Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Daniel Longbottom

Entitled
POLYNOMIAL CURVE FITTING INDICES FOR DYNAMIC EVENT DETECTION IN WIDE-AREA MEASUREMENT SYSTEMS

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Steven Rovnyak
Chair

Lingxi Li

Yaobin Chen

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Steven Rovnyak

Approved by: Yaobin Chen 04/09/2012
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

POLYNOMIAL CURVE FITTING INDICES FOR DYNAMIC EVENT DETECTION IN WIDE-AREA MEASUREMENT SYSTEMS

For the degree of Master of Science in Electrical and Computer Engineering

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Daniel Longbottom

Printed Name and Signature of Candidate

04/17/2012

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

POLYNOMIAL CURVE FITTING INDICES FOR DYNAMIC EVENT DETECTION IN WIDE-AREA MEASUREMENT SYSTEMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Daniel W. Longbottom

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2012

Purdue University

Indianapolis, Indiana

ACKNOWLEDGMENTS

I would like to gratefully acknowledge my thesis advisor, Dr. Steven M. Rovnyak, for his guidance and insight during the entire course of this research and thesis work. Dr. Rovnyak shared his research experience with me and was always willing to meet with me and help answer my questions. He continuously encouraged me to learn and was clear in what his expectations were for the research, for which I am always thankful.

I would also like to thank my advisory committee members, Dr. Lingxi Li and Dr. Yaobin Chen, for their time and insight during the completion of this thesis.

I would like to extend a special thanks to the staff of the Electrical and Computer Engineering department, specifically Jeff Sears, Lab Coordinator, and Sherrie Tucker, Office Coordinator. Their support and willingness to help me with any of the issues I encountered during the course of this research is much appreciated.

I would also like to thank Valerie Lim Diemer and Sherrie Tucker, who gave me assistance and direction in formatting this thesis. Finally, I would like to thank my wife and family for their immense support and encouragement during my time in college.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

CFE	Curve Fitting Error
COA	Center Of Angle
DT	Decision Tree
ISBA	Integral Square Bus Angle
ISGA	Integral Square Generator Angle
FN	False Negative
FP	False Positive
MW	Megawatt
PMU	Phasor Measurement Unit
SBA	Squared Bus Angle
SNR	Signal-to-Noise Ratio
TN	True Negative
TP	True Positive
TSAT	Transient Security Assessment Tool
WAMS	Wide-Area Measurement System

ABSTRACT

Longbottom, Daniel W. M.S.E.C.E., Purdue University, May 2012. Polynomial Curve Fitting Indices for Dynamic Event Detection in Wide-Area Measurement Systems. Major Professor: Steven M. Rovnyak.

In a wide-area power system, detecting dynamic events is critical to maintaining system stability. Large events, such as the loss of a generator or fault on a transmission line, can compromise the stability of the system by causing the generator rotor angles to diverge and lose synchronism with the rest of the system. If these events can be detected as they happen, controls can be applied to the system to prevent it from losing synchronous stability. In order to detect these events, pattern recognition tools can be applied to system measurements. In this thesis, the pattern recognition tool decision trees (DTs) were used for event detection. A single DT produced rules distinguishing between and the event and no event cases by learning on a training set of simulations of a power system model. The rules were then be applied to test cases to determine the accuracy of the event detection.

To use a DT to detect events, the variables used to produce the rules must be chosen. These variables can be direct system measurements, such as the phase angle of bus voltages, or indices created by a combination of system measurements. One index used in this thesis was the integral square bus angle (ISBA) index, which provided a measure of the overall activity of the bus angles in the system. Other indices used were the variance and rate of change of the ISBA. Fitting a polynomial curve to a sliding window of these indices and then taking the difference between the polynomial and the actual index was found to produce a new index that was non-zero during the event and zero all other times for most simulations.

After the index to detect events was chosen to be the error between the curve fit and the ISBA indices, a set of power system cases were created to be used as the training data set for the DT. All of these cases contained one event, either a small or large power injection at a load bus in the system model. The DT was then trained to detect the large power injection but not the small one. This was done so that the rules produced would detect large events on the system that could potentially cause the system to lose synchronous stability but ignore small events that have no effect on the overall system. This DT was then combined with a second DT that predicted instability such that the second DT made the decision whether or not to apply controls only for a short time after the end of every event, when controls would be most effective in stabilizing the system.

1. INTRODUCTION

1.1 Problem Statement

Power systems today are becoming increasingly complex and therefore need to be carefully monitored and controlled to keep the system within certain operating conditions. With the increased use of wide-area measurement systems (WAMS), it is possible to receive data back from the system and have an accurate observation of its dynamics in real time. That data can then be used to detect dynamic events such as the loss of a generator or a fault on a transmission line that could affect the system's stability. The use of phasor measurement units (PMUs) allows for bus phase angles on the system to be measured.

The synchronous stability of the system can be determined from generator rotor angles, such as those shown in Figure 1.1. As the figure shows, the angles of all the generators in the system begin to diverge around 3 seconds, causing several of the generators to lose synchronism with the rest of the system and making the system unstable. The goal of response-based control is to prevent this instability by monitoring system measurements such as bus phase angles, predict whether the system is about to lose synchronism, and apply controls to prevent the system from becoming unstable. If a control action consisting of fast power changes on two high voltage DC (HVDC) lines is applied at 0.67 seconds, the same simulation is stabilized, as can be seen in Figure 1.2.

The controls used in this thesis are response-based and one-shot. Response-based control means that if an event occurs, dynamic data such as voltages and currents are used to determine what control action, if any, should be taken. This type of control is a research topic of more interest than event-based control, which relies on the direct detection of specific outages in the system to apply control. The term one-

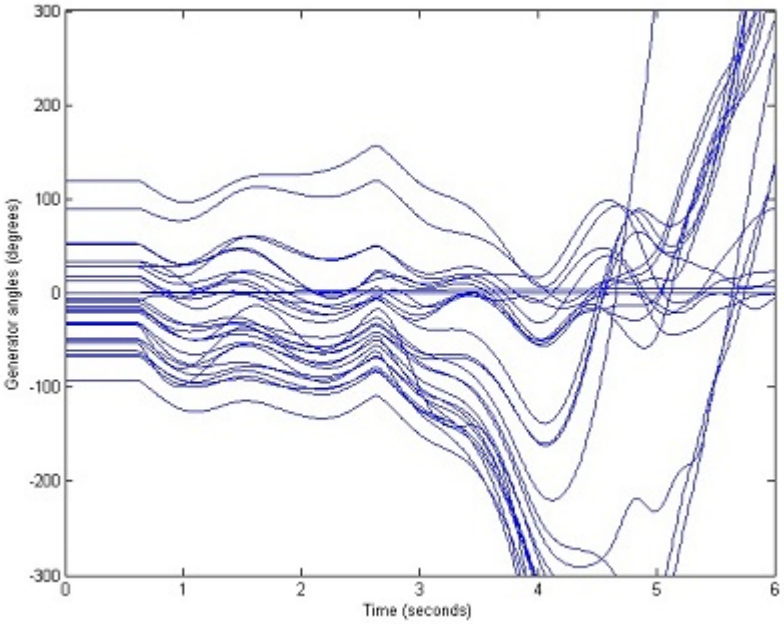

Fig. 1.1. Diverging generator angles

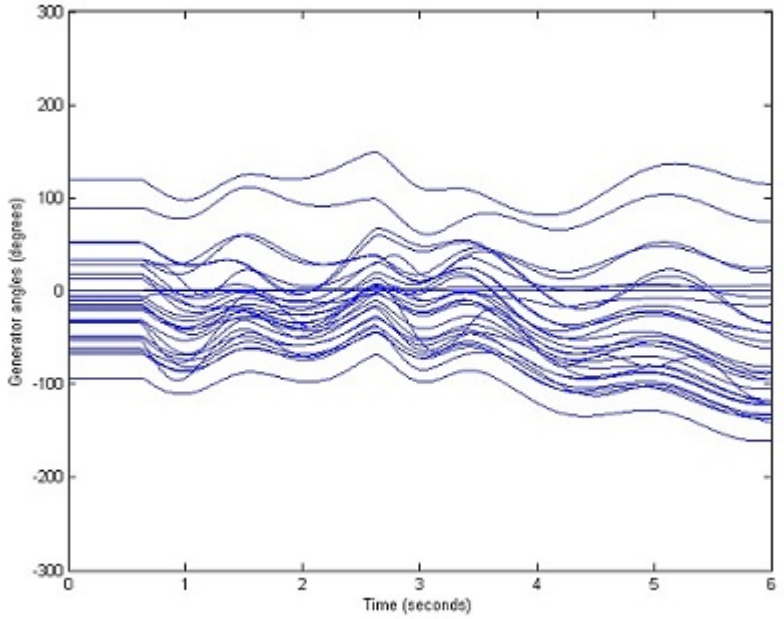

Fig. 1.2. Generator angles with control applied at 0.67 seconds

shot controls means that the controls are applied once to the system and do not occur again unless they are reset. Therefore they are applied based on system measurements only from the time frame before the control is applied. The control actions taken can include load shedding, generator tripping, and reactive power changes. In this thesis, the control action used was the same control applied in Figure 1.2.

1.2 Previous Research

Using response-based controls in power systems has been a topic of previous research and many different indices have been used to predict the loss of synchronism. In [1], generator angles measured immediately after a fault were used to predict transient stability using decision trees (DTs). Pattern recognition tools such as DTs are used to classify data points based on prior knowledge of data in the classes. The phase angle of one bus and the rate of change of another bus phase angle in the power system model in [2] were used as inputs in a DT to predict instability. Other indices used for detecting instability have been researched, such as the ISGA, a weighted average of all the generator angle differences in the system [3], the apparent resistance of a bus [4], and a combination of the maximum difference between generator rotor angles and speeds [5].

Using pattern recognition tools to predict instability can be difficult without knowing when and where the event that caused instability occurred in the system because measurements before the event happens will be the same for both stable and unstable cases. Therefore it is useful to be able to detect the event on the system before trying to predict instability. The work in [6] provided a method to detect events and their location in the system. The indices used were based on the variance of the generator rotor frequencies. The variance of a sliding window of frequency measurements gave better results as an index for detecting events than using one data sample at a time. This was because the variance-based index was less sensitive to glitches, which are not significant events in the system, than the frequency-based index.

1.3 Importance of Timing in Control Application

Once an event on the system is detected, the next step is to decide whether or not to apply control. The sooner the decision is made, the more effective the controls will be in stabilizing the system, because the system will not be close to losing synchronism yet. This presents a challenge to response-based controls, because as the system gets closer to losing synchronism, it becomes easier to predict the impending instability. Therefore, the timing of the controls must be delicately balanced. If the controls are applied too soon, before the instability can be predicted, their application may turn out to have been unnecessary. However, if they are applied too late, the controls might not be effective in maintaining system stability. The allowable time to apply control is discussed in [7]. Figure 1.3 illustrates the fact that controls are more effective when applied soon after a detected event. A set of 24 power system simulation cases that lost stability after two transmission line faults was used. If control was applied at the fault clearing time, which is the end of the fault, all of the cases were stabilized. However, if the control was applied later, the number of cases stabilized decreased. The figure shows the number of cases stabilized from that set as a function of the delay between the fault clearing time and control application.

1.4 Goals of This Thesis

The objective of this thesis is to find an index from system measurements that can be monitored in real time and set a threshold on that index that will trigger when an event that could compromise system stability happens in the system. Once this is done, it can be combined with instability prediction in order to improve the accuracy of instability prediction and the timing of the controls being applied. To accomplish this, it is proposed that two DTs be used, one to detect events and the other to predict instability. The first DT will always check for an event, and when it detects one, it will wait until it detects that the event ended and then enable the second DT for a short period of time. If the second DT predicts instability in that

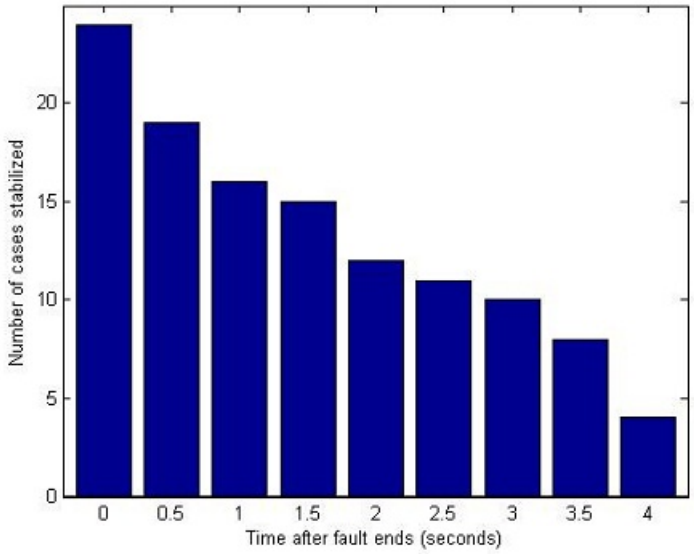

Fig. 1.3. Cases stabilized versus time control is applied

window, control is applied to the system. If it does not detect instability in that window, it is disabled until another event occurs. The second DT is effectively forced to make an instability prediction immediately after the fault, when controls are most effective. The block diagram of the two DTs can be seen in Figure 1.4.

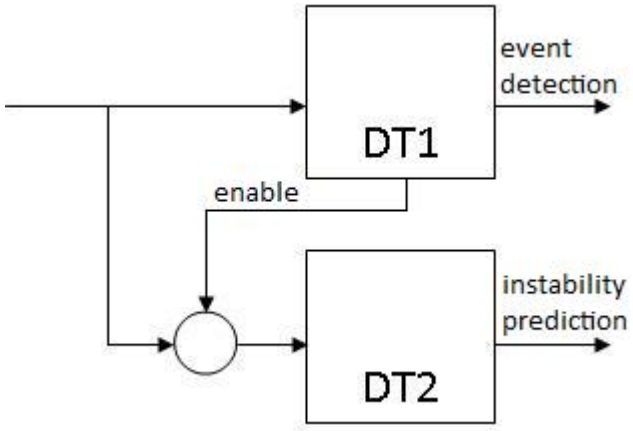

Fig. 1.4. Two decision tree system

1.5 Tools Used in This Thesis

The program Transient Security Assessment Tool (TSAT), made by Powertech Labs Inc. was used to run the power system simulations referred to in this thesis. TSAT is used to model and assess the dynamic behavior of power systems [8], and produces data that includes generator rotor angle and bus angle quantities. The power system modeled was a 176-bus model of the Western Electric Coordinating Council power grid. Of these buses, 17 had PMUs and 71 were load buses. There were also 29 generator buses whose rotor angles were monitored. Simulations were run for six seconds, and the step size was 1/120 of a second. One electrical cycle at 60 Hz equals two steps, or 1/60 of a second.

Two sets of simulations were used in this thesis. The first set contained faults on 40 lines in the model that could cause instability if they were faulted. The set consisted of four cases of single phase to ground or three phase short circuit faults of varying lengths on each of the 40 lines, and 225 cases of a single phase fault followed by a three phase fault on another line for a total of 385 cases. The second set consisted of three phase faults of length varying from momentary (zero cycles) to 11 cycles on each of the 40 lines, for a total of 480 cases. Both sets of simulations produced both stable and unstable cases. In order for a case to be classified as unstable, the generators in the system had to lose synchronism. A loss of synchronism was defined as the maximum difference between any two generators in the system exceeding 300 degrees. Reference [9] shows why this threshold is more accurate than the commonly used threshold of 360 degrees. If the fault or faults in a case did not cause this angle difference to exceed 300 degrees at any point in the simulation, that case was classified as stable. The stability of a case depended on several factors, such as the location and duration of the fault. Throughout this thesis, a highly unstable case refers to a case that included a long fault which caused the generator angles to diverge quickly and greatly exceed the threshold.

2. DECISION TREES

2.1 Background

Decision trees are a widely used pattern recognition tool. They learn on a training set, which is a set of data points consisting of input variables and a value assigned to each point, called the target variable. The target variable for a binary DT can take on two values, either 1 or 0. These values represent the class that each data point belongs to, either the positive class or negative class. The DT tries to find some correlation between the inputs and the target variable so that it can predict what class a data point belongs to based on the input variables. A trained DT consists of a set of if-then rules that are then used to classify any data point with the same input variables. The name "decision tree" comes from the shape of the flowchart of rules, which resembles an upside-down tree. A data point starts at the top or root node, and makes its way down the branches based on the rules until it reaches a leaf node, where it is classified (assigned a target value).

Figure 2.1 shows an example DT. The title at the top of the figure says that the the file "WtrainOriginal.csv" was used to build the DT and the target variable is named "istab". The input variables in this example are "V9ADot" and "V6A". The rules are at the top of each node. For instance, the first rule is "$V9ADot >=< 66.82$", which means if V9ADot is greater than or equal to 66.82, the left branch is taken where the data point reaches a leaf node and is classified. Otherwise, the right branch is taken, where other rules are applied until a leaf node is reached. The first number under every leaf node represents the rule number (not relevant to this discussion), while the number below it is the target variable, either 1 or 0. The number below the target variable is the number of data points in the training set that fell into that particular leaf node's category, and the percentage below it is the accuracy of the classification

of those data points. For instance, the information contained in the leftmost leaf node says that rule number 2 was used, the target variable for data points in that node was 0, there were 43441 data points in the training data set that fell into that category, and of those, $43441 * 0.988 = 42938$ data points had the correct target variable of 0.

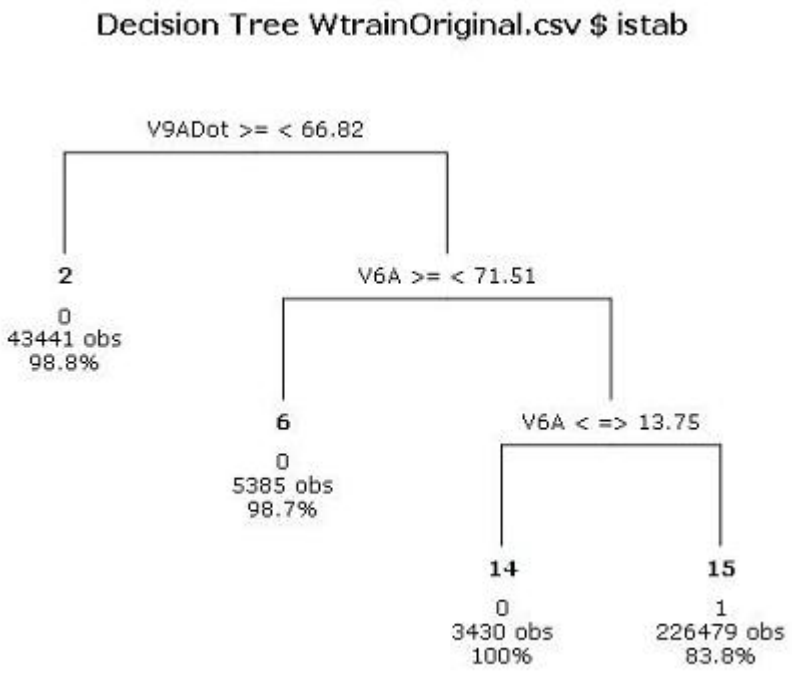

Fig. 2.1. Example DT with four leaf nodes

2.2 Decision Tree Tools

The DT software used in this thesis was Rattle, a graphical user interface for data mining using the R statistical language [10]. Two parameters were used to adjust the output of the decision tree: the complexity cost and the loss matrix. The complexity cost controls the size of the decision tree. Setting the complexity cost very low will most likely produce a complex decision tree with many levels and branches. On the other hand, raising the complexity cost causes Rattle to decrease the number of branches and nodes. Having a higher complexity cost makes the generated rules less

complicated and less specific to the training data, but can eliminate important nodes and even ignore some or all of the input variables if set too high. The loss matrix assigns relative weights to the types of misclassifications. There are four different outcome classes, true negative (TN), true positive (TP), false negative (FN), and false positive (FP). A false negative is a data point that belongs in the positive class but that the DT classifies in the negative class, while a false negative is the reverse. The loss matrix entered into Rattle is in the form of four numbers, one weight for each outcome class as TN, FN, FP, TP. For example, a loss matrix of 0,10,1,0 penalizes the DT for a false negative 10 times more than for a false positive and does not assign weights to the TN and TP classes. If no loss matrix is assigned, the FN and FP classes are given equal weights.

2.3 Using Decision Trees With Simulation Results

The results of this thesis were produced using the set of 385 cases as the training data set and the set of 480 cases for testing. To create the training data set, the 385 cases were run. Each time a case was run, TSAT produced bus and generator angle measurements at each time step. These values, and other indices such as the ISBA, were written to an output file. If the case was stable, the output variable for every data point in that case was assigned 1. If the case was unstable, all of the output variables were assigned the value 0. This data set was then was used in Rattle to train the DT and produce rules that could be used on cases such as the testing set to predict instability.

3. THE INTEGRAL SQUARE BUS ANGLE INDEX

3.1 Previous Index Developed

Throughout the research, many different indices based on system measurements were used for applying response-based control. The index discussed in [3], the integral square generator angle (ISGA) index, provides an overall measure of the relative angles of all the generators in the system and is defined by the equation

$$ISGA = \int_0^T \sum_i M_i(\delta_i(t) - \delta_{coa}(t))^2 dt. \tag{3.1}$$

The constants M_i are machine inertias, the terms $\delta_i(t)$ are the generator angles, and $\delta_{coa}(t)$ is center of angle (COA)

$$\delta_{coa}(t) = \frac{\sum_i M_i \delta_i(t)}{\sum_i M_i}. \tag{3.2}$$

Generators farther from the COA are penalized more by the ISGA. Disconnected generators are removed from the calculation of the COA and ISGA.

A variant of the ISGA, the integral square bus angle (ISBA) index, was used in this research. The ISBA provides a measure of the overall activity of the bus angles in the system. It was chosen over the ISGA because in a real-time power system, PMU technology enables system operators to monitor the phase angles of buses in real time, but measuring generator rotor angles in real time cannot be done directly and is much more difficult.

3.2 Calculation of the ISBA

In order to calculate the ISBA, first the squared bus angle (SBA) index was calculated using the equation

$$SBA = \sum_i M_i(\theta_i(t) - \theta_{coa}(t))^2, \tag{3.3}$$

where

$$\theta_{coa}(t) = \frac{\sum_i M_i\theta_i(t)}{\sum_i M_i}. \tag{3.4}$$

$\theta_i(t)$ represent the phase angles of the buses included in the ISBA calculation and M_i are the weights assigned to each bus. In this thesis, equal weights of one were assigned to all the buses included in the calculation. Also, the equations for SBA and $\theta_{coa}(t)$ were evaluated at discrete sample times because the bus angles in the simulation were measured at discrete points in time. Instead of integrating over a sliding window to find the ISBA, a low pass filter with transfer function $H(s) = 6/(s + 6)$ and cutoff frequency of 6 rad/sec, or about 1 Hz, was applied to the SBA. Figure 3.1 shows the SBA and ISBA after the low pass filter for one case.

3.3 Calculation of the Gradient of the ISBA

Another index used in this research was the rate of change, or gradient, of the ISBA. Since the ISBA was discrete, the gradient had to be approximated. This was accomplished two different ways. In Method 1, the point-to-point difference of all the ISBA data points was taken. Since the ISBA was calculated from the SBA using a low-pass filter instead of integrating, the gradient of the ISBA did not equal the SBA. A second method of approximating the gradient of the ISBA was also used. Since taking the derivative of a quantity in the time domain is equivalent to multiplying by the transfer function $H(s) = s$ in the frequency domain, the derivative can be thought of as a high frequency amplifier. Instead of multiplying the ISBA by s to obtain its gradient, it could be calculated directly from the SBA by combining the low pass filter with the multiplication by s, which produces a high pass filter. The

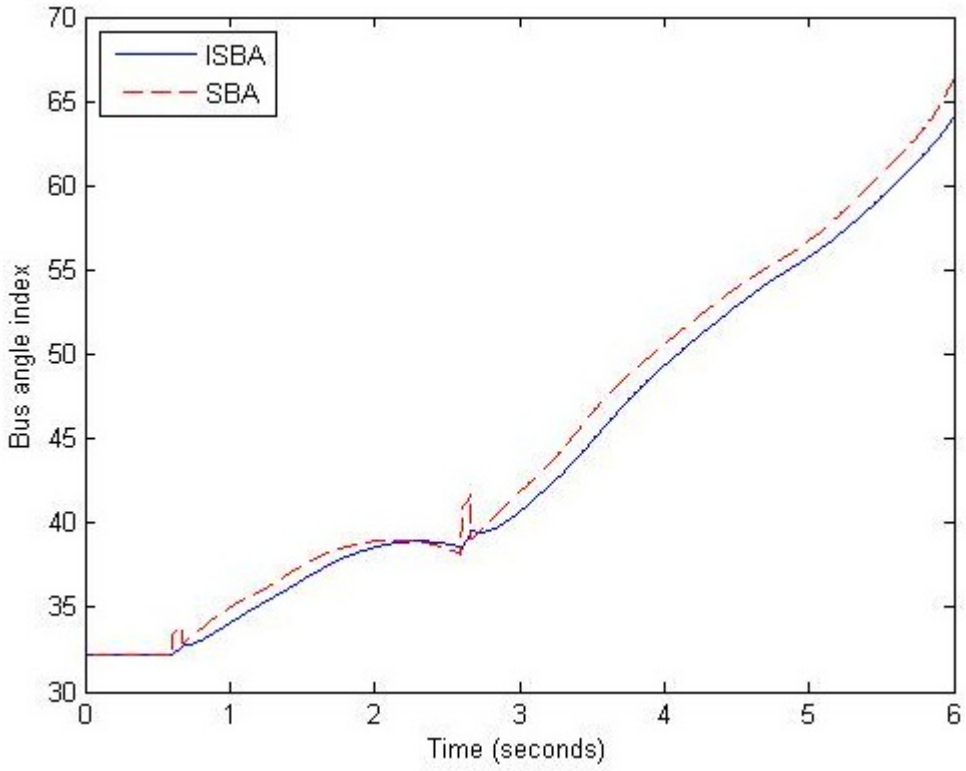

Fig. 3.1. SBA and ISBA indices

high pass filter was derived from the same low pass filter used to obtain the ISBA, so the high pass filter had the transfer function $H(s) = 6s/(s + 6)$.

Figure 3.2 compares the ISBA and its gradient for the case in Figure 3.1. In the figure, the gradient was calculated using Method 1. Using Method 2 would produce the same gradient, with the only difference being a scaling factor of the step size, 1/120.

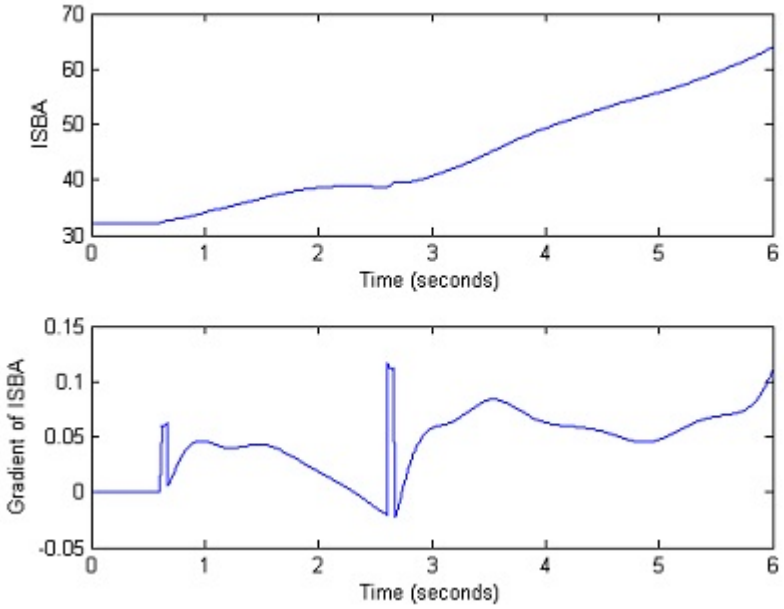

Fig. 3.2. ISBA and its gradient

4. DISCONTINUITY OF BUS ANGLES

4.1 Bus Angle Wrapping

Before choosing an index based on the system bus angles, such as the ISBA, the problem of discontinuous measurements had to be resolved first. The problem was that unlike generator angles, bus angles can be discontinuous and do not go outside the range of ±180 degrees, as shown in Figure 4.1.

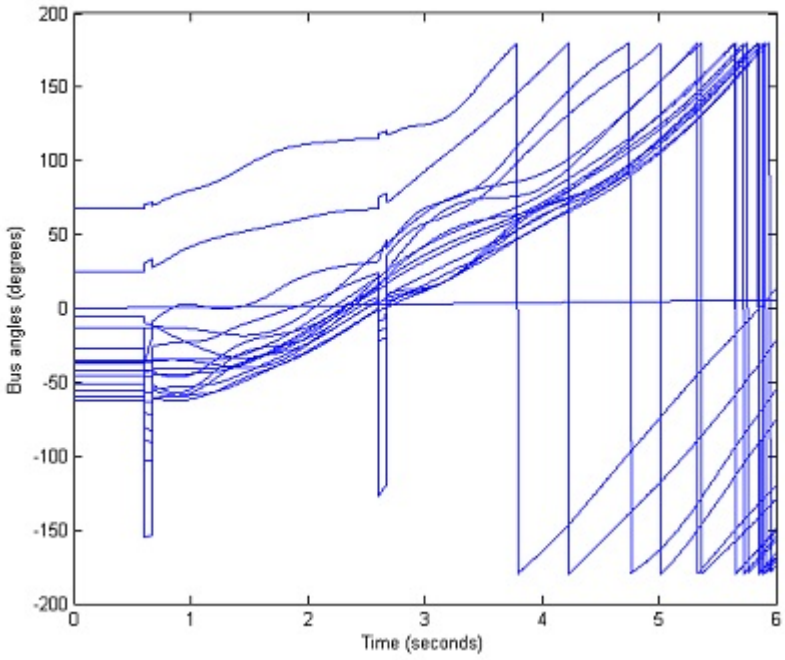

Fig. 4.1. Discontinuous bus angles

If an angle crosses the ±180 degree thresholds, it wraps back around to the opposite limit 360 degrees away. In order for the ISBA calculation work, the angles had to be unwrapped, so that they remained continuous. The angles from Figure 4.1 after reconstruction are shown in Figure 4.2.

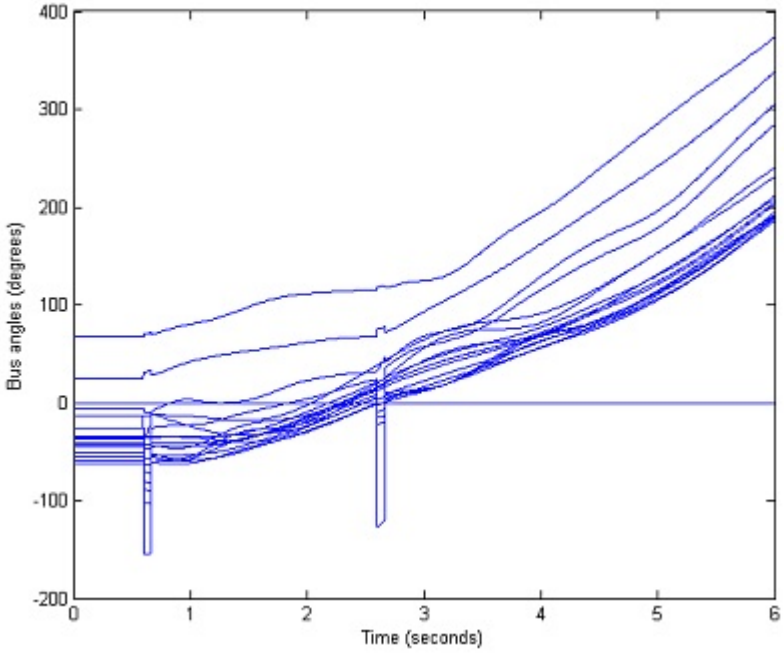

Fig. 4.2. Reconstructed continuous bus angles

4.2 Choosing an Angle Difference Threshold

In order to reconstruct the angles, 360 degrees had to either be added or subtracted from each angle starting at each point of discontinuity. To find each point of discontinuity, the bus angles at every time step were compared to the angles at the previous time step. Then, a threshold was chosen such that if the difference between the current and previous angle measurements exceeded that threshold, 360 degrees was subtracted from every angle measurement after the discontinuity. Likewise, if the difference was smaller than the negative value of that threshold, 360 degrees was added to the subsequent measurements.

Several different thresholds for the difference between consecutive bus angle measurements were tried. The threshold could not be exactly ± 360 degrees, because the absolute value of the difference between two discontinuous measurements would always be less than 360. For instance, if a bus angle at one time step was 179.9 degrees and increased 0.2 degrees the next time step, it would wrap around to -179.9

degrees, for a difference of $(-360 + 0.2) = -358.8$ degrees, not large enough to cross the threshold.

The first threshold tried was 330 degrees. This threshold worked for most cases, but on some highly unstable cases, some of the bus angles still had discontinuity. After investigating the bus angles around the points of discontinuity, it was determined that the threshold was set too high, because if a bus angle was changing more than 30 degrees per time step, like it would in some unstable cases, the difference between the angle before and after the discontinuity would not exceed the threshold.

A lower threshold of 300 degrees was tried next. Again, this threshold did not prevent discontinuities in some highly unstable cases because some bus angles changed more than 60 degrees per time step. Finally, a threshold 180 degrees was chosen. This threshold was determined to be the most effective for removing the discontinuities in the bus angles because it was assumed that a bus angle would not change more than 180 degrees in one time step. The 180 degree threshold unwrapped all measurements that exceeded the ± 180 degree range, but did not add or subtract 360 degrees from any angles that were changing quickly but did not exceed the range. Figure 4.3 shows an individual bus angle of a very unstable case for no threshold and for thresholds of 330, 300, and 180 degrees. Figure 4.4 shows the same four plots zoomed in on a 0.4 second window to show the discontinuities on the first three plots, but not on the fourth plot.

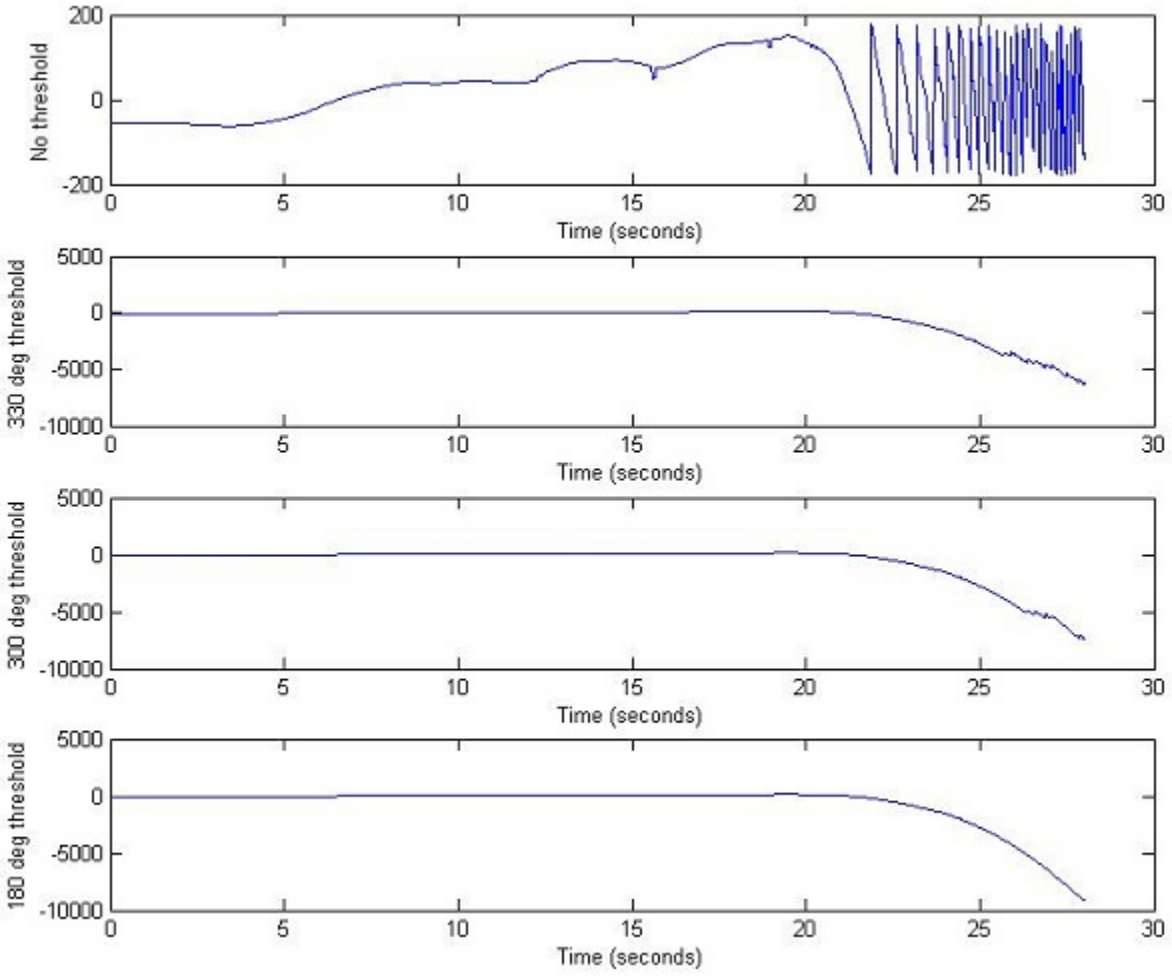

Fig. 4.3. Bus angle with four different thresholds

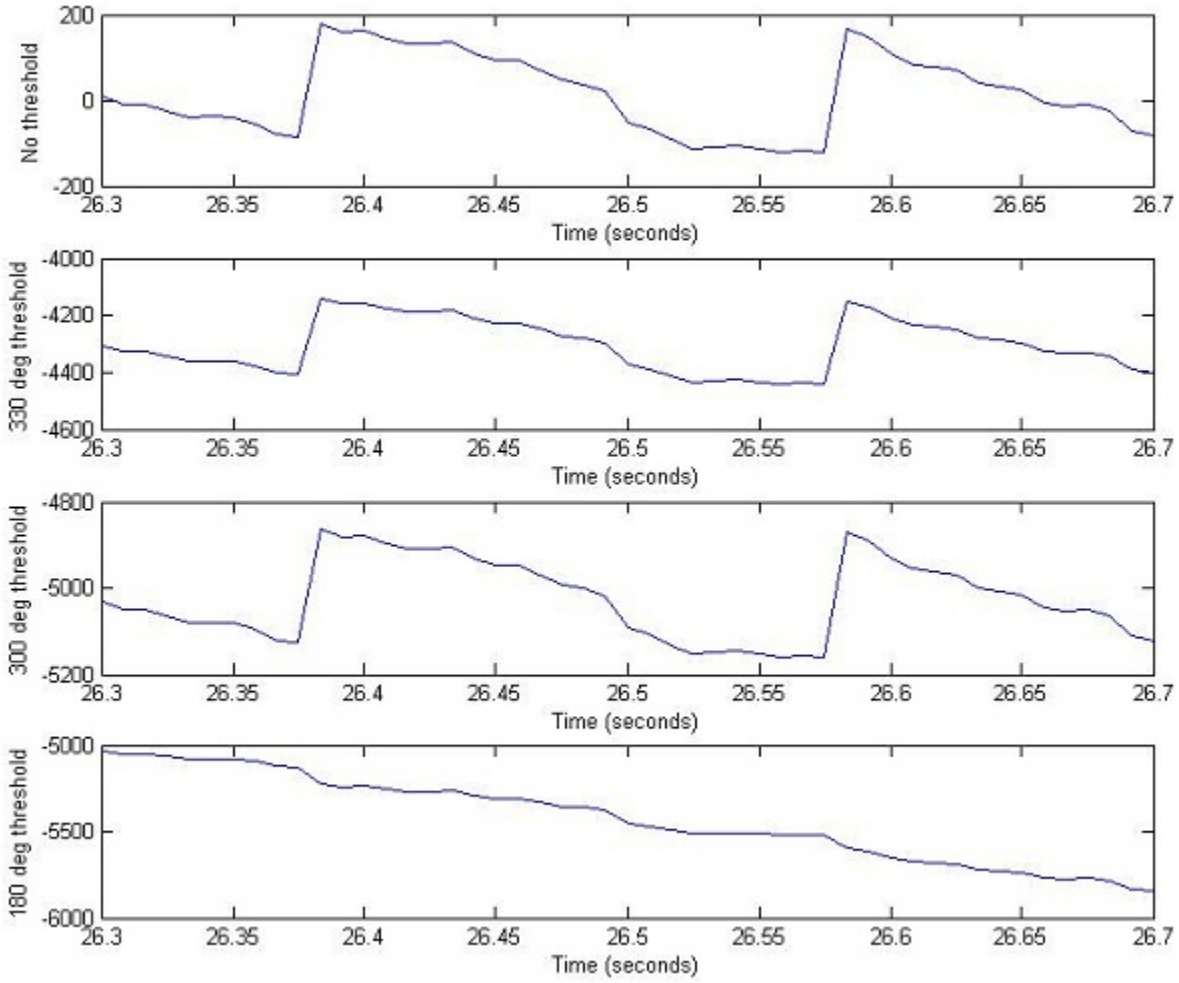

Fig. 4.4. Bus angle between 26.3 and 26.7 seconds with four different thresholds

5. INDICES USED FOR EVENT DETECTION

Once the bus angles were unwrapped so that they remained continuous, they were used to detect events. Event detection with bus angles is a tool that was added to response-based control to improve the timing and accuracy of applying control. To detect an event, a threshold was set on an index such as a bus phase angle frequency or the ISBA. If the threshold was exceeded at any point in time during the simulation, it indicated an event had occurred. Because events are happening all the time on a power system, the system is always changing. However, most events that occur on a power system are normal and do not affect system stability. The goal of event detection in this thesis was to detect the large events on the system that could cause instability, but ignore the small events that do not affect the system. In order to do this, both the appropriate index and its threshold had to be found.

5.1 Individual Bus and Generator Measurements

First, individual bus angles, and their frequency and variance, were tried as parameters for event detection. The variance-based index developed in [6] was tried first. Because the index used in [6] was based on generator angle frequencies, generator angles instead of bus angles were used initially. Their variance was calculated using a 20-step sliding window. The variance for each new time step was calculated from the previous 20 measurements. For the first 20 steps, it was calculated by taking the variance of all the previous time steps. Since the variance was calculated every time step, the size of the variance data array was the same size as the original data. Events were detected using a threshold of $0.0002N_{cyc}$ given in [6] on the variance of the generator angle frequencies, where N_{cyc} is the length of the window in cycles. Since the window length was 20 steps or 10 cycles, the threshold applied was

$0.0002 * 10 = 0.002Hz$. On the set of 385 cases, most of the first events were detected with a delay between 2 and 5 cycles, as can be seen in Figure 5.1.

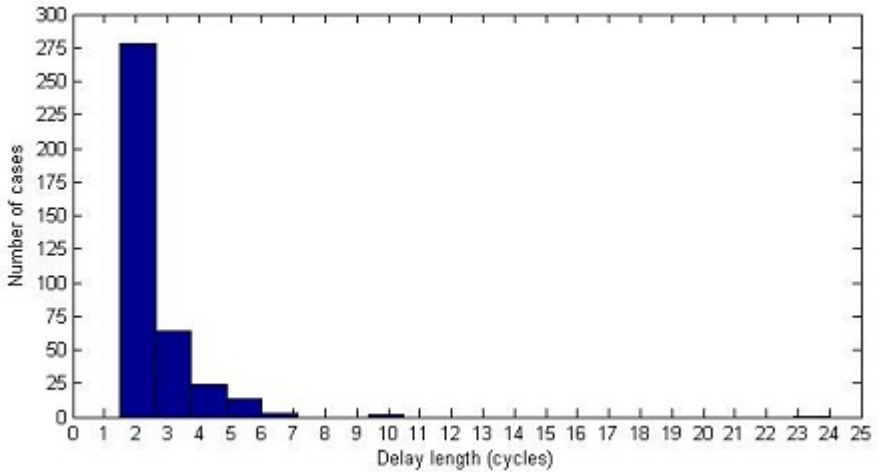

Fig. 5.1. Histogram of event detection delay using generator frequency variance

Another set of measurements used for event detection was the bus angle frequencies. Originally, an arbitrary threshold of 1 Hz was chosen to detect events. If any bus angle frequency deviation from 60 Hz exceeded 1 Hz, the start of an event was recorded. This threshold consistently detected the first event on either the 385 or 480 cases with a delay of 1 cycle.

While using the variance and frequency of individual bus or generator angles correctly detected events with small delays most of the time, these measurements would not be practical in a real-time system for event detection. This is because a small event may occur close to a particular bus, causing a spike in the measurements at that bus, but not affecting the overall system. In other words, it might appear as a large event at a particular bus because of its proximity, when in reality it is insignificant in the overall system. The individual bus thresholds just mentioned would consider these types of events as significant. Also, it was reported in [5] that combining multiple indices into a composite index did a better job of predicting instability than a single index. The rest of this section is a description of the other indices used for event detection.

5.2 The ISBA and Derived Indices

The ISBA was used next for event detection because it only experiences large swings when all the buses on the system change at the same time, since it is a weighted measure of all the bus angles in the system. In addition to the ISBA, it was determined that the derived quantities of the rate of change and variance of the ISBA might be good indicators of an event occurring, because the ISBA experiences rapid changes during a significant event. Therefore, the variance and gradient of the ISBA were also used as indices. The variance was calculated using the same 20-step sliding window as was done on the generator angles earlier. The gradient was calculated using Method 1 from Chapter 3. Figure 5.2 compares the ISBA and its variance and gradient for a case where one event takes place around 0.7 seconds. As can be seen from the figure, the values of both the ISBA and its variance and gradient are steady until the event, then increase once the event starts. The red shaded area shows the time when the fault occurred.

Using the variance or gradient of the ISBA for event detection as opposed to the ISBA directly had a couple of advantages. First, in most cases, including the one in Figure 5.2, it was more obvious when the event began using the variance or gradient because they increased more than the ISBA. Also, the ISBA had a non-zero value even while in steady-state, so knowing that its value was greater than zero did not indicate whether or not an event happened.

While using one of these indices for event detection usually could detect the beginning of a fault, their non-zero values after the fault made it impossible to detect the clearing time of the fault and to detect multiple faults, both of which were necessary for response-based control in a real-time system. The case shown in Figure 5.3 illustrates this. It contains two faults, one occurring around 0.7 seconds and the other occurring around 2.6 seconds. Both the variance and gradient spike during each fault, but also increase after both faults. Any threshold that would trigger for the

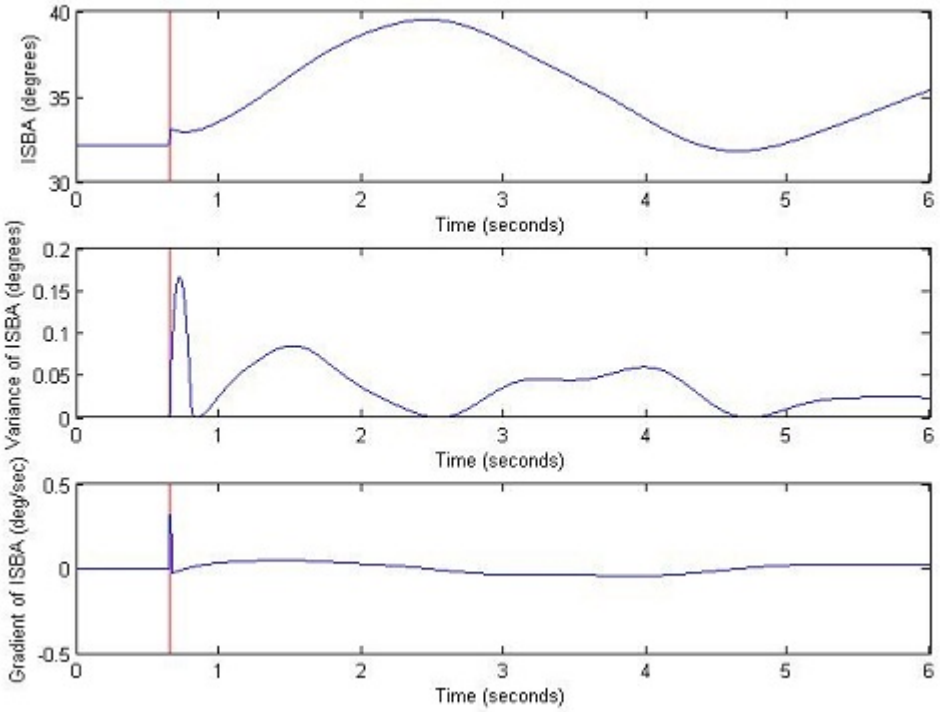

Fig. 5.2. Comparison of ISBA and its variance and gradient

first and second faults would also trigger another time, incorrectly detecting a third fault that did not occur.

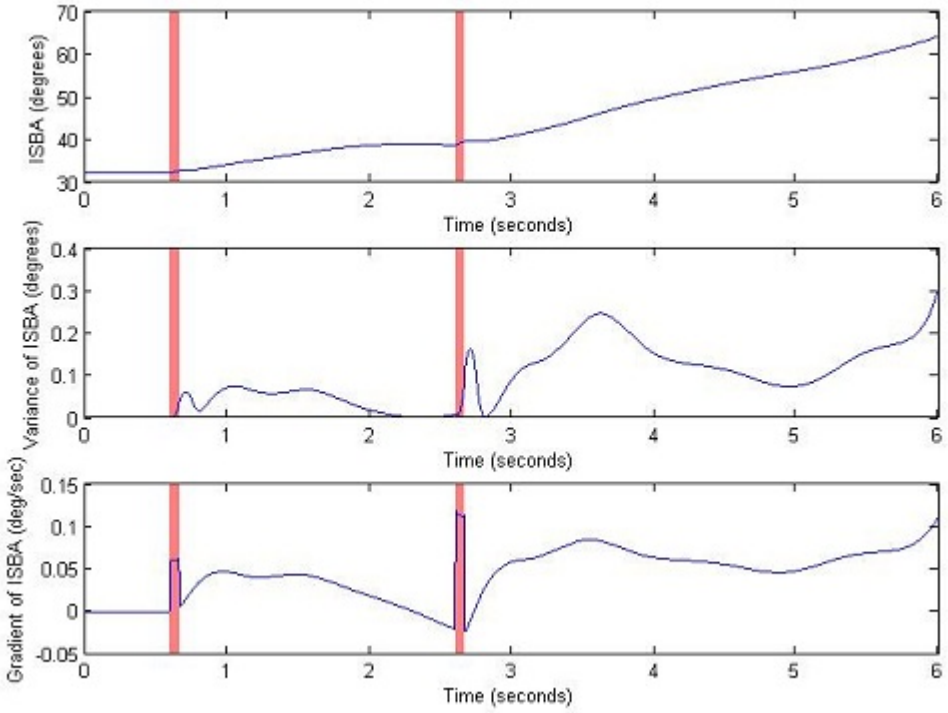

Fig. 5.3. ISBA and its variance and gradient for 2-fault case

5.3 Curve Fitting Applied to the ISBA

In order to detect the clearing time of a fault and multiple faults in succession, another method of detecting events with the ISBA, using polynomial curve fitting, was investigated. Curve fitting had already been used for event detection in [11], where exponentials were fitted to system frequency measurements. In this research, fitting system measurements to a polynomial was tried as an alternative.

5.3.1 Polynomial Curve Fitting

The objective of polynomial curve fitting is to approximate a set of data points with a polynomial function. The method of least squares does this by minimizing the square of the residuals r_i, which are the errors between each data point and the polynomial.

To use the method of least squares, both the degree of the polynomial and the number of data points being fitted must be chosen. If the polynomial has degree k and is being fitted to a set of n data points, then the form of the polynomial will be $a_0 + a_1 x_i + \ldots + a_k x_i^k = \sum_{j=0}^{k} a_j x_i^j$, where the a_j are the coefficients of the polynomial and x_i is the dependent variable in the set of n data points, (x_i, y_i). In order to choose the polynomial that best fits the set of data points, the set of coefficients a_j must be found that minimize the sum of the squared residuals at every data point. That sum, S, is given by the equation

$$S = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} [y_i - \sum_{j=0}^{k} a_j x_i^j]^2. \tag{5.1}$$

The minimum values of this error function can be found by setting the partial derivatives with respect to the coefficients to zero, which yields

$$\frac{\partial S}{\partial a_m} = -2 \sum_{i=1}^{n} [y_i - \sum_{j=0}^{k} a_j x_i^j] x_i^m = 0 \tag{5.2}$$

for $m = 0, 1, \ldots, k$. The next step is to solve the partial derivatives for the coefficients. Let

$$a = \begin{bmatrix} a_0 & a_1 & \ldots & a_k \end{bmatrix} \tag{5.3}$$

$$X = \begin{bmatrix} 1 & x_1 & \ldots & x_1^k \\ 1 & x_2 & \ldots & x_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^k \end{bmatrix} \tag{5.4}$$

$$y = \begin{bmatrix} y_1 & y_2 & \ldots & y_n \end{bmatrix} \tag{5.5}$$

$$r = \begin{bmatrix} r_1 & r_2 & \ldots & r_n \end{bmatrix}. \tag{5.6}$$

Equation 5.2 can be solved for the coefficients by multiplying the x_i^m by the terms in braces and rearranging. In matrix notation, this equation is

$$X^T X a = X^T y. \tag{5.7}$$

Finally, the solution to the coefficients can be found by multiplying both sides of 5.7 by $(X^T X)^{-1}$ and obtaining

$$a = (X^T X)^{-1} X^T y. \tag{5.8}$$

These coefficients minimize S and are used to calculate the polynomial that best approximates the set of data points. Singularity of the matrix $X^T X$ was never encountered during this research; however, in order to avoid that possibility, the pseudoinverse could be used.

Fitting a polynomial to system measurements was chosen because from observation, measurements of the overall system behavior, such as ISBA and its gradient and variance, are generally smooth functions that can be approximated by a polynomial well when a fault is not occurring, but always spike and appear discontinuous when a fault is occurring, which would not be approximated well by a polynomial. The difference between the system measurement and the polynomial approximation could then be used to determine whether or not a fault is occurring. If no event had occurred, the polynomial curve fit would be good and the error between it and the system measurement would be low, even if the measurement was gradually changing. However, if it experienced a sudden spike due to an event, the polynomial at that point in time would not be accurate, and the error would be large.

5.3.2 Curve Fitting the ISBA, Variance, and Gradient

Initially, a polynomial was fitted to the indices of the ISBA, its variance, and its gradient using the method of least squares just described. A polynomial of degree four was chosen, and a 10-step moving window was used. Starting with the first 10 samples, the coefficients of the polynomial were calculated and used to find the

approximate values of the indices at each sample in the current window. Then the absolute value of the difference between the 10 samples from the approximation and actual data points was calculated. The maximum difference in that window was chosen as the error for that window. The window moved one step at a time, so taking the maximum difference from each window produced an index that had the same length as the original index. The first nine samples of the new index were set equal to zero since there were not enough samples to perform the curve fitting calculation. Figure 5.4 shows the polynomial curve fitting error (CFE) indices of the ISBA and its variance and gradient in Figure 5.3.

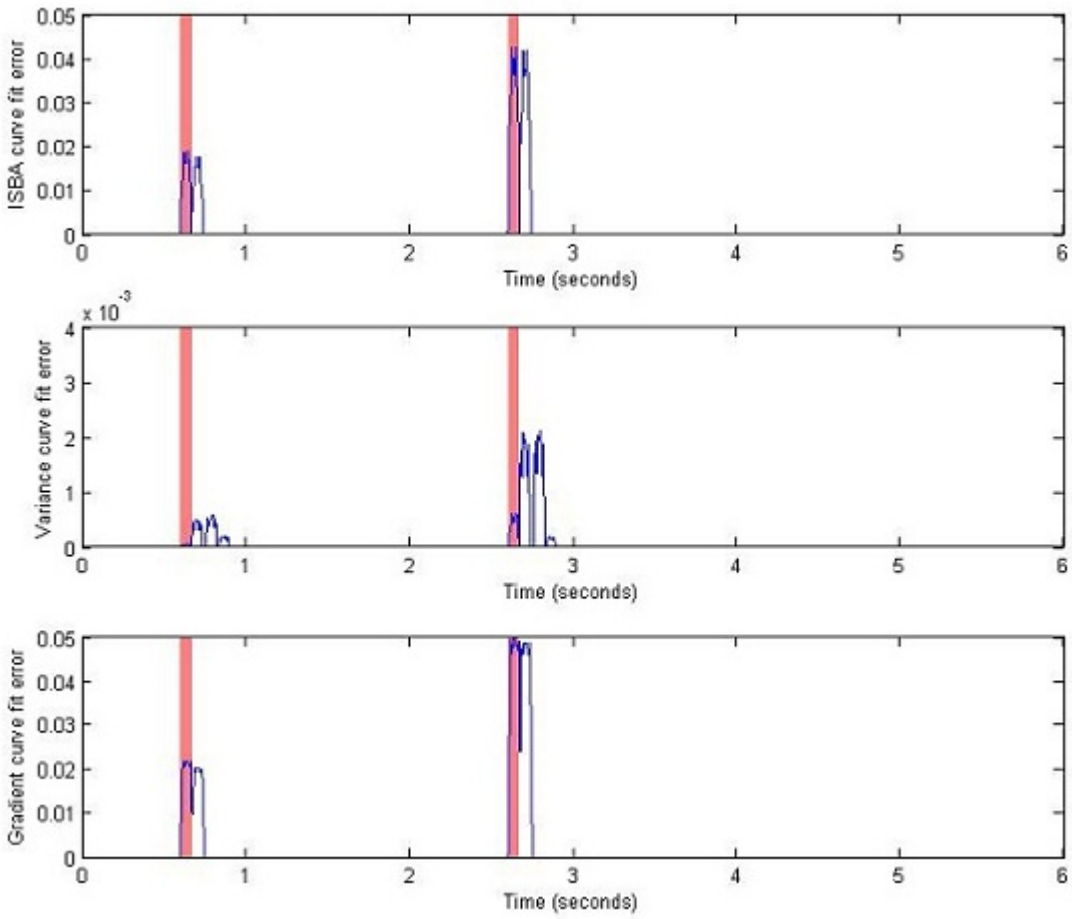

Fig. 5.4. Curve fitting errors from indices in Figure 5.3

All three CFE indices either eliminated or reduced the previous problem of the original three indices, which sometimes increased after a fault. The new indices had non-zero values only during and immediately after the fault for most cases, so a threshold could be applied to them which would not incorrectly detect other faults. Only in some highly unstable cases did the CFE thresholds have non-zero values when a fault was not occurring, as can be seen in Figure 5.5.

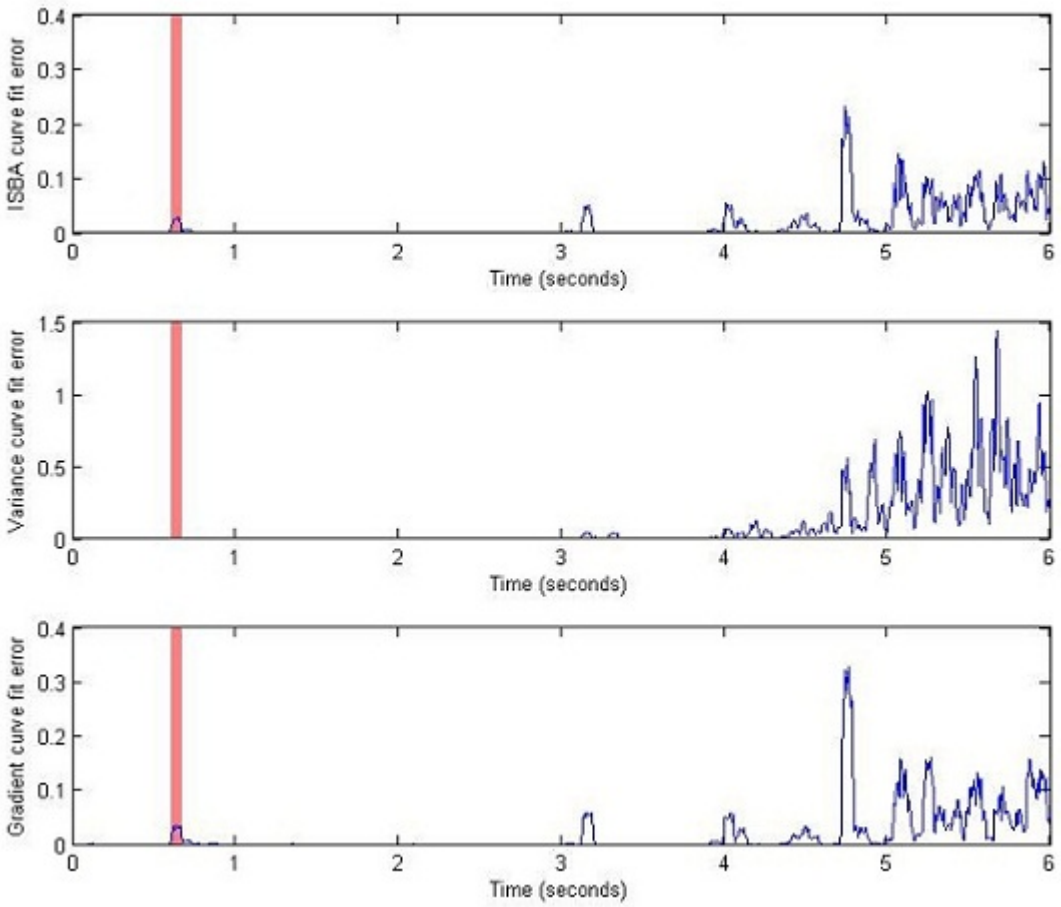

Fig. 5.5. Curve fitting errors for a highly unstable case

In highly unstable cases like the one above, the only time the indices are non-zero not during a fault was near the end of the simulation, when stability had already been lost. By this time, the decision to apply control would have already been made, and the event detection would not be important.

5.3.3 Comparing the Curve Fitting Indices

The three CFE indices were compared to determine which would be the best for event detection. The index on the variance of the ISBA had the poorest correlation to the fault time, which can be seen in Figure 5.4. It was much smaller during the fault than the other two errors and did not decay to zero as quickly as the other two, since the variance did not change quickly like the ISBA or its gradient, which can be seen in Figure 5.3. While it was reported in [12] that variance-based criterion were favored over rate of change criterion because they are not as prone to triggering on small glitches, in this situation the rate of change index would not be susceptible to the same problem because the index it was being used on, the ISBA, was an overall system measurement.

After the CFE for the variance of the ISBA was eliminated as a possible threshold, both the CFE for ISBA and its gradient were used for event detection for several reasons.

1. When both indices were used as input variables to the DT for event detection training, the DT would generally select both.

2. If forced to choose only one index as the input variable, the DT would sometimes choose the CFE of the ISBA and sometimes choose its gradient CFE, depending on factors discussed in the next section on choosing a threshold. (The DT could be forced to choose only one variable by increasing the complexity cost until it ignored one of the input variables.)

3. Visually inspecting both indices for cases such as the one shown in Figure 5.4, it can be seen that both CFE indices have a similar shape and values, so using either one as a threshold would produce similar results.

These reasons indicated that both indices had a similar correlation to the fault time.

5.3.4 Filtering the Curve Fitting Indices

It can be seen in Figure 5.4 that both the CFE indices of the ISBA and its gradient have two consecutive peaks per fault, one at the fault start time and one at the fault clearing time. The ISBA CFE decays all the way to zero in between the two peaks, while the gradient CFE drops about halfway down. Because of this, if a threshold were applied directly to either index, it would probably trigger twice per fault. In order to help prevent this, it was determined that some sort of filtering would need to be performed before a threshold could be applied to either index. The first filter implemented was a simple averaging filter. It took the average of the previous 20 data points as the current data point. The ISBA and gradient CFE indices in Figure 5.4 are shown after being passed through the filter in Figure 5.6.

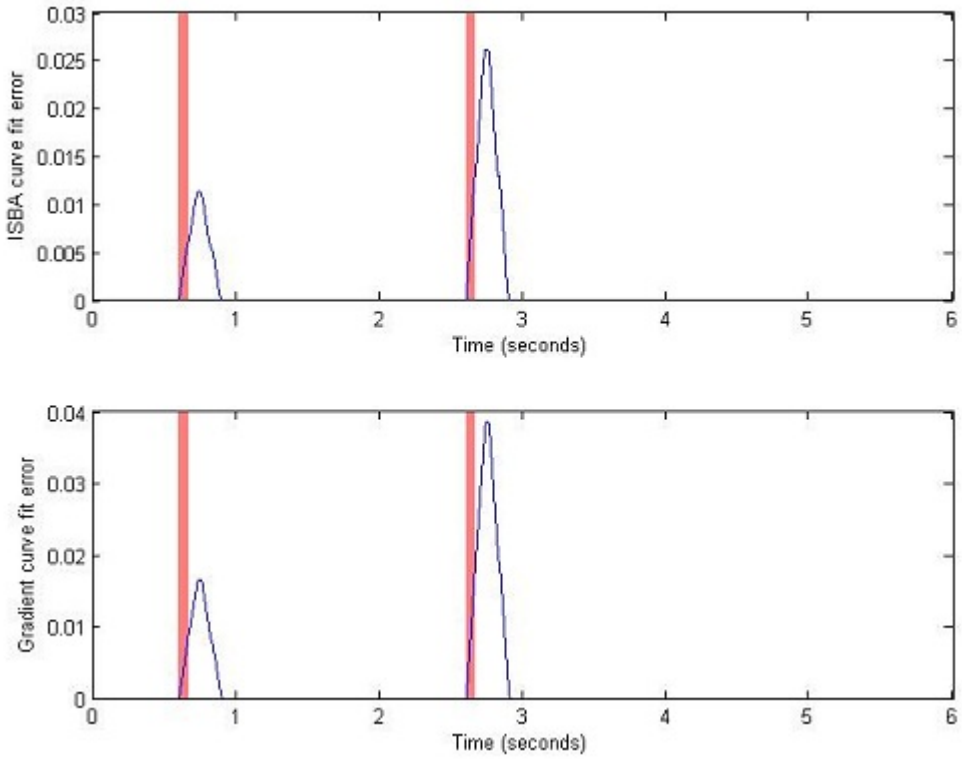

Fig. 5.6. Curve fitting errors after averaging filter

While this type of filtering smoothed out the drop in between the two peaks per fault, it also slowed the decay to zero starting at the fault clearing time, which would delay the detection of the fault clearing time. In order to decrease this delay, a low pass filter defined by a differential equations was applied instead of the averaging filter. By visually inspecting low pass filters with different cut-off frequencies, it was decided that the filter with the best balance between smoothing out the spike in the middle of each fault and minimizing the delay of the waveform had a transfer function of $H(s) = 50/(s + 50)$, with a cut-off frequency of 50 rad/sec, or about 8 Hz. This filter helped to smooth out the high frequencies, such as the spike in the middle of the two peaks, and merge the two peaks together. The CFE ISBA and gradient indices after being passed through the filter are shown in Figure 5.7. As the figure shows, they are smoother in between the two peaks at each fault than the corresponding unfiltered indices.

The decay of the error at the fault clearing time was not as quick as unfiltered error so there was still some delay for detecting the fault clearing time. However, it was decided that this was an acceptable delay, because detecting the fault clearing time late was not as serious as letting the threshold trigger twice for only one fault, which would have occurred without any filtering. Table 5.1 compares the delay associated with detecting the fault clearing time for both the averaging and low pass filters for several different thresholds on the case shown in Figure 5.3. It shows that the low pass filter has smaller delays than the averaging filter for every threshold value.

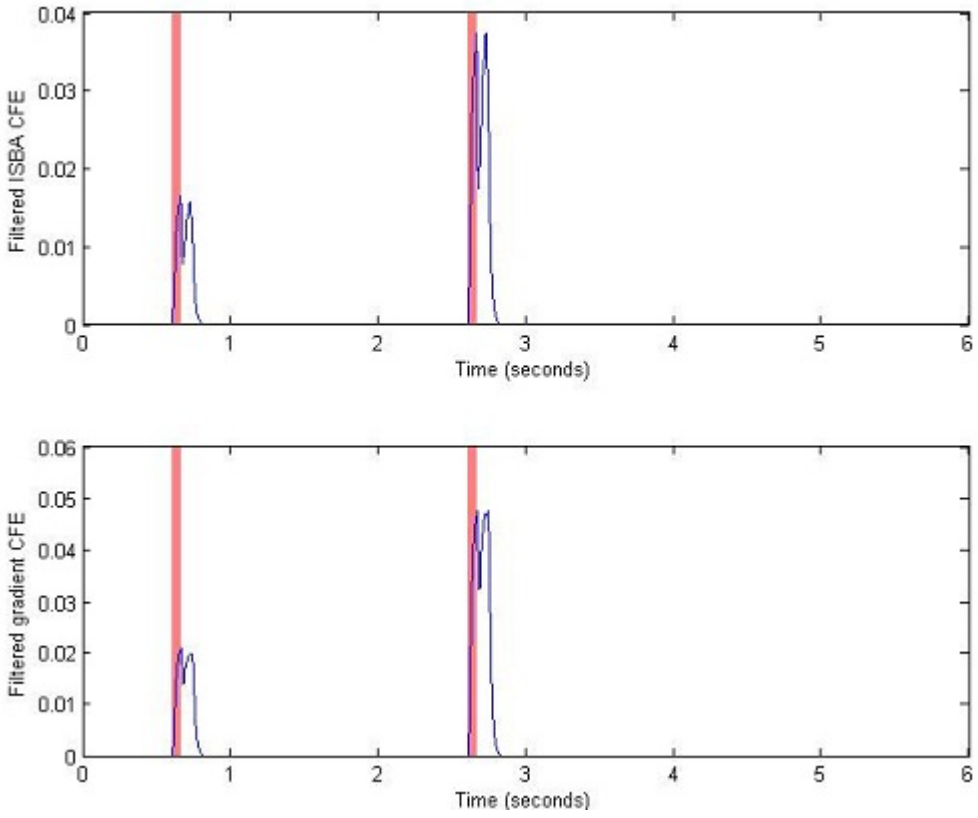

Fig. 5.7. Curve fitting errors after low pass filter $H(s) = 50/(s + 50)$

Table 5.1
Delays of fault clearing time detection on a filtered ISBA CFE index

Threshold	Filter	Delay
0.002	Averaging	14 cycles
0.0025	Averaging	14 cycles
0.003	Averaging	13.5 cycles
0.002	Low pass	8 cycles
0.0025	Low pass	7.5 cycles
0.003	Low pass	7.5 cycles

6. THRESHOLDS FOR EVENT DETECTION

6.1 Cases Used for Decision Tree Training

Along with choosing an appropriate index for event detection, the correct threshold that would be applied to that index needed to be chosen such that it would trigger for all of the significant events in the system that could cause instability, but would ignore the small events and noise on the system. To find this threshold, a set of 142 cases was created to train the DT. Each case had one single event, a power injection of either 200 MW or 500 MW at 40 cycles (about 0.3 seconds) at each of the 71 load buses in the model. Cases were created for these two power changes at each load bus to choose a wide-area threshold, one that could be used for an event anywhere on the system. Power changes of 200 MW and 500 MW were chosen because a 200 MW power injection was a small event that did not affect system stability, while 500 MW was large enough to cause some changes in the system, although not large enough to cause instability. If a threshold could be found that triggered for all the 500 MW events, it would trigger for any larger events that could lead to instability. To train the DT, the cases with the 200 MW change were classified in the positive (no-fault) class, while all the cases with a 500 MW change were classified in the negative (fault-on) class.

While finding a threshold to detect almost all of the events or almost none of the events was simple, the task of differentiating between the 200 and 500 MW cases proved to be much harder. For example, when the DT was trained without a loss matrix and with inputs of all the individual bus angles, frequencies, and variances and tested on the same training cases, all of the 500 MW events and all but two of the 200 MW events were detected. When it was trained again with a 0,10,1,0 loss matrix, it detected all but 6 of the 500 MW events and all but 17 of the 200 MW

events. Using the curve fitting index as the input variable for the DT did a better job of detecting the 500 MW but not 200 MW cases, but it still did not completely separate them.

6.2 Data Points Chosen From Simulations

Choosing what data points from each case to use for DT training and what target variable to assign to the data points in the 500 MW cases were two decisions that had several different options. Several options were experimented with in order to produce the threshold that best distinguished between large and small events. The target variables assigned and data points used fell into three general categories. Following are descriptions of each of those categories. After the descriptions, a figure of the CFE of the ISBA from one 500 MW case is presented for each category. The areas that are greyed out are the times the data points were discarded, and the areas shaded in red show the data points whose output variables were put in the fault-on class.

1. **Data points between 42 and 50 cycles, fault-on target variable for all data points in 500 MW cases.** Data points before 40 cycles were discarded because pre-event conditions were the same for the 200 MW and 500 MW cases. Data points between the beginning of the event (40 cycles) and 42 cycles were discarded because it was determined from inspecting a graph of the CFE of ISBA and its gradient that there was about a 2-cycle delay between the event and the spike in the CFE. The data after 50 cycles was also discarded because the CFE spike ended by that time. With this method, the DT was trained to find a threshold that would trigger based on the CFE spikes due to 500 MW power changes at any load bus, but not trigger for any 200 MW power changes.

2. **Data points between 42 cycles and simulation end, fault-on target variable for data points between 42 and 50 cycles in 500 MW cases.** This method was the same as Method 1, except the data points between 50 cycles and the end of the simulation were kept, and the target value was changed

to no-fault for the points after 50 cycles. This gave the DT more training data in the no-fault class and trained it to find a threshold that triggered only when the CFE spiked on the 500 MW cases, and not any other time.

3. **Data point containing maximum value between 42 and 50 cycles, fault-on target variable for data point in every 500 MW case.** This selection was tried because the threshold set by the DT would trigger the first time a data point exceeded the threshold. Therefore, the most important CFE measurement during the fault was the maximum one. If it were lower than the threshold, no event would be detected, while if it were higher, an event would be detected. Giving the DT only the maximum value during the CFE spikes from both the 200 MW and 500 MW cases was done to find a threshold that would only trigger once for a large event, but never trigger for a small event.

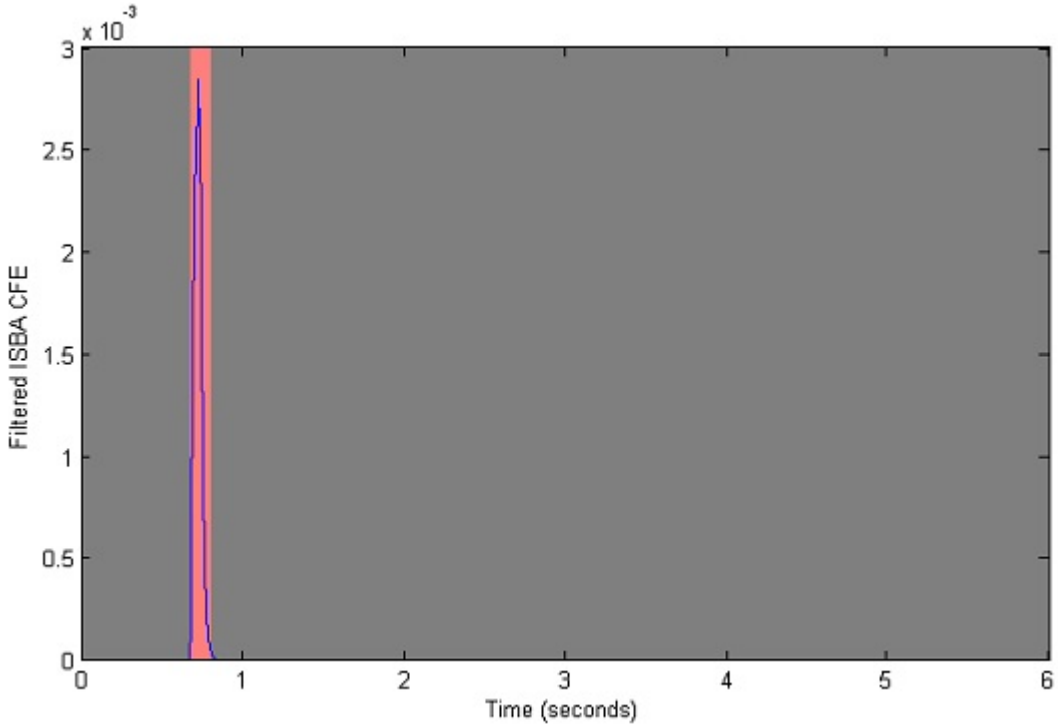

Fig. 6.1. Data points used in a 500 MW case with Method 1

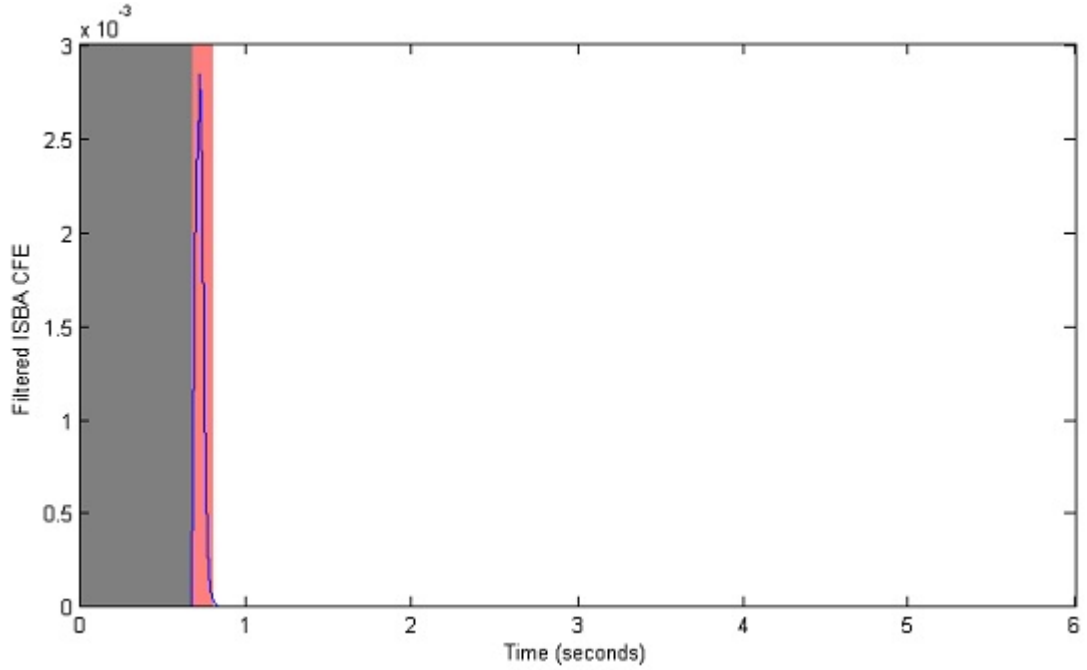

Fig. 6.2. Data points used in a 500 MW case with Method 2

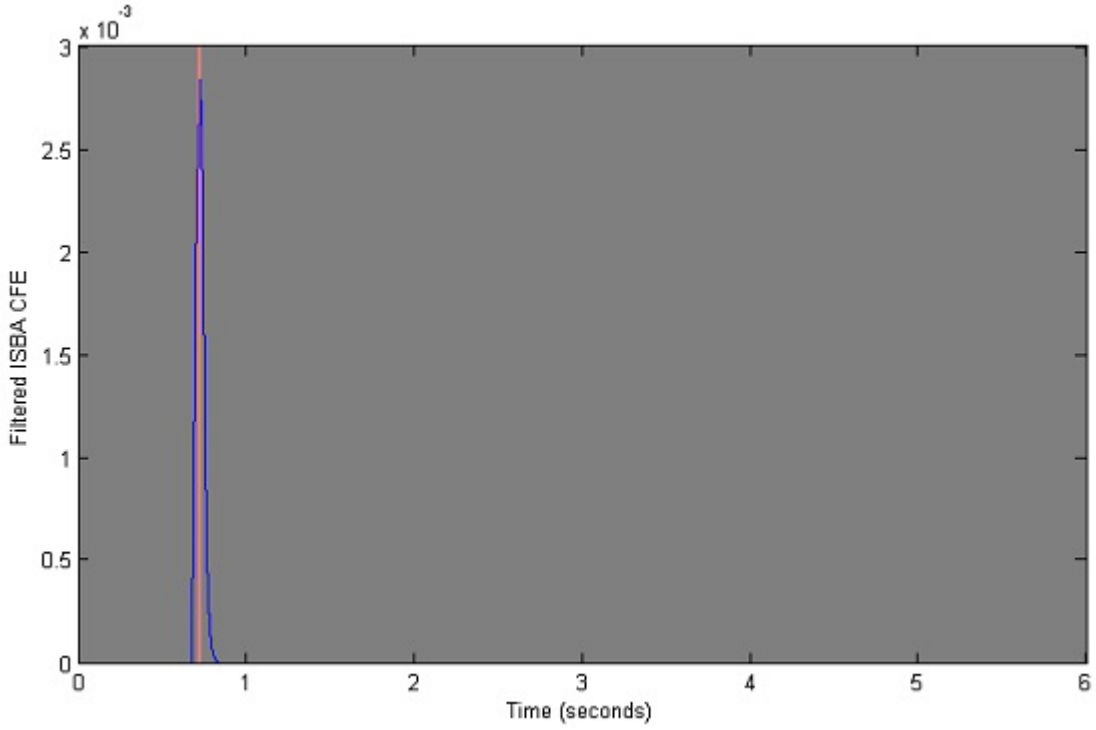

Fig. 6.3. Data points used in a 500 MW case with Method 3

Using the CFE indices of the ISBA and its gradient from each of these four methods, the DT was trained so that it produced a threshold. This threshold was then tested on the 142 cases to determine which method and index/indices would best distinguish between large and small events. During experimentation, it was found that adding the ISBA and its gradient to the two CFE indices already used as input variables to the DT could increase the accuracy of the event detection. By changing the input variables, method of data chosen, and the loss matrix of the DT, different rules for event detection were produced and tested on the 142 cases. A summary of these results can be seen in Table 6.1.

As was discussed in Chapter 5, detecting the start time of an event is only useful in real time response-based control when the clearing time of the event can also be estimated, so that multiple events can be detected. The last two columns in Table 6.1 show for the 500 MW events detected if the end of the events was also detected and the average clearing time of those events. They show that when using the ISBA as one input variable, as was done in the last row, the clearing time of the event is detected extremely late (154 cycles compared to 46-50 cycles), since the ISBA generally increases even after an event ends and only returns to pre-event conditions much later in the simulation, or does not return at all in unstable cases.

6.3 Comparison of Data Points Used for Event Detection

Since no data points after the event ended were used for training the DT in Method 3, the DT produced rules that could detect the occurrence but not necessarily the end of an event. When trained using Method 3, the DT usually chose the ISBA as one input variable, which as just discussed, did not detect the end of the event until long after the event had actually ended. Therefore, it was decided that Method 3 would not be appropriate for event detection.

Since the testing data produced from Method 1 did not include any data points after the events ended, the DT did not have as many data points in the no-fault class,

Table 6.1
Training results for event detection

Input variables	Method	Loss matrix	200 MW events detected	500 MW events detected	End of 500 MW events detected	Average end detection time
CFE of gradient(ISBA)	1	0,10,1,0	3	38	38	46.61 cycles
CFE of gradient(ISBA)	1	0,50,1,0	0	13	12	46.13 cycles
CFE of ISBA	1	0,20,1,0	2	38	34	46.15 cycles
CFE of gradient(ISBA), gradient(ISBA)	2	None	1	42	42	50.33 cycles
CFE of ISBA, ISBA, gradient(ISBA)	3	None	6	65	65	154.07 cycles

so the rules it produced were not as good at distinguishing between the 200 MW and 500 MW events as the rules produced in Method 2. Also, in a few 500 MW cases, the occurrence of an event was detected but the end of the event was not detected using Method 1. For these reasons, it was decided that Method 1 could be eliminated as a possibility for event detection. The last remaining method, Method 2, consistently produced rules that detected the most 500 MW events and the least 200 MW events, and detected the end of all the 500 MW events, so it was chosen as the best option for event detection.

While no loss matrix and set of input variables could completely separate the 200 MW and 500 MW events, adjusting them gave flexibility on the sensitivity of the event detection. In some applications, it might be preferable to detect a lot of events, including small ones, and lower the risk of missing events that could lead to instability, while in other applications, such as systems with a lot of noise in the measurements, it might be preferable to make the thresholds less sensitive so they do not trigger as often.

6.4 Adding White Noise to Bus Angle Measurements

As just mentioned, noise in measurements is a factor in any real power system. In order to test the methods developed for event detection in a more realistic manner, white noise was added to the bus angle measurements. The white noise added was from the standard normal distribution ($\mu = 0, \sigma^2 = 1$), and its magnitude was characterized by the signal-to-noise ratio (SNR). The SNR was chosen to be 50 dB for each bus angle. It was set by visually inspecting a graph of the bus angles with and without the noise. The SNR was increased until it was apparent that the bus angle had some white noise, but the signal was still discernible. For each bus angle, the power of its signal was calculated and the noise was added to the signal so that the value of the SNR was 50 dB. Figures 6.4 and 6.5 compare a bus angle before and after the noise was added.

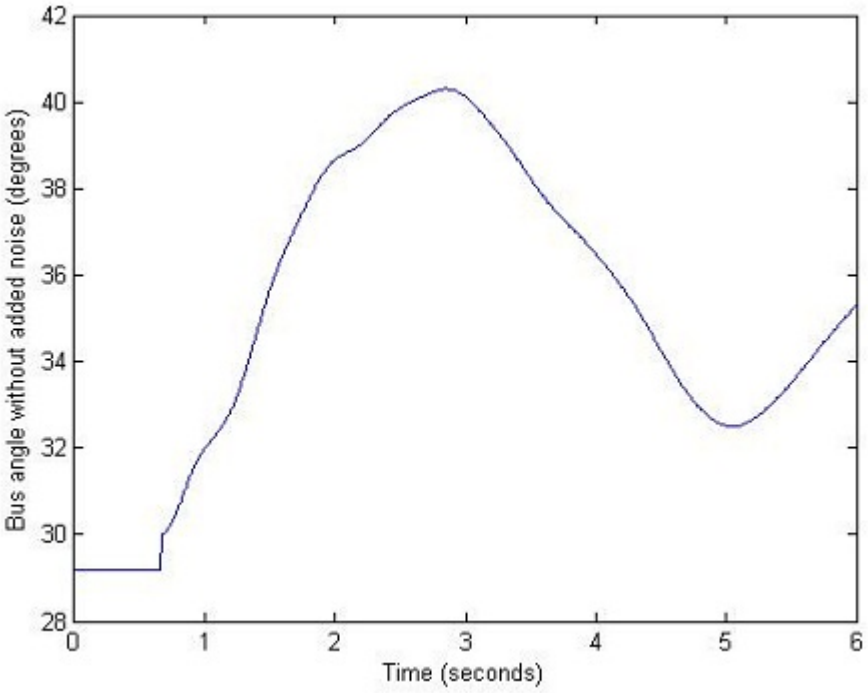

Fig. 6.4. Bus angle measurement with no noise added

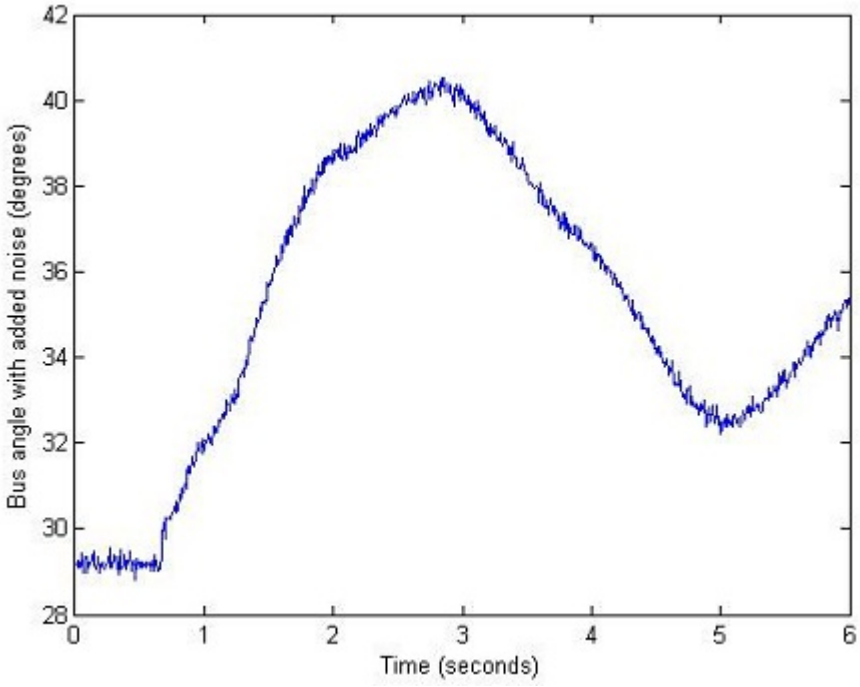

Fig. 6.5. Bus angle measurement with noise added

While adding noise to the system changed the appearance of the individual bus angle measurements, the appearance of the ISBA hardly changed, as can be seen in Figures 6.6 and 6.7. The ISBA with noise was almost identical to the ISBA without noise because the calculation of the ISBA mostly cancelled out the noise from the individual buses, since it is an average of all the bus angles on the system. However, the curve fitting indices changed because of the small fluctuations in the ISBA with noise. The noise from the 200 MW and 500 MW cases caused the spike during the event to be indiscernible, as shown in Figure 6.8.

Because of this noise, the DT could not produce rules for event detection, regardless of the method chosen. Therefore, the threshold had to be set manually by inspecting graphs of the CFE from the 385 cases. When the noise was added to this set of cases, the CFE was noisy, but the faults were still discernible because they were significant enough to cause a large change in the ISBA, as shown in Figure 6.9.

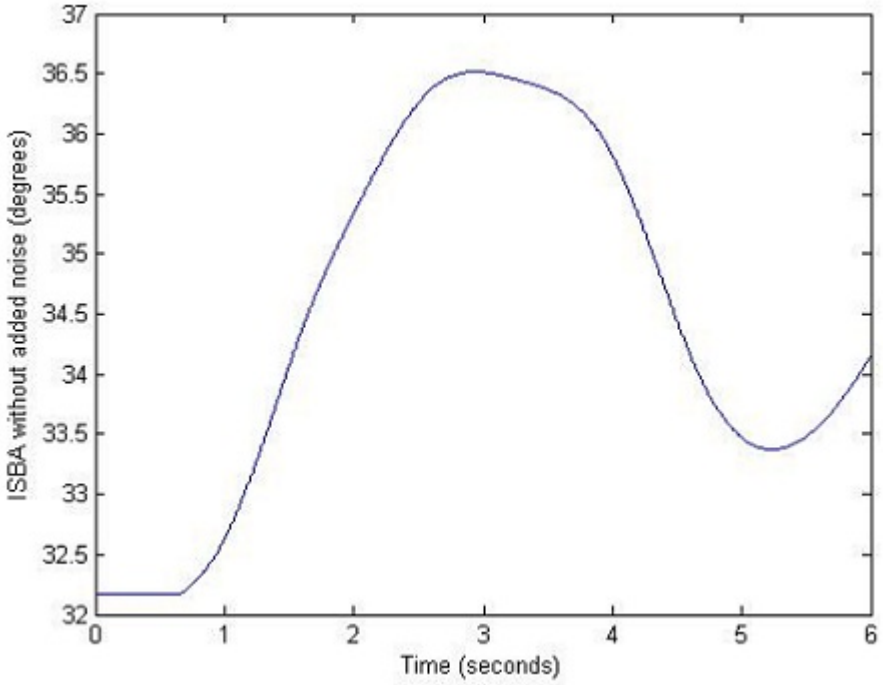

Fig. 6.6. ISBA with no noise added

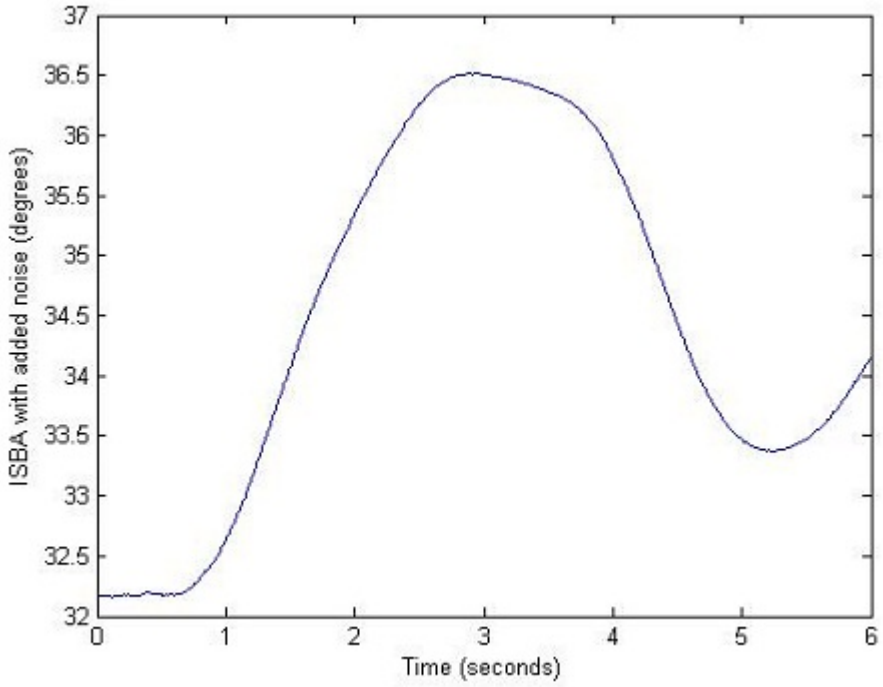

Fig. 6.7. ISBA with noise added

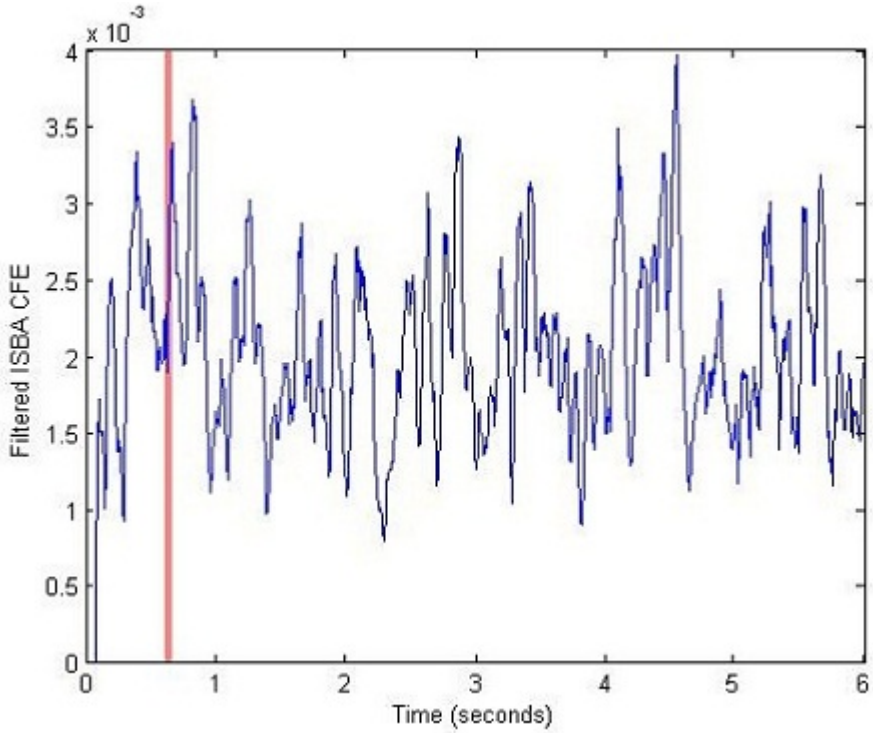

Fig. 6.8. 500 MW case with noise added

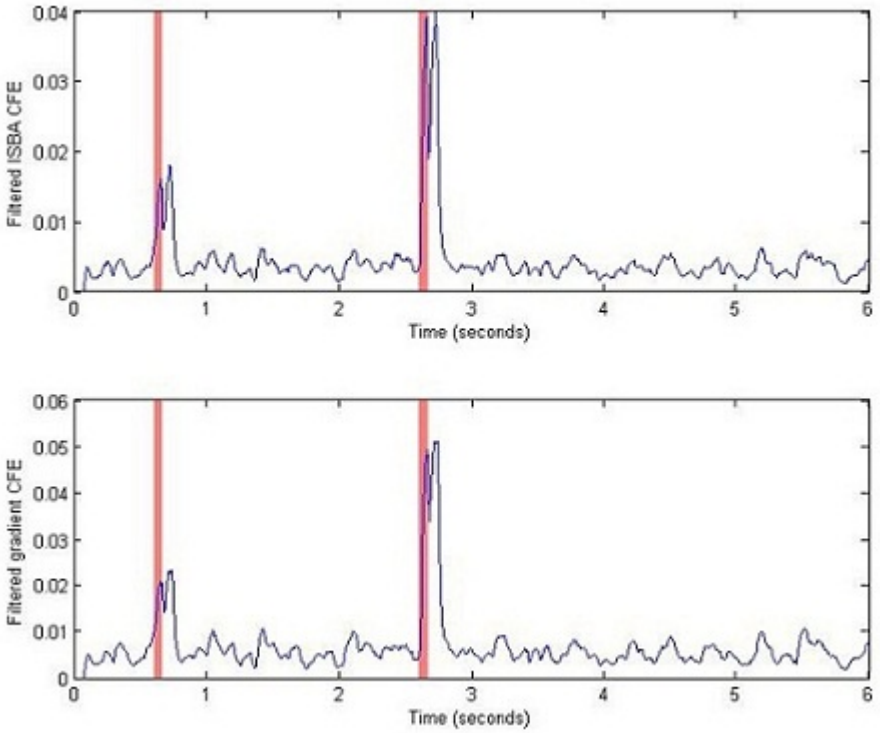

Fig. 6.9. Curve fitting errors from Figure 5.7 with noise added

7. THE TWO DECISION TREE STRUCTURE

7.1 Combining Two Decision Trees

Once the thresholds discussed in the previous chapter that could detect multiple events discussed were obtained, either by DT training or manually setting a threshold, they were combined with instability prediction to create the model shown in Figure 1.4. In this model, DT1 is the decision tree that applies the thresholds for event detection to the CFE indices. When implemented in a real-time system, DT1 would always be active, reading system measurements, calculating the CFE of the ISBA and/or its gradient, and triggering each time the CFE crosses the pre-set threshold. Immediately after the CFE measurement crosses below the threshold, signalling the end of the event, the second decision tree, DT2, would be enabled for a pre-determined amount of time. This second DT would predict impending loss of synchronism and resulting instability, and apply controls to the system in an attempt to prevent the loss of synchronism. If it does not predict instability in the window of time after the end of the event, DT2 would be disabled, and would not be enabled again unless DT1 detects another significant system event.

7.1.1 Window Length of Training Data

To train the second DT to predict instability, it was necessary to apply the CFE thresholds developed in the previous chapter to the test set of 385 cases. In order to produce the training data for DT2, the length of the window of measurements after the detected events ended had to be chosen. Since controls applied in response to a fault become less effective as more time passes, as Figure 1.3 showed, a small window of time, five cycles, was chosen. This window gave the second DT 10 data measurements

to train on and then predict instability when used for testing, but stayed in the time immediately after the fault when controls are most effective. Other window lengths not used in this thesis might be effective as well.

7.1.2 Training the Second Decision Tree

After determining the window length of data points, the measurements used as input variables to the second DT had to be chosen. One input variable used was the ISBA, since it was a measure of the activity of the bus angles across the system. Its gradient was also used, since the ISBA tended to change quickly when the system was about to lose stability. Two other input variables used were V6A, the voltage phase angle of one bus on the model used for the simulations, and V9ADot, the frequency of a voltage phase angle of another bus on the model. These two measurements were chosen as input variables because they were shown to be good predictors of instability in [2]. The second DT was trained with samples of these input variables from the first five cycles after DT1 detected the end of the first fault in each of the 385 cases.

7.2 Testing the Thresholds Obtained From the Two Decision Trees

After the rules for DT2 to predict instability were obtained from the 385 cases, both the event detection and instability prediction thresholds were applied to the 480 cases. Listed below are a summary of the steps taken to obtain these thresholds and apply control to the 480 cases.

1. Train DT1 with 142 cases of 200 MW/500 MW events to obtain thresholds for event detection. Input variables: ISBA and its gradient, CFE of ISBA and its gradient. If noise is present, manually set the threshold.

2. Test event detection thresholds on 385 cases. Use data points from window of five cycles after the first detected fault ends to train DT2 for instability prediction. Input variables: ISBA and its gradient, V6A, V9ADot.

3. Test event detection and instability prediction thresholds on 480 cases. Enable DT2 to predict instability for five cycles after DT1 detects end of first fault. Apply control if instability is predicted.

In each of these steps, there are several variables that can be changed to produce different thresholds for event detection and instability prediction. These adjustments give flexibility to the power system engineers to choose the sensitivity of the event detection and control application. The changes that can be made are:

- Ignoring different input variables for either DT
- Choosing Method 1, 2, or 3 for event detection
- Changing the loss matrix of either DT
- Changing the complexity cost of either DT

Weighting the FN class of the second DT more heavily than the FP class causes the DT to produce thresholds that are less sensitive to triggering, while assigning a larger weight to the FP class produces thresholds that correspond to a larger unstable region and are therefore more likely to trigger.

7.3 Comparison With Single Decision Tree Structure

The two DT structure was tested on the set of 480 cases for several different parameters to show how it compares to the single decision tree structure that was predicting instability continuously used in [2]. Figure 7.1 plots the number of cases that had control applied because instability was predicted versus the number of those cases that were stabilized out of the 480 cases. The first five data points were from the 1 DT model, while the last 5 were from the 2 DT model. For all of the data points from the 2 DT model, Method 2 was used for event detection. Following is a description of the parameters used to obtain each point.

1. Loss matrix of 0,35,1,0 on DT; input variable: ISBA

2. Loss matrix of 0,10,1,0 on DT; input variables: ISBA, gradient(ISBA)
3. Loss matrix of 0,50,1,0 on DT; input variables: V6A, V9ADot
4. Loss matrix of 0,20,1,0 on DT; input variables: V6A, V9ADot
5. No loss matrix on DT; input variables: V6A, V9ADot
6. No loss matrix on DT1; loss matrix of 0,10,1,0 on DT2; input variables for DT2: V6A, V9ADot, gradient(ISBA)
7. Loss matrix of 0,1,20,0 on DT1; no loss matrix on DT2; input variables for DT2: V9ADot, gradient(ISBA)
8. Manual threshold of 0.003 applied to CFE of ISBA for DT1; no loss matrix on DT2; input variables for DT2: V9ADot, gradient(ISBA)
9. No loss matrix on DT1; loss matrix of 0,1,10,0 on DT2; input variables for DT2: V9ADot, ISBA, gradient(ISBA)
10. No loss matrix on DT1; loss matrix of 0,1,30,0 on DT2; input variables for DT2: V9ADot, ISBA

As Figure 7.1 shows, the 2 DT structure provides improved control response over the 1 DT structure. All of the points associated with the 2 DT structure can stabilize a similar number of cases as the 1 DT structure, but with fewer applications of control to the system.

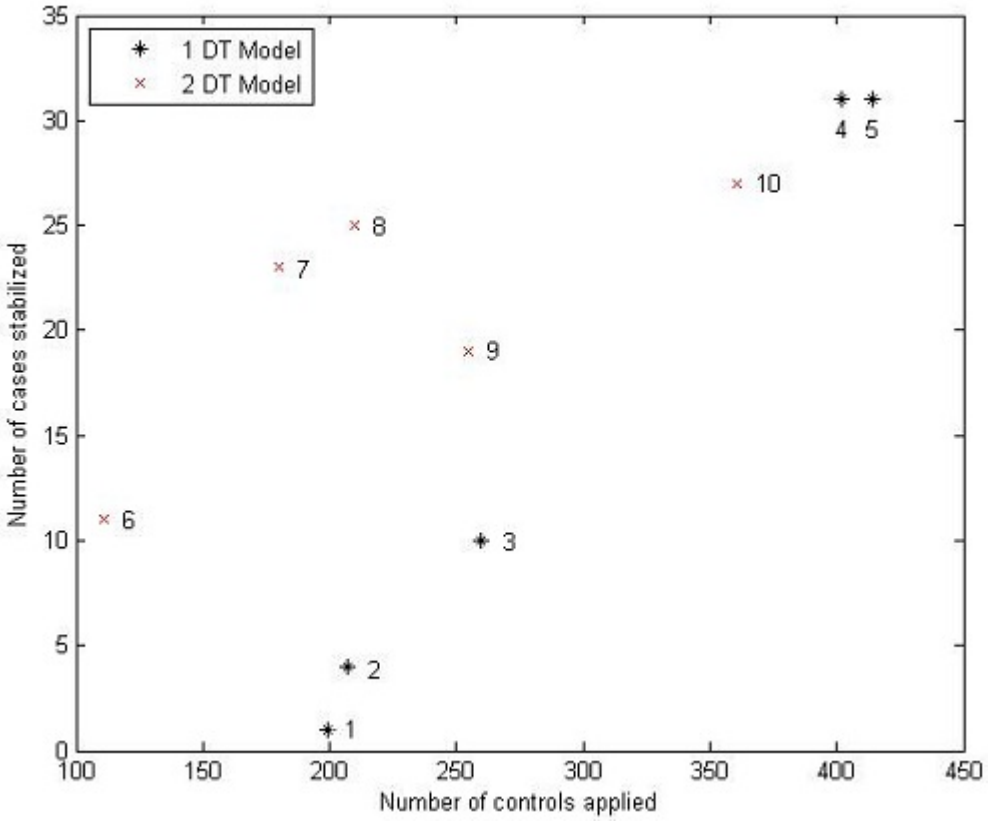

Fig. 7.1. Comparison of 1 and 2 DT Models

8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

Event detection can be used to improve the application of response-based controls in power systems. Measurements of the overall stress of the system, such as the ISBA, contain information that can be used to detect events. These types of measurements are more reliable than individual bus measurements, which are susceptible to glitches. Fitting a polynomial curve using the method of least squares to a moving window of ISBA measurements provided an index that could detect events anywhere on the simulated power system. It was shown that a DT training program could produce a threshold that would trigger for most significant events on the system but ignore small events and glitches. It was also shown that event detection could be combined with instability prediction in order to keep the system stable in several test cases, and that this model performed better than a single DT predicting instability without event detection.

8.2 Recommendations for Future Improvements

Other indices not used in this thesis could be tried to improve the accuracy of event detection, even in the presence of noise. One possibility is to not only determine if an event occurred based on the height of the CFE spike, but also its width, since events with longer fault times are more likely to cause instability. In this case, the CFE would have to stay above the set threshold for a set amount of time before an event would be detected. Another index that might contain accurate information about when an event is occurring is some measurement of the energy in the ISBA.

To increase the number of cases stabilized when event detection is combined with instability prediction, different types of control actions could be tried. In this thesis, only one control action consisting of fast power changes on two HVDC lines was used, but changing power flows on different lines or tripping generators could be better for maintaining system stability.

Another extension of the work presented in this thesis would be to detect not only if an event occurred, but also its approximate location on the system. If the location of the event is known and instability is predicted, then it may be possible to choose a control action that would be most effective for that location, as opposed to a general control action applied regardless of the location of the event on the system.

LIST OF REFERENCES

LIST OF REFERENCES

[1] S. M. Rovnyak, S. E. Kretsinger, J. S. Thorp, and D. E. Brown, "Decision trees for real-time transient stability prediction," *IEEE Transactions on Power Systems*, vol. 9, pp. 1417-1426, August 1994.

[2] Q. Gao and S. M. Rovnyak, "Decision trees using synchronized phasor measurements for wide-area response-based control," *IEEE Transactions on Power Systems*, vol. 26, pp. 855-861, May 2011.

[3] G. Li and S. M. Rovnyak, "Integral square generator angle index for stability ranking and control," *IEEE Transactions on Power Systems*, vol. 20, pp. 926-934, May 2005.

[4] S. M. Rovnyak, G. Li, and K. Mei, "One-shot controls for preventing loss of synchronism," in *Proceedings of the IEEE PES General Meeting*, vol. 4, July 2003.

[5] V. Brandwajn, A. B. R. Kumar, A. Ipakchi, A. Bose, and S. D. Kuo, "Severity indices for contingency screening in dynamic security assessment," *IEEE Transactions on Power Systems*, vol. 12, pp. 1136-1142, August 1997.

[6] S. M. Rovnyak and K. Mei, "Dynamic event detection and location using wide area phasor measurements," *European Transactions on Electrical Power*, vol. 21, pp. 1589-1599, May 2011.

[7] C. W. Taylor, D. C. Erickson, K. E. Martin, R. E. Wilson, and V. Venkatasubramanian, "WACS-Wide-Area Stability and Voltage Control System: R D and Online Demonstration," *Proceedings of the IEEE*, vol. 93, pp. 892-906, May 2005.

[8] PowerTech Labs Inc., "TSAT: Transient Security Assessment Tool," June 2010. [Online]. Available: http://www.dsatools.com/downloads/TSAT.pdf. Last accessed April 2012.

[9] S. M. Rovnyak, M. N. Nilchi, D. W. Longbottom, and D. C. Vasquez, "Angle stability predictive indices," *Proceedings of the IEEE PES General Meeting*, July 2012.

[10] G. J. Williams, "Rattle: A Data Mining GUI for R," *The R Journal*, vol. 1, no. 2, pp. 45-55, 2009.

[11] K. Mei, S. M. Rovnyak, and C.-M. Ong, "Dynamic event detection using wavelet analysis," in *Proceedings of the IEEE PES General Meeting*, June 2006.

[12] S. M. Rovnyak, G. Li, and K. Mei, Study Support for Wide Area Monitoring System Enhancement, Entergy/NSF, Tech. Rep. for Entergy Res. Agreement 10020386, 2003. [Online]. Available: http://www.engr.iupui.edu/ srovnyak/pubs/NSF_Entergy03.pdf. Last accessed April 2012.

APPENDIX

APPENDIX: MATLAB CODE FOR POWER SYSTEM SIMULATIONS

```
clc
clear all;

% control lines consisting of power changes on two HVDC lines
confile(1) = cellstr('ADD ADMITTANCE
;ADELANTO 500  -500.0   -200.0 MVA /');
confile(2) = cellstr('ADD ADMITTANCE
;INTERMT  345   500.0    200.0 MVA /');
confile(3) = cellstr('ADD ADMITTANCE
;SYLMARLA 230   -500.0    -200.0 MVA /');
confile(4) = cellstr('ADD ADMITTANCE
;CELILO   230    500.0     200.0 MVA /');

% counters for cases after control is applied
itotUnst = 0;
itotLose = 0;
itotStab = 0;
itotSave = 0;

% open csv file for writing training data for DT
fid=fopen('Wtrain.csv','w+');
fprintf(fid,'%s\n','istab,t,isba,gradIsba,V6A,V9ADot');
fclose(fid);
```

```
% open file containing cases
fid1 = fopen('swifile.txt', 'r');
nline=0;
% count number of lines in swifile.txt
while ~feof(fid1)
    line = fgetl(fid1);
    nline = nline + 1;
end
nevent = 0; % counter for number of events in swifile.txt
% read swifile.txt
swifile = textread('swifile.txt','%s','delimiter','\n','whitespace','');
% find event locations in swifile
for iline=1:nline
    record = swifile(iline);
    record = char(record);
    findspace = isspace(record);
    if ~isempty(record) && ~findspace(1)
        if strcmp(record(1:5),'DESCR')
            nevent = nevent + 1;
            swistart(nevent) = iline;
        end
        if strcmp(record(1:5),'NOMOR')
            swistop(nevent) = iline;
        end
    end
end
fclose(fid1);
```

```
% for each event do the following
for ievent = 1:nevent

    %-----------------RUN THE EVENT WITH NO CONTROL-----------------%
    % write fault.swi file
    fid2 = fopen('fault.swi','w+');
    fseek(fid2,0,-1);
    for irec = swistart(ievent):swistop(ievent)
        charline = char(swifile(irec));
        fprintf(fid2,'%s\r\n',charline);
    end
    fclose(fid2);

    % time delay
    time = timer('TimerFcn',@(x,y)disp('Hello World!'),'StartDelay',10);
    start(time);
    wait(time);

    % call TSAT
    !C:\dsa_powertools_5_net\tsat\bin\tsat_batch.exe  C:\Document\Course06\
    TSAT\WSCC29\WECC29.tsa
    !C:\dsa_powertools_5_net\tsat\bin\sim2txt WECC29.bin -quan=gen_relang
    -all > WECC29GA.csv
    !C:\dsa_powertools_5_net\tsat\bin\sim2txt WECC29.bin -quan=bus_va -all
    > WECC29BA.csv

    % read generator angles
    gen_angles=dlmread('WECC29GA.csv', '', 1, 0);
    % read bus angles
```

```
bus_angles=dlmread('WECC29BA.csv','', 1, 0);
% store number of generators and angles
[length1,Ngen]=size(gen_angles);
diffmax(ievent) = -1.0; % variable for largest angle difference
% calculate max generator angle difference
for z = 1:length1
    for x = 2:Ngen
        for y = x+1:Ngen
            % take difference of angles
            diffa = abs(gen_angles(z,y) - gen_angles(z,x));
            if diffa > diffmax(ievent)
                % store larger of two angle differences
                diffmax(ievent) = diffa;
            end
        end
    end
end
fprintf('%f\n',diffmax(ievent));
% store number of buses and angles
[length2,Nbus]=size(bus_angles);

% unwrap bus_angles and store in newbus_angles
for z = 2:Nbus
    newbus_angles(1,z) = bus_angles(1,z);
    addangles(1,z) = 0;
end
for irec = 2:length2
    for z = 2:Nbus
        % compare angles at consecutive time steps
```

```
        diffa = bus_angles(irec,z) - bus_angles(irec-1,z);
        addangles(irec,z) = addangles(irec-1,z);
        % add 360 deg to bus angles if diff is less than -180 deg
        if diffa <= -180
            addangles(irec,z) = addangles(irec-1,z) + 360;
        end
        % subtract 360 degrees from bus angles if diff is greater than
        % 180 degrees
        if diffa >= 180
            addangles(irec,z) = addangles(irec-1,z) - 360;
        end
        diffa = bus_angles(irec,z) + addangles(irec,z);
        newbus_angles(irec,z) = diffa;
    end
end

% add random noise with SNR = 50 to the bus angle measurements
for z = 2:Nbus
    newbus_angles(:,z) = awgn(newbus_angles(:,z),50,'measured');
end

% make one bus be the reference angle
for irec = 1:length2
    coa(irec) = 0.0;
    for z = 2:Nbus
        diffa = newbus_angles(irec,z) -  newbus_angles(irec,18);
        newbus_angles(irec,z) = diffa;
        coa(irec) = coa(irec) + newbus_angles(irec,z)/17.0;
    end
```

```
end

% calculate the isba
for irec = 2:length2
    sba(irec) = 0.0;
    for z = 2:Nbus
        diffa = newbus_angles(irec,z) - coa(irec);
        sba(irec) = sba(irec) + diffa*diffa/17.0;
    end
    sba(irec) = sqrt(sba(irec));
    sba(1) = sba(2);
    isba(1) = sba(1);
    % pass the sba through a LPF to get the isba
    isba(irec) = isba(irec-1) + 6*1/120*(sba(irec-1) - isba(irec-1));
end

% store time for each measurement
for z = 1:length2
    t(z) = bus_angles(z,1);
end

% calculate successive bus angle differences
% set velocity for first sample to be zero
for z = 2:Nbus
    bus_freqs(1,z) = 0;
end
for irec = 2:length2
    for z = 2:Nbus
        diffa = newbus_angles(irec,z) - newbus_angles(irec-1,z);
```

```
        if diffa < 180
            diffa = diffa + 360;
        end
        if diffa > 180
            diffa = diffa - 360;
        end
        deltat = max(t(irec) - t(irec-1), 0.1*(t(9) - t(1)));
        bus_freqs(irec,z) = diffa/deltat;
    end
end

% determine stability of case without control
istab(ievent) = 1;
if diffmax(ievent)>300
    istab(ievent) = 0;
end

% calculate variance of all angles and frequencies
for irec = 1:length2
    for z = 2:Nbus
        if irec<20
            GenVars(irec,z) = var(gen_angles(1:irec,z));
            varAngles(irec,z) = var(newbus_angles(1:irec,z));
            varFreqs(irec,z) = var(bus_freqs(1:irec,z));
        else
            GenVars(irec,z) = var(gen_angles(irec-19:irec,z));
            varAngles(irec,z) = var(newbus_angles(irec-19:irec,z));
            varFreqs(irec,z) = var(bus_freqs(irec-19:irec,z));
        end
```

```
    end
    if irec<20
        varIsba(irec) = var(isba(1:irec));
    else
        varIsba(irec) = var(isba(irec-19:irec));
    end
end
% calculate gradient of isba
gradIsba = [0 diff(isba)];

% calculate curve fitting errors
x = (1:10); % length of curve fitting window
k = 4; % degree of polynomial
% calculate the matrix X
for i=1:10
    for j=1:k+1
        X(i,j) = x(i)^(j-1);
    end
end
% calculate CFE of isba
for i=10:length2
    y = isba(i-9:i);
    a = (X'*X)\X'*y';
    ynew = (X*a)';
    error(i) = max(abs(ynew-y));
end
% calculate CFE of gradient of isba
for i=10:length2
```

```
    y = gradIsba(i-9:i);
    a = (X'*X)\X'*y';
    ynew = (X*a)';
    error1(i) = max(abs(ynew-y));
end
% calculate CFE of variance of isba
for i=10:length2
    y = varIsba(i-9:i);
    a = (X'*X)\X'*y';
    ynew = (X*a)';
    error2(i) = max(abs(ynew-y));
end
% use averaging filter on CFE
for i=20:length2
    eavg(i) = sum(error(i-19:i))/20;
    eavg1(i) = sum(error1(i-19:i))/20;
end
% use LPF on CFE
eLPF(1) = error(1);
eLPF1(1) = error1(1);
for i=1:length2
    ep = 50*(error(i)-eLPF(i));
    eLPF(i+1) = eLPF(i) + (1/120)*ep;
    ep1 = 50*(error1(i)-eLPF1(i));
    eLPF1(i+1) = eLPF1(i) + (1/120)*ep1;
end

% detect and count faults
nfault(ievent) = 0; % counter for number of detected events in case
```

```
faulton(1) = 0; % variable that is 1 whenever an event is occurring
for irec=2:length2
    if eLPF(irec) < .05 % event detection threshold from chapter 6
        if faulton(irec-1)==1
            % stop time of each event
            istop(ievent,nfault(ievent)+1) = irec;
            nfault(ievent) = nfault(ievent)+1;
        end
        faulton(irec) = 0;
    else
        if faulton(irec-1)==0
            % start time of each event
            istart(ievent,nfault(ievent)+1) = irec;
        end
        faulton(irec) = 1;
    end
end

% check for instability for 5 cycles after the end of each event
icontrol(ievent) = 0; % variable that contains time control is applied
if nfault(ievent)>0
    for icount=1:nfault(ievent)
        if (istop(ievent,icount)+10)<=length2
            for irec=istop(ievent,icount)+1:istop(ievent,icount)+10
                ipred(irec) = 0; % variable used to predict instability
                % rules generated for instability prediction
                if bus_freqs(irec,10)<-1.22
                    ipred(irec) = 1;
                elseif bus_freqs(irec,10)>=-1.22 && bus_freqs(irec,10)<
```

```
                12.95 && gradIsba(irec)>=.03968
                    ipred(irec) = 1;
                end
                % apply control at time when instability is first
                % detected
                if ipred(irec) == 0 && icontrol(ievent) == 0
                    icontrol(ievent) = irec;
                end
            end
        end
    end
end
% in this example, write training data for instability prediction
% use measurements from 5 cycles after end of first detected event
if nfault(ievent)>0
    for irec=1:(istop(ievent,1)+1):(istop(ievent,1)+10)
        DT(irec,1) = istab(ievent);
        DT(irec,2) = t(irec);
        DT(irec,3) = isba(irec);
        DT(irec,4) = gradIsba(irec);
        DT(irec,5) = newbus_angles(irec,7);
        DT(irec,6) = bus_freqs(irec,10);
    end
    dlmwrite('Wtrain.csv',DT,'-append');
end

clear DT bus_angles gen_angles newbus_angles bus_freqs length1 Ngen;
clear diffa length2 Nbus addangles t deltat DT sba isba;
```

```
%-----------------RUN THE EVENT WITH CONTROL-----------------%
delete('WECC29GA.csv');
delete('WECC29BA.csv');
% write fault.swi file
fid2 = fopen('fault.swi','w+');
fseek(fid2,0,-1);
for irec = swistart(ievent):swistop(ievent)-1
    charline = char(swifile(irec));
    fprintf(fid2,'%s\r\n',charline);
end
% if instability is detected, add control lines to swifile
if icontrol(ievent)>0
    % add 6 cycle delay between detection and time control is applied
    ncyc = 6 + icontrol(ievent)/2;
    fprintf(fid2, 'AT TIME %4.0f CYCLES /\r\n', ncyc);
    for irec = 1:4
        charline = char(confile(irec));
        fprintf(fid2,'%s\r\n',charline);
    end
end
fprintf(fid2,'%s\r\n', char(swifile(swistop(ievent))));
fclose(fid2);
% time delay
time = timer('TimerFcn',@(x,y)disp('Hello World!'),'StartDelay',15);
start(time);
wait(time);
% call TSAT
!C:\dsa_powertools_5_net\tsat\bin\tsat_batch.exe  C:\Document\Course06\
```

```
TSAT\WSCC29\WECC29.tsa
!C:\dsa_powertools_5_net\tsat\bin\sim2txt WECC29.bin -quan=gen_relang
-all > WECC29GA.csv
!C:\dsa_powertools_5_net\tsat\bin\sim2txt WECC29.bin -quan=bus_va -all
> WECC29BA.csv

% read generator angles
gen_angles=dlmread('WECC29GA.csv', '', 1, 0);
% read bus angles
bus_angles=dlmread('WECC29BA.csv','', 1, 0);
% store number of generators and angles
[length1,Ngen]=size(gen_angles);
diffmax2(ievent) = -1.0; % variable for largest angle difference
% calculate max generator angle difference
for z = 1:length1
    for x = 2:Ngen
        for y = x+1:Ngen
            % take difference of angles
            diffa = abs(gen_angles(z,y) - gen_angles(z,x));
            if diffa > diffmax2(ievent)
                % store larger of two angle differences
                diffmax2(ievent) = diffa;
            end
        end
    end
end
% store number of buses and angles
[length2,Nbus]=size(bus_angles);
```

```
% unwrap bus_angles and store in newbus_angles
for z = 2:Nbus
    newbus_angles(1,z) = bus_angles(1,z);
    addangles(1,z) = 0;
end
for irec = 2:length2
    for z = 2:Nbus
        % compare angles at consecutive time steps
        diffa = bus_angles(irec,z) - bus_angles(irec-1,z);
        addangles(irec,z) = addangles(irec-1,z);
        % add 360 deg to bus angles if diff is less than -180 deg
        if diffa <= -180
            addangles(irec,z) = addangles(irec-1,z) + 360;
        end
        % subtract 360 degrees from bus angles if diff is greater than
        % 180 degrees
        if diffa >= 180
            addangles(irec,z) = addangles(irec-1,z) - 360;
        end
        diffa = bus_angles(irec,z) + addangles(irec,z);
        newbus_angles(irec,z) = diffa;
    end
end

% add random noise with SNR = 50 to the bus angle measurements
for z = 2:Nbus
    newbus_angles(:,z) = awgn(newbus_angles(:,z),50,'measured');
end
```

```
% make one bus be the reference angle
for irec = 1:length2
    coa(irec) = 0.0;
    for z = 2:Nbus
        diffa = newbus_angles(irec,z) -  newbus_angles(irec,18);
        newbus_angles(irec,z) = diffa;
        coa(irec) = coa(irec) + newbus_angles(irec,z)/17.0;
    end
end

% store time for each measurement
for z = 1:length2
    t(z) = bus_angles(z,1);
end

% calculate successive bus angle differences
% set velocity for first sample to be zero
for z = 2:Nbus
    bus_freqs(1,z) = 0;
end
for irec = 2:length2
    for z = 2:Nbus
        diffa = newbus_angles(irec,z) - newbus_angles(irec-1,z);
        if diffa < 180
            diffa = diffa + 360;
        end
        if diffa > 180
            diffa = diffa - 360;
        end
```

```
        deltat = max(t(irec) - t(irec-1), 0.1*(t(9) - t(1)));
        bus_freqs(irec,z) = diffa/deltat;
    end
end

% determine stability of case with control
istabnew(ievent) = 1;
if diffmax2(ievent) > 300
    istabnew(ievent) = 0;
end

% write cases stabilized to new file
if (istab(ievent) == 0 ) && (istabnew(ievent) == 1)
    itotSave = itotSave + 1;
    fid3=fopen('C:\Document\Course06\TSAT\WSCC29\swifile.txt.save',
    'a+');
    for irec = swistart(ievent):swistop(ievent)-1
        charline = char(swifile(irec));
        fprintf(fid3,'%s\r\n',charline);
    end
    if icontrol(ievent)>0
        ncyc = 6 + icontrol(ievent)/2;
        fprintf(fid3, 'AT TIME %4.0f CYCLES /\r\n', ncyc);
        for irec = 1:4
            charline = char(confile(irec));
            fprintf(fid3,'%s\r\n',charline);
        end
    end
    fprintf(fid3,'%s\r\n', char(swifile(swistop(ievent))));
```

```
        fclose(fid3);
    end

    % count cases destabilized by control
    if (istab(ievent) == 1) && (istabnew(ievent) == 0)
        itotLose = itotLose + 1;
    end
    % count cases kept stable
    if (istab(ievent) == 1 ) && (istabnew(ievent) == 1)
        itotStab = itotStab + 1;
    end
    % count cases kept unstable
    if (istab(ievent) == 0 ) && (istabnew(ievent) == 0)
        itotUnst = itotUnst + 1;
    end
end

fprintf('Events Stabilized: %4.0f\n',itotSave);
fprintf('Events Destabilized: %4.0f\n',itotLose);
fprintf('Events Kept Stable: %4.0f\n', itotStab);
fprintf('Events Kept Unstable: %4.0f\n', itotUnst);
```