Designing Massive 3-Dimensional Neural Networks with Chromosomal-Based Simulated Development

By

Robert Glen Schinazi

Dissertation Submitted to the Faculty of the
Virginia Polytechnic Institute and State University
In partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
in
INDUSTRIAL AND SYSTEMS ENGINEERING

APPROVED:

_____
Roderick Reasor, Ph.D. Advisor

_____ _____
Guo-Quan Lu, Ph.D. Subhash C. Sarin, Ph.D.

_____ _____
Bevlee A. Watford, Ph.D. Loren Paul Rees, Ph.D.

_____
William G. Sullivan, Ph.D.

November 1997
Blacksburg, Virginia

Designing Massive 3-Dimensional Neural Networks with
Chromosomal-Based Simulated Development

By
Robert Glen Schinazi

ABSTRACT

A technique for designing and optimizing the next generation of smart process controllers has been developed in this dissertation. The literature review indicated that neural networks held the most promise for this application, yet fundamental limitations have prevented their introduction to commercial settings thus far. This fundamental limitation has been overcome through the enhancement of neural network theory.

The approach taken in this research was to produce highly intelligent process control systems by accurately modeling the nervous structures of higher biological organisms. The mammalian cerebral cortex was selected as the primary model since it is the only computational element capable of interpreting and complex patterns that develop over time. However the choice of the mammalian cerebral cortex as the model introduced two new levels of network complexity. First, the cerebral cortex is a three dimensional structure with extremely complicated patterns of interconnectivity. Second, the structure of the cerebral cortex can only be realized when thousands or millions of neurons are integrated into a massive scale neural network. The neural networks developed in this research were designed around the Hebbian adaptation, the only training technique proven by the literature review to be applicable to massive scale networks.

These design difficulties were resolved by not only modeling the cerebral cortex, but the process by which it develops and evolves in biological systems. To complete this model, an advanced genetic algorithm was produced, and a technique was developed to encode all functional and structural parameters that define the cerebral cortex into the artificial chromosome. The neural networks were designed by a cell growth simulation program that decoded the structural and functional information on the chromosome. The cell growth simulation program is capable of producing patterns of differentiation unique for any slight variations in the genetic parameters. These growth patterns are similar to patterns of cellular differentiation seen in biological systems. While the computational resources needed to implement a massive scale neural network are beyond that available in existing computer systems, the technique has produced output lists which fully define the interconnections and functional characteristic of the neurons, thereby laying the foundation for their future use in process control.

ACKNOWLEDGMENT

First and for most, thanks to my family for their endless support. I want to thank my mom, the "iron woman" for her gifts of tenacity, self-discipline, and unyielding energy. I also want to thank my mom for being strong and raising me to be strong, a gift that has allowed me to survive all that life has dealt me so far. My father, "the chief" can never be thanked enough, for it is he who has given me the gift of deep thought, and is primarily responsible for training my mind to think in the way that has allowed this work to be done. I also want to thank my best friend and brother, Alan, for everything, and mostly for always being there to catch me when I fall. Alan was the sound board for everything in this work, and provided creative insight that no one else ever could.

I would like to thank my committee for allowing me the opportunity to pursue this research, and being extremely patient in awaiting its competition. Special thanks to Professor Reasor for his prudent guidance, and for remaining my chairman under difficult geographic circumstances. Special thanks are also due to Professor Rees for encouraging me to pursue the theoretical aspects of advanced network design. Professor Watford cannot be thanked enough for her wisdom and honest counsel on every aspect of my life. I would also like to thank Professor Sarin for introducing me to many of the mathematical tools used in this study. I would also like to thank Professor Sullivan for introducing me to chaotic and fractal mathematics, both of which were extremely useful for analyzing the results produced by the simulation program.

Extra special thanks are due to Professor Lu for his friendship and countless hours of counsel. I would like to thank him for his assistance in developing my presentation skills and for the opportunity to work in his lab. I would also like to thank him for introducing me to Professor Spaepen and Joe Bell at Harvard and Dr. Krader at NIST so that I could complete the work on thin film thermocouples.

I would also like to thank Keith, Randy, and Jeff for giving me an unauthorized office in their machine shop, without which conducting this research would have been much more difficult.

Special thanks to Professor Susan Gerbi and Professor Jim McIlwain of Brown University, Alex Fair, Dave Shearer, Ross May, Dave McCarey, and Garrick Evans, for providing me with valuable information and direction during the development of this work.

Table of Contents

Table of Tables

Table of Figures

1. INTRODUCTION

This dissertation describes a design technique to produce neural networks that mimic the structure of the mammalian cerebral cortex. A computer based cellular automata simulation of cell growth has been created. The approach taken models and replicates the stages of development in order to produce the final structure. All automata instructions are stored in an artificial chromosome such that optimization and design alterations can be accomplished via traditional genetic algorithms. The development of the simulation program relied on detailed modeling of neurological anatomy. Although the mammalian cortex is more complex than traditional engineering neural networks, it is a working system with long range structure. It is the long range structure that is of particular interest because it allows the development of dramatically larger neural networks than what is currently possible.

This document is organized into five chapters. The first provides an introduction to the science of smart processing and briefly describes other current approaches to process control. The scope, research objectives, approach of this study, and synopsis of results are also discussed in the first chapter. The second chapter reviews work done in contributing areas up to the present time and provides the scientific foundation upon which this research was based. Chapter 3 establishes the theoretical background and mathematical representations used for the design and optimization technique. Chapter 4 describes the workings of the technique, the selection of cellular and cortical data, and the design of a genetic algorithm. In Chapter 5 the technique is evaluated and the behavior of the system is discussed.

1.1.1 The relations of neural networks to industrial engineering.

Perhaps the best place to begin the introduction is to explain why an industrial engineering study is focused on neural networks. It is the primary purpose of an industrial engineer to optimize all aspects of production, including management science, systems, human factors, or manufacturing. Manufacturing has four subclasses; machines, processes, materials and controls. Neural networks belong to the subclass controls. Neural networks are actually a class of smart systems, which is a class of adaptive controller (see Figure 1.1). While figure 1 may define the relationship of neural networks to industrial engineering, there are three other areas of confusion which must be addressed in this section.

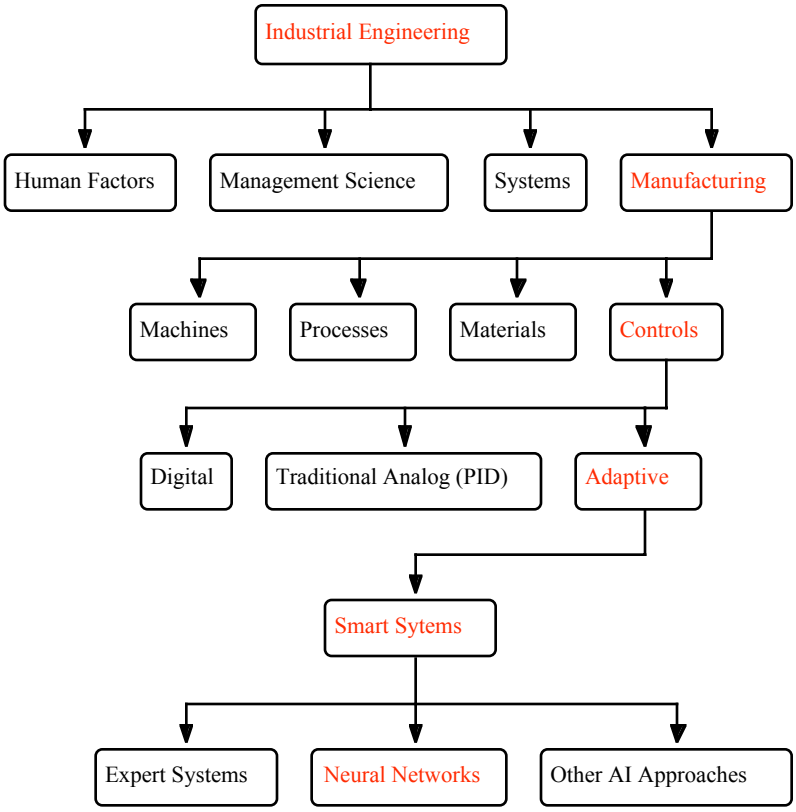


Figure 1.1 Relationship of neural networks to industrial engineering.

1.1.2 Association of Neural networks to Other Disciples

The first area of confusion concerns the relationship of neural networks to other disciplines. Neural networks are not unique to industrial engineering, in fact they enjoy membership to may other disciplines including; neuroscience, cognitive science, mathematics, and computer science. The first neural network was the product of a cross discipline study between a mathematician and a neurologist. Neural networks thus began as a mathematical representation of biological neuro-circuitry. Neural networks excel in handling ambiguous or fuzzy data and have since proven to be very useful for pattern recognition and control systems. Their early success and novelty has attracted researchers in nearly every discipline to adopt and apply neural network to their area. As a result, neural networks applications span nearly the entire engineering spectrum.

1.1.3 The relationship between engineering and biological neural networks

The second area of confusion is a gray line separating biological and artificial neural networks. Neural networks are mathematical models of neural circuitry. The neural network model describes functional elements (neurons) and a set of rule which define how the neurons may be connected. It is quite interesting that the McCulloch-Pitts model can be used to create systems that are not observed in biology. In other words the neural network model, though originally modeled from biology describes the super-set of all possible networks of which biological networks are a subset, (see Figure 1.2)

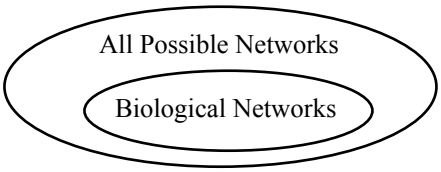


Figure 1.2 Relationship of biological networks to all possible networks.

1.1.4 The relationship between neural networks and other AI systems.

There are two basic types of artificial intelligence (AI) systems. The first attempts to replicate the behavior of intelligence while the second attempts to replicate the structure that produces intelligence. Expert systems are a good example of the first category. An expert system is a computer program that demonstrates a form of intelligence through a carefully constructed set of logical statements. Although an expert system is very useful, it is only a facsimile of intelligence. The second category includes neural networks. Neural network are a very good model of the actual circuitry of intelligence. It is hoped that by carefully replicating the neurological structure a truly intelligent system will be produced. The level of intelligence observed in neural networks does seem to increase with size and level of detail. This dissertation is but one more study that attempts to provide greater detail to the general network model.

1.2 Smart systems

This section provides an overview of the design of smart systems. A smart system is a modified control systems that utilizes one AI technique as the controller. There is a considerable discussion on smart controllers and smart processing in chapter 2. The purpose of presenting this material is to show the area of application for the neural networks developed in this study.

A smart system has two elements, a battery of sensors and a smart controller. There are two types of sensors, process condition and process state sensors, where the former report parameters related to the processing vessel and the latter detect parameters associated with the actual process, directly or indirectly. The smart controller, which is typically an expert system or neural network, will adapt the process variables, such as temperature and pressure, based on the sensory input, historical data, and programmed logic.

The main difference between smart control and previous control techniques is that part-specific optimization has been made possible by recent advances in process sensor technology and artificial intelligence programming techniques [1].

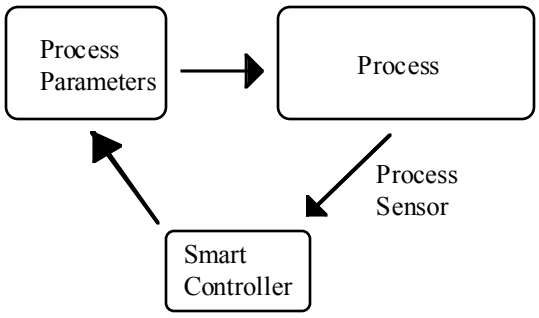


Figure 1.3 Schematic of smart controller closed loop architecture.

Smart processing is one of the most promising areas of materials processing research; however, the field is still in its infancy, and few guidelines have been established for designing smart systems. Currently, the development of smart systems is done on a case study basis.

1.3 Artificial Intelligence - General Definition

Artificial intelligence is the area of computer science which deals specifically with the development of adaptive "intelligent" systems. Artificial intelligence seeks to replicate the human ability to autonomously adapt and learn over time. At the present time, expert systems, genetic algorithms, and neural networks are considered by many authors [2,3,4] to be the most promising techniques. While none of these techniques rival the intelligence of the human mind, each has captured at least one aspect of the cognitive process. Following is an introductory discussion on each.

1.3.1 Expert Systems

The expert system is not the most recent, sophisticated, or advanced artificial intelligence technique, but it is the most successful, both in terms of providing a useful tool and in gaining widespread use [3,5]. Although expert systems are not used in this dissertation, the following brief overview is included since they are employed in many of the smart processing systems described in the literature review. An expert system is a software program that captures the working knowledge of a human expert and puts it in a format so that it becomes a tool for other people to use to solve complex problems and perform difficult diagnostics. The origin of expert systems can be traced to the problem-solving programming languages of the 1950's, such as LISP. LISP was designed to allow the programmer to treat objects as words and assign those words inherent meaning. The use of LISP led researchers to first develop the expert system.

An expert system has two separate and distinct parts, the knowledge base and the inference engine. The knowledge base stores the knowledge typical of an expert, and the inference engine contains the logical operations that allow inferences to be drawn from the knowledge base. The most clever aspect of this structure is that it allows the programmer to add and delete knowledge without affecting the logic of the program. The development of the expert system's architecture allowed the management of a very large knowledge base. Expert systems solve problems by making inferences about the present situation by referencing their knowledge base. The limitation of expert systems is that the time to search through the knowledge base is somewhat stochastic, meaning that the number of inferences to be drawn is unknown at the initiation of the search. As the complexity of the logic increases, the number of logical operations needed to define the problem grows exponentially. Expert

systems are especially well suited for diagnostic work; however, the stochastic nature of the search makes them ill-suited for control applications where it is very important to know the length of a data processing procedure.

Most modern expert systems utilize fuzzy factors to contend with the issue of partial membership. Partial membership describes the condition where a the state of a given entity can belong to two or more classes. For example, a cut apple can be described as: an apple, a slice, or a half of an apple. Fuzzy factors have enabled expert systems to deal with problems where exact information is not known, but they have also greatly increased the amount of programming time required. To program fuzzy factors, it is necessary to evaluate the responses of the expert system against the responses of an actual human expert, then iteratively adjust the factors until the two responses equal. Not only do fuzzy factors increase the amount of programming time, they complicate information modification because the addition or deletion of knowledge shifts many, if not all, of the fuzzy factors.

Expert systems are especially useful in areas where the knowledge base is closed and very specific. Campbell's Soup Corporation developed an expert system based on one human expert who could diagnose problems with their large soup vats. The Campbell's Soup expert system performed very well because the knowledge of the human expert was focused only to a particular type of very large soup vats.

When the knowledge base is expanded, the efficiency and performance of the expert system drops dramatically. Current research is being directed to developing multi-component expert systems such that each subsystem is specialized and capable of redirecting the search appropriately. The overall system produces a tree-like structure. In theory a tree structure can be expanded to a high level of complexity. Unfortunately, these systems quickly grow to unmanageable proportions [3].

In summary, the four principle limitations of expert systems to large control applications include the stochastic nature of the search process, the exponential increase in search time with an increased knowledge base, the complex interconnectivity of the knowledge making modifications of large databases complicated, and the inability of the human expert systems to deal with nonspecialized data.

1.3.2 Genetic Algorithms

A genetic algorithm is a pseudo-random search technique based on the process of evolution. Genetic algorithms borrow features from both nearest neighbor and random search techniques. Genetic algorithms have proven most productive in the solving of complex or NP hard problems where their pseudo-random, "survival of the fittest" search methods can outpace traditional search techniques [6].

The process of evolution is based on slight mutations of the large DNA molecules called chromosomes. Since DNA molecules serve as the blueprints for all living organisms, any mutation of an expressed gene will manifest itself as a physical alteration of the organism. In nature these mutations may be benign, advantageous, harmful, or even fatal. Traditional genetic algorithms are based on the concept that mutations lead to optimization through evolution.

A genetic algorithm has three main parts: the chromosome, the fitness function, and the breeding algorithm. The chromosome is divided into a series of genes. Each gene is used to represent one of the possible solution values. Consider, for example, a genetic algorithm created to solve a triangulation problem in three space. In this case the chromosome would have three genes, representing the **X**, **Y**, and **Z** coordinates. Each coordinate is represented as a binary string such that the entire chromosome is an extended binary string. The second part of the genetic algorithm, the fitness function, evaluates the chromosome with respect to a particular desired outcome. In the triangulation problem, the fitness function could simply be the root mean error from the target goal. The last part of the genetic algorithm, the breeding algorithm, selects the fittest individuals from the population and then breeds them to create a new population. During the breeding process, some of the genes of the parents are interchanged to produce new, unique genes and offspring. Also during breeding, mutations are introduced to the chromosomes to further diversify the new generation. Several types of mutations can occur, such as total inversion, where all 1's are changed to 0's and vice versa, and partial crossover, where the **Z** coordinate gene could be interchanged with the **X** coordinate gene of a given chromosome. The breeding process is much more complicated than described here, and there are many other types of mutations to be described in a later section.

There are two main disadvantages of genetic algorithms; the stochastic nature of the search and the dependence of the rate of convergence on mutation parameters. Due to the random nature of the mutation process, it is impossible to determine when the algorithm is

going to converge on the solution. Since the problem space is typically large for genetic algorithm applications, it is unlikely that the true optimum will be found, rather a state of convergence is accepted when the overall population has a high average fitness value. Due to the random nature of the algorithm, it is not possible to determine how many generations should be produced before the desired level of convergence is obtained. The literature is somewhat misleading because authors often apply genetic algorithms to very simple problems for demonstration purposes, then report the rate of convergence for that particular problem as if it were the rate of convergence for that genetic algorithm applied to the general case. Since the search time is stochastic, genetic algorithms are ill-suited for control applications where the cycle time of each process must be known.

The second problem with genetic algorithms is that the rate of convergence is dependent on the specific problem and the parameters used to describe the mutation rates. In the literature the rate of convergence is usually reported as a function of the mutation parameters. At the present time, no theory exists to predict which mutation parameters will produce the optimal convergence to a given problem before a solution attempt is made. Because of this problem the development of genetic algorithms is a specialized art that produces an optimal algorithm for a very narrow class of problems.

1.3.3 Neural Networks

Neural networks are parallel data processing structures that are modeled after the nervous system of higher mammals. The basic computational element of the mammalian system is the neuron (Figure 1.4). A neuron is a completely separate data processing element that has no direct physical contact with any other neuron. Neurons are however electrically connected by synapses to other neurons. The artificial neuron is modeled after the biological neuron and has three basic elements:, an axon, an axon hillock and dendrites [2]. Input signals in the form of voltages are received at the dendritic synapse. The size of the synapse is generally assumed to determine what fraction of the incoming signal will be transmitted to the neuron. In essence, the dendritic synapse is both a connecting point and a weighting function.

Each neuron has numerous dendritic synapses. The voltages from all the dendrites are summated in the axon hillock. The axon hillock takes this total value, amplifies it by an amount of gain specific to the neuron, then transmits this voltage through the axon. The artificial neuron is modeled such that all of the axial synapses have the same value, which is equal to the output of the axon hillock. The axial synapses connect to the dendritic synapses

of other neurons. This brief description of the neuron is incomplete and is presented here only for reference. Refer to section 2.3.2 for more details.

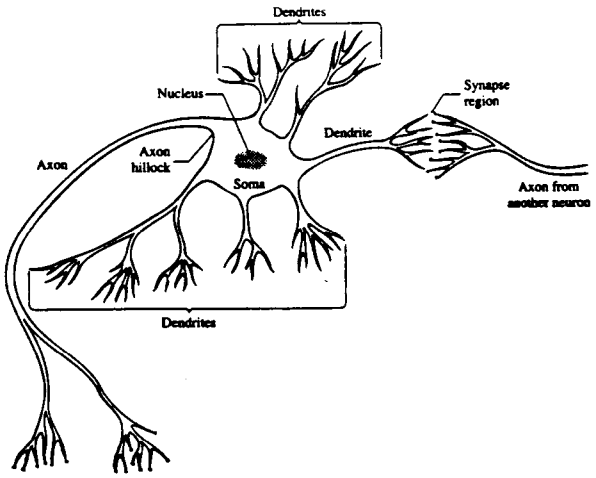


Figure 1.4 Simplified biological neuron showing key elements [2].

In an artificial neural network, the neuron is replaced by a node that sums the weighted average of the input signals from the other nodes to which it is connected. The neurons are typically arranged in a three layer array where the top layer is connected to the inputs and the bottom layer corresponds to the output vector. This type of network is commonly referred to as a pattern classifier where different patterns of the input layer are mapped to a specific output node. The mapping of the input to the output involves as a complex interplay of the different synaptic values that interconnect the nodes. While there are some hardware applications, most artificial neural networks are emulated on computers using a matrix to store the synaptic weights and linear algebra to solve for the output vector.

The main advantage of neural networks over other systems is the high degree of parallelism that allows very rapid calculations. Neural networks have demonstrated great proficiency at rapidly handling complex and incomplete data [2,7]. The speed of neural networks, combined with their well-proven abilities to process noisy data, makes them a good candidate for control applications. Unfortunately, the current state of neural network technology only allows for relatively small neural networks to be produced. It is well known from neuroscience that the intelligence of a network, biological or artificial, increases with the number of neurons and synapses. While most engineers practicing neural network development would like larger networks for their increased intelligence, no one has yet proposed a technique to allow the design of a massive network.

There are three main challenges to the development of massively large neural networks: economically feasible hardware, training techniques, and modifiable architecture. In the past five years network designers have been aggressively designing very large scale integration, (VLSI), circuits for neural networks. The area of VLSI neural chip development is growing rapidly and will likely provide the necessary hardware needed for massive networks; however, there is no feasible model to describe how these elements can be integrated into a massive network. The training of a massive network presents a difficult challenge. The most successful technique, backpropagation, is iterative in nature and thus training time increases exponentially with respect to the size of the network. This makes backpropagation unfeasible for massive networks. The second training technique, referred as Hebbian adaptation, is not upwardly bounded by network size. Hebbian adaptation, described further in section 2.1.3.2, is the actual process by which biological neural systems learn. The difficulty in implementing Hebbian adaptation is that it introduces a much greater computational load on the individual neurons. With the Hebbian technique, training is accomplished by modifying a preexisting neural pathway, implying that certain fundamental pathways must exist in the network at its inception. The need to use Hebbian adaptation for massive networks provides an added impetus for developing a technique to define the architecture of a massive network in a controlled and predicted fashion. The development of a generalized neural architecture has not yet been successful because a fundamental understanding of the biological nervous system has been absent in engineering design. By most engineering standards, the nervous system of living organisms is a chaotic mass of interconnected neurons; therefore, very few attempts have been made to replicate its structure accurately.

In summary, the three most common artificial intelligence systems are expert systems, genetic algorithms, and neural networks. Expert systems are extremely useful in situations where information is very specific, however they have practical limitations when attempts are made to upscale or increase their size. The genetic algorithm is a very powerful optimization technique for multi-parameter problems but is not particularly suited for control applications. Neural networks have the most potential for the real-time control of complex systems. However the development of the next generation of large scale neural controllers is limited by the absence of a technique to design the network architecture and a fundamental lack of understanding of how signals are processed in large scale neural networks.

1.4 Research Objectives

The purpose of this research is to develop a technique for the design of massive scale neural networks. This study enhances the current neural network model by adding greater detail and defines a set of rules that allow long range structure to be realized. It is the goal of this work to develop a design technique that is robust enough to be used in many application including smart systems.

1.5 Scope of Research

This dissertation investigates smart control systems and neural controllers, identifies the limitations and deficiencies of those systems, and proposes an alternative in the form of massive scale neural networks. In Chapter 2, it is shown that the fundamental limitation of neural controllers is in fact a fundamental limitation of neural networks. Neural networks were developed in the 1950's, based primarily on the nervous systems of lower organisms such as *Aplysia Americana.* Not only is the nervous structure of lower organisms less complex than those of higher mammals, but the computer and mathematical limitations of the equipment of the 1950s forced substantial simplifying assumptions to be made in order to develop a working model. Advancements in the fields of neural, cognitive, and computer sciences now allow a far more detailed and accurate model of the neuron and neural structures. While the body of knowledge surrounding neurology has grown since the 1950s, it is not yet complete. There are many parameters not yet quantified that are needed to fully describe the structures we seek to replicate. To overcome this limitation, the approach was taken to develop a system with the flexibility and the capacity to allow the easy manipulation of the unknown neural, and cortical values. Genetic algorithms were used to maintain and optimize the unknown neural parameters based on their success in other areas of multi-parameter optimization. By encoding all of the parameters that define the structure of the neural network in the form of a genetic algorithm, it was possible to optimize the design through the known process of artificial genetic operations.

To convert the neural parameters into the structure of a neural network, a cell growth simulation program was developed. The simulation program is based on von Neumann's automata. Von Neumann's simple chromosome was replaced with the complex chromosome that contains the neural parameters. The simulation program also differs from von

Neumann's work in that multiple stages were utilized to fully define the final neural network structure.

By combining the power of genetic algorithms to optimize multi-parameter systems with the self-organizing abilities of automata, a system has been produced that generates output structures with the diversity and the chaotic uniqueness found in natural systems. In other words, the system that has been developed can produce neural networks with the specificity and complexity similar to their biological counterpart. The development of an advanced genetic algorithm and the automata programming relied heavily on mathematical modeling of biological systems.

1.6 Accomplishments

This dissertation has produced a complete technique for designing and optimizing massive scale neural networks. To develop the technique, a mathematical representation of neurological anatomy, an advanced genetic algorithm, and an automata based cell growth simulator were produced.

The power of the technique is demonstrated and can be evaluated by the high degree of control of the output structures that is possible through simple and minute alterations of the chromosomal parameters. The chaotic nature of the patterns attests to the validity of the model. The technique developed in this dissertation produces neural network structures that closely resemble the organization of the cerebral hemispheres of the brains of higher mammals. The degree to which they are similar is a measure of the effectiveness of this study. Yet another measure of the work done in this dissertation is the general applicability of this technique to replicate nearly any functional region of a mammalian brain simply by varying a few of the parameters in the artificial chromosome.

In summary, the development of an adequate neural controller was limited by deficiencies in the previously existing neural network model. This dissertation has proposed a solution to those limitations by producing an advanced neural network model based on updated neural, cognitive, and computer science technologies. The technique developed in this study is capable of replicating the higher processing regions of mammalian nervous systems. This technique allows a high degree of adaptability and variability in the resulting network structures which is necessary to allow future work to be directed towards developing specific neural controllers. By encoding structural parameters into an artificial chromosome, this technique has an embedded method of optimization.

The technique developed in this dissertation is based on a very broad body of knowledge. Several areas of science and technology must be reviewed before the individual steps and components of the technique can be understood. In the following chapter a historical foundation for this study is presented.

2. HISTORICAL REVIEW

The technique developed in this dissertation for designing massive scale neural networks shall be referred to as simulated growth for convenience. Simulated growth is an involved process that borrows concepts from general neural network theory, neural science, genetics, cognitive science, and automata programming. There is one section devoted to each of this five primary topics. There is an additional section on smart processing. The topics are arranged logically with respect to understanding the simulated growth technique. First, neural networks will be discussed in order to examine the capabilities of that technology. There are four primary focuses of neural networks: theoretical operation, network architecture, training techniques, and hardware implementations. The second section discusses smart processing techniques that have been implemented in industry. It also demonstrates how neural networks are utilized in the real word. The third section reviews neural science and contains three subsections including discussions on the neuron, the brain, and mapping. The review of neural science presents the additional neurological information that has been used to enhance the traditional neural network model. The fourth section discusses genetics and provides a framework for the advanced genetic algorithm. The genetic algorithm is used to store and optimize the neural network structural parameters. The process of biological development is also reviewed in the fourth section and serves as the model used to design the growth simulation program. The fifth section discusses recent contributions made in the area of cognitive science. Cognitive science describes how information is stored and represented in the brain and is the foundation of the working dynamics of this study. The final section contains a brief discussion of automata theory upon which the growth simulation program is based.

2.1 Neural Networks

Neural networks were first developed in the late 1940's and 1950's. A neural network is a computational device modeled after nervous tissue in living organisms. The primary advantages of neural networks are that they process data in a parallel fashion and allow for a great deal of overlap or "fuzziness" in the data. These advantages make neural networks well suited for rapid processing and pattern recognition applications. A neural network is comprised of simple summing devices called neurons or nodes. The neurons can be arranged in a variety of patterns as will be discussed in the following section. The overall performance and capabilities of a network are directly related to the network's architecture,

the number of neurons, and the degree to which they are interconnected. As the technology associated with the neural networks has progressed with time, the size and complexity of the architecture has also increased, as documented in the following section. Both computational, derived purely from mathematics, and evolutionary, heavily influenced by biological systems, networks are discussed. Neural networks can be divided into two general classes separated by the technique by which they are trained. The most commonly employed technique is backpropagation which calculates adjustments to the network based on the output error. The second form of training, referred to as Hebbian adaptation, adapts the network only when associations are made. Both training approaches are discussed, as is their applicability to massive scale neural networks. While the majority of neural networks exist as software, there is a growing trend to develop high speed hardware neural networks and several examples are cited. The last part of this section describes neural controllers and smart process controllers, giving special attention to those directly related to composite manufacture.

2.1.1 Definition of a Neural Network Layer and Three Layer Networks

Neural networks, regardless of their complexity or designs, perform one basic function, which is to correlate some input to an output. Clearly, a neural network must accept input. Therefore, the network must have a functional region that is designed to receive input. Once the input signal enters the network, it is process in some meaningful fashion before it is passed out of the network. The region where the signal is processed is called the hidden layer. The hidden layer passes the result to the third, and final, output layer.

Note that the description of the layers of the neural network is completely independent of topology. The topology of each layer may be very complex or extremely simple. Typically a layer is represented as a one-dimensional array of neurons. This is a simplification that has lead to some confusion. There is no actual requirement that a layer has to be a one-dimensional array. If the definition of the hidden layer is expanded to represent that portion of the neural network where the complex relationships and interdependencies are represented then all networks may be described as three layer networks. Essentially, the term layer needs to be broadened to describe functional regions as opposed to topological region.

To illustrate this point, consider an extremely complex three layer neural network such as the human brain, (Figure 2.1). The human brain has an input layer, which is comprised of five cortexes, one for each of the senses: sight, sound, tough, taste and smell. These five cortexes stimulate the hidden layer, which is the bulk of the brain's 1.5 Kilogram mass. This hidden layer is also divided up into numerous cortexes; each specialized to form a

specific class of association. Of course, the brain also has an output layer that drives motor functions including speech.

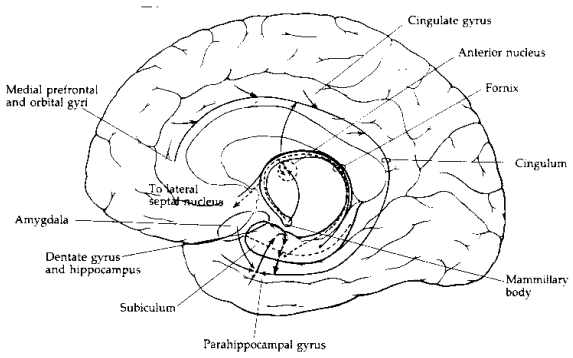


Figure 2.1 A complex three layer neural network

Although this extended definition of "layer" facilitates the description of the massive scale neural networks described in this paper, it does somewhat complicate the description of existing neural networks. The confusion is created by authors who use the term "layer" interchangeably with "array" to describe a topological domain. There are networks that have less, and more than three arrays of neurons. In the case where there are more than three arrays, all those layers which do not perform input or output functions may be clumped into a large hidden layer. The ambiguity associated with fourth and fifth arrays, typically called layers, will hopefully be resolved by the publication of this work.

In the case where there are less than three arrays the broad definition still holds, however the application of the three-layer model is less intuitive. With these simplified networks the functionality of the layers is combined through a process akin to Boolean reduction. However, these simplified networks are of little interest with respect to the body of this work. This dissertation attempts to model complex hidden layer structure of the mammalian brain. In essence this work adds detail to the existing model.

2.1.2 Classical Neural Networks

This first section on neural networks begins with McCullough and Pitts and the first few successful neural network applications. As the discussion progresses, more advanced architectures are introduced, and finally evolutionary networks are described.

McCullough and Pitts [8] were first to describe the neuron in mathematical terms.

The McCullough-Pitts neuron is a simple summing device whose output is the sum of the weighted inputs multiplied a sigmoidal function. The value of neuron 'i' is transmitted (transferred) to neuron 'j' by multiplying the value of i by the synaptic weight w_{ij} that connects the two neurons. The value of the synaptic weight in an artificial neural network typically ranges between 0 and 1; however, there are many ways to mathematically implement weighing functions. A neuron will not transmit (fire) unless its internal value is above a given threshold value. The internal value of the j^{th} neuron in a network, $\_j$, is given by:

$$\_j = \_ w_{ij}x_i \tag{2.1}$$

where "i" denotes the neurons connected to j, and x_i is the value or voltage of node i. The output value of the j^{th} neuron is given by:

$$x_j = (1 + e^{-k\_})^{-1} \tag{2.2}$$

where k is a shape parameter. The sigmoidal function $(1 + e^{-k\_})^{-1}$ is used to model the actual transfer function of biological neurons. The complete artificial neuron is given in Figure 2.2.

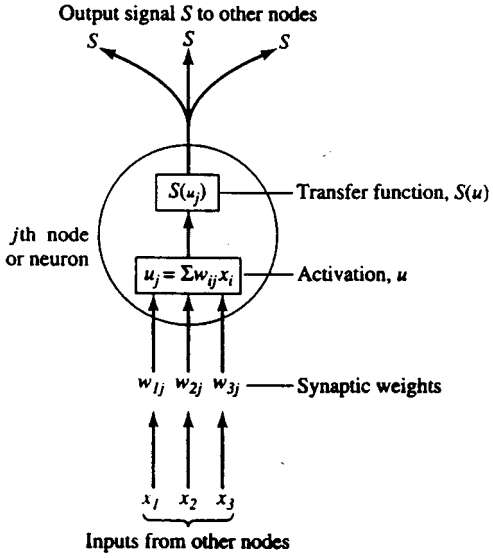


Figure 2.2 Schematic representation of the artificial neuron [2].

One of the first successful forward feeding neural network was the Perceptron. Rosenblatt made his Perceptron from vacuum tubes and servo motors. The synaptic weights

were adjusted by servo motors which turned resistors (a common analog computer technique), and the summing and multiplication were done by vacuum tube amplifiers. The Perceptron used a crude Hebbian rule for learning, (section 2.3.2) and is credited by most authors as being the first learning machine.

2.1.2.1 One Hidden Layer

The most common type of neural network is the forward feeding, three-layer network. The first, second, and third layers of the network are referred to as the input, hidden, and output layers, respectively. Each layer contains several nodes or neurons, the number of which may be different for each layer and depends on the design. The values of the input layer, called the *input vector*, collectively form a [1 x N] matrix, where N is the number of nodes on the input layer. Sometimes an additional bias node is included in the input node to give a [1 x (N+1)] input vector. Likewise, the values of the output layer are referred to as the output vector.

The most common use of three-layer neural networks is pattern recognition [6]. Pattern recognition involves mapping the input vector onto the output vector. Usually each node of the output layer represents a unique class. The recognition involves determining which input node, or combination of input nodes, most closely represents a particular class. Neural networks are especially well suited for this role because no knowledge of the relationship between the input vector and the output vector is necessary at the onset of training.

Three-layer, feed-forward neural networks are usually fully interconnected, meaning that each node on one layer is connected to all of the nodes on the next layer. A given output node, which represents a class, is connected to all the nodes in the hidden layer, and thus all the nodes in the input layer. For a three-layer network, each output node has (m x n) + m synapses, where m and n are the numbers of nodes on the input and hidden layers respectively. Through complex interplay of all these synapses, virtually any input pattern can be mapped to the output layer.

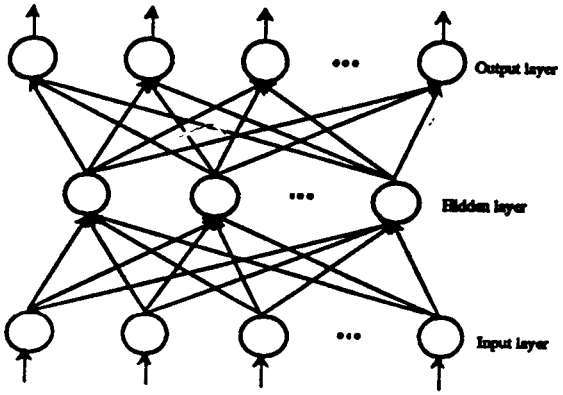


Figure 2.3 Generalized three layer feed-forward neural network [13].

The three-layer forward feeding neural network is by far the most common because it has a general architecture that performs well for most applications [7]. Three-layer neural networks have been used to perform a variety of pattern recognition tasks including infrared (IR) spectroscopy (Figure 2.4 [9]) and voice recognition. Although it was once thought that the three-layer architecture was limited to relatively small networks, Morgan has produced a speech recognition network with 4,000 nodes in the hidden layer [10].

The fully-connected architecture that has been described so far can be modified by permanently setting some of the synaptic weights to 0, thereby effectively eliminating their connections. Another variation of the theme suggests a network with constant synapses and variable gain. Codrington and Tenorio [11] demonstrated an adaptive gain system on a basic, three-layer neural network. Although this is an interesting idea, the paper did not indicate that it performed particularly well. The novelty of this idea is somewhat watered down by the fact that mathematically there is no difference between varying the gain and varying the synaptic weights [12].

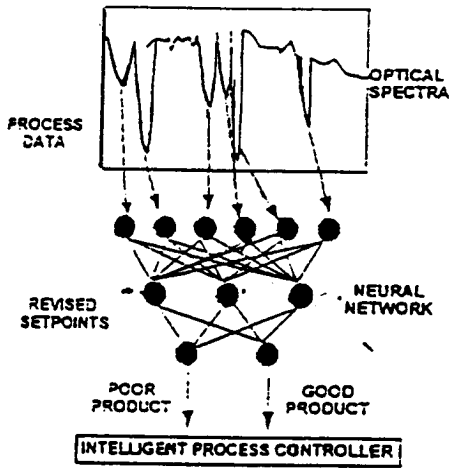


Figure 2.4 Three-layer, forward feeding neural network used for IR spectroscopy [9].

2.1.2.2 Computational vs. Evolutionary Neural Networks

There are two basic classes of neural networks, those produced for engineering purposes and those produced to model biological systems. The first class is referred to as computational networks, and the latter is referred to as evolutionary networks, although they do not evolve in any sense of the word. The vast majority of computational networks are derivations of the Perceptron with only a very small percentage of academic-based networks having architectures with unique origins. Computational networks are designed for specific pattern recognition problems where on-the-job-training is not performed. It is commonplace to discuss a computational network's efficiency in terms of computational power versus network size, and to optimize it towards that goal. Evolutionary networks on the other hand, are much closer representations of biological nervous tissue. They are used for modeling dynamic learning behavior.

The vast majority of computational networks are mathematical emulations performed on computers. These emulations often utilize a matrix that contains the synaptic values of the different neurons. An input vector is crossed with the synaptic weight matrix to produce the output vector. In his book, Freeman [13] demonstrates several neural network designs using Mathematica®, and discusses such issues as training time, convergent rates, and overall performance of similar networks on different problem types.

2.1.2.3 Hierarchical Networks

Hierarchical networks are actually a series of networks set up in tree formation to allow the analysis of a problem beyond the scope of single sub-network. The networks are usually not connected in a hard sense, but rather the results direct the computer program running the emulation towards another network. Hierarchical networks are computational networks that are quite useful from an engineering point of view, but have no biological counterparts. This approach was used by Paradis and Dietrich [14] when they combined a recurrent cascade network, a Kohonen network, and a recurrent backpropagation network.

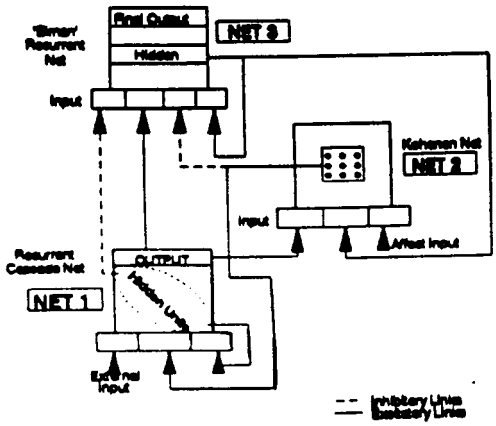


Figure 2.5 Schematic illustration of the architecture of the hierarchical network developed by Paradis and Dietrich [14].

2.1.2.4 Modular Networks

A modular network is a modification of hierarchical networks that directly "hard" connect the subnetworks into one massive whole. Proponents suggest that this method of network design offers greater power with less training. Wan, Low, Lau, and Lui demonstrated this architecture on a speech recognition network [15].

The best example of a modular network was developed by Taber and Diech [16,17] for the identification and classification of underwater sounds. They developed a system that was capable of identifying a short sampling of a cyclic pattern taken anywhere in the cycle. Their technique utilized a series of neural rings that, unlike traditional networks, have no start or end (Figure 2.6). The rings were divided into 20 sectors that each contained 60 neurons, one each for the 60 frequency bands under investigation. When an echo was detected, it was broken into the 60 frequency bands and then used as input for the sectors of all the rings. If one of the sectors identified a 1 millisecond clip, it would pass a positive response to the next

sector and so forth. When several consecutive sectors made a positive identification, it was believed that the ring had identified the source that it was trained for. The rings were fed into a larger network that would evaluate the situation when two or more rings returned a positive identification. Each ring was trained separately and could be removed or added to the system with little additional work.

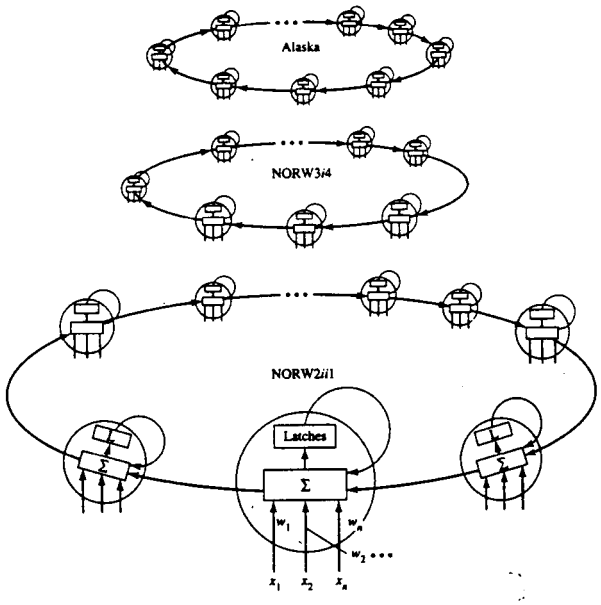


Figure 2.6 Modular network of Tabor and Dietch. Note Alaska, NORW3i4, and NOR2ii1 are sound source types that the network was trained to detect.

2.1.2.5 Binary and Bipolar Networks

There is a small class of networks that appears in the literature known as binary networks. Binary networks were first developed from a misconception of the nature of the threshold mechanism in the McCullough-Pitts neuron. McCullough and Pitts modeled their neuron after its biological counterpart, but the model is incomplete and neglects the current-to-voltage and voltage-to-frequency conversion (section 2.7) where a neuron stores a voltage up to a threshold value before discharging a pulse. The rate at which the neuron will pulse is proportional to the current, but below the threshold voltage, no pulsation will occur [27]. Without this information one can observe the equations in the McCullough-Pitts model and realize that by modifying the shape constant k in the sigmoidal function and raising the threshold value, it is possible to produce neurons with a binary response. Binary networks have little correlation with biological systems but they have proven to be useful engineering

tools. Binary networks are commonly converted to bipolar networks by converting the 0's to -1's. Kosko [18] reports that under nearly every circumstance, bipolar networks converge more rapidly. The ART and BAM paradigms (section 2.1.2.7) are good examples of binary networks.

2.1.2.6 Time-Sequence Networks

Many patterns of interest, such as cure cycles, are difficult to recognize from a frozen frame perspective. Neural networks can be modified to capture elements of time by adding recurrent or context vectors. Recurrent vectors are the vectors from previous iterations that are reprocessed in the current iteration. A crude Elman network is illustrated in Figure 2.7.

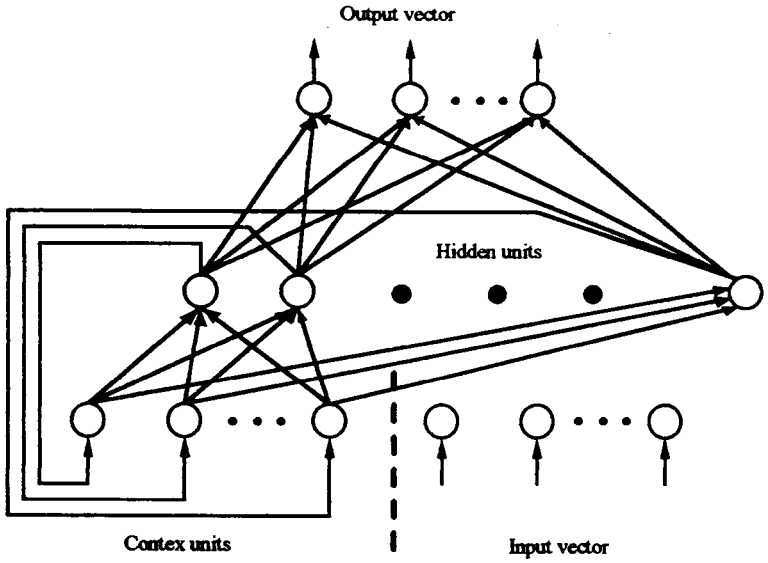


Figure 2.7 Schematic of a basic Elman network, Freeman [13].

Recurrent vectors may be the previous inputs, outputs, or in the case of the *Elman* network, the hidden vectors. Freeman [13] reports the most rapid convergence when using the hidden vector as the recurrent vector, although the reasons for this are unclear. By using multiple recurrent vectors Elman networks can recognize a pattern developing over time. Consider the case were there are *n* patterns to be recognized. In this case there would be n-1 recurrent vectors for periods (t-1), (t-2), (t-3), ..., (t-(n-1)), and one input vector for period (t). The structure will allow the Elman network to differentiate between the sequence 1,2,5,2,4 and 3,2,5,2,4 even though the four most recent input patterns are identical.

2.1.2.7 Evolutionary Networks

There are four popular network architectures based on biological models: ART, BAM, CMAC, and the *neural-solver*. ART and BAM address the issue of producing internal representations as activity patterns between groups of neurons, while the CMAC, and neural-solver models demonstrate that replicating neural structures replicates their functionality.

2.1.2.7.1 Adaptive Resonance Theory (ART)

One of the main deficiencies of traditional forward feeding neural networks is that once an input vector is placed in the input layer, the network will classify the entity, and yield its 'best' guess as the output vector. Grossberg [19] recognized that people are often able to interpret otherwise ambiguous information. Grossberg presumed that somewhere in one's mind is a representation of the idea that is being presented, and this representation is produced by activity patterns between groups of cells. Since a significant amount of data compression occurs in the translation of sensory input to brain representation, it was presumed (correctly) that errors could occur when the brain representation is decompressed. If the brain's representation is decompressed to be in the format of the input, then the two representations should be equal, under which condition, the two activity patterns should resonate. If the two representations are not equivalent, no resonance will occur, and thus a error has been made and the network should make alternate selections until the correct classification has been made. ART is designed to produce *grandmother cells*, i.e. single output nodes representing an entire class, a feature desirable only from an engineering point of view.

The architecture that Grossberg [19] eventually developed is called ART, for adaptive resonance theory. ART employs three elements, two networks and a logic node (Figure 2.8). The two networks, F^x and F^y represent neurons close to the sensory input and the neurons in the brain which produce the brain representation, respectively. The logic element, called the nonspecific arousal signal, is inhibited from functioning except when F^y returns a mismatch signal. Upon receipt of a mismatch signal, the nonspecific element will send a signal to F^y to suppress the pattern and allow for error correction (Figure 2.9).

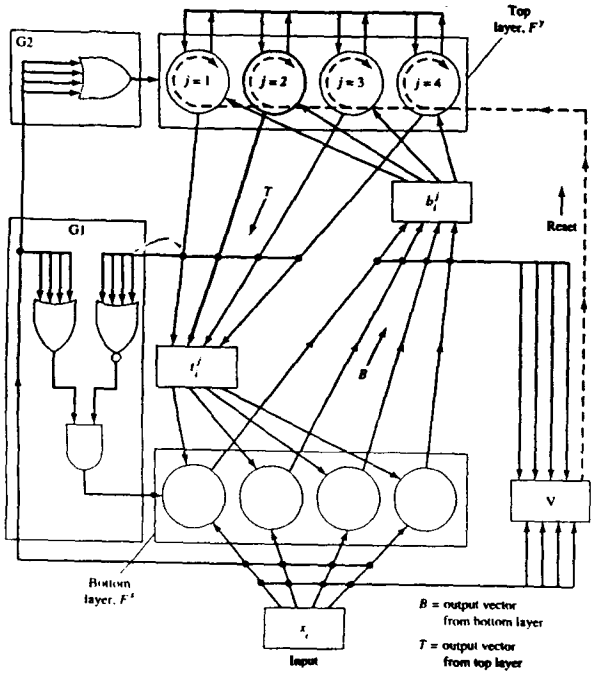


Figure 2.8 Schematic of an ART layout. Note G1 and G1 switch between F^x and F^y and V is the nonspecific arousal logic element [2].

There are actually four ART paradigms: ART1 is a binary network, ART2 is an analog network, ART3 models chemical gating, and ART4 employs fuzzy sets. Only the first two are commonly referred to in the literature. The ART architecture has proven itself to be a powerful pattern classifier, especially for analyzing spectrographic data. ART systems have been developed for a range of problems, including monitoring the condition of drill bits based on ultrasonic data [20]. Heileman, Georgiopoulos, and Hwang [21] give a review of the learning capabilities of ART1 networks.

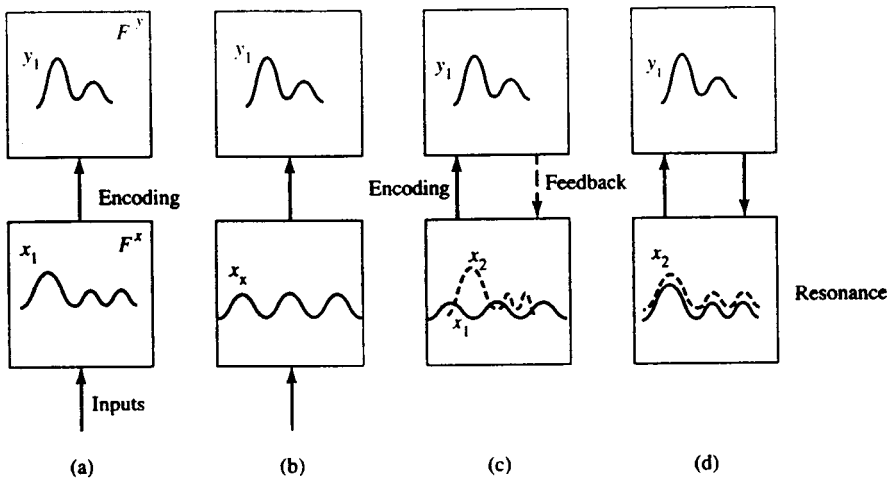


Figure 2.9 Correcting false coding. (a) correct coding of x_1 onto y_1, (b) x_x incorrectly coding to y_1, (c) F^y decodes y_1 and sends the signal to F^x, which does not resonate, and (d) resonance when decoded y1 is equal to x_1 [2].

2.1.2.7.2 Bi-Directional Associative Memory (BAM)

BAM, was developed by Bart Kosko [18] as a simpler version of the ART architecture. BAM, like ART, represents stored memories as activity patterns between cells; however, BAM does not require the nonspecific logic element. The BAM architecture relies on the designer to control the learning. BAM was designed to allow the association of patterns in either direction with respect to the sequence. For example, if pattern A was memorized to initiate pattern B, B would also initiate pattern A. The BAM model captures the human ability to cross-reference information. The original BAM paradigm was bipolar and only employed two vectors. Input vectors are placed in the network which then begins to pass the information cyclically between the two layers (Figure 2.10) until the system stabilizes, at which point the output can be read.

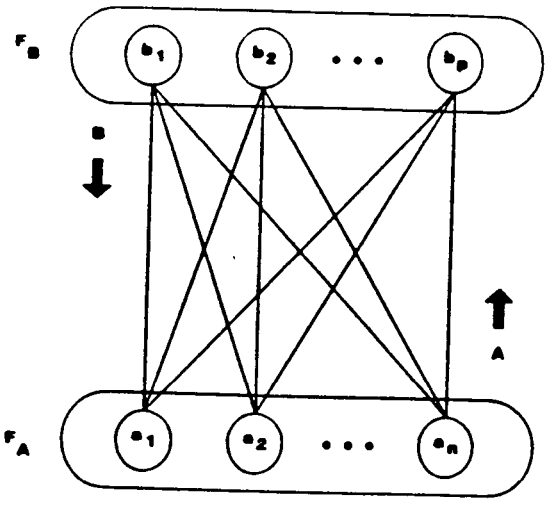


Figure 2.10 BAM architecture as described by Kosko. Note the similarity to ART with the exception of the nonspecific arousal element [18].

2.1.2.7.3 Cerebellar Modular Articulation Controller (CMAC)

The CMAC is designed to replicate the ability of the cerebellum to rapidly coordinate the multi-axis motion of the motor systems of biological organisms. The cerebellar cortex, the center of the brain responsible for motor coordination, has two specialized cells, Purkinje neurons and horizontal neurons. The Purkinje neurons, which provide stimuli for motor neurons, have numerous dendrites in one plane, giving the appearance of a sheet. The Purkinje cells are arranged in the cerebellum with their dendrite sheets aligned in parallel, such that staggered stacks of dendrite sheets are created, (Figure 2.12a). These stacks are interconnected by the horizontal cells. The parallel fibers form *beams* such that the Purkinje cells on the beam are excited, while those off-beam Purkinje cells are inhibited by *basket cells*, (Figure 2.12b).

Since there are many different paths that a multi-jointed appendage can use to move its terminus between two points and only one path can be used, Pellianisz and Llinas [22] speculate that the massive number of interconnections allow the cerebellum to convert a covariant sensory signal into a contravariant motor control signal (Figure 2.12). Albus [23,24] proposed that this structure could be approximated by stacking a series of Perceptrons (Figure 2.14). He developed a general algorithm which has since gained wide acceptance as a neural controller and many researchers have published work using it. The CMAC is well suited for complex motion, such as the control of a 10-axis biped, walking

robot [25] (section 2.2.2). In addition to motion control, the CMAC has proven to be useful in signal processing, such as color correction [26].

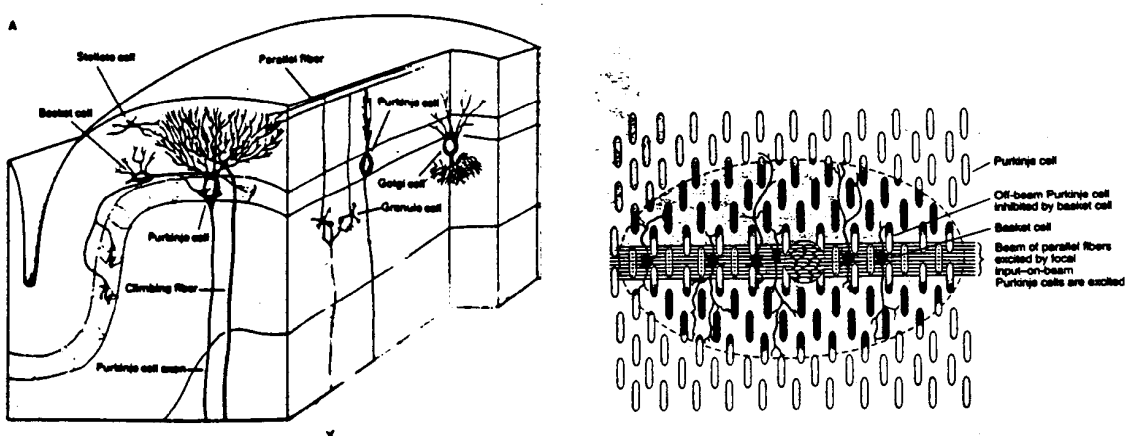


Figure 2.11 Anatomical structure of the cerebellar cortex. **Left**, longitudinal and transverse cross section of the cerebellar cortex showing basic cell types and three stratum structure, and **right** schematic representation of excited on beam, and inhibited off-beam Purkinje cells [27].

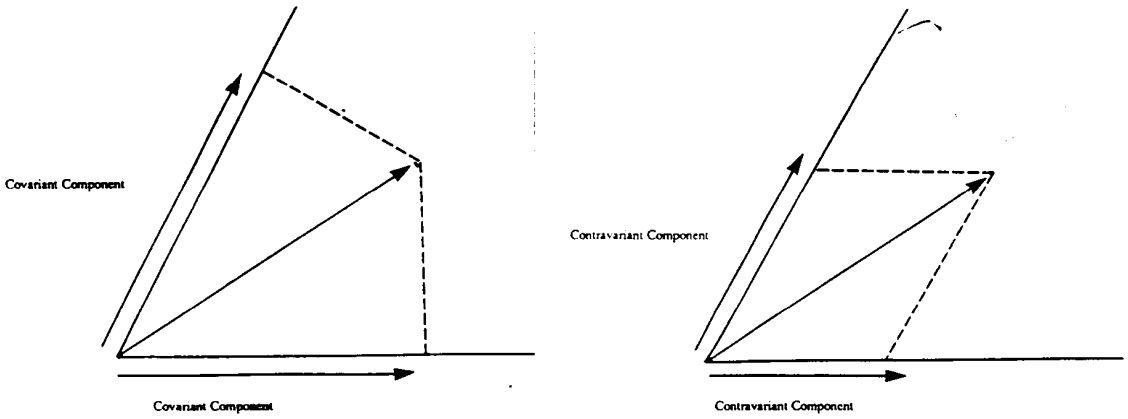


Figure 2.12 Graphical illustration of **left** covariant and **right** contrvariant vectorial representation of the same motion path [28].

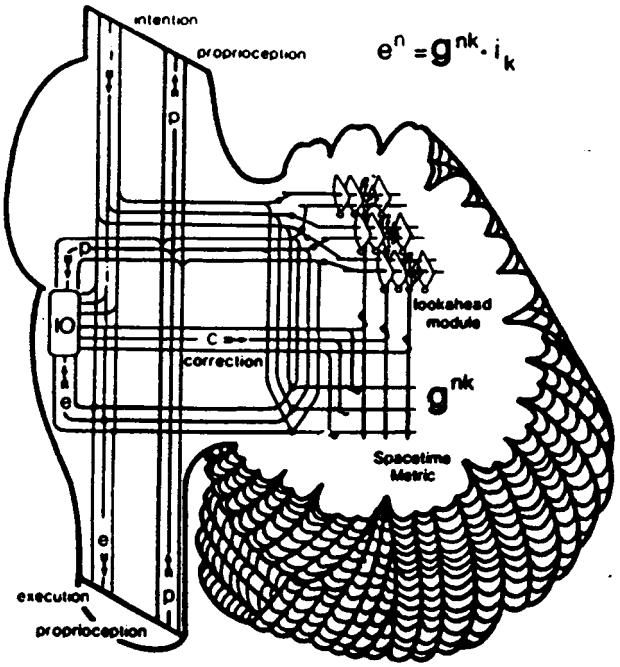


Figure 2.13 Schematic illustration of a the CMAC model. Note that the artificial cerebellum is not on the direct path between the brain and motor control, but rather a serves a secondary, signal modification role [22].

2.1.2.7.4 Neural-Solver

Montcastle [29], and Zeki and Shipp [30] proposed that the cerebral cortex is comprised of modules or cortical columns. These columns are groups of cells that process information both in a vertical and horizontal fashion. The intragroup processing is done vertically, and the intergroup communication is done horizontally. Bieszczad [31] proposed a network topology modeled after the human cerebral cortex that integrates these cortical columns. His model had two layers, referred to as the upper and lower layers. The *neuralsolver* was comprised of a grid of these cortical columns defining a [N x M x 2] lattice. An external pattern was entered at the lower level of the columns. As can be seen in Figure 2.15a, internal and external column communications were possible and *lower-upper* connections, depicted as a black arrow, are inhibitory. Information is processed by passing it between columns, (Figure 2.15b). Complex trajectories are represented as a path about the lattice surface called a *tree* as seen in Figure 2.16. The neuralsolver is capable of representing time dependent patterns, as well as simultaneously mapping several trees.

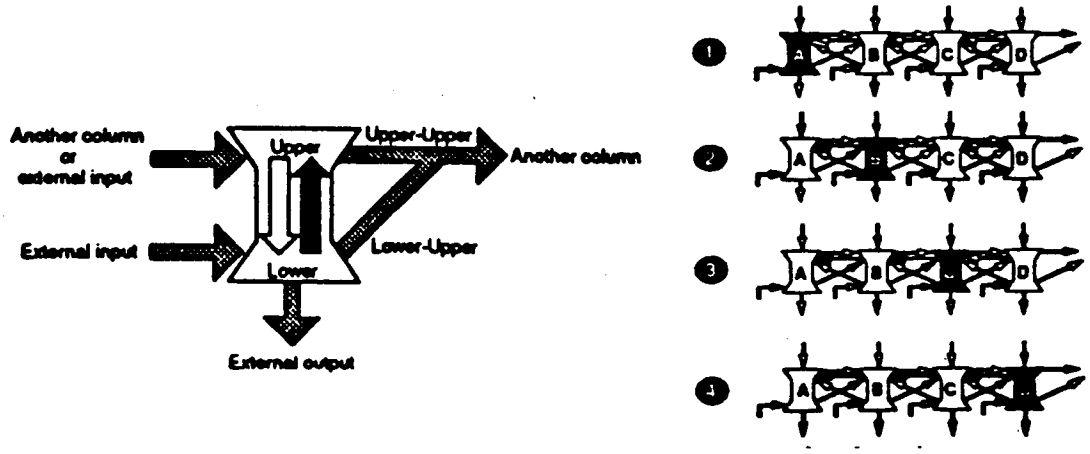


Figure 2.14 Cortical columns. **Left**, the connections of the cortical columns, where black arrows are inhibitory, and **right**, learning a pattern [31].

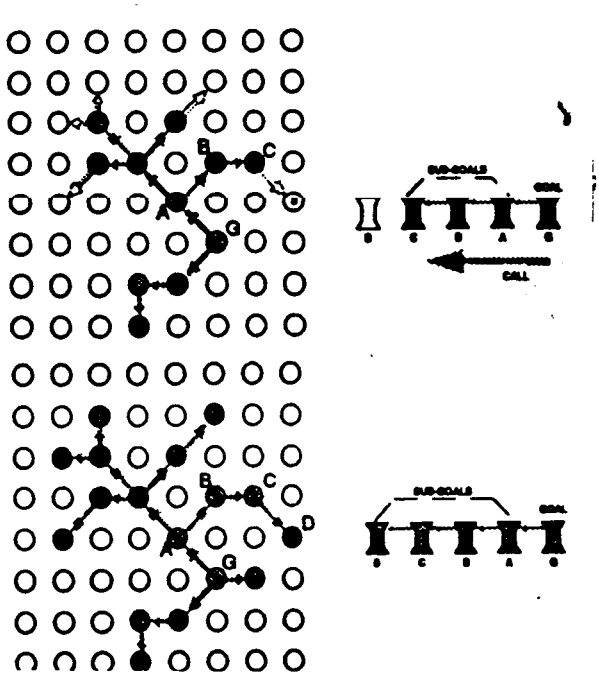


Figure 2.15 Construction of a call tree [31].

2.1.2.8 Large Networks

Although the most common approach to solving problems requiring a large amount of intelligence is to produce a sophisticated, highly intelligent network, there is a school of thought that believes that big, dumb networks can perform equally well. Most notably among these advocates is Anderson [32] who provides a logical argument based on biological evidence for his position. He states that neurons are the most expensive type of cell to have in a biological system, where expensive refers to increased nutritional requirements, energy consumption, and susceptibility to damage. He further assumes that the biologists are correct in stating that living creatures represent optimized designs; therefore, no structure present in those systems can be less than optimal. From this he concludes that there must be an important reason why human beings have 10^{14} neurons. This simple conclusion became the direction of Anderson's research where he produced very large networks of basic design. He reported that big "dumb" networks produced correct recognitions about as often as the more sophisticated architectures that have become the mainstream design; however, they had many more neurons and were less efficient. Anderson's work has continued in this area of very large networks and has indicated that a large number of neurons must be present before any interesting self-organizing behavior occurs. This observation is further supported by Edelman et al. [33,34] who performed self organization of large networks and noted that numerous neurons were needed before distinct topologies would develop.

Through a review of classical neural network architectures, it is seen that a steady progression in power accompanies increases in network size and architectural sophistication. By varying the architecture, the behavior of the network can be altered to allow the recognition of patterns that develop over time. The evolutionary networks, which are modeled closely after biological systems demonstrate a great deal of promise, but computer hardware limitations delay their development. Although the trend to increase the size of the network is evident from the literature review, it has been shown that classical designs are ill suited for one or two order of magnitude increases in network size. Also, classical networks retain a predominantly two dimensional structure. The two dimensional structure is only slightly altered when layers or vectors are stacked in order to capture elements of time; however, very little work has been done in developing three dimensional architectures. While the development of new network architectures is a recent area of research, the performance of any network is ultimately dependent on the method by which it is trained. In

the next section the two most common methods of training, backpropagation and Hebbian adaption are discussed.

2.1.3 Training

Once a network is produced, either in hardware or software, it must be trained to correctly map input patterns to output vectors. All training methods involve the systematic adjustment of synaptic weights. Although some limited work has been done using random approaches, such as genetic algorithms to train the networks, nothing has been nearly as successful as backpropagation and Hebbian learning. In the first half of this section backpropagation and its variations will be discussed in reference to their applicability to neural networks. The second half of this section describes Hebbian adaption as a more abstract method of training, but better suited for massive scale neural networks.

2.1.3.1 Backpropagation

The backpropagation paradigm utilizes a forward feeding neural network that initially has all of its synapses set to random values. The network is trained in a two step process using a "training" set that contains input vectors and the corresponding correct output vectors. First, the input vectors from the training set are used by the untrained network to produce an incorrect output. Second, the synapses are incrementally adjusted using a partial differential equation that relates the synaptic weights to the error between the incorrect output and actual value. These two steps are repeated until the error has decreased to the desired level. Training can be done on-line using live data, but is most commonly done off-line with a training set. Backpropagation obtains its name because the algorithm adjusts the synapse from output layer to the input layer, thus "propagating" the corrections "backward" through the network. Once the desired level of performance has been obtained, the synapses are "frozen," i.e., never adjusted again, so that the network will invariably perform the pattern recognition as trained.

Backpropagation was invented by Werbos [7,35] and was later refined by numerous others. The most commonly used refinement of backpropagation, the delta rule, was proposed by Rumelhart, Hinton, and Williams [36]. The delta rule put Werbos' technique in an algorithm form that is easier to implement. The delta rule is so similar to its antecedent that it is quite common to find the term backpropagation being used to describe the delta rule in the literature, so this notation will be used henceforth in this paper. Backpropagation is extremely effective on small networks; however, the size of the training set increases

exponentially with the number of synapses, nodes, and the degree of precision desired. This is a non-trivial problem and imposes a real upper limit on the feasibility of the method. While this limitation prohibits the realization of massive networks, there are three common approaches to expand the utility of the method: pruning nodes, developing more efficient backpropagation networks, and developing systems that integrate the learning phase into the actual operational phase.

2.1.3.1.1 Pruning Nodes

The goal of most engineering applications of neural networks is to rapidly process information. For this role, it is essential to optimize the network in the traditional engineering sense. Since network efficiency _ is given by:

_ = (processing power) / (number of nodes or synapses)

where processing power is a specific parameter to a given problem type, and optimization can be accomplished by decreasing the number of nodes or the number of synapses. In any non-optimized network, only a small percentage of the nodes and synapses are responsible for the vast majority of the processing that occurs in the network. One school of thought recommends that the nodes and synapses that do not have a highly active role in the data processing can be eliminated without significantly degrading the performance of the network [37]. This approach has been demonstrated on numerous accounts and is highly effective under certain conditions.

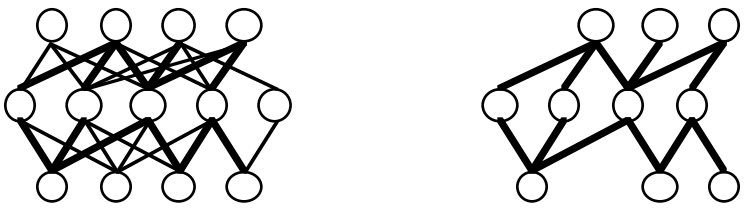


Figure 2.16 Pruning nodes and connections. **Left**, network after training, showing all initial connection and nodes with line thickness corresponding to synaptic weight, and **right**. same network after pruning.

There are two ways to remove "nonessential" nodes. The first approach uses genetic algorithms to facilitate the design of the network under development. Estevez and Okabe [38] developed an artificial chromosome that contained genes for synaptic weights and switches. The switches were proposed to modify the architecture of the network by activating or blocking select interneural connections. Although in theory their technique should have allowed a wide variety of network configurations, no clear outcome of this technique was reported.

A second approach to the node pruning problem employs algorithms that evaluate the network during the training phase to predict which nodes can be eliminated. This second approach is an organized version of the 'trial, error, and modify' approach that has been traditionally used to design networks. Lange, Voigt, and Wolf [39] developed an "evolution" algorithm that systematically adds nodes to a primitive network. The algorithm has four steps including: (1) the addition of a node, (2) some limited training, (3) the evaluation of the squared errors, and (4) acceptance or rejection of the node based on the error. The algorithm will iterate through these four steps until the squared error cannot be reduced through the addition of any more nodes. This approach is fairly effective in growing simple networks, but is far too iterative to be used in the design of a large network.

The two main limitations to pruning nodes are (1) the lack of an efficient evaluation technique, and (2) the reduction in the network's ability to learn (section 2.5). Pruning is an effective technique when the network is highly specialized and will be put into service in a static environment with no on-the-fly training or modification. While pruning increases the domain of applicability of backpropagation, it does not overcome the fundamental problems of the technique.

2.1.3.1.2 Advanced Backpropagation Algorithms

Another approach to increase the utility of backpropagation is to develop better algorithms that allow for faster training [36]. The philosophy of this approach assumes that a larger network can be trained in a given amount of time if the time of each iteration is reduced. This presumption is valid and numerous authors have published work both proposing and demonstrating better learning rules. Advanced backpropagation algorithms have demonstrated their utility for increasing the size of networks that can feasibly be trained with backpropagation; however, they have not overcome the exponential increase in training time that results from a linear increase in the number of nodes.

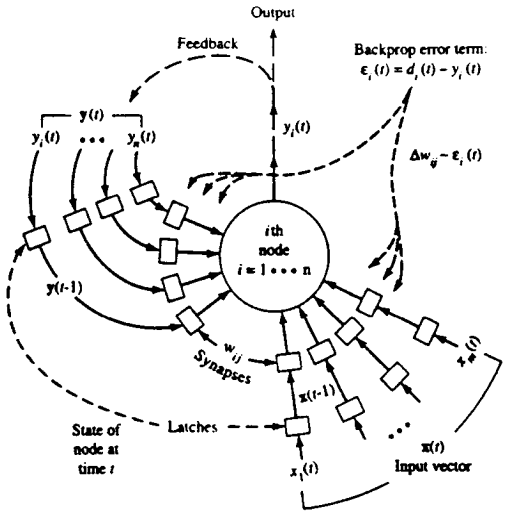


Figure 2.17 Diagram of recurrent backpropagation network [41].

The most common technique to increase backpropagation efficiency is to incorporate feedback and feed-forward mechanisms. This technique, originally suggested by Rumelhart, Hinton, and Williams [36], employs temporal nodes that 'remember' previous learning steps, and 'target outputs' that are used instead of the actual outputs for error calculations [40]. Temporal nodes are loaded with the values from the input vector of the previous cycle, (Figure 2.18). The 'target' outputs are used to force the backpropagation algorithm onto a presumably better convergence path. The temporal nodes not only help in the learning phase, but allow the network to process time-varying sequences [41].

Rojas [42] has suggested using fractal geometry to predict the convergence rates of backpropagation. His paper demonstrates that for certain networks, the iteration paths of the backpropagation paradigm follow fractal patterns. Although he did demonstrate that fractal geometry could be used on some networks under certain conditions to predict convergence of a backpropagation algorithm, he did not prove his technique to be universally applicable.

Abbas and Bayoumi [43] developed a faster backpropagation algorithm utilizing an "anti-Hebbian" rule. The algorithm actively weakens the terms in a conventional backpropagation differential equation associated with weak synapses so that training is focused on those synapses that are strongly correlated. This approach seems to be quite clever, but it is named incorrectly. Anti-Hebbian refers to the neural-dynamics associated with the weakening of the actual synapse, not the weakening of the terms of a training differential equation. The authors stated that a shortcoming of Hebbian modification is that

the synaptic weights can grow indefinitely. While this statement is true in the context of traditional engineering applications of Hebbian modification, it is irrespective of the biological process where synapses can only grow to a maximum physical size.

In cases where traditional backpropagation becomes trapped in local minima, alternative training approaches, such as genetic algorithms, have been suggested. Lueng, Luk, and Ng [44] used a genetic algorithm to train a simple, forward-feeding neural network. Their approach used genes to represent synaptic weights. Compared to backpropagation, genetic algorithms find the area of the true minimum rapidly, but then are quite slow to converge on the absolute minimum.

2.1.3.2 Hebbian and Anti-Hebbian Networks

In 1948, Donald Hebb [45] proposed that if a neuron receives a signal that initiates a response, then that synapse will become stronger. The Hebbian mechanism is completely contained in the neuron, and therefore the neuron can learn without knowing the states of any other neurons. The criteria of self-sufficiency is well supported by physiological evidence. The mechanism Hebb described is the basis for all learning that occurs in biological systems. Recent advances in micro-neurology have since better defined the actual mechanics of the mechanisms Hebb proposed, but his basic premise is still valid.

Hebbian learning is based on two biological mechanisms, long term potentiation, LTP and long term depression, LTD, where the former refers to an increase in synaptic strength and latter corresponds to a decrease in synaptic strength. LTP is initiated when a neuron receives a signal from a weak synapse at the same time another signal, received over a strong synapse, initiates the firing of the neuron [46]. LTP is driven by the release of chemical signals that strengthens the synapse temporally and initiates permanent strength modification of that synapse. LTD, a weakening of synaptic strength, on the other hand is prompted by prolonged stimulation by a weak synapse that does not initiate neural firing [47]. LTD is similarly regulated by the release of chemical agents. LTP and LTD are also discussed in section 2.3.2.8.

There are two types of neurons, excitatory and inhibitory. If associative learning occurs on an excitatory cell, the process is called Hebbian adaptation. Likewise, associative learning that occurs on an inhibitory neuron is referred to as anti-Hebbian adaptation. Hebbian adaptation can either be correlational (LTP) or decorrelational (LTD). All of these terms are relative to the resting potential, which is usually zero for artificial neurons, but is not theoretically constrained to this value. A correlational Hebbian adaptation moves the postsynaptic value away from the resting potential in the positive direction, while a decorrelational Hebbian adaptation moves the postsynaptic potential towards the resting potential. The maximum postsynaptic value of a Hebbian synapse is typically 1, and the minimum value is the resting potential, zero. Hebbian synapses are constrained within this range, and once their value has reached either of these limits, further adaptation in the same direction is impossible.

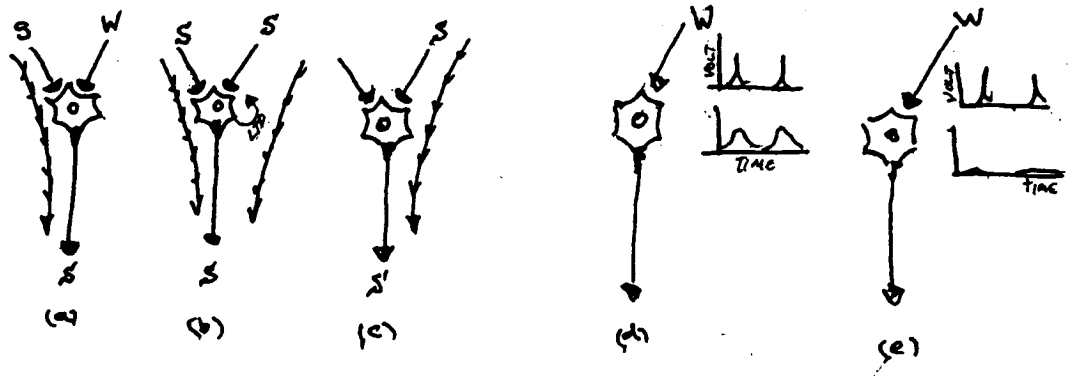


Figure 2.18 LTP and LTD mechanisms. (a) Correlation of a weak signal triggering an LTP to occur in (b) strengthening the synapse so that in (c) either signal alone can initiate a response. (d) Decorrelation of a weak stimuli to (e) where the input is hardly registered in the neuron.

Similarly, a correlational anti-Hebbian adaptation moves the postsynaptic value away from the resting potential in the negative direction, and a decorrelational anti-Hebbian adaptation returns the postsynaptic value to the resting potential. Again, similar to the Hebbian case, the anti-Hebbian mechanism is constrained within a range between 0 and -1 (Figure 2.19). The Hebbian and anti-Hebbian mechanisms work in conjunction to form tightly compacted neuron groups which represent different classes with a mechanism called self organization (section 2.5.2).

The main difference between Hebbian adaptation and backpropagation is the location

of the intelligence. Backpropagation places all of the training intelligence in an external computer that manipulates the synapses based on their final output. Backpropagation requires a global knowledge of all neurons and their synapses. Since each iteration of backpropagation requires numerous calculations to be performed, the computational power of the computer performing the backpropagation is proportional to the network size. Recognizing that the human brain has 10^{16} synapses, of which approximately 10^{14} are active per second, it becomes clear that even the Gigahertz, (10^{9} $^{cycles}/_{second}$), speed of a supercomputer is 5 to 7 orders of magnitude slower than what is needed to feasibly perform backpropagation on a network the size of the brain.

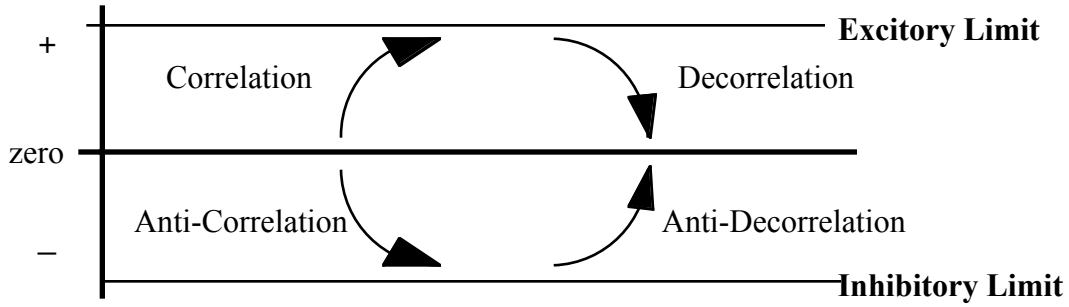


Figure 2.19 Depiction of the relative changes in synaptic weights due to correlation and decorrelation of Hebbian and anti-Hebbian mechanisms.

In contrast, Hebbian learning is based purely on local information. To implement a Hebbian learning rule, one must design or program each neuron to be capable of simulating the LTP and LTD mechanisms. The correct modeling of the LTP and LTD mechanisms is quite computationally intensive, requiring each neuron to perform all activities associated with synaptic adjustment. While this places a heavy computational load on each neuron, the design is essentially infinitely expandable. It is the potential of high parallelism and great expandability that has attracted so much interest to the true form of Hebbian learning.

More commonly, Hebbian learning is implemented by designing a program that keeps track of all neural activity and adjusts synaptic weights accordingly. Although this latter approach is easier to implement initially, it is restricted much like backpropagation in that the upper bound on the network size is proportional to the power of the computer. For classification purposes Hebbian learning will refer to this latter supervised technique and Hebbian adaptation will refer to the previously described unsupervised technique. Hebbian adaptation actually pre-dated backpropagation; however, until recently the computational power has not existed to drive the more complex Hebbian model. Hebbian adaptation has

always been recognized by cognitive science theoreticians as the training technique for massive neural networks because it is the mechanism used by the brain.

It is becoming more common to find examples of Hebbian learning appearing in the literature. The overall trend is that Hebbian learning is more efficient for categorizing different entities when performing pattern recognition with many entities involved. Chatterjee and Roychowdhury [48] recognized the tedium associated with training a large network with backpropagation. They demonstrated that a Hebbian network could be trained to perform the character recognition done by the U.S. Postal Service more rapidly than its backpropagation counterpart. Palmieri [49] noted that Hebbian networks were trained more rapidly when employing both Hebbian and anti-Hebbian mechanisms.

However until recently it was not known which parameters affect the learning rate. Fomin and Lórincz [50] showed the sensitivity of Hebbian learning to the learning parameters, the sharpness of the sigmoid function, and the saturation value. The learning parameters define how much the synapse will change per modification and what synaptic values are considered strong and weak. The shape of the sigmoid function essentially determines the gain of the node, with a sharp sigmoidal function corresponding to a high gain. Lastly, the saturation value corresponds to the input level, above which the neuron will fire. These authors found that Hebbian learning was very sensitive to saturation value and relatively insensitive to the other terms.

In summary, backpropagation is a powerful tool for engineering neural networks but is ill suited for massive scale systems because the amount of training data and time needed for training increases exponentially with network size. Hebbian adaption, which relies only on local information is better suited for large and massive scale systems, especially when the adaption process is local rather than orchestrated by a supervising computer. While performance gains can be obtained through alterations or varying the training method, the most promising avenue for increasing network performance is the development of specialized hardware. In the following section trends in hardware development are discussed and several examples are cited showing the current endeavors in producing large scale neural networks.

2.1.4 Hardware Implementations

The first part of this section describes the basic differences between representing the network with analog versus digital elements from a hardware point of view. The discussion progresses to describe the basic elements: the neuron, the synapse, and the membrane, and the

roles they play in signal logic. The latter part of this section describes hardware applications. Hardware applications fall into one of two categories, high speed and large size. High speed applications are used when very rapid pattern recognition is desired. These systems tend to be compact and highly integrated. The second category has evolved due to the inability of serial processing to adequately emulate the parallel processing of large scale networks. Large size hardware applications are usually comprised of multiple chip sets that can be arranged or reconfigured easily to allow various architectures to be produced.

There is no question that a hardware implementation operates more rapidly than an equivalent software counterpart. While most authors predict that future neural network applications will be done in hardware, currently most are software emulations. Software implementations have been facilitated by math environment programs, such as *Mathematica®* [13], special neural net programs [51], and the development of special C++ functions. Hylander and Meador [52] developed a library of sophisticated C++ functions that allowed the neurons to be represented as producing a pulse modulated width output, which, in fact, is what biological neurons actually do (section 2.3.2).

Unfortunately, the interesting functionality of biological neural nets does not express itself until the network is of sufficient size [32]. Large neural nets with complex neuron models place high demands on a traditional serial computer as it tries to simulate the nature of the network. To increase the rate at which a traditional CPU based computer can emulate a neural network, accelerator boards have been developed, some of which claim up to 250,000 associations per second [53]. Of course, an accelerator board cannot yield the performance of a dedicated computer or supercomputer specifically designed for neural net applications.

Asanovic et al. [54] have proposed a 128 node, 32 bit parallel processor capable of performing some 160 billion operations per second. The design of the supercomputer links the 128 nodes by 1024 parallel data paths and has a single central processor that coordinates all of its activities. The supercomputer is being built to allow real-time speech and video processing. The system has not been built yet, so no evaluation of its performance can be made. Although the authors have proposed an interesting system, the effort is mainly a computer engineering project because living organisms do real-time audio/video processing with neurons that operate in the kilohertz range.

Table 1. Advantages and disadvantages of analog, optoelectronic and digital approaches.

	Advantages	Disadvantages
Analog	compact high speed, but depends on precision	susceptible to process-parameter variation susceptible to noise lacks scalability the precision of synaptic weights is limited difficult to make modifiable synapses
Optoelectronics	large fan-in and fan-out modifiable synapse by SLM	opto-electronic and electro-optic transformation susceptible to noise and parameter variation
Digital	high precision high scalability modifiable synapses	large circuit size

Table 2. Representation of input and output.

Approach	Representation	Circuit technology
Analog	analog voltage	buffer amp.
	pulse encoding	VCO
	1bit digital output	comparator
Optoelectronics	analog voltage	buffer amp
	light intensity	LED and photodiode
Digital	binary digital	bit parallel bus
	(bit serial transmission)	serial/parallel conversion
	pulse density	rate multiplier

Table 3. Representation, storage and multiplication of synaptic weights.

	Representation	*Storage*	*Multiplication*
Analog	resistor array	fixed or digital switch array	Kirchhoff's low
	MOS channel resistance	storage device	V-I characteristics
	amplifier gain	memory for bias voltage	transconductance gain
	storage capacitor	floating gate	Gilbert-Multiplier
		refreshing capacitor with digital memory and D/A converter	Gilbert-Multiplier
	D/A converter	digital memory	multiplying D/A converter
	switched capacitor	refreshing capacitor	switching frequency
Optoelectronics	SLM	electric charge to modurate polarizor	difference in polarizing angles
		density of transmission media	transmissivility
	photodiode array	light source (CRT or LED array)	gain of photodiode
		memory for bias voltage	gain of photodiode
Binary digital	digital memory	digital memory	digital multiplier
Pulse density	pulse density	digital memory	logical AND
	digital memory	digital memory	rate multiplier

Table 4. Representation of spatial summation

Analog	current summation by a differential amplifier with wired-OR inputs
Optoelectronics	convergence of light by lens
	current summation
Binary digital	accumulator
Pulse density	nonlinear summation by OR gates

Table 5. Representation of temporal summation.

Analog	RC integrator
Optoelectronics	no example
Binary digital	numerical solution
Pulse density	up/down counter and rate multiplier solution of differential equation by integral form

Table 6. Representation of output function.

circuits	output functions	realizing methods
Analog	sigmoid tanh	transfer characteristics of amplifier
	unit	comparator
Optoelectronics	sigmoid tanh	transfer characteristics of amplifier
Binary digital	any function	table look-up
Pulse density	tanh	nonlinear spatial summation
	ramp	thresholding gate to suppress negative output

Since the highest processing rates are always obtained from dedicated hardware, substantial work has been focused in this area. Hirai [55] gives an excellent discussion on this topic, addressing various issues including: analog vs. digital, neuron representation, and hardware types, and has produced the following classification tables.

Unfortunately, the novelty of neural net technology precludes a historical perspective for evaluating the optimal performance systems [56]. Additionally, current systems cannot be readily compared because no benchmark for neural network processing has been established. van Keulen [56] suggests that an evaluation criteria should not be based on system design or clock speeds, but rather on general parameters such as learning rates, mapping speeds, solution accuracy, and resource consumption.

2.1.4.1 Analog vs. Digital

The two principle technologies competing for neural net applications are digital and analog hardware. Although there are numerous ways that either technology can be realized, some basic trends can be identified. Analog hardware has the advantage of being faster, having fewer elements, allowing tighter on-chip neuron densities, and being representative of the actual summing process of biological neurons. Unfortunately the precision of the analog element is a function of the processing quality and the physical size of the element [57]. Since the cost of producing the element is also a function of precision and size, Shi [57] identifies that doubling the analog precision increases the cost at a much greater rate than the addition of a single bit of a digital system (which would double its precision). Arreguit and Vittoz [58] identify that the development of an architecture to allow the interconnection of a large number of analog elements is a nontrivial problem that has prevented the commercial

realization of analog systems. Additionally analog neural networks are not well suited for backpropagation training. Withagen [59] developed an analog implementation that could be trained with backpropagation; however, his system needed four additional multipliers, making the layout cumbersome and slowing down convergence.

The digital neuron is slower, requires many more elements, and therefore, in theory, would give a lower packing density. Digital technology has the advantage of its low power requirement and linear coefficient of resolution expansion. To double the resolution of a digital system one need add only a single additional bit, while to double the resolution of an analog system, one needs to double the size, and thus square the power consumption of the element. The main advantage of a digital system is that the tolerance of the individual transistors is very low since only a high or low state is needed to be detected, reducing manufacturing constraints. The digital systems also allow the designer to use network buses and RAM storage to produce the synapses while the analog systems require actual hardwiring.

2.1.4.2 Basic Elements

To produce an artificial neural network in hardware, two things must be done: an electronic representation of the neuron must be created, and the artificial neurons must be connected to form a network. The basic elements of the artificial neuron include the synaptic weight memory, weight-input multipliers, a post-synaptic adder, and a post-synaptic multiplier. The analog approach usually represents the synaptic weights as variable resistors or as currents in a CMOS-Gilbert multiplier (Figure 2.20). The input signals are multiplied by the synaptic weights using an analog multiplier, and these post-synaptic values, typically in the form of voltages, are added. The summed post-synaptic value is multiplied by an analog circuit designed to replicate the sigmoidal function of a biological neuron. This approach is demonstrated by Abusland and Landy [60].

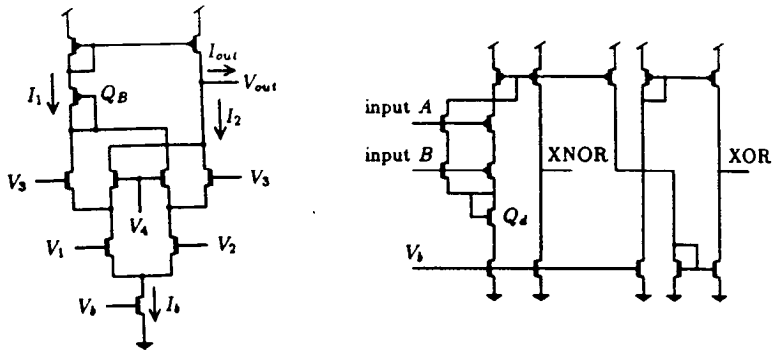


Figure 2.20 Hardware synapses. CMOS Gilbert floating gate multiplier **left** [60] and hybrid synapse **right** [61].

The storage of the synaptic weight is the most challenging aspect of the analog approach. Not only is the basic Gilbert multiplier difficult to upload values into, but it is volatile, meaning that all memory is lost when the power is turned off. Montalvo, Paulos, and Gyurcsik [61] developed an alternate analog synapse, which is a hybrid of volatile and nonvolatile elements (Figure 2.22). The volatile elements allow more rapid training, while the nonvolatile elements facilitate programming.

The digital representation of a neuron uses RAM memory to store a digital value corresponding to a synaptic weight. The incoming signal, also represented by a digital word, is crossed with the synaptic value by the digital multiplier. The various post-synaptic values are added digitally. The sum is then modified by a digital circuit designed to produce a sigmoidal response. Beichter et al. [62] developed a VLSI digital neural network processor employing this format that had four neurons, each consisting of four multipliers and four adders.

One of the shortcomings of the digital approach is that the memory arrays that store the synaptic weights grow rapidly to unmanageable proportions as the degree of interconnectivity and network size increase. Chung, Tsai, and Sun [63] investigated the dependence of network performance on synaptic resolution. They designed a neural network with a few "quantized" levels of resolution in an attempt to facilitate hardware implementation.

Zaman and Wunsch [64] developed a hybrid neural network chip which uses multiplying-digital-to-analog converters (MDACs). The MDAC system stores the synaptic weights in a 6 bit register and converts these weights to an analog value, which is temporarily

stored in an EEPROM floating gate synapse. When the synaptic values are in analog form, they can readily be multiplied with the input signal. The MDACs allow the chip to retain the asynchronous parallel behavior of analog circuits while allowing digital storage.

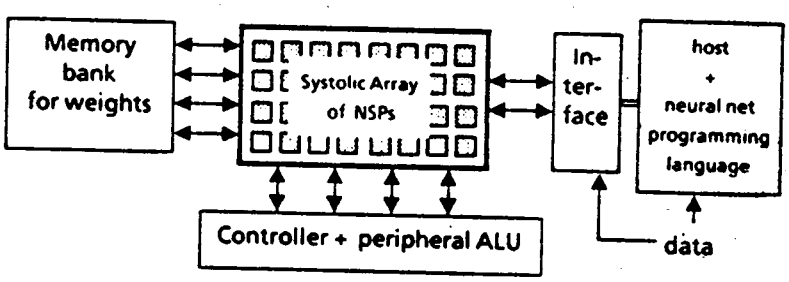


Figure 2.21 Beichter's neural network processor [62].

2.1.4.3 Advanced Synapses

The McCullough-Pitts neuron [8] uses a very crude representation of the synapse. Several researchers suggest that improved network performance can be realized by developing a more representative synapse. Elias and Northmore [65] developed an electronic representation of the membranes that constitute a synapse. Their model, which only characterized sodium channels, consisted of five basic elements (Figure 2.24). These five elements were adjusted during the training process to allow both inhibitory and stimulatory responses.

The true biological synapse has a certain amount of randomness and time delay associated with its functioning. Gopalsamy [66] developed a model that contained interneural transmission delays. His study investigated the effect of these delays on convergence, but found that the convergence was independent of all but large time delays. Delgado-Restituto and Rodriguez-Vazquez [67] introduced more complex functions of the synaptic transmission to more closely resemble the sigmoidal function of the biological neuron. They found that the shape of this function did have an effect on convergence. Douglas, Mahowald, and Martin [68] arranged their network into functional groups that were interconnected to allow the various signals to be regenerated. Their model was mostly based on the refresh feature of RAM memory and also abstractly based on the visual cortex. They suggested that by grouping the neurons in this fashion that their network would be less sensitive to interneural transmission noise.

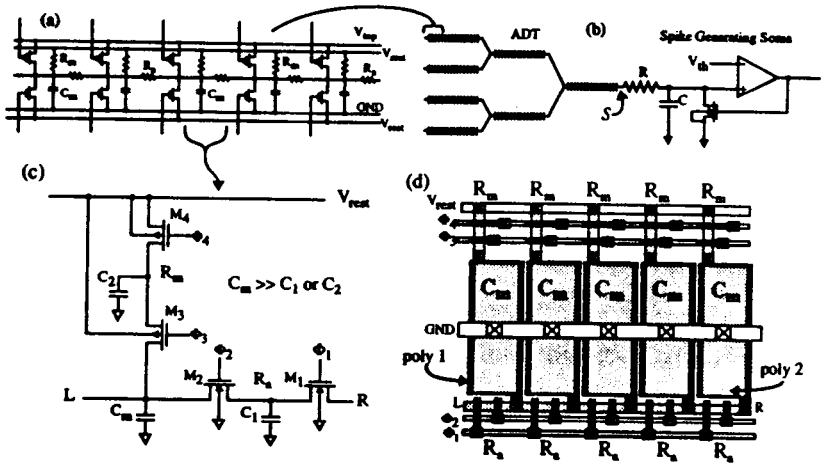


Figure 2.22 Electronic membrane. (a) The membrane's five compartments (b) are represented by five small lines on the artificial dendritic tree (ADT). Synaptic value are stored in switched-capacitors (c). VLSI layout (d) [65].

2.1.4.4 Basic Hardware Systems

There are two basic types of hardware implementations of computational networks commonly seen in the literature, ultra high-speed, pattern recognition systems and generalized, expandable computation networks.

Figure 2.23 High speed pattern recognitions system. (a) Network schematic and (b) system block diagram [69].

Ultra high speed pattern recognition systems tend to have simple, fixed architectures, relatively simple neuron representations, and few neurons. Masa, Hoen, and Wallinga [69]

developed a three layer feedforward neural network with 70 input nodes that could classify a pattern in 20 nanoseconds (Figure 2.25). Their system was developed for identifying particles in nuclear research.

The second type of neural hardware implementation is designed to allow the assemblage of large neural networks through the integration of numerous microchips. This philosophy is based on the ability of digital microchips to be cascaded. Microchips of this kind are referred to as VLSI neural building blocks. Myers, Vincent, and Orrey [70] developed a three-layer neural building block by integrating four node processors per chip. The integration of these chips is to be supervised by an external computer. The obvious weakness of this approach is the use of an external computer; however, its replacement substantially increased the system's complexity.

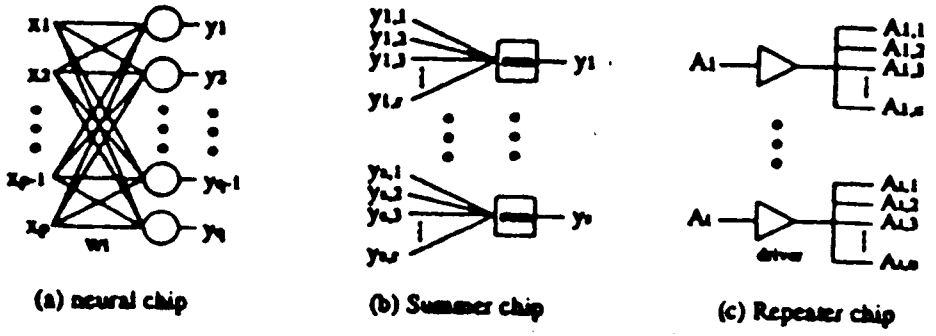


Figure 2.24 The HAVENN chip set. (a) Neural chip, (b) summer chip, and (c) repeater chip [71].

Lo and Fisher [71] suggested a system HAVENN, composed of three chips that can be arranged in various orders to produce a desired network configuration (Figure 2.26). The chips are fully digital and have 6-bit synaptic weights. The first, the neural chip, contains a series of neurons with input and output layers that are essentially fully interconnected. The second type of chip, referred to as a summing chip, has many inputs, a few neurons, and a few outputs. Unlike the neural chip that has the same number of inputs and outputs, the summing chip has far less outputs than inputs, and serves to anastomose the inputs from various other neurons. The last chip, the repeater, performs the inverse function of the summing chip, producing numerous outputs from a single input. In essence, neural "trees" can be produced by branching with the repeaters, processing with the neural chips, and assembling with the summing chips.

Although Lo and Fisher's system allows for some flexibility in the network's topology, it is hard wired, and thus architectural modification requires physical modification. Van der Spiegal et al. [72] suggested a system with numerous hard switches that could be controlled through software, essentially creating "soft" reconfigurable hard interconnections. Their system was also composed of three chips, but the chips functions were completely different (Figure 2.28). Their system functioned in the analog mode; however, all parameters, such as time constants and synaptic weights, were stored and addressed digitally. Parameters are uploaded in the digital mode then converted by a D/A conversion to a value that the network used during normal operation. The analog representations were periodically refreshed because they tended to drift during operation. The system was connected by "switch" chips that served as soft programmable junctions. The switch chips allowed the outputs of any neuron to be redirected to the synapse of any other. The synapse chip received its inputs from the switch chips. The synapse chip then multiplied the incoming value by the appropriate weight, sending the post-synaptic values to the neuron chips. The neuron chip summed the post-synaptic values and then applied a transfer function that was representative of an actual neuron's response. The chips are arranged on boards that are connected as seen in Figure 2.28, producing a network comprised of 1,024 neurons with 98,304 synapses.

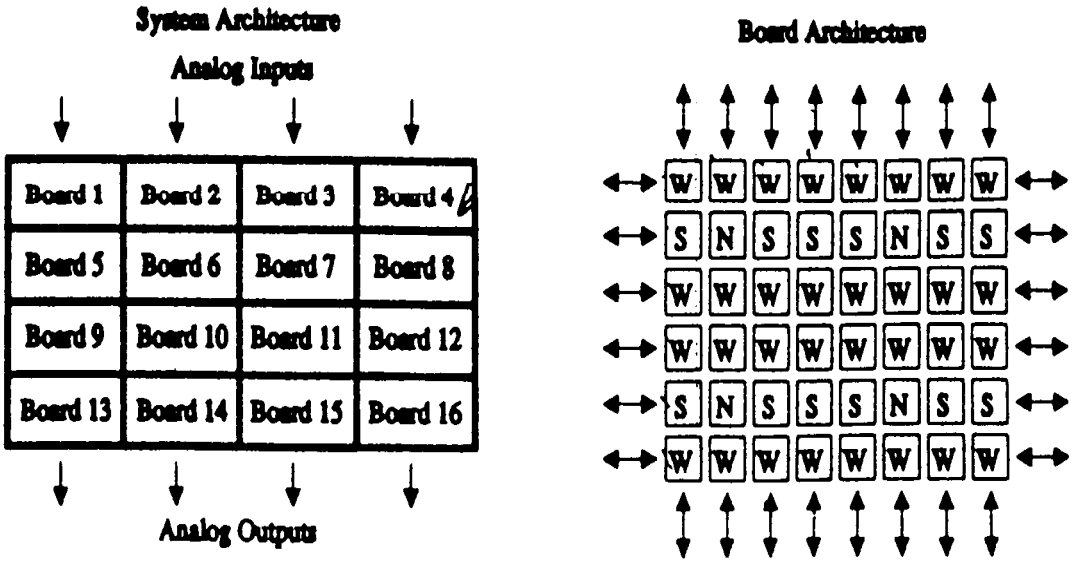


Figure 2.25 Van der Speigal's system layout, **left**, and board architecture, **right** [72].

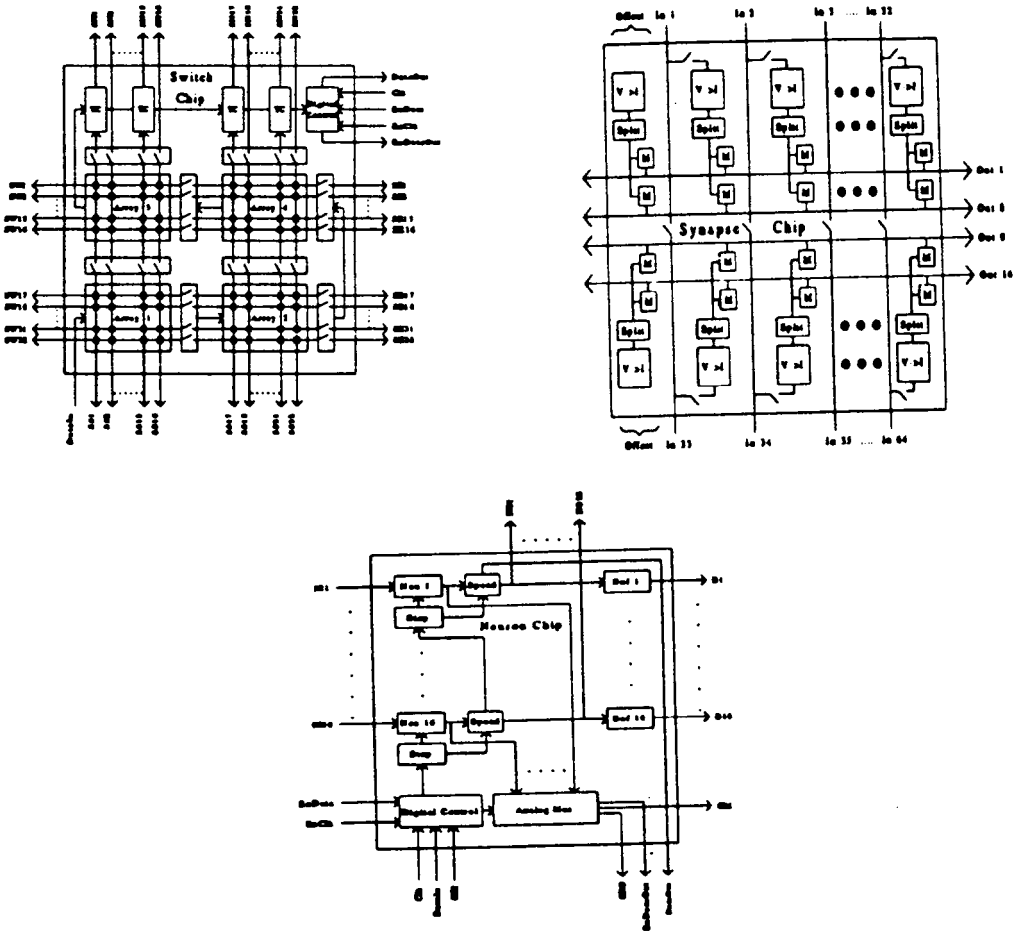


Figure 2.26 Block diagrams of Van der Spiegal's chip set. Switch chip, top right; synapse chip, top left; and neuron chip bottom [72]. See text for details.

2.1.4.5 Advanced Biological Models

While most hardware implementations are computational in nature, the capabilities of biological systems to perform signal processing has aroused considerable interest. The two biological systems most commonly researched in a hardware context are hearing and vision. Research in these areas has been focused on developing artificial sensing organs and artificial sensory cortices.

Biological systems are "distributed" in that all elements of a particular sensory system contribute equally to the processing of stimuli. Unlike a man-made microphone, the cochlea of the human ear performs a considerable amount of signal processing before it sends a signal back to the brain. Lyon and Mead [73] first proposed to produce a silicon cochlea

that was capable of doing minor signal processing, and directly produced a parallel output consisting of the intensities of various frequencies. Work has continued on the silicon cochlea, most recently incorporating second-order, discrete-time, switch-current filters [74].

The interpretation of the signals received from the cochlea are also of considerable interest. While direct speech recognition has always been an area of interest, recent findings suggest that the human auditory system performs several other functions that may be critical to the interpretation of sound. One of these secondary functions is the binaural localization of sound, which is based on the time delays between the sounds as heard by the two different ears. Once the time delays have been established, the two signals from each of the two ears, can then be used for error correction to improve on recognition efficiency. Additionally the identification of the source location of multiple sounds can be used to neglect the "noise sources" and concentrate on the significant source. These abilities prompted Bhadkamkar [75] to develop a neural network to identify the location of a sound source (Figure 2.27). Using binaural localization, his system successfully recognized the original sound over random environmental noise and echoes.

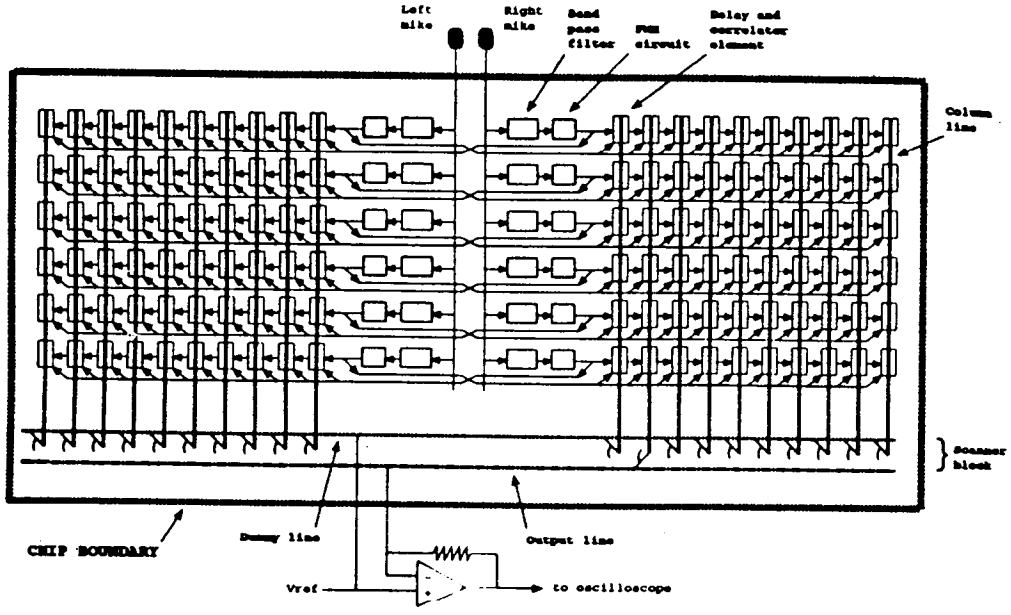


Figure 2.27 Simplified schematic of the sound localization chip, note that real chip has 25 rows and 20 elements per side as opposed to what is depicted here [75].

A similar approach has been taken to model the visual system. An artificial retina was developed by Mead et al. [76,77,78] in which the sensory units are interconnected by neurons to enhance edge detection and motion detection. Girod et al. [79] successfully demonstrated that comparing the two images from binocular vision can be used to enhance edge detection (Figure 2.29).

Investigations in biological systems have also provided useful information to how a network's topology should be arranged. The stratified structure of the visual cortex (), has been partially replicated by Wu and Nishiwaka [80], and the complete integration of sensor and data processing elements has been demonstrated by Duong et al. [81] who needed to process complete visual images at very high rates for guided missile applications. The system they developed contained a sensing array, three strata of neural elements vertically stacked in what was described as a sugar cube (Figure 2.28) The image received at the top strata was directly sent to the succeeding strata with data buses located on the edge of the cube. In total there were 134 neurons, 4,864 synapses, and the system was capable of processing 1,000 video frames per second.

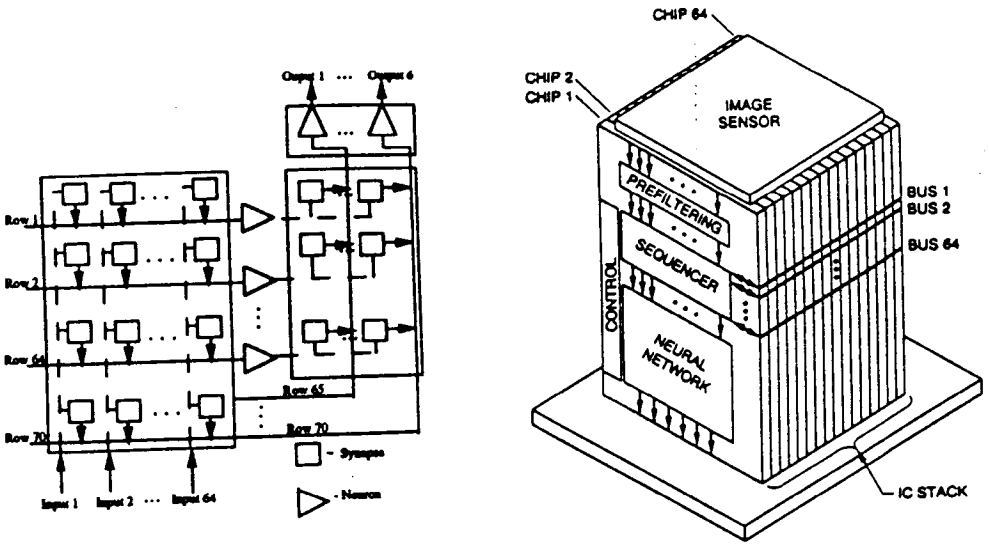


Figure 2.28 High speed video processing "sugar cube." The neural network architecture **left** and the assembled cube appears **right** [81].

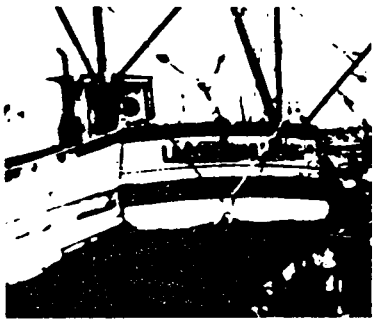


Figure 5 : image "LACORNOU"

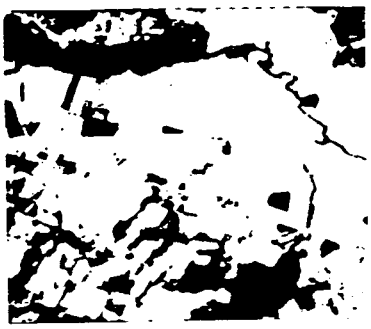


Figure 8 : image "AQUITAIN"

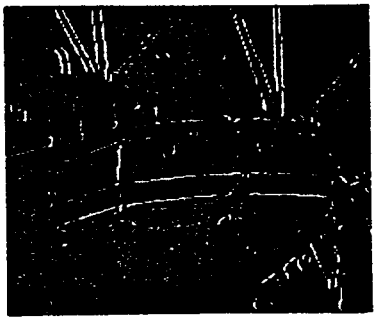


Figure 6 : edge image "LACORNOU" with Canny-Deriche operator

Figure 9 : edge image "AQUITAIN" with Canny-Deriche operator

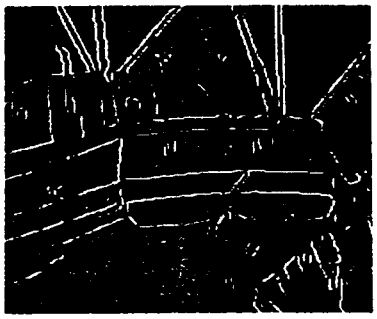


Figure 7 : edge image "LACORNOU" with biological model

Figure 10 : edge image "AQUITAIN" with biological model

Figure 2.29 Edge enhancement based on binocular vision. The original image, top, is processed with a traditional image processor, center, and the binocular neural network bottom, which better defines the edges [79].

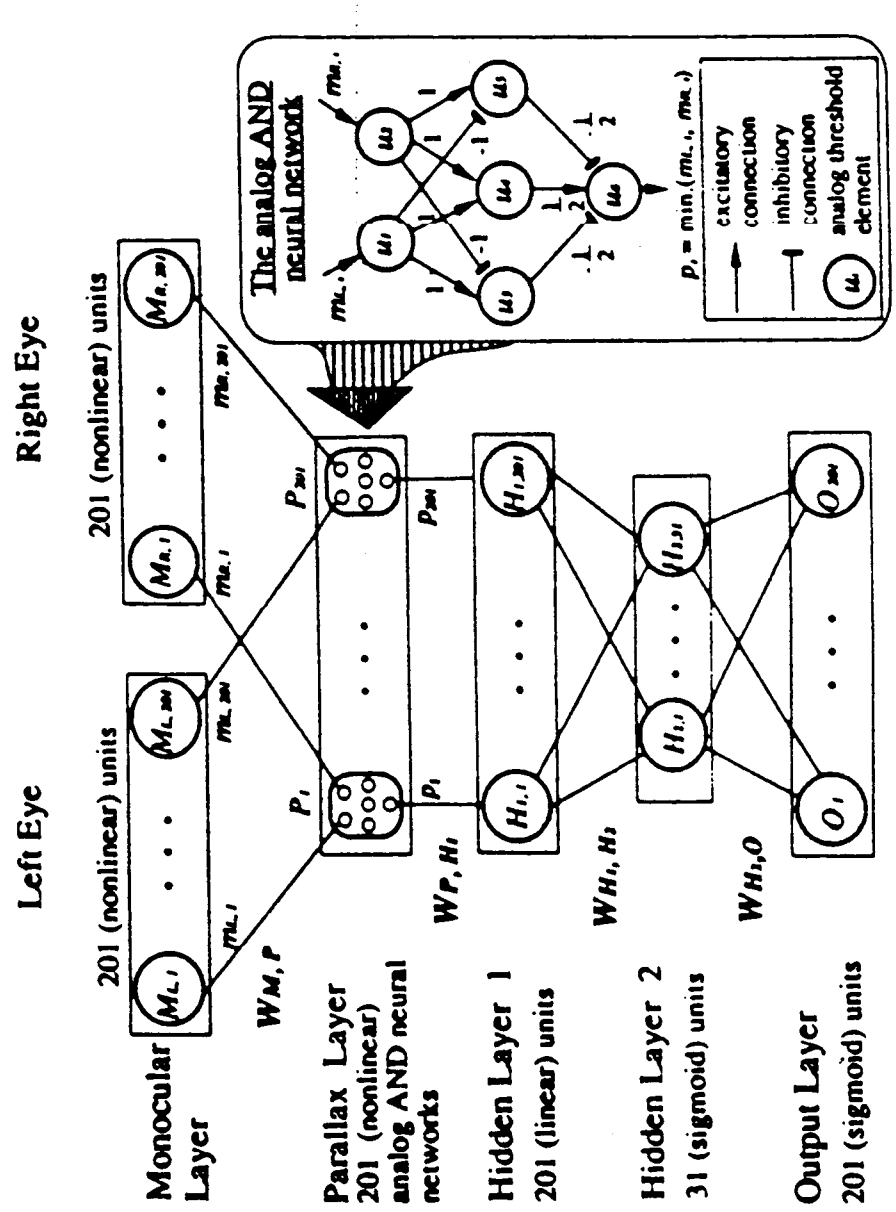


Figure 2.30 Multiple strata artificial visual cortex [80].

2.1.5 Summary

The field of neural networks is growing rapidly, spanning many different network architectures, training methods, and hardware designs. Nearly all current network designs are two dimensional and are limited to a fairly small size by the universal inability to train large systems. Hardware applications hold substantial promise for large systems; however, there is no clear logic or method for designing the architecture of large systems. All of the networks discussed in this section were either developed by modeling a specific system or through an iterative trial and error process. To date, no general method exists for designing a network that allows optimization for specific problems or needs. This section has demonstrated a clear and present need for a scientific method of designing and optimizing network architectures.

2.2 Smart Processing

In the previous section the fundamentals of neural network design were discussed. In this section the focus will shift to application of neural networks to process control. The are three basic types neural network process control; neural-sensors, adaptive critics and neurocontrollers. Neural-sensors describe a class of systems where neural networks are located between the sensing devices and the process controller. In a sense the neural network are used as signal preprocessors for non-linear or noisy data. The output of the neural-sensor is a clean signal that is readily interpreted by a traditional process controller. Neural-sensors are by far the most popular. There are more examples of neural-sensors than the other two categories combined. The second way neural network can be used in process control is to adjust or tune the existing process controller. Such an architecture is called an adaptive critic. The networks are again used to compensate for non-linearities, however they are implemented at a higher level. The adaptive critic design has been used to integrate neural networks to many other forms of controllers such as PID, and expert systems. The last method, neurocontrol, uses neural networks to directly control the system. The following discussion includes wide range of industrial applications to demonstrate breadth of applicability of neural networks to manufacturing and process control

2.2.1 Neural-Sensors

This section describes the use of neural networks as signal preprocessors. Traditionally, a single neural network is used to fuse all inputs into a common output signal. A good example of this approach was demonstrated by Smith and Dagli [82]. They developed a forward-feeding neural network trained with a backpropagation rule to aid in the production of PVC pipe. Their network used process data such as bulk density, feed rate, temperature, etc., to produce a recommendation used by a traditional controller directly connected to the process equipment.

Fildes [9] developed a smart system that used the output of the spectral-analyzer to adjust the control parameters of a composite epoxy cure system. The measurement of the cure state of epoxies is complicated by impurities, which add a substantial amount of noise in the IR spectra emitted during the curing process. A simple three layer neural network was demonstrated to be capable of interpreting these "dirty" spectra and predicting the quality of the cure.

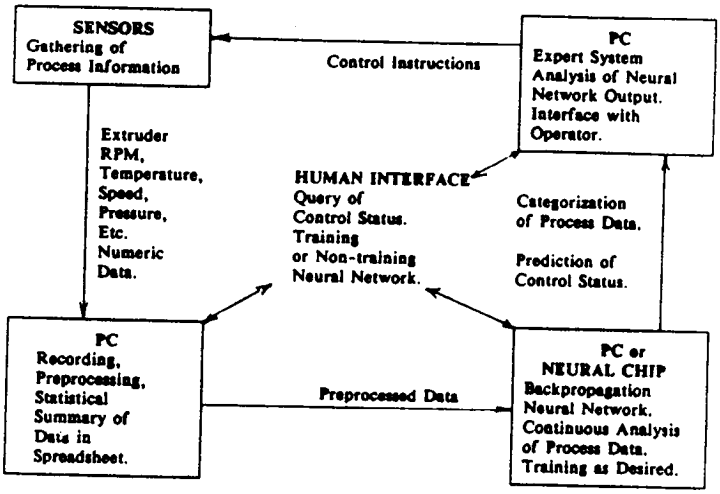


Figure 2.31 Smart system for PVC pipe extrusion process control [82].

Cook and Shannon [83] developed a forward-feeding recursive network for the production of particle board (plywood) and reported favorable results. The network recommended the moisture content and resin treatment based on a variety of process parameters. The network's structure used the input vectors of previous time periods for the context units.

Koenig, Memis and Tansel [84] developed a scheme to model and control plunge grinding. They used two networks to predict the cutting forces and acoustical emissions respectively. The latter network was of greater interest as the acoustical emissions are extremely difficult to analyze with other approaches.

One of the most interesting applications was demonstrated by Fechner, Neumerkel, and Keller [85] in the field of metal rolling. Although rolling metal has few process variables, the data is extremely difficult to interpret. The quality of the sheet metal is determined by the ability to control the gap size. There is no direct way to measure the gap size. Instead fluctuations in the roller pressure are used to indirectly estimate the gap width. The neural network was not only able to filter out the inherent noise, but was able to compensate for eccentricity in the rollers.

Neural-sensor have been used to assist traditional processes such as drilling and tuning. Tarng and Li [86] implemented a ART2 network to interpret signals from a dynamometer during boring. The system readily detected drill chatter, and was used to optimize boring rates. Dimla, Lister, and Leighton [87] developed a neuro-sensor to predict the sharpness of a cutting tool during turning. They used a single layer perceptron modeled in MATLAB to interpret a dynamometer and a three axis accelerometer. A single output represented tool condition; 1 and 0 for dull and sharp respectively. The output of the network is used to adjust the feeds and speeds of the lathe.

MATLAB has a very useful set of tools for modeling neural networks and has been employed by several other authors. Steele, Archuleta and Hooley [88] produced a back-propagation neural network in MATLAB to detect cavitation in hydraulic pumps. Their system uses the temperature, pressure and flow rate for input, and provides a signal 0 to 1 for none to full cavitation respectively. The output is used to adjust the pump motor spindle rate. The paper has a good discussion on the effect of various thresholds and the number of neurons in the hidden layer on prediction accuracy.

Chen, Khedkar and Weeme[89] developed neural networks for the manufacture of plastic. To maintain control over a polymerization, a chain-stopping agent is added in proportion to the so called melt-index. The authors first used a 3 layer neural network with 7 input, 7 hidden and a single output neuron. The inputs were the mole percentages of the various monomers and impurities and the chain stopper target. The single output was the desired melt index. The authors found the output naturally clustered around two index values. To increase performance two smaller 7,5,1 networks were trained, each optimized for one of the index ranges. The system switched between the networks according to the index

value. The decomposition reduced the training from 20,000 to 2,000 iterations, and increased performance by 10 to 20 percent.

In the manufacture of microchips, a fine gold wire is used to connect the silicon substrate to the terminal of the packaging. Traditionally, the only ways to evaluate the wire bonding process are visual inspection and destructive testing of random samples. Sun, Golden and Kwon[90] produced a neural network to predict the strength of the wire bonds based ultrasonic measurements taken during processing. First a three layer (16,4,1) network was trained with backpropagation to predict 85 percent of the defective bonds. By modifying the backpropagation error function, adjusting the slope of the sigmoid output function and pruning the network, accuracy was improved to 96.8 percent. Apparently pruning was most effective. The afore mentioned performance increase was reported after pruning the initial network from 68 to 48 synapses. The authors describe a simple method of pruning where the network is iteratively trained, with weak synapse being removed between cycles. The "weak" synapses are defined to be below a user selected value.

Neural networks have also been developed to predict the strength of weld joints in the manufacture of steel pipe [91, 92] . Three layer backpropagation networks have been designed for this purpose. The networks a basic in design but have a very high neuron count of 172, 204 and 102 in the input, hidden and output layers respectively. The input consisted of 86 pairs of welding current, and speed values. Each pair corresponded to a different time sample. The output of the network is 51 pairs of weld pool width and cooling time values. The cooling time is used to determine the metallurgic properties, and thus the strength of the weld.

A wave soldering machine a passes a circuit board over a bath of molten solder to form all connections at once. According to Coit et al. [93] the quality of the final product is very sensitive to the temperature of the board as it enters the wave. Coit has developed a series of neural networks to predict the thermal condition of the boards. The system uses five networks to predict mean temperature, temperature range, temperature standard deviation, mean and maximum temperature gradient. All five networks uses the same inputs, which are; 4 preheated temperatures, line speed, number of power and ground planes, card mass loaded, card mass unloaded, and card thickness. The published data of the networks predictions is within 5 percent of the actual values.

Khalid, Omatu and Yusof[94] produced an adaptive critic which uses a neural network to correct an adaptive fuzzy logic controller. The application was to regulate the temperature of a water bath used in chemical batch processing. The interesting aspect of this

application is the use of a second network to predict the state of the plant. Essentially, the fuzzy logic controller looks at the current state (temperature) and the prediction of the future state prior to making any control adjustments. Any discrepancy between the fuzzy logic controller's output and the measure response is used by the first network (critic) to adjust the fuzzy factors. Using this scheme, they claim to eliminated the need for mathematically modeling the plant, which is otherwise required for the fuzzy logic controller. In essence they replaced a traditional mathematical model of the process with a neural network model.

2.2.2 Neurocontrollers

Neurocontrollers are artificial neural networks whose output vector is used to directly control the process devices. Neurocontrollers are self-sufficient, thus they do not need additional algorithms, or control units to complete their functions. Neurocontrollers have the potential of predicting the states of non-linear systems and operating in complex control environments.

A neurocontroller for a refrigerator was developed by Graviss and Zurada [95]. The refrigerator had three on/off motorized components; fan, compressor and damper. They used 5 temperatures and 3 digital component control signals for inputs. The output of the network is an array composed of the three control signals for the afore mentioned components. The authors report 2 to 5 percent efficiency improvement depending on the operating conditions.

Carriere, Cela, Hamam[96] applied neural networks to controlling heat pumps. They demonstrated two approaches. The first used four networks to model each of the four heat pump components; compressor, evaporator, condenser, and expansion valve. The second approach, which appears to work better uses a single network to model the entire system. The output of the networks predicted the performance of the heat pump. The published data indicates the multi-layer network's predictions nearly exactly match the actual measured values.

Youlal et al. [97] designed an elegant neurocontroller for the distillation of methanol from a water based solution. The system used two neural networks; a neurocontroller and a neural sensor. The neurocontroller was a three layer (4,5,1) network whose sole output adjusted the valve that added more mixture to the distillation column. The system was able to decrease the duration of the transient phases and smooth the control inputs.

Perhaps the best example of neurocontrol was produce by Luecke and Edwards [98] with a fluidized bed power generator. Fluidized beds are combustors that allow the use of low grade fuels. They are notoriously non linear, and thus variation in fuel composition are

both a common and difficult problem. The three layer network controlled the air and fuel flow rates using; temperature rate, oxygen rate, temperature set point, residual oxygen set point, residual oxygen, and temperature for inputs. This is a text book example of a neurocontroller which demonstrated 53% and 61% improvement in the control of residual oxygen and temperature respectively.

Vega, Zamarreno et al. [99] developed a neurocontroller for the control of water sulfitation, a process used in the refinement of sugar. The process was traditionally controlled with an array of 8 PID controllers, each tuned for a different range. Depending on the pH of the system, the appropriate PID controller was selected. Using a three layer (3,6,1) network a valve was adjusted that regulated the flow of SO_2. The only inputs required were; e(t), e(t-1) and pH, where "e" and "t" represents error between measured and desired pH values, and time respectively. The single neural network was able to operated over the entire range. The neurocontroller was tested in both a simulation and the actual environment and always surpassed the PID array in performance.

Neurocontrollers have also been developed for traditional manufacturing process. Chiang et al. [100] developed a system to optimize the material removal rates during end mill machining. Their system employed two neural networks. The first network, a three layer design, converted the feed rates, and depths of cut into x and y forces, power consumption and surface finish. The second network took the output of the first and generated the optimal cutting parameters. Using the neurocontroller the authors report reducing cutting time by over 20%. The results are impressive, though one might question the original cutting condition against which such large productivity gains were realized.

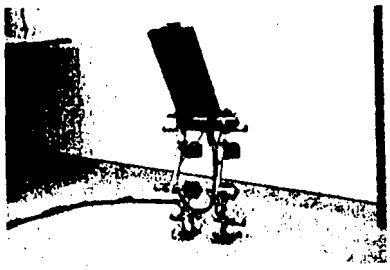


Figure 2.32 A front view of the biped robot with 10 degrees of freedom [25].

The CMAC paradigm which was described in section 2.2.7.3 is also commonly employed as a neurocontroller, however the CMAC is generally used for motion control. Since this paper is primarily concerned with process control, only one CMAC example will be discussed. Miller [25] developed a CMAC neurocontroller for a 10 axis biped robot (Figure 2.32). The robot had 21 sensors, four force sensors per foot, 10 joint sensors, and three orthogonal accelerometers. The controller was composed of three CMAC networks to control side-to-side lean, forward-to-back lean, and posture. The CMAC networks worked in conjunction with a *gait generator* which generated the servo motor control signal that the networks modified (Figure 2.332.34). With this control system they were able to teach the robot to take a few steps without losing its balance. More advanced motion will require additional nodes to be integrated into the network. This is not currently possible in real time given their 486 - PC computer constraints.

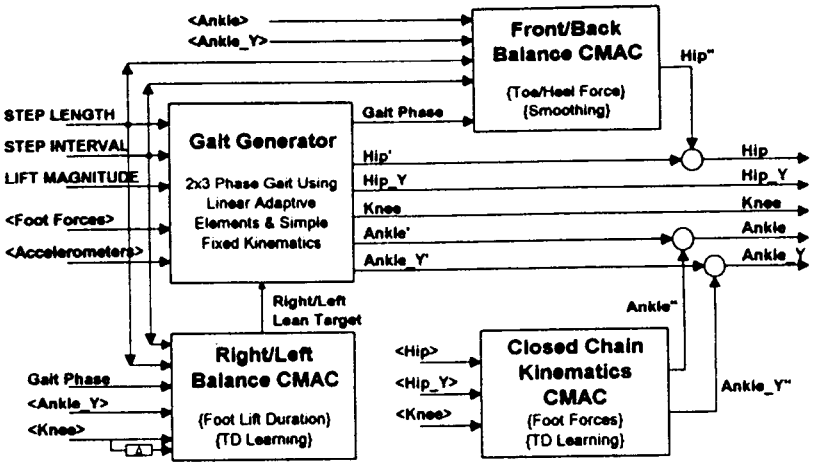


Figure 2.332.34 A block diagram of the biped gait control system, see text for details [25].

2.2.3 Adaptive Critic

Werbos [7] proposed a paradigm that allows the network to train itself during normal production. The system he describes, the adaptive critic, has two parts, a backpropagation network and a "critic" (Figure 2.35). The backpropagation network is first trained with whatever data is available. Once the system is brought on-line, it begins to cycle between two modes of operation. In the first mode the network will produce an output vector based on the current state of the input vector, just like a regular backpropagation network. This

output vector is used both by the system controllers to adjust the process and by the critic which calculates the error. The adaptive critic calculates the error between the input vector at time (t) and the output vector at time (t-1). Of course some manipulation of the two vectors may be necessary before error calculation is performed, but essentially the adaptive critic is comparing the output of the network to a known response. This error is used to calculate synaptic adjustments with a backpropagation training rule. After synaptic adjustments are complete, a new input vector will be fed into the network and a new cycle will repeat. The adaptive critic paradigm is the most popular neural control technique at the present time. Adaptive critics have been implemented to numerous processes, a few of which are listed below.

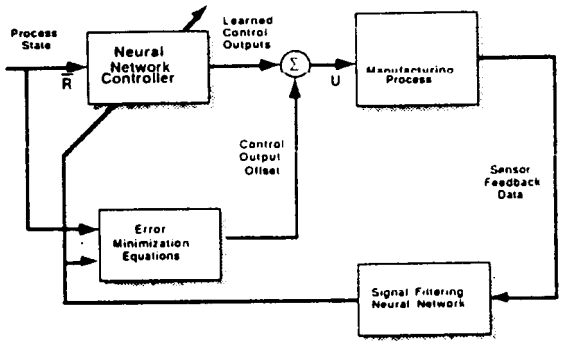


Figure 2.35 Basic organization of the adaptive critic architecture [106].

Park and Yang [101] developed an adaptive critic using a recursive network to predict the quality of glass that would be produced 8 hours from a given point in time. The adaptive critic that they developed was used by management to select the optimal processing conditions for the glass tank. Park and Yang performed a sensitivity analysis to determine which parameters could be removed from the input vector without substantially degrading the quality of the output. This work is particularly impressive considering the seemingly unmanageable complexity of a glass tank [102]. Although this application did not employ the neural net to directly control the glass tank, it is still one of the best papers describing industrial adaptive critics. Pao, Phillips, and Sobajic [103] discuss a nonspecific system that was nearly identical to that of Park and Yang, but in much greater detail.

The adaptive critic architecture is suited for use as a process controller for composite manufacturing. White [104] developed the adaptive critic controller based on Werbos' theories [7] for the purpose of improving the yield rates of large autoclave composites.

Unlike the adaptive critic of Park and Yang, White's directly controlled the autoclave. In a later collaboration with Sofge, White developed an adaptive critic for the filament winding of complex geometries, such as those used for aircraft planforms (Figure 2.36 and Figure 2.37) [105,106]. The authors discuss and compare their adaptive critic against a proportional feedback controller and a PID controller. They later developed an adaptive critic that could adapt to new conditions such as the loss of a wing on an F-15 simulator, reducing its crash rate from 100% to 50% [104].

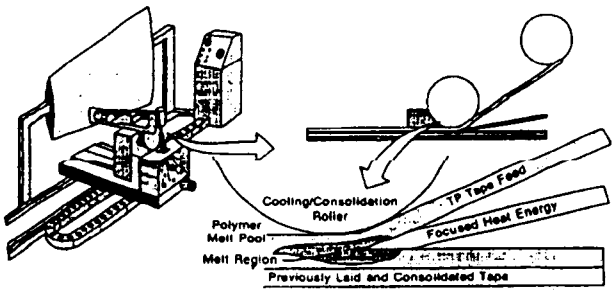


Figure 2.36 Filament winding application that was controlled with an adaptive critic [105].

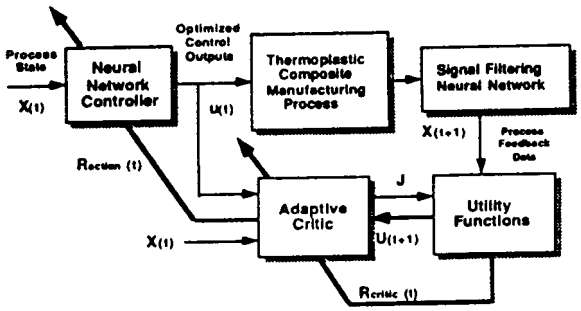


Figure 2.37 Adaptive critic filament winding system architecture [106].

Li and Wu [107] developed an adaptive critic controller with a fuzzy logic controller, a derivative controller and a neural network. The main controller is comprised of the fuzzy logic and derivative controllers. The neural network is configured to optimize the membership functions of the fuzzy logic controller. The input to the three layer backpropagation network is the error term e which is also the primary input to the fuzzy logic

controller. The output of the network is a vector k* which defines fuzziness. The authors claim this approach can be used in the absence of analytical models as the control scheme adjusts its own fuzzy factors.

Katsumata et al. [108] developed an adaptive critic for systems with long dead times. Such systems require I-PD control used in conjunction with a state predictor. The authors devised a scheme where two neural networks are integrated with a I-PD controller. The first network is trained to be a state predictor, replacing the math models typically used. The second network adjust the gains of he I-PD controller. Both networks are simple three layer feed forward backpropagation designs. The published results suggest the system is more effective than even a tuned I-PD controller.

Zhao[109, 110] developed a similar method for PID controllers. Zhao's method uses a single three layer neural network to tune the PID controller. The controller is designed for uncharacterized system. The author applied the technique to an industrial furnace an presented favorable result under limited, controlled conditions.

Lin, Cheng, and Kim [111] have done some recent work to improve the learning rate of adaptive critics by replacing the backpropagation neural network traditionally used with adaptive critics with a radial basis function network (RBFN). RBFNs are composed of "basis" neurons that have basis functions to modify their output. The basis neurons are more powerful because fewer basis neurons are required to represent a given pattern compared to traditional artificial neurons

2.2.4 Summary

This section described the use of neural network in smart processing. The three types of smart controllers; neural-sensors, neurocontroller and adaptive critics have been shown to be capable of operating with non-linear and noisy data. Neural sensors are extremely useful for interpreting process signal an are used in virtually ever field of manufacturing. Neurocontrollers are an emerging technology and have demonstrated the ability to increase product quality and decrease processing time. The adaptive critic architecture is suitable both for direct use and for controller auto-tuning. This section actually completes the review of engineering neural networks. The next section focuses on neural science in an attempt to describe features and functionality that have not yet been integrated into engineering models.

2.3 Neural Science

The following section on neural science is comprised of three parts. The first describes the basic element of all neural networks, the neuron. The neuron is presented here in considerable detail as it is necessary to understand that multiple mechanisms are responsible for the generation and transmission of signals in the brain. Many of these mechanisms are not present in the McCullough-Pitts neuron as their significance only becomes evident in massive scale systems. The information presented on the neuron also serves as the foundation from which the chromosomal and structural parameters are developed in Chapter 3. The second part of this section describes the brain. The topics addressed concerning the brain relate to the organization of functional regions called cortices and serve as the foundation for the discussion of integrating multiple cortices in Chapter 5.

Neural science is predominately concerned with developing a theory that describes how the interaction of neurons in biological organisms produces the behaviors seen in nature. The majority of the work in neural science is strict modeling of biological organisms and actual physiological investigation. A smaller segment of neural science is dedicated to developing math models based on the observations of the first group. The following brief review describes the basic areas of neural science and discusses several key papers and their contributions to a holistic understanding of the elusive cognitive process.

2.3.1 Cell Types of the Nervous System

A discussion of neural cell types is presented here since the classification of cells by type is a useful means of separating functionality. Cell types are used extensively in Chapters 3 and 4 for the development of the growth simulation program.

The nervous system of living organisms is composed of two basic types of cells, glial cells and neurons. There are many different types of neurons (Figure 2.38) and three different types glial cells (Figure 2.39). The neurons are the signal processing elements that are so specialized for this task that they cannot perform their own metabolic functions. The astrocytes, one type of glial cells, function as the support team, not only providing the neurons with nourishment, but also removing their by-products and restoring ion concentrations so signal logic can be maintained (Figure 2.39c). The second and third type of glial cells form the myelin sheaths, essentially the insulators for the conductive neurons. The myelin sheaths are formed by oligodendricytes and Schwann cells, depending on whether the neuron is located in the central or peripheral nervous system respectively (Figure 2.39a, b).

The value of the myelin sheath is demonstrated by the *shiverer* mutation which causes a myelin deficiency and loss of motor coordination because the control signals become jumbled as they pass down the nerve cords [112].

There are two basic types of nerve cells, sensory and signal. The sensory neurons are those that contain a sensor on one end and a junction on the other. There are a myriad of sensor neurons, but the most recognized types are visual, olfactory, auditory, and gustatory receptors (Figure 2.40). Signal neurons are equally diverse but can be classified into two basic types, inhibitory and stimulatory. The inhibitory neurons function by drawing off the charge of other neurons, while the stimulatory neurons provide charge to other neurons. The exact distributions between these two types of neurons is unknown, but it is thought to be approximately 2/3 stimulatory, 1/3 inhibitory. Neurons are often classified by their cytoarchitecture, dimensional characteristics such as axon length, number of dendrites, cell body size, etc.

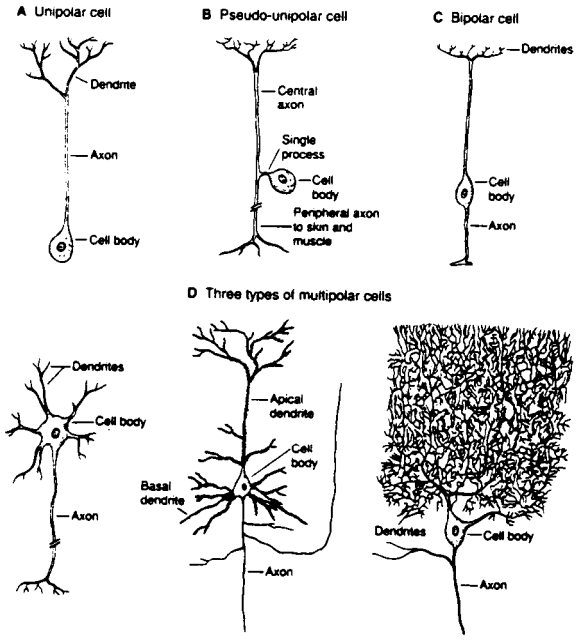


Figure 2.38 General neuron types. (a) Invertebrate unipolar cell, (b) spinal cord pseudo-unipolar cell, (c) bipolar cell of the retina, (d) multipolar cells; **left** spinal motor neuron, **center** hippocampal pyramidal cell, **right** cerebellar Purkinje cell.

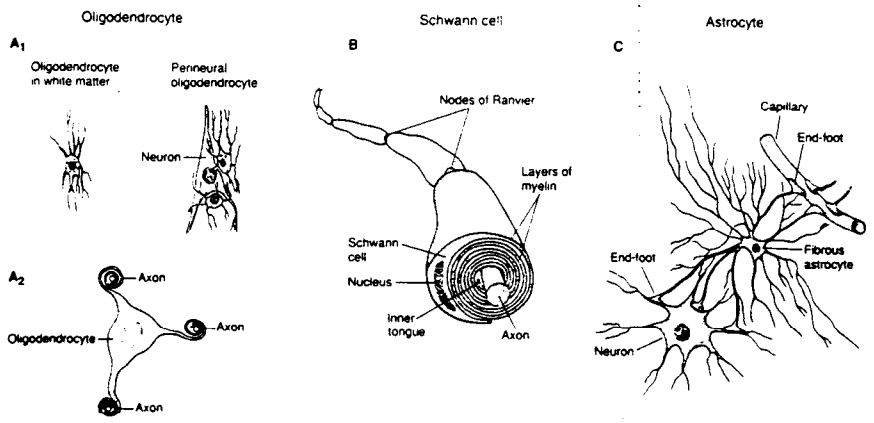


Figure 2.39 Glial Cells. (a) 1. Oligodendrocytes in the central nervous system form myelination in white matter, **left**, and surround gray matter, **right**. (a) 2 a single oligodendrocyte forms many myelin sheaths. (b) Schwann cells of the peripheral nervous system form myelination over long neurons, (c) star shaped astrocytes seen connecting neurons to capillaries in gray matter [112].

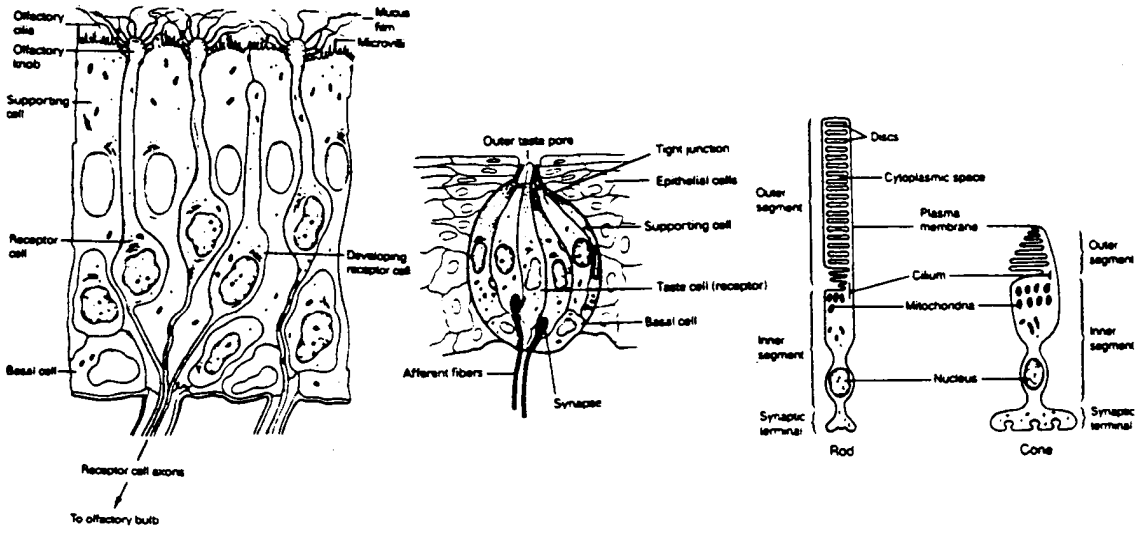


Figure 2.40 Sensory neurons. From left to right, olfactory receptors, gustatory receptor (taste bud), black/white photo receptor and color photo receptor [27].

2.3.2 The Neuron

The following discussion introduces the neuron and the mechanisms by which it transmits signals through the nervous system. A detailed understanding of these mechanisms is necessary to allow the accurate modeling of neural systems. The section begins with an overview of the neuron and proceeds in greater depth to describe those mechanisms specifically related to signal transmission.

Neurons have three functional elements: dendrites, axons, and the axon hillock (Figure 2.41). As mentioned before, input signals are received in the dendrites, then summated in the axon hillock, and the resulting signal is transmitted down the axon to other neurons. The cell body and nucleus have very little to do with the signal transmission and are often neglected from analysis. The cell body does actually receive some signals from other neurons, but these signals are analytically treated as if they were from other dendrites. The dendrites and axons are extensions of the cell body through which signals propagate. A synapse is defined as the junction where a signal is transmitted from the axon of the presynaptic neuron, to the dendrite of the postsynaptic neuron.

2.3.2.1 Overview of the Interneural Electric Signal

Neurons produce an electric current through the selective movement of ions, not through the flow of electrons. Depending on the location along the neuron's body, the signal is observed to be analog or digital. The signal begins in the axon hillock and then propagates down the axon toward other neurons. The axon hillock produces a digital pulse once the voltage of the cell body reaches a certain threshold level. When the axon hillock fires, the voltage of the neuron is decreased and can only be raised by the stimuli of other neurons. The axon hillock effectively performs an analog-to-digital conversion. The rate at which the axon hillock will fire is proportional to the rate the neuron regenerates the threshold value, which is proportion to the amount of stimulation it receives. The digital pulse width modulated (PWM) signal is distributed to other neurons through the branching of the axon terminal. Each neuron receives stimuli from several thousand other neurons. The PWM signals are converted to analog signals in the dendrites, so they may be summed in the cell body in an attempt to reach the threshold value.

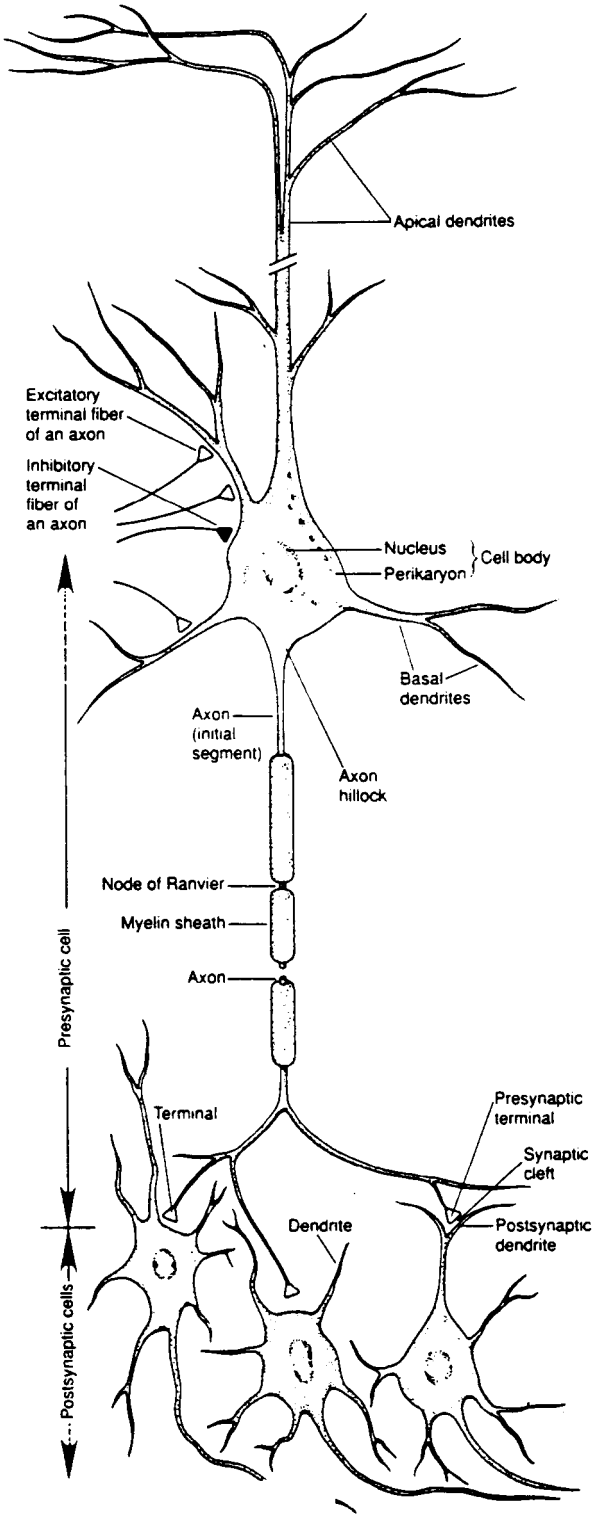


Figure 2.41 Typical vertebrate neuron showing key elements [27]. See text for details.

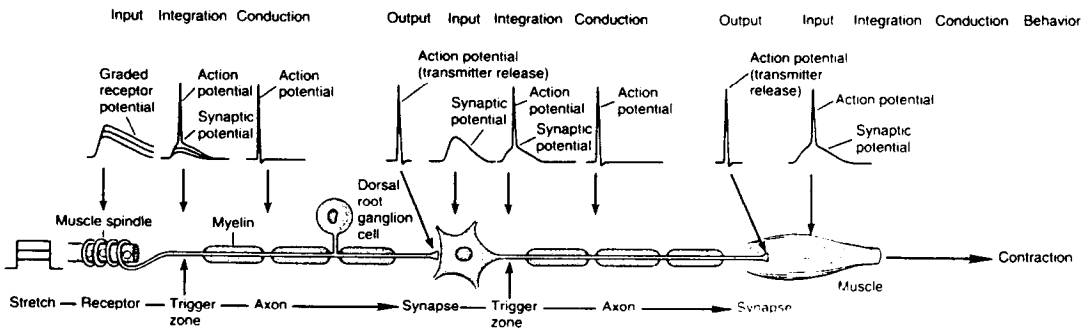


Figure 2.42 Schematic illustration of an electrical signal as it passes through neurons [27].

2.3.2.2 Electrical Properties of Neuron Terminals

Axons and dendrites not only look like wires, they behave like wires. Cable theory, which describes a wire as a series of resistors and capacitors, has been successfully used to describe neurons. The results of this description suggest that a neuron can be represented by the resistance and capacitance of the membrane and cytoplasm. Standard notation is used: R = total resistance, C = total capacitance. Lower case *r* and *c* are used to represent specific, per unit values. The subscripts *a* and *m* denote cytoplasm and membrane, respectively. The voltage measured down the length of a neuron terminal can be represented as a function of time *t*, or distance as *x*, in equations 2.7a and b, respectively.

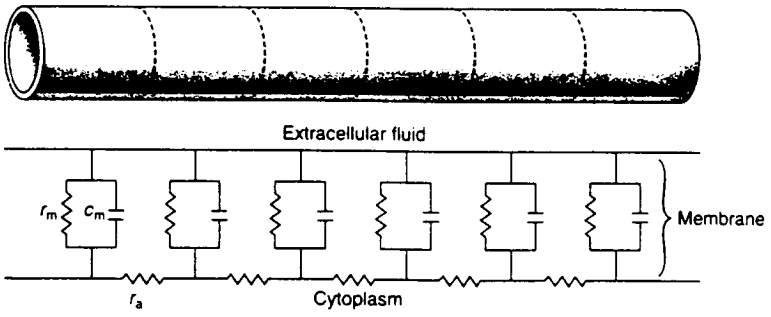


Figure 2.43 Representing a neuron terminal with cable theory [27].

$$\Delta V_m(t) = I_m R_m (1 - e^{-t/\tau}) \quad 2.7a$$

$$\Delta V_m(x) = \Delta V_o \, e^{-x/\lambda} \quad 2.7b$$

$$\lambda = \sqrt{\frac{r_m}{r_a}} \quad 2.7c$$

$$\tau = r_m c_m \quad 2.7d$$

I_m is the total membrane current, V_o is the source voltage at (t = 0), λ is the length constant, and τ is the time constant. τ is defined to be the time for 37% of the initial current, I_o to be lost. Similarly, the membrane length constant, λ, is defined to be the length that a signal travels before it loses 37% of its original voltage, V_o. Another term often mentioned is the passive transmission velocity, υ, which describes the rate that a signal propagates through a neuron terminal. υ is inversely proportional to r_m and c_m as seen in equation 2.7e.

$$v \propto \frac{1}{r_a c_m} \quad 2.7e$$

2.3.2.3 The Propagation of Neuron Signals in Axons

As the signal propagates down a neuron terminal, its voltage and current are dissipated. This power loss has a pronounced effect on the transmission of neural signals. The simplest form of neural transmission, *passive transmission*, occurs in the primitive, unsheathed neurons typical of invertebrates. The neural terminal can be viewed as a tube, surrounded by an infinite ion reservoir, filled with an ionic gelatin, with 'poppet' valves evenly distributed across the surface. The gel, in this case, is the cytoplasm, which allows little longitudinal diffusion in the time interval under consideration. The ionic imbalance causes the poppet valves (channels) in the immediate area to open, allowing ions to flow in from the infinite reservoir to equilibrate the cytoplasm. This creates a "depolarized" wave front, where the channels ahead of the wave are opened as the depolarized or "active" regions approach, and the channels in the wake close as the membrane repolarizes, and return to the resting potential (Figure 2.44).

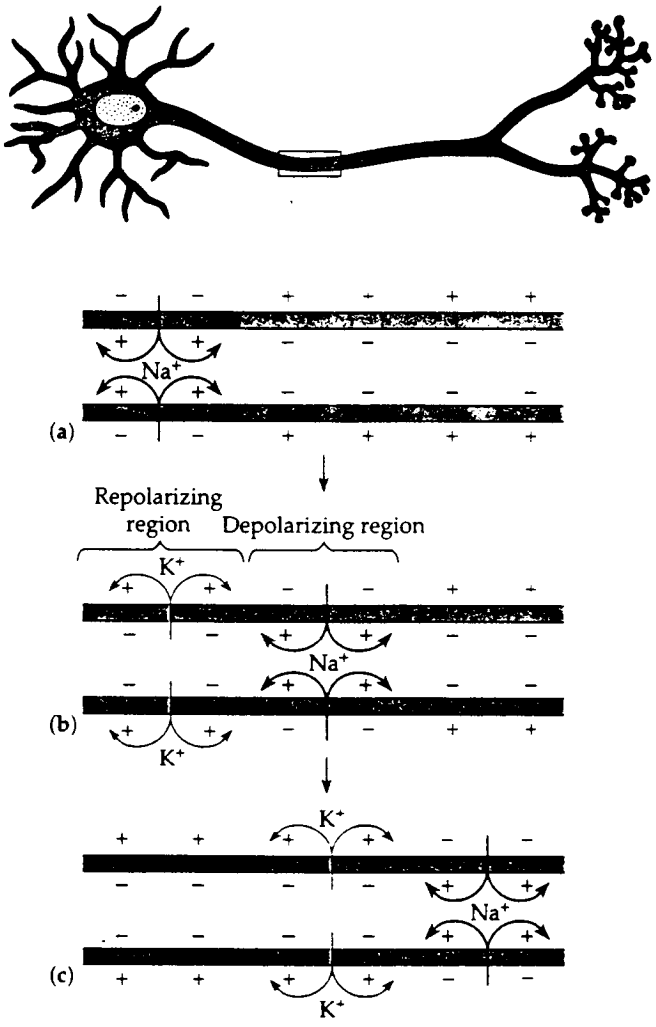


Figure 2.44 Propagation of the action potential. (a) As the sodium ions flow inward across the membrane, the resulting depolarization spreads to the portion of membrane just ahead of the impulse. (b) Depolarization of this axon segment then initiates an action potential at that site which in turn depolarizes the next portion of the axon. In this manner, the action potential progresses along the axon. (c) As the action potential propagates, the preceding segment repolarizes [113].

Passive propagation will continue as long as the membrane potential of the active region is above the threshold of the ion channels. Since the membrane potential is a function of distance (equation 2.7b), there is a critical distance after which the membrane potential is insufficient to open any additional channels, and the depolarized wave is arrested. This critical distance is related to the length constant λ, though not equal to it. From this

relationship it can be seen that the critical length can be increased by increasing the membrane resistance and/or decreasing the resistance of the neuron terminal core. The resistance of the neuron terminal can be decreased by increasing the diameter of terminal, which increases the cross sectional area, (recall $R \propto {}^{\rho}/_{A}$ [114]), as demonstrated by the giant axon of the pink squid. However, Krandel states that a higher per-volume gain in λ can be achieved by increasing r_c through myelination, as demonstrated in mammalian vertebrates.

2.3.2.4 Myelination

Myelination increases the membrane resistance by effectively increasing its thickness. This thickening is accomplished by glial cells that form a 'jelly roll' out of their own membrane around the axons. Myelinated neurons do not use passive depolarized wave propagation to transmit their signals; instead they use an active, regenerative system called *saltatory conduction* (from Latin *saltare*, to leap). The signal propagates down the axon in little jumps, or leaps. Each leap is the length of the myelination , i.e. 1-2 mm. The myelination appears as small cylindrical sections, separated by a *nodes of Ranvier*. The nodes of Ranvier behave as repeaters. When a current is detected, a node of Ranvier will produce a new signal. This process is repeated. The pulse that is generated at the axon hillock is regenerated at each node of Ranvier. Regardless of the pulse frequency, the power of each pulse, $P = \int_{t_0}^{t} VI \frac{dv}{dt}$, is essential the same, thus signal integrity can be maintained over large distances.

In addition to increasing the range that a neuron may transmit, myelination greatly improves the transmission velocity when compared to unsheathed passive transmission (Figure 2.45). Saltatory conduction receives its name because each myelin sheath behaves as a single current loop. When a pulse is generated at the proximal end of a myelin sheath, a current is detected at the distal end and initiates local depolarization at the node of Ranvier, without depolarizing any of the sheathed area of the axon (Figure 2.46). In essence sheathed neurons transmit their signal as a series of current loops that jump, or leap, between nodes.

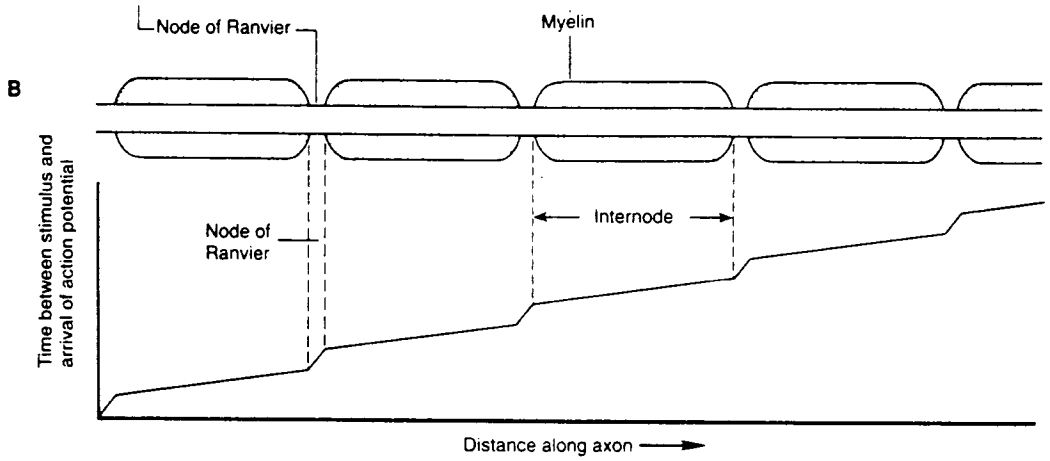


Figure 2.45 Distance vs. time plot and corresponding schematic. Note the internode velocity is much greater than the intranode velocity [27].

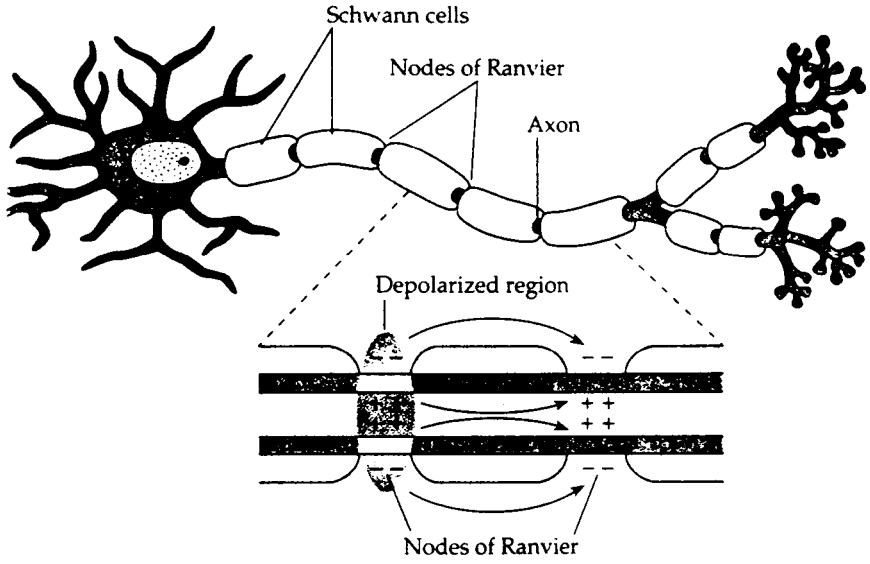


Figure 2.46 Saltatory conduction jumps from one node of Ranvier to the next without depolarizing the intermediate membrane [113].

2.3.2.5 Division of the Presynaptic Current and Voltage

The last node of Ranvier is referred to as the terminal node, after which the signal is divided among the various axon terminals. The axon terminals branch out, serving as the presynaptic connections to the postsynaptic neurons. Unlike the McCullough-Pitts model that does not describe any presynaptic division, in actuality, a division of current does occur. The division of current is based on the resistance of the various axon terminals, which is inversely proportional to their diameters.

After the terminal node, each sub-signal, s_i, begins to degrade as the axon terminal operates by passive transmission. Both the voltage and the current of a given sub-signal s_i would have decreased by the time it reaches a synapse as described by equations 2.7a and b. This indicates that short, thick terminals have a much greater probability of initiating a post-synaptic response than long, skinny neuron terminals.

2.3.2.6 Cytoskeletons

Since the physical dimensions of the neuron have an impact on the signal transmission logic, it is worthwhile to discuss the parameters which determine the cytoarchitecture. The shape of the cell is determined by the cytoskeleton, a network of microtubules that resembles an iso-grid. The shape of the cell is modified by adding or subtracting protein segments to and from the microtubules. In nerve cells these microtubules are in a constant state of flux described by three conditions: true equilibrium, treadmilling, and dynamic instability (Figure 2.47). True equilibrium is the condition where proteins are being added and subtracted from the microtubule at equal rates. Treadmilling occurs when proteins are being added to one end and subtracted from the other end at equal rates. Unlike the first two conditions where the length of the microtubule remains constant, dynamic instability describes a net gain or loss of proteins, translating to growth or shrinkage respectively.

It is suspected that modification of the cytoskeleton plays a crucial role in transmission logic since nerve cells at any given time have at least half of their microtubular proteins in the monomer form, which is a much greater percentage than for any other cell type. Hammeroff, Dayhoff, and Koruga [115] proposed a paradigm for network training based on the modification of an artificial cytoskeleton, treating the proteins as automatas.

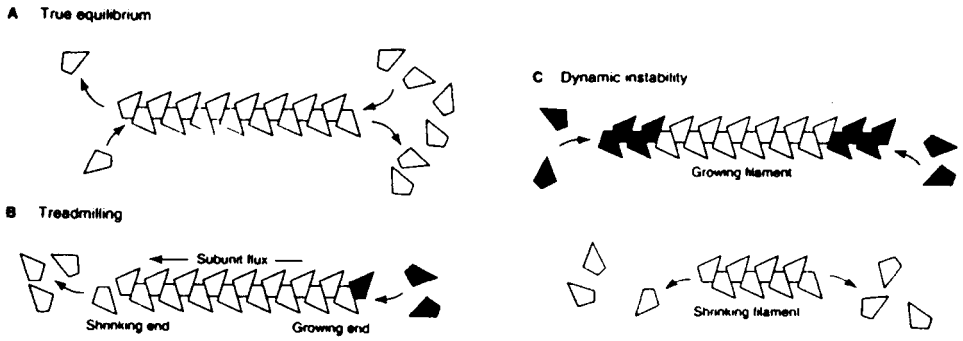


Figure 2.47 Microtubule modification. (a) True equilibrium, (b) treadmilling, and (c) dynamic instability; growth, **top**, and shrinkage, **bottom** [27].

2.3.2.7 The Synapse

A synapse is the junction of nerve cells. Nerve cells do not physically connect, rather the ion channels of the pre-synaptic and post-synaptic neurons align. The actual biological synapse has very little resistance. When an incoming pulse reaches the synapse, the change in membrane potential initiates the opening of some of the ion channels to reduce the membrane potential. The flow of sodium can either be with the extracellular fluid or with another cell. If the flow of sodium ions is with another cell, then successful interneuron transmission has occurred. To increase the amount of current that the adjacent cell receives, the efficiency of the synapse can increase, or the amount of available sodium ions can increase. The first scenario requires a modification of the synapse, which is known to occur, while the latter would require a physical change in the diameter of the neuron terminal.

2.3.2.8 Synaptic Modification

Synapses, and neurons in general, have several different types of channels. In addition to the primary sodium and potassium channels used for direct neural transmission, there are several auxiliary ion channels, such as magnesium, that assist in learning. There are also chemical channels that serve to regulate neuron transmission.

Long-term potentiation is initiated by the opening of magnesium channels which temporarily increases the efficiency of the synapse. Once the sodium ion channels are open, the release of other chemicals, including NMDA, may lead to growth of the axon terminal or other features related to the synapse, permanently increasing the amount of power being transferred. Neurologists believe that actual physical growth does occur during learning

based on the fact that half of the cellular protein that comprises the cytoskeleton of neurons exists in its monomer form at all times.

2.3.2.9 Postsynaptic Current-to-Voltage Conversion

Once the charge is transferred across the synapse, it becomes more or less static in the cell body. Until the charge exceeds a threshold level, the axon hillock will not fire, thus there is essentially zero current. Because of this phenomenon, neurologists refer to the charge as the *cell-body (somatic) potential*. Each time a stimulus reaches the cell body, the voltage of the stimulus is added to that of the cell body (Figure 2.48). The time period during which the stimulus has an effect on the firing of the neuron is related to the time constant τ. The postsynaptic activation potential is produced by the accumulation of overlapping stimuli. There are two types of constructive signal overlap, spatial and temporal. Temporal summation occurs when the time interval between stimuli from the same or similar axo-dendritic location is less than the time constant τ. Spatial summation occurs when the separation between two stimulating synapses is less than the length constant λ, as seen in Figure 2.49.

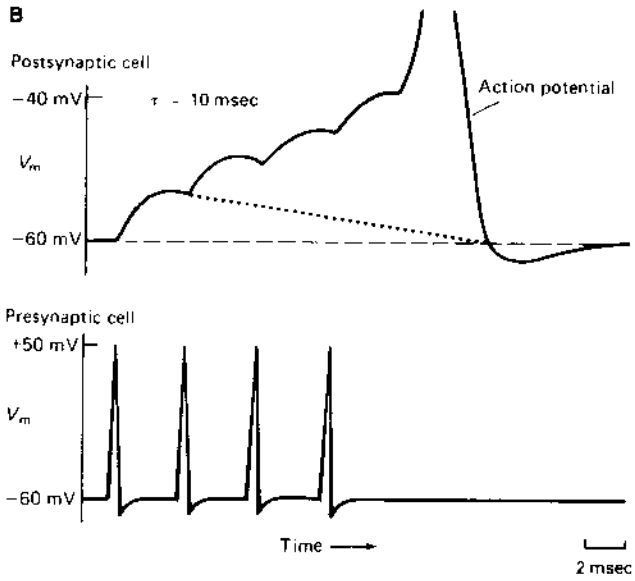


Figure 2.48 Integration of presynaptic stimulus in the postsynaptic neuron. Note that once the cumulative voltage exceeds the threshold, (-40mV), the neuron fires [27].

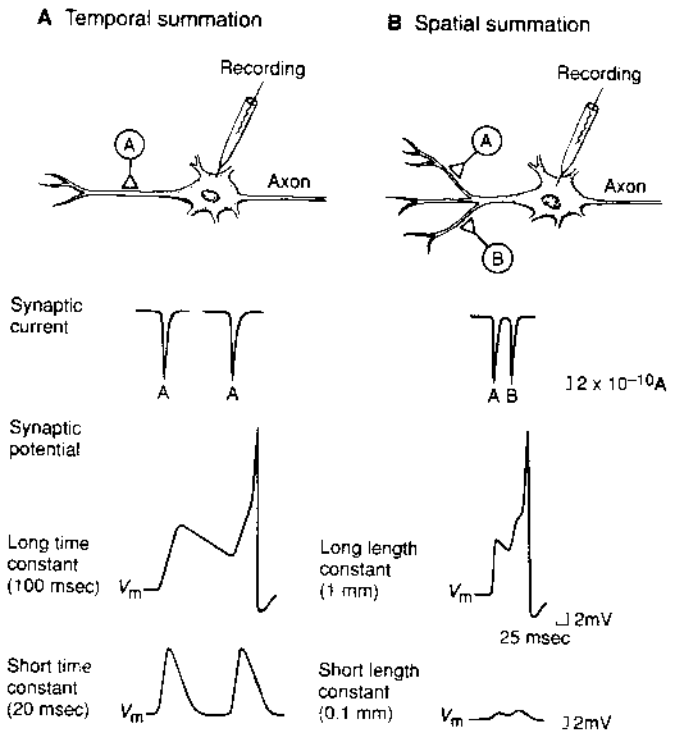


Figure 2.49 Temporal and spatial summation. The effect of membrane time and length constants on temporal summation and spatial summation [27].

2.3.2.10 Inhibitory Neurons

Inhibitory neurons have three functional mechanisms: hyperpolarization, opening of chloride channels, and shunting. The first mechanism describes the generation of an activation potential inside the inhibitory neuron that is more negative than the resting potential of the neuron being inhibited. This hyperpolarization is affected by the simultaneous opening of chloride and potassium channels on the surface of the inhibiting neuron. Chloride, which carries a negative charge rushes into the neuron while potassium, which carries a positive charge, leaves the neuron. The influx of negative charge, combined with the efflux of positive charge, gives the inhibitory neurons a strongly negative charge. The 'negative' inhibitory signal subsequently lowers the somatic voltage of the inhibited soma (cell body), along with the other input signal. The hyperpolarized signal has a deactivating effect, decreasing the likelihood that the membrane voltage of the inhibited neuron will reach the action potential. The second mechanism opens the chloride channels on the membrane of the

inhibited neuron. This second mechanism works by increasing the inhibited neuron's permeability to chloride. The increased permeability serves to clamp the inhibited neuron's membrane potential near E_{Cl-}, which is below the threshold potential. The last mechanism increases the conductance of the inhibited neuron's membrane which effectively shorts-out, or shunts the cell.

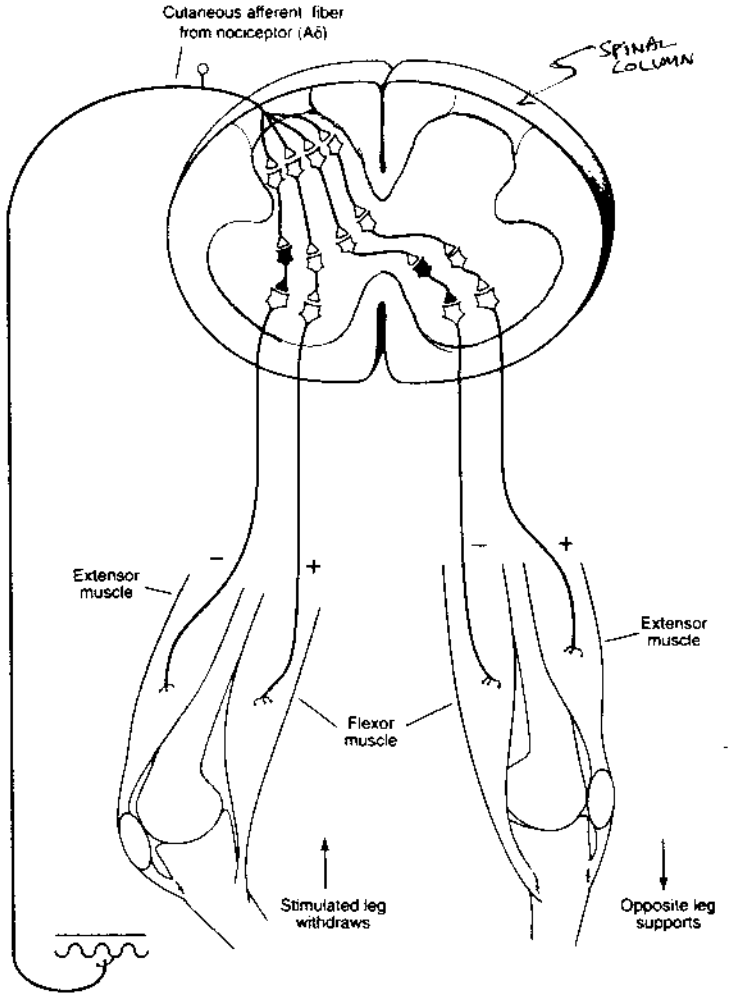


Figure 2.50 The role of inhibitory neurons in motor functions [27]. See text for details.

Inhibitory neurons play an important role in both motor control and signal separation. In the withdrawal reflex, which is responsible for retracting a limb from a negative stimulus,

a single sensory neuron drives the simultaneous contraction and relaxation of muscle groups. A surface receptor transmits a stimulus through the cutaneous afferent fiber to the spinal column (Figure 2.50). There the signal is processed by the gray matter in the spinal column to retract the stimulated leg and simultaneously send signals to the other leg in order to support the weight being transferred from the stimulated leg. This coordinated motion requires the activation and deactivation of opposing muscle groups. Inhibitory neurons are responsible for deactivation.

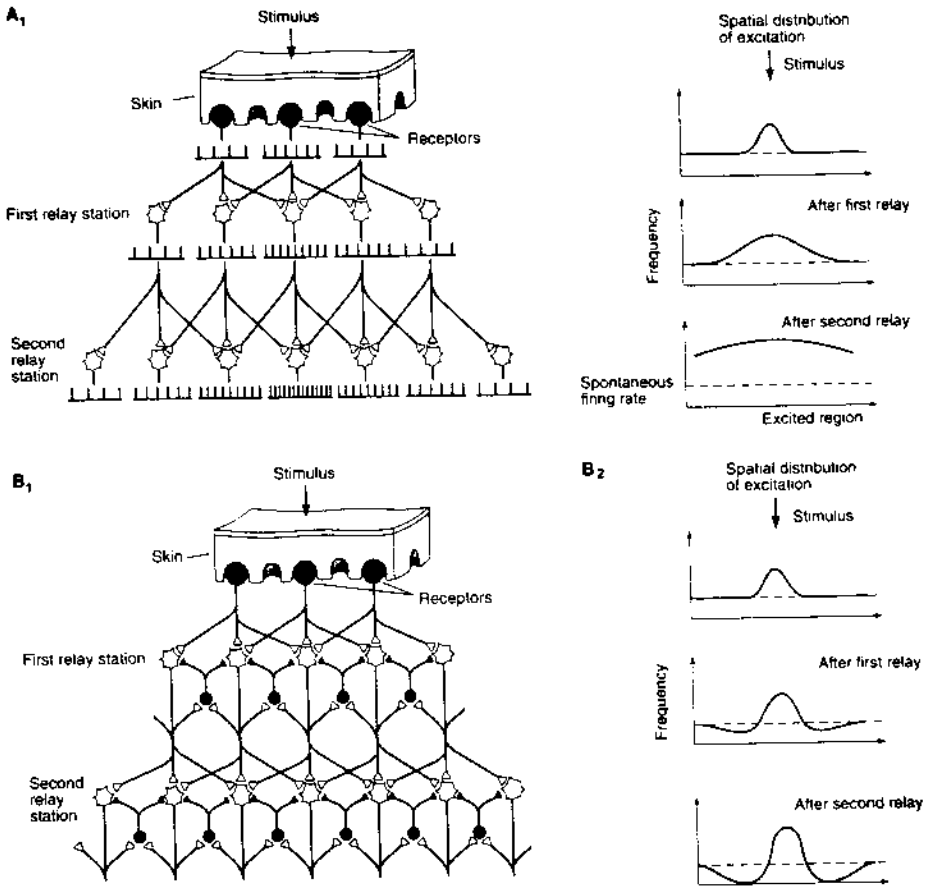


Figure 2.51 The role of inhibitory neurons on stimulus localization [27]. See text for details.

Inhibitory neurons are also responsible for the localization of surface stimuli. As seen in Figure 2.51, skin, which contains numerous receptors, is stimulated by a point source. The output of the receptors is depicted in pulse width modulation (PWM) format directly below each element. Without lateral inhibition, the stimulus propagates through the first and

second relay stations to produce a broad dome of activation seen on the right of Figure 2.51a. In contrast, with lateral inhibition the single point stimulus is represented by a sharp *Mexican hat*-like region of activation. This Mexican hat function facilitates the localization of the stimulus. The need for localizing the stimulus is apparent in Figure 2.52, where two stimuli, separated by a small distance, can be resolved with lateral inhibition, but appear to be a single, large stimulus without lateral inhibition. A three dimensional representation of the Mexican hat is given in Figure 2.52.

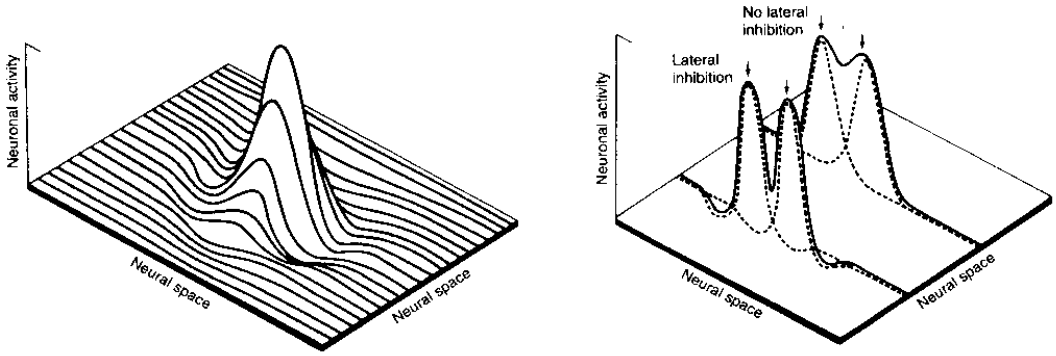


Figure 2.52 The localization of stimuli produces the Mexican hat shown in three space on the **left**. The Mexican hat function serves to separate two point stimuli, **right** [27].

The ability to form the Mexican hat through the integration of excitatory and inhibitory neurons is the most important ability of nervous tissue. Literally every single representation that occurs in the brain, be it sensory, motor, perceptual, or conceptual, is based on the ability to form these tight neuronal groups defined by the Mexican hat function (section 2.5).

In summary, multiple mechanisms such as passive conduction and saltatory conduction, are responsible for the transmission of signals throughout the nervous system. These mechanisms are both structural and chemical in nature, and can produce temporal or spatial delays and/or increase or dissipate signal intensity. These mechanisms are utilized to separate different signals. The mechanisms have been discussed so that they can be used to develop cytoarchitectural and electrical parameters in Chapter 3. Having concluded a discussion on the single neuron, the following discussion describes the collective interaction of the neurons and the brain.

2.3.3 The Brain

There are three main issues concerning the brain that need to be discussed to allow the development of a massive scale neural network. The first describes the physical arrangement of the neurons in the brain and serves as the foundation for approximating the cerebral cortex as a uniform, three dimensional grid. The second issue addresses the organized, six strata structure of the cerebral cortex. The definition of the term layer used in this paper is different than the traditional definition found in biological descriptions of neural physiology. Therefore, in this work, the term strata replaces the term layer as used in a biological references. The six strata structure is the foundation for the generalized neocortex model presented in Chapters 3 and 4. The last issue concerns the interconnection of the cortices and the primary pathways of signal transmission that are evident in the brain. These pathways serve as guidelines for the interconnection of artificial cortices in Chapter 5.

2.3.3.1 Overview of the Brain

The nervous system, for all its amazing complexity, is actually very well ordered. Unfortunately it is ordered by means of its development process instead of the orthogonal Euclidean space with which engineers are comfortable. The nervous system originates as a spinal tube in the early stages of the embryo. The tube grows in both directions, forming the spinal cord on one side and the brain on the other side. The brain is formed as the tube continues to grow, but it is physically constrained and thus folds on itself (Figure 2.53). The initial folds develop into the various cortices of the brain. As the brain continues to develop, each cortex forms additional folds which eventually give the brain its characteristic, undulated surface.

It is important to realize that the brain begins as a thin film that folds and folds upon itself. In the early stages of development these films are very close to each other, and the various cortices can easily interconnect. As the brain continues to grow in size, the interconnections grow as well. Even though the brain is a solid structure, it still maintains evidence of its simple origins. A cross-section of the brain reveals a thin, gray strata that runs along its perimeter (Figure 2.54). The center of the brain is called the brain stem. The brain stem is a series of relay points at the end of the spinal cord. These relay points are referred to as subcortical regions and connect the rest of the brain to the spinal cord. The space between the brain stem and the gray strata is filled with white matter. The white matter consists of axons, the interconnecting wires of the brain. The gray strata is referred to as gray matter and

consists of the cell bodies of the neurons, the signal processing elements of cells. Despite its undulated surface, the brain is essentially a 2-dimensional film that is wrapped over an undulated surface. These undulations maximize the surface area and thus allow a greater number of neurons to be contained within the skull [27].

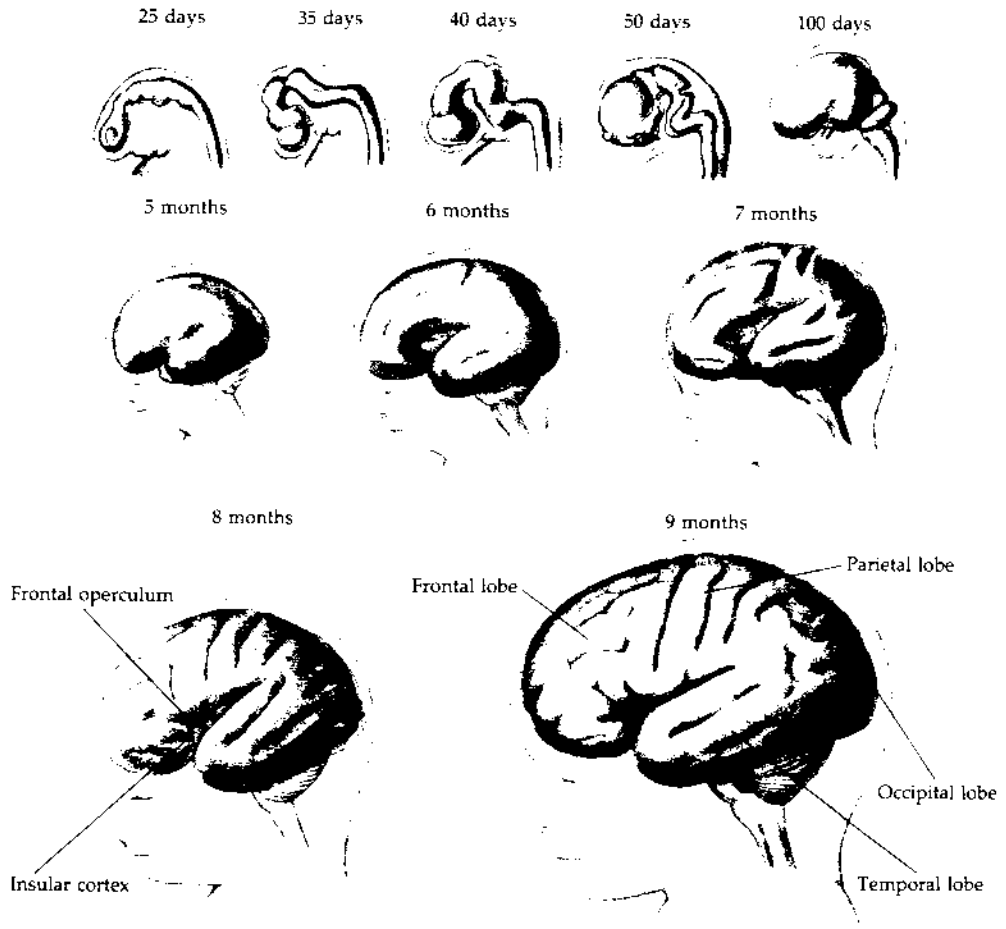


Figure 2.53 The development of the human brain [116].

The brain is divided into various functional regions called cortices (Figure 2.55). The advanced, or modern cortices are located in the neocortex. Although each cortex performs a different function, those in the neocortex all have the same basic structure. A closer examination of the peripheral gray matter reveals a six strata, concentric structure. This basic six strata structure is found in all cortices in the cerebrum (neocortex), although the relative

thickness of the strata varies among the different cortices. These strata are distinguishable from one another by the nerve cell types they contain. Although volumes of information exist describing the entire brain, this paper is primarily interested in the structure of the neocortex. Nine neuron types have been identified, as seen in Figure 2.56. All information that enters the cerebral cortex projects into stratum 4 from the thalamus, one of the subcortical regions of the brain stem. The thalamus serves as a junction site and is connected to all other centers of the brain.

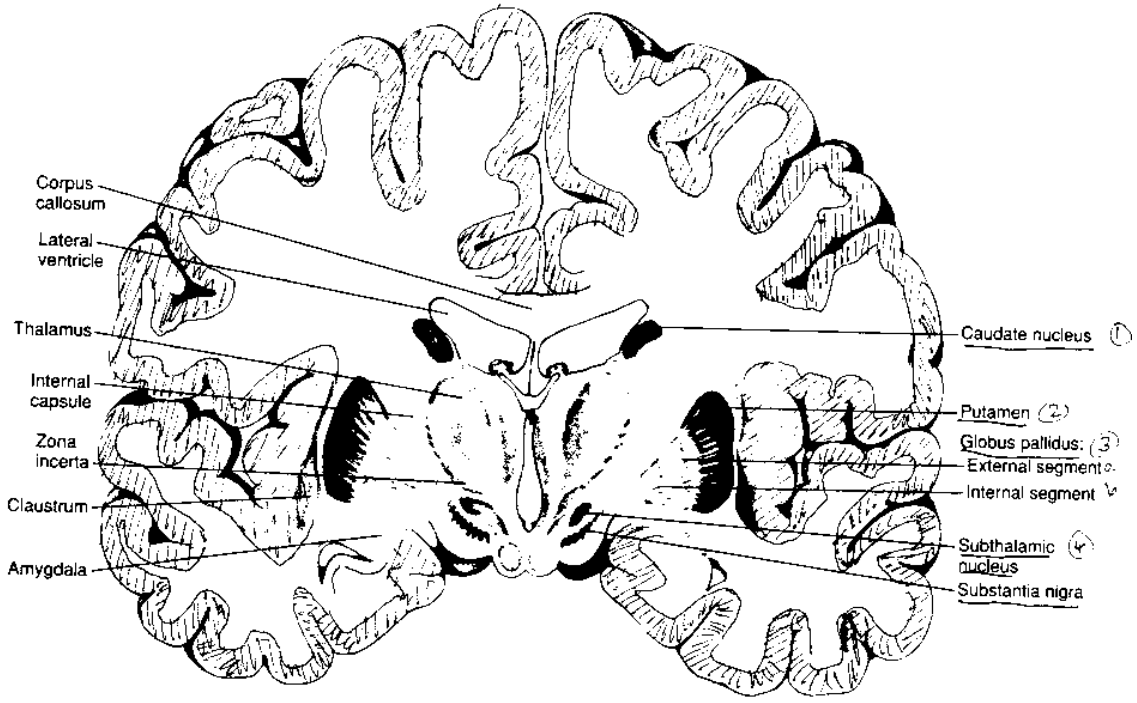


Figure 2.54 Cross section of the brain showing perimetrical gray matter, interior white matter, and subcortical regions [116].

There are four types of output from strata 2,3,5, and 6. Stratum 2 contains small pyramidal cells and is believed to form direct intercortical communications. The pyramidal cells in stratum 3 are moderate in size and form direct transhemispherical intercortical communications. Stratum 5 has the largest pyramidal cells which go through the basal ganglia and down the spinal cord. The pyramidal cells in stratum 6, also of moderate size, directly connect to the thalamus, thereby producing feedback. The rest of the cells seen in Figure 2.56, referred to as nonpyramidal cells, are responsible for the processing of information. The neuroglialform cell is thought to be the primary element associated with

cognition. The other cell types are thought to perform distributive functions by transferring information across different regions. Type B and type D cells demonstrate a vertical transfer of information. The top stratum of the cerebral cortex contains no neurons. Rather the axons run in this strata for short distances, performing moderate range intracortical communications.

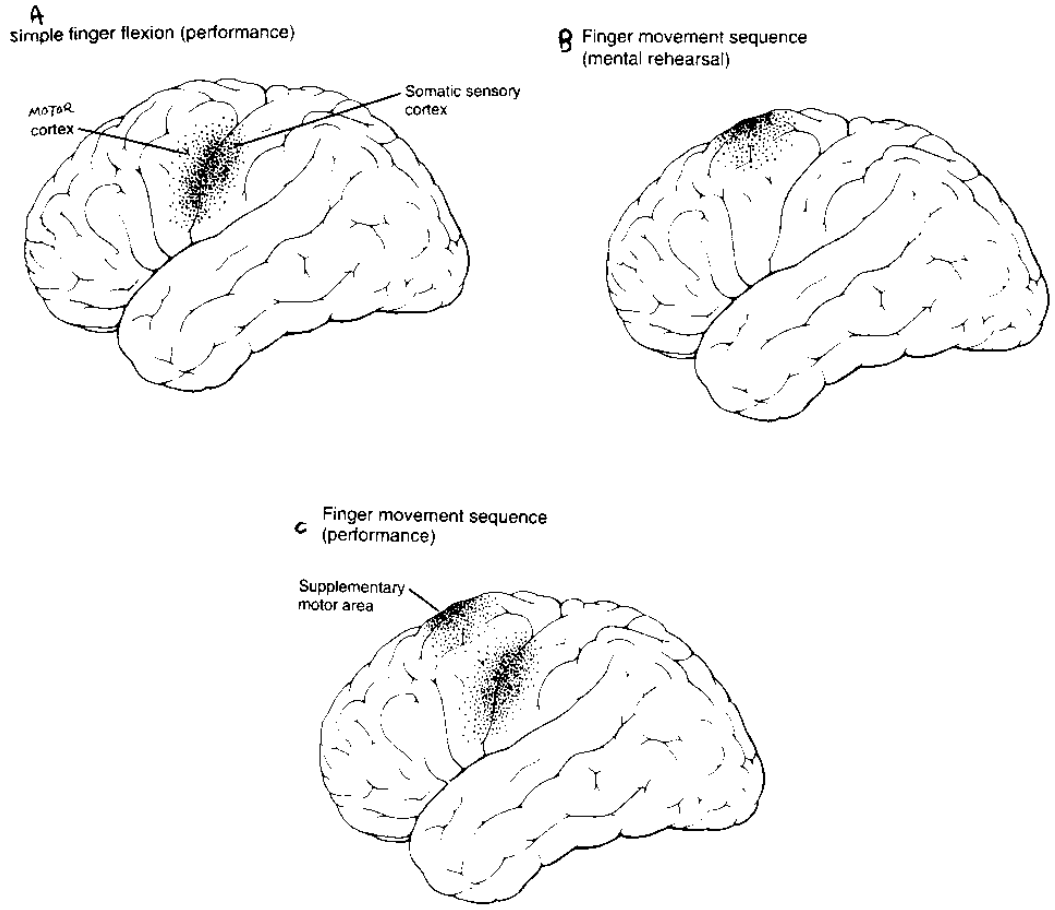


Figure 2.55 Cerebral cortices. The role of the different cortices in various functions. (a) During simple movements, only the portions of the sensory and motor cortices are active. (b) During a mental rehearsal of the activity, a completely separate cortex is active. (c) When a complex movement is performed, all cortices from (a) and (b) are simultaneously active [27].

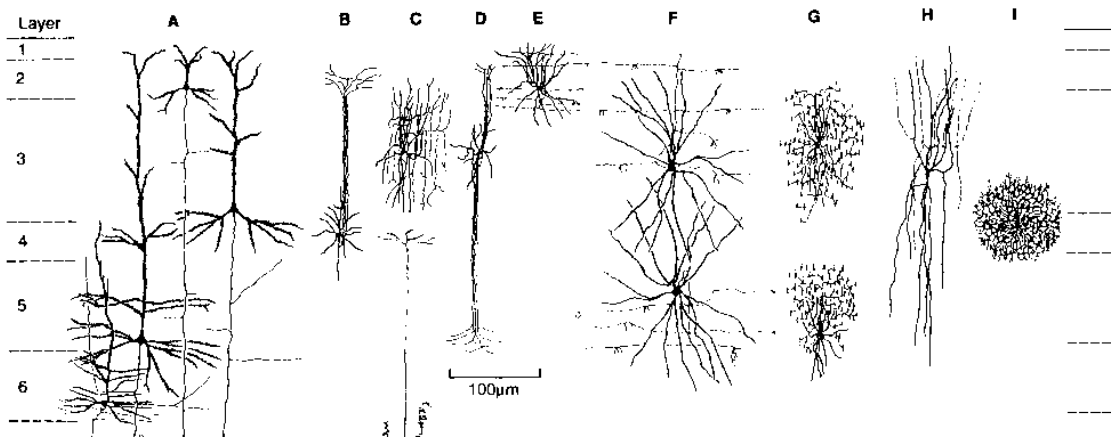


Figure 2.56 Illustration of the variety of cell types in the cerebral cortex as seen by the diversity of the cytoarchitecture [27].

2.3.3.2 Neural Pathways of Indirect Interconnections

The various cortices of the brain are connected in a well ordered manner through the limbic system. The limbic system consists of numerous neural features, all of which can be classified as either data conduits or junction points. There are three principle data conduits, the *basal ganglia,* the *corpus callosum,* and the *cingulum*. The basal ganglia is the collection of axons which forms a series of four loops in the frontal lobe: (1) sensory-motor, (2) oculomotor, (3) association, and (4) limbic. The first two are associated with motion control of the body, and the latter two play roles in memory and cognition. The corpus callosum is the conduit which connects the two hemispheres of the brain. The corpus callosum is shaped like a hyperboloid with a 'C'-shaped neck. The corpus callosum originates in the frontal lobe, forms a graceful spiral toward the occipital lobe, and terminates in the *hippocampal* formation located at the bottom-center of the brain. The cingulum is much like the river of the brain, receiving tributaries from all cortices, where each cortex contributes representations of primitive objects, and the total sensory perception is assembled as the multitude of signals approach the hippocampus. The assembled representations can then be stored by the hippocampus.

In summary, the cerebral cortex can be represented as a uniform, six strata film wrapped over the surface of the brain. The brain is arranged in a very symmetric and organized fashion which will serve as a guideline for interconnecting artificial cortices.

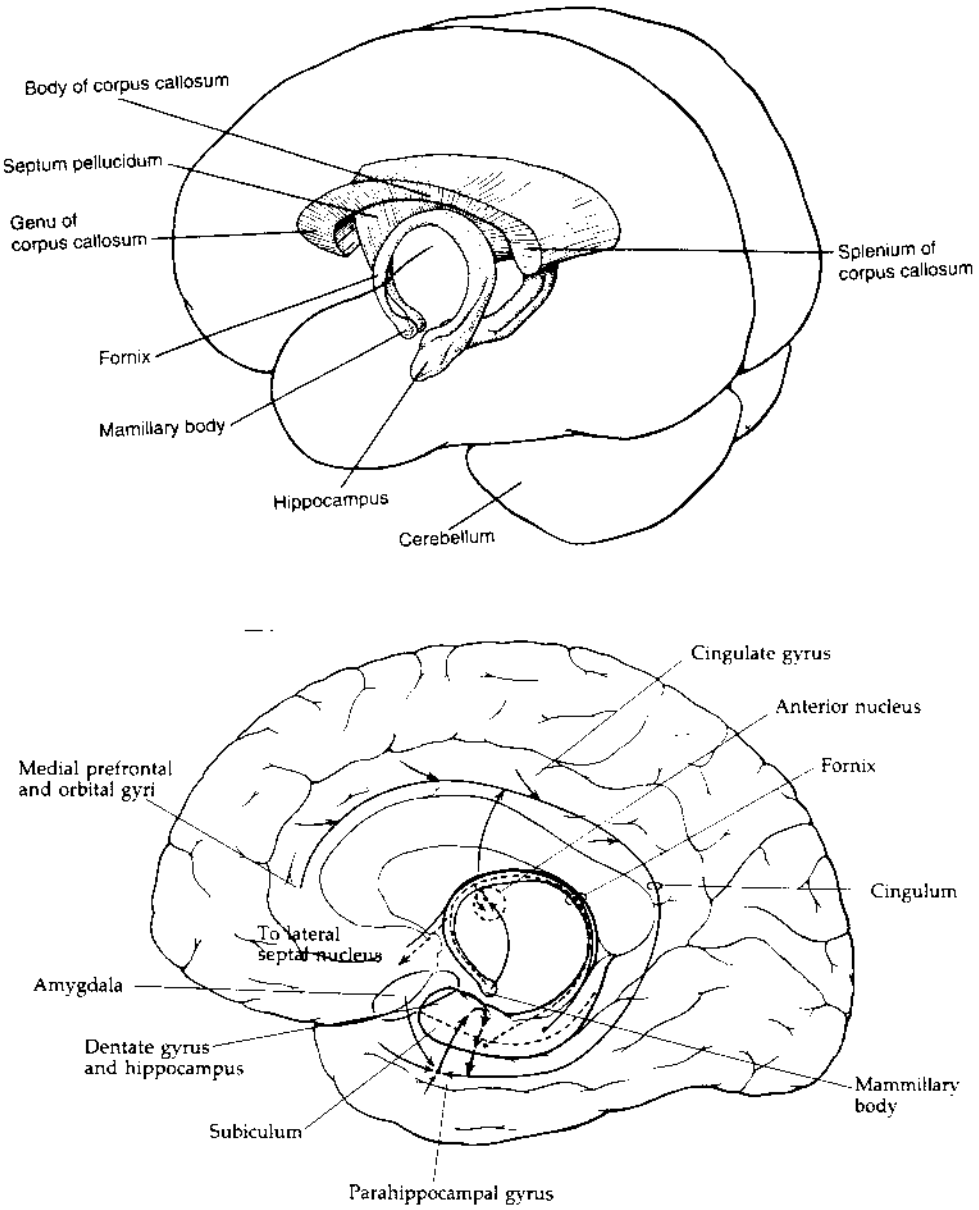


Figure 2.57 Neural pathways in the brain. The corpus callosum, **top**, connects the two hemispheres [27]. The cingulum, **bottom**, interconnects cortices in the same hemisphere [116].

2.3.4 Mapping

The last topic in neural science to be discussed is mapping. Mapping describes how neurons from one region project onto another region. It is necessary to discuss the phenomenon of mapping as it describes how the artificial cortices are connected to their sensory and output devices.

For each sensory input on the body's surface there is a corresponding neuron on the brain's surface. As the brain develops, nerve cells extend to the various parts of the body in a surprisingly systematic way. Two theories have been proposed to describe this fact. The first claims that the embryonic cells were initially adjacent, then as the embryo grew, the neurons essentially stretched to span the distance. The second theory suggests that a specific cell releases a chemical agent that stimulates a particular neuron to extend and eventually make contact with that cell. Both theories are probably correct to some extent.

The most extensively studied sensory system is the mammalian visual cortex. The vision system has five principle elements: the retina, the optic nerve, the subcortical regions, the visual cortex, and the secondary visual cortex, which perform: image collection, image transfer, eye control/coordination, image detection, and image association, respectively. The retina has five strata. The first collects the image, the middle three enhance acuity and edge perception, and the fifth projects the image, via the optic nerve, to three subcortical regions: the *lateral geniculate*, the *pretectal array*, and the *superior colliculus*. The lateral geniculate is a relay point to the primary visual cortex, the pretectal array controls pupil size, and the superior colliculus coordinates eye movements.

The superior colliculus contains seven maps, including three of the retina. If one could imagine applying a unit grid to the retina, he would find that the corresponding grid on the superior colliculus has been distorted such that the elements that contain the fovea (focal point of retina) are much greater in size, while those of the peripheral vision are smaller (Figure 2.58). In actuality there is no such grid; however, each photo receptor group in the retina is connected to the same number of neurons in the superior colliculus. Essentially each photo receptor has equal resolution regardless of its location on the retina. Higher perceived resolution is achieved by concentrating more photo-neurons at the fovea.

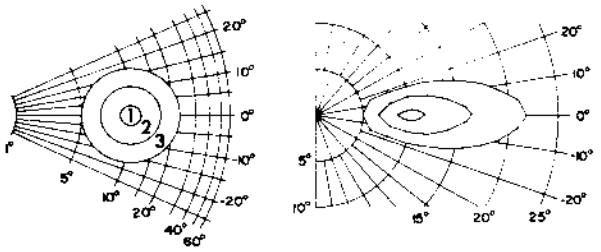


Figure 2.58 Distortion between the retina, **left**, and the superior coliculus, **right**. Note that circles are used to illustrated the distortion [117].

Mapping does not only occur in the visual system, but it occurs in all other sensory and motor cortices (Figure 2.59). Figure 2.60 shows the sensory input for the various parts of the body as mapped onto the surface of the brain. The distortion that McIlwain [117] described in the superior colliculus is given visual representation here. The distortion of sensory maps in different animals is shown in Figure 2.61.

In summary, sensory and motor neurons map to their respective cortices in a logical and organized manner. While the proximity of the neurons is maintained on the map, the overall proportions of the map are distorted to emphasize regions of greater sensitivity or control.

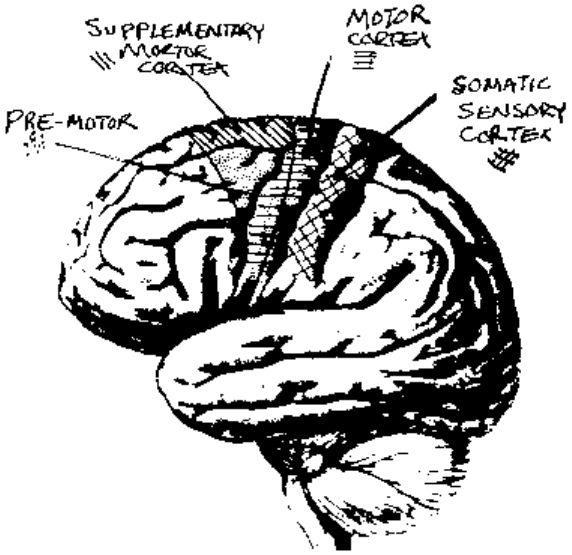


Figure 2.59 Side view of brain showing motor and sensory cortices [116].

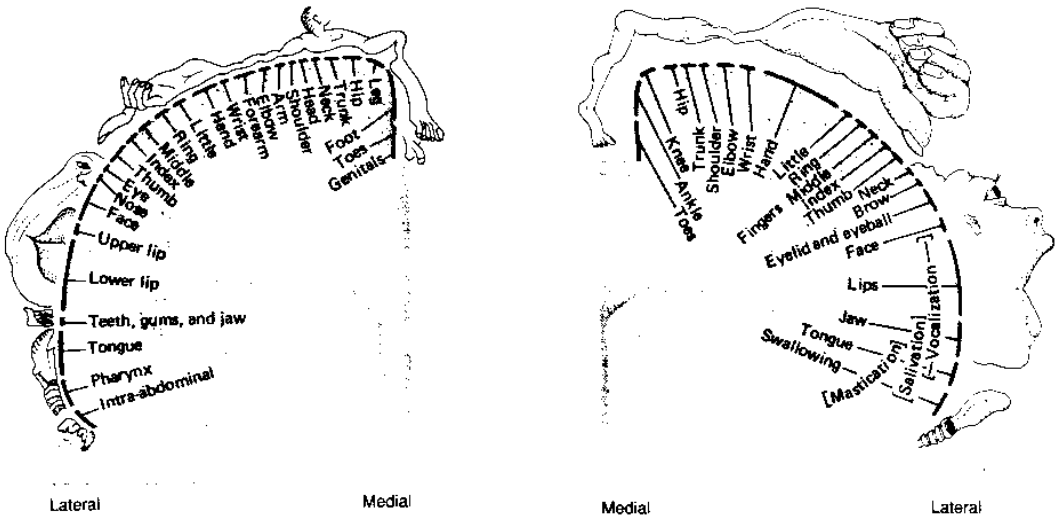


Figure 2.60 Mapping of motor and sensory cortex. Note the relative sizes of the body features are proportional to their importance in the respective cortex [27].

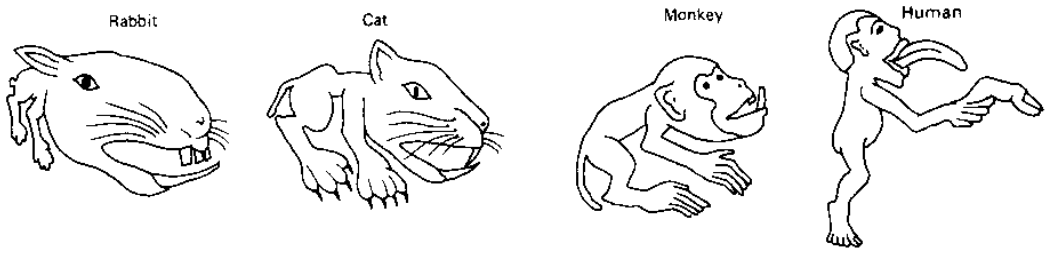


Figure 2.61 Relative importance figures of the sensory cortices of various animals [27].

2.3.5 Summary

The neuron has been presented to allow the development of a set of parameters needed for the mathematical representation of the nervous system. The identification of the six strata structure of the cerebral cortex is crucial for the development of the technique described in this dissertation. The organization of the brain provides a useful guide which allows both the separation of the functional regions and provides a method for connecting the cortices. The last section of neural science, mapping, provides a tool to allow the artificial cortex to be connected to its input and output devices. The next section changes the focus from the cells of the brain to the molecular level of control of the cell.

2.4 Genetics and Heredity

Genetics and heredity play a crucial role in the growth simulation program. The ability to encode structural parameters into an artificial chromosome allows the optimization of the system. Genetics provides four extremely useful contributions to the dissertation. The first describes the nature of DNA, the information contained therein, and how that information relates to the global structure. Second, genetics describes how traits are passed across generations in a semi-random fashion that balances diversity and specificity, while optimizing the organism for its environment. The third topic in this section is that of development. The sequence of development has been used as the outline for the cell growth simulation program. The last topic describes the role of concentrations and chemical messages in genetic determination. This last topic has been shown to be one of the most powerful tools to control the structural morphology of massive scale artificial neural networks.

In biological systems, deoxyribonucleic acid, DNA, represents the blueprint of all organisms. DNA provides the template for the formation of proteins which constitute all living tissue. DNA plays a crucial role in the reproductive cycle, serving both as the carrier of the genetic record, and the mechanism for evolutionary change..

2.4.1 The Structure of DNA

DNA is a polymer chain of four nucleotides, adenine, thymine, cytosine, and guanine, which form a double helix structure. Except for times of reproduction when the DNA assumes neat and orderly bundles called *chromosomes*, the DNA appears as a knotted mass, called *chromatin*, in the nucleus. The chromatin is held together with proteins, such that most genes are buried in the center, and only a small fraction is exposed. The buried genes are called euchromatin, and the exposed genes are called heterochromatin [118]. Most complex organisms have several strands of DNA, or chromosomes, which are collectively called the *genome.* A chromosome, or chromatin strand is divided into subunits called genes (Figure 2.62). Each gene represents a unique compound or trait. The genes are made up of codons or triplets, which are sets of three nucleotides. Each codon represents a specific amino acid, the basic building blocks of proteins. When a gene is converted into a protein, or some other compound, the gene is said to have been *expressed*. Although every cell contains the complete genome, each cell selectively uses only a small fraction of the genome throughout its life.

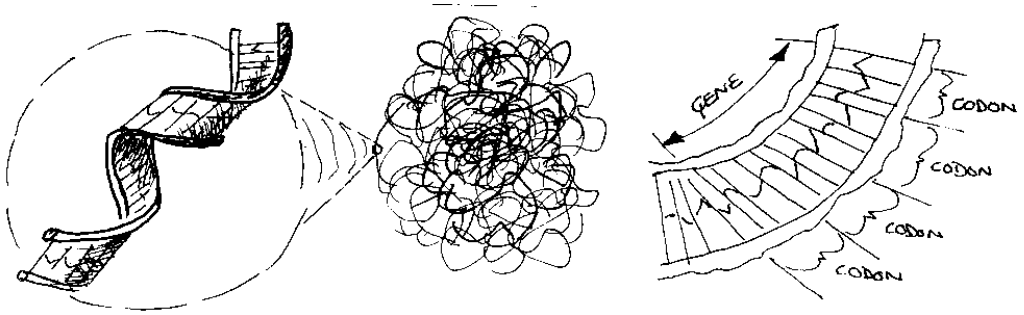


Figure 2.62 Schematic representation of DNA structure. (a) Double helix, (b) DNA as it appears in the cell nucleus, and (c) location of genes and codons.

2.4.2 DNA to Protein Conversion

All biological compounds are made from DNA, either by direct translation which yields polypeptide chains (proteins), or indirectly, where an enzyme (type of protein) is produced to create all other compounds. The conversion of DNA to a polypeptide chain is a three step process. The first involves the separation of the double helix to form two *DNA* strand, one *coding* strand and one *non-coding* strand (Figure 2.63a). Since the DNA never leaves the nucleus, a working copy, called messenger ribonucleic acid, (mRNA), is *transcribed* from the DNA coding strand in the second step. In the last step the completed mRNA leaves the nucleus and enters the cytoplasm where it is *translated* by the *ribosome* to produce a polypeptide chain (Figure 2.63).

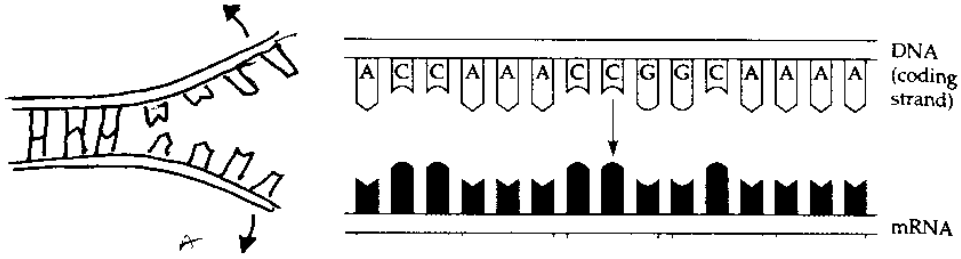


Figure 2.63 Schematic representation of mRNA formation: **left** separation of the double helix, and **right** transcription and translation of mRNA [27].

Ribosomes are organelles that essentially walk along a strand of mRNA and attach

segments of translational-RNA, (tRNA), that correspond to the codon that the ribosomes cover. Each new tRNA segment has an amino acid attached to it, and is called an aminoacyl-tRNA. As the ribosome continues to walk along the mRNA, the tRNA segments are released and the polypeptide chain is formed (Figure 2.64).

2.4.3 DNA Regulation of Cellular Activity

Most people who are unfamiliar with cellular biology attribute an intelligence to the chromosomes that simply does not exist. It was previously believed that the DNA somehow *decided* when different cell activities should occur. In contrast, it is now believed that the DNA is simply the blueprint for the proteins that the cell needs to function. The driving force behind the selection of which protein to produce at a given time is a complex interaction between the concentrations of all the different substances within the cell and its immediate environment. When the concentration of a particular substance becomes either too great or too low, a conformational change will take place in a protein that binds the DNA, exposing a different set of heterochromatin genes (Figure 2.65). Since the euchromatin genes are buried, they cannot be expressed; thus by changing the set of genes that appear on the surface, the conformational change selects a different set of proteins to be produced. The conformational change can simultaneously stop the production of the other protein, or it can cause another series of reactions that will force the cell's chemical balance into equilibrium. The process is both amazingly complex and inefficient. Roughly 90% of all cellular material that is produced is subsequently destroyed during cell regulation.

2.4.4 DNA in Reproduction

Most cells in the body have paired chromosomes, meaning that there is a redundant set of genes that are never expressed during the organism's life. Both chromosomes of the pair have dominant and recessive genes. The reproductive cycle of multicelluar organisms begins with the formation of the sex cells or *gametes*, in which a regular cell divides and gives one chromosome from each pair to each gamete (Figure 2.66). The selection of which chromosome will go to which gamete is random. A human with 23 chromosome pairs can form at least 2^{23} (8,388,608) different gametes. When the gametes from two individuals join, there are over 2^{46} (70 trillion) possible different offspring.

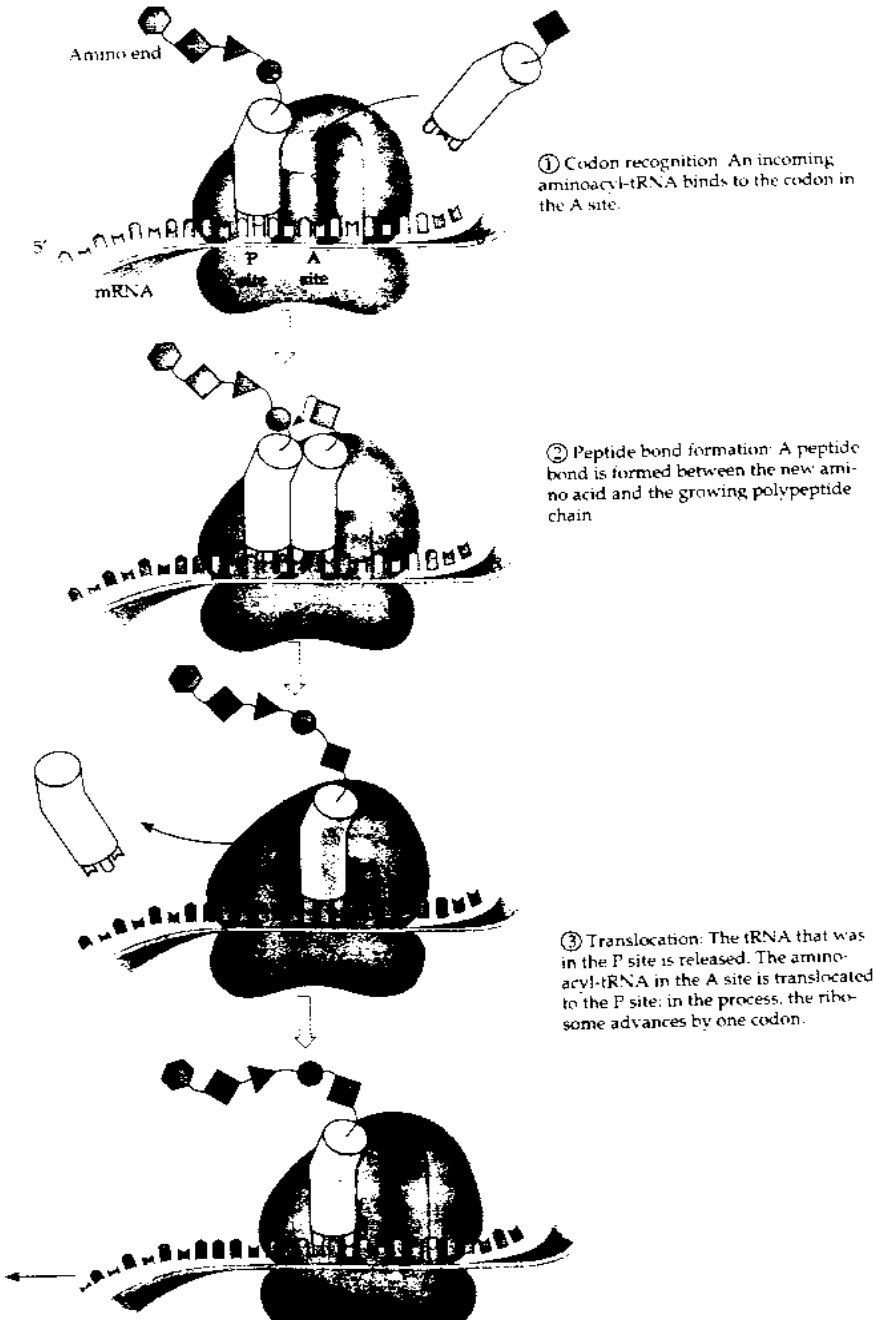


Figure 2.64 Production of a peptide chain by ribosome translation of mRNA. (a) Codon recognition: an incoming aminoacyl-tRNA to the codon in the A site, (b) peptide bond formation: a peptide bond is formed between the new amino acid and the growing polypeptide chain, (c) release of the tRNA that was in the P site, and (d) translocation: the aminoacyl-tRNA moves from the A site to the P site [113].

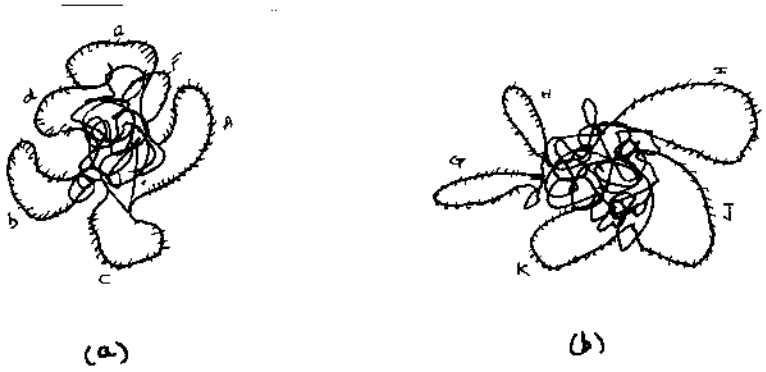


Figure 2.65 Schematic representation of the effects of a protein binder conformational change on the morphology of chromatin. (a) DNA before conformational change, and (b) DNA after. Note that new genes appear on the surface.

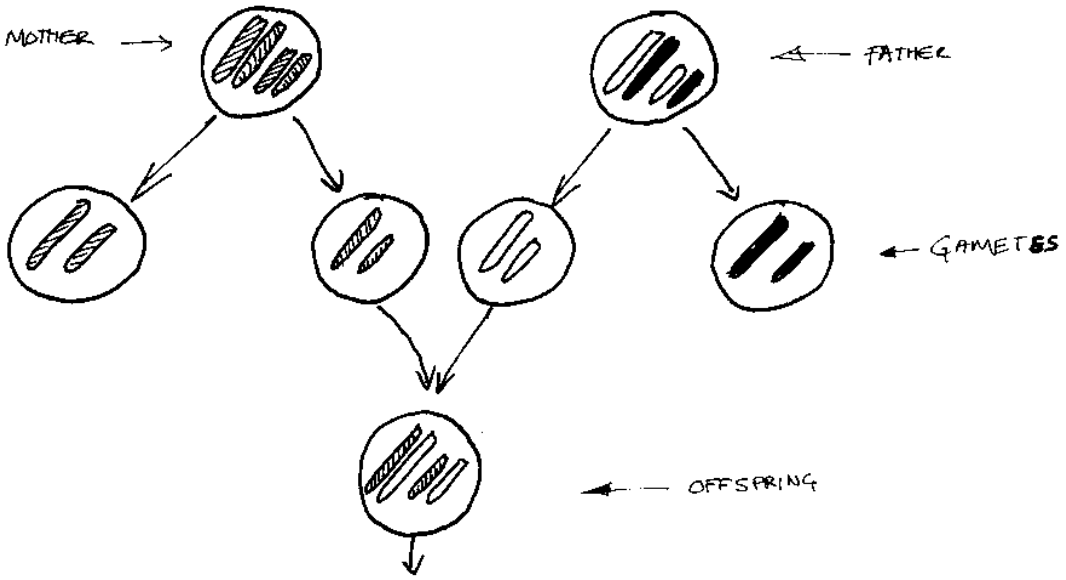


Figure 2.66 Multicellular reproduction from the DNA level .

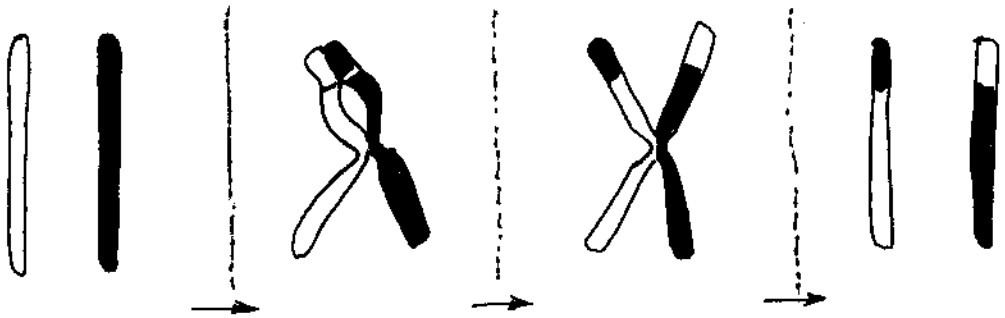


Figure 2.67 Crossing over of Chromosomes.

Actually there are considerably more possible offspring than 2^{46} due to a mechanism called crossing over. Crossing over allows genes from one chromosome to be exchanged with its paired partner. Consider the chromosome pair, X1 and X2, that have 1000 genes each. If they undergo a single cross over, then the resultant chromosomes could be: X1' = {(1 to 237) $_{X1}$ + (238 to1000) $_{X2}$} and X2' = {(1 to 237) $_{X2}$ + (238 to1000) $_{X1}$}, where the subscripts denote from which original chromosome the genes came from. Since all chromosomes can undergo numerous cross-over operations, the number of possible unique offspring far exceeds 70 trillion.

2.4.5 Heredity

When gametes join, the resultant cell, called the zygote, has a new genome comprised of genes from both parents. In essence, each parent provides one gene for each trait. For any given organism, only one of the two available genes will be expressed. The gene which is expressed is the most *dominant* and the other is *recessive*. Because only half of the genes are expressed, the appearance, or *phenotype*, of the organism will not be representative of all of its genes, or *genotype*. Zygotes that have all recessive or all dominant genes are called homozygous, while those containing a distribution of both dominant and recessive genes are called heterozygous. The laws of heredity, as defined by Mendel, show that the distribution of homozygous and heterozygous offspring follow predictable patterns (Figure 2.68).

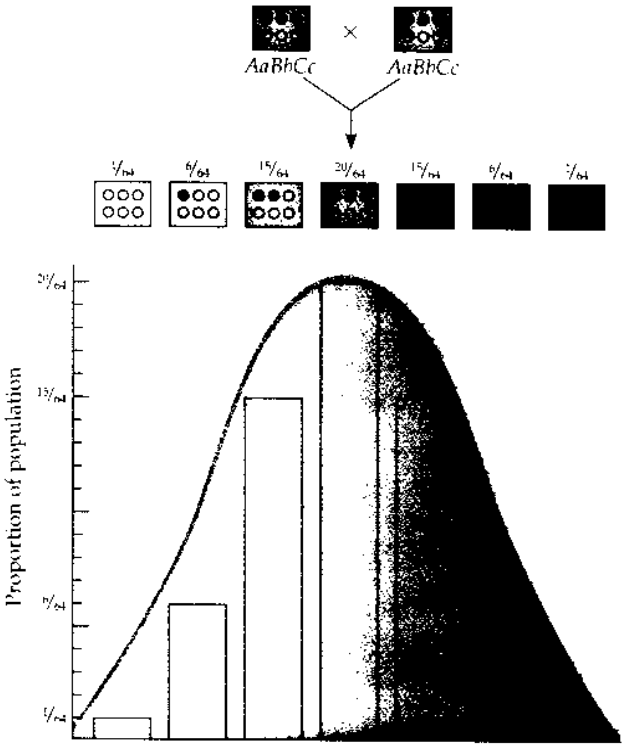


Figure 2.68 Proportion of offspring population vs. genotype [113].

From the above discussion, it is implied that for a given gene there are only two possible states, dominant and recessive. This is not the case. A given gene may have numerous slightly different forms called *alleles*. Alleles basically code for the same trait but produce slightly different characteristics. An allele is said to be dominant if it completely suppresses the other allele for that particular trait (Figure 2.69). Partial dominance, on the other hand, refers to the condition when both genes are partially expressed (Figure 2.70). The appearance of an organism is further varied by *pleiotropy* and *epistasis*. Pleiotropy is the condition where one gene is responsible for the expression of more than one trait, and epistasis describes the condition when the expression of one gene affects the expression of another (Figure 2.71). Independent assortment (Figure 2.72) is similar to pleiotropy, in that more one gene defines a trait; however, in the case of the former, there is no chemical or physical interaction.

2.4.6 Development

After the zygote has formed and the genotype has been determined based on the dominance rules described above, the single cell will begin to grow and multiply. After a few days the zygote has formed a mass of nearly identical cells called a blastula. As the cells begin to differentiate, an internal void is formed, marking the entrance of the gastrula stage. The developing organism is called an embryo when all of the basic cell types have differentiated and primitive organs and features are recognizable. The fetal stage is marked by well developed organs and features and well defined cell types.

Since every cell in the fetus originated from the zygote and contains the same genome, it is presumed during development a mechanism activates selective genes for each cell. The most plausible hypothesis is based on knot theory, suggesting each cell type has a unique knot [119]. Knot theory describes how continuous strands loop upon themselves. The knot hypothesis is consistent with the concept of heterochromatin and euchromatin.

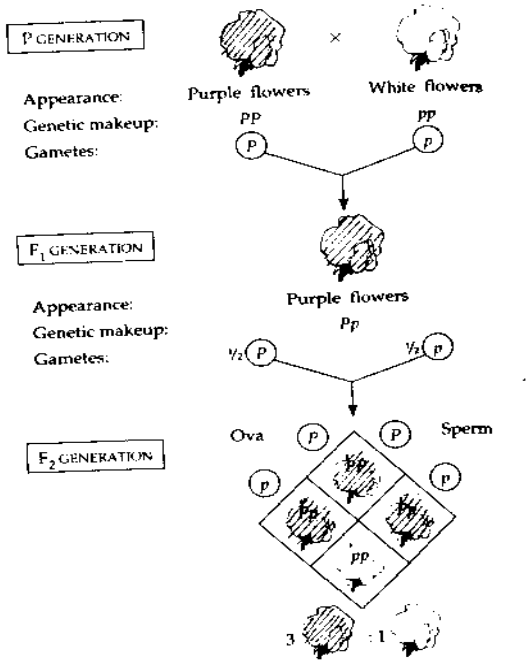


Figure 2.69 Complete dominance [113].

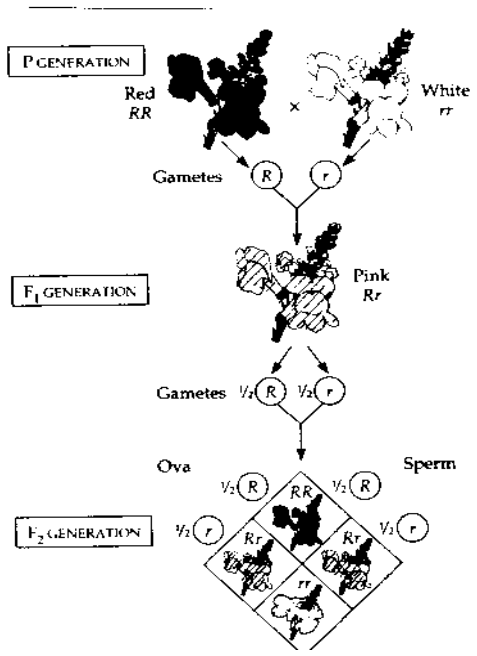


Figure 2.70 Incomplete dominance [113].

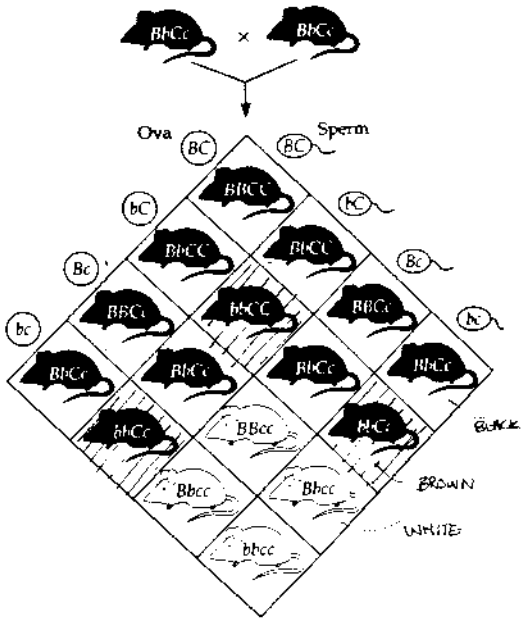


Figure 2.71 Epistasis [113].

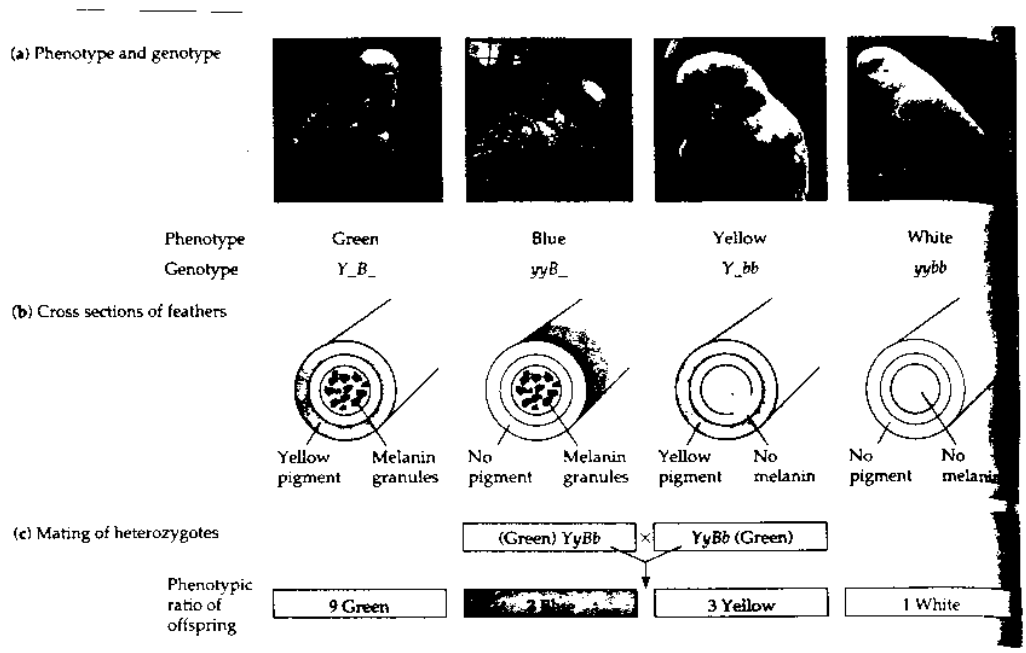


Figure 2.72 Independent assortment [113].

2.4.7 Concentrations and Growth

The role of chemical gradients in the growth of cells during development cannot be underplayed. There are two basic driving effects that cause growth in cells, the absence of a certain chemical signal or the presence of one in excessive concentrations. It is assumed, but not yet proven, that during development the gradient field serves as a guide for the cell's path of growth. It is further assumed that some cells grow towards higher concentrations of chemical signals, while others grow towards lower concentrations.

Chemical signals are detected by features on the cell's surface called receptors. A ligand is a general term used to describe any extracellular substance that binds with a receptor. When a ligand mates with a receptor, a chain reaction occurs which initiates the cell to perform a certain task. In the case of fetal growth, the detection of a ligand could initiate a conformational change in the chromatin, thereby initiating the production of a certain protein that could only be incorporated into the cytoskeleton in a certain location, yielding physical growth in a certain direction.

One of the more interesting aspects of concentration gradients is the role of cells in reducing local concentrations [120]. It has been found that some cells have special receptors

that function to remove the growth ligand out of the intercellular fluid. Consider three cells, **X**, **Y**, and **Z**, where **X** secretes growth ligand **A** and **Z** has the receptor for that ligand. If **Y** is located between cells **X** and **Z**, and it is equipped with the special receptor for ligand **A**, **Y** can effectively remove the vast majority of ligand **A** out of the solution before it can reach cell **Z,** thus the effect that ligand **A** would have had on **Z** can effectively be suppressed.

2.4.8 Evolution and Mutation

Evolution as described by Darwin [121] is based on the survival of the fittest individuals of a species. The variation between individuals within a species is based on differential gene expression, as defined by dominance rules. Darwinian survival biases future generations towards having those genes that were either beneficial or non-impeding on the organism's ability to compete in its environment. The biasing of the gene pool away from harmful alleles does not prevent the carry-over of benign, harmful, or recessive traits. The future survival of a species relies on its ability to produce genotypes that are competitive in the new environments from the differential expression of various traits. Since there is no way to predict which allele will be necessary for the survival of future generations, all but fatal alleles must be kept in the gene pool.

Mutations does occur sometimes, but most often the results are cancer or death. Mutation usually occurs before birth; The result is frequently miscarriage so that the mutation will not be passed on to any offspring. The development of a new, superior individual from mutation in biological systems is a very rare occurrence. Note that this is in contrasts to traditional genetic algorithms which rely on mutation as a mechanism for evolution [6].

2.4.9 Summary

This section has established that the DNA is both a blueprint for proteins and a regulation for production of those proteins. The distinction and separation of this information is the foundation of the regulating process of the cell growth simulation program. The dual functionality of the chromosome is an extremely useful tool and is a significant contribution to the area of genetic modeling. It is believed that this is the first time that DNA has been modeled with dual functionality. The development process follows a steady and predictable sequence of events and serves as the guideline for the cell growth simulation program. By maintaining a high level of detail in evaluating the role of chemical messengers, a useful tool for regulating network morphology has been identified.

2.5 Cognitive Science

The previous sections have focused on physical, chemical, and anatomical aspects of the nervous system. Although these issues are important and necessary, alone they do not describe how the collective interactions of the neurons produce the functionality commonly associated with intelligence. Cognitive science is the discipline that bridges the philosophy of thought and the science of the brain in an attempt to describe how intelligence can reside in the nervous system. There are three main areas of cognitive science that have been utilized in this dissertation. The first area describes how a thought or idea is represented in the brain. This topic is addressed because classical neural networks use the older, "Grandmother" representation while the massive scale neural networks developed in this dissertation use the more advanced distributed model. The second topic, self-organization, describes how the interaction of different cell types allows the formation of cortical columns and how these columns compete with each other for cortical area. The cortical column model is fairly recent and is the foundation for learning and training massive scale systems. The last topic is that of automata, the most accurate model of multicellular systems. The automata is the core of the cell growth simulation program.

Artificial intelligence was originally mankind's attempt to create a near duplicate of his own cognitive self. After the 1960's, the term *artificial intelligence (AI)* came to specifically describe the development of thinking machines and computer programs. The original multi-disciplined study of human intelligence has come to be referred to as *cognitive science (CS)*. The term cognitive science was coined by the Sloan Foundation when they awarded several multimillion dollar grants to MIT, Berkeley, and San Diego to pursue the multidiscipline approach [122]. Cognitive science differs from artificial intelligence in that its goal is to accurately describe the actual processes that enable human thought. The models discussed in cognitive science are much more closely linked to the actual physiology of the nervous system. Unlike the easily implemented AI models, the CS models tend to be more complex and are intended to help theoreticians gain a better understanding of the cognitive process. Cognitive scientists are primarily interested with the logic associated with the representations of thoughts, sensations, and memories. The models used in CS to describe these processes are too abstract for direct implementation; however, they do identify several issues that are important from an artificial intelligence point of view.

2.5.1 Representation

2.5.1.1 The Grandmother Neuron

The issue of how an object is represented in the brain is one of the most fundamental questions confronting neural scientists today. Prior to the late 1960's, the most prevalent theory was referred to as the *grandmother* neuron. It was thought that for a given object or idea there exists a single neuron somewhere within the brain that represents the item in question. For example, when an individual would see his grandmother, a single neuron would fire that had been associated with the sensory inputs stimulated by the presence of his grandmother [123]. The grandmother cell theory was an outgrowth of observations made in simple invertebrates. In invertebrates, *command* neurons have been identified that are responsible for a particular response. The most famous example is the giant axon of the squid which is solely responsible for the coordination of the squid's escape response. Based on the observations made in invertebrates, it was conjectured that some command or grandmother cells existed in higher mammals.

To verify the theory, single micron diameter probes were developed to locate the location of the grandmother neurons. These probes enabled researchers to place probes within the brain and take actual measurements of single neuron activity. As the body of data accumulated from this technique, it became increasingly clear that the theory was in error. No matter where the probes were placed, the same thing was observed, one stimulus activated a clusters of neurons.

2.5.1.2 Distributed Memories

The neural clusters suggest that the brain represented objects and ideas through the combination of numerous neurons, such that no single neuron represented any one thing in particular [123]. This new understanding of neural representation initiated Anderson [124] in 1968, and many others to discuss the implications of distributed intelligence. Anderson sites four characteristics of neuro-processing. "First, information is represented as patterns, not by single cells. Second, use of patterns mean that interference occurs between stored items, and so information loss must be accepted. Third, memory and cognitive operations are generally statistical. And fourth, biological computation is quite different from engineering computation, and the correct questions must be asked [124]."

After the emergence of the distributed theory of memory, neural scientists became

aware that brains behaved vastly differently than digital computers. The digital computer has exact memories, where each memory is stored in a unique location. If a digital computer loses one bit, the entire meaning of that memory is corrupted. In contrast, memories are stored across innumerable neurons such that 100 or more memories could share thousands of the same neurons. This distributed type of storage presents the problem to the modeler of determining the degree of integrity of a given memory. While distributed memory is far less precise than its digital counterpart, it is far more resistant to information loss. This resistance of neural memories to damage is demonstrated by patients that suffer partial damage to the temporal lobes, the brain center associated with memory storage. Theses patients do not lose a specific or group of memories, rather all memories are partially degraded [125].

The second aspect of neural memory systems of engineering interest is the high rate of memory recollection. Anderson [124] observed that when subjects were given a list of items to memorize, the time for the subject to determine if the item was on the list was the same regardless of whether the item was actually on the list or not. This observation indicated that knowledge was, in fact, being stored in a parallel, distributed fashion. Had the memories been stored in a serial fashion, then it would have taken much longer to identify an item that was not contained on the list, as all other memories would have to be searched. Although the recollection rates are good, the accuracy is not. Subjects could not accurately determine if a nearly similar pattern was or was not the one contained in the list. The incredibly high speed at which the human mind can recall memories is of unquestioned value, and while the actual quality of the memory may still be up for discussion, the issue of speed overrides the quality concern.

2.5.1.3 Representing Distributed Storage

Until the mid 1970's neural scientists and engineers were in agreement that digital computers were capable of reproducing the behaviors of the nervous system; but as the distributed theory became more prevalent, neural scientists began seeking more abstract systems with which to draw parallels. Holograms, now commonly seen on credit cards and drivers licenses, possess the same resilience to damage as the memory systems of living organisms. When a portion of the plastic that contains the hologram is scratched the image is still seen in the area of the local damage, but the overall image quality of the hologram is degraded. Gabor [126] proposed that holograms could be used as memory by storing the wavefront of the object rather than the actual image. Although there are many technical problems with the implementation of a hologram based memory system, the model has

proven useful for developing some mathematical relationship governing distributed memory storage [127].

2.5.2 Self Organization

Once an organism is born, its nervous system must organize the multitude of sensory inputs so it can understand and function in its environment. Considering the size of biological nervous systems and the amount of sensory input, this correlation process is a nontrivial problem. It is assumed that real biological network do not have to self organize from zero [128], but rather are born with some pre-programmed pathways. These pre-programmed pathways are commonly referred to as instinctive behaviors. The fact that virtually all organisms of a given species possess the same instinctive behaviors is strong evidence that these pathways are genetically coded. While no one argues the genetic nature of the instinctive behavior, the mechanism by which the pathways are produced from genetic information is unknown.

Since it is believed that the only learning mechanism that biological organisms have for training is Hebbian adaptation, it is presumed that all post-natal learning is based on associations with the fundamental pathways of instinctive behavior. The limit to which an organism can adapt to the environment is the amount of redundant, untrained pathways with which it is born that can later be correlated [128]. Thus, compact and efficient networks will not be able to adapt to new environments or situations because the redundant pathways needed to support the new behaviors would not exist.

Self organization was briefly discussed in section 2.3.2 where it was mentioned that the combination of short range Hebbian and long range anti-Hebbian mechanisms induced group formation through the Mexican hat function [129]. Although the self organization of artificial networks is interesting, from a modeling point of view, the actual biological process is more pertinent.

Merzenich et al. [130] outlined five empirical rules of somatosensory map organization. They are reprinted here for convenience:

1. *The Mo*vement Rule: Sites of representation of particular localized skin surfaces can move hundreds of microns, and can differ several fold in areal extent. Representational borders remain sharp throughout this process of movement.
2. *The Continuity Rule*: As the representations of uncut nerves expand into silenced cortical areas, they maintain topographical continuity.
3. *The Distance Limit Rule*: The distance limit for reorganization (as shown by amputation of digits) is roughly 600-μm on either side of the cortical map border originally presented between the representation of the deprived and normal skin.
4. *The Overlap Rule*: The percentage overlap of the receptive field of two cortical neurons

decreases as the separation between them in the cortex increases. The receptive fields of neurons located more than about 600 μm apart do not overlap at all

5. *The Inverse Rule:* As the representations expand in the cortex, their receptive field sizes decrease in area.

These rules were later simplified by Edelman and Finkel [33] when they studied the mechanics of neuronal group selection of the cerebral cortex. The neuronal groups of the cerebral cortex have been previously identified by Mountcastel [131] to exist in vertical *cortical coulums* connected by bouquet and chandelier cells (section 2.3.3). Edelman and Finkel first described the cerebral cortex as a "degenerate anatomical substrate capable of giving rise to numerous possible maps." The process by which a particular map is selected is a concurrent struggle between neuronal groups for domination of cell activity. This concurrent struggle could be described with the three basic mechanisms that unified the five rules presented by Merzenich et al. The concurrent group selection mechanisms, *group confinement*, *group selection*, and *group competition*, are described with regard to the anatomical seven strata structure of the cerebral cortex.

1. *Group Confinement*: Cortical domains are restricted by local horizontal inhibitory connections.
2. *Group Selection*: Translational movement of the extensively overlapping adjacent neurogliaform groups toward coactivated sub-clusters and away from uncorrelated peripheral stimuli.
3. *Group Competition*: A set of six guidelines that predict which group will dominate in a Darwinian competition for cortical space; a) group sizes above and below min/max values are unstable, b) partial overlap with adjacent groups is desirable, c) distal group members can be easily captured by other groups, d) older groups dominate over younger groups, e) density of receptor field, and f) most frequently simulated groups dominate.

The previously mentioned group formation mechanisms have since been integrated into a computer simulation to demonstrate the formation of neuronal groups. The results are given graphically in Figure 2.73.

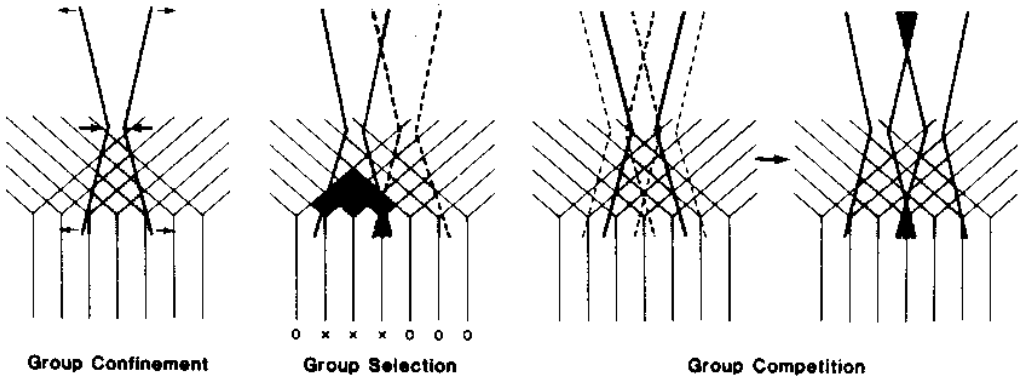


Figure 2.73 Group formation by the Edelman-Finkel model [33]. See text for details.

2.5.3 Automata and Artificial Life

The study of automata is a subset of computer science dedicated to developing programmable parallel hardware. The term *artificial life* is derived from the biological models employed in the development of the "cellular" topologies that automata theory is based upon. The concept of developing programmable parallel hardware originated when the shortcomings of serial hardware were realized. The originator of artificial life was mathematician John von Neumann, who not surprisingly also developed the CPU-memory architecture utilized by every digital computer since the second world war. To understand "artificial life" it is necessary to review von Neumann's contributions in the field of digital computing.

2.5.3.1 Historical Perspective

The first digital computer, the ENIAC, did not have a CPU, and programs were done in hardware. The ENIAC was basically a mass of unconnected digital logic elements, (20 accumulators, 1 multiplier, 1 divider/square rooter, 3 function tables, 1 input unit, 1 output unit, 1 master programmer, and 2 supplemental controllers). A program was produced by changing the states of mechanical switches, and physically wiring different combinations of these elements [132]. Since these elements were essentially separate, the ENIAC performed many operations in parallel. von Neumann proposed a that soft-programmable computer could be produced by having one CPU that contained all possible logic operations and a program stored in memory that instructed the CPU to behave a particular way at a particular

time. In operation, a von Neumann computer fetches an instruction from memory, performs that function on the data item(s), then fetches the next program instruction, and repeats the process until the program is completed.

2.5.3.2 Overcoming the Fetch Cycle

While the von Neumann architecture is extremely powerful, it is limited by the need to fetch each instruction prior to every operation. Crevier [122] cites that computer engineers have tried to work around this problem somewhat by (1) increasing the number of bits in the data bus, which decreases the fetch time, and (2) through miniaturization, which increases the feasible clock frequency. Unfortunately, neither of these approaches overcome the problem; they merely make it less visible.

One way to overcome the fetch cycle is to return to the ENIAC's parallel architecture with "soft" connection replacing the "hard" wires. The concept of a "soft" programmable ENIAC was also proposed by von Neumann. He envisioned a chip with elements that could reconfigure themselves into various digital logic elements based on an instruction set contained in "soft" code. The proposed architecture would load the entire program to all of the cells in one step, then execute the program, also in a single step. Once the program was completed, the next program would be loaded and run in a similar, parallel fashion.

The "soft" ENIAC theoretically has very high processing rates; however, the actual development of the system is hindered by the enormous complexity associated with coordinating and performing numerous operations. von Neumann [132] knew that living organisms routinely perform billions of parallel functions every second, and soon began modeling the "soft" ENIAC based on biological mechanisms.

2.5.3.3 The Mechanical Cell

Taking a decidedly mathematical approach, von Neumann assumed cells were biochemical machines. Since machines obey predictable rules, von Neumann proposed that cells could be defined as a series of predictable rules. His ultimate goal was to prove that a series of rules could be created that would allow a machine to create a copy of itself. Because reproduction is a primary criterion for life, von Neumann believed that the design of a self-replicating machine would be proof that machine life was possible.

2.5.3.4 von Neumann's Automata

The architecture von Neumann proposed was a two dimensional unit grid that had cells called automata on each lattice point. The organism was a collection of these automata arranged in a pattern. Each automata could communicate with its nearest neighbors and receive commands from the tape, a global set of instructions conceptually analogous to DNA. The automata were arranged to form a path that defined a shape. The path is a closed loop, consisting of concentric rings forming walls that bound the tape to continually circulate through the middle. Based on the tape and adjacency information, each automata can assume a state that directs the tape, such as to turn right, go straight, or branch, etc.

Using several thousand automata that each had 28 states, von Neumann produced a *universal constructor* that could replicate any pattern that could be defined by the tape. He demonstrated that his universal constructor was *self replicating* when the machine construction produced an identical copy of itself and the tape. von Neumann became absorbed with the theoretical aspects of automata and spent the rest of his life tying to quantify the mathematical basis for reproduction. He believed that below a certain level, complexity becomes degenerative, and reproduction would be impossible [132].

2.5.3.5 Post-von Neumann Automata Developments

von Neumann's model is quite complex, and most of the work that has since followed has focused on producing a system with the same functionality but with fewer constraints. Codd [133] proposed a simpler model that had fewer automata and only required 8 states to self replicate. Codd eventually used his model to propose some reconfigurable hardware elements; however, Codd's models were too complex to be implemented on the computers of his day.

Langton [134] further simplified the model by removing the constraint that each automaton be capable of reproducing all of the other possible states. Langton integrated some genetic mechanisms into the model by specifying that the information embedded in the automaton should have translated (utilized) and transcribed (nonutilized) elements. Langton's simplified model was simulated on a computer and successfully replicated some simple structures such as circles (Figure 2.74).

Byl [135] further simplified the model by reducing the amount of intercellular communication needed for reproduction. Byl's adaptations essentially removed some of the aspects of biological differentiation from the model.

The simplified automata theory has become the basis of a class of computer science that treats the automata as computation elements. Accelerator boards [136, 137] have been developed for automata programming, thus allowing complex geometric patterns to be produced. Toffoli and Margolus [138] have published numerous examples of automata geometries, two of which are presented here in Figure 2.75

2.5.3.6 Automata and DNA

Hofstadter [139] was the first to propose that automata could be used to model DNA. He developed a one-dimensional model based on the concept of typogenetics that produced character strings symbolically analogous to the four basic nucleotides (section 2.4.1). Morris [140] later expanded Hofstadter's model to the two-dimensional case. Morris' model, like Hofstadter's, performed no function other than self-replication.

In a recent paper Marchal et al. [141] described an automata system that could translated a crude artificial chromosome into a functional structure. The tape of traditional automata was replaced with an artificial chromosome. The chromosome contains the structural information for all of the cells. A 3x3 cell grid was constructed such that each cell extracted only the segments of the chromosome that were needed for its particular function. Marchal et al. used their model in a hardware demonstration to produce a 4-state reversible counter (Figure 2.76).

Automata were first developed as an alternative computational architecture to the traditional fetch-cycle-serial computers. Automata have been shown to be very accurate models of cellular dynamics capable of self replication. Automata have also been shown to be capable of modeling fluid and particle dynamics and able to produce complex geometric patterns. To date the automata is the most accurate mathematical tool for modeling cell interactions and has been used as the foundation for the cell growth simulation program.

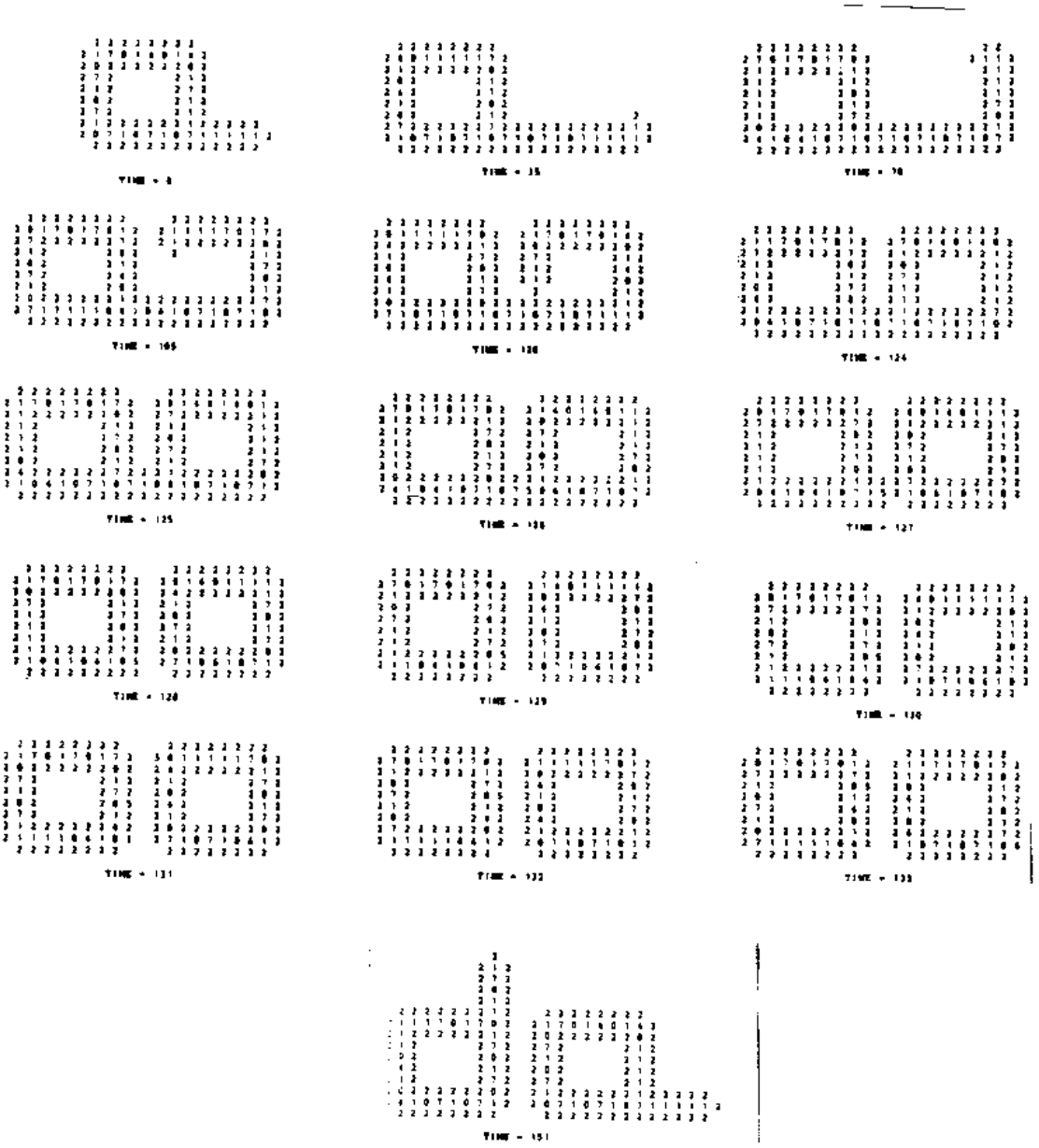


Figure 2.74 Langton's automata. This automata is designed to produce rings. From upper left, (time = 0), to lower right, (time = 151), the automata first spouts an arm that grows linearly at first, then curls to form a second ring. Once the second ring is formed, the bridging element is severed and the two separate rings restart the cycle [134].

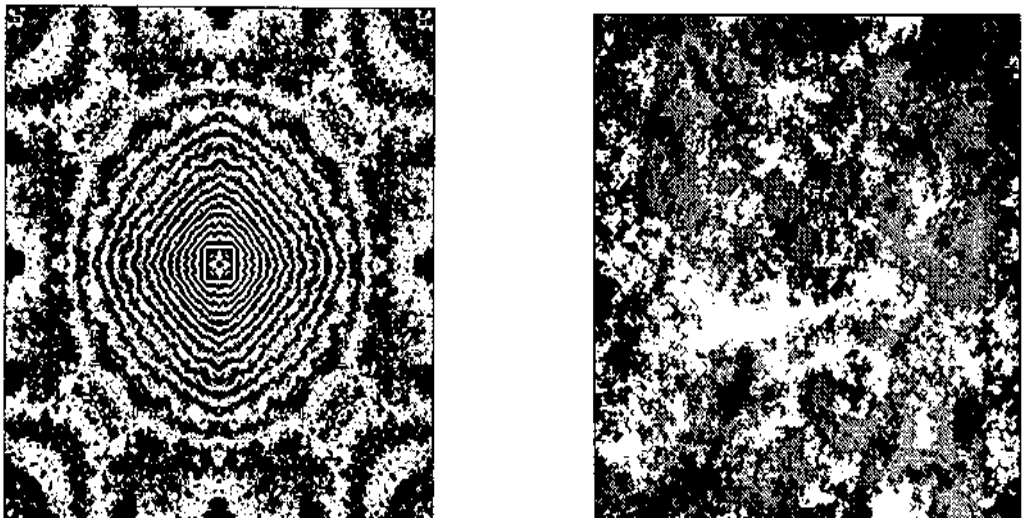


Figure 2.75 Geometric patterns generated with automata [138]. The patterns can be symmetric, **left** or random **right**, by allowing the genetic tape to "drift."

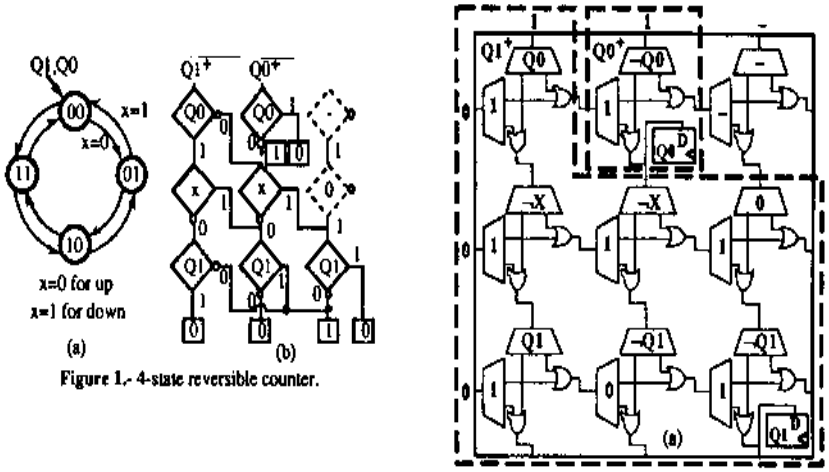


Figure 2.76 4-state reversible counter produced by Marchal's automata. System schematic **left** is reproduced as logic elements **right** [141].

2.6 Chapter Summary

This chapter has provided the knowledge base upon which this dissertation is based. The power of neural networks was shown to increase with network size and that no single accepted method of designing neural network currently exists. Numerous architectures were discussed, none of which are either sufficient for controlling composite manufacturing or have a feasibly capability to be substantially enlarged. Hebbian adaption was shown to be difficult to implement as a training rule, but more analogous to the biological model, having greater potential for use in large scale systems. The area of hardware neural networks is developing rapidly for high speed and large scale systems, yet no guiding principle or theory exists for the design of the network architecture. Neural controllers were demonstrated to be effective for controlling relatively simple system's however, lack the power to interpret the complex output of many of the process sensors proposed for composite manufacture. Neural controllers were shown to be quite effective when designed to emulate or "clone" the abilities of the human operator they were replacing. This established that more effective control of complex processes could be achieved by more accurately replicating human abilities. The success of the CMAC architecture suggests that network processing power can be increased by more accurately replicating the structure of the nervous system.

The neuron and the nervous system were discussed in detail to establish the parameters responsible for and regulating the transmission of electrical signals in the brain. These parameters are used in the next chapter as the foundation for the genetic parameters in the artificial cell growth simulation. The structure of the brain was also discussed and shown to be well organized, providing a guideline for interconnecting multiple artificial cortices. The cerebral hemispheres, which contain virtually all of the higher processing centers that are of interest for control applications, were shown to have a general six strata structure used as the foundation for the mathematical model in the next chapter. The phenomenon of mapping was also discussed and serves as a method for connecting the input and output devices of the cortex.

The structure and function of DNA was described, providing the foundation for the development of an advanced genetic algorithm. It was shown that a gene determines not only the function, but the quantity of the trait it represents. The advanced genetic algorithm that has been discussed is used for the automata based cell growth simulation and for optimization.

A discussion on cognitive science was included to describe how entity representation is thought to occur in nervous systems. The concept of cortical columns and their ability to self organize serves as the foundation for how the massive scale neural networks developed in this dissertation operate, are trained, and learn. The existence of cortical columns relies on the interactions of the various cell types that occur in the cerebral cortex. The need to establish a system where cortical columns can exist and function ultimately requires a system to have multiple cell types. The requirement of having multiple cell types dictates the development of cellular parameters in Chapter 3 that are used to differentiate between the various cell types. Lastly, the automata was introduced as the most accurate model of cellular interactions and its use in the development of the cell growth simulation program.

In the next chapter, the concepts and functionality described in the literature review are converted into parameters and mathematical expressions upon which the cell growth simulation is based.

3. THEORETICAL BACKGROUND

This chapter takes the mechanisms and concepts discussed in the literature review and develops a series of models which shall be integrated in Chapter 4 to produce the cell growth simulation program. Chapter 3 has four main sections. The first discusses the structure of the cerebral cortex and establishes a series of indicial expressions used for defining the location of neurons in the network. The first section also establishes a model that treats the cerebral cortex as a flat, six strata, planar structure. The second section discusses the parameters that define each neuron and how manipulation of these parameters can be used to discern between the various cell types. The neuron parameters also serve as the basis for the genetic parameters or genes, of the artificial chromosome. The third section presents a model for the artificial chromosome. The artificial chromosome is comprised of overlapping subsets of genes that are used to define the various cell types and network structure. The last section of this chapter describes the growth process in mathematical terms. The growth process which has five stages: zygotic, blastular, gastrular, embryonic, and fetal, has been defined as a series of specific changes or differentiations. The process where by the cells differentiate is based solely on the artificial chromosome, adjacency, and diffusional information; thus the algorithm can be applied to a network of virtually any size. The last section also describes the crossover process that occurs during reproduction and the role of chemical signals in cell regulation. This final section describes the events that occur in the simulation program and is perhaps the most important part of this chapter.

3.1 The Basic Neuron Grid

The first step in producing a neural network is defining the architecture. In the case of this dissertation, defining the network architecture is an issue of modeling the mammalian neocortex. This section describes the structure of neocortex for the purpose of establishing an indicial representation used by a computer program to locate the individual neurons. The section begins by converting the complex structure of the cerebral cortex into a three dimensional matrix. The following paragraph describes the mental exercise necessary to understand this mathematical representation.

Referring to neural anatomy, one discovers the gray matter (neuron containing) is actually a thin strata covering the convoluted surface of the brain, as seen in Figure 3.1a. An enlarged detail of this gray matter reveals six distinct strata. If one can visualize the gray matter as a film, then it should be possible to image peeling that film off as seen in Figure

3.1b. Further imagine completely removing the film and laying it flat on a table to produce Figure 3.1c. Once the film is flat, it should be easy to visualize assigning a Cartesian coordinate system to the film. Neurons are located at the lattice points, as seen in Figure 3.1d. In this way, a three dimensional computer array can be used to store the locations of all neurons.

Unfortunately a three dimensional array will not allow a computer program to efficiently locate neurons by layer, which is essential for simulation purposes. To rectify this short coming a four dimensional array, called the unit neuron grid (UNG) will be used. The UNG has four basic parameters, M,N,L, and η representing the width, length, layer and layer thickness. With this representation, M and N define the over all size and L is reduced to being a simple index.

To simplify the program all neurons are located on grid points such that UNG parameter are 16 bit integers instead of 32 bit floats. One of the most important parameters of the neocortex model is layer thickness. There are three basic ways to model thickness; uniform, columnar, and clustered. The first assume that η is a single value. Since the model only uses integers to define the UNG parameters, this approach would limit layer thickness to integer values. Such an integer approximation is not very representative of the biological structure, nor desirable. The second approach assumes each layer is comprised of adjacent columns such that the base of each column is located at a grid point. Under this second model η corresponds to the height of each column. Since each column can be a different height, it is possible to construct a network that has an non integer average for layer thickness, while still only using integers for UNG parameters. While this second approach is dramatically better than the first, it makes interneural distance calculation, (necessary for determining concentration values) cumbersome. The last method, which is used in the program, assumes that the neurons are arranged in clusters centered about the grid points. In essence each point (M,N,L) defines a square sub grid that contains η discrete points. The advantages of this last technique are that it allows non-integer average layer thickness and easy interneuron distance calculations.

To complete the definition of UNG, it is necessary to fully define what part of a neuron is located on a grid point. In nature, a neuron has dendrites and axial terminations in many, if not all of the layers. Even though the nucleus has no function from an engineering point of view, it will be used to define the location of the neuron. Under this scheme, the UNG is used to store the locations of the nuclei.

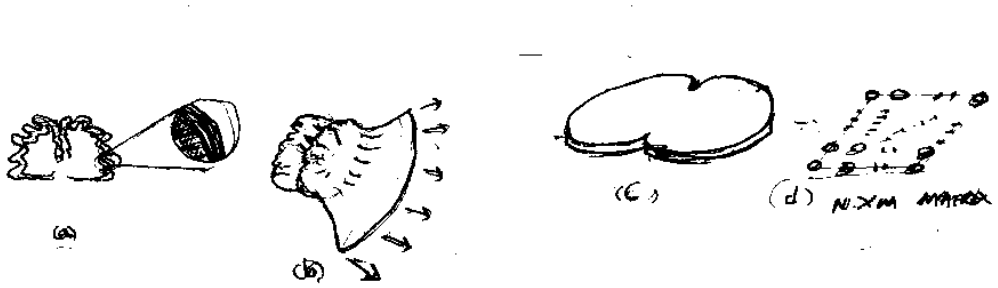


Figure 3.1 From 3-D brain to 2-D unit grid. (a) Cross section of the brain showing interior white matter and the stratified structure of peripheral gray matter, **inset**. (b) Conceptual removal of idealized, elastic peripheral gray mater by peeling. (c) The removed, flattened, idealized, peripheral gray matter from (b), and (d) the N x M unit grid representation of (c).

Recall stratum 1 serves only as an axon conduit for moderate-ranged intracortical communications and thus contains no neuron nuclei. It is therefore not necessary to include stratum 1 as a neuron grid parameter (Figure 3.2). The cerebral cortex can be represented by the three dimensional unit grid:

$$\text{Unit Neural Grid} = M * N * L \tag{3.1}$$

where M and N are the number of neurons along the length and width respectively, and L=5 corresponding to strata 2, 3, 4, 5, and 6.

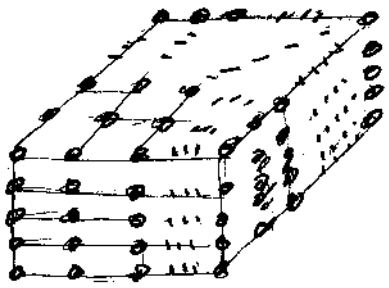


Figure 3.2 M x N x 5 matrix representation of an artificial cortex.

3.1.1 Modeling Cortical Differences

Every cortex of the brain has six strata; however, functional differences require each cortex to have a unique distribution of neuron types and quantities in each strata (Figure 3.3a). Since all strata are bound by the same length and width parameters, deviation in the intrastrata neuron count must be compensated by modulation of the strata thickness (Figure 3.3b).

Equation 3.1 can be readily modified to allow the expression of strata thickness:

$$\text{Unit Neuron Grid} = M * N * H_i \qquad (i = 2, 3,..., 6) \tag{3.2}$$

where H_i is the unique height (thickness) of stratum i. Unfortunately, the strata height uniformity that equation 3.2 produces is not observed in actual cortical strata. The cortical strata have average heights; however, samples collected at random locations across a given cortex reveal variations in the strata thickness, suggesting the following expression:

$$\text{Unit Neuron Grid} = M * N * H_{ijk} \qquad \begin{matrix}(i = 1, 2,..., M)\\ (j = 1, 2,..., N)\\ (k = 2, 3,..., 6)\end{matrix} \tag{3.3}$$

where i, j, and k are the indices for length, width, and height respectively, and H_{ijk} is the height at a given grid point. Equation 3.3 produces a primitive structure with a uniform length and width unit grid with a nonuniform surface (Figure 3.4).

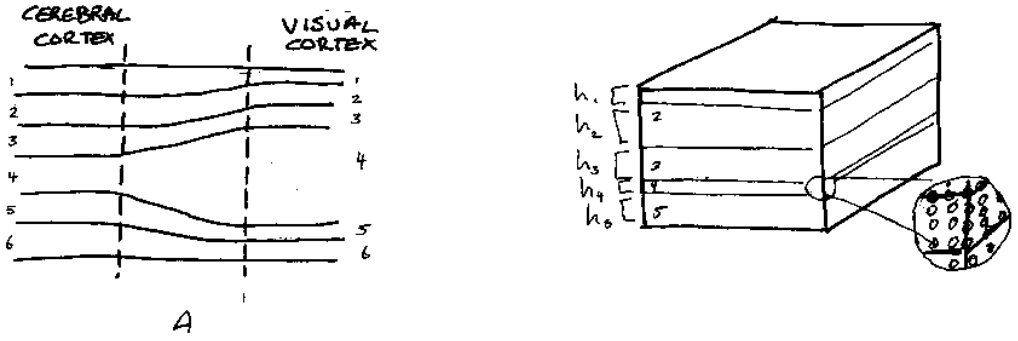


Figure 3.3 Integrating intrastrata height into the cortical grid. (a) Comparison of strata thickness between the cerebral and visual cortices, and (b) corresponding artificial cortical grid with adjustable strata thickness affected by modulation of intrastrata neuron count **inset**.

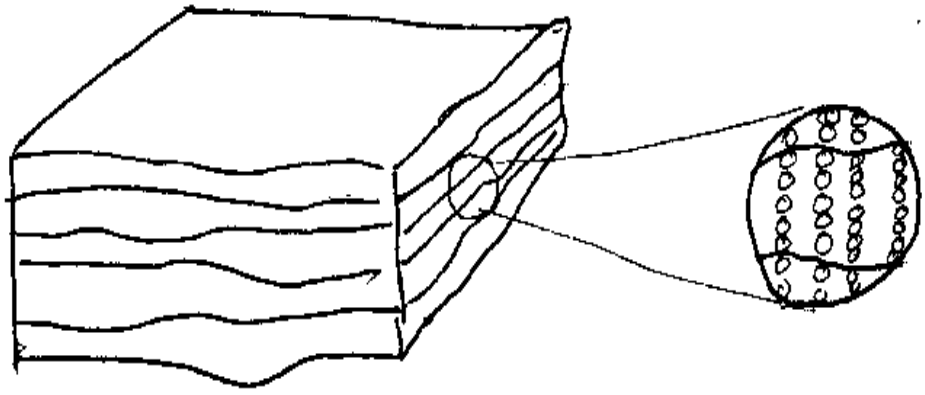


Figure 3.4 Schematic representation of an artificial cortex lattice with nonuniform strata thickness, **inset** affected by modulating the intrastrata count on each point $G_{ijk.}$

The physical representation of H_{ijk} in cell form is awkward from the point of artificial chromosomal encryption and adjacency calculation, therefore on rearrangement:

$$\text{Unit Neuron Grid} = M * N * 5 * \eta_{ijk} \qquad \begin{matrix} (\ i = 1, 2,..., M\ ) \\ (\ j = 1, 2,..., N\ ) \\ (\ k = 2, 3,..., 6\ ) \end{matrix} \qquad (3.4)$$

where η_{ij} is the cell count at grid point (i, j) and can be any positive integer. Equation 3.4 allows multiple cells to occupy the same grid point (Figure 3.5), which allows the representation of nonuniform cortical strata heights to be expressed with a simple three dimensional matrix.

The superposition of multiple cells on a single point is justified given the non-Euclidean packing of actual neurons in the cerebellum and the fact that actual cortical columns (Figure 3.6) are groups of neurons represented by equation 3.5.

$$\text{Cortical Column } (\ i, j\ ) = 5 * \eta_{ijk}(k = 2, 3, ...6) \qquad (3.5)$$

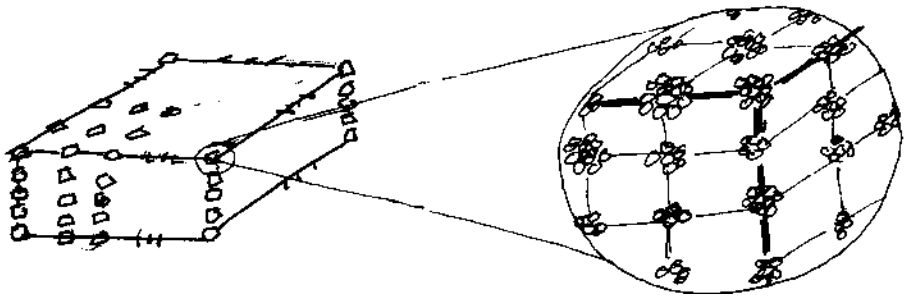


Figure 3.5 M * N * 5 * η_{ijk} unit neural grid **left** and enlarged view showing neuron clusters at each lattice point **right**.

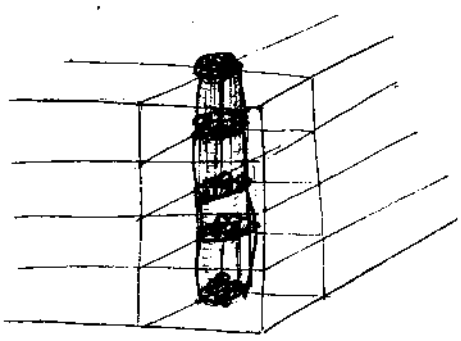


Figure 3.6 Cortical column in the lattice of Figure 3.5.

Under this scheme, a fourth dimensional matrix can be used to store neuron locations:

$$\text{Neuron “Unit” Grid} = M * N * H * (\eta) \tag{3.6}$$

allowing the use of Einstein’s summation convention to represent a single neuron. Thus a given grid location becomes:

$$\text{Neuron } \underline{\mathbf{G}}\text{rid Location} = G_{ijkl} \qquad \begin{array}{l}(\, i = 1, 2, \ldots, M\,) \\ (\, j = 1, 2, \ldots, N\,) \\ (\, k = 1, 2, \ldots, 5\,) \\ (\, l = 1, 2, \ldots, \eta_{ijk}\,)\end{array} \tag{3.7}$$

where i, j, k, M, N and η_{ijk} are defined as before, l is the index for the intrastrata neuron cluster, η_{ijk} and G_{ijkl} are the indicial locations of neurons.

Although the above scheme does fully describe all neuron location and allow strata

thickness to be modeled, it complicates projection, the growth phase where neurons are interconnected. To facilitate neural projections, and simplify the programming effort a convention has been accepted to replace each gird point with a square sub-grid. The sub-grid is defined to be square. In this scheme, not all of the sub-grid locations will be occupied. The size or side of the sub-grid is defined to be smallest whole square root of the actual thickness. For example, actual stratum thickness of 4, 9, and 10 would be represented by sub-grids with sides of 2, 3, and 4. In the later cases, a sub-grid of size 4 would contain 4^2 = 16 sites and thus 16-10=6 vacancies. To avoid pattern effects from developing, the vacancies will be randomly assigned to the sub-grid sites. To allow a fractional stratum thickness such as 10.25, 25% of the sub-grids would have 11 neurons and the remain 75% would have 10 neurons. The implementation of the sub-grid technique is discussed in the next capture. By adopting this convention, a UNG stratum can be uniformly expanded to describe any desired stratum thickness. The sub-grid size is contained in separate vector (t_L), accessed by the UNG parameter L.

In summary, a three dimensional matrix is used for representing the neuron locations in a massive scale neural network. A separate vector t_L is used to store the stratum thickness. This scheme allows each neuron to be easily addressed, variations in stratum thickness to be modeled and facilitates the process of neural projection. Having established the basic neural grid the next section discusses the parameters that define the neurons which when placed in the matrix, define the network.

3.2 Modeling Intrastrata Cell Types

This section mathematically defines cell types. The first part of this section shows that strata are differentiated by cell type and that each stratum is composed of a set of cell types. The cell types in a given stratum exist in determined proportions, that vary between functional regions. A set of parameters is established next which defines the neuron. Different cell types are created by varying the cellular parameters. For a given cell type it is shown that the cell properties are distributed probabilistically, indicating that each parameter has a distribution of values. By defining the cell types parametrically, a general model for the a neuron is produced.

The cortical strata are differentiated by the distributions of the various cell types. The variations seen in the intrastratum cell type populations in different cortices are testimony to the dependence of region functionality on cytoarchitecture (Figure 3.7). To

enable this technique to evolve highly specialized artificial cortices the model developed in this paper has been expanded to allow the encryption of cell type populations.

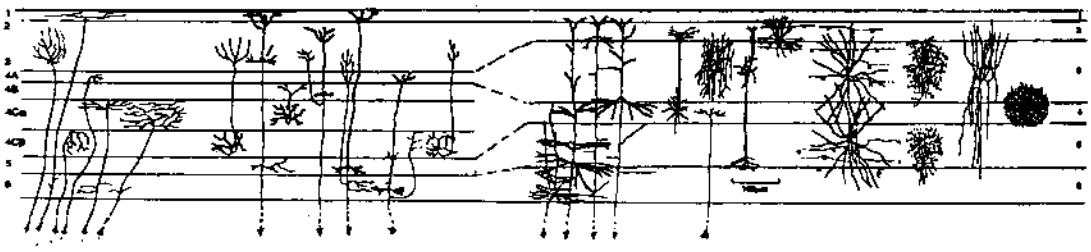


Figure 3.7 Comparison between cell types in (a) cerebral and (b) visual cortices.

Let C equal the set that contains all cell types that can be expressed in a given artificial cortex:

$$C = \{ t_1, t_2, \ldots, t_T \} \qquad t = \#\text{ of cell types.} \tag{3.8}$$

For a given stratum, k, there will be a set C_k of cell types that can be expressed in stratum k such that $C_k \subset C$. C_k is used to produce P_k which contains the population fractions of the cell types in C_k. For example:

$$\begin{aligned} C_1 &= \{t_1, t_2, t_4, t_7\} \rightarrow P_1 = \{p_{t1}, p_{t2}, p_{t4}, p_{t7}\} \\ C_2 &= \{t_2, t_3\} \rightarrow P_2 = \{p_{t2}, p_{t3}\} \\ &\vdots \\ C_5 &= \{t_5, t_8, t_{12}\} \rightarrow P_5 = \{p_{t5}, p_{t8}, p_{t12}\} \end{aligned}$$

where:

$$\sum_{p_i \in P_k} p_i = 1$$

A given neuron can be defined as:

$$\text{Cell Type} = G_{ijkl} * P_k,$$

which indicates that the assignment of cell types can be treated as probabilistically.

3.2.1 Modeling Neuron Parameters Probabilistically

The cell types are differentiated by cytoarchitecture and cellular function (Section 2.3.2) which together are termed features. These features do not exhibit discrete, unique values for each cell type. The values of the features are observed across all cell types as

continuous values. A cell type has features normally distributed across a certain range; therefore, biologists usually use the average value to describe the feature for a cell type. To simplify the discussion it is assumed that all cell features can be described with a Gaussian distribution, unless otherwise noted. The notation that used throughout this document assigns each cell feature a single parameter. The cell features are listed below. To continue this discussion a dummy feature (X) will be used. A given feature, in this case X, denotes its mean, X^{μ}, and variance, X^{σ^2}. The right superscript is reserved for statistical parameters, while the right subscript is used to differentiate between related neural parameters.

$$\mathrm{X} = \begin{bmatrix} X^{\mu} \\ X^{\sigma^2} \end{bmatrix}$$

In addition, the left superscript is used to denote intrastratum dependence, and the left subscript is used to identify to which cell type the parameter belongs. If the left subscript or left superscript is not used, then it is assumed that the cell feature is universal and thus has no cell type or stratum dependence. For example, the following expression is used to characterize parameter X that is associated with length (l) for cell type t_1 and has a dependence stratum k:

$$\underset{t_1}{\overset{k}{}}X_l = \begin{bmatrix} {}^{1}X^{\mu} & {}^{2}X^{\mu} & {}^{3}X^{\mu} & {}^{4}X^{\mu} & {}^{5}X^{\mu} \\ {}^{1}X^{\sigma^2} & {}^{2}X^{\sigma^2} & {}^{3}X^{\sigma^2} & {}^{4}X^{\sigma^2} & {}^{5}X^{\sigma^2} \end{bmatrix}$$

The need for this 2 x 5 matrix is apparent in Figure 3.8 which schematically represents the dendrite terminal lengths per stratum for a pyramidal cell.

The following 13 parameters define the cytoarchitectural features related to primary signal transfer in the cerebral cortex:

1. Dendrite Count 5dc
2. Dendrite Length ..... ${}^{5}d_l$
3. Dendrite Aspect Ratio f(d/l) ..... ${}^{5}d_f$
4. Dendrite Diameter ..... ${}^{5}d_d$
5. Primary Axon Length ..... p_l
6. Primary Axon Diameter ..... p_d
7. Terminal Axon Count ..... ${}^{5}a_c$
8. Terminal Axon Length ..... ${}^{5}a_l$
9. Terminal Axon Aspect Ratio f(d/l) ..... ${}^{5}a_f$
10. Terminal Axon Diameter ..... ${}^{5}a_d$
11. Myelination ..... m^{*}
12. Myelin Sheath Length ..... m_l
13. Excitatory ..... e^{*}

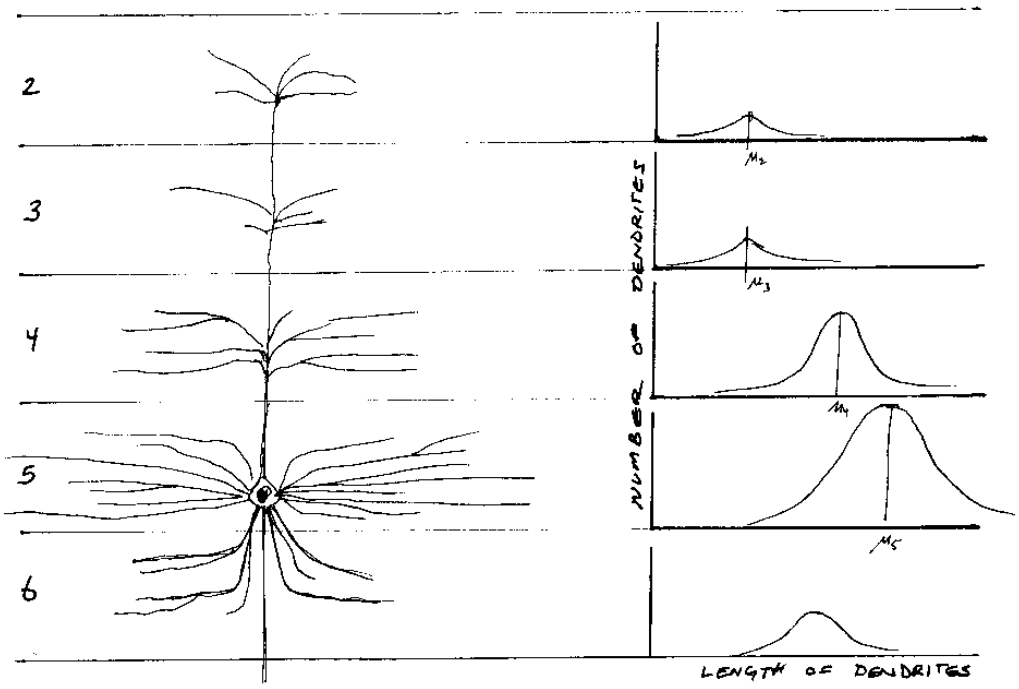


Figure 3.8 The dependence of cytoarchitectural features on a cortical stratum. Illustration (a) of a cerebral pyramidal cell showing dendrite length in each stratum, and (b) corresponding dendrite length distributions per stratum.

The dendrite and axon count terms, ${}^5d_c$ and ${}^5a_c$ respectively, describe the numbers of terminals in the various strata. A similar set of terms, ${}^5d_l$ and ${}^5a_l$, define the average length of the dendrite and axon terminals. To derive a mathematical representation of the terminal diameter, it is assumed that the diameter is dependent on length, and can be related using an aspect-ratio. The diameters are modulated with a pseudo-random generator , which introduce variability to avoid the solution from being biased toward a length dependency.

To ensure that strong, long range connections will be made, the model has two special parameters. The first, the primary length, p_l , defines the length that an axon traverses before a terminal forms. The second term, the primary diameter, p_d, is used to control signal strength through the power-to-diameter ratio defined in section 2.3.2. The terminal myelination and excitation factors, m^* and e^*, are the only non-Gaussian parameters, assuming 0 or 1 for their values. The logic of the two factors is given in Table 3.1. The last term, m_l representing the myelin sheath length is used to calculate the saltatory transmission velocity.

Table 3.1 Cytoarchitecture Factor Definitions

	Logic Definition	
	0	1
Myelination factor - m*	Unmyelinated	Myelinated
Excitatory factor - e*	Inhibitory	Excitatory

The neuron also has a set of parameters that defines the electrical properties of the transmitted signal:

14. Threshold Voltage.....V_t
15. Channel Threshold Voltage.....V_c
16. Sigmoid Shape Parameter.....k
17. Membrane Current.....I_o
18. Activation Potential.....V_o
19. Pulse Duration.....Δt_o
20. Membrane Time Constant.....τ
21. Membrane Length Constant.....λ

These parameters were described in sections 2.3.2.2 - 2.3.2.4 and can be seen in Figure 3.10.

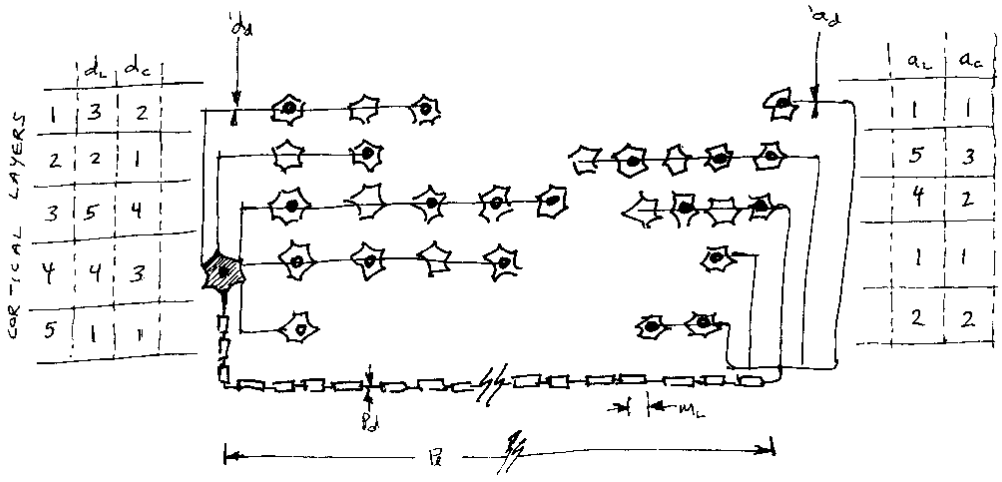


Figure 3.9 Schematic illustration of a neuron and corresponding parameters.

There is also a set secondary parameters, calculated from the primary parameters, that is useful for describing and visualizing neuron behavior:

1. Dendrite Radius.....^5d_r
2. Axon Radius.....^5a_r
3. Dendrite-Synapse Density.....^5d_p
4. Terminal Axon-Synapse Density.....^5a_p

5. Primary to Terminal Cross Sectional Area Ratio .....r_{pt}
6. Signal Power.....P
7. Passive Transmission Velocity.....υ_p
8. Saltatory Transmission Velocity.....υ_s

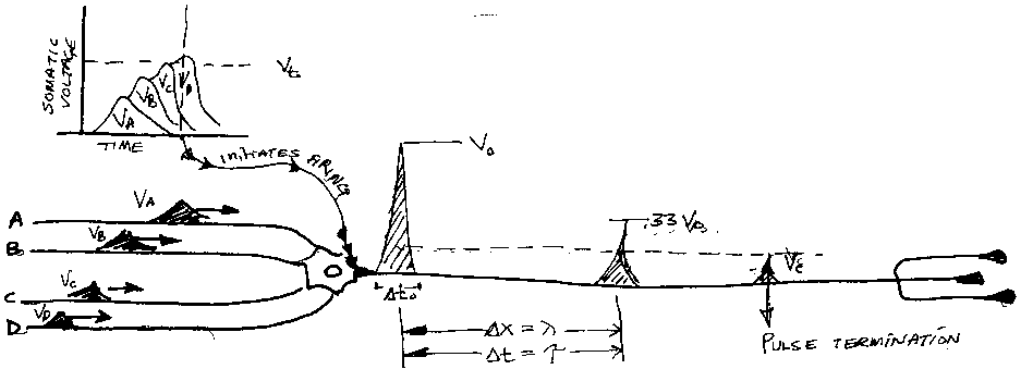


Figure 3.10 Neuron transmission signal and electrical parameters.

The terminal radii, ^5d_r and ^5a_r, define the area within which 99.99% of all interconnections are formed, (3.5 sigma). The terminal radius values are used in conjunction with the average dendrite and axon lengths, d_l^{μ} and a_l^{μ}, to find the area densities $^5d_\rho$ and $^5a_\rho$, which correspond to the fraction of neurons that form synapses with the neurons in question inside the area bound by the terminal radii, ^5d_r and ^5a_r (Figure 3.11) The next term, r_{pt}, is useful for determining the voltage and current split that occurs at the terminal node based on the diameters of the various axon terminals. The signal power and the transmission velocities, defined in section 2.3.2, are used to calculate the temporal and spatial delays, which ultimately determine the long range effectiveness of a neuron.

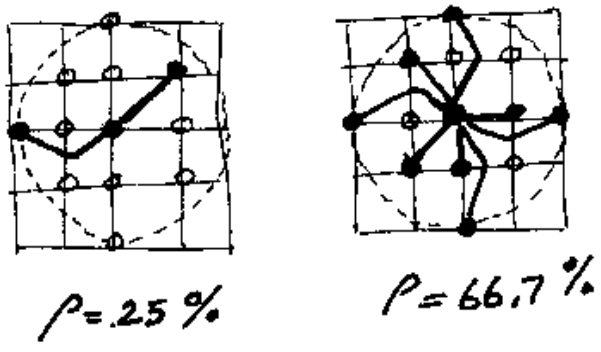


Figure 3.11 The effect of terminal density on the connectivity of neurons. The dotted line defines the terminal perimeter.

3.2.2 Summary

An artificial neuron can be fully defined by 21 parameters, which are Gaussian distributions and are characterized by mean and variance. Several of these parameters have intrastratum dependence, and are therefore represented by 2 x 5 matrices. The complete cell can thus be defined as a single matrix, F_{t_k}, composed of the 21 matrices defining the primary parameters:

$$F_{t_k} = [[d_c] + [d_l] + \cdots + [\lambda]]$$

The expansion of F_{t_k} is below, showing the intrastratum dependence of the parameters.

$$F_{t_k} = \begin{bmatrix} {}^1d_c^{\mu} & {}^2d_c^{\mu} \cdots {}^5d_c^{\mu} & {}^1d_l^{\mu} & {}^2d_l^{\mu} \cdots {}^5d_l^{\mu} & \lambda^{\mu} \\ {}^1d_c^{\sigma^2} & {}^2d_c^{\sigma^2} \cdots {}^5d_c^{\sigma^2} & {}^1d_l^{\sigma^2} & {}^2d_l^{\sigma^2} \cdots {}^5d_l^{\sigma^2} & \lambda^{\sigma^2} \end{bmatrix} + \cdots +$$

In summary, once a cell type has been selected, a whole family of distributions has been selected to describe the cytoarchitecture and behavior contained in the matrix F_{t_k}.

3.3 Artificial Chromosomes

With the neural grid and cell types mathematically defined, this section defines an artificial chromosome that allows the encryption of the parameters discussed earlier in this chapter. This section begins by defining the chromosome as being comprised of a number of genes. Subsets of the chromosome are shown to be associated with structural, functional, and cellular information. The relationship between these subsets is used by the cell growth simulation program. The actual encryption of cellular parameters is described in section 4.6.4. It is interesting to note that this is the first documented description of the encryption of a three dimensional structure into an artificial chromosome.

The artificial chromosome **G** contains hundreds of genes $g_{m.}$ (Figure 3.12).

$$\mathbf{G} = \{g_m\} \qquad (m = 1, 2, ..., m_{tot}) \qquad (3.15)$$

where m_{tot} is the total number of members in **G**. Two types of genes are needed to encode the artificial cortex, structural and functional. Structural genes, G_s, and functional genes, G_f, are subsets of **G** (Figure 3.13) and can be represented as follows:

$$G_s \subset \mathbf{G} \qquad (3.16a)$$
$$G_f \subset \mathbf{G} \qquad (3.16a)$$

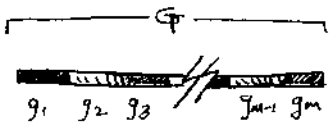


Figure 3.12 Schematic representation of an artificial chromosome showing genes.

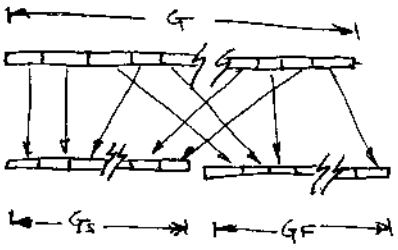


Figure 3.13 Schematic representation of the formation of the subsets G_s and G_f from **G**. Note that there is no restriction on the location or sequence of the genes.

Structural genes are further classified into two subsets. The first subset, ${}^{b}G_s$, defines the neural grid, and the second, ${}^{c}G_s$ is the encryption of cytoarchitecture data contained in F_{t_k}. ${}^{c}G_s$ contains all the genes for all cell types. Each cell type is encoded as subset ${}_{t_k}{}^{c}G_s$ of ${}^{c}G_s$, (Figure 3.14). There is no restriction dictating that each cell type subset must have a unique set of genes. In fact, a more accurate model includes shared genes.

$$ {}^{b}G_s \subset G_s \tag{3.17a} $$

$$ {}_{t_k}{}^{c}G_s \subset {}^{c}G_s \subset G_s \tag{3.17b} $$

Unlike traditional genetic algorithms that represent a feature with a single determinist value, our method uses a range of 2 to 10 genes to define a probabilistic range for a given feature. The encryption of F_{tk} involves the conversion of the two dimensional matrix into a binary string. The mapping of the values is proportional, but not necessarily equal.

Functional genes are used to stove membership information and other parameters that the algorithm needs to decode the chromosome. Aside from membership data, the functional genes determine the order that the various genes are expressed, thereby sequencing various stages of the development process.

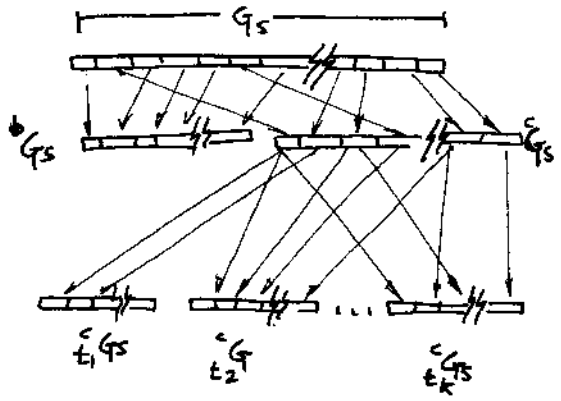


Figure 3.14 Schematic formation of ${}^{c}_{t_k}G_s$, the cell type gene subsets, from ${}^{c}G_s$. Also shown, the formation of ${}^{b}G_s$ and ${}^{c}G_s$ from G_s.

In summary, the chromosome is shown to be comprised of overlapping subsets of genes. The subsets define structural, functional, and cellular parameters. This section laid presented the mathematical foundation for genetic inheritance of a cell line. The concept of inheritance is used to reduce the number of differentiation steps thereby simplifying the cell growth simulation program.

3.4 Artificial Development

In the previous section mathematical representations of the structure of the cerebral cortex, the structure of the neuron, and the artificial chromosome were established. This section discusses the artificial development process and explains how the final structure is produced from the chromosome. In other words the previous section described the encryption of the cortex parameters into the artificial chromosome and this section describes the decryption of the parameters to produce the neural network.

The first five parts of this section describe the basic stages of the development process. During, in the zygotic stage, the chromosome is separated between the expressed and unexpressed genes. Then during in the blastula stage, the overall MxNx5 matrix is generated from a single lattice point. In the gastrular stage the cells divide to produce the various stratum thickness. In the embryonic stage the cell types are defined. Finally in the fetal stage the interconnections and other neurons parameters are defined. The last part of this section describes a way to creating specific neural pathways.

3.4.1 Zygotic Stage

In terms of neural networks, the zygote is the first cell of what eventually becomes the artificial cortex. During this first stage the zygote assumes its phenotype. The genotype is defined by the complete set of genes **G**, which is comprised of equal contributions from both parents. The subset of **G** that is expressed to form the offspring defines the phenotype **P** (Figure 3.15). This stage is necessary to ensure that sufficient variability exists in the genome to allow the differential gene expression needed for evolution. Mathematically, the phenotype separation can be represented as:

$$\mathbf{P} \subset \mathbf{G}$$

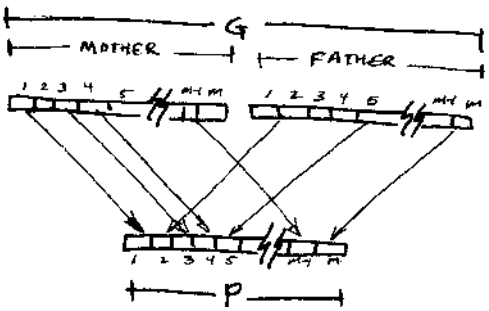


Figure 3.15 Formation of the phenotype from the genotype.

where the selection of **P** from **G** is based on an apriori set of dominance rules, **D**. The dominance rules determine the relative strength of each allele. For the purposes of this study it was sufficient to use the activation threshold values. Each allele has a slightly different activation threshold and the one with the lowest threshold would be expressed, and thus dominant. Since biologists have not verified the threshold concept, the model also allows the alternative dominance rules to be contained either in the functional genes or in the development algorithm. During the zygote stage, all the relationships described in section 3.3 are converted into phenotypic expressions.

$$P_s \subset \mathbf{P} \tag{3.18a}$$

$$P_f \subset \mathbf{P} \tag{3.18a}$$

$${}^{b}P_s \subset P_s \subset \mathbf{P} \tag{3.19a}$$

$${}^{c}_{t_k}P_c \subset {}^{c}P_s \subset P_s \tag{3.19a}$$

Genes that are members of subset **P** are referred to as *phenogenes*.

3.4.2 Blastula Stage

The blastula stage is where the unit neural grid is formed from the single cell zygote. The process involves four steps where the height, length, width, and stratum thickness are produced. The first stage involves the zygote, z, producing four identical copies of itself to form a line of five cells, z_1, z_2, z_3, z_4, and z_5 (Figure 3.16a). This first division is denoted by the subscript *i* corresponding to the first matrix parameter. Next, each of the z_i cells divides M-1 times to produce a 5 x M grid (Figure 3.16b), where each cell is denoted z_{ij}. The third step projects the 5 x M grid by N-1 units to yield a 5 x M x N three dimensional lattice, and assigns cell z_{ijk} to each lattice element G_{ijk} (Figure 3.16d).

Although each cell is essentially "featureless" during the blastula stage, each step further defines the indicia through the addition of indices. The cell's indicia is its generation code or marker that it uses to determine which subset of phenogenes is expressed. At the end of the blastula phase, each cell z_{ijk} assigns a subset P_{ijk} that contains all the data for each possible cell type that could form at that lattice point.

$$z_{ijk} = P_{ijk} \subset \mathbf{P} \subset \mathbf{G} \tag{3.20}$$

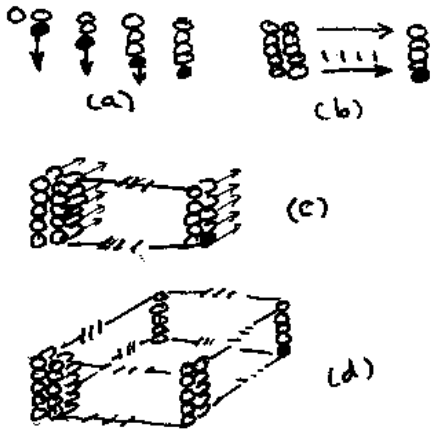


Figure 3.16 Production of the M x N x 5 blastula lattice from a single cell. (a) Lattice development begins with the formation of a line of 5 cells that (b) divide to form a 2-D lattice, which is projected (c) to yield the final lattice (d).

3.4.3 Gastrula Stage

Next the cells next undergo partial differentiation and stratum formation. In the gastrula stage, each lattice point decodes a gene that regulates cell group size. Each cell z_{ijk} has its own potentially unique set of phenogenes, P_{ijk}, which can introduce a large amount of variability to the intrastratum cell group size. Since the phenogene subsets are allowed to intersect, it is possible to produce strata of equal or near equal thickness. The cells at the end of the gastrula stage have already assumed partial identity, and although they still are "featureless," the grouping of the phenogenes that occurs in each cell is irreversibly classified z_{ijkl} to stratum k. At the end of the gastrula stage, the total cell count has been established, and all cells have been assigned a unique lattice location and a unique generation indicia.

3.4.4 Embryonic Stage

Once all the cells have been assigned, they must be differentiated into their final cell type. This differentiation process can be approximated by a three step iterative process. The first step involves the determination of the type and quantity of ligands and receptors a cell produces. The ligands are binary codes that represent the intercellular chemical signals used by real cells. Similarly, the receptors are logic elements that mimic the structures in biological cells that detect particular ligands (Figure 3.17).

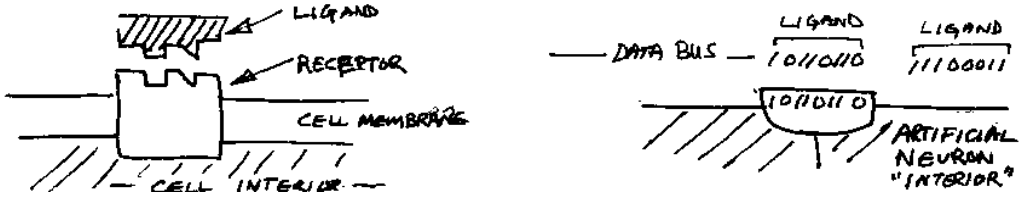


Figure 3.17 Schematic representation of a ligand-receptor pair corresponding to (a) conformational molecules in the biological model and (b) the corresponding digital AND gate of our model.

For simplicity, our model assumes that the number of receptors is proportional to the cell's sensitivity, or responsiveness to that particular ligand. It should be noted that the cells sensitivity includes all possible mechanism that collectively define a gene's threshold of activation. The number and type of ligands and receptors is proportional to the cell type

probability vector, P_k, and are translated to the ligand vector, L_{ijkl} and the receptor vector R_{ijkl}.

$$L_{ijkl} = [\, l_1, l_2, \ldots l_q \,] \tag{3.21}$$

$$R_{ijkl} = [\, r_1, r_2, \ldots r_s \,] \tag{3.22}$$

In the second step, a ligand concentration gradient field is produced. The contribution of one cell's production to the concentration of a particular ligand *[l]* as observed by another cell, is proportional to the inverse of the cube of the intercellular distance, $[l] \propto 1/d^3$ (Figure 3.18).

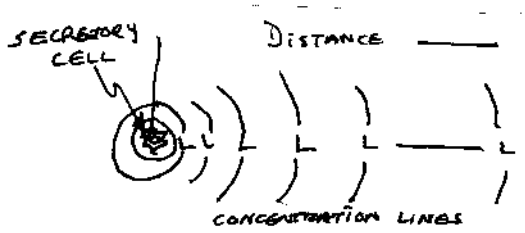


Figure 3.18 Effect of distance from the secretory cell on ligand concentration.

From the gradient field it is possible to determine the total concentration of the various ligands at any given lattice point. Vector χ_{ijkl} contains the concentrations of the various ligands, Λ_n at lattice point G_{ijkl}.

$$\chi_{ijkl} = [\, \Lambda_1, \Lambda_2, \ldots \Lambda_n] \tag{3.23}$$

where the subscript *n* denotes the total number of ligands in the system. The transpose of L_{ijkl} is multiplied with R_{ijkl} to yield χ'_{ijkl}.

$$\chi'_{ijkl} = \underline{\delta\chi}^{t}_{ijkl} \bullet R_{ijkl} \tag{3.24}$$

The last step is the assignment of cell types based on the largest element of χ'_{ijkl}. Once all of the cells have been assigned a cell type, the phenogene subset P_{ijk} further differentiates to allow other genes to be expressed. The new subset of the phenogene, ${}_tP_{ijk}$, defines a cell type that may or may not include genes encompassed by $P_{ijkl,}$.

$$P_{ijkl} \subset P_{ijk} \subset \mathbf{P} \tag{3.25}$$

$${}_tP_{ijk} \subset P_{ijk} \subset \mathbf{P} \tag{3.26}$$

$${}_tP_{ijk} \neq P_{ijkl} \tag{3.27}$$

Once the cells have been assigned a type, the process returns to step one and the new cells once again produce ligands; however, the type and concentration of this second wave of ligand release is governed by the new phenogene subsets, ${}_tP_{ijk}$. A new ligand concentration gradient is produced, from which new χ'_{ijkl} vectors are generated. It is possible that some reassignment of cell types may occur for several iterations. The process continues until the percentage of cell type conversions has decreased to a desired level.

3.4.5 Fetal Stage

At the end of the embryo stage, all of the cells have been assigned a cell type, but the interconnections are not yet formed. The interconnections form during the fetal stage. There are three types of interconnections; intracortical, intercortical and extracortical. All three follow the same basic mechanism. There are three main discussion associated with the process of interconnection. The first describes the complete theoretical versions of the events responsible for dendrite and axon growth. The second describes the assumptions used in this technique that simplify the process. The last discussion describes projection and mapping, which are the processes through which intercortical and extracortical connections are formed.

3.4.5.1 Theoretical Description

The fetal stage is divided into a series of components referred to as growing cycles, Γ_t. At the beginning of Γ_1, the cells releases ligands from which a differential gradient $\delta\chi_{\Gamma 1}$ is calculated. In a similar fashion, as demonstrated in the previous section, the transpose of the ligand concentration gradient vector of χ_{ijk} is multiplied with the receptor vector R_{ijkl} to yield the χ'_{ijkl} . In the fetal stage, the largest element χ'_{ijkl} is used to determine the type of growth, and the derivative of the gradient, $\delta\chi_{ijkl}$ is used to determine the direction of the growth (Figure 3.19).

For the discussion of the fetal growth cycle, the cell that releases the ligand is referred to as the secretory cell, and the cell that receives the ligand is called the receptory cell. Two forms of growth occur during the fetal stage, axial and dendritic. Both types of growth obey the same mechanisms. The secretory cell releases a ligand that initiates growth in the receptory cell. Receptory cell growth initiates at the site of the receptor closest to the sharpest gradient of the particular ligand to which it is tuned (Figure 3.20a). As the growth proceeds in the direction of the sharpest gradient, the receptor is carried by the developing terminal Figure 3.20b). When the receptory terminal reaches the secretory cell, a synapse is

formed at the location of the nearest ligand release site. In this model, both the receptor and the release site are lost during the formation of the synapse (Figure 3.20c). Thus after synapse formation, the secretory neuron's capacity to produce the ligand, and the sensitivity of the receptory neuron to that ligand have diminished.

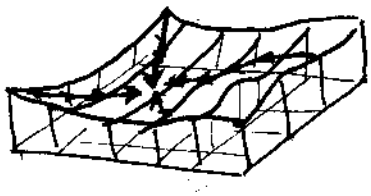


Figure 3.19 Inverse concentration gradient showing the direction of growth.

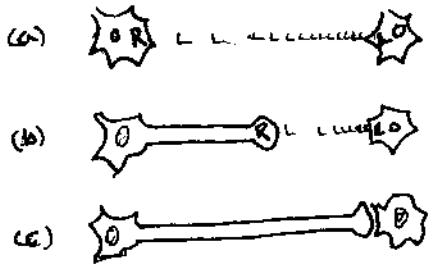


Figure 3.20 Formation of an axon or dendrite based on a ligand signal. (a) Detection of a signal, (b) transient growth, and (c) synapse formation and simultaneous loss of ligand release and receptor sites.

Since the growth of the terminal occurs at the site of the ligand receptor, the determination between axon or dendrite formation is based solely on the locations of the receptor on the receptory cell's surface. Those receptors on the top half of the cell (*animal pole*) will form dendrites, while receptors on the bottom (*plant pole*) form axons. Our model will assume that the set of receptors R_{ijkl} can be divided into two unique subsets, plant receptors, ${}^{a}R_{ijkl}$, and animal receptors, ${}^{d}R_{ijkl}$:

$$ {}^{a}R_{ijkl} \subset R_{ijkl} \tag{3.28} $$

$$ {}^{d}R_{ijkl} \subset R_{ijkl} \tag{3.29} $$

$$ {}^{a}R_{ijkl} \cap {}^{d}R_{ijkl} = \phi \tag{3.30} $$

Thus an activated ${}^{a}R_{ijkl}$ receptor initiates axial growth and ${}^{d}R_{ijkl}$ initiates dendritic growth. Similarly, the type of synapse is determined by the location of the ligand release site. Our model assumes ligand release sites, L_{ijkl}, thus ligands can be separated into dendritic, ${}^{d}L_{ijkl}$, somatic, ${}^{s}L_{ijkl}$, and axial, ${}^{a}L_{ijkl}$, subsets (Figure 3.21):

$$ {}^{a}L_{ijkl} \subset L_{ijkl} \tag{3.31} $$

$$ {}^{s}L_{ijkl} \subset L_{ijkl} \tag{3.32} $$

$$ {}^{d}L_{ijkl} \subset L_{ijkl} \tag{3.33} $$

$$ {}^{a}L_{ijkl} \cap {}^{s}L_{ijkl} \cap {}^{d}L_{ijkl} = \phi \tag{3.34} $$

The model allows all permutations of the ligand-receptor subsets to occur; for example, axodendritic, (${}^{a}R_{ijkl}$-${}^{d}L_{ijkl}$), axosomatic, (${}^{a}R_{ijkl}$-${}^{s}L_{ijkl}$),...etc.

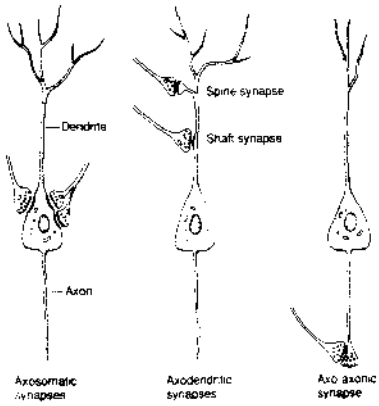


Figure 3.21 Depiction of synaptic locations on biological neurons. From left to right, axosomatic, axodendritic, axo-axonic synapses [27].

In each growing cycle, each receptory cell that has an activated receptor produces one new terminal. Since several receptory cells may have the same receptor, it is probable that the secretory cell located at a local maximum of the concentration gradient, χ, will form many synapses in one cycle. When this occurs, the majority, if not all, of the secretory cell's ligand release sites are lost. Thus the cell is no longer the local maximum of χ, allowing other secretory cells of the same ligand to form synapses during the next growth cycle. Growth cycles continue until no new synapses form.

3.4.5.2 Model Simplifications

The previous section a complete, though complicated technique for forming interconnections was introduced. In this section, a less complicated alternative is introduced.

The method utilizes the intrinsic order of the stratified cortex, and inherited cytoarchitectural properties from the embryonic stage. Originally it was believed that the actual growth of the dendrites and axons would have to be simulated in order to produce the correct structure. This assumption was found to be false. After the embryonic stage, all the neurons have been assigned a type, and thus a set of cytoarchitectural features. Since the inherited features define how the cell will grow, it is possible to predict the interconnections probabilisticly.

The two most important terms for the probabilistic determination of the interconnections are the radius and density. First consider the case where interconnection densities (d_ρ and a_ρ sec. 3.5) are equal to 1. In this simple case, a given neuron will form connections with all other neurons that are within its radius. It is easy to visualize selecting one neuron, drawing a circle of radius a_r, and counting all other neurons within that circle.

Next consider forming the connections across two strata. Regardless of how many neurons a given stratum contains, its external dimensions are the same as all other strata. Consider a basic cortex with strata 2,3,4,5 and 6 that have thickness of 1,4,3,11, and 4 respectively. Obviously the sub-grids will have sizes of: 1,2,2,4, and 2 respectively, and stratum 5 will contain 8 time as many neurons as stratum 2. In order to allow the strata to be overlaid, a compression-expansion term will be used. The convention of compressing the "thicker" strata has been selected as it requires no manipulation of the unity case. To stack strata 2 and 5, stratum five will be effectively compressed by 4. Note the compression term is equal to the sub-grid size, and thus explains the extended derivation of the square sub-grid convention. To form interconnections, the computer will raster across the stratum (2), than adjust the M and N terms by the compression value so that the inherent logic of the vertical stack is not lost. For example, neuron (3,4) of stratum 2 would be directly above neuron (12,16) of stratum 5. In a similar way, the interconnection radius of the stratum 2 neuron would be adjusted from 3.5 to 14. In this way strata of different thickness can be easily overlaid.

At this point, it should be obvious, that forming connections is a simple task of rasterring through a stratum, identifying the cell type, cross referencing its properties, compressing/expanding other strata, and storing location of neurons with which synapses will form. To form an axodendritic connection, the dendritic and axial radii are combined to increase the effective radius.

In the case that the interconnection density is less than 1, set of "fetal" differentiations will be required to form sub-cell types. The fetal differentiation are very similar to those of the gastrula stage. There will be 1/ (d_ρ or a_ρ) sub-cell types per

interconnection per stratum. In other words, for a d_p =.25, there will be 1/.25= 4 sub cell types such as a type 1A, 1B, 1C, and 1D. Any fraction is possible, but the computer program developed in this work was limited to whole number fractions less than sixteenths. For a density of 0.75, the neuron in question will connect with three of the four sub-types. After the fetal differentiations are complete, the program will raster though the stratum, compressing and expanding when ever necessary as before, but only forming connections between similar subtypes. The method is highly repetitive and has a simple elegance that only nature could have originated.

3.4.5.3 Intercortical and Extracortical Connection

Intercortical and extracortical connections are formed using the same basic procedures used in forming intracortical connections, with the exception that the strata belong to different cortices. First consider the case where the cortices have the same M and N dimensions. In the case, forming intercortical connections between strata 2 and 5 of cortices C1 and C2 respectively follows exactly the same procedure as described above. If the M and N dimensions are dissimilar, an additional compression/expansion term will be used to scale the two cortices. The convention of compression the larger cortex has again been adopted. When intercortical connections are being formed, it is assumed that the (0,0), and (M,N) neurons are located at the same corners respectively. The formation of intercortical connections is referred to as projection, as in the neurons of one cortex project onto another. Projection allow a multi-cortex neural network to be developed.

Forming extracortical connections follows nearly identical steps as describe above. Extracortical connections are the input and outputs of the massive scale neural networks. To form an input, the output of the sensory device must be place on a M by N grid called the input array. For example, an CCD camera's image plane could be used as in put array. The input array is treated as a external cortical stratum, that projects onto a cortex. When there is some distortion of the input array such that certain portions receive preferentially more cortex real estate the process is called mapping (see chapter 2). Extracortical connections also provide the output of the massive scale neural networks. The extracortical connections only differ from those of other cortices in that the signals project or map onto an output array. The output arrays thus drive the output devices.

In summary a simplified technique for forming interneural connections probabilistically has been developed. The technique is applicable for intracortical,

intercortical, and extracortical connections. This section has laid the ground work for designing multi-cortex neural networks and integrating input and output devices.

3.4.6 Genetically Hardwired Neural Pathways

There are two types of Ligand and Receptors Pairs (LRPs), general and specific. The general LRPs produce a pseudo-random structure that is used to produce the redundant synapses used for learning. The discussion thus far has concentrated on the case of the general LRPs, where a large percentage of the total cell population contains either the ligand or a receptor. The specific LRPs occur in a very low percentage of the neurons. If it is desired to connect one particular cell with another particular cell, a specific LRP pair is coded into the chromosome that can only be expressed by the two cells in question. Again, the expression of the specific LRP is controlled by the generation indicia and the cell type selection from the second stage of the development process.

LRPs are responsible for the formation of fundamental pathways and the mapping of sensory or motor neurons onto the surface of the artificial cortex. The fundamental pathways are responsible for instinctive behavior. Since all learning is associative, it is crucial that some fundamental pathways be integrated into the artificial cortex so that new inputs may be associated with existing neuronal groups. The ability to genetically map sensory or motor stimuli onto the surface of an artificial cortex allows the development of highly specialized controllers and signal processors.

3.5 Chapter Summary

Chapter 3 has established the theoretical background for the cell growth simulation program. A four dimensional matrix has been shown to be suitable for modeling the cerebral cortex. The neuron has been parametized and the properties and features of the neuron have been shown to exist as a distribution of values. The cerebral cortex was shown to be comprised of various cell types. These cell types can be derived by modulating the neuron parameters of the generalized neuron. The chromosome was shown to be made up of overlapping subsets of genes which define structural, functional, and cellular properties. The development process has been shown to proceed through distinct stages. The decisions of one stage cause the inheritance of properties to the next. The development process has been described with the mathematical expressions developed in first half of the chapter. In Chapter 4 the mathematical tools developed in this chapter are used to develop the cell

growth simulation program which is used to develop massive scale neural networks capable of controlling composite manufacturing.

4. DESIGNING AN ARTIFICIAL NEURAL CORTEX

In the last section the mathematical and theoretical foundations were established upon which a growth simulation program has been developed and used to design massive scale neural networks. This chapter takes the expressions and concepts presented in Chapter 3 and demonstrates their implementation. There are three main sections in this chapter. The first describes which parameters were used and from where their appropriate values were obtained. The second section describes the process of converting the values collected in the first section into the artificial chromosome that is actually used by the simulation program. The second section also includes a detailed discussion on the development of the advanced genetic algorithm upon which the simulation is based. The last section describes the actual development of the program. This section specifically describes how cellular interactions are modeled, how differentiation decisions are made, and the mechanisms through which the program functions. The chapter concludes with a series of plots showing the incremental development and spontaneous self organization that occurs during the operation of the simulation program.

4.1 Collecting Properties

This section describes the process through which the properties of the cells of the artificial cortices are collected. Essentially it describes the conversion of biological observation into engineering data. There are three basic steps in this process:

1. Identifying regions or corticies of the neural system that perform similar functions as those desired for the particular application.

2. Identifying the cell types which constitute the cortex from step 1.

3. Collecting the cell parameters that fully define the cell's electrical and structural properties.

The main emphasis of this section is to describe why the particular values were used and the significance of these values. This section also describes the concept of inheritance which uses primary properties to determine secondary properties.

4.1.1 Identifying Cortical Regions

The process of selecting an artificial cortex is rather straightforward. It involves the matching of existing cortical functionality with the user's particular application. The cortex to be selected need not be of human origin. Actually, the human brain is probably a poor choice for most engineering applications because it not optimized for specific tasks, but rather general in structure and function. Most engineering applications are very specific and more closely resemble the behaviors performed by simpler animals, such as frogs or bats, which have specialized neural structures for visual and acoustical detection of prey respectively. To clarify this point, consider the following engineering applications:

Table 4.1 Sensory Corticies

Application	**Animal**	**Cortex**	**Properties**
Part Location	Frog	Visual	Pre-programmed to locate a few specific visual entities only.
Visual Inspection	Eagle	Visual	Optimized to locate small aberrations (prey) in visual field.
Tumor Location	Human	Primary and Secondary Visual	Able to compile several different image types, such as X-ray and MRI, of the same objects to enhance feature separation. Able to handle extremely large number of similar entities.
High Frequency SONAR	Bat	Auditory	Optimized for operating in air.
Low Frequency SONAR	Whale	Auditory	Optimized for operating in water.

In the first example, part selection, the visual system would only need to locate entities and make gross classification. The frog was selected as a model because its visual system is genetically "pre-programmed" only to respond to certain images (flies) thus reducing the time needed for training. The second example requires a visual system able to detect small abnormalities in the visual field. The frog's visual system is inadequate because it lacks the resolution and structure required to detect small variations in the entities that it is optimized to locate. On the other hand, an eagle routinely locates prey from great distances, an operation comparable to the needs of the second example. Automated visual inspection systems require the detection of small defects; however, classification of those defects is of secondary importance and most engineering applications only have a few types of defects that

need to be identified. These criteria match the functionality of the eagle's visual cortex which has great acuity and resolution, but limited ability to process the visual information much beyond the classification of prey types. The third example, tumor location, is customarily performed by a physician who is able to compile a variety of diagnostic image types (X-ray, CT scan, MRI, etc.). The compilation of multiple image types is complicated by inter-patient variability, and the poor quality of most images of soft tissue. The poor image quality in a sense creates the need to compile several different tests because a complete diagnoses can rarely be made after only one test. Since the ability to correlate a variety of images and data types into a single meaningful form is the unique ability of man, one would like to replicate the human secondary visual cortex for medical imaging.

SONAR and its close cousin RADAR are rich areas for neural network applications. The U.S. Navy has been conducting experiments on bats and whales to this end. These last two examples are introduced to illustrate that the animal kingdom has produced a variety of related systems that operate in different frequency ranges and are set up in quite different ways. These latter cases serve to alert the engineer that although the selection of a cortex model can be straightforward, a careful review of comparative animal physiology may locate systems more closely suitable for modeling.

In the case of this initial study, the generalized model of the cerebral cortex, as described by neural scientists, was selected because it can be modified to yield any particular cortex. The generalized cerebral cortex turned out to be a good selection because it has a well defined six strata structure with well documented cell types facilitating the model's development.

4.1.2 Identifying Cortical Strata

Once a cortex has been selected for replication, several properties must be extracted, the first of which is stratum thickness. For simplicity, Figure 4.1 is treated as a true-scale figure, and is used as the source of the cortical properties needed for the growth simulator.

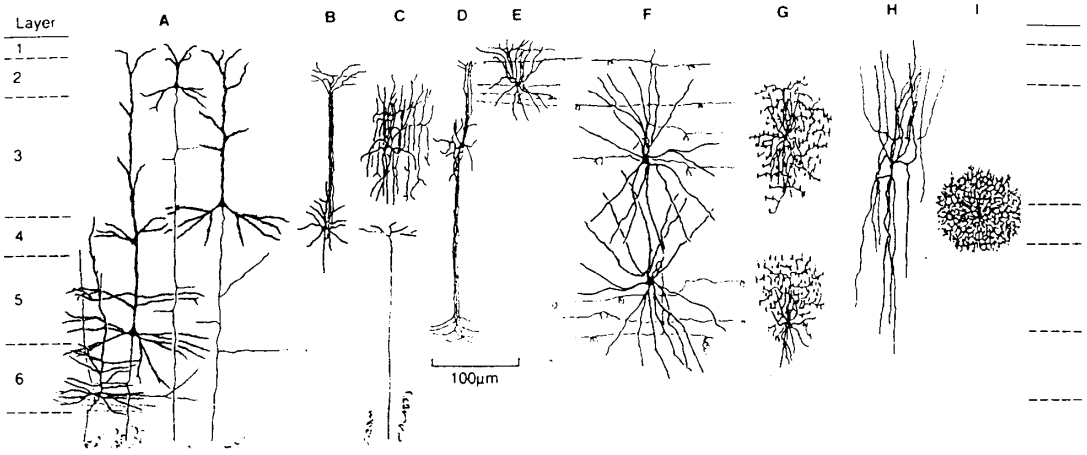


Figure 4.1 Neurons of the Cerebral Cortex

As discussed in sections 3.3 and 3.4, the cerebrum could be represented by a MxNx5xJ matrix. However, this representation tends to be cumbersome from a software point of view. Instead, each stratum of the cortex is represented by a two dimensional array represented by [($\sqrt{J}$ *M) x ($\sqrt{J}$ *N)]. To perform the conversion to a two dimensional array, it is first necessary to normalize the strata. Given that the average cell diameter is approximately 10 µm, the unit cell count of a particular stratum can be found by dividing the stratum thickness by 10 µm. The unit cell count represents the number of cells that can be vertically stacked in a column perpendicular to the surface of the stratum. Since each stratum has a different size, each stratum will have a different unit cell count. Using the scale indicated in Figure 4.1, the stratum thickness and the unit cell count can be easily obtained, see Table 4.2.

Table 4.2 Stratum Thickness of the Generalized Cerebral Cortex

	Stratum 1	Stratum 2	Stratum 3	Stratum 4	Stratum 5	Stratum 6
Stratum Thickness (µm)	17.5	28	105	34	77.5	62
Unit Cell Count (cells)	1.75	2.8	10.5	3.4	7.75	6.2

4.1.3 Identifying Cell Types

Having identified the stratum thickness, the next step is to develop an appropriate means of classifying the cell types. There are several ways to do this, and all are necessary. First the cells shall be numbered from left to right, as seen in Figure 4.1. In addition to the 17 cells, there are two afferents, numbered 18 and 19, that are the axons of cells from other areas and correspond to the inputs. The first classification is between pyramidal and nonpyramidal cells. Pyramidal cells, as discussed in Chapter 2, each have a long axon that projects out of the cortex and can be considered the output. Cells 1-4 are pyramidal cells having different sizes and stratum locations. Each of these four pyramidal cells constitutes a different output type, the significance of which becomes apparent when the integration of multiple cortices is addressed in a later section.

Table 4.3 Excitatory and Inhibitory Neurons

Type	**Cell Number**	**Cell Symbol**
Excitatory	1,2,3,4,5,6,7,8,18,19	
Inhibitory	9,10,11,12,13,14,15,16	

The second classification is between excitatory and inhibitory cells, (see Table 4.3). The pyramidal cells are actually a subset of the excitatory cells, which include cells 1-8. The third way to classify cells is by stratum. Since many cells have axons and dendrites which project through several strata, the convention of classifying the cells according to the stratum in which their cell body and nucleus are found is employed. This classification scheme is consistent with Equation 3.8 and yields the subclasses listed in Table 4.4.

4.1.4 Cell Parameters

In section 3.5 a variety of parameters are listed, all of which control different aspects of cellular function. While all of these parameters are necessary for the operation of the neural network, very few are needed during the development stage. All of these properties are inherited once the cell type has been assigned, but some properties are only partially defined when inherited. Those that are fully defined are secondary properties, and those that are only partially defined are primary parameters (Table 4.5). Only the properties listed as primary need to be considered as developmental parameters to define the network. The primary parameters are considered partially defined because their value has the potential of becoming one of several possibilities. In other words, there may be three genes related to

dendrite length, but only one of the three is expressed in a particular cell. Which of the three possible genes are expressed is determined by concentration gradients that develop during the simulated development.

Table 4.4 Cell Type per Stratum

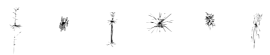
Stratum	**Cell Number**	**Cell Symbol**
1	19	
2	3,5,11	
3	4,9,10,12,14,16	
4	6,8,17,18	
5	2,13	
6	1,7	

Table 4.5 Primary and Secondary Cell Parameters

Primary Parameter	Secondary Parameter
Cell Type	Excitatory
Cell Type Fractions	Terminal Thickness
Maximum Dendrite Length	Dendrite Aspect Ratio
Maximum Axon Length	Axon Aspect Ration
Minimum Dendrite Length	Dendrite Diameter
Minimum Axon Length	Axon Diameter
Intercellular Associations	Myelination
Dendrite Count	
Axon Count	**All Electrical Properties

Although the primary parameters are discussed here and are used during the growth simulation, it is not to downplay the significance of the secondary parameters. The primary parameters are used first, during the growth simulation development phase of the network design. The output of the growth simulator is a series of lists of interconnections that fully define what all of the cells have become and to which other cells they are connected. This data is then used by an assembler program that actually generates the functional neurons that

are used during the training phase. During the second phase, assemblage, the secondary parameters are used to fully characterize the electrical behavior of the cells and to establish initial synaptic weights. In other words, as the cells emerge out of the growth simulator, they are fully defined even though all of their characteristics have not yet been calculated. They are fully defined because the transcription of the secondary properties that occurs during the assemblage program is a deterministic process based on the primary parameters. All variability, be it random, probabilistic, or chaotic, occurs solely in the growth simulation program and only over the primary parameters.

4.1.4.1 Cell Types and Fractions

The most important parameter is cell type. Essentially, it defines all other properties that the cell inherits. The cell types listed in Table 4.4 are used in the example. While cells have been assigned to particular strata in last section, the numerical proportions of their occurrence, (P_k of Equation 3.8) has not yet been specified. The cell type fractions are listed in the table below. Table 4.6 is comprised of the actual input used in the example solutions discussed in this paper.

Table 4.6 Cell Type Stratum Fractions

Stratum	Cell Number	Cell Symbol	Fraction
1	19		0.0
2	3,5,11		0.412, 0.294, 0.294
3	4,9,10,12,14,16		0.179, 0.108, 0.108, 0.162, 0.162, 0.262
4	6,8,17,18		0.333, 0.333, 0.333, 0.0
5	2,13		0.433, 0.333, 0.233
6	1,7		0.5, 0.5

Note that the afferents, cell types 18 and 19, both have occupancy fractions of 0.0, indicating that they do not occupy a lattice point. This occurs because the afferents are axons of cells located in a different cortex, thus they are already associated with another lattice point. If the afferents were assigned a lattice point in this cortex, then they would essentially

exist in two places in the network simultaneously, thereby creating an illogical and impossible situation. Note also that no neurons are assigned to the first stratum which is reserved for interconnection purposes only.

4.1.4.2 Terminal Lengths

The terminal lengths are properties inherited from the cell type; however, they are extremely important for the simulation program because they define the range over which the cells can interconnect. While the importance of the maximum dendrite and axon lengths for defining a cell's interconnection range is intuitive, the importance of the minimum length parameters is less evident. The minimum lengths are crucial parameters for several of the inhibitory neurons, specifically the basket cell type. The basket cells are primarily responsible for the separation of the cortical columns. This function occurs because the basket cells inhibit only their distant neighbors. The interconnections of a basket cell form a doughnut shaped connection pattern (Figure 4.2), where the hole corresponds to the cortical column and the cell body occurs at the center of the hole. Without this minimum terminal length parameter this *doughnut* cannot form, therefore neither can the cortical columns.

Initially the collection of the terminal lengths seemed to be an overwhelming task as all of the nerve cells appear to be highly branched, with terminals appearing in multiple strata. However, a few guidelines greatly simplified the task. First, nonpyramidal cells have either very short dendritic spines or none at all, very short meaning less than 10μm or one cell diameter. Second, inhibitory neurons predominantly form axosomatic synapses. This second rule dictates that inhibitory neurons never form connections on the dendritic spines of the pyramidal cells.

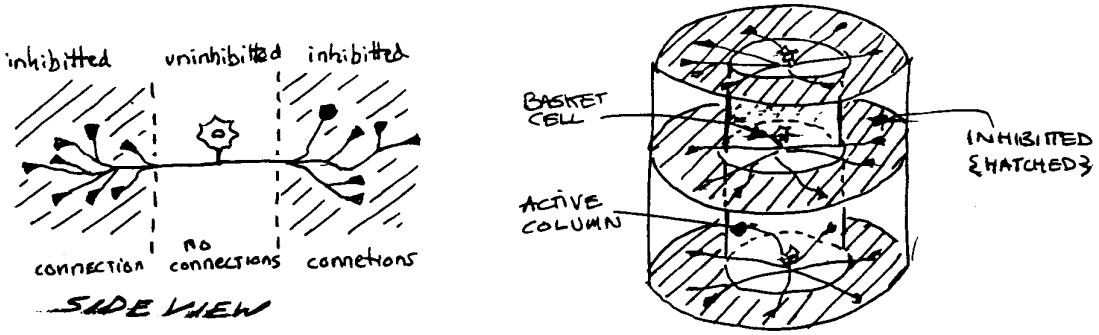


Figure 4.2 Connection doughnut formed by basket type cell.

When considering the degree of terminal branching in Figure 4.1, it may appear that the number of possible interconnections is beyond the scope of computational solvability. However, the application of the two rules described above reduces the problem complexity significantly. The first rule specifies that the terminals projecting from cells 5-17 are in fact axons. This rule has two implications. First, it dictates that any excitation or inhibition of these nonpyramidal cells can only be produced by axons from other cells that project to the immediate vicinity or the surface of the cell body. In other words, nonpyramidal cells grow connections in only one direction; from the first cell's axon hillock to the body of the second cell. Second, it dictates that the only cells that can provide a nonpyramidal cell with stimulation are those that can project axons into the stratum of that nonpyramidal cell. For example, to determine if cell A will form a connection with cell B, where cell A has axial projections in several strata and cell B is nonpyramidal, then one only need determine if cell A has projections in the same stratum where cell B occurs.

The second rule greatly reduces the complexity of determining the interconnections of pyramidal and other dendrite-bearing cells. Since an inhibitory neuron can only form an axosomatic synapse (the axon forms a connection with the cell body), the dendritic spines can be ignored when evaluating the formation of interconnections between inhibitory and pyramidal and/or dendritic cells. The second rule also indicates that only excitatory cells will form synapses on the dendrites. Since an afferent is actually the output of an excitatory cell located in a different cortex, the afferents will form axodendritic synapses. While these two rules greatly reduce the computational complexity of interconnection evaluation, the excitatory-to-pyramidal cell problem remains fairly complex. When considering this last subclass of interconnection types, one must evaluate all strata where both cells have common axons and dendrites; in some cases, this could be all six strata. If one neglects the neural science data presented in Chapter 2, they might create a series of assumptions to eliminate the connections that two cells form in multiple strata; however, the effects of passive transmission clearly indicate that all of the synapses will have unique signals, and therefore must be included.

The following tables contain the actual data that was used in this example to run the simulation, and were collected by measuring the lengths of the terminals appearing in Figure 4.1. Again, Figure 4.1is being used as a data source for clarity of the explanation, but any convenient source of cellular data may be used. Note that all measurements are presented in Unit-Cell-Lengths (1 UCL = 10μm). The minimum dendrite lengths are not given as all values for all cells are set at zero.

Table 4.7 Dendrite Maximum Length (UCL)

Cell Number	Symbol	Stratum 1	Stratum 2	Stratum 3	Stratum 4	Stratum 5	Stratum 6
1		0	0	0	0.5	3	4
2		1.5	0.5	0.9	3.2	6.5	6
3		1.5	3	2	0	0	0
4		1.5	1.5	3	5	0	0
5		1.5	0.5	0	0	0	0
6		0	0	0	1	0	0
7		0	0	0	0	2	2
8		0	0	1	3	1	0
9		0	0	1	0	0	0
10		0	0	1	0	0	0
11		0	1	0	0	0	0
12		0	0	1	0	0	0
13		0	0	0	0	1	0
14		0	0	1	0	0	0
15		0	0	0	0	1	0
16		0	0	1	0	0	0
17		0	0	0	1	0	0
18		0	0	0	0	0	0
19		0	0	0	0	0	0

Table 4.8 Axon Maximum Length (UCL)

Cell Number	Symbol	Stratum 1	Stratum 2	Stratum 3	Stratum 4	Stratum 5	Stratum 6
1		1	1	1	1	3	4
2		1	1	1	1	3	5
3		1	1	1	1	1	1
4		1	1	1	1	3	3
5		2	1	2	0	0	0
6		0	2	0.5	2	0.5	0
7		2.3	0	0	0	0	0
8		2.2	0	0	0	0	0
9		0	1	5	0	0	0
10		0	1	2	0	2	0
11		3.5	4	3.5	0	0	0
12		0	5	9	5	0	0
13		0	0	0	5	9	5
14		0	2	5	0	0	0
15		0	0	0	0	5	2
16		0.5	3	2.5	2	2	0.5
17		0	0	3	3.5	3	0
18		0	0	0	2	0	0
19		2	0	0	0	0	0

Table 4.9 Minimum Axon Lengths (UCL)

Cell Number	Symbol	Stratum 1	Stratum 2	Stratum 3	Stratum 4	Stratum 5	Stratum 6
1		0.0	0.0	0.0	0.0	0.0	0.0
2		0.0	0.0	0.0	0.0	0.0	0.0
3		0.0	0.0	0.0	0.0	0.0	0.0
4		0.0	0.0	0.0	0.0	0.0	0.0
5		0.0	0.0	0.0	0.0	0.0	0.0
6		0.0	0.0	0.0	0.0	0.0	0.0
7		0.0	0.0	0.0	0.0	0.0	0.0
8		0.0	0.0	0.0	0.0	0.0	0.0
9		0.0	0.0	0.0	0.0	0.0	0.0
10		0.0	0.0	0.0	0.0	0.0	0.0
11		0.3	0.4	0.3	0.0	0.0	0.0
12		0.0	0.5	0.9	0.5	0.0	0.0
13		0.0	0.0	0.0	0.5	0.9	0.5
14		0.0	0.0	0.0	0.0	0.0	0.0
15		0.0	0.0	0.0	0.0	0.0	0.0
16		0.0	0.0	0.0	0.0	0.0	0.0
17		0.0	0.0	0.0	0.0	0.0	0.0
18		0.0	0.0	0.0	0.0	0.0	0.0
19		0.0	0.0	0.0	0.0	0.0	0.0

4.1.4.3 Interconnection Density

The interconnection density, another parameter defining the structure of the neural network, is not well documented. The interconnection density refers to the fraction of cells within a cell's connection radius that actually form synapses with the projecting neuron. The significance of the interconnection density becomes apparent when considering the difference between the structures of type 12 and type 17 neurons. Type 12 neurons have few, but fairly long axons, whereas type 17 neurons have numerous short terminals that would appear capable of connecting to every single cell within its connection radius. In contrast, the few long axons of type 12 cells are only capable of connecting to a few of the cells within its connection radius. It is logical to assume that the interconnection density is a crucial parameter for preventing excessive short-range interconnections which may otherwise inhibit the formation and separation of neural pathways. Collecting values for the interconnection density is fairly time consuming, especially considering that each cell type has the potential of having a unique density for each cell type that lies within its connection radius. Even if a connection is not formed between two cell types, a density value of zero must be recorded to allow genetic mutation and evolution to occur, as is discussed later. In the case of the dendrite-bearing cells, it is possible for two cells to have multiple densities corresponding to the potentials of forming synapses in the strata to which they project connecting terminals.

The simplest case of interconnection density is that of fully interconnected, in which case $\rho_{AB} = 1$. The significance of a fully interconnected network is that all cell parameters reduce to secondary status, thus the cells are fully defined after the cell types have been defined. The fully interconnection assumption yields an association matrix **A** for each cell. In the example problems there are 19 association matrices listed at the end of this paper. Note that a value of 1 appears in all the locations where a synapse may possibly exist. The convention of using a right subscript to denote the cell type with which a particular association matrix is affiliated had been adopted. For example $\mathbf{A_2}$, seen in Table 4.10 below is the association matrix of cell type 2. The rest of the association matrices are listed in the appendix.

Table 4.10 A_2 - Association Matrix of Cell Type 2

		Cell Type																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
																				
	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0	0	1	0	0	1
r	**2**	1	1	1	1	1	1	0	0	1	1	1	1	0	1	0	1	0	0	0
e	**3**	1	1	1	1	1	1	0	0	1	1	1	1	0	1	0	1	1	0	0
y	**4**	1	1	1	1	0	1	0	0	0	0	0	1	1	0	0	1	1	1	0
a	**5**	1	1	1	1	0	1	0	0	0	1	0	0	1	0	1	1	1	0	0
L	**6**	1	1	1	1	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

The next level of complexity assumes that the density values are not simple binary 0, 1 values, but are continuous throughout that range. In this case a single density value ρ_{AB} is used to describe the interconnection density between cells A and B for all strata. This uniform density assumption is implemented by establishing a matrix ρ comprised of the various interconnection densities (ρ_{AB}). The rows of the uniform density matrix ρ constitute the interconnection density vectors of each cell, which, when multiplied by the appropriate cell's association matrix, yield a cell's uniform density association matrix denoted by the right superscript *uf*. The right subscript denotes the cell to which the matrix belongs.

$$\rho = \text{uniform density matrix}$$

$$\rho_1 = \{\rho_{11}, \rho_{12}, \rho_{13}, \ldots \rho_{1n}\}$$

$$\mathbf{A}_1^{uf} = \mathbf{A}_1 * \rho_1$$

A further degree of specificity is possible by employing a unique density assumption in which cells have unique densities between all cell types on all strata. The value of the unique density between cells A and B is given by :

$$\rho_{AB\text{-}1} = \text{specific interconnection density}$$

where the -1 represents the stratum in which cell types A and B form a synapse. The unique density assumption is implemented by assigning unique values to all of the elements of interconnection matrix A, to yield A^{uq}. Mathematically, the unique density assumption allows the conditions where:

$$\rho_{AB} \neq \rho_{AC}$$

$\rho_{AB\text{-}n} \neq \rho_{AB\text{-}m}$

4.1.5 Summary

A set of values has been collected for the cellular and cortical properties used in the cell growth simulation program. The neurons are classified by stratum, excitatory versus inhibitory status, and shape. The primary and secondary cell parameters have been identified, and the use of the primary parameters in the selection or inheritance of the secondary parameters has been discussed. Cell types have been associated with given strata, and the relative volume fractions have been identified. Some of the primary differences among the cell types were shown to be the size and number of axons and dendrites. Neurons with long range, short range, and even *doughnut* like connection patterns have been identified, and their role in cortical organization discussed. An interconnection matrix has been established, providing the foundation for the interconnection density parameter which describes the fraction of neurons that form synapses with a particular type of neuron within its interconnection radius. The next section converts the parameters into an artificial chromosome.

4.2 Converting Parameters into Genes

The cell growth simulation program is based on von Neumann's automata, and as such, requires an artificial chromosome that contains the complete instruction set for the cell. This section describes the conversion of parameters into a chromosome. This section establishes the significance of genes and describes a type of robustness and fragility that must be captured in the model to accurately mimic the biological system. This section also includes an important comparison between the genetic algorithm used in this study versus traditional genetic algorithms. This section also describes biological genes with respect to binary logic, and establishes LRP values which are analogous to activation thresholds and production rates. A worked example of converting a cellular parameter into a chromosomal representation is also included.

4.2.1 Gene Parameters

Converting parameters into genes requires the knowledge of the interrelationship between genes and the mechanisms whereby genes are expressed. To correctly model gene based growth it is first necessary to model a mechanism that produces a desired structure or function. Then it is necessary to determine what value a gene must possess such that its expression in each cell will collectively produce the overall network structure or function. From this it should be obvious that the conversion process does not follow direct or linear relationships. Before this process can begin it is necessary to identify the mechanisms to be incorporated into the model. Then, the effect of the gene on the mechanism must be characterized so that a relationship between the parameter, the gene, and the final structure produced by the mechanism can be established. Once such a relationship has been established, the gene may be assigned a value that is characteristic of the parameter. The first stage in this conversion process is to establish the engineering significance of a gene in terms of the amount and type of information it contains.

4.2.2 Significance of Genes and Genetic Algorithms

The conversion of a desired set of parameters into an artificial chromosome is a question of establishing the significance of the genes. The complexity of this problem is compounded by the fact that chromosomes exist within a system, and only within the system are their full meaning and significance observed. In the strictest sense, a gene represents the blueprint for the production of a protein. The protein can have a variety of functions, as discussed in Chapter 2, such as providing structure, serving as a messenger, or becoming an enzyme to produce another type of molecule. The placement of a gene within a chromosome also represents a regulatory process. Through mechanisms that are not well understood, the presence or absence of certain trigger chemicals in specific concentrations can initiate or deactivate protein production. This implies that either the structure of the gene or the placement of the gene within the chromatin, (or decondensed chromosome), affects the responsiveness of a gene to chemical signals. In layman's terms, this means that the gene also represents that quantity of the substance produced as well as serving as the blueprint for its production.

When attempting to produce a genetic algorithm, it is also important to represent the fragility and robustness of the chromosome. Chromosomes are robust in that several

different alleles can produce different, but functionally similar proteins. Chromosomes are also robust in that genes can be moved across tremendous distances and still be useful. On the other hand, chromosomes are fragile in the sense that the absence of a single gene can be fatal, near fatal, or detrimental to the survival of an organism. Such is the case with diabetes and many other genetic disorders. Also, chromosomes are especially fragile in the sense that the loss of a single nucleotide can render an entire gene useless.

In order to capture all of this behavior in a genetic algorithm, it is necessary to recreate a standard format analogous to DNA. Traditional genetic algorithms sought to replicate the appearance of the nucleotide base pairs with a binary string. Although this approach is successful in limited applications, it has a severe limitation in that certain genetic operations during artificial evolution may produce illogical or impossible offspring. These offspring are called fatal mutations in a misguided attempt to make the overall system appear consistent with true genetics. Traditional genetic algorithms are not consistent with true genetics for two reasons. First, genetic algorithms assign definite significance to the linear position of the gene on the chromosome. If the gene is mutated in such a way where is it no longer in the position that the algorithm would look for it, then an inconsistent or illogical offspring is produced. This is not the case in true genetics where genes can be quite mobile. The assignment of linear position also dictates that the genes must be a constant size, because a change in binary string length can potentially disrupt the transcription logic for the rest of the chromosome. In some cases, however, this process is used as a form of mutation. True genes are of variable length, having a start and a stop sequence to mark their beginning and end. The start and stop sequences allow alleles of the same gene to be of different length. Perhaps the most serious shortcoming of traditional genetic algorithms is that they directly code their parameters into the binary string, a condition that may produce multiple copies of the same gene appearing at different axial locations on the artificial chromosome after breeding. This deficiency is acutely pronounced when attempting to perform machine grouping problems where a machine number is encoded as a gene and the grouping sequence is read linearly from the chromosome. During crossing over, it is possible to produce an offspring that has the same machine appearing multiple times, a physical impossibility. In contrast, biological systems commonly have multiple gene copies.

4.2.3 Developing a Differential Gene Expression Based Algorithm

The discussion of genetic algorithms and their biological equivalents in the previous section has identified several shortcomings which can potentially be eliminated by applying

the following assumptions. First, relax the constraint that the artificial chromosome be represented by a binary string. This traditional constraint is purely cosmetic and does not substantially facilitate the mutation process or calculations. Second, assume that the gene carries both a functional and structural meaning, as well as the production quantity. Third, assume that all values to be represented in the chromosome must be in a standard format such that crossing over these values will not yield an illogical offspring. Lastly, it must be assumed that the chromosome will only have true significance when it is integrated within the simulation program.

Referring to Chapter 2, it should be clear that all decisions made within the cell are based on concentrations of the ligand receptor pairs, LRPs. It was also discussed in Chapter 2 that the transcription of one gene could possibly alter the transcription of another. From this, the LRP Standard Format has been produced, in which all genes simply carry a value corresponding to the responsiveness or rate of production of the LRP; thus, the encoded value corresponds to the production rate. The actual meaning of the gene is assigned as auxiliary data. In a general format, the auxiliary data can be stored in the file header, although it would probably be larger than the actual genetic information. Using the general format, the header would define the function associated with the gene. For example, if the simulation program determines that gene A should be expressed, then the program would read the auxiliary information in the file header to determine the effect of A. Examples of auxiliary data are: number of divisions, release ligand, produced receptors, etc. Note that although the gene values would undergo a crossing over operation, the header file would remain constant.

While the use of such a general format is an elegant solution, it requires the development of a smart, generalized simulation program. An alternative approach, which was used to develop the growth simulation program, is to incorporate the auxiliary data directly into the program as hard code. While at first this may appear as a diversion from the biological model, it is in fact more consistent. The transcription and translation of a gene yields a protein. If this protein is removed from the cell, it has no meaning. In the cell, there are countless mechanisms that respond to the proteins produced by the translation process yielding the structure or behavior that is observed. Using standard LRP format, the values for the chromosome can be varied to a great extent, as demonstrated in the simulation program, without the loss of logic.

At first thought these assumptions seem to complicate the development of the genetic algorithm, but in fact they greatly reduce the logical complexity of the process. Relaxing the cosmetic constraint of using a binary string allows the DNA to be assembled with standard

integer or real values. To increase computational speed, the convention of representing everything with integer values was adopted. The ability to store genes as integers allows much easier revision and interpretation from a human factors point of view, and also allows storage in the form of a text file.

In Chapter 3 a theoretical basis was established for producing a genetic algorithm based on differential gene expression. Since the genetic algorithm used in this dissertation evolves based on a differential gene expression mechanism instead of a mutagenic process, the possibility of an illogical mutation is greatly reduced. The key element for this new technique is establishing double redundancy in the amount of information stored in the organisms. Each organism has two complete sets of chromosomes and randomly contributes half to its offspring. Through this mechanism, different organisms are produced by permutating the allele combinations. If mutation occurs in this new format, the result would be a new allele, which is a new gene value, that could be readily interpreted by the hard code.

When considering the new system as a whole, three things become apparent. First, since the genes are assigned to a text file in a fixed sequence, the meaning of each location in the text file has a permanent significance to the growth simulation program. The values of the genes can be interchanged freely without loss of logic. The appearance of a zero in any given gene simply means that the gene will not be expressed. It is demonstrated that zero values can produce viable offspring with my technique. The ability to understand a zero value without producing an illogical or fatal mutation is a clear distinction from traditional genetic algorithms. Second, the separation of the gene value from the auxiliary data means that totally new genes can only be added by the user. Since zero values can be interpreted, genes can essentially be turned off, thereby disabling unwanted mechanisms without incurring reprogramming costs. In the course of development of an algorithm for a particular application, it may be found that additional mechanisms need to be incorporated into the model. In this case, new genes can simply be added to the end of the file since the sequence of the genes in the chromosome has no intrinsic meaning. Only the pointer address of the gene in reference to the auxiliary data has meaning to the simulation program. Third, the differential gene expression model does not allow the same actual physical reordering of the gene sequence as the biological model, but it does allow the same logical transfer of information that occurs during crossing over. Where cells actually cut their DNA and splice segments together to form new chromosomes during reproduction, this new genetic algorithm reads data from the parental text files, randomly selecting one value at a given pointer address, and writes that to the offspring text file.

4.2.4 LRP Convention

There are three ways that an LRP can cause a differential or selective effect in biological organisms. The first is to assign each ligand a unit value. With this convention, the greater the quantity of ligands produced, the stronger the response of the cell. The same convention can be applied to receptors where cells with more receptors are more responsive to a particular ligand. A second convention is to assume that different ligands can have varying degrees of significance, and under the same convention, certain receptors would be more responsive than others given the same ligand concentration. The last convention is a combination of the first two, where both the number and sensitivity of the LRPs dictate the responsiveness of the cell. Since all three have the same mathematical result, and the first convention is computationally less intensive, it was selected as the method of choice in this dissertation. To summarize, the number of ligands and receptors is proportional to the responsiveness of the cell, and the value of the gene is this number of the particular LRPs.

4.2.5 Worked Example of Parameter to LRP Conversion

Consider the case of deciding that a stratum thickness would be 3.4. In this example, which corresponds to the gastrula stage, each of the cells in the particular stratum must produce a family of size 3.4 members. Since a cell can only divide into discrete elements, it becomes clear that to produce an average of 3.4, some cells will produce 2 daughters, (2 + 1 mother = 3 members) and others 3 daughters. The possibilities of families with less than 3 members and greater than 4 members are excluded in order to reduce the problem complexity. From the given data, it is known that:

$$60\% \rightarrow 3 \text{ members}$$

$$40\% \rightarrow 4 \text{ members}$$

giving an overall average of:

$$0.6(3) + 0.4(4) = 1.8 + 1.6 = 3.4$$

To implement this value, two genes are created, A and B. If A is expressed, then the mother cell will divide twice to produce two daughters or a family with 3 members; and if B is expressed, 3 divisions of the mother cell will occur to produce a family size of 4. The ratio of the values of A to B gives the probability of the particular gene being expressed. To write an artificial chromosome, the auxiliary data is separated and placed in the file header, and the gene values are listed in the body of the file.

4.2.5.1.1.1.1.1.1 Artificial Chromosome

A	Make 3	Auxiliary Data
B	Make 4	
A	60	LRP values
B	40	

The auxiliary data in this example contains the command "make" and the modifier "3" giving the complete expression: *make three cells from one.* The 0.6 probability value of the expression of gene A is obtained from the fraction of the gene values as seen below:

$$0.6 = 60 \Big/ (60 + 40)$$

4.2.6 Summary

The significance of the genes has been established. It has been shown that the gene not only describes a function, but the amount that the function is expressed. It has been shown that the traditional genetic algorithm relies on a mutagenic process for evolution. In contrast, nature relies on a differential gene expression process which has been modeled. This process serves as the foundation for the advanced genetic algorithm used in this dissertation. The advanced genetic algorithm greatly reduces the possibility of an illogical offspring being produced. The LRP value has been introduced and a worked example of converting a parameter into an artificial chromosome was shown. In the next section, the design of the complete chromosome will be discussed.

4.3 Generating the Artificial Chromosome

The generation of the artificial chromosome is a process dependent on the development of the simulation program, meaning that one cannot effectively be done without the other. The process discussed here involves establishing a very primitive piece of code that when interpreted, yields a final structure. The first stage of this process is establishing a sequence in which the artificial cells develop. This sequence serves as the foundation for both the chromosome and the program and has two basic requirements. The first requirement is that the sequence be comprised of independent steps. An independent step is one that requires only one gene to be expressed to complete the step. In true biological systems, the likelihood of only one gene being expressed at a time is quite low. The biological system is parallel in every sense of the word in that each cell operates independently and within each

cell, organelles operate independently, allowing the simultaneous transcription of many genes. Ideally, one would like to be able to replicate the parallel capabilities of the biological system; however, the limitations of traditional computers prevent the full modeling of such a parallel system as it would require memory resources beyond the limits of current hardware. To assume that a step is independent, it must be assumed that when the gene is read, it does not affect any other genes during the transcription process. This is not to say that the expression of one gene cannot control another. Rather, it is assumed that the effect of a gene is only felt by other genes at the completion of its transcription. While the independence assumption is partially consistent with the biological model, it does not include conditions that require more than one gene to fully define the trait, such as epistasis, independent assortment, or incomplete dominance. When computer resources become available, the independence constraint can be removed or relaxed to allow a more comprehensive model. Despite the apparent limitations of the independence constraint, it actually facilitates the design of the system by reducing the complexity of the gene selection process, and for all of the procedures encountered in this study, none required multiple gene expression.

The second requirement is that the sequence contains all of the steps for the development of every cell. Certain steps can be skipped by cells, but no cell can contain a step outside of the master sequence. This master sequence requirement arises from the fact that the growth simulation program must treat each cell essentially the same; therefore, the program must progress down a common path for all cells. A common path is defined as a linear path that does not branch. While it is possible to develop a system without this master sequence assumption, the complexity of the growth simulation program increases dramatically. After due consideration, it was determined that the master sequence assumption would not prevent or inhibit the development of any of the traits of interest to this project, while greatly simplifying the development process. The master sequence assumption is partially characteristic of the biological model. It is known that all cells follow the same initial steps, but soon thereafter follow their own separate paths. Whether the development of true cells actually branches or follows a master sequence is not known. Since no mention of a master gene sequence occurs in the biological literature, it is assumed that the master gene sequence is an artifact of this technique. This master sequence is not a hard requirement of this method and was only used in this study because it was benign to the processes under investigation. In the case of an application that would be more easily described in a branching type sequence, this assumption can readily be relaxed.

4.3.1 The Master Sequence

A bi-directional approach was used to develop the master sequence since the beginning and final products were known at the onset. It was known that the network initiates as a single cell that undergoes a five-fold split, and each of the resulting five cells undergo a series of divisions to produce a five stratum thick square lattice. This lattice then undergoes a series of transformations to produce a structure such that each stratum has a morphology analogous to neocortex tissue. To generate the master sequence to be used to design the growth simulation program, refer to Chapter 3 which contains an outline of the steps of development. In Chapter 3, it was stated that development occurs through several stages including the zygotic, blastular, gastrular, embryonic, and fetal stages. These stages have been interpreted here in the context of developing a program. When evaluating the steps outlined in Chapter 3, one goal is to identify or classify processes in such a way that a small set of fundamental mechanisms can be used to fully define the structure. While the majority of the discussion to this point has focused on keeping the technique philosophically analogous to the true biological process, at this stage it is necessary to make some adaptations to the underlying biological theories in order to allow computer implementation. The adaptations or assumptions made to generate the computer program are by no means to be interpreted as absolute constraints on the theory in general, but rather temporary assumptions that were implemented for this specific example problem based on the limitations current serial processing computers.

4.3.1.1 Zygotic Stage

During zygotic development the artificial chromosome, which has double redundancy, is divided between expressed genes t and genes that are held idle until reproduction. The zygotic stage is implemented by the breeding program directly after the generation of the offspring chromosome from the parents. Thus the output of the breeding program is a text file that contains two complete sets of genes. The first set of genes defines the phenotype of the developing network and is separated and listed first. The second set of redundant genes is listed in the same file after the phenogenes to avoid the accidental loss of genetic material. In summary, the zygotic stage occurs as the last step of the breeding program.

4.3.1.2 Blastula and Gastrula Stages

The blastula and gastrula stages are discussed jointly as together they define the hardware resources needed to run the simulation program. The blastula stage involves the transformation of the single cell zygote into the multicellular unit-neural-grid. This unit-neural-grid is comprised of predominantly undifferentiated cells and has the same number of cells in each stratum. The growth theory put forth in this document states that as the cells divide, a pointer of undefined specification keeps track of each cell's lineage. This pointer is used to specify which subset of genes can manifest themselves in the particular cell during later stages of development. Since the simplified growth theory of this model directs cell development in the Euclidean fashion, the cell's (x, y, z) coordinates can be used for this pointer value.

During the gastrula stage the strata undergo a series of divisions to allow each stratum to have a unique unit cell count. The stratum thickness, which is based on the unit cell count as previously described, is collected from anatomical data, (Chapter 2), and is proportional to the number of cells per stratum. When reviewing the blastula and gastrula stages, the systematic divisions of cells are seen from which genes can be generated. The first three genes correspond to the first three divisions in the blastula stage.

In this example the gene values are set at MAXINT, the largest value that a 16 bit integer can obtain. This value denotes that no decision is necessary and that the gene in question must be expressed. The auxiliary information associated with these first three genes indicates that all three steps require the cells to divide. The modifiers, the values following *Divide*, are obtained from the number of strata and the x and y dimensions of the unit-neural-grid. Note that there are actually six strata, but only five develop since stratum 1 does not actually contain any cell bodies, and only serves as a region for interconnections.

4.3.1.2.1.1.1.1.1 Artificial Chromosome

A	Divide 5	*Auxiliary Data*
B	Divide M	
C	Divide N	
A	MAXINT	*Ligand Value*
B	MAXINT	
C	MAXINT	

It was deemed unnecessary to simulate these three steps since all three steps definitely occur, since they have the highest possible probability of occurring, as indicated by the MAXINT value. Instead, the values corresponding to the number of strata and the size of

the unit-neural-grid were used directly in order to reserve the memory resources of the computer used to run the program.

During the gastrula stage, the size of the unit-neural-grid expands based on genes; however, unlike the blastula stage, considerable variability must be simulated to fully define the network. While one might prefer to allow the genes to acquire any value, it is necessary to implement a few physical constraints to prevent memory mismanagement errors from occurring during the operation of the program. The first constraint is on the unit cell count whose value is set at 16. In other words, the greatest number of divisions that any one cell can have during the gastrula stage is 15, producing a total of 16 cells. This limit not only controls the number of divisions in a given stratum, but sets the maximum ratio between stratum thickness at 1:16. Having employed this limit, the variable arrays used in the program can be fully defined as (M*4) x (N*4).

Next, the gastrular genes are designed. When the program begins, the unit-neural-grid is defined in memory. The program immediately begins to convert the raw data concerning stratum thickness into an LRP format. In all, 10 LRPs are used; two for each stratum. Using the method described in Section 4.2.5, the following artificial chromosome is generated.

In the environment of the computer it is necessary to maintain a proportionality between strata of different sizes. The overall picture of the gastrula stage is that each cell referred to as a mother produces a certain number of daughters. All of the daughters from the same mother are collectively termed a family. At the beginning of the gastrula stage all of the strata have the same number of elements, and each cell in each stratum becomes the mother of a family. If one considers the uniform grid produced at the end of the blastula stage in the context of these mother cells, it becomes apparent that at a given (x, y) lattice point, five mother cells are aligned in a vertical column. It is necessary to ensure, regardless of the size of the stratum, that the families of the mothers that constituted the initial vertical column are still aligned at the end of the gastrula stage.

4.3.1.2.1.1.1.1.2 Artificial Chromosome

A	Divide 1	*Auxiliary Information*
B	Divide 2	
C	Divide 2	
D	Divide 3	
E	Divide 10	
F	Divide 11	
G	Divide 3	
H	Divide 4	
I	Divide 7	
J	Divide 8	
K	Divide 6	
L	Divide 7	
A	25	*LRP Values*
B	75	
C	20	
D	80	
E	50	
F	50	
G	60	
H	40	
I	25	
J	75	
K	80	
L	20	

To ensure inter-stratum family alignment given strata of different thickness and families of different size, it is necessary to employ the smallest square standard. This standard states that a family is allocated a square, the size of which is determined by the number of elements in the family. For a given stratum there exists only two possible family sizes. The smallest square that can accommodate the larger of the two family sizes is allocated for every family in that stratum. It is necessary to use a square for each family to ensure that the stratum is being scaled proportionally. The largest possible square would be 4x4, based on the memory resource limit of a 1:16 growth ratio.

The next stage of the process of ensuring family alignment is to scale the interneural distance to be inversely proportional to the family size. Since all strata are being represented by a two-dimensional array in the computer and all strata must maintain alignment, then strata with more cells are treated as if they are packed closer together. Consider two strata, one with a family square of 9 and the other 16. After the gastrula stage the two strata are assigned grid size scaling parameters. The grid scaling parameter is equal to the inverse of

the square root of the family size. To identify the elements at grid location (3,5), one would multiply the coordinates sought by the scaling factor to obtain (1, 1.67), (0.75, 1.25) for the 9 and 16 family size strata respectively.

Employing the smallest square method dictates that vacant lattice sights will occur in those families whose membership is less than the allocated square. In the case of a stratum size of 10.3, there are families with 5 and 6 vacant sights. Since the decision making process of this method is based on the generation of concentration gradients, and those gradients are dependent on the proximity of cells to each other, it is necessary to ensure that the central elements of the families are occupied and that vacancies only occur in the distal sights of the family square. The assignment of vacancies to distal sights is performed by a depleting population probability algorithm. It is necessary to use a depleting probability algorithm since the program operates in a serial fashion and each distal sight has equal probability of being vacant.

4.3.1.3 The Embryonic Stage

At the completion of the gastrula stage the total cell count of the complete network has been established; however, with the exception of the pointer values associated with the cell lineage, the cell types have not yet been defined. In a similar fashion to the gastrula stage, the differentiation that occurs within each stratum is treated as an independent event. When the program is executing, it differentiates one stratum at a time and carries over no information between strata.

Recall from Section 4.1.3 that cells can be classified by stratum and type. To be consistent with the methodology previously described, it is necessary to convert the cell fraction data into standard LRP format. During the embryonic stage another stratum of complexity over the gastrula stage is observed in that the number of cells per stratum is variable and unique. Unlike the case of the gastrula stage where a decision had to be made between one of two possibilities, in the embryonic stage, many possible selections exist. Consider the example of stratum 3, which has six possible cell types in it. Each cell must decide independently one of the six possibilities. Collectively the stratum should have approximately the same fractions of cell types as described in the initial data.

Following the same basic method, a gene is generated for each of the cell types, the value of which is proportional to the probability of there occurance in that stratum. The auxiliary data determines which cell type is expressed. From this the following genes corresponding to the embryonic stage of stratum 3 are obtained.

Artificial Chromosome

AA	Become 4	*Auxiliary Data*
BB	Become 9	
CC	Become 10	
DD	Become 12	
EE	Become 14	
FF	Become 16	
AA	179	*LRP Values*
BB	108	
CC	108	
DD	162	
EE	162	
FF	262	

Because the number of cells in a given stratum is unknown at the time that the program generates the LRP formatted chromosome, it is necessary to include information associated with the number of cells in the stratum. In essence information is encoded into the chromosome that allows the treatment of genes as if they were families or subgroups. Using the "family" gene it is possible to associate several genes with a single function, and furthermore allow the number of genes to be controlled. This introduces the ability to make more complex decisions to the model. A set of embryonic genes for each stratum is produced in a similar fashion.

4.3.1.4 Fetal Stage

During the fetal stage, the cells grow their terminals and form interconnections. As mentioned in the previous section, this step can be greatly simplified by using the assumption of total interconnection which states that a cell will form connections with all of the cells permitted by its association matrix that lie within the range of its terminal limits. To determine the interconnections under such a scenario, a simple search routine can be employed that checks the previously mentioned conditions. However, the growth of the synapses is greatly complicated by the introduction of interconnection density.

There are several cases, or classes, of interconnection densities, each of which must be treated slightly differently. The first condition to be considered is that of an interconnection density of 50%. Consider cells *A* and *B* in stratum *N*, where *A* is projecting an axon towards *B*. In this a scenario, the *B* cells secrete ligands and the A cells have the corresponding receptors. To produce an interconnection density of 50%, two LRPs must be assigned to the single step. Each *B* cell secretes one of the two ligands and each of the A

cells have only one of the two possible receptors. In essence, to allow an interconnection density of 50% the cells must further differentiate themselves. Therefore using a similar process as described, a set of genes is generated where the auxiliary information contains the ligand or receptor code and the value is proportional to the interconnection density. The family of this gene set contains both the genes that differentiated the *A* and the *B* cells.

The same basic strategy is employed for interconnection densities less than 50%. For example, if an interconnection density of 33% is sought then the *family-size* contains three genes and each cell associated with that family makes a 1 out of 3 decision. The growth simulation program written for this dissertation limit has a software limit of 6 ligands or a minimum interconnection density of 1/6 or 16.7%. In the case of an interconnection density value lying between two whole fractions, such as 30%, which lies between 1/4 and 1/3, is desired then the family size is expanded to 9 elements. The first two elements determine how many cells will undergo 1/3 and 1/4 differentiations respectively. This condition is called the *complex single*, referring to multiple differentiations where a single gene out of a number of possibilities is selected. Complex single differentiation is basically the combination of the mechanisms of gastrular differentiation and embryonic differentiation performed in one step. In this example the family has three subgroups. The first subgroup, called the membership subgroup, contains two genes which determine the membership of the other two subgroups. The remaining two subgroups, called size subgroups, have three and four genes, respectively. It is important to not that a receptory cell only have membership to one size subgroup, whereas a secretory cell has membership to both size subgroups.

In the case where an interconnection density greater than 50% is desired, further differentiation must also occur; however, unlike the previous example where one out of a given subgroup is selected, all but one will now be selected. Consider the case of an interconnection density of 75%. This example requires that the family contain four genes, and that each cell select three of the four. In a similar fashion to the complex single discussed above, if an interconnection density is sought that lies between the multiples of two whole fractions, then the condition of *complex multiple* exists. The complex multiple condition requires three subgroups; one defines membership and the other tow define the size subgroups. For the example of an interconnection density of 70% which lies between 2/3 and 3/4, the two genes in the membership subgroup decide how many cells would belong to the 2/3 and 3/4 size subgroups respectively. Once the cells have been divided between A and B subtypes, each cell selects which gene of its subset it expresses. Continuing the example, a subtype *A* cell selects all but one of the genes associated with the 2/3 size-subgroup and a

type *B* cell selects all of the genes but one from the 3/4 size-subgroup. Again, receptory cells only have membership in only one of the two size subgroups, whereas secretory cells have membership to both size subgroups.

4.3.2 Summary

In this section the generation of an artificial chromosome has been demonstrated. It was shown that the development process follows distinct stages which collectively define the master sequence. All of the stages of the master sequence must be followed in the correct order to produce the correct final structure. The chromosome, which contains all of the genes, has been organized with respect to the master sequence. The master sequence must be followed exactly. Any deviation from the stages specified in the master sequence will not produce the correct final structure since the events that take place in any given stage are dependent on the structures produced in the previous stage(s). The stages of the master sequence correspond to the stages of development. It has been shown that the sequence that the gene are expressed can be deduced by observing the changes that occur during development. During each stage of development a unique set of events transpires. The final structure begins as a single cell or zygote. During the first or zygotic stage, the genes to be used throughout the rest of the development process are selected. The cell then divides to form the overall mass (blastula) which then differentiates to obtain specific physical dimensions or size (gastrula). During the embryonic stage the cell types are defined. The remainder of the interconnections are formed during the final or fetal stage. At the end of the simulated growth process a complete network has been produced with all traits, features and structures specified. In the following section the critical aspects of the program needed for computer implementation.

4.4 Implementations

This section describes the vital details of the model that have been implemented to produce the working program. These four key details are discussed in the first part of this section. The first key detail describes the differential gradient which essentially acts as the engine of the system. Secondly, it is shown that the cells respond to variations in control ligand concentrations. The collective production of the different control ligands by all of the cells in the network produce the concentration gradient. In order to produce a concentration gradient field it was necessary to introduce minute differences between the cells since a matrix consisting of identical cells would produce a homogeneous concentration field. These

minute differences are primarily based on nutrient availability. The third key detail describes the nutrient availablity which defines the growth rate and the rates of ligand production. The last section models the effect of distance on the concentration of the control ligands using a simple exponential function. The last part of this section gives a brief introduction to the program and contains plots of the cell growth simulation.

4.4.1 Generating The Synapses

During the fetal stage, interconnections are specified but not truly grown. The interconnections are specified because the secretory and receptory cells are assigned ligands and receptors that define which cells may interconnect within their terminal length limits. During program execution, the two cell types under consideration are loaded into memory and the appropriate family of genes are read by the simulation program, thereby differentiating the cells. Once this is completed, an interconnection list is generated from a simple search routine that identifies appropriate secretory cells that are in the terminal range of the receptory cells. Although a complete interconnection or synaptic list generated, this is not considered a growth process. If the terminals were actually grown, then each of the terminals would have to follow the chemical gradient of the specific ligand until it reaches the appropriate secretory cell. This process would involve numerous iterations and consume tremendous computer resources. Purists might argue that unless the synapses are grown, the network being defined has not undergone a process that is truly analogous to the biological counterpart. To this an engineer would retort that the network is fully defined once all the genes associated with interconnections have been expressed, and that the actual connecting process is merely a mechanical operation. While both positions are correct, the latter was adopted in the writing of this dissertation.

4.4.2 The Differential Gradient

The development of the simulation growth program is based on the assumption that cells secrete signals which communally these secretions produce a differential gradient. From this differential gradient, differentiation decisions are made. The chemical signal that is at a higher concentration than the other chemical signals, at a given cell location, initiates some mechanism whereby the transcription of a specific gene produces an observed behavior. The problem with this model is that if all the cells are essentially identical and all of the cells

contain the same genetic information, then no differential gradient is produced, but rather a homogeneous or uniform field is observed.

Consider first the blastula stage in which all of the cells in stratum 2 must decide between producing 2 or 3 offspring (see *Artificial Chromosome*, page 165). To enact such a decision, the cells must decide to express gene *C* or gene *D,* where genes *C* and *D* yield 2 and 3 divisions respectively. This decision is based on the cell's ability to communally generate a differential ligand field based on the ratios contained in the genetic code. At the beginning of the cycle, all the cells release 20 type *C* ligands and 80 type *D* ligands. No matter what equation is used to calculate the dissipation over distance, all cells will observe the same ratio of 20*C*:80*D*. Such a ratio would cause all of the cells to select gene *D* and therefore produce three offspring yielding a stratum thickness of 3, not the desired 2.8 unit-neural-lengths. To correct this problem one might consider incrementally activating the cells. Under this scenario, for each iteration the cells secrete an additional fraction of their total, until finally they reach the limit specified by the LRP value. The problem with this technique is that the gradient, although activated incrementally, the 20*C*:80*D* ratio is still produced.

Creating a differential concentration gradient from nearly uniform cells implies that very slight differences between cells produce a pronounced effect on their development. This deduction is consistent with the biological model which customarily identifies the effect of compounds whose concentrations are below measurable limits with today's equipment. Accepting the assumption that minute differences have a pronounced effect, and are responsible for the development of a differential gradient, requires modeling a source for these minute differences.

Since the DNA is assumed to be the same in every cell of an organism it is illogical to assume that there would be minute differences in the DNA accounting for the generation of the concentration gradient. Further, assume that cells with the same DNA, the same nutrient availability, and the same stimuli, develop in exactly the same way. This last statement indicates that the minute differences probably arise from variations in stimuli and/or nutrient availability. It is already known from the previous work in this study that the stimuli of neighboring cells plays a significant role in differentiation. However, the role of nutrient availability has not been discussed. If a cell is assumed to be a tiny machine operating as quickly as nutrients (fuel) can be supplied to it, then it is obvious that those cells receiving more nutrients develop at a more rapid rate compared to those that receive kess nutrients.

This phenomenon still does not address the problem of identical cells having very similar nutrient availabilities but developing at different rates. To accept the notion of similar

nutrient availabilities affecting the rate of development, it is necessary to accept that the diffusion process by which the nutrients are delivered to the cells is a process that is highly susceptible to concentration. The diffusion process is in fact driven by concentration differentials and one would expect it to be sensitive to minute differences in concentration. In summary, the diffusion process is sensitive to minute differences in the nutrient supply, and those differences produce the differential gradients given the non-linear dynamical nature of the simulation model.

4.4.2.1 Modeling the Rate to Maturity Based on Nutrient Availability

In the closed circulatory system common among mammals, very small blood vessels called capillaries deliver nutrients to the cells. A given capillary can supply sufficient nutrients to a certain number of cells. As the cluster of cells surrounding the capillary develops and grows, the cells on the periphery of the cluster become nutrient starved, at which time they begin secreting signals to initiate a new capillary to grow and increase the nutrient supply being delivered. From this it is concluded that there is a dependence on minute changes in nutrient levels and that the supply of nutrients, in this case the growth of capillaries, is a non-linear dynamical process that follows the same basic set rules used to simulate the growth of the neural network.

Ideally one might consider modeling not only the growth of the neurons, but also the growth of the capillaries to determine which cells are differentially preferred in terms of nutrient resources. However this level of detail in a simulation is unwarranted. The growth of capillaries is a non-linear dynamical system, not only dependent on growth stimuli from surrounding tissue, but also on the hydrodynamics of the colloid solution flowing through the vessels. The hydrodynamics is in turn linked to every other system in the body and is ultimately affected by the environment as well. A more reasonable approach is to assume that the overall trend is a random or near random distribution of nutrient supply across the developing neural tissue. Using this approach it is possible to assign a random number, termed the ***MATURE*** parameter, to each cell. ***MATURE*** is a uniform random integer, with a range of 0-100, that is used to determine the growth rate of a given cell. The cellular growth rate is then used to determine the rate of LRP production.

4.4.2.2 Modeling the Effects of the MATURE Parameter.

The ***MATURE*** parameter regulates the development of a differential chemical gradient in three ways. First the ***MATURE*** parameter is used to determine the rate at which the ligands

and receptors are produced once production begins. Second, the ***MATURE*** parameter determines how many iterations occur before the cell begins to produce ligands and receptors. Lastly, the combination of rate and starting time ultimately determine when a cell reaches maturity. Maturity is defined to occur when a given cell has produced the total number of receptors specified by the gene; thus, the ***MATURE*** parameter, by determining the rate and starting time of LRP production, ultimately determines the time to maturity.

4.4.2.3 Differential Gradient Based Gene Selection

To generate a differential gradient, the program first establishes the number of iterations, ***it_count***. The variable ***it_count*** is user specified and roughly defines how many iterations are necessary before all cells produce the LRPs. Values between 5 and 10 have been experimentally found to perform well. A value below 5 tends to disrupt the formation of patterns that have been found to be characteristic of this system, while a value above 10 has not been found to improve pattern formation. Assuming that ***it_count*** is set at 10, approximately 10% of the cells, those with a ***MATURE*** parameter greater than 90%, begin to produce the LRPs. The rate of LRP production, or the percentage of the total value produced per iteration, is proportional to the maturation value, thus the cells that begin producing the LRPs first, also produce them at the greatest rate, and subsequently reach maturity first. The equation which determines the rate of LRP production is given below:

$$\text{Growth Rate} = (A+(B * \textbf{\textit{MATURE}}/100))/100$$

The two terms A and B regulate production rates. A is used as a minimum value to ensure that at least A % of the total value is added each iteration. The maximum rate is given by A+B, which prevents cells from reaching maturity too rapidly. Experimentally, the values of 10 and 20 for A and B respectively, giving a minimum of 10% and a maximum of 30%, have been found to perform well.

Once a cell reaches maturity, it expresses the gene whose ligand concentration is the greatest at its location. Once this has occurred, it no longer secretes the LRP associated with its own type, but continues to secrete the other LRPs of its family. This has a tendency to bias the local concentrations away from its own kind, thereby decreasing the likelihood of a neighboring cell differentiating to the same type. During each iteration the concentration gradient is recalculated and the process continues until all cells differentiate.

4.4.2.4 Calculating a Differential Gradient

The calculation of the concentration gradient is a two step process. At the beginning of each iteration the number of ligands that each cell is secreting is calculated and stored in an array. These values are not changed until the beginning of the next iteration even if a cell differentiates during the current iteration. It is necessary to employ this strategy to prevent biasing that could occur as the program sequentially develops the cells. In the second step the program calculates the gradient that each cell is exposed to by calculating the contributions of the adjacent cells. The equation given below is used to approximate the dispersion associated with a molecule entering solution. The effect of this equation is to bias the decision making process of a given cell by making the contributions of adjacent cells more significant than those of more distant cells. Based on the gradient calculated in the second stage, a gene can be selected to be expressed; however, the expression of the gene is not manifested until the end of the stage, again to prevent biasing. The expression for calculating the magnitude of the concentration from cell B at the site of cell A is given below.

$$\text{Magnitude}_{(\text{B on A})} = [\ \text{B}\ ]*(1\text{-ABS})^{\text{length}} \qquad \text{Equation 4.1}$$

The absorption term ABS refers to the fraction of the initial concentration entering a cell that is absorbed or otherwise metabolized in that cell. Length is the distance between cells in unit-cell-lengths (UCL). Note that Equation 4.1 is a power function in the form of a^u, where a is a real number and u, also a real number, is the variable of the equation. Equation 4.1 can be converted to be an exponential equation by converting the base:

$$a^u = e^{u \ln a} \qquad \text{Equation 4.2}$$

Thus the new expression for magnitude is:

$$\text{Magnitude}_{(\text{B on A})} = e^{\text{length}*\ln(1\text{-ABS})} \qquad \text{Equation 4.3}$$

The smallest contribution that cell B can have on the gradient of A is given by:

$$\text{Min}\ [\text{Magnitude}_{(\text{B on A})}\ ] = e^{RAD*\ln(1\text{-ABS})} \qquad \text{Equation 4.4}$$

where *RAD* is a variable that defines the maximum range that the program uses for calculating the gradient.

4.4.3 Simulating Development

The simulation program developed in this study has three main functional elements. The first element scans in the cell parameters stored in text files and creates the LRP

formatted chromosome. The second element actually simulates growth based on the LRP coded genes. The stages of development represented in the second module are the gastrula, embryonic, and fetal stages. During each stage the program treats each stratum independently and determines the cellular differentiation through an iterative process whereby the concentration gradients are generated and used as the decision making criteria. The following tables demonstrate the development of the network showing each stage of the iteration and the resulting structure. Two examples are presented here. The first has a stratum thickness of 16 UCL which yields a network with no vacancies. The second example, which has a stratum thickness of 15.3, is shown to illustrate how the system handles non-integer thickness values. Each successive block represents the state of the developing network at the end of an iteration. Since the underlying mechanism responsible for the development of the network is identical for all strata, the development follows the same trends for all strata, therefore only stratum 3 is shown here.

The blastula stage is defined by the first six iterations and is responsible for determining the final size of the network. At the beginning the blastula stage, the cells are completely undifferentiated and are symbolized by a small black squares. As the blastula stage progresses, the undifferentiated cells form neural clusters denoted by circles, the size of which is specified by the inscribed number. After completion of the blastula stage, the network is expanded such that the neurons in the clusters are evenly spaced. In this example, a 16 fold increase occurs, easily observed by the 4-fold increase of the network's side dimensions. At the beginning of the embryonic stage all of the partially differentiated cells are shown as black circles with an inscribed 1. As the embryonic stage progresses, the cell types are selected. Stratum 2 has three cell types, represented by large black, gray, and white squares. Vacant sights, which appear in the second example, are symbolized by white squares with an inscribed "x".

Table 4.11 Reduced Symbol Key

Symbol	Definition	Symbol	Definition
▪	Undifferentiated cell	■	Type 2 cell
❶	Partially differentiated cell	▩	Type 2 cell
⓯	Neuron cluster with 15 members.	□	Type 2 cell
⓰	Neuron cluster with 16 members.	☒	Vacant site

4.5 Chapter Summary

This chapter describes the implementation of the cell growth simulation program. The properties used have been organized and discussed. Cellular properties are related to each cell type and the cell type fractions are related to the cortical strata. The cellular properties are classified between the primary and secondary properties, where the latter are defined or inherited from the former. Interconnection density was introduced and shown to be important in the modeling of an artificial cortex and the design of the artificial chromosome. A differential expression based genetic algorithm was developed and the cellular and cortical parameters were encoded into an artificial chromosome. A technique to model fractions and probabilities using the LRP convention was developed, where the combination of two or more genes is used to produce fractional values in the structures. The chromosome is organized through the master gene sequence. The order of the master sequence is critical for the correct development of the final structure. The stages of the development process were described and related to the master sequence.

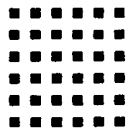
This chapter also describes the vital details of the cell growth simulation model. These details introduce minute variations into the cells so that a differential gradient of control ligand concentration is produced. The differential gradient is the driving mechanism through which cellular interactions occur. The parameters which define the gradient and the equation used in its calculation were discussed. The overall program was briefly described and example output plots were listed. Having described the workings of the model, its theoretical background, and the details of its mechanisms, the next chapter analyzes the program and discusses the results.

EXAMPLE 1

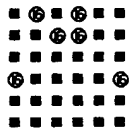
Layer Thickness	16
glbl_low_lay	2
glbl_up_lay	3
glbl_absorb	0.10
crit_radius	13.16
glbl_rad	5.00
mature_incr	5
low_mat_lim	10
up_mat_lim	30

Blastular Stage

Iteration 1



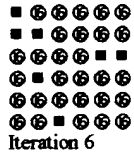
Iteration 2



Iteration 3

Iteration 4

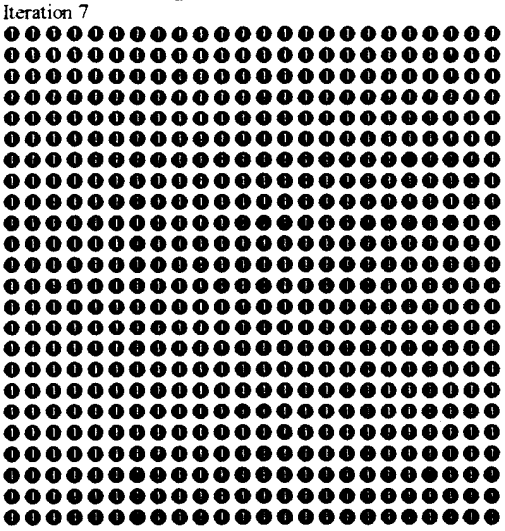
Iteraion 5



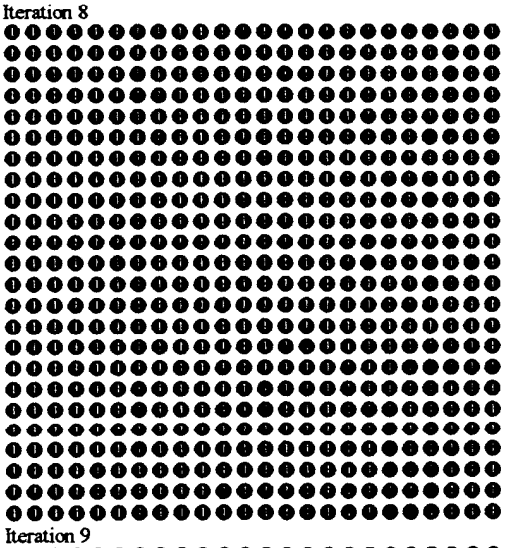
Iteration 6

Embryonic Stage

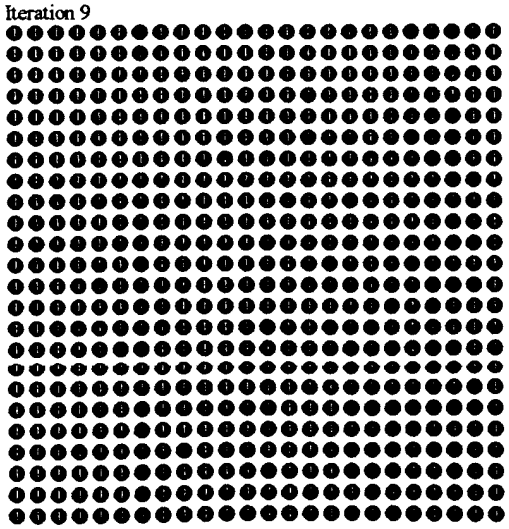
Iteration 7



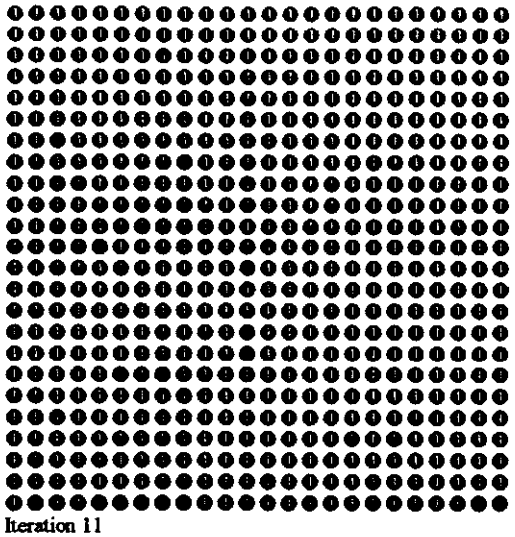
Iteration 8



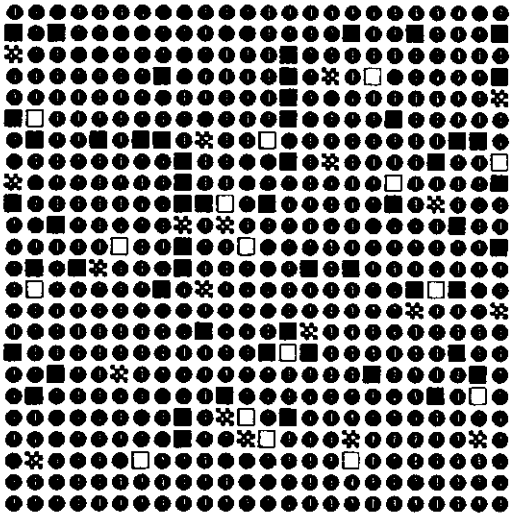
Iteration 9



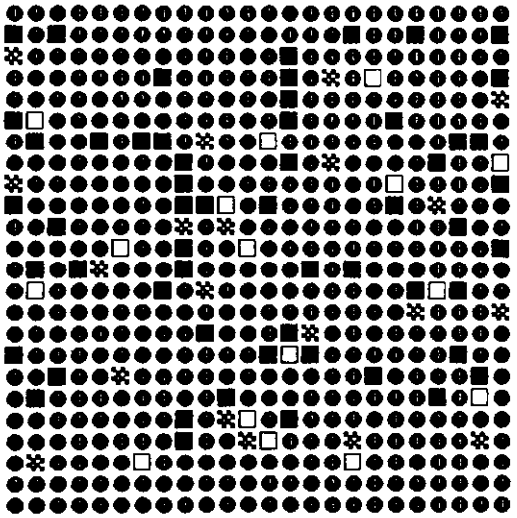
Iteration 10



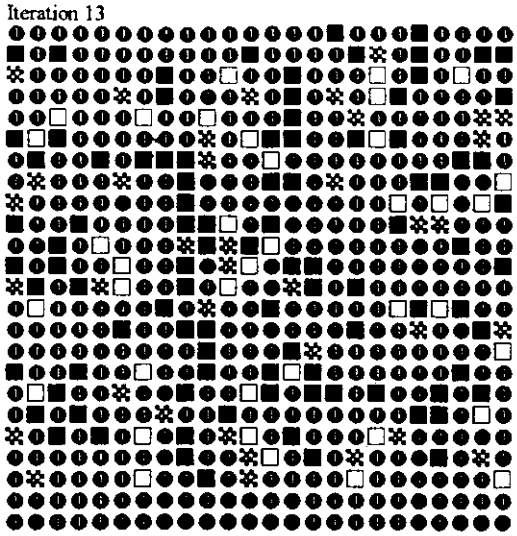
Iteration 11



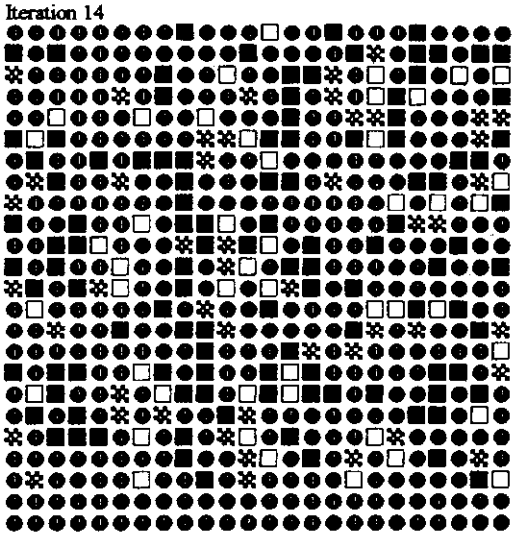
Iteration 12



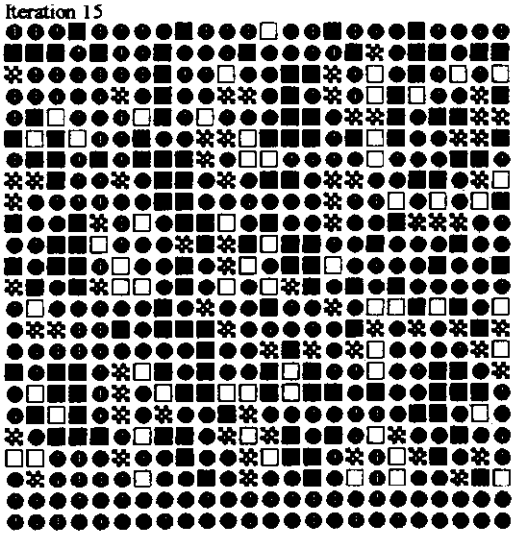
Iteration 13



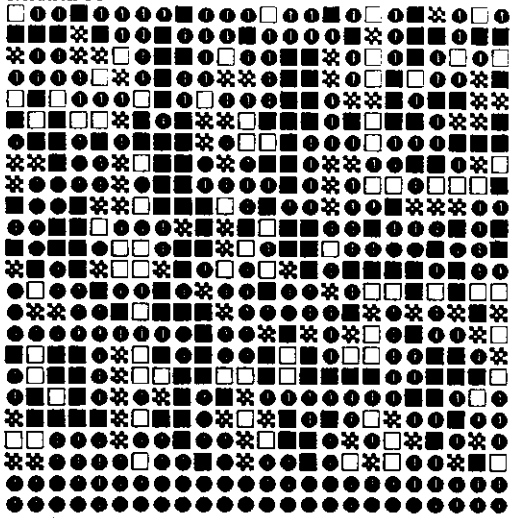
Iteration 14



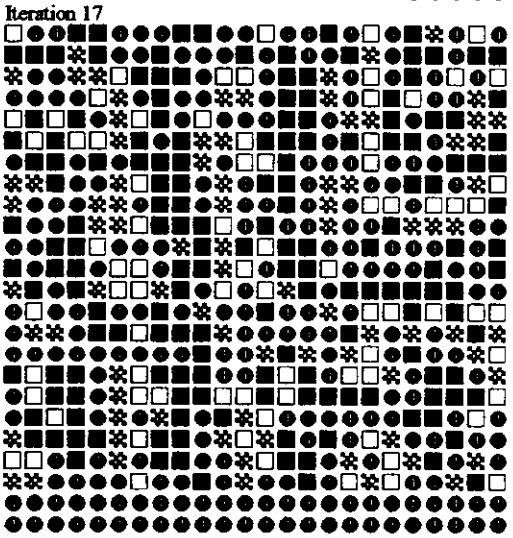
Iteration 15



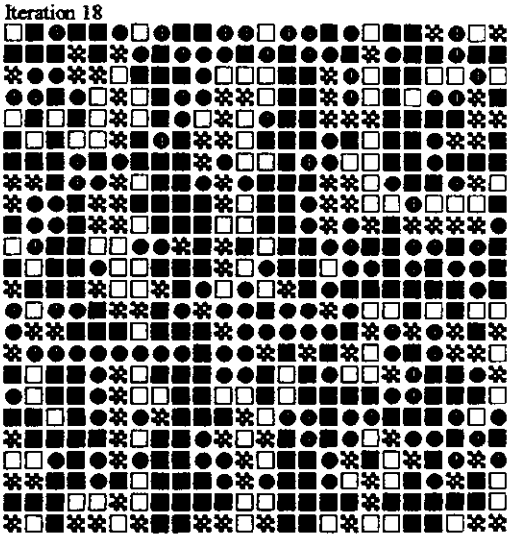
Iteration 16



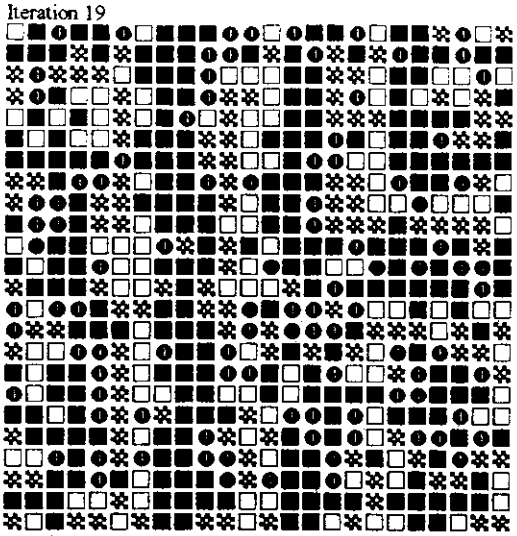
Iteration 17



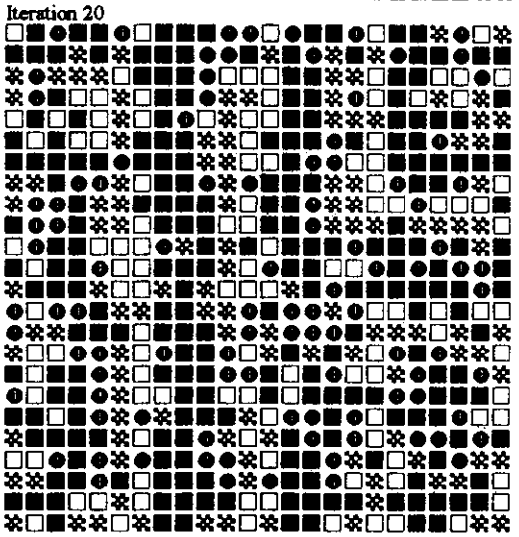
Iteration 18



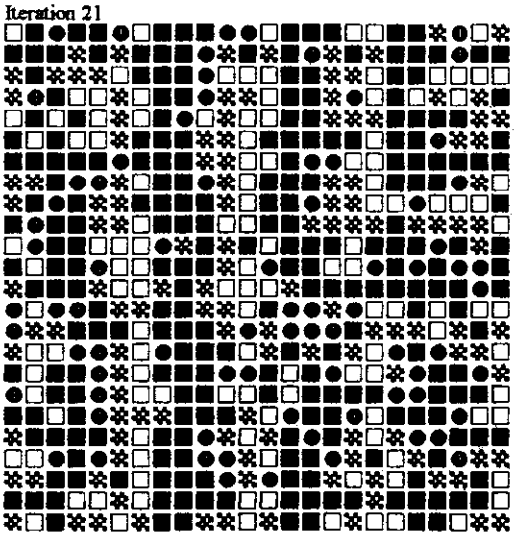
Iteration 19



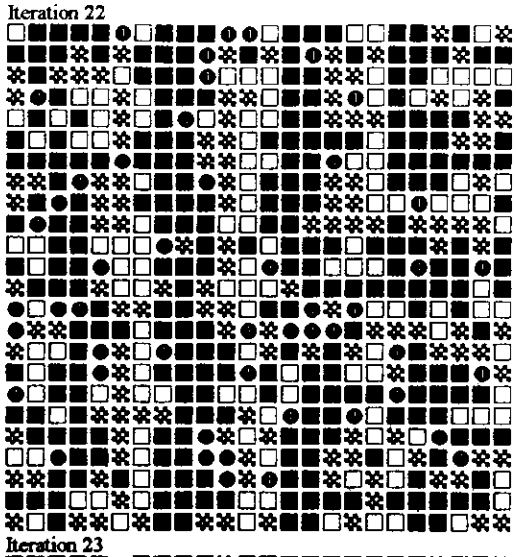
Iteration 20



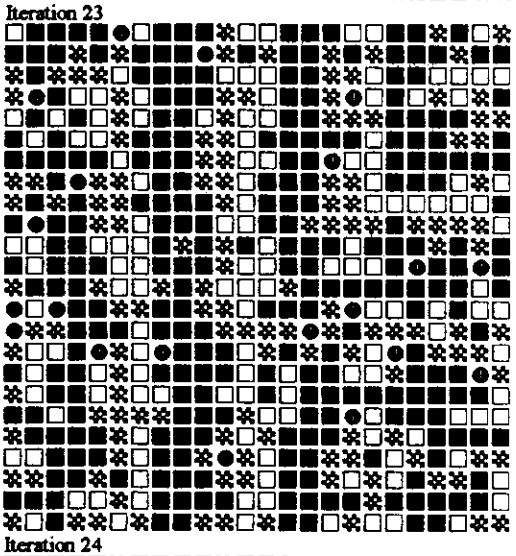
Iteration 21



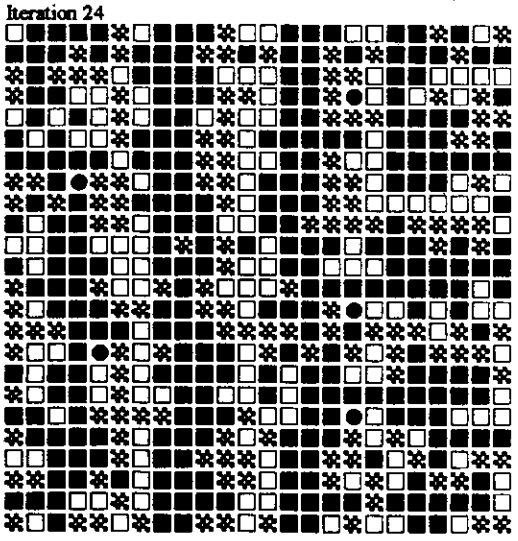
Iteration 22



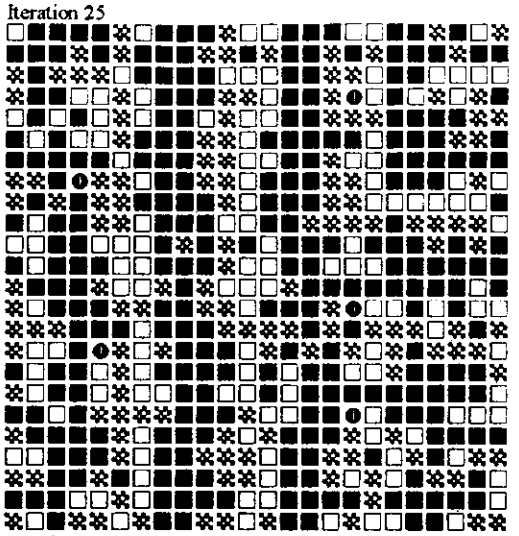
Iteration 23



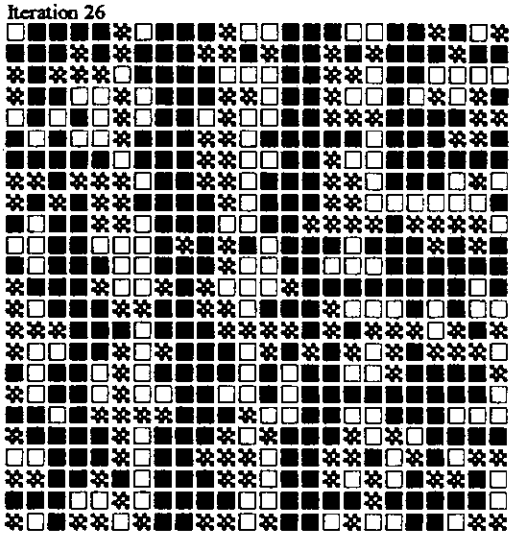
Iteration 24



Iteration 25



Iteration 26

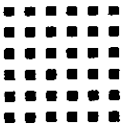


EXAMPLE 2

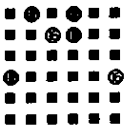
Layer Thickness	15.25
glbl_low_lay	2
glbl_up_lay	3
glbl_absorb	0.10
crit_radius	13.16
glbl_rad	5.00
mature_incr	5
low_mat_lim	10
up_mat_lim	30

Blastual Stage

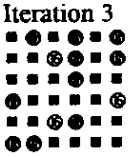
Iteration 1



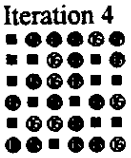
Iteration 2



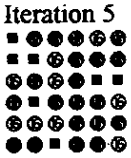
Iteration 3



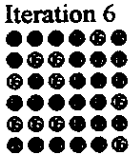
Iteration 4



Iteration 5

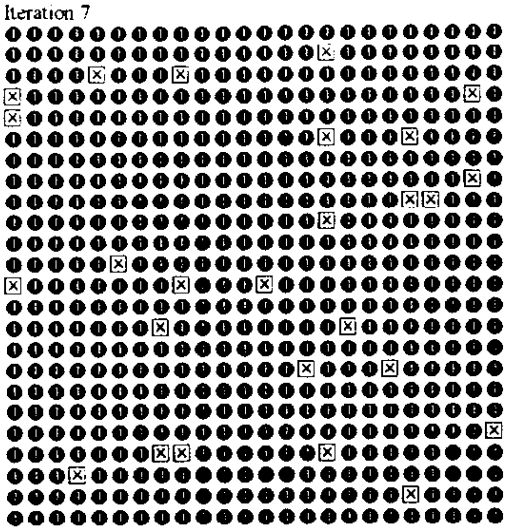


Iteration 6

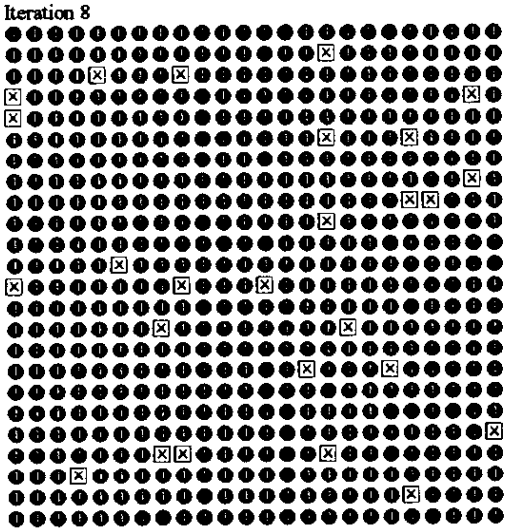


Embrionic Stage

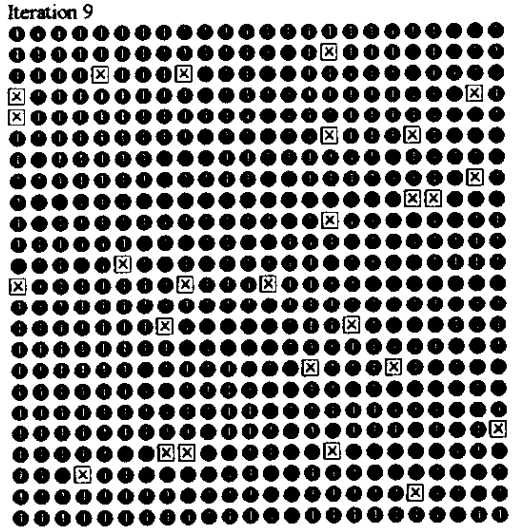
Iteration 7



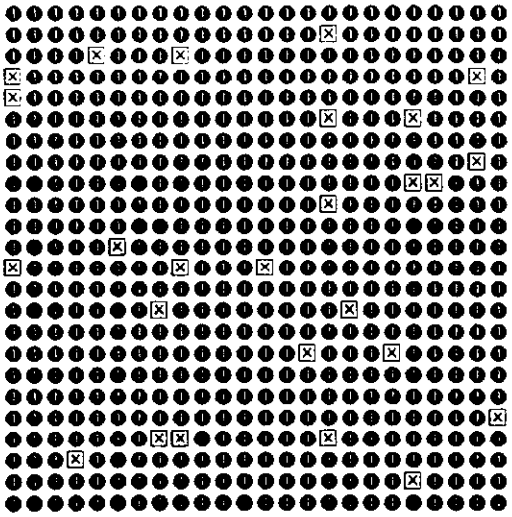
Iteration 8



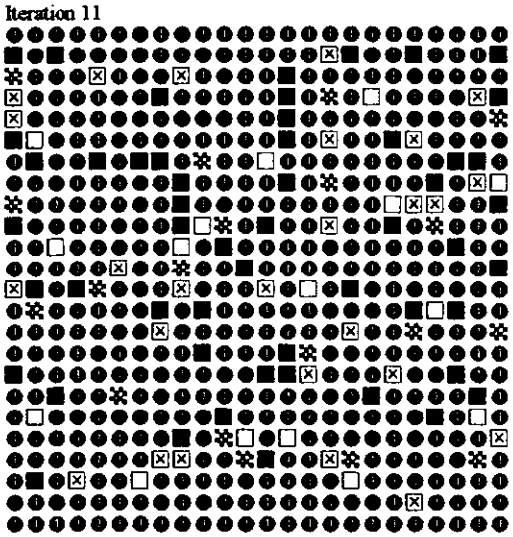
Iteration 9



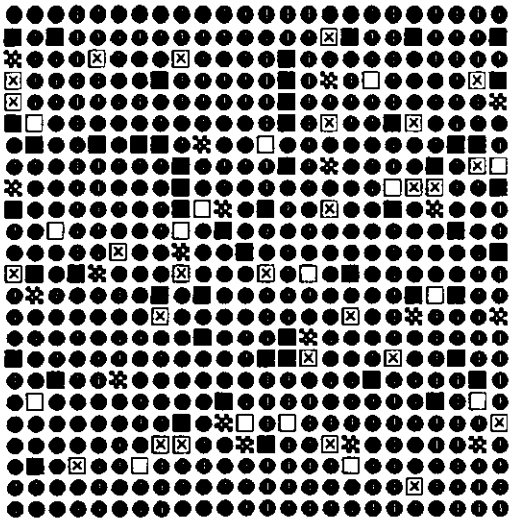
Iteration 10



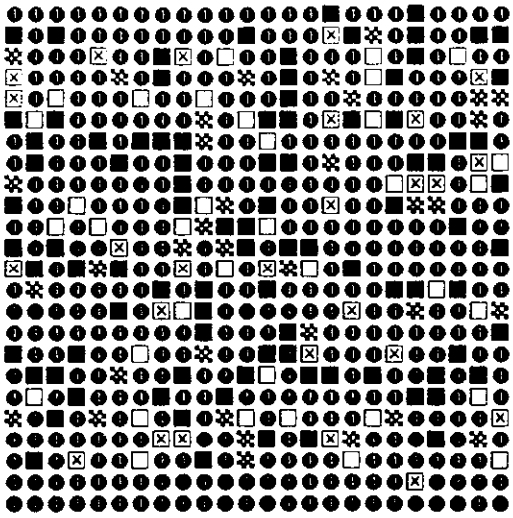
Iteration 11



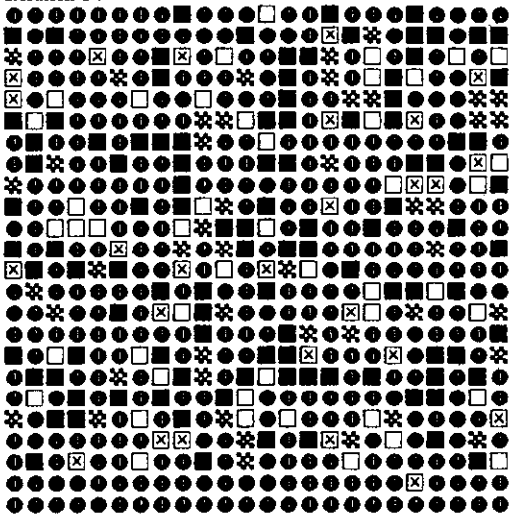
Iteration 12



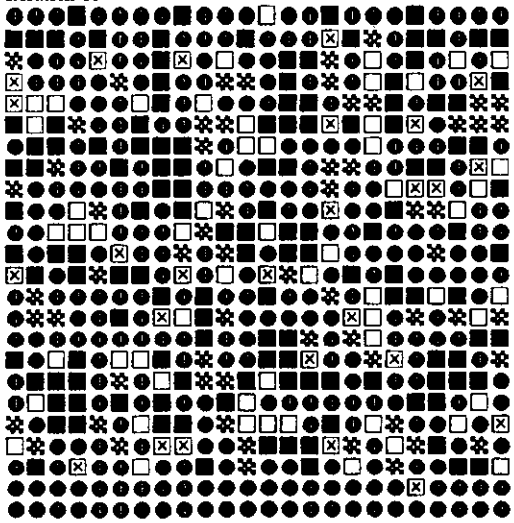
Iteration 13



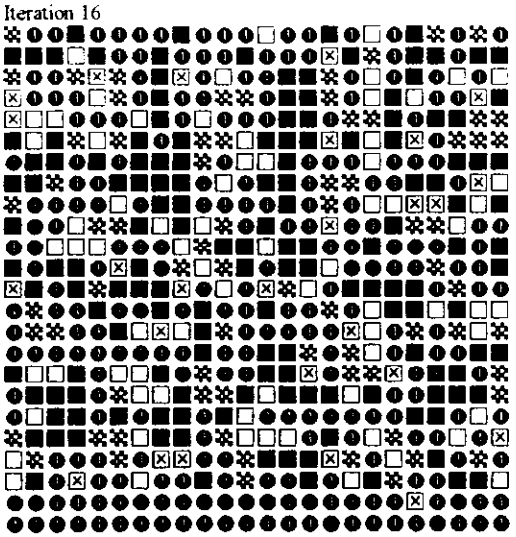
Iteration 14



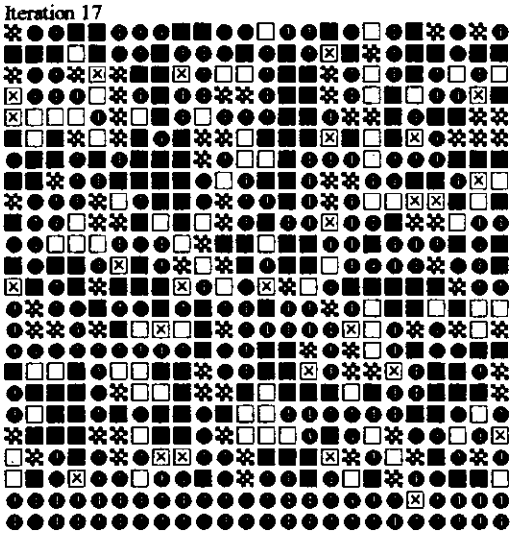
Iteration 15



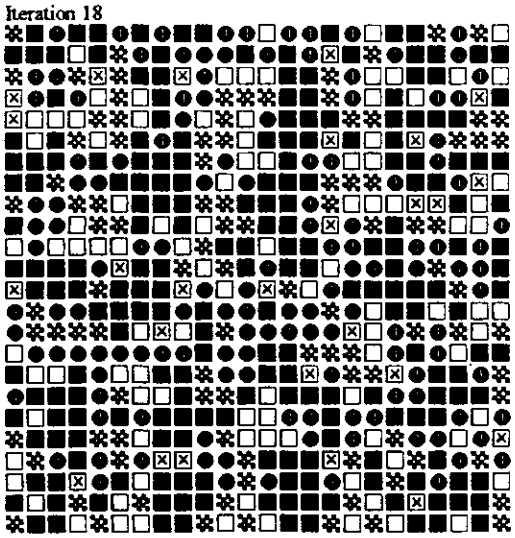
Iteration 16



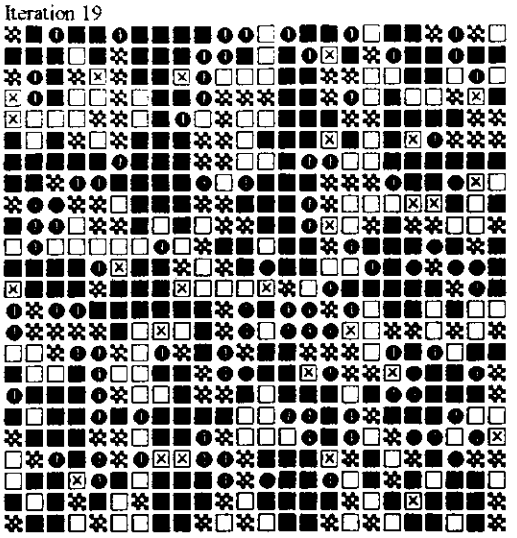
Iteration 17



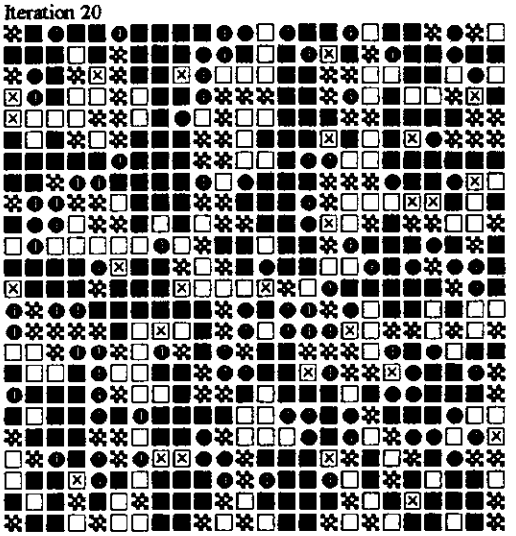
Iteration 18



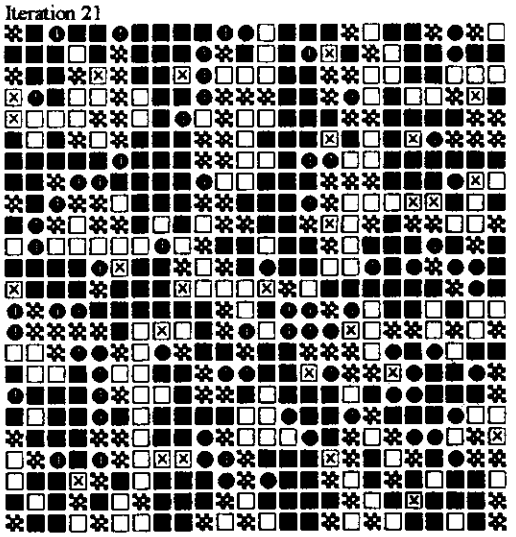
Iteration 19



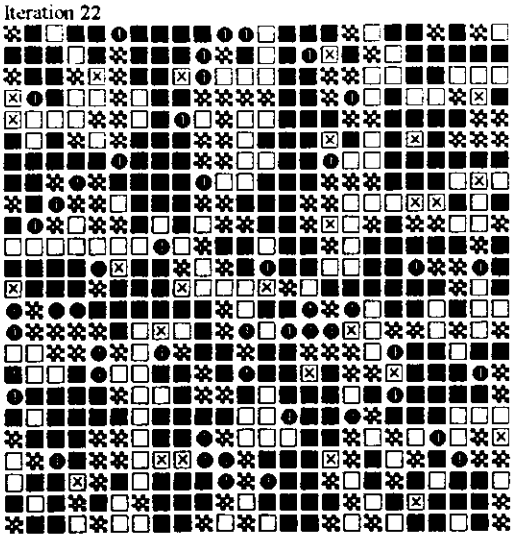
Iteration 20



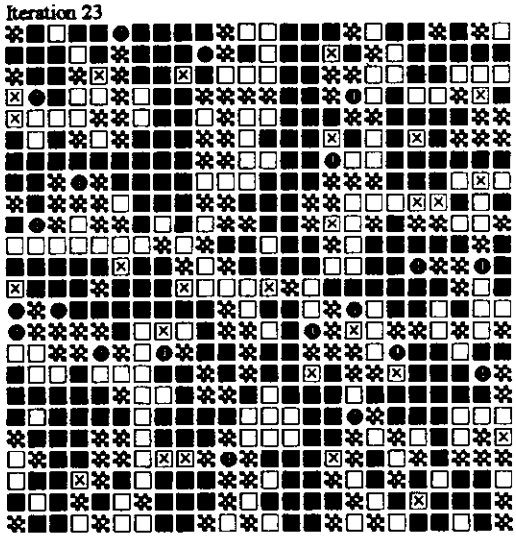
Iteration 21



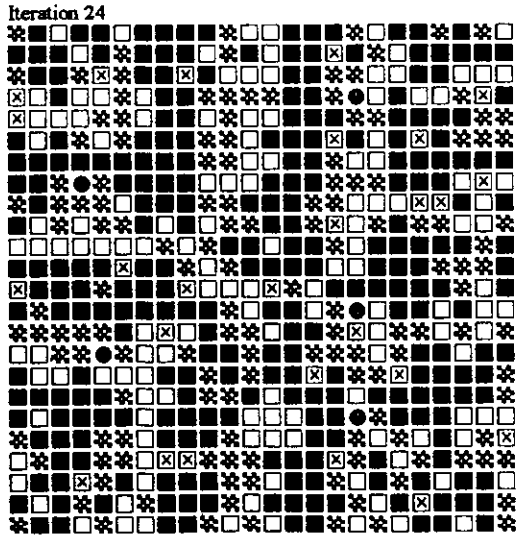
Iteration 22



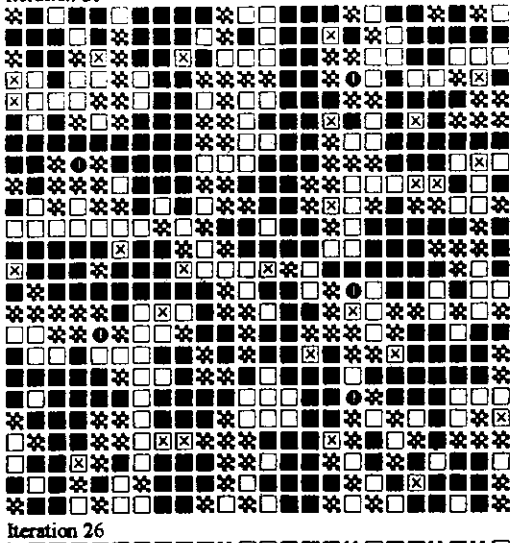
Iteration 23



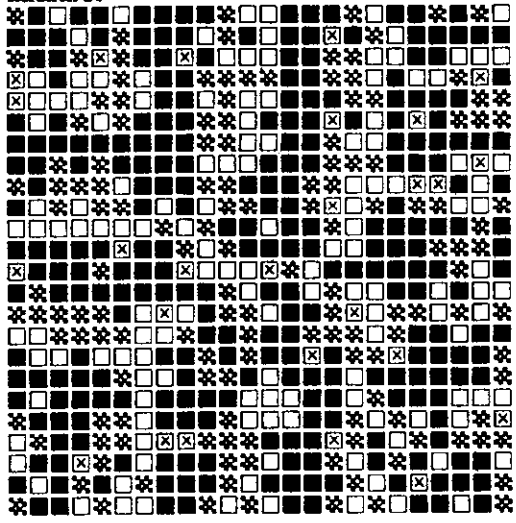
Iteration 24



Iteration 25



Iteration 26



5. DISCUSSION

This chapter discusses the output of the growth simulation program, and describes future work. There are two major sections in the chapter. The first is a discussion centered around the development of the program, its significance with respect to the model, and the parameters that were experimentally found to have significant effects on the performance of the simulation program. The second section is a discussion of recommendations for future work which addresses the issue of integrating the network into other systems and the direction for future network expansion, including a section discussing the implementation of this technique to the problem of automated visual inspection. A discussion of differential expression based genetic algorithms is also included in the future study discussion.

5.1 The Simulation Program

This first section describes in detail the behavior and output of the growth simulation program. The section begins with a brief description of the evolution of this research and the order in which several of the critical assumptions were realized. Following that discussion, discrepancies between the theoretical model and the program are described. The program demonstrated that the traditional definitions of cell types may not be necessary since the cell types evolved simply by the manifestation of cellular characteristics. This discussion is continued to describe cellular properties and emphasizes that observed global features or patterns can be the result of local or cellular mechanisms. The cellular or local mechanisms do not necessarily have to be equal or similar to the global features they produce. The third part of this section discusses the effects of converting the model into a software environment. Finally this section discusses the actual output and the parameters which control the morphology of the observed patterns. The effect of each parameter is separately demonstrated and the overall accuracy of the program is evaluated. The error analysis indicates the parametric limits of the software and the reasons for these limits are discussed.

5.1.1 Evolution of the Program

The development of the simulation program was both based on the model, and simultaneously contributed to the model's development. The architecture of the simulation program reflects its experimental development in two ways. First, the program is clearly is divided into three major sections as discussed earlier: data conversion, growth simulation, and output file generation. These modules were actually written separately and later

combined. Second, the simulated development has clearly defined boundaries between the gastrular, embryonic, and fetal stages of development. In fact, these three stages were initially written as three separate programs.

The discussion that follows describes the traditional views of genes. The *genes* in this discussion refer to traditional genetic algorithms, not biological genetics. At the onset of programming, the genes were still being treated in the traditional sense, and it was assumed that the value of a specific gene was a parameter. However, this view requires that the program be written in an algorithmic form such that the genes are used only to complete an equation. For example, consider trying to define the network as it would appear at the end of the blastula stage. One might consider writing a function that would directly read in the parameters of the stratum thickness and dynamically allocate arrays to those dimensions. This traditional parametric treatment of a gene obviously does not parallel the biological mechanisms. Biological genes code for proteins, and these proteins initiate a series of chemical reactions. In other words, a gene represents a mechanism, not a parameter. Any deviation from this concept was viewed as not only inaccurate, but detrimental to successful development of the cell growth simulation program.

The second traditional view of the function of a gene is the assumption that each gene performs a unique function that is wholly separate from the function of other genes. This view leads to the creation of a system that has unique functions for each gene, or more specifically, a program that interprets each and every gene in a unique fashion. This second assumption introduces implied complexity to the system.. While this argument is not presented to suggest that the behaviors and mechanisms in biological systems are fundamentally simple, it is to say that a high degree of repeatability of the mechanisms should be observed across different organisms. Modern mathematical analysis of system consistently identifies the fractal nature of natural growth patterns. The fractal grow patterns indicate the biological process are fundamentally fractal in nature. In fact, there is only one fundamental way to interpret the DNA. In all organisms, from viruses to bacteria to plant and animal cells, the same triplets code for the same amino acids. The sequences of the triplets change, but the fundamental mechanism by which they are interpreted do not. Furthermore, the most fundamental gene that codes for ATP, a molecule that supplies cellular energy, is approximately 90% identical across all organisms that use and produce ATP. These observations clearly indicate that the underlying mechanisms behind the functions of cells have a high degree of uniformity and repetitiveness in different cells. In other words, the

mechanism of the cell growth simulation program should represent fractal system, not complex or unique system.

It is the underlying assumption of this work that the mechanisms encoded by genes, though very simple individually, can occur simultaneously with many others. It is further assumed that a cell is a pure parallel processing machine and that the behaviors observed by scientists are not the effects of a single mechanism, but rather the composite effects of numerous mechanisms. If each gene is considered responsible for one mechanism, then the number of possible behaviors defined in the genome is given by the factorial of the number of genes contained in that genome.

In summary, to correctly model the biological system it must be assumed that only a few types of basic mechanisms exist. There may be hundreds to millions of variants of these basic mechanism types; however, from a modeling point of view, all of these variants reduce to a small, finite set. All of the behaviors and mechanisms observed are the result of the expression of these basic mechanisms. It must further be assumed, based on the fact that there are more behaviors than there are genes, that these mechanisms can operate collectively to produce a behavior. By assuming that behaviors are the result of the expression of numerous simple mechanisms operating in unison, it possible to design a system that can be infinitely upscaled since all local mechanisms function in a cooperative manner to produce the global behaviors.

5.1.2 Cell Types

When this study began it was believed that the assignment of cell types was critical for the overall performance of the model. From a biological point of view, cell type is a crucial parameter because it is observable and describes how the cell behaves. However, cell type, in the context of the model developed in this study, is essentially an artifact of selectively expressed cellular properties. In other words, the cell type does not define the properties, rather the properties define the cell type, which means that it is possible to develop the various cell types simply by differentiating and expressing genes associated with the cell properties. Since the true biological mechanism that leads to the appearance of cell types is not known, it is appropriate to model the generation of cell types either as a single differentiation, or as the result of the collective property differentiations. From the standpoint of computational efficiency, it is easier to undergo one differentiation step to determine cell type and to inherit the corresponding properties.

5.1.3 Cell Properties

When this study began it was believed that it was absolutely necessary to obtain cellular properties as close to the biological counterparts as possible. While it is true that if all cell properties and parameters could be measured, it would be possible to produce a nearly identical artificial counterpart, it is important to recognize the technical difficulties of cortical replication. The technique presented in this dissertation was not simply developed to allow the design of massive scale neural networks, but it is a development tool in which the design and optimization of neural networks are combined into in a single cohesive model. This approach was taken because in reality it may not be possible, at least in the near future, to obtain an accurate set of cell property data. A perfect set of data from an optimized network will yield an optimal network. An accurate set of data will yield a nearly optimal network when implemented; and a marginal set of data will yield a suboptimal network. All networks developed with this technique can be bred and the offspring can be used to create successive generations that should eventually yield an optimal network. Therefore the more accurate the initial data, the fewer the number of generations that needed before the optimal network is obtained. In other words, the system will converge faster with good data, but will also converge with less accurate data. Thus the accuracy of the initial data is more of a luxury than a necessity.

The very statement that an optimal network is sought implies that there exists some means by which to define the optimal state, and as is the case with many industrial engineering problems, optimal is at best a relative condition. As one delves deeper into this technique and becomes more closely aware of its implications when upscaled, one realizes the enormous number of possible networks that can be developed from a gene pool that contains dozens of alleles for all of the cell properties for all of the cell types.

5.1.4 Two Dimensional Grid Transformation and Grid Parameters

The three dimensional structure of the brain can in theory be truly modeled in a computer environment by storing the three-dimensional positions of the cells. The consequences of treating each stratum as a three dimensional system include greater programming complexity and slower processing rates to run the simulation. The loss of program speed is a result of the increased memory needs and calculation complexity associated with determining the intercellular distances that the differential gradients are based upon.

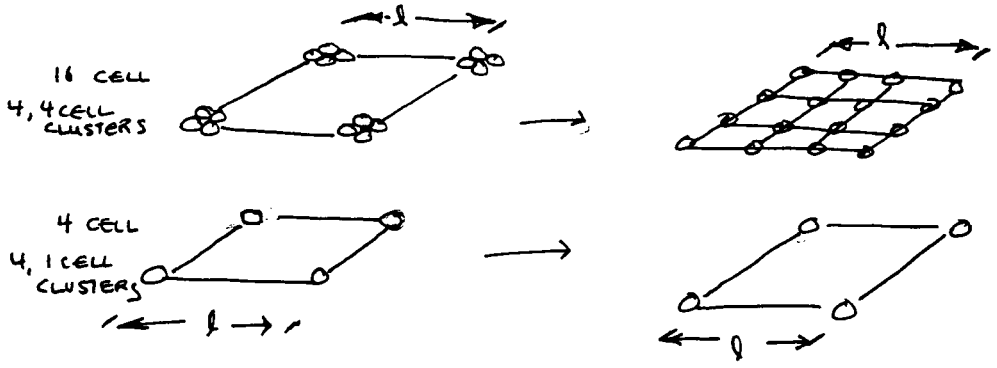


Figure 5.1 Representing Stratum Thickness by Modifying the Lattice Parameter

The model presented in this paper attempts to represent a three dimensional structure with a two dimensional grid by proportionally scaling the lattice parameters of the thicker strata so that a higher neuron per area density is achieved (Figure 5.2). This approach is a reasonable proximity and allows the model to partially compensate for the fact that the different strata have different unit cell counts. However, by modifying the lattice parameters, the number of cells reached within a cell's connection radius is affected. It can be argued that a thicker stratum gives the cells within it a greater probability of obtaining their theoretical synaptic count, because on average, a larger fraction of the volume encompassed by the effective synaptic radius can be reached within the stratum (Figure 5.2).

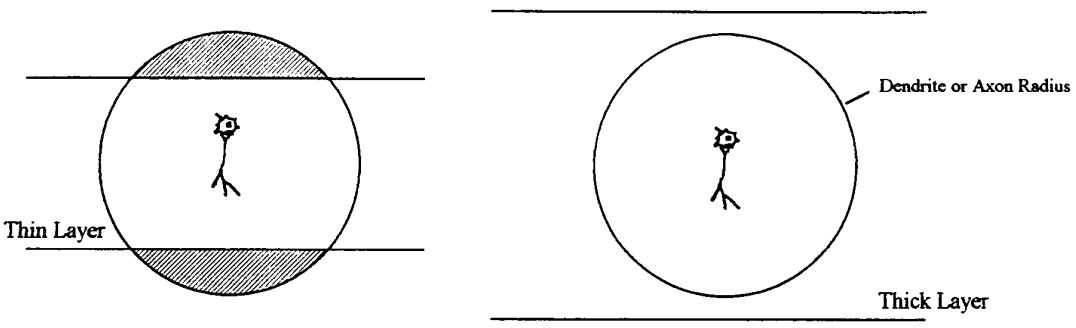


Figure 5.2 Effect of Stratum Thickness on the Number of Connected Cells

However, simply treating the thicker strata as having a higher per cell synaptic count ignores the fact that the three dimensional volume may actually be used in the generation and separation of neural pathways. Since it is known that the complex neural networks targeted

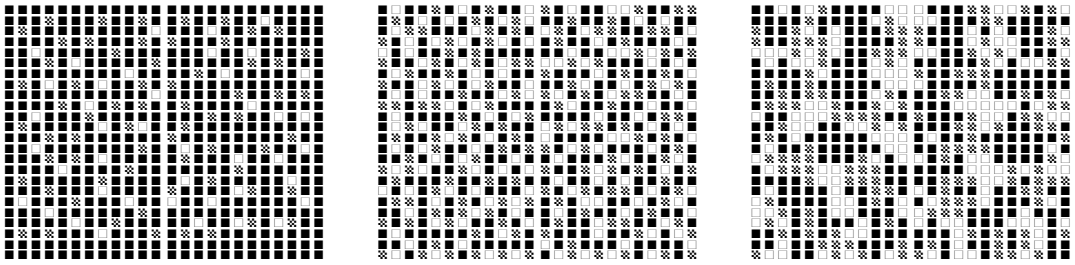
for replication are capable of having numerous neural pathways and that all of those pathways are functionally separate such that they can be used independently, it must be assumed that the actual physical placement of the cells plays an important role in preventing pathways from crossing illogically. It is the basic premise of this work that the functionality of neural networks is intimately and inseparably related to the structure. The proven dependence of the function of a neural network on its structure, shown in the literature review, suggests that the geometric placement or relationship of the cells with relation to each other plays a crucial role in the design and development of the mechanisms sought for replication.

Although the two dimensional representation of the strata is a valid starting place considering this is the first work of its kind, at the conclusion of this study, it has become apparent that the three dimensional nature of the neural network should be represented in the model. In summary, the two dimensional representation used in this study has produced a functional system closely related to the true biological model, but incapable of capturing all geometric relationships.

5.1.5 Effects of the First Cells to Differentiate and Edges on Pattern Morphology

The simulation program bases all decisions on differential ligand concentration gradients. The differential gradients, as discussed in section 4.2.3, are calculated by adding the secretions of neighboring cells. The final definition of any given cell in the network is dependent on the development of its neighbors, which is in turn dependent on their neighbors, and so on. In other words, the local differentiation of a given cell is at least indirectly related to the differentiation of every other cell in the network. If the neural network was of infinite size, then it would be believed that all of the cells develop in an equal fashion; however, these networks are of finite size, thus the cells in the peripheral regions of the network receive substantially less stimulation than the interior counterparts. As a result the development of the exterior elements should not be viewed as truly characteristic of the rest of the network. Such *edge effects* are especially pronounced in systems that have very high absorption coefficients. In such systems, the cells only receive signals from immediate neighbors, thus edge and corner elements are lacking 1/3 and 1/2 of the stimulation of interior elements respectively. The result of under-stimulation biases the cells to differentiate to the type with the highest LRP value. In the case of Figure 5.3(a), the bordering cells uniformly differentiated to the same cell type. Edge effects are less pronounced in systems with moderate to low absorption coefficients since a differential gradient can develop at peripheral

sites based on the significant long range contribution of interior elements, (see Figure 5.3b, c). In systems with moderate to low absorption coefficients, the characteristic pattern develops in peripheral cells with very minor distortion. The overall result of the effect of edges suggests that the network under study should be slightly larger than needed so that the edge portions can be neglected. Only the central portions of the network should be considered truly characteristic of the pattern developed from the artificial chromosome and growth simulation program.



(a) (b) (c)

Figure 5.3 The Effects of Edges on Pattern Formation for Absorption Coefficients: (a), 0.95; (b), 0.50; and (c), 0.1.

If one reviews the development versus iteration plots in Section 4.4.3, it becomes apparent that certain cells mature more rapidly and effectively determine or bias the development of the other cells around them. These rapidly maturing cells are analogous to the seeds observed in the solidification of polycrystalline materials. The magnitude of the effect of these seed neurons on their neighbors is fairly difficult to calculate. In addition it is not known to what degree the formation of these seed neurons deviates from the biological model, clearly suggesting that certain cells mature more rapidly due to nutrient availability. However, the biological model is a continuous function, whereas the simulation program operates in discrete iterations. The effect of these seed neurons probably more closely resembles that of their biological counterparts with increasing numbers of iterations, and the overall process more closely resembles continuum.

5.1.6 Controlling Parameters

There are six parameters related to the differential gradient that have been found to affect the morphology of the final structure. The morphology of the structure is used here to describe the overall appearance of a stratum. Characterizing the morphology of the strata is a process that can only be done after reviewing numerous structures. To the inexperienced

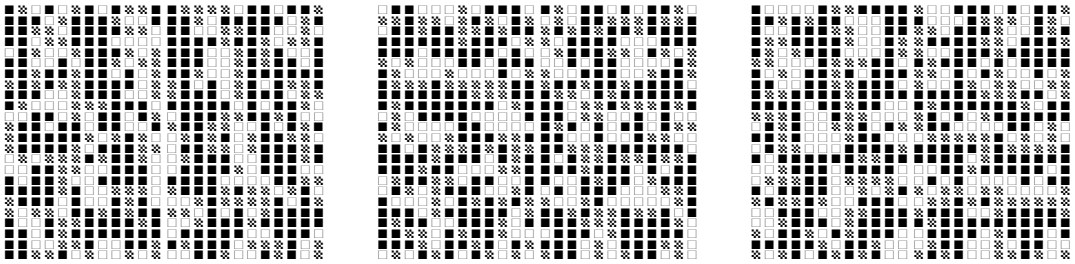
observer, structures produced by the growth simulation program appear as randomly distributed numbers in a simple matrix; however, the trained eye is capable of detecting certain abstract patterns and features. The features most easily characterized are clumpiness and homogeneity. Clumpiness refers to the number of cells of the same type that are adjacent. Stringiness is a variant of clumpiness in which the similar groups are arranged in patterns with high aspect ratios. Homogeneity describes the condition where the overall pattern of neural distribution is identical in all portions of the network. The most homogeneous of networks are those with very low clumpiness. The four parameters discussed in this section were created based on the modeling considerations previously discussed. As the development of the program proceeded, certain modifications were made to enhance the effectiveness of these parameters to facilitate the user's ability to control the morphology of the strata.

5.1.6.1 Puberty Iterations

The number of puberty iterations determines the number of cycles the program performs before all cells release ligands and produce receptors. When a cell begins to release ligands and produce receptors, it is said to have reached puberty. Before puberty, the cell is referred to as being immature and after puberty, when the cell has selected which gene to express, it is referred to as mature. When a differentiation cycle begins, all of the cells are immature and release no ligands. As discussed in an earlier section, all cells are assigned a maturity value which is a uniform random number between 0 and 100. The number of cells that reach puberty is on average equal to the total cell count divided by the number of puberty iterations. The duration of the puberty cycle is the number of iterations needed to reach maturity, and is regulated by the growth rate as discussed in the next section. Thus the total number of iterations needed to differentiate a stratum is greater than the number of puberty iterations.

The number of puberty iterations has little effect on the morphology on the resulting pattern. Group separation appears to be assisted by more puberty iterations. Changes in the number of puberty iterations for the same initial conditions does result in unique patterns, as seen in Figure 5.4. Figure 5.4 shows that increasing the number of puberty iterations increases the amount of time necessary to converge on the final pattern. Since the morphology does not change by decreasing the number of puberty iterations, from a computational point of view, it is more logical to have a very small number of puberty

iterations to decrease the overall processing time. Since the number of puberty iterations is only a theoretical parameter, decreasing or eliminating it is justified.



(a) (b) (c)

Figure 5.4 The effect of the number of puberty iterations on pattern formation. 1,5 and 10 puberty iteration for figures (a), (b) and (c) respectively.

(glbl_low_lay 2) (glbl_up_lay 3) (glbl_absorb 0.10) (crit_radius 13.16)
(glbl_rad 5.00) (mature_incr 1) (low_mat_lim 10) (up_mat_lim 30)

5.1.6.2 Rate

Once a cell has entered puberty, it begins to produce ligands and receptors at a certain growth rate until the total number specified in the chromosomes is obtained. In the computer model, the differential gradient is produced in a certain number of iterations and the growth rate corresponds to the fraction of total ligands and receptors additionally produced each cycle. Since the growth rate is assumed to be dependent on the availability of nutrients, the equation given in Section 4.10.1 incorporates the ***MATURE*** term. The rate equation repeated here for convenience has an upper and lower limit specified by terms A and B, where A is the minimum percentage produced each cycle and A+B is the maximum.

The expression for rate was experimentally evaluated using three methods. First, B was set to equal a constant value of 10, while the effect of the lower limit A was evaluated independently. The results of this study, seen in Figure 5.6, show that the number of iterations to convergence decreases exponentially with the value of A and that the processing time decreases about 40% across the A values of 1 to 7, remaining constant above 7. The stability of the processing time to variations in the minimum growth parameter A is due to the fact that the program only evaluates undifferentiated cells, therefore the total amount of computations necessary to converge on the final pattern remains more or less constant regardless the number of iterations. The 40% decrease in processing time in the lower range A can be associated to a five-fold drop in the number of iterations that occurs over that range.

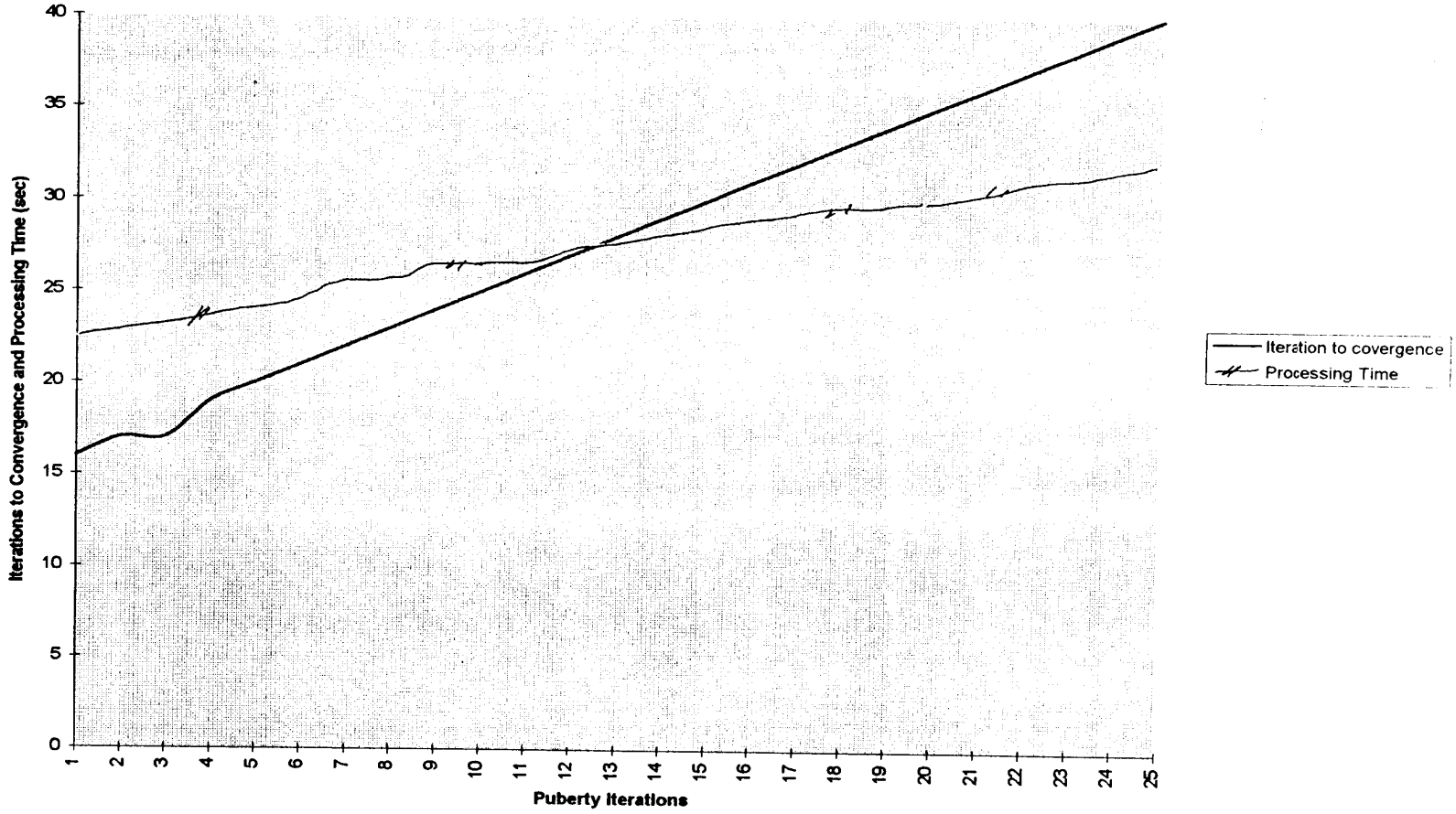


Figure 5.5 Iterations to Convergence and Processing Time vs. Number of Puberty Iterations

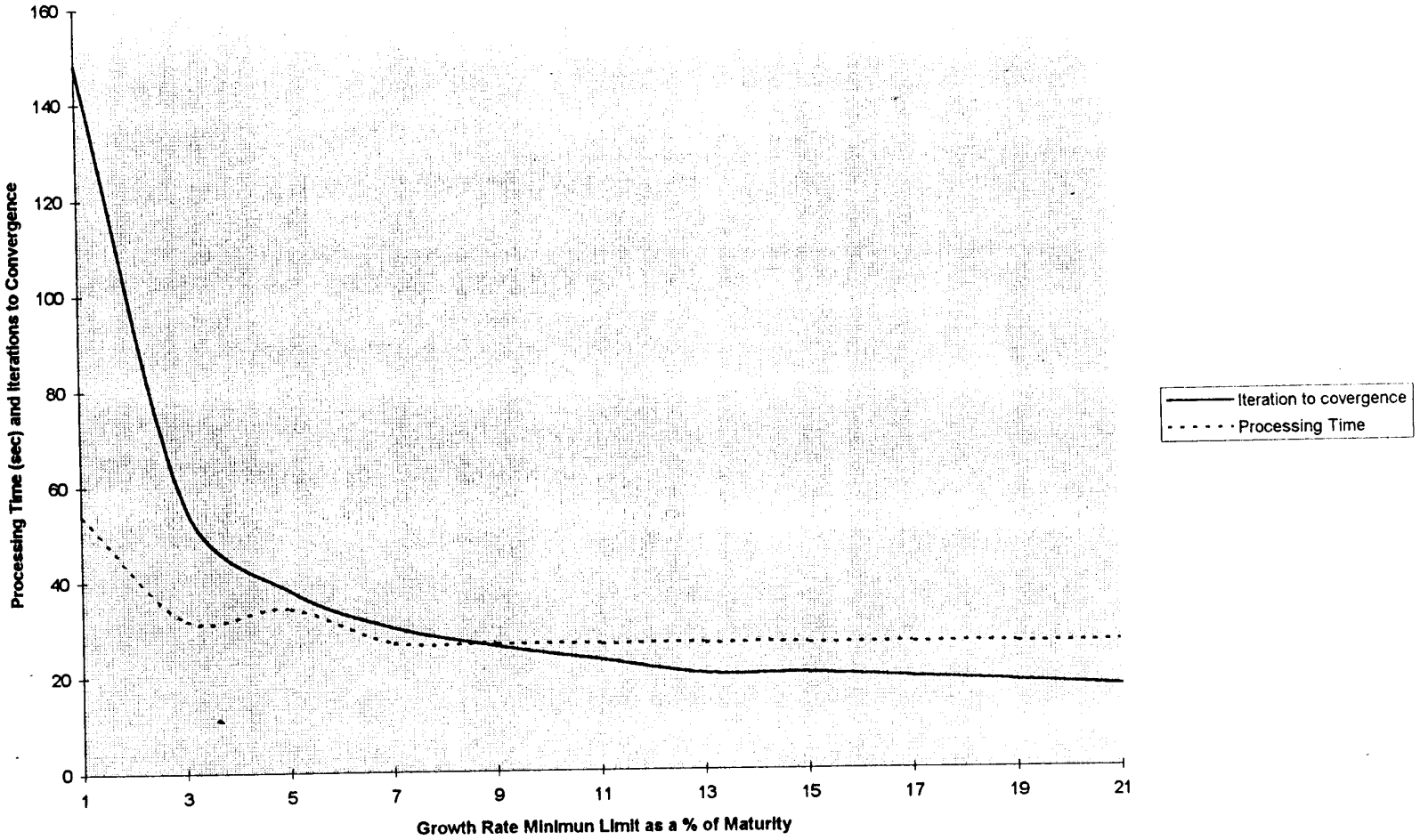
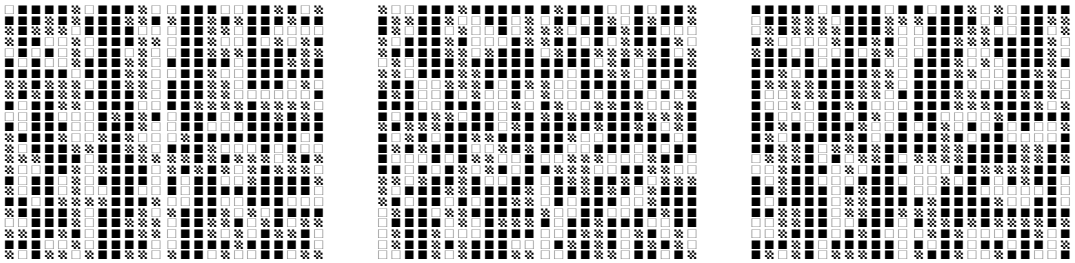


Figure 5.6 Iterations to Convergence and Processing Time vs. Growth Rate Minimum Limit

The effect of parameter A on pattern morphology is maximized for moderate values such 30-40 %. A moderate A parameter appears to assist in group separation, where small and large values tend to yield long range clumping, or banding (Figure 5.9).



(a) (b) (c)

Figure 5.7 Effects of Lower Growth Rate Limit on Pattern Formation.

Since the A parameter, or minimum growth rate, ultimately determines when the last cell reaches maturity, it determines the number of iterations to convergence. An analytical expression for the number of iterations can therefore be produced, as seen below.

Iterations to Convergence = Number of Allowed Iterations * (A/100) +
Number of Puberty Iterations

where the number of allowed iterations is a parameter in the simulation program used to model the continuous nature of the differentiation process as a number of discrete intervals. The effect of the maximum growth rate limit, given by (A + B), on the pattern morphology was found to be inconsequential. It was also found that by increasing the maximum growth rate to its highest value, a 3-4% decrease in processing time could be achieved, (Figure 5.9). From this it was concluded that the maximum growth rate should be set to the highest possible value to increase computational efficiency.

A uniform growth rate that is independent of the random ***MATURE*** parameter can be tested by setting the B term equal to zero. The results of this study, shown in Figure 5.10, have similar features to the first two experiments with the rate parameters. Both show a fairly sharp drop-off in the number of iterations to convergence at lower growth rate values and fairly uniform processing times over the entire range.

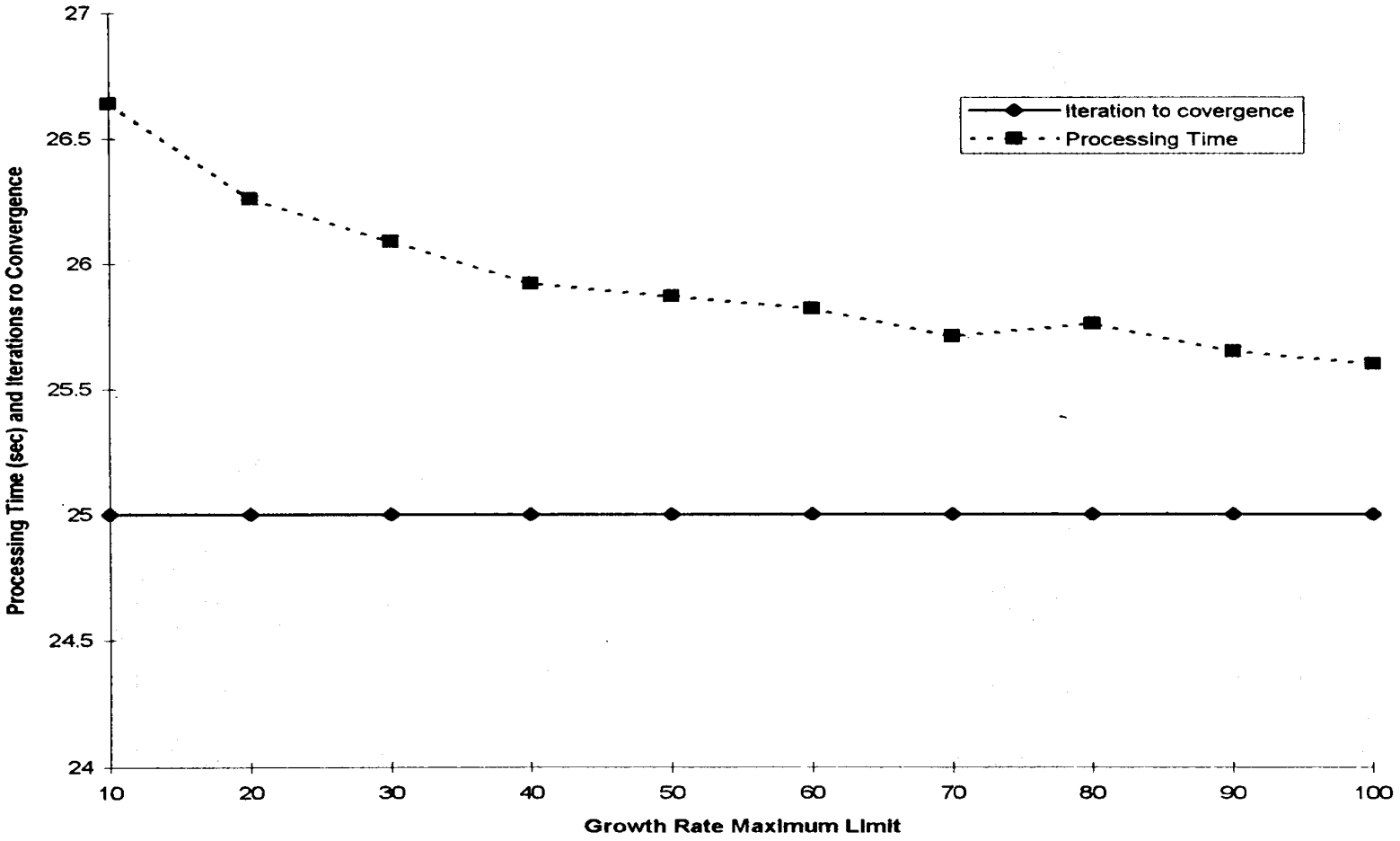


Figure 5.8 Iterations to Convergence and Processing Time for Constant Minimum Growth Rate vs. Growth Rate Maximum Limit

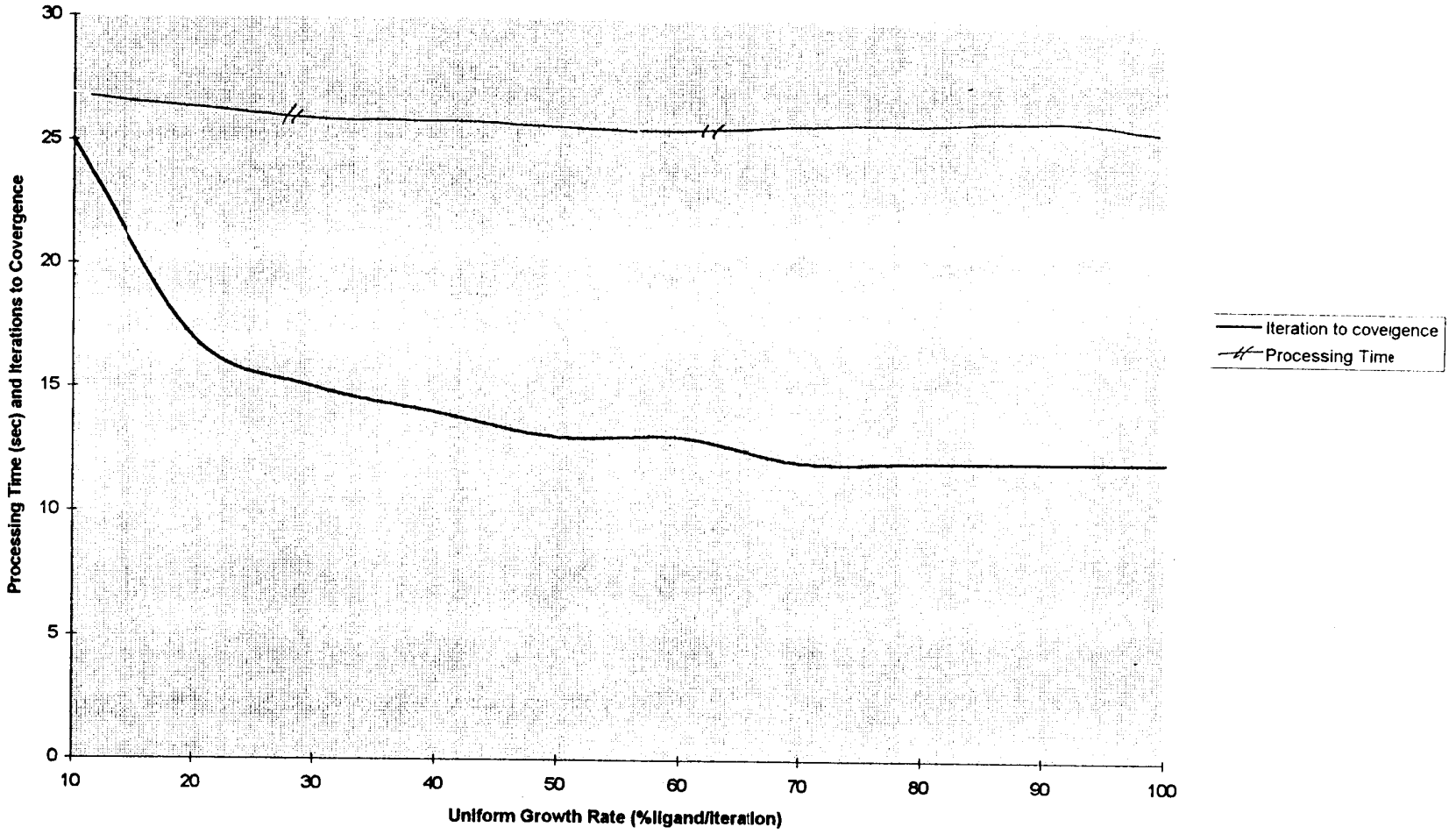


Figure 5.9 Processing Time and Iterations to Convergence vs. Uniform Growth Rate

5.1.6.3 Radius

The first simulation parameter found to have a pronounced effect on pattern formation was the gradient radius. The gradient radius is the parameter in the simulation program the controls how many cells are used to calculate the ligand concentration at a given cell location. When the simulation program was first written, the expression for the ligand concentration dispersion (Equation 4.1) had not been developed fully. The radius term was initially used only to control the processing time (Figure 5.11) to allow the collection of more data. After the first series of data was collected, it was observed that by changing the radius, while holding all other parameters constant, the pattern morphology could be controlled. It was found that small radii produced homogeneous morphologies and that radii greater than 5 caused clumping. As the radius is increased above 5 UCLs, the clumpy patches of similar cell types increase in size until they eventually connect, producing sigmoidal bands. As the radius is further increased, the sigmoidal bands grew to form distinct regions. Examples of these morphologies are illustrated in Figure 5.10.

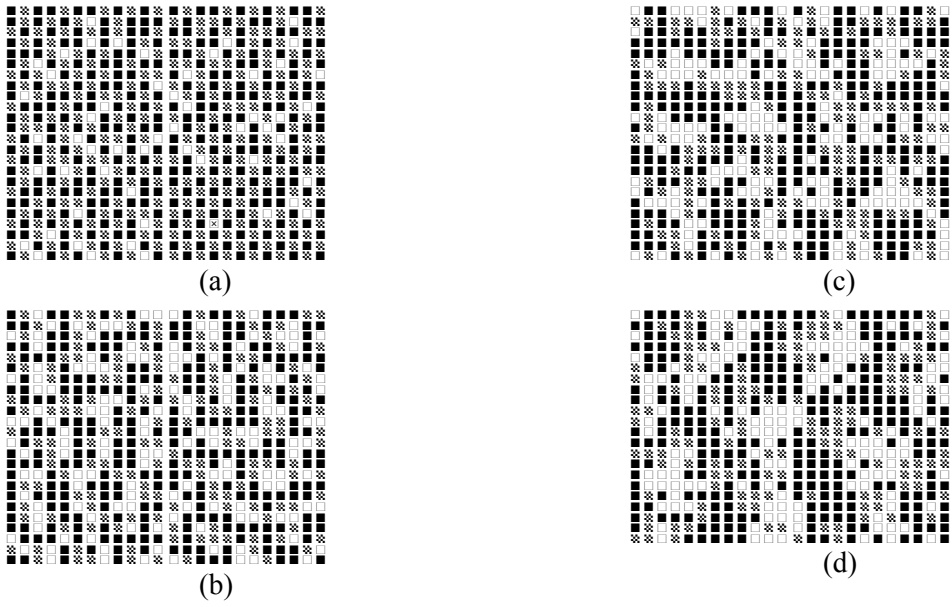


Figure 5.10 Effect of Radii on Pattern Morphology

(glbl_low_lay	*2)*	*(glbl_up_lay*	*3)*	*(glbl_absorb*	*0.10)*	*(crit_radius*	*13.16)*
(glbl_rad	*1.00)*	*(mature_incr*	*5)*	*(low_mat_lim*	*10)*	*(up_mat_lim*	*30)*

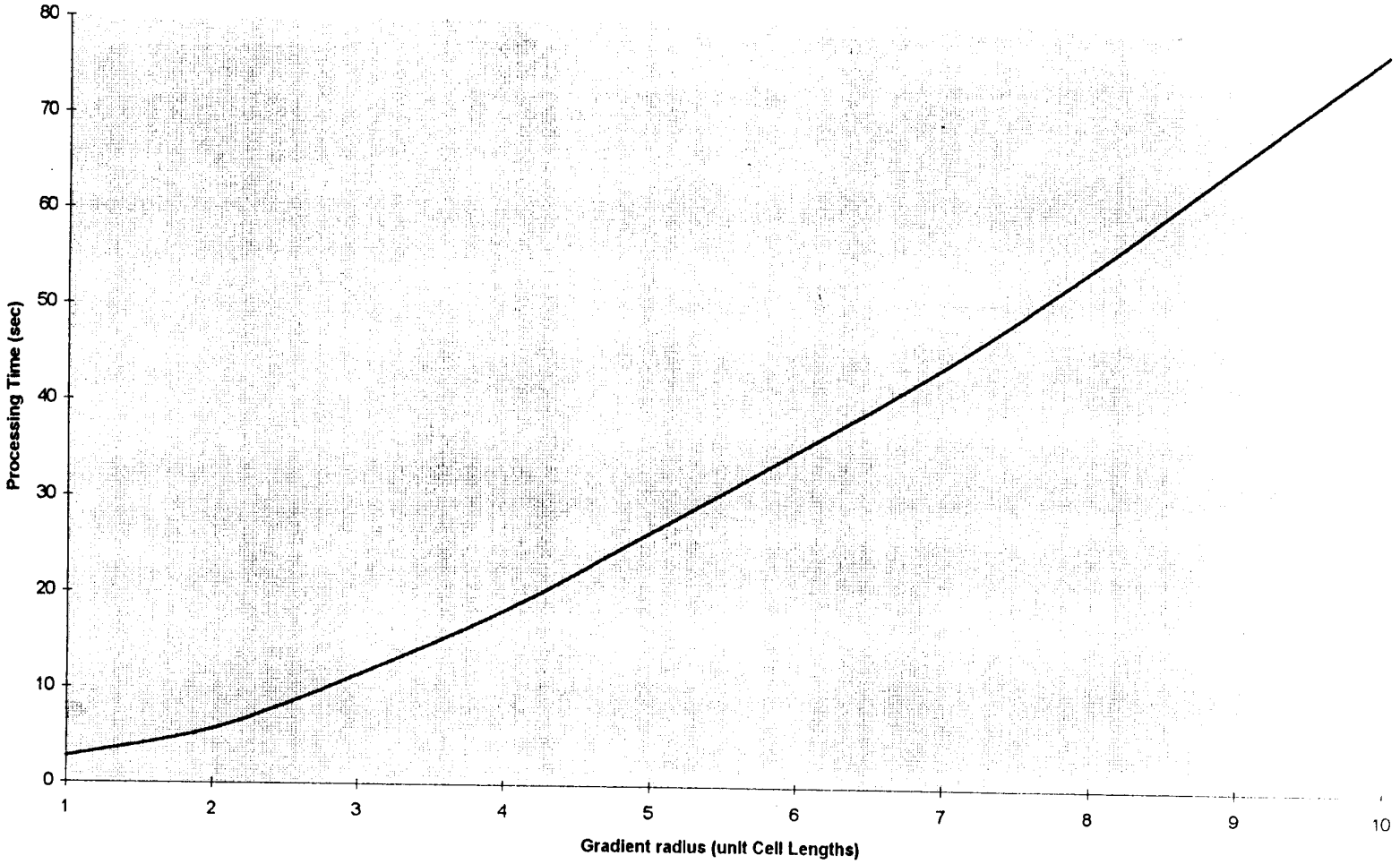


Figure 5.11 Processing Time vs. Gradient Radius

Initially strata with evenly distributed cells were thought to be preferable. However, the development of a technique which can control the pattern morphology is advantageous since some tissues, although not necessarily those under investigation, have similar, overall nonuniform cell type distributions. Thus *radius* is a useful design tool.

The radius parameter could either be controlled directly by the user or incorporated into the artificial chromosome so that it may be controlled by simulated evolution. To incorporate the radius, which is a global parameter, into the artificial chromosome, it must be converted to reflect a local mechanism. To produce the effect of the radius parameter, each cell must, without global information, modify the ligand concentration such that after a critical distance, the effect of the one cell on another is negligible. The power function of section 4.2.3 was produced to model absorption as a local mechanism. The magnitude expression was shown to be an exponential function taking the form,

$$\text{Magnitude}_{(\text{B on A})} = e^{\text{length}*\ln(1-\text{ABS})} \qquad \text{Equation 4.3}$$

which is a general exponential decay type system. The magnitude expression has a similar effect as the radius parameter in that the contributions of distant cells to the total concentration at a given cell site approaches zero. A low rate of decay approximates a large radius, and vice versa. Thus, the global effect of the radius can be achieved by the collective expression of one local parameter, namely the absorption coefficient, ABS.

To utilize the magnitude expression, it must be recognized that there is no limit to the radius, meaning very distant cells, *B*, will still contribute to the local concentration, at cell site *A,* even though their effects are singularly insignificant. This implies that there is a critical radius beyond which the secretions of a given cell are so severely decayed as they pass through the network that their signal alone will not significantly affect the development of the cell in question. After simulating the growth of a series of networks while varying the radius and maintaining the absorption coefficient, it was found that there is a critical radius beyond which no significant changes in the pattern morphology are observed. At the critical radius, only X% of the initial concentration remains, where X is an emprical value. The critical radius can be found through simple algebraic manipulation of Equation 4.3 as seen below:

$$\text{Critical Radius} = \ln(X)/\ln(1-\text{ABS}) \qquad \text{Equation 5.1}$$

From experimental observation, an X value of approximately 25% appears to provide good system characterization. When the critical radius is plotted against the absorption coefficient, the critical radius increases exponentially with a decreasing absorption coefficient (Figure 5.12), a straightforward result that correlates to the processing time (Figure 5.13). The

critical radius is an extremely important parameter in understanding pattern morphology. A growth simulation with the radius set to the critical value yields a morphology that is independent of matrix size. The critical radius is proportional to the average clump or cluster size, though the relationship is not linear. Using this relationship, a very large cluster size would be expected from a very small absorption coefficient; a prediction verified by the 6400 neuron network seen in Figure 5.14 not only shows that large neural clusters are possible with this model, but also demonstrate its potential for up-scaling.

5.1.6.3.1 Modeling the Cellular Absorption Mechanism

There are actually two possible mechanisms that can control ligand diffusivity. The first model assumes that ligands are absorbed through the cell membrane and are utilized to some capacity within the cell. This first model suggests that ligands can easily enter the cell, but once inside, are readily consumed and therefore have difficulty leaving through the other side. The second mechanism assumes that ligands have tremendous difficulty entering the cell and can only migrate between cells through the extracellular fluid. By the second model, a low diffusivity would suggest that the ligand's ability to migrate between cells is limited. Either or both of these mechanisms may be applied; however, from the standpoint of creating an artificial system, the first method is easier to implement. The first method has been implemented by defining the rate of ligand consumption to be proportional to a gene value. The magnitude expression, in this study, was derived by considering a two dimensional system comprised of 11 cells in a row, numbered from left to right, where the left-most cell is secreting ligands at concentration $[C_0]$. Assuming that all cells consume 90% of all ligands that enter their cell membranes, the concentration entering the right-most cell can be calculated as:

$$[C_{10}] = [C_0](1-0.90)^9$$

where the subscripts denote the cell number from left to right and the exponent 9 refers to the number of cells through which the ligand signal must pass.

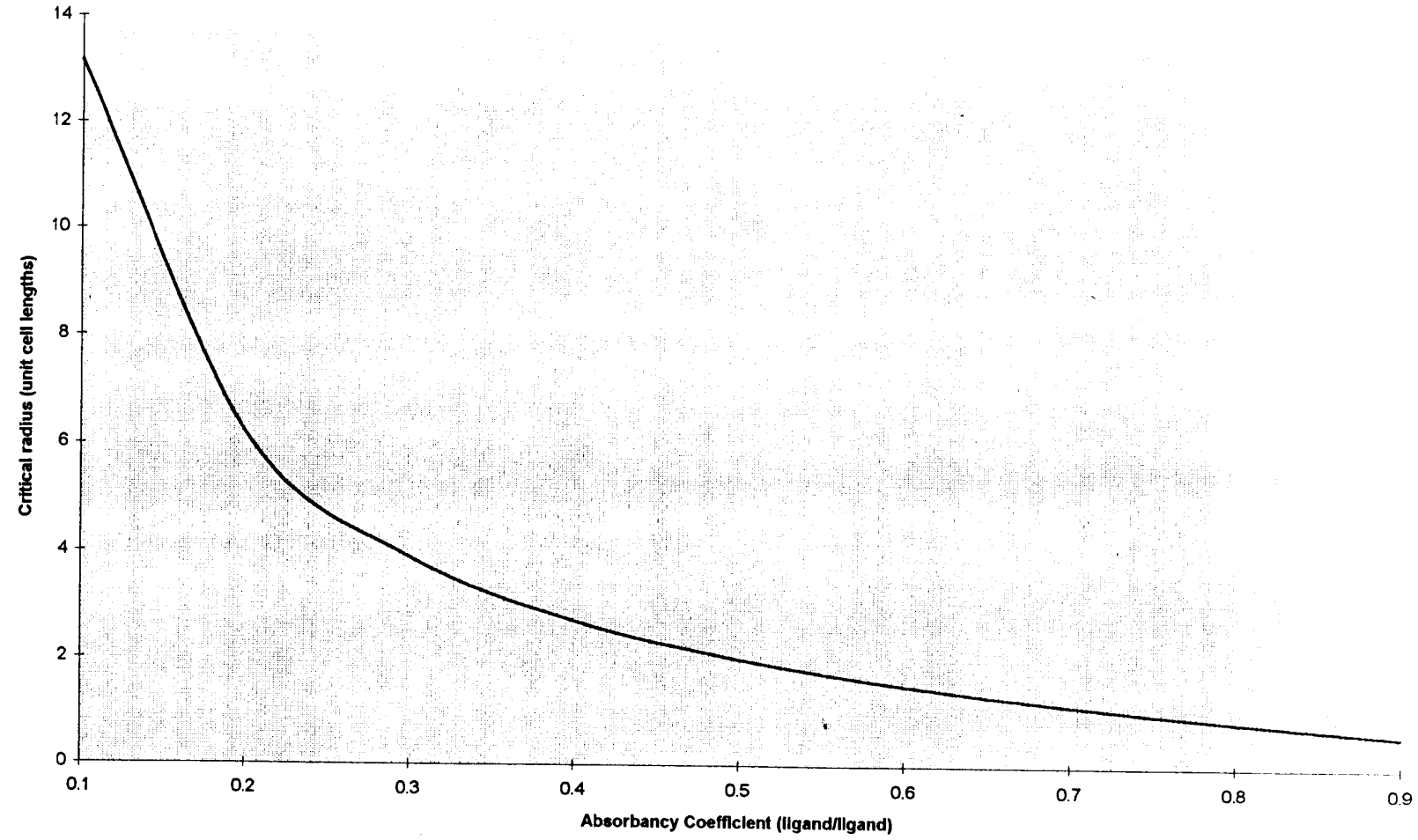


Figure 5.12 Critical Radius vs. Absorption Coefficient.

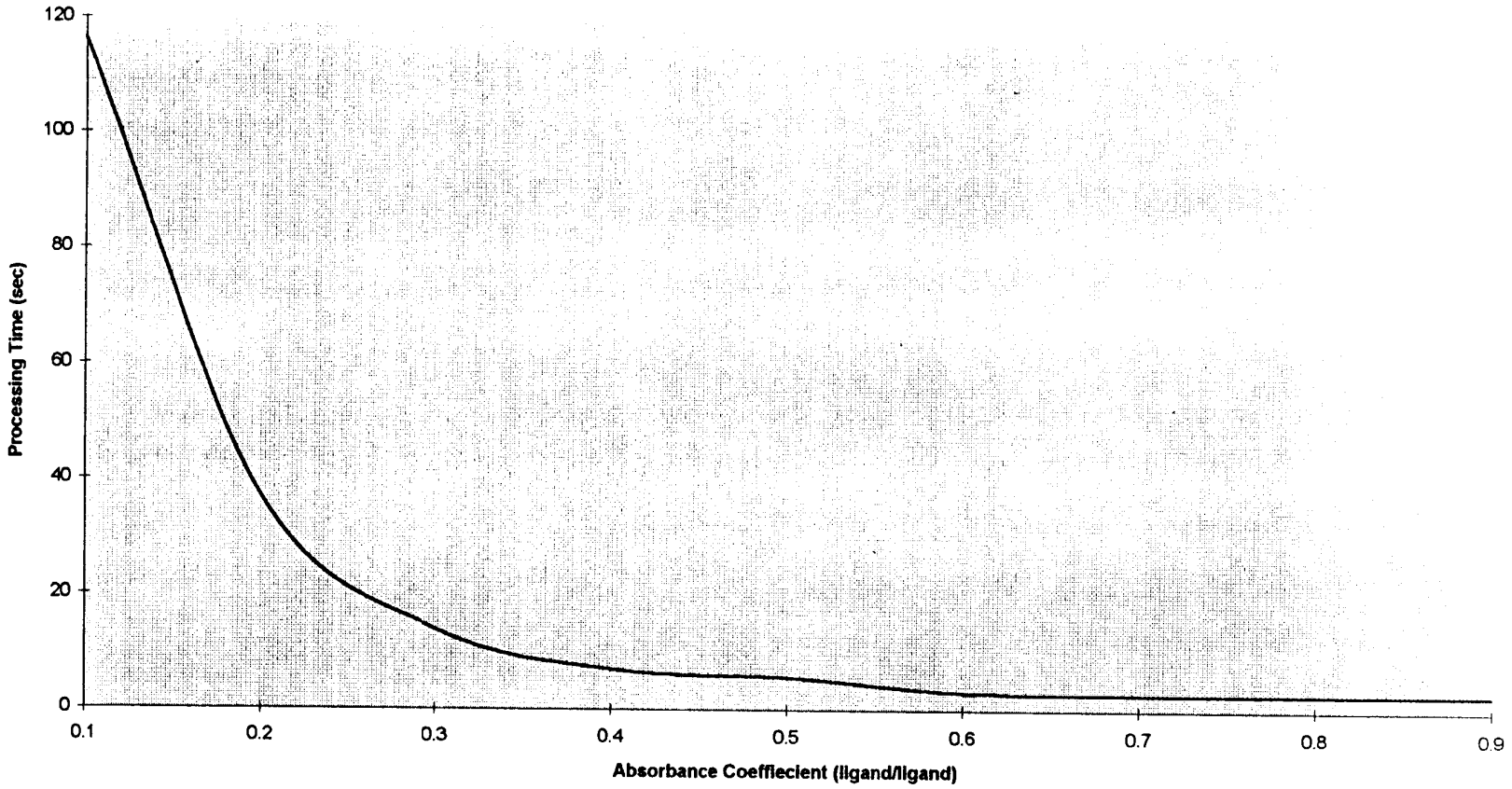


Figure 5.13 Processing Time vs. Absorption Coefficient for Critical Radius

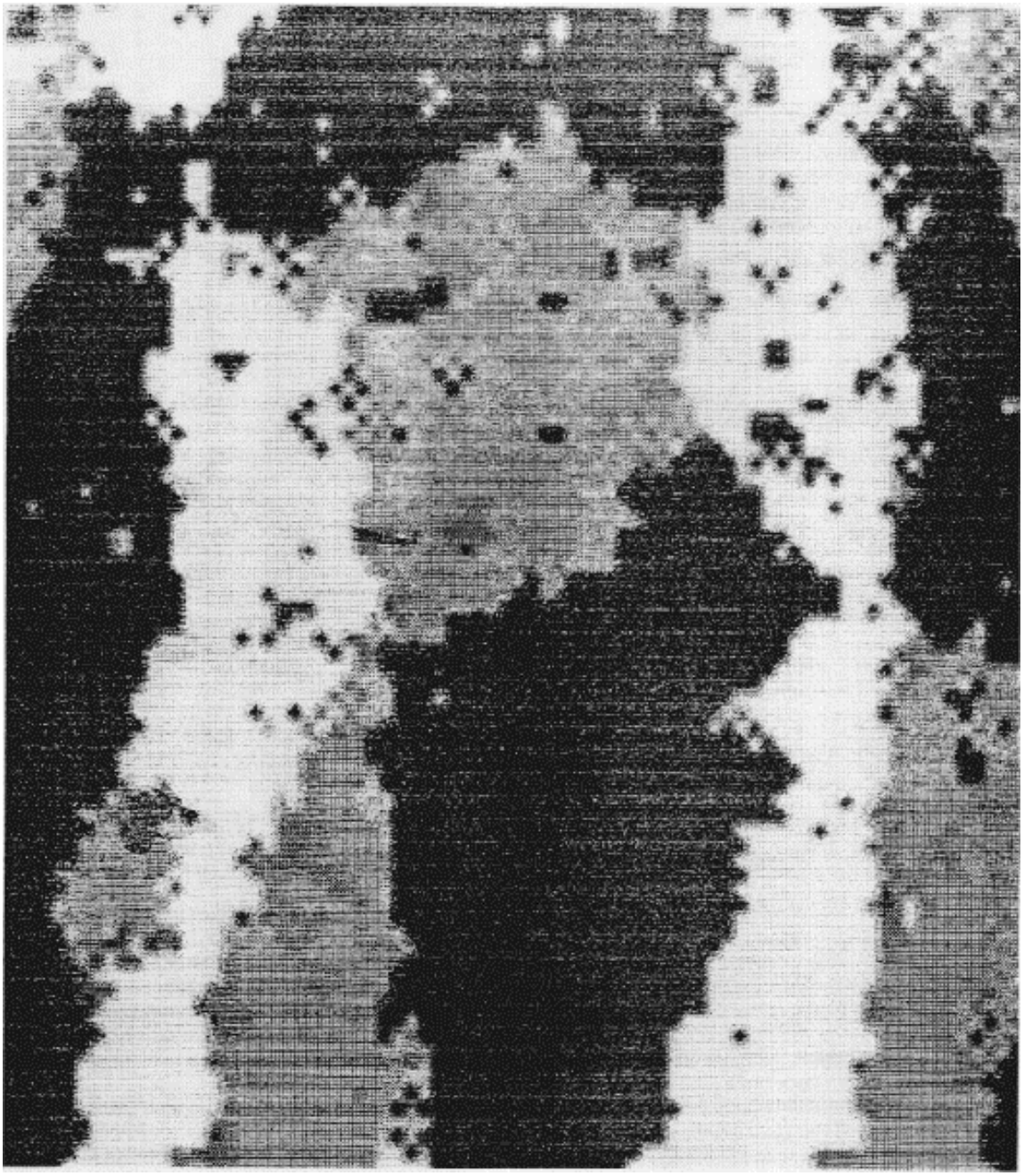


Figure 5.14 6400 Neural Network for 0.01 Absorption Coefficient

5.1.6.3.2 Biological Local Absorption: A Comparison of Simulation Studies to Experiment.

It has been suggested that some developing cells will absorb certain ligands effectively, biasing local concentrations thereby affecting the development of their neighbors. The computer simulation was modified from the initial condition, where a mature cell secretes ligands other than those associated with its type and does not secrete its own specific ligand, to a condition where the mature cell absorbs ligands of its own kind. Several simulations were run, modifying the amount of ligands that the mature cells were able to absorb, and revealed some surprising information. In the simulations where all cells were programmed to have the same proportional absorption, no change in the resultant assignment of cellular position or type was observed. Regardless of the value of the absorption, the same exact pattern would develop. From this it was believed that the absorption was acting as a scalar value in the overall scheme of the differentiation process, changing only the magnitude of the gradient, but not the direction of the vectors. In order to observe a change in the resultant differentiated pattern, it was necessary to assign different values of absorption for the different cells. The first simulation that did reveal a change in the embryonic development, the stage used to test this model, was observed when only one cell type in a given stratum was allowed to absorb ligands. The overall effect of this mechanism is equivalent to lowering the inter-stratum fraction of the cell that absorbs its own ligand type. It is believed that although this mechanism is analogous to the biological model, it has the identical mathematical effect as shifting the cell type probabilities, at least in the current model. This mechanism is therefore deemed unnecessary because it provides no new functionality while increasing computational time. The results of this simulation are quite interesting as they suggest that if absorption was observed as a mechanism that did shift the differentiation process in the biological model, then the cells in which it was observed were probably absorbing comparatively more than the rest of the cells in the tissue.

5.1.7 Accuracy of Methods

When this study began, traditional statistical analysis was perceived as being very useful for characterizing the system; however, as knowledge and insight was gained about the behavior of the simulation model, it became apparent that traditional measures could not quantify the characteristic patterns that were developing. It is now believed that the pattern morphology, and specifically the parameters that affect that morphology, are of the greatest significance in terms of characterizing the behavior of the overall system. While statistical approaches may be applied

to characterize the patterns by measuring the number of elements associated with a clump of similar neurons, the aspect ratio of the clumps, or the proximity of clumps to each other, etc., these approaches should be viewed with a degree of skepticism as the optimal type of pattern is not yet known and the premature development of measures to characterize these patterns may inadvertently lead future research in the wrong direction.

A full statistical evaluation of this system has not been conducted; however, some of the data from the simulation study has been analyzed in reference to the original parameters. The transcription error is defined to be the difference between the values obtained from the output of the simulation program and the desired value divided by the desired value. The transcription error graphs were produced only to determine if the system produced structures that are characteristic of the intended design. Future work can be conducted to characterize the system to develop relationships between the gene values and those yielded from the growth simulator. The presence of transcription error was predicted (recall the discussion on the need to separate the parameter from its gene value) and in no way indicates that the performance of the program is inconsistent with the model, rather the constant error supports the assumptions and predictions of the system's performance. The reason that transcription error is expected is that there is no mechanism that directly forces the system to produce an output identical to the input. Once the program begins to run, all global information is converted into a series of local parameters that collectively interact to produce the output. Thus once the global parameter has been converted the system has no way to maintain the integrity of the global parameter.

Transcription error has one additional use; it can be used to characterize the domain that the simulation program performs best. The following section has a variety of graphs that depict transcription error and processor time versus specific parameters. Four systems were studies. The first and second systems have unequal cell type fractions with 3 and 5 cells respectively. The third and fourth systems have equal cell type fractions with 3 and 5 cells respectively. Each of the parameters that govern the cell growth simulation program was tested over its respective full range for the 4 systems.

5.1.7.1 Growth Rate

The growth rate term functionally determines the number of iteration between the minimum and maximum growth rate values. The growth rate term does not have a strong effect on error. Growth rate does however regulate processor time. Since growth rate does not seem to have an effect on error, but does effect pattern formation, its value should be selected based on the desired morphology.

5.1.7.2 Minimum And Maximum Growth Rate

Minimum and maximum growth rates determine the starting and final values of maturing cycle. They are mostly set points. Neither term seem to have a strong effect on either processor time or transcription error. Figure 5.19 through Figure 5.26 show an essentially constant level of transcription error for the minimum and maximum growth rates. Their effect, much like that of the growth rate (above), term is primarily on pattern formation.

5.1.7.3 Absorption Coef With Constant Radius

The absorption coefficient is the most powerful term for controlling the cell growth simulation program. There are many facets to these terms, which require three separate sections to describe. The absorption coefficient determines how much of the original concentration is absorbed by each cell. The first set of figures, (Figure 5.27 through Figure 5.30) depict the effect of the absorption coefficient on transcription error and processor time while holding the radius constant Under such conditions absorption is found to have no effect on processor time and a slight effect on error. Figure *** shows the transcription error versus the absorption coefficient for 5 UCLs.

5.1.7.4 Radius

The interpretation of the effective radius on the transcription error is supported by figure*** which shows the transcription error as a function of radius for a constant absorption coefficient of 0.1. This absorption coefficient was selected because it is known to yield low transcription error. This figure shows that the transcription error maintains a constant level for radii above 2. Below a radius of 2, there is a marked increase in transcription error, which is again attributed to insufficient stimulation from distant neural elements. Radius has a strong effect on transcription error, but it is a fake effect. The radius term is not real; it was introduced in the model only to limit computational time. The radius term is somewhat obsolete. As the development of the method proceeded the radius term was essentially replaced the critical radius term discussed in the next paragraph with limits the computational time base on the absorption coefficient. This section is included to help illustrated that the number radius is the dominant term in regulating processor time, but has little effect on error. Figure 5.31 through Figure 5.34 clearly show that there little change in error above a radius of 2. Below two, the error increases rapidly as too few cells are incorporated in the concentration calculations.

5.1.7.5 Critical Radius

Figure 5.35 through Figure 5.38 shows the error associated with the absorption coefficient when the radius is dynamically adjusted to the critical values. Recall the critical value is determined to be a value above which less than 10 percent of the original concentration is measured. The error is constant at an absorption coefficient below 0.4, then gradually increases to a critical value around 0.75, where the transcription error begins to increase dramatically. When the critical radii of these three points are reviewed, it is found that the system is stable when the critical radius is over 2.5 UCLs, unstable with a critical radius below 1 UCL, and partially stable between these values. This behavior can be directly linked to the neural network lattice. A critical radius below 1 UCL does not allow the significance signal to be received from neighboring neurons, thus the cell is developing essentially independently from the rest of the network. In the partially stable region the cell is receiving significant contributions from neighboring elements; however, less than six cells are contributing which apparently is too few to produce ligand concentrations representative of the rest of the network. As the radius is increased, the concentration at any one given cell becomes representative of the rest of the network because the network can be viewed as being fully interconnected through overlapping gradient fields. An absorption coefficient below 0.2 (6.21 UCL critical radius) is seen to increase the amount of transcription error. This observation is actually a manifestation of the critical radii being approximately 25% of the length of the neural grid (24 elements/side). Under this condition, edge effects become pronounced, and a representative ligand field cannot develop.

The last series of graphs are perhaps the most revealing. These graphs depict transcription error versus the ratio between cell type fractions. The results of this reveal a surprising sweet spot of performance around a ratio of .3 where transcription error is virtually non existent. This sweet spot is believed to occur when the error associated with under and over transcription cancel out. Over transcription describes when more than the fractions develop. Over transcription is driven by a large cell type fraction dominating the system. Under transcription error occurs when a cell type fraction is disproportional small and fails to promote the development of the expected fraction of cells. Below a cell type fraction ratio of .3 under transcription is the dominant source of error. Under transcription error has the interesting property of being able to completely suppress cellular development for extremely low fractions, seen clearly in Figure 5.39 through Figure 5.41. Above a cell type fraction of .3 over transcription error dominates and seem to reach a near steady-state value above ratios of .75. The significance of Figure 5.39 through Figure 5.41 is they show repeatable, consistent behavior over a large operating range, suggesting a useful range between cell type fractions above .2.

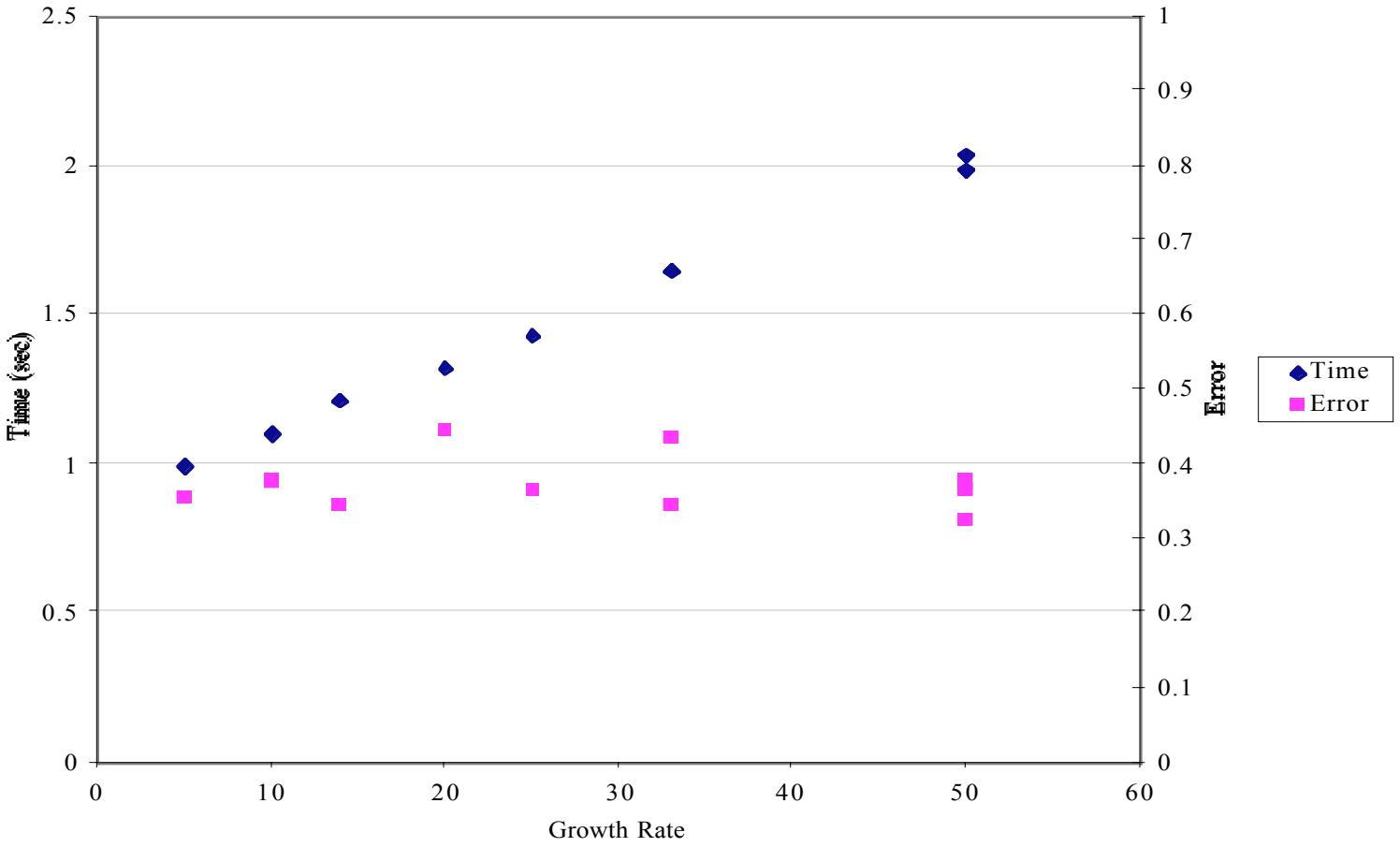


Figure 5.15 Transcription Error & Time vs. Growth Rate for Three Cell types with Unequal Cell Type Fractions

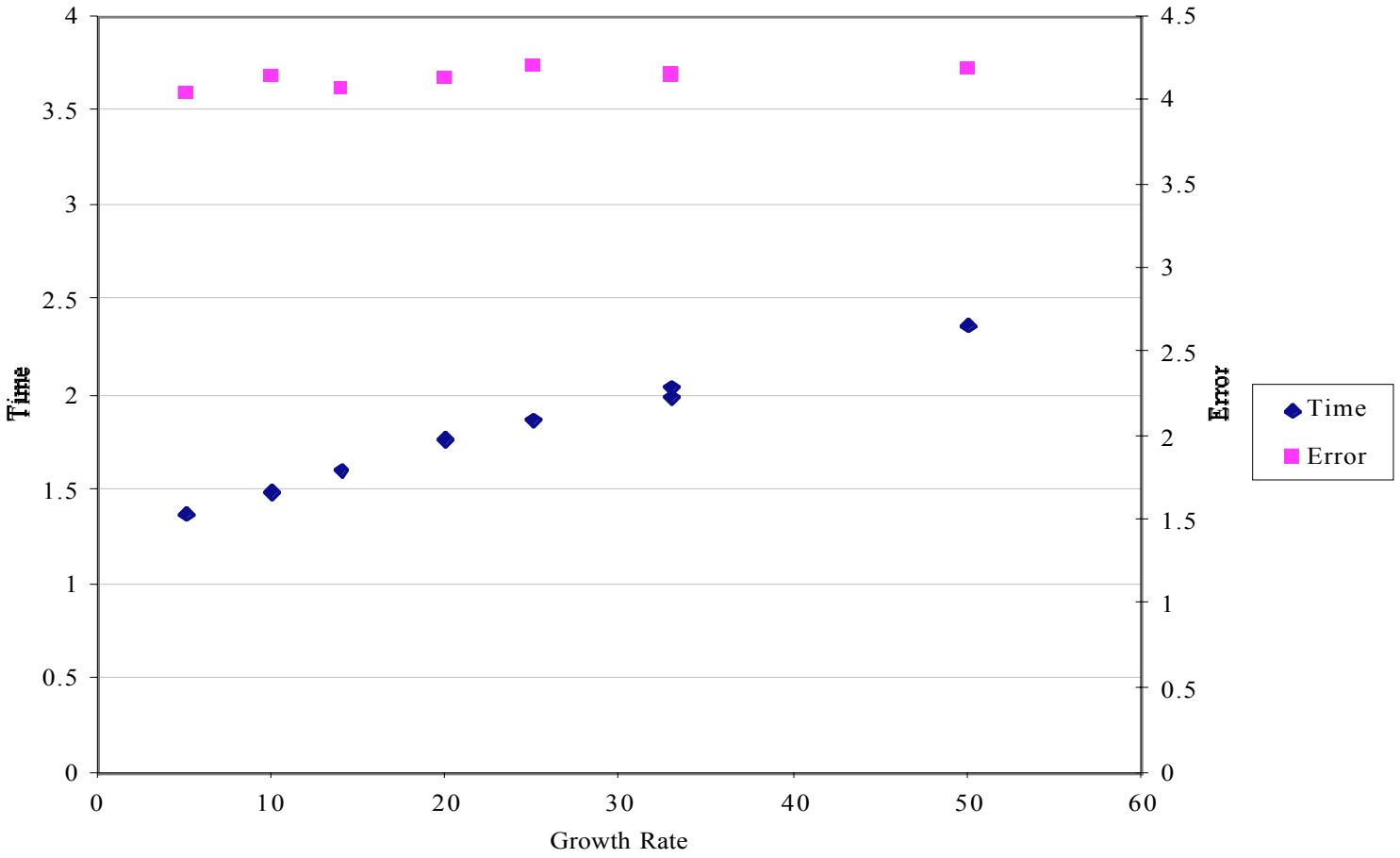


Figure 5.16 Transcription Error & Time vs. Growth Rate for Five Cell types with Unequal Cell Type Fractions

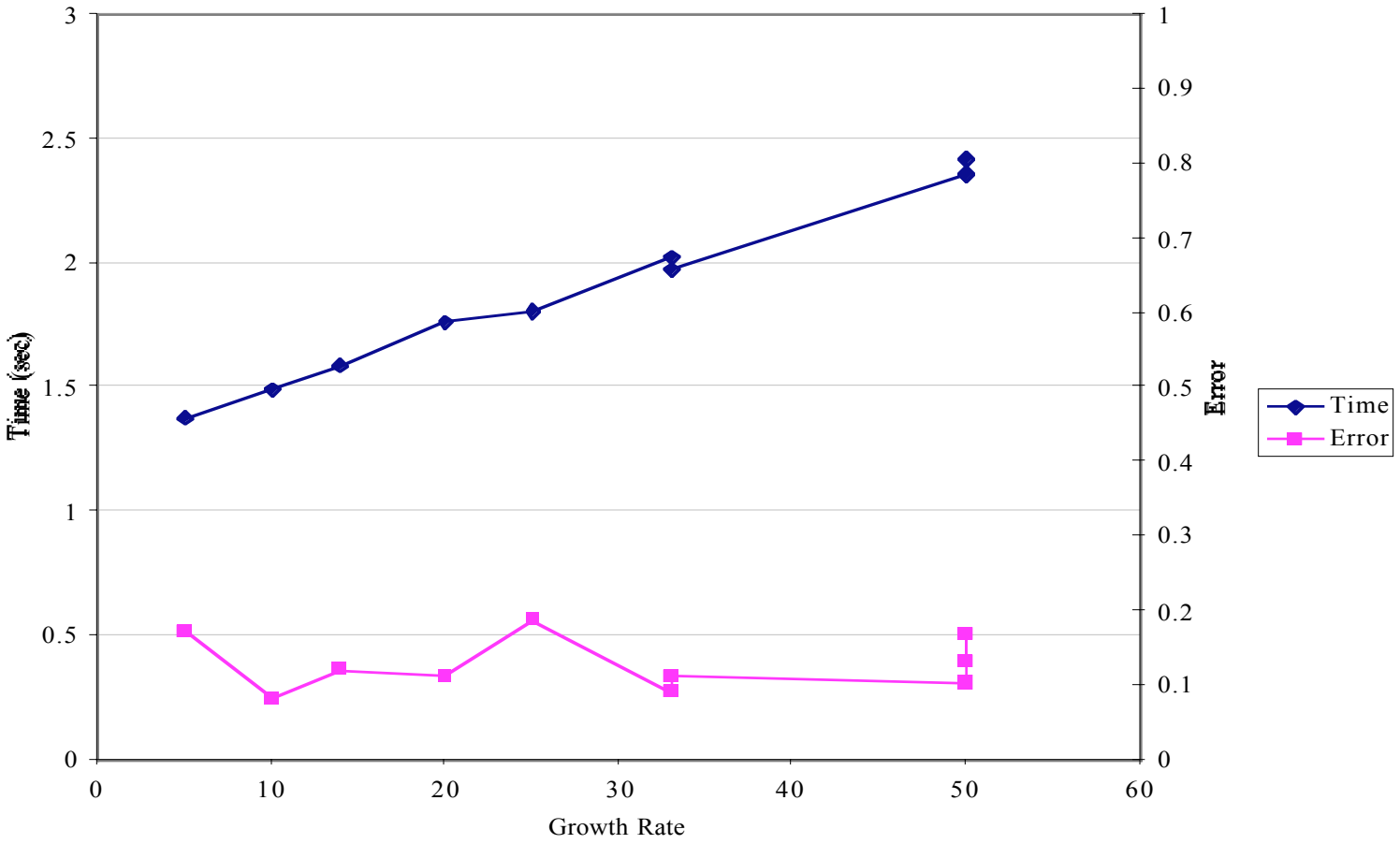


Figure 5.17 Transcription Error & Time vs. Growth Rate for Three Cell types with Equal Cell Type Fractions

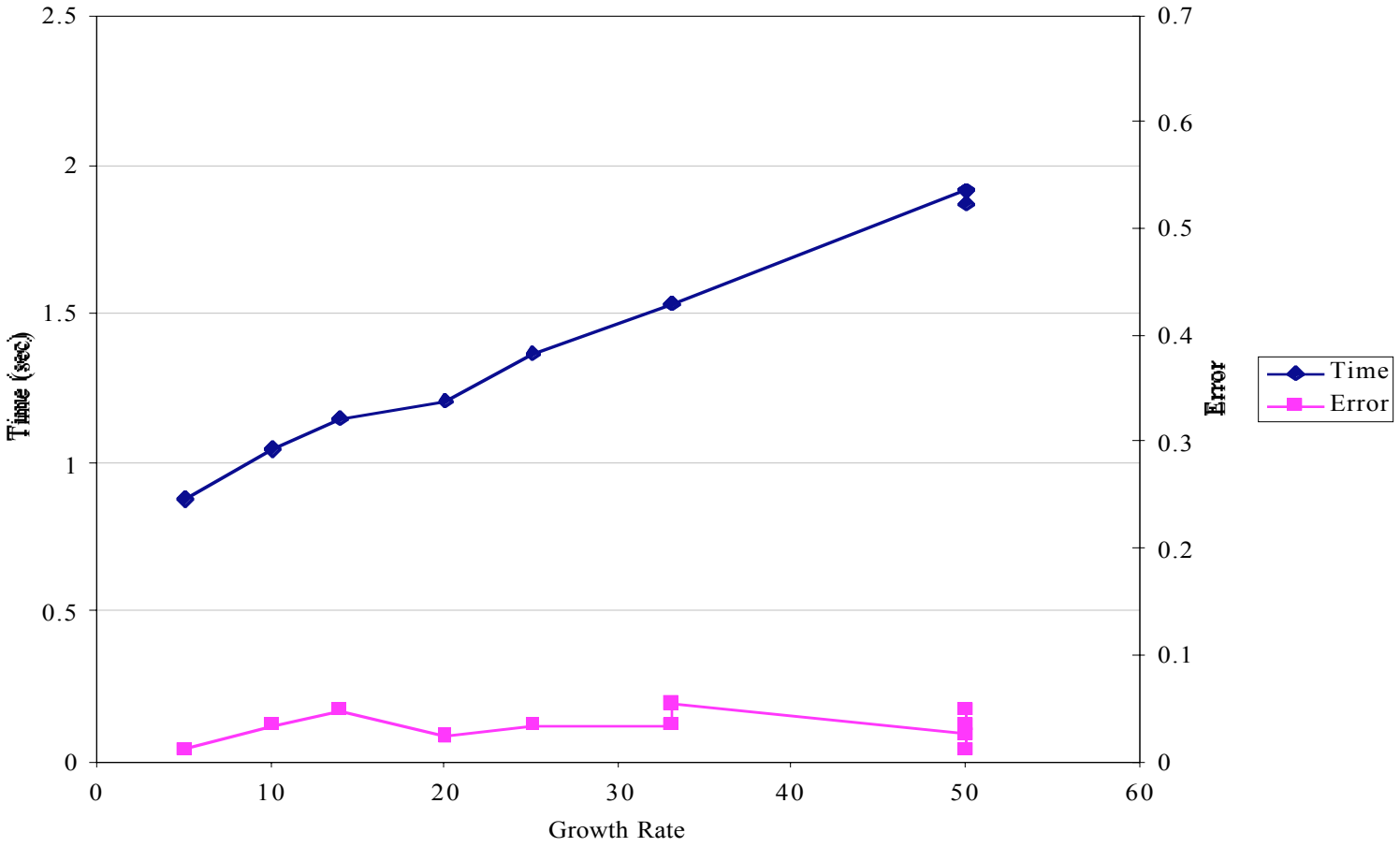


Figure 5.18 Transcription Error & Time vs. Growth Rate for Five Cell types with Equal Cell Type Fractions

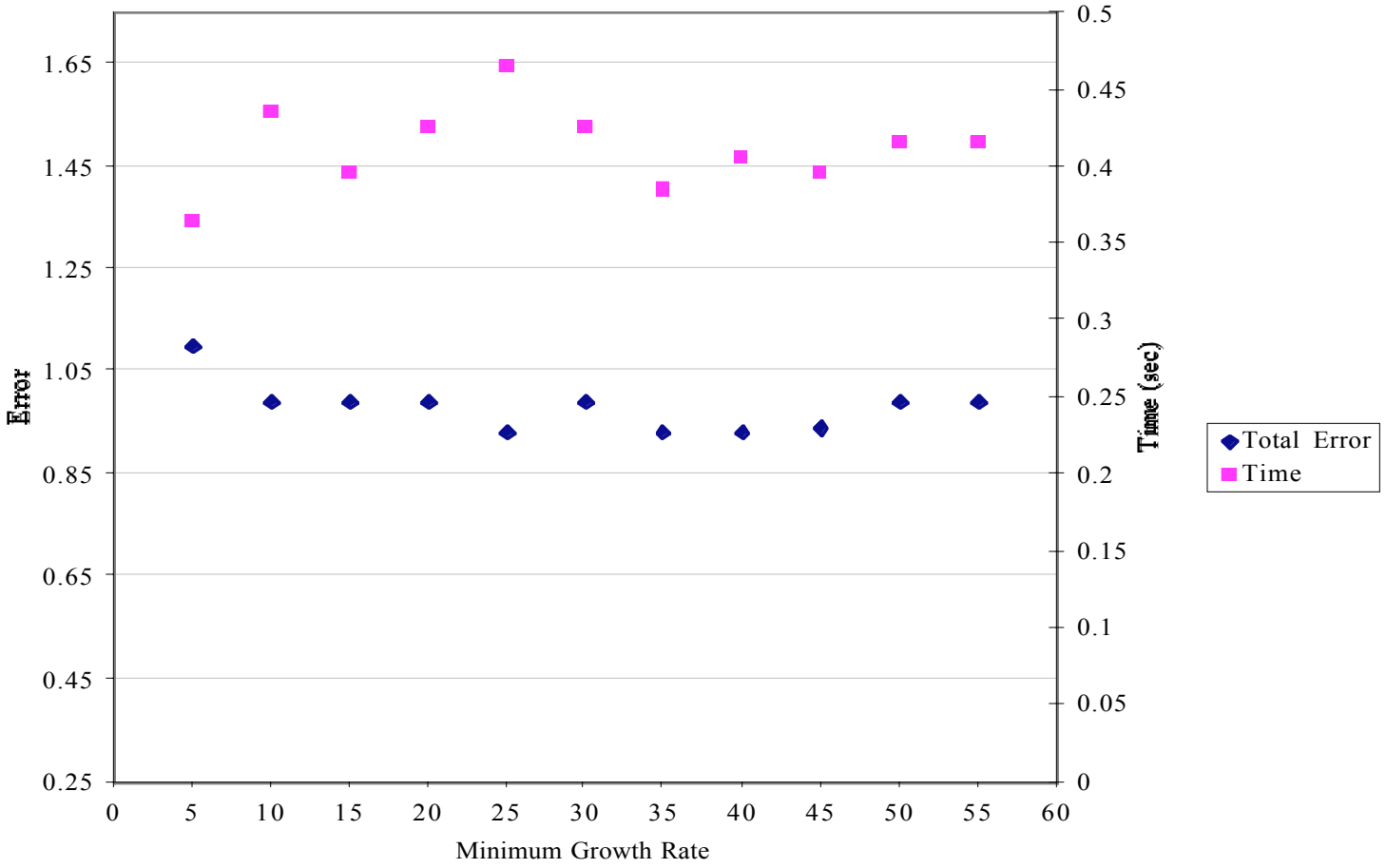


Figure 5.19 Transcription Error & Time vs. Minimum Growth Rate for Three Cell types with Unequal Cell Type Fractions

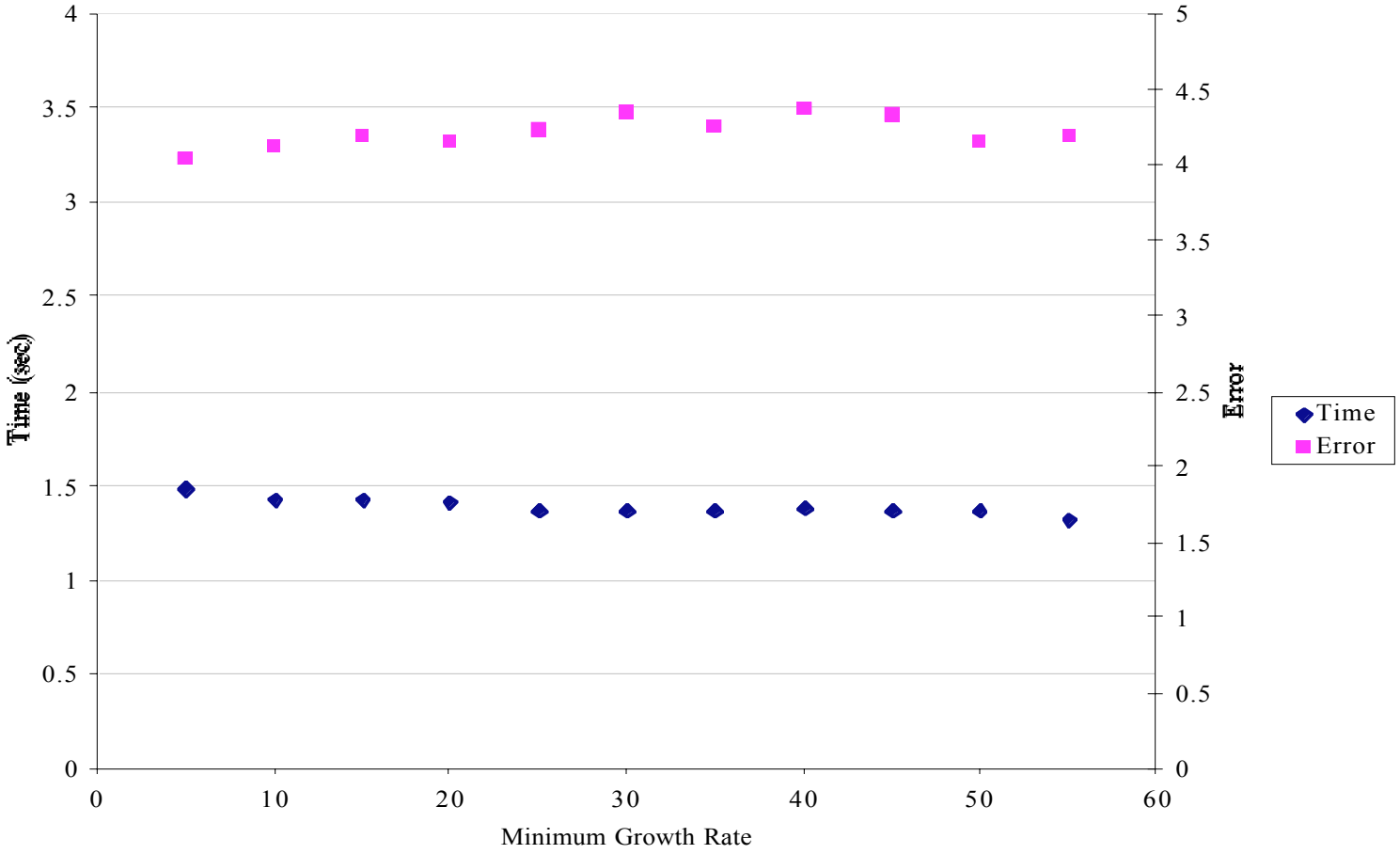


Figure 5.20 Transcription Error & Time vs. Minimum Growth Rate for Five Cell types with Unequal Cell Type Fractions

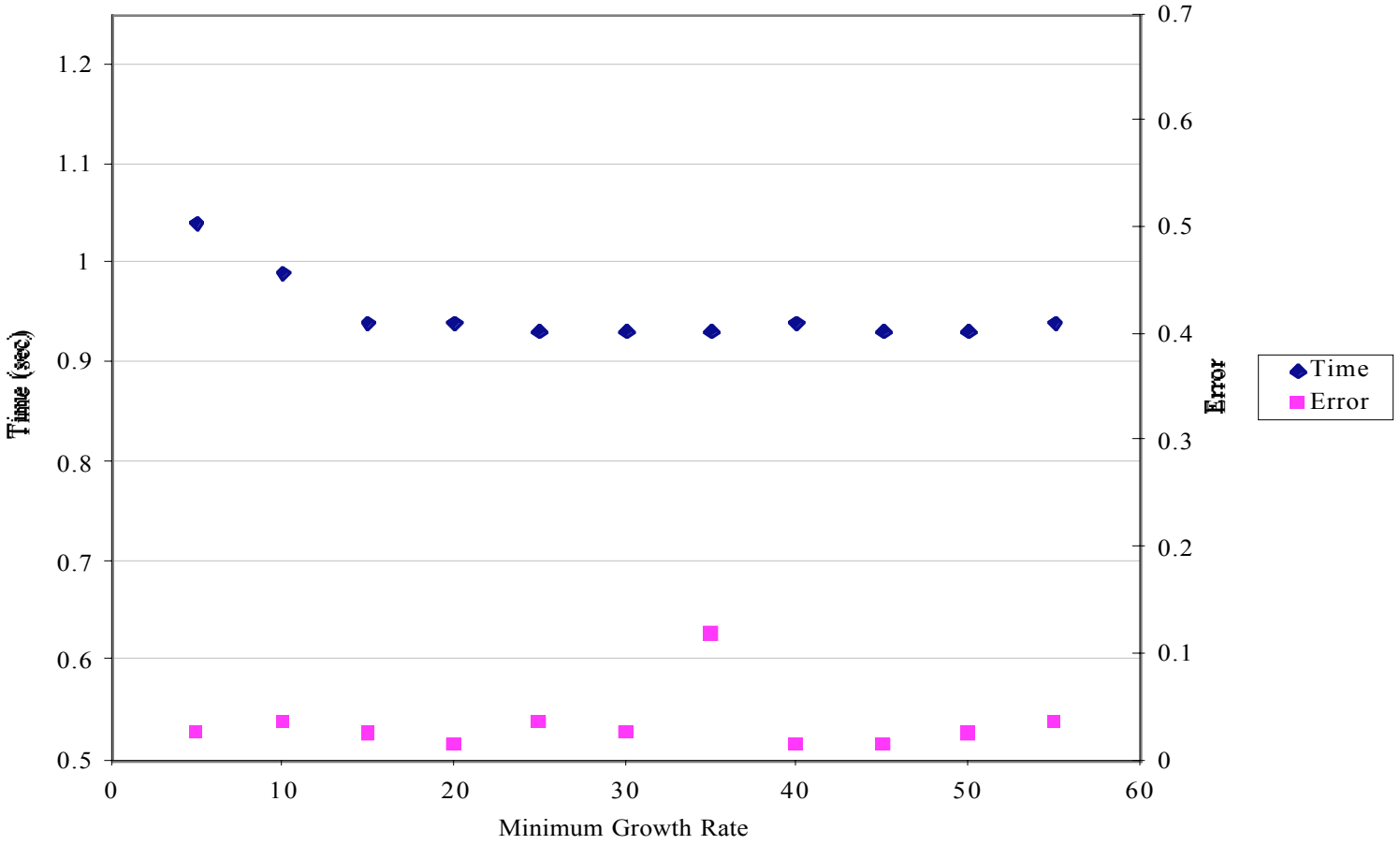


Figure 5.21 Transcription Error & Time vs. Minimum Growth Rate for Three Cell types with Equal Cell Type Fractions

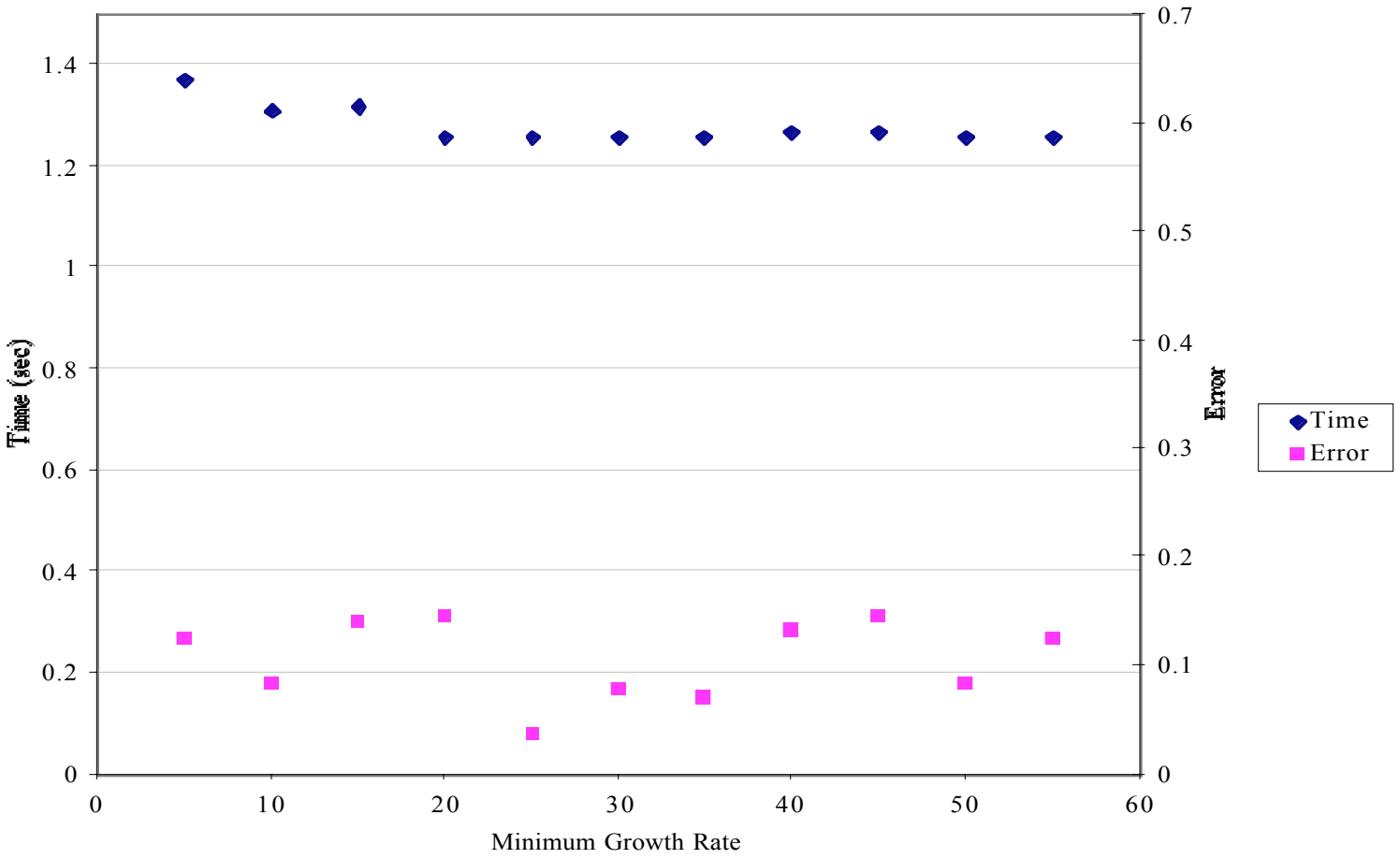


Figure 5.22 Transcription Error & Time vs. Minimum Growth Rate for Five Cell types with Equal Cell Type Fractions

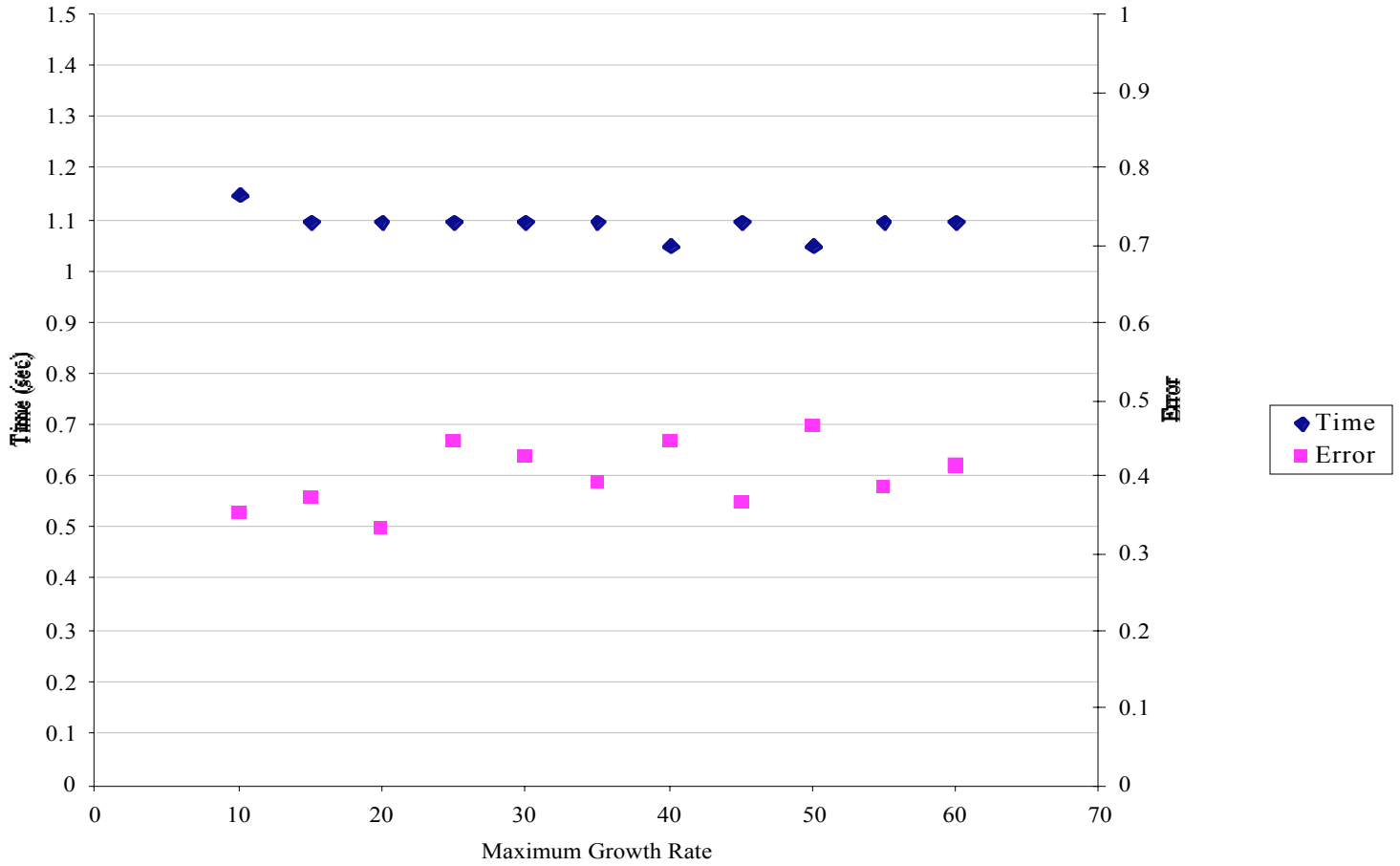


Figure 5.23 Transcription Error & Time vs. Maximum Growth Rate for Three Cell types with Unequal Cell Type Fractions

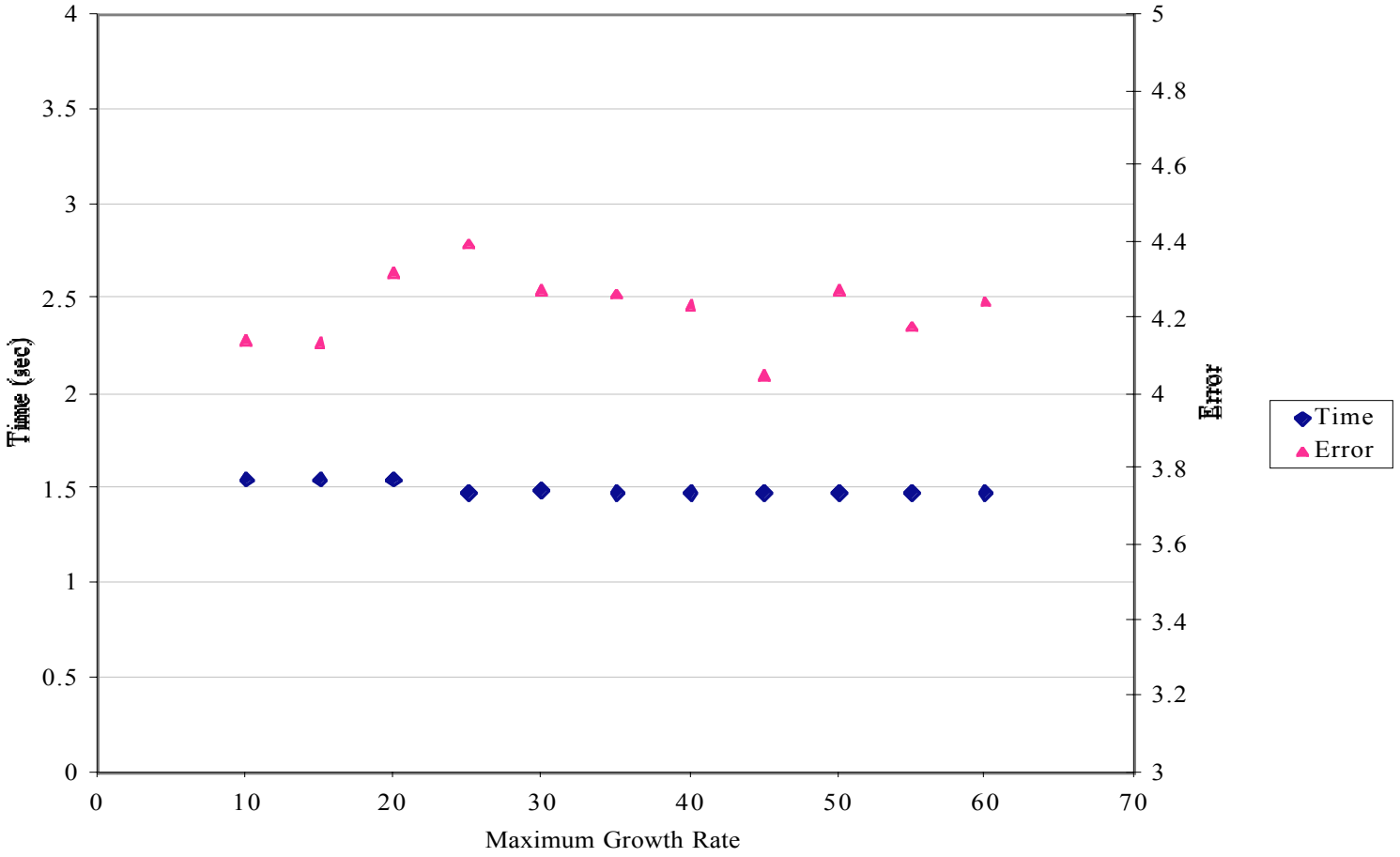


Figure 5.24 Transcription Error & Time vs. Maximum Growth Rate for Five Cell types with Unequal Cell Type Fractions

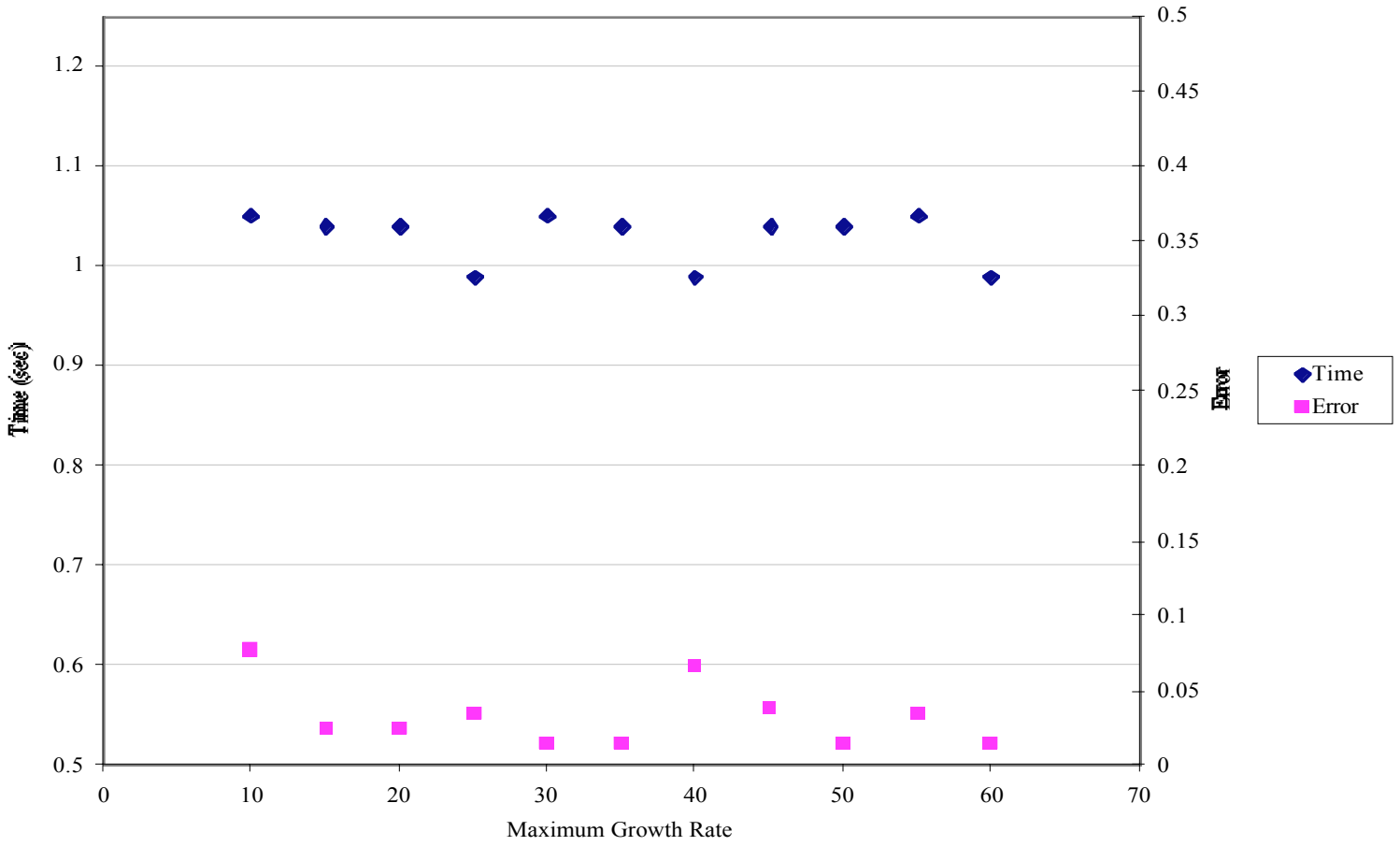


Figure 5.25 Transcription Error & Time vs. Maximum Growth Rate for Three Cell types with Equal Cell Type Fractions

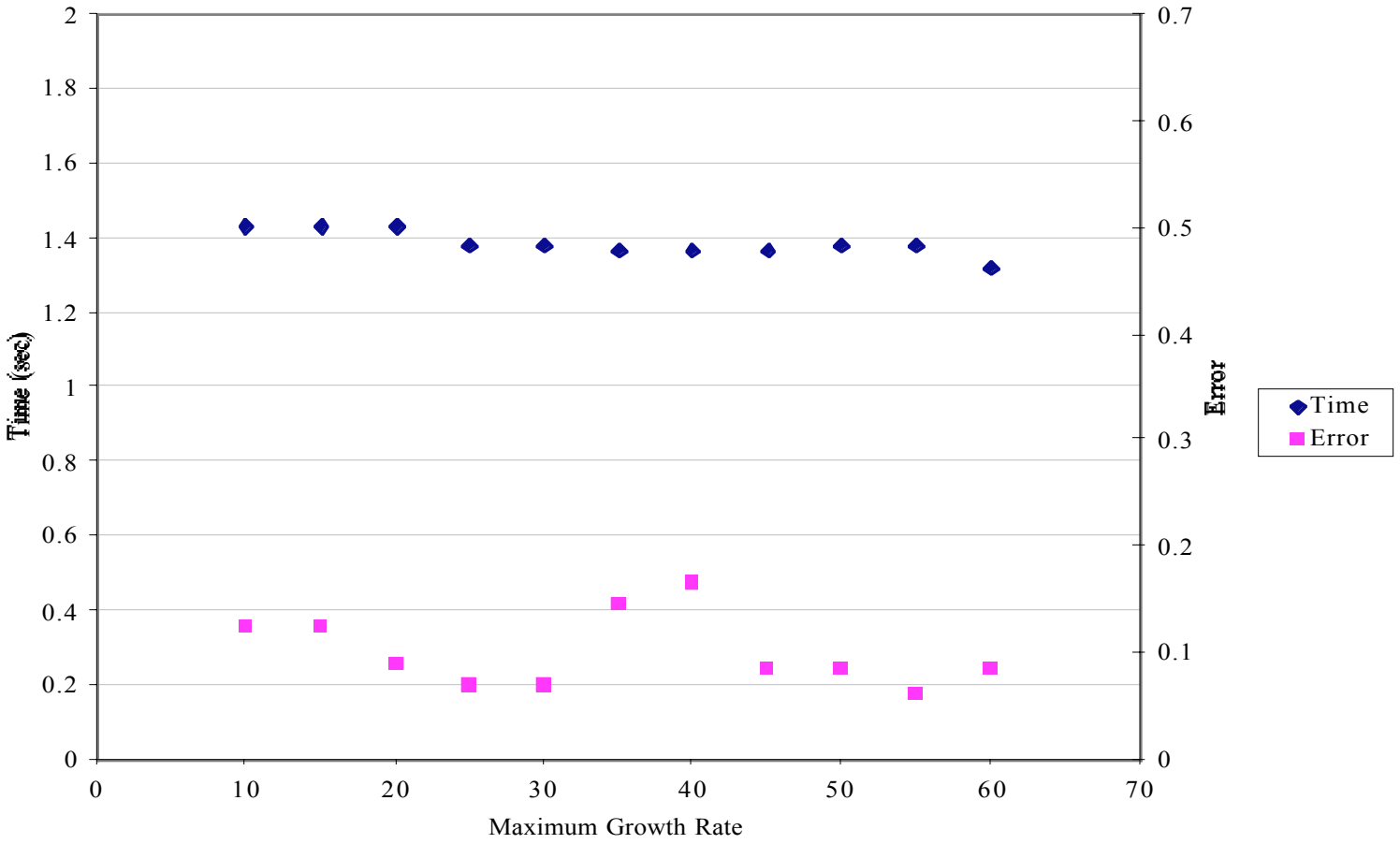


Figure 5.26 Transcription Error & Time vs. Maximum Growth Rate for Five Cell types with Equal Cell Type Fractions

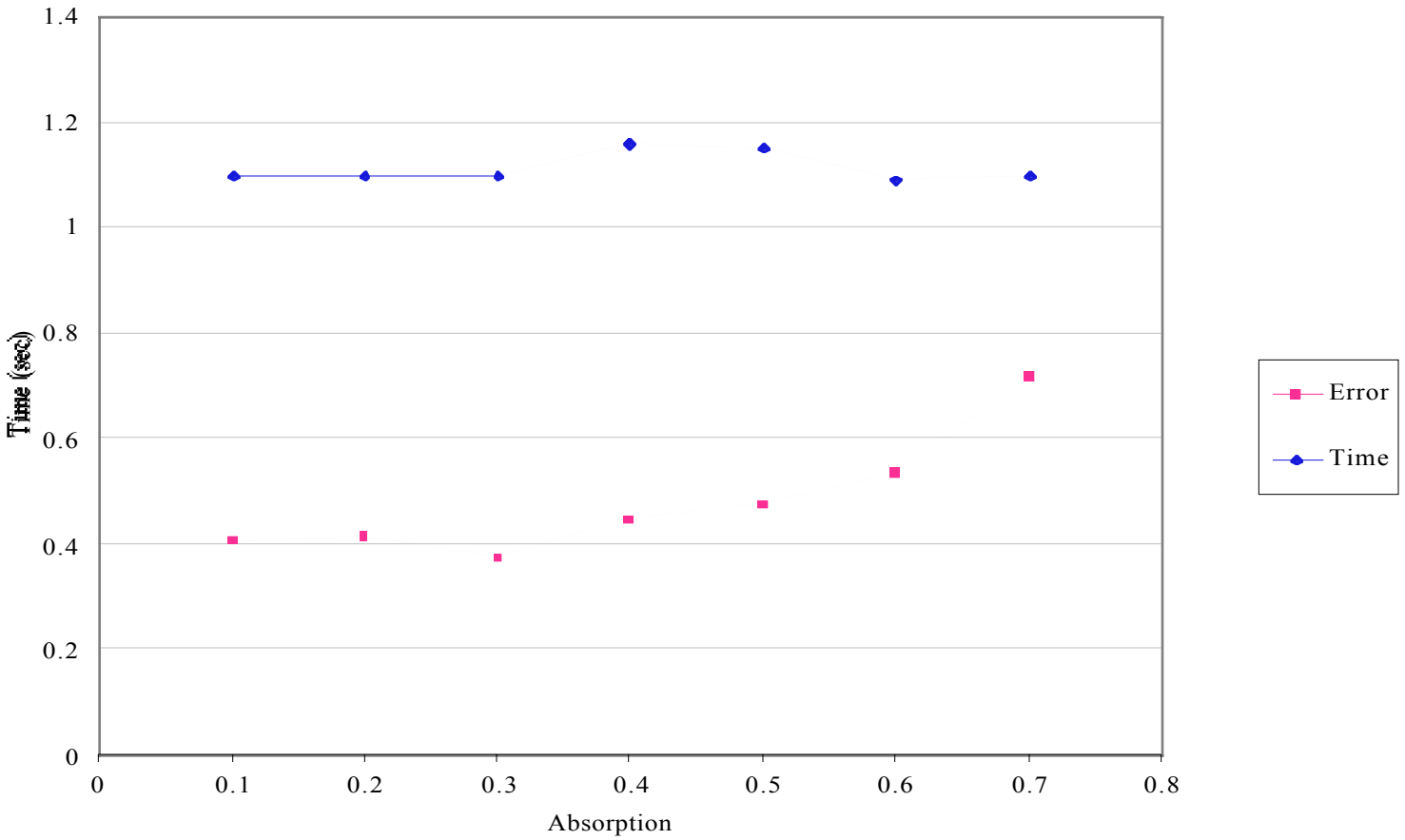


Figure 5.27 Transcription Error & time vs. Absorption Coefficient with constant 5 UCL Radius for Three Cell types with Unequal Cell Type Fractions

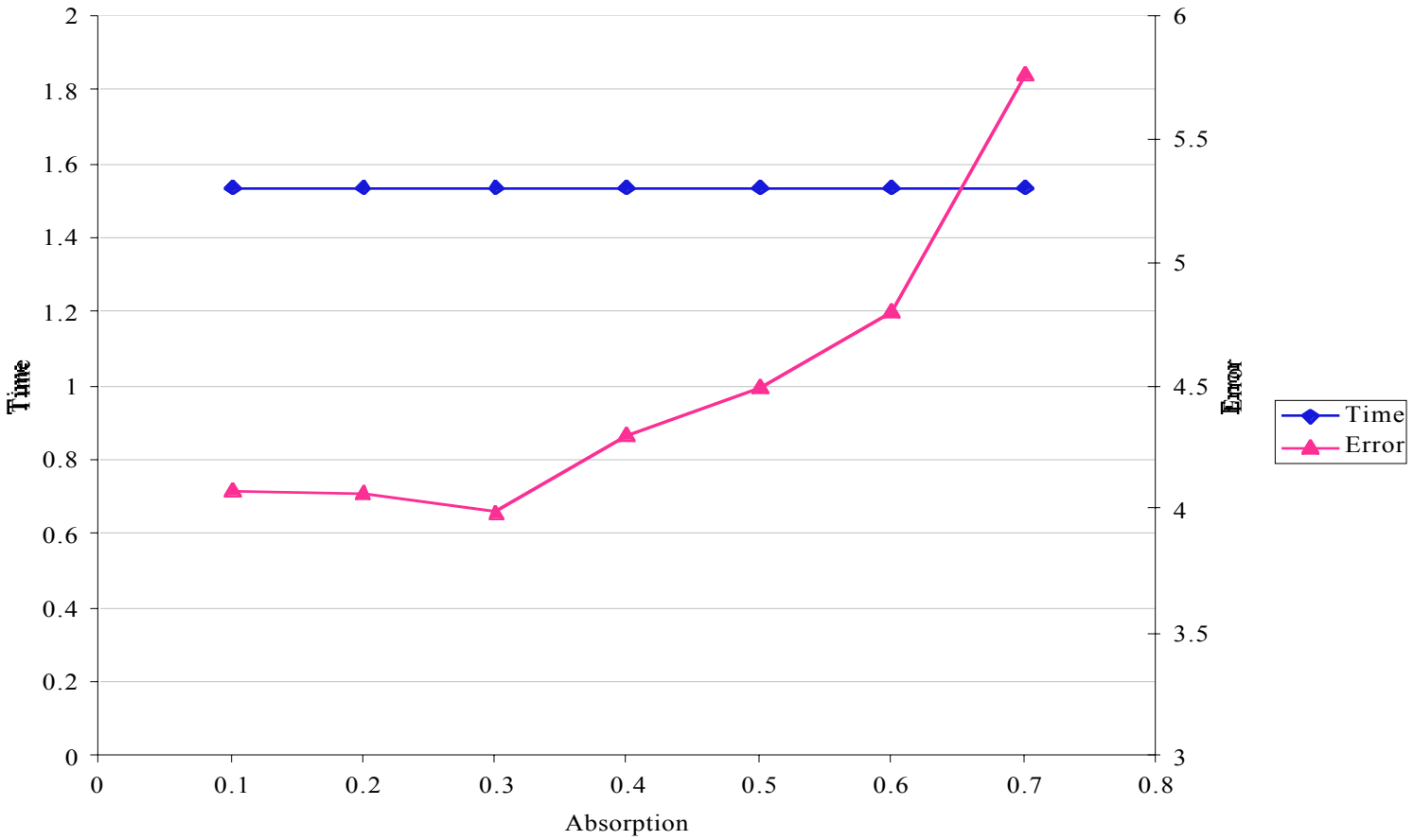


Figure 5.28 Transcription Error & time vs. Absorption Coefficient with constant 5 UCL Radius for Five Cell types with Unequal Cell Type Fractions

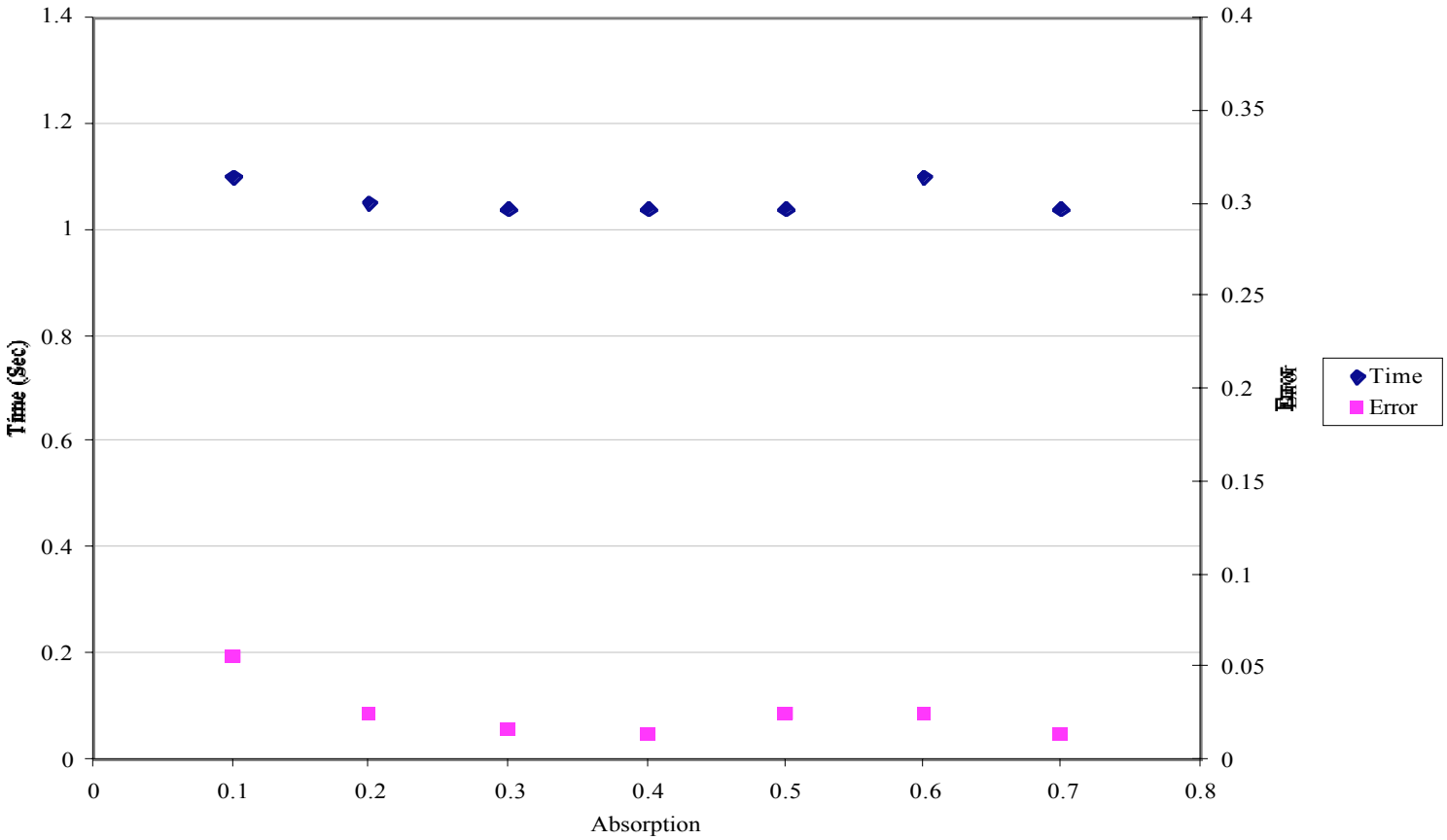


Figure 5.29 Transcription Error & time vs. Absorption Coefficient with constant 5 UCL Radius for Three Cell types with Equal Cell Type Fractions

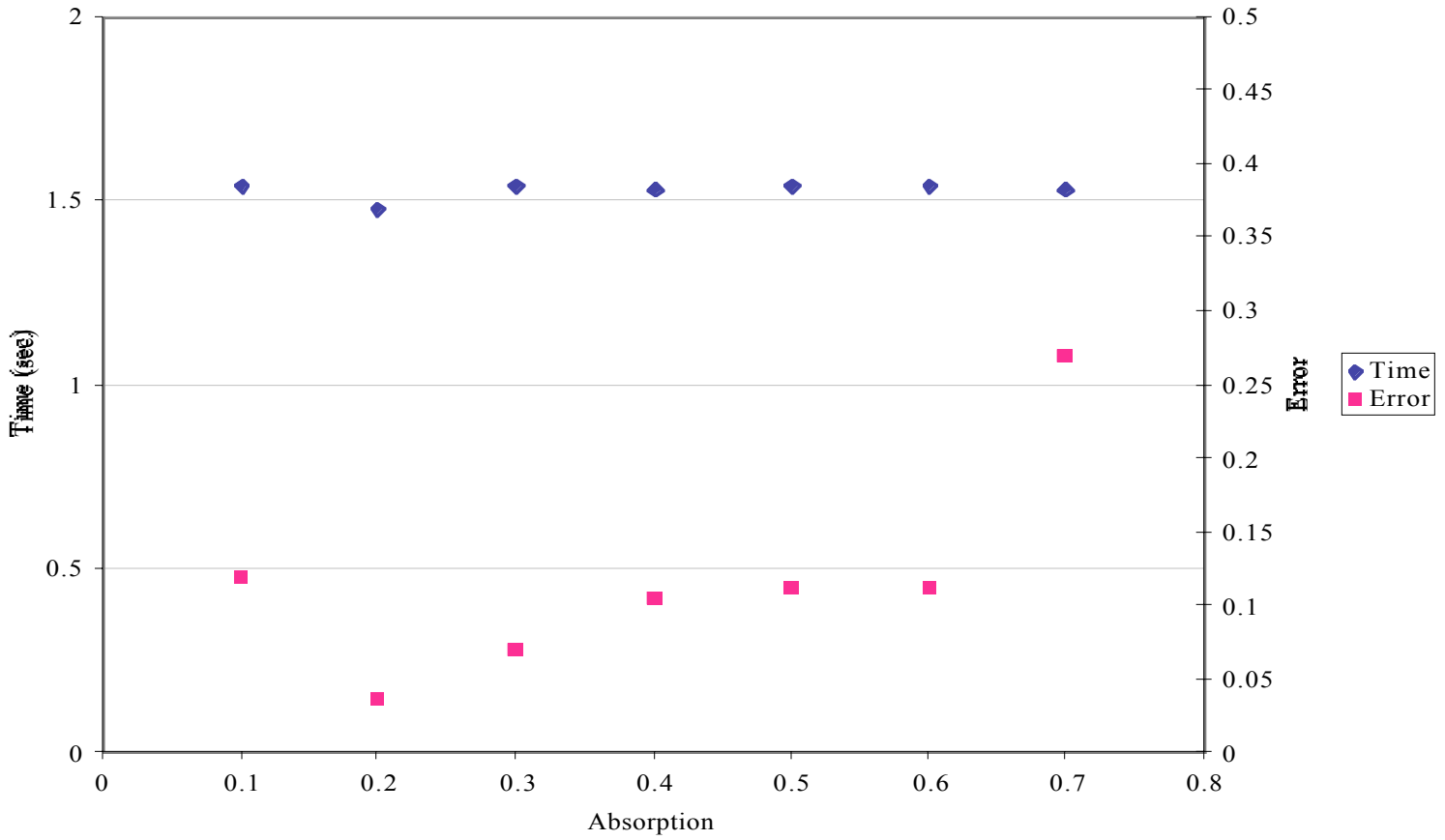


Figure 5.30 Transcription Error & time vs. Absorption Coefficient with constant 5 UCL Radius for Five Cell types with Equal Cell Type Fractions

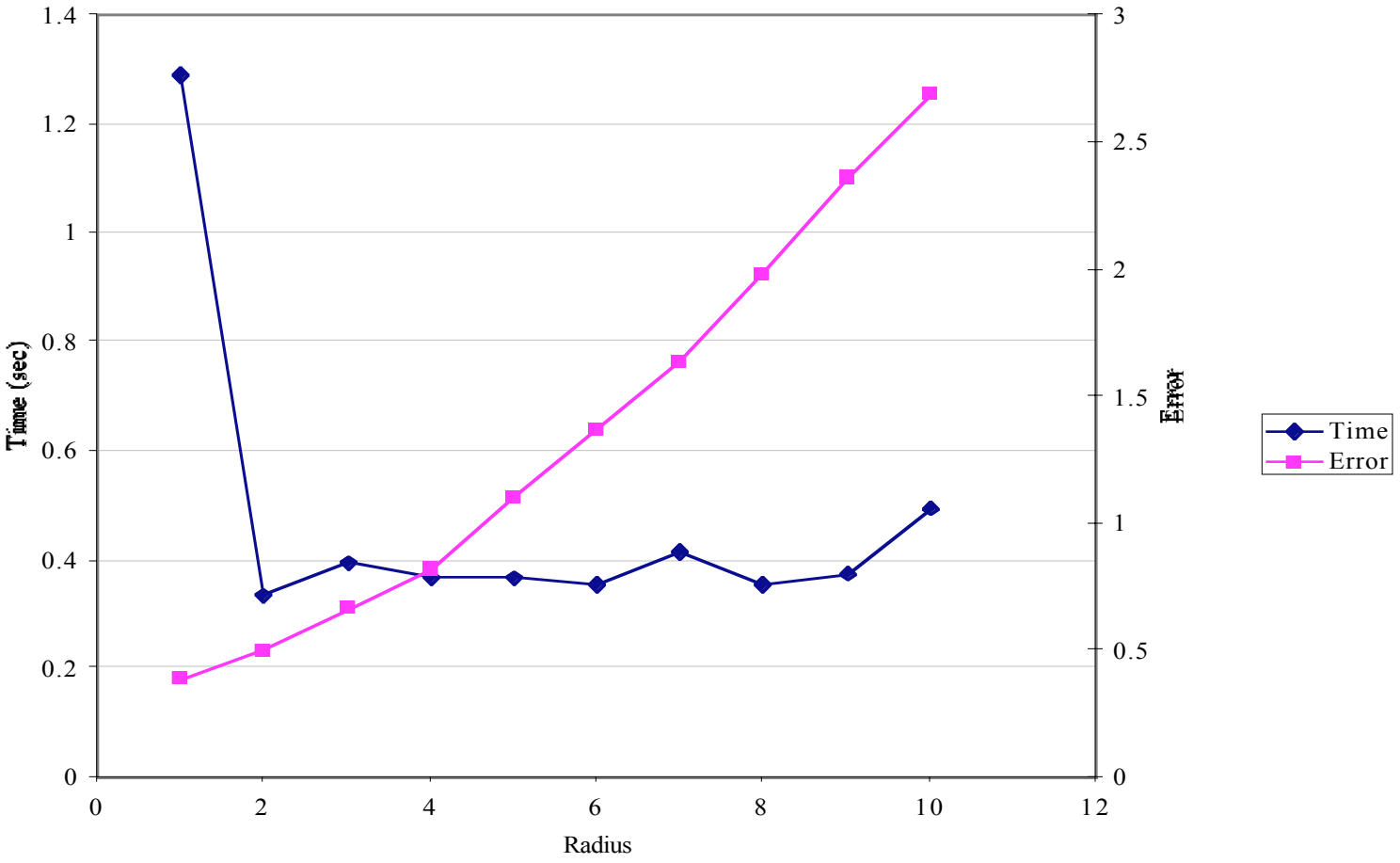


Figure 5.31 Transcription Error & Time vs. Radius for Three Cell Types with Unequal Cell Type Fractions

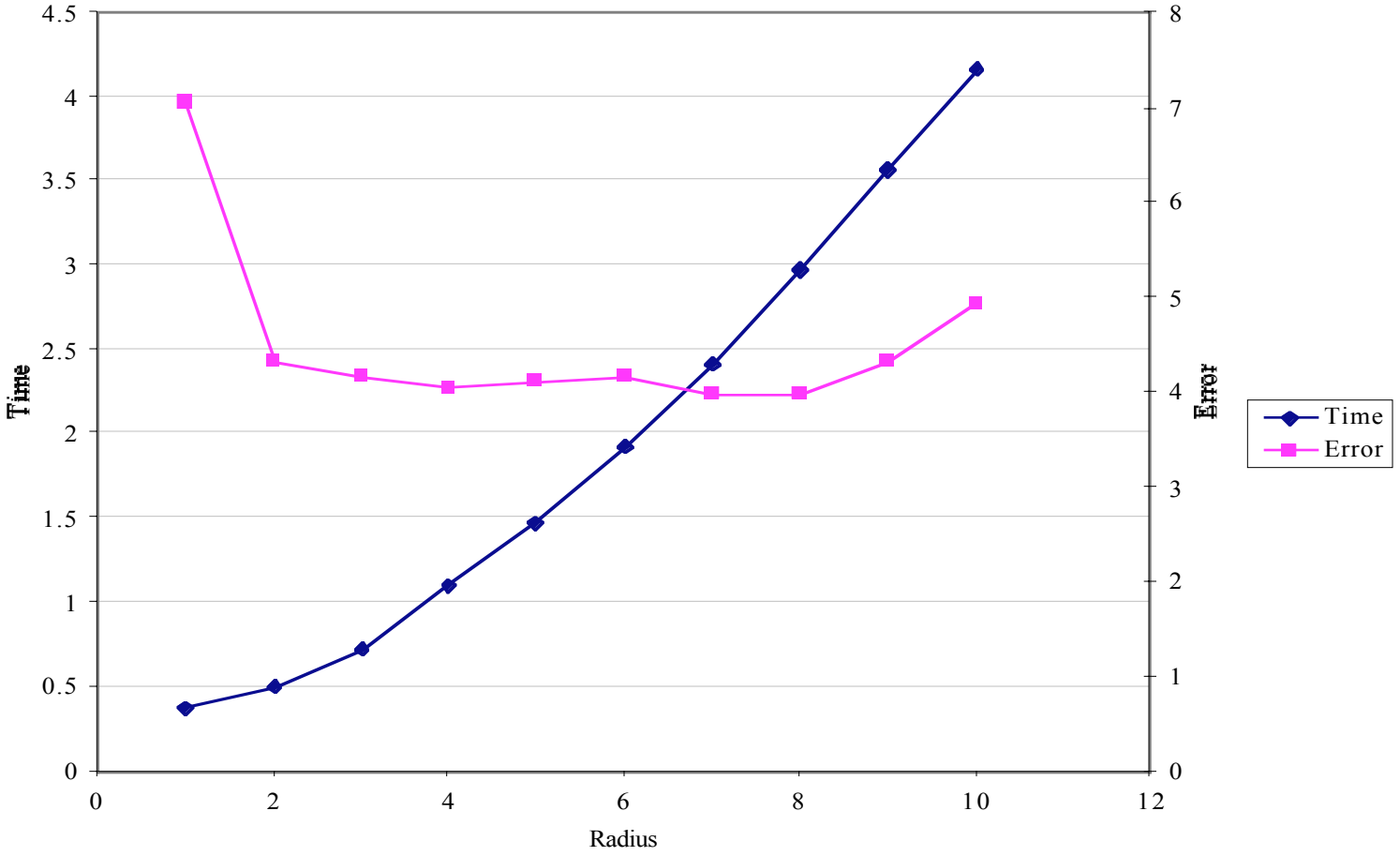


Figure 5.32 Transcription Error & Time vs. Radius for Five Cell Types with Unequal Cell Type Fractions

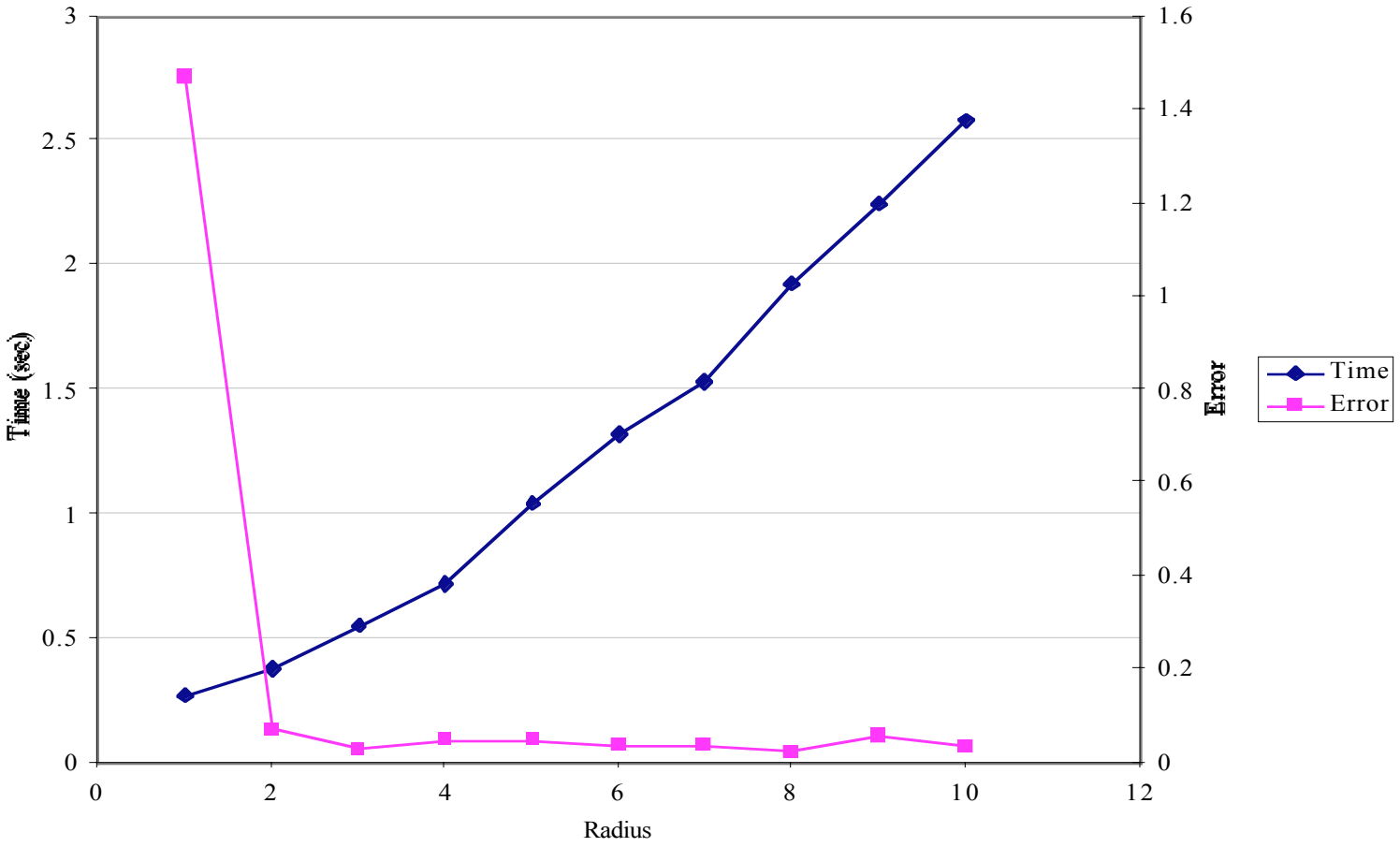


Figure 5.33 Transcription Error & Time vs. Radius for Three Cell Types with Equal Cell Type Fractions

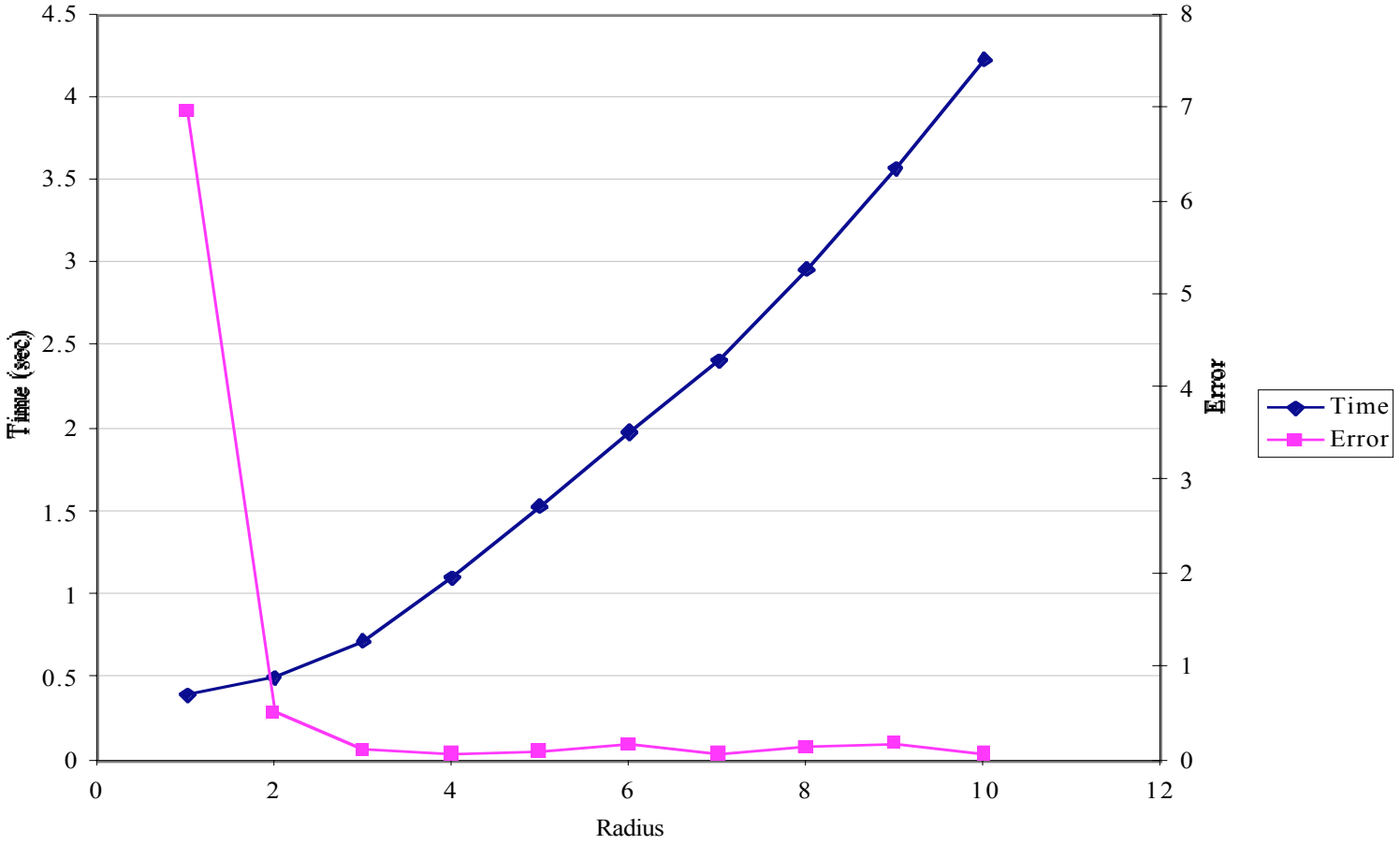


Figure 5.34 Transcription Error & Time vs. Radius for Five Cell Types with Equal Cell Type Fractions

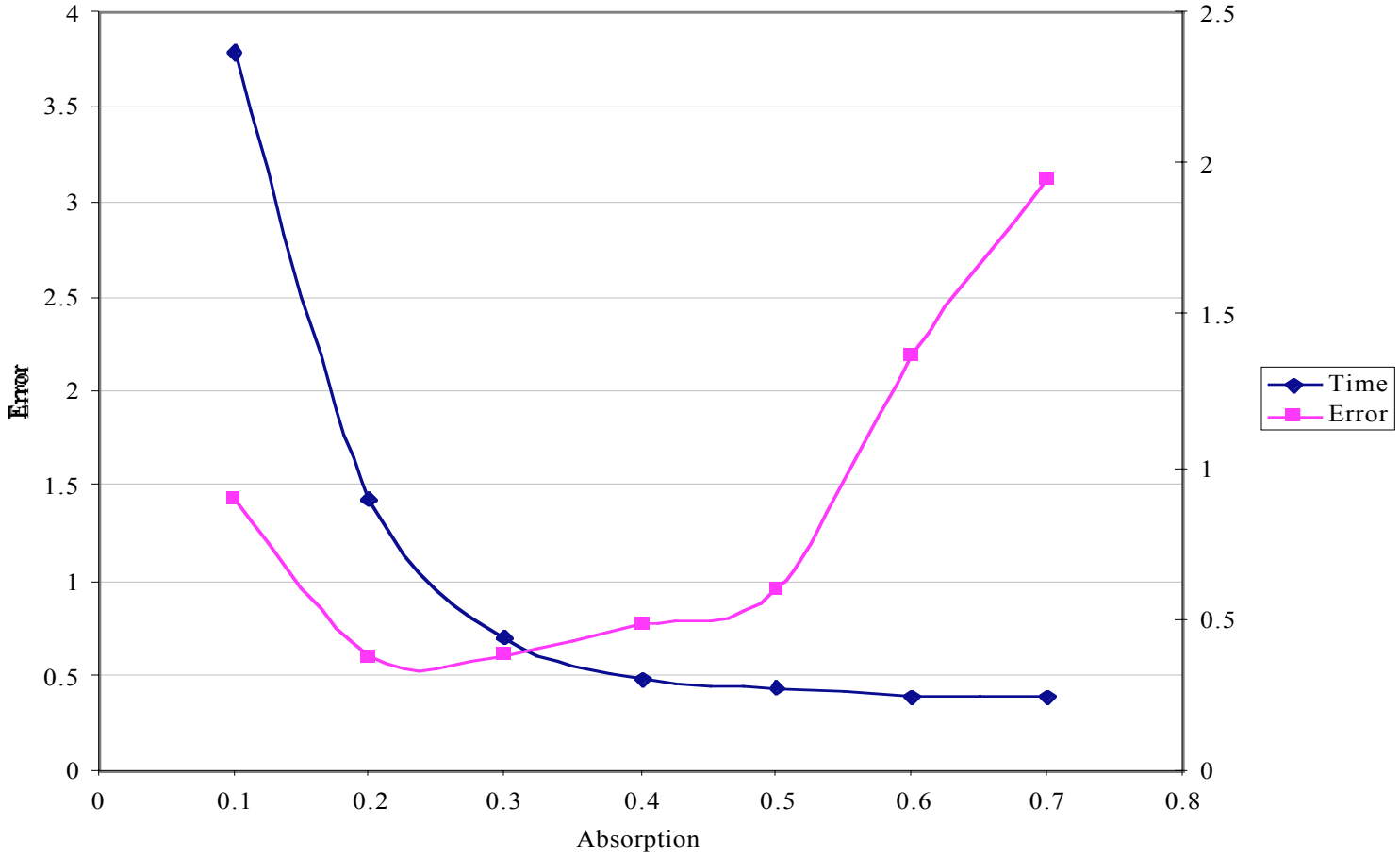


Figure 5.35 Transcription Error & Time vs. Absorption Coefficient for Three Cell Types with Unequal Cell Type Fractions

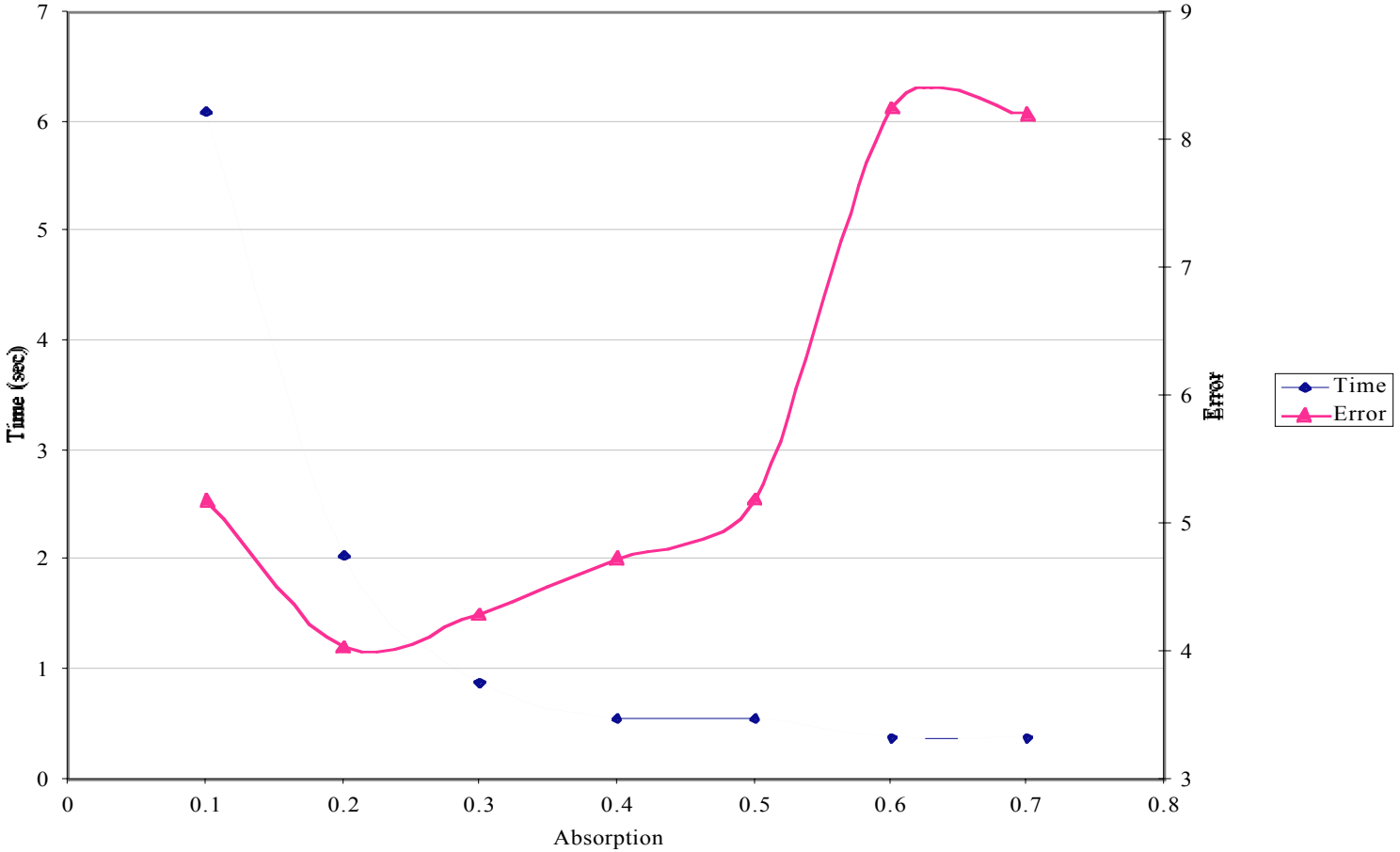


Figure 5.36 Transcription Error & Time vs. Absorption Coefficient for Five Cell Types with Unequal Cell Type Fractions

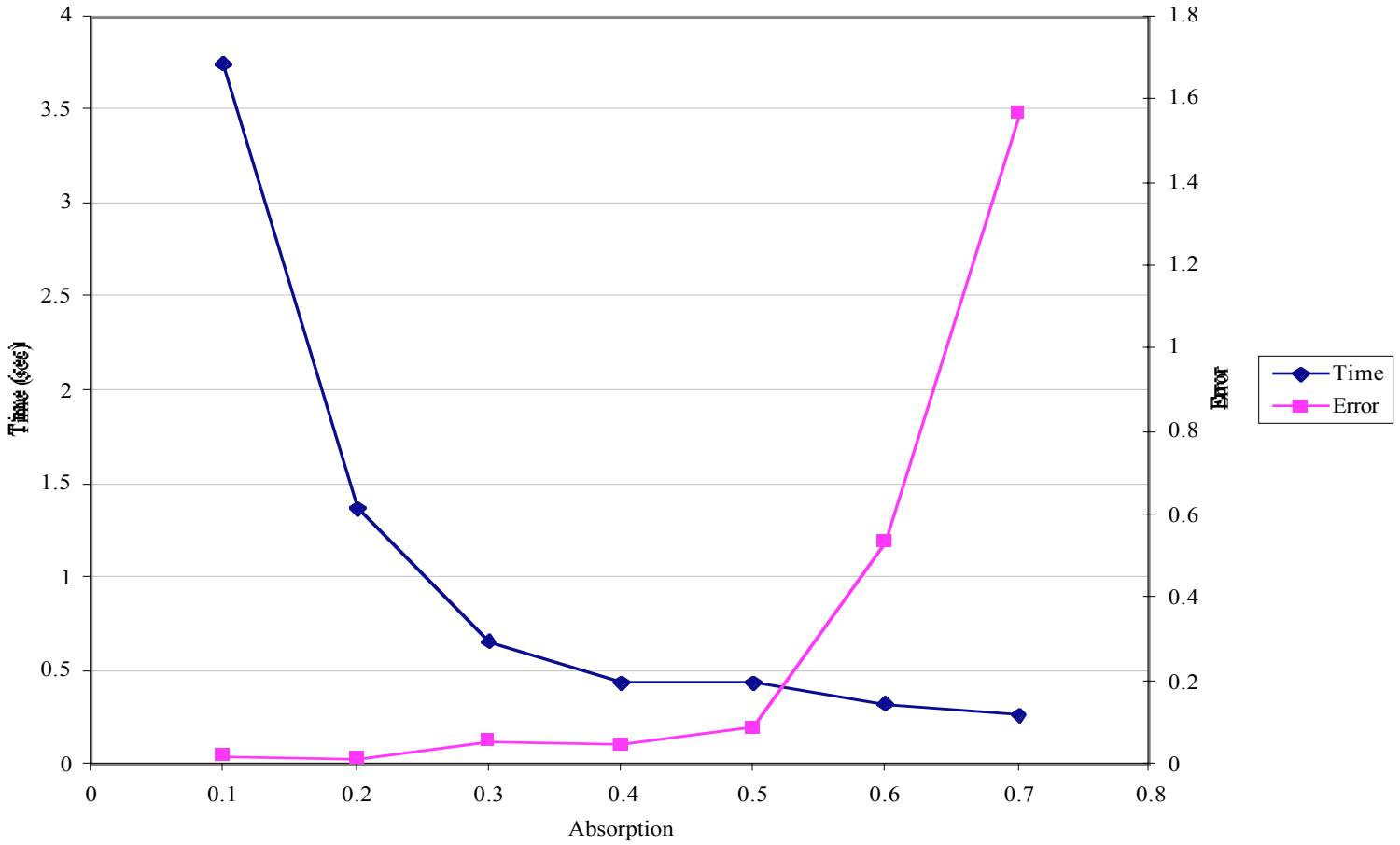


Figure 5.37 Transcription Error & Time vs. Absorption Coefficient for Three Cell Types with Equal Cell Type Fractions

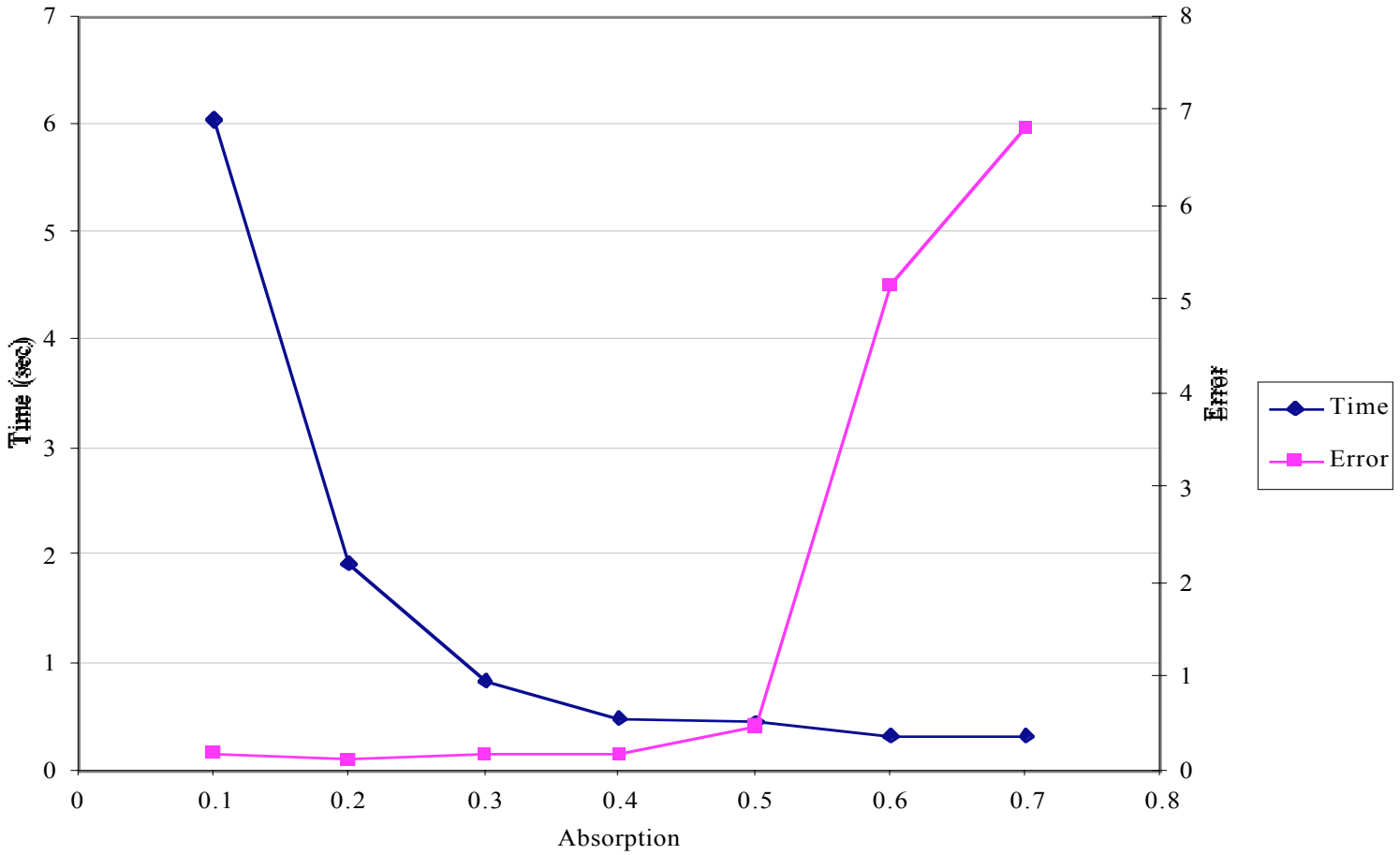


Figure 5.38 Transcription Error & Time vs. Absorption Coefficient for Five Cell Types with Equal Cell Type Fractions

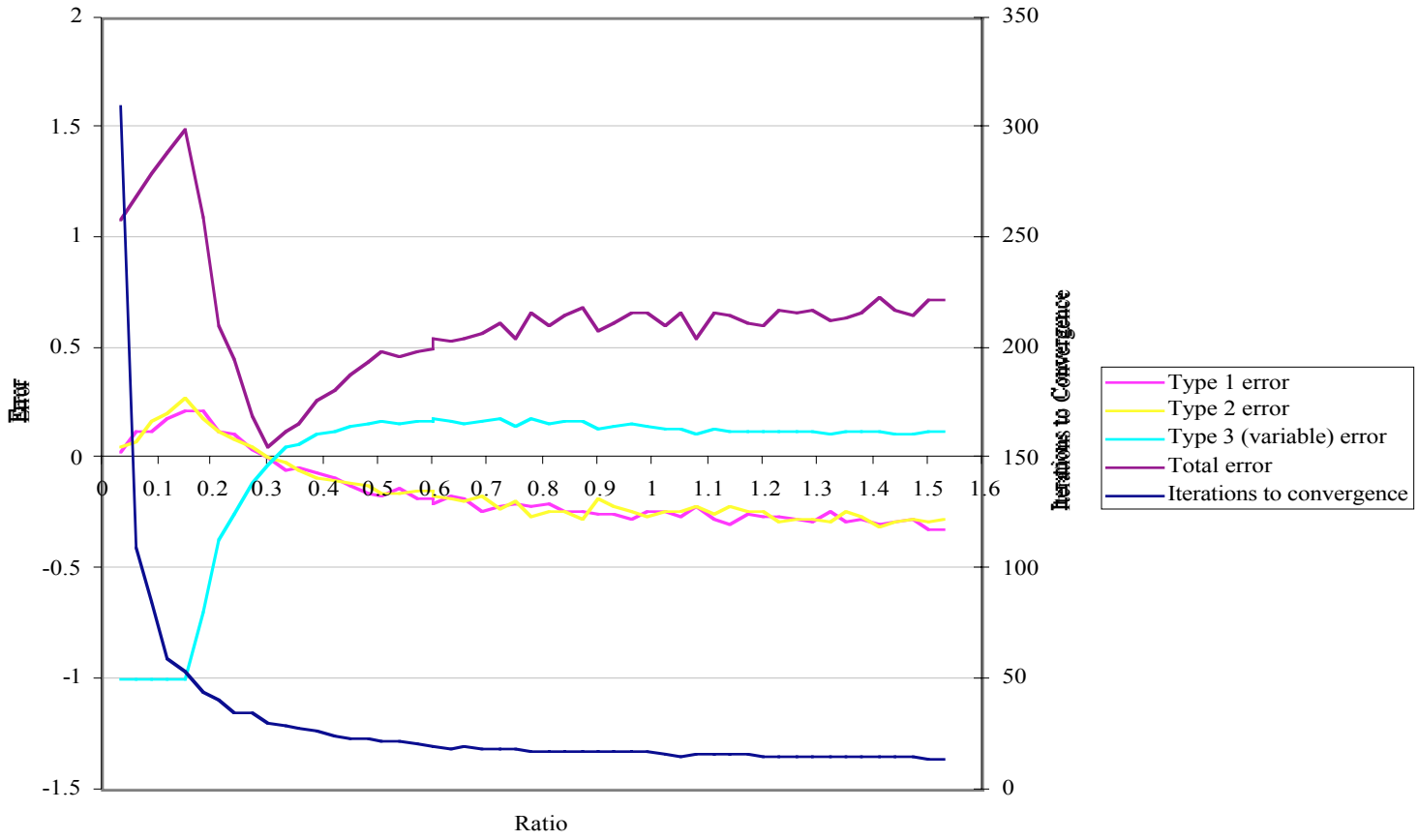


Figure 5.39 Transcription Error & Iterationst to Convergance vs. Ratio of Cell Type Fractions For Three Cells with Two Held Constant and an Absorption Coefficient of .1

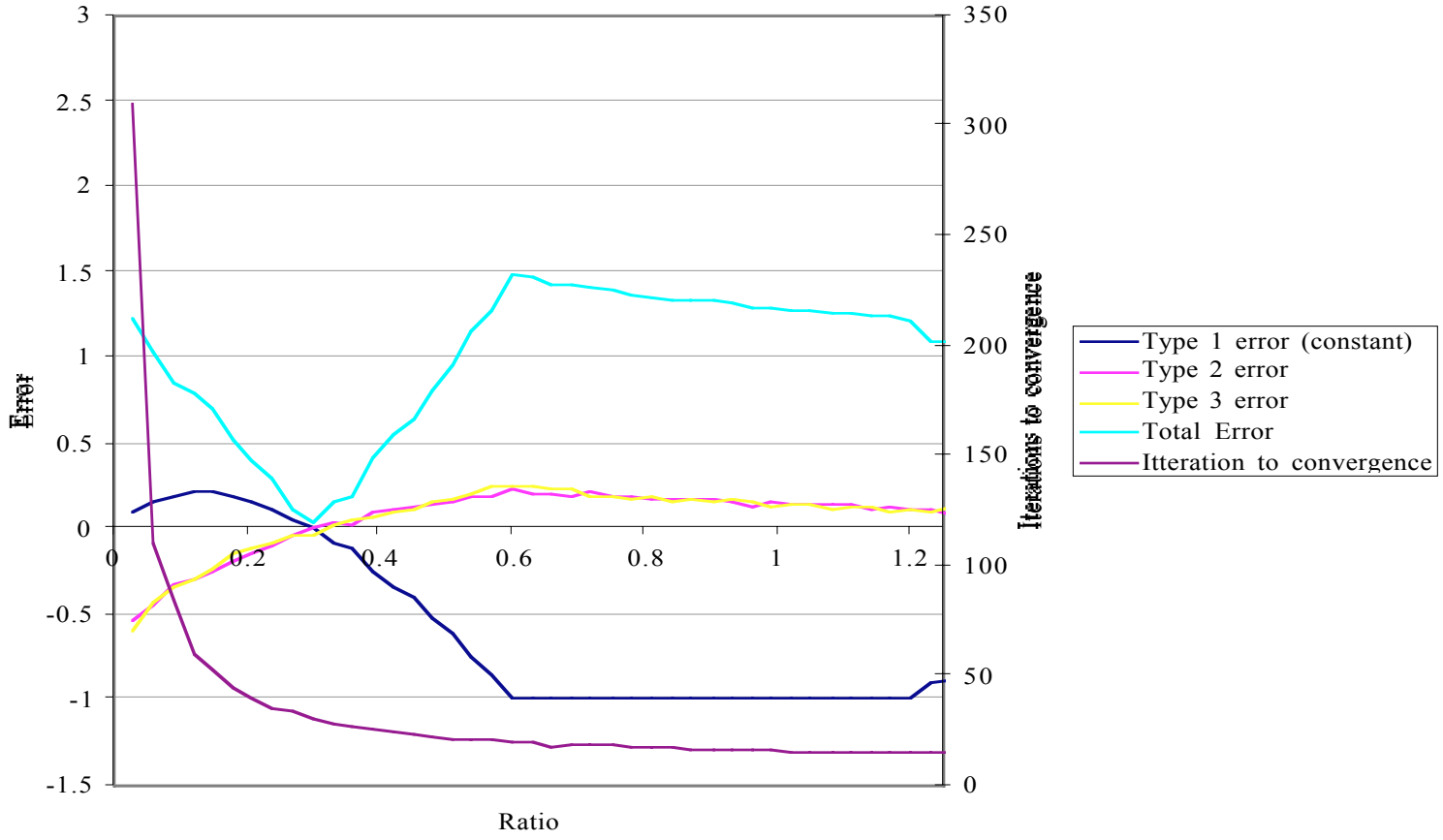


Figure 5.40 Transcription Error & Iterationst to Convergance vs. Ratio of Cell Type Fractions For Three Cells with One Held Constant and an Absorption Coefficient of .1

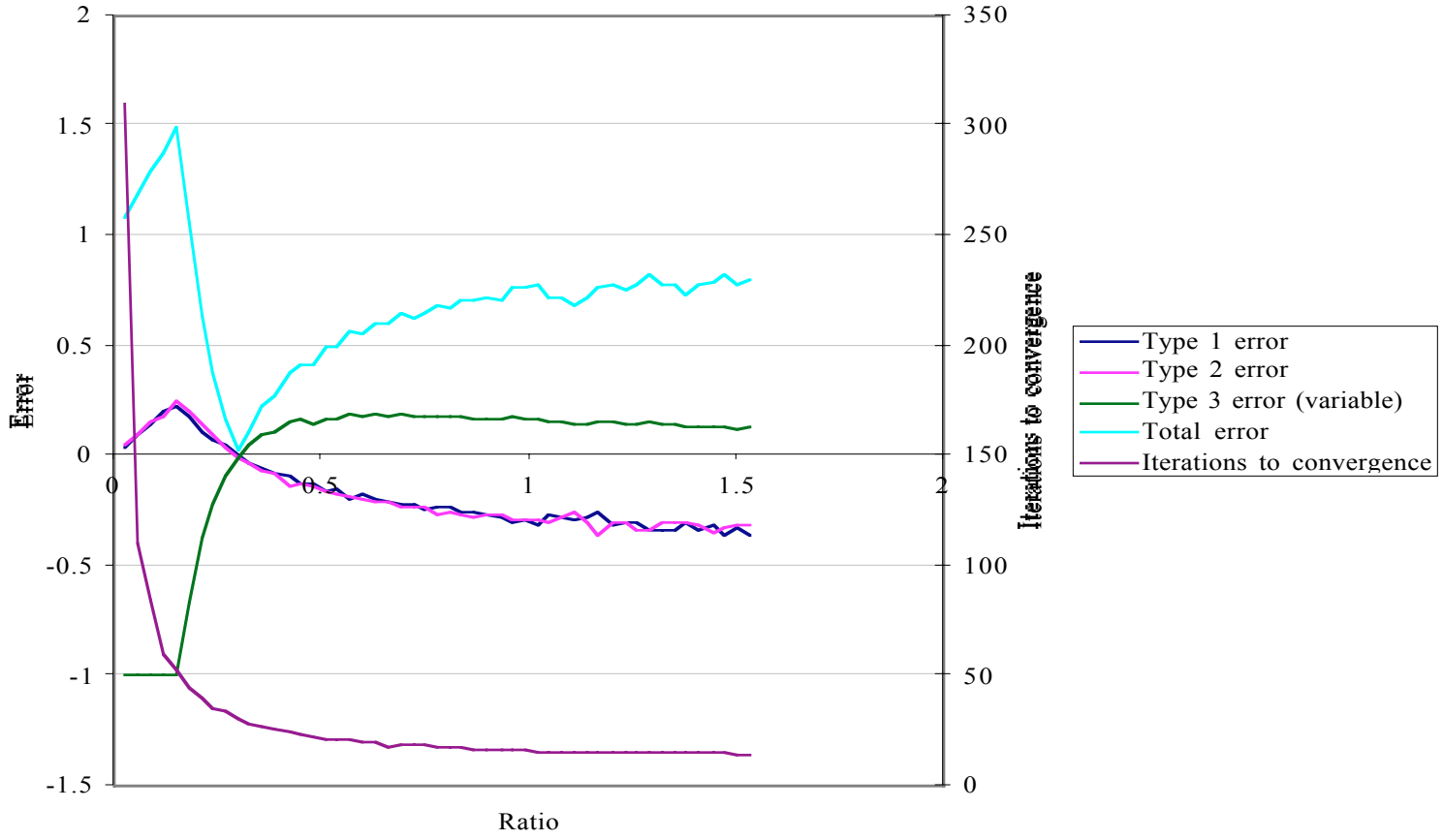


Figure 5.41 Transcription Error & Iterationst to Convergance vs. Ratio of Cell Type Fractions For Three Cells with Two Held Constant and an Absorption Coefficient of .2

In summary, a review of the transcription error has yielded three points of interest. First, there is a proportional relationship between the gene values and those collected from the simulated network that is stable over a wide range of operating conditions. Second, a representative gradient field cannot develop with a radius below 1, and a radius greater than 2 is recommended. Lastly, the maximum radius that can be implemented is equal to 25% of the grid length.

5.1.8 Proof

This section uses the simulated growth technique to produce a three layer neural network. No new or additional steps are required to accomplish this feat. In fact, the three layer network is readily produced by deactivating several of the genes that regulate the growth process. In a sense, the three layer network will be shown to be a diminutive, (retarded) version of the massive scale networks. In other words, the massive scale networks will be shown to be expansions of traditional designs rather than a wholly new system.

The goal of this exercise will be to produce a standard, fully connected, three layer neural network. The network will have 8,12, and 4 neurons in the input, hidden and output layers. For this exercise the three blastular genes which regulate the number of layers, the length and width will be changed from 5, M and N to 3, 4, and 0. These modification will produce a 3 layer network with 4 neurons in each layer. Next, the gastrular genes will be set to produce 2, 3 and 0 divisions per layer to produce the desired neuron count of 8, 12, and 4 for the respective layers.

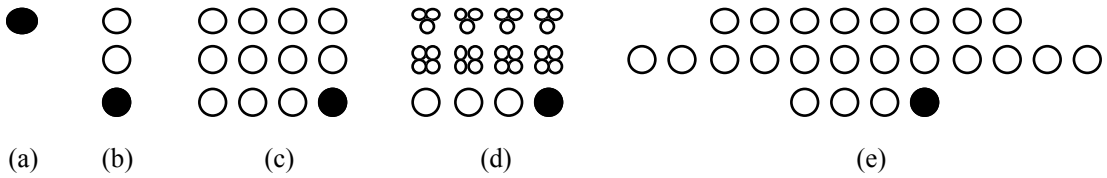


Figure 5.42 Growth of Three Layer Network; (a) initial cell, (b) first growth ccyle, (c) second growth cycle/completed blastula stage, (d) gastrula stage, and (e) complete network.

The simulated growth algorithm assumes that each layer has a different cell class, thus by default this system has three cell types. Normally, the layers would further differentiate to produce more cell types, but in a traditional network, where there is only one

type of neuron, no further differentiation is necessary. Prior to entering the fetal stage the cytoarchitecture table must be constructed.

Table 5.1 Dendrite and Axon Maximum Length per Cell Type.

	Layer 1		Layer 2		Layer 3	
Cell Type	Dendrite	Axon	Dendrite	Axon	Dendrite	Axon
1	0	0	0	100	0	0
2	0	0	0	0	0	100
3	0	0	0	0	0	0

In constructing this table, a maximum length of 100 has been selected to ensure that the network will be fully connected. This length is somewhat arbitrary; its value only needs to be equal to or greater than the largest layer. One should note that cell type 1, (layer 1), only has axons that project into layer 2. Similarly, cell type 2, (layer 2), only has axons that project into layer 3. Dendrites could have been used, but the design is so rudimentary, they were found to be unnecessary. One should also note layer three has no axons at all, as it is the output layer of this design.

At the end of the fetal stage the below network would have been produced. It should be clear at this point that the growth simulation algorithm is robust enough to produce virtually any network architecture.

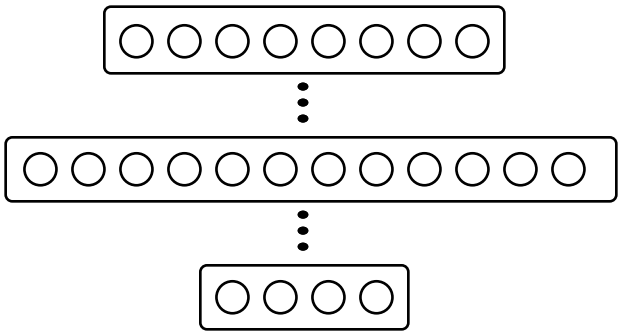


Figure 5.43 Complete Three Layer Nerual Network.

5.1.9 Summary

This discussion demonstrated how the theoretical model was converted into the growth simulation program. Implementation of the growth simulation program suggests that the cell type may be an artifact of observation that occurs when cells receive similar growth controlling signals and produce similar features. The neural grid was converted into a series of two dimensional arrays to facilitate programming, and the effects of this conversion on cell interconnectivity was discussed. The output of the growth simulation program was evaluated and the parameters which control the morphology were also discussed. It was found that parameters associated with growth rates have very little effect, while the absorption coefficient has the greatest control over morphology. By varying the absorption coefficient, morphologies could be produced with varying degrees of clumpiness and banding. Error analysis provided the parametric limits of the software which are a critical radius below 2.5 UCLs and a critical radius above one half the grid length. Within these parametric limits the error analysis showed there was a constant transcription error which indicates that the decryption process is constant but not linear. A short proof has shown that the cell growth simulation is capable of producing traditions designs. After defining and describing the cell growth simulator and its output, the next section describes the implementation of the networks produced by the simulator.

5.2 Recommendations For Future Study

There are four main areas for future study, all of which have been shown to be simple expansions of the current model. The first area is the integration of multiple cortices. The following discussion shows that the full power of the model cannot be realized.until several cortices are linked together. The second area discusses the integration of multiple cortices into other hardware elements to form a complete system. The third area is improving the differential gene expression based algorithm used for optimization. The last area examines various ways to improve the model by making it more representative of biological systems.

5.2.1 Integrating Inputs and Output

The preceding discussion has focused quite heavily on the design of an artificial neural cortex. The model has two types of inputs, referred to as afferents, and four types of output cells, the pyramidal cells. In previous sections it was mentioned that these six types of I/O elements were required to integrate multiple cortices. This section will describe how an artificial cortex is integrated into other hardware elements to form a complete system. The discussion begins with a single cortex, progresses to multiple cortices, and ends with the integration of multiple symmetric cortices.

5.2.2 Cortex Modeling

The artificial cortex model developed in this study is schematically represented with a six-stratum cube as seen in Figure 5.44(a). The horizontal bars on the face of the cube denote the boundaries between its strata. All I/O axons enter and exit the cube from the bottom through stratum 6. These I/O axons will therefore be used to denote stratum 6 so that the cube can be illustrated without the need to enumerate each stratum. An alternative representation of the cortex is seen in Figure 5.44(b) which represents an isometric face view of the neural cortex cube. The neural cortex cube represents potentially hundreds of thousands of neurons, and a single axial projection, either entering or leaving a cortex to go to another, simply marks the path that a bundle of neurons takes. No attempt has been made to numerically represent the bundle size, although this could potentially be done by adjusting the line weight proportional to the bundle size.

This schematic of the cerebral cortex was assembled from the set of schematics shown in Figure 5.45(a) is the front view of an artificial cortex showing all I/O axons evenly distributed across the width of the cube. The second schematic has separated the output axons, or pyramidal cells, from the first schematic. Selecting one pyramidal cell from each stratum in a staggered fashion from the second schematic produces the third image, Figure 5.45(d) and can also be represented with single elements characterizing the total axon count for the given strata Figure 5.45(e). Combining schematics Figure 5.45(c) and (e) yields the final cortex model where the arrows on the bottom of the I/O axons indicate their direction.

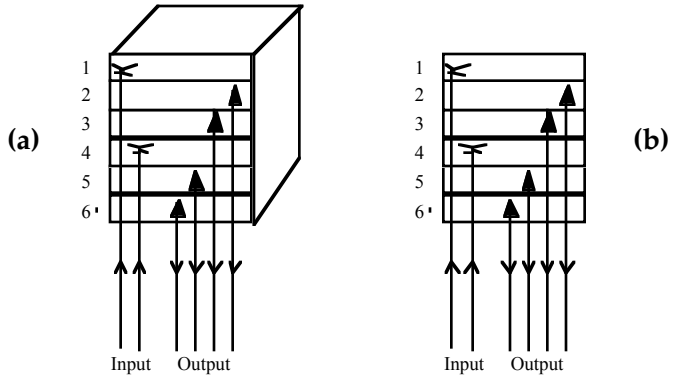


Figure 5.44 Neural Cortex Cube; (a) prospective, (b) isometric face-view.

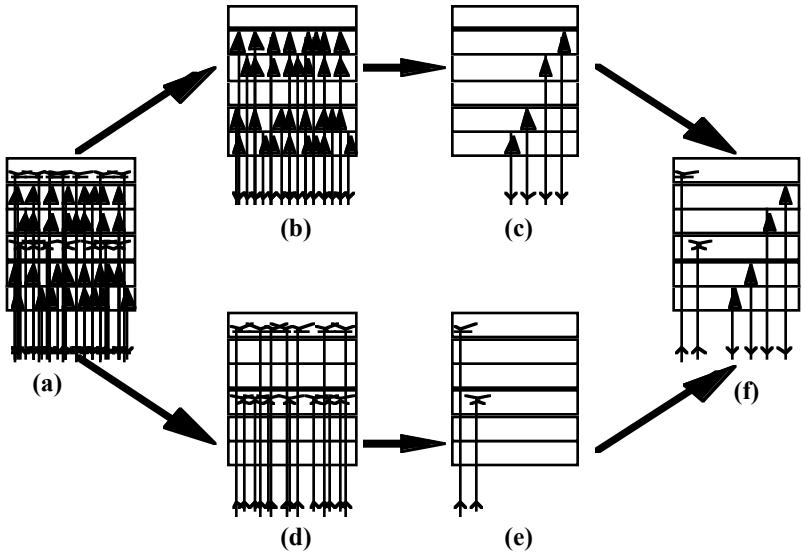


Figure 5.45 Generation of the Cortex Cube Schematic; (a) all elements shown, (b) only output neurons shown, (c) with only characteristic elements, (d) only input afferents, (e) with only characteristic afferents, and (f) final schematic.

5.2.2.1 Single Cortex

The first step in integrating a single cortex with other hardware elements is to produce the I/O arrays. The input array should be proportional in size and geometry to the sensor array to be integrated. The cortex model is based on the assumption that the sensing elements of the system are numerous, functionally equivalent devices arranged in an orderly, geometric fashion. The geometry of the sensing elements must be captured in the

connections of the input array. In the case of a traditional single element sensor, the output must first be broken down into smaller elements to represent the uniform array of simple sensing elements. The process of formatting the output of a single sensing element for use with a neural network can best be explained through example. Consider the design of an artificial visual system. In the first case the sensing element is a CCD array, a device comprised of numerous photo-detectors. The output of the CCD array is already in the format useful to a neural network since the sensor itself is comprised of many small functionally similar elements arranged in a geometric pattern. The actual output of the CCD element is in a serial format, which must be converted to a parallel format for use with the neural network. Alternatively, a visual system could be based on a single ultra sensitive photo-detector, as in the case with nuclear particle detection. In this second example, a single signal is the only output of the sensing element. To convert the output to the format usable by the neural network, the output is customarily separated by frequency such that the input entering the neural network is a series of intensity values at specific frequencies. This latter approach of separating the spectrum is commonly employed in audio applications of neural networks. It is important to maintain the same geometric relationship of the sensing elements because the majority of the associations made by the neural network are formed by looking at adjacent combinations, not necessarily single points.

In the single cortex model, the output array is essentially meaningless on its own. The artificial cortex model developed in this dissertation does not employ the *grandmother* neuron principle of traditional simple networks. Without the *grandmother* principle, where one neuron represents one of the possible classes sought in identification, it becomes very difficult to determine what the network has identified, and it is primarily for this reason that multiple cortex models shall be emphasized.

A single cortex is shown connected to an input and an output array below in Figure 5.46. Note that both associative and specific afferents are connected to the input array. Similarly, all four types of pyramidal cells are connected to the output array. The single cortex model is of little interest because in this configuration the four pyramidal cells are reduced to being of the same type, as are the two afferents. By treating all of the different types of I/O cells in a similar fashion, the inherent structure of the cortex model disintegrates. Since there is no distinction between input and output types, the network is unable to cross-associate information and its overall ability and intelligence are severely degraded.

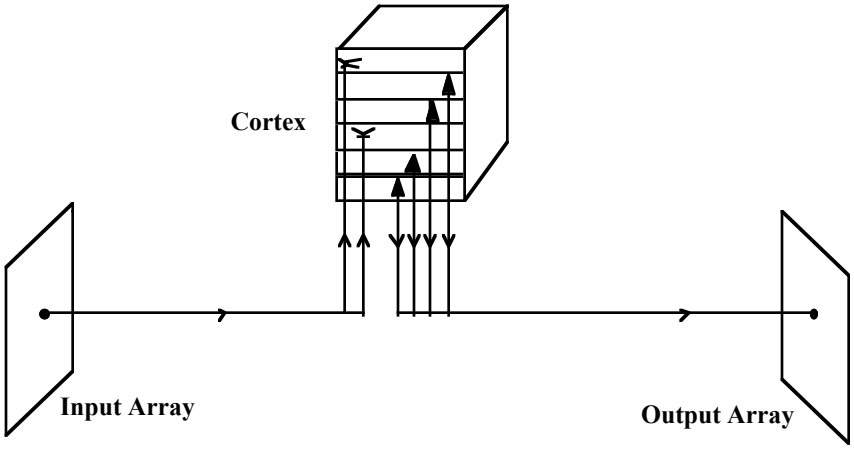


Figure 5.46 Single Cortex Model

5.2.2.2 Paleocortex and Allocortex

One of the interesting consequences of designing a single cortex model is that it demonstrates that a paleocortex, or an allocortex, those having less than six strata or only one stratum respectively, may be produced when the numbers of genes and strata are reduced either by assumption or by turning off genes. Although it was the direct goal of this work to develop the six-stratum cerebral cortex structure, it is useful to realize that this model is capable of producing the more primitive structures of the olfactory cortex and hippocampal formations.

5.2.2.3 Multiple Cortices

The real power of this model is realized in its ability to design and interconnect a multitude of various neural cortices. The interconnection process of multiple cortices requires a specific definition of the different types of I/O cells. Each of these I/O cell types is responsible for communicating information to be utilized in slightly different ways. Three models of multiple cortex systems are presented in order of increasing complexity for illustrative purposes. The first example seen in Figure 5.47 is the interconnection of two cortices. The reason for connecting two cortices is that one cortex becomes responsible for identifying the desired elements in the input field, while the other cortex is used to convert the neural signal into a coherent output, from a human factors standpoint.

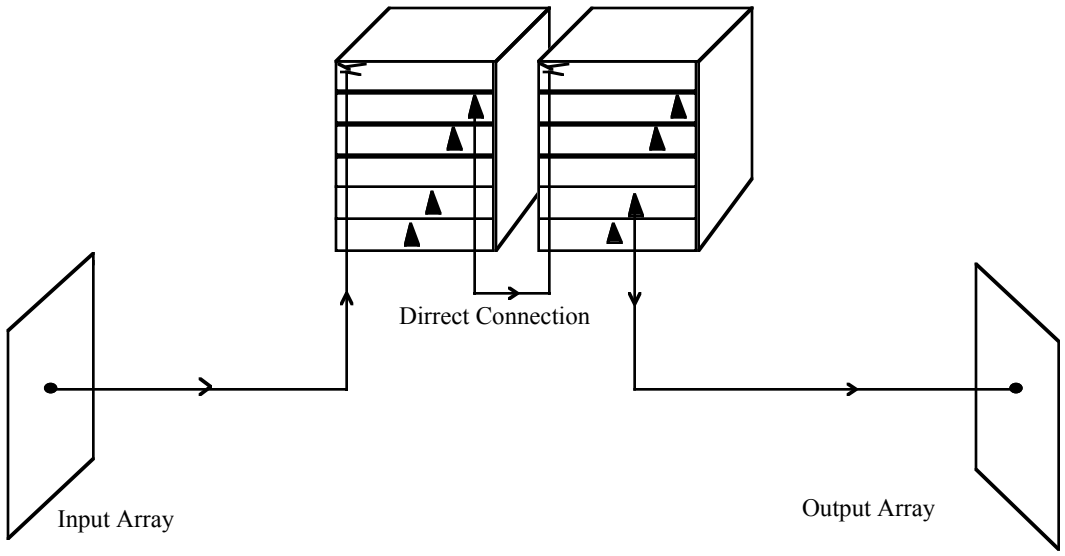


Figure 5.47 Multiple Cortex Model

This system is analogous to how the Broca's region and the visual cortex allow a person to speak the word that represents the object he sees. The first modification that should be observed is that all connections from the input array only enter the first cortex, and that those connections are specific afferents that form synapses on the fourth stratum. The second difference to be observed is that the pyramidal cells of the second stratum of both cortices become associative afferents forming synapses on the first stratum of the other cortex. In this simplified model the second cortex receives its primary input from the pyramidal cells of stratum 2 from the first cortex. The input array project into the afferents of stratum 1 of the first cortex. The stratum 5 pyramidal cells of the second cortex project to the output array. All other elements have no logical connection.

By interconnecting two cortices the processed information from the individual cortices can be used together to form stronger associations. It should be apparent that more of the inherent potential of the model is being utilized compared to the single cortex model through the better utilization and distinction of the different I/O cell types. The separation of functional cortices is viewed as crucial to the development of complex neural controllers and neural network based data interpreters. Unlike the traditional view which suggests one network can be optimized for identifying the feature and producing an interpretable output, the multiple cortex model, as described by neural scientists, states that the overall objective is

made up of separate tasks and that separate cortices should be designed and optimized for each task.

5.2.2.4 Multiple Cortices with Thalamus

In the previous section the concept of breaking a job down into individual tasks, where each task is performed in a specific cortex, was discussed. It was mentioned that the pyramidal cells on stratum 2 are reserved for direct intercortical connections. In the case where there are only a few cortices that need to be integrated, it is possible to directly interconnect them solely with the stratum 2 pyramidal cells. However, as the number of cortices used in a system increases, it soon becomes apparent that the neural density of the stratum 2 pyramidal cells is insufficient to adequately provide a singular means of interconnection without forcing the thickness of stratum 2 beyond the limits of what is observed in nature. A more efficient way to interconnect cortices is to design a system around a relay box. In this system each cortex has two bus lines, one for sending information from the cortex to the relay box, and the other for receiving information from the relay box to the cortex. The signal that is returned from the relay box is comprised of contributions from all other cortices. The use of a relay box is employed in biological systems through the subcortical region called the thalamus. The pyramidal cells of stratum 6 provide the output stimulus to the thalamus. Once the signals from a cortex enter the thalamus, they are essentially evenly distributed to allthe other cortices. Although it is known that the thalamus plays a very important role in the processing of information, its structure is difficult to study, and as a result, the fractional contribution of the various cortices has yet to be quantified. Similarly, the logic concerning how information is relayed inside of the thalamus is not understood, also due to its structural complexity. The output of the thalamus enters the cortex as a specific afferent and projects into stratum 4.

There are two points of interest pertaining to the role of the thalamus in the multiple cortex model. First, the combination of direct interconnections from the stratum 2 pyramidal cells and the thalamic projections suggests that although a given cortex is most directly related in function to one or two other cortices, the information in any one cortex can be used by every single cortex in the brain. In other words, there are direct data buses between cortices that perform tasks associated with the same job, and there are thalamic connections that allow information to be distributed to cortices that are not normally associated with the principle pathway of that job. This implies that there are principle pathways that are

associated with commonly performed tasks. This also implies that there are secondary pathways that can be used to associate different primary pathways so that a new job can be performed as a coordinated effort comprised of multiple primary pathways.

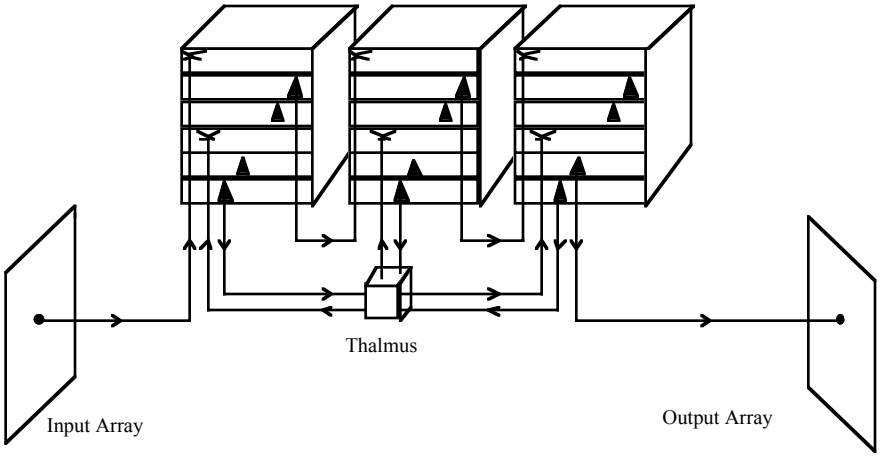


Figure 5.48 Multiple Cortices with Thalamus Model

The second major implication of the thalamic connections is made from studying the relative dependence that various cortices have on their thalamic connections. It is quickly found that the cortices most closely related to the sensory input have relatively less thalamic projections, as is evident by the small size of their stratum 6. It is also found that cortices that perform higher associative functions, such as those in the frontal lobe, receive and project many thalamic connections. This suggests that the thalamus is more closely related to higher brain functions than to primary sensory and motor cortices, further implying that an artificial system developed to coalesce various input types should probably have a well developed artificial thalamus.

5.2.2.5 Symmetric Multiple Cortices with Thalamus

The last feature of brain structure to be discussed is symmetry. The brain is comprised of two halves that perform nearly identical functions. It has been suggested that the two halves perform a type of proof-checking function, where the associations made in one cortex are compared with the other in order to eliminate ambiguities and detect discrepancies. The proof checking abilities of the human brain are well recognized and have been used as the foundation for the development for several neural network models, such as adaptive resonance theory (ART) and bi-directional associative memory (BAM). However, ART and BAM structures focus on localized effects of small clusters of neurons

and do not directly address the issue of the logical integration of multiple cortices. The concept behind the ART system is that two elements performing similar functions pass information between themselves until a resonance type condition is made which corresponds to a positive identification. This basic premise can be applied to cortices by assuming that each cortex can be treated as a separate logical element. While a cortex does not actually act as a united element, the same type of associations are being made in the cortex. In other words, a cortex is a region where numerous sub-elements are performing a similar type task while partially connected but also partially independent. By treating the cortical columns as elements of the ART system, and assuming that one half of the ART system is in the cortex under discussion and the other is in the symmetric element, numerous ART type structures are produced by connecting the symmetric cortices. Interconnecting two symmetric cortices is accomplished by projecting the stratum 3 pyramidal cells into the first stratum of the symmetric cortex and vice versa.

An important aspect of the ART system, as discussed by Grossberg, is that the two elements of the ART system can be, and probably should be, slightly different. When it is assumed that the two symmetric elements are slightly different, it becomes clear that the two symmetric cortices are interpreting the data in a slightly different way, possibly forming two slightly different conclusions. Thus it may possible that one half may successfully identify a feature that the other may not, and vice versa, so that the combination of the two halves provides a higher reliability of correctness.

The variability between cortices can either be linked to genetic or differentiation processes. The first case assumes that different genes are expressed in the symmetric halves, and the second case is based on the concept of the chaotic congenital development.

The complete symmetric multiple cortex model can now be discussed. The symmetric three cortex model seen in Figure 5.49 is comprised of a primary sensory cortex, a secondary sensory cortex, and an association cortex. The cortices are arranged in a linear fashion, corresponding to the primary pathway defined by the stratum 2 pyramidal cell projections. The symmetric halves in the schematic representation are the top and bottom rows of cortices. Note that the elements on the bottom row are drawn upside down so that their I/O axons are directed towards the center. The input array, seen on the far left, projects to stratum 1 of the left most cortices. The input is evenly distributed between the two symmetric elements. The left most, or primary, cortex is primarily connected to the middle, or secondary, cortex through stratum 6 pyramidal cells which project into stratum 1 of the secondary cortex. However, the primary cortex is also connected to the thalamus by stratum

6 pyramidal cells and stratum 4 afferents, and to its symmetric cortex through stratum 3 pyramidal cells and stratum 4 projections. The secondary cortex directly projects to the associative cortex, the thalamus, and its symmetric element by strata 2, 6, and 3 pyramidal cells respectively. The only output of this network would be the stratum 5 pyramidal cells from the associative cortices. In this simplified model, the primary and secondary cortices are not represented to have stratum 5 pyramidal cells. This simplification was made because the artificial system shown in the figure is too simple to be able to represent all of the possible outputs.

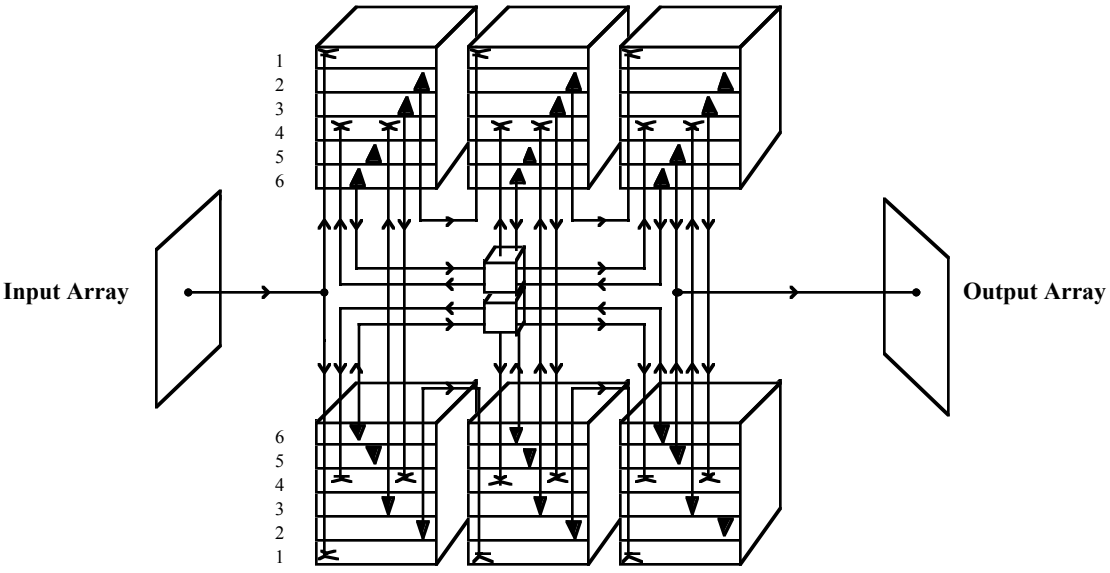


Figure 5.49 Symmetric Multiple Cortex Model with Thalamic Connections

5.2.3 Multiple Primary Pathway Systems

The multiple cortex system discussed in the last section has one severe drawback. The signals analyzed by the sensory cortex cannot be associated with symbols, signals, or patterns that the user can recognize. Since the premise for this work is a massive Hebbian network, any system that is designed must have the capacity to self-associate entities with symbols. The most instructive example is that of identifying objects on an assembly line. In this example, there are three jobs that the network must perform: it must identify entities in the visual field, it must identify symbols in the input field, and it must associate the entities with the symbols and project those symbols to the output field. Each of these jobs must be

represented by a primary pathway involving several cortices (Figure 5.50). The first two jobs, identifying objects in the visual field and recognizing symbols in the input field, can be performed with the basic model of the last section, namely primary and secondary sensory cortices. The third pathway is similar to the others, except that it is comprised of motor cortices, which differ from sensory cortices only in that their overall function is to produce a conditioned output. An associative cortex in the center of the system unites the three pathways.

Training a multiple pathway system to associate visual entities with output symbols is analogous to teaching a child to speak. The process of teaching a child to speak has four stages. The first stage is characterized by the child making incoherent primitive sounds. The second stage involves the child assembling primitive sounds to replicate the words he hears. The third stage involves associating the objects he sees with the words for the objects. The last stage involves associating the words that he hears with the objects.

The training of the network is done in the same steps described in the previous example. First, the network must be trained to create primitive output symbols. This would be accomplished by externally stimulating the motor cortices in a controlled, pseudo-random fashion. This external stimulation is an approximation of the child's internal motivation to create sounds. Since no scientific explanation exists to describe motivation, it is only possible at this stage to simulate it through a software model. Of course, the simulated motivation only serves to give the neural network a basic, primitive set of stimuli from which its initial associations will form. It is necessary to create this initial stimulus because at the beginning of training, the network would have made no associations and no defined neural pathways from which other learning could be based. As the motor cortex is being stimulated, the output field is projected onto the input field, creating a closed loop. With the closed loop the network learns to associate the output symbols to input symbols of the same type through Hebbian adaptation. In the second stage the external stimulus is projected to the output field. The external stimulus in this case is assembled symbols that represent entities. The loop is again closed by connecting the output field to the input field so that the network can learn to associate assembled output symbols to assembled input symbols, again through Hebbian adaptation. At the end of the second stage, well defined pathways will exist from the input field to the output field.

In the next stage, images are projected to the visual field at the same time that assembled symbols are projected to the input field. At this point it is useful to represent the associative cortex as a single neuron and the three principle pathways as terminals leading to

that neuron as seen in Figure 5.51. Using this analogy, the assembled input symbols correspond to a strong input and the assembled output symbols correspond to a strong output, Figure 5.51(c). The visual stimulus, the weak signal, is associated to the strong signal of the input symbols, Figure 5.51(d). Through Hebbian adaptation, the visual pathway is strengthened, Figure 5.51(e), to the point where only the visual stimulus is needed to cause the associative cortex to stimulate the motor cortices to produce the output symbols. The reason it is possible to represent the cortices in this fashion is that Hebbian adaptation is the only mechanism whereby the network learns. The nerve cells in the cortices behave in this fashion and collectively, the cortices also behave in this fashion. The only real difference is that the learning associated with a cortex requires the collective learning of thousands of individual neurons.

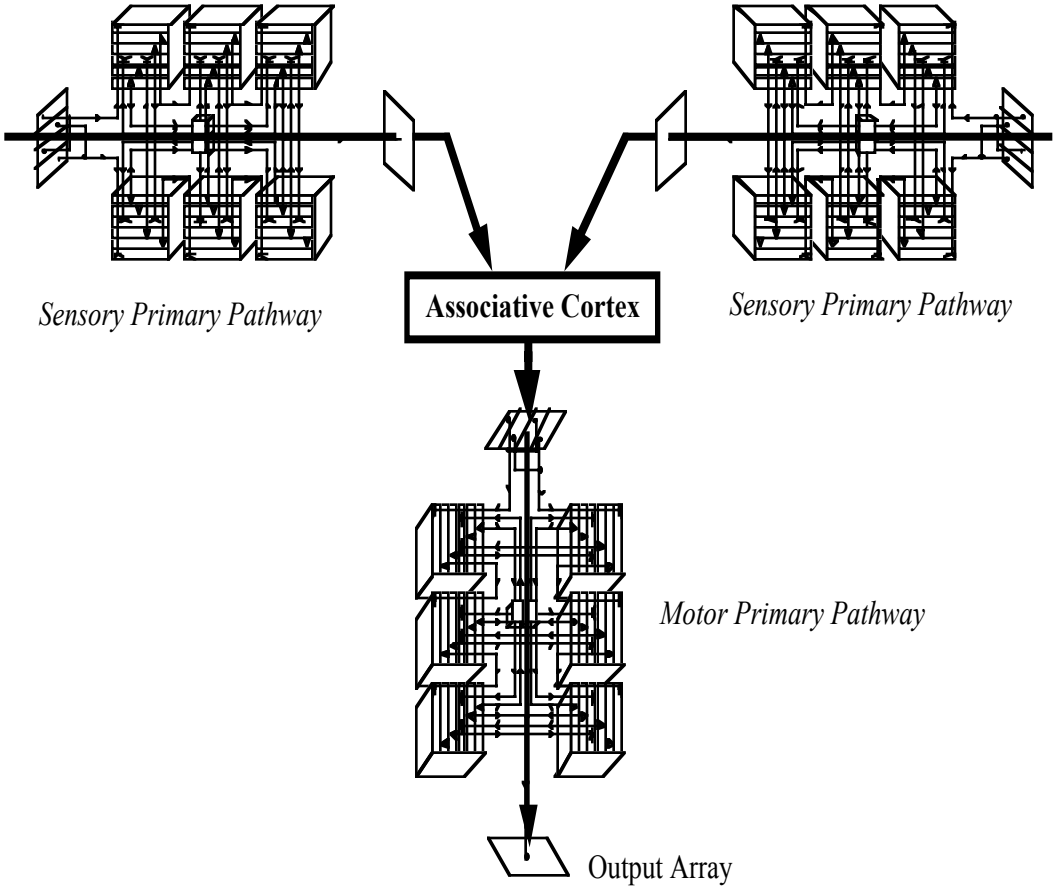


Figure 5.50 Multiple Pathway Network

Once the learning phase is completed, the input array and its associated cortices do not require activation for the network to identify an a object in the visual field. At this point, the synapses can be frozen and the network can be packaged for use.

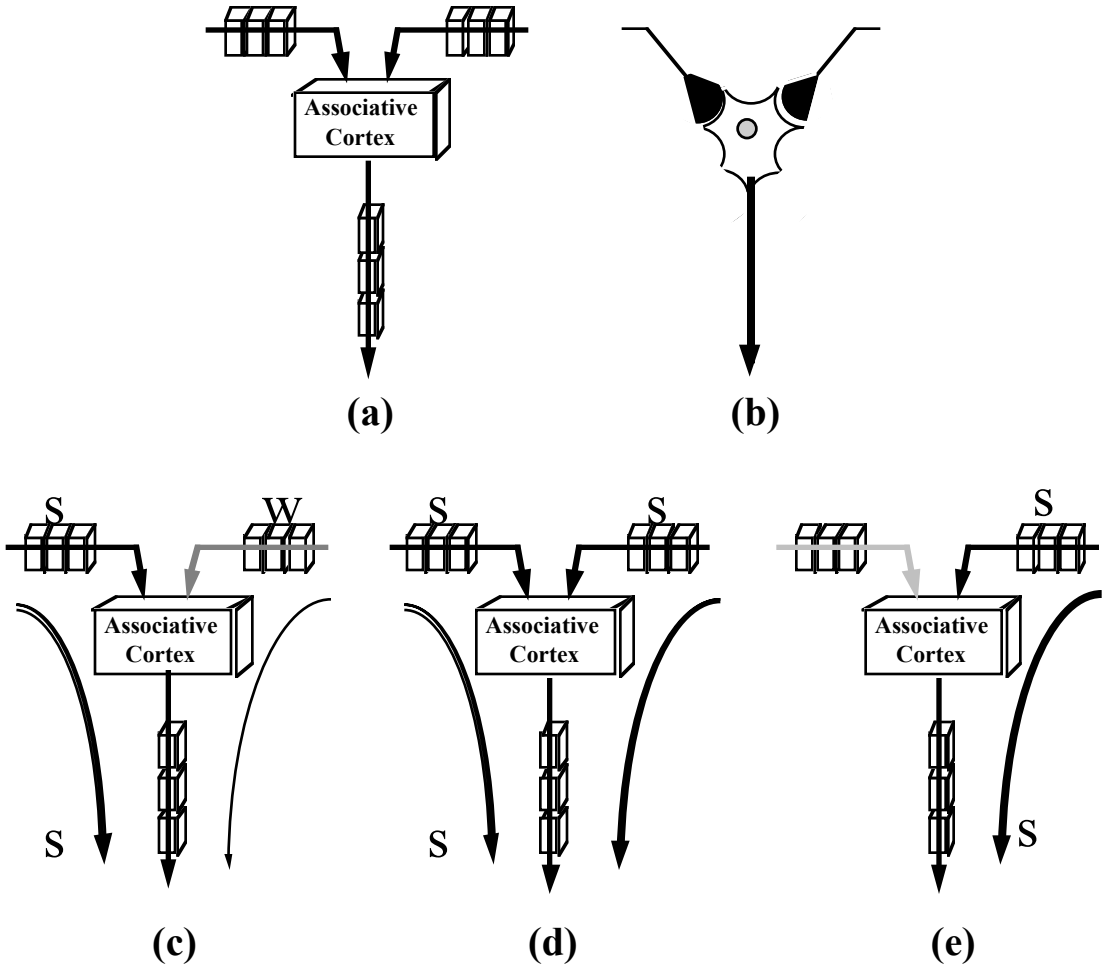


Figure 5.51 Complex Hebbian Learning. The fractal nature of the nervous system shown in similarity between multiple pathway network (a) and single neuron (b). Pathways learn by associating a weak and strong input with a strong output (c). Over time the weak inputs strength increases (d) until it alone cause a strong response (e)

5.2.4 Differential Gene Expression Based Optimization

In previous sections the design of a differential gene expression based genetic algorithm has been discussed and described. The system involves producing a number of

alleles, or different values, for the same gene, then randomly assigning a subset from each parent to form a complete offspring. Although the program that creates the new chromosome from the parents is fairly straightforward to produce, the logic whereby the parents are selected for breeding is not. Once a neural network has been developed and trained, its performance must be evaluated, and then based on a series of performance measures, its probability for breeding, or fitness, must be determined. In order to ensure that the optimization follows patterns similar to natural selection, it is necessary to create a multitude of parameters that push the gene pool away from known undesirable conditions or states without pulling the population towards a perceived optimal state.

The natural selection algorithm can only remove from the breeding pool those individuals that cannot perform the tasks required of them, i.e. those that are unfit for the environment. The unfit members will likely be a very small subset of the total population. From the remaining "fit" members of the population some of the guidelines used in industrial engineering for optimization of systems can be used for biasing the system away from undesirable limits. Depending on the task, a number of performance measures must be obtained. In the case of teaching neural networks to recognize patterns, likely measures might include the accuracy, speed of recognition, tolerance of noise, ability to separate similar entities, etc. It is important to have as many performance measures as possible since in reality, an optimal network should excel in all areas that one could conceive a measure for. It is also necessary to have a performance measure in every area of network performance desired because these performance measures are the driving force for the optimization, and the networks will only evolve in the areas that performance measures exist.

Once the performance measures have been defined, an appropriate way to implement them must be found. The natural tendency of engineers is to preferentially select the members who have the highest performance values; however, this approach as discussed in the chapter on genetics, leads to weak populations and suboptimal solutions. Instead, the population must be biased away from those members with the poorest values. In other words, those elements with poor performance values should have a significantly lower probability of breeding; however, no preference should be given between an individual that excels and one with moderate performance. The type of treatment described above can be accomplished by providing a threshold below which breeding rarely occurs, or by applying a logarithmic function to describe the breeding population, such that above a critical value, all members have approximately the same breeding probability. Figure 5.52(a) shows the probability of breeding versus the performance rating for the discrete and continuous functions described

above. When these figures are compared with Figure 5.52(b) that represents the breeding probability versus performance rating of traditional genetic algorithms, it becomes apparent that the traditional method eliminates a much larger segment of the population each generation. Since the population represents, or is proportional to, the solution space, it should be clear that the traditional approach eliminates a greater number of possible permutable combinations or solutions than the technique proposed in this paper. Although it can be argued that the traditional genetic algorithm performs very well on NP hard problems, no one has truly applied genetic algorithms to a problem with the number of parameters being addressed in this dissertation. In addition, traditional genetic algorithms rely on a mutagenic process to pseudo-randomly search the solution space and escape from being trapped in local minima, a technique that has proven to be fairly effective, but is not truly representative of the biological system.

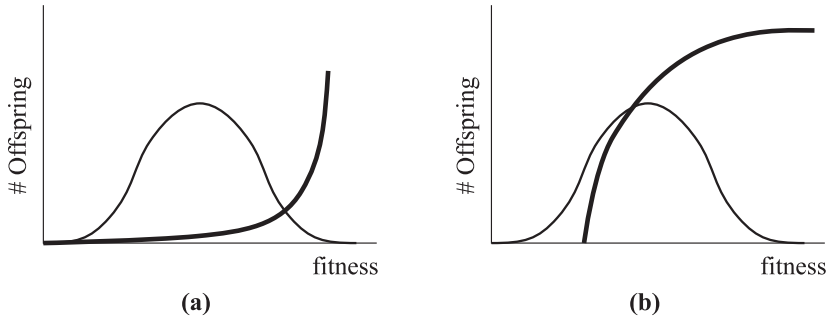


Figure 5.52 Members of Populations Selected for Breeding in (a) Traditional or Selective Breeding and (b) Proposed Natural Selection Genetic Algorithms. Note Hatched areas represent members used in reproduction

Optimization is obtained using the new technique by slowly raising the threshold or critical value of the performance rating. The slow increase of the threshold is analogous to the slow, but steady, changes that occur in the environment. Over time the members with the poorest performance will slowly be removed, and the entire population will, on average, become "more fit." It is believed that keeping a large and diverse population will aid in the optimization of the numerous parameters needed to define a massive scale neural network.

In addition to the environmental changes pushing the population away from undesirable attributes, the species has an internal set of breeding criteria to selectively pull the population towards socially desirable traits. However, these traits are fairly few and tend to be cosmetic in nature; nonetheless, these breeding patterns allow the augmentation of the fitness criteria to allow some elements of a pull system similar to the traditional algorithm.

The combination of the push and pull selection phenomenon in breeding should be utilized to increase the rate of convergence towards the true optimum.

5.2.5 Higher Modeling

Future research should definitely be directed towards developing a way to represent the strata as three dimensional structures. There is undoubtedly a functional dependence on the three dimensional structure which may prove to be crucial for certain applications. There are three additional levels of complexity that should be considered. First, each stratum should be treated as a three dimensional structure, but similarly to the current model, each stratum would develop essentially independently. The next level of complexity would treat the strata as three dimensional structures whose development is dependent on the adjacent strata. In other words, ligands would pass between strata, and the stratum boundaries would become areas of interest. The final layer of complexity would address the issue of the boundary conditions at the boundary of the cortices. In the current model the edges are treated essentially as walls through which no ligands can pass. Future models should evaluate the development of one cortex in the context of being adjacent to others, such that the boundary between cortices becomes an area of interest.

Another area of future research is developing techniques whereby multiple genes can be expressed simultaneously. The simultaneous expression of multiple genes would allow behaviors to be represented as combinations of fundamental mechanisms which in addition to reducing the number of genes to define a structure, would be a more accurate representation of the biological model.

The area of hardware is of course one of considerable interest, both academically and commercially. There are two types of hardware that could be developed from this system. The first would be optimized for simulating the growth process, and the second would be used for actually running the networks.

5.2.6 Summary

In this section the implementation if the massive scale neural networks designed in this dissertation has been discussed. Process sensors map to the input array which projects and maps into the first artificial cortex. The output or control signal projects from the last cortex to the output array from where it is connected to external equipment and used to control the process. The five stratum structure is only fully utilized when multiple cortices

are connected in a symmetric fashion using an artificial thalamus. The lack of full utilization of the structure leaves many if the pyramidal cells unconnected and without purpose. Under the less than fully interconnected, multiple, symmetric, thalamic design the success and power of the cortical model is questionable.

A collection of cortices connected in a progressive fashion is called a primary pathway. Primary pathways that convert sensory input to a complex pattern representation are called sensory pathways and those that do the reverse, converting a complex pattern representation into an output or control signal are called motor pathways. An associative cortex receives input from at least two motor pathways and its output projects to a single motor pathway. The sensory to associative to motor structure is symbolically and functionally similar to the organization of a single neuron showing both the fractal nature of the biological model and providing a means for training massive scale systems. When multiple pathways are connected by associative cortices the entire structure is trained simultaneously through Hebbian rules, thus eliminating the need for single neuron or backpropagation training. The discovery of the fractal nature of this system has provided a viable training technique which ultimately enables the realistic development of massive scale neural systems.

A brief comparative discussion between selective breeding and natural selection breeding patterns was also included in this section. The traditional or selective breeding genetic algorithm was shown to eliminate a greater number of members from the potential gene pool thus decreasing the number of optimal solutions. Selective breeding genetic algorithms have an implied solution and do not truly let the system seek the optimal solution. The selective breeding algorithm pulls the system towards the implied solution. The natural selection algorithm proposed in this dissertation leaves a much larger population intact each iteration and pushes the system away from the nonoptimal states thereby allowing the true optimal solution to be found. The natural selection genetic algorithm is based on the differential gene expression concepts described in Chapter 4.

It was also suggested that future work be directed towards the development of true three dimensional neural structures and systems that can more accurately model the parallel processes that the model developed in this dissertation utilizes.

5.3 Chapter Summary

The cell growth simulation program produces controllable output patterns that are chaotic in nature and analogous to the biological counterpart. The similarities of these structures to the biological model and the tremendous flexibility and control of these patterns made possible through the growth simulation program and the imbedded genetic algorithm ensures the successful development of functional massive scale artificial neural networks. The ability to optimize the functionality of the network by varying the structural and cellular properties allows the development of specialized control systems. Control systems for a variety of industrial process control including composite manufacturing can be designed using the technique developed in this dissertation.

In addition to developing structures that can be optimized and designed for process control this dissertation has shown how to integrate multiple cortices to produce complete control systems. The complete control systems which re comprised of sensory, associative, and motor cortices is organized in a similar fashion to a single neuron allowing training to be accomplished through traditional Hebbian rules.

The next section is a brief list of the accomplishments of this dissertation.

5.4 Conclusions

This dissertation has developed a new technique for the design of massive scale neural networks. There are four main sections of this work. First a review the historic data was collected and the critical parameters which determine the worth of network were identified. Second, mathematical relationships were developed to describe the parameters from the previous section. An artificial chromosome was created and a technique for encoding the network characteristics into the chromosome was also put forth. In the third section, a cell simulation program was developed which used the artificial chromosome to control the growth of the network. Lastly, the results of the simulation program were analyzed and potential of future research was discussed. The following highlights the major accomplishments of each of these sections.

In the first section, the intelligence of the neural network was established to be proportional to the number of neurons and synapses. It was established that networks with the same number of neurons would display functional variations through different connection patterns. The biological system was analyzed and the diverse functionality observed in its

nervous system was attributed to multiple cell types, the six stratum structure of the neocortex, and a multiple cortex design of the brain. The various cell types in the nervous system are distinguished by structural and functional properties. The nerve cell properties have been identified. The significance of these properties to the behavior and performance of the network has been discussed. The structure of the brain was model as a six stratum structure subdivided into functional regions called corticies. The strata of brain were shown to be distinguishable by cell type.

In the second section, the mammalian nervous system was parametized. The mammalian brain was selected as a model because it alone produces the intelligence man seeks to replicate with machines. The main processing region of the brain was shown to be the neocortex. The neocortex was shown to be divided into functional regions called corticies. All corticies have six strata. The corticies are distinguished by variations in the stratum thickness and the cell types per stratum. Cell types were shown to be distinguished by cytoarchitectural properties. A general cortex model was created that allowed nearly any specific cortex to be produced simply by changing a few of the terms that regulated stratum thickness and cell types. Similarly, a general neuron model was created that allowed nearly any specific neuron to be produced by changing the cytoarchitectural parameters. The general cortex and neuron model allow a wide variety of neural structures to be produced and are thus major contributions of this work.

The fundamental theme of this study was to produce massive scale neural networks that could be optimized over time. Such an approach was accomplished by encapsulating all the cortex and cell parameters into an artificial chromosome. The artificial chromosome could then be used with a genetic algorithm to produce new (offspring) networks in a pseudo-random search for the ideal. In order to encapsulate the cell and cortex parameters a mathematical model of the organization of genes was produced. The super set of all genes G was defined and all manners of sub-categories such as cell types, tkGs, and functional genes, Gf, were also identified. By defining cell and cortex parameters into sets, information could be managed in a meaningful way. The organization of this cell and cortex data was a major accomplishment of the work. A method was developed to encapsulate both functional and probabilistic data into the artificial chromosome. Functional data describes the magnitude and purpose of a gene while probability data defines likelihood of the gene expressing. This combination of functional and probabilistic data allowed alleles to be represented simply by varying the probabilistic term. The new format allows optimization to be done by changing

only the probabilistic term, while leaving the functionality intact. This approach is a much closer model to nature than the mutagenic process typically employed. Thus the dual term encapsulating technique for gene encryption is also a major accomplishment. To assist in the encryption the chromosome, a program was written to convert cell and cortex properties, data stored as a text file, into appropriate gene values.

In the third section, a program was written that produced neural networks from the artificial chromosomes. The program simulated the growth of the biological cells. By replicating the growth process, networks of similar topology to mammalian neural structures were produced. The ability to replicate these complex structures was a major accomplishment of this work. The simulation program was based on von Neuman's automata theories. Automata programs establish an array of cells. Each cell's operations are based solely on the artificial chromosome and the artificial chemical signals given off by neighboring cells. The expression of the genes follows digital logic, however, the cell behavior is analog. This digital to analog conversion is produced by averaging all the individual mechanisms of the respective genes. The growth simulation program is a nonlinear dynamical system, producing unique output patterns for very slight variation in the initial data. The production of a non-linear dynamical system from essentially digital data is a significant contribution to the field of chaos mathematics. The output of the simulation program describes what each cell in the network has become and what connections have been formed. The ability to produce an artificial neural network, based only on local mechanisms and the artificial chromosome is yet another major accomplishment of this work.

In the forth section the performance of the cell growth simulation program was analyzed. Several parameters used to define the growth environment were demonstrated to affect the final morphology of the neural networks and the processing time of the simulation program. These parameters include, growth rate, number of puberty iterations, and the absorption coefficient. The absorption coefficient has been found to have the most pronounced effect on the stratum morphology. An inverse relationship has been found to exist between the size of the neural clusters and the absorption coefficient. An error analysis was also performed on the effect of varying the morphology regulating parameters. The error analysis showed that there is a proportional relationship between the encoded parameters and those collected from the simulated network. This relationship was found to be stable over a wide range of operating conditions. Most significantly, the error analysis found that a representative gradient field cannot develop with a radius below 1 UCL. The analysis also

showed that the ligand radius must have a value above 2 UCL and below 25% of the neural network's grid length to ensure pattern stability. The quantification of the operating parameters is a major contribution of this work.

A technique was developed to integrate multiple corticies in the forth sections. The technique allows representation of specific, associative, symmetric, and direct information to be passed between the corticies. This is accomplished by classifying the input and output types of the general cortex model. The integration of the different information types allows functionally different corticies to cooperate. In essence, the over all interconnection logic of corticies has been defined. More significantly, it has been shown that multiple corticies can be directly connected to form a principle pathways. These primary pathway define the macro-performance of the artificial network systems. A fractal relationship has been shown to exists between single neurons, neural-clusters, and corticies. This relationship suggests that the integration of multiple primary pathways can be used to train a massive, multiple cortex neural network with Hebbian rules. While this study has produced some radical network designs, it is significant to note these structures have been shown to be enhancements of traditional systems rather than wholly new entities.

In short, this dissertation has produced a new method to design massive scale neural networks. The technique is modeled after biological nervous systems. The technique stores all network parameters in an artificial chromosome. A program uses the chromosome to simulated the growth of the neurons in order to produce the neural networks. A large scale structure call a cortex has been defined and the integration of these corticies has also been discussed. This study has defined a robust data processing structure that can be used for future neural-computing applications.

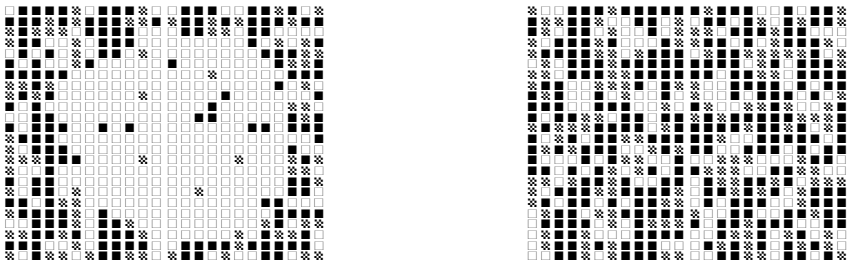
5.5 Retrospective

There is one other area of future research that was not discussed in the main body of the dissertation as it was felt to be distracting from an already abstract study. The discussion is on medical implications of the simulation program and suggests a development for cancer. The dissertation as it stands now, is devoted to modeling massive scale neural networks. The

tool for designing these networks, the cell growth simulation program, took on its own life and produced some truly unexpected results.

Perhaps the most exciting finding of the study was derived from an inadvertent mistake. This finding suggests a radical new interpretation of how genes regulate cellular activity and suggests an explanation for cancer. In the early phases of the study, numerous unsuccessful attempts were made to produce cortex strata with an even distribution of cell types (homogenous tissues.) In these early attempts one cell type would dominate the central portion of the grid forming a blob-like shape. Only on the edges of the artificial tissue array would different cell types develop. Initially this result was considered to be an artifact of incorrect parameters. To achieve the homogenous tissues the LRP values and dispersion coefficients were adjusted. It was, in fact, some time before the significance of these findings was realized and I can not take full credit for the discovery. I was showing my work my brother who commented that artificial tissues looked like a tumor. The comment was largely ignored for nearly a year because I had not yet the knowledge to interpret the statement nor the understanding of automata-interactions needed to analyze the unexpected result.

The analysis has three parts. First, a brief overview of cancer is given to highlight how the new theory diverges from current thought. Next, the dynamics of the cell growth simulation program are discussed in the context of gene regulation. Finally, a model for gene regulation and cancer formation is given. A brief discussion on the implication of the new gene regulation theory follows the analysis.



(a) (b)

Figure 5.53 Effects of unregulated cell growth: (a), cancerous; (b) normal.

Cancer is a condition when the cells cease their normal functions, undifferentiate then grow and reproduce out of control. No one knows why or how exactly this happens, but it is believed to be brought on by both genetic and environmental factors. According to traditional wisdom, cancer occurs when a cancer-gene becomes active. It is believed that these cancer genes lay dormant until something triggers them into activity. The activation of these latent

genes can be metabolic or related to introduction of a carcinogen to body. The basic assumption in the traditional view is the presence of a latent "cancer-gene" that is benign until activation. This assumption leads researchers in two directions: 1) The identification of carcinogens and 2) the development of treatments that suppress the cancer genes once they have become active. The critical reader will note that the actual mechanism of this activation is widely ignored in this description. The purpose of the following section is to shed some light on the mechanism responsible for the onset of cancer by drawing analogies from the simulation model.

To have a discussion comparing the computer models developed in this dissertation to cancer may seem abstract, but is not entirely outside the domain of the sciences used to define the simulation program. The simulation program which has been treated thus far as a mere tool for producing the massive scale networks, will now be regarded as a window for viewing cellular activity. The simulation program is in fact a wonderful model of cellular activity and offers the researcher the same type of information regarding cellular activity that manufacturing simulation provide about factory performance. Before the results of any simulation can be reviewed interpreted or accepted the simulation program must be verified. Simulation verification is traditionally accomplished by loading the program with sample data and running it. The simulation is "verified" if the results obtained from the sample data agree with the collected data. In the case of the cell growth simulation, the preceding 250 pages have essentially been an extended verification. The ability to produce artificial tissues with the same physiological characteristics of the biological systems is essentially a proof of the validity of the simulation program. However, it is impossible to fully verify the simulation program because the artificial gene were generated, not replicated. The generation of the artificial genes was necessary because the actual values and characteristics are not known. Therefore we must accept that the simulation program produces valid results based an assumed data. Thus all discussions regarding the behavior of simulation are compromised. The simulation programming must be viewed as a proposed system, and the following discussion is thus theoretical in nature. The proposed system is one of many possible solutions. Until the underlying assumptions are proven, one must treat the simulation only as a possible explanation.

In an automata program the artificial cells determine their state based on local stimuli. It took some time to understand the true implications of "state." von Neuman used "state" to define one of all possible operating conditions. I originally believed that operating condition referred to the basic functions biologist typically define such as digestion, reproduction, ect. I originally thought I was expanding von Neuman's model by having the states represent cell types. However as the work proceeded, the entire distinction of unique cell types gave way. In

fact cells perform nearly all fundamental cellular activities simultaneously. Furthermore, as was previously discussed, "cell type" is a classification imposed by man, not nature. Cells are considered to be of a particular type if they display certain physical characteristics. These characteristics are latent in all other cells. The appearance of these physicals characteristics are the manifestation of the activation of particular genes. In other words, state refers to which genes are active. A gene can be either zipped or unzipped. In the zipped case, the peptides are not exposed and no transcription occurs. Vice versa, the unzipped case allows transcription to occur. Transcription will occur at some rate "r" so long as the gene is unzipped. Thus cells can be treated as digital systems. A cell contains an instruction set "G". For each instruction, or gene, the on (unzipped) or off (zipped) case is possible. Assuming a state is defined by which genes are active. There are 2^G possible states, where G represents the number of genes in a cell. In chapter 3, G was defined as the superset that contains all genes, G was divided into smaller subset, each of which defined a particular cell types. Thus a cell type that has ${}_{tk}G_s$ genes has 2^{tkGs} states.

Which brings us back to cell type, and the questions that must be answered, "is there a gene that when expressed restricts G to ${}_{tk}G_s$? or is ${}_{tk}G_s$ merely a manifestation of local stimuli? There is no answer at this time, though I have assumed the prior in the dissertation. For the purpose of the discussion we shall assume that differentiation is predominantly an artifact and ${}_{tk}G_s$ is a transient state supported by local stimuli. This assumption is in contrast to traditional thought which views differentiation as permanent. In the following section this assumption will be supported by other observations.

To summarize thus far, we have decided to assume the simulation model to be a valid model of cell dynamics, to define "state" to be a reflection of the genes expressed, and to assume the "cell type" to be a transient state supported merely by local stimuli.

Continuing, let us examine the simulation program in grueling detail. The program was originally designed to produce neocortex structure by replicating differentiation. The program reads an artificial chromosomes and assigns values to cells. The chromosome is comprised of genes. Each gene had 2 parameters, the ID number and the LRP value. The ID number used to identify the particular behavior of that gene. The LRP value was originally intended to be a weighted term. The simulation program would generate an array of cells. Each cell would be assigned a growth rate coefficient, which models the nutrient supply rates. Each cell would grow at a rate defined by this maturation parameter. Each cell would secrete ligands at a rate proportional to the product of the growth rate parameter and the LRP value. Once the cell grows to a certain size, it would differentiate based on the highest concentration of ligands.

In truth both the LRP term and the maturation terms were introduced into the model late, late at night. I created those terms mostly on a hunch, and full explanation of there meaning came later. I knew, fundamentally that some cells grew faster than others and that some genes are preferred, however when I coded the terms I did not knew the mechanism. The growth rate term was readily explained by minute differences in the nutrient supplies. However, the LRP weighting term remained an enigma for quite some time.

Reviewing the model again, one sees the following: Cells grow at rate "r". Cells release ligands at rate "r*(LRP)". Cells differentiate when they mature, a point determined my the mature parameter "m". At least this was the design intent. In actuality, the cells grew at rate "r" secreted ligands at rate "r*(LRP)" and differentiated at "m*(LRP)". In other words, there is a constant "m*(LRP)" which determines when a gene will express. Simply put, I had accidentally incorporated a threshold term into the model.

Reviewing the workings of the model further, one finds the cells grow, release ligands, and when local concentrations of a particular ligand exceeds its respective threshold, the gene will express. The concept of a threshold of expression has profound implications. Biologists tend to speak of gene regulation in analog terms, while the simulation model only predicts digital behavior governed by simple threshold values.

There is still more to the puzzle. There are two remaining problem; 1) What is responsible for the analog measurements and 2) what is the mechanism for control. Both questions are answered by pulse width modulation. When the data is critiqued from the view point of an investigator, one finds two simple facts:

1) Something turns genes on,
2) Something turns genes off.

Yet these facts do not stand completely alone. The previous discussion theorized that genes are activated once their respective thresholds have been reached. However, once a gene is on, what turns it off? Again the answer is found in the simulation program. One will recall that a necessary criteria for the simulation program was that each stratum had to have at least two cell types. Initially I thought cells had to chose (differentiate) between at least two different cell types. However, when one makes the assumption that cell type is little more than a gene expression, then one must also assume that certain genes exist in a Darwinian competition with each other. These groups compete for the right to express. There is a subtle implication here; among a group, only one gene can express at a time.

In chapter 2 it was explained that the DNA exists as a knot in the nucleus, and only those genes on the surface can be expressed. Furthermore changes to the physical characteristics

called conformational changes alter the set of genes that can be expressed. It was previously suggested that conformational changes may be responsible for separating tkG from G. Now we introduce the postulate that the expression of each gene yields a conformational change. If one assumes that only one gene can be expressed from a group at a time, then one must postulate that the expression of one gene from a group evokes a conformational change that simultaneously deactivates the other genes from that group. This implies the genes within a group regulate themselves. From this discussion the term "regulation group" will be used to describe those genes which interact and regulate themselves.

There is yet a further implication of the model, that of pulse width modulation, (PWM). PWM is a technique employed by digital circuits designers to approximate analog output. The technique divides the time domain into discrete periods. The maximum and minimum analog values are obtained by leaving the digital signal on or off respectively for the entire time period. The digital signal is turned on a fraction of the period proportion to the desired analog value. Thus a 50% value is obtained by leaving the digital signal on for half of the time period. Recall that each gene has a specific threshold of activation. Consider a regulation group comprised of two genes, A and B. For this exercise, assume A has just been activated. Gene A produces protein (P_A) at some rate (R_A). P_A is consumed, metabolized or otherwise used at a rate of C_A. Meanwhile, the protein produced from gene B, P_B which was resident from the previous cycle is consumed at a rate C_B. Thus for time period 1, the concentration of P_A is increasing while that of P_B is decreasing (Figure 5.54).

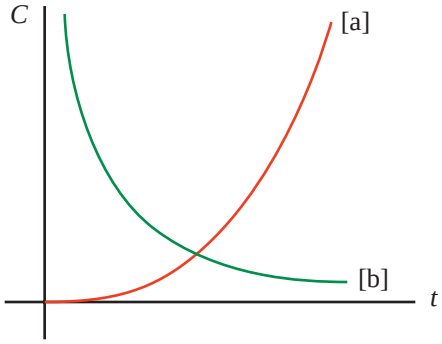


Figure 5.54 Concentrations of proteins P_A and P_B vs. time for the first time period.

In this simple 2 gene regulation group P_A and P_B are the ligands for B and A respectively. At some time, the concentration of P_B will be below the threshold of A (T_A) and the concentration of P_B will be above the threshold of B (T_B). At this time, A is subcritical, meaning that nothing is causing it to remain unzipped and B supercritical, meaning there is

sufficient ligand concentrations to cause B to unzip. The instance B-unzips, a conformational change occurs to initiate A to zip. During the second period, the concentrations of P_B will increase, and those of P_A will decrease following the same mechanisms described above. In theory the cell will cycle between producing P_A and P_B indefinitely, (Figure 5.55).

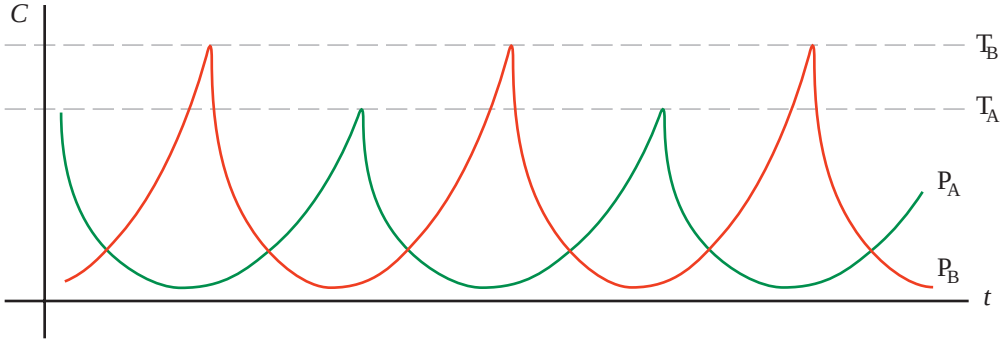


Figure 5.55 Concentrations of P_A and P_B vs. time for extended duration. Note T_A and T_B are the threshold concentration values for A and B respectively.

Various combinations of production and consumption rates of P_A and P_B combined with various threshold values produce a system where genes within a regulation group can be active for an unequal fraction of the time period, (Figure 5.56a). Figure 5.56b shows the digital signal corresponding to the production of P_A and P_B. When one considers a single time period then the protein production graph (Figure 5.56b) appears much like a signal being controlled by PWM. In this example A and B are active roughly 25% and 75% of the time respectively. Even though A and B are digital in nature the PWM effect allows their expression to appear analog.

The two gene regulation group is of course a gross simplification of the system. It is likely that regulation groups comprise dozens, perhaps hundreds of genes. The example also simplifies the problem by having the proteins be the ligands, and the genes acting independently. In the biological system it is probable that several genes linked as a conglomerate producing a chain-reaction type response where the activation of one gene causes the activation of many others. Also it is unlikely that the gene regulation groups are so local. It is most likely the ligands originate from many other cells from far distant locations, more like the computer model. Added complexity and multiple stages do not affect the basic mechanism.

The simulation program is interesting in that it provides a glimpse at the complex, dependent interactions that biologist have been theorizing. More important, it allows one to destabilize the system and observe the results. Which brings us back to the original purpose of the discussion. When the simulation program was first written, the coefficients were not given the correct values. Later, when the original simulation experiments were reviewed, one finds

unacceptable large consumption rates, and low production rates. In other words, the original simulations had P_A less than or equal to C_A. Under such conditions, the concentration of A will not become critical, and B will therfore not activate. This produces a condition where the cell will not have enough P_A and absolutely no P_B.

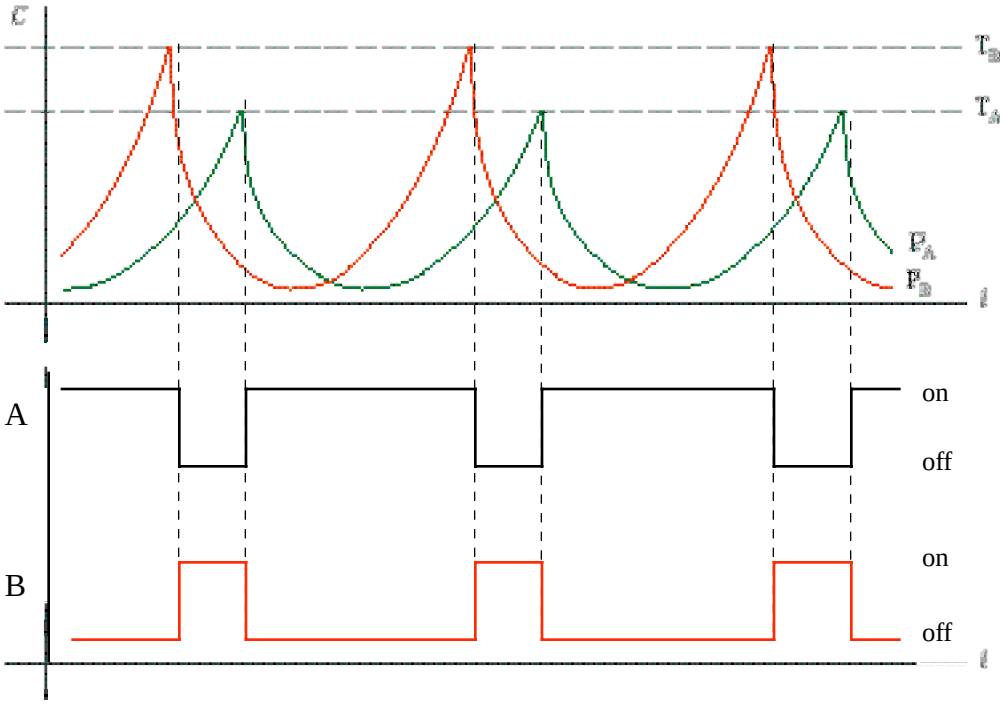


Figure 5.56 Generation of a pulse width modulated signal from two genes' co-regulation. (a) protein production vs time, top, and (b) gene expression vs time, bottom.

One last fact is required prior to putting forth a complete theory for cancer formation. As the years take their toll, there is a gradual reduction of one's metabolism. The slowing of the metabolism is measured by a decrease in hormone levels and a general slowing of all functions. In the context of the previous discussion, the production rates and consumption rates all decrease. In the case where the production rates decrease slightly faster than the consumption rates, a dangerous situation becomes apparent. Over time, due to the slowing of one's metabolism it is possible that, one of the genes among a regulation group will not produce enough ligands to cause other genes to activate. This hypothesis has profound implications.

In contrast to the traditional view which requires a external carcinogen to enter the body, the new model provides a mechanism where by the natural decrease of metabolic rate due to aging produces a condition where control signals that governs normal cell behavior is interrupted. The fact that the new model is dependent on aging correlates well to the medical

data that clearly shows cancer to be an age related disease. Secondly, the traditional view suggest that a latent "cancer" gene becomes active, while the new model predicts that a gene becomes inactive. Most significantly the new model implies an explanation for one of the more contradicting findings to the single cancer gene theory, namely the apparent ability of some individuals to have a "cancer" gene but never suffer form the disease. If one assume that there is single gene responsible for cancer then one must assume any individual carrying it will develop the disease. However the historic data indicates that only a percentage of the individuals who carry a cancer gene will ever develop the disease. By assuming genes operate in regulation groups, one must conclude that there are "cancer groups" as opposed to single genes. If one incorporates the concept of alleles, chapter 2, an interesting exercise can be constructed.

Assume there is a regulation group comprised of genes A and B. Further assume B has 4 alleles, B_1, B_2, B_3, and B_4. Let the combination of A and B_1 be cancer causing. Here we have produced a system where the is a 25% chance of cancer if A is present and 100% chance if B_4 is present. Continuing with the exercise, assume A has 3 alleles, A_1, A_2 and A_3. If we only allow cancer to occur with the combination of A_1 and B_4 we find a situation where there is a 1 in 12 chance of cancer occurring. In short the new model agrees with historic data that show a "cancer" gene to only induce the disease a fraction of the time.

The concept of cancer inducing regulation groups has a profound impact on cancer research. The new model suggest that the gene currently called the "cancer" gene may not be the problem. Rather it is the "cancer" gene's complement, or regulation partner, that is to blame. Unfortunately it will be nearly impossible to directly measure the presence of the complimentary gene because it is dormant at the time that the cancer is active.

There is one further implication. The model predicts that cancer is reversible. According to the new model, cancer occurs when one of the regulating signal remains subcritcal. If the deficient substance is introduced to make the ligand levels super-critical, the dormant gene should become active again and return the cells to their non-cancerous state.

The simulation program suggests new avenues of research for biologist. The program suggest that genes exist in regulation groups. These groups must be identified and analyzed to understand the nature of the disease. Furthermore the model suggest that cancer can be treated by reactivating the gene that lays dormant during the disease.

5.5.1 Summary

A simulation program has been developed and shown to be representative of the biological system. The program produced some unexpected and intriguing results; 1)

unpredicted intercellular interactions and 2) tumor like growths under certain types of stimulation. Upon analysis a new theory for gene regulation of cells was created and a possible explanation for cancer formation was put forth.

The simulation model indicates that genes function in groups. Their interactions are subject to a threshold stimulation. A gene is activated by a signal reaching the threshold value. It is deactivated when its complimentary gene become active. In such a way, genes within the group are auto-regulating. Through differences in protein production rates, consumption rates and threshold values, analog behavior is observed though a pulse width modulation process.

The traditional model of cancer formation is based on the concept of latent cancer genes and the introduction of carcinogens to the body. This traditional model has two main short comings. First, only a percentage of the individuals caring a cancer gene will develop the disease. Second, all the historic data indicates cancer is a age related condition, where as the traditional model does not stipulate age as a causative agent. The new model proposes that genes funtion in groups and only certain combination of alleles may produce cancer. Secondly the new model suggests that the slowing of the metabolism disrupts the fundamental signals that regulates the group of genes involved in cancer formation. Lastly the new model suggest that cancer can be reversed by supplementing or otherwise restoring the deficient signal.

This research though initially intended to produce massive scale neural networks has inadvertently opened many new areas for genetic and cancer research.

6. BIBLIOGRAPHY

1 LeClair, S.R., Abrams, L., and Matejka, R.F., "Qualitative Process Automation: Self Directed Manufacturing of Composite Material", AI EDAM, Vol. 3, No. 2, pp. 125-136, 1989.

2 Chester, M., "Neural Networks, a Tutorial", PTR Prentice Hall, Englewood Cliffs, NJ, 1993.

3 Durkin, J., "Expert Systems, Design and Development", Macmillan Publishing Company, New York, NY, 1994.

4 Winston, P.H., "Artificial Intelligence, 3rd Ed.", Addison-Wessley Publishing Company. Reading, MA, 1992.

5 Watkins, P.R., and Eliot, L.B., Eds., "Expert Systems in Business and Finance, Issues and Application", John Wiley & Sons, New York, NY, 1992.

6 Davis, L., "Handbook of Genetic Algorithms", Van Nostrand Reinhold Co., New York, NY, 1991.

7 Werbos, P.J., "The Roots of Back Propagation, From Order Derivatives to Neural Networks and Political Forecasting", John Wiley & Sons, New York, NY, 1993.

8 McCulloch, W.S., and Pitts, W., "A Logical Calculus of Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, Vol. 9, pp. 127-147, 1943.

9 Fildes, J.M., Milkovich, S.M., Altkorn, R., Haidle, R., and Neatrour, J., "In Situ Infrared Spectroscopy and Neural Network Analysis of Composite Cure Monitoring", Northwest University, BIRL., 1801 Maple Avenue, Evanston, IL, 60201, 1993.

10 Morgan, N., "Big Dumb Neural Nets: A Working Brute Force Approach to Speech Recognition", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 4462-4465, June 27-29, 1994.

11 Codrington, C.W., and Tenorio, M.F., "Adaptive Gain Networks", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 339-344, June 27-29, 1994.

12 McIlwain, Personal Interview.

13 Freeman, J.A., "Simulating Neural Networks with Mathmatica_ ",Addison-Wessley Publishing Company. Reading, MA, 1994.

14 Paradis, R., and Dietrich, E., "Concept Development in a Scaffold Neural Network" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2339-2343, June 27-29, 1994.

15 Wan, K.W., Tian, Q., Low, K.C., Lau, S.L., and Lui, H.C., "An Integrated Multiple Neural Network Architecture For Reading Alphanumeric Characters in Complex Scenes" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 4384-4389, June 27-29, 1994.

16 Taber, W.R., and Deich, R.O., "Fuzzy sets and Neural Networks", Proceedings of the First Joint Technology Workshop on Neural Networks and Fuzzy Logic. NASA/University of Houston, Houston: May 2-3, 1988.

17 Taber, W.R., and Deich, R.O., Simpson, P.K., and Fagg, A.H., "The Recognition of Orca Calls with a Neural Network", International workshop on Fuzzy Application, Iizuka, Japan, August 20-24, 1988.

18 Kosko, B., "Bi-directional Associative Memories", (Neurocomputing), IEEE Transactions on Systems, Man and Cybernetics, Vol. 18, pp. 49-60, 1988.

19 Carpenter, G.A., and Grossberg, S., "Art2: Self-Organization of Stable Category Recognition Codes for Analog Input Patters", (Neurocomputing), Applied Optics, Vol. 26 pp. 4919-4930, 1987.

20 Ly, S., and Choi, J.J., "Drill Monitoring using ART-1", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1226-1229, June 27-29, 1994.

21 Heileman, G.L., Georgiopoulos, M., and Hwang, J., "A survey of Learning Results for ART1 Networks", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1222-1225, June 27-29, 1994.

22 Pellianisz, A., and Llinás, R., "Tensor Network Theory of the Metaorganization of Functional Geometries in the Central Nervous System", (Neurocomputing), Neuroscience, Vol. 16, pp. 245-273, 1985.

23 Albus, J.S, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller ($CMAC^1$)", Journal of Dynamic Systems, Measurement, and Control, September, pp. 220-227, 1975.

24 Albus, J.S, "Data Storage in the Cerebellar Model Articulation Controller ($CMAC^1$)", Journal of Dynamic Systems, Measurement, and Control, September, pp. 228-233, 1975.

25 Miller, W.T., "Learning Dynamic Balance of a Biped Walking Robot" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2771-2776, June 27-29, 1994.

26 Wen, R.C., Ker, J.S., Kuo, Y.H., Liu, B.D., and Chang, G.W., "A CMAC Neural Network Chip for Color Correction", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1943-1948, June 27-29, 1994.

27 Krandel, E.R., Schwartz, J.H., and Jessel, T.M., "Principle of Neural Science", Appleton & Lange, Norwalk, CT, 1991.

28 Anderson, J.A., Pellionisz, A. and Rosenfeld, E. (Eds.), "Neurocomputing 2, Directions for Research", MIT Press, Cambridge, MA, 1990.

29 Mountcastle, V.B., "An organization principle for cerebral function: the unit module and the distributed system", The Mindful Brain, Schmitt, F.O., (Ed.), MIT Press, Cambridge, MA, 1978.

30 Zeki, S., and Shipp, S., "The functional logic of cortical connections", Nature, Vol. 335, pp. 311-317.

31 Bieszczad, A., "Neurosolver: A Step Toward a Neuromorphic General Problem Solver", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1313-1318, June 27-29, 1994.

32 Anderson, J.A., Rossen, M.L., Viscuso, S.R., and Sereno, M.E., "Experiments with Representation in Neural Networks: Objects Motion, Speech, and Arithmetic", (Neurocomputing), Synergetic of Cognition, Haken, H., and Stadler, M. (Eds.), Berlin: Springer, 1990.

33 Edelman. G.M., and Finkel, L.H., "Neuronal Group Selection in the Cerebral Cortex", (Neurocomputing), Dynamic Aspects of Neocortial Function, Edelman, G.M., Gall, W.E., and Cowan, W.M. (Eds), Wiley-Interscience, New York, NY, pp. 653-695, 1984.

34 Pearson, J.C., Finkel, L.H., and Edelman, G.M., "Plasticity in the Organization of Adult Cerebral Cortical Maps: A Computer Simulation Based on Neuronal Group Formation", (Neurocomputing), The Journal of Neuroscience, Vol. 7, pp. 4209-4223, 1987.

35 Werbos, P.J., "Neural Networks & the Human Mind: New Mathematics Fits Humanistic Insight", IEEE International Conference on Systems, Man, and Cybernetics. 1970.

36 Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning Internal Representation by Error Propagation", Parallel Distributed Processing: Exploration in the Microstructures of Cognition, Vol. 1, Runelhart, D.E., and McClelland (Eds.), MIT Press, Cambridge, MA, pp. 318-362, 1986.

37 "Proceedings of the Neural Network Development Conference", Army Research Laboratory, Adelphi, MD, February 2, 1994.

38 Estévez, P.A., and Okabe, Y., "Genetic Synthesis of Piecewise - Linear Neural Networks", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1783-1786, June 27-29, 1994.

39 Lange, J.M., Voigt, H.M. and Wolf, D., "Growing Artificial Neural Networks Based on Correlation Measures, Task Decomposition and Local Attention Neurons", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1355-1358, June 27-29, 1994.

40 Williams, R.J., and Zisper, D., "Experimental Analysis of the Real-Time Recurrent Learning Algorithm", Connection Science, Vol. 1, No. 1, MIT Press, Cambridge, MA, pp. 87-111, 1989.

41 Williams, R.J., and Zisper, D., "Gradient-Based Learning Algorithms for Recurrent Learning Algorithm", College of Computer Science Technical Report NU-CSS-90-9, Northeastern University, Boston, MA, 1990.

42 Rojas, R., "The Fractal Geometry of Back Propagation", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 233-238, June 27-29, 1994

43 Abbas, H.M., Bayoumi, M.M., "Anti-Hebbian Rule for Faster Backpropagation Learning", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 101-106, June 27-29, 1994.

44 Leung, S.H., Luk, A., and Ng, S.G., "A Weight Evolution Algorithm for Multi-layered Network", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 892-896, June 27-29, 1994.

45 Hebb, D.O., "The Organization of Behavior", John Whiley and Sons Inc., New York, NY, 1949.

46 Brown, T.H., Chapman, P.F., Kairiss, E.W., and Keenan, C.L., "Long-Term Synaptic Potentiation", (Neurocomputing) Science, vol. 242, pp. 724-728, 1988.

47 Lisman, J., "The CaM Kinase II Hypothesis for the Storage of Synaptic Memory", TINS, Vol. 17, No.10, pp 406- 412, 1994.

48 Chatterjee, C., and Roychowdhury, V., "A New Training Rule for Optical Recognition of Binary Character Images by Spatial Correlation", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 4095-4100, June 27-29, 1994.

49 Palmieri, F., "Hebbian Learning and Self-Association in Nonlinear Neural Networks", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1258-1263, June 27-29, 1994.

50 Forman, T., and Lórincz, A., "Robustness of Hebbian and Anti-Hebbian Learning", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 731-735, June 27-29, 1994.

51 Brarath, R., and Drosen, J., "Neural Network Computing", Windcrest_ /McGraw-Hill, New York, NY, 1994.

52 Hylander, P., and Meador, J., "Object Oriented VLSI Design Automation for Pulse Coded Neural Networks", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1825-1829, June 27-29, 1994.

53 Rückert, U., Funker, A., and Pintaske, Ch., “Accelerator Boards for Neural Associative Memories”, Neurocomputing, Vol. 5, pp. 39-49, 1993.

54 Asanovic, K., Beck, J., Feldman, J., Morgan, N., and Wawrzynek, J., "A Super Computer for Neural Computation", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 5-9, June 27-29, 1994.

55 Hirai, Y., "Hardware Implementation of Neural Networks in Japan", Neurocomputing, Vol. 5, pp. 3-15, 1993.

56 van Keulen, E., Colak, S., Withagen, H., and Hegt, H., "Neural Network Hardware Performance Criteria", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1885-1888, June 27-29, 1994.

57 Shi, B.E., Roska, T., and Chua, L.O., "Random Parameter Variation in Analog VLSI Neural Networks for Linear Image Filtering", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1917-1922, June 27-29, 1994.

58 Arreguit, X., and Vittoz, E.A., "Analog VLSI Hardware: What is Missing for Industrial Realizations ?", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1883-1884, June 27-29, 1994.

59 Withagen, H., "Implementing Backpropagation with Analog Hardware" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2015-2017, June 27-29, 1994.

60 Abusland, A., and Lande, T.S., "An Analog Continuous-Time Micropower Hopfield Net", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1860-1865, June 27-29, 1994.

61 Montalvo, A.J., Paulos, J.J., and Gyurcsik, R.S., "An Analog VLSI Neural Network Architecture with On-Chip Learning", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1364-1368, June 27-29, 1994.

62 Beichter, B., Brüls, N., Sicheneder, E., Ramacher, U., and Klar, H., "Design of a General-Purpose Neural Signal Processor", Neurocomputing, Vol. 5, pp. 17-23, 1993.

63 Chung, P.C., Tsai, C.T., and Sun, Y.N., "Linear Quantization of Hebbian-Type Associative Memories in Interconnection Implementations", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1092-1096, June 27-29, 1994.

64 Zaman, R.U., and Wunsch, D.C., "An Adaptive VLSI Neural Network Chip" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2018-2021, June 27-29, 1994.

65 Elais, J.G., and Northmore, D.P.M., “Programmable Dynamics in an Analog VLSI Neuromorph”, 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2028-2033, June 27-29, 1994.

66 Gopalsamy, K., "Convergence in Neural Networks With Interneural Transmission Delays", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 976-979, June 27-29, 1994.

67 Delgado-Restituto, M., and Rodríguez-Vázquez, A., "Current-Mode Building Blocks for CMOS-VLSI Design of Chaotic Neural Networks" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 3973-3977, June 27-29, 1994.

68 Douglas, R.J., Mahowald, M.A., and Martin, K.A.C., "Hybrid Analog-Digital Architectures For Neuromorphic Systems", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1848-1853, June 27-29, 1994.

69 Masa, P., Hoen, K., and Wallinga, H., "70 Input, 20 Nanosecond Pattern Classifier", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1854-1859, June 27-29, 1994.

70 Myers, D.J., Vincent, M.J., and Orrey, D.A., "HANNIBAL: A VLSI Building Block for Neural Networks with on Chip Backpropagation Learning", Neurocomputing, Vol. 5, pp. 25-37, 1993.

71 Lo J.C., and Fisher, G., "HAVENN: Horizontally and Vertically Expandable Neural Networks" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2074-2077, June 27-29, 1994.

72 Van der Spiegel, J., Donham, C., Etienne-Cummings, R., Fernando, S., Mueller, P., and Blackman, D., "Large Scale Analog Neural Computer with Programmable Architecture and

Programmable Time Constants for Temporal Pattern Analysis", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1830-1835, June 27-29, 1994.

73 Lyon, R.F., and Mead, C., "Electronic Cochlea", *Analog VLSI and Neural Systems*, Mead, C., (Ed.), Addison-Wesley Pub. Co. , Reading, MA, pp. 279-302, 1989.

74 Abel, C., Park, J., Ismail, M., Lohiser, B., Justice, S., Bibyk, S., and Fiez, T., "A Switched - Current Silicon Cochlea", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1842-1847, June 27-29, 1994.

75 Bhadkamkar, N.A., "Binaural Source Localizer chip using subthreshold analog CMOS", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1866-1870, June 27-29, 1994.

76 Sivilotti, M.A., Mahowald, M.A., Mead, C.A., "Real-Time Visual Computations Using Analog CMOS Processing Arrays", Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference, Losleben, P. (Ed.), MIT Press, Cambridge, MA, pp. 295-312, 1987.

77 Mahowald, M.A. and Mead, C., "A Silicon Model of Early Visual Processing", Neural Networks: Vol. 1, Pergamon Press, New York, NY, 1988.

78 Mead, C., "Analog VLSI and Neural Systems", Addison-Wesley, Reading, MA, 1989.

79 Girod, J.P., Martin, G., Heit, B., and Bremont, J., "Image Segmentation by the Modelisation of the Biological Visual Systems", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2233-2238, June 27-29, 1994.

80 Wu, J.L., and Nishikawa, Y., "A Neural Network Model of the Binocular Fusion in the Human Vision", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 4169-4175, June 27-29, 1994.

81 Duong, T., Kemeny, S., Tran, M., Daud, T., and Thakoor, A., "Low Power Analog Neurosynapse Chips for a 3-D 'Sugarcube' Neuroprocessor", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 1907-1911, June 27-29, 1994.

82 Smith, A., and Dagli, C.H., "Controlling Industrial Process Through Supervised, Feedfoward Neural Networks", Computers Ind. Engng., Vol. 21 Nos. 1-4, pp. 247-251, 1991.

83 Cook, D.H, and Shannon, R.E., "A Predictive Neural Network Modeling System for Manufacturing Process Parameters", Int. J. Prod. Res., Vol. 30, No. 7, pp. 1537-1550, 1992.

84 Prof. Dr. -Ing., Dr. -H.C. Mult. W. Koenig, Dipl, Ing. F. Memis, and Assoc. Prof. I. N. Tansen, "Modeling of the Plunge Grinding Process with Self Learning System for Monitoring and Adaptive Control", Proceedings of the Artificial Nerual Networks in Engineering (ANNIE'94) conference, St. Louis, Missouri, pp. 981-986, November 13-16, 1994.

85 Thomas Fechner, Dietmar Neumerkel, and Ivo Keller, "Adaptive Neural Network Filter for Steel Rolling", Proceedings of the 1994 IEEE International Conference on Neural Networks, Orlando, Florida, pp. 3915-3920, June 27-29, 1994.

86 Y.S. Tarng and T.C. Li, "Adaptive Pattern Recognition of Drilling Chatter", Journal of Materials Processing Technology, Vol 48 pp. 247-253, 1995.

87 Dimla E Dimla, JNR., Paul M. Lister, Nigel J Leighton, “An Investigation of the Sensitivity of a Single-Layer Perceptron Neural Network to Tool Wear Inception During a Turning Process”, Proceedings of the Artificial Neural Networks in Engineering (ANNIE ’94) Conference, St. Louis, Missouri, pp. 867-873, November 10-13, 1996.

88 John P.H. Steele, Michelle Archuleta, and Tyler Hooley, “Detecting Cavitation In Hydraulic Pumps Using Artificial Neural Network”, Proceedings of the Artificial Neural Networks in Engineering (ANNIE ’96) Conference, St. Louis, Missouri, pp. 909-914, November 10-13, 1996.

89 Yu-To Chen, Pratap Khedkar, and Maarten P. Ter Weeme, “A Nerual Network Prediction for Reengineering of Resin Manufacture”, Proceedings of the Artificial Neural Networks in Engineering (ANNIE ’96) Conference, St. Louis, Missouri, pp. 855-860, November 10-13, 1996.

90 Xioyun Sun, Bruce Golden, Ohseok Kwon, and Edward Wasil, “Enhancing the Performance of Neural Network Models for the Wire Bonding Process”, Proceedings of the Artificial Neural Networks in Engineering (ANNIE ’94) Conference, St. Louis, Missouri, pp. 1041-1047, November 13-16, 1994.

91 Kenji Ohshima, Akira Hirai, and Satoshi Yamane, “Knowledge Based Information Processing of the Weld Pool Using Neural Network in the Robotic Welding”, 1995 International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies, pp. 47-51, 1995.

92 Bingzhe Jin, Wenhuan Liu, and Kenji Ohshima, “Control of Weld Pool Width and Cooling Rate in Circumfrential GTA Welding of Pipe by using Neural Network Model”, 1995 International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies, pp. 41-46, 1995.

93 David W. Coit, Jay Billa, Darren Leonard, Alice E. Smith, Willian Clark, and Amro El-Jaroudi, “Wave Solder Process Control Modeling Using a Neural Network Approach”, Preceedings of the Artificial Neural Networks in Engineering (ANNIE ’94) Conference, St. Louis, Missouri, pp. 999-1004, November 13-16, 1994.

94 Marzuki Khalid, Sigeru Omatu, and Rubiyah Yusof, “Adaptive Fuzzy-Neuro Control with Application to a Water Bath Process”, Proceeding of the 1994 IEEE Conference on Control Applications, Glasgow, United Kingdom, pp 173-178, 1995.

95 Kent Graviss and Jacek M. Zurada, “Optimal Temperature Control of a Dual Chamber Plant using Neural Networks”, Preceedings of the Artificial Neural Networks in Engineering (ANNIE ’95) Conference, St. Louis, Missouri, pp. 581-586, November 12-15, 1995.

96 A. Carriere, A. Cela and Y. Hamam, “Neural Network Based Adaptive Control of a Non-Linear System: Application to a Thermal Process”, 1994 IEEE International Conference on System, Man, and Cybernetics, San Antonio, Texas, pp. 1133-1138, 1994.

97 H. Youlal, A. Kada, M. Haloua, A. Majdoul, and M. Ramzi, “Multilayered Neural Networks for Identification and Control of a Multivariable Distillation Process”, Proceedings of the 1994 IEEE Conference on Control Applications, Glasgow, UK, pp. 289-294, 1994.

98 G.R. Luecke and J.C. Edwards, “Learning Control of Fluidized Bed Combustors Using Neural Networks”, Preceedings of the Artificial Neural Networks in Engineering (ANNIE ’94) Conference, St. Louis, Missouri, pp. 923-928, November 13-16, 1994.

99 P. Vega, J.M. Zamarreno, S. D'Amico and L. Alonso, "Modeling and Control of an Industrial Processing using Neural Networks", Preceedings of the Artificial Neural Networks in Engineering (ANNIE '94) Conference, St. Louis, Missouri, pp. 1029-1034, November 13-16, 1994.

100 Shiuh-Tarng Chiang, Ding-I Liu, An-Chen Lee and Wei-Hua Chieng, "Adaptive Control Optimization in End Milling using Neural Networks", International Journal of Machine Tools & Manufacture, Vol. 34 No. 5 pp. 637-660, 1995.

101 Park, S.J., and Yang, J.S., "Intelligent Process Control: Application of Neural Networks to Glass Melting Furnace Control", Proceeding of the 2nd International Conference on Expert Systems for Development, Bangkok, Thailand, pp. 255-260, March 28-30, 1994.

102 Scholes, S.R., and Greene, C.H., "Modern Glass Practice: Seventh Revised Edition", Cahners Publishing Co. Inc., Marietta, OH, 1975.

103 Pao, Y.H., Phillips, S.M., and Sobajic, D.J., "Neural-Net Computing and the Intelligent Control of Systems", Int, J. Control, Vol. 56, No. 2, pp. 263-289, 1992.

104 White, D., and Sofge, D., (Eds.), "Handbook of Intelligent Control: Neural, Adaptive and Fuzzy Approaches", Van Nostrand Reinhold Co., New York, NY, 1992.

105 Sofge, D.A., and White, D.A., "Neural Network Based Process Optimization and Control", Proceedings of the 29th Conference on Decision an Control, IEEE, Honolulu, Hawaii, pp.3270-3276, 1990.

106 White, D., and Sofge, D., "Neural Network Based Control of Composite Manufacturing", Intelligent processing of materials : presented at the Winter Annual Meeting of the American Society of Mechanical Engineers, Dallas, TX, November 25-30, pp. 89-97, 1990.

107 Wie Li and Zuowei Wu, "A Self-Organizing Fuzzy Controller Using Neural Network", Proceedings of the 1994 ASME International Computers in Engineering Conference and Exhibition, Minneapolis, Minnesota, pp. 807-812 September 11-14, 1994.

108 Kouichi Katsumata, Toshinobu Matsuoka, Yoshihisa Ishida and Takashi Honda, "Predicative I-PD Controller for Processes with Long Deadtimes Using Neural Networks", The 1994 IEEE Internation Conference on Neural Networks, Orlando, Florida, pp. 3154-3157, June 27-29, 1994.

109 Mingwang Zhao, "Neural-Net-Based Adaptive PID Regulator with Attenuating Excitation Signal", Proceedings of the American Control Conference, Baltimore, Maryland, pp. 2931-2932, June, 1994.

110 Mingwang Zhoa, "Neural-Net-Based Model-Free-Self-Tuning Controller with On-Line Self-Learning Ability for Industrial Furnace", , Proceedings of the 1994 IEEE Conference on Control Applications, Glasgow, UK, pp. 423-426, 1994.

111 Lin, C.S., Cheng, Y.H.E., and Kim, H., "Radial Basis Function Networks of Adaptive Critic Learning", 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 903-906, June 27-29, 1994.

112 Krandel, E.R., Schwartz, J.H., and Jessel, T.M., "Principle of Neural Science", Appleton & Lange, Norwalk, CT, 1991.

113 Campbell, N.A., “Biology, 3rd Ed.”, Benjamin/Cummings Publishing Co. Inc., Red Wood, CA, 1992.

114 Halliday, D., and Resnick, R., “Physics: Part 2, 3rd Ed., extended version”, John Wiley & Sons., New York, NY, 1986.

115 Hameroff, S., Dayhoff, J., and Koruga, D., “Cytoskeletal Conformational Automata: Intra-Neuronal Support of Neural Networks”, IEEE 0-7803-0720-8/92, pp. 84-88, 1992.

116 Martin, J.H., "Neuroanatomy: Text and Atlas", Elsevier, New York, NY, 1989.

117 McIlwain, J.T., "Large Receptive Fields and Spatial Transformations in the Visual System", (Neuroscience), International Review of Physiology, Neurophysiology II, Porter, R. (Ed.), Vol. 10, pp. 223-248, 1976.

118 Fair, A., Personal interview 4/5/95

119 “knot theory” Popular Science, no 7, 1994.

120 Gerbi, S.A., Personal interview 3/2/95.

121 Darwin, C., “On the origin of species”, in “A concordance to Darwin's Origin of species, first edition” edited by Barrett P.H., Weinshank, D.J., and Gottleber, T.T., Cornell University Press, Ithaca, N.Y. 1981.

122 Crevier, D., "AI, The Tumultuous History of the Search for Artificial Intelligence", Basic Books, New York, NY, 1993.

123 Barlow, B., "Single Units and Sensation: A Neuron Doctrine for Perpetual Psychology?", (Neuroscience), Perception, Vol. 1, pp. 371-394, 1972.

124 Anderson, J.A., “A model for memory using spatial correlation functions”, (neurocomputing), Kybernetik, Vol. 5, pp. 113-119, 1969.

125 Lashley, K., “In search of the engram”, (NC, F.R., #5) Society for experimental biology (Great Britain): Physiological mechanisms in animal behavior”, Academic Press, New York, NY, 1950.

126 Gabor, D., “A new microscopic principle”, Nature, Vol. 161, pp. 777-778, 1948.

127 Primbram, K.H., Nuwer, M., and Baron, R.J., "The Holographic Hypothesis of Memory Structure in Brain Function an Perception", (Neurocomputing), Contemporary Developments in Mathematical Psychology, Volume II, Krantz, D.H., Atkinson, R.C., Luce, R.D., and Suppes, P. (Eds.), W.H. Freeman, San Francisco, CA, pp. 416-457, 1974.

128 Changeux, J.P., Heidmann, T., and Patte, P., "Learning by Selection", (Neurocomputing), The Biology of Learning, Marler, P., and Terrace, H.S. (Eds.), Springer-Verlag, Berlin, pp. 115-133, 1984.

129 Kohonen, T., “The 'Neural' Phonetic Typewriter", (Neurocomputing), Computer, Vol. 21 pp. 11-22, March, 1988.

130 Merzenich, M.M., Edelman, G.M., Sur, M., and Kaas, J.H., “Origins of Topographical Order in Cortical Fields: Some observations and Hypotheses on the Selection Dynamics of Functional Maps”, J. Neurosci., 1984.

131 Mountcastel, V.B., “An Organizing principle for cerebral function: the Unit Module and the Distributed System”, in *The Mindful Brain: Cortical Organization and the Group-Selective Theory of Higher Brain Function*, Elman, G.M., and Mountcastel, V.B. (Eds.), MIT Press, Cambridge, MA, pp. 7-50, 1978.

132 von Neumann, J., "The Theory of Self-Reproducing Automata", A.W. Burks, Ed., University of Illinois Press, Urbana, IL, 1966.

133 Codd, E.F., "Cellular Automata", Academic Press, New York, NY, 1968.

134 Langton, C.G., "Self-Reproduction in Cellular Automata", Physica 10D, pp. 135-144, 1984.

135 Byl, J., "Self-Reproduction in Small Cellular Automata", Physica 34D, pp. 295-299, 1989.

136 Toffoli, T., “CAM: A high-performance Cellular-automaton machine”, Physica, Vol. 10D, pp. 195-204, 1984.

137 Tucker, J., “Cellular automata machine: the Ultimate parallel computer”, High Technology, Vol. 4, No. 6, pp. 85-87, 1984.

138 Toffoli, T., and Margolus, N., “Cellular Automata Machines”, MIT Press, Cambridge, MA, 1985.

139 Hofstadter, D.R., “Gödel Escher Bach : an eternal golden braid”, Basic Book, New York, NY, 1979.

140 Morris, H.C., "Typogenetics: A Logic of Artificial Life", Artificial Life, Langton, C.G. (Eds.), Addison-Wesley Publishing Co., Redwood City, 1989.

141 Marchal, P., Piguet, C., Mange, D., Stauffer, A., and Durand, S., "Achieving von Neumann's Dream: Artificial Life on Silicon" 1994 IEEE International Conference on Neural Networks, Orlando, Fla., pp. 2321-2326, June 27-29, 1994.

7. VITA

Robert Schinazi is a native of Providence, Rhode Island. In addition to this Ph.D. from Virginia Tech in Industrial Engineering, he holds an MS in Ceramics and Materials Engineering from Clemson University and a BS in Mechanical Engineering from Rensselaer Polytechnic Institute. As of May 1998 Dr. Schinazi is president of Rhode Flux, Inc. and a design consultant for the Bose Corporation. His future plans focus on growing his company and pursuing other ventures as their enabling conditions occur.