ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

COMBINED WAVELET-NEURAL CLASIFIER FOR POWER DISTRIBUTION SYSTEMS

M.Sc. Thesis by

Oben DAĞ, B.Sc.

Department : Electrical Engineering

Programme: Electrical Engineering

MAY 2002

ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

COMBINED WAVELET-NEURAL CLASIFIER FOR POWER DISTRIBUTION SYSTEMS

M.Sc. Thesis by
Oben DAĞ, B.Sc.

(504991175)

Date of submission : 13 May 2002

Date of defence examination: 31 May 2002

Supervisor (Chairman): Assoc. Prof. Dr. Canbolat UÇAK

Members of the Examining Committee Prof. Dr. R. Nejat TUNÇAY (İTÜ.)

Assis. Prof. Dr. Neslihan ŞENGÖR (İTÜ.)

MAY 2002

ACKNOWLEDGEMENTS

First and foremost, I would like to thank to my family for their endless support, love and patience without which I could not have achieved this thesis.

Then, I would like to thank my supervisor, Assoc. Prof. Dr. Canbolat UÇAK for his continuous support and for the opportunities given in the studies of the thesis at Electrical Engineering Department. I am so pleased to have him as a supervisor in my masters career.

I am grateful to Prof. Dr. Nejat Tuncay for his assistance in providing the simulation software during the thesis study.

Also I appreciate the contributions of chief engineer Mehmet Gönen and engineer Nazım Kenc from "Bogazici Elektrik Dağıtım A.S." for providing me the model distribution system.

My special thanks goes to all my friends in Electrical Engineering Department, especially research assistants engineer Hakan Özcan and Ekrem Gürsoy, M.Sc. for their valuable contributions and critics during the thesis study.

This study is sponsored by the Graduate Thesis Support Program of Institute of Science and Technology, ITU.

May 2002 Oben Dağ

CONTENTS

ABBREVIATIONS

ANN : Artificial Neural Network
AI : Artificial Intelligence
BP : Back Propagation
SOM : Self Organizing Map
RBF : Radial Basis Function
CPN : Counter Propagation Network
FFNN : Feed Forward Neural Network
TDNN : Time Delay Neural Network
EMTP : Electromagnetic Transient Program
LVQ : Learning Vector Quantization
ART : Adaptive Resonance Theory
MSD : Multi-resolution Signal Decomposition
HVDC : High Voltage DC
AC : Alternative Current
DC : Direct Current
FT : Fourier Transform
STFT : Short Time Fourier Transform
WT : Wavelet Transform
DFT : Discrete Fourier Transform
WFT : Windowed Fourier Transform
FFT : Fast Fourier Transform
DWT : Discrete Wavelet Transform
CWT : Continuous Wavelet Transform
LP : Low Pass
HP : High Pass
PR : Perfect Reconstruction
FIR : Finite Impulse Response
MRA : Multi Resolution Analysis
VLSI : Very Large Scale Integrate
LMS : Least Mean Square
VQ : Vector Quantization
SOFM : Self Organizing Feature Map
BMU : Best Matching Unit
TRF : Transformer
FIA : Fault Inception Angle
HF : High Frequency
LF : Low Frequency
WTC : Wavelet Transform Coefficient
QE : Average Quantization Error
U-matrix : Unified Distance Matrix

LIST OF TABLES

LIST OF FIGURES

Page No

LIST OF SYMBOLS

$f(t)$	: A time-domain signal
$L^2(R)$	: Lebesgue vector space for square integrable functions
$\Psi(t), h_c(x)$	: Wavelet functions
$\Phi(x), g_c(x)$	: Scaling functions
$W(a,b)$	: CWT of the signal $f(t)$
$DPWT(m,n)$	: Discrete parameter Wavelet Transform coefficients
$g(n)$	: Half-band LP filter
$h(n)$	: Half-band HP filter
$\varphi(v)$	: Transfer function
$\alpha(t), \eta(t)$	: Learning rate functions
w_{kj}	: Synaptic weight function
E	: Error function
$m_i(t)$	: A set of variable codebook (prototype) vectors
$N_b(t), h_{bi}(t)$	: Neighborhood function centered on the best matching unit b
$\sigma(t)$	: Width of the topological neighborhood function
V_a, V_b, V_c	: Phase-to-ground voltages for *a,b,c* phases
I_a, I_b, I_c	: Line currents for *a,b,c* phases
$Daub-n$	: Daubechies wavelet function with an *n*-coefficient filter
$d_1(n)$	: Detail signal at scale *1*
$c_1(n)$	: Approximation signal at scale *1*
mu	: Number of codebook vectors (number of map units)
V	: Vector spaces

ELEKTRİK DAĞITIM SİSTEMLERİNDE BİRLEŞİK DALGACIK-SİNİR AĞI TABANLI SINIFLAYICI

ÖZET

Bu çalışmada, dağıtım sistemlerinde hibrid "Dalgacık-Yapay Sinir Ağı (YSA) tabanlı" bir yaklaşımla arıza sınıflama işlemi gerçeklenmiştir. 34.5 kV "Sağmalcılar-Maltepe" dağıtım sistemi PSCAD/EMTDC yazılımı kullanılarak arıza sınıflayıcı için gereken veri üretilmiştir. Tezin amacı, on farklı kısa-devre sistem arızalarını tanımlayabilecek bir sınıflayıcı tasarlamaktır. Sistemde kullanılan arıza işaretleri 5 kHZ lik örnekleme frekansı ile üretilmiştir. Farklı arıza noktaları ve farklı arıza oluşum açılarındaki hat-akımları ve hat-toprak gerilimlerini içeren sistem arızaları ile bir veri-tabanı oluşturulmuştur. "Çoklu-çözünürlük İşaret Ayrıştırma" tekniği kullanılarak altı-kanal akım ve gerilim örneklerinden karakteristik bilgi çıkarılmıştır. PSCAD/EMTDC ile üretilen veri bu şekilde bir ön işlemden geçirildikten sonra YSA-tabanlı bir yapı ile sınıflama işlemi gerçekleştirilmiştir. Bu yapının görevi çeşitli sistem ve arıza koşullarını kapsayan karmaşık arıza sınıflama problemini çözebilmektir. Bu çalışmada, Kohonen'in öğrenme algoritmasını kullanan bir "Kendine-Organize Harita" ile "Eğitilebilen Vektör Kuantalama" teknikleri kullanılmıştır. Bu "dalgacık-sinir ağı" tabanlı arıza sınıflayıcı ile eğitim kümesi için % 99-100 arasında ve sınıflayıcıya daha önce hiç verilmemiş test kümesi ile de % 85-92 arasında sınıflama oranları elde edilmiştir. Elde edilen başarım oranları literatürdeki sonuçlara yakındır. Geliştirilen birleşik "dalgacık-sinir ağı" tabanlı sınıflayıcı elektrik dağıtım sistemlerindeki arızaların belirlenmesinde iyi sonuçlar vermiş ve iyi bir performans sağlamıştır.

COMBINED WAVELET-NEURAL CLASIFIER FOR POWER DISTRIBUTION SYSTEMS

SUMMARY

In this study an integrated design of fault classifier in a distribution system by using a hybrid "Wavelet-Artificial Neural Network (ANN) based" approach is implemented. Data for the fault classifier is produced by using PSCAD/EMTDC simulation program on 34.5 kV "Sagmalcılar-Maltepe" distribution system in Istanbul. The objective is to design a classifier capable of recognizing ten classes of three-phase system faults. The signals are generated at an equivalent sampling rate of 5 KHz per channel. A database of line currents and line-to-ground voltages is built up including system faults at different fault inception angles and fault locations. The characteristic information over six-channel of current and voltage samples is extracted by the "Wavelet Multi-resolution Analysis" technique, which is a preprocessing unit to obtain a small size of interpretable features from the raw data. After preprocessing the raw data, an ANN-based tool was employed for classification task. The main idea in this approach is solving the complex fault (three-phase short-circuit) classification problem under various system and fault conditions. In this project, a self-organizing map, with Kohonen's learning algorithm and type-one learning vector quantization technique is implemented into the fault classification study. The performance of the wavelet-neural fault classification scheme is found to be around "99-100%" for the training data and around "85-92%" for the test data, which the classifier has not been trained on. This result is comparable to the studied fault classifiers in the literature. Combined wavelet-neural classifier showed a promising future to identify the faults in electric distribution systems.

1. INTRODUCTION

The quality of electric power has become an important issue for electric utilities and their customers. Customers, in particular, have become less tolerant of power quality disturbances and faults because these phenomena degrade the performance and efficiency of customer loads.

In order to improve the quality of power, electric utilities continuously monitor power delivered at customer sites. Disturbance waveforms are captured and recorded continuously using power monitoring instruments.

Existing methods to analyze and identify power disturbances are delicate and laborious since the primary methods are based on visual inspection of the waveforms. So, power quality engineers are swamped with an enormous amount of data to inspect [1].

It would be desirable if the data collection process could be further automated and the monitoring device not only monitors and records the disturbances, but also classifies them according to appropriate criteria. This would help to immediately detect a disturbance or fault and then make the appropriate decision to eliminate the fault. Therefore, this would minimize the customer displeasure and provide to obtain optimum efficiency from the power system.

1.1 Introduction and Background

One of the most common techniques utilized for fault analysis is one based on the symmetrical components theory [2]. This technique requires computation of symmetrical components phasors, resulting in positive, negative, and zero sequence phasors. The computation of phasors requires appropriate processing considerations. The most important considerations in computing power system phasors are the sampling rate, antialising filters, and data window. The sampling rate is determined

by the method used for phasor computation such as Fourier transform. The antialising filters are used to band limit the frequency spectrum of the input current and voltage to meet the sampling theorem, which states that the sampling frequency should be at least twice the highest frequency in the spectrum. The data window consideration relates to the number of samples required to compute a phasor. The most common data window is one cycle. When symmetrical component phasors are calculated; known theory of fault analysis is applied to determine fault occurrence and fault type.

Fault type and phase classification utilizes the negative and zero sequence components (magnitude and phase) of the currents and voltages, to classify the fault type and phase [3]. Under normal and symmetrical fault (three line-to-ground) conditions, the zero and negative sequence components in the line currents are nearly zero. The presence of only the negative sequence component in the fault current indicates that a line-to-line fault has occurred. The presence of negative and zero components indicates that a fault of single line-to-ground or double line-to-ground has occurred [4].

Another approach that can be used for fault classification is to utilize samples of currents and voltages directly without computation of phasors and related symmetrical components. There is no need to perform extensive filtering to obtain phasors. Instead, transient waveform data can be utilized directly to perform the required processing. In addition, the data window does not need to satisfy particular rules present for the phasor calculation. This approach is based on the use of artificial neural networks [2].

An artificial neural network (ANN) is a parallel, distributed, information processing structure consisting of processing elements, which can possess a local memory and carry out localized information processing operations [5]. Each processing element has a single output connection that branches into as many connections as desired (each carrying the same signal, that the processing element produced). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local; i.e., it must depend only upon the current values of the input signal arriving at the processing element by the connections and upon values stored in the processing

element's local memory. The key elements of most neural net descriptions are distributed representation, the local operations, and nonlinear processing. ANNs are primarily used in situations in which only a few decisions are required from a massive amount of data and situations in which a complex nonlinear mapping must be learned. Main applications of neural computing include:

- Functional approximation,
- Clustering,
- Data compression,
- Optimization,
- Topological mapping.

ANNs are useful in cases where the nature of the input-output functional relationship is neither well defined nor easily computable. Furthermore, ANNs are able to compute the answer quickly by using associations learned from previous experience. When an ANN is fully trained, it is capable of mapping an unfamiliar input vector to an arbitrary surface. In other words, an artificial neural net generalizes instead of performing "table-lookup". ANNs have been successfully applied to various pattern classification problems in terms of their learning ability, high discrimination power, and generalization ability. Classification, by definition, means to assign a physical object or event into one of several prespecified classes based on the extraction of significant features and the processing or analysis of these features [6]. In this respect, ANN technique provides the ability to classify the faulted phase/phases by identifying different patterns of the associated voltages and currents.

Recently, artificial intelligence (AI) techniques that include artificial neural networks (ANNs), fuzzy logic, genetic algorithms, and expert systems have been used to solve many nonlinear classification problems [7]. Since each branch has its own advantages/disadvantages, for any complex classification task, it is essential to compare all possible AI techniques and then choose the one appropriate for solving a specific problem. For example, in the case of fault classification, an ANN on its own or an ANN integrated with fuzzy logic or genetic algorithm for training purposes can be employed; it should be mentioned that an ANN on its own requires a longer

training time compared to the latter approach, but it is important to note that once trained and provided the size of the ANN is not too large, its testing and application is fast and hence a fault classifier based solely on an ANN can satisfy the speed requirements on an on-line fault classification scheme. Another technique, such as that based on combined fuzzy logic and an expert system, has been found to be useful for fault detection in power systems but such an approach is not particularly well suited for fault classification [8]. Also, though much promise has been shown by expert systems technology, a fundamental limitation is the a priori formulation of "rules" for the device or system application. In other words, an expert system solution to a given problem presumes that some human expert can solve the problem. This is not a serious limitation for many problems, since heuristic rules can be derived in most cases; however, expert systems cannot be applied to problems for which little human expertise exists, as in the case of fault classification with many variables and nonlinear characteristics.

From the view of ANN training techniques, they can be defined as supervised, unsupervised, and reinforced learning algorithms. For example, one of the typical supervised learning algorithm is error back-propagation (BP), which employs a nonlinear regression technique to achieve minimum error goal. Even though it has been reported that BP is adequate for though pattern classification problems owing to its high discrimination power and excellent generalization ability [6], the number of classes to be allowable is too small to apply it directly to large-set classification problems without preclassification. In other words, as the number of classes increases, the computational complexity of the learning problem quickly reaches unmanageable proportions. Furthermore, it is very difficult to determine the structure and size of the network for the classification of large-set and complex patterns. It is presented in [7] that for a fault classification study, when the voltage and current waveforms are preprocessed into nonstationary waveforms, although a fault classifier with the supervised training technique can reach the desired global minimum, the classification rate is only about 79%. However, when a fault classifier is based on an unsupervised (i.e., Kohonen ANN) or combined supervised and unsupervised training technique, such as self-organizing mapping (SOM), radial-basis function (RBF), Counter Propagation Network (CPN), the classification rate can reach a high level of about 95%. The last ANN training technique is reinforced learning

algorithm, i.e., a genetic algorithm which is used to search the weight space of a multilayer feed-forward ANN without the use of any information. The basic concept behind this technique is that complete sets of weights are coded into a string, which has an associated fitness finding attribute for the optimal weight. Although the reinforced learning performs a global search and therefore minimizes the possibility of getting stuck in local minima, the training is very time-consuming; classification rate is around 85% [7].

It is well known that, there is usually a high volume of recorded event data to be processed and classified when dealing with power quality analysis. This makes it very difficult and time-consuming to interpret the data and provide useful operations. Then, large dimensionality of the data is one general problem that exists. Moreover, a major concern arising from the classification of a large data set is the complexity of the discrimination process. Due to changes in the disturbance type, duration and its frequency components (which may overlap in time), the parameters in the discriminant model become highly variable. This leads to a considerable deterioration in classification performance of the classifier. To overcome this problem, it is often necessary to decrease the number of variables and/or data to a manageable size [9].

Wavelet analysis techniques have been applied with success in a wide variety of research areas such as signal analysis, image processing, data compression, de-noising and numerical solution of differential equations. The wavelet analysis techniques have been proposed extensively in the literature as an approach for fault detection, localization and classification of different power system transients.

When a fault occurs in a distribution system, disturbance signals like transients would present in the voltage and current signals. These high-frequency parts of the signals carry essential information that could be used in classifying the fault types. By careful observation of current and voltage waveforms and frequency spectra, some characteristics may be identified for each fault type. Wavelet transform provides the task of extracting the information in the current and voltage waveforms. As a means to reduce the number of inputs into the ANN, Wavelet transform may be used to find the reduced model of the event data.

1.2 Literature Review

This section provides a comprehensive literature review of studies related to fault classification and power quality disturbance analysis with artificial intelligence techniques such as artificial neural networks, fuzzy-logic techniques, and expert systems.

In [10], A. K. Ghosh and D. L. Lubkeman introduced an artificial neural network methodology for the classification of waveforms that are captured, as part of a larger scheme to automate the data collection process of recorders. They investigated two different neural network paradigms: feed-forward neural network (FFNN) and the time-delay neural network (TDNN), which has the ability to encode temporal relationships found in the input data and exhibit a translation-shift invariance property. Also a comparison of these paradigms based on a typical distribution circuit configuration is presented by the authors. They showed that the classification rate of FFNN (72%) is better than TDNN (57%) under different events simulated by EMTP program on a distribution system including different system and fault conditions. They also used a modified TDNN architecture and obtained classification rate up to (92%). The authors concluded that future work would involve the implementation of the classification/data collection scheme in the real systems as a part of data monitoring and classifying implementation.

Another work described in [11] addresses the problems encountered by conventional techniques in fault type classification in double-circuit transmission lines, which arise due to the mutual coupling between the two circuits under fault conditions, and this mutual coupling is highly variable in nature. It is shown that a neural network based on combined unsupervised/supervised training methodology provides the ability to accurately classify the fault type by identifying different patterns of the associated voltage and currents. The authors also compared their technique with a supervised training algorithm (back-propagation (BP) network classifier). The proposed system was tested under different fault types, location, resistance and inception angle; different source capacities and load angles. The authors showed that the technique based on hybrid SOM-LVQ network correctly identifies the faulted

phases in spite of the presence of the highly variable mutual coupling effect between the two circuits. Also they figured out that the performance of the proposed fault classification scheme is better than the BP network with supervised learning. They declared that the BP network needs much larger number of training sets and the training phase of BP is very slow and time consuming. Moreover, retraining the BP network with new data associated with contingencies may not converge to the desired value. When the learning gets stuck on local minima, the requisite performance can never reached. They also included that the number of neurons in the Kohonen map is very much dependent on the number of training sets. But, in practice, this problem is mitigated by the fact that there is a requirement for a much more smaller number of data sets as compared to the BP networks to cover all types of practically encountered different fault conditions.

The study of F. N. Chowdhury and J. L. Arevena presents a modular and integrated approach to the problem of fast fault detection and classification [12]. They emphasized that although the specific application example studied is a power system, their method would be applicable to dynamic systems as well. They proposed a model-free case that use the concepts from signal processing and wavelet theory to create fast and sensitive fault indicators. The method to create indicators utilizes multirate filter banks based on wavelet decomposition of a given data. Then, they used the indicators to be analyzed by artificial neural networks to create intelligent decision rules. After a detection, the fault indicator is processed by a Kohonen network to classify the fault. They included that results of computer experiments with simulated faulty transmission lines are satisfactory. They concluded that their integrated and modular approach can eventually be developed into a widely applicable tool in detection and classification of faults in dynamic systems.

The study in [13] explores the possibility of using neural networks to identify faults that may have occurred in an ac-dc power system. This study showed that based on the ability of neural networks to distinguish reliably between different types of faults, appropriate control measures can be taken to improve the dynamic performance of the ac-dc power system. The authors emphasized that in their study the CPN based on Kohonen layer was used for its simplicity, easy training features and good statistical model representation of the input environment. They concluded that the artificial neural networks could be used to distinguish typical faults (single line-to-

ground, double line-to-ground, three phases-to-ground, line-to-line faults; dc line fault) that can occur in an ac-dc power system. Also they figured out that once the work on fault identification has been completed, it is intended to use the resulting information to adapt the dc controller parameters to optimize the dynamic behavior of the ac-dc power system.

A study on an artificial neural network to classify power system disturbances according to power system response characteristics is presented by K.L. Frick and S.K. Starrett in [14]. They used Prony analysis to represent the original time series data as a sum of exponential terms defined by frequency, phase, amplitude, and damping coefficients. These variables are presented as an input to the competitive layer architecture with Kohonen learning rule. They found that there is a correlation between the classifications found by the competitive layer neural network and the geographic location of the disturbance. They also figured out that the system conditions also play a significant role in the disturbance characteristics.

Fuzzy logic and expert systems have also found applications in power systems for fault classification. In the following paragraphs the studies related to fuzzy-logic, and expert systems for fault classification will be mentioned.

In [15] the authors presented a new approach to real-time fault detection and classification in power transmission systems using fuzzy-neuro techniques. They pointed out that the integration with neural network technology enhances fuzzy logic systems on learning capabilities. In this study the symmetrical components in combination with three line currents are utilized to detect fault types such as single line-to-ground, line-to-line, double line-to-ground and three line-to-ground. The authors proved that the proposed approach gives a fast, accurate and robust classification for various system conditions. They concluded that the fuzzy-neuro model with further refinement could be implemented in an actual power system to monitor occurrence of faults and take necessary action.

In [7], the problems of fault diagnosis in complex parallel transmission systems is addressed. In this study, a fuzzy ARTmap (adaptive Resonance Theory) neural network is employed and is found to be well suited for solving the fault classification problem under various system conditions. They proved that the artificial neural

network based on the supervised adaptive resonance theory can identify the faulted phase with a high degree of accuracy. The classification technique is compared with a Neural Network technique based on the error back-propagation training algorithm, and found that the fuzzy ARTmap technique is better suited for solving the fault diagnosis problem, the classification rate is higher and also the training times required are shorter for the same training sets. They concluded that this proposed fault diagnosis technique based on fuzzy ARTmap network is well suited for the complex transmission systems than other more conventional ANN-based techniques.

In [16], a system for the identification of power quality violations is proposed. A two-stage system that employs the potentials of the wavelet transform and the adaptive neurofuzzy networks is presented in the study. For the first stage, the authors used the wavelet multiresolution signal decomposition technique to denoise and then decompose the monitored signals of the power quality events to extract their detailed information. This stage provides a set of reduced data set for the training data. A modified organization map of the neurofuzzy classifier is trained with the extracted features to recognize ten event categories. The authors concluded that the proposed scheme can be extended to accommodate hybrid disturbed signals which can assist as a step towards building an intelligent recorder capable of the detection and identification of power quality violation events automatically.

A hybrid scheme using a Fourier Linear Combiner and a fuzzy expert system for the classification of transient disturbance waveforms in a power system is presented in [17] by P.K. Dash, S. Mishra, M. M. A. Salama, and a. C. Liew. In this study the captured voltage and current waveforms are passed through a Fourier Linear Combiner block to provide normalized peak amplitude and phase at every sampling instant. These features are than passed on to a diagnostic module that computes the truth value of the signal combination and determines the class to which the waveform belongs. The authors depicted that the fuzzy expert system yields a robust and accurate classification scheme for a variety of simulated waveforms containing harmonic distortions and noise. They concluded that the approach is found to be computationally simpler than the ANN and Wavelet approaches which are currently used for transient disturbance classification.

In the following paragraphs the signal processing studies related to Wavelet transform and Multiresolution signal decomposition techniques for fault detection and classification will be given.

In [18], the use of the Wavelet transform and Multiresolution signal decomposition as an analysis tool to detect and localize transient events and classify different power quality disturbances is presented. The authors emphasized that the property of Multiresolution signal decomposition (MSD) shows the ability to extract important information from the analyzed distorted signal and furthermore separate power quality problems that overlap in both time and frequency. They used std_MSD curve technique, which is the standard deviation of the MSD coefficients, to construct a time-frequency picture of the distorted signal. This technique presents a classification role for the operator to detect, localize, and classify different power quality problems. They concluded that using std_MSD it is possible to distinguish among similar power quality problems. Also it can help in finding the source of disturbance.

An algorithm for detecting and classifying fault transients in underground cable systems based on the use of discrete wavelet transform is presented in [19] by W. Zhao, Y.H. Song, and Y. Min. The authors developed an algorithm for fault detection and classification based on discrete wavelet analysis on power cables. The authors emphasized that the property of multiresolution in time and frequency provided by wavelets allows an accurate time location of fault transients while simultaneously retaining information about the fundamental frequency and its high-order harmonics, which is efficient to extract characteristics of different types of fault in underground cable systems. They concluded that it is necessary to fully evaluate the proposed technique as part of protection relays or fault locators under a wide range of system and fault conditions.

1.3 Objective of The Thesis

In this thesis, it is intended to provide a contribution to the fault classification studies in the literature to further improve the classification performance and to present a new approach to the previous studies. The objective is to design a classifier capable of recognizing ten classes of three-phase system faults.

The thesis is organized as follows. An introduction to the PSCAD/EMTDC software, which is used for generating the data from the 34.5 kV distribution system, is presented in Chapter II. Chapter III presents the theoretical aspects of the wavelet transform and multi-resolution signal decomposition. Also a comparison between the wavelet transform, Fourier transform and short-time Fourier transform are presented in this section. Chapter IV illustrates the theoretical aspects of the artificial neural networks. The theoretical background is provided in the thesis to help the reader with the background of the techniques and also they are referred by other chapters to clarify some points. The application of the proposed technique on a distribution system is presented and the results are demonstrated in Chapter V. Finally, the conclusion is presented in Chapter VI.

2. ELECTROMAGNETIC TRANSIENTS SIMULATION PROGRAM

EMTDC is a transients simulator of electric networks with the capability of modeling complex power electronics, controls and the non-linear network. It has been evolving since the mid 1970's. The acronym stands for "Electro-Magnetic Transients in DC systems" since the program was originally developed for studying high-voltage DC (HVDC) system.

The origin of EMTDC is based on the study of Dr. Hermann Dommel's paper published in the IEEE Transactions of Power Apparatus and Systems in April 1969 [20].

The Manitoba HVDC Research Center created PSCAD/EMTDC in the early 1990s for use on Unix workstations. PSCAD is the graphical user interface. With the emergence of Windows OS for personal computers and its expanding capabilities, the Manitoba HVDC research Center developed a new version of PSCAD known as PSCAD/EMTDC Version 3, which is a versatile tool to study AC as well as DC power system problems.

One of the methods of understanding the behaviour of a complex system is to study its response for disturbances or parametric variations. Simulation is one way of producing these responses. In power systems; these responses can be studied by observing either the time domain instantaneous values, time domain "rms" values, or the frequency components of the response.

PSCAD/EMTDC is a simulation tool for analyzing power systems. PSCAD is the graphical user interface and EMTDC is the simulation engine. PSCAD/EMTDC is most suitable for simulating the time domain instantaneous responses, known as electromagnetic transients of electrical systems. The PSCAD Graphical Interface enhances the power of EMTDC. It allows to schematically construct a circuit, run a simulation, analyze the results, and manage the data in a graphical environment.

Simulation is one method of analysis which can be used to examine a complicated or non-linear model or process. The operation of that model can be tested by subjecting it to disturbances and parameter variations and the stability of its response can be observed. In electric power systems, some of the components which EMTDC can model are [20]:

- Resistor, inductor and capacitor circuit elements.
- Mutually coupled windings such as transformers.
- Distributed frequency dependent transmission lines and cables.
- Sources, both Thevenin (voltage) and Norton (current).
- Switches, breakers, thyristors, diodes.
- Analogue and digital control functions.
- AC machines.
- Meters and measuring functions.
- Generic DC and AC controls.
- HVDC, and Static Var Compensator.

EMTDC is generally used in planning, operation, design, teaching and advanced research by engineers, manufactures, consultants, research and academic institutions. Examples of typical studies which have been investigated using EMTDC are as follows [20]:

- Studies of AC networks consisting of rotating machines, exciters, governors, turbines, transformers, transmission lines, cables, loads.
- Relay coordination.
- Transformer saturation effects.
- Insulation coordination of transformers, breakers and arrestors.

- Impulse testing of transformers.
- Sub-synchorous resonance (SSR) studies of networks with machines, transmission lines and HVDC systems.
- Filter design and harmonic analysis
- Control system design and coordination of HVDC; including STATCOM and VSC.
- Studies to determine the worst case over voltage due to lightining strikes, faults or breaker operations.
- Investigate the pulsing effects of diesel engines and wind turbines on electric networks.

3. TIME-FREQUENCY ANALYSIS

3.1 Introduction

The aim of signal analysis is to extract relevant information from a signal $f(t)$ by transforming it from one domain to another. The transformation of a function is a mathematical operation that results in a different representation of it. One example is Fourier Transform (FT) that gives superposition of functions by the building blocks or basis functions: sines and cosines. As an illustration to Fourier Transform, a prism acts as a transformer by decomposing sunlight into its visual spectrum of different colors (frequencies). So a transform reveals the composition of a signal in terms of the building blocks, or basis functions (of the transformed domain) that are not readily available from the original signal [21].

Fourier series are ideal for analyzing periodic signals, since the harmonic modes used in the expansions are themselves periodic. By contrast, the Fourier analysis is a far less natural tool because it uses periodic functions to expand non-periodic functions. Two possible substitutes are the windowed Fourier Transform and the Wavelet Transform [22].

While classical Fourier analysis manages to deal with periodic and stationary signals, it is inadequate to analyze non-stationary signals, signals with discontinuities and transients. The limitations of Fourier analysis are that they require all time functions involved to be periodic, and the Fourier analysis does not consider frequencies that evolve in time.

The analysis of non-stationary signals often involves a compromise between how well transitions or discontinuities can be located, and how finely long-term behavior can be identified. A typical example is the choice of window length in the Short Time (windowed) Fourier Transform (STFT). In STFT, the signal is divided into

small enough segments, where these segments (portions) of the signal can be assumed to be stationary. For this purpose, a window function is chosen [23].

Another concept in non-stationary signal analysis is the Wavelet Analysis. In contrast to a Fourier sinusoidal signal, which oscillates forever, a wavelet is localized in time and it lasts for only a few cycles. The wavelet transform (WT), like the STFT, provides an understandable transient signal representation corresponding to a time-frequency plane. This plane gives time and frequency related information of analyzed signal. WT is more efficient than Fourier analysis when a signal is dominated by transient behavior and discontinuities. Wavelet algorithm process data at different scales or resolutions such that a rough approximation of the signal might look stationary, while at a detailed level, for instance with small window, discontinuities may become apparent. Moreover, unlike STFT, which uses a single analysis window, the WT uses short windows at high frequencies and long windows at low frequencies. Thus the windowing of wavelet transform is adjusted automatically for low or high frequencies. The result in wavelet analysis is to see both the forest and the trees [24].

The basic concept in wavelet analysis is to select an appropriate wavelet prototype (or kernel) function, called an analyzing wavelet or mother wavelet, and then perform an analysis using shifted and scaled versions of the mother wavelet. Time (or space) analysis is performed with a contracted (high frequency) version of the prototype wavelet, while frequency analysis is performed with the dilated (low frequency) version of the same mother wavelet.

Since the original signal or function can be represented in terms of a wavelet expansion, the operations can be performed using just the corresponding wavelet coefficients.

Wavelet transforms have been proven to be very efficient in signal analysis. This efficiency comes from the reduction in the number of coefficients as the scaling factor increases. The wavelet expansion separates signal components that overlap in both time and frequency. Wavelets can be designed to fit different applications. The calculation of the discrete wavelet transform is well matched to digital computer [23].

3.1.1 Historical Perspective

The French mathematician Joseph Fourier asserted in 1807 that any 2π-periodic function $f(t)$ can be expressed as an infinite sum of periodic complex exponential functions [25]. He showed that $f(t)$ is the sum

$$f(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt) \tag{3. 1}$$

of its Fourier series, where the coefficients a_0, a_k, and b_k are calculated by

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(t)dt \tag{3. 2a}$$

$$a_k = \frac{1}{\pi} \int_0^{2\pi} f(t)\cos(kt)dt \tag{3. 2b}$$

$$b_k = \frac{1}{\pi} \int_0^{2\pi} f(t)\sin(kt)dt \tag{3. 2c}$$

Fourier's assertion played an essential role in the evolution of the functions in mathematics. After 1807, by exploring the meaning of functions, Fourier series convergence, and orthogonal systems, mathematicians gradually were led from their previous notion of frequency analysis to the notion of scale analysis. This provides analyzing $f(t)$ by creating mathematical structures that vary in scale. The procedure, briefly, is as follows: first construct a function and shift it by some amount then change its scale. Apply that structure in approximating a signal. Repeat the procedure. Take that basic structure, shift it, and scale it again. Apply it to the same signal to get a new approximation, and so on [25].

The first mention of wavelets appeared in an appendix to the thesis of A. Haar in 1909. One property of the Haar wavelet is that it has compact support, which means that it vanishes outside of a finite interval. Unfortunately, Haar wavelets are not continuously differentiable, which limits their applications [25].

In the 1930s, several groups working independently researched the representation of functions using scale-varying basis functions. Paul Levy, a physicist, investigated Brownian motion, a type of random signal by using a scale-varying basis function called the Haar basis function in 1930. He found that the Haar basis function is superior to Fourier basis functions for studying small-complicated details in the Brownian motion [25].

The beginning of the wavelet transform as a specialized field can be traced to the work of Grossman and Morlet in 1984. Their motivation in studying wavelet transforms was provided by the fact that certain seismic signals can be modeled suitably by combining translations and dilations of a simple, oscillatory function of finite duration called a wavelet [26].

In 1985, Stephane Mallat gave wavelets an additional jump-start through his work in digital signal processing. Mallat discovered some relationships between quadrature mirror filters, pyramidal algorithms, and orthonormal wavelet bases [25].

In 1993, Y. Meyer constructed the first wavelets. Unlike Haar wavelets, Meyer wavelets are continuously differentiable; however, they do not have compact support [25].

In 1996, Ingrid Daubechies used Mallat's work to construct a set of wavelet orthonormal basis functions that have become the cornerstone of wavelet applications today [25].

In the next section, the fundamentals of Fourier Transform and its variances will be given. Besides that, the similarities and dissimilarities between Fourier Transform and Wavelet Transform will also be described.

3.1.2 Fourier Transforms

The Fourier Transform's utility lies in its ability to analyze a signal in the time domain for its frequency content. The transform works by first translating a function in the time domain into a function in the frequency domain. The signal can then be analyzed for its frequency content because the Fourier coefficients of the transformed function represents the contribution of each sine and cosine function at each

frequency. The reconstruction of the signal can be done with Fourier coefficients by inverse Fourier transform that gives a mapping from the frequency domain to the time domain.

The Discrete Fourier Transform (DFT) estimates the Fourier Transform of a function from a finite number of its sampled points. The DFT has symmetry properties almost exactly the same as the Continuous Fourier Transform [25].

The classical Fourier Transform is a standard tool for stationary and periodic signals [24]. However, in the case of non-periodic signal, the summation of the periodic functions (sine and cosine) does not accurately represent the signal. And in the case of non-stationary signal, the classical FT does not give a good performance to adapt any abrupt change in the signal.

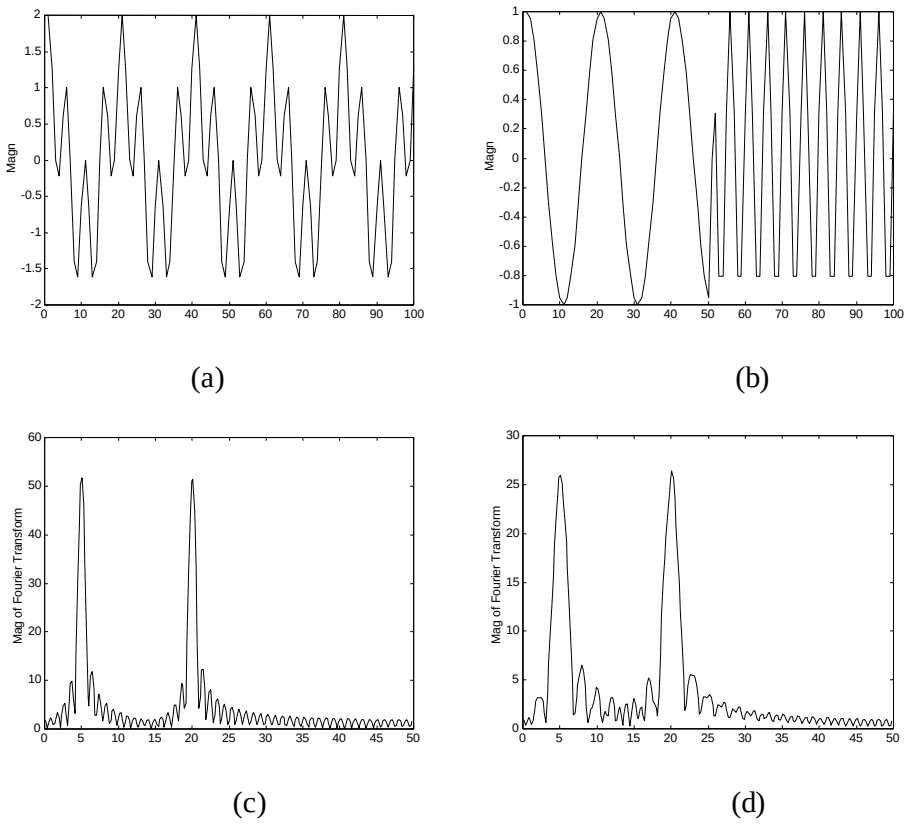


Figure 3.1 The illustration of frequency response for stationary and nonstationary signals (a) Stationary signal $f(t)$, (b) Non-stationary signal $g(t)$ (c) FT of $f(t)$, (d) FT of $g(t)$

Figure 3.1 illustrates the case of non-stationary signal problem in classical Fourier Transform. Figure 3.1a represents a stationary signal, $f(t)$ with two frequency components of *5* and *20* Hz., all-occurring at all times. And Figure 3.1c is the FT of

$f(t)$; that the two peaks in the figure correspond to two different frequencies. Another signal $g(t)$ with the same frequency components, *5* and *20* Hz. respectively, occurring at different times is given in Figure 3.1b, and Figure 3.1d represents its FT. It is clear that the FT of $f(t)$ is almost the same with the FT of $g(t)$. (The ripples in both of the Figure 3.1c and Figure 3.1d show that, those frequencies also exist in the signal; but the reason they have small amplitude is because, they are not major spectral components of the given signal.) To summarize, FT cannot distinguish the two signals very well. According to FT both of the signals are the same since they contain the same frequency components. Therefore, FT is not suitable for analyzing non-stationary signals.

The Windowed Fourier Transform (WFT) is a solution to the problem of better representing the non-periodic and non-stationary signals. WFT has the ability to give information about signals simultaneously in time and frequency domains. With WFT the input signal is segmented into sections, and each segment is analyzed for its frequency content separately. The idea is to introduce a local frequency parameter (local in time) so that the local FT looks at the signal through a window over which the signal is approximately stationary. So the effect of the window is to localize the signal in time [24].

To approximate a function by samples, and to approximate the Fourier integral by the discrete Fourier transform, requires applying a matrix whose order is the number of sample points (n). Since multiplying an ($n \times n$) matrix by a vector costs on the order of (n^2) arithmetic operations, the problem gets quickly worse as the number of sample points increase. If the samples are uniformly spaced, then the Fourier matrix can be factored into a product of just a few sparse matrices, and the resulting factors can be applied to a vector in total of order ($n \log n$) arithmetic operations. This is the so-called Fast Fourier Transform [23].

3.1.3 Wavelet Transform Versus Fourier transform

3.1.3.1 Similarities

The mathematical properties of the matrices involved in the FFT and Discrete Wavelet Transform (DWT) are similar. The inverse transform matrix for both the FFT and the DWT is the transpose of the original. So, both of the transforms can be viewed as a rotation in function space to a different domain. For the FFT, this domain contains basis functions that are sines and cosines. For the wavelet transform, this domain contains more complicated functions called wavelets, mother wavelets, or analyzing wavelets.

Besides that, the basis functions for both of the FFT and DWT are localized in frequency, which makes mathematical tools such as power spectra and scalograms useful for picking out frequencies and calculating power distributions.

3.1.3.2 Dissimilarities

A limitation of STFT is the fact that only one single window is used for all frequencies. So, once a window is chosen for STFT, then the time-frequency resolution is fixed over the entire time-frequency plane because the same window is used for all frequencies. This is shown in Figure 3.2b, while Figure 3.2a shows the basis functions of the STFT. For instance, if the signal is composed of small bursts associated with long quasi-stationary components, then each type of component can be analyzed with good time resolution or frequency resolution, but not both.

Another important case related to STFT is the Uncertainty Principle, which prevents the possibility of having arbitrarily high resolution in both time and frequency. It lower bounds the time-bandwidth product of possible basis functions by

$$\Delta T \cdot \Delta\Omega \geq (1/4\pi) \tag{3. 3}$$

where ΔT and $\Delta\Omega$ are the resolution in time and frequency of the STFT analysis [26]. Equation 3.3 is referred to as the uncertainty principle, or Heisenberg inequality (see Appendix 6). It means that one can only trade time resolution for frequency resolution, or vice versa in STFT.

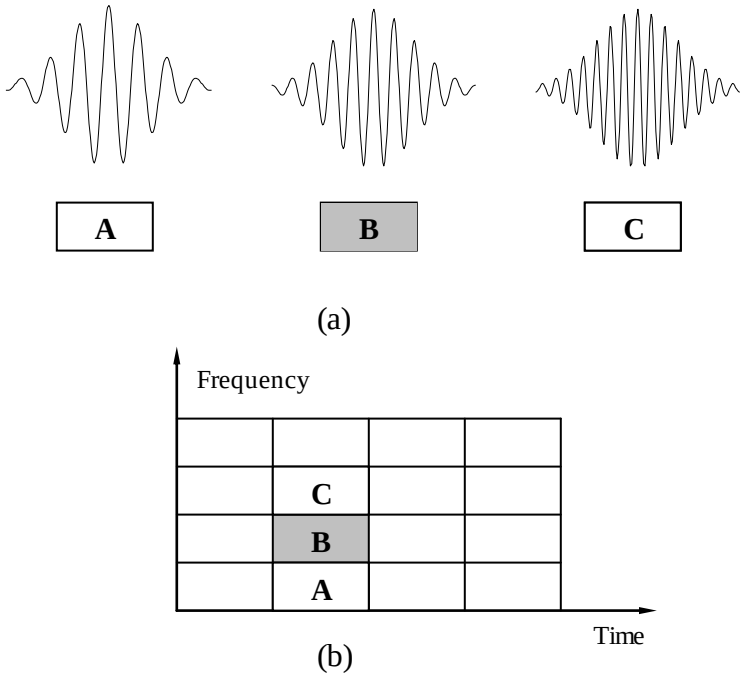


Figure 3.2 Basis functions and time-frequency resolution of the STFT. (a) Basis functions, (b) Coverage of time-frequency plane [24]

However by varying the window used, it is possible to trade resolution in time for resolution in frequency [23]. In order to isolate discontinuities in signals one would like to have some basis functions which are very short, while some long ones are required to obtain fine frequency analysis. One possibility for this task is having short high frequency basis functions, and long low frequency basis functions. This is exactly what is achieved with the WT, where the basis functions are obtained from a single kernel (prototype) wavelet by translation and dilation [23].

The time-frequency resolution of the WT involves a different tradeoff to the one used by the STFT; at high frequencies the WT is sharper in time, while at low frequencies, the WT is sharper in frequencies. The basis functions and time-frequency resolution of the WT is given in Figure 3.3. The middle functions in Figure 3.2a and Figure 3.3a are identical, and hence the time-frequency resolutions of the two methods are the same at that frequency.

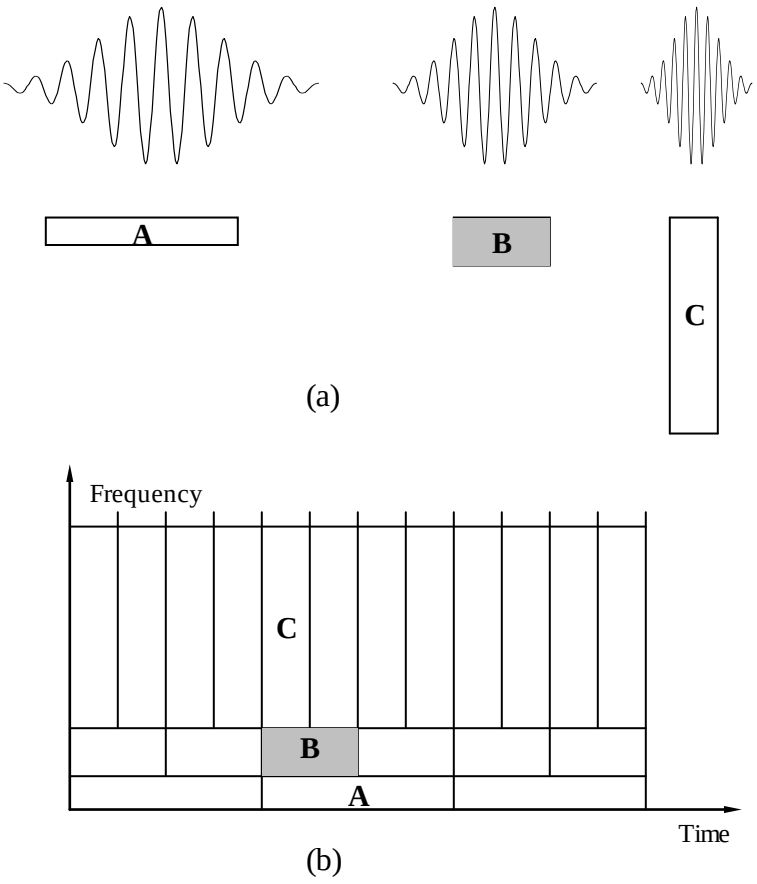


Figure 3.3 Basis functions and time-frequency resolution of the WT. (a) Basis functions, (b) Coverage of time-frequency plane [24]

As described above to overcome the resolution limitation of the STFT, the resolution ΔT and $\Delta\Omega$ chosen to vary in the time-frequency plane. When the analysis is viewed as a filter bank, the time resolution must increase with the central frequency of the analysis filters. So this makes $\Delta\Omega$ proportional with Ω, as $\Delta\Omega/\Omega = c$, where c is a constant. The analysis filter bank is then composed of band-pass filters with constant relative bandwidth (known as Constant-Q analysis). So instead of the frequency responses of the analysis filter to be regularly spaced over the frequency axis (this is the case of STFT shown in Figure 3.4a), they are regularly spread in a logarithmic scale as shown in Figure 3.4b [26].

The Heisenberg inequality in Equation 3.3 will still be satisfied for the WT, but now, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. One example of this is the case of two very close bursts that can always be eventually separated in the analysis by going up to higher analysis frequencies in order to increase time resolution. This

kind of analysis works best if the signal is composed of high frequency components of short duration plus low frequency components of long duration, which is often the case with signals encountered in practice [24].

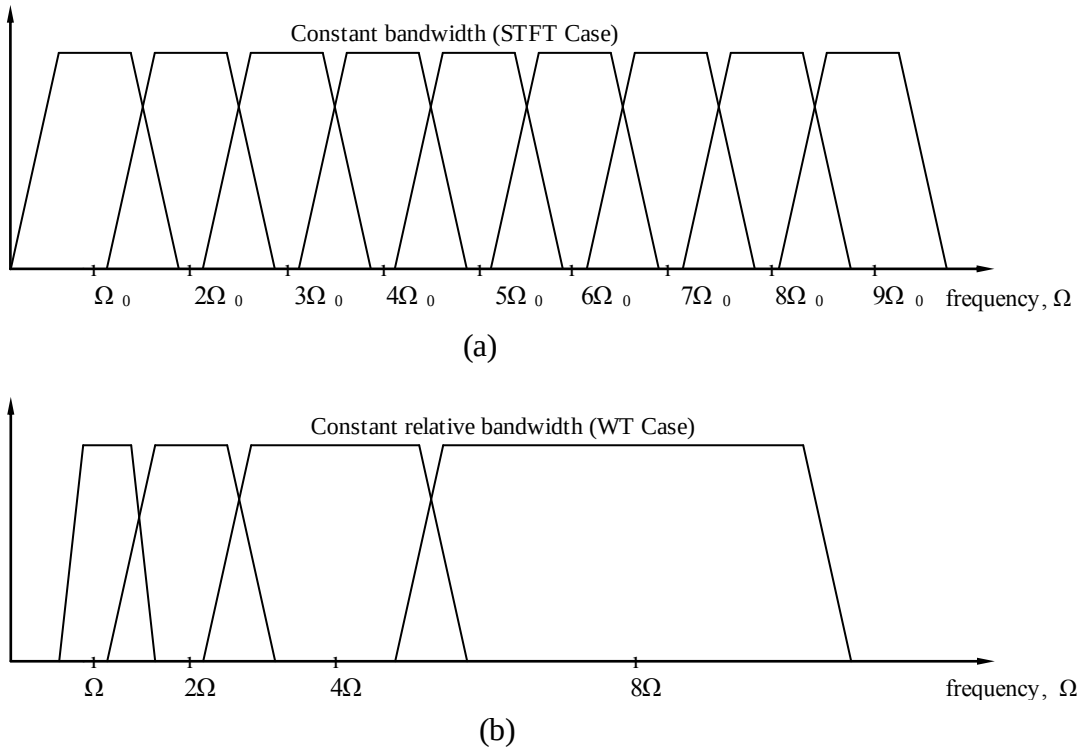


Figure 3.4 Division of the frequency domain, (a) for the STFT (uniform coverage), and (b) for the WT (logarithmic coverage) [24]

Another important point is about the size of the cells in time-frequency planes in Figure 3.2b and Figure 3.3b. In STFT the time and frequency resolutions are determined by the width of the analysis window, which is selected once for the entire analysis, i.e., both time and frequency resolutions are constant. Therefore the time-frequency plane consists of squares in the STFT case.

Regardless of the dimensions of the cells, the areas of all cells, both in STFT and WT, are the same and determined by Heisenberg's inequality. The area of a cell is fixed for each window function (STFT) or mother wavelet (WT), whereas different windows or mother wavelets can result in different areas. However, Equation 3.3 implies that all areas are lower bounded by $(1/4\pi)$. That is, the areas of the cells cannot be reduced arbitrary due to the Heisenberg's uncertainty principle. On the other hand, for a given mother wavelet the dimensions of the cells can be changed, while keeping the area the same. This is exactly what wavelet transform does.

As the main idea of time-frequency analysis is described, in the next sections of part three the mathematical formulations for both FT and Wavelet analysis will be given.

First the STFT and its features in mathematical point of view, afterwards the Wavelet analysis and synthesis with respect to CWT, and discrete-time WT will be defined. And finally the Multi-resolution Analysis, DWT, and digital filters with respect to pyramid scheme, sub-band coding and DWT in octave band filters will be described.

3.2 Short-Time (Windowed) Fourier Transform (STFT)

Given a signal $f(t)$, the standard Fourier Transform (FT) is [27],

$$(Ff)(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \tag{3. 4}$$

for every $t \in R$ where R is the set of real numbers. $(Ff)(\omega)$ gives a representation of the frequency content of $f(t)$, but information related to time-localization of, e.g., high frequency bursts cannot be seen from $(Ff)(\omega)$. Time localization can be provided by first windowing the signal $f(t)$ to obtain a well-localized slice of $f(t)$, and then taking its FT as,

$$(F^{win} f)(\omega, t) = \int_{-\infty}^{\infty} f(s) g(s-t) e^{-j\omega s} ds \tag{3. 5a}$$

that gives the windowed Fourier transform of $f(t)$.

The reconstruction formula is [27],

$$f(s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (F^{win} f)(\omega, t) g(s-t) e^{j\omega s} d\omega dt \tag{3. 5b}$$

In order for the STFT to make sense, as well as for the reconstruction formula to be valid, it is necessary that g(s) is a square integrable function [22], i.e. $g \in L^2(R)$ (see Appendix 3 for vector spaces):

$$\|g(s)\|^2 = \int_{-\infty}^{\infty} |g(s)|^2 ds = 1 \qquad (3.6)$$

If t and ω are assigned regularly spaced values as $t = nt_0$ and $\omega = m\omega_0$, where $\{m,n\}$ range over the set of integers, $\mathbb{Z}$, and $\omega_0 t_0 > 0$ are fixed, then the discrete version of Equation 3.5a becomes

$$F_{m,n}^{win}(f) = \int_{-\infty}^{\infty} f(s)g(s - n\,t_0)e^{-jmw_0 s}\,ds \qquad (3.7)$$

Figure 3.5 shows the schematic representation of this procedure: for fixed *n*, $F_{m,n}^{win}(f)$ corresponds to the Fourier coefficients of $f(s)g(s-n\,t_0)$ in Equation 3.7. If the window function g is compactly supported, then with appropriately chosen ω_0, the Fourier coefficients are sufficient to characterize and to reconstruct $f(s)g(s-n\,t_0)$. Changing n amounts to shifting the slices by steps of t_0 and its multiples, allow the recovery of all of the f from $F_{m,n}^{win}(f)$.

In the Figure 3.5 the function $f(t)$ is multiplied with the window function $g(t)$, and the Fourier coefficients of the product $f(t)\,g(t)$ are computed; this procedure is then repeated for translated versions of the window, ($g(t-t_0)$, $g(t-2t_0)$, ...).

Consequently, the WFT localizes a signal simultaneously in time and frequency by looking at it through a window that is translated in time, and then translated in frequency (i.e., modulated in time).

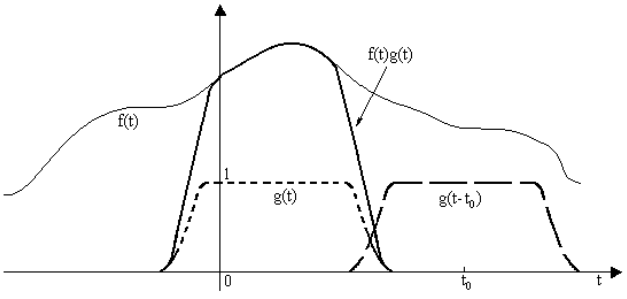


Figure 3.5 The Windowed Fourier Transform [27]

The FT tells whether a certain frequency component exists or not. This information is independent of where in time this component appears. FT gives perfect frequency resolution, but no time information. FT is not suitable if the signal has time varying frequency, i.e., the signal is non-stationary, as described in Section 3.1.2.

If the spectrum of a signal is time dependent, it is then necessary to use sufficiently short segments of it to compute the spectrum. In STFT, the signal is divided into small enough segments (windowing), where these segments of the signal can be assumed to be stationary. For this purpose, a window function $g(t)$ is chosen. The windowing procedure is represented in Figure 3.5.

The problem with the STFT is width of the window function that is used. (The width of the window function is known as the support of the window. If the window function is narrow, than it is known as compactly supported [27].) Choosing the window function is a hard task in STFT. Narrow windows give good time resolution, but poor frequency resolution. Wide windows give good frequency resolution, but poor time resolution; furthermore, wide windows may violate the condition of stationarity. The problem is a result of choosing a window function, once and for all, and using that window in the entire analysis [21].

3.3 Wavelet Analysis and Synthesis

If a real or complex value function $\Psi(t)$ satisfies the following two properties, then it is a mother wavelet or wavelet function [27]:

i. The function integrates to zero:

$$\int_{-\infty}^{\infty} \Psi(t)dt = 0 \tag{3. 8}$$

ii. It's square integrable, $\Psi(t) \in L^2(R)$, or has finite energy:

$$\int_{-\infty}^{\infty} |\Psi(t)|^2 dt < \infty \tag{3. 9}$$

Property 2 implies that most of the energy in $\Psi(t)$ is restricted to a finite duration. Property 1 is suggestive of a function that is oscillatory or that has a wavy appearance. Property 1 is the admissibility condition and forces the Wavelet function to have a band-pass nature. So the transform will be able to zoom in the singularities of the signal to be analyzed.

3.3.1 Continuous Wavelet Transform (CWT)

Let $f(t)$ be a square integrable function, denoted as $f(t) \in L^2(R)$. The CWT or continuous-time wavelet transform of $f(t)$ with respect to a wavelet $\Psi(t)$ is [26],

$$W(a,b) \equiv \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{|a|}} \Psi^* \left(\frac{t-b}{a} \right) dt \qquad (3.10)$$

where a and b are real and $*$ defines complex conjugate. Therefore, the wavelet transform is a function of two variables. Both $f(t)$ and $\Psi(t)$ belong to $L^2(R)$, the set of square integrable functions, also called the set of energy signals (by Cauchy in 1997).

In Equation 3.10 the basis (or mother) function $\Psi(t)$ is defined as

$$\Psi_{a,b}(t) \equiv \frac{1}{\sqrt{|a|}} \Psi \left(\frac{t-b}{a} \right) \qquad (3.11)$$

Then, combining Equations 3.10 and 3.11 yields

$$W(a,b) = \int_{-\infty}^{\infty} f(t) \Psi_{a,b}^*(t) dt \qquad (3.12)$$

In Equation 3.11 when *a=1* and *b=0*, $\Psi_{1,0}(t) = \Psi(t)$.

The normalizing factor of $1/\sqrt{|a|}$ ensures that the energy stays the same for all *a* and *b*; that is,

$$\int_{-\infty}^{\infty} |\Psi_{a,b}(t)|^2 dt = \int_{-\infty}^{\infty} |\Psi(t)|^2 dt \qquad (3.\ 13)$$

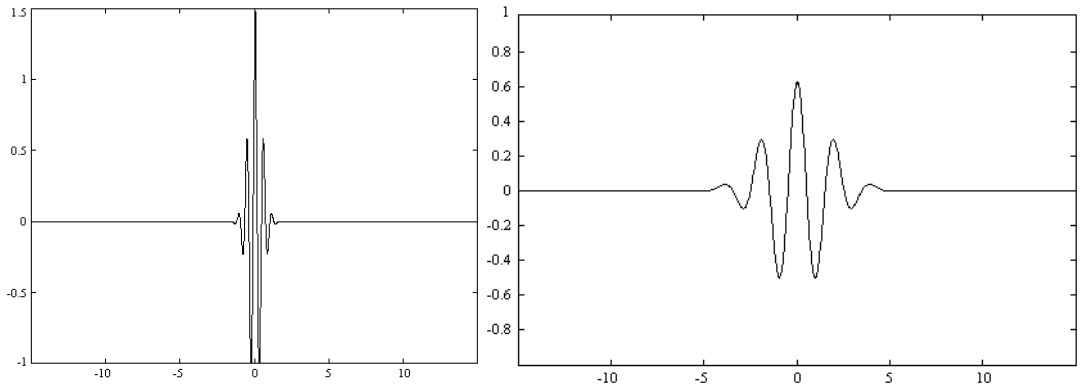
for all a and b. For any given value of a, the function $\Psi_{a,b}(t)$ is a shift of $\Psi_{a,0}(t)$ by an amount b, along the time axis. So, the variable b represents time shift or translation. $\Psi_{a,0}(t)$, that is a time-scaled and amplitude-scaled version of $\Psi(t)$ is denoted as

$$\Psi_{a.0}(t) = \frac{1}{\sqrt{|a|}} \Psi\left(\frac{t}{a}\right) \qquad (3.\ 14)$$

Due to determining the amount of time scaling or dilation, the variable a is referred to as the scale or dilation variable. As an illustration, two dilations of the Morlet (modulated Gaussian) wavelet are shown in Figure 3.6.

A Morlet wavelet is constructed by modulating a sinusoidal function by a Gaussian function [27]. A Morlet wavelet has infinite duration and it is a member of wavelets that are not supported compactly. However, most of the energy in Morlet wavelet is limited to a finite interval. The real value of Morlet wavelet is [26]

$$\Psi(t) = e^{-t^2} \cos\left(\pi \sqrt{\frac{2}{\ln 2}} t\right) \qquad (3.\ 15)$$



(a) (b)

Figure 3.6 A Morlet wavelet dilated by factors of (a) *a = 1/2* and (b) *a = 3* [R1]

Figure 3.6a shows the illustration of the real value of Morlet wavelet. More than 99% of the total energy of the function is confined to an interval $|t| \leq 2.5$ sec.

If *a>1*, there is a stretching of $\Psi(t)$ along the time axis, while if *0<a<1*, there is a contraction of $\Psi(t)$. Since CWT is generated using dilates and translates of the single function; $\Psi(t)$, the wavelet for the transform, is referred to as the mother wavelet.

The effect of dilation on time and frequency resolution is examined above. The CWT also involves translation of the wavelet. The translation parameter *b* affects the location of the wavelet, whereas the duration or bandwidth is affected by the dilation "*a*" [26]. If the $\Psi(t)$ is such that the Equation 3.10 is invertible, then

$$f(t) = \frac{1}{C_\Psi} \int_{a=-\infty}^{\infty} \int_{b=-\infty}^{\infty} W(a,b)\Psi_{a,b}(t) \frac{1}{|a|^2} dadb \tag{3. 16}$$

is the inverse CWT that gives a mapping from the set of W(a,b) back to $L^2(R)$.

C_Ψ is a constant that depends on $\Psi(t)$. The constant has value

$$C_\Psi = \int_{-\infty}^{\infty} \frac{|\Psi(w)|^2}{\omega} d\omega < \infty \tag{3. 17}$$

and is such that $0< C_\Psi <\infty$ which in turn imposes an admissibility condition on $\Psi(t)$. For $C_\Psi <\infty$, $\Psi(t)$ must be such that

$$|\Psi(w)| < \infty, \text{ for any } \omega \tag{3. 18}$$

and $\Psi(0) = 0$, implying that

$$\int \Psi(t)dt = 0 \tag{3. 19}$$

means $\Psi(t)$ cannot have nonzero 'dc' (average).

The admissibility condition stated here is a sufficient condition that leads to the expression in Equation 3.16. It is not a necessary condition to obtain a mapping from the set of W(a,b) back to $L^2(R)$ [26].

Another important point about the CWT is that, W(a,b) can be thought of as the inner product of the signal with the basis functions (see Appendix 2 for information on inner product of a function). Therefore, the definition of the W(a,b) in Equation 3.10, shows that the wavelet analysis is a measure of similarity between the basis functions (wavelets) and the signal. (Here the similarity is in the sense of similar frequency content.) The calculated CWT coefficients refer to the closeness of the signal to the wavelet at the current scale.

This case points on the correlation of the signal with the wavelet at a certain scale. If the signal has a major component of the frequency corresponding to the current scale, then the wavelet (the basis function) at the current scale will be similar or close to the signal at the particular location where this frequency component occurs. Therefore, the CWT coefficient computed at this point in the time-scale plane will be a relatively large number.

3.3.2 Discretization of Time-Scale Parameters

Both the STFT and WT are highly redundant when the frequency-time parameters *(Ω,T)* and scale-translation parameters *(a,b)* are continuous. In Equation (3.10), both *(a,b)* are continuous variables and there is a redundancy in the CWT representation of $f(t)$. There is certainly no need to compute W(a,b) for all possible *(a,b)*. Additionally, it is of practice necessity that *(a,b)* take on only a finite number of values. The idea is, if the sampling of *(a,b)* is sufficiently dense then $f(t)$ can be recovered from W(a,b) with *(a,b)* discrete, since there is already a parallelism in the perfect recovery of a signal from its samples taken at or above the Nyquist rate. The transforms are usually evaluated on a discrete grid on the time-frequency and time-scale plane, respectively, corresponding to a discrete set of continuous basis functions. There are various ways of discretizing time-scale parameters *(a,b)*. Since two scales $a_0 < a_1$ roughly correspond to two frequencies $f_0 > f_1$, the wavelet coefficients at scale a_1 can be sub-sampled at $(f_0 / f_1)^{th}$ rate of the coefficients at

scale a_0, according to Nyquist's rule. Therefore the time-scale parameters chosen to discretized on the sampling grid drawn in Figure 3. 7. Depending on the type of $\Psi(t)$ and the sampling grid of *(a,b)*, sometimes duals (see Appendix 5 for information) are required for perfect reconstruction [21].

A special case occurs when *(a,b)* are samples of a dyadic grid, when certain $\Psi(t)$ produce orthonormal basis functions $\Psi((t-b)/a)/\sqrt{a}$, with *(a,b)* discrete. Therefore, $f(t)$ can be exactly synthesized as a weighted sum of these orthonormal basis functions [21].

When *(a,b)* is discrete and given by

$$a = a_0^m \ , \ b = nb_0a_0^m \ \text{ m, n integer,} \tag{3. 20}$$

the discrete parameter wavelet transform (wavelet coefficients) is

$$DPWT(m,n) = \int f(t)\Psi_{mn}(t)dt \tag{3. 21}$$

where

$$\Psi_{mn}(t) = a_0^{-\frac{m}{2}}\Psi(a_0^{-m}t - nb_0) \tag{3. 22}$$

and a_0 and b_0 are constants that determine the sampling intervals. The translation step *b* depends on the dilation step *a*, since long wavelets are advanced by large steps, and short wavelets are advanced by small steps. Both $f(t)$ and $\Psi(t)$ are continuous functions of time [21].

For the sake of computational efficiency the constants a_0 and b_0 are taken as a_0=2 and b_0=1 ($a = 2^m$, $b = n2^m$) so that

$$\Psi_{mn}(t) = 2^{-\frac{m}{2}}\Psi(2^{-m}t - n) \tag{3. 23}$$

for $\Psi \in L^2(R)$, and $m,n \in \mathbb{Z}$. With this octave time scaling and dyadic translation, the sampled values of *(a,b)* are as shown in the dyadic grid of Figure 3.7.

Since the FT of $\Psi(at)/\sqrt{a}$ is $\Psi(w/a)/a\sqrt{a}$, the center frequency and bandwidth of a wavelet are both scaled by 1/*a* for a time scaling of *a*. So for all mother wavelets,

$$\frac{centre\_frequency}{bandwidth} = \frac{f}{\Delta f} = Q = \text{constant} \qquad (3.\ 24)$$

giving rise to the so-called "Constant-*Q*" or "Constant Relative-bandwidth" analysis capability of wavelets [21].

The basic difference between STFT and WT is that STFT, which gives time-frequency representation, uses a single analysis window with constant bandwidth. However in WT, wavelet functions (daughter or baby wavelets) are localized in time and frequency domains and are generated from a single kernel function (mother wavelet) by dilation and translation. So WT uses short windows at high frequencies and long windows at low frequencies. In STFT, if the analyzing functions are not wide enough, they are unable to capture the low frequency information and wider they get, they loose short time duration changes in the signal. So it is felt that the analyzing functions should have a constant center frequency to bandwidth ratio (constant-Q). The mother wavelet can be thought of as a band-pass filter. The bandpass filters have constant relative bandwidth or constant-Q property.

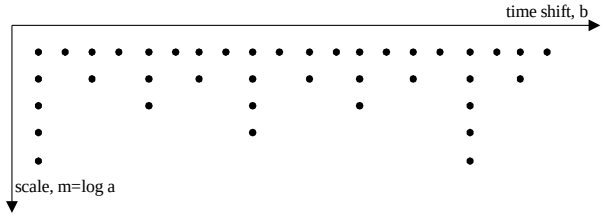


Figure 3.7 Dyadic sampling grid in the time-scale plane. Each node corresponds to a wavelet basis function with scale 2^{-m} and shift b= $n2^{-m}$ [28]

To overcome the resolution limitation of the STFT, it is possible to derive time and frequency resolutions vary in the time-frequency plane in order to obtain a multi-resolution analysis. When the analysis is viewed as a filter bank, then Δf must increase with the central frequency of the analysis filters. This yields to Equation

3.24. The analysis filter bank is then composed of band-pass filters with constant relative bandwidth. Therefore, instead of the frequency responses of the analysis filter to be regularly spaced over the frequency axis (as for the STFT) case, they are regularly spread in a logarithmic scale (see Figure 3.4).

When Equation 3.24 is satisfied, Δf and also ΔT changes with the center frequency of the analysis filter. (they still satisfy the Heisenberg inequality given in Section 3.1.3.2). But now, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies.

The reconstruction formula for $f(t)$ is [28]

$$f(t) = c_{\Psi} \sum_m \sum_n DPWT(m,n) \cdot \psi_{m,n}(t) \qquad (3.25)$$

where c_{Ψ} is a constant that does not depend on the signal. Evidently, if a_0 is close enough to 1 and if b_0 is small enough, then the wavelet functions are defined to be over-complete. Equation 3.25 is then still very close to Equation 3.16 and signal reconstruction takes place within non-restrictive conditions on $\Psi(t)$. On the other hand, if the sampling is sparse, e.g. the computation is done octave by octave (a_0=2), a true orthonormal basis will be obtained only for very special choices of $\Psi(t)$ [28].

Two examples of orthonormal discrete parameters (dyadic sampling) can be given as Haar wavelets and Shannon wavelets. The Haar wavelet has good time localization but poor frequency localization. Its spectrum is non-zero for $\omega \to \infty$. It does not have compact support in the frequency domain. In contrast, the Shannon wavelet has non-compact support in time and decays only as fast as $1/t$, so it has poor time localization. Its frequency localization is good because it has the spectrum of an ideal band-pass filter. There are orthonormal wavelets that are between these two given types, giving both acceptable localizations in time and frequency [21].

There are several benefits of the orthonormal basis in dyadic-orthonormal wavelet transform which is obtained by a_0=2 and b_0=1. The first is that there will be no

information redundancy among the decomposed signals due to the orthonormal properties. The second is that with this choice of a_0 and b_0 there exists an elegant algorithm, which is known as the multi-resolution signal decomposition technique, to decompose a signal into scales with different time and frequency resolution [28].

3.3.3 Wavelet Frames

The theory of wavelet frames [see Appendix 5 for information about frames in vector spaces] permits one to balance redundancy, i.e. sampling frequency in Figure 3.7.

The energy of the wavelet coefficients; *DPWT(m,n)* relative to the signal $f(t)$, in a Hilbert space, lies between two positive frame bounds A and B

$$A.E_f \leq \sum_{m,n} \left| DPWT(m,n) \right|^2 \leq B.E_f \qquad (3.\ 26)$$

with $0 < A < B < \infty$, where E_f is the energy of the signal $f(t)$ [24]. Equation 3.26 is known as Parseval relation. The Wavelet function satisfying Parseval relation constitutes a frame. Once the Parseval and the admissibility conditions are satisfied, the transform is complete; the signal can be reconstructed from transformation coefficients.

3.4 Multi-resolution Analysis, Discrete WT and Digital Filters

A wavelet is a band-pass filter from a signal processing point of view [28]. In the dyadic case given by Equation 3.23, it forms an octave band filter. So, the wavelet transform can be interpreted as constant-Q filtering (given by Equation 3.24) with a set of octave-band filters followed by sampling at the respective Nyquist frequencies. Therefore, by adding higher octave bands, details or resolution is added to the signal.

In the discrete time case, two methods were developed independently which lead naturally to discrete wavelet transforms. They are sub-band coding and pyramidal coding. In this section sub-band coding, pyramidal coding and discrete wavelet transform (DWT) will be described respectively.

An important point in DWT is the "scale" and the "resolution" parameters. Scale is related to the size of the signal, while resolution is related to the amount of detail present in the signal. For large scales, dilated wavelets take "global views" of a sub-sampled signal, while for small scales, contracted wavelets analyze small details in the signal. The resolution and scale change in discrete time is given in Figure 3.8.

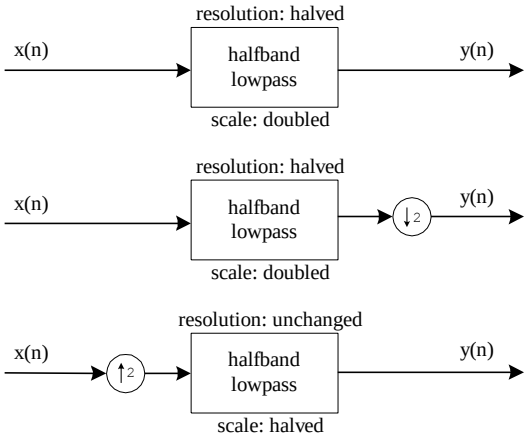


Figure 3.8 Resolution and scale changes in discrete time [28]

3.4.1 The Pyramid Algorithm

Given an original sequence $x(n)$, $n \in \mathbb{Z}$, derive a lower resolution signal by low-pass filtering with a half-band low-pass filter having impulse response $g(n)$. Following the Nyquist's rule, it is possible to down-sample by two (see Appendix 7 for down-sampling operation), hence doubling the scale in the analysis. This results in a signal $y(n)$ given by

$$y(n) = \sum_{r=-\infty}^{\infty} g(r)x(2n-r) \tag{3. 27}$$

The resolution change is obtained by the low-pass filter (loss of high frequency detail). The scale change is due to sub-sampling by two, since a shift by two in the original signal $x(n)$ results in a shift by one in $y(n)$.

Based on this low-pass and sub-sampled version of $x(n)$, an approximation to the original signal will be found. This is done by first up-sampling $y(n)$ by two (see Appendix 8 for up-sampling operation):

$$y'(2n) = y(n), \text{ and } y'(2n+1) = 0 \tag{3. 28}$$

Then, $y'(n)$ is interpolated with a filter with impulse response $g'(n)$ to obtain the approximation $a(n)$:

$$a(n) = \sum_{r=-\infty}^{\infty} g'(r)y'(n-r) \tag{3. 29}$$

If $g(n)$ and $g'(n)$ were "perfect" half-band filters, then the Fourier transform of $a(n)$ would be equal to the Fourier transform of $x(n)$. That is, $a(n)$ would be a perfect half-band LP approximation to $x(n)$.

In general case, $a(n)$ is not equal to $x(n)$. So, the difference between $a(n)$ and $x(n)$ is as follows;

$$d(n) = x(n) - a(n) \tag{3. 30}$$

$x(n)$ can be reconstructed by adding $d(n)$ and $a(n)$. As it can be seen in Figure 3.9, derivation of a low-pass, sub-sampled approximation $y(n)$, from which an approximation $a(n)$ to $x(n)$ is derived by up-sampling and interpolation. Then, the difference between $a(n)$ and $x(n)$ is computed as $d(n)$. Perfect reconstruction is obtained by adding $x(n)$ back. However, there is some sort of redundancy, because a signal with sampling rate f_S is mapped into two signals $d(n)$ and $y(n)$ with sampling rates f_S and $f_S/2$, respectively. In the case of a perfect half-band LP filter, $d(n)$ contains exactly the frequencies above $\pi/2$ of $x(n)$, and therefore $d(n)$, can be sub-sampled by two as well without loss of information [24].

The separation of the original signal $x(n)$ into a coarse approximation $a(n)$ plus some additional detail $d(n)$ is important. Because of the resolution change involved

(low-pass filtering followed by sub-sampling by two produces a signal with half the resolution and at twice the scale of the original), this method is part of the multi-resolution signal analysis. The scheme can be iterated on $y(n)$, creating a hierarchy of lower resolution signals at lower scales. Because of that hierarchy and the fact that signals become shorter, this scheme is called signal pyramids.

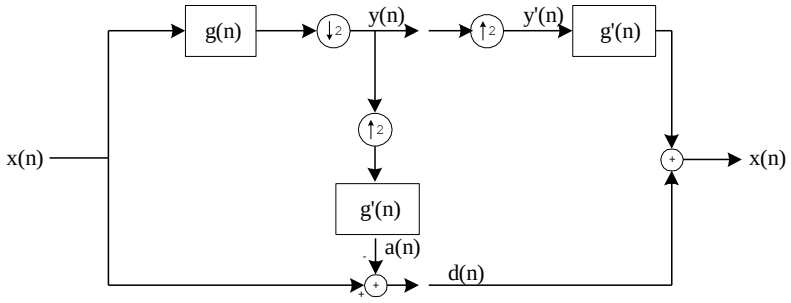


Figure 3.9 The pyramid scheme [24]

The pyramid scheme creates a redundant set of samples. One stage of a pyramid decomposition leads to both a half rate low-resolution signal and a full rate difference signal, resulting in an increase, in the number of filters by fifty percent. The redundancy problem (oversampling) can be avoided if the samples $g(n)$ and $g'(n)$ meet certain conditions [28].

3.4.2 Sub-band Coding

A different scheme is sub-band coding where no such redundancy appears as there is in the pyramidal scheme. The LP, sub-sampled approximation is obtained exactly as in the case of multi-resolution pyramid scheme; but instead of a difference signal, the added detail as a high-pass filtered version of $x(n)$, followed by sub-sampling by two, is computed. So, the original signal is mapped into a LP approximation and an added detail signal. The added detail to the LP approximation has to be a HP signal, and if $g(n)$ is an ideal half-band LP filter, then an ideal half-band HP filter $h(n)$ will lead to a perfect representation of the signal into two sub-sampled versions.

An important point is that it is not necessary to use ideal filters, and $x(n)$ can be recovered from its two filtered and sub-sampled versions. (Lets call them $y_0(n)$ and $y_1(n)$ respectively.) To do so, both are upsampled and filtered by $h'(n)$ and $g'(n)$

respectively, and finally added together as shown in Figure 3.10. Unlike the pyramid scheme the reconstructed signal $\hat{x}(n)$ is not identical to $x(n)$ unless the filters meet some specific constraints. Filters that meet these constraints have 'perfect reconstruction' (PR) property. More information related to this topic can be found in [28].

In the Figure 3.10 the filters $h(n)$ and $g(n)$ are FIR filters. And the relation between the HP and LP filters in this concept is given by,

$$h(L-1-n) = (-1)^n g(n) \tag{3. 31}$$

where L is the filter length (which is even). The modulation by $(-1)^n$ transforms indeed the LP filter into a HP one [24].

The filter bank in Figure 3.10, computes convolutions followed by sub-sampling by two, evaluate inner products of the sequence $x(n)$ and the sequences $\{g(-n+2r), h(-n+2r)\}$. Hence

$$y_0(r) = \sum_n x(n) g(-n+2r) \tag{3. 32a}$$

$$y_1(r) = \sum_n x(n) h(-n+2r) \tag{3.32b}$$

Since the filter responses form an orthonormal set (see Appendix 4 for information about orthogonality and orthonormality), it is very simple to reconstruct $x(n)$ as

$$x(n) = \sum_{r=-\infty}^{\infty} \left[y_o(r) g(-n+2r) + y_1(r) h(-n+2r) \right] \tag{3. 33}$$

that is, as a weighted sum of the orthogonal impulse responses, where the weights are the inner products of the signal with the impulse responses.

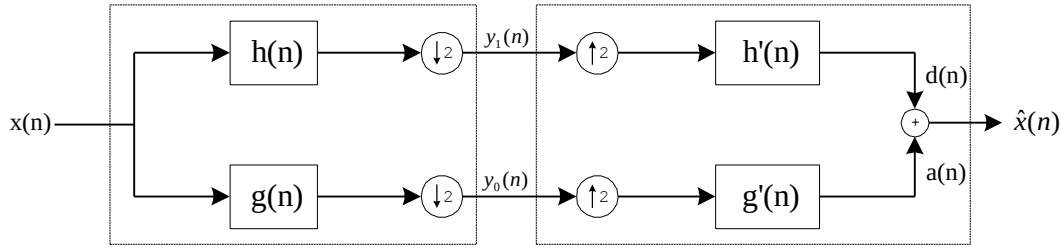


Figure 3.10 Sub-band coding scheme [24]

In the Figure 3.10 two sampled approximations; one corresponding to low and latter to high frequencies are computed. Re-interpolating the approximations and summing them obtain the reconstruction signal. The filters on the left form an analysis filter bank, whereas the ones on the right is a synthesis filter bank.

3.4.3 The Discrete Wavelet Transform

Decomposition of a sequence $x(n)$ into two subsequences at half rate, or half resolution, by means of orthogonal filters is defined in Section 3.4.2. This process can be iterated on either or both subsequences [24]. In particular to achieve finer frequency resolution at lower frequencies, the scheme on the lower band is iterated only. If $g(n)$ is a good half-band LP filter, $h(n)$ will also be a good half-band HP according to Equation 3.31. So, one iteration of the scheme on the first low-band creates a new low-band that corresponds to the lower quarter of the frequency spectrum. Each further iteration halves the width of the low-band, but due to sub-sampling by two, its time resolution is halved as well. At each iteration, the current high band portion corresponds to the difference between the previous low-band portion and the current one, which is a pass-band [24]. The procedure is represented in Figure 3.11, and the frequency resolution is as in the Figure 3.4b.

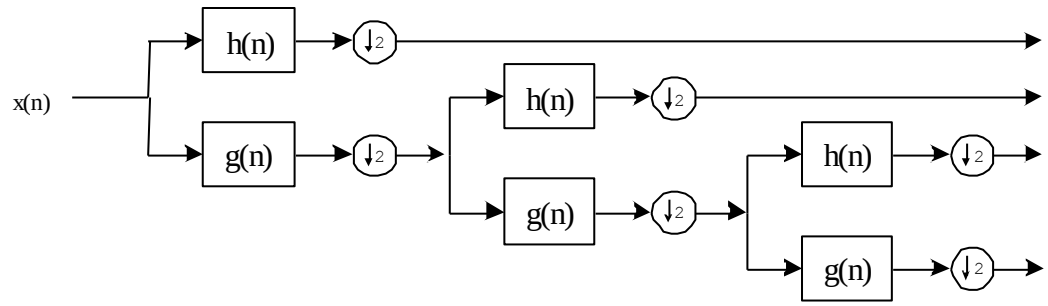


Figure 3.11 Block diagram (Filter bank tree) of the DWT implemented with discrete time filters and sub-sampling by two. The frequency resolution is given by Figure 3.4b [24]

There is an important difference between the discrete scheme and the continuous time WT. In the discrete time case, the role of the wavelet is played by the HP filter $h(n)$ and the cascade of sub-sampled LP filters followed by a HP filter. These filters, which correspond to octave band filters, unlike in the CWT, are not exact scaled versions of each other. In particular, because of being in discrete time, scaling is not as easily defined: it involves interpolation as well as time expansion. However, under certain conditions, the discrete system converges to a system where subsequent filters are scaled versions of each other. Actually, this convergence is the basis for construction of continuous time compactly supported wavelet bases [27].

The equivalent filter or the iterated low-pass filter corresponding to the lower branch in Figure 3.11, will be described in the following paragraphs.

The z-transform of the half-band filter in the lower branch of Figure 3.11 is,

$$G(z) = \sum_n g(n) z^{-n} \tag{3. 34}$$

Sub-sampling by two followed by filtering with *G(z)* is equivalent to filtering with $G(z^2)$. So, the first two steps of low-pass filtering can be replaced by a filter with z-transform $G(z).G(z^2)$, followed by sub-sampling by four. Calling $G^i(z)$ the equivalent filter to i stages of low-pass filtering and sub-sampling by two (a total sub-sampling by 2^i),

$$G^i(z) = \prod_{I=0}^{i-1} G(z^{2^I}) \tag{3. 35}$$

with an impulse response of $g^i(n)$ is obtained. As i infinitely increase, this filter becomes infinitely long. Instead, consider a function $f^i(x)$, which is piecewise constant on intervals of length $1/2^i$ and has value $2^{i/2} g^i(n)$ in the interval $[n/2^i, (n+1)/2^i]$. That is, $f^i(x)$ is a staircase function with the value given by the samples of g and intervals, which decrease as 2^i. It can be verified that the function is supported (see Appendix 3 for "support" property) on the interval [*0,L-1*], where L is the length of the filter *g(n)*, For i going to infinity, $f^i(x)$ can converge to a

continuous function $g_c(x)$, or a function with finitely many continuities, or not converge at all. A necessary condition for the iterated functions to converge to a continuous limit is that the filter *G(z)* should have a sufficient number of zeros at *z= -1*, or half sampling frequency. Using this condition, one can construct filters, which are both orthogonal and converge to continuous functions with compact support. (See Appendix 3 for "compact-support" property) Such filters are called regular filters. The above condition can be interpreted as a flatness condition on the spectrum of *G(z)* at half sampling frequency. Wavelet filters are chosen so as to be regular. Moreover the Daubechies orthonormal filters are deduced from "maximally flat" low-pass filters [24].

The $g_c(x)$ is the final (limit) function to which $f^i(x)$ converges. Because it is the product of low-pass filters, the final function is itself low-pass and is called a "scaling function" (also shown as $\Phi(x)$) because it is used to go from a fine scale to a coarser scale. Because of the product in Equation 3.35, from which the scaling function is derived, $g_c(x)$ satisfies the following two-scale difference equation [24]:

$$g_c(x) = \sum_{n=-\infty}^{\infty} g(n) g_c(2x - n) \tag{3. 36}$$

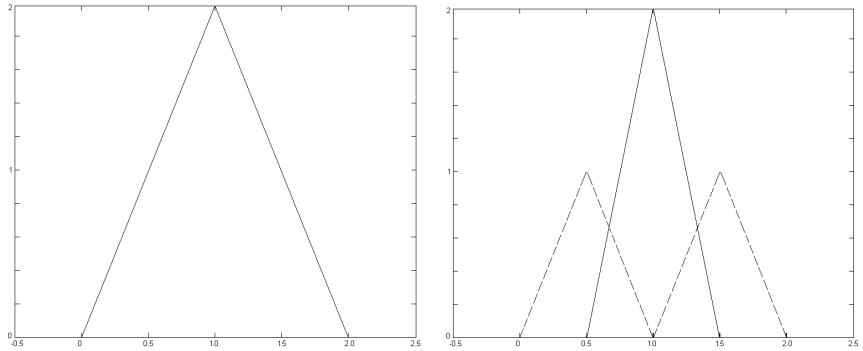
Figure 3.12 shows scaling functions that satisfy two-scale difference equations. It shows how a scaling function can be obtained from a linear combination of its scaled versions.

Equation 3.36, gives the scaling function of the iterated low-pass filter. From Figure 3.11 it is shown that also a band-pass filter is obtained in the same way as the low-pass filter, except for a final high-pass filter. Hence, the wavelet function $h_c(x)$ (also shown as $\Psi(x)$) is obtained as

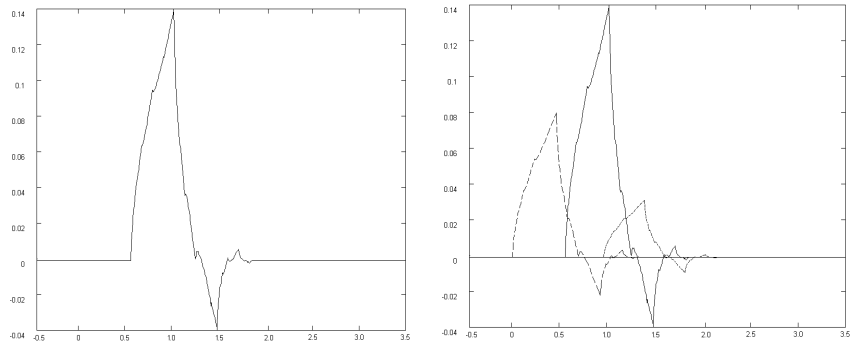
$$h_c(x) = \sum_{n=-\infty}^{\infty} h(n) g_c(2x - n) \tag{3. 37}$$

that also satisfies a two scale equation. If the filters *h(n)* and *g(n)* form an orthonormal set with respect to even shifts, then the functions $g_c(x - l)$ and

$h_c(x-r)$ form an orthonormal set. Because they also satisfy two scale difference equations, the set forms an orthonormal basis for the set of square integrable functions $L^2(Z)$ [28].



(a)



(b)

Figure 3.12 Scaling function $\Phi(x)$ as a linear combination of scaled and shifted versions $\Phi(2x-n)$. (a) Hat function (not an orthonormal example), (b) the Daub-4 wavelet obtained from a regular orthogonal filter [28]

As mentioned above, wavelet filters are chosen so as to be regular. This means that the piecewise constant function associated with the discrete wavelet sequence $h_j(n)$ of z-transform $G^j(z)H(z^{2^j})$ converges, as j indefinitely increases, to a regular limit function $h_c(x)$. Equivalently, the piecewise constant function associated with the discrete "scaling" sequence g(n) of a z-transform $G^j(z)$ converges to a regular limit function $g_c(x)$. "Regular" means that the continuous-time wavelet $h_c(x)$ (or the scaling function $g_c(x)$) is at least continuous, or better, once or twice continuously differentiable [28].

3.4.4 The Multi-resolution Analysis (MRA)

The multi-resolution analysis of functions was introduced by Meyer and Mallat [29], and provides a better view of understanding the wavelet decomposition.

The MRA is a tool that utilizes the DWT to represent a time-varying signal in terms of its frequency components. It essentially maps a one-dimensional (1D) signal of time into a two-dimensional (2D) signal of scale and time. The goal of MRA is to develop representations of a complicated signal $f(t)$ in terms of its orthonormal basis, which are the scaling and the wavelet functions. These functions can be scaled and translated to decompose $f(t)$ and represent it at different resolutions (scales). So, using MRA, the time domain signal $f(t)$ can be mapped into the wavelet domain and represented at different resolution levels in terms of wavelet coefficients.

Assume that there is a ladder of spaces such that:

$$\Lambda\, V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2}\Lambda \tag{3. 38}$$

with the property that if $f(x) \in V_i$ then $f(x-2^{-i}r) \in V_i, r \in Z$, and $f(2x) \in V_{i-1}$. Let's call W_i the orthogonal complement of V_i in V_{i-1}. This is written as

$$V_{i-1} = V_i \oplus W_i \tag{3. 39}$$

Thus, W_i contains the "detail" necessary to go from V_i to V_{i-1}. Iterating Equation 3.39,

$$V_{i-1} = W_i \oplus W_{i+1} \oplus W_{i+2} \oplus W_{i+3} \oplus \Lambda \tag{3. 40}$$

is obtained, that is, a given resolution can be attained by a sum of added details.

Assume that there is an orthonormal basis for V_0 made up of a function $g_c(x)$ and its integer translates. Since $V_0 \in V_{-1}$, $g_c(x)$ can be written in terms of the basis in V_{i-1}, Equation 3.36 is satisfied:

$$g_c(x) = \sum_n c_n \cdot g_c(2x - n) \tag{3. 41}$$

Then it can be verified that the function $h_c(x)$ in Equation 3.37, with the relation in Equation 3.31, and its integer translates form an orthonormal basis for W_0. Because of Equation 3.40, $h_c(x)$ and its scaled and translated versions form a wavelet basis [28].

Assume that there is an approximation of a signal at a resolution corresponding to V_0. Then a better approximation is obtained by adding the details corresponding to W_0. This amounts to a weighted sum of wavelets at that scale. Hence, by iterating this idea, a square integrable signal can be seen as the successive approximation or a weighted sum of wavelets at finer scale.

In this section, it is shown that the Short Time Fourier Transform and the Wavelet Transform represent alternative ways to divide the time-frequency and time-scale planes respectively. The main advantages of Wavelet Transform are that it can zoom into time discontinuities, and that orthonormal bases localized in time and frequency can be constructed. WT provides looking at a signal at various scales and analyzing it with various resolutions. For large scales, dilated wavelets take global views of a sub-sampled signal, while for small scales, contracted wavelets analyze small details in the signal. In the discrete case, the WT is equivalent to a logarithmic filter bank, with the added constraint of regularity on the low-pass filter.

One of the best-developed application areas of Wavelets is in signal compression. Discrete Wavelet Transforms have essential sub-band coding systems and sub-band coders are successful in speech and image compression [28].

4. ARTIFICIAL NEURAL NETWORKS

4.1 Introduction

Artificial Neural Networks (ANN) represent an engineering discipline concerned with non-programmed adaptive information processing systems that develop associations (transforms or mappings) between objects in response to their environment. That is, they learn from examples. ANNs are a type of massively parallel computing architecture, based on brain-like information encoding and processing models. They can exhibit brain-like behaviors such as learning, association, categorization, generalization, feature extraction, and optimization.

A more formal definition of an ANN according to Haykin [5] is:

"A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

i. Knowledge is acquired by the network through a learning process.

ii. Inter-neuron connection strengths known as synaptic weights are used to store knowledge."

The key element of the ANNs is the novel structure of the information processing system. An ANN is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. The ANN achieves its ability to learn and then recall that learning through the weighted interconnections of those processing elements. Given noisy sensory inputs, they build up their internal computational structures through experience rather than pre-programming according to a known algorithm. The interconnection architecture can be very different for different networks. Architectures can vary from feed-forward, and recurrent structures to latticed structures.

4.1.1 The Biological Neural Network

The basic computational unit in the nervous system is the nerve cell, or neuron. The human brain is composed of a very large number (about a hundred billion) of neurons, massively interconnected (with an average of several thousand interconnects per neuron). The structure of a neuron is represented in Figure 4.1.

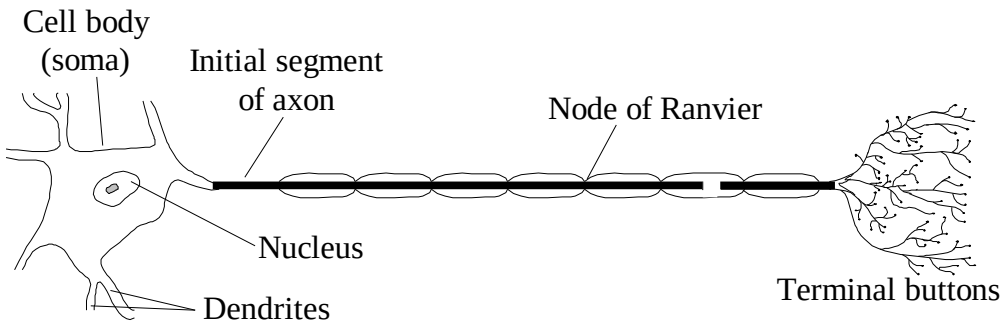


Figure 4.1 The basic structure of a neuron

A neuron is made up of a nucleus, a cell body (soma), dendrites (inputs) and an axon (output). The cell body is a place for the mechanisms that provide the cell its energy and cause the activation. The neuron collects signals from others through a host of fine structures called dendrites. Once the total input signal received at the cell body from the dendrites exceeds a certain level (the firing threshold), the neuron activates and fires an electrochemical signal through a long, thin structure known as an *axon*, which splits into thousands of branches. This event is also called depolarization, and is followed by a refractory period, during which the neuron is unable to fire. The action potential is generally a one-millisecond electrical pulse of 0.1 mV amplitude. The axon endings almost touch the dendrites or cell body of the next neuron. At the junction of the signal-sending axon and the signal-receiving dendrite lies a small gap called a synapse. When the action potential reaches a synapse at the end of the axon, the electrical signal is converted to a chemical signal to be communicated across the synaptic gap to the post synaptic neuron. At the membrane of the postsynaptic neuron, the chemical signal is converted back to an electrical signal to be conveyed along the dendrite to the soma. A synapse can either be excitory or inhibitory. Input from an excitory synapse increases the internal activation level of the neuron while input from an inhibitory synapse reduces it [30]. The neurons learn to react to certain

signals; the synaptic connections between neurons either get stronger or weaker. The strength of the synaptic connection determines how strong the receiving neuron finds the signal. The signals from different neurons are thus weighted differently based on the strength of the synaptic connections. If the total effect of all the received signals is adequate, the neuron is activated and it will begin to send a signal to the other neurons via its axon.

4.1.2 Historical Background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

The modern age of ANNs began by the classic paper of McCulloch and Pitts in 1943. They described a logical calculus of neural networks. Their formal model of a neuron was assumed to follow an "all-or-none" law. McCulloch and Pitts showed that with a sufficient number of such simple units, and synaptic connections set properly and synchronously, a network can compute any computable function [5].

In 1948, Wiener's famous book Cybernetics, describing the concepts for control, communication, and statistical signal processing, was published. The second edition of this book published in 1961, adds a new material on learning and self-organization.

In 1949, Hebb's book The Organization of Behavior, which presents an explicit statement of a physiological learning rule for synaptic modification, was published. Hebb proposed that the connectivity of the brain is continually changing as an organism learns differing functional tasks.

In 1952, Ashby's book, Design for a Brain: The Origin of Adaptive Behavior was published, which was concerned with the basic notion that, adaptive behavior is not inborn but rather learned, and by learning the behavior of a system usually changes for the better.

In 1954, Gabor proposed the idea of a nonlinear adaptive filter. He went on to build such a machine that learning was accomplished by feeding samples of a stochastic

process into the machine, together with the target function that the machine was expected to produce.

In 1956, Taylor initiated the work on associative memory. This was followed by the introduction of the learning matrix by Steinbuch in 1961. This matrix consists of a planar network of switches interposed between arrays of sensory receptors and motor effectors.

In 1957, Bellmann reported his work on the dynamic programming, which provides the mathematical formalism for sequential decision-making.

In 1958, Rosenblatt introduced a new approach to the pattern recognition problem in his work on the perceptron, a novel method of supervised learning. Frank Rosenblatt used the perceptron to solve some image recognition problems [30].

In 1960, Widrow and Hoff introduced the least mean-square (LMS) algorithm and used it to formulate the Adaline (adaptive linear element). The difference between the Adaline and the perceptron lies in the training procedure. One of the earliest trainable-layered neural networks with multiple adaptive elements was the Madaline (multiple adaline) structure proposed by Widrow in 1962.

In 1965, Nillson's book, Learning Machines, was published, which concerns the linearly separable patterns in hyper surfaces. In 1967, Amari used the stochastic gradient method for adaptive pattern classification [31].

In 1969, Minsky and Papert published their book, which concerns the use of mathematics to demonstrate that there are fundamental limits on what single layer perceptrons can compute. They also stated that, there was no reason to assume that any of the limitations of single layer perceptrons could be overcome in the multilayer version.

An important activity that did emerge in 1973 was self-organizing maps using competitive learning. Von Der Malsburg demonstrated self-organization with some computer simulations. In 1976, Willshaw and Von Der Malsburg published the first paper on the formation of self organizing maps, motivated by topologically ordered maps in the brain.

In 1980, Grossberg established a new principle of self-organization known as ART (Adaptive Resonance Theory). The theory involves a bottom-up recognition layer and a top-down generative layer. If the input pattern and learned feedback pattern match, a dynamical state called adaptive resonance takes place. [5]

In 1982, Hopfield used the idea of an energy function to formulate a new way of understanding the computation performed by recurrent networks with symmetric synaptic connections. He established the connection between neural networks and physical systems of the type considered in statistical mechanics.

Another important development in 1982 was the publication of Kohonen's paper on self-organizing maps using a one or two dimensional lattice structure, which was different from the earlier work by Willshaw and Von Der Malsburg. Many applications have been developed since Kohonen first proposed the Kohonen networks. Kohonen networks have been applied in the field of combinatorics, for example, to solve the Traveling Salesman Problem with the elastic net algorithm. Teuvo Kohonen has carried out some research on the application of topology preserving networks in such diverse fields as speech recognition and structuring of semantic networks [30].

In 1983, Cohen and Grossberg established a general principle for assessing the stability of a content-addressable memory that includes the continuous time version of the Hopfield network. Also in 1983, Kirkpatrick, Gelatt, and Vecchi described a new procedure called Simulated Annealing, rooted in statistical mechanics, for solving combinatorial optimization problems. The simulated annealing has been used for many years in the field of numerical optimization. The technique is a special case of the Monte Carlo method [30].

In 1985, Ackley, Hinton, and Sejnowski developed the Bolztmann Machine, which was the first successful realization of a multilayer neural network. Their learning algorithm was the first proposed for stochastic networks and allowed dealing with hidden units in the networks of Hopfield type [5].

In 1986, Rumelhart, Hinton, and Williams reported the development of the backpropagation algorithm; which has been a major influence in the use of backpropagation learning. The back-propagation algorithm has emerged as the most common learning algorithm for the training of multilayer perceptrons [5].

In 1988, Linsker described a new principle for self-organization in a perceptual network. The principle is designed to preserve maximum information about input activity patterns, subject to such constraints as synaptic connections and synapse dynamic range. Linsker used abstract concepts rooted in information theory to formulate the maximum mutual information (Infomax) principle. Also in 1988, Broomhead and Lowe described a procedure for the design of layered feed-forward networks using Radial Basis functions (RBF), which provide an alternative to multilayer perceptrons.

In 1989, Mead's book, Analogue VLSI and Neural Systems, was published. It provides an unusual mix of concepts drawn from neurobiology and VLSI technology; also includes chapters on silicon retina and silicon cochlea. In the early 1990s, Vapnik invented a computationally powerful class of supervised learning networks, called Support Vector Machines, for solving pattern recognition, regression, and density estimation problems (Boser, Guyon, and Vapnik, 1992; Cortes and Vapnik, 1995; Vapnik, 1995, 1998) [30].

4.1.3 A Taxonomy of Artificial Neural Networks

In this section, taxonomy of ANNs according to learning algorithms is given. The taxonomy is divided into two tables, Table 4.1 and Table 4.2 respectively, according to learning algorithms: supervised and unsupervised learning, which will be given in more detail in Section 4.2.4.

Table 4.1 shows three common network architectures: feed-forward, feedback, and competitive which will be given in detail in Section 4.2.3.

Table 4.1 – Taxonomy according to supervised neural networks [32]

Supervised Neural Networks		
Feedforward	Feedback	Competitive
• Linear: Hebbian, Perceptron, Adaline • Multilayer Perceptron: Backpropagation • RBF Networks: Orthogonal Least Squares • Classification only: Learning Vector Quant. Probabilistic NN • Regression only: General Regression NN	• Bi-directional Associative Memory • Boltzmann Machine • Recurrent Time Series: Backpropagation Through Time, Elman Network, Finite Impulse Response, Jordan Network, Real Time Recurrent Network, Recurrent Backpropagation, Time Delay Neural Networks	• ARTMAP-1991 Carpenter, Grossberg, Reynolds • Fuzzy ARTMAP-1992 Carpenter, Grossberg, Reynolds, Markuzon, Rosen • Gaussian ARTmap-1995 Williamson • Counter-propagation Hecht-Nielsen-1987/ 88/ 90 Fausett-1994 • Neocognitron-1983 Fukushima, Miyake, Ito

Table 4.2 – Taxonomy according to unsupervised neural networks [32]

Unsupervised Neural Networks		
Competitive	Dimension Reduction	Auto-association
• Vector Quantization: Grossberg-1976 Kohonen-1984 • Self-Organizing Map: Kohonen-1995 • Adaptive Resonance Th.: ART 1-1987 Carp. /Gross. ART 2-1987 Carp. /Gross. ART 3-1991 Carp. /Gross. and Rosen Fuzzy ART-1991 Carp. / Gross. and Rosen • Differential Competitive Learning-Kosko 1992	• Hebbian Hebbian-1949 Fausett-1994 • Oja-1989 • Sanger-1989 • Differential Hebbian Kosko-1992	• Linear Auto-associator Anderson-1977 Fausett-1994 • Brain State in Box Anderson-1977 Fausett-1994 • Hopfield Hopfield-1982

The definitions of the learning algorithms for both supervised and unsupervised learning according to Table 4.1 and Table 4.2 will be given in the Section 4.3 and Section 4.4.

4.1.4 The Benefits and Application Areas of ANNs

A neural network derives its computing power through its massively parallel-distributed structure and its ability to learn. Learning refers to producing reasonable outputs for inputs not encountered during training. This is also known as generalization. This capability makes it possible for neural networks to solve complex problems that are currently intractable.

The ANNs can provide suitable solutions for problems that generally are characterised by:

i. nonlinearities,

ii. high dimensionality,

iii. noisy, complex, imprecise, imperfect and error prone sensor data,

iv. a lack of a clearly stated mathematical solution or algorithm.

The use of ANNs bring several benefits as described below [5]:

i. Nonlinearity: An artificial neuron can be either linear or nonlinear. The nonlinearity is distributed throughout the network. The nonlinearity is an important property, particularly if the underlying physical mechanism responsible for generation of the input signal is inherently nonlinear.

ii. Input-Output Mapping: The learning process involves modification of the synaptic weights of a neural network by applying a set of labeled training samples. Each sample consists of an input signal and a corresponding desired response. The network is presented with an example picked at random from the set, and the synaptic weights of the network are modified to minimize the difference between the actual response and desired response according to a learning rule. The training case is repeated several times until the system reaches a state where there are no further significant changes in the synaptic weights. As a result the network learns from the examples by constructing a mapping between the input space and output space.

iii. Adaptivity: Traditionally, intelligence within the current computer software is the result of the programmers' efforts only. During the laborious design phase of a computer software, the future operation of the software is determined. A certain set of operations is defined for each input that the program is expected to receive. Although this approach can be very effective within the problem area to which it is applied, it is usually a very difficult and time-consuming process to update the rules as the problem changes. In addition, a totally distinct set of heuristic rules is needed for every problem. However the ANNs, like biological neural networks, have a built in capability to adapt their synaptic weights to changes in the surrounding environment. A neural network trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environmental conditions.

iv. Contextual Information: Knowledge is represented by the structure and activation state of an ANN. Every neuron in the network is affected by the global activity of all other neurons in the network as in the case of biological neural networks. Hence, an ANN deals with contextual information naturally.

v. Fault Tolerance: An ANN has the potential to be inherently fault tolerant. For example, if a neuron or its connecting links are damaged, recall of a stored pattern is impaired in quality. However, due to the distributed nature of information stored in the network, the damage has to be extensive before the overall response of the network is degraded seriously. But in the case of traditional computer software, this kind of damage in only one entry of the code would cause the failure of the program.

vi. VLSI Implementation: the massively parallel nature of an ANN makes it potentially fast for the computation of certain tasks. This feature makes an ANN well suited for implementation using VLSI technology. One particular beneficial property of VLSI is that it provides a means of capturing truly complex behavior in a highly hierarchical fashion [30].

Although ANNs has been designed to address certain kinds of problems, there exist no definite rule as to what the exact application domains for certain ANNs are. The general application areas of ANNs are: robust pattern recognition, filtering, data

segmentation, data compression, adaptive control, optimization, modeling complex functions and associative pattern recognition. Table 4.3 illustrates the use of well-known neural networks, and Table 4.4 lists the application areas grouped according to the ANN structure [32].

Table 4.3 – Application areas of different neural networks [32]

Application	Back-propagation	Hopfield	Bolzmann machine	Kohonen SOM
Classification	•	•	•	•
Image processing	•			•
Decision making	•		•	•
Optimization		•	•	•

Table 4.4 – Application areas of different ANNs grouped by network structure [32]

Structure	Single layer, lateral connections	Topological vector map	Two layer, feedforward feedbackward	Multi-layer, feedforward
Network type	Hopfield	LVQ Kohonen SOM	ART	Perceptron network, Boltzmann machine
Application area	Autoassociation, Optimization	Autoassociation, Pattern recognition, Data compression, Optimization	Heteroassociation, Pattern recognition	Heteroassociation, Pattern recognition, Data compression, Optimization

4.1.5 The Neural Network Design Process

The neural network design process involves at least five main tasks. These are data collection, raw data preprocessing, feature extraction from the preprocessed data, selection of an ANN type and topology (architecture), and finally training and testing of ANN.

After suitable data is collected and pre-processed the features are chosen by the designer based on the knowledge and experience with the problem. Features should be chosen because they are believed to have some correlation to the desired output. It can be useful to eliminate redundant or ineffective features. It is also possible to determine which sets of features are the most significant by comparative analysis. An ANN design should incorporate a minimum of two sets of independent input/ output vector pairs representative of the process: There should be training, and testing vector sets.

In the following sections the general theoretical structure of artificial neural networks will be given.

4.2 Fundamentals of ANNs

4.2.1 The Basic Model of the Neuron

A neuron is an information-processing unit, which is the fundamental unit of an artificial neural network. Its basic model is illustrated in Figure 4.2.

A neuron is consist of three main elements:

i. Synapses, each of which is characterized by a weight. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}.

ii. An adder for summing the input signals, weighted by the respective synapses of the neuron.

iii. An activation function for limiting the amplitude of the output of a neuron. The activation function is also referred to as a squashing function in that it squashes the permissible amplitude range of the output signal to a finite interval.

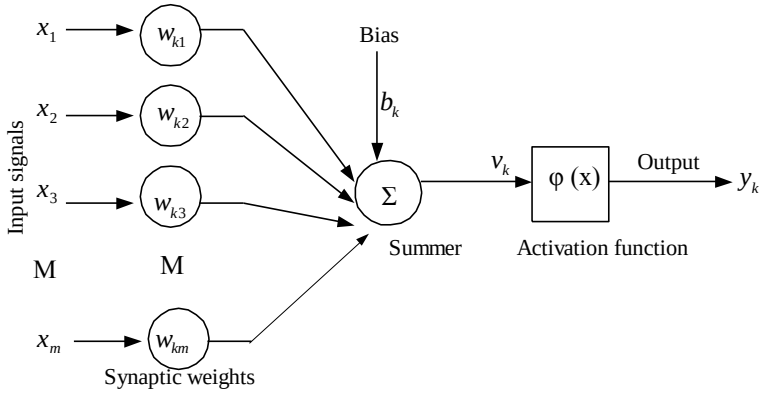


Figure 4.2 A nonlinear model of a neuron [5]

The model also includes an externally applied bias term, denoted by b_k, which has the effect of increasing or decreasing the net input of the activation function depending on whether it's positive or negative.

In Figure 4.2 y_k, the output of the neuron, can be described as,

$$y_k = \varphi(v_k) \tag{4. 1}$$

where v_k, the induced local field or activation potential of neuron k is, $v_k = u_k + b_k$. Here b_k is the bias and u_k is the linear combiner output due to the input signals. The linear combiner is formulated as,

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{4. 2}$$

So from Equations 4.1 and 4.2 the output of the neuron is derived as,

$$y_k = \varphi(\sum_{j=1}^{m} w_{kj} x_j + b_k) \tag{4. 3}$$

4.2.2 Transfer Function

The behavior of an ANN depends on both the weights and the input-output function (transfer function) that is specified for the neurons [5]. The transfer function, also known as activation function, typically falls into one of three categories:

i. Threshold function: This function is described as,

$$\varphi(v) = 1 \text{ if } v \geq 0 \tag{4. 4}$$

$$\varphi(v) = 0 \text{ if } v < 0$$

The Equation 4.4 is illustrated in Figure 4.3a. This form of threshold function with a simple difference in output, is also known as Heaviside (or signum or hard-limiter) function. The output of a neuron k is expressed as

$$y_k = +1 \text{ if } v \geq 0 \tag{4. 5}$$

$$y_k = -1 \text{ if } v < 0$$

where v_k is the induced local field of the neuron k that is,

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{4. 6}$$

Such a neuron is referred to as the McCulloch- Pitts model. In this model, the output of a neuron takes on the value of *1* if the induced local field of that neuron is nonnegative, and *0* otherwise. This statement describes the all-or one property of the model.

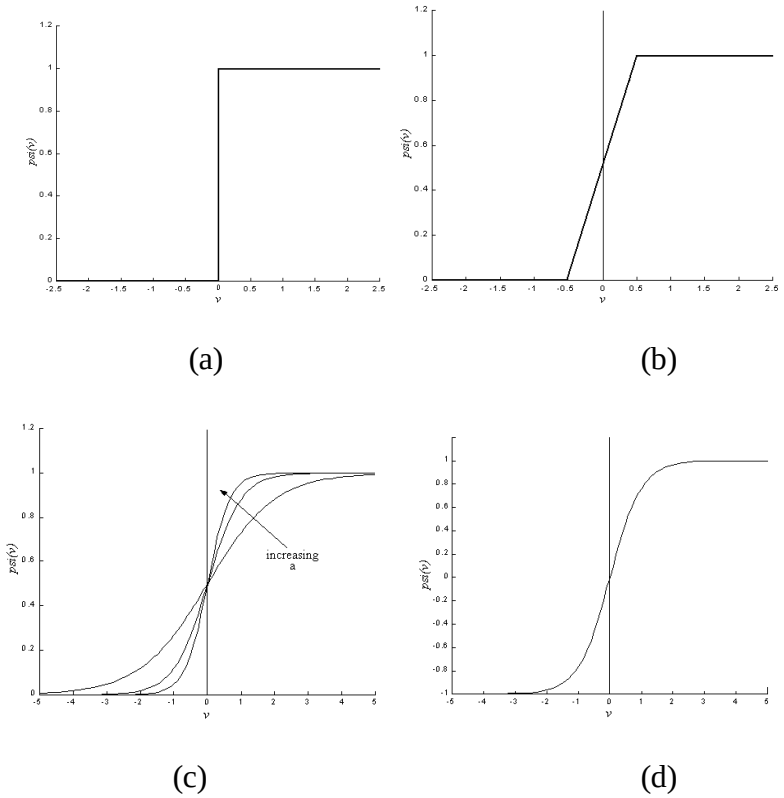
ii. Piecewise Linear function: This function is described as,

$$\varphi(v) = 1 \text{ if } v \geq +1/2 \tag{4. 7}$$

$$\varphi(v) = v \text{ if } +1/2 > v > -1/2$$

$$\varphi(v) = 0 \quad \text{if} \quad v \leq -1/2$$

where the amplification factor inside the linear region of operation is assumed to be unity. The Equation 4.7 is illustrated in Figure 4.3b. An important case is if the amplification factor of the linear region is made infinitely large, then the piecewise linear function reduces to a threshold function given by Equation 4.4.



(a) (b)

(c) (d)

Figure 4.3 The transfer function: (a) Threshold function (b) Piecewise-linear function (c) Logistic function (d) Hyperbolic tangent function

iii. Sigmoid Function: This function is defined as a strictly increasing function that exhibits a balance between linear and nonlinear behavior. Logistic function, that is a kind of sigmoid function is defined by

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{4. 8}$$

where a is the slope parameter. Changing the value of the parameter a, provides sigmoid functions of different slopes as shown in Figure 4.3c.

Another type of sigmoid function is the hyperbolic tangent function, defined by $\varphi(v) = \tanh(v)$, is shown in Figure 4.3d. The hyperbolic tangent function range from *–1* to *+1*, whereas the functions defined by Equation 4.4, Equation 4.7, and Equation 4.8 range from *0* to *+1*.

4.2.3 Architectures of ANNs

The artificial neural networks can be classified according to the structure that they exhibit. The structure of the neurons in an ANN is related with the learning algorithm used to train the network. Thus it is possible to consider that the learning rules used in the design of ANNs are structured [5].

In general the network architecture can be identified into three fundamental classes:

i. Single Layer Feed-forward Networks:

This is the simplest form of a layered network, with an input layer of source nodes that projects onto an output layer of neurons. The important point is that this network is completely a feed-forward type, so the direction of signal flow is from input layer of source nodes (I.L.) to output layer (O.L). Figure 4.4 shows a single layer network. Here, the single layer refers to output layer of neurons since there is no computation in the input layer of source nodes.

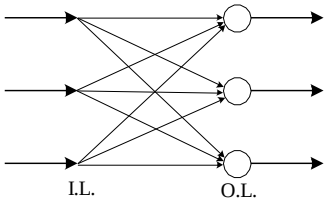


Figure 4.4 A single layer feed-forward neural network

ii. Multilayer Feed-forward Networks:

Figure 4.5 represents the structure of a multi-layered feed-forward network. This model has one or more hidden layers that are composed of one or more hidden neurons. The function of this neurons is to intervene between the input and output layers in a useful manner. By adding one or more hidden layers,

this model is capable of extracting higher-order statistics. This model is also a completely feed-forward type. There are no lateral connections within each layer and also no feed-backward connections within the network. The best-known ANN of this type is the perceptron network.

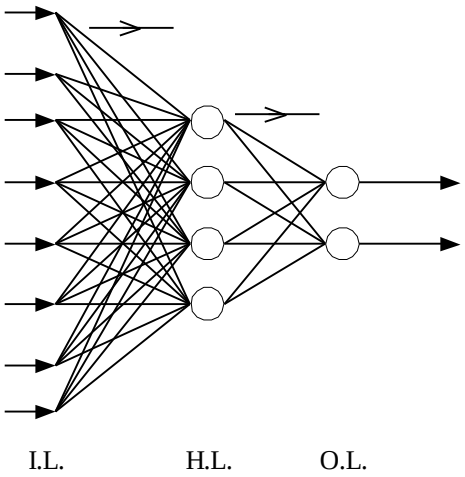
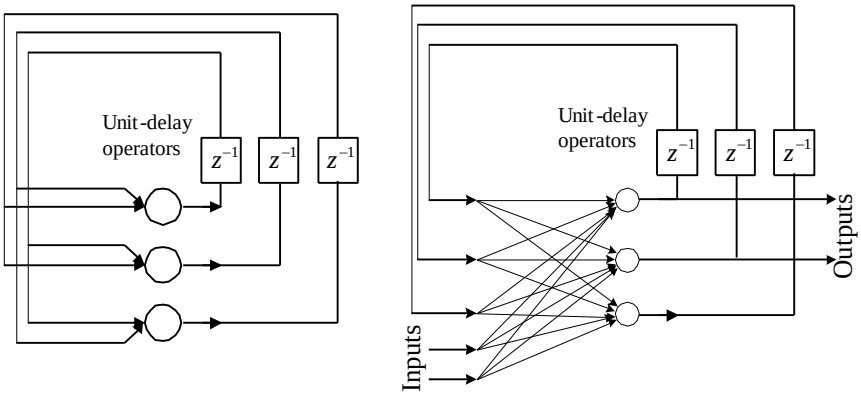


Figure 4.5 A multilayer feedforward neural network

iii. Recurrent Networks

This model is distinguished from feed-forward networks due to its feedback loops (at least one). One kind of recurrent network is shown in Figure 4.6a. Here the network consists of one single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons. Another possibility is shown in Figure 4.7b, with hidden neurons. Here the feedback connections originate from the hidden neurons as well as from the output neurons. The presence of feedback loops, has an important role on the learning capability of the ANN. The feedback loops involve the use of particular branches composed of unit-delay elements that result in a nonlinear dynamical behavior.



(a) (b)

Figure 4.6 Recurrent network with (a) no hidden, (b) hidden neurons [5]

4.2.4 The Learning Process

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms [30]:

i. The first paradigm is Associative Memory: The network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms as the auto-association and the hetero-association. In the case of auto-association, an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completion, i.e. to produce a pattern whenever a portion of it or a distorted pattern is presented. The second case, the hetero-association is related to a recall mechanism:

   - Nearest-neighbor recall mechanism, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.

ii. The second paradigm is Regularity detection: The units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge that is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights. Information is stored in the weight matrix *W* of a neural network. Learning is the determination of the weights. Following the way learning is performed; we can distinguish two major categories of neural networks:

- Fixed Networks, in which the weights cannot be changed. In such networks, the weights are fixed a priori according to the problem to solve.
- Adaptive networks, which are able to change their weights.

All learning methods used for adaptive neural networks can be classified into two major categories: learning with a teacher and learning without a teacher [5].

4.2.4.1 Learning with a Teacher

The first major category of learning is learning with a teacher, which is also referred to as supervised learning. It incorporates an external teacher, having knowledge of the environment, with that knowledge being represented by a set of input-output examples, as shown in Figure 4.7. The environment is not known by the neural network. During the learning process the teacher is able to provide the neural network with a desired response for a given training vector, from the environment. The network parameters are adjusted according to the training vector and the error signal, which is the difference between the desired response and actual response of the system. The aim is to provide the emulation of the teacher by the network. After this stage the network can generate desired response without the teacher. This kind of learning is error correction learning.

So, an important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights, which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

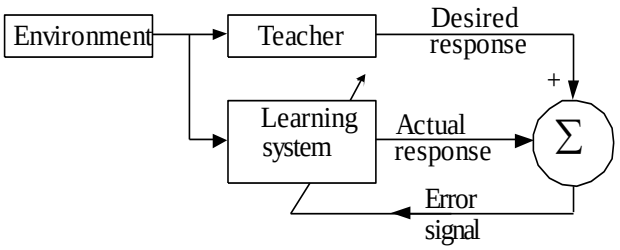


Figure 4.7 Learning with a teacher [5]

4.2.4.2 Learning without a teacher

In this category of learning, there is no external teacher to oversee the learning process. This category can be divided into two subdivisions:

i. Reinforcement Learning: The learning of an input-output mapping is performed through continued interaction with the environment in order to minimize a scalar index of performance. Figure 4.8 illustrates the general structure of reinforcement learning. Here, the critic converts a primary reinforcement signal received from the environment into heuristic reinforcement signal, which is a higher quality reinforcement signal. The aim of the system is learning under delayed reinforcement, in other words the system observes a temporal sequence of stimuli also received from the environment, which provides the generation of the heuristic reinforcement signal. The goal of the learning procedure is to minimize the expectation of the cumulative cost of actions taken over a sequence of steps. The function of the Learning system is to discover these actions and feed them back to the environment. Reinforcement learning is closely related to dynamic programming, which was developed by Bellmann in the context of optimal control theory [5].

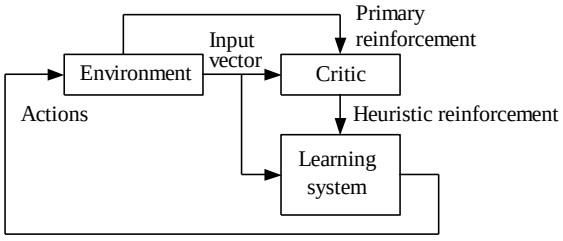


Figure 4.8 Reinforcement learning [5]

ii. Unsupervised learning: There is no external teacher or critic to oversee the learning process. Here the provision is made for a task independent measure of the quality of representation that the network is required to learn, and the parameters of the network are optimized according to that measure. When the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input and therefore to create new classes automatically [5]. This is illustrated in Figure 4.9.

Unsupervised learning is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning.

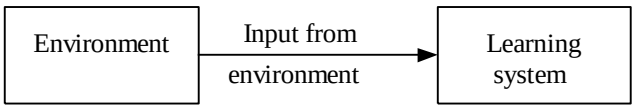


Figure 4.9 Unsupervised learning

4.3 Supervised Learning

4.3.1 Hebbian Learning

Hebb's learning rule is the oldest and the most famous of all learning rules. A simple definition according to Haykin [5] is:

- If two neurons on either side of a synapse (connection) are activated simultaneously, then the strength of that synapse is selectively increased.
- If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

This type of synapse is called a Hebbian synapse [5]. Hebbian synapse uses a time-dependent, highly local, and strongly interactive mechanism to increase synaptic

efficiency as a function of the correlation between the presynaptic and postsynaptic activities.

The mathematical model of Hebbian learning is denoted by

$$\Delta w_{kj}(n) = \eta . y_k(n) . x_j(n) \tag{4. 9}$$

where η is the rate of learning, a positive constant; w_{kj} is the synaptic weight of neuron k with presynaptic and postsynaptic signals denoted by x_j and y_k, and Δw_{kj} is the adjustment applied to the synaptic weight at time step n.

The basic idea behind Hebbian learning is that, two neurons, which are simultaneously active, should develop a degree of interaction higher than those neurons whose activities are not correlated. In the latter case, the interaction between the elements should be very low or zero.

The Hebbian learning rule underlies most of the self-organizing neural network models. Self-organization is a fundamental ability for a neural system to adapt to its environmental information structure.

4.3.2 Perceptron and Adaline

4.3.2.1 Perceptron

The perceptron was proposed by Frank Rosenblatt in 1958, as a more general computational model than McCulloch-Pitts model (see Equation 4.6). In the original Rosenblatt model the computing units are threshold elements and the connectivity is determined stochastically. The learning is provided by adapting the weights of the network with a numerical algorithm. The perceptron is built around a nonlinear neuron, namely, the McCulloch-Pitts model of a neuron. This model consists of a linear combiner followed by a hard limiter (see Equation 4.5). Rosenblatt's model was refined and perfected in 1969, by Minsky and Papert.

The classical perceptron (Rosenblatt) is in fact a whole network for the solution of pattern recognition problems. In its simplest form it consists of an N-element input layer (retina), which transmits binary values to a layer of computing units in the

projection area. The binary values from projection area feed into M-element layer, association (or predicate) area. The goal of the operation of the perceptron is to learn a given transformation $d:\{-1,1\}^N \rightarrow \{-1,1\}$ using learning samples with input x and corresponding output $y=d(x)$. In other words the idea is to train the system to recognize certain input patterns in the connection region as shown in Figure 4.10.

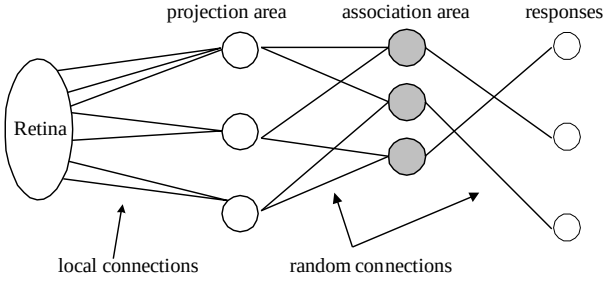


Figure 4.10 The classical perceptron [30]

The only difference between McCulloch-Pitts elements and perceptrons is the presence of weights in the networks.

Minsky and Papert refined and perfected Rosenblatt's model. In this model (shown in Figure 4.11) there is also a retina of pixels with binary values on which patterns are projected. Some pixels from the retina are directly connected to logic elements called predicates, which can compute any single bit according to the input. These predicates transmit their binary values to a weighted threshold element that is in charge of reaching the final decision in a pattern recognition problem.

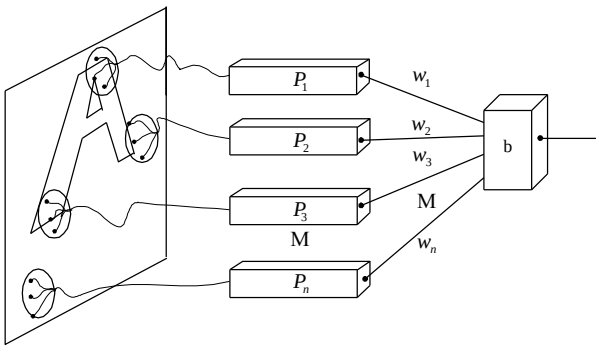


Figure 4.11 The perceptron [30]

In Figure 4.11 the predicates P_1 to P_n deliver information about the points in the projection surface that comprise their receptive fields, and the only restriction of them is that they produce a binary value and the receptive field cannot cover the

whole retina. The system consists in general of *n predicates* P_1 to P_n and the corresponding weights w_1 to w_n. The system fires only when,

$$\sum_{i=1}^{m} w_i.P_i \geq b \tag{4. 10}$$

where b is the threshold of the computing unit at the output.

From Equation 4.3 and Equation 4.5 and Equation 4.6, it is seen that the output of the network is either +1 or –1, depending on the input. The network can now be used for a classification task: it can decide whether an input pattern belongs to one of two classes. If the total input is positive, the pattern will be assigned to class +1, if the total input is negative, the sample will be assigned to class –1. the separation between the two classes, i.e. for $m=2$ inputs, will be derived by the equation:

$$w_1 x_1 + w_1 x_1 + b = 0 \tag{4. 11}$$

So this single layer network represents a linear discriminant function [31]. Here, the important case is computing the weights and biases for the network. There are two general rules for this task: The perceptron learning rule and the delta or LMS (least mean-square) rule. Both methods are iterative procedures that adjust the weights.

For a set of learning samples consisting of an input vector x and a desired output vector $d(x)$, the "perceptron learning rule" is stated as follows:

i. Start with random weights for the connections;

ii. Select an input vector x from the set of training samples;

iii. If $y \neq d(x)$, so the perceptron gives an incorrect response, modify all connections w_i according to $\Delta w_i = d(x)x_i$;

iv. Go back to step 2.

The difference between this procedure and Hebb rule in Section 4.3.1, is that, when the network responses correctly, no connection weights are modified. Besides modifying the weights, also the bias is modified. The bias term, b, is considered as a

connection w_0 between the output neuron and a 'dummy' predicate unit, which is always on: $x_0 = 1$ [30]. Given the perceptron-learning rule as stated above, this threshold is modified according to:

$$\Delta b = 0 \qquad \text{if the perceptron responds correctly;} \qquad (4.\ 12)$$

$$\Delta b = d(x) \qquad \text{otherwise.}$$

4.3.2.2 The Adaptive Linear Element (Adaline)

Widrow and Hoff presented an important generalization of the perceptron training algorithm, as the "least mean square" (LMS) learning procedure, also known as the delta rule. The main difference with the perceptron rule is the way the output of the system is used in the learning rule. The perceptron learning rule uses the output of the threshold function (either *–1* or *+1*) for learning. The delta-rule uses the net output without further mapping into output values *–1* or *+1*.

The delta rule was applied to the Adaline, developed by Widrow and Hoff in 1960. In a simple physical implementation, this device consists of a set of controllable resistors connected to a circuit which can sum up currents caused by the input voltage signals as shown in Figure 4.12. Usually the central block, the summer, is also followed by a quantizer which outputs either *+1* or *–1*, depending on the polarity of the sum [30].

In Figure 4.12, if the input conductances are denoted by $w_i, i = 0,1,...,m,$ and the input and output signals by x_i and y. The output of the central block is,

$$y = \sum_{i=1}^{m} w_i x_i + b \qquad (4.\ 13)$$

The purpose of this device is to yield a given value $y = d^p$ at its output when the set of values $x_i^p, i = 1,2,3,...,m$ is applied at the inputs. The problem is to determine the coefficients $w_i, i = 0,1,...,m,$ in such a way that the input-output response is correct for a large number of signal sets. If an exact mapping is not possible, the average

error must be minimized. For Adaline, Widrow introduced the delta rule to adjust the weights [30].

For a single layer network with one output unit with a linear activation function the output is,

$$y = \sum_j w_j x_j + b \tag{4. 14}$$

Equation 4.14 represents the linear relationship between the input and output. By thresholding the output value, a classifier such as Widrow's Adaline, can be constructed.

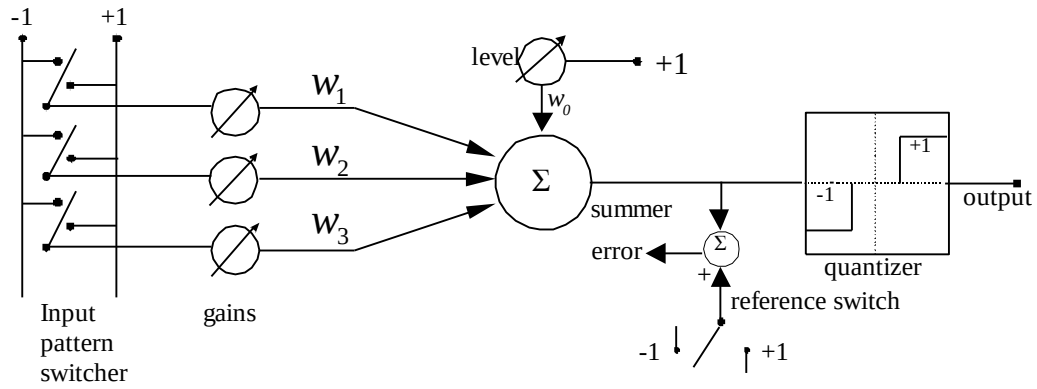


Figure 4.12 The Adaline [30]

In high dimensional input spaces the network represents a hyper-plane and also multiple output units may be defined.

For a given network, such that a hyper-plane is fitted as well as possible to a set of training samples consisting of input values x^p and desired (or target) values d^p; the output value (for every given input sample) of the network will be $(d^p - y^p)$, where y^p is the actual output. The delta-rule uses an error function based on these differences to adjust the weights.

The error (or total error) function is the summed squared error,

$$E = \sum_p E^p = \frac{1}{2}\sum_p (d^p - y^p)^2 \tag{4. 15}$$

where E^p represents the error on pattern *p*. The LMS procedure finds the values of all the weights that minimize the error function by a method called gradient descent. The idea is to make a change in the weight proportional to the negative of the derivative of the error as measured on the current pattern with respect to each weight,

$$\Delta_p w_j = -\eta \frac{\partial E^p}{\partial w_j} \tag{4. 16}$$

where η is a constant of proportionality. The derivative is,

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p} \frac{\partial y^p}{\partial w_j} \tag{4. 17}$$

By using the derivatives of the functions in Equation 4.14 and Equation 4.15 in Equation 4.17, the Equation 4.16 can be extracted as "the delta-rule",

$$\Delta_p w_j = -\eta \delta^p x_j \tag{4. 18}$$

where $\delta^p = d^p - y^p$ is the difference between the target output and the actual output of the pattern *p*. The delta-rule modifies weight appropriately for target and actual outputs of either polarity and for both continuous and binary input and output units [5].

4.3.3 The Back-Propagation Algorithm

The single layer feed forward networks, given above have some advantages and disadvantages. The disadvantage is the limited representational power; only linear classifiers can be constructed or, in case of function approximation, only linear functions can be represented. The advantage is that, due to the linearity of the system, the training algorithm converges to the optimal solution, however this is not the case for nonlinear systems such as multiple layer networks.

Minsky and Papert showed that a two layer feed-forward network can overcome the restrictions in the single layer perceptron. But they did not present a solution to the problem of how to adjust the weights from input to hidden units. Rumelhart, Hinton,

and Williams in 1986 solved this problem by introducing back-propagation (BP) algorithm [5].Figure 4.13 shows a multilayer feedforward network. Each layer consists of units that receive their input from units of a layer directly below and send their output to units in a layer directly above the unit. There are no connections within the layer.

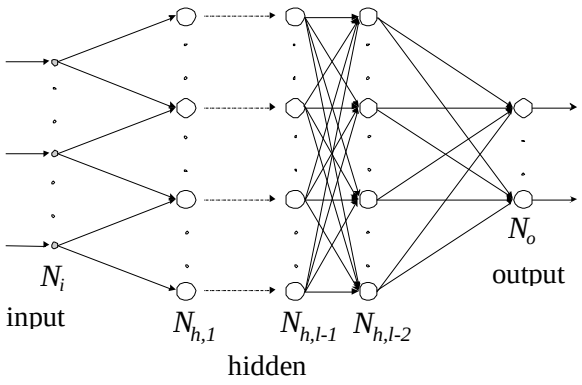


Figure 4.13 A multi-layer network with *l* layers of hidden units [5]

Since activation functions of the units in multi-layer feed-forward networks are nonlinear the delta rule must be generalized. The activation is a differentiable function of the total input given by $y_k^p = \varphi(s_k^p)$, in which $s_k^p = \sum_j w_{jk} y_j^p + b_k$. To get the generalization of the delta rule, $\Delta_p w_{jk} = -\eta \frac{\partial E^p}{\partial w_{jk}}$ is obtained from Equation 4.16. The error measure E^p is defined as the total quadratic error for pattern *p* at the output units; $E^p = \frac{1}{2}\sum_{o=1}^{N_o}(d_o^p - y_o^p)^2$, where d_o^p is the desired output for unit *o* and pattern *p*. The summed squared error is $E = \sum_p E^p$. From here, the update rule which is equivalent to the delta rule, is obtained if the weight changes are done according to,

$$\Delta_p w_{jj} = -\eta \delta_k{}^p y_j^p \tag{4. 19}$$

similar to Equation 4.18. For any hidden unit *h*, δ^p is derived as,

$$\delta_h^p = \varphi'(s_h^p)\sum_{o=1}^{N_o}\delta_o^p w_{ho} \tag{4. 20}$$

And for any output unit o, δ^p is derived as,

$$\delta_h^p = (d_o^p - y_o^p)\varphi_o'(s_o^p) \tag{4. 21}$$

So for a learning pattern, the activation values are propagated to the output units, and the actual network output is compared with the desired output values, the procedure ends up with an error, e_o, in each of the output units. The goal is to bring e_o to zero. From the delta–rule, in order to reduce an error, its incoming weights must be adapted by $\Delta w_{ho} = (d_o - y_o)y_h$. That is step one of the procedure. In order to adapt the weights from input to hidden units, the delta rule should be used again, by the help of the chain rule. Chain rule distributes the error of an output unit o to all the hidden units.

The application of the generalized delta rule thus involves two phases [31]: During the first phase the input x is presented and propagated forward through the network to compute the output values y_o^p for each output unit. This output is compared with its desired value d_o, resulting in an error signal δ_o^p for each output unit. The second phase involves a backward pass through the network during which the error signal is passed to each unit in the network and appropriate weight changes are calculated.

An important point is that, the learning procedure requires the change in weight to be proportional with $\partial E^p / \partial w$. For practical purposes, a learning rate that is as large as possible without causing oscillation is necessary .One way to avoid the oscillation at large values of η, is to make the change in weight dependent of the past weight change by adding a momentum term [30];

$$\Delta w_{jk}(t+1) = \eta \delta_k^p y_j^p + \alpha \Delta w_{jk}(t) \tag{4. 22}$$

where t indexes the presentation number and α is a constant, which determines the effect of the previous weight change.

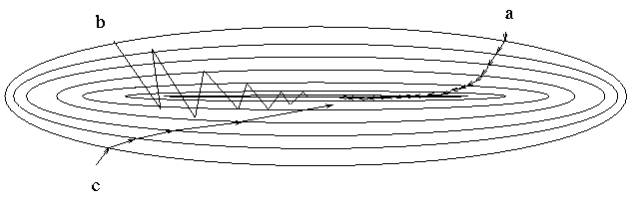


Figure 4.14 The descent in weight space, (a) for small rate; (b) for large learning rate with oscillation: (c) with large rate and momentum included [30]

The role of the momentum term is showed in Figure 4.14. When no momentum term is used, it takes a long time before the minimum has been reached with a low learning rate, whereas for high learning rates the minimum is never reached because of the oscillation. When the momentum term is included, the minimum will be reached faster.

4.4 Unsupervised Learning

In the case of supervised learning, the goal is to perform a mapping: $F = R^n \rightarrow R^m$ by presenting the network 'examples' (x^p, d^p) with $d^p = F(x^p)$ of this mapping. However, problem exists where such training data, consisting of input and desired output pairs are not available, but where the only information is provided by a set of input patterns x^p. In this case the information has to be found within the training samples x^p. Some examples, [33], of such problems are:

- Clustering: Clustering algorithms attempt to organize unlabelled feature vectors into clusters such that points within a cluster are more similar to each other than to vectors belonging to different clusters.

- Vector quantization: When a continuous space has to be discretised, this problem occurs. The input of the system is the n-dimensional vector x, the output is a discrete representation of the input space. The system has to find optimal discretisation of the input space.

- Dimensionality reduction: the input data are grouped in a subspace that has lower dimension than the dimension of the data. The system has to learn an optimal mapping, such that most of the variance in the input data is preserved in the output data.

- Feature extraction: The system has to extract features from the input signal. This often means a dimensionality reduction as mentioned above.

In unsupervised learning, training is done without the presence of the external teacher. The unsupervised weight adapting algorithms are usually based on some form of global competition between the neurons.

4.4.1 Competitive Learning

In competitive learning, the output neurons of an ANN compete among themselves to become active. While in the case of an ANN based on Hebbian learning several output neurons may be active simultaneously, in competitive learning only one single output neuron is active (fired) at any one time. Competitive learning is suitable for discovering the statistical features that may be used to classify a set of input patterns [5].

4.4.1.1 Clustering

The competitive learning is a learning procedure that divides a set of input patterns in clusters that are embedded in the input data. A competitive learning network is provided only with input vectors and thus implements an unsupervised learning procedure. Figure 4.15 shows a simple competitive learning network.

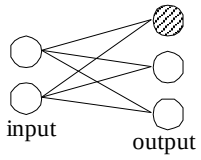


Figure 4.15 A simple competitive learning network

In Figure 4.15 all output units "*o*" are connected to all input units *"i"* with weights w_{io}. When an input pattern x is presented, only a single output unit of the network (the winner) will be activated. In a correctly trained network, all x in one cluster will have the same winner. For the determination of winner and the corresponding learning rule, two methods exist [33]:

i. Dot Product: For given input vectors x and weight vectors w_{io}, which are normalized to unit length, each output unit "o" calculates its activation value y_o according to the dot product of the input and weight vector:

$$y_o = \sum_i w_{io} x_i = w_o{}^T x \tag{4. 23}$$

In a next pass, output neuron k is selected with maximum activation,

$$\forall_o \neq k : y_o \leq y_k \tag{4. 24}$$

And the activations are set to $y_k = 1$ and $y_{o \neq k} = 0$. The output layer is referred to as winner-take-all layer. After a winner k is selected by Equation 4.24, the weights are updated according to,

$$w_k(t+1) = \frac{w_k(t) + \eta(x(t) - w_k(t))}{\left\| w_k(t) + \eta(x(t) - w_k(t)) \right\|} \tag{4. 25}$$

The denominator ensures that all weight vectors w are normalized. As a result only the weights of the winner k are updated.

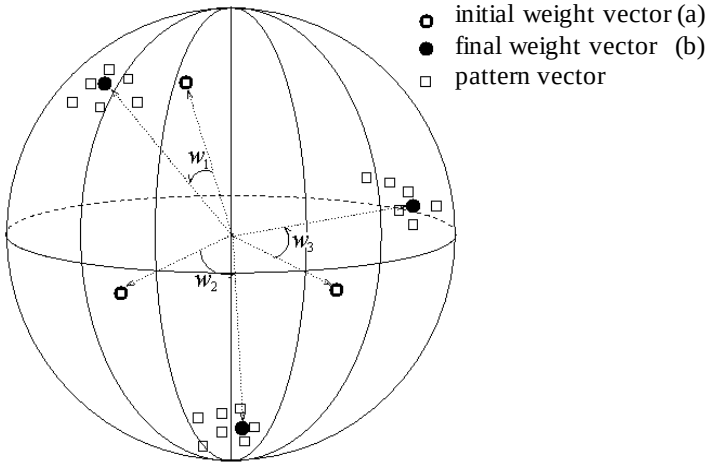
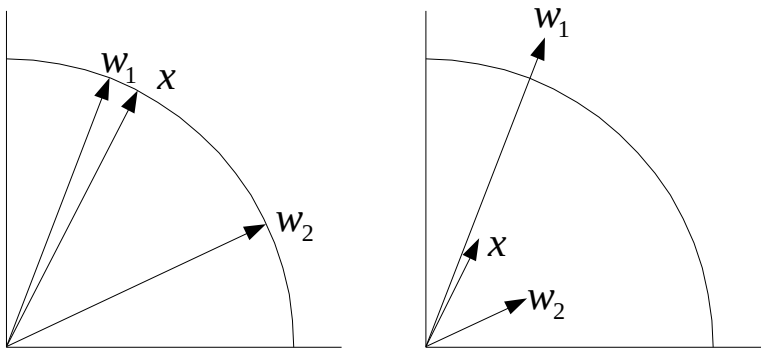


Figure 4.16 Geometric illustration of clustering with normalized vectors. The three weight vectors are rotated towards the centers of three different input clusters (a) Initial state of the network. (b) Final state of the network [5]

The procedure used in 'dot product' is illustrated in Figure 4.16. The weight update in Equation 4.25 rotates the weight vector w_{io} towards the input vector

x. Each time an input is presented, the weight vector closest to this input is selected and is rotated towards the input. As a result, weight vectors are rotated towards the areas of the clusters in the input.

ii. Euclidean Measure: Previously it was assumed that both inputs and weight vectors are normalized. In Figure 4.17, it is shown that the algorithm would fail if un-normalized vectors were used.



(a) (b)

Figure 4.17 Determining the winner. (a) Three normalized vectors (b) Three vectors with different lengths [5]

In Figure 4.17a, vectors x and w_1 are nearest to each other, and their dot product $x^T w_1 = |x||w_1|\cos\alpha$ is larger than the dot product of x and w_2. In Figure 4.17b, however, the pattern and weight vectors are not normalized, and in this case w_2 should be considered the 'winner' when x is applied. But, the dot product $x^T w_1$ is still larger than $x^T w_2$.

To be able to use unnormalized input data, the winning neuron k is selected with its weight vector w_k closest to the input pattern x, using the Euclidean distance measure:

$$k : \|w_k - x\| \leq \|w_o - x\| \quad \forall_o \tag{4. 26}$$

If all vectors were normalized the Equation 4.26 would be reduced to Equation 4.23 and Equation 4.24. In Euclidean measure, instead of rotating

the weight vector towards the input, the weight update is changed to implement a shift towards the input,

$$w_k(t+1) = w_k(t) + \eta(t)(x(t) - w_k(t)) \qquad (4.\ 27)$$

$$w_i(t+1) = w_i(t) \quad \text{for } i \neq k$$

where $\eta(t)$ is a suitable, monotonically decreasing scalar-valued gain coefficients $0 < \eta(t) < 1$. Then this is the simplest definition of "Competitive learning".

The competitive learning stems from "cluster analysis". Assume a sequence of statistical samples of a vectorial observable $x = x(t) \in R^n$ where t is the time coordinate, and a set of variable reference vectors $\{m_i(t); m_i \in R^i, i = 1,2,\Lambda\ ,r\}$. Assume that the $m_i(0)$ have been initialized in some proper way. Competitive learning then means that if the input *x(t)* can be compared in parallel with all the $m_i(t)$ at each successive time instant, to be an integer **(***t=1,2,3,...)*, then the best matching $m_i(t)$ is updated to better comply with *x(t)*. If comparison is based on some distance measure $d(x, m_i)$**,** updating must be such that if $i = k$ is the index of the best-matching reference vector, then $d(x, m_k)$**)** shall be decreased, and all the other reference vectors with $i \neq k$ left intact. In this way, in the long run, the different reference vectors tend to become specifically "tuned" to different domains of the input variable x. If the probability density function of *p(x)* is clustered, then the m_i tend to describe the clusters. In general, it can be shown that the m_i tend to be placed into the input space R in such a way that they approximate *p(x)* in the sense of some minimal residual error [33].

So, a competitive network performs a clustering process on the input data by dividing the input patterns in disjoint clusters such that similarities between input patterns in the same cluster are much bigger than similarities between input patterns in different clusters. Similarity is measured by a distance function on the input vectors. A common criterion to measure the quality of a given clustering is the

square criterion, given by $E = \sum_p \left\| w_k - x^p \right\|^2$ where k is the winning neuron when input x^p is presented. The competitive learning seeks to find a minimum for this square error by following the gradient of the error-function

$$E = \frac{1}{2}\sum_i (w_{ki} - x_i^p)^2 \tag{4. 28}$$

where k is the winning unit, minimized by the weight update rule in Equation 4.27.

4.4.1.2 Vector Quantization (VQ)

The VQ is a classical method in signal processing to produce an approximation to the distribution of a single class by a reproduction (codebook) vector [33]. Each incoming signal is mapped to the nearest codebook vector, and that vector sent instead of the original signal. One way to choose the codebook is to minimize some measure of the approximation error averaged over the distribution of the signal, and over the training patterns of that class. Taking the measure as the squared distance from the nearest codebook vector leads to the *k*-means algorithm, which aims to minimize the sum of squares of distances within clusters. The *k*-means algorithm is applied as follows:

i. Begin with an arbitrary set of cluster centers in the vector space and assign the sample vectors to the nearest centers,

ii. Compute the sample vector mean of each center,

iii. Reassign each and every sample vector to the cluster with the nearest mean,

iv. If the classification of all sample vectors has not changed, stop: else go to step 2

The distance measure is Euclidean and a similarity measure J is used as defined by;

$$J = \sum_{k=1}^{N_C} \sum_{i \approx k} \left| y_i - m_k \right|^2 \tag{4. 29}$$

where N_c is the number of clusters, y_i are the sample vectors, and m_k are the cluster centers or cluster means. For fixed set of sample vectors, J is minimized by choosing m_k to be the sample mean of the kth cluster. When m_k is fixed. J is minimized by choosing the class of y_i as the class of the cluster with the nearest mean. The set of means or centers is often called the 'codebook' and the problem of choosing the centers is called 'code book' design. VQ is similar to clustering or finding centers of vectors that may be said to be correlated or related in some way. The self-organizing map developed by Kohonen [34], maps some given vectors to finite set of output nodes, which can be defined as cluster centers for related groups of vectors.

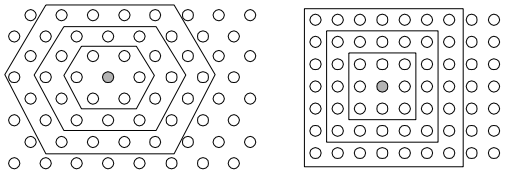
4.4.2 Kohonen Networks

Given a set of information signals *m*, an ANN is capable of automatically forming an inner representation of signals. In particular, when each signal in *m* is represented at a specific position of neural field, a map of the signals on the neural field is obtained. A neural field implies a network in which neurons are arranged on a two-dimensional space like cortex. This is called a cortical map or neural map of information [31]. The basic properties of a map are; the topological relation between the map and the original signal space *m*, the resolution and stability of a map.

Willshaw and Von Der Malsburg proposed a self-organizing mechanism of this type of a cortical map. Kohonen also proposed a simplified but powerful model, and studied its properties. Moreover, he utilized the map to form a vector quantizer, emphasizing its discrete characteristics [34].

4.4.2.1 Kohonen Self-organizing Feature Maps (SOFM)

A self-organizing map (SOM) is formed of neurons located on a regular, usually one or two-dimensional grid. Also higher dimensional grids are possible, but since their visualization is difficult, they are not generally used. The neurons are connected to adjacent neurons by a neighborhood relation dictating the structure of the map. In the two-dimensional case the neurons of the map can be arranged either on a hexagonal or a rectangular lattice, as shown in Figure 4.18.



(a) (b)

Figure 4.18 Neighborhood (size 1,2 and 3) of the unit marked with black dot: (a) hexagonal lattice, (b) rectangular lattice

Each unit i in the grid is represented by a prototype vector m_i. The number of map units, which typically varies from a few dozen up to several thousands, determines the accuracy and generalization capability of the SOM. During training, the SOM forms an elastic net that folds onto the cloud formed by the input data. Data points lying near each other in the input space are mapped onto nearby map units. So, the SOM can be interpreted as a topology preserving mapping from input space onto the two dimensional grid of map units.

The principal goal of SOM is to transform an incoming signal pattern of arbitrary dimension into a discrete map, and to perform this transformation adaptively in a topologically ordered fashion.

Figure 4.19 shows the diagram of a two dimensional lattice of neurons commonly used as the discrete map. Each neuron in the lattice is fully connected to all the source nodes in the input layer. This network represents a feed-forward structure with a single computational layer consisting of neurons arranged in rows and columns.

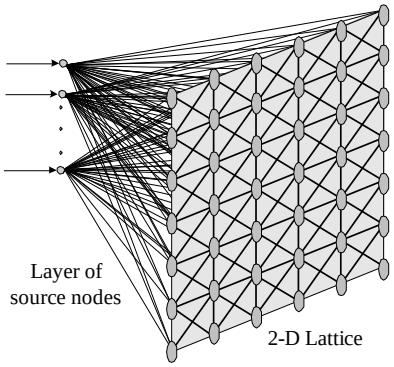


Figure 4.19 Two-dimensional lattice of neurons

The training procedure of SOM is iteratively. At each training step, a sample vector x is randomly chosen from the input data set. Distances between x and all the prototype vectors are computed. The best matching unit (BMU), which is denoted by b, is the map unit with prototype closest to x:

$$\|x - m_b\| = \min_i \{\|x - m_i\|\} \tag{4. 30}$$

Then, the prototype vectors are updated. The BMU and its topological neighbors are moved closer to the input vector in the input space. The update rule for the prototype vector of unit i, according to [34] is

$$m_i(t+1) = m_i(t) + \eta(t) h_{bi}(t)[x - m_i(t)] \tag{4. 31}$$

where t is time, $0<\eta(t)<1$ is time dependent learning rate parameter, $h_{bi}(t)$ is the neighborhood kernel function centered on the winner unit. A typical choice of $h_{bi}(t)$ is the Gaussian function [5]

$$h_{bi}(t) = \exp\left(-\frac{\|r_b - r_i\|^2}{2\sigma^2(t)}\right) \tag{4. 32}$$

where r_b and r_i are positions of neurons b and i on the SOM grid, and $\sigma(t)$ is the width of the topological neighborhood function. It is denoted as,

$$\sigma(t) = \sigma_o \exp\left(-\frac{t}{\tau_1}\right) \tag{4. 33}$$

where σ_o is the value of $\sigma(t)$ at the initiation of the SOM algorithm, and τ_1 is a time constant. In Equation 4.31, $\eta(t)$ *is*

$$\eta(t) = \eta_o \exp\left(-\frac{t}{\tau_2}\right) \tag{4. 34}$$

where τ_2 is another time constant and η_o is the initial state of learning rate. Both $\eta(t)$ and $\sigma(t)$ decrease monotonically with time.

In the case of a discrete data set and fixed neighborhood kernel, the error function of SOM can be shown to be

$$E = \sum_{i=1}^{N} \sum_{j=1}^{M} h_{bi} \left\| x_i - m_j \right\|^2 \qquad (4.\ 35)$$

where N is number of training samples in the input data set, and M is the number of map units. Neighborhood kernel is centered at unit b, which is the BMU of vector x_i, and evaluated for unit j [33].

The adaptation of the synaptic weights in the network, computed by Equation 4.31, is generated in two phases: an ordering or self-organizing phase followed by a convergence phase [34]:

Self-organizing or ordering phase: It is during the first phase of the adaptive process that the topological ordering of the weight vectors takes place. The ordering phase may take as many as 1000 iterations of the SOM algorithm, and possibly more.

Convergence phase: This phase of the adaptive process is needed to fine-tune the feature map and to provide an accurate statistical quantification of the input space.

As described in Section 4.4.1.2, VQ, an input space is divided into a number of distinct regions, and for each region a reconstruction vector is defined. When the quantizer is presented a new input vector, the region in which the vector lies is first determined, and is then represented by the reproduction vector for that region. So, by using an encoded version of this reproduction vector for storage or transmission in place of the original input vector, considerable savings in storage can be obtained, at the expense of some distortion. The collection of possible reproduction vectors is called the codebook of the quantizer, and its members are called code words, as mentioned earlier in Section 4.4.1.2.

A VQ with minimum encoding distortion is called a “Voronoi” or “Nearest-neighbor quantizer”, since the Voronoi cells about a set of points in an input space correspond

to a portion of that space according to the nearest neighbor rule based on the Euclidean metric [35,36]. The SOM algorithm provides an approximate method for computing the "Voronoi vectors" in an unsupervised manner, with the approximation being specified by the synaptic weight vectors of the neurons in the feature map. Thus, computation of the feature map may be viewed as the first of two stages for adaptively solving a pattern classification problem. The second stage is provided by learning vector quantization, which provides a mechanism for the final fine-tuning of a feature map [5].

4.4.2.2 Learning Vector Quantization (LVQ)

LVQ is a supervised learning technique that uses class information to move the Voronoi vectors slightly, so as to improve the quality of the classifier decision regions. An input vector x is picked at random from the input space. If the class labels of the input vector x and a Voronoi vector w agree, the Voronoi vector w is moved in the direction of the input vector x. However, if the class labels of the input vector x and the Voronoi vector w disagree, the Voronoi vector w is moved away from the input vector x.

In the case of classifying a number of given input signal sets, into a 'finite' number of categories, several codebook vectors are usually made to represent each class, and their identity within the classes is not important. But the important case is the decisions made at "class borders". So, it is then possible to define effective values for the codebook vectors such that they directly define optimal decision borders between the classes, even in the sense of classical Bayesian decision theory [33]. For this reason, Kohonen proposed LVQ1, which will be described below [34].

According to "LVQ1" algorithm, if several codebook vectors "m_i" are assigned to each class, and each of them is labeled with the corresponding class symbol, the class regions in the input space are defined by simple nearest neighbor comparison of x with the m_i; the label of the closest m_i defines the classification of x.

To define the optimal placement of m_i in an iterative learning process, initial values for them must be first set using any classical VQ method or by the SOM algorithm. The initial values in both cases roughly correspond to the statistical density function

$p(x)$ of the input. The next phase is to determine the labels of the codebook vectors, by presenting a number of input vectors with known classification, and assigning the cells to different classes by majority voting, according to the frequency with which each m_i is closest to the calibration vectors of a particular class.

It is proved [34] that, the classification accuracy is improved if the m_i are updated according to the following algorithm. The main idea is to pull codebook vectors away from the decision surfaces to demarcate the class borders more accurately. Let m_c be the codebook vector closest to x in the Euclidean metric; this then also defines the classification of x. Then apply training vectors x, the classification of which is known. Update the $m_i = m_i(t)$ as follows:

$$m_c(t+1) = m_c(t) + \alpha(t)\left[x(t) - m_c(t)\right] \tag{4. 36}$$

if x is classified correctly;

$$m_c(t+1) = m_c(t) - \alpha(t)\left[x(t) - m_c(t)\right]$$

if x is classified incorrectly;

$$m_i(t+1) = m_i(t) \quad \text{for} \quad i \neq c$$

Here $\alpha(t)$ is a scalar adaptation gain (learning rate) $(0 < \alpha(t) < 1)$, which is decreasing monotonically in time. Since this is a fine-tuning method, its initial value should be selected a small value, i.e. 0.01 or 0.02 for 100.000 steps.

5. IMPLEMENTATION OF THE PROPOSED TOOL FOR DISTRIBUTION SYSTEM FAULT CLASSIFICATION

5.1 Introduction

This section discusses the implementation of an integrated design of fault classifier in a 34.5 kV distribution system by using the hybrid "Wavelet-ANN-based" approach. The section is divided into three subsections: The first subsection introduces the 34.5 kV test system, all the features of the distribution system and the monitored data at PSCAD/EMTDC simulation software. The second subsection introduces the features of the wavelet transform module as a preprocessor to extract relevant information from the raw data monitored from the test system. Finally the last subsection describes the hybrid neural classifier, all the parameters of the feature vectors, input-output structures and the results obtained from the ANN-based classifier.

5.2 Simulation of a 34.5 kV Distribution System

In this thesis a distribution system in Istanbul, Turkey is simulated by the educational edition of PSCAD/EMTDC simulation program. This is the 34.5 kV "Sagmalcılar-Maltepe" distribution system with a 61-bus configuration. Due to software limitations and practical reasons, a reduced model shown in Figure 5.1 is used to generate the test data. One of the limitations in the software is related to the node number. In the actual distribution system the node number is 326. However in this version of the simulation program the node number is limited to 200 nodes. The system is reduced to be 66 nodes as shown in Figure 5.1 to generate the simulation data.

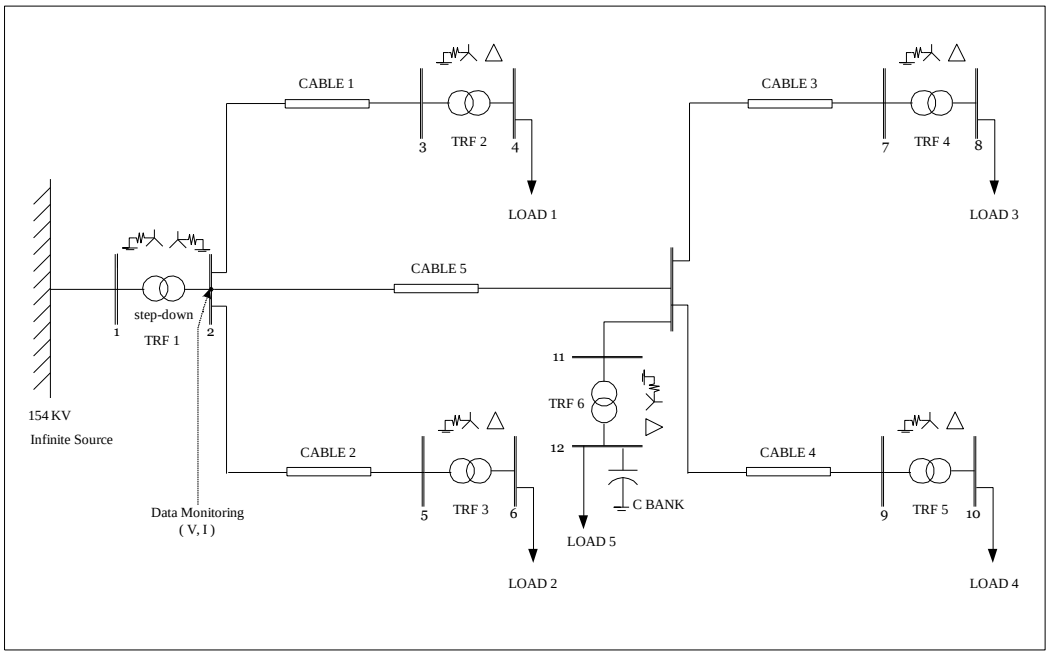
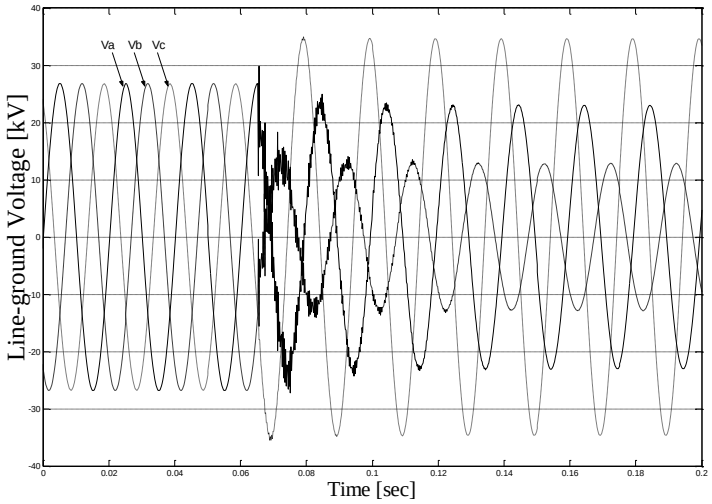


Figure 5.1 One-line diagram of the reduced 34.5 kV Sagmalcılar-Maltepe Distribution System

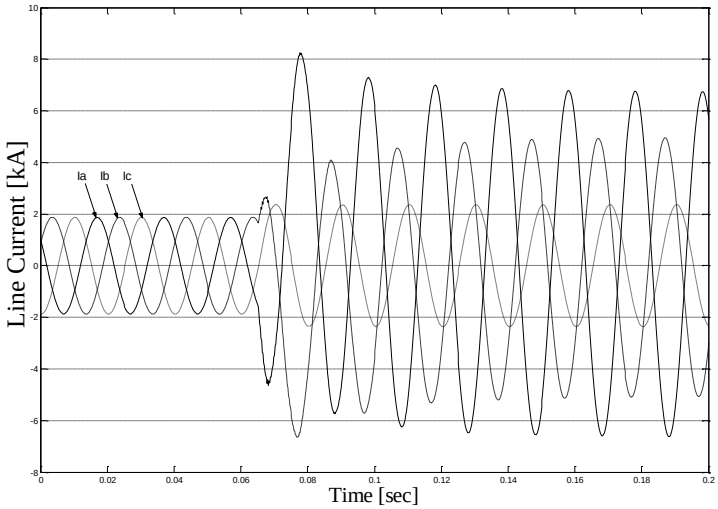
The network parameters of the test system given in Figure 5.1 is provided in Appendix B in Table B 1. The test system is a 12-bus distribution system with a base power of 100 MVA and a base voltage of 34.5 kV. All of the test data were monitored from the secondary of the main step-down transformer (TRF 1) located at the Sagmalcılar substation as shown in Figure 5.1, node 2. The objective is to monitor the voltage and the current at the 34.5 kV bus and identify the fault classes. Each sample of data contained six channels, a set of three line current and three line-to-ground voltages, which is typically what a recording device would measure in a "real system". The signals are generated at an equivalent sampling rate of 5 KHz per channel, which could be increased if needed. The fundamental frequency of the waveforms is 50 Hz. Ten cycles of data per channel with a total time duration of 200 milliseconds were captured for each sample from which features were extracted.

Various fault conditions were simulated using the PSCAD / EMTDC software (refer to Section 2 for information about the software), which is an ideal tool for fault simulation and transient analysis. (The test system layout used in the simulation software is provided in Appendix B, Figure B.1). A database of line currents and line-to-ground voltages is built up under normal and fault conditions. Specific events simulated including system faults at different fault inception angles and fault locations. The fault inception angles (FIA) used in the simulations are in the range of

0^o ~ 180^o. Since the waves are periodic, it is sufficient to study angles in the range of 0^o ~ 180^o. The selected angles are: 0^o, 30^o, 60^o, 90^o, 120^o, 150^o, 180^o. Short-circuit faults are simulated at various locations of the test system. The five fault locations are shown in Appendix B, Figure B.1. Four main types of system faults are generated: single-phase to ground, two-phase fault, two-phase to ground, and three-phase to ground fault.



(a)



(b)

Figure 5.2 Typical measured voltage and current patterns. (a) voltage waveforms, (b) current waveforms

Figure 5.2 shows the three-phase line currents and the line-to-ground voltage signals for a "phase A-phase B to ground" fault occurred at bus 4 on 0.0636 sec. with a fault

inception angle of 90^{o}. It is possible to investigate various types of disturbances in this example. An abrupt change can be observed in both of the current and voltage signals at the time 0.0636 sec. Figure 5.2a shows a 15% and 50% sag disturbance on the faulted phase voltages "V_a" and "V_b" respectively, and a 30% swell disturbance on the unfaulted phase "V_c". There is some high-frequency (HF) distortion on the voltage waveforms, in particular on the faulted "a" and "b" phases at fault occurrence time and this is so by virtue of the fact that there is a large step change in the "a" phase voltage and the "b" phase voltage when the fault occurs. In addition to voltage signals, the faulted line (I_a and I_b) currents increase greater than the current (I_c) in healthy line as it can be seen from Figure 5.2b. In the case of double-line-ground fault, only the faulty line currents increases greatly: the increase in the magnitude of the faulted "a" phase current and "b" phase current is larger than "c" phase current, as expected. Unlike the voltage waveforms, the current signals are relatively distortion free from a HF point of view. The abrupt change observed in the current and voltage signal at 0.0636 sec. enables the classification scheme to identify the fault phase and fault type.

Since the waveforms have certain distinct characteristics, any successful classification tool would be able to pick out these relevant features and associate the waveforms with those of a certain fault class. Here, fault classification is defined as a multiclass problem. The ten types of faults (A-g, B-g, C-g, A-B, A-C, B-C, AB-g, AC-g, BC-g, ABC-g) produce a ten-class classification problem. Having chosen the classes, the next step in developing a classifier is the selection and extraction of desired features. This refers to the preprocessing of raw data into a smaller set of features that would be the input to the classifier. This is probably the most critical step in the analysis. The criterion used in feature selection was to represent the important characteristics that distinguish each class from another. Here the "wavelet multi-resolution analysis" technique (for information about MRA refer to Section 3.4.4) is used as a preprocessing unit to obtain a smaller set of data to represent each of the class. The key idea in using wavelet transform analysis for classifying fault types is based on the uniqueness of the wavelet transform coefficients (WTCs) for each type of signals. The second technique is "descriptive statistics" that summarize the data into a small set of numbers that contain most of

the relevant information. After the uniqueness of each signal is found, then an ANN-based classification tool was employed for classification task. The main idea in this approach is solving the complex fault (three-phase short-circuit) classification problem under various system and fault conditions. This classification is a pattern recognition problem where the process must be able to discriminate various fault classes. The ANN technique provides the ability to classify the classes by identifying different patterns of the associated voltages and currents. The performance of the proposed fault classification scheme is evaluated based on a database of about 350 sample cases simulated under different fault types, fault inception angles and various fault locations monitored from the 34.5 kV test system given in Figure 5.1 and Appendix B, Figure B.1.

In the next sections the preprocessing module and the ANN-based classification technique will be described.

5.3 Feature Detection and Extraction

The neural network approach to the detection and classification of system faults consists of three general tasks; generating sets of line current and line-to-ground voltages, using these sets to train a neural network, and testing the network on separate sets of line currents and line-to-ground voltages. The preprocessor is an internal part of this scheme. Training cases were generated using an electro-magnetic transient simulation program on a distribution system as described in the Section 5.2. To enhance the competence of the classifier system, it is necessary to pre-process the event signals to extract characteristic information. Also, it is impractical to use the raw waveforms directly as input for a neural network. Thus, certain characteristics of the waveforms must be identified and reduced to quantitative form in order for the network to distinguish between faulty conditions.

5.3.1 Introduction

Feature extraction is a pre-processing operation that transforms a pattern from its original form to a new form suitable for further analysis. It reduces the high dimensionality of the initial system description. The feature extraction method proposed in this thesis is based on the “wavelet MRA technique”, and the

distribution of the energy of given signal within different frequency sub-bands. In addition to the MRA technique, the "descriptive statistics" analyze is also used to obtain the measure of dispersion of the signals and the detail coefficients. As a result, the raw data generated by EMTDC is mapped into a small size of interpretable features.

The MRA is a tool that utilizes the DWT to represent a time-varying signal in terms of its frequency components. It essentially maps a one-dimensional signal of time into a two-dimensional signal of time and scale. Wavelet analysis involves representing signals in terms of simpler, fixed building blocks (wavelets) at different scales and positions. The main idea is to develop representations of a complicated signal $f(t)$ in terms of its orthonormal basis, which are the scaling and the wavelet functions. These two functions are translated and scaled to produce wavelets at different locations (positions) and on different scales (durations). Fine-scale wavelets are narrow and brief; coarse-scale wavelets are wide and long lasting. The wavelet functions represent the high frequencies corresponding to the detailed parts of a given signal, and scaling functions represent the signal's low frequencies or smooth parts. These functions can be scaled and translated to decompose $f(t)$ and represent it at different resolutions or scales. This decomposition technique is called multi-resolution signal decomposition (MSD).

The purpose of feature extraction task is to identify specific signatures of the fault types in the system. The wavelet transform breaks down the signal into different time-frequency scales. Each scale represents the signal in the corresponding sub-band. By using wavelet analysis, the sub-band information can be extracted from the simulated waveforms, which contain useful fault features. Some bands are intensive to some types of fault. The energy content of the scale signals relative to the given signal changes depending upon the type of disturbance. By analyzing these features of the detail signals, different types of fault can be detected and classified.

In the proposed feature extraction method, disturbance detection is performed in the wavelet domain rather than time or frequency domain. Using the MSD technique, it is possible to decompose the given signal into different resolution levels. Any changes in the smoothness of the signal can be detected and localized at the finer resolution levels. The detection and localization behavior of five-level MSD of a

current signal is shown in Figure 5.3. (The "analysis" block is defined in Figure 5.4). The input signal is the I_b current for a three-phase to ground fault in bus-4 with a FIA of 150 °. Here, the MSD approach is based on a dyadic-orthonormal wavelet transform analysis with Daubechies' wavelet with a ten-coefficient filter having five-vanishing moments. The value of "vanishing-moments" determines the constants leading to zeros in the wavelet spectrum. For example it can be seen from Figure 5.3 that regions of the given signal that are constant lead to corresponding zeros in the associated wavelet spectrum. Generally regions that are linear, quadratic, and cubic, etc. lead to zeros in the wavelet spectrum, which improves the scale decompositions, because scales tend to separate and localize better. This generalization is known as the property of vanishing moments [38]. As the number of vanishing moments, hence the number of filter coefficients grows the coarse approximation becomes smoother and the small-scale oscillations are separated better. The choice of number of filter coefficient depends on the type of signal analyzed. More vanishing moments lead to better localization of scales and poorer time localization. After examinations of several types of wavelets like Daubechies 4, Daubechies 8, Daubechies 10, Daubechies 40; Daubechies 10 is chosen in this thesis and used in the entire analysis.

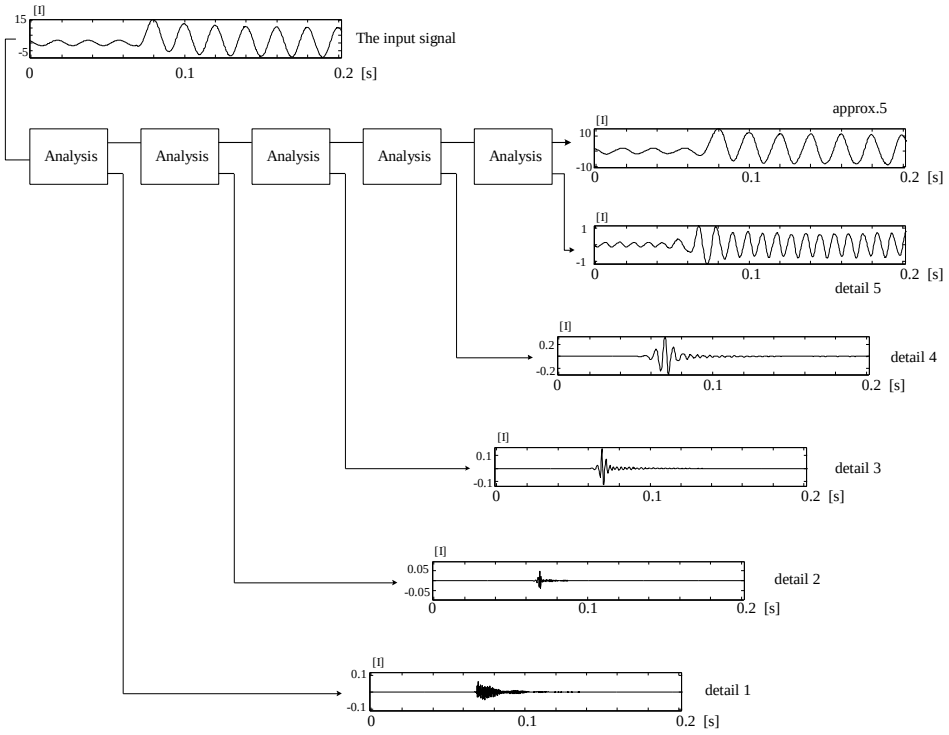


Figure 5.3 Five-level MSD of a distorted signal

As shown in Figure 5.3, a five-scale signal decomposition is performed to ensure that all disturbance features in both high and low frequency are extracted. The given input signal is decomposed into other signals, which represent a smoother version and detailed versions of the original signal. The output of the wavelet transform is six decomposed scale signals with different level of resolution. Therefore, the distorted signal is represented as a sum of wavelets. It is seen from Figure 5.3 that the first finer decomposition levels of the distorted signal may be adequate to detect and localize the disturbance. However, the other coarser resolution levels are also used to extract more features that can help in the classification scheme.

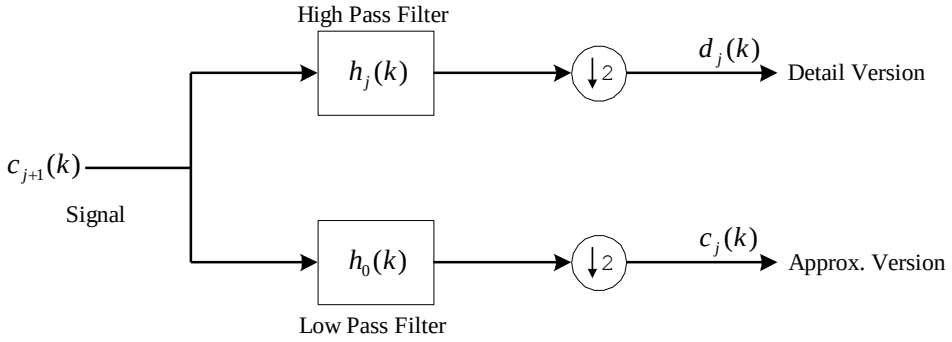


Figure 5.4 One stage MSD using convolution and decimation by factor 2

The input signal in Figure 5.3 has 1001 sample points lasting 0.2 sec. The signal consists of 10 cycle data, each cycle lasting 0.02 sec. The detail signal in scale 1, $d_1(n)$, has 510 sample points due to decimation by a factor of two, as shown in Figure 5.4. The other detail signals have 264, 141, 80, 49 sample points for $d_2(n)$, $d_3(n)$, $d_4(n)$, $d_5(n)$ respectively. The approximation signal at the output of last "analysis" module is $c_5(n)$ with 49 sample points. The input signal has been sampled at 5 KHz. Thus, a five-scale decomposition of a signal yields 5 detailed signals having a frequency band of 2.5-1.25 KHz at scale 1; 1.25-0.625 KHz at scale 2; 625-312.5 Hz at scale 3; 312.5-156.25 Hz at scale 4; 156.25-78.125 Hz at scale 5 and one smooth signal contains frequency band 78.125 Hz to DC level.

All of the six channels ($V_a, V_b, V_c, I_a, I_b, I_c$) were decomposed by the five-level MSD with Daubechies 10 filters for the entire simulation in this thesis. The six

channel's data and their sub-bands have the characteristics described in the previous paragraphs.

In addition to "wavelet MRA technique" briefly described in the previous paragraphs, the "descriptive statistics" analyze is also used to obtain the measure of dispersion of the signals and the detail coefficients. Descriptive statistics are a way to summarize the data into a small set of numbers that contain most of the relevant information. Statistical information like standard deviation, variance and other average quantities like maximum amplitudes of line currents and line-to-ground voltages and their sub-band information obtained by five-level MSD were also extracted. These parameters are different for each fault class, thus they are unique identifying features.

For each set of six channel data *27* parameters are computed. Generally, the computed parameters are the current for three phases before and immediately after the fault occurred, energy of the current signals over the five detailed components, power and energy of the voltage channels for each phase. The detailed description of the parameters is given in the Section 5.3.4.

The "data pre-processing" scheme is represented in Figure 5.5. The pre-processor extracts pertinent information over 10 cycles of operation. The voltage and current signals monitored from the secondary of the main step-down transformer (TRF 1) in Figure 5.1 are fed into signal-processing unit. These modules extract the features required by the fault detection and classification network. Figure 5.5 shows the different modules belonging to the signal-processing unit. Module I extracts the five-level MSD detail coefficients of three current channels. The statistical information and other average quantities are extracted by the Module II and Module III: Module II extracts the distribution of the energy of the sub-bands obtained by the Module I. Module III extracts the statistical information and average quantities of the three channel voltage and the three channel current information. The "feature collector" module collects the information produced by the previous three modules. The information is then put in the order by feature collector module and produced a feature vector of *27* parameters. The detailed parameters of the feature vector will be provided in the Section 5.3.4.

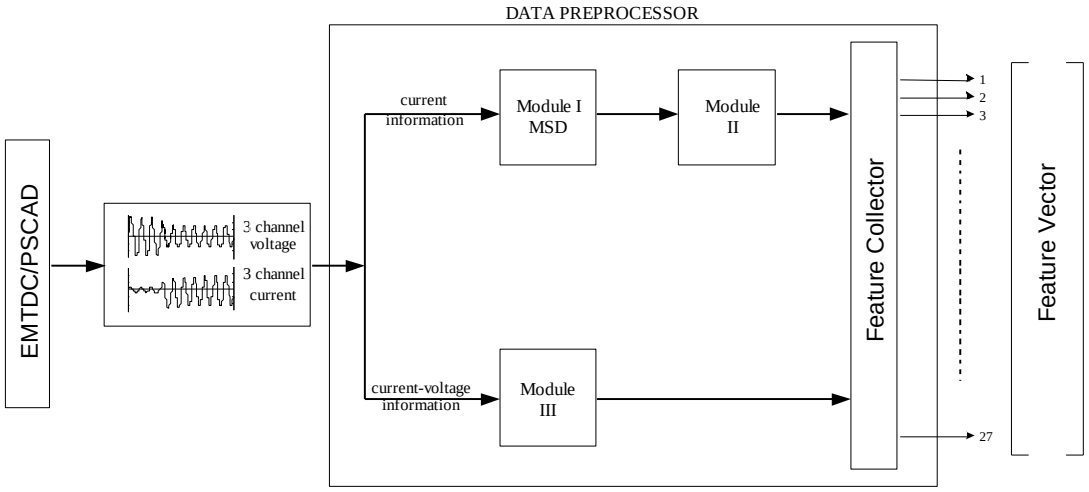


Figure 5.5 Data processing and feature extraction architecture

5.3.2 Parseval's Theorem

As mentioned in the previous section, feature extraction is a pre-processing operation, which transforms a pattern from its original form to a new form suitable for further processing. The features extracted by the Module I in Figure 5.5 are processed by a second module to extract the energy distribution of the given pattern. Parseval's theorem relates the energy of the distorted signal to the energy in each of the expansion components and their wavelet coefficients if the selected scaling function and the wavelet function form an orthonormal basis. This means that the energy of the signal $f(t)$ can be partitioned in terms of the expansion coefficients as in [18]:

$$\int |f(t)|^2 dt = \sum_{k=-\infty} |c_0(k)|^2 + \sum_{j\_0}^{\infty} \sum_{k=-\infty}^{\infty} |d_j(k)|^2 \qquad (5.1)$$

$$W_{Signal} = W_{c_0} + \sum_{j=0}^{j-1} W_{d_j}$$

where $d_j(k)$ is the detail coefficients at scale j and $c_0(k)$ presents the last approximate coefficients as shown in Figure 5.4.

The energy of the current signals will be partitioned at different resolution levels by the property of Parseval's theorem. The standard deviation at different resolution

levels of the decomposed signal can be considered as a measure of the energy as it was used in [9] and [18] as a feature to classify different power quality problems. The energy of the detail coefficients, where extracted at different resolution levels, is used to generate the translation invariant feature vector. The term "translation invariant" denotes that the features remain unchanged if the position of the distortion changes.

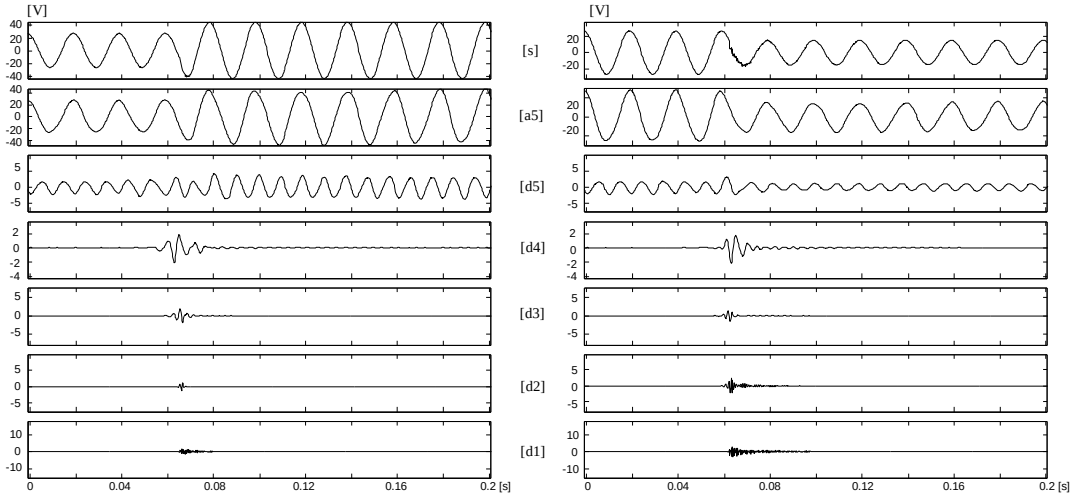
This property is used as a feature to classify different fault classes. The process is shown in Figure 5.5 with Module I and Module II. In the first step the three channel current waveforms are decomposed into different resolution levels by module I. In the second step the distribution of the signal energy for each detail version at different resolution levels is computed by Module II. So, the energy of the signals at different frequency ranges are decomposed and represented, which gives an idea about the frequency content of the signal and is used as a feature to classify different fault classes.

5.3.3 Some Other Important Features

In addition to the features extracted by modules one and two, other discriminative features like voltage signal power and statistical information is obtained by Module III as shown in Figure 5.5.

The voltage waveforms for "A-g" fault and "ABC-g" fault at bus-4 with a fault inception angle of 30^{o} are given in Figure 5.6a and Figure 5.6b, respectively. The two distorted signals belong to V_c channel. Figure 5.6a shows a 50% swell disturbance and Figure 5.6b shows a 50% sag disturbance. These are slow varying disturbances. For both of the disturbances, the abrupt change in the magnitude of signals when fault occurs is seen at scales *1*, *2*, *3*, and *4*. The rapid oscillation disturbances (high frequency) in voltage sag in Figure 5.6b are seen in scales 1 and 2. The change in the magnitude of the signals is best seen in scales 4 and 5. Although these two waveforms belong to two different disturbance classes, their detail components have similar characteristics. And it is difficult to separate single-phase to ground faults and three-phase to ground faults, since the difference between the two fault classes are the sag and swell disturbances. Besides that, the MSD technique is not adequate in detecting slowly varying events like voltage sags, swells, and outages

owing to the poor time resolution at low frequency [16]. This limitation of the MSD can be overcome by tracking the voltage signal power, which is its mean square value [16].



(a) (b)

Figure 5.6 The five-level MSD analysis with Daubechies-10. (a) The voltage swell disturbance signal. (b) The voltage sag disturbance signal

5.3.4 The Feature vector

In the previous paragraphs the functions of the modules in Figure 5.5 were described. In this section the feature vector, which is obtained at the last step of data pre-processing in Figure 5.5 will be explained.

The distribution system shown in Figure 5.1, is simulated using EMTDC/PSCAD simulation software for the purpose of generating line current waveforms and line-to-ground voltage waveforms under various fault conditions. A database of three channel line currents and three channel line-to-ground voltages is built up for various types of faults at different locations and fault inception angles. Then, a data set of six channel waveforms is created for further processing by the data pre-processor. One channel consists of a 10 cycle signal generated at an equivalent sampling rate of 5 KHz (fundamental frequency of 50 Hz) with 1001 sample points lasting 200 msec. Thus, the six-channel data set is a data matrix of (1001,6).

The characteristic information over six-channel current and voltage samples is extracted by the data pre-processor shown in Figure 5.5. The six-channel data set is

then reduced to a feature vector of a small set with 27 parameters. Accordingly, the event feature vector parameters are as follows:

i. The maximum modulus current for each three channel before the fault occurs,

ii. The maximum modulus current for each three channel immediately after the fault occurs,

iii. Energy of each current channel over the 0.078-0.156 kHz band range,

iv. Energy of each current channel over the 0.156-0.312 kHz band range,

v. Energy of each current channel over the 0.312-0.625 kHz band range,

vi. Energy of each current channel over the 0.625-1.250 kHz band range,

vii. Energy of each current channel over the 1.250-2.500 kHz band range,

viii. The signal power for each three channel voltage waveforms,

ix. The energy for each three channel voltage waveforms,

As described in Section 5.3.2, the standard deviation of the DWT coefficients at the resolution levels of five-level MSD serves as a representative of the current signal energy partitioning and hence to aid in classification task as it is used in [9] and [18] as a feature to classify different power quality problems. The standard deviation of five-level detail coefficients of each current signal channel is computed, yielding fifteen parameters.

The magnitude change in the current waveforms is used as a feature to aid in classification of fault classes. This yields six parameters for three cannel current waveforms.

To detect and separate slowly varying events like voltage sags and swells, the voltage signal power is computed for three channels as it was used in the [16] to classify power quality disturbances. The calculated mean square value for three channel voltage waveforms yields three parameters for feature vector.

Furthermore to enhance the competence of the classifier system and to extract more relevant information the standard deviation of voltage channels is computed, which yields three parameters for the feature vector.

Altogether, the feature extraction task is a combination of the "wavelet multi-resolution analysis" technique and the "descriptive statistics" to extract characteristic information from the raw data set and produce a reduced data set of feature vector for being input to the neural network.

In the following paragraph the structure of input vectors will be given and the detailed parameters will be described.

As described above, a feature vector consists of 27 parameters, which will be given to the input layer of the neural network. The structure of a feature vector is as follows:

$$FV = [f_1; f_2; f_3; f_4; f_5; f_6; f_7;...; f_{27}] \tag{5. 2}$$

Here the parameters f_1, f_2 correspond to maximum modulus value before and immediately after the fault for I_a; parameters f_3, f_4 correspond to maximum modulus value before and immediately after the fault for I_b; parameters f_5, f_6 correspond to maximum modulus value before and immediately after the fault for I_c. The parameters $f_7, f_8, f_9, f_{10}, f_{11}$ correspond to standard deviation for five-level MSD sub-bands for I_a. Similarly the parameters $f_{12}, f_{13}, f_{14}, f_{15}, f_{16}$ correspond to standard deviation for five-level MSD sub-bands for I_b; and the parameters $f_{17}, f_{18}, f_{19}, f_{20}, f_{21}$ correspond to standard deviation for five-level MSD sub-bands for I_c. The parameters f_{22}, f_{23}, f_{24} correspond to mean square value of absolute value of V_a, V_b and V_c; finally the parameters f_{25}, f_{26}, f_{27} correspond to standard deviation V_a, V_b and V_c.

The discriminative features of the input vector can be obtained by the neural classifier, which has the property to solve non-linear problems. All of the

components in the input vector give important features of each fault classes and consequently classification with an appropriate algorithm can give adequate results.

By repeatedly executing EMTDC data sets and reducing the resulting samples to *27* element vectors, the data pre-processor in Figure 5.5 collect training sets for the neural classifier. In the next section the neural classification algorithm and the classification results for the complex fault (short-circuit) classification problem under various system and fault conditions will be explained.

5.4 Adaptive Pattern Classification

5.4.1 Introduction

As mentioned in Section 5.2, the fault classification scheme in this thesis is defined as a multi-class problem with ten types of faults (A-g, B-g, C-g, A-B, A-C, B-C, AB-g, AC-g, BC-g, ABC-g).

A literature search shows that most of the ANN studies for fault classification are based on multilayer, feed-forward nets. In the case of the typical supervised back-propagation (BP) network, sets of associated input-output pairs are presented to the ANN that learns a model of the mapping between input and output. However, training of a BP network is very time consuming, needs very large training sets, and easily gets stuck on local minima. Furthermore, it can be difficult to retrain the ANN with new training data. Therefore it may not be sufficient for the task of fault classification.

Another approach for the ANN application for fault classification is using data self-organization obtained through the use of unsupervised learning. Here the task of the classification network is to cluster the faults into separate classes. So, it is a pattern recognition problem. Self-organization refers to the specific learning method without external examples. This is also called unsupervised learning. Given a set of input patterns, neighboring processing units (neurons) in a self-organizing net develop into detectors of specific categories of patterns. So, each local neuron-group acts as a decoder for the inputs [5]. After the learning phase through unsupervised learning, then the ANN is ready for the classification task, where the features

selected from the input data are assigned to individual classes. Although a self-organizing map is equipped to perform the role of classification, it is recommended in literature [5,34] that for best performance it should be accompanied with a supervised learning scheme. Computation of the self-organizing map (feature map) may be viewed as the first of two stages for adaptively solving a pattern classification problem. The second stage is provided by learning vector quantization, which performs a mechanism for the final fine-tuning of a feature map. The combination of a self-organizing map and a supervised learning scheme forms an adaptive pattern classification that is hybrid in nature [5].

In this thesis, a self-organizing map (SOM), with Kohonen's learning algorithm [34,39] and learning vector quantization [LVQ] technique [34] is implemented into the fault classification study. The SOM is intended to discover significant patterns or features from a set of feature vectors obtained by the data preprocessor, as demonstrated in Figure 5.5. SOM obtains the information hidden in high dimensional data that is otherwise difficult to interpret. The SOM converts the complex nonlinear relationship between high-dimensional data into a simple geometric relationship on a low-dimensional display. So it is a vector quantization technique: it compresses the information, while preserving the most important topological relationship of the primary data elements. The SOM is especially suitable for data analysis because it has important visualization properties. It creates a set of prototype vectors representing the data set and carries out a topology preserving projection of the prototypes from *d*-dimensional input space onto a low-dimensional grid. This ordered grid can be used as a convenient visualization surface for showing different features of the SOM and the data, i.e. the cluster structure. The visualization of high dimensional data and discovery of categories is known as exploratory data analysis [40]. It is emphasized in [34] that the map is only intended to visualize topological relationships of signals. The maps should not be used for pattern recognition or other decision processes, because it is possible to increase the recognition accuracy by a significant amount if the maps are fine tuned, i.e. by the learning vector quantization algorithms.

After the hidden patterns in a set of feature vectors are discovered and initial classification is performed by the SOM, the LVQ technique is applied to improve the quality of the classifier. LVQ, developed by Kohonen, is a technique based on a

supervised learning algorithm. The purpose of LVQ is to group a set of related input signals into a finite number of categories based on similarity of the input signals, thereby fine-tuning the initial map, with the number of such categories predetermined by the SOM.

The whole fault classification scheme with the hybrid neural network structure is demonstrated in Figure 5.7. The hybrid neural network forms a two-level classification approach. The whole feature extraction-classification system forms a pyramid, where the number of patterns and connections decrease. The primary benefit of the two-level approach is the reduction of the computational cost.

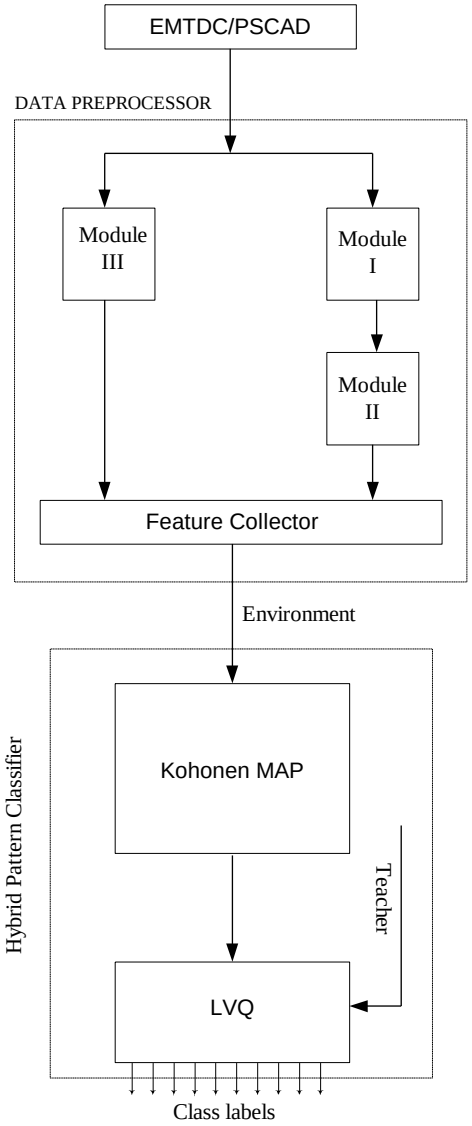


Figure 5.7 The Adaptive Pattern Classification

The major blocks of the system in Figure 5.7 are:

i. Data preprocessor that extracts the feature vectors from the raw six-channel signals as described in Section 5.3.1 and Figure 5.5.

ii. Self-Organizing Map: Unsupervised layer that clusters the data vectors taken from "data preprocessor" to separate clusters.

iii. Learning Vector Quantization: Supervised layer carrying out the classification.

5.4.1.1 Input-Output Structures

The training data set for the hybrid classification scheme in Figure 5.7 is generated by EMTDC/PSCAD simulation software. It is aimed to have a classifier capable of recognizing ten classes of system faults. A three-phase power system, shown in Figure 5.1, was chosen for the purpose of generating line currents and line-to-ground voltages under different fault conditions as described in Section 5.2. A set of *350* cases were generated by changing fault type, fault inception angle, and fault location. For each fault condition the raw data is preprocessed by the three modules in Figure 5.5 and then *350* input vectors with *27* variables were generated. Thus, there are a total of *27* input units, which include the invariant parameters obtained by the data preprocessor demonstrated in Figure 5.5. All of the different cases were then divided into two sets, one to be used for neural-network training and the other for testing. The training set consists of *250* training examples (*25* examples per class) with ten fault classes, five fault locations, and five inception angles in the range of $0^\circ \sim 120^\circ$. The test set consists of *100* examples (*10* examples per class) with ten fault classes, five fault locations, and two inception angles 150° and 180°. Verification of training results is performed so that the ANN is first tested with training patterns, which were used in training, then with samples, which were not used in training.

An important point related to most of the neural network models, is about the choice of input data set. It would often be absurd to use primary signal elements, such as temporal samples of current and voltage waveforms for the components of input vector directly. It may not be possible to achieve any invariance in perception unless the primary information is first transformed, using various convolutions with, i.e.

wavelet transforms or other nonlinear functionals of the signals, as components of input vector [34]. Which particular choice of functionals should be used for preprocessing in fault classification is described in Section 5.3. So, it is generally necessary to use some kind of preprocessing to extract a set of invariant features for the components of the input vector.

The normalization of the data can be thought of the second phase of data preprocessing. Normalization is not necessary in principle, but it may improve numerical accuracy because the resulting vectors then tend to have the same dynamic range. Normalizing the variables of an input vector is important so that none of them has an overwhelming influence on the training result. Since the SOM algorithm in this thesis uses Euclidean metric to measure distances between vectors, scaling of variables is of special importance. If one variable has values in the range of (*0,…,100*) and another in the range of (*0,…,1*) the former will almost dominate the map organization because of its greater impact on the distances measured.

The normalization method used in this thesis is based on the "logarithmic transformation" [40]. This is useful if the values of the variable are exponentially distributed with a lot of small values, and increasingly smaller number of big values as it is the case in this thesis. This transformation gives more resolution to the low end of that vector component. The logarithmic transformation is a non-linear transformation:

$$x' = \ln(x - \min(x) + 1) \tag{5. 3}$$

where "*ln*" is the natural logarithm. The resulting values will be non-negative [40]. In the entire train and test cases of fault classification scheme in this thesis, "logarithmic transformation" is used to normalize the input vectors.

As it is indicated above, the data set for neural classifier consists of a database of *350* vectors with *27* invariant variables. Thus, the input vector for neural classifier has *27* input units. Moreover, since the problem is a multi-class problem with ten-classes, the hybrid system has *10* output units that each belonging to one fault class. For the three-phase distribution system studied here, ten types of faults are to be classified by the neural network (Table 5.1).

Table 5.1 – Fault classifier categories

Category	Fault Type	Label
1	Phase A to Ground Fault	AG
2	Phase B to Ground Fault	BG
3	Phase C to Ground Fault	CG
4	Phase A to Phase B Fault	AB
5	Phase B to Phase C Fault	BC
6	Phase A to Phase C Fault	AC
7	Phase A-B to Ground Fault	ABG
8	Phase B-C to Ground Fault	BCG
9	Phase A-C to Ground Fault	ACG
10	Phase A-B-C to Ground Fault	ABCG

In the next sections the structure of the "Adaptive Pattern Classifier" will be given. Both of the models in hybrid system will be described in separate subsections, and the classifier performance will be demonstrated with an example simulation. Also a comparison of all the simulation results with varying parameters in both of the SOM and LVQ will be demonstrated.

5.4.2 SOFM Algorithm

The Self-Organizing Feature Map (SOFM) is a neural network motivated by the biological nervous system. The various cortices in the cell mass of the animal brain contain many kinds of maps such that a particular location of the neural response in a map often directly corresponds to a specific modality and quality of sensory signal. Similarly, in an artificial Kohonen's feature map the nodes are specifically tuned to various input signal patterns or classes of patterns through self-organization. Kohonen's analogy is that both in the animal brain and the artificial feature map, the

internal representation of information is organized spatially and the maps are formed adaptively through unsupervised learning [41].

The SOFM is a vector quantization method that places the prototype vectors on a regular low-dimensional grid in an ordered fashion. The purpose of Kohonen's self-organizing feature map is to capture the topology and probability distribution of input data. The main idea is to store a large set of input vectors by finding a smaller set of prototypes, so as to provide a good approximation to the original input space. The basis of the idea is rooted in vector quantization theory, which produces an approximation to probability density function of the vectorial input variable using a finite number of codebook vectors. Once the codebook is chosen, the approximation of input variable involves finding the reference vector that is closest to the input variable. After the SOM algorithm converges, the feature map computed by the SOFM algorithm displays important statistical characteristics of the input data [34].

The SOM consists of neurons organized on a regular low-dimensional grid. Each neuron is a d-dimensional weight vector (prototype vector, codebook vector, model vector) where d is equal to the dimension of the input vectors. The neurons are connected to adjacent neurons by neighborhood relation that dictates the topology or structure of the map. The SOM can be thought of as a net, which is spread to the data cloud [34]. The SOFM training algorithm moves the weight vectors so that they span across the data cloud and so that the map is organized. The feature map computed by the SOFM algorithm is topologically ordered in the sense that the spatial location of a neuron in the lattice corresponds to a particular domain or feature of input patterns [5]. The topological ordering property is that when the synaptic weight vector of a winning neuron moves toward the input vector, it also has the effect of moving the synaptic weight vectors of the closest neurons along with the winning neuron so that the map becomes topologically ordered.

The SOFM reflects variations in the statistics of the input distribution. Regions in the input space from which sample vectors are drawn with a high probability of occurrence are mapped onto larger domains of the output space, and therefore with better resolution than regions in the input space from which sample vectors are drawn with a low probability of occurrence. This is the "density matching" property of the feature map [5]. So, this property implies that if a particular region of the input

space contains frequently occurring stimuli, it will be represented by a larger area in the feature map than a region of the input space where the stimuli occur less frequently.

The topology preserving mapping algorithm of Kohonen is an iterative process for training a class of neural networks [34]. The learning procedure is unsupervised or self organizing and is used to train a network of units or neurons that are arranged in a low-dimensional sheet-like structure. In this thesis, a two-dimensional structure for the network is used (as shown in Figure 5.8), but also the application of one or more dimensional structure is possible.

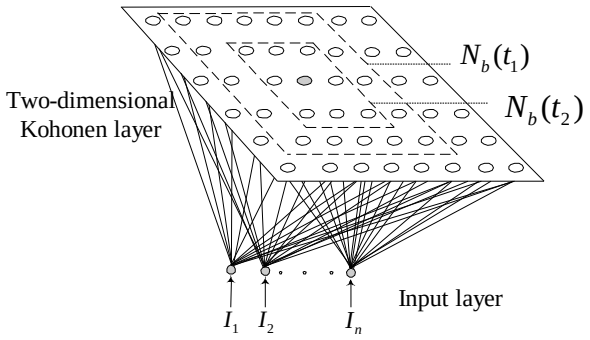


Figure 5.8 A two-dimensional Kohonen layer with $N_b(t)$, topological neighborhood where $t_1 < t_2$

In the SOFM, there are four issues that need to be decided in the beginning of the algorithm. They are the number of neurons, dimensions of the map grid, map lattice and shape.

The number of neurons should usually be selected as big as possible, with the neighborhood size controlling the smoothness and generalization of the mapping. However, as the size of the map increases, the training phase becomes computationally heavy. In this thesis, the default number of neurons is selected according to an empiric formula of Kohonen [42]. The default number of neurons is selected to be $(5*\sqrt{(n)})$ where n is the number of training samples. In the simulations, the number of map units is between *80-110*.

In the simulations in this thesis, a sheet shaped map is used according to [42], however it may be possible to use toroid and cylinder shapes according to data structure. For the selected sheet shaped maps in this thesis, side length along one

dimension is longer than the other, e.g. (8,10), so that the map can orientate itself properly.

The local topology type of the map can be selected to be either rectangular or hexagonal (see Section 4.4.2.1 Figure 4.18 for different lattice structures). The important difference between rectangular and hexagonal lattices is that in the former all *8* neighbors of a neuron are at the same distance and in the latter *6* neighbors of a neuron are at the same distance, as shown in Figure 4.18. In this thesis, both of the lattice structures are used in simulations to obtain different neighborhood relations.

Another important issue that has to be determined before the SOFM algorithm proceeded is the neighborhood function, which was described in Section 4.4.2.1. The neighborhood function determines how strongly the neurons are connected to each other. The simplest neighborhood function is the bubble function, which is constant over the whole neighborhood of the winner unit and zero elsewhere. Generally, at each learning step, all the cells within N_b are updated, whereas cells outside N_b are left intact. N_b is the neighborhood set of the best matching unit (BMU), which is denoted by b. The neighborhood N_b is centered on that unit for which the best match with an input pattern is found according to Equation 4.30 given in Section 4.4.2.1. The width or radius of N_b can be time-variable, in fact, for good ordering of map, the N_b should be large in the beginning of the training process (the "ordering phase"), and then shrink with time so that toward the end of the process (the "convergence phase"), N_b should include only the closest neighbors of the winning neuron b [34]. Also, it has been demonstrated that in biological neurons, there is lateral interaction, which means that when a neuron is firing, it excites other neurons in its closest neighborhood more than those farther away from it [41]. To incorporate this feature in the algorithm, usually the neighborhood around the winning neuron is made to decay gradually [34]. One of the typical choices is to let the amplitude of the topological neighborhood (centered on the winning neuron) decay according to Gaussian function, which was described in Section 4.4.2.1 with Equation 4.32. In this way, the weight update is the strongest for the winning neuron, and becomes weaker with increasing lateral distance.

In this thesis, the neighborhood function is chosen to be an "Epanechicov function" according to Kohonen [42]. Also "gaussian", "cut-gaussian", and "bubble" functions [42] were used as neighborhood function, but the performance obtained by "epanechicov function" was better then former. So, it is used in the whole simulations. The "epanechicov" neighborhood function is

$$h_{bi}(t) = \max\left\{0, 1-(\sigma_t - d_{bi})^2\right\} \tag{5. 4}$$

where $\sigma(t)$ is the width of the topological neighborhood function at time t, $d_{bi} = \|r_b - r_i\|$ is the distance between map units b and i on the map grid as shown in Figure 5.8. Also, $\sigma(t)$ is given in Section 4.4.2.1 in Equation 4.33.

The training of SOM is usually performed in two phases. In the first phase, relatively large initial learning rate and neighborhood radius are used. In the second phase both learning rate and neighborhood radius are small right from the beginning. This procedure corresponds to first tuning the SOM approximately to the space as the input data and then fine-tuning the map as described in Section 4.4.2.1.

The SOFM algorithm can be implemented in two ways: as sequential and batch training algorithms. In the traditional sequential training, samples are presented to the map one at a time, and the algorithm gradually moves the weight vectors towards them as described in Section 4.4.1.2. In the batch training, the data set is presented to the SOM as a whole, and the new weight vectors are weighted averages of the data vectors.

In an attempt to accelerate the computation of the SOM, the batch algorithm [42] is used in this thesis. In batch map principle, the whole training set is gone through at once and only after this the map is updated with the net effect of the samples. Actually, the updating is done by replacing the prototype vector with a weighted average over the samples, where the weighting factors are the neighborhood function values. In each training step, the data set is partitioned according to the Voronoi regions of the map weight vectors, i.e. each data vector belongs to the data set of the map unit to which it is closest; this set is called the Voronoi set. After this, the new weight vectors are calculated as:

$$m_i(t+1) = \frac{\sum_{j=1}^{n} h_{ib}(t) x_j}{\sum_{j=1}^{n} h_{ib}(t)} \qquad (5.5)$$

where $b = \arg\min_k \left\{ \left\| x_j - m_k \right\| \right\}$ is the index of the BMU of data sample x_j. The new weight vector is a weighted average of the data samples, where the weight of each data sample is the neighborhood function value $h_{ib}(t)$ at its BMU *"b"* [38]. This is the way batch algorithm has been implemented in this thesis.

In batch version of the SOFM algorithm the order in which the input patterns are presented to the network has no effect on the final form of the feature map, and there is no need for a learning-rate schedule. But the algorithm still requires the use of a neighborhood function [5].

Finally, the "batch algorithm" can be summarized as follows [42]:

If all observation samples $\{x(t) = 1,2,\Lambda\ ,N\}$ are available prior to computations, they can be applied as a batch in the SOFM algorithm, whereby the following computational scheme can be used:

i. Initialization: Choose random values for the initial weight vectors $m_i(0)$ (model vectors). The only restriction is that the $m_i(0)$ be different for $\{j = 1,2,\Lambda\ ,l\}$, where l is the number of neurons in the lattice.

ii. Similarity: For each map unit *i*, collect a list of all those observation samples *x(t),* whose most similar model vector belongs to the neighborhood set N_i of node *i*.

iii. Updating: Take for each new model vector the mean over the respective list. And, update the m_i according to Equation 5.7.

iv. Continuation: Continue with steps 2 and 3 until the m_i can be regarded as stationary, or until the end of the iteration time.

In the following paragraphs of this section simulation results for the training data set described in Section 5.4.1.1 will be demonstrated by SOFM algorithm. Also the

topological relationships of fault classes will be visualized and analyzed by the visualization properties of the map.

A SOM was trained using the batch-training algorithm for the training set described in Section 5.4.1.1. The data set consists of 250 example cases with 27 variables. The data set is normalized according to "logarithmic transformation". The neighborhood function is selected to be "epanechicov" function, which was given in Equation 5.4. A sheet shaped map with hexagonal lattice structure is used in this simulation. The default number of map units is $((5*\sqrt{(n)})=80)$ according to the empiric formula of Kohonen [42]. It is selected to be *110* in this simulation to further compare the results with default size of the map. The selected map size is (10,11) so, side length along one dimension is longer than the other that make map orientated properly. The initial radius of the N_b is nine, to make the N_b cover the majority of the neurons in the (10,11) map. Special caution is required in the choice of the initial radius of the $N_b = N_b(0)$. If the initial neighborhood is too small to start with, the map will not be ordered globally. This phenomenon can be avoided by starting with a fairly wide $N_b = N_b(0)$ and letting it shrink with time. In this simulation the radius of N_b is decreased from the value nine (covering the majority of the neurons) to one (covering neuron b and its six neighbors) with $\sigma(t)$ given in Equation 4.33. Before the training, initial values are given to the prototype vectors of the self-organizing map. Properly accomplishing the initialization allows the SOM algorithm to converge faster to an appropriate solution. Typically the map can be initialized by one of the three-initialization procedures [42]:

i. random initialization, where the map weight vectors are initialized with small random values

ii. sample initialization, where the map weight vectors are initialized with random samples drawn from the input data set

iii. linear initialization, where the map weight vectors are initialized in an orderly fashion along the two greatest eigenvectors of the covariance matrix of the training data [42].

In the simulations in this thesis, random and linear initializations have been implemented.

In the SOFM batch algorithm, the training is performed in two phases. In the first phase, relatively large initial neighborhood radius is used, and in the second phase neighborhood radius is small right from the beginning as described above. This procedure corresponds to first tuning the SOM approximately to the same space as the input data and then fine-tuning the SOM approximately to the same space as the input data.

Another important property before training the map is the total training time or the number of training steps (iteration). The number of training steps should be at least ten times the number of map units in the first phase. And the length of second phase is at least four times that of the first phase [42]. In this simulation the number of training steps is selected to be "*1100*" steps in the first phase of SOM algorithm, and "*4400*" steps in the second phase. So totally "*5500*" steps is considered in the simulation.

Each neuron in the SOM has actually two positions: one in the input space (the prototype vector) and another in the output space (on the map grid). Thus SOM is a vector projection method defining a nonlinear projection from the input space to a lower-dimensional output space [42]. In this thesis the input space is a *27*-dimensional vector, and the output space is a *2*-dimensional map. The SOFM gives topological relations of different classes on the *2*-dimensional map by examining the features of the input vector in a nonlinear fashion. These properties are visualized in Figure 5.9 and Figure 5.10.

Figure 5.9a shows the average quantization error (QE) of the map at the second phase of the learning. The average quantization error, denoted by Equation 4.35, is a measure of the quality of the map [42]. It is the average distance from each data vector to its best matching unit (the closest model vector). It is data-dependent: it measures the map in terms of the given training data. The initial value of the QE was "*1.890*" in the beginning of the first phase and "*0.340*" at the beginning of the second phase of training the map. Finally at the end of the training it reached to "*0.1648*". In the first phase, initial neighborhood radius was relatively large (nine). In

the second phase neighborhood radius is small (one) right from the beginning as described above. The iteration time step was 1100 and 4400 for first and second phase respectively. So, in the first phase the value of the QE changed faster than the second phase. Since, the second phase corresponds to fine-tuning the map.

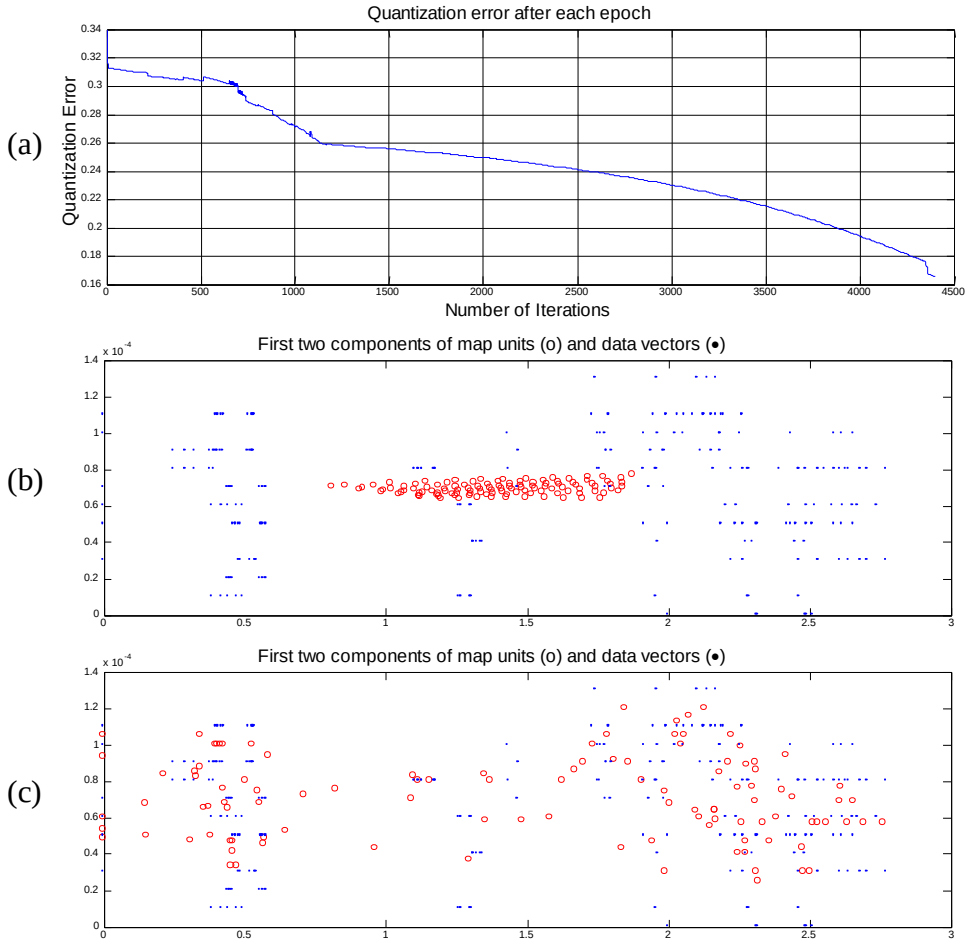


Figure 5.9 (a) The QE in the second phase of SOM algorithm. (b) Initial state of the distribution of prototype vectors on the input space. (c) Final state of the distribution of prototype vectors on the input space

Figure 5.9b and Figure 5.9c shows the distribution of the prototype vectors on the input space. An important issue should be clarified at this point that although the prototype vector and the input vector is a 27-dimensional vector, to be able to plot them on the 2-dimensional space, the first two components of the vectors are selected. Figure 5.9b shows the linear initialization of the prototype vectors of the map. During training, the map organizes itself and folds to the training data as shown in Figure 5.9c. The feature map reflects variations in the statistics of the input distribution. During training the prototype vector most similar to a data vector is

modified so that it is even more similar to it. This way the map learns the position of the data cloud. Also, not only the most similar prototype vector, but also its neighbors on the map are moved towards the data vector. This way the map self-organizes. Another case is that, the region of the input space containing frequently occurring stimuli is represented by a larger area in the feature map than a region of the input space where the stimuli occur less frequently. This reflects the so called "density matching" property of the feature map [5]. So, the density of the prototype vectors assigned to an input region approximates the density of the input occupying this region. In other words, after training has been completed, the map reflects the statistical characteristics of the inputs. And, the prototype vectors tend to be ordered according to their mutual similarity (topology preserving property). This second issue will be clarified in the next two figures.

An initial idea of the number of the clusters in the SOM, as well as their relationships, is usually acquired by visual inspection of the map. For the trained Kohonen map, the properties of the clusters can be further explained by analyzing the weight vectors (prototype vectors) of the neurons in the clusters corresponding to the given data. The most widely used methods for visualizing the cluster structure of the SOM are distance matrix techniques, especially the unified distance matrix [43]. The unified distance matrix (U-matrix) shows distances between prototype vectors of neighboring map units and thus shows the cluster structure of the map..

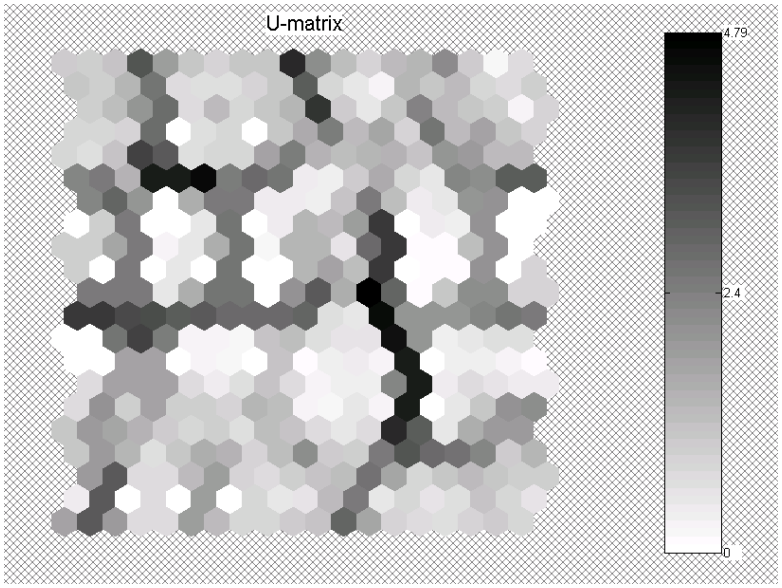


Figure 5.10 U-matrix of the (10,11) SOM of the training data set

Figure 5.10 shows the U-matrix of the SOM of the training dataset in input space. In the U-matrix, dark color indicates large distance between neighboring map units (indicates class borders). Clusters are typically uniform areas of low areas with light color. The colorbar in Figure 5.10 shows the meaning of the light and dark colors with respect to the map. From the U-matrix, one can clearly distinguish several separate areas. There are ten clusters, which were given in Table 5.1. The neurons with the same color belong to the same category. Since the SOFM algorithm gives a rough approximation to the probability density function of the data cloud, the borders between some clusters are not very clear. Also, it is seen from Figure 5.10 that the U-matrix visualization has much more hexagons than the map size (10,11). This is because distances between map units are shown, and not only the distance values at the map units.

Another visualization method is hit histograms [42]. They are formed by taking a data set, finding the BMU of each data sample from the map, and increasing a counter in a map unit each time it is the BMU. The hit histogram shows the distribution of the data set on the map. In Figure 5.11a multiple hit histograms are shown simultaneously to investigate whole training data set using the map. Here the hit histograms of ten data sets are shown with respect to color code on the U-matrix. Here U-mat uses interpolated shading of colors [42]. (Interpolation is a process for estimating values that lie between known data points). The size of the hit histogram determines how many times the corresponding map unit is selected BMU.

It is not easy to visualize the structure of the data and distinguish each of the clusters clearly when the category information is not available as seen in Figure 5.10. Since the category information of the data is available, it is possible to label each unit in the map by the class label of the patterns, which are projected onto the map. As shown in Figure 5.11b, when the class labels are assigned to the map units after training, the map clearly shows that the data is clustered and thereby demonstrates the topology preserving property of the SOFM algorithm.

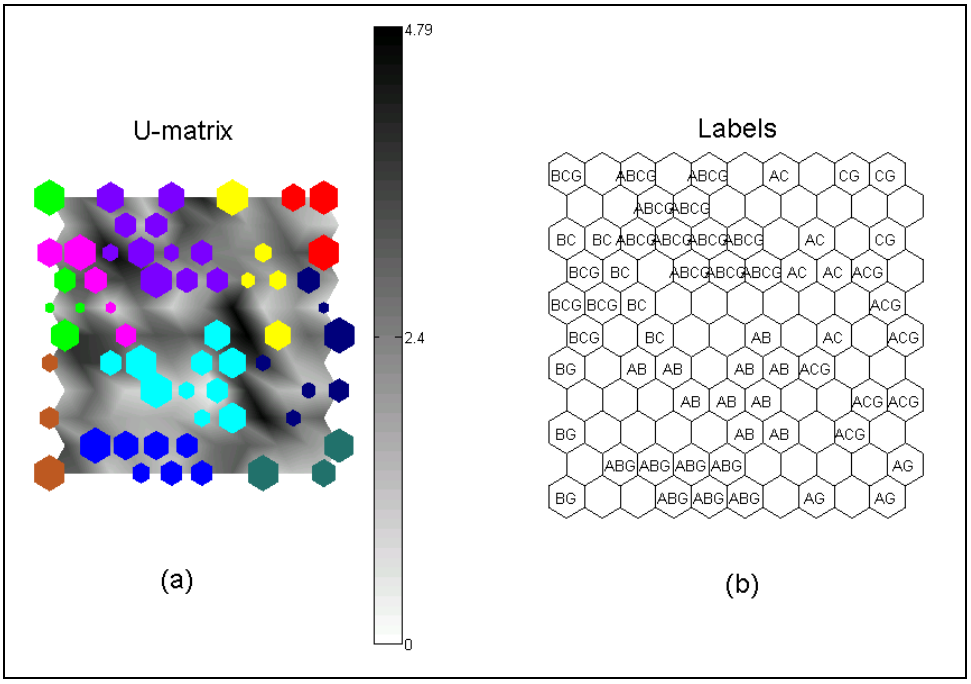


Figure 5.11 (a) The hit histograms on the U-matrix. (b) The labeled SOM

Each node in the feature map is ”labeled” to learn what category of inputs it denotes. This labeling of nodes of a feature map is accomplished by presenting a number of input patterns whose class memberships are known, although these labels were not referred to during the learning. In this simulations *250* input patterns that had been used for training the feature map were presented to the trained map in order to determine the classes denoted by the nodes of the feature map. The class of an input pattern determines the label of the node it activates in the feature map. For example, if an input pattern that represents the “category 5” according to Table 5.1, activates a node in the feature map then this node is labeled to denote this category. If a node of the feature map is activated by the input patterns of different classes, the label of the node is decided by majority voting. In this case all of the map units are given a counter of each label. And when the map unit wins a given data vector, the corresponding counter is increased. After presenting all of the data vectors, the higher counter determines the label of the map unit. If two or more counters are equal, then the map unit is not labeled [34].

By comparing Figure 5.1a and Figure 5.1b it is possible to see the topology preserving property of the map. The map units labeled with the same category are generally clustered together. Since the SOFM algorithm gives a rough approximation

to the probability density function of the data cloud, some of the clusters, especially "category 5" and the "category 8", are not well separable on the feature map.

According to Kohonen [34], if the SOM is to be used as a pattern classifier that the map units are grouped into subsets, each of which correspond to a class of patterns, then the problem becomes a decision process. One should not use the maps as such for pattern recognition or decision processes, because it is possible to increase the recognition accuracy by a significant amount if the maps are fine tuned according to supervised learning algorithms. The SOFM algorithm is intended to approximate input signal values, or their probability density function, by codebook vectors. If the signal sets are to be classified into a number of categories, then several codebook vectors are usually represent each class, and decisions made at class borders will be important but the identity of the codebook vectors within the classes is no longer important. It is possible to define values for the codebook vectors that they directly define the decision borders between the classes. In this thesis "Type One-Learning Vector Quantization (LVQ1) algorithm [34] is used as a supervised classifier that uses class information to move the Voronoi vectors slightly to improve the quality of the classifier decision regions.

5.4.3 Fine-tuning of Map By Type-1 Learning Vector Quantization (LVQ1)

The learning vector quantization (LVQ1) is called a binary output pattern classifier since its output is either zero or one. It is a supervised version of the self-organizing map networks, suitable particularly for pattern recognition problems.

The LVQ1 neural network consists of a single layer of nodes. The weight vectors for these nodes are termed "codebook vectors", since they serve as the reference vectors against which the input is matched. With total lateral inhibition among all nodes, only one node remains active for a given input, providing a binary output. This active node has the codebook vector "closest" to the input sample. Here, the closeness measure is the Euclidean distance, which was explained in Section 4.4.1.2.

To define the optimal placement of the codebook vectors, initial values for them must first be set using randomly initialization or the Self-Organizing Map algorithm. The initial values in the second method roughly correspond to the overall statistical

density function of the input. The next phase is to determine the labels of the codebook vectors, by presenting a number of input vectors with known classification, and assigning the cells to different classes by majority voting, according to the frequency with which each codebook vector is closest to the calibration vectors of a particular class. The detailed description of labeling the map units was described in the previous section.

After the codebook vectors are assigned to each class, and each of them is labeled with the corresponding class symbol, the class regions in the input space are defined by a simple nearest-neighbor comparison of the codebook vectors with the input data vectors. The label of the closest codebook vector defines the classification of the given input data. The LVQ1 algorithm was described in Section 4.4.2.2. The idea in LVQ1 algorithm is to pull the codebook vectors away from the decision borders, to demarcate the class borders more accurately. The training phase of the LVQ1 is said to achieve convergence when the tuned LVQ recognizes input signals with nearly 100% accuracy. The next step is then testing the system with a new set of patterns that were never presented to the pattern classification network.

The "LVQ1" can be summarized as follows [5]:

i. Initialization: Assign random values for the initial weight vectors $m_i(0)$ (model vectors) using randomly initialization or the Self-Organizing Map algorithm.

ii. Calibration: Determine the labels of the codebook vectors, by presenting a number of codebook vectors with known classification

iii. Updating: Update the $m_i = m_i(t)$ according to the rule in Equation 4.36.

iv. Continuation: Continue with step 3 until the end of the iteration time.

It is desirable for the learning rate in Equation 4.36 given in Section 4.4.2.2 to decrease monotonically with the number of iterations. Since LVQ1 is a fine-tuning method, the initial value should be small, i.e. 0.01 or 0.02 and decrease to zero [34]. In this thesis, several functions for the learning rate was used, such as power series, linear, inverse-of-time functions. In all cases, the learning rate function was defined

on the interval [0,1] and was monotonically decreasing with time. In the following paragraph the learning rate functions used in this thesis will be described.

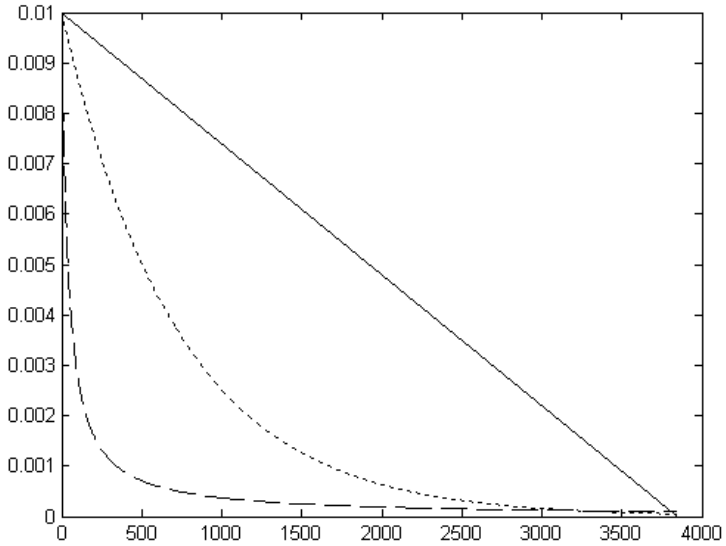


Figure 5.12 Different learning rate functions: linear (solid line), power series (dot-dashed) and inverse-of-time (dashed) functions

The learning rate $\alpha(t)$ is a scalar adaptation gain ($0 < \alpha(t) < 1$), which is decreasing monotonically in time. The learning rate functions are defined as [42]

$$\alpha(t) = \alpha_0(1 - t/T) \tag{5. 6a}$$

$$\alpha(t) = \alpha_0(\alpha/\alpha_0)^{t/T} \tag{5.6b}$$

$$\alpha(t) = \alpha_0(1 + 100t/T) \tag{5.6c}$$

for "linear", "power series", and "inverse-of-time" functions respectively. Here α_0 is the initial value of the learning rate, α is the final value of the learning rate, T is the training length and t is the time steps. These are illustrated in Figure 5.12. According to Kohonen, [34] the initial learning rate is taken between *0.01* or *0.02*. The final value of the learning rate is approximately zero ($0.00005 \cong 0$). In this thesis all of these functions are used, and since better classification rate is obtained with the "inverse" function, it is used in the whole simulations as the learning rate function.

In the following paragraphs of this section simulation results for the training-test data set described in Section 5.4.1.1 will be described. The codebook vectors obtained by the self-organizing map in the previous section (Section 5.4.2) will be used as the initial values of the codebook vectors in the LVQ1, as described in the first step of the LVQ1 algorithm.

The "inverse function" shown in Figure 5.12, is used as the learning rate function. The initial learning rate is "*0.01"*.

Another important parameter is the number of training steps that determines how many times training sequence is performed. According to Kohonen [42], LVQ1 algorithm may be stopped after a number of steps, that is "*30-50*" times the number of codebook vectors. In this simulation the selected map size was (10,11): there were "*110*" map units (codebook vectors). In this simulation the number of training steps is selected to be "*35*mu=3850*", where "*mu*" is the number of codebook vectors.

Before proceeding to the next step and analyzing the test results, the "adaptive pattern classification" network will be illustrated.

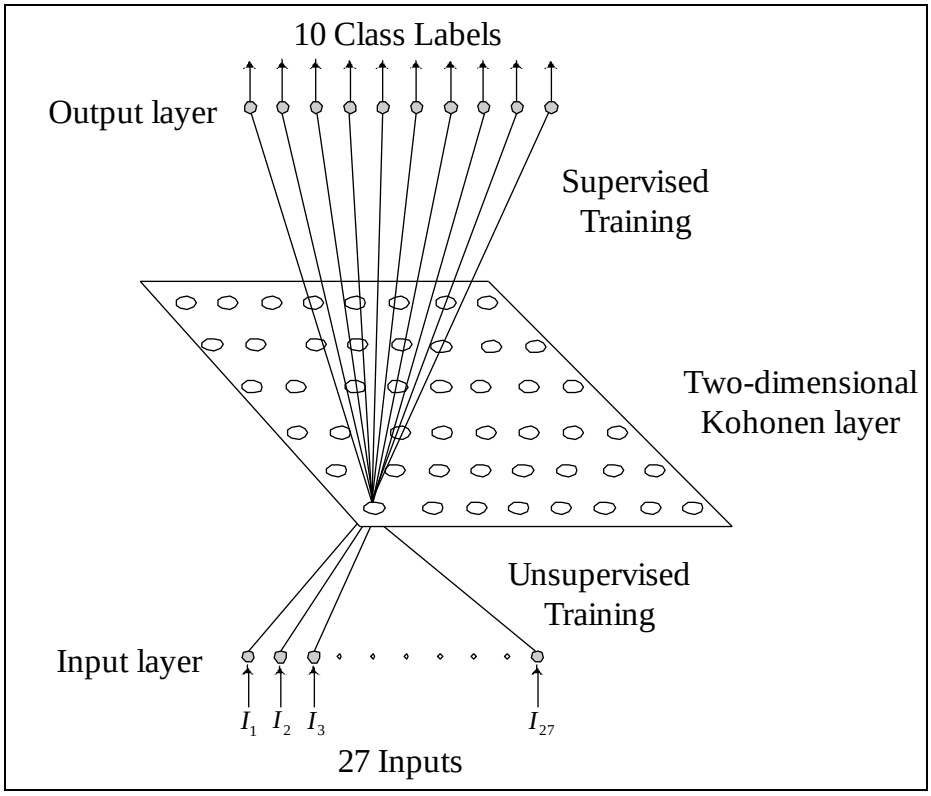


Figure 5.13 Adaptive pattern classification with combined unsupervised-supervised learning

Since the problem is a multiclass problem with ten types of faults as given in the Table 5.1, and learning vector quantization is a binary output pattern classifier; there is ten units in the output layer of the LVQ1 structure.

The combined unsupervised/supervised training architecture is illustrated in Figure 5.13. The pattern classification task can be described in two steps with respect to Figure 5.13. In pattern classification, the first and most important step is feature extraction. The techniques to extract relevant information from the raw data were described in Section 5.3. Here the SOFM can be thought as a second feature extractor to further examine the data and class characteristics. So, the objective of the SOFM is to select a reasonably small set of features, in which the essential information of the input data is concentrated. The self-organizing map is suitable for the task of feature extraction, as described in Section 5.4.1 and Section 5.4.2. The second step in pattern classification is the actual classification, where the features selected from the input data are assigned to individual classes. It is recommended by Kohonen [34] to combine a supervised learning scheme, especially the LVQ for the second stage of classification with the SOFM for pattern recognition problems. The combination of self-organizing map and a supervised learning scheme forms "Adaptive Pattern Classification", which has hybrid architecture [5].

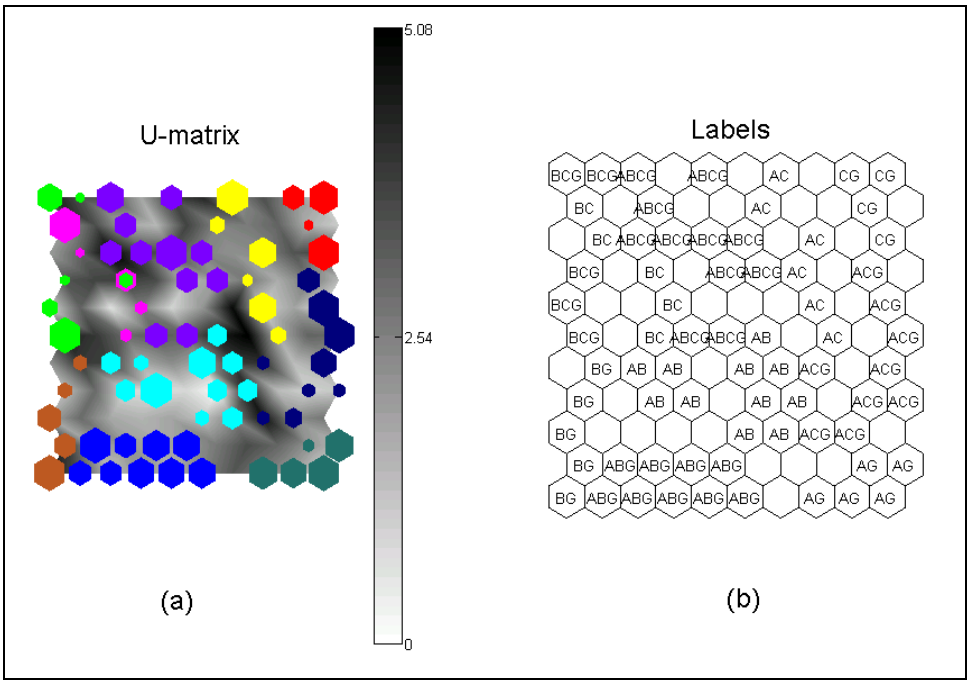


Figure 5.14 (a) The hit histograms on the U-matrix. (b) The labeled SOM

Figure 5.14 shows the hit histograms on the U-matrix and the labeled map after fine-tuning the SOFM (described in Section 5.4.2) with LVQ1 algorithm. As seen from the Figure 5.14a the final state of the map gives a better approximation to the input data, and the cluster borders are represented better than Figure 5.11a. Also, as it is seen in Figure 5.14b of the map units not labeled in the first phase of adaptive pattern classification are now labeled with the class labels, and thus the clusters are represented better than the case in Figure 5.11b. Then another measure of the map is the count of labeled map units or cells. In rough training of the SOM the number of map units that are not labeled is "*51*", and after fine-tuning the SOM with LVQ1 the number of the map units that are not labeled decreased to "*40*". Also when the results of the labeled maps in Figure 5.11b and Figure 5.14b are compared it can be seen that "category 5" and the "category 8" are better represented and the border between this two clusters is better demarcated. Moreover, some of the map units' labels are changed because they were incorrectly labeled by the SOFM algorithm in the first phase of pattern classification. These results reflect the properties of the LVQ1 algorithm. In order to clarify this point, it would be better to describe the main idea of the LVQ1. LVQ employs supervised competitive learning, based on the "winner takes all" strategy, to find the output node, which best matches the input pattern [5]. If the input pattern's class differs from the best-matched neuron's class, the best-matched neuron is moved away from the exemplar. If the best matched neuron gives the same class as that of the input pattern, then the best matched neuron is moved closer to the input pattern vector. As a result the borders between the data clusters can be demarcated better, and the map can better approximate the given data, furthermore the initially incorrectly labeled map units can be labeled with the correct class label.

To obtain the classification rate of the pattern classification network, the following simple procedure is followed according to [44]. The data set (unlabeled) to be classified is submitted to the trained prototype vectors. And the labels with respect to nearest neighboring rule of LVQ are assigned to the given input vectors. In other words, the codebook vectors best matching to each input vector gives its label to this input vector. To ensure that the prototypes really represent the "right" classes, the labels given to each vector in the data set is compared with the "actual" labels of each vector in the data set. If the labels of a vector before and after classification is

the same, then the binary output "*1*" is produced. However, if the labels differ, that is the pattern vector is "incorrectly" classified, than a binary output "*0*" is produced. According to this procedure, all of the input vectors in the data set are given a label. And the sum of the incorrectly labeled vectors determine the misclassification rate of the pattern classifier.

To further clarify the measure of classification rate, the procedure is given below:

i. Obtain the unlabeled pattern vector from the data set to be classified.

ii. Label the pattern vector according to nearest neighbor rule of LVQ algorithm.

iii. Compare the identifier of the pattern vector by comparing its given label and actual label.

iv. If the labels are the same then produce "*1*".

v. Else if the labels are different then produce "*0*".

vi. Proceed with the other pattern vectors in the data set from step *2* to *5*.

vii. Count the "errors", which represent the pattern vectors incorrectly labeled.

viii. Find the error percentage of the classifier for the given data set.

For the data set described in Section 5.4.1.1 and the simulation described in this section, the misclassification rate for the training data set is "*1%*" and the misclassification rate for the test data is "*15%*". So the classification performance of the Adaptive pattern classifier is "*99%*" for the training data, and "*85%*" for the test data.

There are several simulations performed in this thesis, and the maximum classification rate is "*92%*" for the test data, which was never given to the "hybrid" pattern classifier during the training phase. In the following section, the results of the simulations with varying parameters will be given and the performance of the hybrid classifier will be described.

5.4.4 Simulation Results

In this thesis it is aimed to design a classifier capable of recognizing ten classes of three-phase system faults. The ten types of faults to be classified in this thesis are described in Table 5.1 in Section 5.4.1.1. The fault classifier model was illustrated and described in Figure 5.7 in Section 5.4.

In this section several simulations with varying parameters related to the hybrid pattern classifier illustrated in Figure 5.7 and Figure 5.13 will be given. The simulations include six cases, which are obtained with the best performance overall the simulations. As the network parameters related to these six cases are explained, the test results are given in a table as well.

The simulations are performed on a platform with the following properties:

- CPU: Pentium III-MMX Processor at 1 GHz.
- Memory: SDRAM with 3x256 MB.
- Operating System: Windows 2000 Professional Edition.
- MATLAB Version 6.0.0.88 (R12).
- Preprocessor: Wavelet Toolbox Version 2.0 (R12).
- Adaptive Pattern Classifier: SOM Toolbox Version 2.0 [42].

Feature detection and extraction with the "wavelet multi-resolution analysis" technique was performed using the "Wavelet Toolbox Version 2.0" in "Matlab 6.0". The classification of ten-fault classes is based on the "SOM Toolbox", which is a freely available Matlab package developed by "Helsinki University of Technology" [42].

The six simulation cases are described in the following paragraphs.

The common properties of the simulations are as follows. The simulations include several map sizes between "80" map units to "110" map units. A sheet shaped map with a local topology of rectangular and hexagonal structure is used for the simulations. The map sizes are (8,10), (9,10), and (11,10). The initial values of the

prototype vectors of SOM are given according to random initialization and linear initialization as described in Section 5.4.2. The neighborhood function for SOFM algorithm is chosen to be an "Epanechicov function" given by Equation 5.4. The SOFM algorithm is implemented in batch training algorithm. The number of training steps for the SOFM algorithm was between "4000" and "5500" iterations, which will be described in more detail in the following paragraphs. For the LVQ1 algorithm the initialization of the codebook vectors is performed according to the final state of the SOFM corresponding to each simulation. The learning rate function for LVQ1 algorithm is selected to be the "inverse-of-time" function given by Equation 5.6c. The initial learning rate is "0.01" for all cases. The number of training steps for the LVQ1 algorithm was between "2800" and "3850" iterations, which will be described in more detail in the following paragraphs. For all simulations the data set described in Section 5.4.1.1 is used. First the hybrid classifier (shown in Figure 5.7 and Figure 5.13 in more detail) is trained with the training set that consists of 250 training examples (25 examples per class). Afterwards it is tested with the test set that consists of 100 training examples (10 examples per class). (The data set was normalized according to "logarithmic transformation" before being used in adaptive pattern classifier). The rated performance of the test data is between "85%" and "92%".

The following six paragraphs describe the parameters of the six simulation cases.

The first simulation case (will be denoted by "SIM_1") consists of a sheet shaped map with a local topology of hexagonal structure of (8,10) map. The initial values of the map units are given according to random initialization. "Epanechicov function" is used as the neighboring function for the SOFM batch algorithm. In the first phase of SOFM algorithm the initial neighborhood radius was seven and in the second phase neighborhood radius is one. So, the N_b is large in the beginning of the training process, and then shrink with time so that toward the end of the process, N_b includes only the closest neighbors of the winning neuron b. N_b is the neighborhood set of the best matching unit (BMU), which is denoted by b. The number of training steps should be at least ten times the number of map units in the first phase. And the length of second phase is at least four times that of the first phase [42]. So, in this respect, the number of training steps is selected to be "800" steps in the first phase of SOM

algorithm, and “3200” steps in the second phase. So totally “4000” steps is considered in the simulation. The initial value of the QE (that was described in the simulation in Section 5.4.2) was “2.415” in the beginning of the first phase and “0.475” at the beginning of the second phase of training the map. Finally at the end of the training it reached to “0.2376”. The quantization error in the second phase of SOM algorithm, initial state of the distribution of prototype vectors on the input space, and final state of the distribution of prototype vectors on the input space are illustrated in the Figure C.1 in Appendix C. As seen from Figure C.1b and Figure C.1c after training has been completed, the map reflects the statistical characteristics of the input space. The feature map tends to imitate the distribution of the data cloud. After adaptation of the map with the SOFM algorithm, it is calibrated according to “majority voting” as described in Section 5.4.2 and Section 5.4.3. The classification task is performed with LVQ1 algorithm as described in Section 5.4.3. In this simulation the number of training steps is selected to be “35*mu=2800”, where “*mu*” is the number of codebook vectors. The classification performance of the Adaptive pattern classifier is “100%” for the training data, and “89%” for the test data. The hit histograms on the U-matrix and the labeled map before and after fine-tuning the SOFM with LVQ1 algorithm are illustrated in the Figure C.2 and Figure C.3 in Appendix C. It is seen that the cluster border demarcation is better approximated after training the SOFM with LVQ1 algorithm. The clusters are better represented on the U-mat with the “hit-histograms”. Since there is some correlation between the cluster labeled with “BC” and the cluster labeled with “BCG”, the separation between them is not very clear on the U-mat, but from the labels and “hit-histograms” it seems that they correspond to two different clusters. Moreover, as seen from Figure C.2b and Figure C.3b the number of missing labels (or the map units not assigned to any of the clusters) is decreased after fine-tuning the feature map with LVQ1. This is because the LVQ1 provides a better approximation to the data cloud. So the map units without cluster labels are now assigned to clusters.

The second simulation case (will be denoted by “SIM_2”) consists of a sheet shaped map with a local topology of rectangular structure of (8,10) map. The initial values of the map units are given according to linear initialization. “Epanechicov function” is used as the neighboring function for the SOFM batch algorithm. In the first phase of SOFM algorithm the initial neighborhood radius was seven and in the second

phase neighborhood radius is one. The number of training steps is selected to be "800" steps in the first phase of SOM algorithm, and "3200" steps in the second phase. So totally "4000" steps is considered in the simulation. The initial value of the QE was "1.893" in the beginning of the first phase and "0.421" at the beginning of the second phase of training the map. Finally at the end of the training it reached to "0.2132". The quantization error in the second phase of SOM algorithm, initial state of the distribution of prototype vectors on the input space, and final state of the distribution of prototype vectors on the input space are illustrated in the Figure C.4 in Appendix C. As seen from Figure C.4b and Figure C.4c after training has been completed, the map reflects the statistical characteristics of the input space. The feature map tends to imitate the distribution of the data cloud. After adaptation of the map with the SOFM algorithm, it is calibrated according to "majority voting". The classification task is performed with LVQ1 algorithm. In this simulation the number of training steps is selected to be "35*mu=2800", where "mu" is the number of codebook vectors. The classification performance of the Adaptive pattern classifier is "99%" for the training data, and "92%" for the test data. The hit histograms on the U-matrix and the labeled map before and after fine-tuning the SOFM with LVQ1 algorithm are illustrated in the Figure C.5 and Figure C.6 in Appendix C. It is seen that the cluster border demarcation is better approximated after training the SOFM with LVQ1 algorithm. The clusters are better represented on the U-mat with the "hit-histograms". Since there is some correlation between the cluster labeled with "AB" and the cluster labeled with "ABG", the separation between them is not very clear on the U-mat, but from the labels and "hit-histograms" it seems that they correspond to two different clusters. Also it is seen from the U-mat plots in Figure C.5a and Figure C.6a that there is some relation between "AC" and "ACG". Moreover, as seen from Figure C.5b and Figure C.6b the number of missing labels (or the map units not assigned to any of the clusters) is decreased after fine-tuning the feature map with LVQ1. This is because the LVQ1 provides a better approximation to the data cloud. So the map units without cluster labels are now assigned to clusters.

The third simulation case (will be denoted by "SIM_3") consists of a sheet shaped map with a local topology of hexagonal structure of (9,10) map. The initial values of the map units are given according to random initialization. "Epanechicov function" is

used as the neighboring function for the SOFM batch algorithm. In the first phase of SOFM algorithm the initial neighborhood radius was eight and in the second phase neighborhood radius is one. The number of training steps is selected to be "900" steps in the first phase of SOM algorithm, and "3600" steps in the second phase. So totally "4500" steps is considered in the simulation. The initial value of the QE was "2.523" in the beginning of the first phase and "0.365" at the beginning of the second phase of training the map. Finally at the end of the training it reached to "0.236". The quantization error in the second phase of SOM algorithm, initial state of the distribution of prototype vectors on the input space, and final state of the distribution of prototype vectors on the input space are illustrated in the Figure C.7 in Appendix C. As seen from Figure C.7b and Figure C.7c after training has been completed, the map reflects the statistical characteristics of the input space. The feature map tends to imitate the distribution of the data cloud. After adaptation of the map with the SOFM algorithm, it is calibrated according to "majority voting". The classification task is performed with LVQ1 algorithm. In this simulation the number of training steps is selected to be "35*mu=3150", where "mu" is the number of codebook vectors. The classification performance of the Adaptive pattern classifier is "100%" for the training data, and "89%" for the test data. The hit histograms on the U-matrix and the labeled map before and after fine-tuning the SOFM with LVQ1 algorithm are illustrated in the Figure C.8 and Figure C.9 in Appendix C. It is seen that the cluster border demarcation is better approximated after training the SOFM with LVQ1 algorithm. The clusters are better represented on the U-mat with the "hit-histograms". Since there is some correlation between the cluster labeled with "BC" and the cluster labeled with "BCG", the separation between them is not very clear on the U-mat, but from the labels and "hit-histograms" it seems that they correspond to two different clusters. Also it is seen from the U-mat plots in Figure C.8a and Figure C.9a that there is some relation between "AC" and "ACG". Moreover, as seen from Figure C.8b and Figure C.9b the number of missing labels (or the map units not assigned to any of the clusters) is decreased after fine-tuning the feature map with LVQ1. This is because the LVQ1 provides a better approximation to the data cloud. So the map units without cluster labels are now assigned to clusters.

The forth simulation case (will be denoted by “SIM_4”) consists of a sheet shaped map with a local topology of rectangular structure of (9,10) map. The initial values of the map units are given according to linear initialization. “Epanechicov function” is used as the neighboring function for the SOFM batch algorithm. In the first phase of SOFM algorithm the initial neighborhood radius was eight and in the second phase neighborhood radius is one. The number of training steps is selected to be “900” steps in the first phase of SOM algorithm, and “3600” steps in the second phase. So totally “4500” steps is considered in the simulation. The initial value of the QE was “1.891” in the beginning of the first phase and “0.376” at the beginning of the second phase of training the map. Finally at the end of the training it reached to “0.1691”. The quantization error in the second phase of SOM algorithm, initial state of the distribution of prototype vectors on the input space, and final state of the distribution of prototype vectors on the input space are illustrated in the Figure C.10 in Appendix C. As seen from Figure C.10b and Figure C.10c after training has been completed, the map reflects the statistical characteristics of the input space. The feature map tends to imitate the distribution of the data cloud. After adaptation of the map with the SOFM algorithm, it is calibrated according to “majority voting”. The classification task is performed with LVQ1 algorithm. In this simulation the number of training steps is selected to be “35*mu=3150”, where “mu” is the number of codebook vectors. The classification performance of the Adaptive pattern classifier is “99%” for the training data, and “91%” for the test data. The hit histograms on the U-matrix and the labeled map before and after fine-tuning the SOFM with LVQ1 algorithm are illustrated in the Figure C.11 and Figure C.12 in Appendix C. It is seen that the cluster border demarcation is better approximated after training the SOFM with LVQ1 algorithm. The clusters are better represented on the U-mat with the “hit-histograms”. Since there is some correlation between the cluster labeled with “AB” and the cluster labeled with “ABG”, the separation between them is not very clear on the U-mat, but from the labels and “hit-histograms” it seems that they correspond to two different clusters. Moreover, as seen from Figure C.11b and Figure C.12b the number of missing labels (or the map units not assigned to any of the clusters) is decreased after fine-tuning the feature map with LVQ1. This is because the LVQ1 provides a better approximation to the data cloud. So the map units without cluster labels are now assigned to clusters.

The fifth simulation case (will be denoted by "SIM_5") consists of a sheet shaped map with a local topology of hexagonal structure of (11,10) map. The initial values of the map units are given according to linear initialization. "Epanechicov function" is used as the neighboring function for the SOFM batch algorithm. In the first phase of SOFM algorithm the initial neighborhood radius was nine and in the second phase neighborhood radius is one. The number of training steps is selected to be "1100" steps in the first phase of SOM algorithm, and "4400" steps in the second phase. So totally "5500" steps is considered in the simulation. The initial value of the QE was "1.890" in the beginning of the first phase and "0.34" at the beginning of the second phase of training the map. Finally at the end of the training it reached to "0.1648". The quantization error in the second phase of SOM algorithm, initial state of the distribution of prototype vectors on the input space, and final state of the distribution of prototype vectors on the input space are illustrated in the Figure 5.9. As seen from Figure 5.9b and Figure 5.9c after training has been completed, the map reflects the statistical characteristics of the input space. The feature map tends to imitate the distribution of the data cloud. After adaptation of the map with the SOFM algorithm, it is calibrated according to "majority voting". The classification task is performed with LVQ1 algorithm. In this simulation the number of training steps is selected to be "35*mu=3850", where "mu" is the number of codebook vectors. The classification performance of the Adaptive pattern classifier is "99%" for the training data, and "85%" for the test data. The hit histograms on the U-matrix and the labeled map before and after fine-tuning the SOFM with LVQ1 algorithm are illustrated in the Figure 5.11 and Figure 5.14. It is seen that the cluster border demarcation is better approximated after training the SOFM with LVQ1 algorithm. The clusters are better represented on the U-mat with the "hit-histograms". Since there is some correlation between the cluster labeled with "BC" and the cluster labeled with "BCG", the separation between them is not very clear on the U-mat, but from the labels and "hit-histograms" it seems that they correspond to two different clusters. Moreover, as seen from Figure 5.11b and Figure 5.14b the number of missing labels (or the map units not assigned to any of the clusters) is decreased after fine-tuning the feature map with LVQ1. This is because the LVQ1 provides a better approximation to the data cloud. So the map units without cluster labels are now assigned to clusters.

The last simulation case (will be denoted by “SIM_6”) consists of a sheet shaped map with a local topology of rectangular structure of (11,10) map. The initial values of the map units are given according to random initialization. “Epanechicov function” is used as the neighboring function for the SOFM batch algorithm. In the first phase of SOFM algorithm the initial neighborhood radius was nine and in the second phase neighborhood radius is one. The number of training steps is selected to be “1100” steps in the first phase of SOM algorithm, and “4400” steps in the second phase. So totally “5500” steps is considered in the simulation. The initial value of the QE was “2.294” in the beginning of the first phase and “0.280” at the beginning of the second phase of training the map. Finally at the end of the training it reached to “0.1390”. The quantization error in the second phase of SOM algorithm, initial state of the distribution of prototype vectors on the input space, and final state of the distribution of prototype vectors on the input space are illustrated in the Figure C.13 in Appendix C. As seen from Figure C.13b and Figure C.13c after training has been completed, the map reflects the statistical characteristics of the input space. The feature map tends to imitate the distribution of the data cloud. After adaptation of the map with the SOFM algorithm, it is calibrated according to “majority voting”. The classification task is performed with LVQ1 algorithm. In this simulation the number of training steps is selected to be “35*mu=3850”, where “mu” is the number of codebook vectors. The classification performance of the Adaptive pattern classifier is “100%” for the training data, and “88%” for the test data. The hit histograms on the U-matrix and the labeled map before and after fine-tuning the SOFM with LVQ1 algorithm are illustrated in the Figure C.14 and Figure C.15 in Appendix C. It is seen that the cluster border demarcation is better approximated after training the SOFM with LVQ1 algorithm. The clusters are better represented on the U-mat with the “hit-histograms”. Since there is some correlation between the cluster labeled with “AC” and the cluster labeled with “ACG”, the separation between them is not very clear on the U-mat, but from the labels and “hit-histograms” it seems that they correspond to two different clusters. Moreover, as seen from Figure C.14b and Figure C.15b the number of missing labels (or the map units not assigned to any of the clusters) is decreased after fine-tuning the feature map with LVQ1. This is because the LVQ1 provides a better approximation to the data cloud. So the map units without cluster labels are now assigned to clusters.

"SIM_5"; the fifth simulation was explained in the Section 5.4.2 and Section 5.4.3 in more detail.

In the next paragraph the results of the adaptive pattern classification system for the six simulation cases described above are summarized in a table. The results shown in Table 5.2 were obtained after the hybrid system in Figure 5.7 and Figure 5.13 is trained using *250* different fault patterns as described previously. During the testing, the pattern classifier was presented with *100* new fault patterns. The neural network never saw these patterns, and its task was to classify new patterns based on the previous experience (i.e., using the information learned during the training).

Table 5.2 – Hybrid neural network classification results

Simulation	**Description**	**Classification rate (%)**	
		Training data set	**Test data set**
SIM_1	Random initialized hexagonal structure of (8,10) map	100	89
SIM_2	Linear initialized rectangular structure of (8,10) map	99	92
SIM_3	Random initialized hexagonal structure of (9,10) map	100	89
SIM_4	Linear initialized rectangular structure of (9,10) map	99	91
SIM_5	Linear initialized hexagonal structure of (11,10) map	99	85
SIM_6	Random initialized rectangular structure of (11,10) map	100	88

Classification was based on the following procedure:

The data set (unlabeled) to be classified is submitted to the trained prototype vectors. And the labels with respect to nearest-neighboring-rule of LVQ are assigned to the given input vectors. In other words, the codebook vectors best matching to each input vector gives its label to this input vector. To ensure that the prototypes really represent the "right" classes, the labels given to each vector in the data set is compared with the "actual" labels of each vector in the data set. If the labels of a vector before and after classification is the same, then the binary output "*1*" is produced. However, if the labels differ, that is the pattern vector is "incorrectly" classified, than a binary output "*0*" is produced. According to this procedure, all of the input vectors in the data set are given a label. And the sum of the incorrectly labeled vectors determines the misclassification rate of the pattern classifier.

As the test results in Table 5.2 and the figures in the Appendix C are examined, some properties of the parameters used in the SOFM algorithm and the LVQ1 algorithm is obtained. These properties are described below.

As the number of the units in the self-organizing map is increased, the map become smoother and the topological relations and the clusters can be examined better in the SOFM. So the map can be visualized better.

In this simulation, the best classification value is obtained using the empiric formula of Kohonen [42], which was given in Section 5.4.2. The number of map units is $((5*\sqrt{(n)})=80)$ for this case. In the literature, increasing the number of map units generally improves the classification performance of the map [42]. However the results of the simulations were close to each other as the number of the map units are increased for the cases in Table 5.2. This may be because of the initial states of the weight vectors on the map. It may be possible to obtain some improvements on the performance if some more experiments are done.

Using rectangular map structure generally increases the classification rate of the classifier. For example, in the case of a (8,10) map the result obtained by a rectangular lattice (*92%*) is better than the hexagonal lattice (*89%*). Also in the case of [9 10] map rectangular lattice (*91%*) is better than the hexagonal lattice (*89%*).

Finally in the case of (11,10) map rectangular lattice (*88%*) is better than the hexagonal lattice (*85%*). This may be possible because as opposed to the six neighbors in a hexagonal lattice, the eight neighbors of a neuron are at the same distance (as seen in Figure 4.18 in Section 4.4.2.1). So, the map can organize itself better.

6. CONCLUSION

A Combined Wavelet-ANN based fault classifier has been investigated for electrical distribution systems in this thesis. Ten fault categories have been selected to be identified by using the proposed approach. It is shown that the technique presented correctly recognizes and discriminates the fault type and faulted phases(s) with a high degree of accuracy for different location and time of occurrence in the simulated model distribution system.

The underlying approach of the proposed classifier is to carry out (preprocessed) waveform recognition in the self-organizing feature map. The SOFM is intended to discover significant patterns or features from a set of feature vectors obtained by the data preprocessor. SOFM obtains the information hidden in high dimensional data that is otherwise difficult to interpret. The test results show that the decision regions between different fault classes are quite clearly defined. A final decision about the fault type is made by combining the information extracted by SOFM with a supervised learning algorithm: type-one learning vector quantization.

The performance of the proposed fault classification technique is comparable and close to the classifiers in the literature. As described in Section 5.4.4, the results shown in Table 5.2 were obtained after the hybrid system in Figure 5.7 and Figure 5.13 is trained using *250* different fault patterns as described previously. During the testing, the pattern classifier was presented with *100* new fault patterns. The rated performance of the test data is between “85%” and “92%”. All the test results presented show that the proposed fault classification technique based on SOM is well suited for fault classification problems. This hybrid method is easy and very promising for fault classification problem.

Furthermore, the classification performance of the combined Wavelet-SOM network may be improved by performing some modifications to the fault classifier. First of all, depending on the computing power of the test platform, using a large (above 100

units) map size may improve the clustering performance of the map since in this case the data and the cluster borders may be better represented by the feature map. Also some more experiments with varying system parameters may be performed to investigate if better performance can be obtained. Moreover, modifying the SOM algorithm with conscience mechanism may improve the performance of the classifier: A problem with competitive learning algorithms is that they sometimes lead to solutions where several nodes of the network remain unchanged. For example, if some region of the input space is more crowded than others and the initial density of weight vectors is too low in this region, specific nodes may be winning the competitions. The Kohonen learning algorithm attempts to overcome this problem by using topological neighborhoods. Although this approach is very effective, some more modifications may be possible to completely alleviate this case. Conscience mechanism is a technique developed for this issue. The idea is that each neuron keeps track of how many times it has won the competition (i.e., how many times its synaptic weight vector has been the neuron closest to the input vector in Euclidean distance). The notion used here is that if a neuron wins too often, it "feels guilty" and therefore pulls itself out of the competition. This method adapts the weights of the winning node only and the learning rate is assumed to be constant [5].

This research has showed that combined Wavelet-ANN technique can be used for the classification of power system short-circuit faults. More work is needed to further explore the other aspects of cluster characteristics and to better classifying each of them. Future work can involve the use of actual recorded field data to verify initial results obtained in this study.

REFERENCES

[1] **Santoso, S., Powers, E.J., Grady, W.M., Parsons, A.J.,** 2000. Power Quality Disturbance Waveform Recognition Using Wavelet-Based Neural Classifier-Part 1: Theoretical Foundation, *IEEE Transactions on Power Delivery*, **15/1**, 222-228, January.

[2] **Kezunovic, M., Rikalo, I.,** 1996. Detect and Classify Faults Using Neural Nets, *IEEE Computer Applications in Power*, October, 43-47.

[3] **Momoh, J.A., Dias, L.G., Laird, D.N.,** 1997. An Implementation of a Hybrid Intelligent tool for Distribution System Fault Diagnosis, *IEEE Transactions on Power Delivery*, **12/2**, 1035-1040, April.

[4] **Anderson, P.M.,** 1995. Analysis of Faulted Power Systems, IEEE Press, USA.

[5] **Haykin, S.**, 1999. Neural Networks: A Comprehensive Foundation, Prentice Hall International. Inc., New Jersey.

[6] **Song, H., Lee, S.,** 1998. A Self-Organizing Neural Tree for Large-Set Pattern Classification, *IEEE Transactions on Neural Networks*, **9/3**, 369-380, May.

[7] **Aggarwal, R.K., Xuan, Q.Y., Johns, A.T., Li, F., Bennett, A.,** 1999. A Novel Approach to Fault Diagnosis in Multicircuit Transmission Lines Using Fuzzy ARTmap Neural Networks, *IEEE Transactions on Neural Networks*, **10/5**, 1214-1221, September.

[8] **Ebron, S., Lubkeman, D.L., White, M.,** 1990. A Neural Network Approach to the Detection of Incipient Faults on Power Distribution Feeders, *IEEE Transactions on Power Delivery*, **5/2**, 905-914, April.

[9] **Gaouda, A.M., El-Saadany, E.F., Salama, M.M.A., Sood, V.K., and Chikhani, A.Y.**, 2001. Monitoring HVDC Systems Using Wavelet Multi-Resolution Analysis, *IEEE Transactions on Power Systems*, ***16*/4,** 662-670, November.

[10] **Ghosh, A.K., Lubkeman, D.L.,** 1995. The Classification of Power System Disturbance Waveforms using a Neural Network Approach, *IEEE Transactions on Power Delivery*, **10/1**, 109-115, January.

[11] **Aggarwal, R.K., Xuan, Q.Y., Dunn, R.W., Johns, A.T., Bennett, A.,** 1999. A Novel Fault Classification Technique for Double-circuit Lines Based on a Combined Unsupervised/Supervised Neural Network, *IEEE Transactions on Power Delivery*, **14/4**, 1250-1256, October.

[12] **Fahmida, N.C., Arevena, J.L., Grady, W.M., Parsons, A.J.,** 1998. A Modular Methodology for Fast Fault Detection and Classification in Power Systems, *IEEE Transactions on Control Systems Technology*, **6/5**, 623-633, September.

[13] **Kandil, N., Sood, V.K., Khorasani, K., Patel, R.V.,** 1992. Fault Identification in AC-DC Transmission System Using Neural Networks, *IEEE Transactions on Power Systems,* **7/2**, 812-819, May.

[14] **Frick, K.L., Starrett, S.K.,** 2000. Classification of Disturbance Characteristics Using a Kohonen Neural Network, *Large Engineering Systems Conference on Power Engineering*, Halifax, Canada, July 2000, 124-128.

[15] **Wang, H., Keerthipala, W.W.L.,** 1998. Fuzzy-Neuro Approach to Fault Classification for Transmission Line Protection, *IEEE Transactions on Power Delivery,* **15/1**, 222-228, January.

[16] **Elmitwally, A., Farghal, S., Kandil, M., Abdelkader, S., and Elkateb, M.**, 2001. Proposed Wavelet-neurofuzzy Combined System for Power Quality Violations Detection and Diagnosis, *IEE Proc.-Gener. Transm. Distrib.,* ***148*/1,** 662-670, January.

[17] **Dash, P.K., Mishra, S., Salama, M.M.A., Liew, A.C.,** 2000. Classification of Power System Disturbances Using a Fuzzy Expert System and a Fourier Linear Combiner, *IEEE Transactions on Power Delivery,* **15/2**, 472-477, April.

[18] **Gaouda, A.M., Salama, M.M.A., Sultan, M.R., and Chikhani, A.Y.**, 1999. Power Quality Detection and Classification Using Wavelet-Multiresolution Signal Decomposition, *IEEE Transactions on Power Delivery, **14**/**4,*** 16-25, October.

[19] **Zhao, W., Song, Y.H., Min, Y.,** 1998. Wavelet Analysis Based Scheme for Fault Detection and Classification in Underground Power Cable Systems, *Electric Power Systems Research,* Elsevier, December 1978, 23-30.

[20] **EMTDC User's manual,** Manitoba HVDC Research Center, Manitoba, Canada.

[21] **Chan, Y.T.,** 1996. Wavelet Basics, Kluver Acedemic Publisher, Norwell, Massachusetts.

[22] **Kaiser, G.,** 1994. A Friendly Guide To Wavelets, Boston: Birkhauser, Cambridge.

[23] **Galli, A.W., Heydt, G.T. and Ribeiro, P.F.**, 1996. Exploring the Power of Wavelets, *IEEE Computer Applications in Power,* October, 37-41.

[24] **Vetterli, M. and Rioul, O.**, 1991. Wavelets and Signal Processing, *IEEE SP Magazine,* October, 14-38.

[25] **Graphs, A.**, 1995. An Introduction to Wavelets, *IEEE Computational Science & Engineering,* Summer, 50-61.

[26] **Rao, R.M. and Bopardikar, A.S.**, 1998. Wavelet Transforms: Introduction to Theory and Applications, Addison-Wesley Longman, Massachusetts.

[27] **Daubechies, I.**, 1992. Ten Lectures On Wavelets, Society for Industrial and Applied Mathematics, Pennsylvania.

[28] **Vetterli, M. and Herley, C.**, 1992. Wavelets and Filter banks: Theory and Design, *IEEE Transactions on Signal Processing*, **40/9**, 2207-2232.

[29] **Mallat, S.**, 1999. A Wavelet Tour Of Signal Processing, Academic Press, Cambridge.

[30] **Rojas, R.**, 1996. Neural Networks: A Systematic Introduction, Springer-Verlag Berlin Heidelberg, Germany.

[31] **Amari, S.**, 1990. Mathematical Foundations of Neurocomputing, *Proceedings of the IEEE*, **78/9**, 54-73, September.

[32] **Sarle, W.S.**, 1997. Neural Network FAQ, part 1 of 7: Introduction, periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html.

[33] **Ripley, B.D.**, 1996. Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge.

[34] **Kangas, J.A., Kohonen, T. and Laaksonen, J.T.**, 1990. Variants of Self-Organizing Maps, *IEEE Transactions on Neural Networks*, **1/1**, 93-99, March.

[35] **Kohonen, T.**, 1990. The Self-Organizing Map, *Proceedings of the IEEE*, ***78*/9,** 74-90, September.

[36] **Bose, N.K. and Garga, A.K.**, 1993. Neural Network design Using Voronoi Diagrams, *IEEE Transactions on Neural Networks*, ***4*/5,** 778-787, September.

[37] **Chu, W.C. and Bose, N.K.**, 1998. Vector Quantization of Neural Networks, *IEEE Transactions on Neural Networks*, ***9*/6,** 1235-1244, November.

[38] **Galli, A.W. and Nielsen, O.M.**, 1999. Wavelet Analysis for Power System Transients, *IEEE Computer Application in Power*, 16-25, January.

[39] **Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., and Saarela, A.**, 2000. Self-Organization of a Massive Document Collection, *IEEE Transactions on Neural Networks*, **11/3**, 574-585, May.

[40] **De, A., Chatterjee, N.**, 2001. Impulse Fault diagnosis in Power Transformers Using Self-Organizing Map and Learning Vector Quantization, *IEE Proc.-Gener. Transm. Distrib.*, ***148*/5,** 397-405, September.

[41] **Gaugh, J.L., Weinberger, N.M., and Lynch, G.**, 1990. Brain Organization and Memory**,** 323-337., Oxford University Press, USA.

[42] **Vesanto, J., Himberg, J., Alhoniemi, E., and Parkankangas, J.**, 2000. SOM Toolbox for Matlab 5, *Helsinki University of Technology*, URL: http://www.cis.hut.fi/projects/somtoolbox/.

[43] **Vesanto, J., Alhoniemi, E.**, 2000. Clustering of the Self-Organizing Map, *IEEE Transactions on Neural Networks*, **11/3**, 586-600, May.

[44] **Nikhil, R.P., Bezdek, J.C., Erik, C., and Tsao, K.**, 1993. Generalized Clustering Networks and Kohonen's Self-Organizing Scheme *IEEE Transactions on Neural Networks*, **4/4**, 549-557, May.

APPENDICES

Appendix A

A 1. Basis Vectors

A vector υ in *V*, vector space, is said to be linear combination of a set of vectors $\upsilon_1, \upsilon_2, \mathrm{K}, \upsilon_n$ in *V* provided there exist constants $a_1, a_2, \mathrm{K}, a_n$ such that

$$\upsilon = \sum_{k=1}^{N} a_k \upsilon_k \tag{A.1}$$

Equation A.1 shows how any vector υ can be written as a linear combination of the basis vectors υ_k and the corresponding coefficients a_k [22].

A set of vectors $\upsilon_1, \upsilon_2, \mathrm{K}, \upsilon_n$ is said to be linearly dependent if there exist constants $a_1, a_2, \mathrm{K}, a_n$ not all zero such that the linear combination

$$\sum_{k=1}^{N} a_k \upsilon_k = 0 \tag{A.2}$$

A set that is not linearly dependent is linearly independent. Linear independence is a prerequisite for a set of vectors to be a basis of a vector space. Another requirement is that every vector in vector space should be expressible as a linear combination of members of this set. In other words, the basis vectors should span the vector space. Therefore, if $b_1, b_2,$ and so on are the basis of a vector space *V*, then a vector υ in *V* can be represented as

$$\upsilon = \sum_{k=1}^{N} a_k b_k \tag{A.3}$$

The number of vectors in any basis of a finite dimensional vector space gives its dimension. An example of basis vectors is the standard basis of R^3 with $b_1 = (1,0,0)$, $b_2 = (0,1,0)$, $b_3 = (0,0,1)$.

This concept, given in terms of vectors, can easily be generalized to functions, by replacing the basis vectors b_k with basis functions $\Phi_k(t)$, and the vector υ with a function $f(t)$. Equation A.3 then becomes

$$f(t) = \sum_{k} a_k \Phi_k(t) \tag{A.4}$$

A 2. Inner Product

An inner product on a vector space *V* is a rule that associates a real number, defined as $\langle u, \upsilon \rangle$ with each pair of vectors u and υ in V.

The following axioms satisfied by the inner product for all vectors u, υ, and ω in *V*:

1. Symmetry: $\langle u,\upsilon\rangle = \langle \upsilon,u\rangle$.
2. Additivity: $\langle u+\upsilon,\omega\rangle = \langle u,\omega\rangle + \langle \upsilon,\omega\rangle$.
3. Homogeneity: $\langle ku,\upsilon\rangle = k\langle u,\upsilon\rangle$, if k is a scalar.
4. Positivity: $\langle \upsilon,\upsilon\rangle \geq 0$ and $\langle \upsilon,\upsilon\rangle = 0$ if and only if v = 0

A vector space with an inner product defined on it is called an inner product space [26]. An example of an inner product space is the space of all polynomials of degree *n*. If $f(t)$ and $g(t)$ are two polynomials in this space, then the inner product is

$$\langle f(t), g(t)\rangle = \int_{-\infty}^{\infty} f(t)g(t)dt \tag{A.5}$$

If *V* is an inner product space and if u is a vector in this space, then the norm of u denoted by $\|u\|$ is defined by

$$\|u\| = \sqrt{\langle u,u\rangle} \tag{A.6}$$

So, the squared norm is an inner product of a vector with itself. In this manner, the inner product operation induces a norm.

The distance between two vectors u and υ in an inner product space is defined as

$$d(u,v) = \|u-\upsilon\| \tag{A.7}$$

Equation A.5 defines the similarity between $f(t)$ and $g(t)$.

A 3. Hilbert Spaces

A Hilbert space is defined as any vector space with an inner product satisfying positivity, homogenity, and symmetry (see Appendix 2) that is; moreover, complete with respect to this norm [22].

Given a set of infinitive sequences of vectors υ_1, υ_2, and so on, in the vector space *V*; this sequence is convergent if there is a vector υ in *V* such that

$$\lim_{n\to\infty} \upsilon_n = \upsilon \tag{A.9}$$

A Cauchy sequence is one that has the property that as $n \to \infty$, successive points tend to be closer together. Mathematically, this property is as follows: given $\varepsilon > 0$ there exists a positive integer n_0 such that for $m,n > n_0$, then $\|\upsilon_n - \upsilon_m\| < \varepsilon$, ($\|\cdot\|$ is the norm defined on *V*). A convergent sequence must be a Cauchy sequence. However, the converse is not true. Every Cauchy sequence is not necessarily convergent because the element to which the sequence tends may not be in *V*. There are certain inner product spaces where every Cauchy sequence does converge to a vector in that space. Such vector spaces are called complete [22].

An example of a Hilbert space is Lebesgue vector space; $L^2(R)$, for one dimensional, measurable, square integrable functions defined on the real values *R*. [26].

A function $f \in L^2(R)$ is said to be supported in an interval $[a,b] \subset R$ if $f(t) = 0$ outside of $[a,b]$. So f satisfies; $f \subset [a,b]$. The set of square integrable functions supported in $[a,b]$ is denoted by $L^2([a,b])$. It is a subspace of $L^2(R)$ and is itself a Hilbert space. Given an arbitrary function $f \in L^2(R)$, f has "compact support" if $f \subset [a,b]$ for some bounded interval $[a,b] \subset R$ [22]. It is common to use the term "compact support" instead of "finite length".

A 4. Orthogonality and Orthonormality

A set of vectors $\{\upsilon_i\}, i = 1,2,\mathrm{K}\,,n$ is orthogonal if

$$\langle \upsilon_i, \upsilon_j \rangle = c\delta_{ij} \tag{A.10}$$

where the Kronecker delta

$$\delta_{ij} = 0,\ i \neq j \tag{A.11}$$

$$\delta_{ij} = 1,\ i = j$$

and c is a constant. It is orthonormal if c=1, in Equation A.10. When two vectors are orthogonal, they have no correlation or common components. The projection of one onto another is zero, and their inner product is zero. When a set of vectors $\{\upsilon_i\}, i = 1,2,\mathrm{K}\,,n$ is said to be orthonormal, they are pair-wise orthogonal to each other, and all have length one [21].

Similarly, a set of functions $\{\Phi_k(t), k = 1,2,3,\mathrm{K}\,,n\}$ is said to be orthonormal if

$$\int_a^b \Phi_k(t)\Phi_l^*(t)dt = c\delta_{kl} \tag{A.12}$$

There may be more than one set of basis functions (or vectors). Among them, the orthonormal basis functions (or vectors) are of particular importance because of the properties they provide in finding the analysis coefficients. The orthonormal bases allow computation of these coefficients in a very simple and straightforward way using the orthonormality property [21].

For orthonormal bases, the coefficient a_k, can be calculated as

$$a_k = \langle f, \Phi_k \rangle = \int f(t) \cdot \Phi_k^*(t)dt \tag{A.13}$$

and the function $f(t)$ can then be reconstructed by Equation A.4 by substituting the a_k coefficients. This yields

$$f(t) = \sum_k a_k \Phi_k(t) = \sum_k \langle f, \Phi_k \rangle \Phi_k(t) \tag{A.14}$$

Orthonormal bases may not be available for every type of applications where a generalized version, biorthogonal bases can be used. The term biorthogonal refers to two different bases that are orthogonal (pair-wise orthogonal) to each other, but each do not form an orthogonal set [21].

In some applications, however, biorthogonal bases also may not be available in which case frames can be used. Frames constitute an important part of wavelet theory [21].

A 5. Frames in Vector Space

When two vectors are orthogonal, they have no correlation or common components. The projection of one onto another is zero, and their inner product is zero. Decomposition of a vector into its components of ortonormal basis vectors is therefore a simple inner product operation as described in Appendix 4.

Let $\{\upsilon_i\}$ be a set of orthonormal vectors that spans the n-dimensional space, then any $n\times 1$ vector g is a linear combination of the υ_i given by

$$g = \sum_{i=1}^{n} \langle g, \upsilon_i \rangle \upsilon_i \tag{A.15}$$

If $\{\upsilon_i\}$ is not orthonormal (or orthogonal), g can still be expressed as a linear combination of $\{\upsilon_i\}$, but the coefficients of υ_i are no longer simple inner products of $\langle g, \upsilon_i \rangle$. Note that basis vectors need not be orthonormal; they can even be linearly dependent and hence redundant. The only requirement is that they span the vector space so that any vector can be represented in terms of them. The theory of frames is a generalization of the orthonormal decomposition principle and gives a representation of an $m\times 1$ vector as

$$g = \sum_{i=1}^{n} \langle g, \upsilon_i \rangle \tilde{\upsilon}_i \quad , \; n \geq m \tag{A.16}$$

This is similar to Equation A.15, except $\{\upsilon_i\}$ is not necessarily orthonormal and since $n \geq m$, the υ_i basis vectors can be linearly dependent. The $\{\tilde{\upsilon}_i\}$ is called the dual of $\{\upsilon_i\}$. Equation A.16 states that the simple inner product form is still valid in decomposition, but at the expense of introducing dual in the reconstruction [21].

The $\{\upsilon_i\}$ are called elements of a frame, and $\{\tilde{\upsilon}_i\}$ the dual frame of $\{\upsilon_i\}$. For simplicity, υ_i will be assumed as unit vectors. A frame $\{\upsilon_i\}$ is a set of vectors that satisfies, for any nonzero $m\times 1$ vector g,

$$A\|g\|^2 \leq \sum_{i=1}^{n} \left| \langle g, \upsilon_i \rangle \right|^2 \leq B\|g\|^2 \quad , \; n \geq m \tag{A.17}$$

when A and B are constants dependent on $\{\upsilon_i\}$ only, called frame bounds, with $0 < A \leq 1 \leq B < \infty$. They are the highest lower bound and lowest upper bound. The lower bound guarantees that the set $\{\upsilon_i\}$ spans the vector space, i.e., $\{\upsilon_i\}$ is a complete frame, otherwise $\sum_{i=1}^{n} \left| \langle g, \upsilon_i \rangle \right|^2$ can become zero for some $\|g\| \neq 0$.

The theory of frames provides the representation of a set of basis vectors that are not necessarily orthonormal, nor linearly independent. The coefficients are still inner products of the vector with the basis vectors. Reconstruction, however, requires new basis vectors called duals. As long as $\{\upsilon_i\}$ obey Equation A.17, any vector g can be synthesized according to Equation A.16. If $A=B$, then $\tilde{\upsilon}_i = \upsilon_i / A$ and if $A=B=1$, then $\tilde{\upsilon}_i = \upsilon_i$ and $\{\upsilon_i\}$ forms an orthonormal basis. When $A=B$ the frame is said to be tight. Also, if $A>1$ then the frame is redundant and A can be interpreted as a minimum redundancy factor [29].

If $\{\upsilon_i\}$ is a tight frame, then $\tilde{\upsilon}_i = c\upsilon_i$, where c is a constant. However to find the dual frame $\{\tilde{\upsilon}_i\}$ when $\{\upsilon_i\}$ is not tight, suppose $V_{m\times n} = [\upsilon_1 \Lambda \; \upsilon_n]$, and $\tilde{V}_{m\times n} = [\tilde{\upsilon}_1 \Lambda \; \tilde{\upsilon}_n]$. $\{\upsilon_i\}$ obey Equation A.16, so that it follows from Equation A.15 that $g = VV^T g$ for any $m\times 1$ vector g. So V must satisfy

$$\tilde{V}_{m\times n} V_{n\times m}{}^T = I_{m\times m} \tag{A.18}$$

where I is an identity matrix. The solution of Equation A.18 is

$$\tilde{V} = (VV^T)^{-1}V \tag{A.19}$$

The theory of frames is necessary in the decomposition and reconstruction of a function by wavelets.

A last point about frames is the resolution of the identity property which states that; if a transformation is invertible, then the signal energy in the original domain must be equal to, within a constant, the signal energy in the transform domain. The reconstruction of the signal in terms of the basis functions is feasible if energy preservation holds within a constant.

Frames in general satisfy resolution of the identity. In the case of vector transformation, it is easy to verify that if $\{\upsilon_i\}$ is an orthonormal set, then $g = \sum \langle g, \upsilon_i \rangle \upsilon_i$ and $\|g\|^2 = \sum |\langle g, \upsilon_i \rangle|^2$, so resolution of the identity property holds. But when a transform violates resolution of the identity, duals are needed for reconstruction. Finally, from Equation A.16, since $\sum_i \langle g, \upsilon_i \rangle \tilde{\upsilon}_i = \tilde{V}V^T g$, then g can be found as

$$g = \sum_i \langle g, \upsilon_i \rangle \tilde{\upsilon}_i = \sum_i \langle g, \tilde{\upsilon}_i \rangle \upsilon_i \tag{A.20}$$

A 6. Heisenberg Uncertainty Principle

All functions, including windows, in time-frequency analysis obey the uncertainty principle, which states that sharp localizations in time and in frequency are mutually exclusive [22].

If a nonzero function $g(t)$ is small outside a time-interval of length T and its Fourier transform is small outside a frequency band of width Ω, then an inequality of the type $\Omega T \geq c$ must hold for some positive constant $c \approx 1$. The precise value of c depends on how the widths T and Ω of the signal in time and frequency are measured. For instance, given a normalized function g is $\|g\| = 1$; $|g(t)|^2$ is weight distribution of the window in time and $|\hat{g}(w)|^2$ is a weight distribution of the window in frequency. The "centers of gravity" of the window in time and frequency are then

$$t_0 \equiv \int_{-\infty}^{\infty} t|g(t)|^2 dt\,,\ \omega_0 \equiv \int_{-\infty}^{\infty} \omega|\hat{g}(w)|^2 d\omega \tag{A.21}$$

for $g \in L^2(R)$ [22]. A common way of defining T and Ω is as the standard deviations from t_0 and ω_0:

$$T^2 \equiv \int_{-\infty}^{\infty} (t-t_0)^2 |g(t)|^2 dt\,, \quad \Omega^2 \equiv \int_{-\infty}^{\infty} (\omega-\omega_0)^2 |\hat{g}(w)|^2 d\omega \tag{A.22}$$

With these definitions, it can be shown that $4\pi\Omega \cdot T \geq 1$, which is the Heisenberg form of the uncertainty relation. For Gaussian windows, the Heisenberg inequality becomes an equality: $4\pi\Omega \cdot T = 1$ [22].

Heisenberg Uncertainty is an important principle in time-frequency analysis. Time and frequency energy conservations are restricted by the Heisenberg uncertainty principle. The fundamental fact about the T and Ω can be summarized as follows: The precise measurements of time and frequency are fundamentally incomplete, because frequency cannot be measured instantaneously. That is, if a signal has frequency ω_0, then the signal must be observed for at least one period, i.e., for a time

interval $\Delta t \geq 1/\omega_0$. The larger the number of periods for which the signal is observed, the more meaningful it becomes to say that it has frequency ω_0. Hence it is not possible to say with certainty exactly when the signal has a constant frequency [22].

A 7. The Downsampler

The downsampler is also called the sampling rate compressor and the subsampler. Figure A.1 shows an *M*-fold downsampler. It takes an input sequence $x(n)$ and produces an output sequence

$$y(n) = x(Mn) \tag{A.23}$$

where *M* is an integer. In other words, a downsampler by *M* retains only those samples of $x(n)$ that occur at times that are multiples of *M*. Figure A.1b shows how the downsampler acts on a sequence for the case of *M=2* [26].

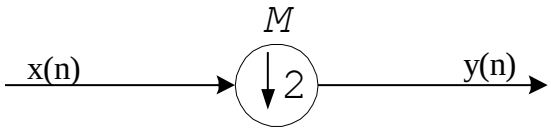
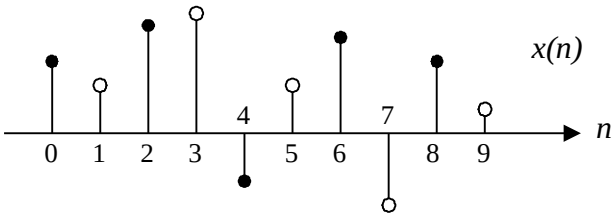
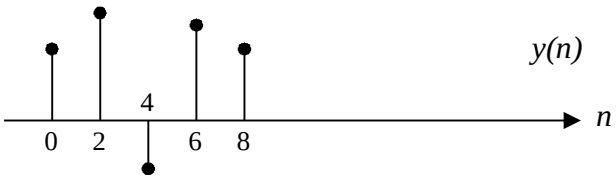


Figure A.1 An M-fold downsampler



(a)



(b)

Figure A.2 Demonstration of a downsampler for the case of *M=2*. (a) Original sequence. (b) The downsampled sequence

A 8. The Upsampler

The upsampler is also called a sampling rate expander or simply an expander. Figure A.3 shows an *L*-fold upsampler. It takes an input sequence $x(n)$ and produces an output sequence

$$y(n) = x(n/L) \quad \text{if n is an integer multiple of } L \qquad \text{(A.24)}$$

$$y(n) = 0 \quad \text{otherwise}$$

So, upsampling by a factor of L involves inserting $L-1$ zeros between consecutive samples of the input sequence. The process of upsampling is demonstrated in Figure A. 4 for $L=2$ [26].

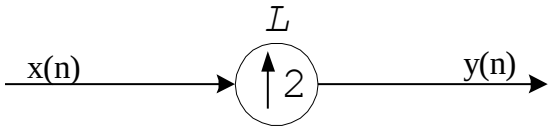
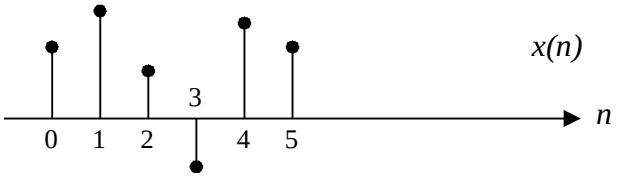
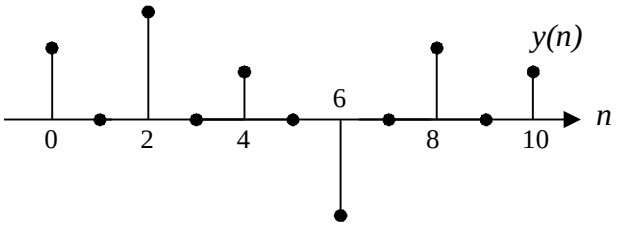


Figure A.3 An L-fold upsampler



(a)



(b)

Figure A.4 Demonstration of an upsampler for the case of $L=2$. (a) Original sequence. (b) The upsampled sequence

Appendix B

B 1. Power System Data of the Simulation System

Table B.1 – Power system parameters of Figure 5.1

Component Name	Component Parameters
Source*	154 kV, 10000 MVASC, 50 Hz, X_1 =j2.3716, X_2 =j2.3716, X_0 =j0.7905,
Transformer TRF 1	100 MVA, 154 Y / 34.5 Y kV, 50 Hz Leakage Inductance= 10.0 %
TRF 2,3,4,5,6	20 MVA, 34.5 Y / 10.5 Δ kV, 50 Hz Leakage Inductance= 10.0 %
Capacitor Bank	7.2 MVAr, Delta connected, C= 34.5 μF
Cables 1,2,3,4	20.3 / 34.5 kV, 3.0 km, 3x(1x240 mm^2) XLPE Cable
Cable 5	20.3 / 34.5 kV, 3.5 km, 3x(1x240 mm^2) XLPE Cable
Loads 1,2,3,4	19 MVA, Cos φ=0.906, Delta Connected, R= 5.259 [Ω], L=0.007806
Load 5	19 MVA, Cos φ=0.707, Delta Connected, R= 4.103 [Ω], L=0.01306

The parameters of the "infinite source" are based on the assumption that the "short-circuit power" is "100" times the maximum rated power on the distribution system, which is "100 MVA". This assumption is based on an empiric formula according to [4]. Also, the zero sequence impedance of the source is based on an assumption that the zero sequence impedance is between (*1/3*) and (*1/6*) of the positive sequence impedance for the source according to [4]. Positive sequence impedance is calculated by $(U^2 / S_k'') = (154^2 / 10000)$ according to [4].

B 2. The Simulation System Layout in PSCAD/EMTDC

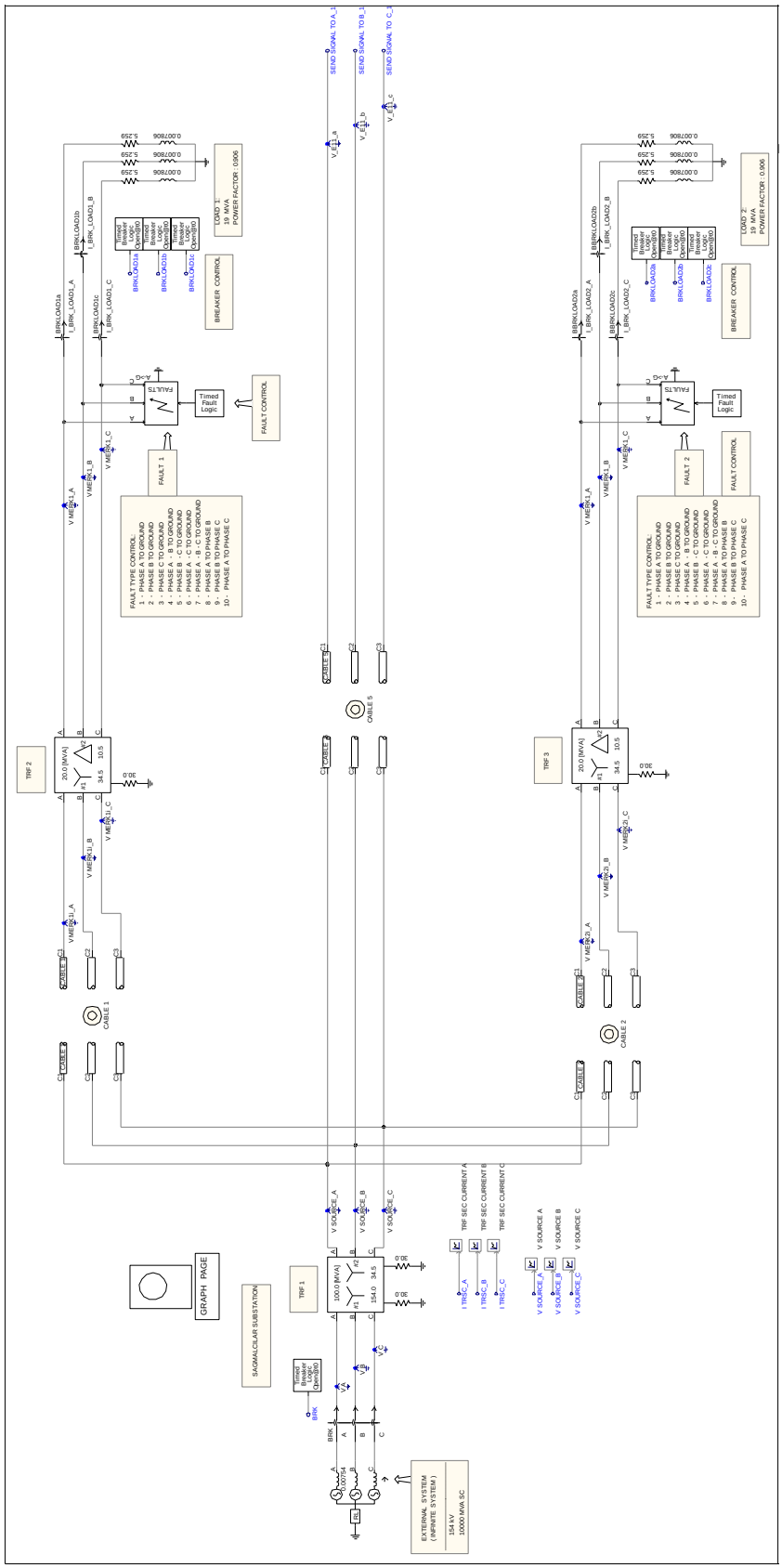


Figure B. 1a The Reduced 34.5 kV Sagmalcılar-Maltepe Substation System Model (part 1)

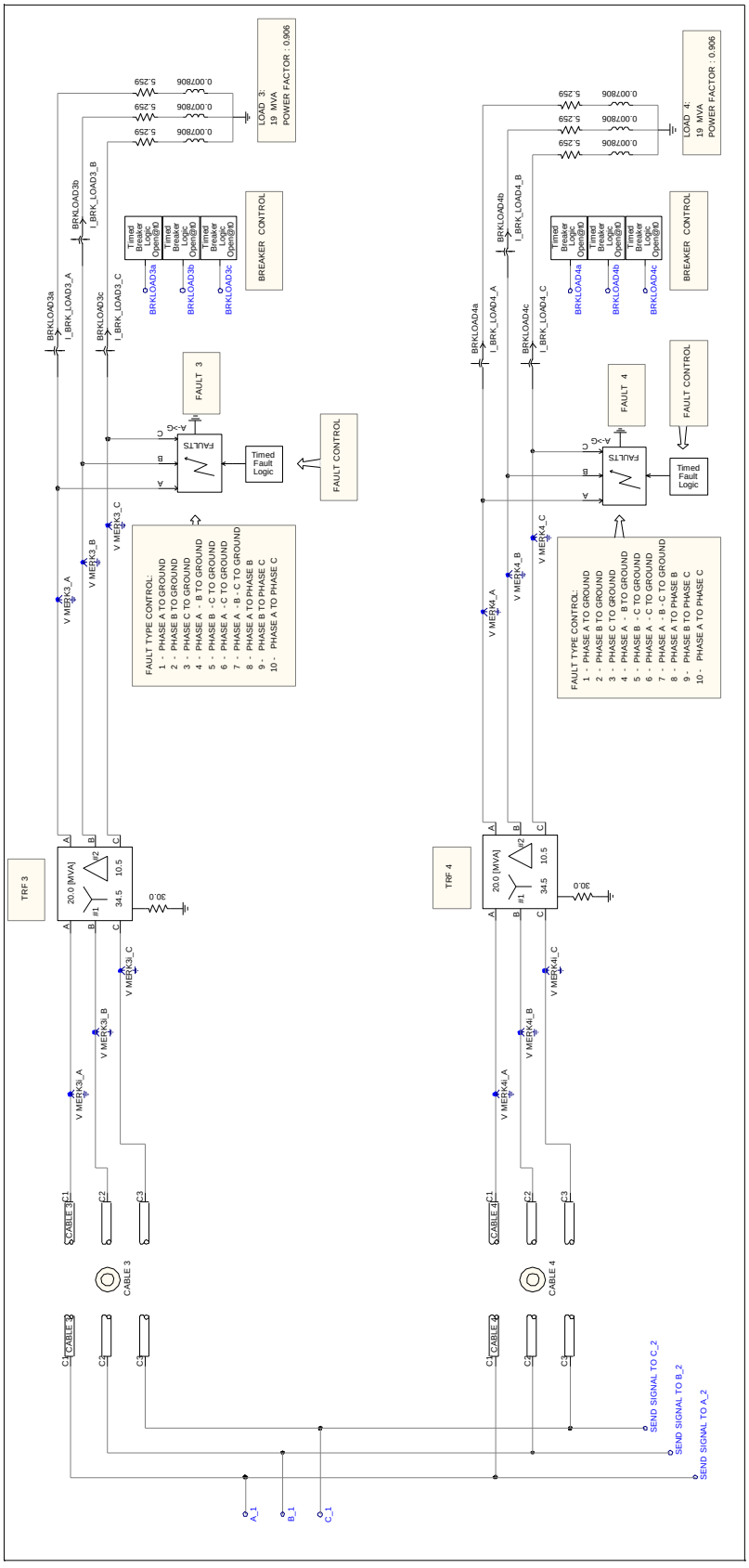
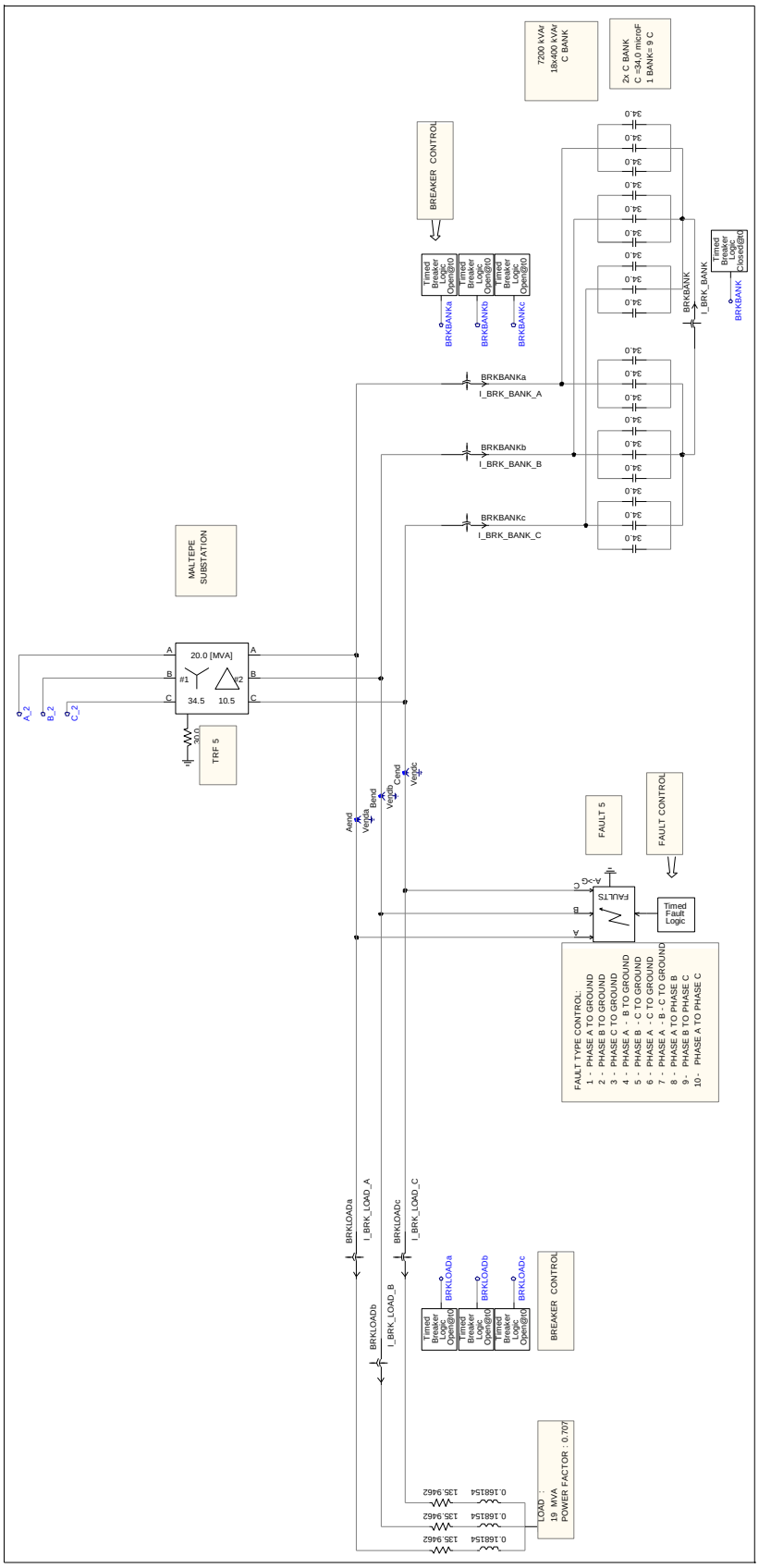
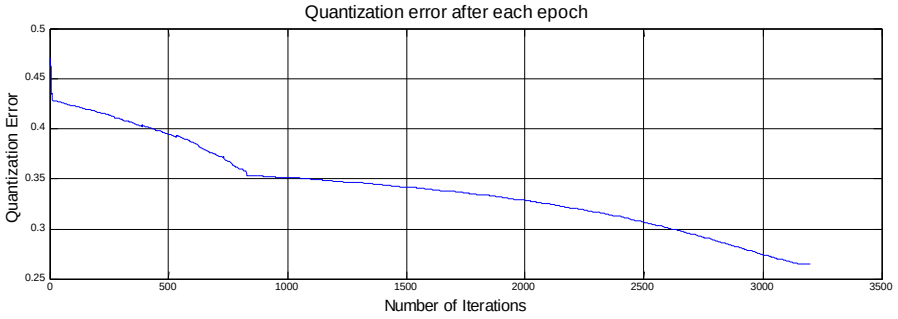


Figure B. 2a The Reduced 34.5 kV Sagmalcılar-Maltepe Substation System Model (part 1)

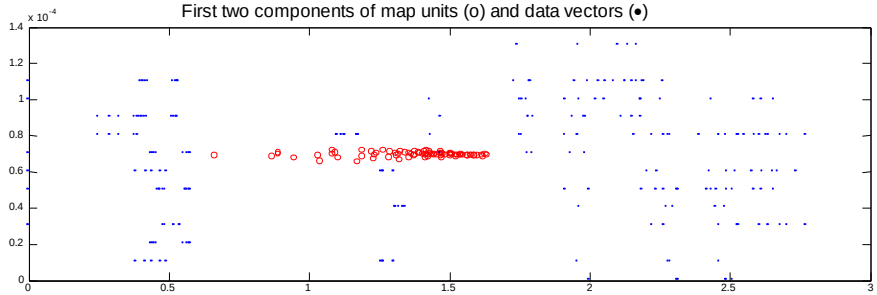

7200 kVAr
18x400 kVAr
C BANK
2x C BANK
C =34.0 microF
1 BANK= 9 C
BREAKER CONTROL
Timed Breaker Logic Open@t0
BRKBANKa
BRKBANKb
BRKBANKc
Timed Breaker Logic Closed@t0
BRKBANK
I_BRK_BANK
BRKBANKa
I_BRK_BANK_A
BRKBANKb
I_BRK_BANK_B
BRKBANKc
I_BRK_BANK_C
MALTEPE SUBSTATION
20.0 [MVA]
#1
#2
34.5
10.5
TRF 5
A_2
B_2
C_2
Aend
Vendb
Bend
Vendc
Cend
Vendc
FAULT 5
FAULT CONTROL
FAULTS
Timed Fault Logic
FAULT TYPE CONTROL:
1 - PHASE A TO GROUND
2 - PHASE B TO GROUND
3 - PHASE C TO GROUND
4 - PHASE A - B TO GROUND
5 - PHASE B - C TO GROUND
6 - PHASE A - C TO GROUND
7 - PHASE A - B - C TO GROUND
8 - PHASE A TO PHASE B
9 - PHASE B TO PHASE C
10 - PHASE A TO PHASE C
BRKLOADa
I_BRK_LOAD_A
BRKLOADb
I_BRK_LOAD_B
BRKLOADc
I_BRK_LOAD_C
BREAKER CONTROL
LOAD : 19 MVA POWER FACTOR : 0.707
0.168154
135.9462

Appendix C

(a)



(b)



(c)

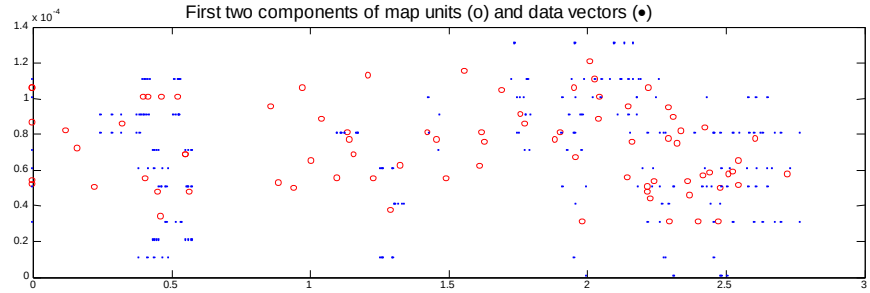


Figure C.1 (a) The QE in the second phase of SOM algorithm (b) Initial state of the distribution of prototype vectors on the input space (c) Final state of the distribution of prototype vectors on the input space (for SIM_1)

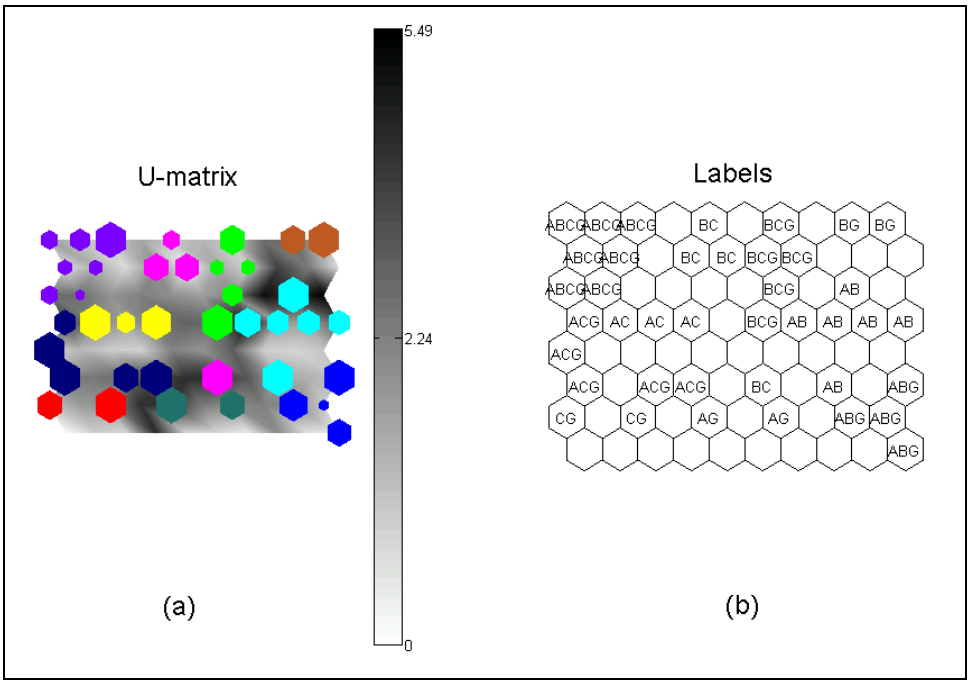


Figure C.2(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 40 for SIM_1)

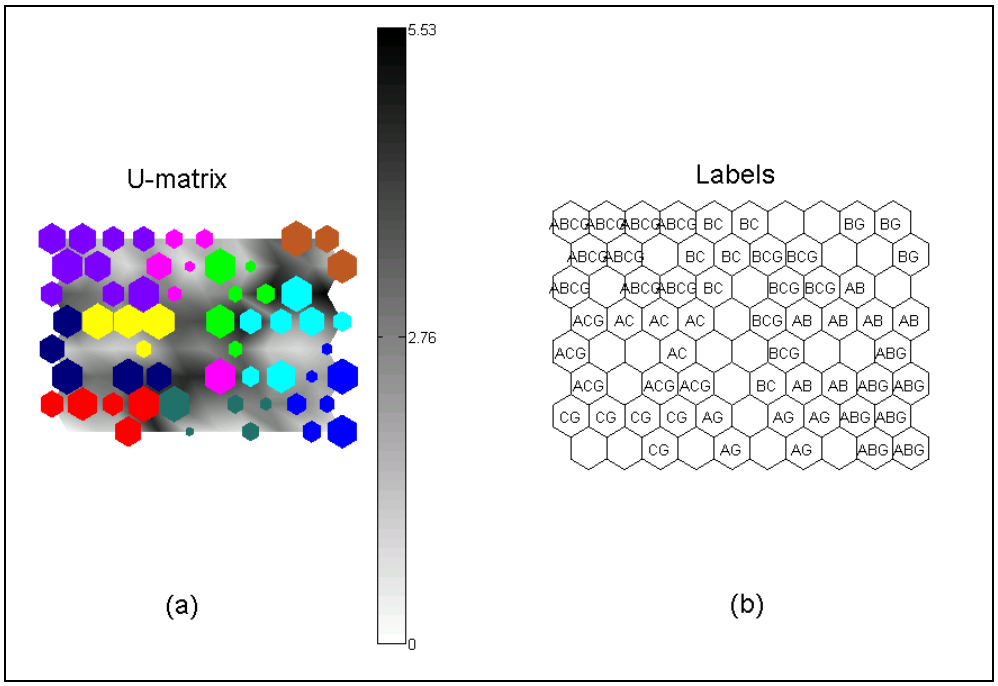


Figure C.3(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 57 for SIM_1)

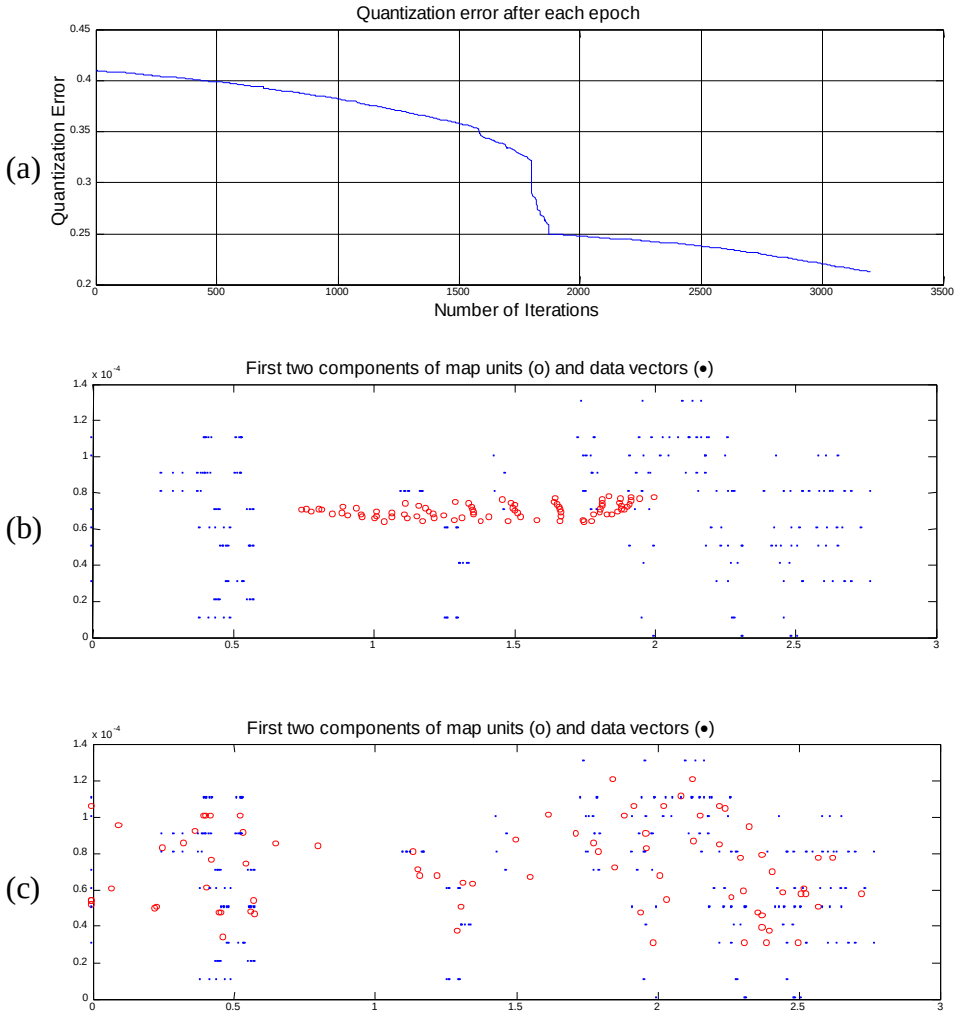


Figure C.4 (a) The QE in the second phase of SOM algorithm. (b) Initial state of the distribution of prototype vectors on the input space. (c) Final state of the distribution of prototype vectors on the input space. (for SIM_2)

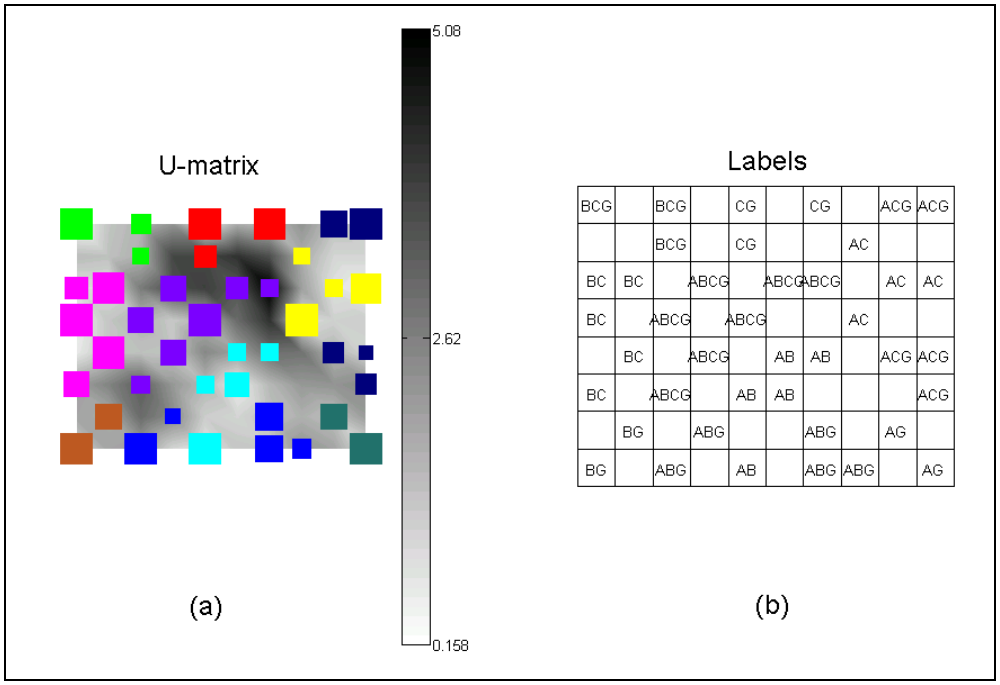


Figure C.4(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 41 for SIM_2)

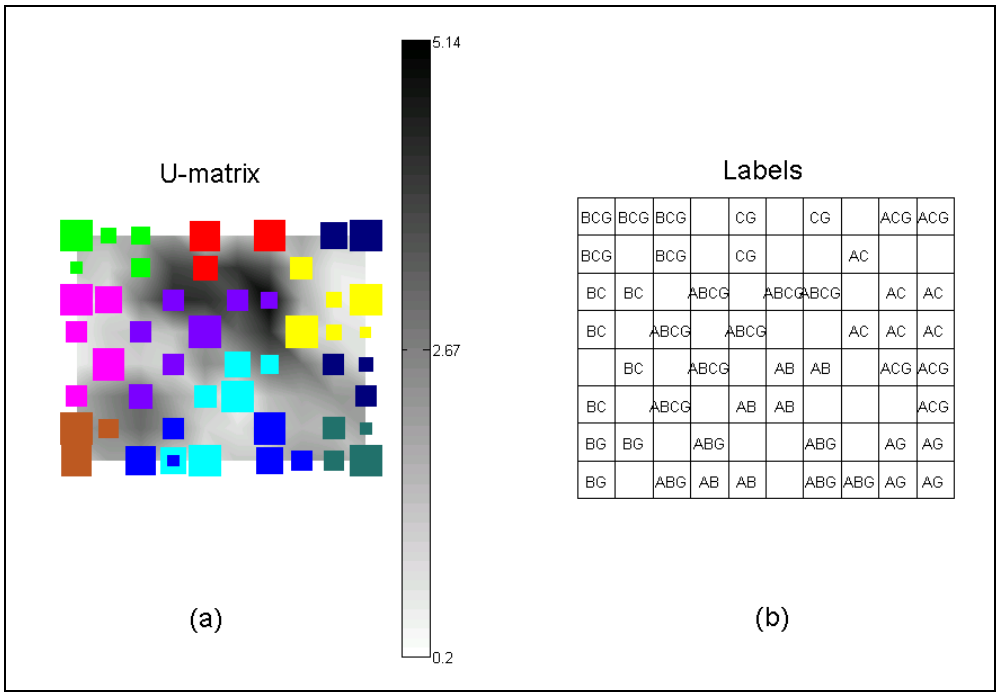
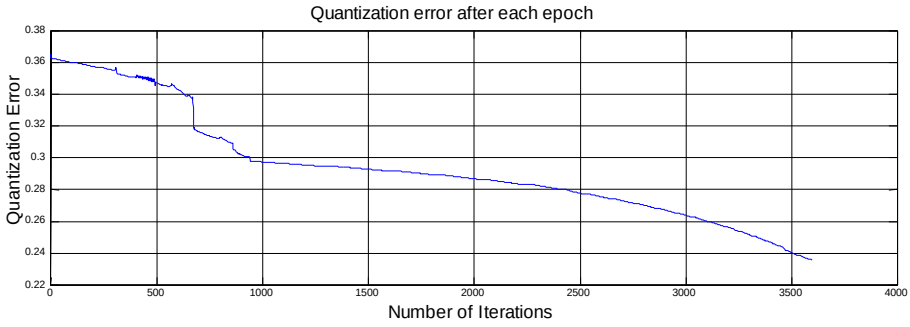
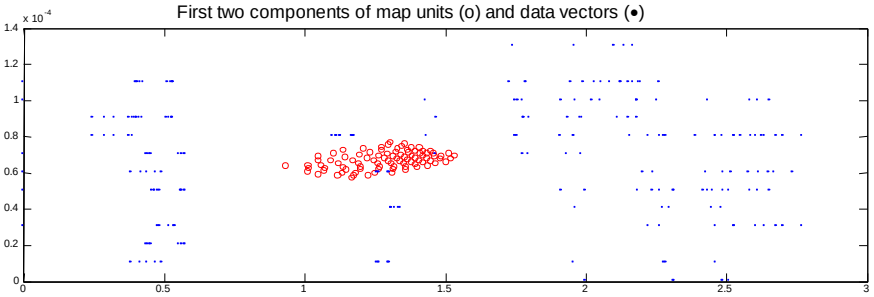


Figure C.5(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 49 for SIM_2)

(a)



(b)



(c)

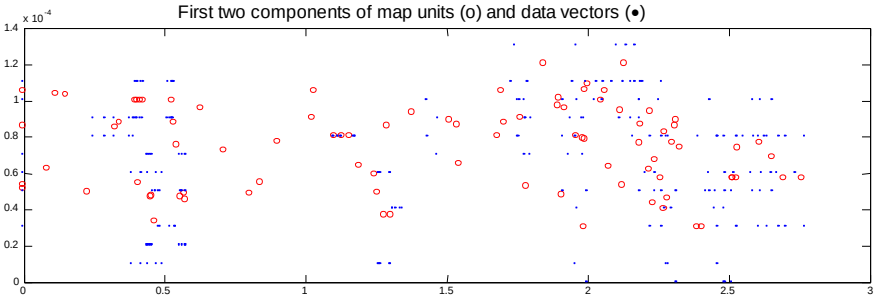


Figure C.7 (a) The QE in the second phase of SOM algorithm. (b) Initial state of the distribution of prototype vectors on the input space. (c) Final state of the distribution of prototype vectors on the input space. (for SIM_3)

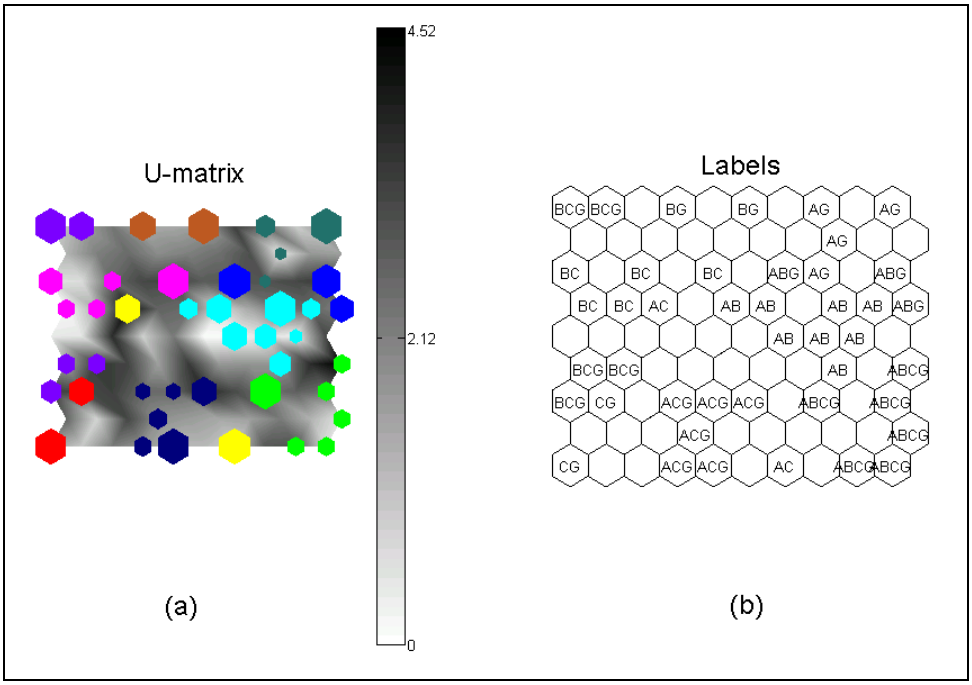


Figure C.6(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 43 for SIM_3)

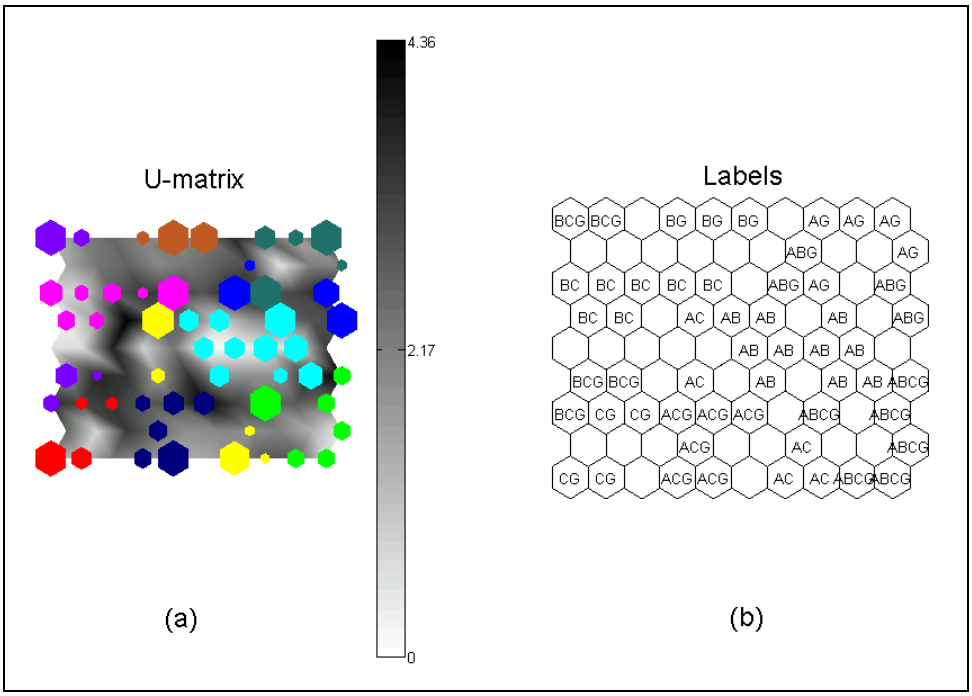
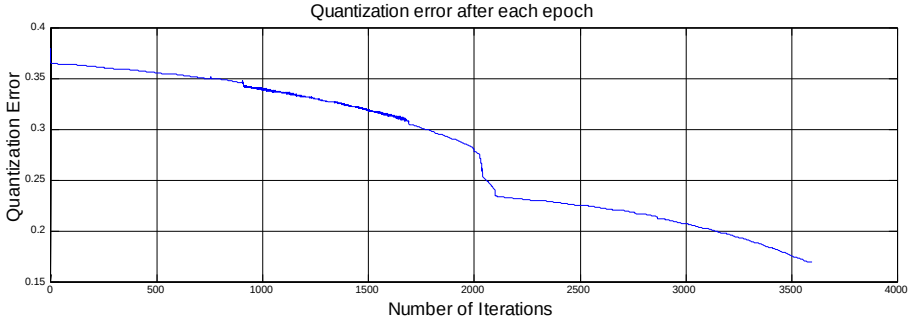
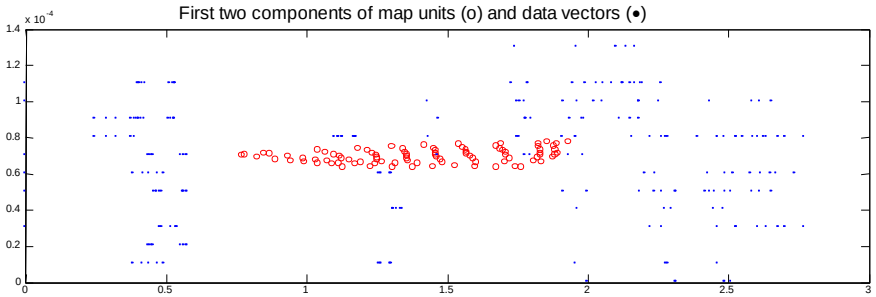


Figure C.7(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 52 for SIM_3)

(a)



(b)



(c)

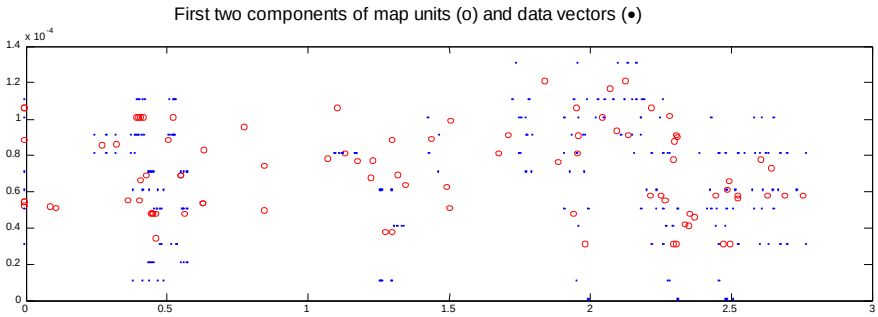


Figure C.10 (a) The QE in the second phase of SOM algorithm. (b) Initial state of the distribution of prototype vectors on the input space. (c) Final state of the distribution of prototype vectors on the input space. (for SIM_4)

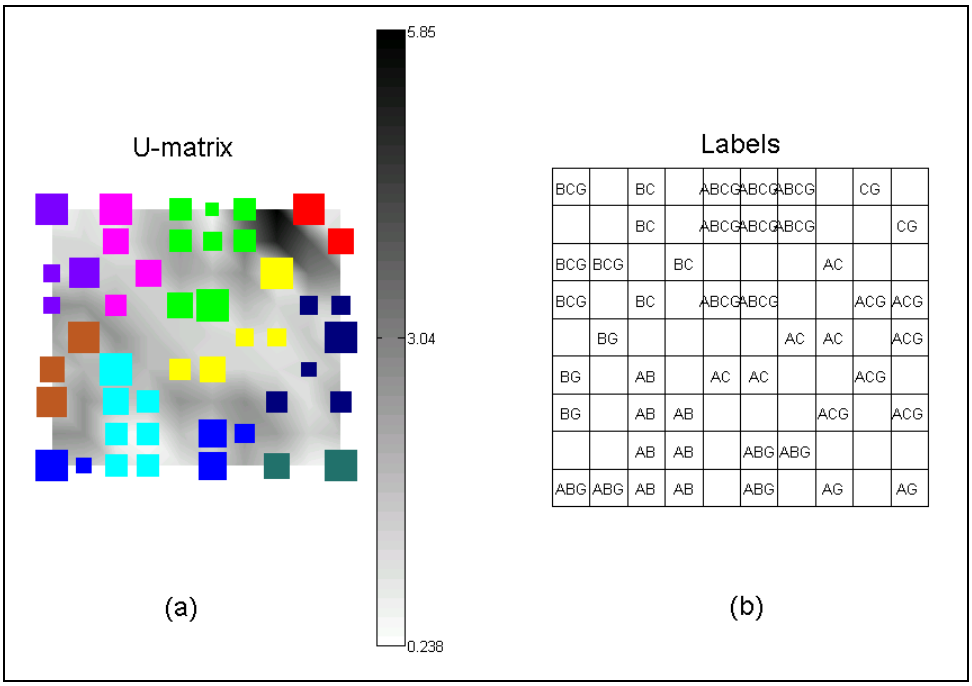


Figure C.8(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 46 for SIM_4).

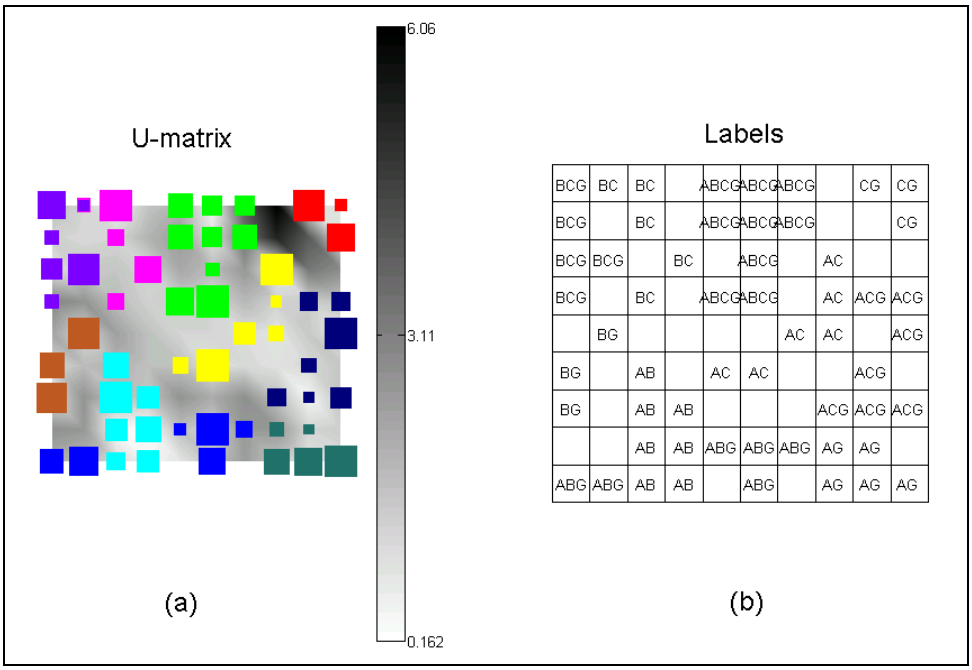


Figure C.9(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 56 for SIM_4).

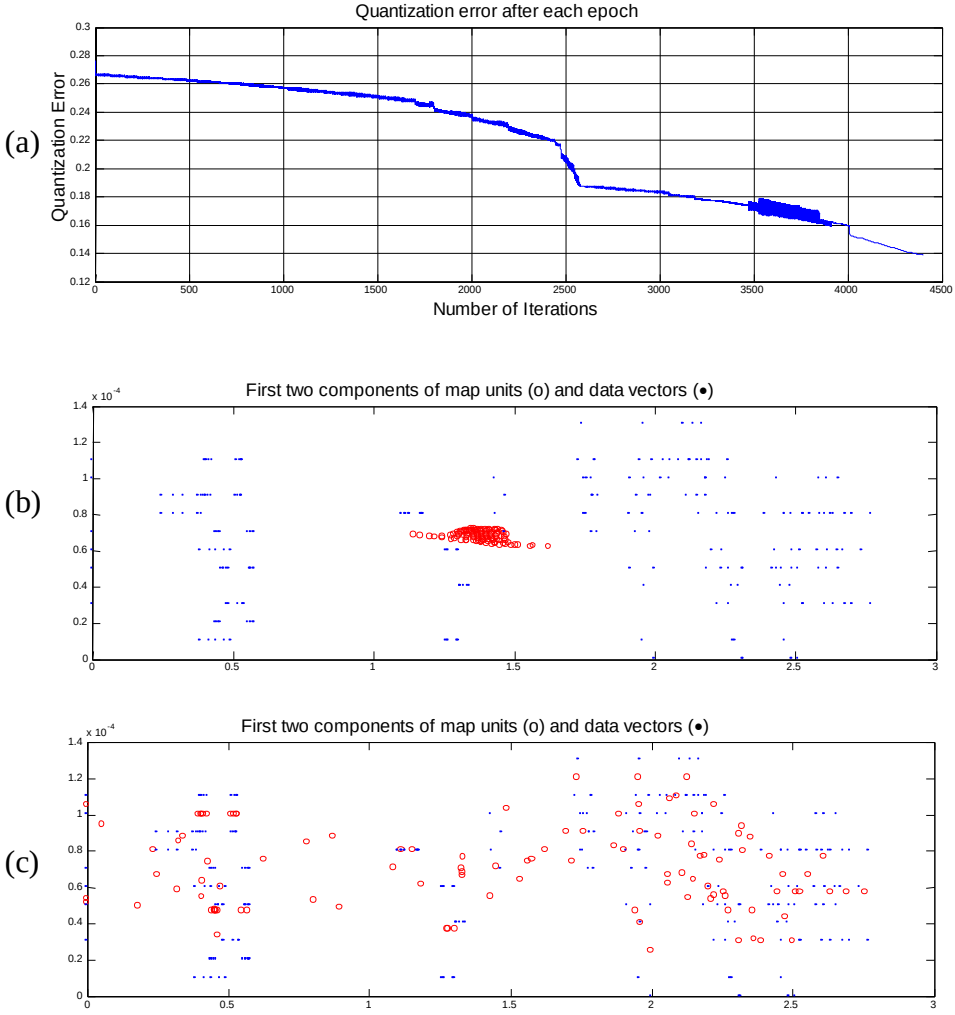


Figure C.13 (a) The QE in the second phase of SOM algorithm. (b) Initial state of the distribution of prototype vectors on the input space. (c) Final state of the distribution of prototype vectors on the input space. (for SIM_6)

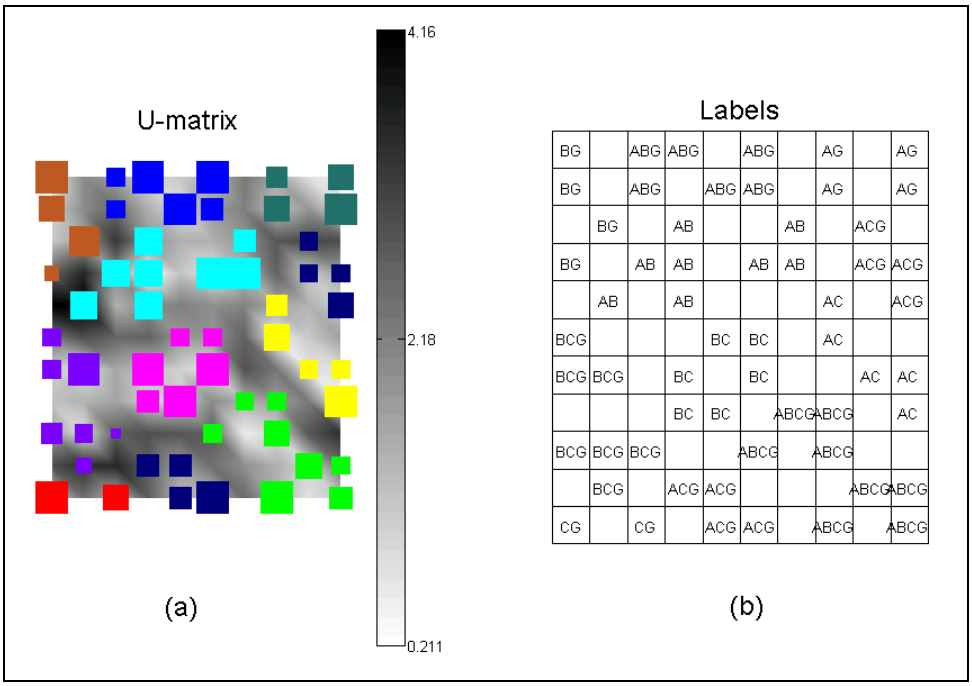


Figure C.14(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 58 for SIM_6)

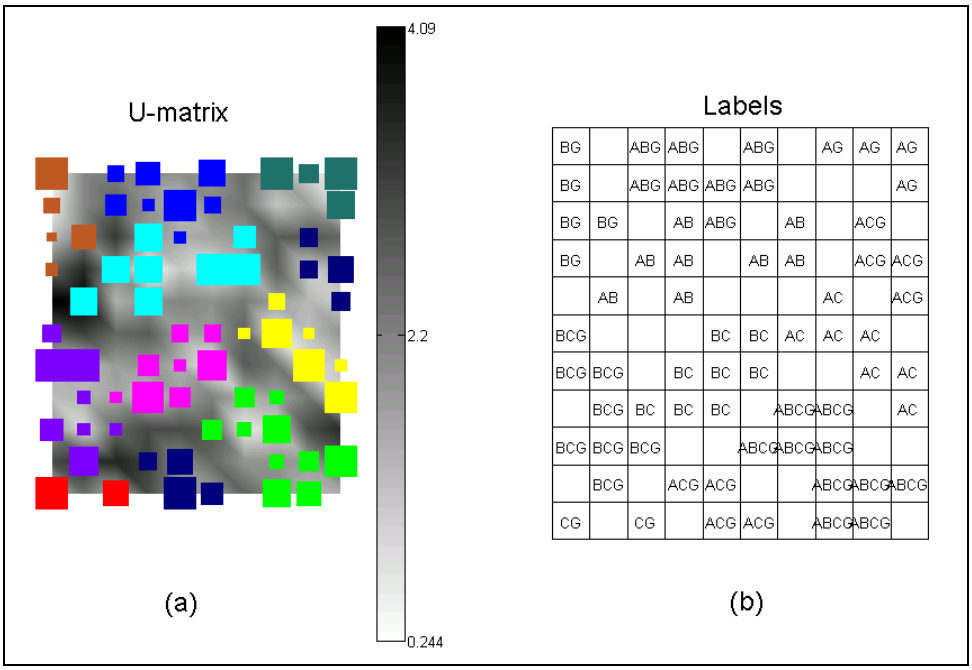


Figure C.15(a) The hit histograms on the U-matrix. (b) The labeled SOM. (labeled map unit amount = 68 for SIM_6)

BIOGRAPHY

Oben Dağ was born in 1978 in Istanbul. After graduating from Tercuman College, he started studying Electrical Engineering at Yıldız Technical University. He graduated from YTU in 1999 with the forth-highest degree. His undergraduate thesis was the control of a "0.7 kV AC Asynchronous Motor" using Siemens Access System and to transfer data from S7-200 PLC to workstation via Dynamic Data Exchange communication protocol. He is currently enrolled in Electrical Engineering programme at the Institute of Science and Technology in Istanbul Technical University as a M.Sc. student. His research areas cover signal processing, artificial neural networks, and data communication. He is a student member of IEEE and he is also currently a Cisco Certified Network Associative.