Artificial Neural Network Modeling of Czochralski Single Crystal Growth

Roy Smeal
University Undergraduate Fellow, 1995-1996
Texas A&M University
Department of Electrical Engineering

APPROVED

Fellows Advisor Rx. Ramsey

Honors Director Alexandra Fennell

ARTIFICIAL NEURAL NETWORK MODELING OF CZOCHRALSKI SINGLE CRYSTAL GROWTH. *Roy M. Smeal (Dr. R. K. Pandey), Electrical Engineering, Texas A&M University.*

The adaptive nature of Artificial Neural Networks (ANNs) makes them a powerful tool for modeling complex processes. ANNs are inherently nonlinear and process signals in a parallel manner. These qualities make them suitable for applications requiring the model to identify the dynamics of a nonlinear system and applications in which the speed of the model's response is important. Czochralski single crystal growth is a widely used technique for growing semiconductor crystals to be used for substrate material in integrated circuits. The trend in the microelectronics industry is to fabricate smaller components and more intricate circuits while increasing the wafer size on which these circuits are made. This trend requires that larger crystals be grown. In addition, the purity and defect requirements are becoming more stringent. Czochralski crystal growth is a nonlinear process and difficult to control. The trends in industry make the growth even more sensitive to the control. An ANN is developed in this research that could be used for a model based control of the Czochralski crystal growth process. The ANN model is compared with an existing model developed for germanium single crystal growth.

ACKNOWLEDGEMENT

The author wishes to thank Jayakumar Muthusami for his instruction about the inner workings of neural networks, modeling, and crystal growth. The advice that Jayakumar has given throughout this research has been invaluable. The meals his wife was kind enough to cook for me are much appreciated also.

The author would like to thank Dr. Pandey for his advice and for providing the opportunity to participate in this research. It has been a rewarding experience.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER 1

INTRODUCTION

The objective of this research is to develop an Artificial Neural Network (ANN) model for Czochralski crystal growth. Czochralski crystal growth is the process currently used to grow silicon, germanium, and ilmenite single crystals as well as other types of semiconductor crystals. These crystals are sliced into wafers which are then used as substrates for integrated circuits. The proper growth of these crystals is essential to the eventual successful fabrication of integrated circuits. Ilmenite is a new semiconductor material with many potential applications. The growth of ilmenite poses an interesting problem because there is not much data collected on how to effectively grow ilmenite using the Czochralski growth process. The development of a good model for the growth process is important to understand and control the dynamics of the growth especially for a new crystal technology. Czochralski crystal growth technology has advanced in silicon due to various modeling efforts.

The semiconductor, ilmenite, is a new material whose properties are currently being determined. It has many potential applications in the semiconductor industry including high temperature circuits, blue-green lasers, and radiation resistant solar cells[1,2]. The potential application as a blue-green laser arises from ilmenite's large band gap of 2.54 eV [3]. This would have a significant impact for optical storage technology. This same band gap may make ilmenite a suitable substrate for high temperature circuits. The identification of a model to the growth of ilmenite is key to the

understanding of the important growth parameters and for the large scale industrial growth of the semiconductor.

The development of new ANNs is an active field of research. The implementation of nonlinear squashing functions in ANNs has allowed for a nonlinear function approximation capability that was previously not available[4]. This new ability has spurred the application of ANNs to a much wider range of problems including the modeling of processes ranging from etching, thin film growth and crystal growth[5]. Excitement in this area also stems from ANNs adaptive nature and the capacity to generalize beyond the examples used to train the network. There are many different types and variations on established ANN architectures. The Adaptive Back Propagation (ABP) network [6,7] was chosen for this application. It is a variation of standard Back Propagation (BP) which is a well established learning technique[8]. Originally BP and ABP were developed for static applications which would not be suitable to model a dynamical process. These networks were modified to deal with data that changes in time. Since crystal growth is a dynamical process, the data is collected in the form of a time series. It is important that the model recognize trends and potential deviations from desired growth. ANN modeling is an experimental modeling approach. ANNs learn from actual data from the system to be modeled. Learning is accomplished by comparing the output of the network with known experimental data. This type of learning is known as supervised learning. ANNs and the specific network used in this research will be discussed in detail in chapter III.

The advances in integrated circuit technology and the demand for faster, denser, and larger circuits has created the need for new materials and a more refined

crystal growth process. The variations in crystal diameter and imperfections have to be controlled more rigorously than ever before. In addition to these improvements, the trend in industry is towards larger wafer diameters and therefore larger crystal diameters. The larger diameter requirement makes the dynamics and control even more difficult. A precise control relies heavily on a model that accurately predicts how the system would respond based on current conditions. The control, usually a Proportional Integral Differential (PID) controller, will be adjusted appropriately knowing these conditions. The model's importance in the effective controlling of a complex process has spawned much research in this area. The importance of modeling and some of the models developed in previous research will be discussed next.

I.1. Why is Modeling Needed?

As the demands on the crystal growth industry by the integrated circuit fabrication community grow more stringent, new questions are to be answered. How are the best process conditions determined? The need for a more sophisticated closed loop control becomes apparent. Previously, the control parameters were adjusted manually by a technician during the process as the technician saw fit. As the exactness required in the growth increased, the human error that this practice introduced became unacceptable. The question of how to implement a better closed-loop control becomes important. The answer to both the questions of process conditions and closed-loop control could be found in the appropriate process model. If a sophisticated enough model could be developed, the relationships between the process conditions and the important crystal

variables like diameter, defect distribution, and solute distribution could be found. With this information, the optimal process conditions could be found.

I.2. Established Modeling Techniques

After the need for models was recognized, the task of developing a model for Czochralski growth needed to be addressed. The first models were simple. Kim et al.[9] came up with a simple algebraic model that modeled all phases of the growth process. This model was implemented on a computer and the crystal growth was simulated. The simulation results were compared with actual growth and there was good agreement between them. This model was developed from purely empirical observations. This model could not capture the complex dynamics involved in the process. A more sophisticated crystal growth system would require much more accuracy.

A model similar to Kim's was developed by Srivastava et al[10]. This model used semiquantitative arguments and was more complicated. The model provided empirical relationships between pulling rate, interface shape, crystal radius, temperature, and melt level. More accurate prediction is still required. Models needed to be developed taking into account all the relevant physics of crystal growth. These first principle models had much to deal with. Heat flow, fluid dynamics, mass transfer, and a myriad of process parameters were just a few things that needed to be included in a comprehensive model. To help deal with these complexities, usually only one growth phase was considered. The phase chosen is always the constant diameter growth. This is the part of the crystal which will actually be sliced into wafers and is usually considered the most important. Although, it is now recognized that the other phases, neck in and tailing off, are also very

important in determining how defects in the crystal nucleate. These phases will be discussed in chapter II. Even considering only one phase, the models based on first principles are very complex.

A sophisticated model based on first principles was developed by Gevelber and Stephanopoulos[11]. This model uses nonlinear differential equations and requires many process parameters which would vary depending on the crystal growth setup. The complication of the model makes it difficult for a computer to solve and is difficult to implement online. In addition to this drawback, the model is difficult to adapt to different crystal growth furnaces and melt types. With the realization that complex first principle models are difficult to implement, concentration focused on identifying models using experimental data. Ghassempoory et al. [12] modeled the growth process by identifying a transfer function that fits the experimental data. This model predicted the change in grown crystal weight which is a measure of the diameter reasonably well, but there was some error introduced by the linearity assumptions of the identifying technique. If an experimental model could be developed that is good at identifying data from a nonlinear system, it might be able to satisfy both the requirements of speed and accuracy. ANN modeling could be used to achieve both these requirements.

I.3 ANN Modeling of Crystal Growth

The nonlinear nature of Czochralski crystal growth and the difficulty of using standard modeling techniques suggests ANN modeling might be appropriate. ANNs exhibit characteristics similar to associative memory. Neural networks learn from exposure to previous information from the system. Data can be collected from the sensor

readings of process parameters, like temperature and crystal weight, and used to train the ANN. The ANN can be trained to recognize potential defects and undesired changes by using data from both acceptable and defective crystals. The architecture of ANNs is parallel by nature and the information flows through them very quickly. Since ANNs are nonlinear, the limitations of linear models are not an issue. The distributed representation of ANNs allows them to generalize well when problems are encountered beyond their training experience[8].

I.4 Research Objectives

The goal of this research is to develop an ANN model for Czochralski single crystal growth. The ANN model should have relatively short training time and have a good generalization. The model developed will be compared to an existing model. The existing model is the linear transfer function model of Ghassempoory et al.[13] that uses the temperature and weight derivative as input and output parameters respectively. After this comparison, the modeling approach could be applied to a ilmenite single crystal growth problem in the Center for Electronic Materials, Devices and Systems(CEMDAS) lab at Texas A&M University.

CHAPTER II

CZOCHRALSKI CYSTAL GROWTH

II.1 Introduction

The Czochralski single crystal growth technique is one of the most widely used techniques in industry. The process represents half of all high quality crystals grown. Czochralski growth was first developed by Czochralski in 1918. While this technique is more demanding as far as equipment and skill are concerned, it is one of the fastest single crystal growth processes and produces quality crystals. For most modern devices high purity as well as quantity is essential. It is because of these considerations that industry often chooses this technique despite its complications. The largest application of Czochralski growth is for silicon, but at least 100 other materials using the crystal pulling process are commercially available. New materials, like ilmenite, with potential applications as substrate material need to become established in this growth technique. Once it can be grown reliably using this process, it becomes feasible to market it commercially. Crystal growth is often considered an art due to the subtle manipulation required of the pulling rate, rotation rate, thermal geometry, and atmosphere which all must be simultaneously controlled to minimize imperfections. Of course, this is not a positive state of affairs as far as industry is concerned. This is why large amounts of money and time has been spent on research by the industry to develop better models and more efficient controls for crystal growth.

The basic mechanism for crystal growth is the relative motion of the seed and the melt. Since the growth is accomplished through a solid-liquid interface the materials selected for this type of growth must have certain properties to avoid problems. The composition of the material in solid state must be the same as in the liquid state. It should not decompose before or during the melting process and it should have a low vapor pressure. Incongruent melting or vaporization (i.e. different components of the material melt or vaporize at different temperatures) will lead to an imbalance of certain components. If the material does have one or more of these characteristics, it could lead to a stoichiometry different from what is desired. There should be no first order solid-solid phase transitions or reconstructive phase transitions as this could lead to a polycrystalline structure instead of the desired single crystal structure[14]. The material should not react with the crucible or any other object it contacts during the growth as this would contaminate the crystal. Despite all these restrictions, many materials can be grown using this technique. Ilmenite has the properties necessary to utilize the Czochralski growth process. Research still needs to be done to determine the optimal conditions for Ilmenite growth.

II.2 Czochralski Growth

The basic growth process consists of the following steps[14,15]. First the temperature of the melt is raised a few degrees above the melting point of the material. During this period radial and vertical temperature gradients need to be kept as small as possible, usually plus or minus 20% of its average temperature[14]. This mean temperature will differ depending on the conductivity of the material being grown. For

materials with high conductivity, a temperature 25 °C above the melting point is normal. Materials with low conductivity are kept at a temperature closer to the melting point. In the second step, the seed crystal, which is in its purest form and has a well defined crystallographic direction of interest, is slowly brought into contact with the melt. The seed crystal rotates during the whole process. After it contacts the melt, the downward movement is stopped. If the temperature of the melt is set correctly, then the seed crystal should start to melt slowly without losing contact with the melt. Crystal growth will not occur if the temperature is less than the nucleation temperature. The crystal will lose contact with the melt if the melt temperature is too high. To minimize dislocations, the crystal should have a narrow region before the main growth starts. To achieve this a neck is formed. Generally when the temperature or the pull rate is increased, the crystal diameter will decrease and vice versa. To start the neck-in, the pull rate or temperature is increased. The diameter will decrease due to this increase. After the neck-in step is attained successfully, the temperature and pull rate is lowered and consequently the diameter starts to increase. This stage is referred to as shoulder growth. The temperature is then leveled off and the crystal soon after grows to its constant diameter. The point at which the temperature is leveled off is important and this point is determined by experience with the particular crystal being grown. The constant diameter crystal is grown until the final length is reached. The constant diameter growth is complex to accomplish because the temperature and pull rate usually need to be adjusted to maintain the desired diameter. These rates also affect many other growth conditions like fluid dynamics, heat flow, and dopant impurity levels. The relationships between these factors

are nonlinear making correct adjustments difficult. This is why modeling and automating the process is desirable for the crystal growth industry. In the final stage, the growth is ended, also known as tail off. This can be done in different ways. The pull rate can be increased so the crystal breaks contact with the melt or the temperature can be raised which slowly decreases the diameter. The latter is usually preferred since it causes fewer dislocations in the crystal.

II.3 Crystal Growth Equipment

The main equipment considerations for the crystal growth are the furnace, the heat source, the crucible, and the computers used to implement the control. Furnaces can come in a wide range of complexity depending on the chemical and thermal requirements of the system. The furnace used in Ilmenite research is made off water cooled steel. It can be sealed so that the atmosphere inside the furnace can be controlled.

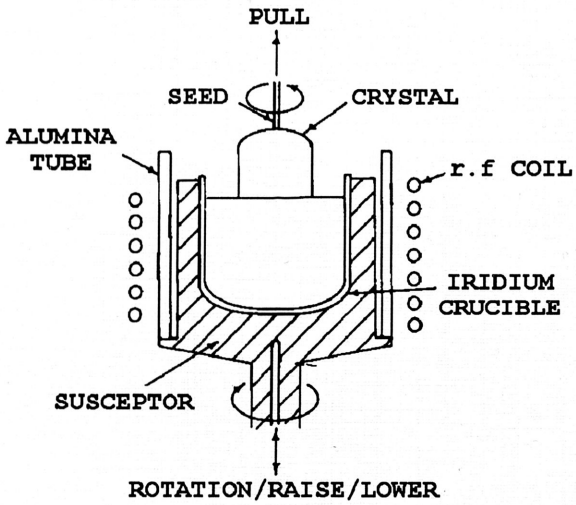

Figure II.1 Radio Frequency Czochralski Grower

A typical Czochralski setup is shown in the diagram above. The apparatus shown above has an alumina tube and an iridium crucible for ilmenite growth. The crucible material can vary on the type of crystal to be grown. The selection depends on the melting temperature of the material, the material's chemical activity, and the type of heating among others.

There are two types of heat sources. The choice of a heat source is a crucial decision in the design of the furnace. The simpler of the two is the resistive heater. A current is passed through a resistive element which heats up due to the power losses in the element. This type of source has very uniform heating and makes the placement of the crucible in the furnace less crucial. Resistive heating has a higher electrical efficiency and is easier to control. This type of heating is only good for the lower temperature ranges. Because of this limitation, resistive heating is primarily used for semiconductor materials with low melting points; usually up to 1500°C. The second type of heating is inductive heating. The heating is done by coils carrying a current with a frequency range of about 250-500 kHz. These coils are magnetically coupled with a susceptor that surround the crucible or the crucible itself. This coupling creates eddy currents in the susceptor or crucible that cause heating. This type of heat source can operate at higher temperatures, but it has several drawbacks. This process consumes more power. The susceptor or crucible have to be conductive so the eddy currents can be created. There are higher thermal gradients in the furnace and the crucible, so the placement of the crucible is much more critical. The crucible must be of high quality as defects can cause hot spots which can melt it. These hot spots result from resistivity changes near the defect.

Crucible selection depends on many factors and can be difficult. Crucibles are usually cylindrical but may be tapered. The ratio of the height to diameter is between 0.5 and 1.5. In most cases it is between 0.8 and 1.2. If the ratio is much larger than this, small irregularities in the crucible diameter produce large changes in the growth rate. The crucible can not react chemically with the melt and must have a higher melting point. The type of heating must be taken into consideration for reasons mentioned above. The following table gives a listing of some commonly used crucibles.

Material	Maximum Operating Temperature (°C)	Melting Point (°C)	Use
Platinum	1400	1773	1,2
Iridium	2150	2452	1,2
Molybdenum	2300	2620	1,2
Tungsten	2800	3370	1,2
Carbon	3000		1,2
Silica	1550	1700	1
Alumina	1800	2050	1

1 = resistance; 2 = r.f. heating.

Figure II.2 Crucible Materials, Melting Point, and Use

II.4 Control Difficulties

The effective control of the growth rate and the factors which affect it is essential for successful crystal growth. Ideally the pull rate would depend only on the growth rate and the melt temperature. Unfortunately, it is very sensitive to other factors including fluid dynamics, thermal gradients, liquid/solid density ratios, and the geometry of the system.

The basic fluid flow problem for Czochralski crystal growth is a rotating fluid next to a surface rotating at a different velocity. This fluid flow is sometimes modeled using a shear layer model. This model is based on the Taylor-Proudman theorem which

states that all steady motions in a rotating fluid are two dimensional with respect to the rotating axis[14]. This model was developed for ideal fluids and is only approximately true for the fluid flow in Czochralski crystal growth. It only takes into account forced convection that is caused by the mechanical driving forces of the system like rotation. In addition to the ideal fluid assumption, the model also assumes a steady state. This is not always true. Since the dynamics is nonlinear, small deviations from the steady state can create large variances in the fluid dynamics. These deviations can arise from thermal gradients, vibrations, or changes in melt level. The thermal gradient can sometimes add significantly to the fluid dynamics and is referred to as natural convection. When both types of convection occur, a more complicated fluid flow results. Some examples of this fluid flow are shown below.

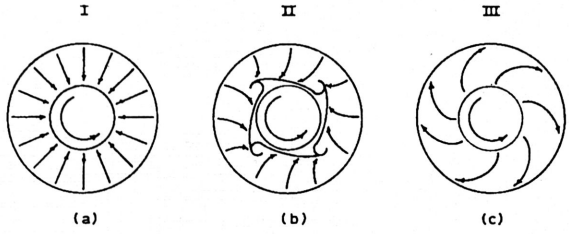

Figure II.4 Fluid Flows

Figure II.4(a) shows the flow for normal dynamics with slow rotation. Figure II.4(b) and (c) picture normal flow for moderate and fast rotation respectively.

The nonlinear relations between fluid flow, thermal gradients, and the growth rate make the minimization of the thermal gradients very important. The rotation of the crystal and the crucible helps to minimize the gradients. This comes with the price of adding to the fluid dynamics. One of the causes of thermal gradients is the deviations between the thermal axis of rotation and the mechanical axis of rotation. By rotating the

crystal and the crucible these axes can be brought together. For semiconductor materials, both types of rotation are used. Fluid flow and temperature variations expose the crystal surface to different temperature. These variations cause growth striations. If the crystal is doped, these striations can cause the development of nonstoichiometric areas in the crystal. If the temperature variation is severe enough, actual remelting can occur. The complex interaction of all these factors is what makes the accurate control of the Czochralski growth process a formidable problem.

II.5 Measurements for Control

Crystal diameter is inversely proportional to the power input in a simple model. The first control systems were designed to take advantage of this relation. The power input would be altered according to an empirically developed program. Most recent crystal growth systems have the temperature control. Temperature measurements are usually taken with a thermocouple or pyrometer. The temperature gradients make it difficult to decide where to take the temperature measurement. Recent control systems use diameter measurements in addition to the temperature measurements. There are several methods of measuring crystal diameter. One method uses a video camera. The camera is usually focused on the meniscus which gives information about the crystal melt interface. The image is processed to get the diameter measurement. This is difficult because the camera angle is usually oblique so some sophisticated processing has to be done. Getting a good contrast between the melt and the meniscus is another problem. Choosing an appropriate wavelength can help create a good contrast. Another measuring technique uses the differing emissivities of convex and concave surfaces. The solid-

liquid-vapor intersection has a curved melt surface for some materials that has a differing emissivity than its surroundings. This difference can be measured with a photo-cell array to determine diameter. Optical reflectivities can also be used to measure diameter. A light is focused on the curved surface and reflected to a photo-cell. These last two methods are good because they require less processing. Their performance varies for different sizes of crystals. The emissivity technique is better for large diameters and the optical technique is better for small diameters. They are both most effective for the constant diameter growth and performance breaks down for the other stages of the growth process. The diameter measurement can be taken indirectly by weighing the crystal and the change in the weight. This technique is commonly used because it is generally easier to implement.

II.6 Practical Modeling of Czochralski Growth

The main idea of modeling is to identify the key relationships between the variables in the process and the parameters of the crystal to be controlled. The model allows the crystal growth personnel to select the specifications of the crystal more selectively. Some of the factors which are important in determining key features such as diameter and defects in the crystal cannot be effectively measured. Measuring fluid dynamics or temperature gradients is not practical in a system that is extremely hot with a sealed atmosphere. The measurement of temperature and diameter can be a significant problem. For a model to be practical, it must be able to predict the behavior using the variables of the system that can be measured. All the dynamics of the system must be inferred from these variables.

The variables that are usually measured in Czochralski single crystal growth include temperature, diameter, and power. In many cases, diameter is indirectly measured by the change in the weight variable. Some system inputs are set externally and do not necessarily need to be measured. Pull rate and crucible rotation are provided by the controller, so these values are known and can be used in the modeling. If these variables were to be measured, it would be to determine any small discrepancies between the set rate and the actual rate. If the system is designed properly these differences should be very small or nonexistent and can be neglected in the model. Models based on first principle physics must ultimately yield a set of equations that can be solved with the variables that can be measured or at least inferred. Empirical models must identify the rich nonlinear dynamics of the system using only the few input and outputs of the system that are available. It is a difficult task for both sophisticated first principle models as well as empirical models. The next chapter discusses some of the theory of neural networks which can be a potent tool for dealing with the problems discussed above.

CHAPTER III

NEURAL NETWORKS: BACKPROPAGATION AND ADAPTIVE BACKPROPAGATION

III.1 Introduction

Before the workings of neural network models are discussed, it would be useful to explain how a model fits into the control of a process like crystal growth. The need for a model for a closed-loop control is explained in the following diagram.

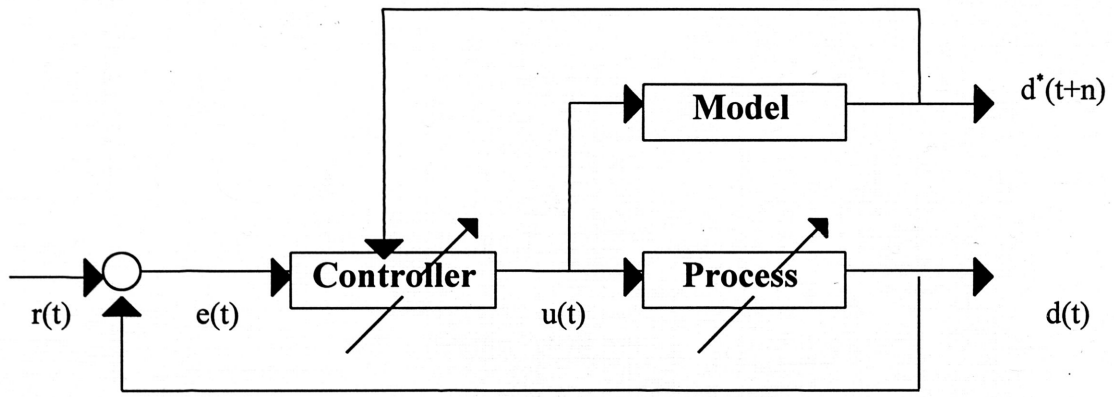

Figure III.1. Closed Loop Control with Model

A command signal r(t) that represents the process requirement is provided as an input to the system. The command signal is compared to the actual output of the process, d(t), and then the error e(t), which is the difference between d(t) and r(t), is sent to the controller. The controller sends the appropriate input signal, u(t), to adjust the process to minimize the error. The signal u(t) from the controller is also sent to the model of the process. The model provides a prediction to a certain number of time steps, n, ahead. This result is then

sent back to the controller which makes a comparison with the desired output. If there is a discrepancy, an adjustment is made to avoid the potential undesired output. This model needs to be implemented on a computer in a way that gives the predictions quickly, and the controller can make the appropriate adjustments in time. These adjustments take the place of the manual adjustment that would normally be done by the technician. In this way, the model allows for a more precise control of the crystal growth process. Good results are more easily repeated and the need for technicians to constantly monitor the process is eliminated.

III.2 What is Backpropagation?

BP[4,8] and ABP[6,7] belong to a class of ANNs called multilayer feedforward networks. These networks are made up of layers. Generally, there is an input layer, one or more hidden layers and an output layer. The input layer receives the input data and propagates it to the hidden layers which usually do the bulk of, if not all, the computation. The output layer provides the results of the network. Signals flow through the network in a parallel manner layer-by-layer. This type of neural network is also called multilayer perceptron(MLP) network. Each layer consists of a number of nodes or neurons which are the actual processing elements. The neurons in one layer are linked to the neurons in the next layer through connection elements called weights. The architecture of the multilayer perceptron ANN is shown in Figure III.2. Each of the connections shown in Figure III.2 has a weight associated with it. As the signal passes from one node to the next, it is multiplied by the corresponding weight. The training of the network consists of

two passes of information through the network: a forward propagation and a backward propagation of information.

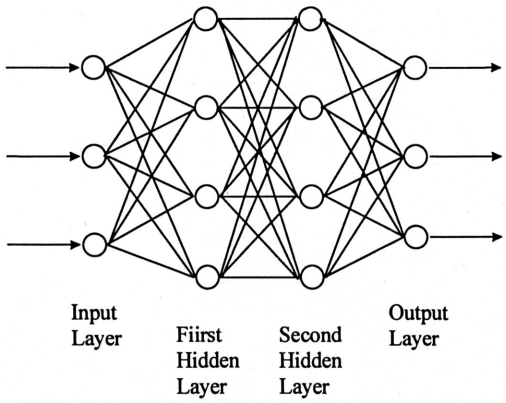

Figure III.2 Multilayer Perceptron Architecture with Two Hidden Layers.

For the forward propagation, a set of inputs is applied to the network. The effect of this input is propagated through the network and an output is generated. During the forward propagation the weights remain unchanged. The difference between the network output and the desired output gives an error. This is when the backward propagation occurs. The error information is sent back through the network and the weights are adjusted to minimize the error. The rule by which the error correction occurs is developed in the next section.

An important aspect of the MLP network is the nonlinearity that is associated with each node. Each neuron sums all of the signals it receives from the previous nodes. This sum is then acted upon by a nonlinear function usually called a squashing function. The nonlinearity added to the network is what enables this architecture to model a wide range of nonlinear problems. A network that does not have these functions can be collapsed into a single layer network which has a very limited function approximating capability[4].

III.3 The Backpropagation Rule

This section describes the backpropagation rule. The weights are characterized by two pairs of numbers, one pair representing the source node and the other representing the destination node. Each neuron is represented by two numbers: a layer number and a node number. The networks used in this research are fully connected, meaning that each node in a layer is connected to every node in the next layer. At each neuron there is a summation of all the input signals multiplied by their corresponding weights. A bias term corresponding to where the neuron is active is also added to this sum. This sum is passed through the squashing function which scales it. The resulting value is passed on to the next neuron via the weight connection or to the output. The action of the neuron is represented in Figure III.3.

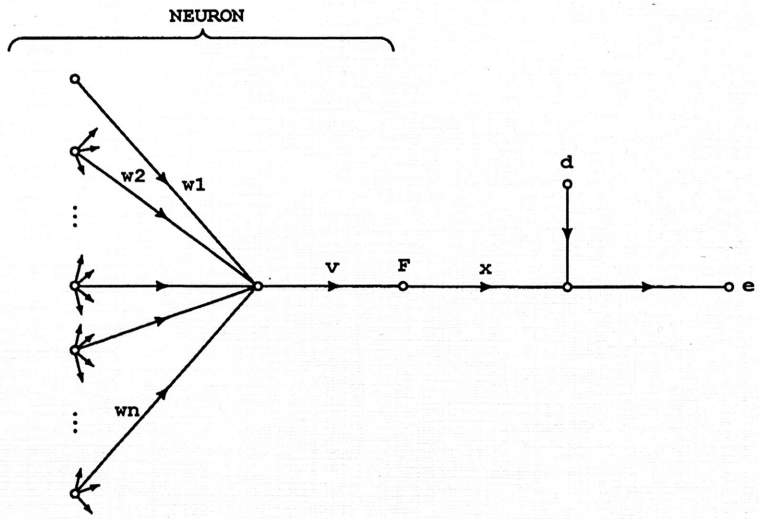

Figure III.3 The Structure of a Neuron

The equations that represent the neurons will now be discussed. The notation quickly becomes cumbersome because of the nature of the architecture. The number of layers will be represented by L. The number of neurons in each layer will be $i = N(l)$ where $l = 1,\ldots,L$. $N(l)$ is a vector with L elements each representing the number of neurons in a layer. The number of training examples will be represented by k. The summation at each neuron is then given by:

$$v_{[l,i]}(k) = \sum_{j=1}^{N(l-1)} w_{[l-1,j][l,i]} x_{[l-1,j]}(k) + b_{[l,i]}(k) \tag{1}$$

where $v_{[l,i]}(k)$ is an internal variable of the neuron which is the value of the signal after the summation and before the squashing function, $w_{[l-1,j][l,i]}$ is the weight connecting the jth neuron in the (l-1)th layer (previous layer) to the ith neuron in the lth layer, $x_{[l-1,j]}(k)$ is the output signal of the jth node in the (l-1)th layer(previous layer), and $b_{[l,i]}$ is the bias term of the ith neuron in the lth layer. Once the summation is calculated, it is acted upon by the squashing function. This operation is expressed by:

$$x_{[l,i]}(k) = F_{[l]}(v_{[l,i]}(k)) \tag{2}$$

where $x_{[l,i]}(k)$ is the output of the ith neuron in the lth layer in response to the kth training vector. The squashing function $F_{[l]}$ is usually linear, sigmoidal, or hyperbolic tangent. The input layer passes the training examples to the hidden neurons and usually does not have a squashing function associated with it. The hidden layers in this research use the hyperbolic tangent squashing function and the output layers use the linear squashing function. Whether sigmoidal or hyperbolic functions are used for the hidden layers depends on the application. The sigmoidal function is given by the following:

$$F_{[l]}(v_{[l,i]}(k)) = \frac{1}{1+\exp(-v_{[l,i]}(k))} \tag{3}$$

The output range of the sigmoidal function is 0-1. The hyberbolic tangent function is given by:

$$F_{[l]}(v_{[l,i]}(k)) = \frac{1-\exp(-2v_{[l,i]}(k))}{1+\exp(-2v_{[l,i]}(k))} \tag{4}$$

The output range of the hyperbolic tangent is from 1 to -1. These functions scale the output to 0 to 1 and -1 to 1 respectively. This is where they get the name squashing function. In general, the sigmoidal function is best for binary applications.

The training of the network is accomplished by applying a set of training examples. These examples are given in input-output pairs $(u_1,d_1),(u_2,d_2),\ldots,(u_k,d_k)$. These examples come from the system to be modeled, $\mathbf{d}_k=f(\mathbf{u}_k)$, where f is a mapping of the input vector $\mathbf{u}_k$ to the output vector $\mathbf{d}_k$. This mapping is unknown. The training examples can be expressed as the following set:

$$S = \{u_{[j]}(k), d_{[i]}(k), \forall k = 1,\ldots,NE; j = 1\ldots N(1); i = 1,\ldots,N(L)\} \tag{5}$$

NE is the total number of training examples. For each training example k, $N(1)$ input values and $N(L)$ output values are given corresponding to the input and output variables of the system.

The network response $x_{[L,i]}(k)$, $i = 1,\ldots,N(L)$, is calculated given the input examples $u_{[j]}(k), j = 1,\ldots,N(1)$. This is done by forward propagating the inputs through the network using equations (1) and (2). When this value is available, it is compared with the desired output $d_{[i]}(k), i = 1,\ldots,N(L)$, and the error is determined. The

network learns by adjusting the weight and bias terms so the mean square error for the entire training set is less than an allowable tolerance. The error between the output neurons and the k^{th} training example is:

$$e_i(k) = d_i(k) - x_{[L,i]}(k) \tag{6}$$

The error for a given training example is given by the sum of the squared errors for all of the neurons in the output layer and is the following:

$$E(k) = \frac{1}{2}\sum_{i=1}^{N(L)}(e_i(k))^2 \tag{7}$$

The mean squared error is given by:

$$E_{av} = \frac{1}{k}\sum_{k=1}^{NE}E(k) \tag{8}$$

which is the sum of the errors over all of the training examples and normalized by the total. These averages are a function of all the weights and bias terms. The network learns by adjusting these parameters to minimize E_{av}.

In BP with individual update, the weights are adjusted on an example-by-example basis instead of the entire data set. The arithmetic average of the individual weight changes for each example is an estimate of the change that would result from taking into consideration the whole training vector set. The change in weight is proportional to the error gradient and is given by:

$$\Delta\overline{w} = -\eta\frac{\partial E(k)}{\partial\overline{w}} \tag{9}$$

where $\overline{w}$ is a vector whose elements are all of the weight terms, and η is a learning rate. There is another commonly used update scheme called batch update. In this scheme, all

of the training examples are applied to the network before the weights are updated. The change in weight is expressed by:

$$\Delta \overline{w} = \overline{w}(new) - \overline{w}(old) = -\eta \frac{\partial E_{av}}{\partial \overline{w}}$$

(10)These gradients are calculated using the chain rule. The error gradient must be calculated for the output layer first as the results from this layer are needed for the hidden layer preceding it. These results in turn are used for the layer before that. The gradient for the output layer is also the easiest to calculate as the error between the network response and the expected value is readily available. The error gradient for the output neurons is expressed as follows:

$$\frac{\partial E(k)}{\partial \overline{w}} = \frac{\partial E(k)}{\partial e_i(k)} \frac{\partial e_i(k)}{\partial x_{[L,i]}(k)} \frac{\partial x_{[L,i]}(k)}{\partial v_{[L,i]}(k)} \frac{\partial v_{[L,i]}(k)}{\partial w_{[L-1,j][L,i]}} \tag{11}$$

Differentiating equation (7) with respect to $e_i(k)$ yields:

$$\frac{\partial E(k)}{\partial e_i(k)} = e_i(k) \tag{12}$$

Differentiating equation (6) with respect to $x_{[L,i]}(k)$ gives:

$$\frac{\partial e_i(k)}{\partial x_{[L,i]}(k)} = -1 \tag{13}$$

Differentiating equation (2) with respect to $v_{[L,i]}(k)$ yields:

$$\frac{\partial x_{[L,i]}(k)}{v_{[L,i]}(k)} = F'_{[L]}(v_{[L,i]}(k)) \tag{14}$$

Finally, differentiating equation (1) with respect to $w_{[L-1,j][L,i]}$ yields:

$$\frac{\partial v_{[L,i]}(k)}{\partial w_{[L-1,j][L,i]}} = x_{[L-1,j]}(k) \tag{15}$$

The derivative $F'_{[L]}(v_{[L,i]}(k))$ will differ depending on the squashing function chosen. For the sigmoidal function the derivative is given by:

$$F'_{[L]}(v_{[L,i]}(k)) = x_{[L,i]}(k)(1 - x_{[L,i]}(k)) \tag{16}$$

The derivative for the hyperbolic tangent squashing function is expressed as:

$$F'_{[L]}(v_{[L,i]}(k)) = (1 - x^2{}_{[L,i]}(k)) \tag{17}$$

The derivative of a linear function would just be a constant. Substituting equations (6), (12), (13), (14), and (15) into equation (11) yields the error gradient for each of the weights in the output layer and is given by:

$$\frac{\partial E(k)}{\partial w_{[L-1,j][L,i]}} = -(d_i(k) - x_{[L,i]}(k))F'_{[L]}(v_{[L,i]}(k))x_{[L-1,i]}(k) \tag{18}$$

where $F'_{[L]}$ depends on the squashing function used. The partial derivative $\frac{\partial E(k)}{\partial v_{[l,i]}}$ in equation (11) is usually referred to as the local gradient δ_i and is defined by:

$$\delta_i(k) = -\frac{\partial E(k)}{\partial e_i(k)} \frac{\partial e_i(k)}{\partial x_{[L,i]}(k)} \frac{\partial x_{[L,i]}(k)}{\partial v_{[L,i]}(k)} \tag{19}$$

For the output layer this can be expressed as:

$$\partial_i(k) = (d_i(k) - x_{[L,i]}(k))F'_{[L]}(v_{[l,i]}(k)) \tag{20}$$

Since the error between the network response and the desired response is not available for the hidden layers, calculating the error gradient is a little complex. The error for a hidden neuron is determined in terms of the errors of all the neurons in the layer to

the right of it and to which it is connected. Again the chain rule of differentiation is used as follows:

$$\frac{\partial E(k)}{\partial w_{[l,m][L-1,j]}} = \sum_{i=1}^{N(L)} \frac{\partial E(k)}{\partial v_{[L,i]}(k)} \frac{\partial v_{[L,i]}(k)}{\partial x_{[L-1,j]}(k)} \frac{\partial x_{[L-1,j]}(k)}{\partial v_{[l-1,j]}(k)} \frac{\partial v_{[L-1,j]}(k)}{\partial w_{[l,m][L-1,j]}} \tag{21}$$

The situation is pictured in figure III.4 below.

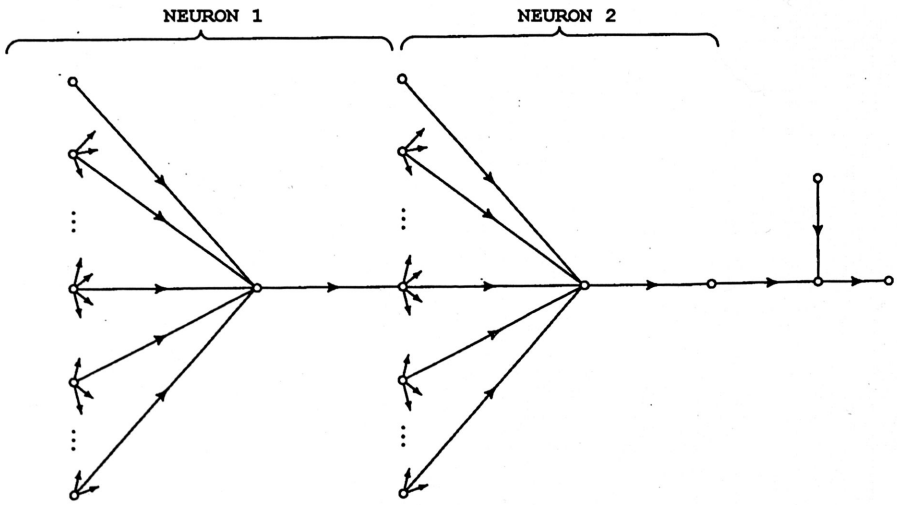

Figure III.4. The Output and One Hidden Layer of a MLP Network

Notice the summation is over the terms $\frac{\partial E(k)}{\partial v_{[L-1,j]}(k)}$, which is the local gradient $\delta_i(k)$ for the output layer, and $\frac{\partial v_{[L,i]}(k)}{\partial x_{[L-1,j]}(k)}$, which is the weight connection between the j^{th} neuron in the $(L-1)^{th}$ layer and the i^{th} neuron in the output layer L, $w_{[L-1,j][L,i]}$. Examining Figure II.3 and calculating the other derivatives in equation (21) yields:

$$\frac{\partial x_{[L-1,j]}(k)}{\partial v_{[L-1,j]}(k)} = F'_{[L-1]}(v_{[L-1,j]}(k)) \tag{22}$$

$$\frac{\partial v_{[L-1,j]}(k)}{\partial w_{[l,m][L-1,j]}} = x_{[l,m]}(k) \tag{23}$$

where $x_{[l,m]}(k)$ is the output of node m in the l^{th} layer and $F'_{[L-1]}(v_{[L-1,j]}(k))$ is the squashing function associated with the $(L-1)^{th}$ layer. Substituting equations (22) and (23) along with the relationships discussed above into equation (21) gives:

$$\frac{\partial e(k)}{\partial w_{[l,m][L-1,j]}} = F'_{[L-1]}(v_{[L-1,j]}(k))x_{[l,m]}(k)\sum_{i=1}^{N(L)}\delta_i(k)w_{[L-1,j][L,i]} \tag{24}$$

Now that the error gradient for the weights in the first hidden layer behind the output layer has been derived the corresponding local gradient $\delta_j(k)$ needs to be found so the process can continue to the rest of the hidden layers if there are any. The local gradient for the hidden layer is defined as:

$$\delta_j(k) = -\frac{\partial E(k)}{\partial x_{[L-1,j]}(k)}\frac{\partial x_{[L-1,j]}(k)}{\partial v_{[L-1,j]}(k)} \tag{25}$$

which is equivalent to:

$$\delta_j(k) = F'_{[L-1]}(v_{[L-1,j]}(k))\sum_{i=1}^{N(L)}\delta_i(k)w_{[L-1,j][L,i]} \tag{26}$$

These new local gradients can in turn be used to calculate the error gradient of the next layer to the left using the above equations with the indices changed and the corresponding local gradient delta. In this fashion the errors backpropagate.

In general, the change in a weight is calculated as follows:

$$\Delta w_{ij} = \eta \times \delta_i(k) \times x_j(k) \tag{27}$$

where η is the learning rate, $\delta_i(k)$ is the local gradient at neuron i, and $x_j(k)$ is the input signal of neuron i. This value is used to update the weights which, of course, is done with the goal of minimizing the error E_{av}. The value of $\delta_i(k)$ will depend on whether neuron i is in an output layer or a hidden layer. If it is in the output layer, then it is simply $F'_{[L]}(v_{[L,i]}(k))$ multiplied by $e_i(k)$. If it is in a hidden layer then delta is given by the product of the derivative of the squashing function and the weighted sum of the local gradients of the neurons in the next layer.

III.4 Drawbacks of ANNs

There are some drawbacks to using ANNs for modeling. For larger networks and more complicated systems, it can take a prohibitive amount of time to train the network. It is also possible to overtrain the network. Overtraining results in a very good fit to the data that is actually used for training, but the ANN loses its ability to effectively generalize to data outside of its training set. ANNs that are overtrained will not be suitable for modeling applications. By judiciously choosing the size and type of network, the training time can be brought down to an acceptable level. The problem of overtraining can be avoided by making sure the ANN performs well for the training data set as well as a test data set. The test data set is an example of system input and outputs that the ANN has not been not trained with. If the ANN still performs well for this set, the network has not lost its generalization capability by overtraining.

A more subtle problem is the local minima of the error function. Sometimes a network will reach a local minimum in its attempt to find the best fit and get stuck there. One way to avoid this problem is to train the network using different initial weight

values. This helps the algorithm to find different minima from which the probable global minimum is selected.

III.5 Adaptive Backpropagation

ABP deals with two of the problems encountered with standard backpropagation. The training time is shorter for many applications and it does not get stuck in a local minimum as easily[7]. The idea of adaptive backpropagation is to adjust the learning rate in relation to the magnitude of the error gradient. If the error gradient is large(i.e. the network is quickly converging to some value), then the learning rate is made smaller to compensate. This reduces the chance that the network will overshoot the desired minimum. If the error is small (i.e. the network is settling down or has hit a flat spot in its descent to a minimum), the learning rate is made larger. This has the effect of speeding up the learning time. A consequence of the varying learning rate is the jumpy behavior of the error as it gets smaller. This behavior has the benefit of knocking the network out of local minimum traps. ABP was chosen for this application because of these qualities.

An idea of how the two algorithms differ can be obtained from examining some relations between the change in the error and the error gradient. These relations were derived using the Taylor series expansions of the error functions[7]. For standard backpropagation the relation follows:

$$\Delta E \approx -\eta \left\| \frac{\partial E}{\partial w} \right\|^2 \tag{28}$$

The change in error is proportional to the error gradient. As the gradient gets smaller, so will the change in error for a fixed learning rate. This is what causes the tendency for BP to slow down or get stuck when the gradient flattens out or reaches a local minimum.

ABP updates the weights in the direction of steepest descent like standard BP. It uses the batch update. This means it updates after the whole training set has been seen. What is different is that the learning rate is a function of the error and of the error gradient norm. The weight update equation is as follows:

$$\overline{w}(new) - \overline{w}(old) = -\rho(E) \left. \frac{\partial E}{\partial \overline{w}} \middle/ \left\| \frac{\partial E}{\partial \overline{w}} \right\| \right. \tag{29}$$

where $\rho(E)$ is analogous to η in BP and is usually chosen to be one of the following:

$$\rho(E) \equiv \begin{cases} \eta \\ \eta E \\ \eta \tanh(E) \\ \eta \tan^{-1}(E) \end{cases} \tag{30}$$

When $\rho(E)$ is chosen to be η, the change in the error is approximately $-\eta$. While this seems to suggest a linear descent to a minimum, it does not actually perform so well. This is probably due to the need for the rate of descent to vary as the error gradient varies. When $\rho(E)$ is chosen to be ηE, the change in error is given by:

$$\Delta E \approx -\eta E \tag{31}$$

This suggests an exponential decay. This makes sense since a large error would create a large change in the weight values and a small error a small weight correction. Simulation studies have shown that the descent is indeed faster[]. The other functions listed above for $\rho(E)$ perform similarly.

III.6 Dynamical Systems and Neural Networks

The above derivations for the neural networks are assuming static problems. Czochralski crystal growth is a dynamical system, so some modifications need to be made to make the networks suitable for a time varying problems. This is basically accomplished by feeding back input and output examples from previous time steps. In this fashion the neural network gets a sense of time variations. This feedback is implemented by increasing the number of inputs to the network. For the static problem, let us consider the network that has two input variables . If the system is time varying, the values of the variables for previous time steps would be added as more inputs. Previous output examples may also be added. If two previous time steps for the inputs and one previous time step for the output are considered, then the number of inputs to the network with two inputs would increase to seven.

$$inputs = \{\ input_1(t), input_2(t), input_1(t-1), input_1(t-2),$$
$$input_2(t-1), input_2(t-2), output(t-1)\ \}$$

In this way the neural network can capture significant trends. The network can then be used to predict time varying signals in the crystal growth such as diameter variation.

III.7 Implementation of the ANN

The neural network was written in C for this research. It is an ABP Neural Network which uses batch update to correct the weights. It was modified to deal with dynamical systems. The code was written so that the number of inputs, outputs, hidden layers, previous inputs, previous outputs, and the number of neurons in each layer could

be modified. The initial weights are created by a random number generator. These weights can be changed by changing the seed of the random number generator or there is an option to read the initial weights from a file. The dependence of the network on its initial state can be examined by varying the initial weights. The network reads the data from a file. The errors for training and testing are written to a file so they can be examined. There is also an option to save the final weights after training to a file. A network that performs well can be saved in this manner.

CHAPTER IV

THE DEVELOPMENT OF THE ARTIFICIAL NEURAL NET MODEL

IV.1 Introduction

The first decision that needs to be made when developing an ANN model is the selection of an architecture to be used. The type selected for the Czochralski model was the MLP architecture discussed in chapter three. A learning rule by which the network parameters(weights) are adjusted, also needs to be selected. Several types of learning rules were examined. The ones examined were the variations of the BP rule discussed in the previous chapter. Of these, ABP was considered to be the best suited for the modeling problem. The ABP network was modified to allow for the dynamical variations of the system as discussed in chapter III. The system input and output variables that will be used in the model need to be selected and an appropriate data collected from experiments. The training of the ANN using a data set and the ANN parameters that effect this learning will be discussed in a later section. After a ANN has been trained, its ability to predict and generalize is tested.

IV.2 BP Rule Comparison

The three BP rules examined were BP with individual update, BP with batch update, and ABP. The main concerns addressed when testing the rules were the

algorithms ability to reach a error minimum and how fast this minimum is reached. While the performance of an ANN is very fast once it is trained, the actual training process can be very slow. For large networks, it is not uncommon for the programs to run for days. It has been demonstrated that ABP has a better chance of finding a global minimum and is more efficient at finding the local minimum[6,7].

The preliminary test data set used to evaluate the algorithms performance[7] was constructed from the function:

$$z = \sin(x)\sin(y) \tag{32}$$

This set is a nonlinear two dimensional problem and is fairly difficult for experimental models to approximate. A plot of the function is given below.

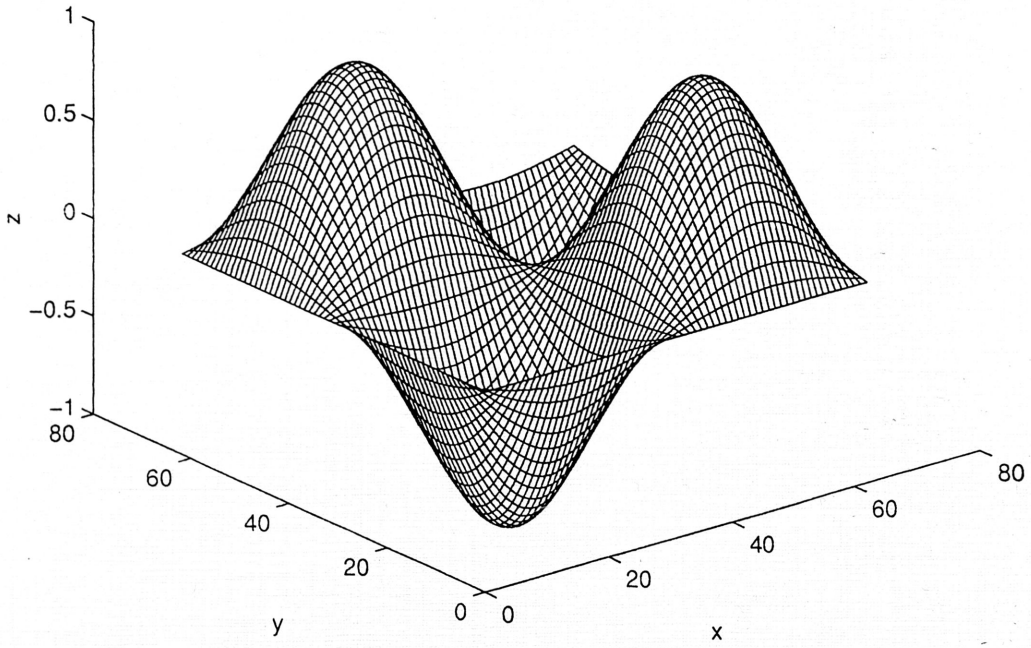

Figure IV.1 Plot of $z = \sin(x)\sin(y)$

The data set used to train the network was created by sampling this function at certain x and y intervals. The network would have two inputs representing x and y, and one output

representing z. Every x and y input has a corresponding output z, and these make up subsets of the data set often called training vectors. For training, the network is exposed to all the training vectors in the data set. When BP with individual update is used, the error for each vector is calculated and the weights updated accordingly to minimize the error. BP with batch update cycles through every vector in the data set and calculates an average error over the whole set. The weights are updated only after the cycle is complete. ABP also uses batch update mode. In addition to this, the learning rate η is varied for each cycle. The minimization of the error is an iterative process for all these rules. An ANN may have to cycle, called an epoch, through all the vectors in the data set many times before an acceptable minimum is reached. This is the cause of the long training times. The training vectors for the function given in equation (32) were cycled through 10000 times for each of the BP rules given above. The graphs of the training errors versus the number of cycles for each BP rule are shown in figure IV.2. The error profile for ABP stands out due to its jumpy nature. These jumps in the error profile are caused by the adaptation of the learning rate. These jumps have the benefit of boosting the error descent out of a local minimum. A drawback of this jumpy behavior is that the same can happen for a global minimum. The ABP also reaches close to the minimum noticeably faster. Overall, the performance of ABP was better.

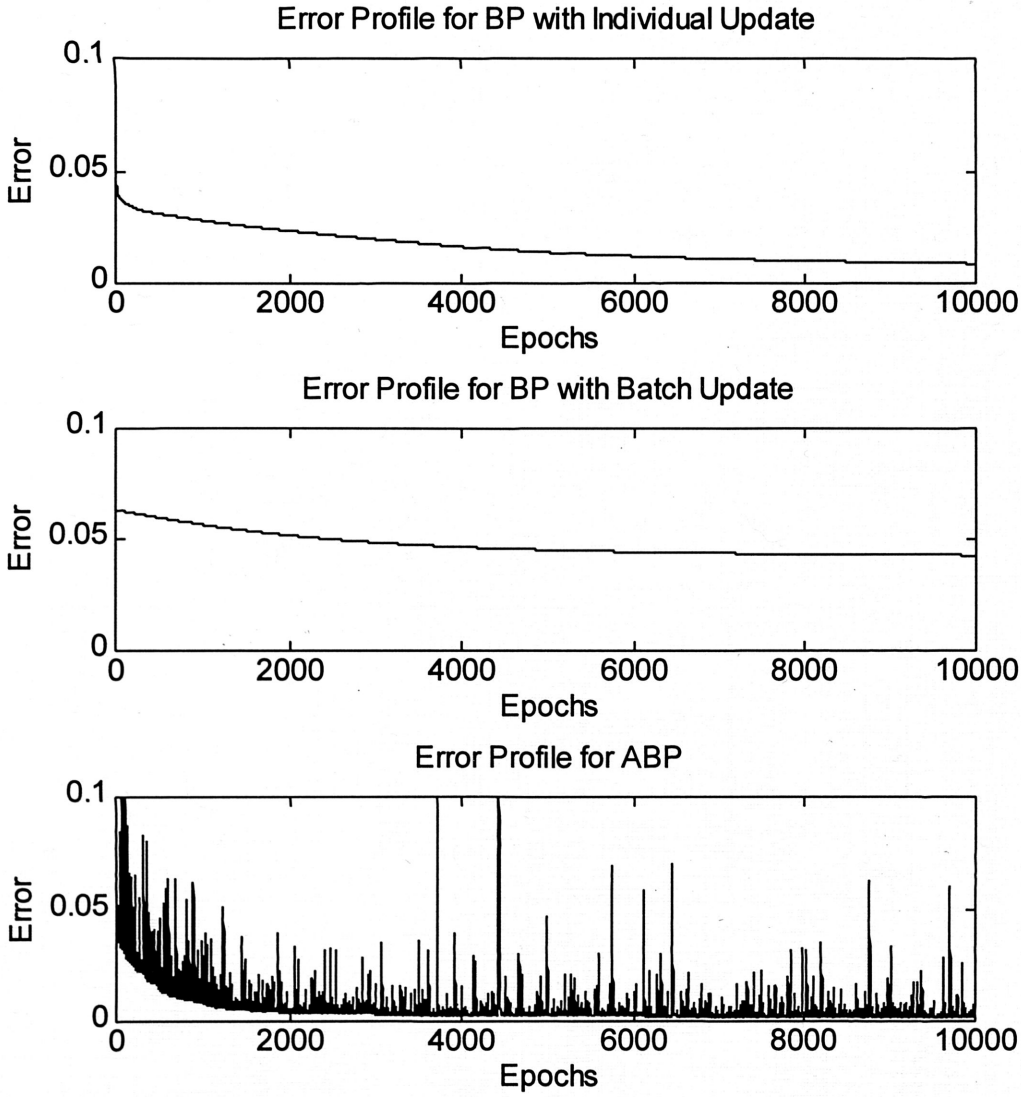

Figure IV.2 Error Profiles

IV.3 Network Variables that Effect Training

Several factors in the ANN can effect the learning capability of the network. The learning rate mentioned in chapter III is one of these factors. The number of layers and

neurons in each layer is another important consideration. The starting weights which are determined by an initial seed for a random number generator can help the network learn by starting it off closer to a minimum. For dynamical systems, the number of previous inputs and previous outputs are two more factors. Once some experience with ANN is developed, the practical ranges of these parameters for certain types of problems becomes apparent and the selection of these values becomes easier. After, a certain point, however, it is a matter of testing different ANN sizes, learning rates, and seeds to see what settings give the best performance.

In general, larger numbers of layers and neurons do not help the overall performance of the network. At first inspection, it may seem a larger network could capture more information but actually the performance usually degrades. Larger ANNs learn the training set very well but loose the capability to generalize. In effect, the ANN has memorized the training data and has learned very little of the underlying dynamics. The testing data set gives the trainer a way to test the ANNs generalization ability. Initial testing is done with smaller network sizes and the size is increased only when necessary. The next section discusses the initial experimental data that was used to test the performance of ANN models when applied to crystal growth problems.

IV.4 Ghassempoory's Transfer Function Model

Ghassempoory et al.[13] modeled the crystal growth of the semiconductor germanium with a transfer function model. He derived a Laplace transform model using linear system identification techniques. This Laplace transform model is linearized about a certain operating point. The model assumes that the system has approximately linear

response for the steady state constant diameter growth. The errors in the model's prediction of the system output most likely arise from the linearity and steady state assumptions. The input and output variables he chose to include in his model were temperature and weight respectively. The model was tested using a germanium crystal growth system in his lab. Two of the input-output sets of data from experimental crystal growths used in the research are pictured on the following page in form of graphs. The graphs show how the melt temperature and weight of the crystal varied with time. These data sets were chosen to train and test the ANN model. The first set is a square wave input and the systems response to this input. These consist of the first 480 time steps in the graphs below. The second set is a pseudo random binary signal PRBS input and the corresponding output, and consists of the last 370 time steps in each plot. Both the input sets are combined in first graph and both the output sets are combined in the second. The data was extracted from Ghassempoory's Ph.D. dissertation[13].

IV.5 Training the ANN

The input data files containing the training vectors for the ANN program were created using the data extracted from Figure IV.3. The square wave input is a much simpler set than the PRBS input set. The best situation would be to train the ANN using the simpler square wave input and to test the network with the random input data set. If the network could predict the response of the system to the PRBS input having only learned from the square wave data set, this would show the ANN was able to infer the

more complicated dynamics of the crystal growth system having only learned from the simpler dynamics excited by the square wave input.

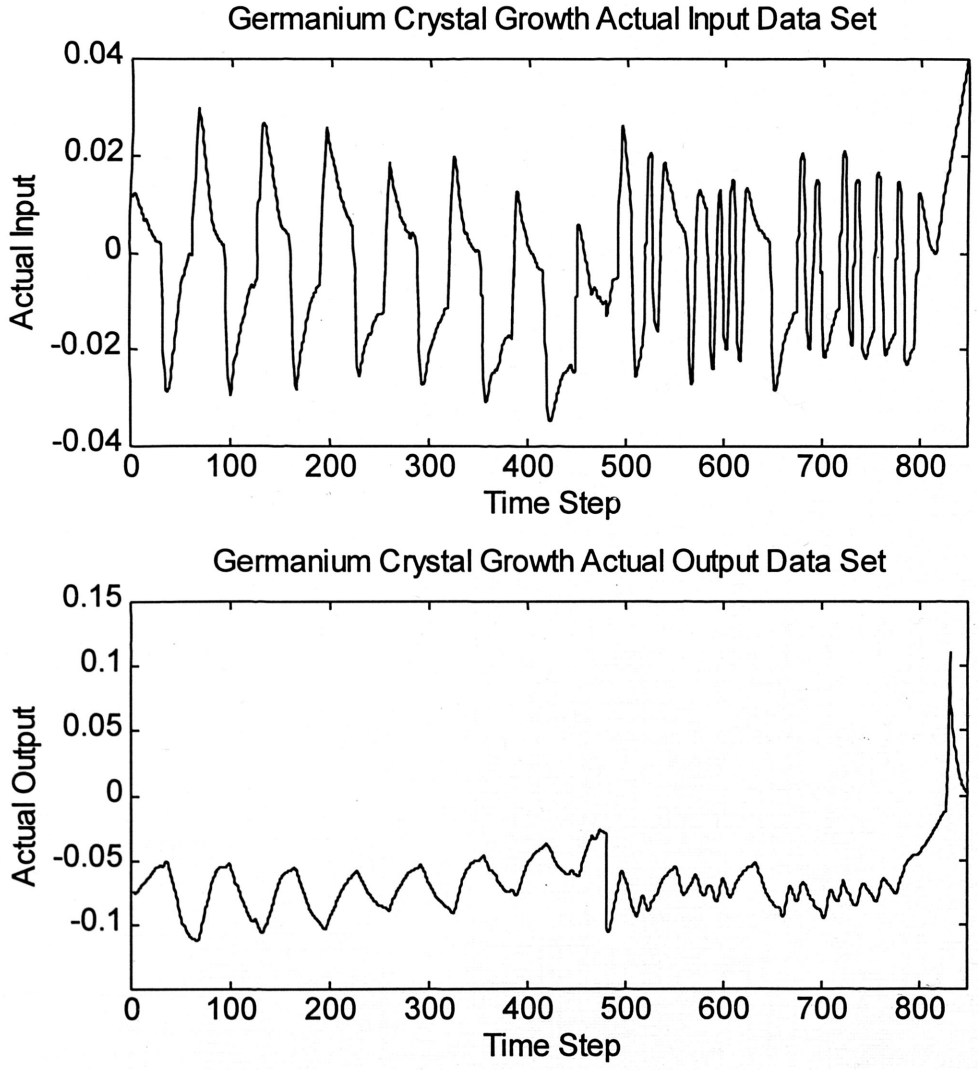

Figure IV.3 Training and Testing Data Set

The first training and testing runs were done with the square wave data set as the training set and the first half of the PRBS input data set as the testing data set. Both of these sets together made up the input file to the ANN program The square wave data consisted of 480 training vectors each corresponding to a time step in the process. The first half of the

PRBS input testing set consisted of 200 input-output pairs. They are not training vectors because they will not alter the weights.

The network was trained many times with different learning rates, network sizes, and starting weights. For each training cycle or epoch in the training run, the network was tested with the PRBS test set. An error was calculated between the ANN response and the known system response. The ANN program was run on a Sparc 20 Sun workstation. It took an average of five to six hours to train the network with 50000 epochs. This demonstrates why training time is a concern. It was found that the square wave input did not excite enough dynamics in the system for the ANN to learn from and to accurately predict the system response to the PRBS input. This was not entirely unexpected as this is a tough task for any model. The results of the runs are shown in the following table.

Training Parameters for Square Wave Training Vectors

# of Previous Inputs	# of Previous Outputs	Starting Seed	# of Hidden Layers	Neurons in the 1st Layer	Neurons in the 2nd Layer	Learning Rate	# of Epochs	Percent Learned
15	4	154197	1	14	0	0.002	22328	55.3
9	3	674172	1	11	0	0.002	37952	54.5
8	2	425879	1	10	0	0.01	50000	Diverge
10	4	723178	1	8	0	0.0005	50000	52.9
6	2	-917156	1	9	0	0.0001	50000	52.3
8	3	559872	2	6	3	0.0001	50000	53.3
9	3	-674172	1	13	0	0.0002	50000	52.4
8	2	425879	1	10	0	0.00001	50000	50.8
15	4	154197	1	14	0	0.0001	50000	47.6

Table IV.1

The table depicts in order of columns from left to right: the number of previous input and the number of previous outputs, the seed for the initial random weight generator, the number of hidden layers, the number of nodes in the first layer, the number of nodes in the second layer, the learning rate, the number of epochs, and finally the percent accuracy with which the network predicted the output for the test data set.

Most of the testing percentages fell in the forty and fifty percent category with the exception of one. This one diverged due to the large learning rate. These are very low accuracy levels. The square wave input did not excite enough dynamics in the system for the network to sufficiently identify the characteristics of the system.

To overcome the above problem the training set was expanded to include 100 points of the PRBS input set. This added 100 more training vectors which contained information about the more complicated dynamics of the system. The remaining 100 points in the PRBS input set were again used for testing the network. This gave a total of 580 training vectors and 100 testing input-output pairs. Again the ANN was trained for many different network parameter combinations to identify the optimal settings. The results are summarized in Table IV.2 on the next page.

The performance of the ANN rose considerably when it was trained with a portion of the PRBS input data set also. The network was not able to generalize as well for network sizes that were too large or small. Runs 3,6, and 9 shown above were to large and noticeable performance degradation can be seen in the testing percentage. The same degradation occurred for networks that were too small as demonstrated in runs 18,19, and 20. Runs 12 and 13 were able to generalize well with a relatively small network size. The network used in run 12 was picked for the validation of the model

Table of Training Parameters for the Square and Random Training Vector Set

# of Previous Inputs	# of Previous Outputs	Starting Seed	# of Hidden Layers	Neurons in the 1st Layer	Neurons in the 2nd Layer	Learning Rate	# of Epochs	Percent Learned
8	2	-423978	1	10	0	0.0005	20000	87.1
8	3	871873	2	6	3	0.0004	20000	87.6
15	4	-154197	1	14	0	0.0003	20000	81.9
9	3	474172	1	13	0	0.00078	20000	88.5
8	2	295879	1	10	0	0.0001	20000	88.6
11	4	-893471	1	7	0	0.0006	50000	86.1
8	4	-392156	2	5	3	0.0004	50000	88.8
8	3	774823	1	9	0	0.001	50000	87.1
12	6	-361192	1	14	0	0.0008	50000	85.4
9	3	674147	1	11	0	0.0007	50000	88.6
9	3	635173	2	7	4	0.0001	50000	87.2
4	2	-795481	1	6	0	0.001	50000	89.6
4	4	-392156	2	4	3	0.0009	50000	89.4
8	2	572923	1	10	0	0.0001	50000	86.8
7	2	-361192	1	10	0	0.0008	50000	88.6
8	2	674147	2	5	3	0.0007	50000	89.3
4	2	439483	1	6	0	0.001	50000	89.7
3	1	563213	1	5	0	0.001	50000	81.8
3	1	-872156	1	5	0	0.001	50000	80.9

Table IV.2

IV.6 Validation of the Model

For the validation of the model, the data set that was used to train the ANN is presented to the network. If the network learned well, then the response should be similar to the system response used to train the network. The test data set originally used to calculate the networks capability to generalize is also presented to the ANN. The response is then compared with the corresponding system response. In addition to this test data, more validation data which the ANN has not been exposed to for both training and testing is applied to the ANN. If the ANN truly has the capability to generalize

outside the data set it has been exposed to, this network response should also accurately reflect the system response. The data set used to verify the ANN is shown in the figure below.

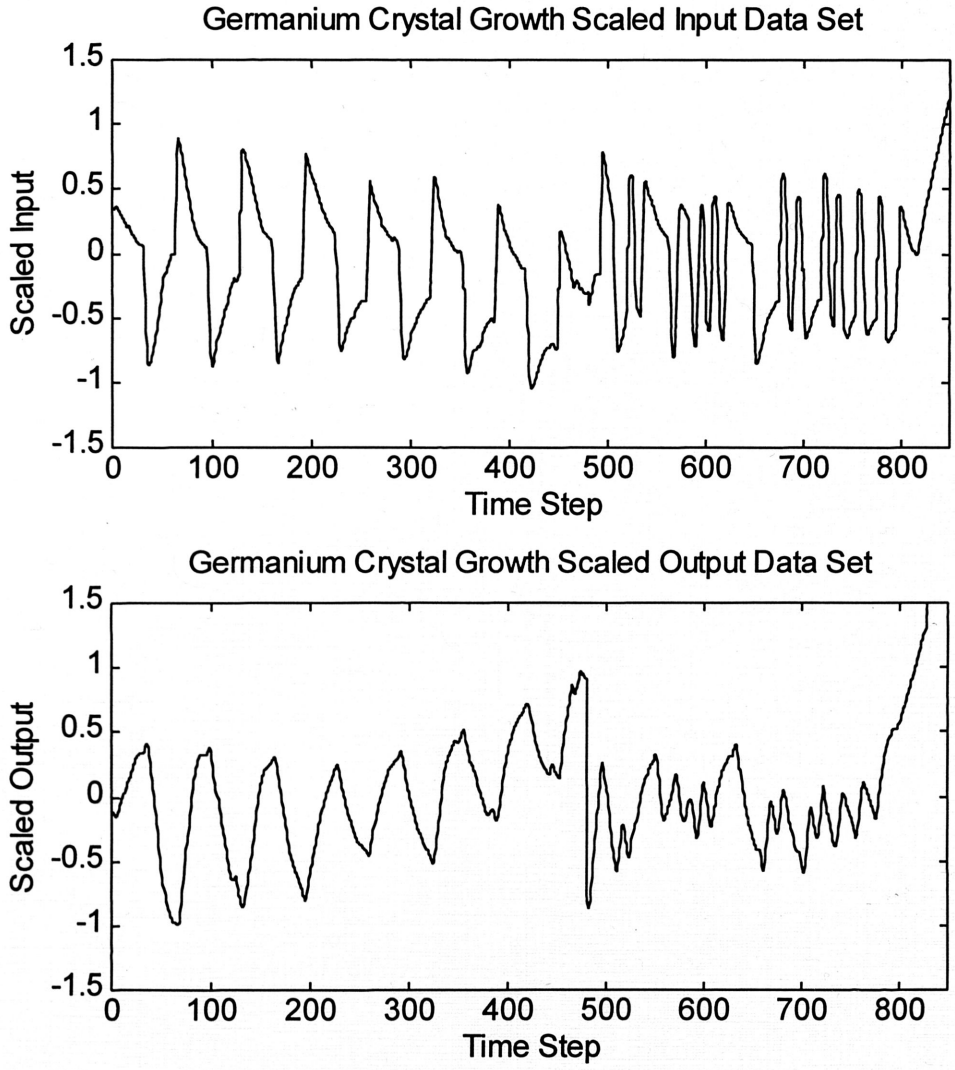

Figure IV.4 Scaled Germanium Growth Data

The data has been scaled. This scaling helps the ANN learn more accurately and quickly. This is due to the hyperbolic tangent squashing functions in the hidden neurons. Since these functions scale data from -1 to 1, it is natural for the ANN to deal with data in this

form. The first 480 vectors of this data set are the square wave signals used for the training. The next 100 (480-580) are the PRBS signals that were also used for the training. The ANN has not been trained with any of the data after this point. Data points 580-680 are used for the test data set. Points 680-850 are new PRBS validation data points for the ANN. These should again validate the networks ability to generalize. Since it is a PRBS signal it will be an indicator of how well the ANN will predict given an unexpected situation.

The verification was done by teacher forcing. This means that the current system output response is not available to the ANN. The system output responses before the current time step are available, though. This means the ANN is predicting one time step ahead. To predict more time steps ahead, a corresponding number of previous system responses would no longer be available to the ANN.

The validation data set and the corresponding ANN response are split into four plots on the following four pages. The plot was split into four so the ANN's performance could be made more clear. The germanium crystal system's response is indicated by the solid line. The ANN model's response is indicated by the dashed line.

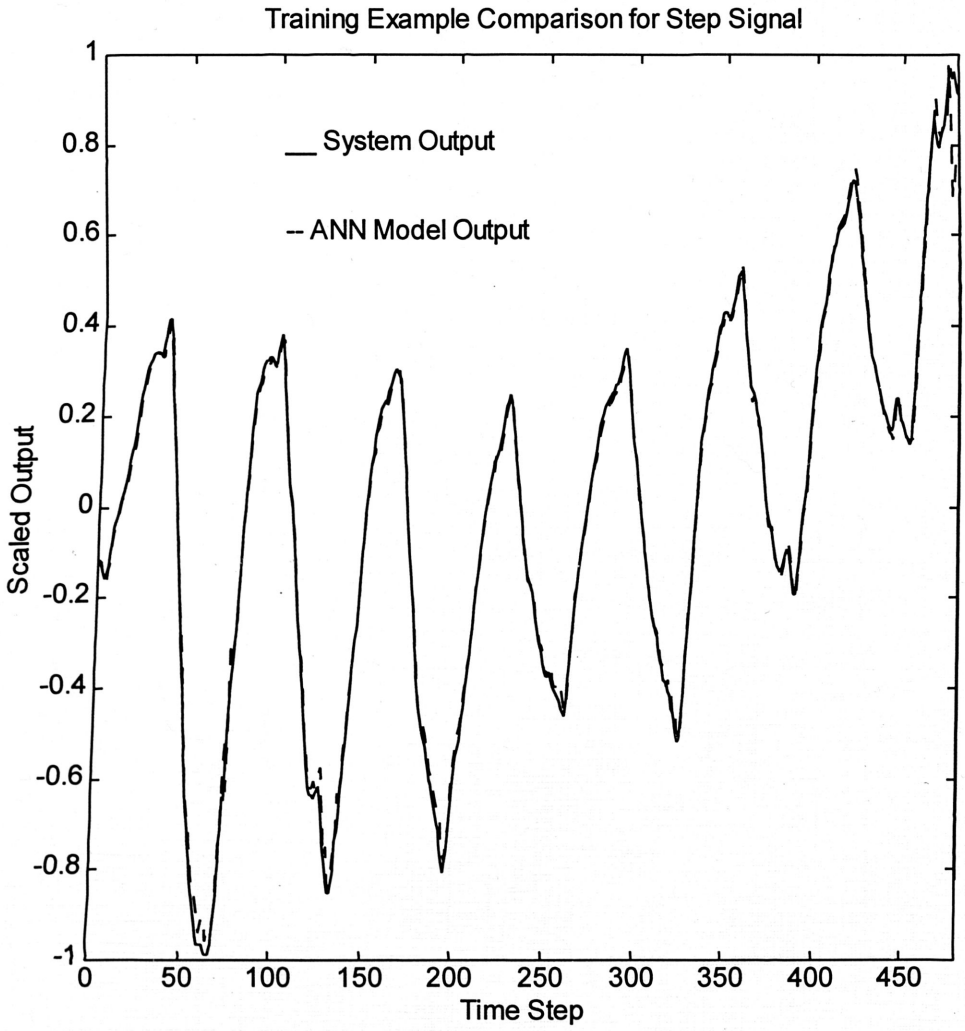

Figure IV.5 Data Points 0-480

These 480 points represent the system and ANN response. The model fits the data very well. This is expected because this was the actual training data. The ANN had seen this data before.

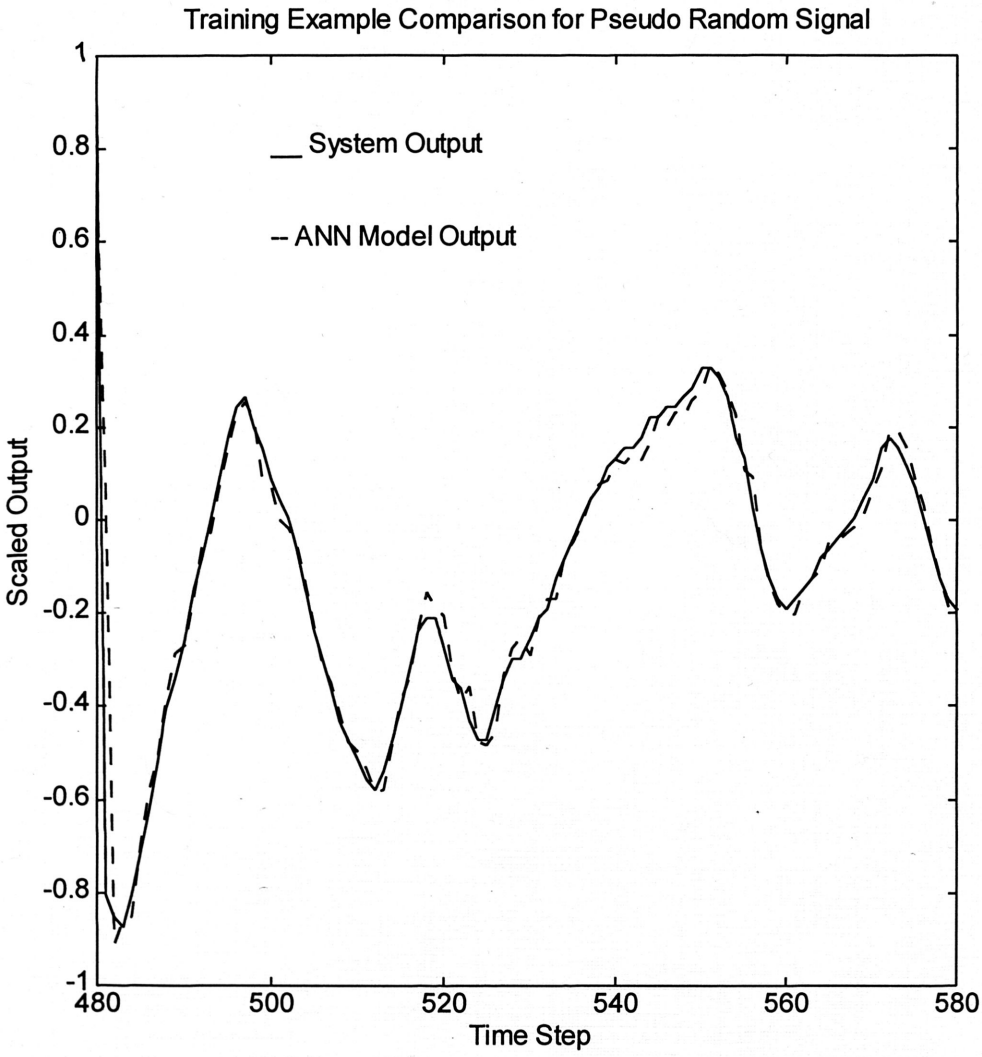

Figure IV.6 Data Points 480-580

The ANN identifies these 100 PRBS response points well also. This is again expected as the ANN was trained with this points. There is a little more error than in the simple square wave response.

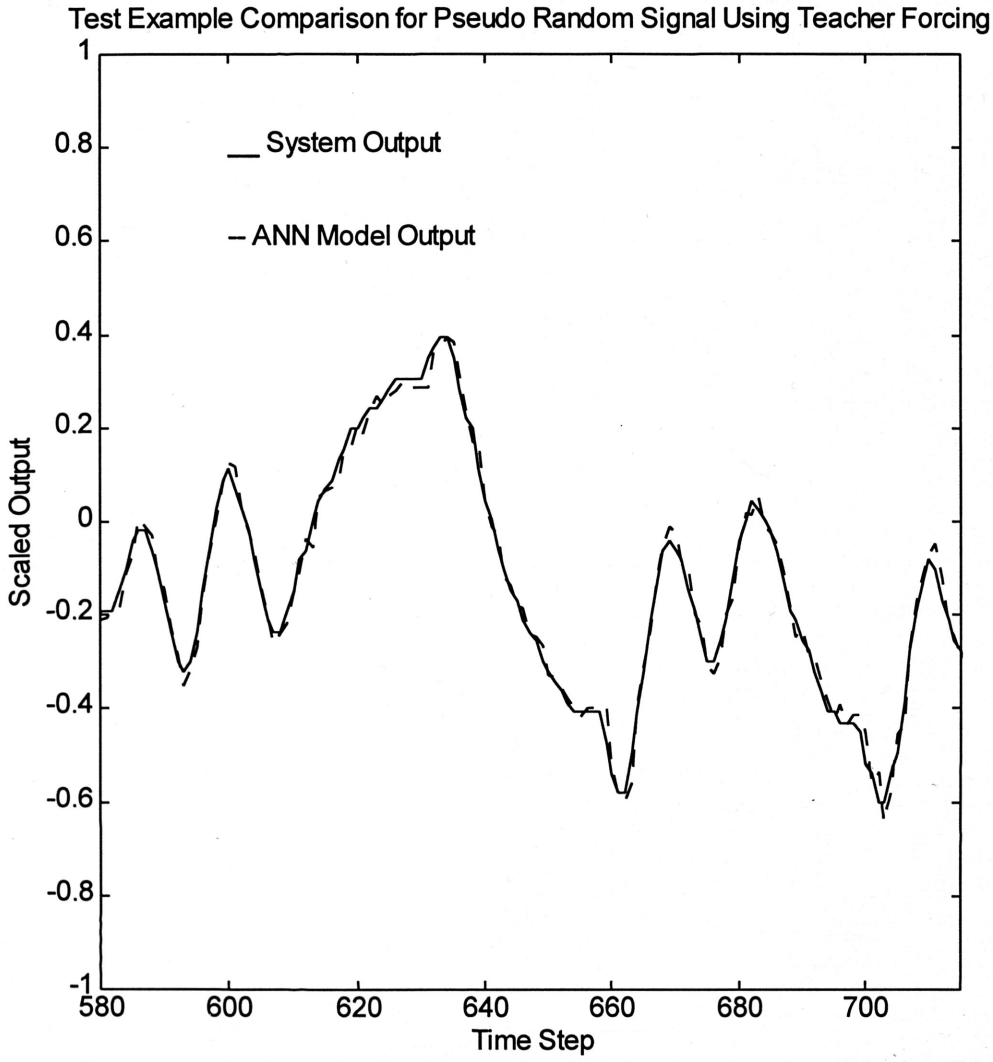

Figure IV.7 Data Points 580-715

These data points are new to the network and test the ANN's ability to generalize to new unforeseen crystal growth situations. The ANN again fits the response well.

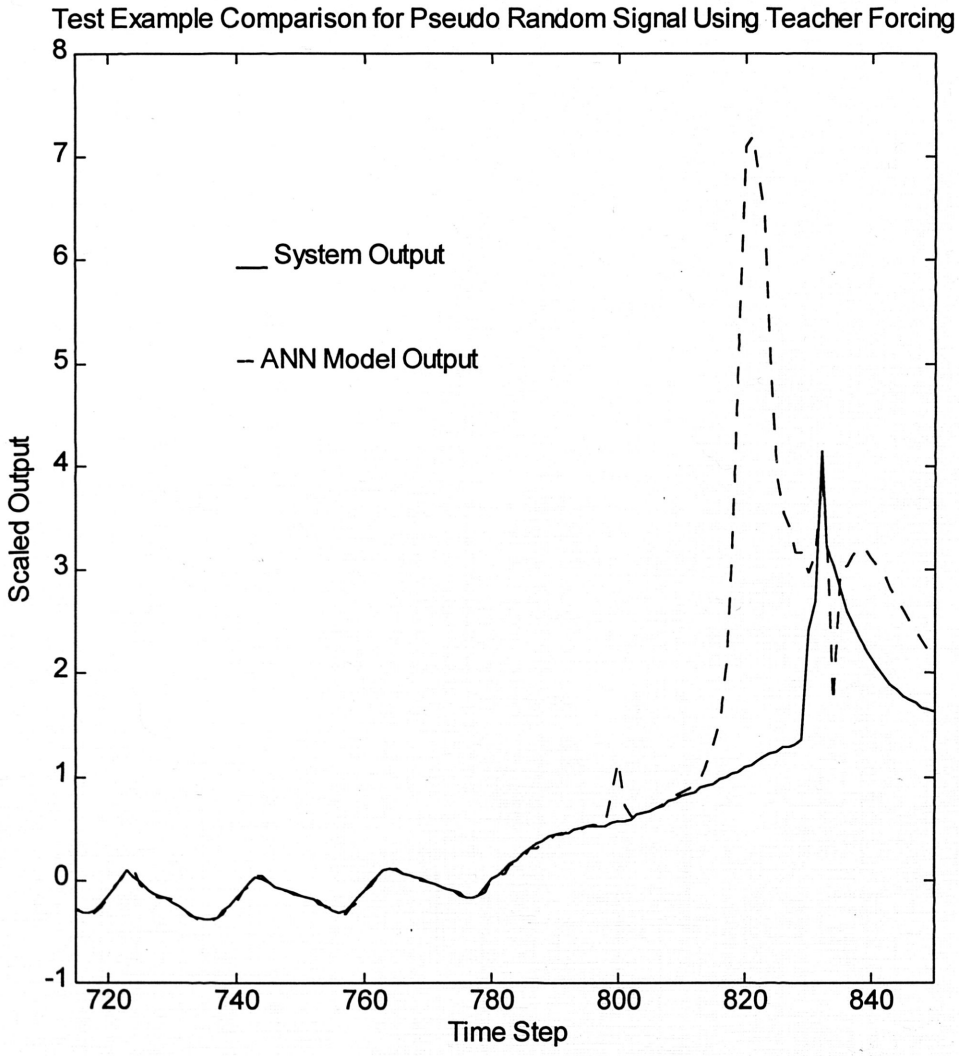

Figure IV.8 Data Points 715-850

This portion of the data set contained a sharp perturbation. This was the only portion of the data that the ANN did not closely follow the system response. It did predict the trend, however. It identified the response well up until the perturbation. The ANN was able to infer the dynamics for the PRBS signal. Even with the large perturbation, the ANN was able to predict the trend.

The response is compared with Ghassempoory's transfer function model's response to examine the improvements. A plot of a typical transfer function model response is shown in Figure IV.9.

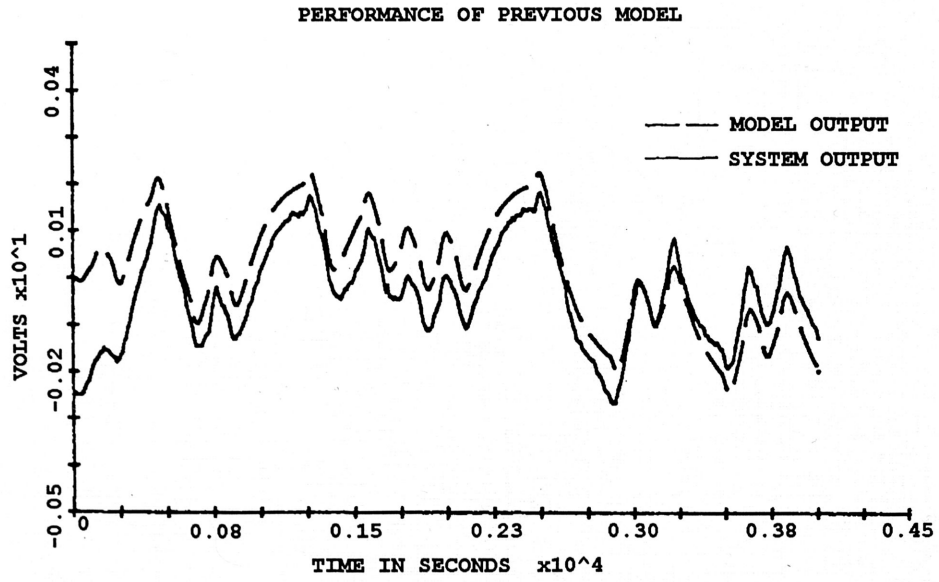

Figure IV.9 Transform Function Model Response

Ghassempoory used the Z-transfer function version of the transfer function to test the model. The general form is given by the following:

$$H(Z) = \frac{a_0 + a_1 Z^{-1} + \ldots + a_n Z^n}{1 + b_1 Z^{-1} + b_2 Z^{-2} + \ldots + b_m Z^{-m}} \qquad (33)$$

The coefficients of the model that Ghassempoory used for the PRBS data set used to test and validate the ANN are given by the following sequence:

{a_0= 0.291E-2, a_1= 0.874E-2, a_2= 0.919E-2, a_3= -0.126E-2, a_4= -0.125E-1,

a_5= -0.101E-1, a_6= -0.124E-1, a_7= -0.102E-1, a_8= -0.136E-1, b_1= -0.105E1,

b_2= -0.571E-1, b_3= 0.520E-1, b_4= -0.602E-1, b_5= -0.316E-1, b_6= 0.107,

b_7= -0164E-1, b_8= 0.614E-1}

This model also is predicting one time step ahead. There is considerably more error in this model's response. The system data the transfer function model is responding to is the same data used to identify the model. Another drawback of Ghassempoory's transform function modeling approach is that he has derived a different model for each type of input. Only one ANN model was needed to effectively predict one time step ahead for the different types of inputs to the system. The ANN responded well to data it had not been exposed to before. The transfer function's response was not shown for any large perturbations.

The ANN model performed well for one time step ahead prediction. The next step, which could be done in future research, would be to test the models capability to predict several time steps ahead (also known as global feedback). The network would have to rely on less information and build its predictions on its last predicted response. In this situation, there is the potential for errors to accumulate. If the ANN model could predict more than a few time steps ahead, it would have a significant impact on the control of complex system dynamics like those in the Czochralski crystal growth process.

CHAPTER V

THE ILMENITE EXPERIMENTAL SETUP AND FURTURE ANN MODELING RESEARCH

V.1 Introduction

The most rigorous test for any model is actual experimental data. The data used in the previous chapter was actual germanium crystal growth data, but it is good to create data for the model yourself. It is the best way to become familiar with the system to be modeled. Another benefit is the ability to acquire the data in a format that is most suitable for modeling. The crystal growth setup at the CEMDAS lab at Texas A&M University was not ready to grow ilmenite crystals and acquire data for testing in time for this thesis. The furnace, growth equipment, control, and data acquisition equipment were being built and developed concurrently with the development of the modeling tool. The progress made during the this period will be discussed in the following sections.

V.2 The Ilmenite Furnace

The furnace the initial research was done on was different than the final setup. It was determined a couple of months into the research that the current setup would not be adequate for growing quality ilmenite single crystals. The main factors that influenced this decision was the amount of vibration and eccentricity in the crystal pulling rod and

the leaks in the furnace which made it difficult to hold a constant atmosphere. A furnace that was designed more for crystal growth research was chosen to replace the current setup. It had not been used for several years and had to be rebuilt.

The new system consists mainly of a main chamber made of steel, a gearing apparatus on top to pull and rotate the crystal rod, and a gearing apparatus on the bottom for raising and rotating the crucible. The furnace is an induction heating furnace. The bottom rod connecting the gears to the crucible's base was significantly warped and had to be replaced. The replacement rod was checked for straightness in a machine shop. The vacuum seal between the main chamber and the bottom gear set had been corroded away. This was replaced by a spare from the previous setup. The ability of the furnace chamber to hold a vacuum was tested next. A quartz view port was cracked in the process and had to be replaced. The seal of the new furnace chamber was found to be a significant improvement over the old setup. The furnace was placed in its final position to test the water cooling system. A leak was found and fixed. Since eccentricity and correct alignment was a main concern, special parts were machined to test these features of the new system. Using these parts, the crystal and crucible mounting rod were tested and some adjustments were made. All the motors and gearing boxes were taken off to place a rubber pad between them and the furnace. This was done to further reduce the amount of vibration that reaches the crystal melt interface. The gearing mechanisms were completely disassembled and cleaned. The platform the furnace rested on was leveled. The water pressure of the source for the cooling system was found to be fluctuating significantly. A pressure regulator was used to correct this problem. A new crucible mount to hold the crucible was machined. It was designed to allow for some adjustment

of the crucible position while mounted and to minimize the coupling of the base to the r.f. heating coils.

After the furnace was assembled, several dry runs were made to test the ovens heating ability and to make sure everything worked properly. The crucible used to melt the ilmenite is made of iridium. Iridium was chosen for its high melting point, inability to react with ilmenite at high temperatures, and compatibility with an inductive heating setup. The crucible is placed in a graphite suceptor in which the eddy currents are linked. Since the graphite suceptor is close to the coils, it couples with the coils instead of the crucible. The graphite heats up, which in turn heats up the iridium crucible. This is done to protect the iridium crucible which is much more valuable. Both the iridium crucible and graphite suceptor are placed in an alumina tube and alumina felt is packed in between. This is done to prevent heat loss. This is then placed on the crucible mount in the furnace. Several runs were made where the furnace was simply heated up to test the new furnace. It worked properly and was able to melt the ilmenite.

V.3 Control and Measurements

The final setup will have temperature and weight measurement equipment. These signals will be sent to a computer which will control the power and the pull rate based on a model. The pyrometer for temperature measurement from the old setup is being used for the new one. The motor control for the pulling motor was switched over from an old computer to a new one. New software was acquired and set up. At this point, the motor is still controlled manually from the computer. A Labview data acquisition card was added to the computer to take in the signals from the measurement devices monitoring the

growth process. The signal from the pyrometer was found to be too small and noise effected it significantly. A RS232 communication card was added to the pyrometer to convert the analog signal to a digital signal to avoid noise problems and for data acquisition using the computer.

Two new motors and the controlling software are currently being ordered. A weight measurement system is soon to be ordered. Once these elements are installed, both temperature and crystal weight information will be available to the control. The control will be implemented using PID controller software that is compatible with the data acquisition card. The power control has to be switched over from a PLC system to a PC based system. When these tasks are accomplished, accurate data can be collected and recorded by the computer. This data can then be used to train and test the ANN model. This model can then be implemented in real time to predict the behavior of the crystal growth and adjust the PID controller appropriately.

V.4 Summary and Conclusions

The recent advances in ANN learning rules has created a lot of excitement about their possible applications in a wide range of problems. One of the areas where ANN are increasingly being used is in modeling and control. Many systems are difficult to control because of their nonlinear nature. Models are used to help predict what the system will do several time steps in advance. Accurate empirical models of these systems are difficult to develop and first principle models are cumbersome and are slow to implement online. ANN's offer a solution because of their ability to adaptively model complex nonlinear systems and the speed with which they provide results once they are trained. The actual

training times can be excessively long and sometimes the networks may fail to identify the system as accurately as possible by getting trapped in a local error minimum. ABP was chosen to avoid these problems. The varying learning rate helps to speed up the convergence to an error minimum while the jumpy nature of the algorithm tends to lift the network out of local minimums.

Czochralski single crystal growth is a highly nonlinear control problem and offers a good test of the ANN's ability to effectively model complex systems. Data on germanium single crystal growth from Ghassempoory's Ph.D. dissertation[13] was used to test the ANN's capability to model this type of process. The ANN's response was found to be significantly better than the response of the linear transfer function model used in the dissertation. In addition to identifying the actual output of the system better for one time step ahead prediction, only one model was needed to model different types of responses and the ANN model was able to accurately predict using data it had not been exposed to before.

The ANN model performed well on single step ahead prediction. The next step in the testing of the model is to see how well it does at prediction for several time steps in advance. This ability would make it a more powerful prediction tool for the control problem. Another good test of the ANN's modeling power would be to use data from a new material for which extensive research is not available. A new semiconductor material, ilmenite, is currently being examined in the CEMDAS lab at Texas A&M University. The crystal growth setup is still being refined at this stage. Once data can be collected on ilmenite single crystal growth, the ANN modeling technique can again be

tested using this data. If it performs well, the model can be implemented and used for the efficient control of ilmenite growth.

REFERENCES

1 Kumar, A. A. Pandey, R. K. Fogarty, T. N. Wilkins, R. , "Ilmenite as a Dual-Use Material," *Dual-Use Space Technology Transfer Conference*, NASA Conference Publication 3263, 1994.

2 Pandey, R. K. Nigli, S. Sunkara, S., "Evaluation of Wide Bandgap Semiconductors in Iron-Titanate for High Temperature Applications", in *Transactions of the Second International High Temperature Electronics*, (Charlotte, N.C.), pp. XIV/13-XIV/17, June 1994.

3 Sunkara, S., *Growth and Evaluation of Ilmenite Wide Bandgap Semiconductor for High Temperature Electronic Applications*, Ph.D. Thesis, Department of Electrical Engineering, Texas A&M University, May 1995.

4 Wasserman, P. D., *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, 1989.

5 Nadi, F. Agogino, A. Hodges, D. "Use of Influence Diagrams and Neural Networks in Modeling Semiconductor Manufacturing Processes," *IEEE Trans. Semi. Manufac.*, vol. 4, no. 1, Feb., 1991.

6 Parlos, A. G. Fernandez, B. Atiya, A. F. Muthusami, J. Tsai, W. K. "An Accelerated Learning Algorithm for Multilayer Perceptron Networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, May 1994.

7 Muthusami, J. *Incipient Fault Detection and Identification on Process Systems Using Artificial Neural Networks*, MS Thesis, Department of Mechanical Engineering, Texas A&M University, 1992.

8 Haykin, S. *Neural Networks: A Comprehensive Foundation*, MacMillan, New Jersey, 1994.

9 Kim, K. M. Kran, A. Smetana, P. Schwuttke, G.H. "Computer Simulation and Controlled Growth of Large Diameter Czochralski Silicon Crystals," *Journal of Electrochemical Society- Solid State Science and Technology*, vol. 130, pp. 1156-1160, May 1983.

10 Srivastava, R.K. Ramachandran, P. A. Dudukovic, M.P. "Czochralski Growth of Crystals: Simple Model for Growth Rate and Interface Shape," *Journal of Electrochem. Soc.: Solid-State Science and Technology*, May 1986, pp. 1009-1015.

11 Gevelber, M.A. Stephanopoulos, G. "Dynamics and Control of the Czochralski Process, I. Modeling and Dynamic Characterization," *Journal of Crystal Growth*, vol. 84, 1987, pp. 647-668.

12 Ghassempoory, M. Morgan, C. Hurle, D.T.J. Joyce, G.C. "Transfer Function Modeling of the Czochralski Crystal Growth Process," *Transactions in Instrument Measurement and Control*, vol. 9, no. 3, July-Sept 1987, pp. 147-151.

13 Ghassempoory, M. *Identification of the Dynamics of Czochralski Crystal Growth*, Ph.D. Thesis, UWIST, Cariff, 1984.

14 Pamplin B.R., *Crystal Growth*, Pergamon Press, Oxford, 1980.

15 Brice, J.C. *Crystal Growth Processes*, Blackie Halsted Press, pp. 129-154, 1986.