Copyright © 1989, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

MODELING COMPLEX MANUFACTURING PROCESSES VIA INTEGRATION OF INFLUENCE DIAGRAMS AND NEURAL NETWORKS

by

Fariborz Nadi

Memorandum No. UCB/ERL M89/123

8 November 1989

COVER PAGE

MODELING COMPLEX MANUFACTURING PROCESSES VIA INTEGRATION OF INFLUENCE DIAGRAMS AND NEURAL NETWORKS

by

Fariborz Nadi

Memorandum No. UCB/ERL M89/123

8 November 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

MODELING COMPLEX MANUFACTURING PROCESSES VIA INTEGRATION OF INFLUENCE DIAGRAMS AND NEURAL NETWORKS

by

Fariborz Nadi

Memorandum No. UCB/ERL M89/123

8 November 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Modeling Complex Manufacturing Processes via Integration of Influence Diagrams and Neural Networks

by

Fariborz Nadi

ABSTRACT

With the increase in the complexity of manufacturing processes comes the need for models that accurately describe their behavior. Accurate models enable us to operate these processes in an optimal manner. These models need to possess learning capabilities in order to capture the complex and dynamic behavior of a manufacturing process. They should also allow random extraction of information from the model.

The work described in this thesis introduces an architecture for modeling complex manufacturing processes. This architecture combines the *qualitative knowledge* of human experts, and the learning abilities of neural networks. Here, the learning abilities of neural networks are used to extract the *quantitative knowledge* that relates parameters of a manufacturing process. In creating a model, the qualitative knowledge of human experts is captured by making use of the relational level of influence diagrams. An influence diagram, at the relational level, is a graphical representation of the process in which relevant parameters are identified as nodes, and directed, acyclic arcs represent the existence of probable relationships amongst the connected nodes. By establishing the influence diagram, a multi-input multi-output process gets broken down into several independent multi-input single-output sub-processes. Next, the quantitative knowledge that relates a group of parameters in each of the sub-processes is extracted by employing feed-forward error-backpropagation neural networks. These networks consist of interconnected simple processors that store their knowledge, in a distributed manner, in the weight of the connections between them. The learning algorithm used for training these networks is a supervised learning procedure in which a sampled set of input and output vectors are chosen to train the networks. The selection of the training set is also done by

the help of human experts. The weights of the connections that represent the knowledge of the networks are calculated by employing a numerical optimization technique, in our case, a gradient descent routine. Next, in order to re-integrate the sub-processes into a process, a single associative memory network, which simultaneously looks at all of the parameters in a model, is used to learn the common and optimal behavioral space of all of the sub-processes. This network uses the same learning procedure as the previous networks, namely, error-backpropagation. Associative memory networks are used as content addressable memory, in which unknown elements of a given vector are calculated based on a best match stored in the network. This concludes the modeling, or the knowledge acquisition phase.

For the knowledge extraction phase, we have introduced a procedure to synthesize single, or groups of neural networks simultaneously. The problem of synthesis can be defined as such: given the value of some of the input and/or output parameters in the model, calculate the value of the remaining parameters, such that they are in agreement with the optimality criteria stored in the associative memory network, and the causal relationships describing the behavior of the sub-processes stored in the feed-forward networks. In this procedure the weights of the connections between processors are held constant, while the value of the unknown input and/or output parameters are calculated by employing a stochastic optimization routine called ALOPEX [42]. The process of synthesis begins with synthesizing the single associative memory network to generate initial estimates for the unknown parameters. This is done to limit the search space to specified areas in the simultaneous synthesis of the feed-forward networks. Next, to provide the final solutions, all of the feed-forward networks are synthesized simultaneously using the initial estimates provided by the associative memory network.

The effectiveness of this architecture to model complex processes is tested by modeling two Very Large Scale Integration (VLSI) manufacturing processes, and then comparing these models to models generated by other techniques, such as statistical regression analysis. In our study, the models generated by the integration of influence diagrams and neural networks are, on the average, at least twice more accurate, while given only half as much information in creating these models. The two processes that were modeled are: dry-oxidation of silicon, and Low Pressure Chemical Vapor Deposition (LPCVD) of polysilicon. For the dry-oxidation modeling, the effects of process time, process temperature, cleaning process, and crystallographic orientation of substrates are studied on the thickness of the resultant oxide film. For polysilicon deposition, three output parameters are related to five input parameters. The output parameters are: deposition rate, film thickness, and stress of the film. The input parameters are: process time, process temperature, ambient pressure, flow rate of the active gas (silane), and wafer position.

Furthermore, by employing the generalization capabilities of neural networks, we have generated novel ideas about the process. Novel ideas are new knowledge created by generalizing the information gained in the learning phase. In our case, two such ideas are generated. The first is a zero stress recipe for LPCVD of polysilicon, and the second is a non-uniform temperature distribution across the processing tube for a uniform deposition rate of polysilicon. In the case of the zero stress recipe, none of the data used to train the networks has zero stress value. In the case of a uniform deposition rate, all of the training data are for uniform temperature depositions, which produce exponentially dropping deposition rates across the processing tube.

In the last part of this work, the noise filtering capacity of the feed-forward networks to filter inherent, normally distributed noise of process parameters, is demonstrated by employing a known non-linear relationship to generate a noise-polluted training set, and by extracting the original noise-free relationship at the end of the learning cycle.

TO

MY PARENTS

AKRAM & RAHIM NADI

THANK YOU FOR EVERYTHING

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank many of the people whose academic, financial, and moral support have contributed to the completion of my thesis.

To my research advisers, Prof. Alice Agogino and Prof. David Hodges, I would like to offer my sincere gratitude, for providing me the opportunity, the freedom, and the guidance to do my research. I can not ever thank you enough.

To Prof. David Dornfeld, thank you for the guidance and help you have provided me through most of my stay here at Berkeley, and your effort in reviewing this thesis.

To Kuang-Kuo Lin, Costas Spanos, Dah-Bin Kao, and Bruce Deal, thank you for providing me with the very valuable and hard to gather experimental data necessary for my work, and for the many helpful discussions we have had together.

To my friend Sabbir Rangwala, for his friendship and his helpful insights in the field of neural networks. Thank you Sabbir.

I would like to thank many of the philosophers, poets, and writers, whose distant voices from the past showed me the light when I could not see.

I would like to thank Semiconductor Research Corporation, MICRO, Harris Semiconductor, IBM, Intel, National Semiconductor, Philips/Signetics, Rockwell International, Sandia National Laboratories, Siemens AG, and Texas Instruments, for providing the funds for my research.

I would like to thank many of my friends, especially Yousef, Reza, Alireza, Essi, and Syrous, who helped me through times of confusion and frustration, and also gave me some of the best memories of my life.

This thesis is dedicated to my parents, Akram and Rahim Nadi, who have nurtured, loved, and supported me throughout my life. I can not ever repay you for what you have done for me. I just hope that you are proud of me.

Table of Contents

Dedication ii

Acknowledgements iii

Table of Contents iv

1. Introduction 1

2. Influence Diagrams

  2.1 Introduction 7

  2.2 Construction of Influence Diagrams

    2.2.1 Relational Level 9

    2.2.2 Structural Level 11

    2.2.3 Numerical Level 11

3. Neural Networks

  3.1 Introduction 14

  3.2 Background on Neural Networks 14

  3.3 Feed-Forward Multi-layered Networks 17

    3.3.1 Learning Procedure for Feed-Forward Multi-layered Networks 19

    3.3.2 Synthesis Procedure for Feed-Forward Multi-layered Networks 23

  3.4 Associative Memory Networks 26

    3.4.1 Learning Procedure for Associative Memory Networks 27

    3.4.2 Synthesis Procedure for Associative Memory Networks 27

4. Integration of Influence Diagrams and Neural Networks

  4.1 Introduction 38

4.2 Architecture of the Integrated Network 38

4.2.1 Learning 39

4.2.2 Synthesis 40

4.2.3 Calculation Procedure for Synthesis 41

4.3 Significance of the Influence Diagrams 43

4.4 Further Explanation of Structural & Numerical Levels of Influence Diagrams

4.4.1 Structural Level 44

4.4.2 Numerical Level 45

5. Experimental Results

5.1 Introduction 53

5.2 Dry-Oxidation of Silicon 54

5.2.1 Oxidation Kinetics 54

5.2.2 Integrated Network 56

5.2.3 Learning 57

5.2.4 Synthesis 59

5.3 Low Pressure Chemical Vapor Deposition of Polysilicon 60

5.3.1 Deposition Parameters 61

5.3.2 Integrated Network 62

5.3.3 Learning 62

5.3.3.1 Comparing Performance of Neural Networks to Other Models 64

5.3.4 Synthesis 66

5.3.4.1 Generating a Recipe for a Zero Stress Polysilicon Film 66

5.3.4.2 Generating a Non-Uniform Temperature Distribution for a Uniform Deposition Rate 67

5.4 Conclusions 69

6. Dealing with Process Noise

6.1 Introduction 87

6.2 Filtering Output Noise for a Single-Input Single-Output, Non-Monotonic, Non-Linear Relationship 88

6.3 Filtering Input and Output Noise for a Multi-Input Single-Output, Monotonic, Non-Linear Relationship 89

7. Conclusions and Future Research Direction 96

References 101

CHAPTER 1
INTRODUCTION

Due to the increased complexity of the modern manufacturing processes, economically feasible production facilities are being required to incorporate more intelligence in their operation. Integration of intelligence results in manufacturing systems that are predictable in their behavior, user-friendly, and less sensitive to human error. To realize these goals more and more manufacturing environments are moving towards Computer Integrated Manufacturing (CIM). CIM environments operate in three main modes:

1) Create a centralized pool of data which captures the required information about the operation of the manufacturing elements. These elements could be: incoming raw materials, processing equipment, human operators, and so on.
2) Use the gathered data to make decisions about the future operation of the plant. This process could be automatic or involve human input.
3) Implement the made decisions via the direct or indirect connections to the manufacturing sites.

The decisions made by CIM are hierarchical in nature and could range from scheduling the production of the plant for the next year, down to specific problems such as setting the parameters of a specific processing unit for a specific product. At the present, however, a lot of these decisions are made by humans, especially at the bottom or machine level. The reason for this is that CIM does not have the required knowledge about the manufacturing processes to make these decisions. The required knowledge could be in the form of an accurate model of the manufacturing process. This is especially true for complex processes, such as Very Large Scale Integration (VLSI) manufacturing. These processes are very sensitive in general and depend on a lot of parameters. There are general models, usually based on physical principles of the process (first principle models), that give an understanding of the process and predict its behavior to some degree. But most of these models fail to predict the behavior of a process with the high accuracy required for an efficient and reliable manufacturing operation.

In most real life situations the process parameters are set or modified by human experts who are closely familiar with specific processes. The knowledge gained by the experts is mostly through observing the behavior of the process for a long time and thereby creating an internal representation of it in their minds. This knowledge, in turn, helps them in setting the process parameters. One possible method of gaining expertise, that is true of human learning in general, is learning generalities by observing a lot of specifics. One might explain the process of becoming an expert in the following manner: first, a novice will observe or learn about a few important parameters which affect the process. Next, by observing the behavior of the process over time he/she will learn of a general relationship between these parameters. At the same time he/she might notice other new parameters which affect the process and therefore update his/her knowledge structure. As the number of observations grows, the knowledge gained becomes more detailed and is partitioned into finer spaces, such that the expert can establish which parameters are affected by each other and which are independent of each other.

Accurate models of manufacturing processes are pre-requisite for automating the process of recipe generation (setting of the process parameters). To create such models, we can combine first principle models (if available), and one of the following general methodologies:

1) Create an analytic model of the process which takes into account the behavior of the processing equipment. This feature is necessary since the physical characteristics of the processing equipment affect the outcome of the process. The parameters in the model are determined by performing numerous experiments and using statistical and/or regression techniques. This approach is beneficial in setting up a process initially, but is expensive otherwise. The model would also lack in terms of accuracy if

there are a lot of parameters involved. Besides, if the algorithm were not adaptive, the accuracy of the derived model would suffer more as the process shifted (a normal behavior of manufacturing processes). This shift is usually due to component aging and other time dependent factors.

2) Capture the knowledge of the human expert by creating a knowledge base and an inference mechanism to retrieve the information. The main problem here is in creating the knowledge base. For example, in rule-based systems the knowledge of the expert is interpreted in terms of if-then type of rules, and for a complex manufacturing process the number of rules grows very rapidly and becomes unmanageable. Although there are areas that lend themselves easily to this type of knowledge transfer, in general there are not many. This technique, if not adaptive, faces the same problems as the first approach in terms of a shift in the behavior of the process.

3) Create an adaptive learning architecture that requires minimal human knowledge. In this approach the knowledge about the process should be gained by observing its behavior over time and adapting to it as it changes.

In this work we have introduced such an architecture, which is capable of learning and synthesizing behavior of complex manufacturing processes. This is an adaptive architecture, which captures both qualitative, and quantitative aspects of the knowledge of the manufacturing process.

The *qualitative knowledge* is captured by making use of the relational level of influence diagrams. An influence diagram, at the relational level, is a graphical representation of the process, in which important process parameters are identified as nodes, and directed, acyclic arcs between these nodes depict existence of conditional dependence between related parameters. These diagrams are an abstraction of the process which can be used to break down the task of modeling a single, large, and complex process (multi-input, multi-output), into modeling several smaller, independent, and less complex sub-processes (multi-input, single-output). Influence diagrams are created with the help of human experts, or they can be induced by observing sampled behavior of the process over time [5]. Unlike any other method of knowledge transfer, it is very easy and natural for human experts to create the relational level of the influence diagrams.

This is the extent of the knowledge required from the experts. They are not required to specify the exact nature of the relationships between the related parameters, although such knowledge can also be incorporated into this architecture. The quantitative relationships between groups of related parameters (sub-processes) are extracted using neural networks. These networks have been proven to be very effective in complex learning tasks, such as human speech recognition [31]. Neural networks crudely resemble the architecture of the brain, in which a lot of simple processors are interconnected and the knowledge is stored, in a distributed manner, in the weight of the connections between processors [13]. These processors operate in parallel and therefore are very efficient in terms of the speed of computations they perform. There is an on-going effort to build these networks on VLSI circuits [27-30].

The neural networks used to capture the *quantitative knowledge* in each of the sub-processes are of the multi-layered feed-forward error-backpropagation type. These networks have been used successfully in a large variety of complex learning tasks [31].

The second contribution of this work is to introduce a very flexible method of synthesizing single, or groups of neural networks simultaneously. Once these networks learn the relationships between groups of related parameters (sub-processes), we need a way of extracting this knowledge in any random manner. That is, given partial information about some of the parameters, we should be able to produce the value of the unknown parameters using the knowledge of the neural networks. To do this we have employed a stochastic optimization technique called ALOPEX [42], which is very similar to simulated annealing [45].

The process of synthesis (knowledge extraction) happens in two phases: (1) generation of initial estimates for the unknown parameters, and (2) fine tunning the initial estimates using the quantitative knowledge. The generation of initial estimates is done by synthesizing a single associative memory network that has captured the common and optimal operating space of all of the sub-processes in the learning phase. This network is also of the multi-layered feed-forward error-backpropagation type. The only difference is in the selection of the input and the output parameters of the network, and in the definition of the objective function, which has to be minimized using the ALOPEX optimization technique.

The final values of the unknown parameters are generated by simultaneous synthesis of the rest of the networks, each possessing the quantitative knowledge about one of the sub-processes. This part is also done by employing the ALOPEX optimization technique.

The final contribution of this work is to employ this architecture in modeling and synthesizing two VLSI manufacturing processes and comparing them to models generated by statistical regression techniques. We will show that, in our study, the models generated by the integration of influence diagrams and neural networks are, on the average, at least twice more accurate, while given only half as much information in creating the models. The two processes are: dry-oxidation of silicon; and Low Pressure Chemical Vapor Deposition (LPCVD) of polysilicon. For the dry-oxidation modeling we study the effects of process time, process temperature, cleaning process, and crystallographic orientation of substrate on the thickness of the resultant oxide film. For polysilicon deposition we look at three output parameters that are film thickness, deposition rate, and stress of the film, and relate them to process time, process temperature, flow rate of the active gas (silane), position of the wafer in the tube, and ambient pressure. We will also show how the generalization capabilities of neural networks can be used to generate novel ideas about the process. Novel ideas are new knowledge generated by generalizing the information gained in the learning phase. Two such ideas are generated in this work. The first is a zero stress recipe for LPCVD of polysilicon, and the second is a non-uniform temperature distribution across the processing tube for a uniform deposition rate of polysilicon. In the case of the zero stress recipe, none of the data used to train the networks has zero stress value. In the case of a uniform deposition rate, all of the training data are for uniform temperature depositions, which produce exponentially dropping deposition rates across the processing tube.

Finally in the last part of our work we will show how neural networks are effective in filtering inherent process noise and extracting the true relationships in each of the sub-processes. This is demonstrated by having a known relationship to generate a noise-polluted training set, and extracting the original noise-free relationship at the end of the learning cycle. This process is repeated for two non-linear relationships: one monotonic, and the other non-monotonic.

The work is broken down into seven chapters. Chapter two is an introduction to Influence diagrams and the way they are constructed along with examples. Chapter three makes us familiar with neural networks and their properties. It covers the type of neural networks used here, and their modes of operation. These modes are learning (knowledge acquisition), and synthesis (knowledge extraction). Chapter four is where influence diagrams and neural networks are combined, and the architecture of this modeling technique is defined. Chapter five is the results part of this work where two VLSI processes are modeled by the application of this approach. Chapter six is where we study the effects of process noise in the operation of neural networks. Chapter seven is the conclusion where the results are summed up and the future direction of research is laid out.

CHAPTER 2
INFLUENCE DIAGRAMS

2.1. INTRODUCTION

Influence diagrams were originally developed by the Decision Analysis Group at SRI International to automate modeling of complex decision problems involving several uncertain variables (Miller et al. [7]). Besides their original objective, influence diagrams have been used for improving communications among people, participative modeling of complex decision problems (Owen [8], and Howard and Matheson [9]), and as a development tool for building expert systems (Agogino and Rege [1-4], and Holtzman [10]).

In an influence diagram the knowledge of the dependencies between variables is presented at two levels: a qualitative level and a quantitative level. In the qualitative (or relational) level, which is a graphic representation of the problem under consideration, the variables of a problem are identified and represented as nodes, and any dependencies between related variables are depicted by directed acyclic arcs. This level of the influence diagrams is perhaps one of the most powerful ways of capturing the qualitative knowledge of human experts in a problem domain. This is due to the fact that it is very easy for human experts to express their knowledge in a graphical frame work where the only required knowledge is identifying critical variables and establishing directed arcs between conditionally dependent variables. The inclusion of uncertain variables or uncertain dependencies at this level does not hinder the ability of influence diagrams in modeling a problem domain. It is only when a true dependency or a critical variable is omitted that the ability of influence diagrams to model a problem domain suffers.

At the quantitative level of the influence diagrams, the knowledge about the conditional dependencies is captured and expressed in a numerical frame work. Traditionally, these dependencies are expressed in terms of probability distributions, where there is a probability distribution associated with each arc, or influence, in the diagram. This makes it possible to retrieve probabilistic information about the parameters in the diagram in any random manner. This is done by using laws of probability which enable us to combine these influences (probability distributions) in a manner that is consistent with the information requested from the diagram.

Despite its power and flexibility in terms of extracting knowledge from an influence diagram, knowledge acquisition is a problem in this approach, where the influences are expressed in terms of probability distributions. This is due to the fact that the probability distributions associated with the arcs of the diagram must be known before hand. The source of this information is usually the human expert, but humans are not good in producing these probability distributions.

In this work we have tried to minimize the amount of knowledge required from the human experts in the knowledge acquisition phase by making use of the learning abilities of neural networks. The reader will be familiarized with neural networks in chapter three. The only thing to be mentioned here is that the quantitative level of the influence diagrams, which itself consists of two levels: structural, and numerical, are handled by employing neural networks. These networks are used to learn the quantitative relationships between related parameters and also to extract this information in any random manner.

In this chapter we will discuss the construction of influence diagrams and explain each of the three levels of an influence diagram. These are the relational level, the structural level, and the numerical level. The relational level is described here, but the in-depth discussion about the structural and numerical levels is postponed to the end of chapter four, after the reader has been familiarized with neural networks.

2.2. CONSTRUCTION OF INFLUENCE DIAGRAMS

2.2.1. RELATIONAL LEVEL

This part is best described by an example. Such an example is given in Fig. 2.1. It is an influence diagram for dry-oxidation of silicon wafers. A brief introduction to the oxidation process is given in chapter 5. The point to be made here is how such a diagram is constructed.

There are three ways of constructing an influence diagram: (1) with the help of human experts, (2) use of first principles (knowledge about the physics of the problem), and (3) inducing the arcs of the diagram by looking at examples stored in a database [5]. It is also possible to construct the diagram by any combination of the three [6]. The first step though, is common for all approaches. This is the identification of the relevant variables. This task is not trivial if one is dealing with a complex process. For example, if one is to construct a diagram for an arc-welding process, with the weld quality being one of the variables under consideration, it is not obvious to know all of the influencing parameters that affect the quality of the weld, unless one has expertise in that area.

The next step is establishing the directed arcs or influences between the variables. If the diagram is being constructed by the help of experts, the dependencies are established from their knowledge. If the diagram is being constructed by use of first principles, the influences are established by the physical laws governing the process. If the diagram is induced from examples, laws of probability are used to extract the dependencies. Interested readers are referred to [5] for a complete description of the last method.

Looking at the influence diagram of Fig. 2.1, we can see that there are 10 variables under consideration, 7 input variables and 3 output variables. Input variables are defined as the parameters that are observable and controllable by the user. Output variables are the result of the input variables and might not be observable during the process. If we look at the output variable oxide thickness, it can be seen from the diagram that thickness is dependent on 4 input variables: process time, process temperature, substrate crystallographic orientation, and the cleaning process type that the substrate has gone through, before being processed. Stress is dependent on annealing time, annealing temperature, oxide thickness, and substrate crystallographic orientation. Surface defects of the resultant oxide is dependent on oxide thickness, stress in the oxide film, cleaning type, and finally, the number of previous runs, which is a measure of the cleanliness of the processing equipment. The essence of the information here is the conditional independencies. For example, stress is conditionally independent of cleanliness of the processing equipment, given information on annealing time, annealing temperature, oxide thickness, and substrate orientation.

It should be noted that influence diagrams are not unique, meaning that different experts could construct different diagrams for the same process. In our oxidation example, another expert could construct another diagram such as given in Fig 2.2. The difference here is that the second expert thinks stress is dependent on cleaning type, process time, process temperature, substrate crystallographic orientation, annealing temperature, and annealing time. And that stress is conditionally independent of oxide thickness. But as can be seen the two diagrams are equivalent, in that the oxide thickness inherits the influences from some of the same variables that affect the stress. But the two diagrams would be inconsistent if the second expert were to say that stress is also dependent on the number of previous runs. It will be shown later that inclusion of uncertain influences does not hinder the ability of influence diagrams in terms of modeling a process. It is only when a relevant variable or a true dependence is omitted that the ability to predict the behavior of a process suffers in terms of accuracy.

2.2.2. STRUCTURAL LEVEL

Once the relational level has produced the conditional dependencies between the variables in the influence diagram, in terms of directed acyclic arcs, the exact nature of these dependencies or relations are learned by employing the learning abilities of neural networks. These networks, once they have captured the quantitative knowledge, are also used for extracting information about the relationships between the variables. The structure of these networks are determined in the structural level. At this point since the reader might not be familiar with neural networks, the discussion about the operation of this level is postponed to the end of chapter four.

2.2.3. NUMERICAL LEVEL

At this level the relationships between the variables of an influence diagram are quantified by employing numerical optimization techniques to calculate the operating parameters of the neural networks. The discussion about this level is also postponed to the end of chapter four.

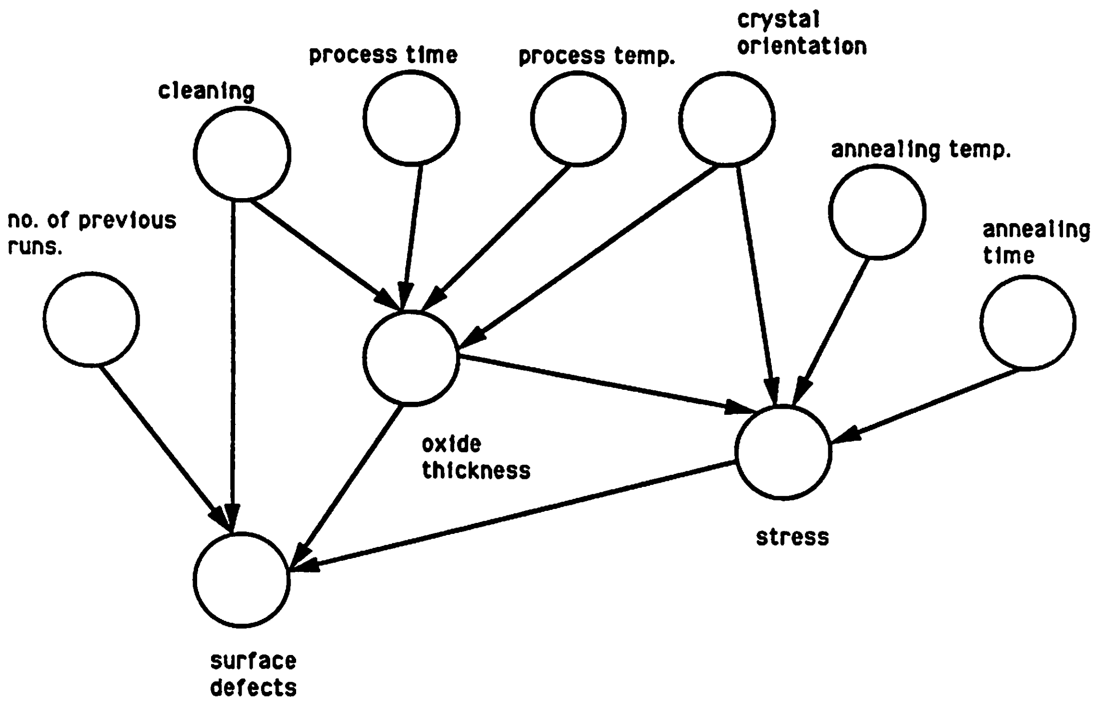


Fig. 2.1 Influence diagram for dry-oxidation of silicon.

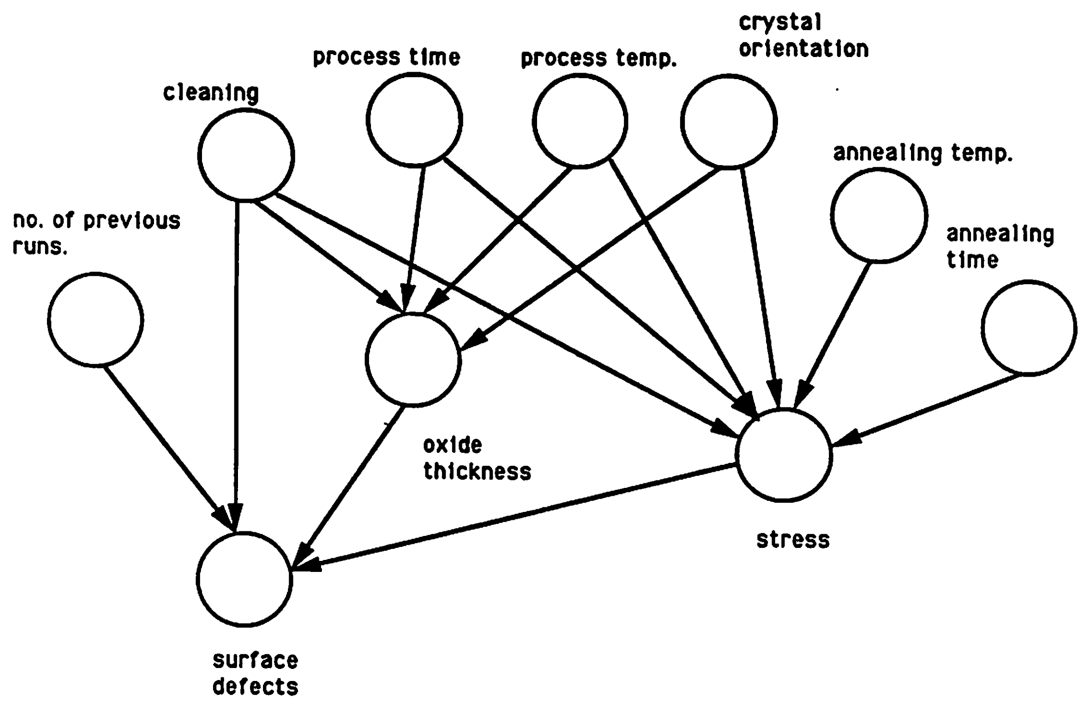


Fig. 2.2 Alternative influence diagram for dry-oxidation of silicon.

CHAPTER 3
NEURAL NETWORKS

3.1. INTRODUCTION

The ability of human beings to process complex types of information and to make inferences based on their knowledge is unequalled [4]. The ability of the human brain to perform such complex tasks with its parallel architecture was the motivation behind the development of neural networks or parallel distributed processing architecture [31]. The human brain is made of a large number of interconnected neurons, each possessing very simple computational abilities. However, the interaction between the neurons allows for parallel processing of information, which greatly enhances the speed of computation and causes a large amount of knowledge to be brought to bear in processing this information [15]. In this chapter we will give a description of neural networks and discuss learning and synthesis for two specific types of such networks.

3.2. BACKGROUND ON NEURAL NETWORKS

Neural networks are collections of simple, interconnected processors, which operate in parallel and store knowledge in the strength of the connections between the individual processors. Such networks of computing elements crudely resemble processing activity in the brain and have been successfully applied to intelligent tasks such as learning and pattern recognition[13].

Although the connectionist computing paradigm was proposed by Rosenblatt [14] in 1950, it did not become popular until recent years. The reasons being, lack of powerful computers for simulation of such architectures and, eventually, inability to implement them on hardware. The recent advances in VLSI technology shows the potential of fabricating electronic neural networks [27-30]. Another advantage of such architectures is graceful degradation. Since the knowledge is stored in a distributed manner over all of the connections, failure of some of the connections or nodes will not result in a complete loss of information.

Neural networks have been used in a wide variety of applications such as pattern recognition in human speech, and vision [31,32], noise filtering [39], content addressable memory [16,17,40,41], intelligent control [33-36], optimization [18,19], diagnostics [11], and others including human cognition [31].

There are different types of neural networks suited for different applications. In general they can be differentiated in three ways: (1) type of nodes or neurons, (2) pattern of connectivity between neurons, and (3) type of operation (mode of learning and computation) of a network.

Each node or neuron has several weighted inputs comming from the output of other nodes, and one output that is connected to the input of several other nodes via another set of weighted connections. In general there is a simple relationship between the input and the output of a node, Fig. 3.1a.

$$output = f\,(\,weighted\ sum\ of\ inputs\,) \tag{3.1}$$

The function f in Eq. 3.1 could be linear or non-linear. A linear relationship would be a simple summation function ($output = weighted\ sum\ of\ inputs$). In which case a network with linear nodes is capable of learning a set of linearly independent vectors. In case of non-linear nodes, f is usually assumed to be a sigmoid function of the input or in some cases, a threshold function may also be used (refer to Fig. 3.1b). A network with non-linear nodes is capable of encoding more complex information than a network with linear nodes.

In terms of pattern of connectivity of a network, there are two general types.

1) Layered networks, Fig. 3.2a, where there is an input layer and an output layer. These layers receive information from the outside world and transmit the results back to it. In some cases the input and the output layers are the same. There could be any number of layers in-between or on top of the input and the output layers. These layers are called hidden layers which are responsible for generating an internal representation of the learned material in the network. It can be shown that a multi-layered network with linear nodes can be collapsed into an equivalent two layer network and thereby lose the advantages of hidden units [13]. Although linear networks are useful in learning a set of linearly independent vectors, to implement complex, non-linear mappings, networks with non-linear nodes are required.

2) Single-layer networks, Fig 3.2b, where the input and the output layers are the same and there are no hidden layers. In this type of networks all of the processors are connected together. This, usually, is not the case in the layered networks where the only connections are between-layer connections.

There are two modes of operation in a neural network: (1) learning mode and (2) synthesis mode. In learning mode, the network is presented with the knowledge to be learned, and to accommodate the new knowledge the weight of the connections between the processors are updated incrementally according to a network specific learning algorithm. This, usually, is an iterative procedure which employs some type of numerical optimization technique. In synthesis mode, when the stored knowledge is to be extracted from a network, the weight of the connections which represent the knowledge of a network are held constant and activation levels (energy) or outputs are assigned to some of the nodes with known values. The computation takes place when the activation energy is spread throughout the network via the weighted connections. Depending on the type of network, the spread of energy happens in a single pass through the network, which is usually the case in layered networks, or it oscillates in the network till the activation level of the nodes settle down to a stable state, which is normally the case in single layered networks. These stable states could be thought of as equilibrium points or basins of attractions in a dynamic system. The location of these equilibrium points, or desired convergence states, could be chosen via the proper choice of weights between the processors. These weights are determined by the learning algorithm.

Now, we will discuss two types of networks used in this work. First, feed-forward multilayered networks that have been shown to be very efficient in learning any arbitrary mapping between a set of input and output vectors. This class of networks are also capable of generalization and feature extraction. These networks are used to extract the causal relationships between each group of related parameters (sub-processes) in an influence diagram. The second type of networks discussed will be associative memory type networks. These networks are used as content addressable memory, in which a partial input vector can be completed to a full vector based on a previously stored value. The associative memory network is used to coordinate the operation of all of the sub-processes in the knowledge extraction (synthesis) phase. We will discuss the learning algorithm and synthesis algorithm for both of these networks.

3.3. FEED-FORWARD MULTI-LAYERED NETWORKS

This class of networks have been proven to be very effective in pattern association[31]. They have been widely used in human speech and vision recognition [31,32] and in other areas such as forecasting and intelligent control [33-36]. The power of these networks come from their use of semi-linear, sigmoidal shape function processors. These networks can incorporate hidden layers that do not interact with the outside world, but rather do classification and feature extraction of the information provided to them by the input and the output layers. The structure of these networks is shown in Fig. 3.2a. The lowest layer, or the input layer, gets the information from the outside world and passes it down to the hidden layers. There can be any number of hidden layers. There are no same layer connections in these networks and the only connections are the adjacent layer connections. Because of the learning algorithm, each layer acts as classifier and feature extractor for its adjacent layers. So it can be seen how very complex information can be encoded using this type of networks. Although each layer is working as a classifier and feature extractor for its adjacent layers, these features and classifications are not necessarily distinguishable in a physical or real world sense. The reason is that the information is distributed over a large number of connections and processors so the classification of the information happens in a distributed way. The top most layer is the output layer which transmits the results to the outside world. These networks, because of their ability for feature extraction and pattern classification, are very good in dealing with the inherent noise of the information in the real world. So the hidden layers also act as a noise filter for the information provided by the outside world. This last statement is proven and dealt with in chapter 6. Feed-forward, multi-layered networks, are also known as backpropagation networks. This is because of the learning algorithm they use in associating the input and the output layers' information. Backpropagation networks can encode either static or dynamic relationships between the input and the output parameters. But care should be taken in choosing the information provided in the learning phase. The choice of input and output parameters is a critical one in how efficient a network learns the relationship between them. Consider encoding the dynamic behavior of a system such as:

$$\frac{d^n y}{dt^n} = f\left(\frac{d^{n-1}y}{dt^{n-1}}, \frac{d^{n-2}y}{dt^{n-2}}, \cdots, Constant\right)$$

Depending on the order of differential equation f, and the sampling time, the network should observe and learn to associate the past and the present behavior of y. This can be done by having a moving window in time that keeps track of a predetermined number of sampled values of y. This will result in a set of inputs that are the values of y sampled a few steps before, and a set of outputs that are the values of y at present. So what the network learns is an anticipation function which relates the behavior of y in the past to its expected behavior at present (refer to Fig. 3.4). Therefore, the network learns the function f just by looking at sampled values of y. On the other hand, if we had presented a network with time as input and values of y as output, the network could not be expected to learn the function f, because time can have an arbitrary initial value. Therefore selection of input and output parameters is a critical factor in the learning process. Next we will discuss the learning algorithm used for these networks.

3.3.1. LEARNING PROCEDURE FOR FEED-FORWARD MULTI-LAYERED NETWORKS

Although the power of non-linear networks to encode complex information was recognized by early researchers[14,37], there was no efficient learning algorithm developed for them. However, the generalized delta rule developed independently by Rumelhart[31] and Le Cun[38] solved this problem. This is a supervised learning procedure, in which examples of input and output patterns, representing the patterns to be associated, are used to train the network [12]. In this algorithm the network starts out with a random set of weights. Next, an input pattern is presented to the network and the output is calculated with the present set of weights. Then, the calculated output is compared to the desired output and the square of the difference between the two output vectors is used as a measure of error. This is then repeated for all of the input-output pairs, summing up the total error in the process. The cumulated error for all of the input-output pairs represents the distance in a Euclidean space of weights that has to be minimized.

If linear processors were used, the error surface in the weight space would look like a bowl with only one minimum (i.e. it is convex). Any gradient descent technique would lead to that solution. However, in our case, because of the non-linear processors, the error surface could have any number of local minima. Therefore, a gradient descent optimization technique does not guarantee an optimal solution. However, as pointed out by Rumelhart[31], this problem is only of theoretical interest. He has shown that by increasing the number of hidden layer nodes or the number of hidden layers, the network will converge to a good enough local minimum. These networks can encode any arbitrary relationship with a maximum of two hidden layers [25]. The generalized delta rule is very similar to the steepest descent optimization routine. In this type of training, the cumulated error is minimized in an iterative fashion. For each of the input-output pairs, the gradient of the error with respect to the weight space is calculated. This is then repeated for all of the input-output pairs. Then, one step is taken in the opposite direction of the total gradient. This whole process is repeated until the network converges to a minimum in which the measure of error has converged to zero or very close to it. Each processor has associated with it a threshold value, which shifts the sigmoid function in a positive or negative direction depending on the value of the threshold. The reason for having a threshold is to add filtering capability to each processor, so that if a processor has a high threshold, it will take a large input before that processor is in its active area, or vice versa. The value of the threshold is also determined in the same manner as calculating the weights. A threshold can be treated as the weight of a connection comming from a node with a fixed unity output. A description of the backpropagation learning scheme [31], with the notations of [12], is given below:

Let (refer to Fig. 3.3):

$t_{i,k} = \textit{threshold of the } i^{th} \textit{ node in the } k^{th} \textit{ layer}$
$w_{i,j,k} = \textit{weight between } j^{th} \textit{ node in the } (k-1)^{th} \textit{ layer to } i^{th} \textit{ node in } k^{th} \textit{ layer}$
$net_{i,k} = \textit{input to } i^{th} \textit{ node in the } k^{th} \textit{ layer}$
$O_{i,k} = \textit{output of } i^{th} \textit{ node in the } k^{th} \textit{ layer}$

The input to a processor is given as:

$$net_{i,k} = \sum_j [\, w_{i,j,k} O_{j,k-1} \,] + t_{i,k} \tag{3.2}$$

where the summation extends over all nodes in the previous layer. The output of a given processor is a sigmoid function of the input and can be expressed as:

$$O_{i,k} = \frac{1}{1+e^{-net_{i,k}}} \tag{3.3}$$

With a random starting point, in the weights and thresholds space, the measure of error is calculated for each of the input-output pairs in the following manner: in this phase the input nodes are assigned a value and the computation takes place by a forward pass of the activation level of the input layer through the network (one layer at a time), until it reaches the output layer. The computed value of the output nodes are then compared to the desired values of the output nodes. The square of the difference between the two vectors is used as the measure of error.

$$E = \tfrac{1}{2} \sum_{j=1}^{q} (d_j - O_{j,n})^2 \tag{3.4}$$

where n is the total number of layers in the network, q is the total number of output nodes, d_j is the desired output of the j^{th} node of the output layer, and $O_{j,n}$ is the actual output for the same node.

This constitutes the forward pass through the network. Next, the error is propagated backward through the network, starting at the output layer. The weights and thresholds are updated one layer at a time, such that the error is minimized. Computation of the error term with respect to each weight and threshold is accomplished using local information at each node, so that gradient calculations at each layer can be accomplished in parallel. Changes in the weight and threshold values in each iteration are calculated as suggested by Rumelhart [31]. Learning relies on minimizing E by suitable adjustments of the learning parameters, $w_{i,j,k}$, $t_{i,k}$. This requires calculation of the derivative of E with respect to the learning parameters. Using 3.2 and 3.3 the following partial derivatives can be computed:

$$\begin{aligned} \frac{\partial net_{i,k}}{\partial w_{i,j,k}} &= O_{j,k-1} \\ \frac{\partial net_{i,k}}{\partial t_{i,k}} &= 1 \\ \frac{\partial O_{i,k}}{\partial net_{i,k}} &= O_{i,k}\,(1 - O_{i,k}) \end{aligned} \tag{3.5}$$

let :

$$\frac{\partial E}{\partial net_{i,k}} = -\delta_{i,k}$$
$$\frac{\partial E}{\partial O_{i,k}} = -\phi_{i,k} \qquad (3.6)$$

The gradients of error with respect to weights and thresholds are calculated using the chain rule.

$$\frac{\partial E}{\partial w_{i,j,k}} = \left(\frac{\partial E}{\partial net_{i,k}}\right)\left(\frac{\partial net_{i,k}}{\partial w_{i,j,k}}\right) = -\delta_{i,k} O_{j,k-1}$$
$$\frac{\partial E}{\partial t_{i,k}} = \left(\frac{\partial E}{\partial net_{i,k}}\right)\left(\frac{\partial net_{i,k}}{\partial t_{i,k}}\right) = -\delta_{i,k} \qquad (3.7)$$

In 3.7, all quantities except $\delta_{i,k}$ is available from the forward pass. The quantity $\delta_{i,k}$ is calculated by propagating the error backward through the network. Consider the output layer. For this layer:

$$-\delta_{i,n} = \frac{\partial E}{\partial net_{i,n}} = \left(\frac{\partial E}{\partial O_{i,n}}\right)\left(\frac{\partial O_{i,n}}{\partial net_{i,n}}\right)$$

Using 3.4 and 3.5, this can be expressed as :

$$\delta_{i,n} = (d_i - O_{i,n})O_{i,n}(1 - O_{i,n})$$

The $\phi_{i,n}$ are calculated as follows :

$$-\phi_{i,n} = \frac{\partial E}{\partial O_{i,n}}$$

Using 3.4 and 3.5 :

$$\phi_{i,n} = (d_i - O_{i,n}) \qquad (3.8)$$

For the lower layers :

$$-\phi_{i,k} = \frac{\partial E}{\partial O_{i,k}} = \sum_j \left(\frac{\partial E}{\partial net_{j,k+1}}\right)\left(\frac{\partial net_{j,k+1}}{\partial O_{i,k}}\right)$$

where the summation extends over all nodes in the $(k+1)^{th}$ layer. Using 3.3 and 3.6, this can be simplified to yield :

$$\phi_{i,k} = \sum_j [\, \delta_{j,k+1} w_{j,i,k+1} \,] \qquad (3.9)$$

The $\delta_{i,k}$ can be determined as :

$$\delta_{i,k} = \phi_{i,k} O_{i,k} (1 - O_{i,k}) \tag{3.10}$$

or

$$\delta_{i,k} = O_{i,k} (1 - O_{i,k}) \sum_j [\delta_{j,k+1} w_{j,i,k+1}] \tag{3.11}$$

Notice that $\phi_{i,k}$ depends only on the δ in the $(k+1)^{th}$ layer, so that within a layer, the ϕ's can be computed in parallel. The gradient of error with respect to the weights and thresholds are calculated for one pair of input-output patterns at a time, keeping the learning parameters fixed. Then, a step is taken in the opposite direction of the total gradient of error with respect to the weights and thresholds [13]. This procedure is repeated until the measure of error is reduced to zero or very close to it.

3.3.2. SYNTHESIS PROCEDURE FOR FEED-FORWARD MULTI-LAYERED NETWORKS

Although a lot of work has been done in the area of learning for the feed-forward type of networks, one can not find a general purpose procedure for synthesizing these networks such that the the stored knowledge about the interrelationships between the input and the output variables can be retrieved in any random manner. This is what we have done in this section by employing a stochastic optimization technique.

Once a network has learned the mapping between the input and the output space by finding the optimum weights and thresholds, we can use this network to extract information about the interdependencies of the input and the output variables. For example, a partial input vector and a partial output vector are given and we are asked to complete both vectors based on the knowledge stored in the network. Therefore, like the learning case, we need to define a measure of error and a way to minimize this error to achieve our goal. Unlike the learning case, the weights and thresholds are fixed and represent the knowledge stored in the network. The variables here are the unspecified elements of the input and the output vectors. This problem is very important in modeling the behavior of a system. If a network learns the relationship between the input and the output space of a system, we would like to be able to use this knowledge in a reverse mode; meaning, if the output is known, we should be able to calculate the input that caused such an output. In non-linear relationships, the solution might not be unique.

In order to solve this problem, let I and O represent the input and the output vectors.

$$I = [\, i_1, i_2, \cdots, i_n \,]$$
$$O = [\, o_1, o_2, \cdots, o_m \,]$$

where n is the number of input nodes and m is the number of output nodes. Each element of these vectors is then specified with an allowable range of values confined between a minimum and a maximum value. For the fixed elements the minimum and the maximum values coincide.

$$i_k^{\max} \geq i_k \geq i_k^{\min}$$
$$o_j^{\max} \geq o_j \geq o_j^{\min}$$

Let :

$$E = \sum_{k=1}^{n} [\text{ if } i_k < i_k^{\min}] \rightarrow (i_k - i_k^{\min})^2 + [\text{ if } i_k > i_k^{\max}] \rightarrow (i_k - i_k^{\max})^2 +$$
$$\sum_{j=1}^{m} [\text{ if } o_j < o_j^{\min}] \rightarrow (o_j - o_j^{\min})^2 + [\text{ if } o_j > o_j^{\max}] \rightarrow (o_j - o_j^{\max})^2$$

The way that the error term is defined, it will penalize only the variables that are outside their allowable range.

The optimization technique chosen to minimize the error term is a stochastic method called ALOPEX [42-44] that is very similar to simulated annealing [45]. The advantages of using such a procedure are:

1) Optimizing a small number of variables. Unlike learning, where we are dealing with a large number of weights and thresholds, here we are only dealing with the input and the output nodes, which are few in numbers.

2) Avoiding local minima. In finding the solution there could be more than one possible answer. This method of optimization tries to avoid the local minima in the

process of finding the global minimum or the best possible answer.

3) Having the flexibility in defining the error term. This feature becomes very important when we need to synthesize several connected networks at the same time.

4) Lacking the need to have any knowledge about the gradients of the error with respect to the input or the output variables.

Here a brief description of the ALOPEX is given to familiarize the reader. For a detailed description of the procedure and proof of convergence the reader is referred to [42-44].

ALOPEX is a stochastic optimization procedure in which the cost function may be non-linear in a large number of variables. Local minima are effectively avoided by the introduction of noise. The ALOPEX process has the following characteristics [42]:

1) *The procedure is iterative. In every iteration all variables that determine the cost function are changed simultaneously by small increments, and the cost function is computed.*

2) *The changes in the variables depend stochastically on the change in the cost function and the change in that variable over the preceding two iterations.*

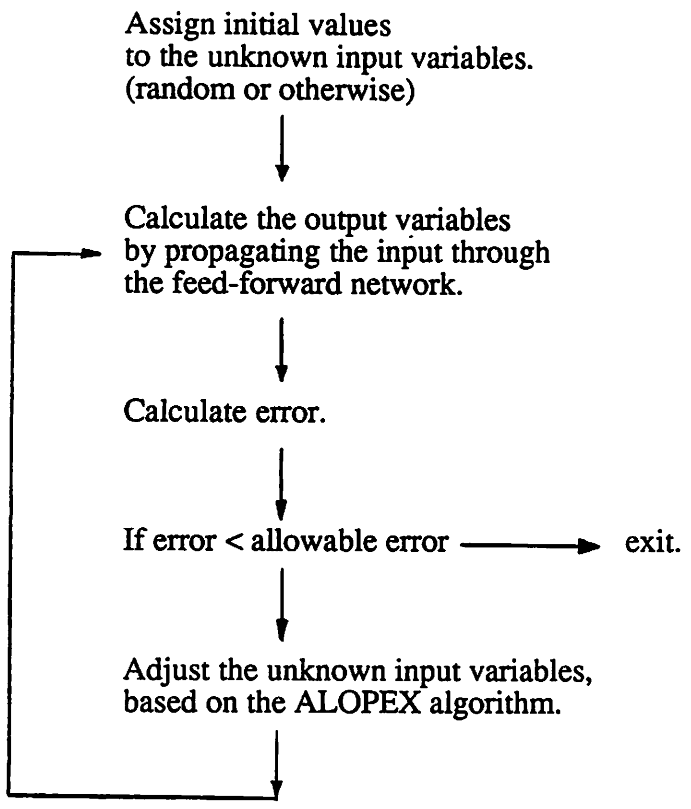
3) *All increments are retained from one iteration to the next.*

4) *The stochastic element in the process is an added noise or an effective temperature.*

5) *The process is guided by two parameters which determine the step size of the increment and the level of the random contributions.*

6) *The procedure contains additional algorithms that automatically adjust the two parameters as the run progresses.*

The flow-chart of the synthesis process is shown here:


Assign initial values
to the unknown input variables.
(random or otherwise)
Calculate the output variables
by propagating the input through
the feed-forward network.
Calculate error.
If error < allowable error
exit.
Adjust the unknown input variables,
based on the ALOPEX algorithm.

In general the relationship between the input and the output is non-linear. There could exist several possible solutions and we might be interested in one of them. The number of solutions can be reduced by reducing the possible range of the variables, or starting the synthesis with good initial values that are near the final solution.

3.4. ASSOCIATIVE MEMORY NETWORKS

In order to come up with good initial estimates for the simultaneous synthesis of back-propagation networks, we use a different type of neural network, called the associative memory network. This network is responsible to learn the common and the optimal space of the quantitative knowledge stored in the backpropagation networks. The reason for using a neural network to perform this task is that the criteria for optimality could be qualitative. The associative memory network can extract the optimality criteria by observing discrete points that satisfy them. This type of network is used as content addressable memory, in which a network can recall all elements of a previously stored vector by starting with partial information about the vector. There are different classes of this type of networks. One such example is the Hopfield network [16,17]. The Hopfield network is a dynamic single layered network. The state space of the network contains locally stable states, or attractors, that correspond to minima of an energy-like function. The basins surrounding these attractors will dominate the flow in phase space, and the system will relax to the attractors from anywhere within the basin [42]. The location of these attractors, or stored memory, can be specified by the proper choice of weights between processors. However, because of the advantages of using similar learning and synthesis algorithms, the associative memory network developed here, is a feed-forward, layered network.

3.4.1. LEARNING PROCEDURE FOR ASSOCIATIVE MEMORY NETWORKS

The network used for associative memory is a feed-forward, error-backpropagation type. The way one can use a backpropagation network as an associative memory network is by training the backpropagation network on identical sets of input and output vectors. Then, given a partial input vector (identical to the partial output vector) one can recall the rest of the elements of the input or the output vectors by using the procedure described in the synthesis section below. We have already discussed the learning algorithm for this type of networks. The only difference here is that the input and the output vectors are identical. Therefore, the network creates a mapping between two identical spaces. This mapping, however, is non-trivial, because each element of the output is related to all of the input parameters.

3.4.2. SYNTHESIS PROCEDURE FOR ASSOCIATIVE MEMORY NETWORKS

Once there is a mapping created by the learning algorithm, we can use it to complete partial vectors. The optimization process is the same as the one used for synthesis of backpropagation networks, namely, ALOPEX procedure. The only difference is in the definition of the error term. Since the input and the output vectors ought to be identical, the error term penalizes any difference between corresponding elements of the input and the output vector. It will also penalize values of the input elements that are outside their permitted range. Therefore, the error term is defined as follows:

$$E = \sum_{j=1}^{n} (i_j - o_j)^2 + [\text{ if } i_j < i_j^{\min}] \rightarrow (i_j - i_j^{\min})^2 + [\text{ if } i_j > i_j^{\max}] \rightarrow (i_j - i_j^{\max})^2$$

Once the error is minimized to zero, the input and the output vectors will be identical and correspond to a previously stored input vector or a generalization of that. As an example, let us look at the following problem:

1) Create a mapping between the coordinates of the corners of a cube, with unity volume, and themselves. Or basically store the coordinates of the corners as content addressable memory. Fig. 3.5.

2) Start the network at points near each of the corners and let it converge freely. If the network converges to the correct corners, it is working as an associative memory network.

The following table shows the learning set that the network was trained on, the testing set, and the values that the network converges to, starting at the testing set points.

Learning set			Testing set			Converged values		
x	y	z	x	y	z	x	y	z
0.0	0.0	0.0	0.2	0.2	0.2	2.0e-4	6.0e-6	3.0e-6
0.0	0.0	1.0	0.2	0.2	0.8	3.0e-5	1.0e-4	1.0
0.0	1.0	0.0	0.2	0.8	0.2	8.0e-5	1.0	6.0e-5
0.0	1.0	1.0	0.2	0.8	0.8	9.0e-5	1.0	1.0
1.0	0.0	0.0	0.8	0.2	0.2	1.0	8.0e-6	6.0e-8
1.0	0.0	1.0	0.8	0.2	0.8	1.0	4.0e-6	1.0
1.0	1.0	0.0	0.8	0.8	0.2	1.0	1.0	2.0e-8
1.0	1.0	1.0	0.8	0.8	0.8	1.0	1.0	1.0

As can be seen from the table, the converged values of the network are almost identical with the training set. In this experiment, each of the elements of the testing set are at a distance of 0.346 from their nearest corner. If we were to increase this distance and test the network, we might find that the final converged values might be different and actually a generalization of the training set. This is due to the non-linear nature of the mapping created in the learning phase. When the network creates a mapping between two identical vectors, some local minima are also created that map an unlearned vector to itself. These can be thought of as basins of attractions that are created due to the other basins of attractions generated in the learning phase. This property is unwanted only in cases where exact recall of memory is required. But for our purposes it actually is a positive quality, because the network has learned to generalize between the two spaces and can give us good initial estimates based on its knowledge for the points that are not covered in the training set. It should be noted that this behavior is also observed in other associative memory networks with non-linear nodes, such as the Hopfield network.

As an example, if we start the network at an equal distance from all of the corners, the final converged values of the network will not be one of the corners. This is shown in the table below.

Initial starting point			Final converged point		
x	y	z	x	y	z
0.50	0.50	0.50	0.54	0.58	0.48

Finally, to get an idea about the number of iterations required in the learning phase and the synthesis phase, we can look at Figs. 3.6 and 3.7. Fig. 3.6 is the error versus number of iterations in learning the coordinates of the cube. It takes about 2000 iterations of changing all of the weights and thresholds, using the backpropagation procedure, for the desired allowable error to be reached. This is a time consuming process, but it can be done off-line. Fig. 3.7 is the error versus number of iterations in synthesizing the same network, using the ALOPEX procedure, in order to retrieve the coordinates of one of the corners, starting at one of the testing points. It takes about 120 iterations, and in each iteration there are only a few parameters updated, namely, the three input nodes. The synthesis procedure is very fast and it only takes a few seconds (on a SUN 3/50) before a solution is reached.

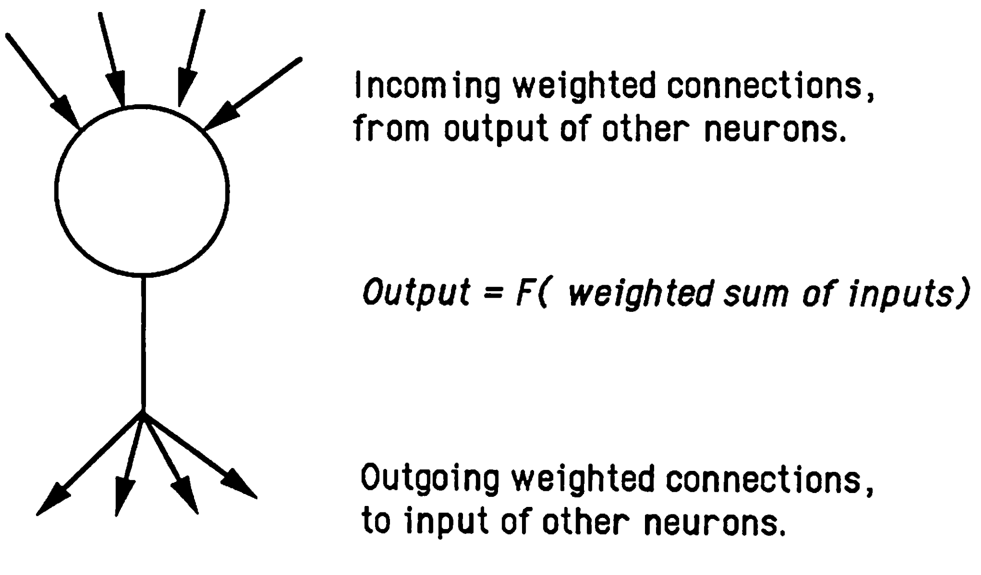

Incoming weighted connections, from output of other neurons.
Output = F(weighted sum of inputs)
Outgoing weighted connections, to input of other neurons.

Fig. 3.1a Schematic of a single neuron.

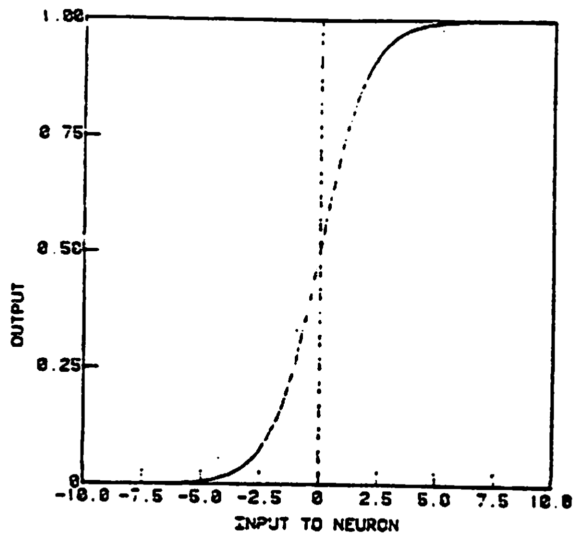


Fig. 3.1b Sigmoid function [12]

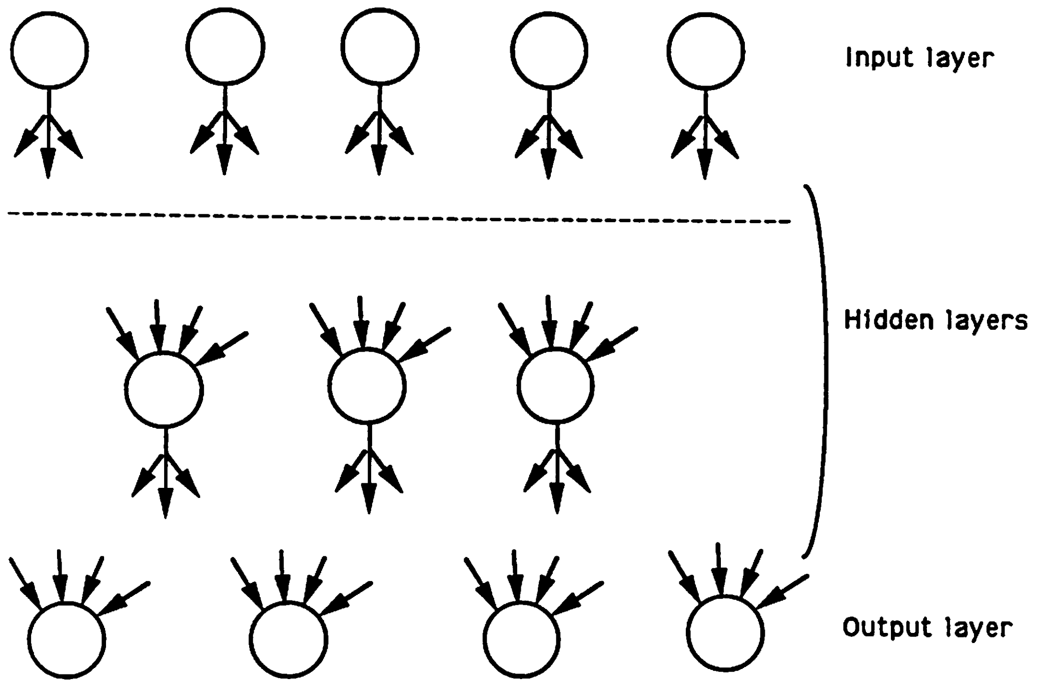


Fig. 3.2a Layered networks.

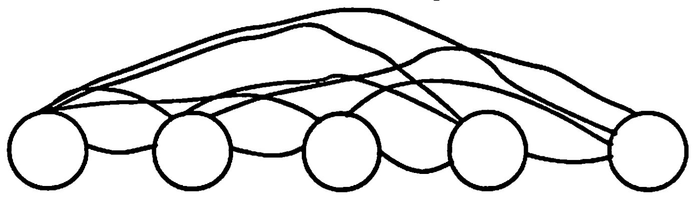


Fig. 3.2b Single layer networks.

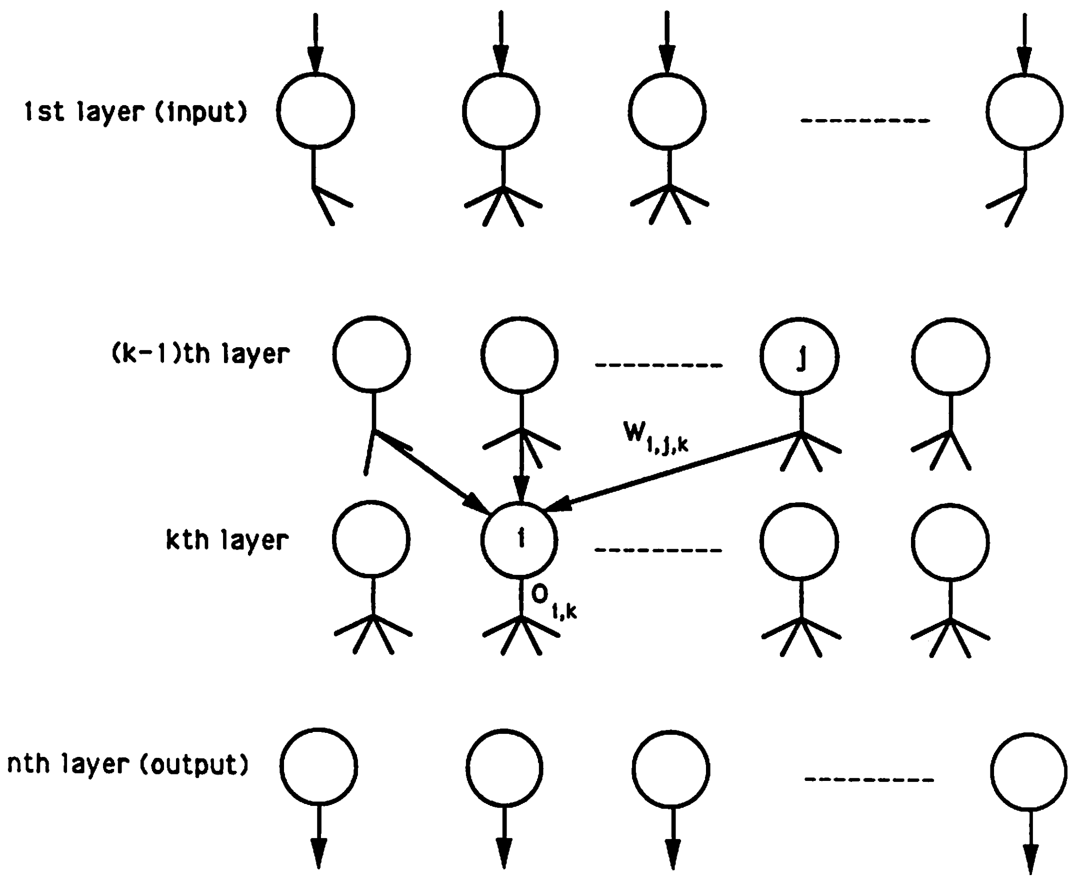


Fig. 3.3 Schematic of a feed-forward error-backpropagation network.

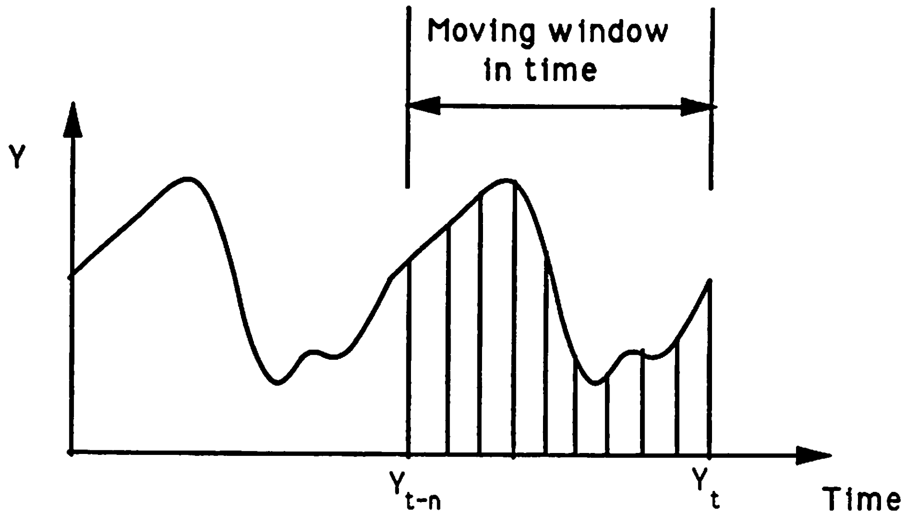


Fig. 3.4a Trace of Y versus time.

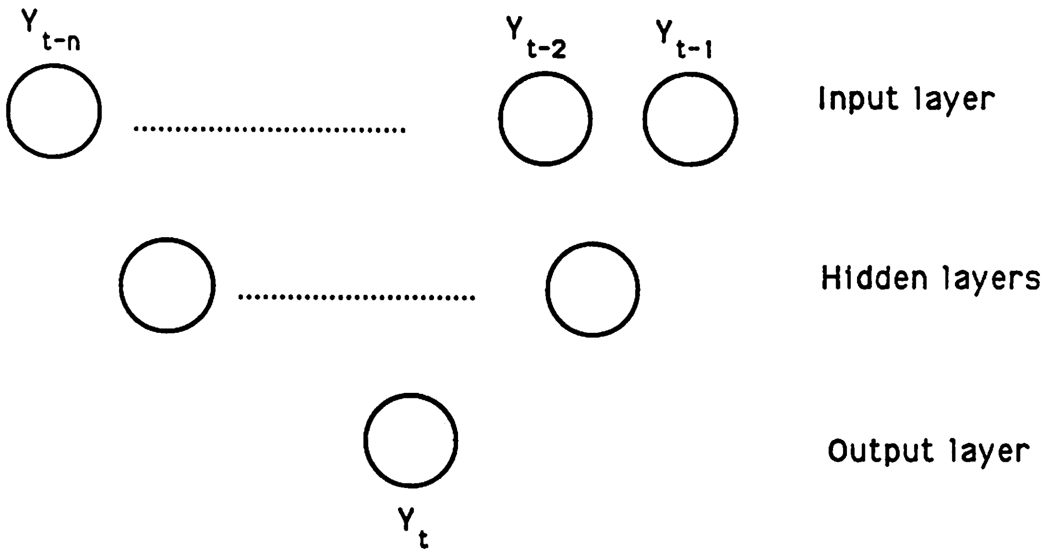


Fig. 3.4b Structure of the network capable of learning $y=f(t)$

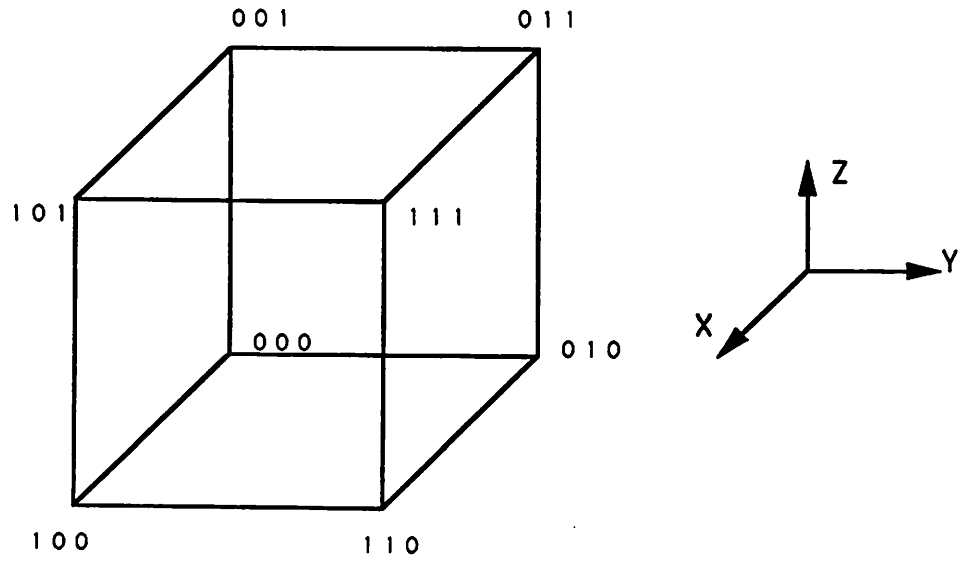


Fig. 3.5a coordinates of a cube with unity volume.

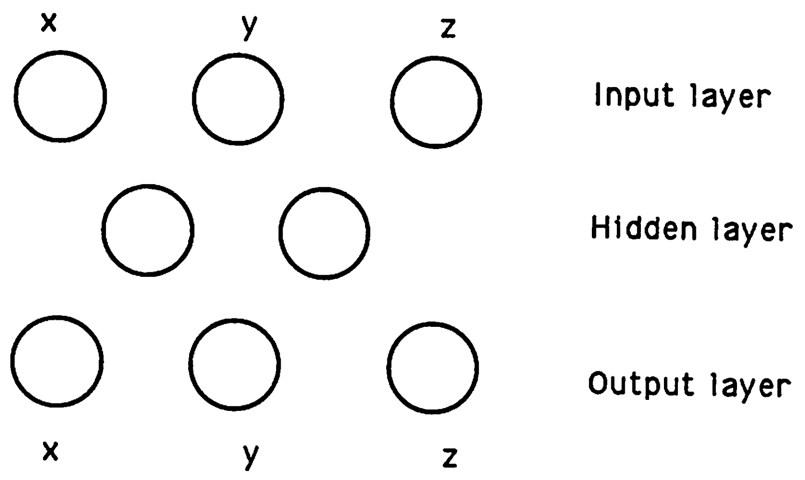


Fig. 3.5b AM network capable of learning coordinates of the cube.

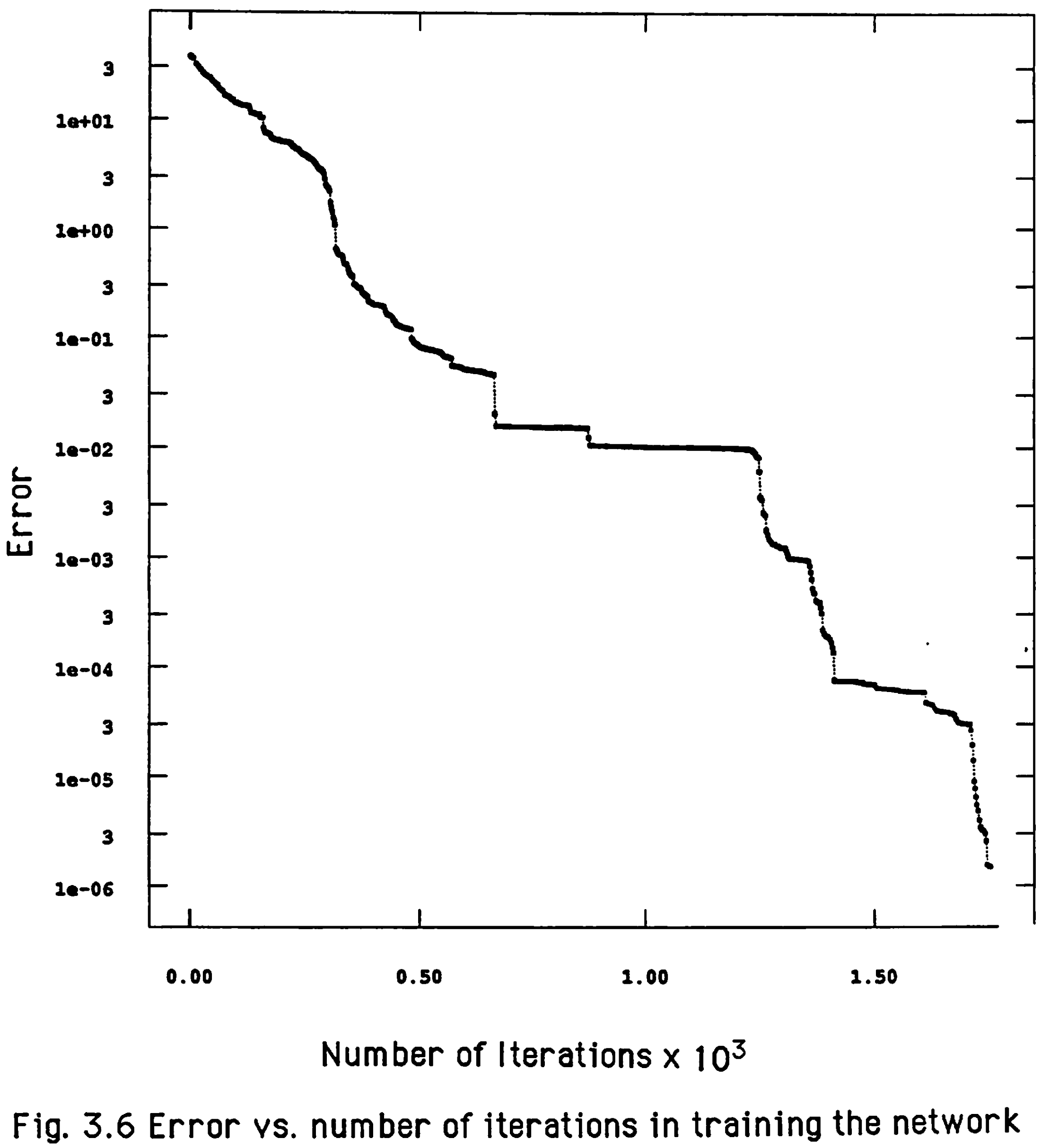


Fig. 3.6 Error vs. number of iterations in training the network for the cube problem

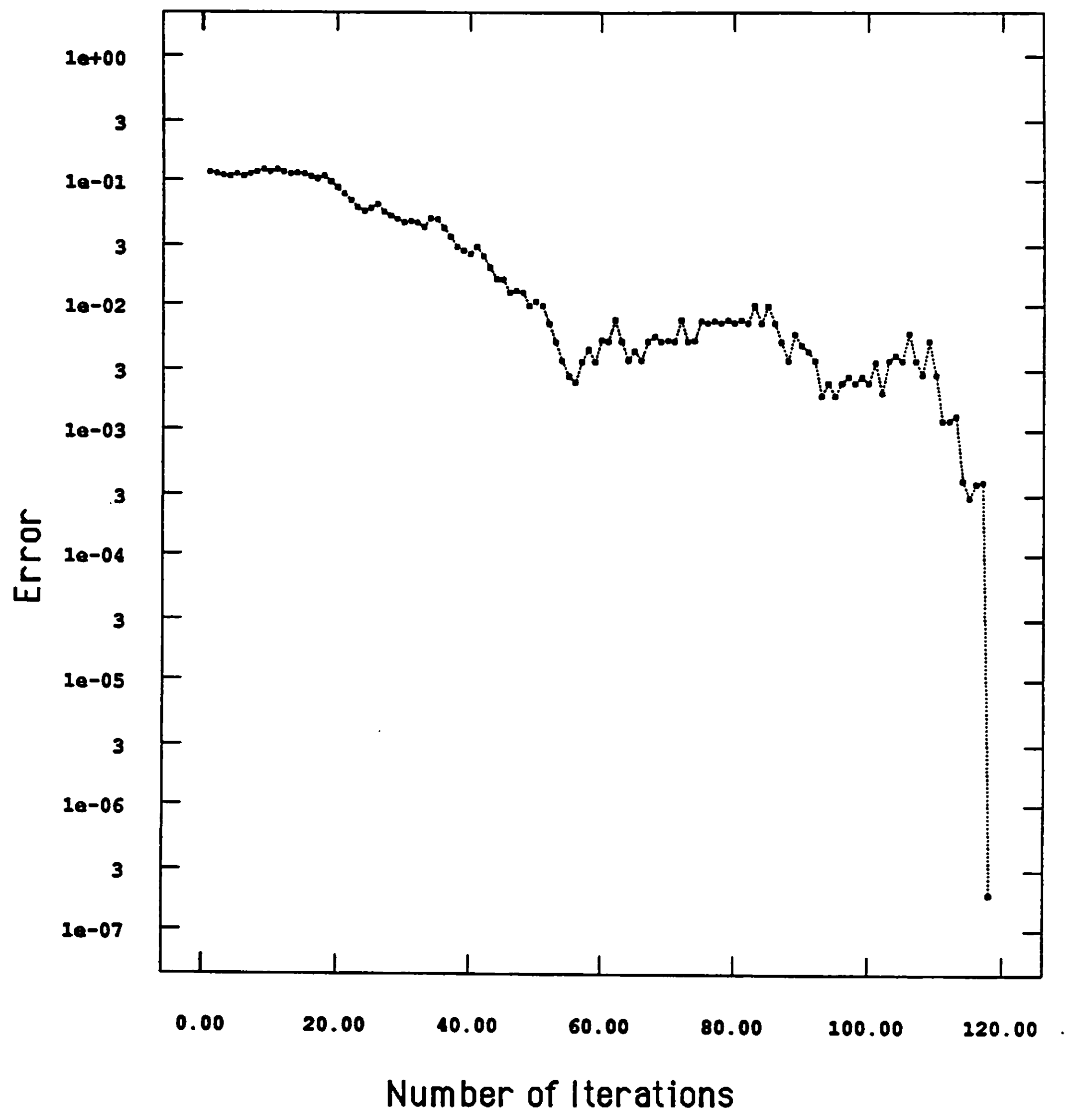


Fig. 3.7 Error vs. number of iterations in synthesizing the cube problem

CHAPTER 4
INTEGRATION OF INFLUENCE DIAGRAMS AND NEURAL NETWORKS

4.1. INTRODUCTION

In order to solve a complicated problem, such as modeling a complex manufacturing process, one efficient approach would be to break down the problem into several smaller independent problems, solve them separately, and integrate them back together. This is the approach chosen in our method of modeling.

Influence diagrams are used to decompose the problem by establishing conditional independence between process parameters. For example, in modeling dry-oxidation oxidation of silicon, Fig. 4.1, instead of trying to come up with one model that relates the three process outputs (oxide thickness, stress, surface defects) to the seven process inputs (rest of the parameters), we will have three separate models, each with only one output and only as many inputs that affect that particular output. One model, for example, will relate oxide thickness to cleaning type, process time, process temperature, and crystal orientation.

In this chapter we will explain how neural networks are used to learn the quantitative relationships in each of the sub-processes, and how they are used to extract the stored knowledge in them, to solve problems such as recipe generation for a desired process output.

4.2. ARCHITECTURE OF THE INTEGRATED NETWORK

First, the topology of the process is laid out by the use of influence diagrams. Next, two types of neural networks are employed for proper operation of the integrated network, Fig. 4.2. There will be a Feed-Forward Error-BackPropagation (FFEBP) type of network for each output node that does not have an exact analytic relationship with the nodes that affect it. For example, there are three of these networks employed in Fig. 4.2. Next, there will be one Associative Memory (AM) type network relating all of the nodes together. The AM network is responsible for re-integrating the sub-processes into a process by learning the common and optimal operating space of all of the FFEBP networks so that it can be used to generate good initial estimates for the unknown variables in the simultaneous synthesis of the FFEBP networks. As mentioned earlier, the criteria for optimality could be qualitative. The AM network can extract the optimality criteria by observing discrete points that satisfy them. The integrated networks operate in two modes: learning, and synthesis, described next.

4.2.1. LEARNING

This is the knowledge acquisition phase of the modeling. Each of the FFEBP networks will learn over a vector of input-output pairs. This vector is a sub-set of points out of the whole possible behavioral space of the process. These points are the possible operating points that are acceptable in terms of their output. In this phase each of the FFEBP networks will extract a relationship, relating their inputs to their output. At the same time the single AM network, which looks at all of the parameters, learns over a subset of points learned by all of the FFEBP networks. These are optimal points of operation for the manufacturing process. The measure of optimality could be quantitative or qualitative. Therefore, FFEBP networks learn over the **possible** operating points of the process and AM network learns over the **optimal** operating points of the process.

A two dimensional example is given here to clarify the subject of learning. If we are looking at two parameters only, namely, A and B in Fig. 4.3a, the influence diagram is a simple arc from A to B, given A is input and B is output. Let Fig. 4.3b describe the general relationship between A and B. And let Fig. 4.3c represent the sampled, acceptable behavior of A and B, in our case, all B's above B_t. Finally, let us assume that we would like to operate where B is less sensitive to changes in A, as shown in Fig. 4.3d. Fig. 4.3c represents the points that the FFEBP network is trained on, and Fig. 4.3d, a subset of points in Fig. 4.3c, represents the points that AM network is trained on. The reason for doing so will become clear in the synthesis part.

The knowledge stored in neural networks should be updated periodically, otherwise there will be no adaptation in the model to changes in the process. The frequency of updating the knowledge is totally process dependent. In general, the knowledge of the FFEBP networks should be up-to-date, but the knowledge of the AM network need not be updated as often, because the AM network acts like a reference point for the FFEBP networks. In other words, if the process degrades gradually, and the FFEBP networks keep up with this knowledge, the AM network will act as an alarm for a corrective action to take place.

4.2.2. SYNTHESIS

Synthesis, or knowledge extraction, is where, given information about part of the parameters, the networks will generate the value of the unknown parameters according to the optimal behavior, stored in the AM network, and the possible behavior, stored in the FFEBP networks. Since there could exist a number of possible solutions for the given constraints, we would like to choose the best possible solution amongst them. For example, if we would like to find a value of A such that $B \geq B_t$, Fig. 4.3b, we can see that there are two possible regions satisfying this constraint. But according to the knowledge stored in the AM network, Fig. 4.3d, $A_2 \geq A \geq A_1$ is the preferred region, because B is less sensitive to changes in A. Therefore, the role of the AM network is to generate the best initial values, given a set of constraints, for the answers provided by the synthesis of FFEBP networks. The operation in the synthesis phase is summarized in Fig. 4.4.

As the number of FFEBP networks increases, the importance of the AM network in providing good initial estimates increases. This is because we would like to find a region of the space of the possible behavior of all of the networks that satisfy our constraints. As the number of FFEBP networks increases, the number of local minima also increases.

These local minima are the regions where only some of the constraints are met satisfactorily.

On the other hand, one might ask "Why are FFEBP networks needed?". To answer this question, we have to remember that the knowledge stored in the FFEBP networks is broader in its scope and covers more of the possible behavioral space of the process than the knowledge stored in the AM network, which covers a sub-space of the FFEBPs' knowledge. Secondly, FFEBP networks extract natural and causal relationships existing between the input and the output vectors, versus the AM network, which creates a mapping between two identical spaces. As shown before, if one deviates too far from the stored memory in the AM network, one might end up in a local minimum (a position that does not match the stored memory, but rather, is a generalization of it). One abstraction would be : AM network provides the ideal operating point, and FFEBP networks provide one possible match between the ideal and reality. Another function that can be provided by FFEBP networks is diagnostics. Since the global knowledge about the process is broken down and stored in several FFEBP networks, the knowledge stored in each of these networks is local information about part of the parameters. Therefore, one can, by varying some of the parameters independently, study their effects on the remaining parameters.

4.2.3. CALCULATION PROCEDURE FOR SYNTHESIS

The synthesis is done in two parts: (1) synthesis of the AM network, and (2) simultaneous synthesis of FFEBP networks. Since there is only one AM network, and its synthesis has been discussed in the previous chapter, we will only describe what a user needs to define in this phase. A user needs to specify an allowable range for each of the nodes in the influence diagram. Some of these nodes might also be rigidly specified; i.e., no variations allowed. Next, an error check is done to make sure that none of the FFEBP networks are over-constrained; this is where all of the input and output nodes of a network are rigidly specified. An FFEBP network must have at least one node that is allowed to change. If this single node is an output node, then its value is determined by a feed-forward pass of the FFEBP network; otherwise, its value will be determined iteratively by the optimization procedure. Once it is determined that none of the networks are over-constrained, the synthesis of the AM network will result in the generation of initial values for nodes that are not rigidly specified. If there are no rigidly specified nodes to start with, then a random number generator is used to produce allowable values for each of the nodes. These values are in turn passed to the AM network to produce initial values that are in agreement with the optimality constraint stored in the AM network.

For the simultaneous synthesis of FFEBP networks, we need to define two terms: dependent and independent nodes. Independent nodes in an influence diagram are the nodes that are not directly influenced by any other nodes (i.e. have no arcs going into them). Input nodes are independent nodes. Dependent nodes, on the other hand, are nodes that receive influences from other nodes (incoming arcs), and that might also influence other nodes (outgoing arcs, refer to Fig. 4.5). In the synthesis phase, independent nodes are the only free parameters that are updated in each iteration of the optimization procedure. All of the nodes, however, are considered in evaluating the error term. As defined in the section on the synthesis of FFEBP networks (chapter 3), the error term will only penalize the nodes that have drifted outside their allowable range. Therefore, the error term is defined as follows:

$$E = \sum_{i=1}^{n} [\text{ if } x < x_{\min}] \rightarrow (x - x_{\min})^2 + [\text{ if } x > x_{\max}] \rightarrow (x - x_{\max})^2$$

$n = \textit{total number of nodes in the influence diagram.}$

$x = \textit{value of each node, input or output.}$

Not all of the FFEBP networks need to be involved in the synthesis procedure if we are only concerned about the behavior of some of the parameters. But all of the FFEBP networks connected to these parameters should be included in the synthesis procedure.

Again, only the independent nodes connected to the same networks will be updated in each iteration of the optimization procedure.

As seen here, the synthesis procedure is very flexible and can be used to extract information about the relationships of all or some of the parameters in the influence diagram. The architecture used here easily lends itself to building an interactive environment where the user can gain detailed knowledge about the process by seeing the interactions of the nodes in the influence diagram.

4.3. SIGNIFICANCE OF THE INFLUENCE DIAGRAMS

In order to reply to an important question: "Why not use only one error-backpropagation network to learn the input-output relationship?", we have to justify the use of influence diagrams, which break down single multi-input multi-output backpropagation networks into several smaller multi-input single-output ones, in the following way:

1) There are variables that are conditionally independent of each other (e.g. deposition rate is conditionally independent of time). Although neural networks are themselves capable of extracting conditional independencies between the input and output variables, influence diagrams significantly reduce the amount of data required to create accurate models for the processes. This is due to the fact that a network would need **enough** data to extract the conditional independencies by itself.

2) The process of training the neural networks gets decoupled by making use of the influence diagrams. Decoupling of the training procedure makes it possible for each network to cover a different region and/or size of the behavioral space of the process. This is very important in the process of modeling, because of the adaptability it gives the model. If new variables are added to a model, or a sudden shift is observed in some of the existing variables, we do not need to produce a lot of new experimental data to re-train the one big neural network. The only additional training sets required are the ones for the new, or existing, neural networks that are

connected to the new, or the shifted, variables.

3) The process of synthesizing a model also gets decoupled by making use of the influence diagrams. This is due to the fact that single-output networks make it possible to synthesize each output variable separately. This point is significant in modeling diagnostic types of problems.

4) There are output variables that have exact relationships with the input variables that affect them. There is no need to use neural networks for extracting these known relationships.

The points mentioned above, besides the very important one that influence diagrams are an effective way of capturing and transferring qualitative knowledge of human experts, are some of the reasons for the use of influence diagrams.

4.4. FURTHER EXPLANATION OF STRUCTURAL AND NUMERICAL LEVELS OF INFLUENCE DIAGRAMS

Since the reader is now familiar with neural networks and their operation, the structural and numerical levels of the influence diagram, which are handled by neural networks, are explained here.

4.4.1. STRUCTURAL LEVEL

As pointed out before, the physical structure of neural networks are determined at this level. The number of input and output nodes of each network is already determined by the variables of the influence diagram; however, the number of hidden layers and the number of nodes in each hidden layer are not predetermined. The choices for these numbers are totally dependent on the complexity of the relationship to be learned by each network. The algorithm used for determining these numbers is an iterative one which usually converges within a few iterations. The procedure is described in Fig. 4.6. The basic goal is to find a structure that drives the measure of error in the learning algorithm to zero or very close to it. The nature of the algorithm is such that it tries to find a network with the minimum number of hidden layers and the minimum number of nodes in each of these layers. The maximum number of hidden layers allowed is two, because these networks can encode any arbitrary relationship with a maximum of two hidden layers [25]. In our experimental results, none of the networks needed more than one hidden layer. Therefore, the algorithm generally needs to find the number of nodes in one hidden layer by starting with one node and increasing it as necessary. Since the error surface in the learning algorithm has more than one minimum, the learning algorithm starts out at a few different random initial points in the weights' space and tries to drive the measure of error to zero, before incrementing the number of hidden layer nodes. A very useful feature of non-linear networks is that they are not very sensitive to the structure. Therefore, as long as the measure of error in the learning algorithm is driven to zero, it does not matter much if the task could have been accomplished with fewer nodes in the hidden layer.

4.4.2. NUMERICAL LEVEL

This is the level where the relations between dependent nodes are quantified by employing two optimization techniques.

In the learning, or knowledge acquisition, phase a gradient optimization technique is used to drive the measure of error to zero by adjusting the weights and thresholds of the networks. This phase is directly related to the structural level where, given a structure produced by the structural level, the optimization routine tries to find weights and thresholds that minimize the error to zero. If not possible, the structural level is flagged to update the structure and the numerical level tries again. This is not usually repeated for more than a few times before the measure of error is successfully driven to zero.

In the synthesis, or knowledge extraction, phase a stochastic optimization technique is used to drive the measure of error to zero, as defined in the synthesis section. Here the structure of the networks are fixed and this part of the numerical level operates independently of the structural level.

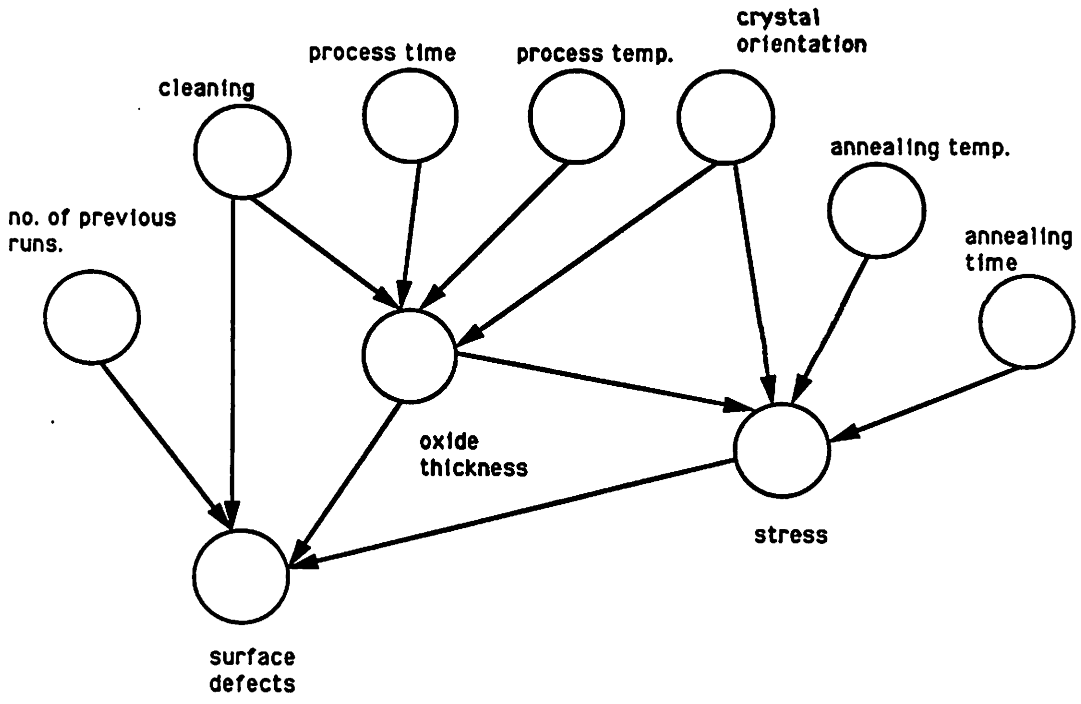


Fig. 4.1 Influence diagram for dry-oxidation of silicon.

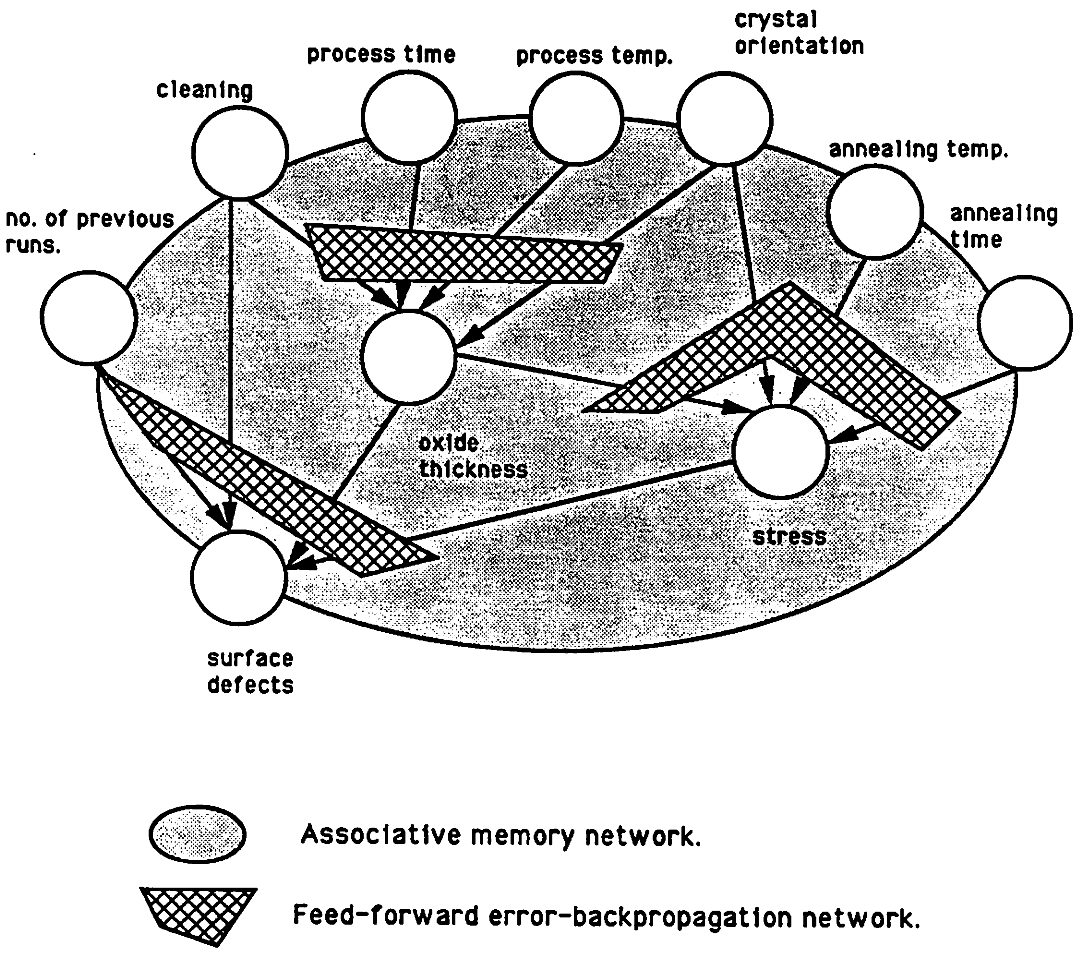


Fig. 4.2 Integrated network for dry-oxidation of silicon.

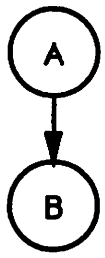


Fig. 4.3a Influence diagram for the two variables A and B.

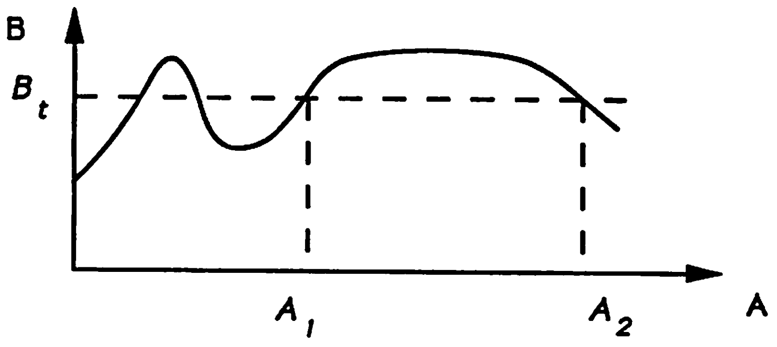


Fig. 4.3b General relationship between A and B.

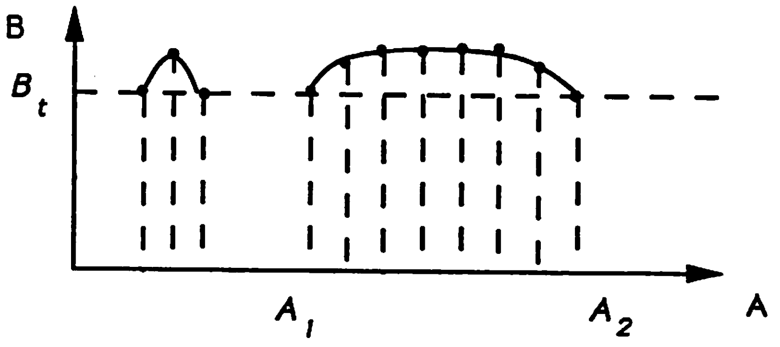


Fig. 4.3c Sampled acceptable regions of the relationship.

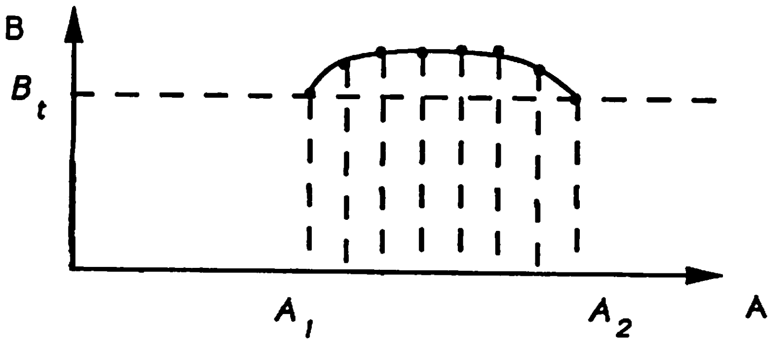


Fig. 4.3d Sampled optimal regions of the relationship.

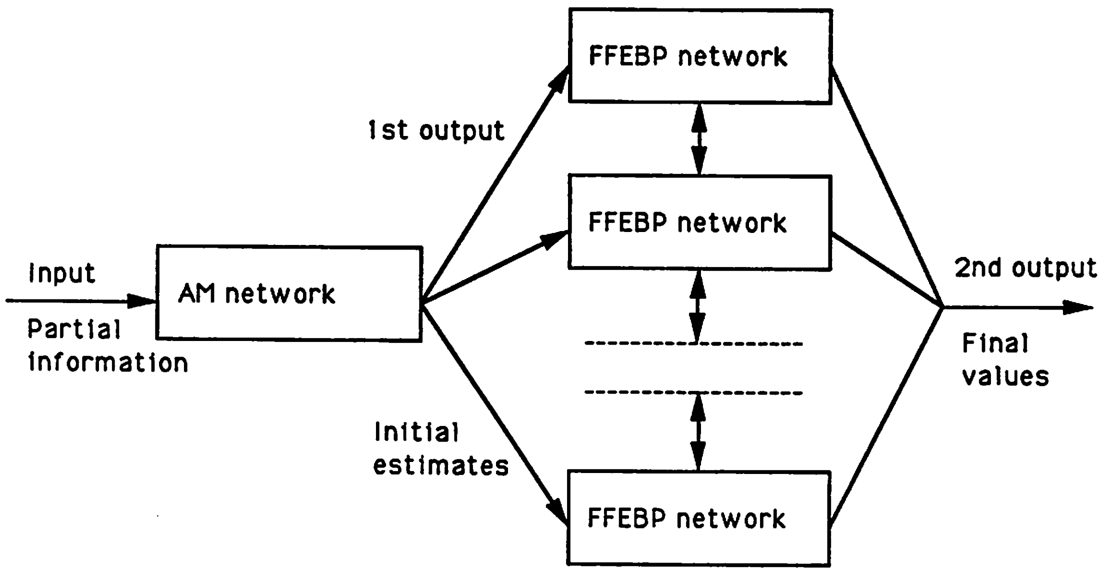
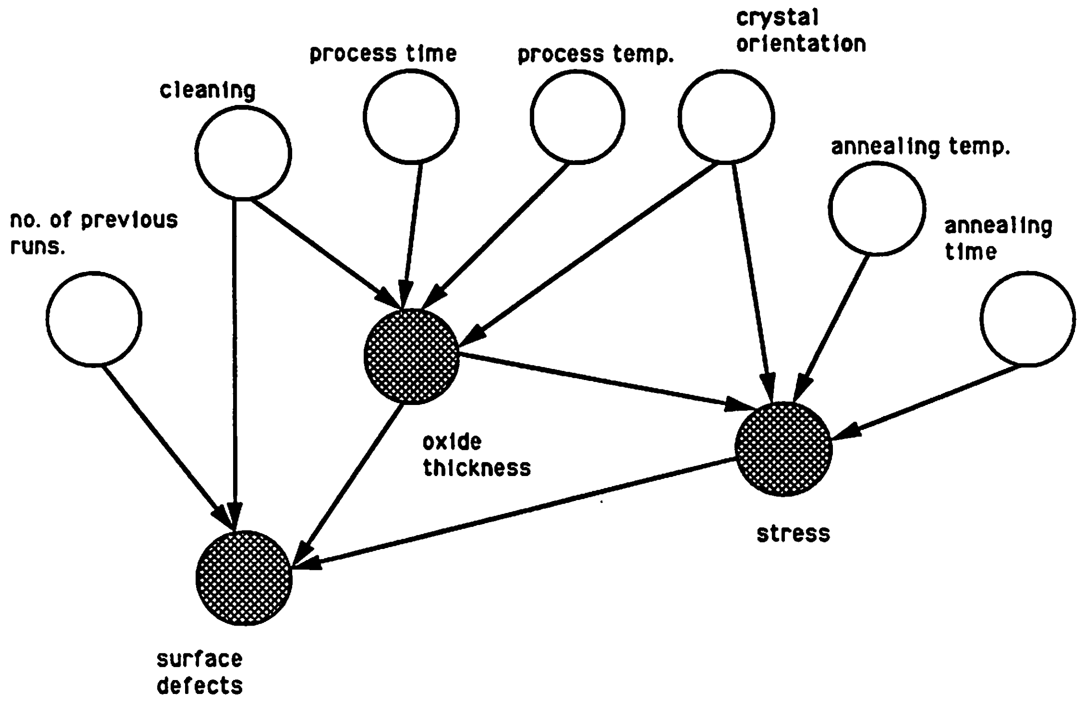


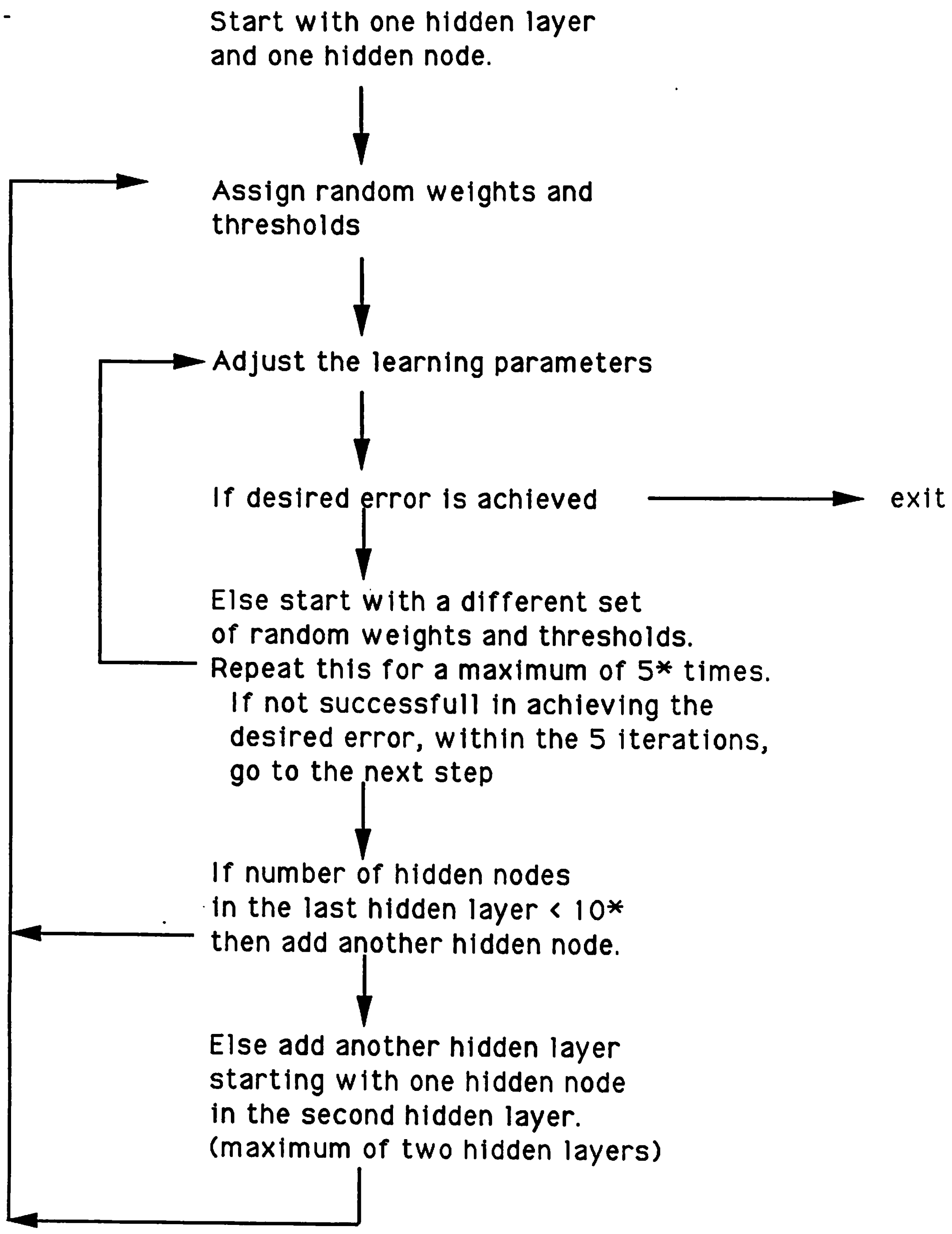
Fig. 4.4 Schematic of the synthesis process



● Dependent nodes.

○ Independent nodes.

Fig. 4.5 Influence diagram for dry-oxidation of silicon.


Start with one hidden layer and one hidden node.
Assign random weights and thresholds
Adjust the learning parameters
If desired error is achieved
exit
Else start with a different set of random weights and thresholds. Repeat this for a maximum of 5* times. If not successfull in achieving the desired error, within the 5 iterations, go to the next step
If number of hidden nodes in the last hidden layer < 10* then add another hidden node.
Else add another hidden layer starting with one hidden node in the second hidden layer. (maximum of two hidden layers)

* These emperical numbers are problem domain dependent.

Fig. 4.6 Flow chart for the structural algorithm.

CHAPTER 5
EXPERIMENTAL RESULTS

5.1. INTRODUCTION

So far we have introduced the reader to influence diagrams and neural networks in chapters 2 and 3, and their integration in chapter 4. In this chapter the experimental results of our work are presented. The techniques developed in chapters 3 and 4 have been applied to modeling two VLSI manufacturing processes. The first process is dry-oxidation of silicon, and the second one is Low Pressure Chemical Vapor Deposition (LPCVD) of polysilicon. There are three parts for modeling each of these processes. In the first part, a brief introduction is given about each process, defining the process parameters and describing the present state of other existing process models. In the second part, called learning, the integrated network of influence diagram and neural networks is set up for the process. The integrated network is then trained on part of the experimental data, and is tested on the remaining data. This test is done to see how by looking at some of the data, the integrated network can predict the behavior of the rest of the data, or how well the integrated network has learned about the behavior of the manufacturing process. For this part, the performance of the integrated network, in modeling LPCVD of polysilicon, is compared to the performance of other models, which are generated by the use of first principles and/or statistical regression techniques, employing the same experimental data. In the third part, which is the synthesis section, the trained integrated network of each process is used to generate a recipe for a desired process output. The integrated networks are also used to generate novel ideas about the process. Novel ideas are process set-ups that the integrated network was not specifically trained on, but could generate them because of its ability to generalize the relationships developed in the learning phase.

5.2. DRY-OXIDATION OF SILICON [51]

The thermal oxidation of silicon has been an integral part of semiconductor device fabrication since the 1950s when thermal oxides were used to selectively mask dopants during the preparation of diffused transistors [53].

In the silicon thermal oxidation process, silicon reacts with either oxygen or water at elevated temperatures (700-1250 $°C$) to form silicon dioxide. The oxidation reaction may be represented by either of the following two reactions:

$$Si+O_2 \rightarrow SiO_2 \tag{1}$$
$$Si+2H_2O \rightarrow SiO_2+2H_2 \tag{2}$$

It has been shown by special marker experiments [54] that the oxidation species, either oxygen or water, diffuses through the oxide already formed to react with silicon at the $Si-SiO_2$ interface. As oxidation proceeds, the interface continues to move into the silicon, thus producing a new clean surface. From the densities and molecular weights of silicon and silicon dioxide, it can be shown that, for every thickness X of oxide formed, 0.45 X of silicon is consumed [51].

Thermal oxidation is normally carried out in a fused quartz tube in a resistance heated furnace. A schematic drawing of a typical oxidation system is shown in Fig. 5.1a. For dry O_2 oxidation, high purity oxygen is transported into the furnace tube through suitable regulators, valves, traps, filters, and flow-meters [51]. Silicon oxidation data are generally obtained by determining oxide thickness as a function of oxidation time, crystallographic orientation of silicon, and temperature.

5.2.1. OXIDATION KINETICS [51]

As indicated earlier, silicon thermal oxidation proceeds by the diffusion of the oxidizing species through the oxide already formed. This process is indicated in Fig. 5.1b. It has been proposed [54] that three consecutive reactions occur during thermal oxidation whose fluxes are equal under steady-state conditions. These are indicated in Fig. 5.1b as (a) transfer of the oxidizing species from the gaseous phase into the outer portion of the oxide, (b) diffusion of some oxidizing species through the oxide to the silicon, and (c) reaction of the oxidizing species with silicon at the $Si-SiO_2$ interface to form SiO_2. By representing each of the three reactions by appropriate flux equations and by equating these fluxes, a general oxidation relationship has been derived [52]:

$$X_0^2 + AX_0 = B\ (t+\tau) \tag{3a}$$

also written in the form

$$\frac{X_0^2 - X_i^2}{B} + \frac{X_0 - X_i}{B/A} = t \tag{3b}$$

where X_0 = oxide thickness, t = oxidation time, and A, B, τ, and X_i are constants as defined below:

$$A = 2D_{eff}(1/k + 1/h)$$
$$B = 2D_{eff}C^*/N_1$$
$$\tau = (X_i^2 + AX_i)/B$$

where D_{eff} = effective oxidant diffusion constant in the oxide; k, h = rate constants at the $Si-SiO_2$ and gas-oxide interfaces; C^* = equilibrium concentration of the oxide species in oxide; N_1= number of oxidant molecules in the oxide unit volume; and X_i = oxide thickness at the start of oxidation.

Two limiting forms of Eq. [3] can be noted. At large oxidation "times," i.e. $t \gg A^2/4B$ and $t \gg \tau$,

$$X_0^2 \simeq Bt$$

This equation represents a parabolic oxidation and B is the parabolic rate constant.

For small oxidation "times," $t \ll A^2/4B$ a linear oxidation expression is obtained:

$$X_0 = B/A\,(t+\tau)$$

B/A is the linear rate constant.

Special mention should be made of the correction factor τ which is related to initial oxide thickness X_i . It has been noted that for oxidation in dry O_2, an initial thickness region does appear to be satisfied by the general relationship Eq. [3]. The plot of oxide thickness versus oxidation time tends to extrapolate through the thickness axis at about X_0=150Å.

The thermal oxidation of silicon can be represented by Eq. [3] for a wide range of temperatures, oxide thicknesses, orientations, and oxidation ambients, provided the dependence of the rate constants B and B/A as a function of these variables is known.

5.2.2. INTEGRATED NETWORK

In the previous section an oxidation model based on first principles was derived by Bruce Deal [51,52]. The constants in the model are determined by experimental procedures. This model, however, looks at only three input parameters, that are time, temperature, and crystallographic orientation of the silicon substrate, and relates them to the resultant oxide thickness. But based on experimental data produced by Dr. Dah-Bin Kao at National Semiconductor Company (Fig. 5.2), the cleaning process of the substrate also does affect the oxide thickness. It should also be noted that the behavior of the process is different for thin and thick oxide films. The oxidation model suggested in the previous section works better for thick oxide films and does not predict the behavior of the process closely for thin oxide films; thickness < 150 Å.

Therefore, we have two objectives here. The first is to have a unified model that works for thin and thick oxide films, the second is to include cleaning type as one of the process parameters.

The influence diagram of the process is shown in Fig. 5.3a. The complexity of the model is directly related to the number of output parameters that we are interested in. In this case there is only one output parameter: oxide thickness. The model though, can be extended if other output parameters become of interest and experimental data are provided for them. Such output variables could be stress or surface defects of the resultant oxide film.

The integrated network for this influence diagram is shown in Fig. 5.3b. There is one FFEBP network and one AM network in the integrated network.

5.2.3. LEARNING

In this part, to train the two networks, two learning sets are chosen from the experimental data in Fig. 5.2. In order to test the learnability of the FFEBP network, one part of the experimental data is chosen for training and the rest is used for testing. There are a total of 70 data points provided. 44 of these are used for training and 26 for testing. The data points for the testing set are chosen such that almost every other data point in each temperature range belongs to the testing set. In other words, the space of the training set covers the space of the testing set completely, because a network can not be expected to effectively generalize its knowledge outside the region of its learning space.

The learning set chosen for training the AM network is a subset of the 44 points chosen for training the FFEBP network. We have arbitrarily chosen the criteria of optimality for training the AM network, to be:

$$2000 \; minutes > process \;\; time > 50 \; minutes$$

There are 30 points in this region that qualify.

The range and type of each variable is given below:

Variable	Range	Type
Temperature	800 - 1200 °*C*	continuous
Time	1.5 - 30000 minutes	continuous
Orientation	(001), (111)	binary
Cleaning type	+HF, -HF	binary
Oxide thickness	50 - 10000 Å	continuous

As can be seen above, three of the parameters are continuous and two of them are binary. One of the advantages of using backpropagation as the training algorithm for both types of networks is the possibility of having a mix of discrete and continuous parameters in modeling a process.

The input parameters are normalized between zero and one, and the output parameter is normalized between 0.3 to 0.7. The input parameters need not be normalized. The sigmoid function maps minus infinity and plus infinity into zero and one, respectively. The normalization of the input is performed so that we do not saturate the sigmoid, and therefore reduce the sensitivity of the networks to small variations in the input. In terms of the output, we are constrained to values between zero and one, since this is the only possible output range of a sigmoid function. But again, in order to increase the sensitivity of the networks, the output parameters are normalized such that they fall into the linear range of the sigmoid's output, away from its saturation regions. One other modification of the data points is the use of the natural log of the variables instead of their actual values. This has been shown to yield better results in creating a mapping for variables with a wide dynamic range. This last point is found to be true experimentally, by mapping known linear and non-linear relations with wide dynamic ranges of parameters. Pre-processing of the raw data is shown in Fig. 5.4.

The network structures capable of successfully creating good mappings were 4-4-1 for the FFEBP network and 5-9-5 for the AM network. Both networks have three layers. The first number corresponds to the number of nodes in the first, or input layer, the second number corresponds to the number of hidden layer nodes, and the last number is the number of nodes in the output layer.

The appropriate structure of each network is found iteratively using the structural algorithm described in the previous chapter.

After both networks were trained on their respective training set, the FFEBP network was tested on its testing set. The result is shown in Fig. 5.5. The graph in Fig. 5.5 represents actual oxide thickness versus predicted oxide thickness for the different experimental conditions in the testing set. If the predicted and actual values of the oxide thickness were the same, the 26 points that the FFEBP network was not trained on would lie on a 45 degree line, which is almost the case in Fig. 5.5. The maximum error in predicting the oxide thickness is 14.6%, and the average error is 6.5%.

The conclusion to be drawn from this experiment is that an FFEBP network can learn the quantitative relationship between process parameters just by observing discrete experimental data points (that are inherently polluted with some statistical noise), without having any knowledge about the physics of the process.

5.2.4. SYNTHESIS

In this part we would like to extract information about the process using the knowledge stored in the networks. The most common form of required knowledge is: given a desired output parameter, in this case oxide thickness, and maybe part of the known input parameters (i.e. cleaning type and substrate crystallographic orientation), find the rest of the input parameters (which are process time and process temperature), such that all of the parameters are in agreement with the optimality criteria stored in the AM network. It should be noted that the use of the AM network is optional and its only function is to limit the search space to specified areas of the knowledge stored in the FFEBP networks.

The process of synthesis has already been explained before, so we will present the results here, summarized in table 5.1. The first row of table 5.1 represents the input or the known parameters. These values are then passed to the AM network. The second row represents the output of the AM network, which is to provide good initial estimates for the FFEBP network. As can be seen here, the process temperature chosen by the AM network is 800 $^{\circ}C$ and the process time is 242 minutes. These values are in agreement with our optimality criteria of 2000 minutes > process time > 50 minutes. If a different temperature was chosen, i.e. 1000 °C, then the process time that would yield the desired oxide thickness would fall outside of the allowable range specified by the optimality criteria. The third row of table 5.1 represents the output of the FFEBP network. These are the values that will be used for actual process set-up. If we compare the values in row 3 to those of row 4, which is an actual experimental data point, it is evident that the answers provided by the integrated network are experimentally verified, and are also in agreement with the optimality criteria.

The results provided in this part show how the integrated network can be used to extract information about a process in any random manner, and specifically its capability to generate process recipes for desired process output values.

5.3. LPCVD OF POLYSILICON [50]

Polysilicon is generally deposited by the pyrolysis (i.e. thermal decomposition) of silane (SiH_4) in the temperature range 580-650 °C. The main technique used to deposit polysilicon is LPCVD because of its uniformity, purity, and economy. The deposition reaction sequence is:

$$SiH_4 + surface\ site \rightarrow SiH_4(adsorbed)$$
$$SiH_4(adsorbed) \rightarrow SiH_2(adsorbed) + H_2(gas)$$
$$SiH_2(adatom) \rightarrow Si(solid) + H_2(gas)$$

where the adsorption of the SiH_4 is followed by decomposition to an intermediate compound, SiH_2. Upon evolution of the remaining hydrogen, the solid film forms. The overall reaction is generally given as

$$SiH_4(vapor) \rightarrow Si(solid) + 2H_2(gas)$$

Horizontal tube, hot wall reactors are the most widely used LPCVD reactors in VLSI processing. They are employed for depositing polysilicon, silicon nitride, and undoped and doped SiO_2 films. They find such broad applicability primarily because of their superior economy, throughput, uniformity, and ability to accommodate larger diameter (e.g. 150 mm) wafers. Their main disadvantages are susceptibility to particulate contamination and low deposition rates. A schematic drawing of such reactors is given in Fig. 5.1.

Three processes are commonly used in conventional LPCVD systems. The first uses 100% SiH_4 at total pressures from 300-1000 mtorr, while the second uses approximately 25% SiH_4 in a nitrogen carrier at approximately the same pressures. A third technique, performed in vertical flow isothermal reactor configurations, uses 25% SiH_4 diluted in hydrogen, also at about 1000 mtorr.

5.3.1. DEPOSITION PARAMETERS [50]

The deposition rate shows an exponential dependence on temperature. Fig. 5.6a is a plot of deposition rate versus reciprocal temperature. Apparent activation energies are found to depend on the silane pressure and range from 1.36 to 1.70 eV. Depositions are limited to the 580-650 $°C$ range, since at higher temperatures gas phase reactions occur (leading to rough and loosely adhering films), and below 580 $°C$ the rate is too slow for practical use (deposition rate < 50 Å/min).

The polysilicon deposition rate depends on the silane (SiH_4) pressure as shown in Fig. 5.6b . The dependence is not linear except for low values of pressure. Such behavior may be due to homogeneous reactions, adsorption of hydrogen on the surface sites, or transport phenomena.

Several surface processes can be important once the gases arrive at the hot substrate surface, but the surface reaction, in general, can be modeled by a thermally activated phenomenon which proceeds at a rate R, given by:

$$R = R_0 e^{[-E_a/KT]}$$

where R_0 is the frequency factor, E_a is the activation energy in eV, and T is temperature in $°K$. R_0 is a function of ambient pressure, silane flow rate, and position of the wafer. These three parameters basically determine the partial pressure of the silane. One such model is discussed later on in this section.

5.3.2. INTEGRATED NETWORK

The integrated network for modeling LPCVD of polysilicon films is given in Fig. 5.7. It consists of five input parameters (ambient pressure, process temperature, process time, flow rate of silane gas, and position of the silicon wafer in the processing tube), and three output parameters (deposition rate, film thickness, and stress). There are only two FFEBP networks used in the model, because there is an exact relationship between deposition rate, deposition time, and the resultant film thickness. The AM network does not cover the film thickness variable, because of the same reason. The type and range of process parameters are given below:

Variable	Range	Type
pressure (mtorr)	300 - 550	continuous
temperature $°C$	605 -650	continuous
flow rate (sccm)	100 - 250	continuous
time (min.)	60 - 150	continuous
position (cm)	3.6 - 28.8	continuous
thickness (Å)	9100 - 18700	continuous
stress($10^9 dyne/cm^2$)	-5.9 to 7.6	continuous
dep. rate ($Å/min.$)	90 - 300	continuous

5.3.3. LEARNING

The data set here is provided by Kuang-Kuo Lin (research assistant) and Costas Spanos (Assistant Professor) of the electrical engineering and computer science department of University of California at Berkeley.

There were 12 experiments performed with 6 wafers positioned along the tube for each experiment. The input parameters were chosen in such a way as to cover a maximum space of the process behavior with a minimal number of experiments. The use of first principle models for such purposes is of great importance. The reader can refer to [48,49] for parameter selection criteria. The film thickness was measured for all of the wafers, resulting in 72 data points. But the stress information was gathered selectively from the six wafers in each of the experiments, resulting in 54 data points.

The data set was then broken into two parts: learning set and testing set. The data gathered for every other wafer in the tube, for all 12 experiments, were used for the learning set, and the rest for the testing set, resulting in 36 data points for the learning set and 36 data points for the testing set. For the stress modeling, however, 34 data points were used for the learning set and 20 data points were used for the testing set. This was due to the lack of stress data for all of the wafers.

The structure of the networks, determined iteratively, for each of the three networks is shown below:

FFEBP network for deposition rate	4-5-1
FFEBP network for stress	5-5-1
AM network	7-7-7

The optimality criteria used for the selection of the training set of the AM network was arbitrarily chosen to be data points with ambient pressures equal or less than 520 mtorr. There are 27 such data points.

Figs. 5.8 and 5.9 are the results of testing the two FFEBP networks on their corresponding testing set. Each figure compares the values produced by the trained FFEBP networks to the empirical data under different experimental conditions.

The maximum and average errors in predicting the values of deposition rate and stress are given below:

	Maximum error	Average error
Deposition rate	11.0 %	1.4 %
Stress	21.2 %	13.2 %

Comparing Fig. 5.8 and 5.9 it can be seen that the values of stress are not predicted as accurately as the values of deposition rate. Although a lot of different FFEBP network configurations were tried in order to achieve a better learning, most of the networks produced the same amount of total error in learning the behavior of stress. This could be due to one or both of the following reasons.

1) Absence of an important variable, affecting the stress, in the influence diagram.

2) Presence of excessive noise in measuring the values of the stress.

Although stress values of the polysilicon film are not predicted as well as its deposition rate, the amount of error in doing so is still very reasonable.

5.3.3.1. COMPARING PERFORMANCE OF NEURAL NETWORKS TO OTHER MODELS

In order to answer the question : "Why use neural networks and not any other statistical or regression techniques ?", the performance of the neural networks in creating the models for deposition rate and stress is compared to models generated in [49] by means of statistical techniques. The models in [49] are developed by Kuang-Kuo Lin and Costas Spanos using the same experimental data. These models are presented in Figs. 5.10a-b. The model for deposition rate was developed by combining statistical regression analysis and the knowledge about the physics of the process. But the model for stress was developed purely based on statistical regression analysis. This is due to the lack of a first principle model for stress. A point to be mentioned here is that the neural networks were only trained on part of the data, whereas the models in [49] used all of the data points in creating the process models.

Fig. 5.11 represents the actual deposition rate versus the deposition rate predicted by the model in [49], for the same 36 data points of the testing set of Fig. 5.8. Comparing Figs. 5.8 and 5.11, one can see how much more accurately the neural network generated model performs, even though it was not trained on these data points.

Fig. 5.12 represents the actual stress versus the stress predicted by the model in [49], for the same 20 data points of the testing set of Fig. 5.9. Notice that the stress model in [49] does not include silane flow rate and position of the wafer as part of the parameters affecting stress. These parameters were found to be statistically insignificant. But the model generated by the neural network includes all five of the input parameters (Fig. 5.7). Comparing Figs. 5.9 and 5.12, it is obvious that the neural network generated model is more accurate in predicting the amount of stress.

The quantitative comparison of the performance of the different models, in terms of maximum and average error in predicting the experimental values, is given in table 5.2.

The important conclusions are:

1) Although the neural networks were given half as much information in creating the models, they were able to create a more accurate model of the process (refer to table 5.2).

2) The inclusion of uncertain influences in modeling a process does not degrade the performance of neural networks in creating an accurate model. Although silane flow rate and position of the wafer were found to be statistically insignificant in modeling the behavior of the stress, their inclusion in the model did not degrade its performance.

Although neural networks are capable of generating more accurate models of a process just by observing selected experimental data points, it does not mean that there is no need for the statistical models. On the contrary, the statistical models are needed in creating bounds on the amount of variability of process parameters and in giving an understanding to the degree of significance of each of the process parameters, features which are of great importance in terms of planning exploratory experiments, or setting up a process initially.

5.3.4. SYNTHESIS

In this part we have done two experiments, both of which test the generalization abilities of the integrated network to produce novel ideas.

5.3.4.1. GENERATING A RECIPE FOR A ZERO STRESS POLYSILICON FILM

In this experiment the integrated network is supplied with information about the desired film thickness, stress, and some of the input conditions such as process time and ambient pressure. The integrated network is then asked to produce values for process temperature and silane flow rate that would result in the asked for outputs. The position of wafer #3 was chosen for the position variable because it is in the middle of the processing tube and, therefore, would yield a better approximate for the rest of the wafers. The novel point about this experiment is that the required stress value is zero, and the FFEBP network for the stress has not seen any such values for stress in its training set. All of the values of the stress in the training set were either compressive or tensile. Therefore, the integrated network is required to generalize its knowledge in order to solve the problem.

Table 5.3 shows the result of the experiment. The input row is the given information. In the first output row the value of the deposition rate is calculated by using the exact relationship between time, thickness, and deposition rate. The second output shows the initial estimates for the temperature and silane flow rate produced by the AM network. The last output shows the final values produced by synthesizing the two FFEBP networks simultaneously.

To test the validity of the answers, the resultant values were plugged into models developed by K.K. Lin and Costas Spanos using the same experimental data [49]. The result of the comparison is presented in table 5.4. It is evident that the results are fairly consistent with each other.

5.3.4.2. GENERATING A NON-UNIFORM TEMPERATURE DISTRIBUTION FOR A UNIFORM DEPOSITION RATE

In this experiment only the deposition rate FFEBP network is used to come up with a non-uniform temperature distribution that results in a uniform deposition rate across the processing tube.

Fig. 5.13 shows a typical deposition rate versus wafer position for constant temperature across the tube. The drop in the deposition rate from the inlet to the outlet is because of the depletion of the active gas (SiH_4) as it passes over the wafers and its concentration is gradually decreased. To compensate for the depletion effect, a non-uniform temperature distribution across the processing tube (lower at the inlet and higher at the outlet) would result in a uniform deposition rate across the tube. This is due to the fact that the deposition rate is an exponential function of the temperature.

To produce the temperature distribution, the deposition rate FFEBP network was synthesized once for each of the six wafer positions, specifying the deposition rate, silane flow rate, and ambient pressure, and then calculating the process temperature that would result in such a deposition rate across the tube. Fig. 5.14 shows the resultant temperature distribution.

In actual practice, however, the length of the process tube is heated up in three heating zones, which can be controlled independently. Therefore, the result is a step-wise temperature distribution that approximates the distribution in Fig. 5.14.

The novel point about this synthesis is that all of the data in the training set of the deposition rate FFEBP network were for uniform temperature distributions across the tube, which resulted in exponentially dropping deposition rates (refer to Fig. 5.13). This again shows how these networks can effectively generalize their knowledge to produce novel ideas.

5.4. CONCLUSIONS

The following is a summary of experimental conclusions from this chapter.

1) FFEBP networks can learn the quantitative relationships between the process parameters just by observing discrete experimental data points (inherently polluted with some statistical noise), without the need for any knowledge about the physics of the process. The error-backpropagation training algorithm also offers the advantage of having a mix of discrete and continuous parameters in modeling a process.

2) Models generated by the FFEBP networks are more accurate in predicting the behavior of a process than models generated by the statistical regression analysis. This is due the fact that a neural network adapts itself directly to the behavior of the data rather than trying to adapt the behavior of the data to a pre-assumed model, which is the case for the regression analysis. This advantage can be attributed to the distributed manner in which knowledge is stored, the learning algorithm, and the large number of free parameters (weights and thresholds) in a neural network.

3) The synthesis procedure developed here can be used to extract information about the relationships between the process parameters in any random manner. This capability is useful for recipe generation, and diagnostics.

4) FFEBP networks can effectively generalize their knowledge to produce novel ideas for the process.

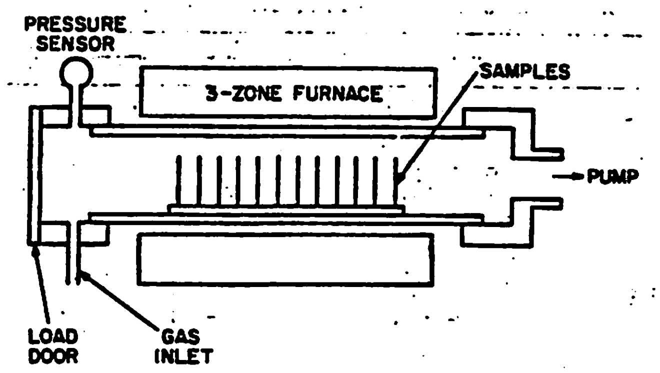


Fig. 5.1a Schematic of a processing tube [51]

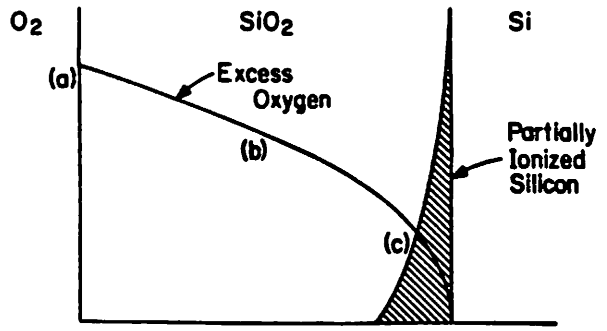


Fig. 5.1b Schematic illustration of the silicon thermal oxidation process [51]

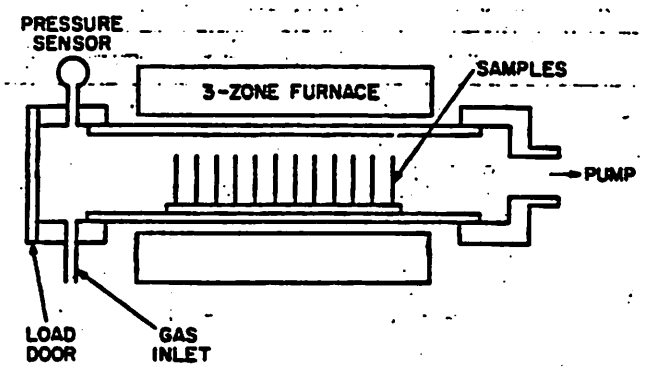


Fig. 5.1a Schematic of a processing tube [51]

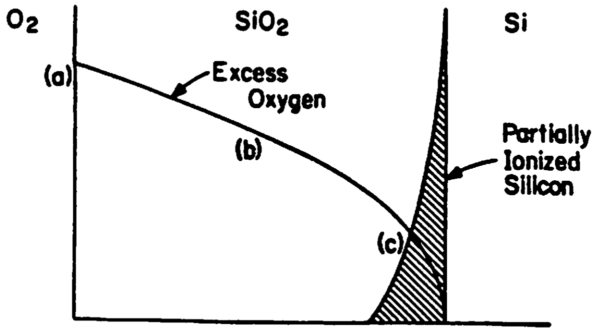


Fig. 5.1b Schematic illustration of the silicon thermal oxidation process [51]

EFFECT OF CHEMICAL CLEANS ON DRY OXIDE KINETICS

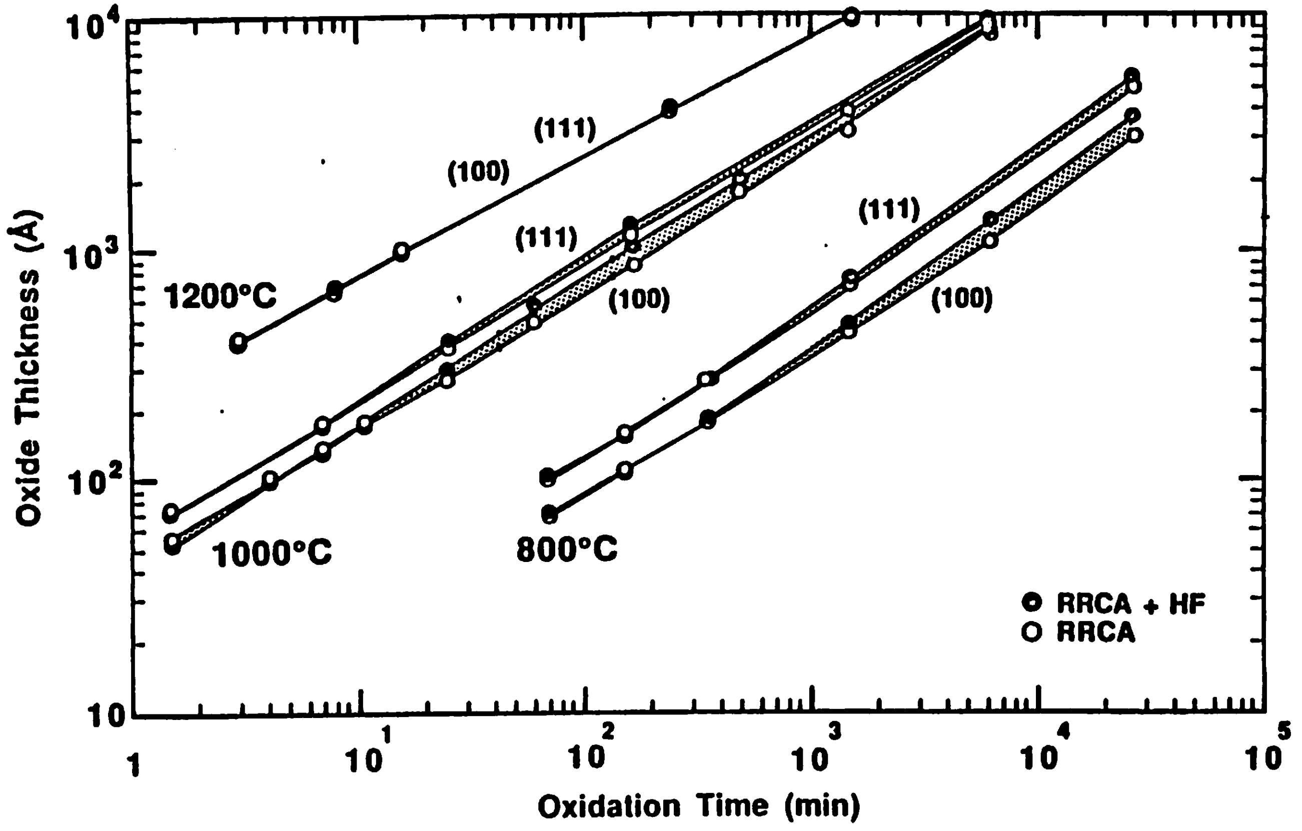


Fig. 5.2 Raw data for dry-oxidation of silicon

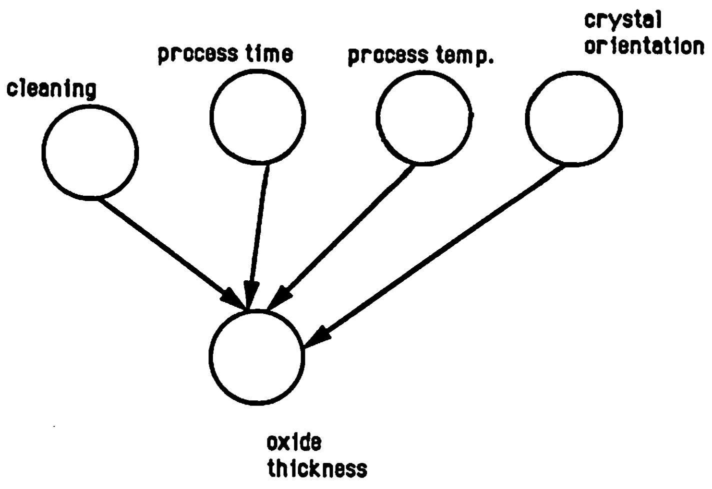


Fig. 5.3a Influence diagram for dry-oxidation of silicon

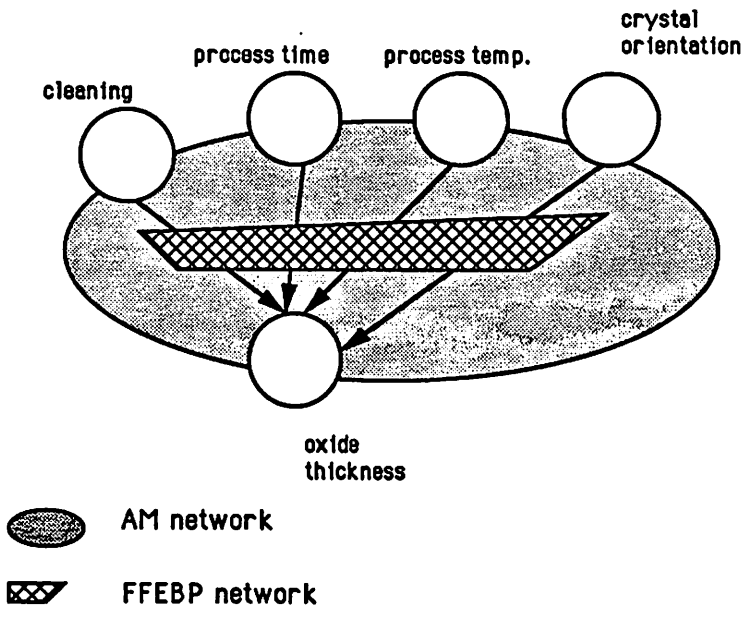


Fig. 5.3b Integrated network for dry-oxidation of silicon.

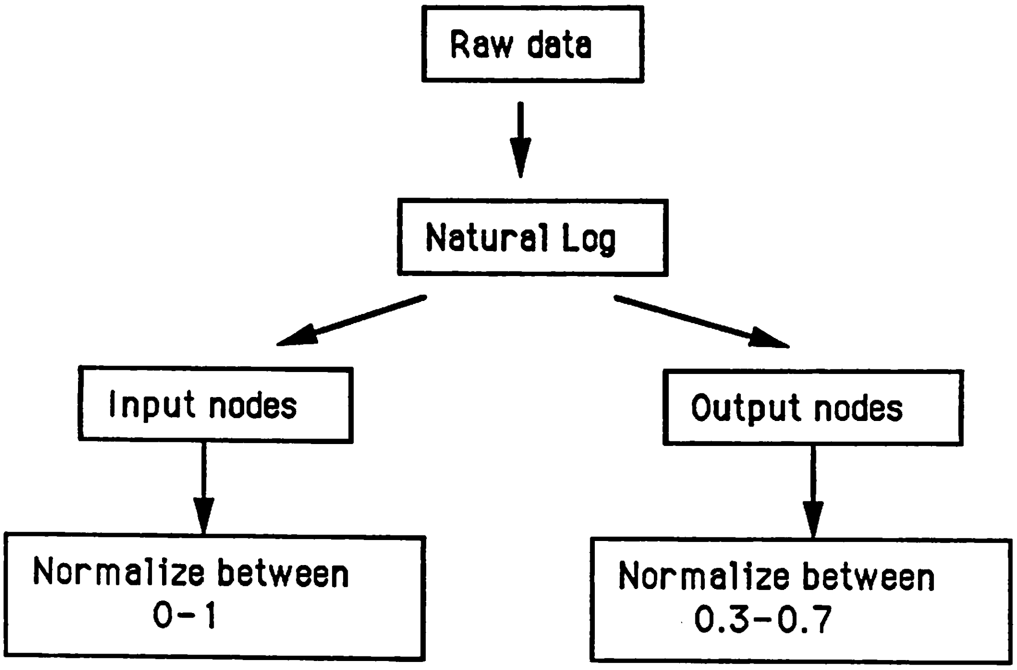


Fig. 5.4 Pre-processing of raw data

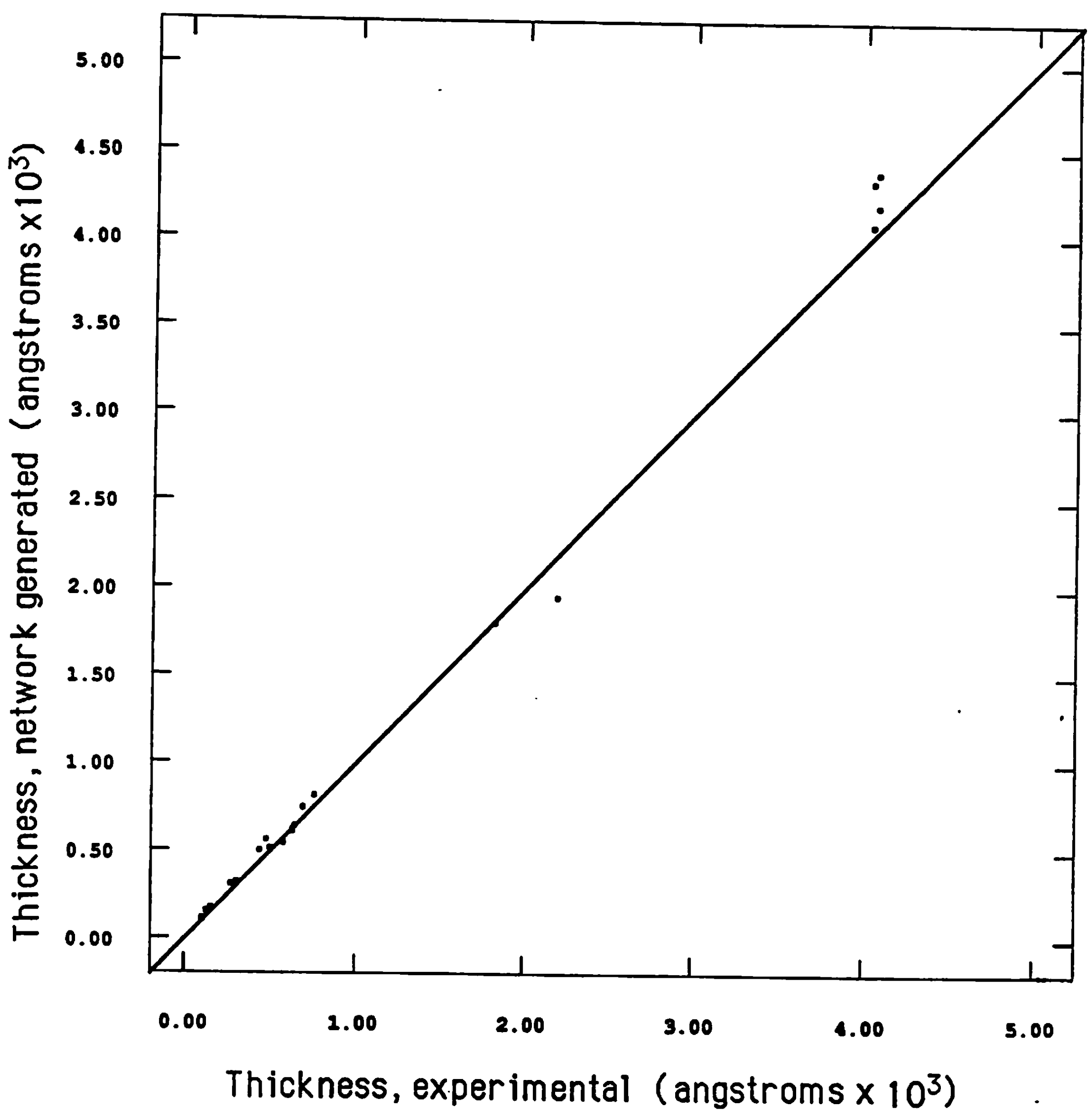


Fig. 5.5 Experimental vs. network generated oxide thickness for different input conditions.

	time	temp	clean	orientation	thickness
	min.	C			A
input	?	?	+HF	(111)	180
1st output	242	800	+HF	(111)	180
2nd output	332	801	+HF	(111)	180
experimental data	342	800	+HF	(111)	180

Table 5.1 Result of the dry oxidation synthesis

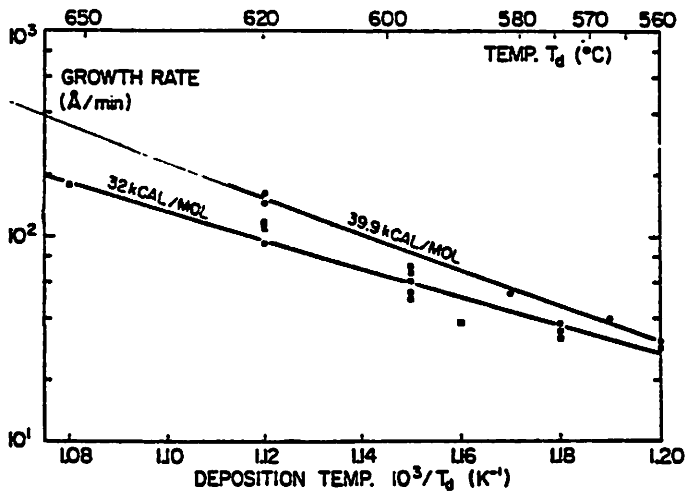


Fig. 5.6a Growth rate as a function of deposition temperature for two different deposition conditions.
1) 350 mtorr, SiH4 = 200 ccm , 2) 120 mtorr, SiH4 = 50 ccm [50]

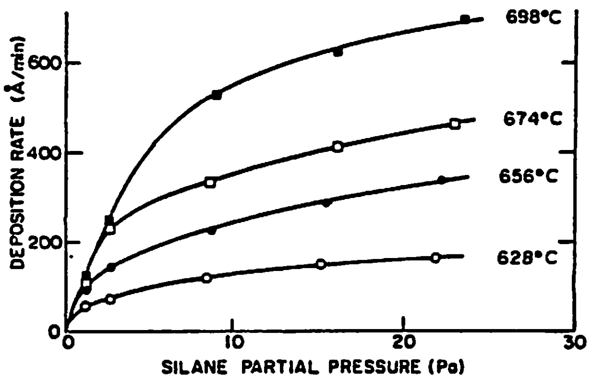


Fig. 5.6b The effect of silane concentration on the polysilicon deposition rate [50]

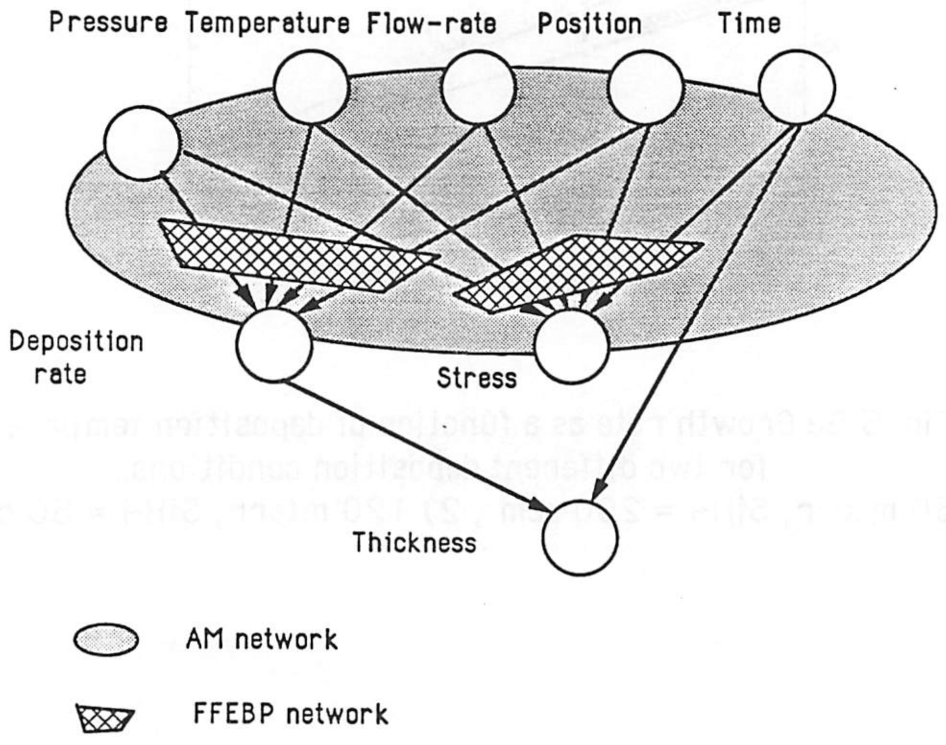
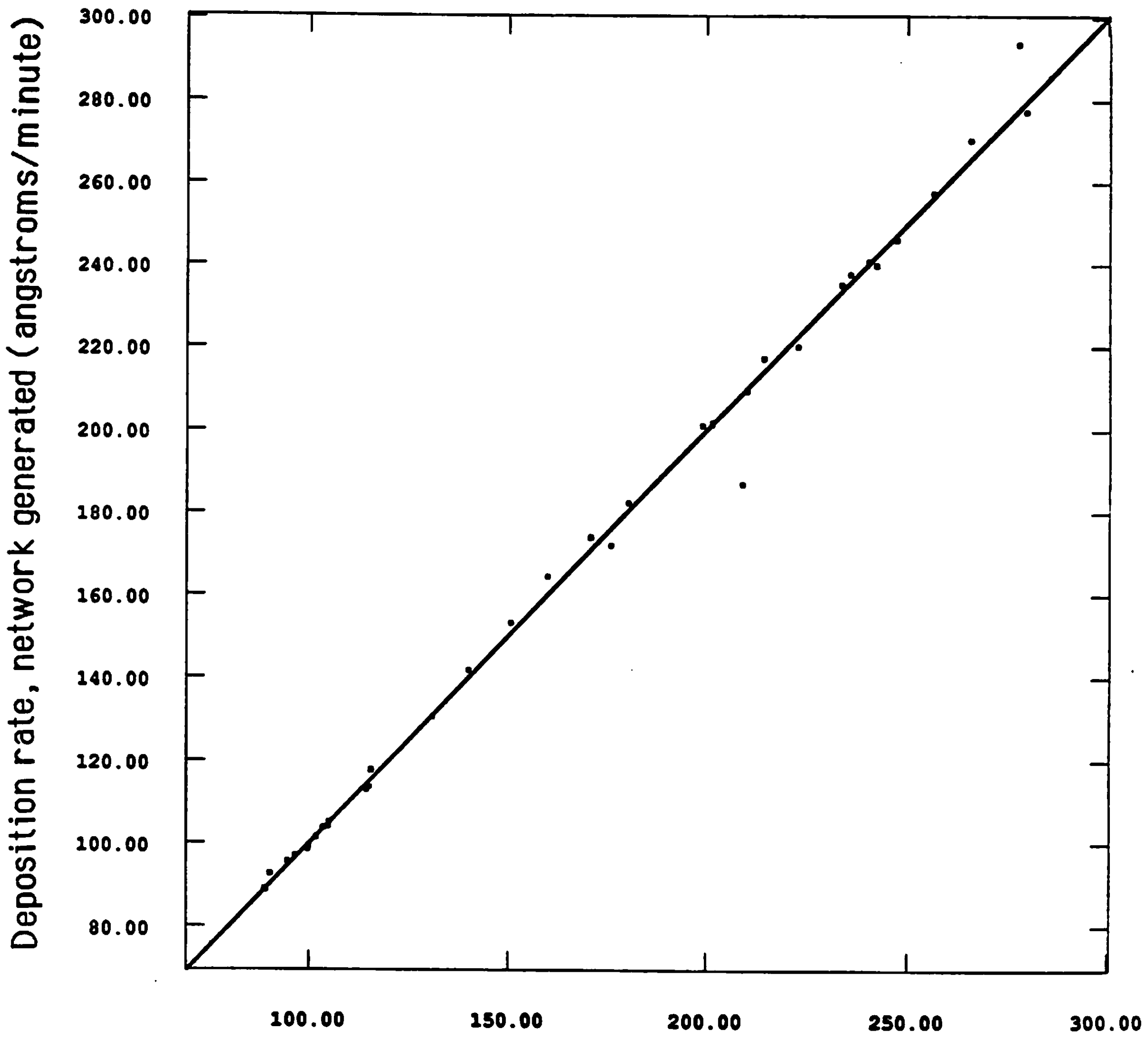


Fig. 5.7 Integrated network for LPCVD of polysilicon



Deposition rate, experimental (angstroms/minute)

Fig. 5.8 Experimental vs. network generated deposition rate for different input conditions.

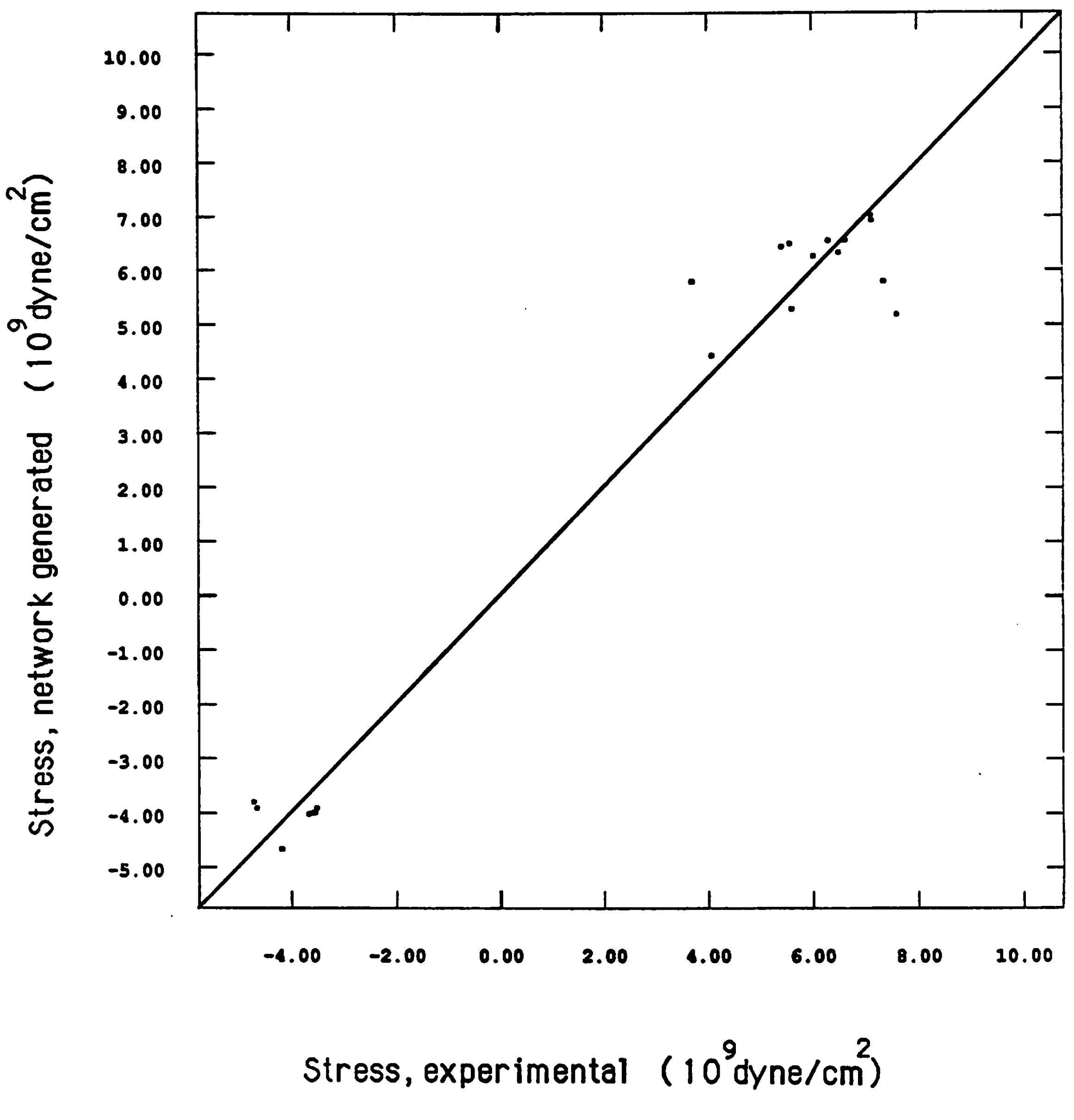


Fig. 5.9 Experimental vs. network generated stress for different input conditions.

Polysilicon Deposition Rate Model for TYLAN 16:

$$R\,(T,P,Q,X) = AP^{\alpha}e^{\frac{-\Delta E}{RT}}\left[\frac{1-\frac{k_3}{Q}}{1+\frac{k_3}{Q}}\right]\left[\frac{1-\eta(P,Q,T,X)}{1+\eta(P,Q,T,X)}\right] \quad \text{in } \frac{\text{Å}}{\text{min}}$$

Mole Fraction of Depleted SiH_4:

$$\eta(P,Q,T,X) = \left[\frac{SC_{gs}}{LQ}R_0\right]X$$

where

$$R_0\,(T,P,Q,X{=}0) = AP^{\alpha}e^{\frac{-\Delta E}{RT}}\left[\frac{1-\frac{k_3}{Q}}{1+\frac{k_3}{Q}}\right]$$

T = deposition temperature in °K. Range 878 – 923 °K.

P = deposition pressure in mtorr. Range 250 – 550 mtorr.

Q = silane flow rate in sccm. Range 100 – 250 sccm.

X = wafer position in cm. Wafers are 1.2 cm apart.
First wafer is at X=0cm, second at X=1.2cm, third at X=2.4cm, etc.

A = the Arrhenius frequencey factor = 9.2935×10^{8} Å/min·mtorr$^{\alpha}$
$\alpha = 0.2910$
ΔE =the activation energy = 30183.8 cal/mol
$k_3 = 23.9845$ sccm^{-1}

R = the universal gas constant = 1.98719 cal/mol–°K

C_{gs} = the gas solid conversion factor = 1.85×10^{-5} cm/Å

S: Surface area of the furnace, 12740.8 cm^2
L: Length of the furnace, 80 cm

Fig. 5.10a Deposition rate model for polysilicon generated by Lin, K.K. and Spanos, C. [49]

Empirical AVERAGE* Residual Stress Model for TYLAN 16:

$$\sigma(T,P,t) = a_1 \left(\frac{898-T}{22.5}\right) + a_2 \left(\frac{120-t}{45}\right) + a_3 \left(\frac{898-T}{22.5}\right)\left(\frac{400-P}{150}\right)$$
$$+ a_4 \left(\frac{400-P}{150}\right)\left(\frac{120-t}{45}\right) + a_5 \left(\frac{898-T}{22.5}\right)^2 + a_6 \left(\frac{120-t}{45}\right)^2 \text{ in } 10^9 \text{ dyne/cm}^2$$

T = deposition temperature in °K. Range 878 - 923 °K.
P = deposition pressure in mtorr. Range 250 - 550 mtorr.
t = deposition time in minutes. Range 60 - 150 minutes.

$a_1 = 3.856$
$a_2 = 3.445$
$a_3 = -2.521$
$a_4 = -1.764$
$a_5 = 3.376$
$a_6 = -5.140$

Fig. 5.10b Stress model for polysilicon generated by Lin, K.K. and Spanos, C. [49]

* AVERAGING over the wafers in 2 boats.

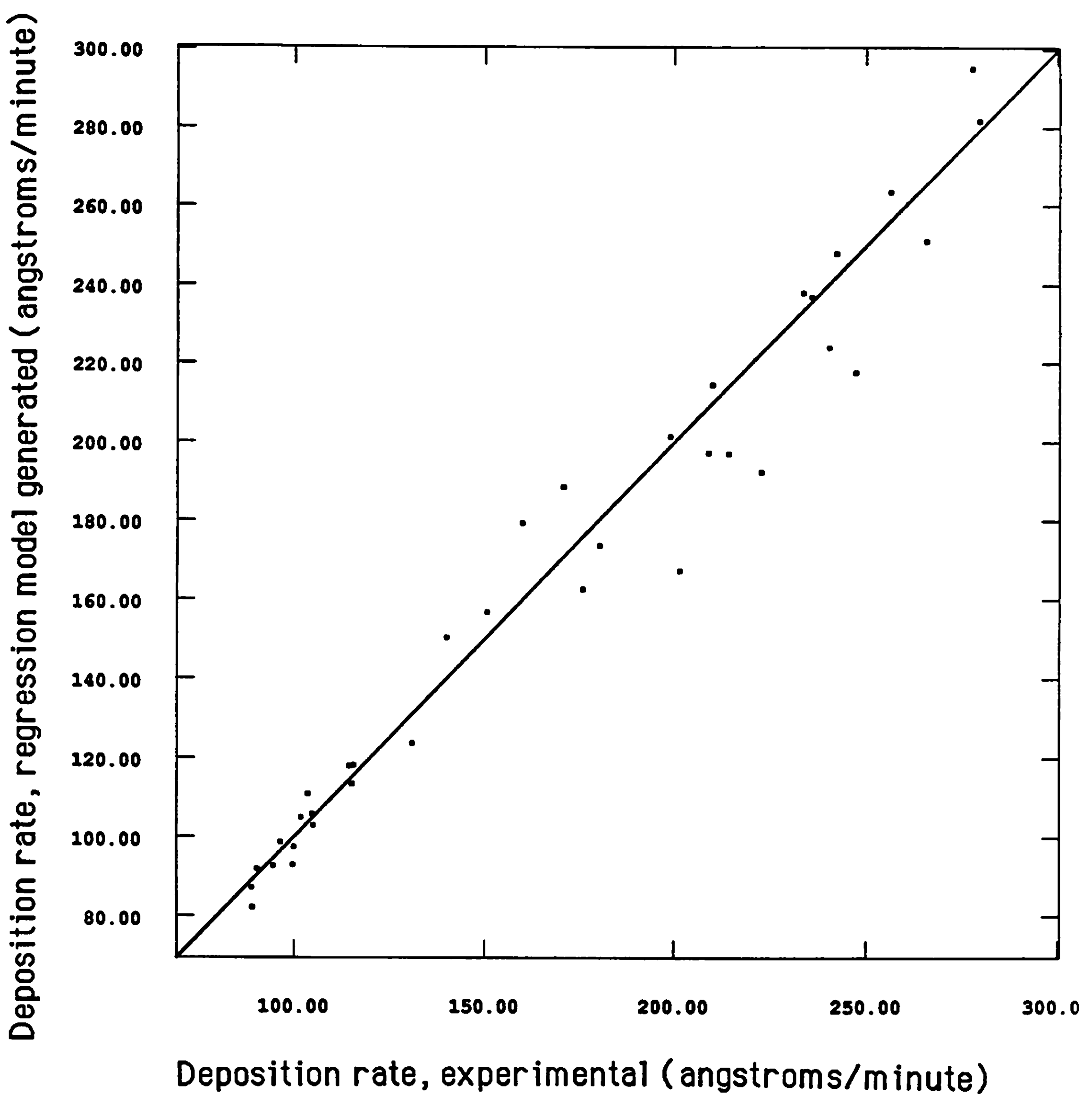


Fig. 5.1 1 Experimental vs. regression model generated deposition rate for different input conditions.

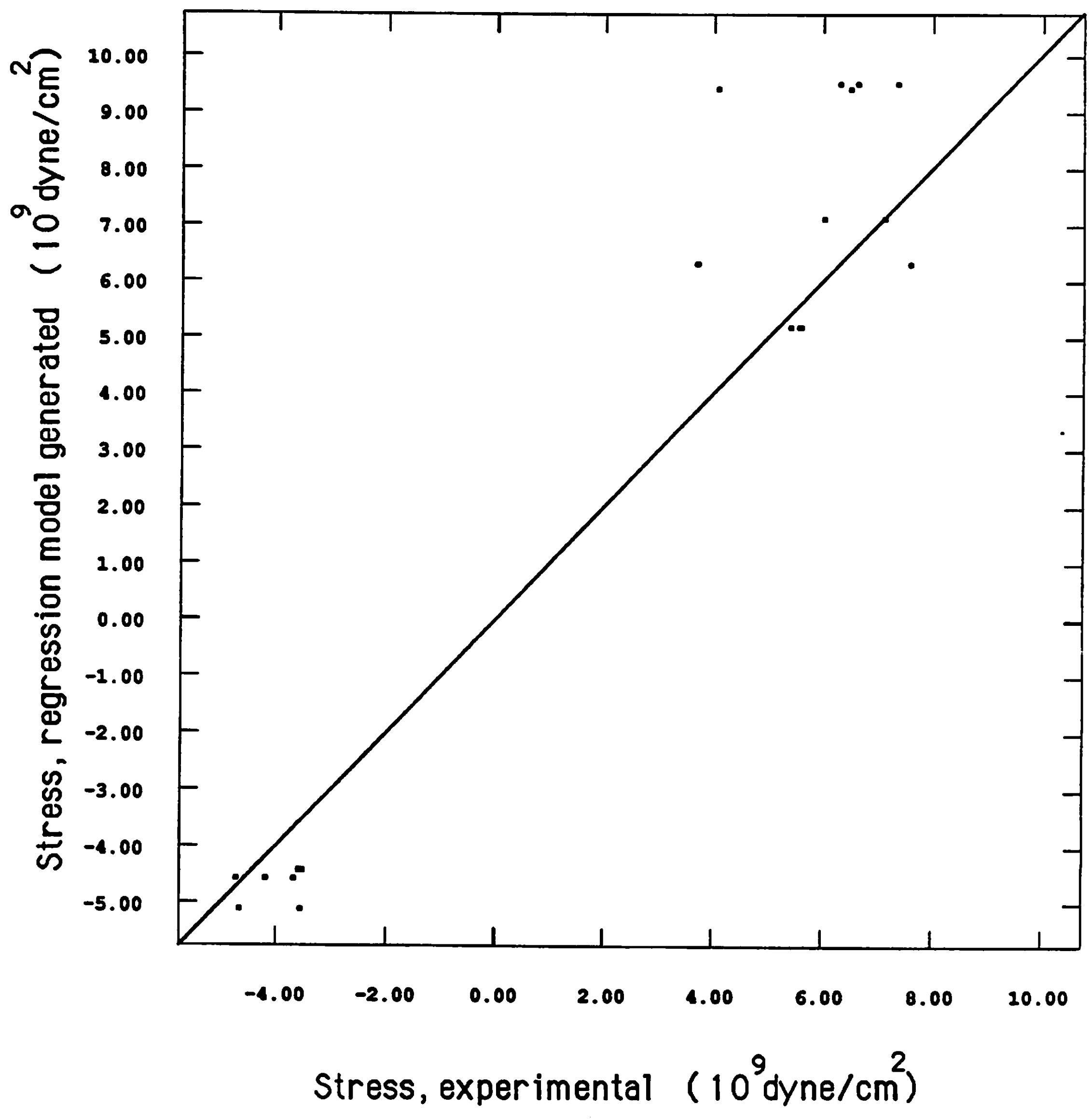


Fig. 5.12 Experimental vs. regression model generated stress for different input conditions.

	Integrated networks	Regression models
Stress	max. error = 21.2 % avg. error = 13.2 %	max. error = 131.1 % avg. error = 28.2 %
Dep. rate	max. error = 11.0 % avg. error = 1.4 %	max. error = 17.0 % avg. error = 5.1 %

Table 5.2 Comparison of the accuracy of two methods in predicting stress and deposition rate

	Pressure	Temperature	Flow rate	Time	Location	Dep-rate	Stress	Thick.
Input	450	?	?	70	13.2	?	0	10000
1st output	450	?	?	70	13.2	142.8	0	10000
2nd output	450	621	177	70	13.2	142.8	0	10000
3rd output	450	615	190	70	13.2	142.8	0	10000

Table 5.3 Result of the zero stress synthesis

	Integrated networks	Regression models
Stress x 10^9 dyne/cm ^{2}	0.0	0.89
Deposition rate angstroms/min.	142.8	151.9

Table 5.4 Comparison of results for the zero stress synthesis

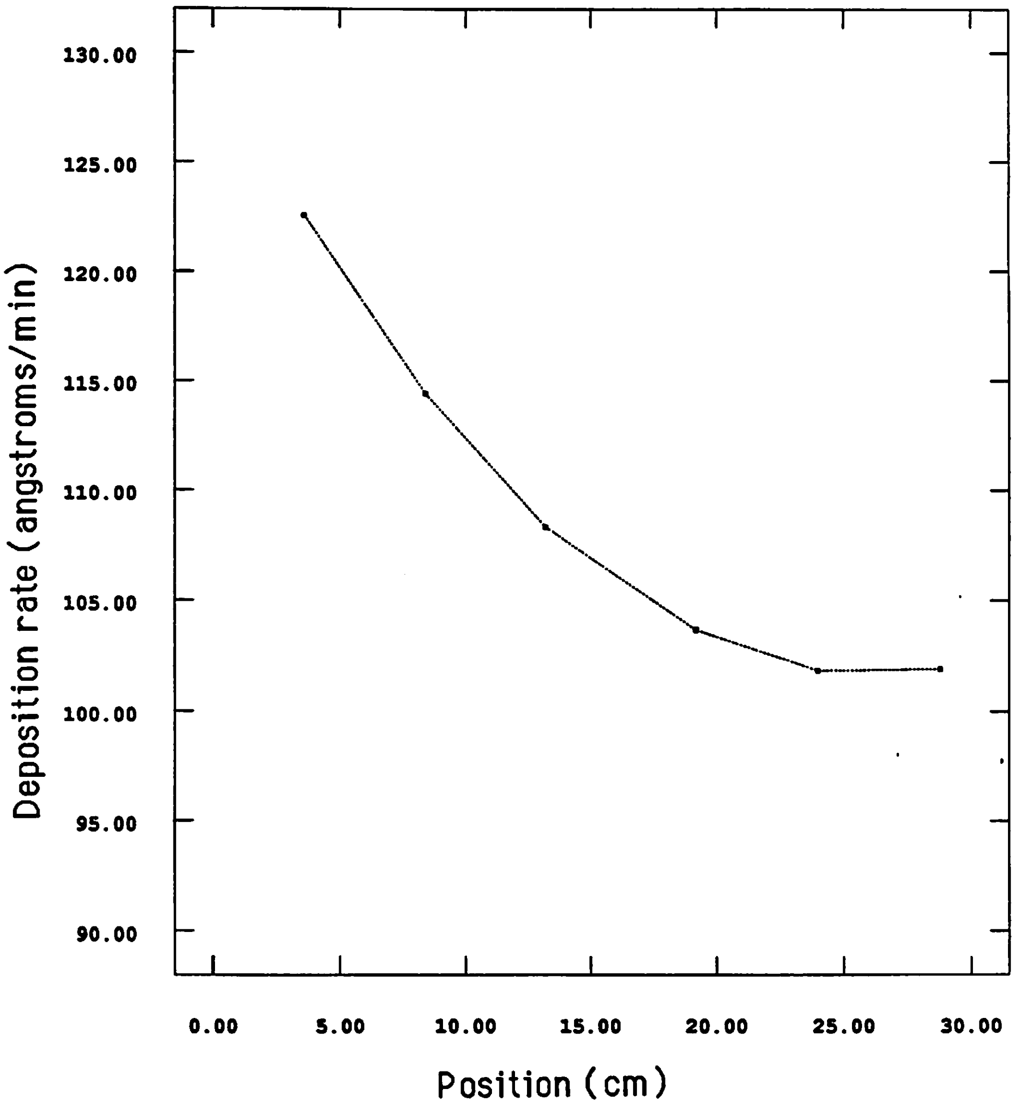


Fig. 5.13 Deposition rate vs. position for constant temperature across the tube.

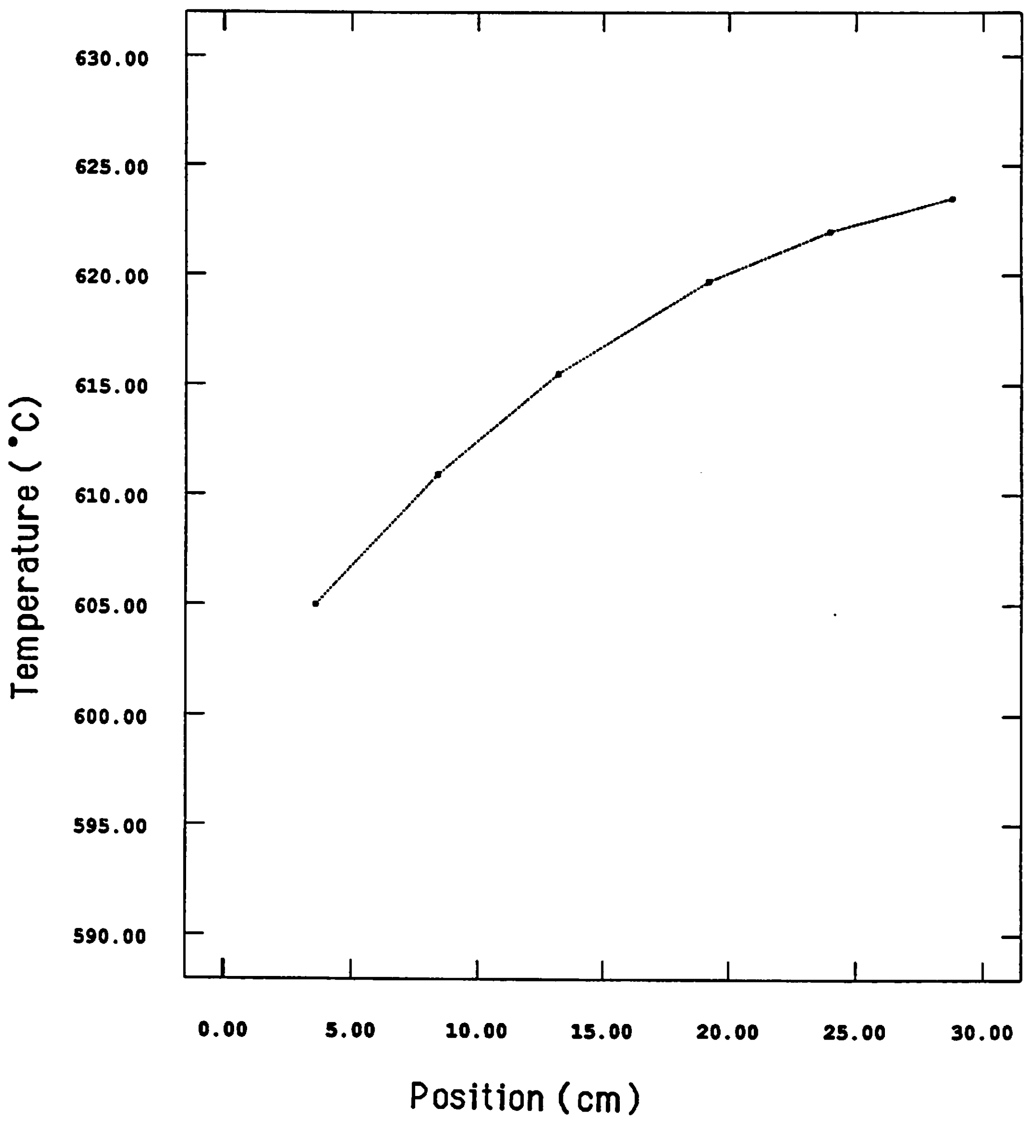


Fig. 5.14 Temperature vs. position for constant deposition rate across the tube.

CHAPTER 6
DEALING WITH PROCESS NOISE

6.1. INTRODUCTION

In a manufacturing environment, the ability of a process model to deal with the inherent noise of process parameters is of critical value. There are many sources that contribute to this statistical variation of the process behavior. Two important sources are: input noise and output noise.

The cause for input noise could be small variations in the input parameters of a process. Such variations could be small changes in the property of the raw materials (e.g. purity of the active gases), small fluctuations in the behavior of the component controllers (e.g. flow-rate controller, temperature controller), and finally human error in setting the input parameters correctly (e.g. setting the temperature or deposition time correctly). Fortunately, this last source of error will play a minor role as there is an effort to put direct communication capabilities in the process equipments, so that most of the parameters can be directly loaded from a data-base in a CIM environment.

The source for the output noise is mostly due to measurement errors, which could be caused by human error or faulty instruments (e.g. measuring thickness of an oxide film with an instrument that is out of calibration).

We have already discussed the training algorithm of FFEBP networks. In this chapter we will examine this learning procedure and show its effectiveness in filtering process noise and generating a noise-free representation of the observed process.

6.2. FILTERING OUTPUT-NOISE FOR A SINGLE-INPUT SINGLE-OUTPUT, NON-MONOTONIC, NON-LINEAR RELATIONSHIP

Assume that the following relationship describes the behavior of a process:

$$Y = Sin(X)$$

where X is the input and Y is the output. We would like to extract the given relationship by just observing sampled pairs of X and Y, where we have also added a normally distributed noise of $3\sigma = 6\%$ of Y to the output.

The normally distributed noise of $3\sigma = 6\%$ of mean value would produce a band-width of 12% allowable error, which is assumed to be a reasonable amount.

The training set consists of 44 pairs of data for X and Y, where :

$$Y = Sin(X) + \textit{normally distributed noise of } 3\sigma = 6\% * Y$$
$$\pi > X > 0$$

The structure of the FFEBP network that produced the minimum error in the training phase was 1-4-1. To see how effectively the FFEBP network has learned the true behavior of the process, a process without noise, we have to look at Figs. 6.1 and 6.2. Fig. 6.1 represents actual output (noisy output, used for training the FFEBP network) versus theoretical output (noise free output, generated by $Y = Sin(X)$). The amount of deviation from the 45 degree line represents the amount of noise present in the process. Fig. 6.2 represents output produced by the trained FFEBP network versus theoretical output, for the given input range. If the FFEBP network was able to filter the noise completely, the points in Fig. 6.2 would all lie on the 45 degree line, which, practically, is the case. Therefore, the backpropagation training procedure was able to extract the true relationship between X and Y by observing noisy samples of X and Y.

6.3. FILTERING INPUT AND OUTPUT NOISE FOR A MULTI-INPUT, SINGLE-OUTPUT, MONOTONIC, NON-LINEAR RELATIONSHIP

In the previous experiment the noise was only added to the output of the process. In this experiment a Gaussian noise will also be added to the inputs of the process. In addition, a cubic relationship is used as the theoretical process model so that the effect of the input noise will be magnified.

Since all FFEBP networks used in the integrated network are multi-input single-output, we have chosen a two input one output process model described as

$$Z = (X+Y)^3$$

where X and Y are the inputs and Z is the output. The training set for the FFEBP network was produced in the following manner:

1) Set a range for the inputs: $1 \geq X \geq 0$, $1 \geq Y \geq 0$
2) Divide the range of each input into 10 equally spaced points. Combination of the two inputs generates 100 output points.
3) Add a Gaussian noise of $3\sigma = 6\%$ of mean value to each pair of the inputs.
4) Calculate the output for all noisy input pairs using $Z=(X+Y)^3$
5) Add a Gaussian noise of $3\sigma = 6\%$ of mean value to the calculated output.
6) Use the input pairs generated in step 2 and the output of step 5 to train the FFEBP network.

Fig. 6.3 shows the procedure described above.

The FFEBP network capable of learning the input-output relationship has a 2-3-1 structure. In order to see how effectively the FFEBP network has learned the true behavior of the process, we have to compare Figs. 6.4 and 6.5. Fig. 6.4 represents actual output (output that the network was trained on) versus theoretical output (generated by $Z=(X+Y)^3$), for different input conditions. The deviation of the points from the 45 degree line represents the amount of noise in the data. Fig. 6.5 represents output produced by the trained network versus theoretical output, for the same input conditions. It can be seen that the amount of deviation from the 45 degree line in Fig. 6.5 is considerably less than the one in Fig. 6.4. Therefore, it is shown once again that the backpropagation training algorithm is very effective in filtering both input and output noise, and extracting the true model describing the behavior of the process.

The reason for the noise filtering capability of FFEBP networks lies in the training algorithm, whereby weights and thresholds of a network are adjusted such that the measure of error, which is the square of the difference between actual and desired values of the network's output, is minimized. In doing so, the network has to create a model that fits the mean value of the data set, very much like regression analysis, with a difference being that the network extracts the model instead of having a prior model and trying to fit the data to it, as is the case in regression analysis.

Since it is shown here that FFEBP networks can filter process noise, we can use the trained network as a true, noise-free representation of the process and get an overall measure of the process noise. To get an overall measure of process noise, the output values produced by a trained network can be compared to the actual output values of a process, given different input conditions. But this overall measure of process noise can not be separated into input noise and output noise, because there is no way of distinguishing the noise due to the input and the noise due to the output.

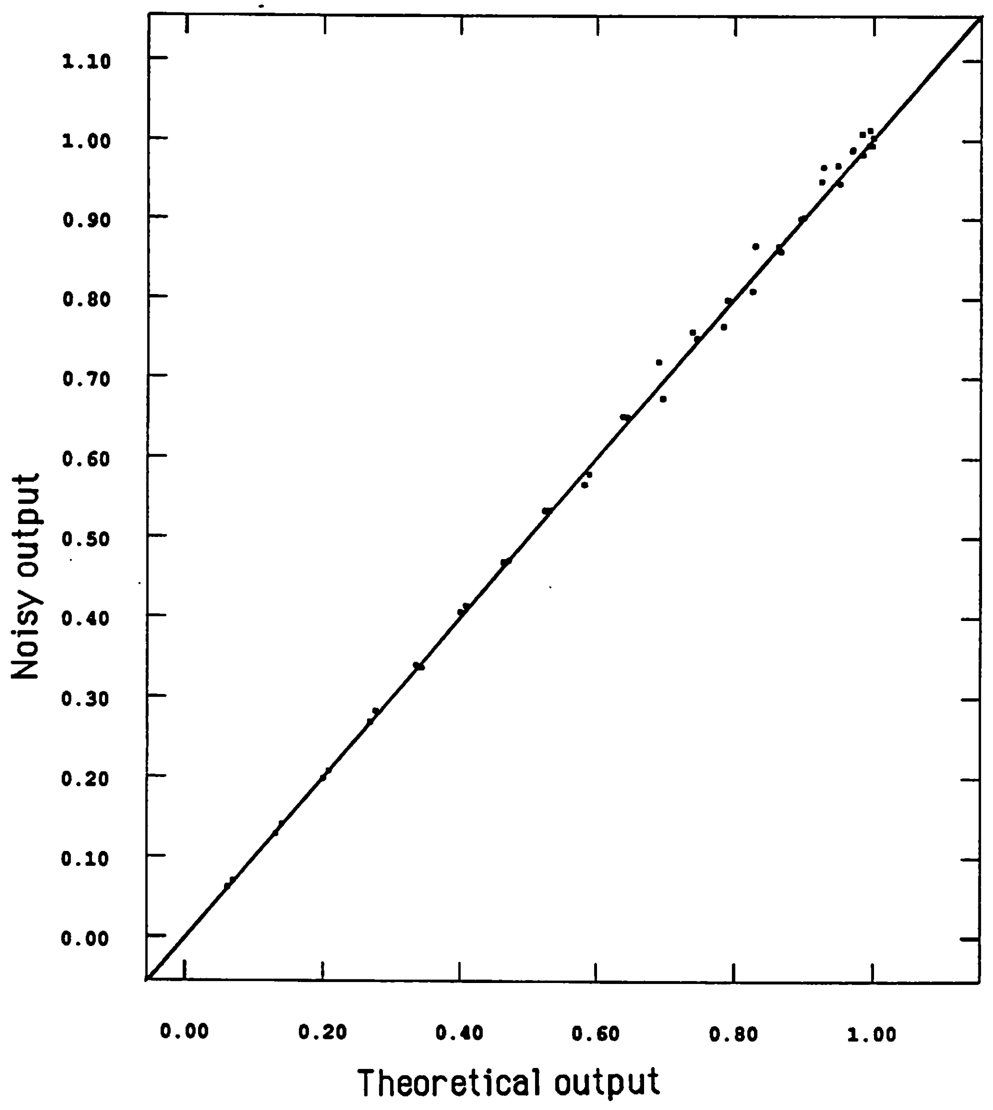


Fig. 6.1 Theoretical vs. noisy output

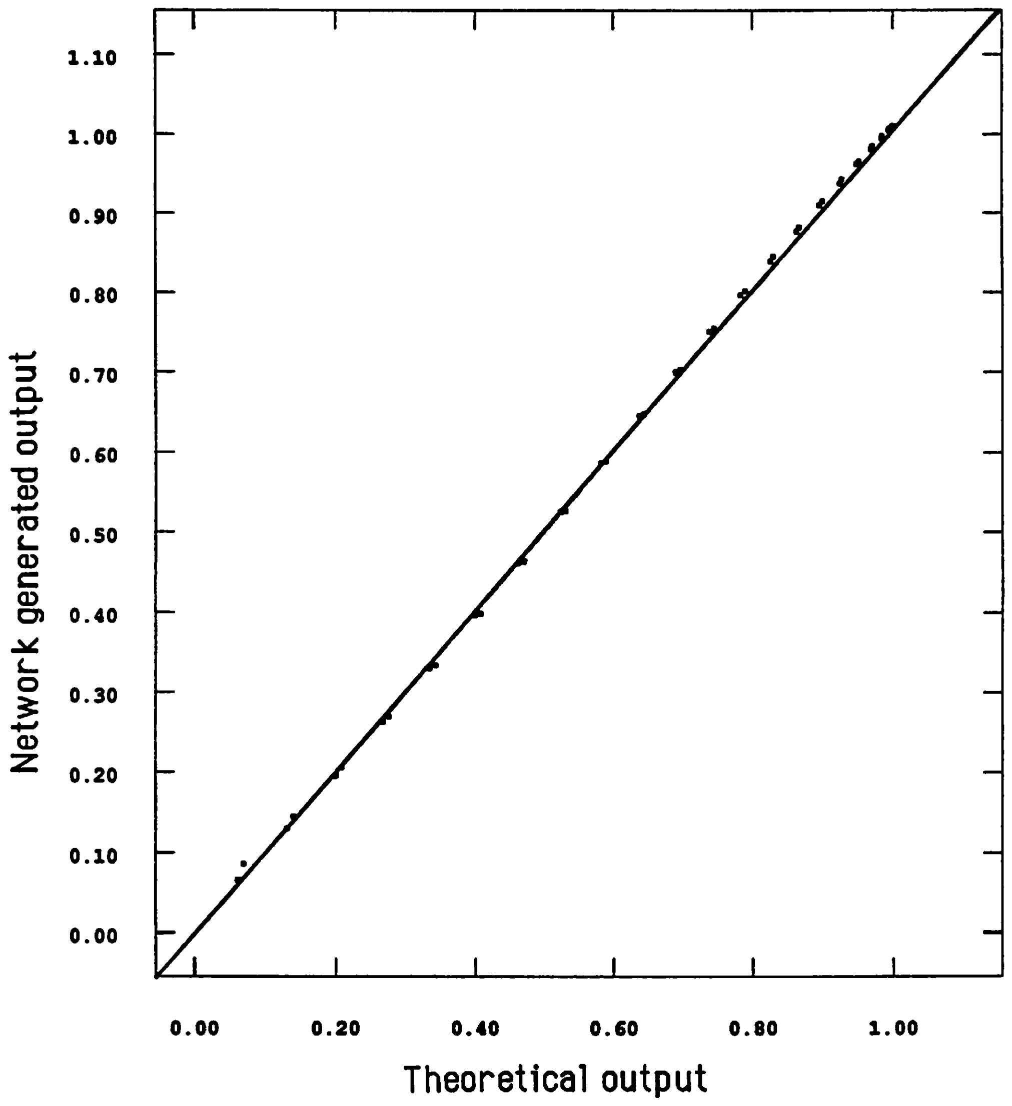


Fig. 6.2 Theoretical vs. network generated output

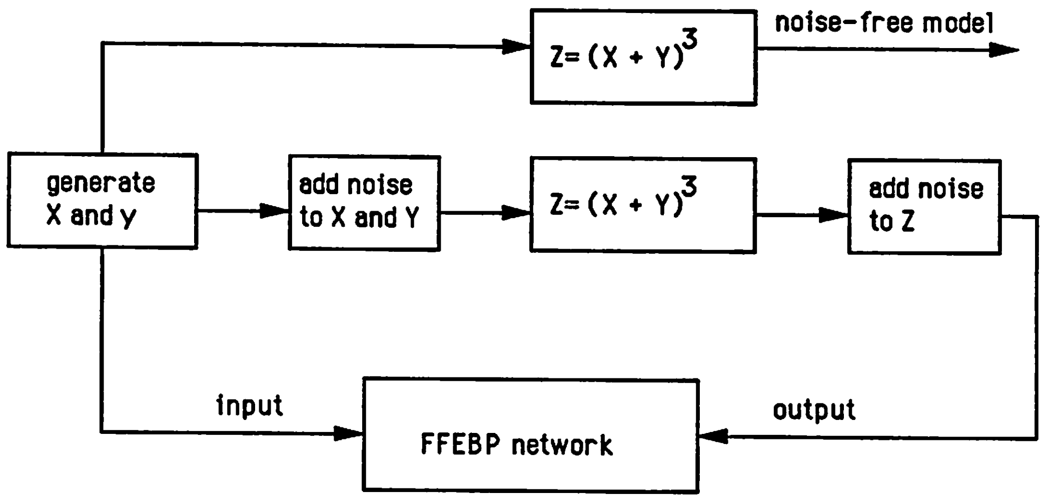


Fig. 6.3 Generation of noisy set of data for FFEBP network

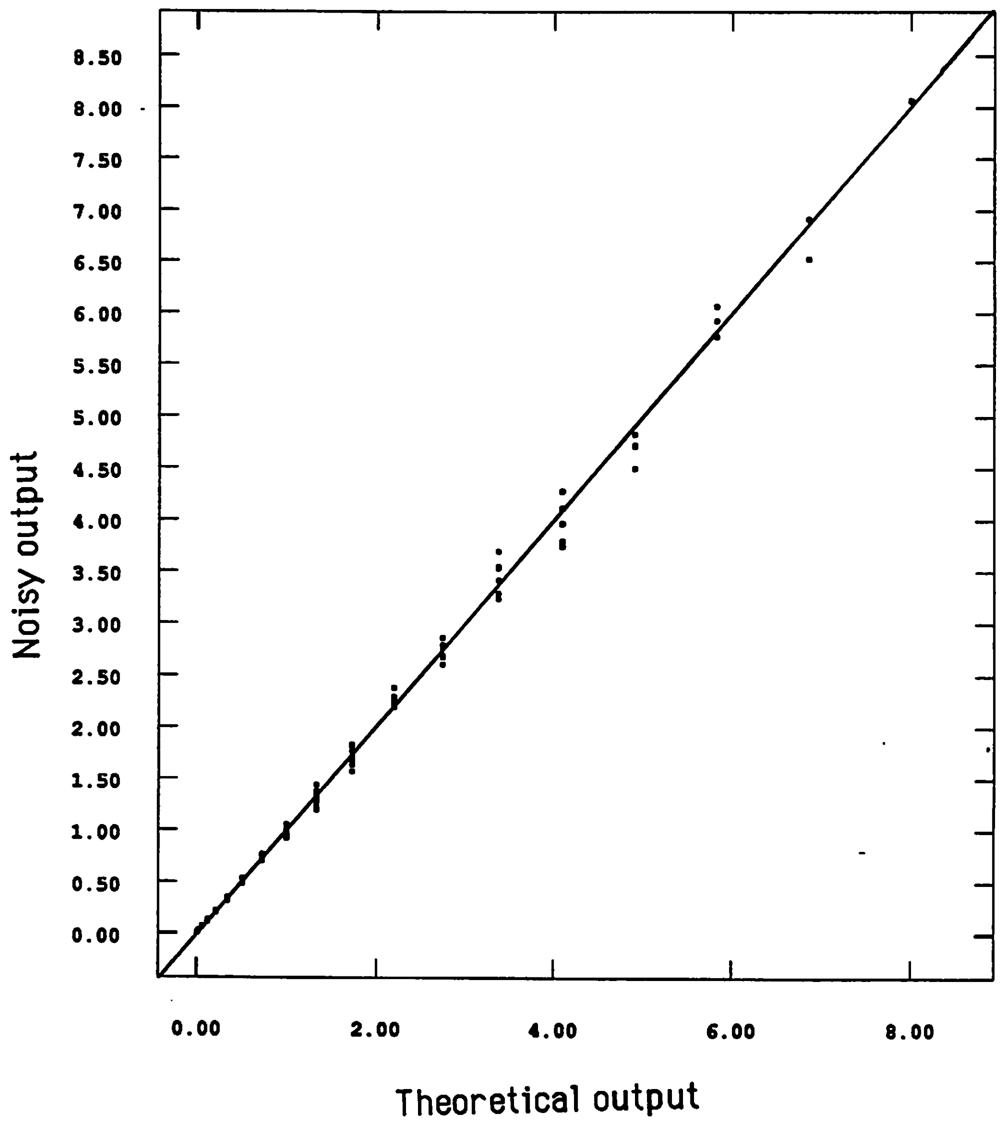


Fig. 6.4 Theoretical vs. noisy output

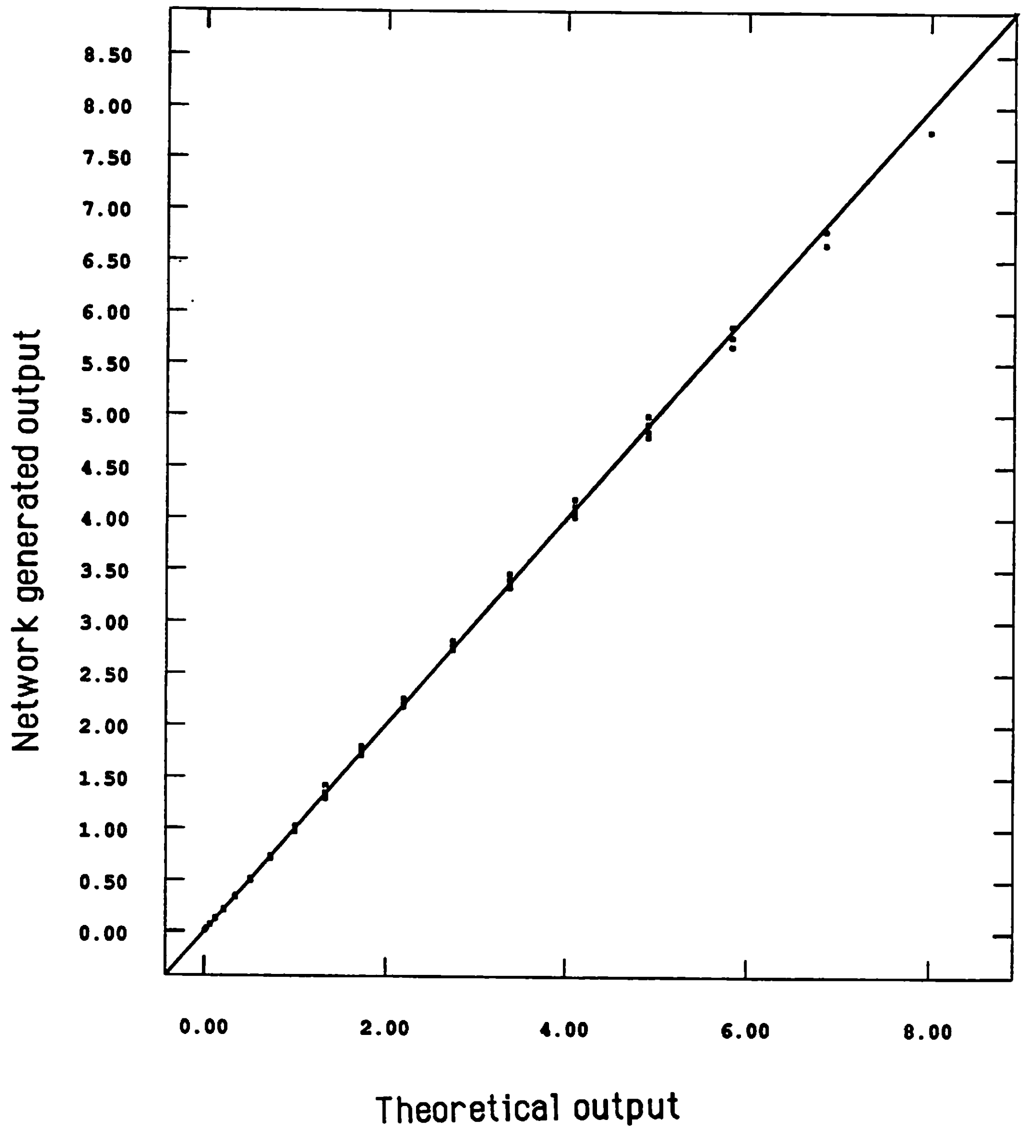


Fig. 6.5 Theoritical vs. network generated output

CHAPTER 7
CONCLUSION AND FUTURE RESEARCH DIRECTION

In this work we presented an architecture for modeling complex manufacturing processes. This architecture possesses both qualitative and quantitative knowledge of the manufacturing process. The qualitative knowledge is captured by use of the relational level of influence diagrams. An influence diagram at the relational level is a graphical representation of the manufacturing process in which critical process parameters are identified and any conditional independence between them is explicitly revealed [3]. By using influence diagrams, the task of modeling one complex process gets broken down into modeling several smaller, less complex, sub-processes. Here, a complex process is defined as a multi-input multi-output process, and a sub-process is defined as a multi-input single-output process.

In chapter two we discussed how these diagrams are constructed, either by the help of one or more human experts, each of whom could be knowledgeable about parts of the process, use of first principles, or by induction from examples [5]. We also showed that influence diagrams are an adaptive knowledge representation scheme, in which addition of new process parameters or deletion of existing process parameters can be easily accommodated, as our knowledge about the process grows.

In chapter three we discussed parallel distributed processing, or neural networks. These networks have been proven to be very effective in complex learning tasks such as human speech recognition [31] and other similar tasks involving pattern recognition, pattern completion, generalization, and categorization. In this work neural networks are used to capture the quantitative knowledge amongst the dependent parameters. We discussed two specific types of neural networks, both of which use the error-backpropagation scheme in the learning phase.

To extract the knowledge stored in the neural networks, we introduced a very flexible method of synthesizing single, or groups of neural networks simultaneously. This method employs a stochastic optimization technique called ALOPEX [42] which allows us to extract the knowledge of the interrelationships between the input and the output parameters of neural networks in any random manner.

Integration of influence diagrams and neural networks was discussed in chapter four, in which Feed-Forward Error-Backpropagation (FFEBP) networks are used to learn the quantitative and causal relationships in each of the sub-processes.

To coordinate the operation of several FFEBP networks in the knowledge extraction or synthesis phase, another type of neural network was applied in this work: the Associative Memory (AM) network. The AM network is responsible for re-integrating the sub-processes into a process by learning the common and optimal operating space of all the process parameters so that it can limit the search space in trying to find solutions that satisfy simultaneous constraints imposed by all of the FFEBP networks. Therefore, this network is responsible for pattern completion, where a partial vector of process parameters is completed according to the best match stored in this AM network. This initial solution is then simultaneously synthesized by the FFEBP networks, using their exact and causal knowledge.

The synthesis procedure used for both types of networks employs the ALOPEX [42-44] optimization technique. The only difference in synthesizing the two types of networks is in the definition of the input and the parameters of the network, and in the definition of the objective functions, as defined in chapter three.

In chapter five we discussed the experimental results of our work. The effectiveness of neural networks in creating accurate models of two semiconductor manufacturing processes (dry-oxidation of silicon, and low pressure chemical vapor deposition of polysilicon), was proven experimentally. In testing each model, neural networks were trained on a part of experimentally gathered data and were then tested on the rest of the data, on which they were not trained. We also compared the performance of the models generated by neural networks to other models developed by means of statistical regression analysis, and knowledge about the physics of the process. In this comparison the modeling technique developed here performed at least twice more accurately (on the average) as the other models, while given half as much information in producing the models.

Again the effectiveness of the integrated networks was proven in the synthesis procedure by generating process recipes for specified process outputs.

We also showed how the generalization capabilities of these networks can be used to create novel ideas (new knowledge, created by generalizing the relationships learned in the knowledge acquisition phase). These were tested successfully against experimental data or other models. Two such ideas were produced here: (1) a zero stress recipe for a polysilicon film, and (2) a non-uniform temperature distribution across the processing tube for a uniform deposition rate of polysilicon film.

The ability of neural networks to filter process noise, that is both input noise and output noise, was demonstrated in chapter six. This was done by starting with a known non-linear model, adding Gaussian noise to input, calculating output, and finally adding more Gaussian noise to the output. Networks trained on the noisy data were able to extract the initial noise-free relationships.

The benefits of this modeling approach can be summarized in the following way:

1) The ability to create an accurate model of the manufacturing process just by observing selected experimental data (polluted with process noise), without the need to have an understanding about the physics of the manufacturing process.

2) Adaptive learning scheme in both qualitative and quantitative phases of knowledge acquisition.

3) An architecture that can be directly implemented on hardware in the near future, and can also act as a knowledge-base for a higher level reasoning.

4) The ease of extracting knowledge from experts by means of creating influence diagrams, which are also very useful in terms of knowledge transfer.

5) Generalization capabilities of neural networks, allowing the model to create novel ideas.

We should also, however, mention the disadvantages of this modeling approach:

1) Computationally intensive in the learning phase; the learning though, can be done off-line. In the near future, the whole architecture can be implemented on hardware, which will increase the speed of computations dramatically because of its parallel distributed processing capabilities.

2) The quantitative knowledge stored in the neural networks is in terms of strengths of connections (weights) between processors, and as such these weights do not give us any understanding about the physics of the process, as opposed to a simple linear, quadratic, or cubic relationship generated by regression analysis.

3) This approach is not suggested for initial process set-up due to the small amount of data available and large level of variability. First principle models should be used for that purpose. Although this technique can be used to provide insight because of its generalization capability.

THE FUTURE DIRECTION OF RESEARCH

Further development of this work can be followed in two different directions: theory and application.

In terms of theory, we can expand this work in two areas. First, instead of using backpropagation networks, which require supervised learning, we can use networks that employ unsupervised learning schemes [20-23]. One such application is discussed by Burke [24]. Second, more intelligent optimization routines than purely stochastic ones can be used for the synthesis of the networks. One possible optimization procedure is discussed by Jain and Agogino [46]. There is also an on-going effort to improve the backpropagation learning scheme in order to speed-up the learning procedure [26].

In terms of application, this method can be applied to model different areas which require learning. One such area is diagnostics. If an influence diagram can be constructed for a problem domain, and enough experimental data can be gathered for the variables of the model, there is no apparent reason why this approach can not model the behavior of the problem under consideration.

One very useful application in diagnostics would be in the area of preventive maintenance [47]. The up-time of processing equipment plays a very definite role in the efficiency of a manufacturing environment, and as such it would be very valuable to detect and correct equipment related problems in their early stages, before they become catastrophic. Most major problems, in their early stages, manifest themselves in minor drifts in some of the process variables. One can create a preventive maintenance and diagnostic model for the processing equipment by creating an integrated network that relates trends in the process and equipment related parameters to an upcoming maintenance problem. The neural networks of the integrated network would each be trained on the history of a set of parameters that relates them to specific equipment problems. Once a model is generated, it can be used as a tool for scheduling preventive maintenance.

References

1. Agogino, A.M., "Use of Probabilistic Inference in Diagnostic Expert Systems," *Proceedings of the ASME International Computers in Mechanical Engineering Conference,* 2, pp. 305-310, 1985.

2. Agogino, A.M., and Rege, A., "IDES: Influence Diagram based Expert system," *Mathematical Modeling in Science and Technology,* 8, pp. 227-233, 1987.

3. Rege, A., and Agogino, A.M., "Topological Framework for Representing and Solving Probabilistic Inference Problems in Expert Systems," *IEEE Transactions on Systems, Man and Cybernetics,* 18, (3), 1988.

4. Rege, A., and Agogino, A.M., "Sensor-Integrated Expert System for Manufacturing and Process Diagnostics," *Knowledge-based Expert Systems for Manufacturing, ASME-PED* 24, pp. 67-83, 1986.

5. Russell, S., Srinivas, S., and Agogino, A.M., "Inducing Influence Diagrams from Examples," *Berkeley Expert Systems Technology Laboratory, Department of Mechanical Engineering, U.C. Berkeley* Working paper #88-0202-1

6. Sampath, S., and Agogino, A.M., "Automated Construction of Sparse Bayesian Network Topologies from an Unstructured Probabilistic Model and Expert Domain information," to appear in *Proceedings of the Fifth Workshop on Uncertainty and AI,* Rev. 0, March, 1989.

7. Miller, A.C., Merkhofer, R.A., Howard, R.A., Matheson, J.E., and Rice, T.R. "Development of Automated Aids for Decision Analysis," *SRI International Technical Report,* 3309, 1976.

8. Owen, D.L., "The Use of Influence Diagrams in Structuring Complex Decision Problems," *Proceedings of the second Lawrence Symposium on Systems and Decision Sciences,* 1978. Also in Bunn, D.W., *Applied Decision Analysis,* McGraw-Hill, pp. 212-216, 1984.

9. Howard, R.A., and Matheson, J.E., "Influence Diagrams," *Readings on the Principles and Applications of Decision Analysis,* Howard, R.A., and Matheson, J.E., (eds.), Strategic Decisions Group, 2, pp. 719-762, 1984.

10. Holtzman, S., "Intelligent Decision Systems," *Ph.D. Dissertation, Stanford University,* 1985.

11. Rangwala, S., and Dornfeld, D., "Integration of Sensors via Neural Networks for Detection of Tool Wear States," *1987 Winter Annual Meeting of the ASME, Symposium on Intelligent and Integrated Manufacturing. Manufacturing Synthesis and Analysis, ASME-PED* vol. 25, pp. 109-120.

12. Rangwala, S., and Dornfeld, D., "Learning and Optimization of Machining Operations Using Computing Abilities of Neural Networks," *IEEE Transactions on Systems, Man and Cybernetics,* March-April, 1989.

13. Rangwala, S., "Machining Process Characterization and Intelligent Tool Condition Monitoring Using Acoustic Emission Signal Analysis," *Ph.D. Dissertation, University of California at Berkeley, Department of Mechanical Engineering,* November, 1988.

14. Rosenblatt, R., *Principals of Neurodynamics,* Spartan Books, New York, 1959.

15. Hinton, G. and Fahlman, S., "Connectionist Architectures for Artificial Intelligence," *IEEE Computer,* pp. 100-109, January, 1987.

16. Hopfield, J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences,* vol. 79, pp. 2554-2558, April, 1982.

17. Hopfield, J., "Neurons with Graded Response have Collective Computation properties like those of Two-State Neurons," *Proceedings of the National Academy of Sciences,* vol. 81, pp. 3088-3092, May 1984.

18. Hopfield, J., and Tank, D., "Computing with Neural Circuits: A Model," *Science,* vol. 233, pp. 625-633, August, 1986.

19. Tank, D., and Hopfield, J., "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," *IEEE Transactions on Circuits and Systems,* vol. CAS-33/5, pp. 533-541, May, 1986.

20. Grossberg, S., "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks,* 1:1, pp. 17-62, 1987.

21. Grossberg, S., "Competitive Learning: From Interactive Activation to Adaptive Resonance," *Cognitive Sciences,* vol. 11, pp. 23-63, 1987.

22. Carpenter, G.A., and Grossberg, S., "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics and Image Processing,* 37, pp. 54-115, 1987.

23. Carpenter, G.A., and Grossberg, S., "ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics,* 26:23, pp. 4919-4946, December, 1987.

24. Burke, L.I., "Automated Identification of Tool Wear States in Machining Processes: An Application of Self-Organizing Neural Networks," *Ph.D. Dissertation, University of California at Berkeley, Department of Industrial Engineering and Operations Research,* July, 1989.

25. Lippmann, R., "An Introduction to Computing with Neural Networks," *IEEE ASSP Magazine,* 4:2, pp. 4-21, 1987.

26. Watrous, R.L., "Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization," *Proceedings of IEEE International Conference on Neural Networks,* II, pp. 619-627, 1987.

27. Spencer, E., "Programmable Bistable Switches and Resistors for Neural Networks," *Proceedings of the American Institute of Physics Conference on Neural Networks for Computing,* pp. 414-419, New York, 1986.

28. Owen, A., Comber, Le P., Sarraybarouse, G., and Spear, W., "New Amorphous-Silicon Electrically Programmable Nonvolatile Switching Device," *IEEE Proceedings,* vol. 129, part 1, no. 2, pp. 51-54, April, 1982.

29. Hubbard, W., Schwartz, D., Denker, J., Graf, H., Howard, R., Jackel, L., Straughn, B., and Tennant, D., "Electronic Neural Networks," *Proceedings of the American Institute of Physics Conference on Neural Networks for Computing,* pp. 227-234, New York, 1986.

30. Graf, H., Jackel, L., Howard, R., Straughn, B., Denker, J., Hubbard, W., Tennant, D., and Schwartz, D., "VLSI Implementation of a Neural Network Memory with Several Hundreds of Neurons," *Proceedings of the American Institute of Physics Conference on Neural Networks for Computing,* pp. 182-187, New York, 1986.

31. Rumelhart, D., and McClelland, J., *Parallel Distributed Processing, Volume 1,* MIT press, Cambridge, MA, 1986.

32. Fukushima, K., Miyake, S., and Ito, T., "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Transactions on Systems, Man and Cybernetics,* vol. SMC-13/5, pp. 826-834, September 1983.

33. Bavarian, B., "Introduction to Neural Networks for Intelligent Control," *IEEE Control Systems Magazine,* pp. 3-7, April, 1988.

34. Psaltis, D., Sideris, A., and Yamamura, A., "A Multi-layered Neural Network Controller," *IEEE Control Systems Magazine,* pp. 17-21, April, 1988.

35. Guez, A., Eilbert, J., and Kam, M., "Neural Network Architecture for Control," *IEEE Control Systems Magazine,* pp. 22-25, April, 1988.

36. Barto, A., Sutton, R., and Anderson, C., "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man and Cybernetics,* vol. SMC-13/5, pp. 834-846, August, 1986.

37. Minsky, M., and Papert, S., *Perceptrons,* MIT Press, Cambridge, MA, 1969.

38. Cun, Le Y., "A Learning Procedure for Asymmetric Threshold Networks," *Proceedings of Cognitiva,* Paris, June, 1985.

39. Pault, D.C., and Hinton, G.E., "Learning Sets of Filters Using Backpropagation," *Computer Speech and Language,* 2, pp. 35-61, 1987.

40. Kohonen, T., and Oja, E., "Fast Adaptive Formation of Orthogonalizing Filters and Associative Memory in Recurrent Networks of Neuron-like Elements," *Biol. Cybernetics,* vol. 21, pp. 85-95, 1976.

41. Kohonen, T., *Associative Memory: A System Theoretical Approach,* Springer, New York, 1977.

42. Harth, E., Kalogeropoulos, T., and Pandya, A.S., "A Universal Optimization Network," *Proceedings of IEEE-EBMS,* November, 1988.

43. Harth, E., and Pandya, A.S., "Dynamics of the ALOPEX Process: Application to Optimization Problems," *Biomathematics and Related Computational Problems,* pp. 459-471, 1988.

44. Harth, E., and Tzanakou, E., "ALOPEX; A Stochastic Method for Determining Visual Receptive Fields," *Vision Res. 14* pp. 1475-1482, 1974.

45. Kirkpatrick, S., Gelatt, Jr. C.D., and Vecchi, M.P., "Optimization by Simulated Annealing," *Science,* vol. 220, pp. 671-680, 1983.

46. Jain, P., and Agogino, A.M., "Global Optimization Using the Multistart Method," to appear in *Proceedings of the 1989 ASME Design Automation Conference,* (September 17-20, 1989, Montreal, Canada).

47. Spanos, C., Personal Communication.

48. Spanos C., and Director, S.W., "Parameter Extraction for Statistical IC Process Characterization," *IEEE Transactions on Computer-Aided Design,* vol. CAD-5, no. 1, January, 1986.

49. Lin, K.K., and Spanos, C., "Statistical Equipment Modeling for VLSI Manufacturing," *to be Presented in the 176th Electrochemical Society Meeting,* Automated Manufacturing Session, October, 1989.

50. Wolf, S., and Tauber, R., *Silicon Processing For the VLSI Era,* vol. 1, Lattice Press, 1986.

51. Deal, B., "The Thermal Oxidation of Silicon," *Proceedings of Electrochemical Society: Tutorial on Semiconductor Technology,* May, 1982.

52. Deal, B., and Grove, J., *Journal of Applied Physics,* pp. 3770, Dec., 1965.

53. Frosch, C., and Derick, L., *Journal of Electrochemical Society,* vol. 104, pp. 547, 1957.

54. Jorgensen, P., *Journal of Chemistry and Physics,* vol. 37, pp. 874, 1962.