CHAPTER 4

PROPOSED ISLANDING DETECTION TECHNIQUE

The proposed technique is based on the identification of the transients associated with islanding, and differentiating them from the transients associated with non-islanding events such as faults, switching, etc.

In this chapter, the proposed Wavelet Transform - ANN based islanding detection technique is explained. The chapter starts with an introduction to wavelet transform including its theory and choice of mother wavelet. Section two gives an overview of ANN followed by an explanation of the training process. Section three presents the methodology of the proposed Wavelet Transform ANN technique.

4.1 Wavelet Transform

Real time transients classification of power transients is very challenging since the high frequency content superimposed on the power frequency signals are usually aperiodic, short term and non stationary waveforms. Wavelet transform is proposed in order to extract discriminative features which will help in differentiating between transients associated with islanding event and those created from any other event such as switching of capacitor bank and temporary fault.

Wavelet transform (WT) is an effective mathematical tool which has been widely used in many engineering applications such as speech and image processing. WT has found many numerous applications in the power systems field some of the applications are power system protection, power quality, and partial discharge.

Unlike Fourier transform (FT) which transforms the signal from the time domain to the frequency domain. The WT extract the frequency components of the signal while preserving the time domain properties. The theory of WT will be explained in the following section.

4.1.1 Theory of Wavelet Transform

Traditionally, the FT has been used extensively in signal processing to analyze stationary or time-invariant signals. In FT the analyzed signal is decomposed into a combination of sinusoidal waves having different frequencies. The FT is defined in (4. 1).

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \tag{4.1}$$

Equation (3.1) defines the Fourier transform $X(f)$ at the specific frequency f to be the time integral over all time of the product of a given signal $x(t)$ with a complex sinusoid at the specified frequency f . If the signal has a strong component at the particular frequency f , the FT will be significant. Otherwise it will be negligible. Since FT is the sum of the coefficient all over time, the coefficient does not give an indication of the time at which a certain frequency exists. Thus, FT provides perfect frequency resolution but no time resolution. In order to overcome the deficiency of FT, a short-time Fourier Transform (STFT) was developed. In STFT the analyzed signal is segmented or windowed using a chosen window. STFT will map the analyzed signal into both time and frequency domain. In other words, STFT computes the FT on the segmented portion of the analyzed signal. If the window size chosen is large then, a larger portion of the segmented signal is considered resulting in good frequency resolution but poor time resolution. On the contrary, if the window size chosen is small, then a smaller portion of the segmented signal will be considered resulting in good time resolution but poor frequency resolution. STFT compromise between time and frequency information could be useful depending on the application. However, once the time window size is chosen it will be fixed for all frequencies. Many signals require a more flexible approach where the window size could be varied to determine more accurately either time or frequency. WT analysis represents the next logical step: a windowing technique with variable-sized regions. WT analysis allows the use of long time window intervals where more precise low-frequency information is needed, and shorter regions where high-frequency information is desired. Hence, Wavelet analysis is capable of providing time localization of the analyzed signal. Similar to FT which breaks the signal into

sinusoidal waves of different frequencies; the wavelet transform decomposes transients into a series of wavelet components, each of which corresponds to a time domain signal that covers a specific frequency band containing more detailed information. Wavelets localize the information in the time frequency plane which is suitable for the analysis of non stationary signals. WT divides up data, functions into different frequency components, and then studies each component with a resolution matched to its scale. In this study, the negative sequence voltage signals are used as the input signals of the wavelet analysis. Daubechies4 (dB4) mother wavelet, is employed since it has been demonstrated to perform well [30, 31]. The islanding of the study cases is detected through discrete wavelet transform (DWT). Both approximation and details information related fault voltages are extracted from the original signal. When the utility grid isolates, it can be seen that variations within the decomposition coefficient of the voltage signals contain useful signatures. Filters of different cut-off frequencies are used to analyze the signal at different scales. The signal is passed through a series of high pass filters to analyze the high frequencies, and it is passed through a series of low pass filters to analyze the low frequencies. Hence the signal (S) is decomposed into two types of components approximation (C) and detail (D). The approximation (C) is the high scale, low-frequency component of the signal. The detail (D) is the low-scale, high-frequency components. The decomposition process can be iterated, with successive approximations being decomposed in turn, so that one signal is divided into many lower resolution components which is called the wavelet decomposition tree and is shown in Fig. 4.1.

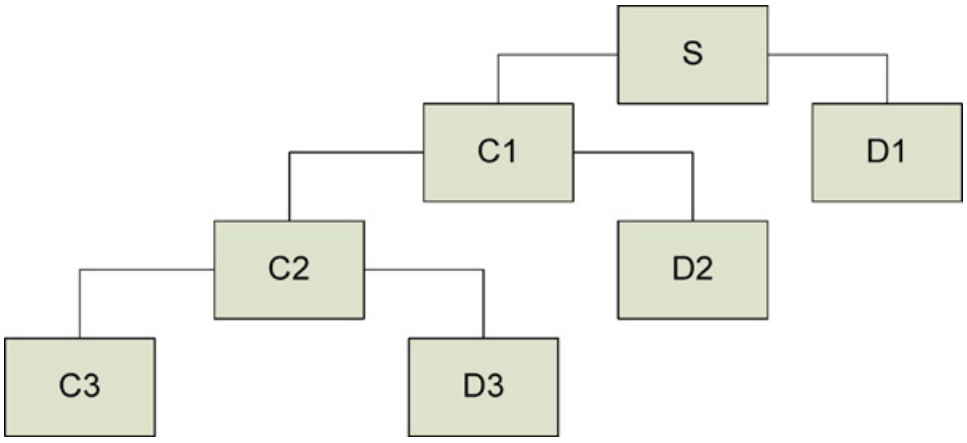

Figure 4.1: Wavelet decomposition tree.

As decompositions are done on higher levels, lower frequency components are filtered out progressively. The wavelet transform not only decomposes a signal into frequency bands, but also, unlike the Fourier transform, provides a non-uniform division of the frequency domain (i.e., the wavelet transform uses short windows at high frequencies and long windows for low frequency components). Wavelet analysis deals with expansion of functions in terms of a set of basis functions (wavelets) which are generated from another wavelet by operations of dilatations and translations. Given a function $v(t)$, its continuous wavelet transform (CWT) can be calculated as

$$CWT(v,x,y) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} v(t)\psi^*(\frac{t-y}{x})dt \tag{4.2}$$

Where x and y are scaling (dilation) and translation (time shift) constants, respectively, and ψ is the wavelet function. Wavelet transform of sampled waveform can be obtained by implementing the discrete wavelet transform as:

$$DWT(v,x,y) = \frac{1}{\sqrt{x_0^m}} \sum_k v(k)\psi^*(\frac{n-kx_0^m}{x_0^m}) \tag{4.3}$$

Where the parameters x and y in Eq. 3.2 are replaced by x_0^m and kx_0^m , k and m being integer variables. In a standard DWT, the coefficients are sampled from the CWT on a dyadic grid. A scaling function $\varphi(t)$ is associated with the wavelet function process multi-resolution analysis (MRA) of the taken signal. The scaling function of one level can be represented as a sum of a scaling function of the next finer level and is given by:

$$\varphi(t) = \sum_{n=-\infty}^{\infty} h(n)\sqrt{2}\varphi(2t-n) \tag{4.4}$$

The wavelet function can be expressed in terms of the scaling function by:

$$\psi(t) = \sum_{n=-\infty}^{\infty} h_1(n)\sqrt{2}\varphi(2t-n) \tag{4.5}$$

Where $h(k)$ and $h_1(k)$ represent the scaling function and wavelet function and are related by the expression:

$$h_1(k) = (-1)^k h(1-k) \tag{4.6}$$

If $v(t) \in R_{j+1} =$ span$\{2^{(j/2)}\phi(2^j t-k)\}$, then it can be expressed as the linear combination of the elements of vector space R_{j+1} :

$$v(t)=\sum_{k=-\infty}^{\infty} c_{j+1}(k)2^{(j+1)/2}\phi(2^{j+1}t-k) \tag{4.7}$$

Where $2^{j/2}$ in each term maintains the unity norm of the basis functions at various scales. Using the scaling function, the signal can be expressed as [32]:

$$v(t)=\sum_{-\infty}^{\infty} c_{j_0}(k)2^{j_0/2}\phi(2^{j_0}t-k)+\sum_{k=-\infty}^{\infty}\sum_{j=j_0}^{\infty} d_j(k)2^{j/2}\phi(2^j t-k) \tag{4.8}$$

Where j_0 represents the coarsest scale spanned by the scaling function. The scaling and wavelet coefficients of the signal $v(t)$ can be evaluated by using a filter bank of quadrature mirror filters given as:

$$c_j(k)=\sum_{m=-\infty}^{\infty} c_{j+1}(m)h(m-2k) \tag{4.9}$$

$$d_j(k)=\sum_{m=-\infty}^{\infty} c_{j+1}(m)h_1(m-2k) \tag{4.10}$$

Eqs. (4.9) and (4.10) shows that the coefficients at coarser level can be attained by passing the coefficients at the finer level to their respective filter followed by a decimation of two. Implementation of DWT involves successive pairs of high pass and low pass filters at each scaling stage of wavelet transform. This can be thought as successive approximations of the same function, each approximation providing the incremental information related to a particular scale (frequency range), and the first scale covering a broad frequency range at the high frequency end of the frequency spectrum, however, with progressively shorter bandwidths. Conversely, the first scale will have the highest time resolution; higher scales will cover increasingly longer time intervals. Actually the voltage signal/function is decomposed into two components, namely approximate $c(k)$ and detail $d(k)$, by passing through low pass and high pass filters, respectively. Further, $c(k)$ can be decomposed into second level and so on the

tree continues till the disturbance in the original signal is filtered out. Hence $c(k)$ contains the low frequency component of the signal while $d(k)$ contains the high frequency component. In this context, the disturbance signal (due to islanding or load rejection) can be decomposed in order to separate the low frequency and high frequency components which are extremely helpful in localization and detection of the disturbance instants.

4.1.2 Choice of mother wavelet

In wavelets applications, the choice of appropriate mother wavelet plays an important role in the analysis. Different basis functions have been proposed. These include Haar, Morlet, Mexican, Daubechies, etc. The choice of mother wavelet depends on the application. For example in signal processing in real time, computation efficiency may be of importance to be considered. For classification of power quality disturbance signals, the choice revolves of discrimination between various transients [30].

Daubechies wavelet family is one of the most suitable wavelet families in analyzing power system transients as investigated in [31], [33], and [30]. In the present work, the db4 wavelet (with four quadrature filter bank) shown in Figure 4.2, 4.3 and 4.4 has been used as the mother wavelet for analyzing the transients associated with islanding. Db4 is a short wavelet and therefore it can efficiently detect transients.

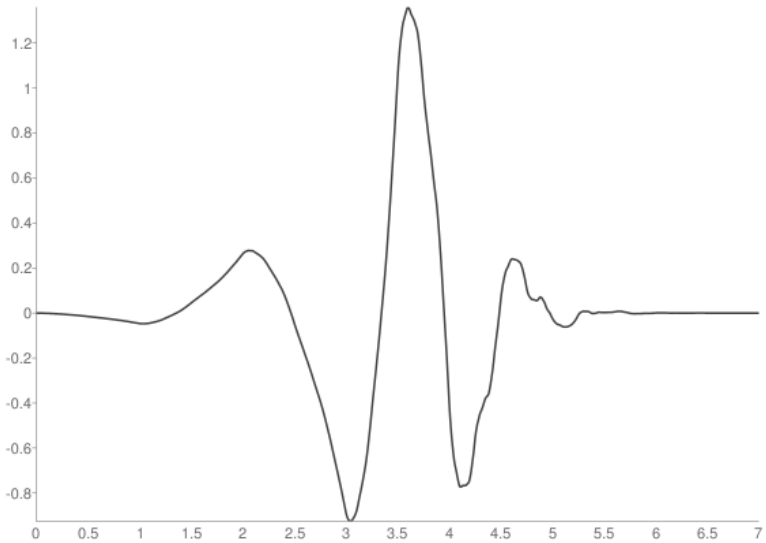

Figure 4.2: Wavelet Function of db4

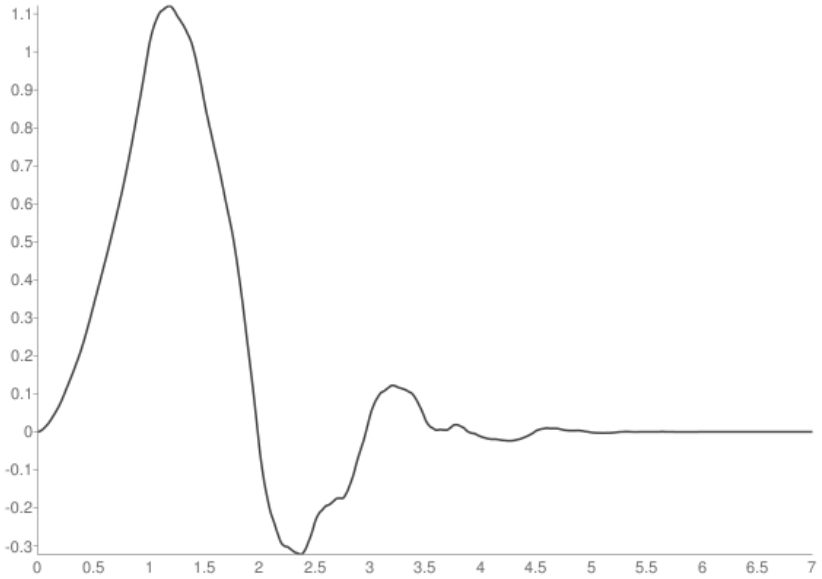

Figure 4.3: Scaling Function of db4

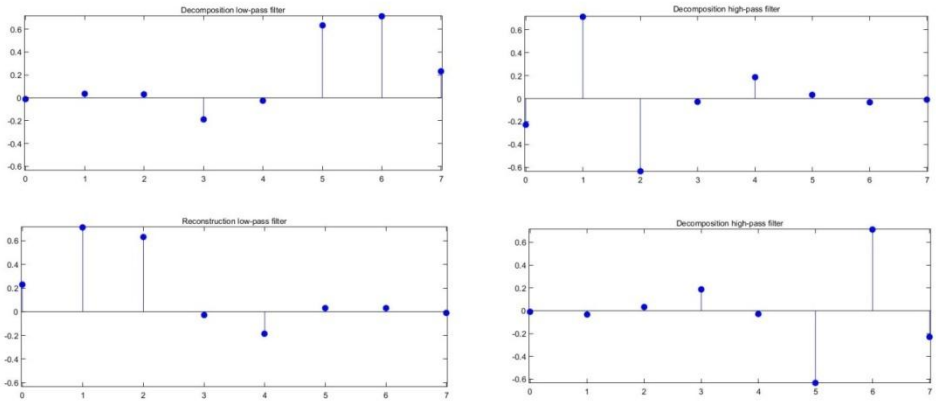

Figure 4.4: Quadrature Filter Bank of db4

4.2 Artificial Neural Networks

Artificial neural networks (ANNs) have been used in many potential applications in power systems operation and control. Load forecasting, fault diagnosis/fault location, economic dispatch, transient stability and harmonics analysis are some of the application in which ANN was adopted as a classifier [35]. ANNs are often used as classifiers since they have the capability of learning complex mapping, linear or nonlinear from the input space to the output space [35]. The architecture and the

training algorithm of the feed forward artificial neural network are described in the following sections.

4.2.1 Artificial Neural Network Architecture

ANN consists of simple processing units, called neurons, operating in parallel to solve specific problems. Figure 3.5 shows a simple neuron with input vector P of dimension $R \times 1$. The input P is multiplied by a weight W of dimension $1 \times R$. Then a bias b is added to the product WP . f is a transfer function (called also the activation function) that takes the argument n and produces the net output a . Both the weight and the bias are adjustable parameters of the neurons.

The idea of ANN is that these parameters (W and P) are adjusted so that the network exhibits some desired behavior. Thus the network can be trained to do a particular job by adjusting the weight or bias parameters [35].

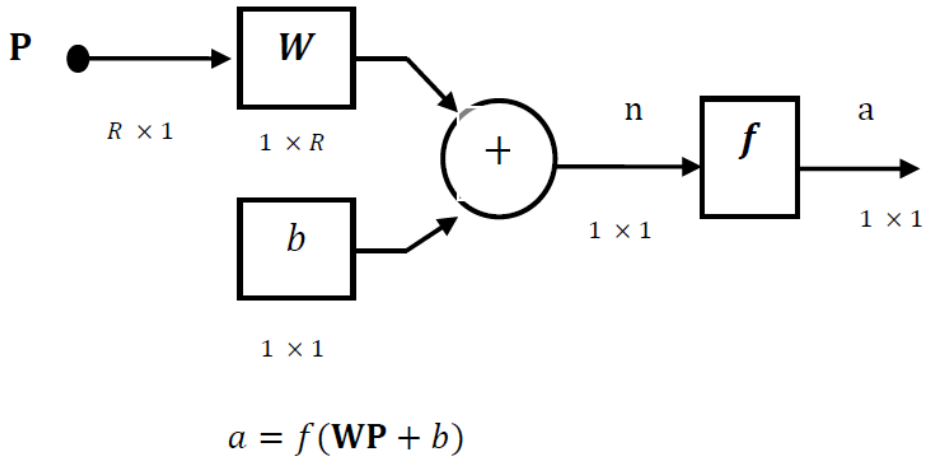

Figure 4.5: Neuron

4.2.2 Training the ANN

A training algorithm is defined as a procedure of updating the weights and biases of a network so the network will be able to perform the particular design task. The training algorithm is divided into two main categorizes: supervised learning, and unsupervised learning. ANN is classified under supervised learning. In the training stage, the training data set and the corresponding targets are entered to the model. Once the network weights and biases are initialized, the network is ready for training. The

weights and biases are then adjusted in order to minimize the mean square error MSE (the average squared error between the networks outputs a and the target outputs t). This can be achieved using the gradient of the MSE. The gradient descent algorithm is determined using a technique called back propagation [35]. The steps of converging to the optimal solution can be summarized as follows:

- The network first uses the input vector to produce its output vector.
- The difference between the obtained target value and the actual target will be computed. This is referred to as the prediction error.
- The error will propagate backward through the network and the gradient (derivatives) of the change in error with respect to changes in weight values will be computed.
- The weights will be updated accordingly to reduce the error.

There are several issues involved when training multiple layers NN:

- Selecting the number of hidden layer :

  Network with one hidden layer is sufficient for solving most of the problems [35]. Though having more than 1 hidden layer for some application may result in faster learning. However networks with more than one hidden layer will increase the probability of converging to local minima.
- Deciding the number of Neurons in the hidden layers :

  If a small number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor. On the other hand, if too many neurons are used, the training time may become long and the network may over fit the data. When over fitting occurs the result is that the model fits the training data extremely well, but it generalizes poorly to novel data. The number of neurons chosen should correspond to the best performance on the validation data.

4.2.3 Back propagation Algorithm

Back propagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the

optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

Back propagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as auto encoders. It is a generalization of the delta rule to multi-layered feed forward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Back propagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable. The backpropagation learning algorithm can be divided into two phases: propagation and weight update.

- Phase 1: Propagation

Each propagation involves the following steps:

1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
2. Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

- Phase 2: Weight update

For each weight-synapse follow the following steps:

1. Multiply its output delta and input activation to get the gradient of the weight.
2. Subtract a ratio (percentage) from the gradient of the weight.

This ratio (percentage) influences the speed and quality of learning; it is called the *learning rate*. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing; this is why the weight must be updated in the opposite direction.

4.3 Methodology

The proposed method utilizes and combines wavelet analysis and artificial neural network. Wavelet transform is capable of decomposing the signals into different frequency bands. It can be utilized in extracting discriminative features from the

acquired negative sequence voltage signal. The features are then fed to a trained ANN model which if well trained is capable of detecting islanding and differentiating between islanding events and any other events that have transients such as switching or temporary fault. In the proposed technique, prescribed events are simulated. The negative sequence voltage signal at the PCC are acquired in order to be analyzed using WT. Negative sequence component is one of the key indicators which quantify the presence of any disturbances in the voltage signal retrieved at point of common coupling (PCC) . Thus, in this thesis, the negative sequence component of the voltage signals retrieved at PCC is considered for analysis towards effective detection of islanding and PQ events. The negative, positive and zero sequence component of the voltage signal at PCC can be expressed by symmetrical component analysis as:

$$V_n = \frac{1}{3}(V_a + \lambda^2 V_b + \lambda V_c) \tag{4.11}$$

$$V_p = \frac{1}{3}(V_a + \lambda V_b + \lambda^2 V_c) \tag{4.12}$$

$$V_z = \frac{1}{3}(V_a + V_b + V_c) \tag{4.13}$$

where V_a , V_b and V_c are the three-phase voltages retrieved at the PCC and V_n , V_p and V_z are the negative, positive and zero sequence voltages, respectively, and $\lambda = 1\angle 120°$ is the complex operator. The negative sequence component of the extracted voltage signal at PCC is obtained by passing it through the three-phase sequence analyzer block in MATLAB/Simulink. Out of these three sequential components, it is only negative sequence component of the voltage signal, considered in this study because it reflects the information under disturbance condition. Quantification of the negative-sequence voltage at PCC is carried out which provides high degree of immunity to noise, for detection of islanding event and other disturbances, thus enable better performance.

The discriminative features are then captured and fed to a trained ANN model. A general block diagram of the proposed method is illustrated in Figure 4.6.

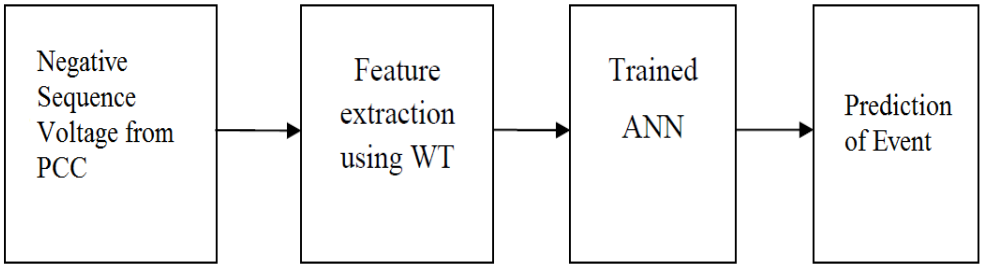

Fig. 4.6: General methodology block diagram.

The system under study is simulated in MATLAB/Simulink as discussed in chapter two. The system model contains two DGs. Two different sets of data which correspond to the two different classes (islanding and non-islanding) are simulated in Simulink. The simulated non-islanding cases include normal condition i.e. grid connected DG without any disturbances, temporary single line to ground (SLG), line to line to ground (LLG) and Non-Linear Load Switching at PCC. Negative Sequence Voltage signals for the different mentioned cases are then acquired from the PCC. WT will be carried out on the obtained negative sequence voltage signals to extract the features. The purpose of feature extraction is to identify specific signature of the negative sequence voltage waveforms that can detect islanding and differentiate between islanding and any other transient condition.

A transient signal can be fully decomposed into smoothed signals and detailed signals for L wavelet levels. In wavelets applications, Daubechies wavelet family is one of the most suitable wavelet families in analyzing power system transients as investigated in [31], [32], and [30]. In the present work, db4 wavelet has been used as the mother wavelet for extracting the energy content and SD of the detail coefficient of negative sequence voltage waveforms. Db4 is a short wavelet and therefore it can efficiently detect transients. The input signals were decomposed for 5 wavelet levels. Table 4.1 gives the frequency band information of the wavelet analysis. The sampling frequency is 1 kHz. Details (D3 and D4) were used for feature extraction since it has the maximum change in amplitude before and after disturbances. Fig. 4.7 shows the change in frequency levels before and after Islanding. Frequency range from 60 Hz to 250 Hz has strengthened significantly after Islanding. Frequency levels below fundamental frequency are ignored.

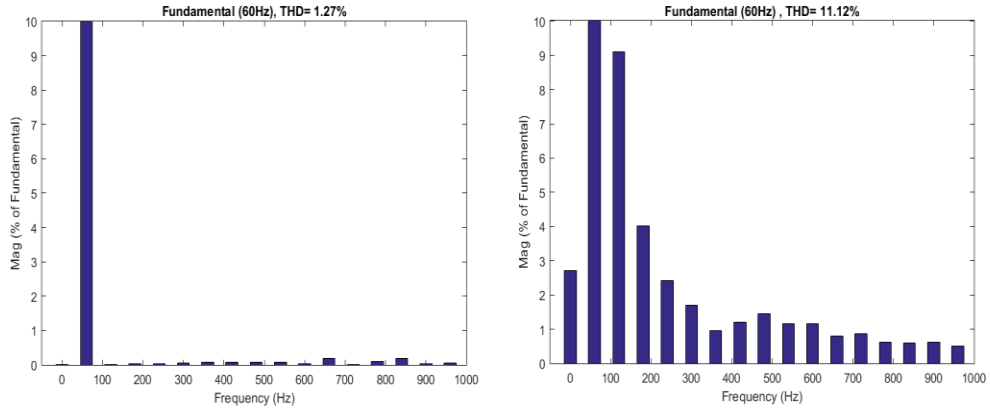

Fig.4.7: Comparison of Harmonic Distribution Before (Left) and After (Right) Islanding

Table 4.1: Frequency band information of different levels

Wavelet Level	Frequency Band (Hz)
D1	500 - 1000
D2	250 - 500
D3	125 - 250
D4	62.5 - 125
D5	31.25 - 62.5

The energy content and Standard Deviation in the details of each decomposition level for all negative sequence voltage signals was calculated using the detail coefficients in the corresponding level. Energy is represented by the L2 Norm of the signal. Equation (4.14) shows the calculation of the energy content (E3) and Equation (4.15) is the formula for calculating Standard Deviation (SD3) for the third detail. The energy content in the other decomposition level can be calculated the same way.

$$\|X\|_2 := \sqrt{\sum_{i=1}^{N} |x_i|^2} \tag{4.14}$$

$$SD = \frac{1}{N-1} \sum_{i-1}^{N} (x_i - \mu)^2 \tag{4.15}$$

Where x_i is amplitude of i^{th} sample of Wavelet Transform of signal at D3, N is the number of samples and μ is the mean of the Wavelet Transform of signal at D3. Loading is considered as a feature since effect of islanding on the system depends heavily on loading of the system. During power mismatch the Islanding is more detectable, but as the load reaches rated output of the system, Islanding detection becomes difficult as discussed in chapter two.

Table 4.2 shows the input feature vector for training the ANN for normal operating conditions i.e. without any disturbances. Since ANN requires supervised learning, the label has to be provided at training time. Table 4.2 shows a sample of input feature vector to train ANN.

Table 4.2: Labels for ANN Training

Event	Label for ANN Training
Normal Operation (No Disturbances)	1
Islanding	2
L-G Fault	3
L-L Fault	4
Non – Linear Load Switch	5

Dataset consist of total 130 examples in which, each of 5 cases of events has 26 examples. The examples is randomly distributed and divided into training and test cases in the ratio of 70(%): 30 (%). Complete data set is given in APPENDIX A and simulation MATLAB code for data collection is given in APPENDIX B. Then the total number of training examples is 91 and test case examples are 39. Once, trained with training examples, the algorithm is tested using test examples and compares the prediction result by ANN with original label to get the prediction accuracy. MATLAB code for ANN is given in APPENDIX C.

CHAPTER 5

SIMULATION RESULTS AND DATA COLLECTION

This section consists of the waveforms obtained on computer simulation of various events at MATLAB/Simulink and the data collected in those disturbances to train ANN. Simulation time is 1.2 seconds. Disturbances are introduced from 0.5 to 0.7 seconds.

5.1 Normal Operation

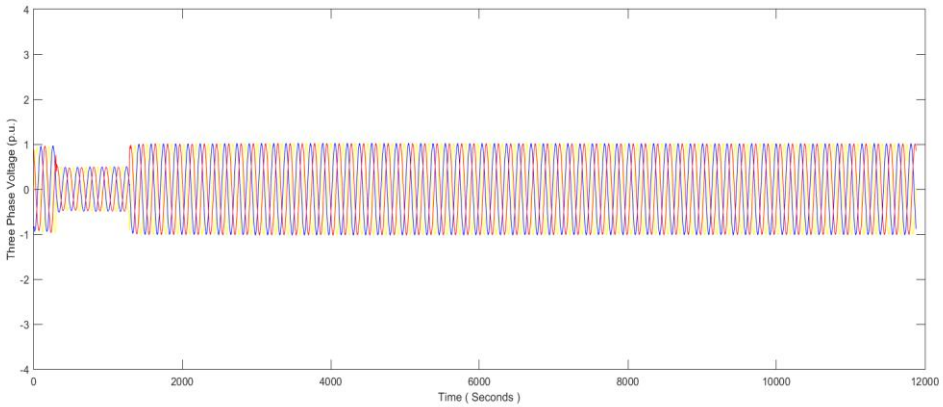

Fig.5.1: Three Phase Voltage at PCC during normal conditions

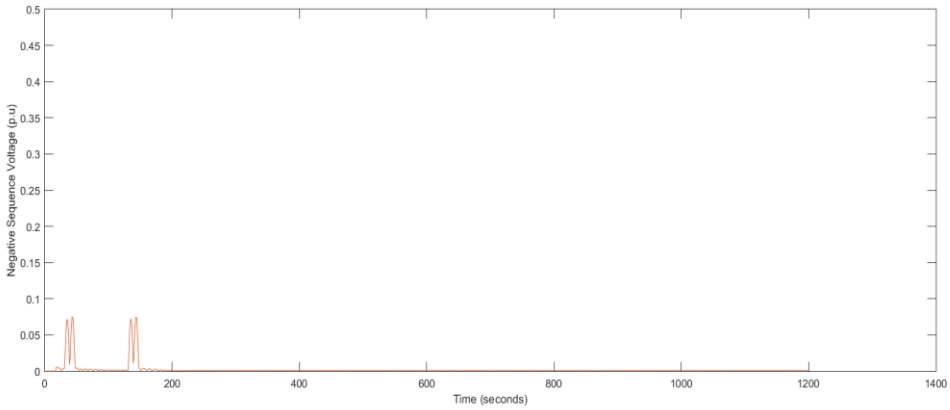

Fig.5.2: Negative Voltage at PCC at normal conditions

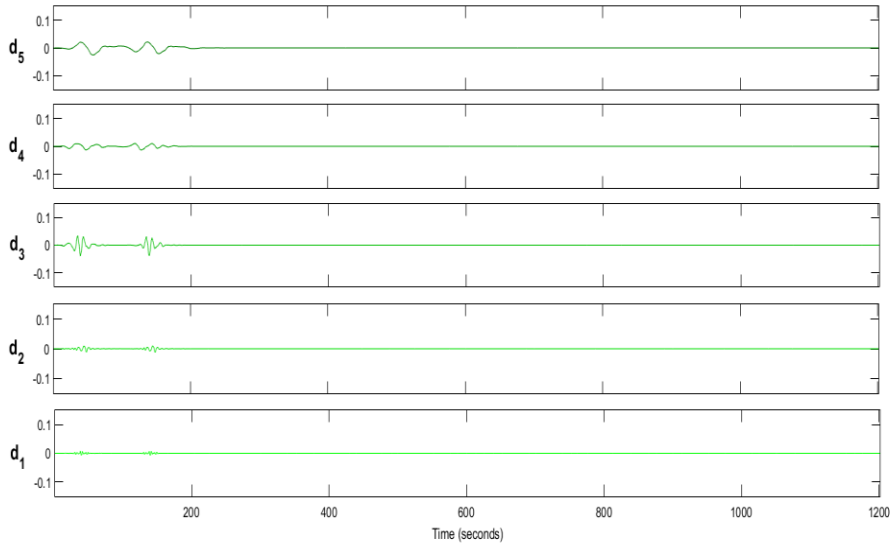

Fig.5.3: Details of Wavelet Transform at normal conditions

Table 5.1: Feature Vector for ANN Training of Normal Operation

Feature Vector					Label
Loading (MW)	SD3	SD4	E3	E4	
0.875	0.00011473	0.00014444	0.00130841	0.00119563	1
1.015	0.00011594	0.00015432	0.00132418	0.00128137	1
1.295	0.00010763	0.00017652	0.00123171	0.00146062	1
1.505	0.00011485	0.00019470	0.00131076	0.00160678	1
1.75	0.00012745	0.00022712	0.00145480	0.00187444	1

5.2 Islanding

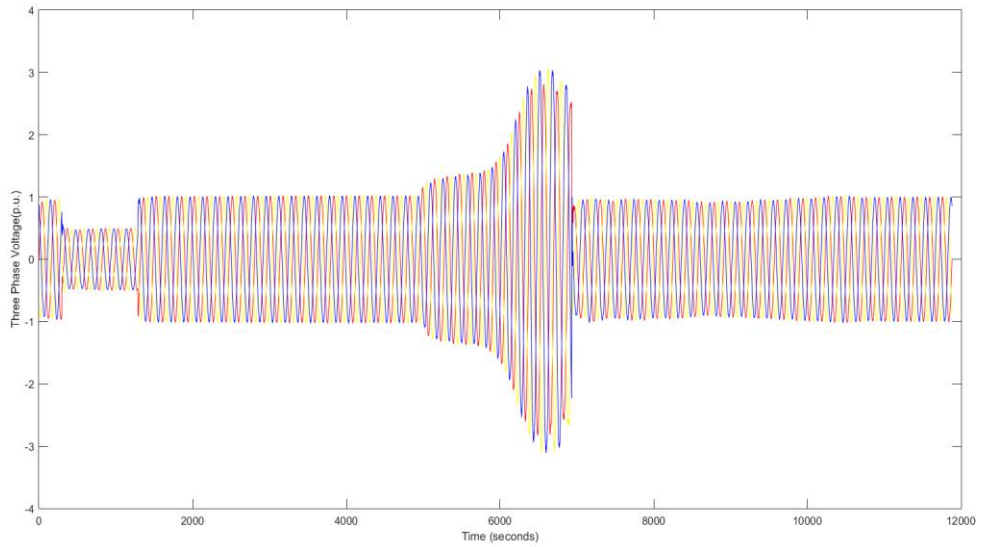

Fig.5.4: Three Phase Voltage at PCC at Islanding

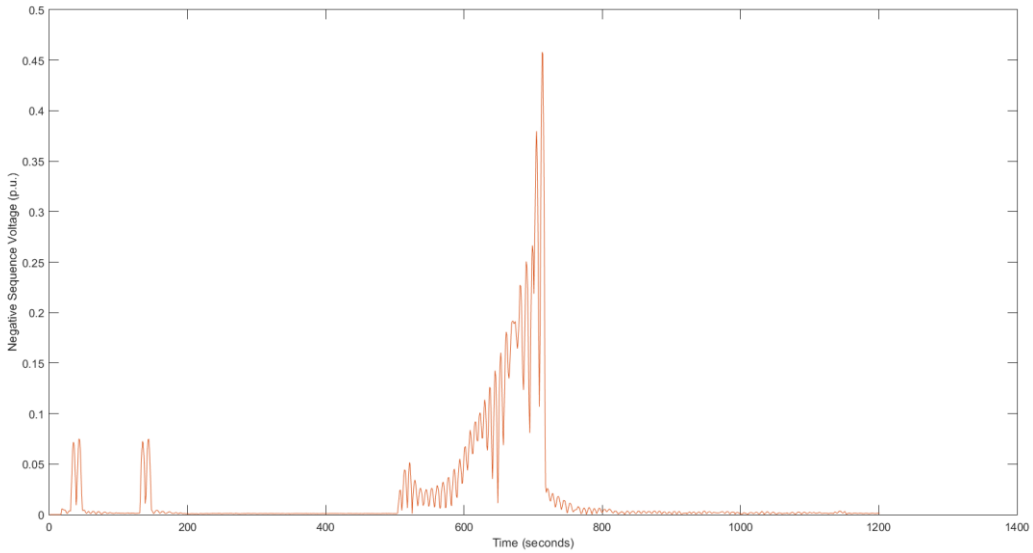

Fig.5.5: Negative Voltage at PCC at Islanding

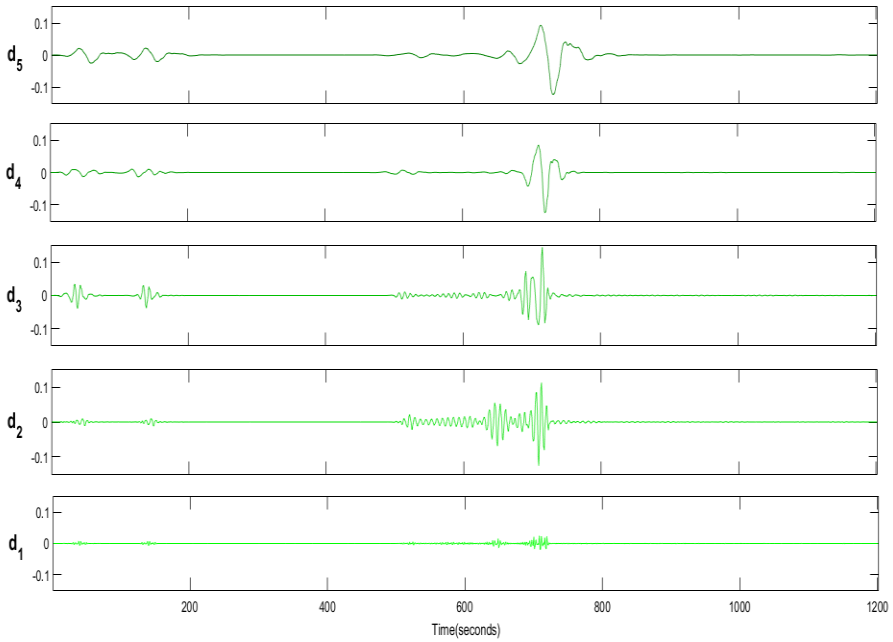

Fig.5.6: Details of Wavelet Transform at Islanding

Table 5.2: Feature Vector for ANN Training of Islanding

Feature Vector					Label
Loading (MW)	SD3	SD4	E3	E4	
0.875	0.03401753	0.03705120	0.39045041	0.30587180	2
1.015	0.03204665	0.04406543	0.36693973	0.36386052	2
1.295	0.02510713	0.02957596	0.29207070	0.24395503	2
1.505	0.00600586	0.00491463	0.06873916	0.04055274	2
1.75	0.00583276	0.00608325	0.06664993	0.05016677	2

5.3 L-G Fault

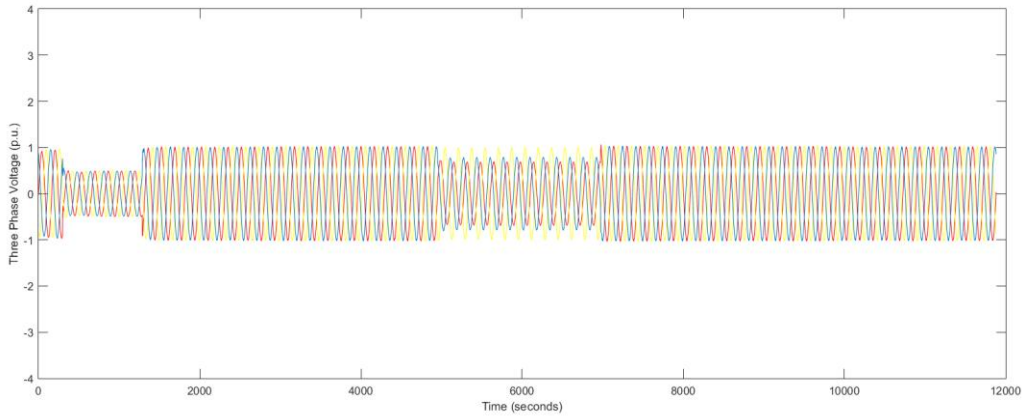

Fig.5.7: Three Phase Voltage at PCC at L-G Fault

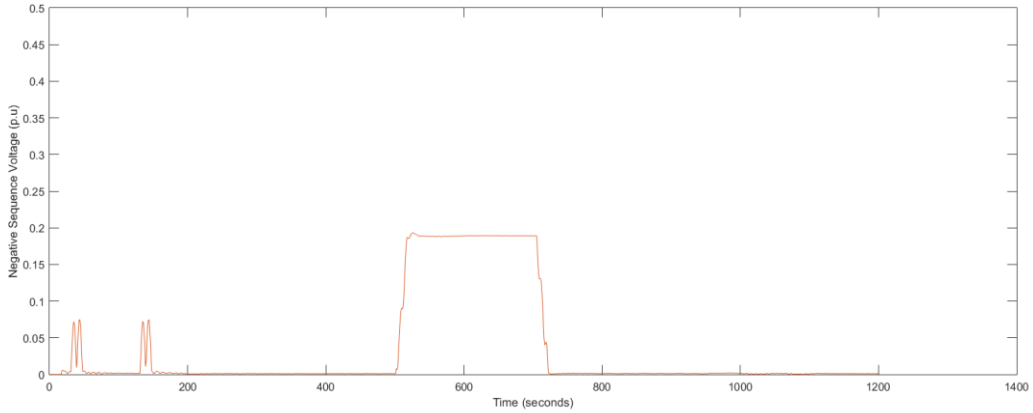

Fig.5.8: Negative Voltage at PCC at L-G fault

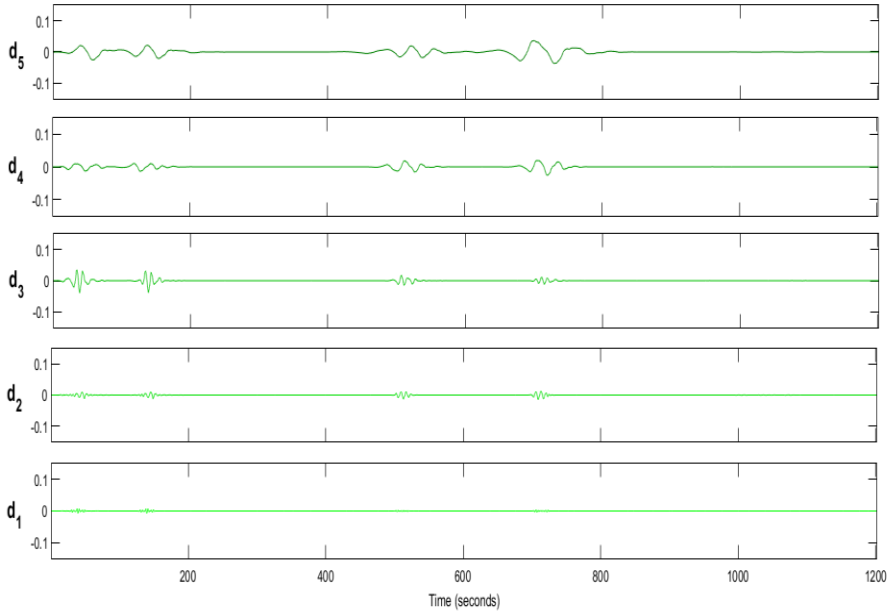

Fig.5.9: Details of Wavelet Transform at L-G fault

Table 5.3: Feature Vector for ANN Training of L-G Fault

Feature Vector					Label
Loading (MW)	SD3	SD4	E3	E4	
0.875	0.00494453	0.01195818	0.05639256	0.09861007	3
1.015	0.00494597	0.01196308	0.05641079	0.09865040	3
1.295	0.00495142	0.01204052	0.05647472	0.09928896	3
1.505	0.00496716	0.01207443	0.05665421	0.09956846	3
1.75	0.00495896	0.01213379	0.05656272	0.10005795	3

5.4 L-L Fault

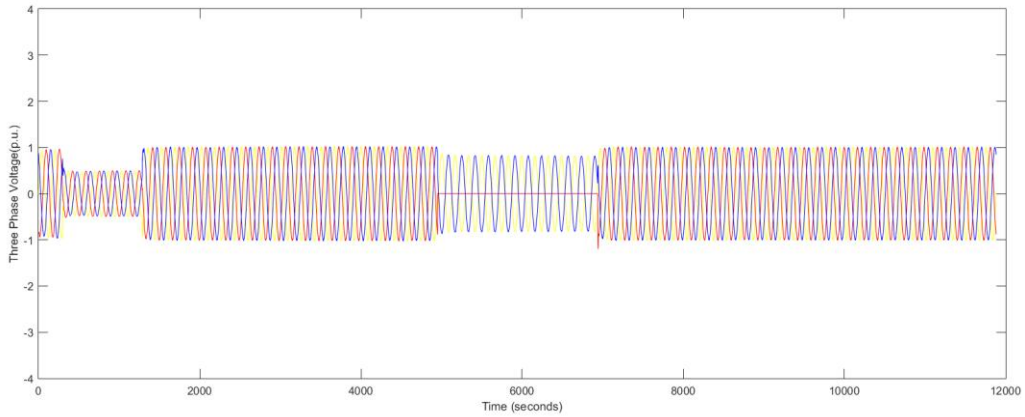

Fig.5.10: Three Phase Voltage at PCC at L-L Fault

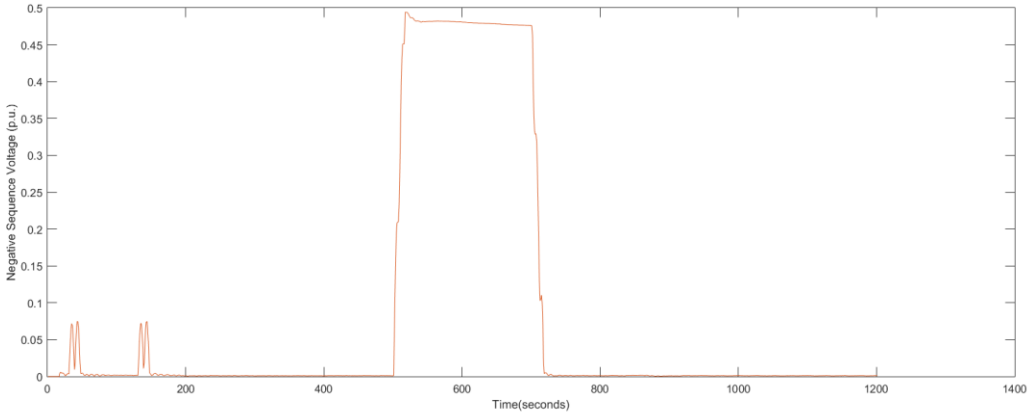

Fig.5.11: Negative Voltage at PCC at L-L fault

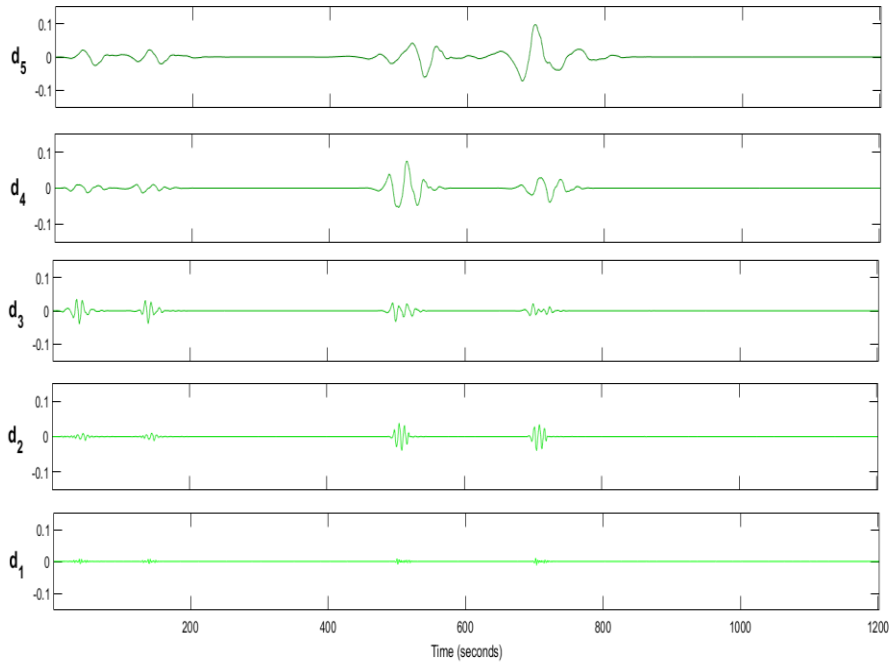

Fig.5.12: Details of Wavelet Transform at L-L fault

Table 5.4: Feature Vector for ANN Training of L-L Fault

Feature Vector					Label
Loading (MW)	SD3	SD4	E3	E4	
0.875	0.00911728	0.02928778	0.10401861	0.24154712	4
1.015	0.00911516	0.02921008	0.10399256	0.24090679	4
1.295	0.00911595	0.02910395	0.10399785	0.24003145	4
1.505	0.00911367	0.02904593	0.10397254	0.23955288	4
1.75	0.00912749	0.02892913	0.10413084	0.23858983	4

5.5 Non Linear Load Switch

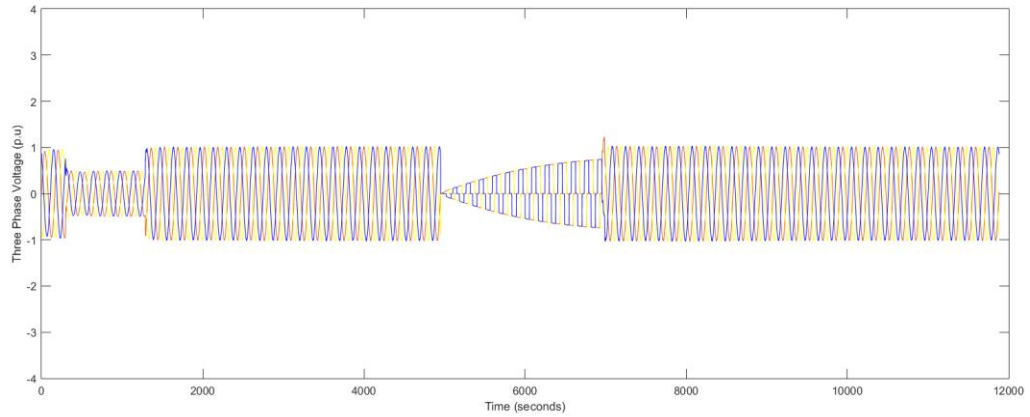

Fig.5.13: Three Phase Voltage at PCC during Non- Linear Load Switch

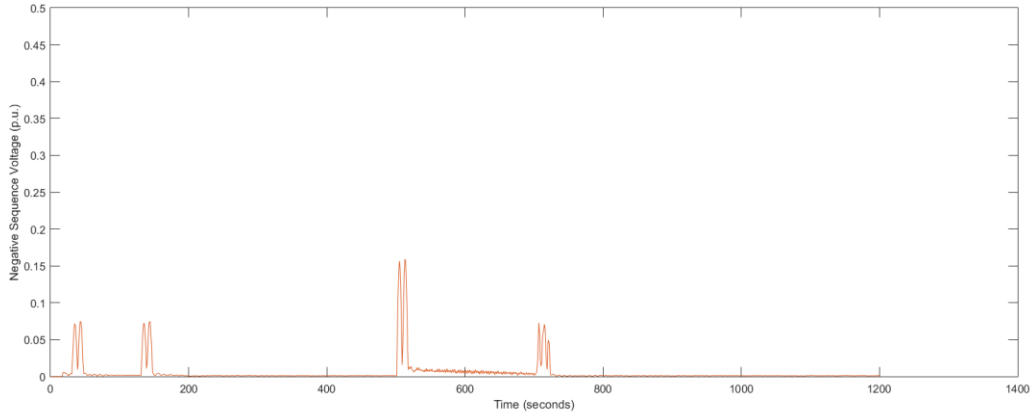

Fig.5.14: Negative Sequence Voltage at PCC during Non- Linear Load Switch

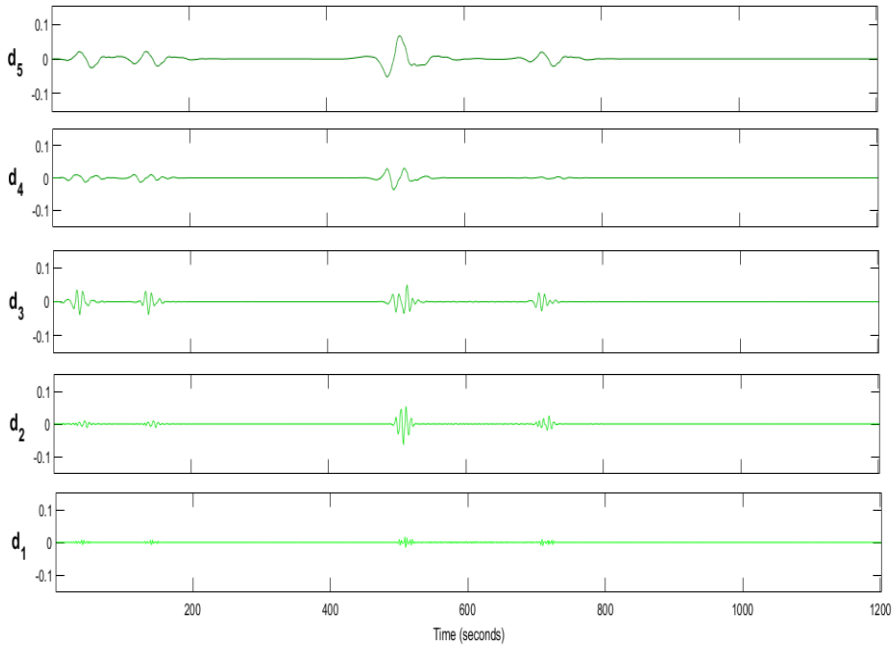

Fig.5.15: Details of Wavelet Transform during Non Linear Load Switch

Table 5.5: Feature Vector for ANN Training of Non-Linear Load Switch

Feature Vector					Label
Loading (MW)	SD3	SD4	E3	E4	
0.875	0.01280362	0.02509573	0.14671166	0.20694654	5
1.015	0.01280433	0.02509539	0.14673016	0.20694387	5
1.295	0.01279189	0.02506549	0.14659020	0.20669716	5
1.505	0.01278282	0.02503957	0.14650191	0.20648373	5
1.75	0.01277151	0.02500862	0.14638457	0.20622875	5

5.6 Results

It is evident from Figures 5.1-5.15 that some differences exist in the negative sequence during different types of events can be noticed by the naked eye. These differences reflect on WT indices and ANN will be able to classify the events with high prediction accuracy.

In order to generalize the ability of the ANN classifier, cross- validation technique is used. In cross-validation, the available simulated data are divided into two sets, first is the training set for which ANN is trained and the parameters are set. The remaining examples are test examples for which the algorithm is tested with new examples. The prediction of the trained ANN is then compared with original label of the test example to find the accuracy. Following table 3.8 shows the results of ANN prediction on test set.

Table 5.6: ANN Prediction Results

No. of Test examples	39
No. of False Positive	1
No. of False Negative	1
Total Accuracy	$\frac{37}{39} \times 100 = 94.87\%$

Where a false positive error, or in short false positive, commonly called a "false alarm", is a result that indicates a given condition has been fulfilled, when it actually has not been fulfilled. I.e. erroneously a positive effect has been assumed. A false negative error, or in short false negative, is where a test result indicates that a condition failed, while it actually was successful. I.e. erroneously no effect has been assumed.

CHAPTER 6

IMPLEMENTATION

6.1 Microsoft Azure Machine Learning Studio

Microsoft Azure Machine Learning (ML) Studio is a collaborative, drag-and-drop tool that can be used to build, test, and deploy predictive analytics solutions on data. Machine Learning Studio publishes models as web services that can easily be consumed by custom apps or BI tools such as Excel. Azure Machine Learning Studio gives an interactive, visual workspace to easily build, test, and iterate on a predictive analysis model. Drag-and-drop datasets and analysis modules are onto an interactive canvas, connecting them together to form an experiment, which can be run in Machine Learning Studio. Then the training experiment can be converted to a predictive experiment, and then publish it as a web service so that the model can be accessed using an API (Application Programmable Interfacing) key. Fig. 6.1 shows the overall representation of Azure ML Studio.

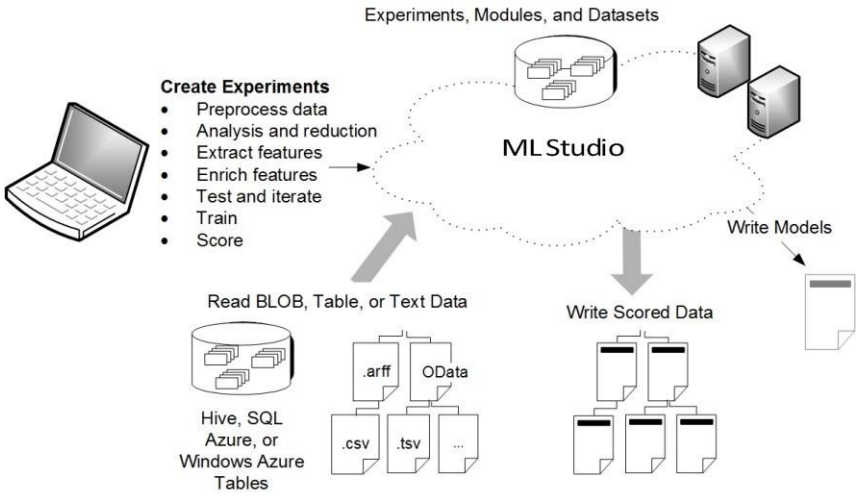

Fig. 6.1: Overview of Azure ML Studio

When deployed as a web service, Azure Machine Learning experiments provide a REST API that can be consumed by a wide range of devices and platforms. This is because the simple REST API accepts and responds with JSON formatted messages. The Azure Machine Learning portal provides code that can be used to call the web

service in R, C#, and Python. But these services can be called with any programming language and from any device that satisfies three criteria:

- Has a network connection
- Has SSL capabilities to perform HTTPS requests
- Has the ability to parse JSON (by hand or support libraries)

This means the services can be consumed from web applications, mobile applications, and custom desktop applications and even from within Excel.

An Azure Machine Learning web service can be consumed in two different ways, either as a request-response service or as a batch execution service. In each scenario the functionality is provided through the RESTFul web service that is made available for consumption once the experiment has been deployed. Deploying a Machine Learning web service in Azure with an Azure web service end-point, where the service is automatically scaled based on usage, you can avoid upfront and ongoing costs for hardware resources.

6.1.1 Request-Response Service (RRS)

A Request-Response Service (RRS) is a low-latency, highly scalable web service used to provide an interface to the stateless models that have been created and deployed from an Azure Machine Learning Studio experiment. It enables scenarios where the consuming application expects a response in real-time.

RRS accepts a single row, or multiple rows, of input parameters and can generate a single row, or multiple rows, as output. The output row(s) can contain multiple columns.

6.1.2 Batch Execution Service (BES)

A Batch Execution Service (BES) is a service that handles high volume, asynchronous, scoring of a batch of data records. The input for the BES contains a batch of records from a variety of sources, such as blobs, tables in Azure, SQL Azure, HDInsight (results of a Hive Query, for example), and HTTP sources. The output for the BES contains the results of the scoring. Results are output to a file in Azure blob storage and data from the storage endpoint is returned in the response. A BES would

be useful when responses are not needed immediately, such as for regularly scheduled scoring for individuals or internet of things (IOT) devices.

Fig: 6.2 show the data diagram drawn in Azure ML Studio to train the model. The Untitled.csv is the input data which is given in APPENDIX A is uploaded to the cloud. Using split data module, the data is split into training and test data in the ratio of 70%: 30%. The training data is fed to Train Model and the algorithm used is Multi Class Neural Network. Once trained, using Score Model, predicts output of the test dataset. The Evaluate model gives the results of the experiment by comparing original labels with the prediction of the trained ANN..

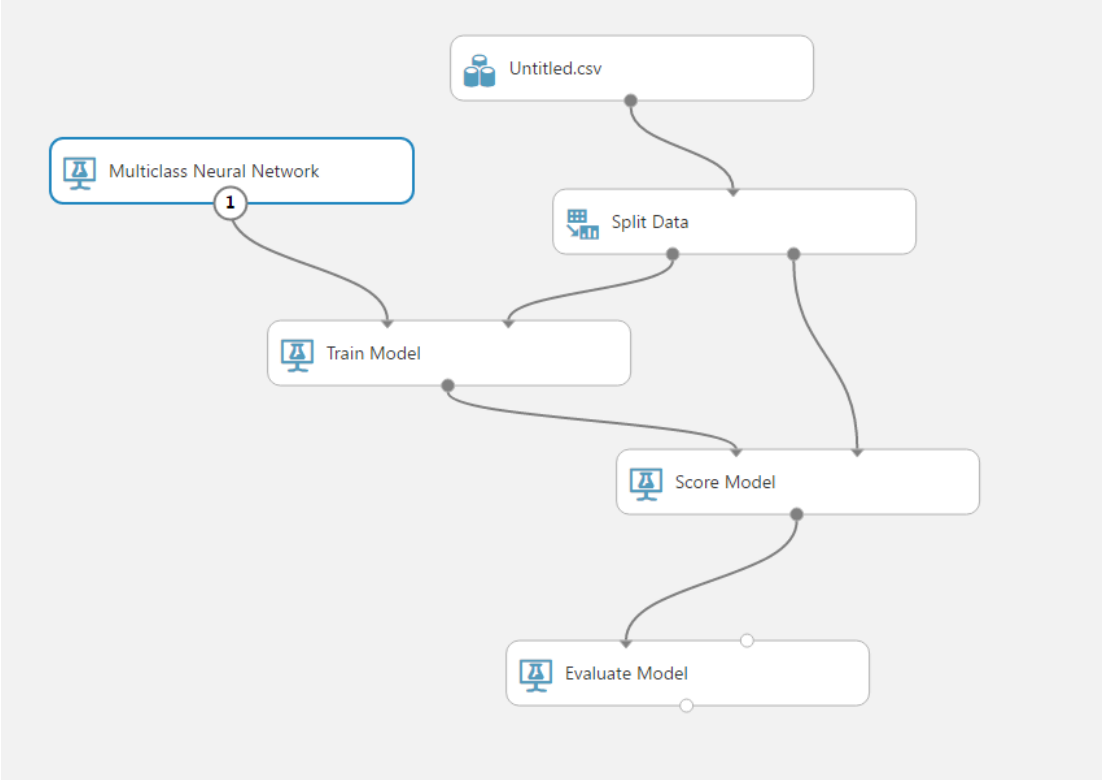

Fig.6.2: Data diagram for training

Once trained, the model can be published as Web Service so that it can be accessed from multiple devices. Fig. 6.3 shows the data flow diagram of publishing the model as a predictive experiment. When web service input is given, the trained model process that input and gives prediction as web service output.

Using Python TkInter library, a GUI is built for requesting the Web Service to access the prediction results from the trained model in Azure ML Studio. The code is

provided in APPENDIX D. Snapshot of GUI is given in Fig. 6.4 (a) and (b). Once the parameters are given, pressing the 'Show' button will show the current status of the system.

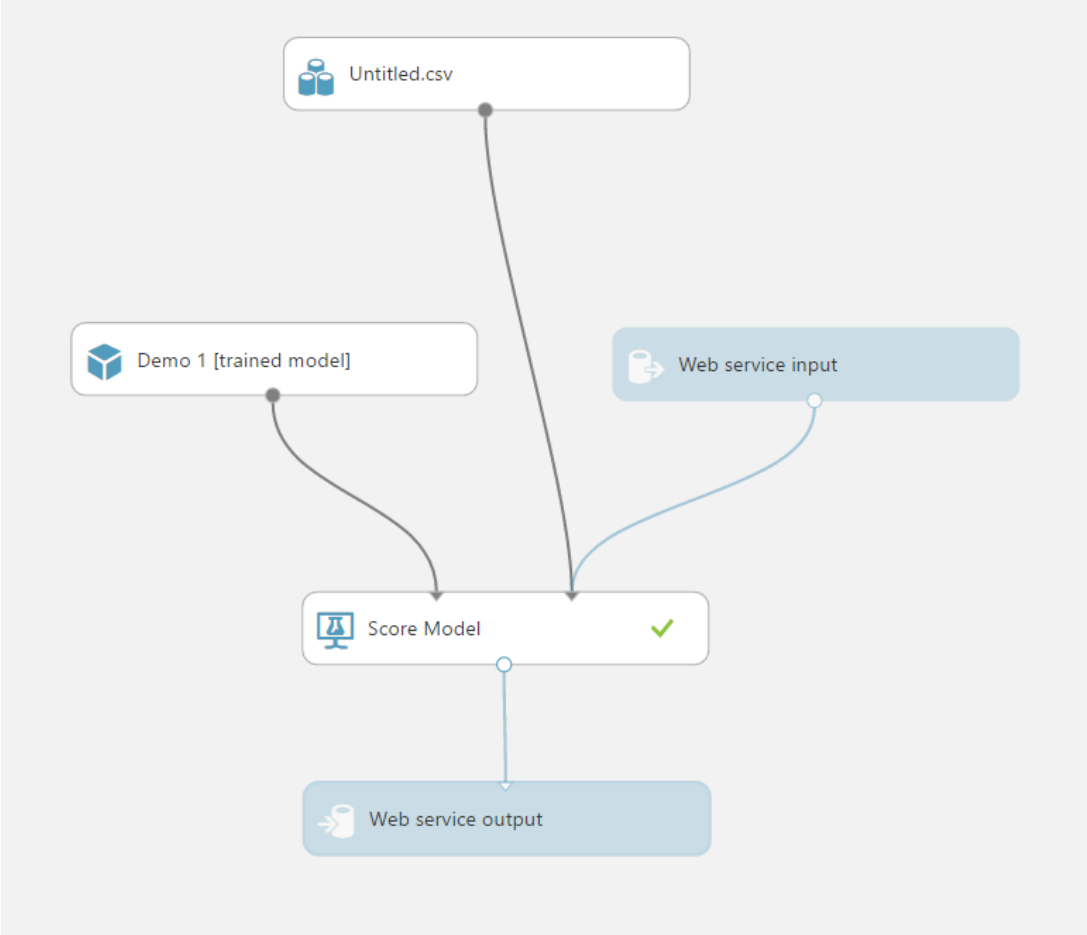

Fig. 6.3: Deploying trained model as a Web Service

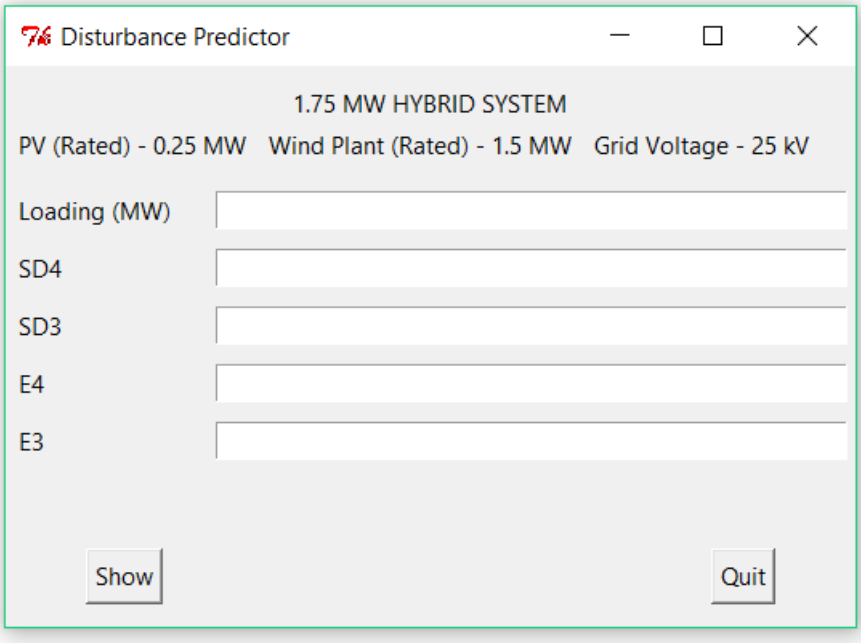

Fig. 6.4(a): Python GUI

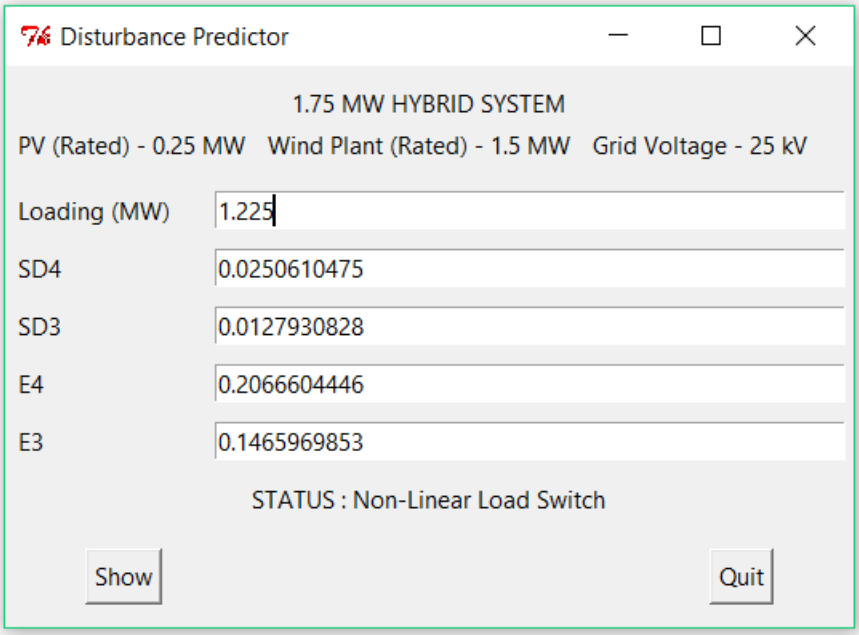

Fig. 6.4(b): Python GUI showing status of system

CHAPTER 7

CONCLUSION

Many schemes have been proposed to detect islanding such as passive, active and communication based techniques. Passive techniques work well when there is power imbalance between the power generated from the DG and the power consumed form the load. On the other hand, active methods affect the power quality and do not perform well in the presence of multiple DGs. Though communication based islanding detection techniques have no NDZ techniques, they are costly and complex.

The performance of a commonly used passive anti-islanding technique was validated by computer simulation. The technique studied in this thesis is detection of harmonics (DH) passive anti-islanding scheme. In the DH method it is difficult to select an appropriate threshold that provide islanding detection but do not lead to false tripping and fails to classify different disturbances. This technique is not capable of detecting islanding alone and should be combined with other techniques in the distribution network.

WT-ANN based islanding technique was successfully implemented for hybrid distribution system. Wavelet transform is capable of decomposing the voltage signals into different frequency bands. It can be utilized in extracting discriminative features from the acquired negative sequence voltage signals. The energy content and standard deviation of wavelet details are then calculated and fed to a trained ANN which is capable to differentiating between islanding and non-islanding events. The high accuracy of ANN validates the quality of WT indices taken.

Implementing WT-ANN based anti-islanding technique using Microsoft Azure Machine Learning Studio brings feasibility for application of Machine Learning in power industry in an easy and user friendly manner. Deploying the trained model as a Web Service will help to access the model from multiple devices and from multiple places at same time. Continuous monitoring of the system is effectively possible with greater accuracy. The implementation cost is less and can be considered as a robust prospect to power system control and detection in future.

7.1 Future work

Though the proposed method resulted in very good accuracy rate, the technique has been validated using relatively small testing set. Increasing the testing data set will result in more general and accurate recognition rate. The proposed technique was implemented on one system configuration. To increase confidence in these results, similar simulations could be performed using different system models. Another direction for future work is to implement and verify the proposed approach with practical and real time implementation. The poor performance of WT in voltage swell events also needs further study.

REFERENCES

[1] El-Saadany, E. F., H. H. Zeineldin, and A. H. Al-Badi. "Distributed generation: benefits and challenges." *International Conference on Communication, Computer & Power*. Vol. 115119. 2007.

[2] Algarni, Ayed. "Operational and planning aspects of distribution systems in deregulated electricity markets." (2009).

[3] Zeineldin, H. H., and M. M. A. Salama. "Islanding detection of inverter-based distributed generation." *IEE Proceedings-Generation, Transmission and Distribution* 153.6 (2006): 644-652..

[4] Eltawil, Mohamed A., and Zhengming Zhao. "Grid-connected photovoltaic power systems: Technical and potential problems—A review." *Renewable and Sustainable Energy Reviews* 14.1 (2010): 112-129.

[5] United States of America. Congress of the U.S., Congressional Budget Office. Prospects for distributed electricity generation. September, 2003.

[6] Tailor, J. K., and A. H. Osman. "Restoration of fuse-recloser coordination in distribution system with high DG penetration." *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*. IEEE, 2008.

[7] Xu, Wilsun, Konrad Mauch, and Sylvain Martel. "An assessment of distributed generation islanding detection methods and issues for Canada."*CANMET Energy Technology Centre-Varennes, Natural Resources Canada, QC-Canada, Tech. Rep. CETC-Varennes* 74 (2004).

[8] Menon, Vivek, and M. Hashem Nehrir. "A hybrid islanding detection technique using voltage unbalance and frequency set point." *IEEE Transactions on Power Systems* 22.1 (2007): 442-448.

[9] PVPS, IEA. "Evaluation of islanding detection methods for photovoltaic utility-interactive power systems." *Report IEA PVPS T5-09* (2002).

[10] Ropp, M. E., et al. "Using power line carrier communications to prevent islanding [of PV power systems]." *Photovoltaic Specialists Conference, 2000. Conference Record of the Twenty-Eighth IEEE*. IEEE, 2000.

[11] Xu, Wilsun, et al. "A power line signaling based technique for anti-islanding protection of distributed generators—part i: scheme and analysis." *IEEE Transactions on Power Delivery* 22.3 (2007): 1758-1766.

[12 Xu, Wilsun, et al. "A power line signaling based technique for anti-islanding protection of distributed generators—part i: scheme and analysis." *IEEE Transactions on Power Delivery* 22.3 (2007): 1758-1766.

[13] Redfern, M. A., O. Usta, and G. Fielding. "Protection against loss of utility grid supply for a dispersed storage and generation unit." *IEEE Transactions on Power Delivery* 8.3 (1993): 948-954.

[15] Wrinch, Michael C. *Negative sequence impedance measurement for distributed generator islanding detection*. Diss. University of British Columbia, 2008.

[16] Yin, Jun, Liuchen Chang, and Chris Diduch. "Recent developments in islanding detection for distributed power generation." *Power Engineering, 2004. LESCOPE-04. 2004 Large Engineering systems Conference on*. IEEE, 2004.

[17] Smith, G. A., P. A. Onions, and D. G. Infield. "Predicting islanding operation of grid connected PV inverters." *IEE Proceedings-Electric Power Applications* 147.1 (2000): 1-6.

[18] Ropp, M. E., Miroslav Begovic, and A. Rohatgi. "Analysis and performance assessment of the active frequency drift method of islanding prevention."*IEEE Transactions on Energy Conversion* 14.3 (1999): 810-816.

[19] Hung, Guo-Kiang, Chih-Chang Chang, and Chern-Lin Chen. "Automatic phase-shift method for islanding detection of grid-connected photovoltaic inverters." *IEEE Transactions on energy conversion* 18.1 (2003): 169-173.

[20] Hernandez-Gonzalez, Guillermo, and Reza Iravani. "Current injection for active islanding detection of electronically-interfaced distributed resources."*IEEE Transactions on power delivery* 21.3 (2006): 1698-1705.

[21] Kim, Byeong-Heon, Seung-Ki Sul, and Chun-Ho Lim. "Anti-islanding detection method using Negative Sequence Voltage." *Power Electronics and Motion Control Conference (IPEMC), 2012 7th International*. Vol. 1. IEEE, 2012.

[22] Rani, B. Indu, et al. "An active islanding detection technique for current controlled inverter." *Renewable energy* 51 (2013): 189-196.

[23] Mahat, Pukar, Zhe Chen, and Birgitte Bak-Jensen. "Review of islanding detection methods for distributed generation." *Electric Utility Deregulation and Restructuring and Power Technologies, 2008. DRPT 2008. Third International Conference on*. IEEE, 2008.

[24] Funabashi, Toshihisa, Kaoruy Koyanagi, and R. Yokoyama. "A review of islanding detection methods for distributed resources." *Power Tech Conference Proceedings*. Vol. 2. No. 1. 2003.

[25] Warin, J., and W. H. Allen. "Loss of mains protection." *Proc. 1990 ERA Conference on Circuit Protection for industrial and Commercial Installation, London, UK*. Vol. 4. 1990.

[26] Redfern, M. A., O. Usta, and G. Fielding. "Protection against loss of utility grid supply for a dispersed storage and generation unit." *IEEE Transactions on Power Delivery* 8.3 (1993): 948-954.

[27] Hopewell, P. D., N. Jenkins, and A. D. Cross. "Loss-of-mains detection for small generators." *IEE Proceedings-Electric Power Applications* 143.3 (1996): 225-230.

[28] Ibrahim, Oladimeji, et al. "Matlab/Simulink model of solar PV array with perturb and observe MPPT for maximising PV array efficiency." *2015 IEEE Conference on Energy Conversion (CENCON)*. IEEE, 2015..

[29] https://en.wikipedia.org/wiki/Wind_power

[30] Chen, S. "Feature selection for identification and classification of power quality disturbances." *IEEE Power Engineering Society General Meeting, 2005*. IEEE, 2005.

[31] Huang, Shyh-Jier, and Cheng-Tao Hsieh. "Coiflet wavelet transform applied to inspect power system disturbance-generated signals." *IEEE Transactions on Aerospace and Electronic Systems* 38.1 (2002): 204-210.

[32] Hsieh, Cheng-Tao, Jeu-Min Lin, and Shyh-Jier Huang. "Enhancement of islanding-detection of distributed generation systems via wavelet transform-based approaches." *International Journal of Electrical Power & Energy Systems* 30.10 (2008): 575-580.

[33] Lidula, N. W. A., N. Perera, and A. D. Rajapakse. "Investigation of a fast islanding detection methodology using transient signals." *2009 IEEE Power & Energy Society General Meeting*. IEEE, 2009.

[34] Kunte, R. "A wavelet transform-based islanding detection algorithm for inverter assisted distributed generators." *Ms. c. thesis, Tennessee Technological University* (2009).

[35] Haque, M. Tarafdar, and A. Kashtiban. "Application of neural networks in power systems; a review." *Trans. Eng. Comput. Technol* 6 (2000).

APPENDIX A

- Complete Dataset for Training and Testing ANN

Loading	SD4	SD3	E4	E3	Label
0.91	0.00016	0.000123	0.001321	0.001398	1
0.945	0.000166	0.00011	0.001375	0.001251	1
0.98	0.000144	0.000112	0.001194	0.001275	1
1.05	0.000148	0.000113	0.001233	0.001295	1
1.085	0.000162	0.000113	0.001339	0.001289	1
1.12	0.000143	0.000112	0.001184	0.001277	1
1.155	0.000165	0.000105	0.001364	0.001199	1
1.19	0.000146	0.000107	0.00121	0.001222	1
1.225	0.00016	0.000107	0.001324	0.001222	1
1.26	0.000172	0.00012	0.00143	0.001374	1
1.33	0.000175	0.000107	0.001444	0.001222	1
1.365	0.000184	0.000108	0.001523	0.001237	1
1.4	0.000202	0.000111	0.001672	0.00127	1
1.435	0.000196	0.000113	0.001615	0.001291	1
1.47	0.000191	0.000113	0.001578	0.001292	1
1.54	0.000188	0.000119	0.001551	0.001356	1
1.575	0.000187	0.000115	0.001546	0.001319	1
1.61	0.000166	0.000122	0.001374	0.001397	1
1.645	0.000188	0.000118	0.001556	0.001344	1
1.68	0.000184	0.000124	0.00152	0.001417	1
1.715	0.000218	0.000121	0.001798	0.001381	1
0.875	0.000144	0.000115	0.001196	0.001308	1
1.015	0.000154	0.000116	0.001281	0.001324	1
1.295	0.000177	0.000108	0.001461	0.001232	1
1.505	0.000195	0.000115	0.001607	0.001311	1

1.75	0.000227	0.000127	0.001874	0.001455	1
0.91	0.03949	0.031775	0.325947	0.365174	2
0.945	0.037571	0.031136	0.310085	0.357678	2
0.98	0.041851	0.03058	0.345552	0.351177	2
1.05	0.046179	0.035238	0.381349	0.403484	2
1.085	0.045764	0.038939	0.377916	0.44633	2
1.12	0.046168	0.041096	0.38108	0.471473	2
1.155	0.048216	0.041723	0.397756	0.478825	2
1.19	0.050753	0.041498	0.418612	0.476796	2
1.225	0.048868	0.038294	0.403062	0.441715	2
1.26	0.043728	0.033	0.36066	0.382166	2
1.33	0.017187	0.014445	0.141731	0.167945	2
1.365	0.013032	0.011371	0.107465	0.131135	2
1.4	0.010913	0.009909	0.089998	0.113876	2
1.435	0.008862	0.00865	0.073088	0.099294	2
1.47	0.006951	0.007265	0.057338	0.083199	2
1.54	0.003336	0.004867	0.027572	0.055604	2
1.575	0.001365	0.00363	0.011347	0.041409	2
1.61	0.000965	0.003169	0.007956	0.03619	2
1.645	0.001291	0.00329	0.01065	0.037512	2
1.68	0.002788	0.003802	0.022992	0.043357	2
1.715	0.00454	0.004716	0.03744	0.053834	2
0.875	0.037051	0.034018	0.305872	0.39045	2
1.015	0.044065	0.032047	0.363861	0.36694	2
1.295	0.029576	0.025107	0.243955	0.292071	2
1.505	0.004915	0.006006	0.040553	0.068739	2
1.75	0.006083	0.005833	0.050167	0.06665	2
0.91	0.011938	0.004949	0.098441	0.05644	3
0.945	0.011977	0.004945	0.098764	0.056402	3
0.98	0.011971	0.004943	0.098714	0.056379	3
1.05	0.011979	0.004942	0.098778	0.05637	3

1.085	0.011999	0.004949	0.098944	0.056448	3
1.12	0.012007	0.004951	0.099009	0.056465	3
1.155	0.012006	0.00495	0.099006	0.056459	3
1.19	0.011998	0.004948	0.098936	0.056433	3
1.225	0.012009	0.004956	0.099033	0.056521	3
1.26	0.012015	0.004946	0.099082	0.056416	3
1.33	0.012041	0.004958	0.099295	0.056547	3
1.365	0.012038	0.004954	0.099269	0.056501	3
1.4	0.012039	0.004956	0.099276	0.056529	3
1.435	0.012075	0.004954	0.099576	0.056505	3
1.47	0.012048	0.004956	0.099352	0.056524	3
1.54	0.012061	0.004958	0.099459	0.056553	3
1.575	0.012096	0.004957	0.099749	0.056542	3
1.61	0.012109	0.004958	0.09985	0.056554	3
1.645	0.012104	0.004964	0.099808	0.056616	3
1.68	0.012123	0.004951	0.099968	0.056469	3
1.715	0.012148	0.004964	0.100173	0.056615	3
0.875	0.011958	0.004945	0.09861	0.056393	3
1.015	0.011963	0.004946	0.09865	0.056411	3
1.295	0.012041	0.004951	0.099289	0.056475	3
1.505	0.012074	0.004967	0.099568	0.056654	3
1.75	0.012134	0.004959	0.100058	0.056563	3
0.91	0.029273	0.009117	0.241425	0.10401	4
0.945	0.029236	0.009115	0.24112	0.103994	4
0.98	0.02923	0.009114	0.241074	0.103984	4
1.05	0.029178	0.009114	0.240646	0.103979	4
1.085	0.029165	0.009124	0.240539	0.104087	4
1.12	0.029149	0.009116	0.240401	0.104003	4
1.155	0.029152	0.009117	0.240427	0.104016	4
1.19	0.029145	0.009111	0.240369	0.103942	4
1.225	0.029117	0.009117	0.240136	0.104012	4

1.26	0.029088	0.009114	0.239902	0.103977	4
1.33	0.029099	0.009116	0.239988	0.103998	4
1.365	0.029071	0.009113	0.239756	0.10396	4
1.4	0.029066	0.009119	0.239717	0.104033	4
1.435	0.029068	0.009117	0.239732	0.104009	4
1.47	0.029062	0.009119	0.239688	0.104038	4
1.54	0.029016	0.009126	0.239307	0.104116	4
1.575	0.029019	0.009124	0.23933	0.104091	4
1.61	0.029006	0.00912	0.23922	0.104043	4
1.645	0.02899	0.009127	0.239093	0.104127	4
1.68	0.028962	0.009125	0.238861	0.10411	4
1.715	0.028943	0.009125	0.238704	0.1041	4
0.875	0.029288	0.009117	0.241547	0.104019	4
1.015	0.02921	0.009115	0.240907	0.103993	4
1.295	0.029104	0.009116	0.240031	0.103998	4
1.505	0.029046	0.009114	0.239553	0.103973	4
1.75	0.028929	0.009127	0.23859	0.104131	4
0.91	0.025104	0.012798	0.207016	0.146653	5
0.945	0.025093	0.012802	0.206921	0.146697	5
0.98	0.025102	0.012802	0.206997	0.146697	5
1.05	0.025095	0.012799	0.206942	0.146672	5
1.085	0.025097	0.012802	0.206954	0.146708	5
1.12	0.025077	0.0128	0.206794	0.14668	5
1.155	0.025087	0.012799	0.206871	0.146669	5
1.19	0.025064	0.012796	0.206687	0.146623	5
1.225	0.025061	0.012793	0.20666	0.146597	5
1.26	0.025068	0.012794	0.20672	0.146612	5
1.33	0.025063	0.012794	0.206677	0.146627	5
1.365	0.025047	0.01279	0.206549	0.146572	5
1.4	0.025067	0.012796	0.206706	0.146644	5
1.435	0.025036	0.012789	0.206452	0.146566	5

1.47	0.025042	0.012789	0.206506	0.146571	5
1.54	0.025054	0.012787	0.206599	0.146544	5
1.575	0.025036	0.01278	0.206457	0.14647	5
1.61	0.025012	0.012776	0.206257	0.14643	5
1.645	0.025029	0.012779	0.206396	0.146463	5
1.68	0.025018	0.012774	0.206306	0.146408	5
1.715	0.025018	0.01277	0.206305	0.146364	5
0.875	0.025096	0.012804	0.206947	0.146712	5
1.015	0.025095	0.012804	0.206944	0.14673	5
1.295	0.025065	0.012792	0.206697	0.14659	5
1.505	0.02504	0.012783	0.206484	0.146502	5
1.75	0.025009	0.012772	0.206229	0.146385	5

APPENDIX B

- MATLAB Code for collecting data from simulation

```
%%MATLAB CODE FOR ACQUIRING DATA USING LOOPED SIMULATION

clc;
activeP=.875e6; %Loading starts from 50 % loading
for j=1:26
    sim('project_model.mdl'); %%simulates the simulink
model
    a=neg_seq_voltage(201:end,2); %% Negative sequence
voltage is acquired
    [C,L] = wavedec(a,5,'db4'); %% Wavelet Transformation
of the signal
    i=1;
    count=L(1);
    while i<=5
        data=C(count+1:count+L(i+1));
        E(j,i)=norm(data,2); %% Calculating Energy
        SD(j,i)=std(data);    %% Calculating Standard
Deviation
        count=count+L(i+1);
        i=i+1;
    end
    activeP=activeP+.035e6; %% Incrementing load
    fprintf('Iterarion Number : %d \n',j);
end
fprintf('\n\n TABLE \n\n');
Loading=.875:.035:1.75;
Loading=Loading';
T = table(Loading,SD,E);
disp(T);
```

APPENDIX C

- MATLAB Code for ANN training and testing

```
%% ANN Training and Prediction - Thesis Project M. Tech
2016 - Sleeba Paul

clear ; close all; clc;

%% Initializing Parameters
input_layer_size  = 5;  % For Input features
hidden_layer_size = 50;   % 40 hidden units
num_labels = 5;          % 4 labels (States of Operation)
load('ex4data.mat');   % Loading the Features and Labels
num_points = length(X);
split_point = round(num_points*0.7);
seq = randperm(num_points);
X_train = X(seq(1:split_point),:);
Y_train = y(seq(1:split_point));
X_test = X(seq(split_point+1:end),:);
Y_test = y(seq(split_point+1:end));

fprintf('\nInitializing Neural Network Parameters ...\n')

initial_Theta1 = randInitializeWeights(input_layer_size,
hidden_layer_size);
initial_Theta2 = randInitializeWeights(hidden_layer_size,
num_labels);

% Unroll parameters to pass to the advanced optimization
techniques (fmincg) in MATLAB
initial_nn_params = [initial_Theta1(:) ;
initial_Theta2(:)];

%%Training NN

fprintf('\nTraining Neural Network... \n')

%MaxIter can be increased to improve the accuracy and
better parameter training
options = optimset('MaxIter', 200);

lambda = 0; % Useful for Over-Fitting situations. For
current data its not needed.

% Create "short hand" for the cost function to be
minimized
costFunction = @(p)
nnCostFunction(p,input_layer_size,hidden_layer_size,num_l
abels, X_train, Y_train, lambda);
```

```
% Now, costFunction is a function that takes in only one
argument (the neural network parameters)
[nn_params, cost] = fmincg(costFunction,
initial_nn_params, options);

% Obtain Theta1 and Theta2 back from nn_params
Theta1 = reshape(nn_params(1:hidden_layer_size *
(input_layer_size + 1)),hidden_layer_size,
(input_layer_size + 1));

Theta2 = reshape(nn_params((1 + (hidden_layer_size *
(input_layer_size + 1))):end),num_labels,
(hidden_layer_size + 1));

fprintf('Program paused. Press enter to continue.\n');
pause;


%% Visualize Weights
fprintf('\nVisualizing Neural Network... \n')
fprintf ('\n Theta1 : \n');
disp(Theta1);
fprintf ('\n Theta2 : \n');
disp(Theta2);

%% Implement Predict
% After training the neural network, it can be used to
predict the labels and compute the prediciton accuracy.
pred = predict(Theta1, Theta2,X_test);
disp(pred);
fprintf('\nTraining Set Accuracy: %f\n', mean(double(pred
== Y_test)) * 100);
```

APPENDIX D

- Python code for GUI and Web Service Consumption

```
from Tkinter import *

import urllib2 # If you are using Python 3+, import urllib instead of urllib2

import json


fields = 'Loading (MW)','SD4', 'SD3', 'E4', 'E3'


def fetch(entries):

   text=[]

   for entry in entries:

      text.append(entry[1].get())

   text.append('1')

   data =  {"Inputs":{"input1":{"ColumnNames":["Loading", "SD4",            "SD3",            "E4",            "E3", "Label"],"Values":[text]}},"GlobalParameters":{}}

   body = str.encode(json.dumps(data))

   url                                                    = 'https://ussouthcentral.services.azureml.net/workspaces/8e3a01b9b5a94cd8be8f69c85bfd1215/services/923ca965d52a46788d91c32cd9cdc9fd/execute?api-version=2.0&details=true'

   api_key                                                = 'zogVJpUKUmTEEh3auMuVYZ1q1tBKBWA+tHeZWIyzuG2sYUfxHKJ7F4IpXhGALe5IiVpunzMHjDF8i0gTImfqfA==' # Replace this with the API key for the web service

   headers     =     {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}

   req = urllib2.Request(url, body, headers)
```

```
   try:
      response = urllib2.urlopen(req)
      # If you are using Python 3+, replace urllib2 with
urllib.request in the above code:
      # req = urllib.request.Request(url, body, headers)
      # response = urllib.request.urlopen(req)
      result = response.read()
      out=json.loads(result)

score=out['Results']['output1']['value']['Values'][0][12]
      if score=='1':
         status=" No Events"
      elif score =='2':
         status="Islanding"
      elif score=='3':
         status="L-G Fault"
      elif score =='4':
         status=" L-L Fault"
      elif score=='5':
         status="Non-Linear Load Switch"
      w4.configure(text="STATUS : "+status)
   except urllib2.HTTPError, error:
      print("The request failed with status code: " +
str(error.code))
      # Print the headers - they include the requert ID
and the timestamp, which are useful for debugging the
failure
      print(error.info())
      print(json.loads(error.read()))
```

```
def makeform(root, fields):
   entries = []
   for field in fields:
      row = Frame(root)
      lab = Label(row, width=15, text=field, anchor='w')
      ent = Entry(row)
      row.pack(side=TOP, fill=X, padx=5, pady=5)
      lab.pack(side=LEFT)
      ent.pack(side=RIGHT, expand=YES, fill=X)
      entries.append([field, ent])
   return entries
def quit():
    root.destroy()
if __name__ == '__main__':
   root = Tk()
   root.title('Disturbance Predictor ')
   root.geometry('530x350')
   new=Frame(root)
   w2=Label(new,text='1.75 MW HYBRID SYSTEM',anchor='w')
   w3=Label(new,text='PV (Rated) - 0.25 MW  ',anchor='w')
   w5=Label(new,text='Wind  Plant  (Rated)  -  1.5  MW
',anchor='w')
   w6=Label(new,text='Grid Voltage - 25 kV  ',anchor='w')
   new.pack(side=TOP, fill=X, padx=5, pady=10)
   w2.pack()
   w3.pack(side=LEFT)
```

```
   w5.pack(side=LEFT)
   w6.pack(side=LEFT)
   ents = makeform(root, fields)
   win=Frame(root)
   w4=Label(win,text='',anchor='w')
   win.pack(side=TOP, fill=X, padx=5, pady=5)
   w4.pack()
   root.bind('<Return>',     (lambda     event,     e=ents:
fetch(e)))
   b1 = Button(root, text='Show',
          command=(lambda e=ents: fetch(e)))
   b1.pack(side=LEFT,padx=50,pady=5)
   b2 = Button(root, text='Quit', command=quit)
   b2.pack(side=RIGHT,padx=50,pady=5)
   root.mainloop()
```