2005

NONLINEAR SYSTEM MODELING UTILIZING NEURAL NETWORKS: AN APPLICATION TO THE DOUBLE SIDED ARC WELDING PROCESS

Earl L. Fugate
University of Kentucky

ABSTRACT

NONLINEAR SYSTEM MODELING UTILIZING NEURAL NETWORKS: AN APPLICATION TO THE DOUBLE SIDED ARC WELDING PROCESS

The need and desire to create robust and accurate joining of materials has been one of up most importance throughout the course of history. Many forms have often been employed, but none exhibit the strength or durability as the weld. This study endeavors to explore some of the aspects of welding, more specifically relating to the Double Sided Arc Welding process and how best to model the dynamic non-linear response of such a system. Concepts of the Volterra series, NARMAX approximation and neural networks are explored. Fundamental methods of the neural network, including radial basis functions, and Back-propagation are investigated.

KEYWORDS: Radial Basis, Neural Network, Volterra Series, Double Sided Arc Welding, Back-propagation

Earl L. Fugate

July 29, 2005

NONLINEAR SYSTEM MODELING UTILIZING NEURAL NETWORKS: AN APPLICATION TO THE DOUBLE SIDED ARC WELDING PROCESS

By

Earl L. Fugate

Dr. YuMing Zhang
Director of Thesis

Dr. YuMing Zhang
Director of Graduate Studies

July 29, 2005

RULES FOR THE USE OF THESIS

Unpublished theses submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this thesis for use by its patrons is expected to secure the signature of each user.

NAME DATE

NONLINEAR SYSTEM MODELING UTILIZING NEURAL NETWORKS: AN APPLICATION TO THE DOUBLE SIDED ARC WELDING PROCESS

Earl L. Fugate

The Graduate School

University of Kentucky

2005

NONLINEAR SYSTEM MODELING UTILIZING NEURAL NETWORKS: AN APPLICATION TO THE DOUBLE SIDED ARC WELDING PROCESS

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in the
Department of Electrical and Computer Engineering
at the University of Kentucky

By

Earl L. Fugate

Lexington, Kentucky

Director: Dr. YuMing Zhang, Professor of Electrical Engineering

Lexington, Kentucky

2005

MASTER'S THESIS RELEASE

I authorize the University of Kentucky
Libraries to reproduce this thesis
in whole or in part for purposes of research.

Signed: Earl L. Fugate

Date: July 29, 2005

ACKNOWLEDGEMENTS

The following thesis, while an individual work, benefited from the insights and direction of several people. First, my Thesis Chair, Dr. YuMing Zhang, exemplifies the high quality scholarship to which I aspire. Next, I wish to thank the complete Thesis Committee: Dr. Bruce Walcott, and Dr. Lawrence Holloway. Each individual provided insights that guided and challenged my thinking, substantially improving the finished product. This work was partially supported by the National Science Foundation under Grant DMI-9812981.

In addition to the technical and instrumental assistance above, I received equally important assistance from family and friends. My wife, Stephanie, provided on-going support throughout the thesis process, as well as motivational assistance critical for completing the project in a timely manner. Finally, I wish to thank the respondents of my study (who remain anonymous for confidentiality purposes). Their comments and insights created an informative and interesting project with opportunities for future work.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

1.1 Introduction

Welding is the most economical and efficient way to join metals permanently. It is the only way of joining two or more pieces of metal to make them act as a single piece. The technology itself, can trace its roots back to ancient times. The earliest examples come from the Bronze Age. Artifacts from this age showed the interest that early man had in creating weapons. The early methods employed to join the metals involved heating the metals in charcoal furnaces to reduce it to a spongier, more pliable material. Once heated, the metals would then be beaten with a hammer to form the weld. This was known as "pressure" or "solid-phase" welding [1]. The advent of the Industrial Revolution showed great potential for advancing the methods and practices surrounding the concept of welding technology. New materials were being discovered, as well as being made that demanded newer techniques.

During recent times, more exotic materials are being utilized. The need to mechanically join them is still of great importance. Today's modern age of industrialization targets metallurgical materials, and in some cases such exotics as carbon fibers, to manufacture and build structures. In the past, common methods to join these materials involved such mechanisms as bolting, riveting, and/or adhesive technologies. While each provides certain benefits, none compare to the overall robustness and strength of the weld.

To this extent, the focus of the research investigates this capability--more specifically as it pertains to a process developed at the University of Kentucky known as Double Sided Arc Welding (DSAW) [2]. The overall concept of this research delves into the ability to physically develop a non-linear model to describe the DSAW process. Expanding from this model, future work may entail developing a robust control mechanism by which the system in its entirety may be controlled reliably.

To gain a fundamental understanding of the DSAW process, a brief overview will be presented. The acronym DSAW as it is aptly named, is short for Double Sided Arc

Weld. The general concept provides for the evolution of two independent processes by which both processes are utilized simultaneously to provide an improved welding process. The DSAW process in simplest terms involves the utilization of Gas Tungsten Arc Welding (GTAW) and Plasma Arc Welding (PAW). Both are well known in industry, with each providing unique characteristics and advantages in particular applications. The pursuit of improving overall process performance stems from the need to improve speed, quality, and reduce the overall cost. This in itself provides for a better-commercialized product.

1.2 Gas Tungsten Arc Welding

A process also known as heliarc or TIG welding, GTAW produces its weld by utilizing a heating mechanism generated by an arc between a non-consumable tungsten electrode (the electrode does not melt and become part of the weld) and the work piece. The oxygen atmosphere is very corrosive to the materials and tends to oxidize them. To this end, a shielding mechanism is employed to prevent the degradation of the materials. The shield is developed around the weld using an inert gas or some form of a gas mixture. Figure 1.1 below provides a schematic diagram of the process.

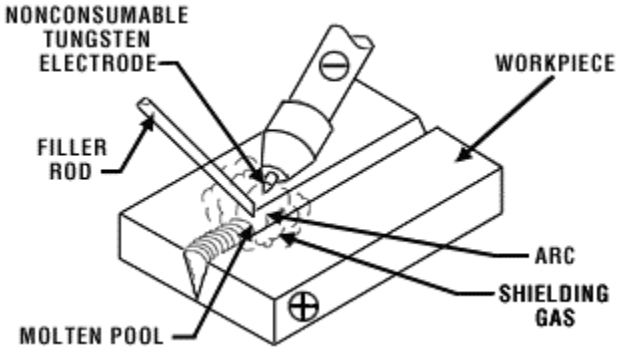

Figure 1.1 GTAW Process

GTAW is often employed in the creation of high performance and quality welds. The welds produced by the GTAW process are usually without flaws or defects regardless of the materials being used. From an industrial standpoint, GTAW is used in the aerospace, shipbuilding, automotive, and power production industries. Overall, GTAW is capable of providing precise control of welding parameters and heat input, and has low equipment cost. While the process provides both cost and quality benefits, it lacks the capability of being a speedy process. Not only does it lack speed, but also it fails to produce deep joint penetration. The lack of first pass penetration often requires multiple passes of the torch to provide the adequate depth and can become burdensome and time consuming.

1.3 Plasma Arc Welding

The Plasma Arc Welding process produces a bonding of metals by heating them with a constricted arc between an electrode and the work piece (transferred arc) or the electrode and the constricting nozzle (non-transferred arc). Like that of the GTAW process, a shielding mechanism is often employed as well. In general, it consists of a hot ionized gas issuing from the orifice, which may be supplemented by an auxiliary source of shielding gas. The shielding gas may be an inert gas such as argon or a mixture of gases. Figure 1.2 below provides a diagram of the process.

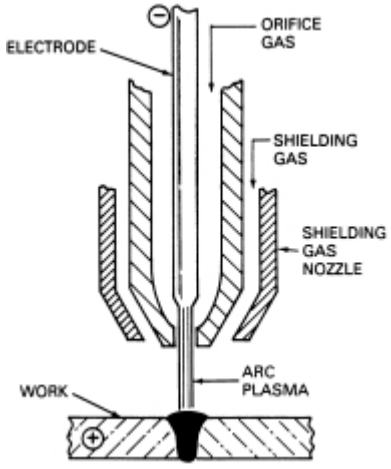

Figure 1.2 PAW Process

The PAW process can be considered a cost effective solution for higher speed welding and keyhole welding. Providing the heat intensity of the plasma arc is great enough, the process can operate in keyhole mode. An unfortunate issue with this process is the occurrence of porosity and holes in the weld, not to mention the spattering due to the highly focused beam, resulting in low quality welds. The keyhole mode is enabled when the gas flow is restricted through a reduction of the gas orifice size. This increases the gas velocity and the overall arc temperature. If too severe a constriction occurs in the orifice, then the gas flow produces a cutting arc. Essentially, the plasma arc blows a hole through the joint or plate on which it is operating [3]. Behind the hole, the molten metal flows together--filling the hole. This phenomenon is due to gravity forces, surface tension and the gas pressure from the shielding gas. While not as fast or energy efficient as Laser Beam Welding (LBW) or Electron Beam Welding (EBW), PAW does provide the distinct advantage in tolerancing to joint gaps and material misalignment. Both LBW and EBW provide a fine column of focused energy, which make them idea for high tolerance welds. They however, require the original joints to be very close in order for the weld to occur. The PAW effectively addresses the closeness issue by having a greater radial heating area. The downside to this, though, is that it tends to affect material distortion.

1.4 Double Sided Arc Welding

As was previously mentioned, DSAW is the merger of the GTAW and PAW processes. Although other torch combination may be utilized, the research thus far has been focused on the employment of these two processes. Namely, we have a configuration in which a PAW torch is positioned on one side of the material and the GTAW torch is placed on the other.

This can be seen in Figure 1.3 below.

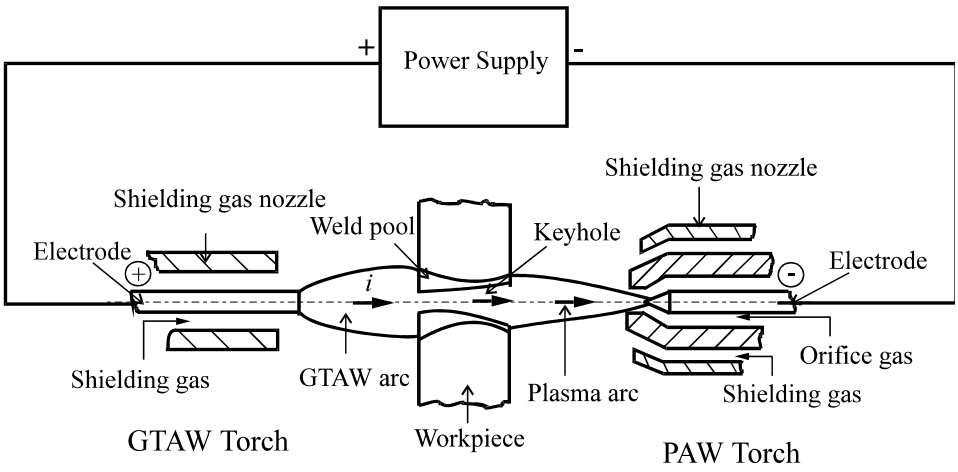

Figure 1.3 DSAW System

The DSAW system is primarily operated in what is known as keyhole model. To this extent, the goal is to switch the welding current from the peak intensity to a base level after keyhole establishment in order to prevent burn-through due to arc pressure, which is proportional to the square of the current [5].

In DSAW, simultaneous electric arcs are formed between the material work piece and the two torches. This characteristic requires that the current pass through the entire thickness of the material. The direction of flow of this current is conveniently termed as through-the-thickness (TTT). The TTT is established along the keyhole providing for a columnar arc to form [4].

From a comparative standpoint, we can see that the total energy being utilized is now being focused into the actual weld and not into the surrounding area that is commonly encountered with more traditional welding techniques. This focusing effect allows the weld to produce a cleaner, deeper joint.

1.5 Proposed Technique

As with most things in nature, the need to accurately control and describe a system is a must. This holds true for the research involving the DSAW process. The process as a whole is not an easy one to define, as there are numerous parameters and variables, which must be taken in consideration. A great amount of information exists regarding the linear modeling and identification of a system, but as will be demonstrated, linear models do not hold true for the DSAW system.

1.6 Organization

This thesis is organized into four main chapters.

In Chapter 2, a survey of some of the existing non-linear modeling techniques and methodologies will be investigated and a proposed method for the DSAW model will be presented.

In Chapter 3, a history of neural networks will be presented, including the origin of the famous back-propagation algorithm. In addition, radial basis functions and black box state space neural networks will be explained.

Chapter 4, will introduce the DSAW model data. The data will be segmented into a training set and test set and provided to a neural network. The net will then generate a model to describe the data.

In Chapter 5, a general conclusion will be provided as well as thoughts and concepts for further research revolving around the neural model.

Chapter 2

2.1 Non-linear Systems

There exists a wealth of information when it comes to system identification and modeling from a linear perspective. Linear system identification, in general, is a very strong and well-developed field of study. Linear system models, however, are beneficial only when the dynamic system is well behaved and tends to follow a linear response around a set operating point. In fact, some phenomena arise precisely because of non-linearity. Linear analysis offers little to no insight in such cases. The fact that all physical systems can be thought as non-linear in nature, has led to a following in which the modeling and analyzes have become a more mainstream area of study.

Non-linear systems are by far, more complex to analyze and model- this fact has often been a reason to abandon a fully realized non-linear model and pursue a more generalized "best-fit" linear model. Part of the complexity of non-linear systems involves the ability to resolve a low order, manageable dimensionality. This "curse of dimensionality" as it is often termed arises, from the fact that non-linear systems exhibit complex dynamic behavior around the operating region of interest.

The main focus of this chapter will be to survey some of the existing well-known techniques often employed in solving these types of dynamic systems. Volterra-series, NARMAX, local modeling, and neural networks will be of most interest.

2.2 Non-linear System Representations

The problem of system identification is in finding a suitable model structure. In general, the identification becomes a problem of relating a series of outputs to corresponding inputs. This is often based upon past input-output pairs as well as being able to predict and extrapolate what the future values will be.

If finite sets of data points are collected, as will be in the research, then a generalized data vector φ can be represented in the form of:

$$\varphi(t) = [y(t-1)\ y(t-2)\ \ldots\ y(t-n_y)\ \ u(t-1)\ u(t-2)\ \ldots\ u(t-n_x)]^T \tag{2.1}$$

where y(•) and u(•) represent the output and input vectors respectively. Upon the formulation of the vector, the goal is to evaluate the relationship between y(t) and φ(t). In other words, a mapping function is defined such that:

$$y(t) \rightarrow f(\varphi(t)) \tag{2.2}$$

It is from this point on that the presentation of several potential techniques that could be utilized for an appropriate solution set.

2.3 Volterra Series

Stemming from the early work of the Italian mathematician Vitto Volterra, and a later expansion by Norbert Weiner, who used the concept as a generalized analysis in the spectrum of an FM system with gaussian noise. The Volterra series marries the concepts of the Taylor series expansion with that of a convolution integral [6][7]. In general, the impulse response of a linear system can be realized such that [8]:

$$y(t) = \int_{-\infty}^{\infty} h(\tau) \cdot u(t-\tau) d\tau \tag{2.3}$$

where *y(t)* is the output, *u(t-τ)* is the input, and *h(τ)* is the impulse response of the system.

The non-linearity of the system can be represented by the Taylor series:

$$y(t) = C_0 + C_1 u + C_2 u^2 + C_3 u^3 + ...C_n u^n \rightarrow \sum_{n=0}^{\infty} C_n u^n \tag{2.4}$$

The Volterra series is then represented as:

$$y(t) = h_0 + \sum_{n=0}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \tau_2, ...\tau_n) \prod_{i=1}^{n} u(t - \tau_i) d\tau_i \tag{2.5}$$

The total output of the system is then said to be the sum of outputs from smaller parallel sub-systems represented by the h_n coefficients, also known as Volterra functionals or kernels. In essence, the series is defining two regions of operation, one being of a global nature, the other being one of a local nature. The overall system is being subdivided into smaller more localized units. These, in turn, are solved for and reassembled--which is commonly known as a divide and conquer technique.

From a graphical perspective, Figure 2.1 illustrates this hierarchy.

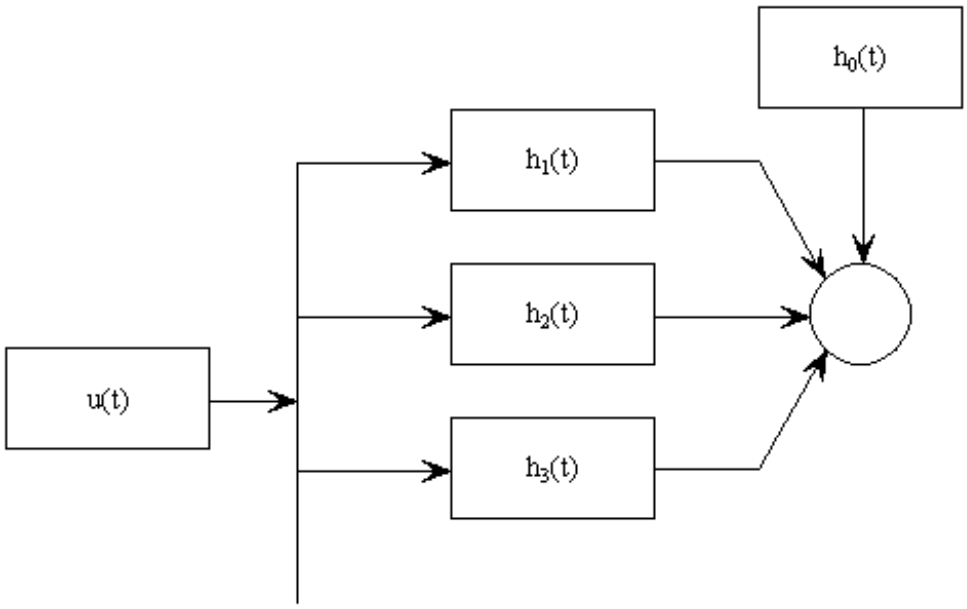

Figure 2.1 Volterra Series

As previously mentioned, the Volterra series involves a solution to system kernels h_n. There are numerous methods that can be leveraged to perform the necessary calculations.

2.3.1 Volterra Kernel

Assume the function f is such that a Taylor series expansion about some fixed point can be calculated such that [6]:

$$y(t) = h_0 + \sum_{m_1=0}^{\infty} h_1(m_1)u(t-m_1) + \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2(m_1,m_2)u(t-m_1)u(t-m_2) + \cdots \qquad (2.6)$$

the coefficients h_n are then defined as:

$$\begin{aligned} h_0 &= f(\bar{u}) \\ h_1(m_i) &= \left(\frac{\partial f}{\partial u(t-m_i)} \right)_{\bar{u}} \\ h_2(m_i, m_j) &= \left(\frac{\partial 2 f}{\partial u(t-m_i)\partial u(t-m_j)} \right)_{\bar{u}} \\ &\vdots \end{aligned} \tag{2.7}$$

where $\bar{u}$ represents the fixed point where the Taylor series is expanded about. In their writing, Boyd and Chua [7], show that the above model provides a good approximation for wide variety of non-linear systems.

Careful observation and deduction show that the computational feasibility of this model quickly evaporates as the order of the model increases. Based on these observations, it is quite realizable that if a model can be well behaved over a particular region of interest, then a small degree model can be utilized to approximate it- thus a Volterra series could be used to approximate a non-linear system. Most non-linear systems, however, tend to require a higher degree of accuracy to better depict the system. From this standpoint, it is a safe assumption that a better modeling technique needs be incorporated.

2.4 NARMAX modeling technique

With such a wealth of knowledge already existing for linear systems, it seems reasonable to ascertain that utilization of these fundamental principles could be expanded to inclusion within a non-linear system. The Non-linear AutoRegressive Moving Average with eXongenous inputs or NARMAX models describe systems in terms of linear-in-the-parameter difference equations. Leontaritis and Billings [8] introduced the NARMAX approach as a means of describing the input-output relationship of a non-linear system. This model takes into account contributions from the present and past

inputs as well as the past outputs of the system. When identifying a NARMAX model structure, two key factors must be defined: structure detection and parameter estimation.

The structure detection can be subdivided to include model order selection and parameter selection. Model order is critical since an infinite number of candidate terms may exist for a particular problem. Once an order has been established, calculation of parameter estimates can be performed.

The NARMAX is an extension to the well known linear model ARMAX and is defined as:

$$\begin{aligned} y(k) = F(&y(k-1),...,y(k-n_y),u(k-1),... \\ &...,u(k-n_u),e(k-1),...,e(k-n_e)) + e(k) \end{aligned} \tag{2.8}$$

where F is a non-linear function mapping; $y(k)$, $u(k)$, and $e(k)$ represent the systems output, input, and error respectively. The n_y, n_u, and n_e terms depict the maximum lags associated with the output, input, and error. The prediction error is defined as:

$$e(k) = y(k) - \hat{y}(k) \tag{2.9}$$

with $\hat{y}(k)$ being the prediction output. Defining F as a polynomial function of degree l allows expansion of equation 2.8 to show the representation of $u(k)$, $y(k)$, and $e(k)$ up to degree l.

It then can be represented as:

$$\begin{aligned} y(k) = &\theta_1 y(k-1) + \theta_2 u(k-1) + \theta_3 u(k-1)y(k-1) \\ &+ \theta_4 y(k-1)e(k-1) + \theta_5 e(k-1)e(k) \end{aligned} \tag{2.10}$$

The above equation can be written in a more concise matrix representation by defining a variable p such that $p_1(k) = y(k-1)$, $p_2(k) = u(k-1)$, $p_3(k) = u(k-1)y(k-1)$, $p_4(k) = y(k-1)e(k-1)$, $p_5(k) = e(k-1)e(k)$.

If N input-output pairs are available and M terms are in the model, then the matrix form of 2.10 is:

$$Y = P\theta + e \tag{2.11}$$

where

$$\begin{aligned} Y^T &= [y(1)\ y(2)\ y(3) \ldots y(N)] \\ \theta^T &= [\theta_0\ \theta_1\ \theta_3 \ldots \theta_M] \\ e^T &= [e(1)\ e(2)\ e(3) \ldots e(N)] \end{aligned}$$

$$p = \begin{bmatrix} p_0(1) & p_1(1) & \cdots & p_M(1) \\ p_0(2) & p_1(2) & \cdots & p_M(2) \\ \vdots & \vdots & \vdots & \vdots \\ p_0(N) & p_1(N) & \cdots & p_M(N) \end{bmatrix} \tag{2.12}$$

In the above, p_x represents a term in the NARMAX model and θ is a parameter that needs to be estimated. There exist numerous methods that can be used in estimating the parameter θ. These include the well known least squares or error estimation algorithms. In their research, Billings and Tsang [9] employed a orthogonal estimator, which provided for simplicity and efficiency.

2.4.1 NARMAX Orthogonal Parameter Estimation

The basis for this algorithm employs transforming equation 2.11 into an equivalent auxiliary equation such that:

$$y(k) = \sum_{i=1}^{n_\theta} g_i w_i(k) + \varepsilon(k), k = 1, 2, ..., N, n_\theta = M \tag{2.13}$$

where $w_i(k)$ is constructed to be orthogonal to the data set and g_i represent constant coefficients. The property of orthogonality allows each of the parameters to be estimated one at a time. Orthogonal vectors can be constructed for the data set so that:

$$w_1(k) = p_1(k), w_j(k) = p_j(k) - \sum_{i=1}^{j-1} \alpha_{ii} w_i(k) \tag{2.14}$$

where

$$\alpha_{ii} = \frac{\sum_{k=1}^{N} w_i(k) p_j(k)}{\sum_{k=1}^{N} w_i^2(k)} \tag{2.15}$$

such that j = 1, 2, … n_θ; and i = 1, 2, …, j – 1, j. To estimate the coefficients g_i for equation 2.13, the parameters are defined as:

$$\hat{g}_i = \frac{\sum_{k=1}^{N} w_i(k) y(k)}{\sum_{k=1}^{N} w_i^2(k)} \tag{2.16}$$

We are now able to obtain the original unknown parameters via $\hat{g}_i$ by the following conversion formulas:

$$\begin{aligned} \hat{\theta}_{n_\theta} &= \hat{g}_{n_\theta} \\ \hat{\theta}_i &= \hat{g}_i - \sum_{j=i+1}^{n_\theta} \alpha_{ij} \theta_j, i = n_\theta - 1, n_\theta - 2, ..., 1 \end{aligned} \tag{2.17}$$

The auxiliary regressors are thus orthogonal (i.e. $w_i(k)w_j(k) = 0, i \neq j$) so additional terms may be added as necessary without computing the previous $\hat{g}_j$, for j < i.

As in the case of the volterra series, it becomes critical to select an order number that can best represent the system, and again we potentially suffer from the "curse of dimensionality".

2.5 Local Modeling Approaches

Stemming from the pseudo-Taylor series idea, local modeling involves localization of a global operating plane. This modeling technique was developed by Johansen and Foss [10, 11]. The architectures thus proposed, provide a promising alternative solution to non-linear model structures with a character that resembles that of a neural network--of which will be discussed in more detail to follow. The common termed values of neuro-fuzzy and fuzzy systems belong to this local modeling technique.

The concept is based upon the decomposition of the input space into a varied series of operating regimes. Within each of these regimes, a local sub-model is held to be valid, or defined such as to describe the operation over the particular input space. Each sub-model is weighted by some activation function Φ_i. The ultimate system output response (i.e. global) $\hat{y}$ is then defined to be the combination of all local sub-models:

$$\hat{y} = \sum_{i=1}^{M} \Phi_i(u) g_i(u) \qquad 2.18$$

Where M is the number of models, $g_i(u)$ is the local model output.

Figure 2.2 below is a graphical representation of the local model scheme [12].

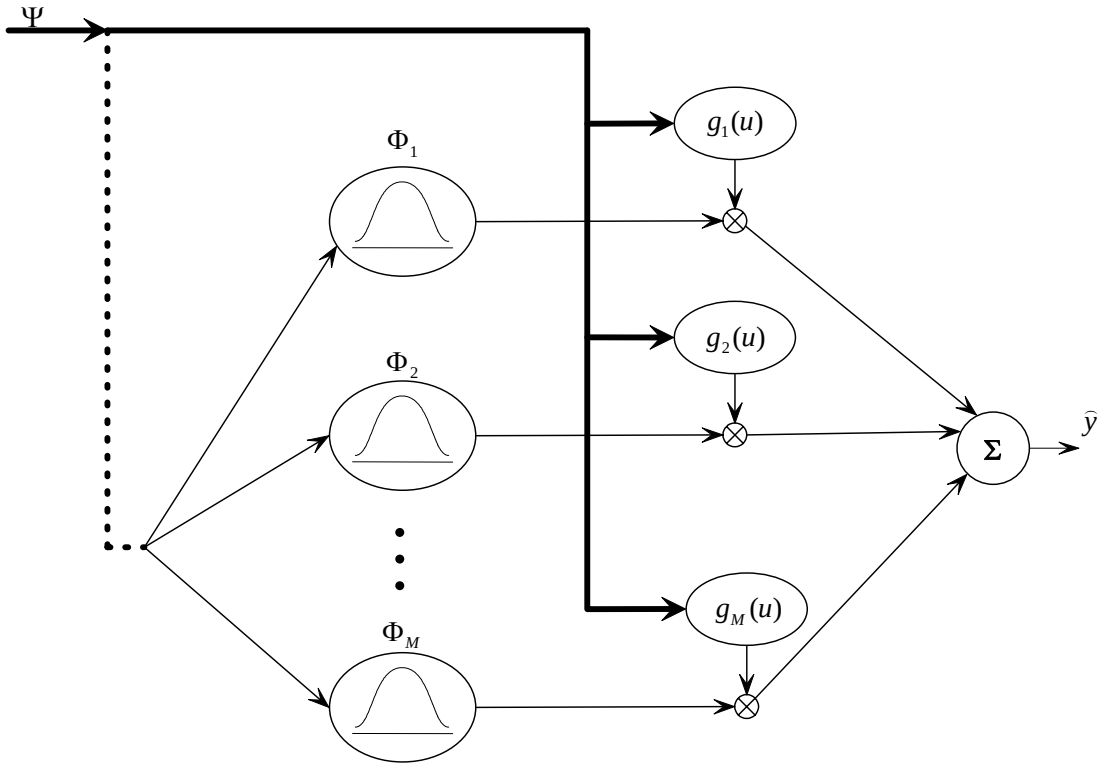

Figure 2.2 Local Modeling Approach

Often times, the activation function is defined to be of a gaussian nature. The shape of this activation function gives a smoothing effect to the overall output of the sub-system and falls into the category of radial basis functions (RBF). The gaussian distribution is defined such that:

$$\Phi = \exp\left[\frac{(\varphi - c_i)^2}{\sigma_i^2}\right] \quad for\ \mathrm{i} = 1,2,...,\mathrm{n} \qquad 2.19$$

where Φ is the activation as previous mentioned, φ is the scheduling variable, c is the center point, and σ is the width for the local model.

Different architectures are distinguishable in relation to the following three properties: [13]

- *Partitioning Principle*: The activation function Φ_i defines a decomposition strategy in relation to: grid structure, recursive partitioning, or partitioning into operating regimes of arbitrary form.
- *Local model structure*: While the structure can be of any form, a linear structure is most often applied due to simplicity. The optimization of the linear model parameters is simply parameter identification. A concept well known in linear system theory.
- *Transition between sub-models*: In general, the transition can be thought as a hard or soft transition. A hard transition being such as a sudden switch occurs. Refer to Figure 2.3 below. The solid curve indicates the soft transition, whereas the dotted depicts the hard transition.

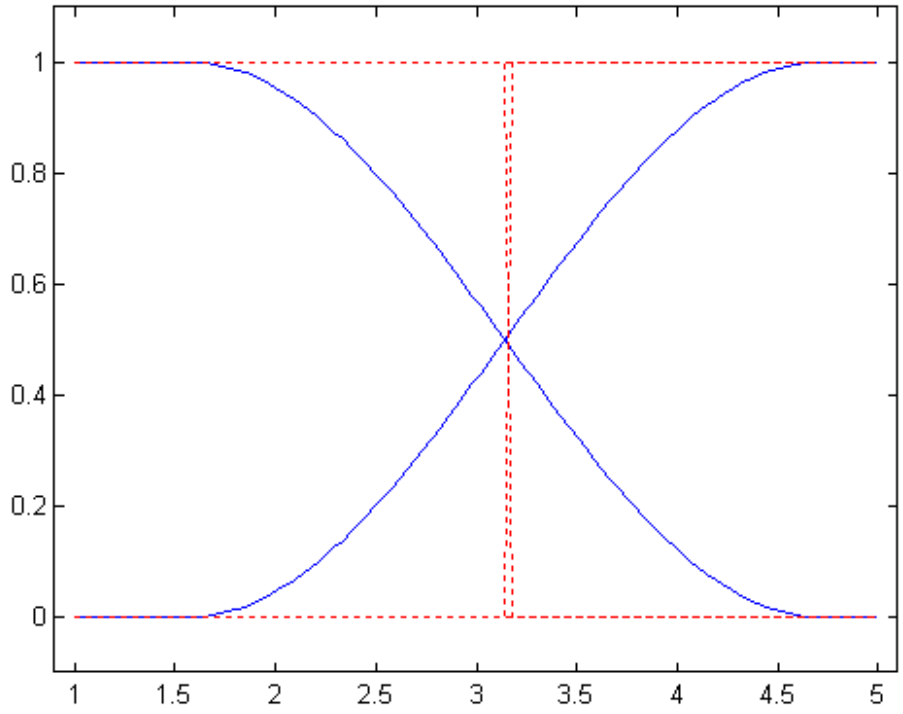

Figure 2.3 Sub-model Transitions

For more detailed and comprehensive description regarding various local model architectures, refer to Murray-Smith and Johanson [14].

2.6 Remarks

While numerous other methodologies exist to provide sufficient modeling capabilities for nonlinear systems, the ascribed methods above have proven beneficial in a wide range of areas. Take for example, some of the more modern circuit simulator programs. These simulators utilize modified methods of the Volterra series to simulate and extrapolate real world circuits. The NARMAX structure is a welcomed extension to the already know linear ARMAX model. With a wealth of knowledge already in existence for such linear systems, an obvious choice would be the modify these and make them work within the nonlinear framework. Finally, the local modeling approach provides a brute force method to provide a solution to a more complex system. It can be paralleled with that of a normal design flow (i.e. breaking a design into smaller manageable units.)

Chapter 3

3.1 History of the Neural Network

The concept of a neural network was first envisioned in a McCullock and Pitts paper [15]. The first proposed artificial neuron came to be known as the Threshold Logic Unit (TLU). The TLU as proposed, summed a number of n-inputs. Each input in turn is multiplied by a weighting function to produce a scaled version of the input signal. Each of the n-scaled signals provide the input to an activation unit, which for this first embodiment, was hard limited to either a logic '1' or '0'. The general structure of the TLU provides a comparable relationship to digital logic circuits. The TLU is graphically represented in Figure 3.1 below.

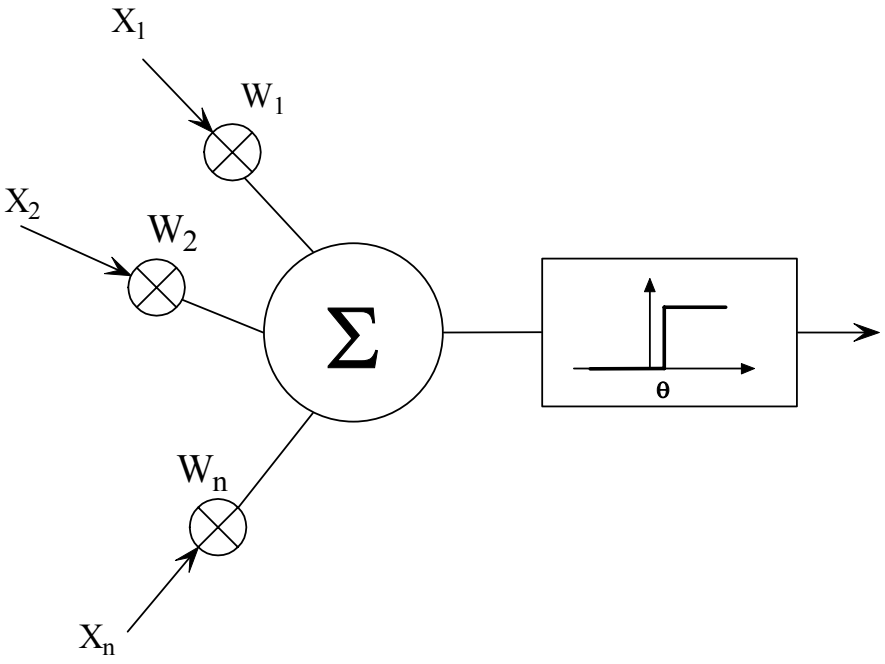

Figure 3.1 TLU

In most common applicable applications, however, the need for the activation function to be continuous and analog is often desired. There are numerous functions that exist. A sigmoid function is often utilized due to its ability to provide smoothing. The general form of the sigmoid functions is defined as:

$$y = \sigma(a) \equiv \frac{1}{1 + e^{-(a-\theta)/\rho}} \tag{3.1}$$

where ρ determines the flatness of the sigmoid and θ adds a non-zero threshold activation or shift as can be seen in Figure 3.2 below.

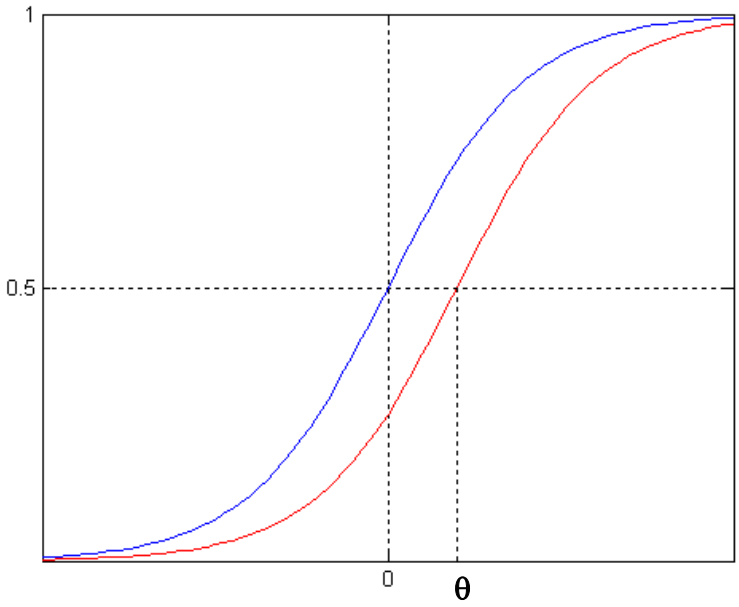

Figure 3.2 Influence of ρ and θ

The TLU offered a new hope for system identification in that a system could be grouped or classified by a decision plane. In essence, the input space is separated into two parts defined by this decision plane. As an example, assume a binary system has an output that produces either an "A" or "B". The TLU classifies the two linearly separable input spaces into those that produce an "A" and those that produce a "B".

The decision line is shown in Figure 3.3 below:

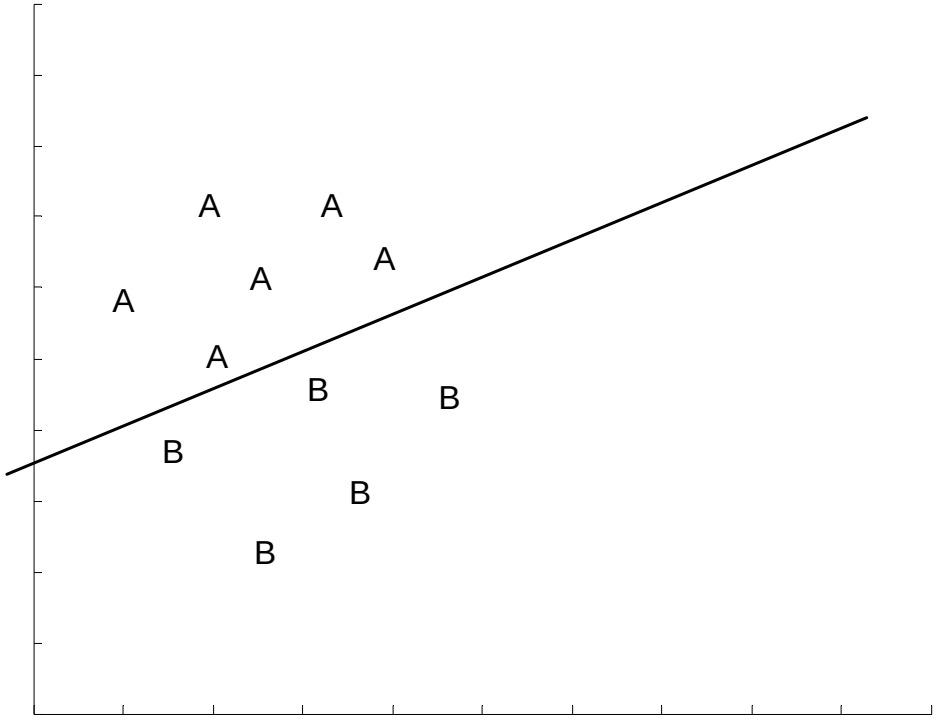

Figure 3.3 Decision plane for a binary system

3.1.1 TLU Training

The concept of a neural network encompasses the ability to train the network to respond to a certain set of stimulus. To this end, training entails utilizing a weighting vector and a threshold unit as a discriminator. A set of vectors ***v*** and ***t*** are defined as the training set and output class respectively, with ***w*** being defined as the weighting vector. The set pairs {***v***, t} are known as a supervised learning scheme, in that, the investigator tells the network what the output should be.

In general, assume there is a set of vectors ***v*** and an initial weighting vector ***w*** that is believed to provide the target class ***t***. The ***w*** vector, however, produces an activation output of y = 0 when ***t*** is expected to be 1. To produce this contrivance, the activation was negative. In order to compensate for this, the weight vector ***w*** must be modified to

correctly resemble the activation. In order to accomplish this, the vector does not need to undergo a major change, as this would destroy prior learning of the network. A small part of the vector $\boldsymbol{v}$ is added to the existing weight vector. Defining a new weight as:

$$\boldsymbol{w'} = \boldsymbol{w} - \alpha\boldsymbol{v} \tag{3.2}$$

such that the new weight is the old weight $\boldsymbol{w}$ adjusted by a fractional part of the input vector $\boldsymbol{v}$. In the case that $\boldsymbol{t}$ is expected to be a 0 and y = 1, the updated weight is defined as:

$$\boldsymbol{w'} = \boldsymbol{w} + \alpha\boldsymbol{v} \tag{3.3}$$

Equations 3.2 and 3.3 and concisely be combined and be rewritten as:

$$\boldsymbol{w'} = \boldsymbol{w} + \alpha(t-y)\boldsymbol{v} \tag{3.4}$$

And further refined in terms of the change of the weight vector $\Delta\boldsymbol{w}$

$$\Delta\boldsymbol{w_i} = \alpha(t-y)\boldsymbol{v_i} \tag{3.5}$$

The preceding equation is commonly known as the training rule. The parameter α is defined to be the learning rate.

A simple training algorithm is as follows in listing 3.1:

```
initialize weight vector to small random number
        for each vector pair (v,t)
                evaluate output y
                        if y ≠ t then
                                create new weight vector w'
                        else
                                do nothing
                        end if
        end for
until y = t for all vectors, repeat for
```

Listing 3.1 Training Algorithm

This concept of adaptive training was first introduced by Rosenblatt [16] and is known more commonly as the Perceptron.

3.1.2 The Delta Rule

Widrow and Hoff [17] proposed a technique to train a network on the activation itself and not on the output. This concept is based upon a gradient descent calculation in determining the error of the network when presented with a training vector. Recall from calculus, that given some arbitrary function y = y(x) where the exact form of the function is not known, a local minimum is able to be calculated . To determine the position x that provides for the minimal value of the function, y is differentiated with respect to x.

The slope Δy / Δx is defined to be the gradient of the tangent and is depicted in Figure 3.4:

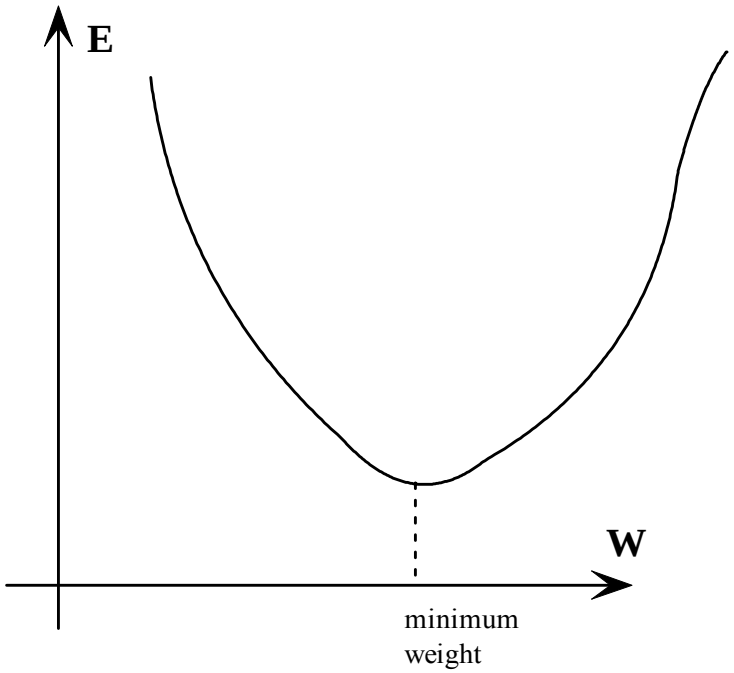

Figure 3.4 Gradient Descent and minimum of unknown function

For each pattern feed into the network, a corresponding error E_p as a function of the weights is given such that:

$$_p = E_p(w_1, w_2, \ldots, w_n). \tag{3.6}$$

The error is commonly defined to be the square of the difference between the actual output and the desired target.

$$E_p = \frac{1}{2}(t-a)^2 \tag{3.7}$$

Equation 3.7 depicts the utilization of the activation *a* rather than the output y as previously described. The total error is then simply the sum of all the errors:

$$E = \sum_{p} E_p \tag{3.8}$$

The new learning rule as defined in (3.5) becomes:

$$\Delta \boldsymbol{w_j} = \alpha(t - a)x_j \tag{3.6}$$

where x_j is the input element of the weight w_j as shown in Figure 3.1.

Unlike the Perceptron, whose theoretical basis is the hyper-plane manipulation of input classification, the Delta (δ) rule is given by the gradient descent on the square of the error. The benefits of the delta rule provide the ability to train more than a single layer network. Up until this point, only single layers were able to perform calculations. With the addition of the delta rule, the sophistication and capability of problems that can be solved improved.

3.1.3 Backpropagation

While the concept of neural networks blossomed into a fledging science in the early 40's through late 60's, a 1969 publication entitled 'Perceptrons' [18] became detrimental to the concept of using neural networks in solving real world problems. It was found that the capabilities of the network where incapable of solving problems associated with linearly inseparable problems. The nexus of the exclusive-or function proved somewhat novel to the network as they were incapable of solving the simple problem. The science remained somewhat dormant until the mid-80's when Back- error-propagation (a.k.a. Backpropagation) was popularized by Rumelhart, Hinton, and Williams [19].

The concept of Backpropagation extends the previously mentioned Delta rule to include more than one node, such that the error is now calculated over all nodes. The error calculation then becomes:

$$E_p = \frac{1}{2}\sum_{j=1}^{N}(t^j - y^j)^2 \tag{3.7}$$

The above equation, takes into account all the weights of the intermediary layers, inclusive of the hidden and output nodes. Note the fact that the hidden layers are not directly connected to any input, thus there is no direct method for training these nodes (i.e. they are unobservable outputs). This new network architecture is depicted in the following Figure 3.5.

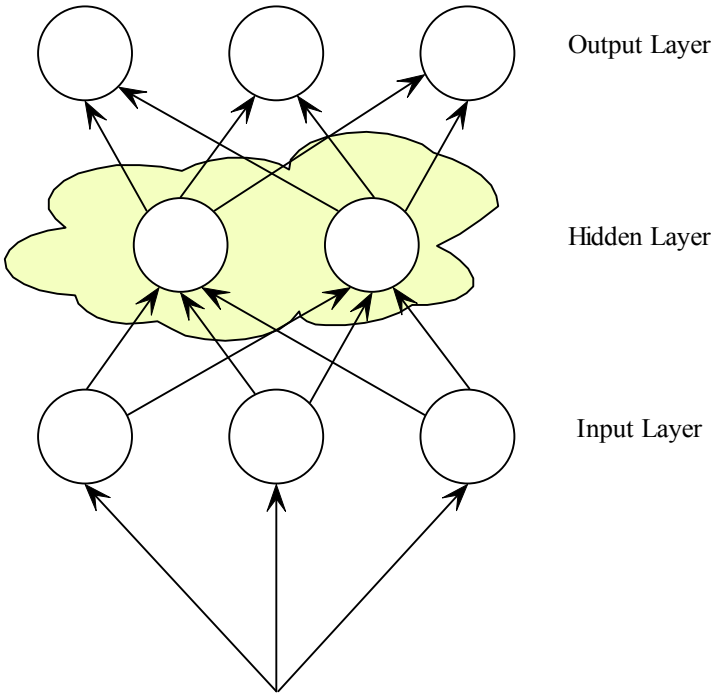

Figure 3.5 General Neural Network

The updated output weighting equation of 3.6 becomes:

$$\Delta w_i^j = \alpha\sigma'(a^j)(t^j - y^j)x_i^j \tag{3.8}$$

The superscripts denote which node in the network is presently being addressed. As previously stated, the $(t^j - y^j)$ term indicates the error between the targeted output *t* and the nodes actual output ***y***. The $\sigma'(a^j)$ and x_i^j terms define how fast the nodes activation function is allowed to change the output and the impact that the input has on the system output respectively. Small values indicate little contribution. The above equation gives us a measure of the rate of change of the error (i.e. gradient descent). Keep in mind, (3.8) only describes the weighting of the output node and does not entail the internal hidden nodes. To this end, the error signal of the hidden nodes is defined as:

$$\delta^j = \sigma'(a^j)(t^j - y^j) \tag{3.9}$$

such that (3.8) becomes:

$$\Delta w_i^j = \alpha\delta^j x_i^j \tag{3.10}$$

for the output node. To gain a more beneficial and fundamental understanding of the hidden network, refer to Figure 3.6 below. The diagram symbolizes the interconnection of a single path output from the input through the hidden node.

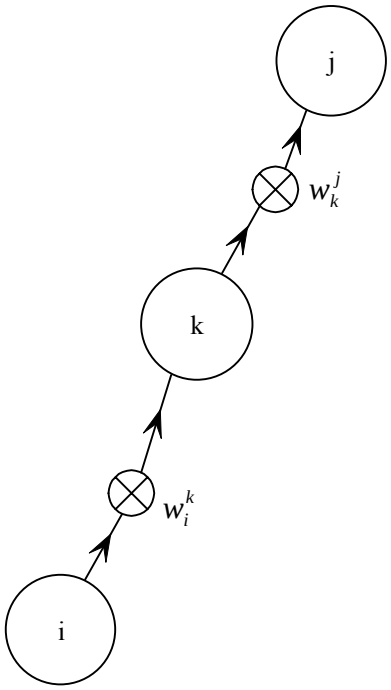

Figure 3.6 Single path Network Interconnection

The two additional terms δ^j and δ^k are defined as being the error of nodes *j* and *k*. The influence that each node *i* has on *k* and *k* has on *j* is defined by the weights, w_i^k and w_k^j. Based on this, δ^j will have a component of δ^k such that each node connected must be summed so that:

$$\delta^k = \sigma'(a^k)\sum_j \delta^j w_k^j \tag{3.11}$$

Combining (3.11), (3.10), and (3.7) provides us with the backpropagation error equation:

$$E = \frac{1}{2}\sum_p \left(\sum_j \left(t_{pj} - y_{pj}\right)^2 \right) \tag{3.12}$$

3.2 Neural Net Models

The current field of neural networks has grown in recent years and thus has provided a vast library on neural modeling. To gain a better appreciate of some of the literature available, one can reference [20]. As with most modeling schemes, one of the key components is to determine the model structure. Neural networks are no different. To maintain focus, only two of the possible schemes will be investigated. These include a Radial Basis Function (RBF) neural network and the other entails utilization of a state-space neural network.

3.2.1 Radial Basis Functions

The RBF neural network can be thought of as a two layer network structure such that the first layer is a hidden layer and contains the RBF activation nodes (neurons). The second layer encompasses the output neurons, which compose a weighted sum of the hidden layers output. One of the interesting facets of the RBF neural network is the input-output correlation pair it provides. The input to the network is nonlinear in nature while the output is linear. For this reason, the RBF network are said to be good curve fitters, such that they are capable of approximations in the higher dimensional spaces. This fact has been studied in [21].

The general definition of the RBF neural network can be defined such that:

$$z_i(x) = r\left(\frac{\|x-\mu_i\|^2}{\sigma_i^2}\right) \tag{3.13}$$

where $z_i(x)$ is the output of the *i*th neuron. The variables μ_i and σ_i^2 dictate the center and width of the corresponding neuron. The function $r(\cdot)$ is chosen to be a suitable radial basis function or activation.

The most common function chosen is gaussian in nature so that (3.13) becomes:

$$z_i(x) = e^{-\|x-\mu_i\|^2/\sigma_i^2} \tag{3.14}$$

here, x is the input vector. To gain a better understanding of the effects that the μ_i and σ_i^2 terms have on the basis function, refer to the following Figure 3.7 below.

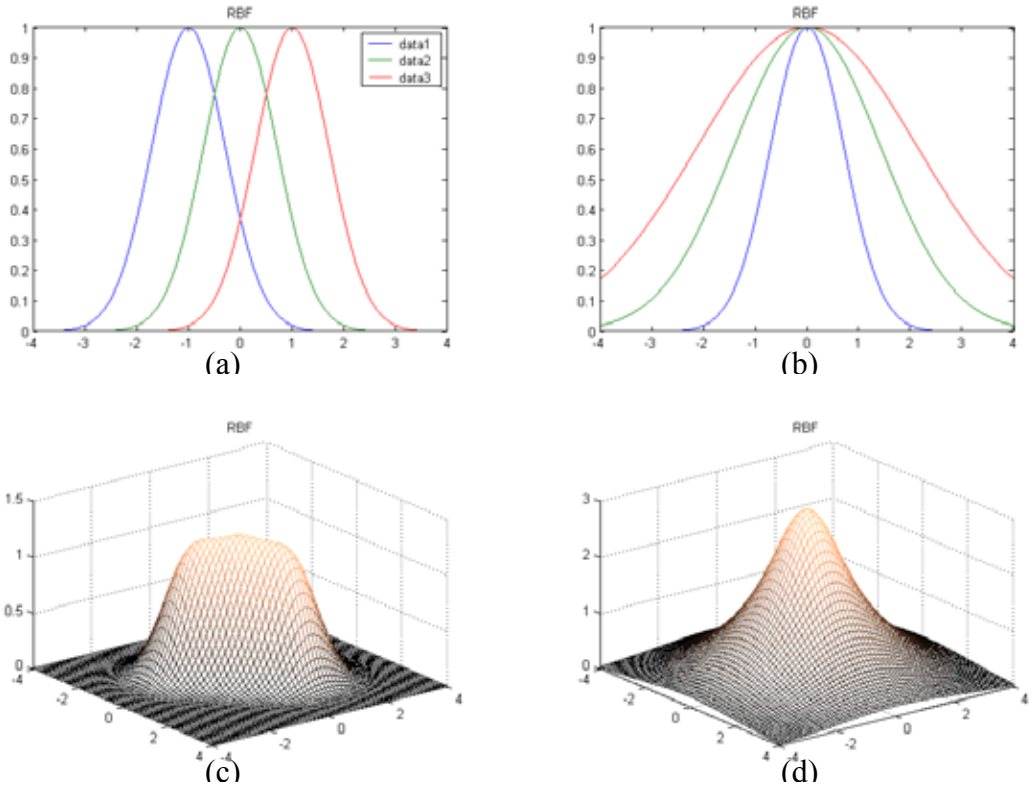

Figure 3.7 Influences of μ_i and σ_i^2

Figure 3.6 (a) and (b) show the impact of μ_i and σ_i^2 while (c) and (d) show the composite representation of the three activations summed. The transfer function associated with any of the intermediate neuron is defined as:

$$y = Wz(x) \tag{3.15}$$

where $y = [y_1\ y_2\ y_3 \cdots y_m]$ assuming ***m*** number of neurons in the secondary layer. The output vector $z(x)$ becomes $[z_1(x)\ z_2(x)\ z_3(x) \cdots z_n]$ defining n number of output neurons. The resulting weight matrix W forms an m x n matrix. This matrix is the adapted weights for connecting the *jth input* neuron with the *ith* neuron in the second hidden layer. This decomposes the overall network into two workable subcategories. The first containing the modified weights, the second vector $z(\cdot)$ being the structural parameters of the network.

3.2.1.1 RBF neural network as nonlinear modelers

There has been recent work in determining the suitability of radial basis functions to accurately approximate non-linear models [21]. To expand, the authors provide a defense for a property known as "*The universal approximation property*". In it, the statement is made that for any continuous function f(·) and some ε defined over a bounded set C, there will always exist a matrix W and vector z(x) such that:

$$\max_{x \in C} \left| f(x) - \overline{W}\overline{z}(x) \right| < \varepsilon \tag{3.16}$$

In essence, the property guarantees that a network exists to describe the system so long as the system is continuous and defined over a compact region of operation. This is a corollary to local modeling techniques described in the previous chapter. In that, assume some function **f** is defined over a finite range, this range can be sub-divided into smaller local regions and it is in these regions that a RBF is applied to sufficiently perform a curve fit. A graphical representation is given in Figure 3.8 below. This graph shows how

three RBF's are utilized to generate a rough outline to that of the original non-linear functions indicated by the solid line. A little thought and insight to the preceding statements indicate that for higher complexity systems, the network of RBF functions can grow considerably large.

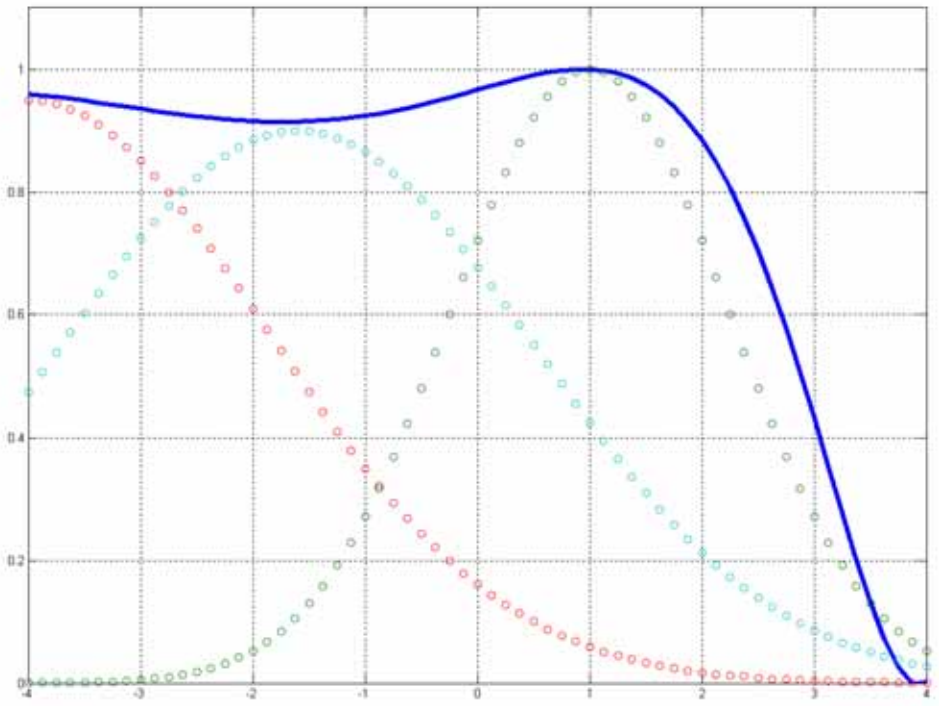

Figure 3.8 RBF

The actual summed version is provided in Figure 3.9. With the lighter line showing the RBF summed curves.

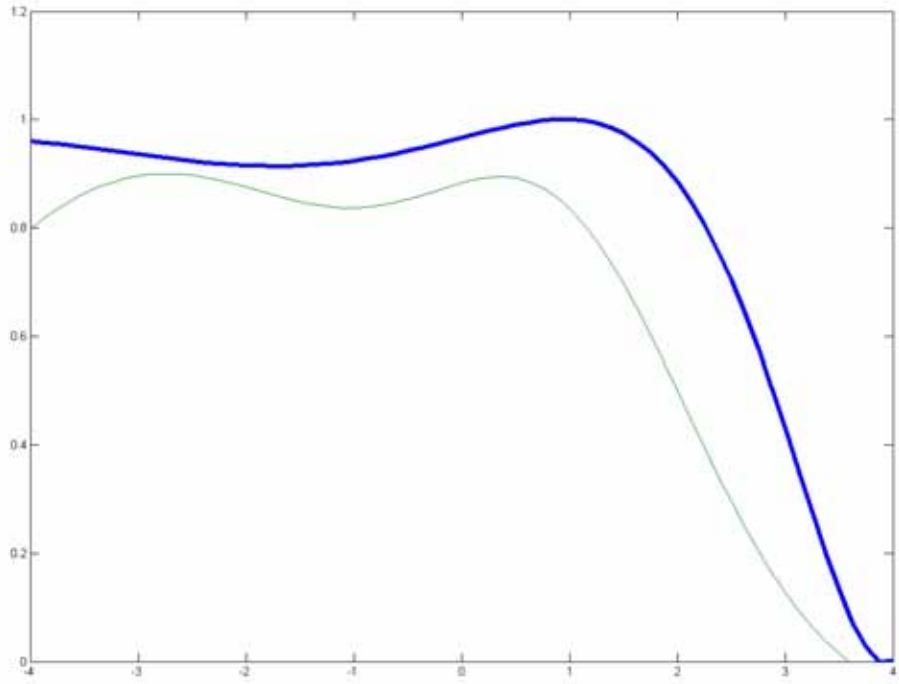

Figure 3.9 Composite RBF signal

While there is still considerable error associated with the above, increasing the number of RBF's increases the fit and reduces the error. To this end, the radial basis network is often employed into a more powerful scheme known as multi-scale modeling. In this, it is known that any system can be represented as having both global and local features. The multi-scale model strikes a balance between the two.

3.2.1.2 Multi-scale Structure

The previous sectioned describe the RBF such that the center locations were defined at some fixed interval Δx between them. Under the multi-scale model an attempt to construct sufficient width levels and center locations for the prescribe RBF is made.

This can be represented as:

$$y = \left[W_{0,} W_1, \ldots, W_d\right] \begin{bmatrix} z_0(x) \\ z_1(x) \\ \vdots \\ z_d(x) \end{bmatrix} \tag{3.17}$$

or more concisely written:

$$y = \sum_{i=0}^{d} W_i z_i(x) \tag{3.18}$$

maintaining the previous nomenclature, W_i is a subset of the overall weight matrix W, z_i(x) is the set of neurons at scale i with d defining the localized minimum operating range.

At each of the scales i, z_i is represented as:

$$z_i(x) = \begin{bmatrix} z_{i1}(x) \\ z_{i2}(x) \\ \vdots \\ z_{in}(x) \end{bmatrix} \tag{3.19}$$

where

$$z_{ij}(x) = e^{\frac{-\left\|2^i x - p_{ij}\right\|^2}{\sigma^2}}, j = 1,2,3,\ldots n \tag{3.20}$$

One of the key features of utilizing this technique is that the multi-scale RBF effectively decomposes the model into several smaller partial models. Once each of these is calculated, they can be collectively summed to produce the global representation of the system [22].

3.2.2 State-space Neural Networks

In general, a state space model can be thought to fall into one of two categories. One being knowledge-based modeling, and the other being black-box modeling. To give a better understanding of what each of these entail some short definitions are in order.

In a knowledge-based model, a model is assumed if it is possible to construct it based upon prior knowledge of the system. These models typically relate their state variables to their physical meaning. The black-box model, on the other hand, can be thought of more in terms of an input-output relationship. This latter model scheme is what is utilized in the modeling of an unknown process. From this standpoint, the concepts of black-box modeling will be expounded upon. It is interesting to note, that in Suykens et al, a state space neural network can be guaranteed to be globally asymptotically stable based upon the NL_q stability theory [26].

3.2.2.1 Black-box modeling

The concept of this scheme is somewhat simple, in that three criteria are set to accomplish the task. First, an appropriate candidate model is chosen to utilize in the modeling process. Second, the systems predictors are determined. Thirdly, the best model is selected.

Consider a nonlinear system in the form:

$$\begin{cases} x_{k+1} = f(x_k, u_k) + \vartheta_k \\ y_k = g(x_k, u_k) + \psi_k \end{cases} \tag{3.21}$$

where y_k denotes the output of the process, u_k is the known external inputs, and x_{k+1} is the state vector. The additive terms ϑ_k, ψ_k denote the state noise and output noise respectively. For our modeling attempts it should be noted that the state x_{k+1} is not measurable.

Continuing, the derivation of the associated predictors is essential. In general, a theoretical predictor is the conditional expectation $E(y_p(k+1)|k)$ of the output. In this, it

is assumed that the past observations $\{y_p(k), y_p(k-1), \ldots, y_p(0)\}$ are given. The predictor is defined as:

$$y(k+1) = h_{pred}(y_p^k, u^k, y^k, k) \tag{3.22}$$

where y is the predictor output, and y_p^k denotes a series of past outputs. The nonlinear mapping function h_{pred} is defined. From 3.22 above, a state-space representation can be formulated as:

$$\begin{cases} x(k+1) = f_{pred}(x^k, u^k, y_p^k, k) \\ y(k+1) = g_{pred}(x^k, u^k, y_p^k, k) \end{cases} \tag{3.23}$$

where, x is the state vector, and f_{pred} and g_{pred} define non-linear mappings. The functions f_{pred} and g_{pred} are then replaced by functions parameterized by a set of parameters θ. The state-space neural predictor becomes:

$$\begin{cases} \xi(k+1) = \phi(\xi(k), u(k), y_p(k); \theta) \\ y(k+1) = \psi(\xi(k), u(k), y_p(k); \theta) \end{cases} \tag{3.23}$$

where ξ is the n-state vector that will be trained to minimize the Mean Square Prediction Error (MSPE) of the training set. It has been shown in Hornik et al. [23] that any universal function approximator can be used. Sontag and Sjoberg [24, 25] show other potential approximators as well. From this point, it becomes important to understand how the predictors are to be properly conditioned for training.

3.3 Training Predictors

The ultimate goal in any neural modeling is to determine the best neural predictors of the provided candidate models. These, in turn, are defined to provide the

best fit for the model. The Prediction Error (PE) or Extended Kalman Filtering (EKF) approach is often employed in the training of the candidates. The EKF approach, however, has a major drawback. In order for this approach to provide sufficiently accurate results, the noise covariance of the system must be known. This in itself is highly unlikely since a knowledge of the system to which the model is being applied is not typically available. To this end, the PE method is used and provides a convergent solution.

To choose the best candidate, a minimization of the cost function is utilized. The cost function is defined as the MSPE of a training sequence having N samples. An iterative approach to this is defined as:

$$J(\theta^i) = \frac{1}{N}\sum_{k=0}^{N-1}(e^i(k+1)^2 \tag{3.24}$$

where e^i is the error of the output in terms of the predictor at times k and iteration i. Substituting this equality into 3.24 yields:

$$J(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}\left((y_p(k+1) - y^i(k+1)\right)^2 \tag{3.25}$$

The next logical question is defining a method by which 3.25 can be minimized. To this end, the cost function $J(\theta)$ can be minimized utilizing the gradient method as defined in the previous chapter as well as utilization of a quasi-Newtonian method. Other methods have also been proposed which include the teacher forcing algorithm [27], and Backpropagation through time [28]. For a more in depth overview of state space neural network training, refer to Rivals [29, 30, 31].

3.4 Remarks

While the concept of neural networks have been studied for more than 50 years, it was not until more recently that they are becoming a more active research focus. Many researchers have shown that neural networks are capable of simulating and modeling real world processes. Many forms and models exist that can be utilized in setting up a suitable network, the key criteria for selecting a model should be based upon its performance measurement. This measurement gives the observers a relative view of how well the network will actually simulate the problem presented to it.

Chapter 4

4.0 The DSAW Process

The DSAW process as outlined in the previous chapter is the merger of two independent welding technologies. The key desire of the process is to be able to control the duration that the process operates in the keyhole process. Remembering that this event, if left unattended, is the same as a cutting torch. The duration of the keyhole cannot always be maintained as desired due to uncertainties in the welding process. To this end, a peak current I_p is applied to establish the keyhole and the current is switched to a lower level called base current once the keyhole is established. The objective of the control is to establish the keyhole within a certain period, which is called the keyhole establishment time. Because the current is switched to the base level once the keyhole is established, the keyhole establishment time is thus the same as the period during which the peak current is applied. Hence, the peak current period T_p is the output and the peak current I_p is the input of the process to be controlled.

4.1 Modeling of the DSAW Data

Up until this point, the focus has been on providing a general synopsis on some of the current techniques employed in nonlinear modeling of dynamic systems. To this end, an overview of the neural network architecture has been presented and expanded upon. The inclusion of both a foundational history and current modeling structures serves to better prepare the reader for the analysis and construction of the Double Sided Arc Welding (DSAW) process model as first outlined in a previous chapter. As was previously mentioned, the establishment of a keyhole in the work material is a function of the current and duration. The input to the network will be the welding peak current per weld cycle while the output is the total time duration of that particular level. Each of the inputs and outputs have been normalized. The peak current ratio is 239:1 while the peak current duration has a ratio of 5074:1.

The normalized sequences are outlined in Figure 4.1 below.

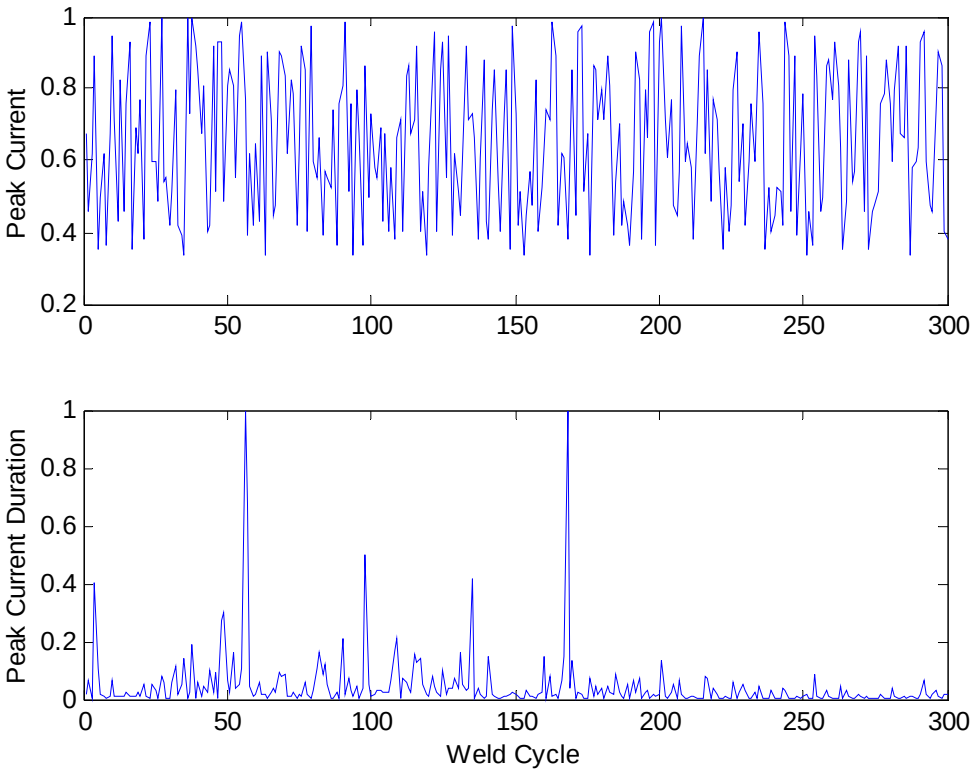

Figure 4.1 Training and Test Sequences

Now that a sample set has been provide, the goal is to separate the sequence into a training set by which the network will be trained and a test set to test the ability of the network to simulate the process.

To this end, the training and test sets are outlined in Figure 4.2 below.

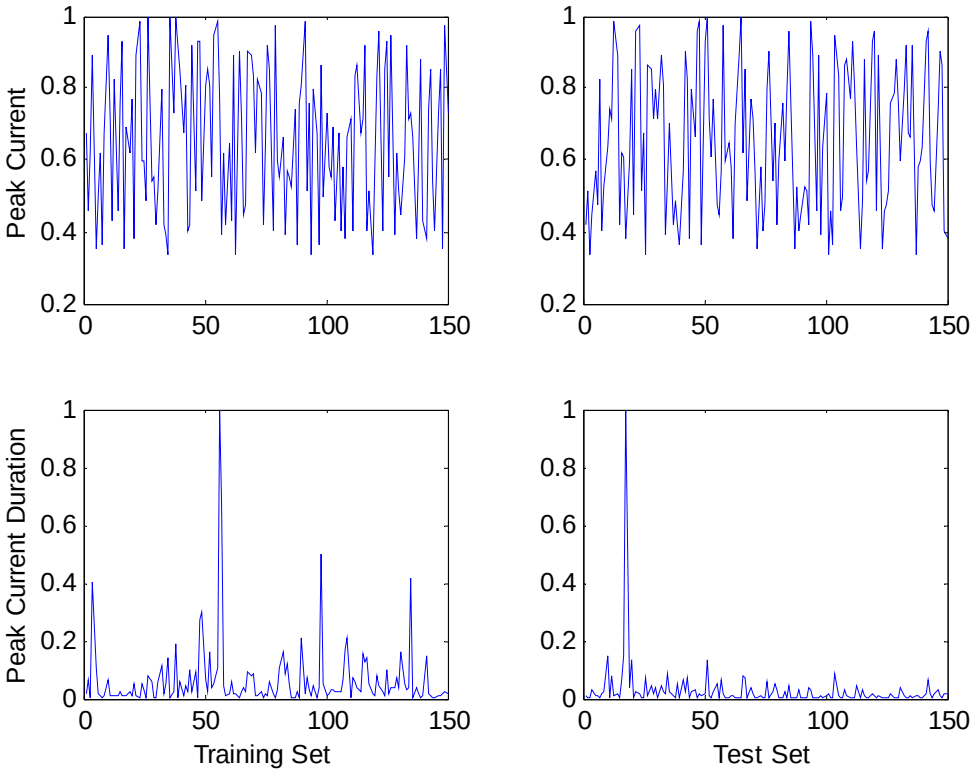

Figure 4.2 Partitioning of the training and test set.

The complexity of this system involves the seemingly noise input for the peak current intensity. The goal of the modeling procedure will be to develop a suitable simulator of the process. In each instance of a chosen predictor, a feedback model is used. Overall performance of the simulated models will be based upon the MSPE of the test sequence.

4.1.1 NARX Assumed Model

The NARX feed-forward predictor will be assumed for the process. The structure of the feed-forward predictor is defined as:

$$y(k+1) = \phi(y_p(k),...,y_p(k-n+1),u(k),...,u(k-m+1)) \tag{4.1}$$

The inputs to our network will be the output y_p and u, where y_p is the predicted output. A generalized block can be realized as presented in Figure 4.3.

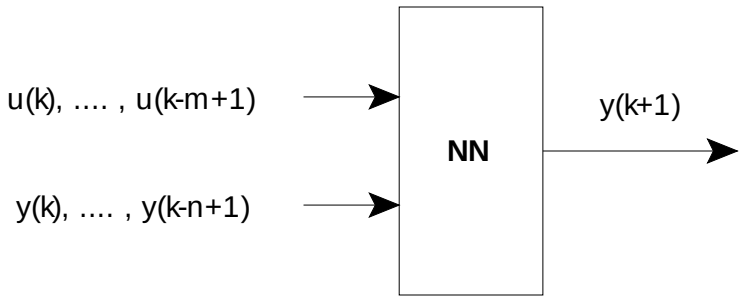

Figure 4.3 Neural NARX predictor

Two hidden neurons will initially be used to learn the system. The performance will then be compared with the same network, but with five hidden neurons. The goal here is to see if increasing the number of neurons actually help the system performance. This methodology will be employed in each of the test cases.

Graphically, the implementation of the network architecture is shown in Figure 4.4. Each node is fully connected.

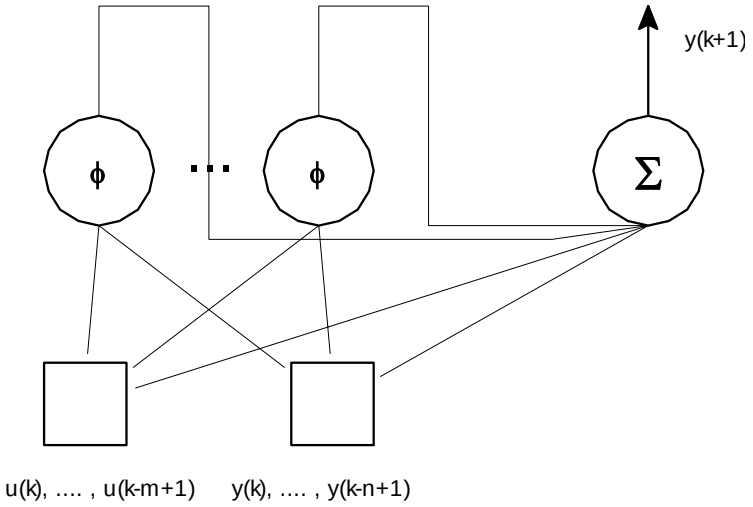

Figure 4.4 Fully connected NARX predictor

4.1.1.1 NARX Network Response

The performance criteria used for the training and testing of the network utilizes the mean square error (MSE). In each trial, a 150 sample training set is presented to the network. The network then trains itself for 2500 epochs in an attempt to establish a suitable process fit. The initial weights of the network are generated randomly and the process is simulated to provide a base line.

The initial response is shown in Figure 4.5.

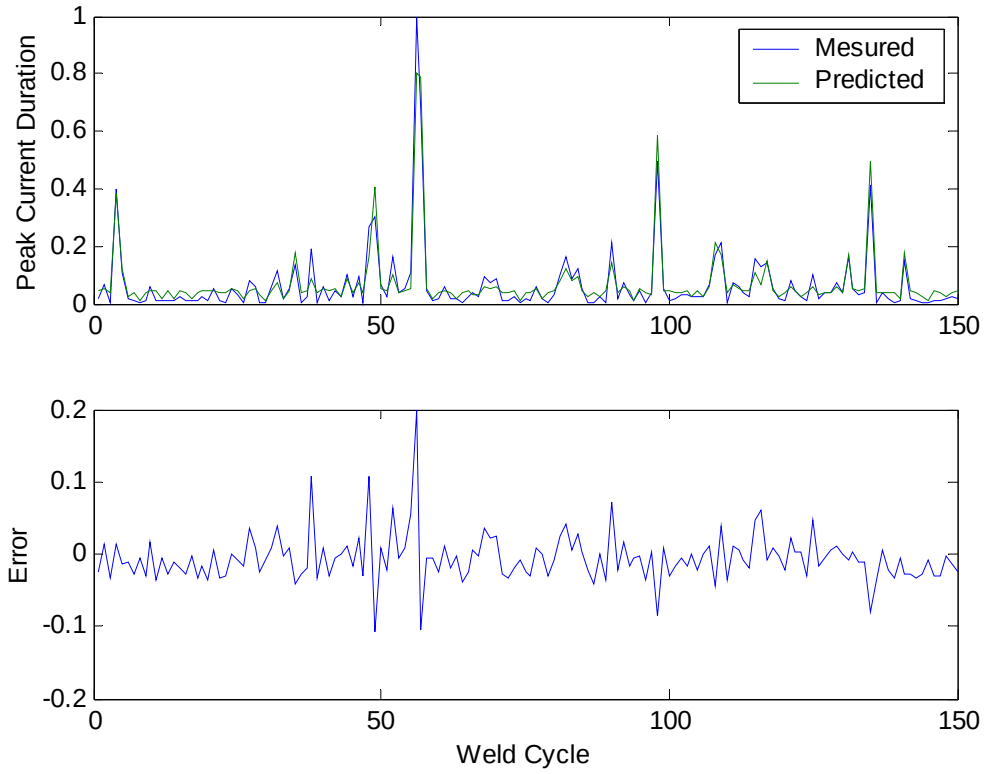

Figure 4.5 NARX response to the training set.

The blue line in the graph indicates the measured data. The predicted output of the network is indicated with the green line. As can be noted, the initial response to the training set appears to have a fairly good fit. After training, it is noticeable that the network acceptably describes the process with minimized error. The MSE performance of the network for this training set is 0.0012.

Figure 4.6 indicates the progression of the error as the training epochs increases.

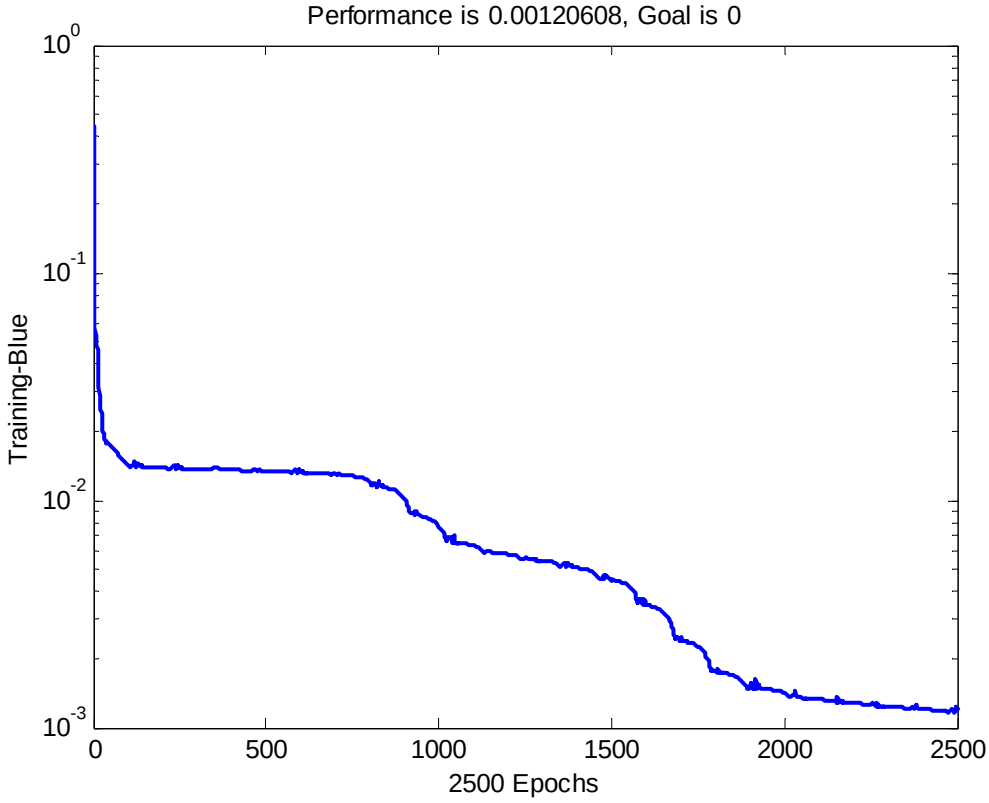

Figure 4.6 MSE Performance of the NARX predictor

After the initial training is accomplished, the task is to verify that the network truly describes the process in question. To accomplish this, our test set is now presented to the network. The network is then simulated with this new set of inputs. After simulation, the MSE of the test set has slightly poorer performance than the training set and comes in at 0.0181.

Comparing the response of the training set and test set is depicted in Figure 4.7.

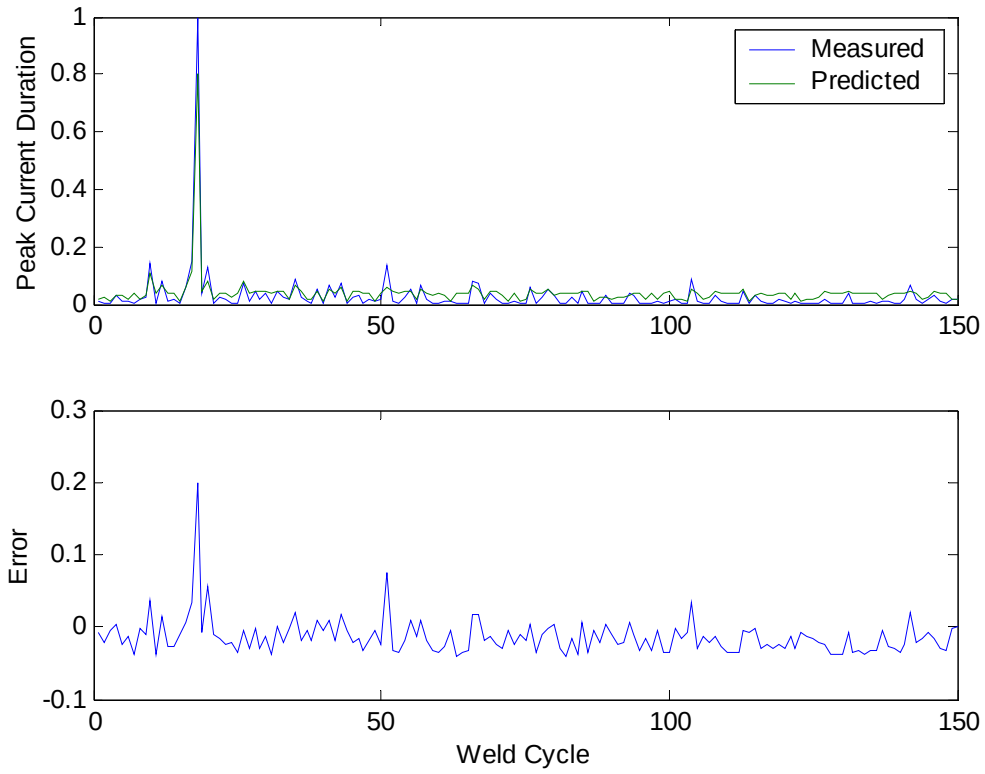

Figure 4.7 NARX Response to the test set.

The interconnecting weight matrix of the network was found to be:

$$\begin{bmatrix} -4.1483 & 3.1312 \\ -1.8142 & 4.5170 \end{bmatrix}$$

Input to hidden node

$$\begin{bmatrix} -0.10001 & 0.48806 \end{bmatrix}$$

Hidden node to output

$$\begin{bmatrix} 1.8124 \\ -0.3609 \end{bmatrix}$$

Hidden node biases

$$\begin{bmatrix} 0.41238 \end{bmatrix}$$

Output node bias

Increasing the number of hidden neurons to five within the network does provide better performance for the system as is shown in Figure 4.8 below.

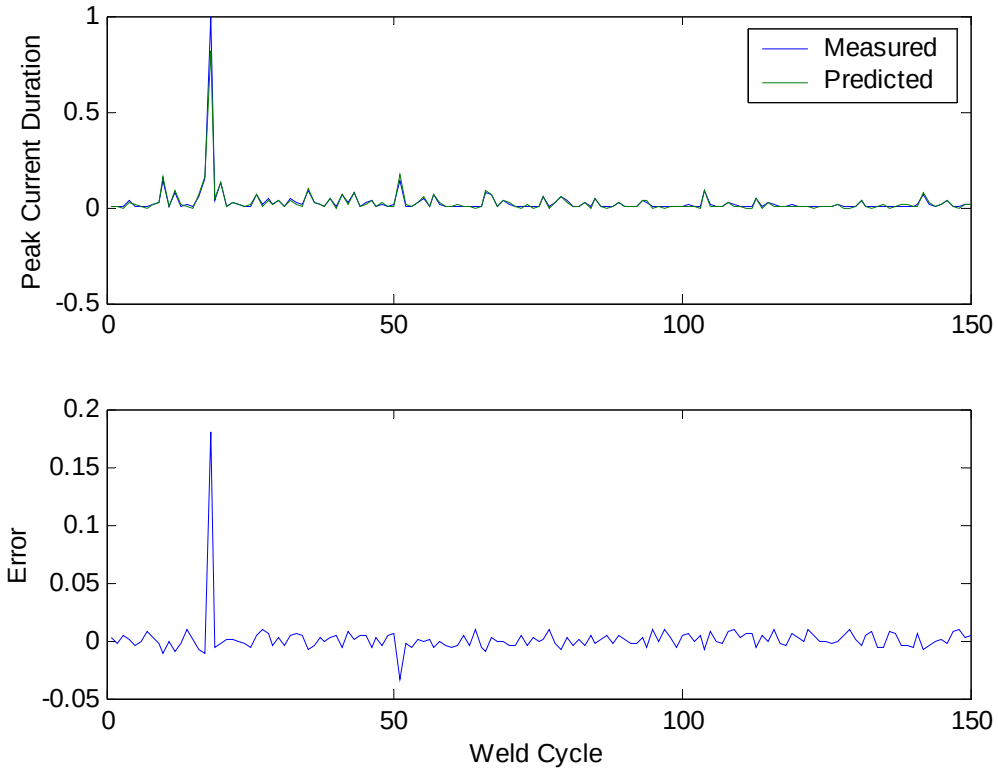

Figure 4.8 Five Neuron NARX response

4.1.2 NARMAX Assumed Model

One of the simplest input-output models that can be represented is the NARMAX. As a second experiment, the NARMAX is assumed to be the model of the process and the predictor is defined as:

$$y(k+1) = \phi(y_p(k),...,y_p(k-n+1),u(k),...,u(k-m+1),e(k),...,e(k-p+1)) \qquad (4.2)$$

Like the NARX model, the NARMAX can be represented as the NARX with the addition of an error term. A proposed neural architecture is presented in Figure 4.9.

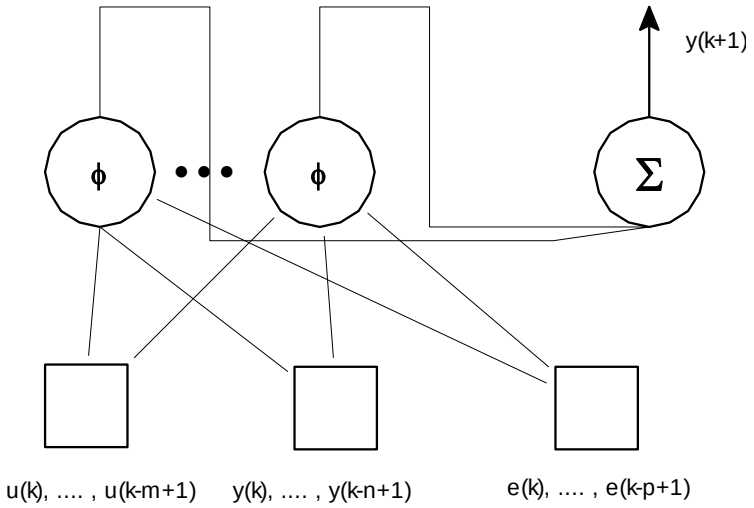

Figure 4.9 NARMAX Predictor

4.1.2.1 NARMAX Network Response

Like the NARX assumption, the performance criteria, number of samples and experimental methodology are the same, so as to provide a valid comparison between the two assumed models. That said, the MSE of the NARMAX predicted system does perform slightly better than the original NARX assumed model. The performance of the MSE comes in at 0.0013, on par with the NARX assumption.

The trained network response is shown in Figure 4.10

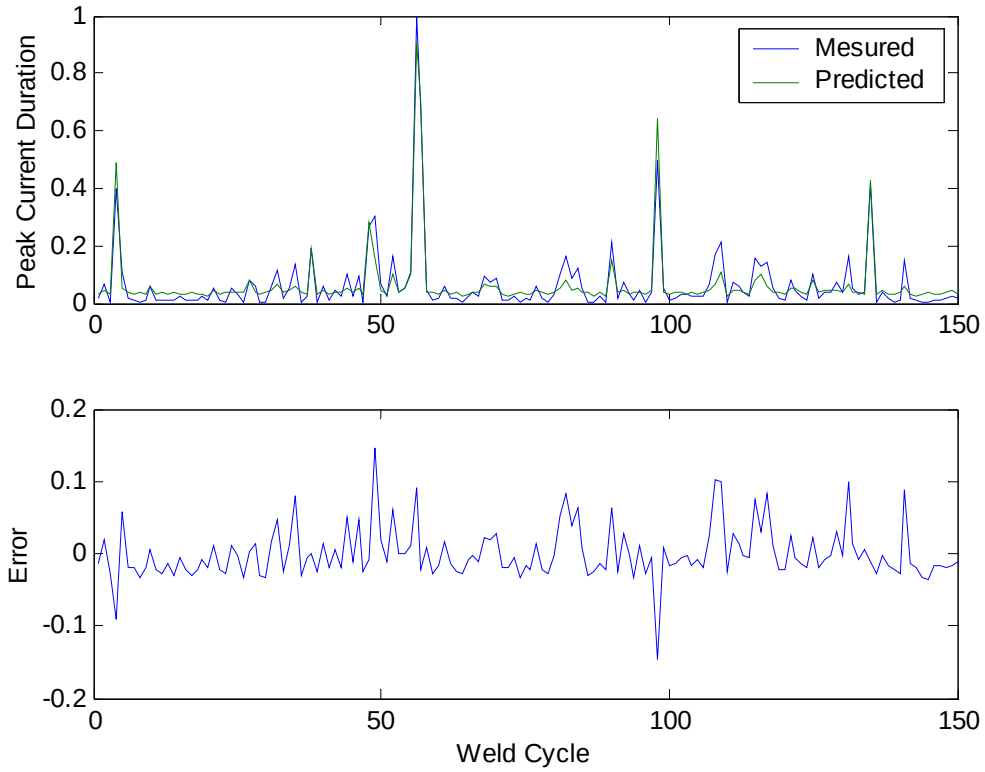

Figure 4.10 NARMAX response to the training set.

The MSE progression over the 2500 epochs of training is shown in Figure 4.11.

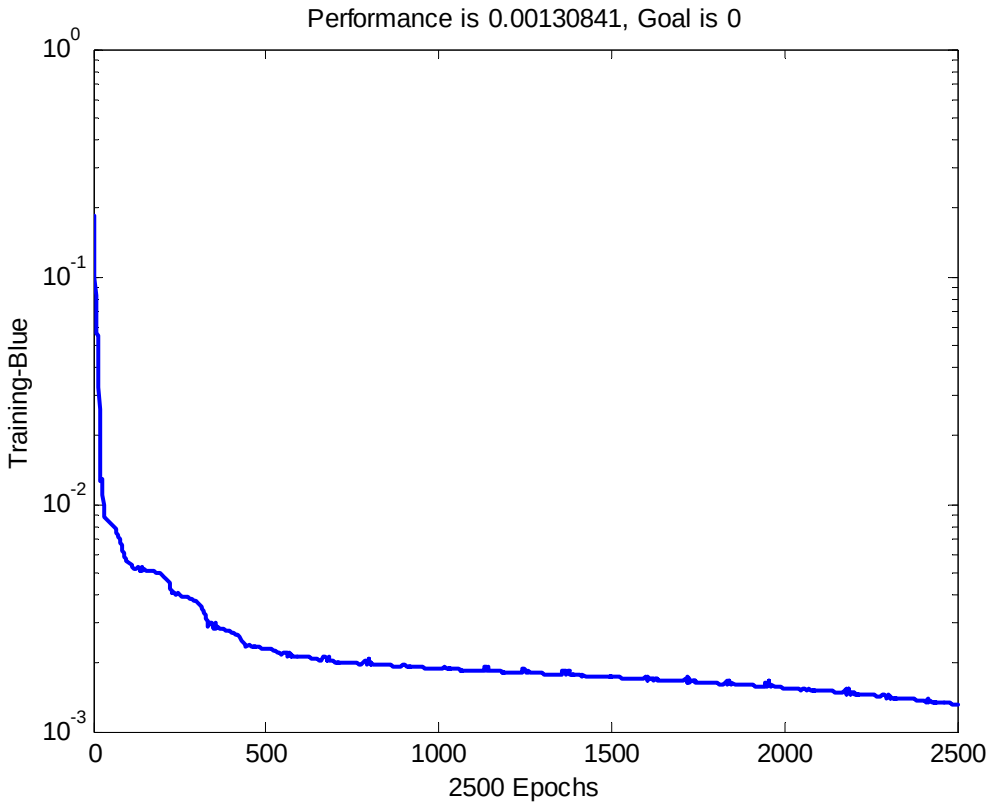

Figure 4.11 MSE Performance of the NARMAX predictor

After simulation with the test set, the MSE performance is 0.0069. This is once again better than the NARX response for the same test set. From that standpoint, it can be safe to draw the conclusion that either one of the assumed models should provide similar responses.

The final comparison for the training set and test set is shown in Figure 4.12.

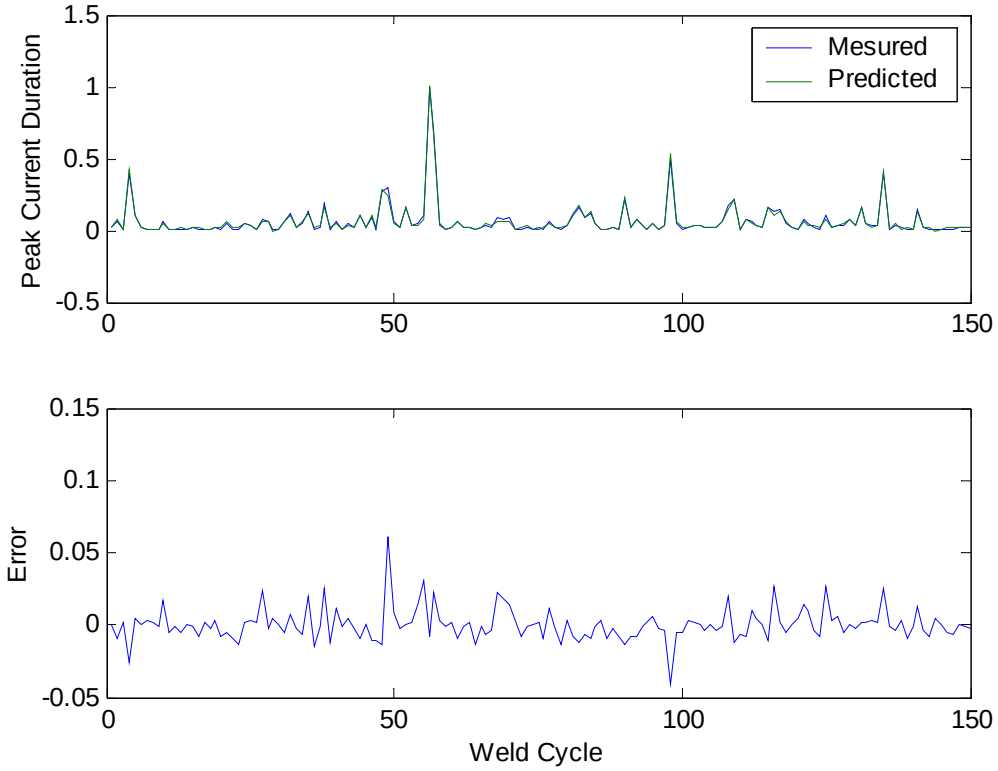

Figure 4.12 NARMAX response to the test set.

The interconnecting weight matrix of the NARMAX network was found to be:
Initial untrained network weights:

$$\begin{bmatrix} 1.2124 & 3.9457 & -0.01461 \\ 3.438 & 2.3186 & 0.66432 \end{bmatrix}$$

Input to hidden node

$$\begin{bmatrix} 0.47567 & -0.032495 \end{bmatrix}$$

Hidden node to output

$$\begin{bmatrix} -2.561 \\ -2.293 \end{bmatrix}$$

Hidden node biases

$$\begin{bmatrix} 0.47535 \end{bmatrix}$$

Output node bias

Increasing the number of hidden neurons to five within the network does not provide better performance for the system as is shown in Figure 4.13 below. In this instance, the network over fits the data and thus the increased errors.

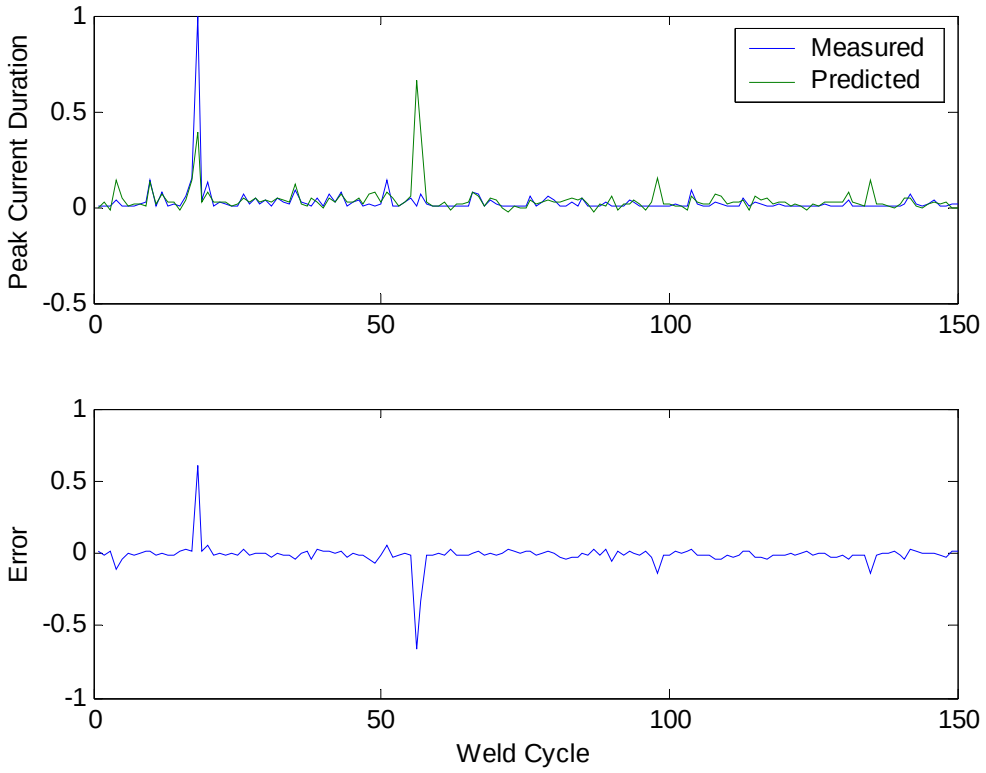

Figure 4.13 Five neuron NARMAX response

4.1.3 State Space Assumed Model

The given predictor for a state-space model is given by the following set of equations:

$$\begin{aligned} \xi_1(k+1) &= \phi_1(\xi_1(k),...,\xi_n(k),u(k),y_p(k)) \\ \xi_n(k+1) &= \phi_n(\xi_n(k),...,\xi_n(k),u(k),y_p(k)) \\ y(k+1) &= \psi(\xi_1(k+1),...,\xi_n(k+1)) \end{aligned} \tag{4.3}$$

The proposed state space model utilizes a two neuron hidden network and is graphically represented in Figure 4.14.

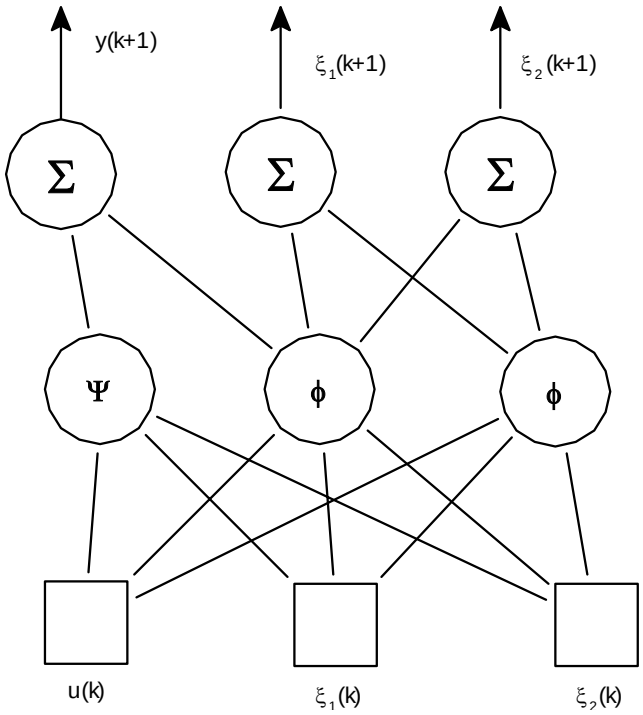

Figure 4.14 State Space Predictor

4.1.3.1 State Space Network Response

Out of the three proposed models, the state-space model performs poorest. The MSE is 0.0133, which, in comparison, is almost an order of magnitude less than the previous two. The response to the training set is shown in Figure 4.15.

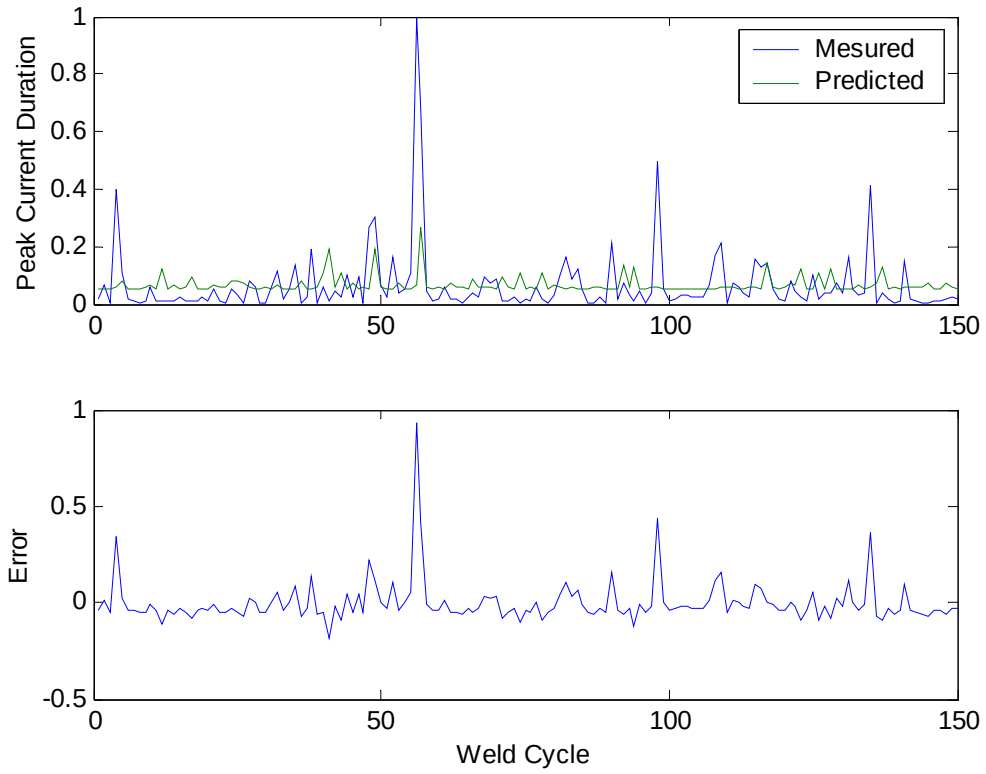

Figure 4.15 State-space response to the training set.

The state-space assumed model lacks in sufficiently predicting the system response .

The MSE progression over the 2500 epochs of training is shown in Figure 4.16.

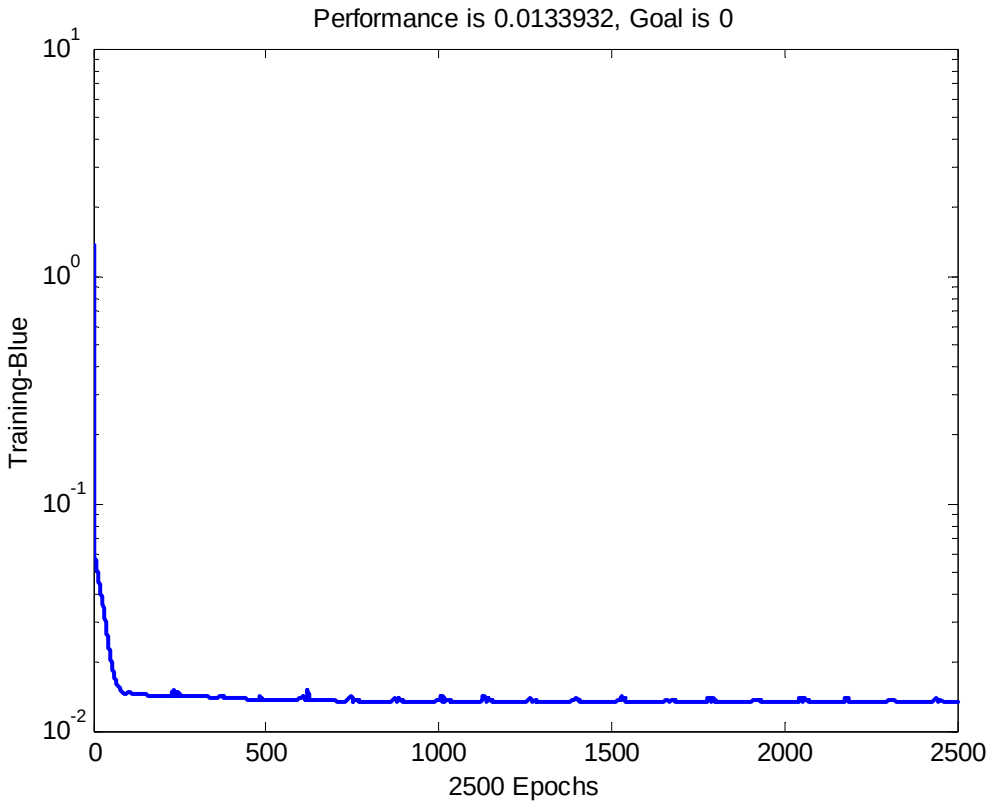

Figure 4.16 MSE Performance of the State-space predictor

As can be seen, the MSE fails to converge to an acceptable level. The model is essentially stuck. After simulation with the test set, the MSE performance is 0.0076. The interesting fact here, is that the MSE of the test set is better than the training set? Perplexing thought, what could this be attributed to? Looking carefully at the training set and test set, one can notice that the test set has less spurious peaks occurring. The overall response of the network was flat for all intensive purpose. The greatest error is due to the one peak.

The final comparison for the test set is shown in Figure 4.17.

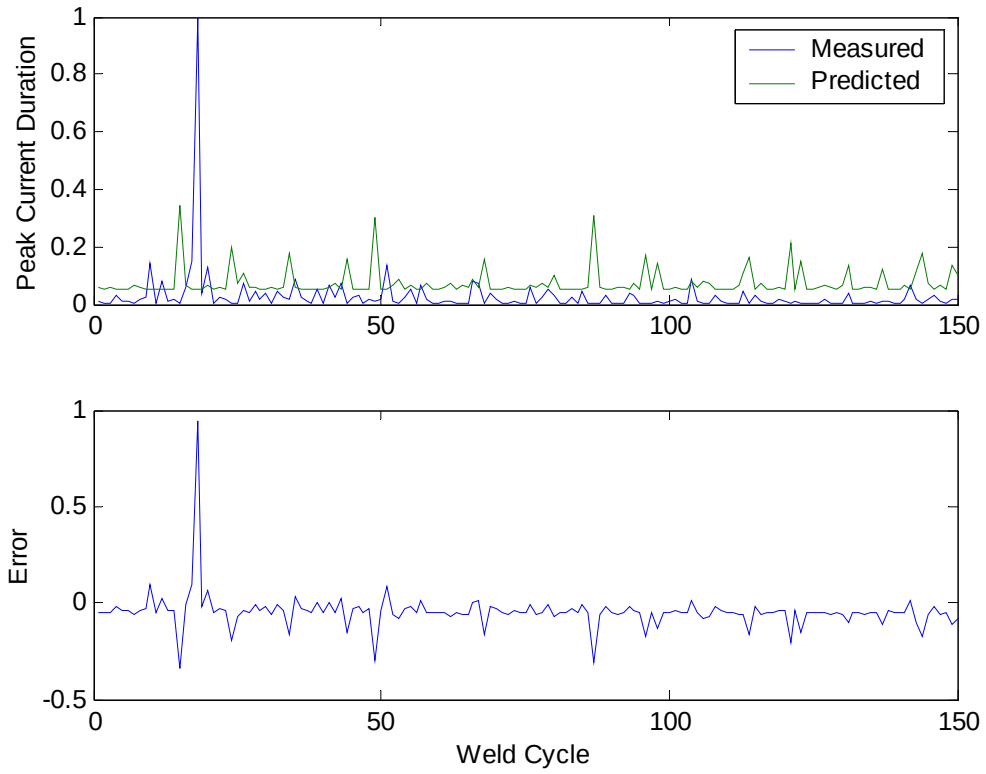

Figure 4.17 Training set to Test Set NARMAX Network Response

The interconnecting weight matrix of the state-space network was found to be:
Initial untrained network weights:

$$\begin{bmatrix} -3.054 & 1.2982 & 3.1384 \\ -3.180 & 2.0701 & 2.7898 \end{bmatrix}$$

Input to hidden node

$$\begin{bmatrix} -0.018746 & 0.59671 \end{bmatrix}$$

Hidden node to output

$$\begin{bmatrix} 0.027708 \\ -4.6835 \end{bmatrix}$$

Hidden node biases

$$\begin{bmatrix} 0.6622 \end{bmatrix}$$

Output node bias

Increasing the number of hidden neurons to five within the network does not provide better performance for the system as is shown in Figure 4.18 below. In this instance, the network over fits the data and thus the increased errors.

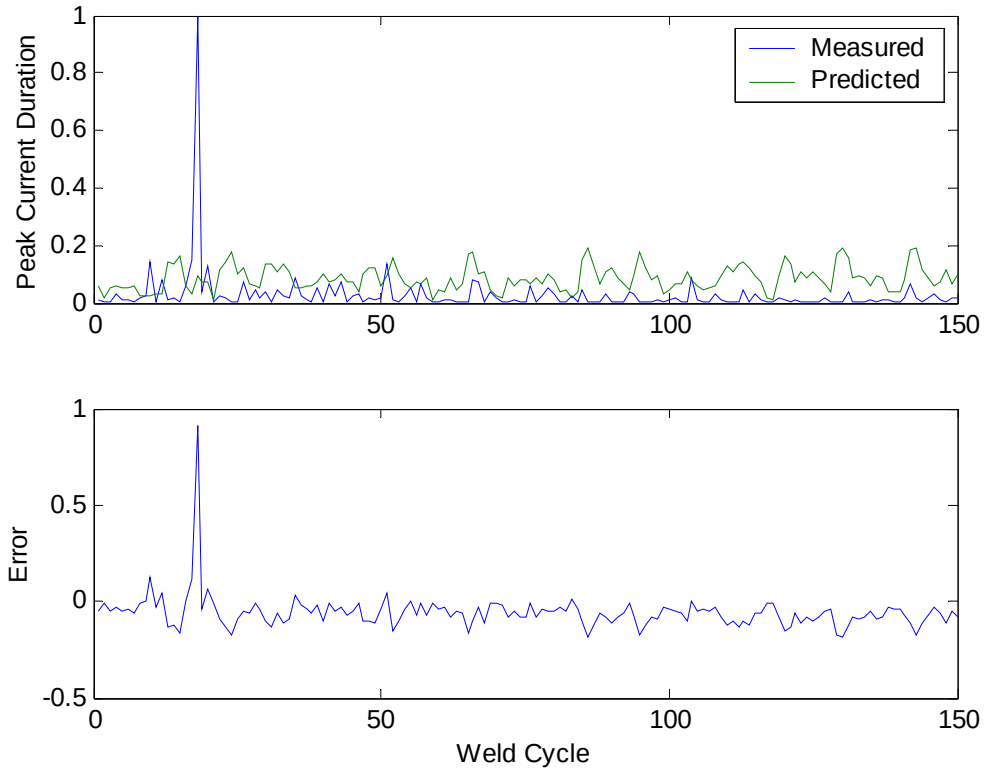

Figure 4.18 Five neuron State Space response.

4.2 Remarks

After careful review of the three proposed network structures, it becomes obvious that the NARX and the NARMAX models provide the best performance. The NARMAX structure provided the best results. Oftentimes, to improve the MSE, the networks are trained to a much higher degree- often approaching tens of thousands of epochs. While this extra training does provide improvements, overall, the performance for these particular cases did not improve significant enough to warrant the extra training. Another goal of the network is to be of minimal size. Increased network size leads to over-fitting

and allows additional system noise in, not to mention that the speed of the network is also decreased.

Chapter 5

5.1 Conclusions

The primary focus of this research has been to provide a fundamental knowledge and understanding of the existing technologies, to solve problems associated with non-linearity. More specifically, a basis of the neural network was provided as an approachable mechanism by which to solve more complex problems.

To develop a working model of a system, an acceptable data sample should be collected of the typical process and provided as the inputs to the network to the be trained. Aptly selecting the data set provides a generality for the network. The goal is not to provide a complete picture to the network but rather a snapshot, so that the network is able to make an "educated" guess of what the response should be.

Within the neural network framework several structures exist to accomplish the modeling task. Structures ranging from the NARMAX, NARX, and state-space models may all be utilized. While these tend to provide good results, there does exist a wealth of other network structures that may be used.

The DSAW process provided a dynamic data set to functionally assert the networks features. The establishment of the keyhole evolution, control of the welding current provide a window into the evolution of an acceptable weld. The neural network provides a means by which to effectively predict at what stage of the development the process may be in. The key once again, is to provide an acceptable date set to show the network how the process behaves.

A key benefit of the network, other than its ability to predict process response, is to be able to control the level of dimensionality of the simulating model. Some of the other non-linear techniques as presented to work but at high cost to processing power and number of variables that must be maintained and controlled. The "curse of dimensionality" is always of great importance.

5.2 Future work

This thesis is but the tip of the iceberg for the realm of possibilities that can stem from it. The DSAW process is very dynamic in nature and the need for an accurate control system is desired. To this end, further research is necessary in the fields of control algorithms associated with neural modeling and process control.

An improved understanding of the environmental aspects that influence the evolution of the weld, as well as, an increased understanding of the heat transfer can provide further development of the neural model. A neural sensor may be developed for online monitoring of the critical parameters. This in turn, would provide the necessary inputs to the neural controller. One of the powerful features of the network is its ability to scale with a process. As a process becomes more complex and other variables are factored into the process, the neural controller is able to adapt and provide the necessary result.

Finally, a process can be expanded, in that, not only one process can be controlled, but rather a multitude of simultaneous weld operations could be performed and controlled in succinct succession.

REFERENCES

[1] Sapp, M (2005) [Online] Available from World Wide Web: http://weldinghistory.org/

[2] Y. M. Zhang and S. B. Zhang, 1999. "Method of arc welding using dual serial opposed torches," U. S. Patent, No. 5,990,446.

[3] S.I. Rokhlin and A.C. Guu, " A study of arc force, pool depression, and weld penetration during gas tungsten arc welding " , *Welding Journal*, Vol. 72, PPs. 381s-390s, 1993

[4] Y. M. Zhang, S. B. Zhang, and M. Jiang, "Keyhole double-sided arc welding," in review for Welding Journal.

[5] S. Subramanian, D.R. White, J.E. Jones, and D.W. Lyons, "Experimental approach to selection of pulsing parameters in pulsed GMAW ", *Welding Journal*, May 1999, 166s-172s.

[6] V. Volterra, *Theory of Functionals and of Integro and Integro-Differential Equations*. New York: Dover, 1959. Reprint of 1930 edition.

[7] N. Wiener, *Nonlinear Problems in Random Theory*. Cambridge, Mass.: M.I.T. Press, 1958.

[8] W. J. Rugh, *Nonlinear System Theory: The Volterra/Weiner Approach*. Baltimore: John Hopkins University Press, 1981.

[9] S. Boyd and L. O. Chua, "Measuring Volterra Kernels," *IEEE Trans. Circuits and Systems*, vol. 30, March 1983, pp. 571–577.

[10] Johansen, T.A. and B.A. Foss, 1992, "A NARMAX model representation for adaptive control based on local models", Modelling, Identification and Control , vol. 13, no. 1:25-39

[11] Johansen, T.A. and B.A. Foss, 1993, "Constructing NARMAX models using ARMAX models", International Journal of Control, 58: 1125-1153.

[12] Gray, G., Murray-Smith, D., Li, Y., and Sharman, K., "Nonlinear system modelling using output error estimation of a local model network"

[13] Topfer, S., Wolfram, A., and Isermann, R., 2002 "Semi-physical modelling of nonlinear processes by means of local model approaches", 15th Triennial World Congress, Barcelona, Spain

[14] Murray-Smith, R. and T.A. Johanson, 1997. “Multiple Model Approaches to Modelling and Control”, Tayler & Francis, London, U.K.

[15] McCulloch and Pitts, W (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115-133

[16] Rosenblatt, F. (1962). Priciples of Neurodynamics. Spartan Books

[17] Widrow, B. and Hoff (1960). Adaptive switching circuits. 1960 IRE WESCON Convention Record, 96-104

[18] M. Minsky and S. Papert. Perceptrons. MIT Press, 1969

[19] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. Nature, 323:533 – 536, 1986

[20] K.J. Hunt, D. Sbarbaro, R. Zbikowski and P.J. Gawthrop (1992), “Neural networks for control systems – A survey”, Automata, Vol. 28, 1083-1112

[21] Park, J., Sandberg, J.W. (1991) “Universal approximation using radial basis functions network”, Neural Computation, vol 3. 246-257

[22] Y. YU, S. Tan, J. Vandewalle, and E. Deprettere (1996), “Near-optimal construction of wavelet netwroks for nonlinear system modeling”, Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS’96), Vol 3. 48 – 51

[23] Hornik, K., Stinchcombe M, White H, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359-266

[24] Sontag E.D, Neural Networks for control, Essays on control:perspectives in the theory and its applications, Trentelman H.L and Willems J.C. eds (Birkhauser, Boston) (1993) 339-380

[25] Sjoberg J., Zhang Q., Benveniste A., Deylon B., Glorennee P., Hjalmarsson H., Juditsky A., Ljung L. Nonlinear black-box modelling in system identification: model structures and algorithms, Automatica (1995)

[26] J. Suykens, J. Vandewalle and B. De Moor, Artificial neural networks for modeling and control of non-linear systems, Kluwer Academic Publishers, Boston (1995)

[27] Jordan M.I., The learning of representations for sequential performance. Doctoral Dissertation, University of California, San Diego (1985)

[28] Rumelhart D.E., Hinton G.E., Williams R.J. Learning internal representations by error back-propagation, Parallel Distributed Processing: explorations in the microsystems of cognition. Vol 1. MIT Press, Cambridge, MA. 1986 318-362

[29] Rivals I., Canas D., Personazz L., Dreyfus G., “Modeling and control of mobile robots and intelligent vehicles by neural networks”, IEEE Conference on Intelligent Vehicles (Paris, 1994) 137- 142

[30] Rivals I., “Modelisation et commande de processus par reseaux de neurones: application au pilotage d’un vehicule autonome”, These de Doctorat de l’Universite Paris 6 (1995)

[31] Rivals I., Personnaz L., Dreyfus G., Ploix J., “Modelisation, calssification et commande par reseaux de neurones: principes fondamentauz, methodologie de conception, et illustrations industrielles”, Recents progres en genie des procedes 9, Lavoisier technique et documentation, Paris (1995)

VITA

Author's Name – Earl L. Fugate

Birthplace – Dayton, Ohio

Birthdate – April 23, 1974

Education

Bachelor of Science in Electrical Engineering
University of Kentucky
December - 1997