UNIVERSITY OF CALIFORNIA

Los Angeles

Neural Network Control of A Chlorine Contact Basin

A thesis submitted in partial satisfaction of the requirements

for the degree Master of Science

in Civil Engineering

by

Michelle Hyeseung Park

1996

The thesis of Michelle Hyeseung Park is approved.

Menachem Elimelech

David Okrent

Michael K. Stenstrom, Committee Chair

University of California, Los Angeles,

1996

Table of Contents

List of Figures

List of Tables

Acknowledgements

I wish to express my deep sense of gratitude to my graduate advisor, Dr. Michael K. Stenstrom for offering me great cooperation, guidance, and encouragment. Also, I wish to acknowledge the assistance and advice of the other members of my thesis committe.

I am thankful to my husband and family for their support and encouragement which have made this study possibe.

Abstract of the Thesis

Neural Network Control of A Chlorine Contact Basin

by

Michelle Hyeseung Park

Master of Science in Civil Engineering

University of California, Los Angeles, 1996

Professor Michael K. Stenstrom, Chair

Studies on the brain function have been conducted for many years. One of the useful results is the introduction of neural networks. Neural networks have been used to solve non-linear problems and provide a simple way to solve complicated problems without using the modeling process.

In this thesis, the application of a neural network technique to solve the problem of chlorination process control has been examined. Chlorination has been the method of choice for disinfection of water and wastewater treatment. The reaction between the dissolved chlorine and the nitrogen compounds has been modeled using very complicated non-linear equations. Moreover, the conventional feedback process control method is not well suited to chlorination process control because of the relatively long retention time and the transportation delay of the process. However, using feed-forward, back-propagation neural network technology, one can achieve relatively stable operation without knowing the precise model for the chlorination process. It is also shown that a neural network can tolerate noise in the input data and still achieve satisfactory performance.

1. Introduction

For the disinfection of wastewater, chlorination has been the method of choice since the early part of this century. Its simplicity and inexpensive operation are the advantages which make it so popular. However, there are people who are very critical about the use of chlorine. They criticize the effectiveness of chlorination for disinfection and the harmful byproducts that are produced. In fact, this criticism is leading to the development of alternative method for disinfection, and better chlorination process control mechanisms.

The chlorination process is a complex non-linear system using reactors with long detention times. Usually a plug flow reactor is used. These facts make it hard to use the conventional feedback control mechanism for chlorination process control. Even though a process model has been developed [Stenstrom 1976, 1980], it is not clear how to build a feed-forward control system using conventional techniques. To overcome these concerns, an artificial neural network was used.

In recent years, many research groups have begun to consider artificial neural network technology for solving various problems which can be characterized as having massive amounts of data, non-linear behavior, and little or no understanding of the deterministic

processes. Advances in computer technology provide tools to accurately measure, transfer, and store vast amounts of data. Consequently, there is a large database to describe empirically many processes for which there is no fundamental understanding. An artificial neural network provides a powerful tool to analyze data and build a model to duplicate the process' behavior.

The non-linear nature of the chlorination process is well suited to demonstrate a neural network's problem solving capability. Since the chlorination process has been used for many years, there is a large amount of available data. This combination lead us to explore the possibility of using a neural network for chlorination process control.

The goal of this thesis is to build a feed-forward neural network controller. The controller must define the proper value for hypochlorous acid concentration at the beginning of chlorination process to keep the concentration of hypochlorous acid at a suitable minimum value at the end of the process. Effluent concentration stability must be maintained during periods of variable flow rate and contaminant concentration. Maintaining this effluent concentration will optimize process efficiency, and minimize chlorine usage and byproduct formation. As the result, a neural network controlled chlorination system can be more economical by reducing the amount of chlorine usage while maintaining proper disinfection efficiency. The complexity of the system makes it a good candidate for an artificial neural network controller.

This thesis is organized as follows. Chapter 2 explains the model of the chlorination process, mainly proposed by Stenstrom [1976, 1980]. Chapter 3 provides a basic description about the neural network technology which is focused on the back-propagation network. The construction of the neural network for chlorination process control is described in Chapter 4. In Chapter 5, the experiments are explained. Chapter 6 provides discussion about the results generated by the experiment. Chapter 7 summarizes the conclusion of the thesis. The last chapter describes the limitations of this study and improvements for future research.

2. Chlorination

Chlorination is currently the most popular method for disinfection in water and wastewater treatment. It serves as an important process in reducing various waterborne diseases.

In this chapter I develop a simple model of the chlorination process for the use of the development of chlorination process control system.

2.1 Chemistry of Chlorination

No model for chlorination would be complete without consideration of its chemistry. When gaseous chlorine is dissolved in water, the following hydrolysis reaction occurs:

$$Cl_2 + H_2O \leftrightarrow HOCl + H^+ + Cl^-, \tag{2.1.1}$$

or

$$Cl_2 + OH^- \leftrightarrow HOCl + Cl^- \tag{2.1.2}$$

Consideration here is restricted to chlorination using gaseous chlorine. Furthermore, changes in pH due to addition of chlorine are not considered. Hypochlorous acid dissociates according to the following equation:

$$HOCl \leftrightarrow OCl^- + H^+ \qquad (2.1.3)$$

Hypochlorous acid is weakly acidic and its dissociation constant at 20°C is 2.611×10^{-8} moles/liter. Therefore the pH level of water influences the distribution between HOCl and OCl^-. As the pH increases from 6 to 9, the relative fraction of HOCl decreases while the corresponding fraction of OCl^- increases [Montgomery 1985]. The time required to complete hydrolysis (equation 2.1.1) is in the order of only a few tenths of a second, while the time to complete the dissociation reaction (equation 2.1.3) is in the order of nano seconds [Stenstrom 1976].

The dissociation constant is defined as follows:

$$K_i = \frac{(H^+)(OCl^-)}{(HOCl)} \qquad (2.1.4)$$

The dissociation constant, K_i, is temperature dependent. The values of this constant have been computed using a best-fit technique [Morris 1996] from the acid dissociation constant, pKa, as follows:

$$pKa = \frac{3000.00}{T} - 10.0686 + 0.0253T \qquad (2.1.5)$$

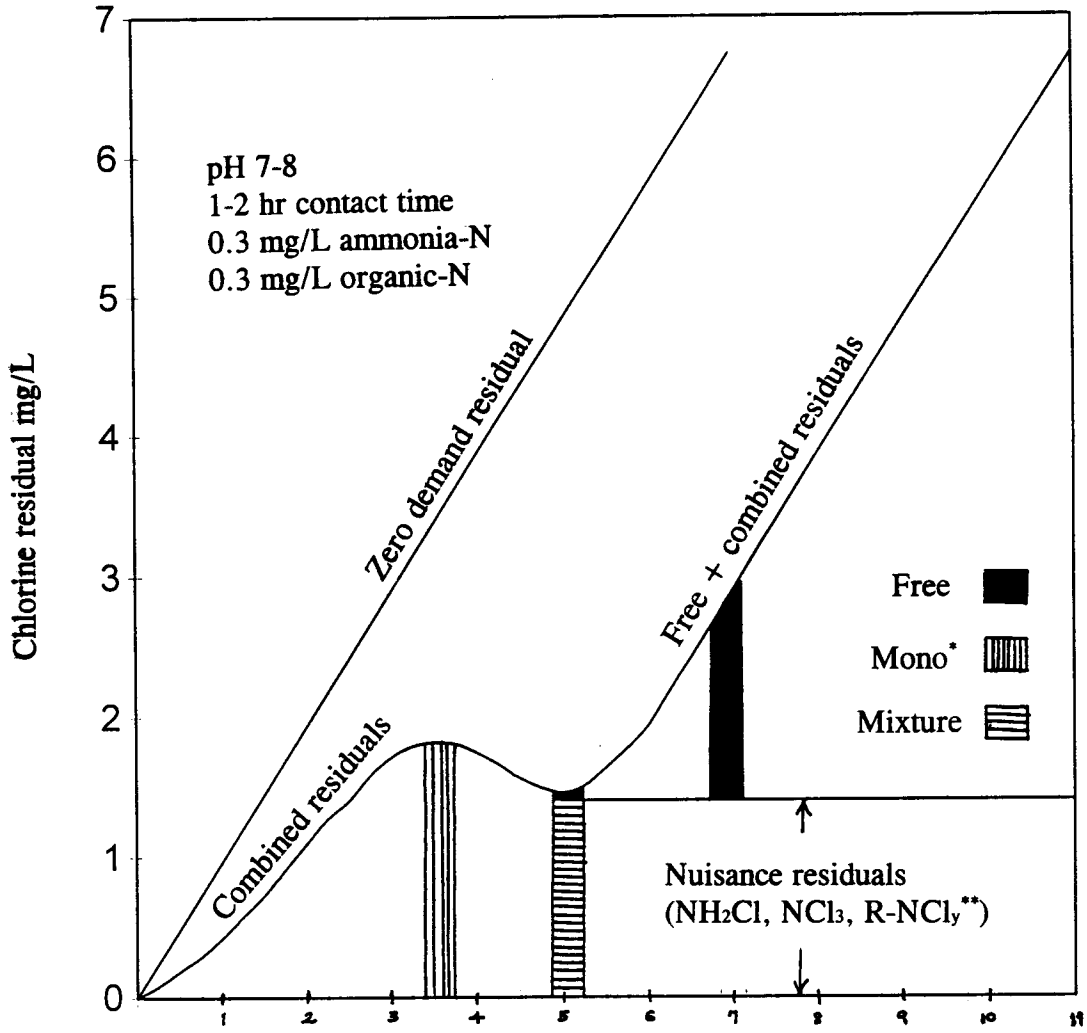
where

T is temperature in °K

It is generally known that hypochlorous acid is a more effective disinfectant than hypochlorite ions, especially at short contact times. However, the bacterial inactivation by these chlorine is similar equivalent to long contact times [Montgomery 1985].

2.2 Reaction with Inorganic Nitrogen

Chlorine (HOCl) reacts with inorganic nitrogen and most waters and wastewaters have measurable concentrations of inorganic nitrogen. Chlorine and ammonia nitrogen react to form several chloroamines and other nitrogeneous compounds such as nitrate. The final products [Stenstrom 1976, 1980] are a function of pH, temperature, the velocity of the flow, and initial chlorine concentration relative to initial ammonia nitrogen concentration. The formation of the three fundamental chloramines containing inorganic and organic nitrogen has been summarized by White [1992] as shown in Figure 2.1. The empirical reactions for chloramine formation in dilute aqueous solutions (1-50 mg/liter) are described as follows [White 1992]:



* Predominantly monochloramine; R-NCl_y titrates as dichloramine

** R-NCl_y = organochloramines

Figure 2.1 Relationship between Ammonia Nitrogen, Organic Nitrogen and Chlorine [White 1992]

$$HOCl + NH_3(aq) \rightarrow NH_2Cl \text{ (monochloramine)} + H_2O \quad (2.2.1)$$

$$HOCL + NH_2Cl \rightarrow NHCl_2 \text{ (dichloramine)} + H_2O \quad (2.2.2)$$

$$HOCl + NHCl_2 \rightarrow NCl_3 \text{ (trichloramine)} + H_2 \quad (2.2.3)$$

The kinetics and equilibrium concentration of these reactions are a function of temperature and pH. The greatest rate of formation of monochloramine(NH_2Cl) is at pH=8.3. The strong relationship between the reaction rate and pH can be modeled by the following equations:

$$r_1 = K_1(HOCl)(NH_3) \quad (2.2.4)$$

or

$$r_1 = K_1(OCl^-)(NH_4^+) \quad (2.2.5)$$

Both equations represent the experimental results equally well; however, the first one is preferable due to the greater electrophilicity of undissociated hypochlorous acid [Morris 1965]. The rate constant is given as:

$$K_1 = 9.7 \times 10^8 e^{(-3000/R \bullet TK)} \quad (2.2.6)$$

where R is gas constant and K is the temperature in °K.

The time required for 99 percent completion of free chlorine to monochloramine at 25°C with molar ratio of $0.2x10^{-3}$ moles/liter HOCl and $1.0x10^{-3}$ moles/liter NH_3 are summarized in Table 2.1.

The rate of formation of monochloramine decreases as the temperature decreases. It requires almost five minutes for 90 percent completion at pH 7 at 0°C.

pH	Seconds
2	421
4	147
7	0.2
8.3	0.069
12	33.2

Table 2.1 Time Required for 99% Completion of Monochlorimine Formation Reaction [White 1992].

Based upon the $HOCl^-$ - OCl^- equilibrium, the pH dependence of this reaction is represented precisely [White 1992].

The reaction of dichloramine($NHCl_2$) formation is second-order and can be described as follows:

$$r_2 = K_2(HOCl)(NH_2Cl) \tag{2.2.7}$$

where, r_2 is the rate of dichloramine formation and K_2 is a rate constant.

The reaction is catalyzed by the hydrogen ion and acetic acid. The reaction can be described by the following equation [Morris 1965]:

$$K_2 = K_{UCl}[1.+(H^+)+(HAC)] \tag{2.2.8}$$

where

K_{UCL} = uncatalyzed rate constant,

H^+ = hydrogen ion concentration,

HAC = acetic acid concentration,

$K_{UCL} = 7.6x10^{(-7300/RT)}$

Compared to the formation of monochloramine, the rate of this formation is slower. It takes approximately one hour for 90 percent completion and up to 5 hours at pH 8.5 and above when ammonia nitrogen concentrations are very low. This reaction depends upon pH, initial ammonia nitrogen, and temperature. The reaction accelerates measurably as the pH approaches 5. The time-to-completion of this reaction is known to be minutes when the initial nitrogen concentration exceeds 1 mg/liter [Morris 1965].

The kinetics of nitrogen trichloride(NCl_3) formation are not known well, especially in concentrations less than 10 mg/L. If the pH is reduced to 5 or less, nitrogen trichloride forms even when the mole ratio of chlorine to ammonia nitrogen is one to one (1/1). At one time it was thought that nitrogen trichloride would not form above pH 5. It is now known to occur in water treatment plants when the pH is as high as 9. This happens at very high chlorine to ammonia weight ratios, 25 to 1 [White 1992]. The formation of nitrogen trichloride by a second-order reversible reaction was proposed as follows [Morris 1969]:

$$HOCl + NHCl_2 \leftrightarrow NCl_3 + H_2 0 \qquad (2.2.9)$$

Only small quantities of nitrogen-trichloride exist after breakpoint chlorination at neutral pH's. At higher pH's, almost no nitrogen-trichloride can be found; however small amounts of nitrate may be produced.

The nitrogen-trichloride reaction seems to be in competition with the desired breakpoint reactions which produce nitrogen gas. One proposed reaction for the nitrogen gas formation is as follows [Rossum 1943]:

$$NH_2Cl + NHCl_2 \rightarrow N_2 + 3HCl \qquad (2.2.10)$$

The decomposition of dichloramine to form an intermediate product, the nitroxyl group, NOH is as follows [Chapin 1931]:

$$NHCl_2 + H_2O \rightarrow NOH + 2H^+ + 2Cl^- \quad (2.2.11)$$

This is followed by three competing reactions involving nitroxyl group, NOH, and the other chlorine species. These equations for the reactions are as follows [Morris 1969]:

$$NOH + NH_2Cl \rightarrow N_2 + H_2O + Cl^- + H^+ \quad (2.2.12)$$

$$NOH + NHCl_2 \rightarrow N_2 + HOCl + H^+ + Cl^- \quad (2.2.13)$$

$$NOH + 2HOCl \rightarrow NO_3^- + 3H^+ + 2Cl^- \quad (2.2.14)$$

2.3 Reactions with Organic Compounds

Chlorine reacts with organic nitrogen as well as inorganic nitrogen; consequently, these reactions must be considered. Griffin [1940] noted that the effect of organic nitrogen compounds on the normal breakpoint curve as early as 1940. He referred to water containing large quantities of organic nitrogen as "offset waters" since they did not

produce the normal breakpoint curve. Griffin and Chamberlin [1945] later noted that the effects of super chlorination of wastewater in various stages of treatment. They chlorinated raw influent, primary effluent, and final effluent from an activated sludge plant. They also noted that the chlorine demand was greatest for raw influent, and that the typical breakpoint curve was not necessarily observed in wastewater chlorination. It was also observed that ammonia nitrogen reappeared after the breakpoint. The effects of chlorination on albumonoid nitrogen were minimal. They observed reaction rates that were first-order with respect to demand and somewhere between first and second-order with respect to chlorine [Bryant 1972].

Feben and Taras [1950] investigated the reactions between chlorine and nitrogen compounds. They were able to characterize the chlorine-organic nitrogen reaction by the following equation:

$$D = kt^n \tag{2.3.1}$$

where,

D is chlorine demand at any time (mg/liter),

k is chlorine demand after one hour (mg/liter),

t is time (hours), and

n is experimentally measured coefficient.

The equation was linearized by plotting on log-log paper. The value of n and k for a number of specific organic compounds as well as raw water had been measured. From this measured data, it is observed that the reaction with ammonia is complete within one hour. Other compounds show one hour specific demands which vary from 0.15 mg/liter (hippuric acid) to 7.0 mg/liter (phenol). These one hour demand values are for initial concentrations of 1 mg/liter of the specific compound and different chlorine concentrations. The exponent n varies from 0.0 (ammonia) to 0.30 (gelatin). The chlorine reactions with nitrogenous compounds can be divided into three categories with respect to speed of reaction: the ammonia reactions were most rapid, being complete in less than one hour; amino acid reactions were slower than ammonia reactions, "reacting slowly over an extended period of time", and reactions with proteins required even greater periods of time to reach completion. In some cases, the protein chlorine reactions were not be complete after 72 hours [Taras 1953].

2.4 Reactions with Other Inorganic Compounds

Chlorine rapidly reacts with many reduced inorganic compounds such as sulfides, which explains chlorine's early use for odor control. Taras [1950] investigated the chlorine demand of sulfide, nitrite, and ferrous sulfate. He did not report the kinetic data necessary to determine a reaction rate, but he stated one hour chlorine demands which indicated that the reactions between chlorine and each of the previously

mentioned reduced species was complete in less than one hour. Later, Kokoropoulos and Manos [1973] measured the rates of reaction of chlorine with nitrite, manganese (II), iron (II), and detergents. They found out that the reactions with these species were two to five orders of magnitude slower than the ammonia or disinfection reactions. They also noticed that the amounts of ammonia, other nitrogen compounds, and impurities in wastewater determined the chlorine dosage necessary to achieve adequate disinfection.

2.5 Assumptions on Chlorination Model Development

Using the previously described reaction rates for chlorine with ammonia, it is possible to mathematically model the disinfection process. The reactions involved are both serial and competitive. The following assumptions are made [Stenstrom 1976, 1980]:

1. Since the hydrolysis of chlorine gas is essentially complete in a few tenths of a second, it is considered to be instantaneous.
2. The dissociation of hypochlorous acid is complete in approximately 10^{-9} seconds; therefore, it is also considered to be instantaneous.
3. The second-order reactions between ammonia and hypochlorous acid to form the various chloramine are used. The concentrations of the unionized hypochlorous acid and ammonia are calculated from the equilibrium relationships. The

equilibrium reaction between ammonia and ammonium is considered to be instantaneous. It is assumed that only the hydrogen ion concentration has significant catalytic effects.

4. Nitrite, iron (II), and manganese (II) are assumed to have no significant effects on the reactions with chlorine.
5. The reactions for the formation of nitrogen gas and nitrogen-trichloride are used.
6. The only chlorine species which exerts disinfecting power are hypochlorous acid (HOCl), the hypochlorite ion (OCl^-), and monochloramine. Any concentrations of dichloramine and nitrogen-trichloride which exists are treated as if they have the same disinfecting power as monochloramine.
7. The model is restricted to cases where well-treated effluent is chlorinated. Therefore, the concentrations of hydrogen sulfide and highly reduced compounds are assumed to be zero.

The previously described model is capable of describing all the breakpoint chlorine reactions. At wastewater treatment plants, breakpoint chlorination is usually not practiced; only enough chlorine is used to produce monochloramine. Monochloramine is usually sufficient to provide the needed disinfection.

The next section describes a simplified model which can describe monochlorination. The simplified model can also be used in a control system.

2.6 Simplified Model Description

This section describes plug flow reactors and a second-order reaction model to simulate monochlorination. The final result is a hyperbolic one dimensional partial differential equation with a single reaction term.

Continuous-flow reactors, such as the reactor basins or aeration tanks used for the activated sludge process, may be classified according to their flow regime as plug-flow, dispersed plug-flow, and completely mixed reactors. In a plug-flow reactor, the elements of the fluid that enter the reactor at the same time flow through it with the same velocity and leave at the same time. The flow through the reactor is similar to a piston or plug moving through it. The travel time of the fluid elements is equal to the theoretical detention time, and there is no longitudinal mixing [Reynold 1982].

The material balance of a plug-flow reactor is performed as follows:

$$Input - Output + Reaction = Accumulation$$

$$Q * C_{ix} - Q * C_{i(x+\Delta x)} + \Delta V * r = \Delta V \frac{\partial C_i}{\partial t} \tag{2.6.1}$$

divide by ΔxA

$$\frac{Q}{A} * \frac{C_{ix}}{\Delta x} - \frac{Q}{A} * \frac{C_{i(x+\Delta x)}}{\Delta x} + r = \frac{\partial C_i}{\partial t} \tag{2.6.2}$$

$$\because \frac{Q}{A} = \overline{V_x}$$

$$\therefore \overline{V_x} * \left\{ \frac{\left(C_{ix} - C_{i(x+\Delta x)} \right)}{\Delta x} \right\} + r = \frac{\partial C_i}{\partial t} \tag{2.6.3}$$

By taking the limit $\Delta x \rightarrow 0$,

$$-\overline{V_x} \frac{\partial C_i}{\partial x} + r = \frac{\partial C_i}{\partial t} \tag{2.6.4}$$

where

x is distance,

t is time,

A is the cross-sectional area of the flow,

V is the volume of the flow,

r is an arbitrary reaction,

Q is the flow rate,

C_i is the reactant concentration, and

V_x is the average velocity.

The simplified model considers only a reaction with free chlorine, represented as HOCl and oxygen demanding materials, represented as S. The reaction is assumed to be second-order as follows:

$$r = -K * S * HOCl \tag{2.6.5}$$

K is the rate coefficient.

After the substitution of C_i =HOCl, the entire simplified model becomes:

$$-\bar{V}_x \frac{\partial HOCl}{\partial x} - K * S * HOCl = \frac{\partial HOCl}{\partial t} \tag{2.6.6}$$

This equation is solved analytically in Chapter 4 for specific conditions and the solution is used with the neural network to simulate chlorination process control.

3. Basic Description of Neural Networks

A neural network is an information processing system that simulates the way a human brain works. It is a computer-based simulation of a human nerve system, which works differently than a conventional computer.

Even though studies of human brain functions began late in the 19th century, it is not understood well enough to construct a system that performs in the same way a human brain works. There are many researchers who are very actively investigating this subject, and they are attempting to build a system which can be called an "artificial brain". The limited results of the studies provide some suggestions to utilize the brain theory to practical applications. The results suggest that the conventional computing paradigm is far different from the behavior of the brain. The brain function can be better described as connectionist model of computing, apart from the conventional von-Neuman model, which proceeds sequentially through a set of instructions. In a connectionist model of computing, a large number of simple processing elements are heavily connected. Information is passed among processing elements, and each processing element collects and processes the information with its simple, non-linear function.

3.1 Brain Structure

Neurons are the processing elements in human brain. A typical neuron consists of three parts; cell body, dendrite, and axon. A cell body contains a nucleus in its middle and extends to dendrites and axons. Dendrites receive the impulses of human nerves from neighboring neurons; the cell body collects and processes them, and the axon sends the resulting signals to other neurons. The point where the signal is exchanged between neurons is called a synapse. According to the results of the studies of brain activities, the synapses have the ability to modify the signals as they pass them. The amount of change in the synapses as the signals pass through is determined by the synaptic strength on the connection. The changes of the synaptic strength are caused by fatigue, oxygen deficiency, and agents such as anesthetics. The brain adapts to external changes by changing the synaptic strengths in the response to the external changes. After signals reach the synapses, they are weighted by the synaptic strengths, and all weighted signals are collected and summed together at the receiving neuron. Finally the cell body processes it using its simple function to generate the corresponding output. A threshold value can be used to determine whether the sum produces output [Nelson 1991].

A human brain consists of ten billion neurons and each neuron can interact with as few as one thousand or as many as two hundred thousand other neurons through synaptic

connections. These huge number of neurons and interconnections make the brain adapt to the external signals to produce reasonable reactions. The real wonder of the brain comes from the fact that there is no central controller to coordinate the operation. Even though so many neurons are interacting with each other through these massive connections, without any form of the central controller, the brain reacts to the external changes properly and promptly, and even to the signals which it has never experienced. The correct reactions are due to the genetic programming capability and the learning ability of brain to respond to the events. This fact is the main focus of the development of an artificial neural network, or simply a neural network [Zupan 1993].

3.2 Artificial Neural Network

Building a neural network is not like conventional programming, nor it is building an artificial brain. It requires constructing a system which utilizes the understanding of brain function to solve a problem. There have been many efforts to build a machine that simulates the function of a human brain. Even though this research is still far from successful in building an artificial brain, there has been some degree of success in understanding the learning mechanism of the brain. In particular, researchers have some understanding of how a brain collects signals from the outside world, processes

them, and responds to them. This progress has resulted in the construction of artificial neural networks.

The structure of a network is conceptually very similar to that of a human brain. It consists of a number of simple, highly interconnected processing elements which are the analog of the biological neurons connected through synapses. The synaptic strengths are implemented as the weights on the connections and the threshold values on neurons are implemented as the biases on the processing elements. Electrical signals are passed through the interconnected network of processing elements. To solve non-linearly separable problems[1], one needs to have several layers of the interconnecting structure. Even though artificial neural networks are very similar in structure to brain, they are far from being artificial brains in terms of capability and adaptability. This is because researchers do not have enough understanding of brain functions and there is no means of constructing the enormous size of the required neural network. Therefore current technology allows them to build a system that can only solve problems in limited domains with limited capabilities. The capability of adapting the external changes makes the resulting artificial neural network useful for certain problems [Nelson 1991].

[1] A non-linearly separable problem is a kind of problem in which the classes of solutions can not be separated using their hyper-plane. For example, exclusive-OR is not linearly separable because two solution classes, 0 and 1, are not separable using a line.

In a manner similar to the way synaptic strength has the ability to adapt the signals from certain external activities and/or situations of human body, the connections in a neural network have assigned weights which determine the effect of signals to the neurons. Besides having weights on the connections, there is an additional parameter, called bias, which affects the adaptability of a neural network to solve a decision problem. Mathematically, the input signal to a neuron is the dot product of the vector of input signals and the weight vector on the connections into the neuron. The result is compared to the threshold value of the neuron to determine the output. If the product is greater than the threshold, the neuron generates a signal according to the input values. If the value is less than the threshold, no signal is generated. Both types of responses are significant for determinating the corresponding result of the input signals [Zupan 1993].

The exact output signal from a neuron is determined by the transfer function of the neuron. The transfer function can be something as simple as a binary function. In this case the output depends upon whether the product is positive or negative. When the product is positive, the output is "1". If the product is negative, then the output is "0". This type of transfer function is called a hard limiter or step function. Another type of transfer function, ramp function or threshold function, mirrors the product only within a given range. As the product value moves outside of the range, the function value stays at certain levels (maximum and minimum values). It can be viewed as a linear function

clipped to minimum and maximum values, which transforms it into a non-linear function. Yet another option would be a sigmoid or 'S' shaped curve. The curve asymptotically approaches a minimum and maximum. Mathematically, the exciting features of these curves are that both the function and its derivatives are continuous. This type of transfer function works fairly well and is often the transfer function of choice. Any of these functions would introduce non-linearity into the neural network. This non-linearity combined with the interconnections of neurons accomplishes proper mapping from input signals to the corresponding output activities [Zupan 1993].

One aspect of building a neural network is determining how one can interconnect several layers of neurons. The input layer receives signals from outside world. This layer typically performs no other function than buffering the input signals. The outputs of a neural network are generated from the neurons of the output layer. Between the input layer and output layer there can be several layers of neurons. These layers are called hidden layers, signifying they are hidden from the outside world. It is the presence of these hidden layers that enables neural networks to be able to solve non-linearly separable problems, which represent many real world problems. The connectivity between layers determines how the outputs of neurons in one layer are sent to the other neurons. The output signal from a neuron in a layer may be passed to a neuron or neurons in other layers as inputs, or possibly sent back as an input to its own layer, or even to itself. Researchers classify neural networks into two groups; feed-

forward and feedback networks. For a feed-forward network, the signals can only flow in the forward direction from the neurons of the input layer to the neurons of the output layer. In contrast to feed-forward networks, a feedback network can direct outputs from one layer backward or even back into the layer itself. In this sense, a feedback network can also be called a recurrent network [Zupan 1993].

3.3 Neural Network Learning

The main advantage of neural networks over the conventional computing paradigm is their adaptability to the problem space. The adaptability is the ability of a system to self-adjust itself according to the changes given to the system. In the conventional programming paradigm, the behavior of a system is determined at the time system is built. In contrast, neural networks can reconfigure themselves to adapt to the signals given to them, so that they can make reasonable response to the signals. The adaptability of a system can be divided into four categories: learning, self-organizing, generalization, and training [Nelson 1991].

Learning can be defined at the level of a single processing element. Learning occurs when the weights on the connections are adjusted to reduce the errors of the outputs generated. Choosing specific algorithms for particular problems would make the solutions so specific that their use would be severely limited. In contrast, a neural

network can generate its own algorithm by adjusting the weights on the connections to solve many problems, even for the problems never included in the network's design.

Self-organizing is the capability of configuring many processing elements at once. During the training sessions, the strength on connections is modified using the rules for learning throughout the entire network system. It functions as if the neural network is developing its own heuristics to solve the problems as it goes through the iterations. It has been shown theoretically that the network will converge to a stable response, even though it sometimes takes too long to be practical.

Generalization is the ability to make hypothetical response to an input that has not seen before. Generalization takes place when a set of values are presented to the system during training. The network compares the values and composes some characteristics which can distinguish one from another. As the result, the system can hypothesize its response to unseen inputs and make a reasonable response.

Training is the way a neural network learns. The method of training can be categorized into two groups: supervised or unsupervised. In supervised training, the neural network uses a set of input vectors and corresponding or desirable output vectors to learn a problem. During training, the network reduces the errors on network generated results by comparing them to the desirable output values and adjusting the weights on the

connections. Unsupervised training does not require the knowledge of the desirable outputs; it only requires a set of input vectors. The network can group the input vectors according to the training rules to find the solution to the problem.

In the case of a feed-forward neural network with supervised training, the network determines the relationship between input and output by looking at examples of many input-output pairs. The network processes the inputs presented to it and adjusts the weights on the connections to produce outputs that correspond to the known inputs. This process can be viewed as a self-organization because the network structure is constantly changing as the training takes place. By knowing what output is expected from each input value, the network learns by adjusting or adapting the strengths of the connections between processing elements to reduce the error. The method used for adjusting weights on the connections is called a learning rule.

There are many suggested and commonly used learning rules. Many researchers are still trying to find new learning rules and make existing learning rules more efficient in order to reduce the training time and increase the accuracy of the resulting solutions. However, researchers do not know much about how learning occurs, and experimental evidence is hard to obtain. The actual learning processes by which a human adapts to the environment are very complicated, and there is no way of achieving the same results by the simplified learning rules proposed by current research. However, there

are some learning rules which are proven to be efficient in solving certain types of problems.

Among many learning rules, the delta rule is one of the most commonly used learning rules. It is based on the simple idea of continuously modifying the strengths of the connections to reduce the difference (delta) between the desired value and the current output value of each processing element. This rule is also known as Widrow-Hoff learning rule or Least Mean Square (LMS) learning rule [Nelson 1991].

The gradient descent rule is an implementation of delta rule. The weights are modified by amounts proportional to the first derivatives of the errors between actual outputs and their desired values. This rule is commonly used, but it is very slow to converge to produce minimum error. To make the learning process more efficient, many rules and suggestions have been made. Each rule has strengths and weaknesses for specific problems. Therefore, it is very important to choose the right learning rule for the proposed application of the neural network [Nelson 1991].

3.4 Back-propagation Learning Algorithm

The most popular learning algorithm is back propagation. It is a generalization of the delta rule and supervised training. It can be described in two phases: forward and

backward. During the forward phase, the training examples are presented to the network and they propagate in the forward direction through layers of neurons to compute output values. At the end of the forward phase, the network compares the outputs to the corresponding, desired outputs in the training set and generates the error values. In the backward phase, these errors values are propagated in the backward direction through the layers of neurons. The network adjusts the weights on the incoming connections and the biases on each of the neurons to reduce the errors presented to the neuron. These forward and backward operations are repeated until a predetermined error goal is reached or the number of iterations of applying the training sets, called epochs, exceeds a predetermined maximum. The later case is called a network training failure, denoting that the neural network failed to find the appropriate relation between input and output values in a specified number of attempts. It has been mathematically proven that this simple approach can find the relations between a set of inputs and outputs when there exists such relation [Zupan 1993].

Even though it is possible to solve any problem with the back-propagation learning algorithm, it is often impracticable to train the neural network because of computational complexity [Freedman 1995]. In other words, too much computer time is required to train the neural network to solve the problem. This is a fundamental difficulty that all neural network developers face, and an enormous amount of research has been devoted to this problem. The problem occurs because of two main reasons: one is that the

problem space is too large to handle with a limited number of observations in the training set; the other problem is local-minima. Since many real world problems have continuous variables, the problem spaces normally have infinitely many possibilities. Only a limited number of states can be presented to the neural network for training. Often the presented values may not represent a good sample of the problem space, so that the neural network can not find the desired solution from the training set. It is necessary to select the training set to include instances of the problem which well describe the entire problem domain.

The local minima problem can be described using the concept of an error surface. An error surface is a multidimensional surface which represents the sum of squared errors (the sum of the squared differences between the desired outputs and the actual outputs). The height of a point on the surface represents the amount of error across the instances in the training set with given values of weights and biases. Neural network training can be described as finding the lowest point of the surface. Weights and biases are adjusted to reach the lowest point of the error surface, which is called the global minimum. There can also be a number of local minima on the error surface. Under certain circumstances, the search for the global minimum terminates at one of these local minima. As a result, the solutions generated by the neural network are not optimal solutions [Caudill 1994].

Even though the basic learning rule is a back-propagation learning using gradient descent, there are many variations to choose from. These variations are attempts to reduce computational complexity and avoid local minima. Each has its own strength and weakness. Among many learning algorithms, Levenburg-Marquardt learning algorithm was chosen because it can converge faster than other learning algorithms. Since it stores all the intermediate results in the memory, it may suffer from a lack of memory which can slow down the training process dramatically. However, the neural network under construction does not show this degradation of the training [Freedman 1995].

To perform training using Levenburg-Marquardt, one needs to determine some parameters to perform the training of a neural network. The parameters are error goal, learning rate, and momentum. These parameters can affect the training and the performance of the network after training [Demuth 1994].

The error goal is the error between the desired outputs and the network generated outputs for the training set. The sum of squared errors is often chosen for the error measure. Training stops when network performance exceeds the arbitrarily set error criteria. The determination of the actual value for the error goal is somewhat arbitrary. If the criteria is too strict, over-training may result. Alternatively, poor performance may result from using loose criteria. The notion of over-training is introduced because

training with too strict an error goal can result in a network that works very well for the cases in the training set, but works poorly for the cases not included in the training set. This phenomenon results because the network fits the training values too tightly, and cannot tolerate variation in the input values.

The learning rate determines how fast the network adjusts itself in the response to errors. The value of learning rate should be set to the maximum value which does not cause oscillation. When a learning rate is too high, the adjusted weights on each instance may oscillate between points with similar heights on the error surface. If the learning rate is too small, training will be slow and the process is more susceptible to "local minima". Therefore the choice of the learning rate should be balanced to provide reasonable training time and overall error reduction [Freedman 1995].

Momentum is introduced as a means to reduce the chance of oscillation for a given learning rate. Momentum is a modification of the generalized delta rule in which the equivalent of physical momentum added to the system to keep the network on its downward path on the error surface. It allows the use of a higher learning rate while avoiding oscillation. Consequently, one can reduce the network training time by introducing momentum [Caudill 1994].

The next chapter applies these criteria to build a neural network to control the chlorination process.

4. Neural Network Chlorination Process Control System

4.1 Process Control System

A typical process control system consists of a plant or process which can be composed of smaller sub-plants or sub-processes. The process has numerous inputs, but these inputs may be divided into two main categories: disturbances and manipulated variables. Manipulated variables are the inputs that can be changed by the plant operator or controller, while disturbances are the inputs which can not be changed by the operator or controller. The object of the control system is to minimize the effects of these disturbances. The controlled variables are the outputs of the process which the controller or the operator tries to keep at some predetermined level, called the set point. There are usually some intermediate variables which may or may not be outputs of the process, but they are usually quantities which can be calculated or determined from a combination of the inputs, outputs, or the specific values of parameters internal to the process. Furthermore, these intermediate variables may become controlled variables in different control schemes [Stenstrom 1976].

Process control systems can be divided into a number of categories, but one of the more common methods of classifying process control systems is using the direction of data flow: feed-forward and feedback. Feedback type controllers are frequently used. They normally have the advantages of simplicity and low cost, but they have a distinct disadvantage in that a disturbance in the controlled variable must occur before any control action is initiated. Alternatively feed-forward type controllers eliminate this disadvantage, but they require sensing or predicting the disturbances in the process inputs. For certain situations, this type of controller is theoretically capable of achieving perfect control. However, this can not be achieved in practice because it is not physically possible to sense the disturbance perfectly or predict the perfect control action. Therefore, a feed-forward controller is usually augmented with a feedback controller.

There are many possible benefits which can be realized from process control. The major benefits are improved performance, reduced capital requirements, increased reliability and process stability, lower operating and maintenance cost, and flexibility.

In this thesis, a neural network for chlorine contact basin is developed. The overall structure is depicted in Figure 4.1 and the following section describes the approach in detail.

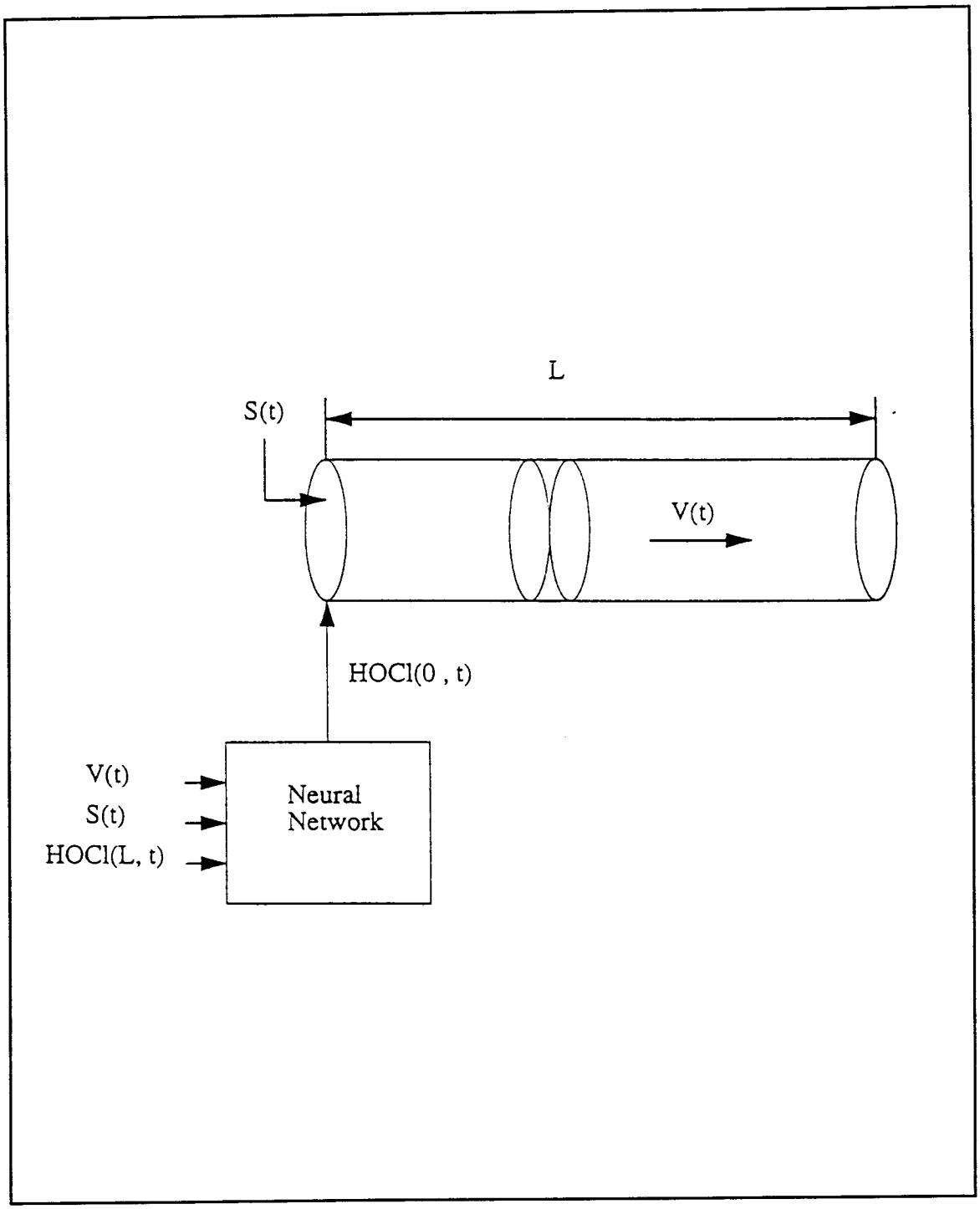


Figure 4. 1 Overall Structure of the Chlorination Process Control System

4.2 Chlorination Process Control using a Neural Network

The benefit of using a neural network is that one can build a system without a deterministic model for the system. One can build a system with representative samples of measured values. An advantage of a neural network is that it can still provide satisfactory performance in the presence of noisy data. The neural network can produce good mapping from a data set with some degree of errors.

4.2.1 Generation of Data Set

To build a neural network that controls a chlorination process, it is first necessary to provide a set of data that well represents an actual chlorination process. Since the necessary data are not available and its collection is beyond the scope of this thesis, the data set can be constructed from the model given in Chapter 2. The partial differential equation for the reaction in chlorination process has the solution as follows:

$$HOCl(x,t) = HOCl(0,t - L/V) * e^{-K*S(t-L/V)*L/V(t)} \qquad (4.2.1.1)$$

By assuming that S(t) and V(t) are stationary, the model equation becomes as follows:

$$HOCl(x,t) = HOCl(0,t-t_0) * e^{-K*S(t-t_0)*L/V(t-t_0)} \qquad (4.2.1.2)$$

where t_0 is the hydraulic retention time of the reactor.

In order to define a realistic problem and to restrict its size to a set of equations solvable in a reasonable time, the ranges of three primary model variables were restricted as follows:

S	:	[5, 15]	(mg/liter)
V	:	[0.0175, 0.0525]	(m/min)
HOCl(L, t)	:	[0.5 1.5]	(mg/liter)

with L = 40 (meters) and K = 1.66×10^{-4} (liter/mg-sec).

These values are necessary to provide a set of good samples that represent the probable range of an actual chlorination process. HOCl(0, t) is the initial concentration of chlorine injected to the system at the beginning of the chlorination process. The system will determine the proper value of HOCl(0, t) in response to the other parameters, S, V, and HOCl(L, t+L/V).

The simplified chlorination model described in Chapter 2 was used with the following additional restrictions:

1. The data values are lump-sum averaged over the bisection area of the process chamber.
2. The changes on the data values are slow, so that each instance can be treated as independent.
3. The velocity of the flow is stationary over the period of chlorination process and all points on the same bisection.

After applying these assumptions, the following relation between HOCl(0, t), S(t), and V(t) to HOCl(x, t) is applicable:

$$HOCl(x,t) = HOCl(0,t-t_0) * e^{-K*S(t-t_0)*L/V(t-t_0)} \qquad (4.2.1.3)$$

where t_0 is the time it takes for water flow through the chlorination process.

Using assumption 1, each data set can be treated as independent so that it is possible to define a set of values for training and testing. In order to cover the problem space, it is necessary to determine the data range for each variable, as follows:

S(t) :	[5, 15]	(mg/liter)
V(t) :	[0.0175, 0.0525]	(m/min)
HOCl(L,t) :	[0.5, 1.5]	(mg/liter)

with L = 40 (meters) and K = 1.66×10^{-4} (liter/mg-sec).

To obtain total coverage of the problem space, I divided each of data ranges in equal distances and found all permutations to form the input data set for the neural network training. From these values, the corresponding value of HOCl(L, t) was computed using equation 4.2.3 to form the training set. To distinguish HOCl(0, t) and HOCl(L, t+L/V), they were renamed HOClB(t) and HOCl(t), respectively.

The same equation can be used to generate the testing set. This time the input data values, S, V, and HOCl(t), are generated using a sine function. The sine function was chosen because it simulates the diurnal variation frequently observed at treatment plants.

$$S(t) = 10.0 + 5.0 * \sin(k_1 t) \quad (4.2.1.4)$$

$$V(t) = 0.035 + 0.0175 * \sin(k_2 t) \quad (4.2.1.5)$$

$$HOCl(0, t) = 1.0 + 0.5 * \sin(k_3 t) \quad (4.2.1.6)$$

The values of the constants, k_1 , k_2 , and k_3, were selected to make the resulting data set represent the actual operation observed in the chlorination process.

4.2.2 Implementation of Neural Network using Matlab

Matlab (Mathworks, Inc., 24 Prime Park Way, Natick, MA) was used for developing the neural network. It has several toolboxes for specific applications including neural network development. Matlab was originally developed as a tool for matrix computations. Since neural network operations are well expressed as set of matrix operations, Matlab is a natural fit for neural network development.

A Sun Sparcstation 5 was used to for simulating the neural network. Since I chose to use the Levengerg-Marquart training algorithm, I needed to find a proper computing platform which has a sufficient computing power and reliable memory management system for the training algorithm. A Sparcstation 5 from Sun Micro Systems was chosen because it met both criteria.

Given the data generated using the method in the previous section and a back-propagation neural network, a neural network was constructed for the chlorination process control. The purpose of the chlorination process control system is to minimize the amount of the chlorine usage to achieve the desired level of disinfection. It will also reduce the amount of chlorine residual at the end of the process as it achieves the optimum value of HOClB(t) for given conditions.

The input data consists of the desired chlorine concentration at the end of the process (HOClB(t)), the velocity of the water (V), and the concentration of the chlorine demanding compound in the influent (S). The output is the initial amount of chlorine for the chlorination process(HOCl(t)) to produce the desired end concentration. Given this information, it was determined that three input neurons and one output neuron were needed.

As mentioned earlier, it was decided to use the feed-forward, back-propagation neural network. The number of layers in the network must be determined. Even though any number of the hidden layers can be chosen, it has been shown that this type of problem can be solved using a three layer back-propagation neural network with tangent-sigmoid and linear transfer functions on the hidden layer and output layer, respectively [Freedman 1995]. Therefore, only one layer of hidden neurons with the tangent-sigmoid transfer function was selected.

The next step was the determination of the number of neurons in the hidden layer. There is a trade-off between the number of neurons, training time, and problem solving ability. With too few neurons, the network can not be trained to solve the given problem or the trained network can not handle the problem properly. Alternatively, a large number of neurons requires excessively long training time. There is no proven method to find the proper number of hidden neurons [Freedman 1995]. It requires trial-

method to find the proper number of hidden neurons [Freedman 1995]. It requires trial-and-error to determine the number of hidden neurons. After several attempts, it was found out that twenty-nine hidden neurons were adequate for the project.

The resulting neural network has the structure depicted in Figure 4.2.

The neural network toolbox of Matlab provides three learning algorithms. One is based on the simple delta rule, and the other two provide some improvement in terms of training convergence time. Since it is very time-consuming to train a neural network, it is important to select a learning algorithm with favorable training characteristics. The Levenberg-Marquardt learning algorithm is very efficient in terms of time required for training. It requires a large memory to store the transient results. According to the technical support personnel at Mathworks, Inc. (24 Prime Park Way, Natick, MA), the memory required for the execution of Levenberg-Marquardt algorithm can be computed by the following equation:

$$M = k * N * n_i * n_h * n_o * n_w \qquad (4.2.2.1)$$

where

M is the memory requirement,

N is the number of instances in the training set,

n_i is the number of neurons in the input layer,

n_h is the number of neurons in the hidden layer,

n_o is the number of neurons in the output layer,

n_w is the unit size of data, and

k is the multiplication factor.

For the case of the neural network in this thesis, the memory requirement is:

$$M = 348{,}000k \text{ (Bytes)}$$

From external measure, k has a value ranging 5 to 15 in most cases. The memory requirement for these conditions is between 1.6 to 4.8 MBytes. For some instances, the memory requirement for the program is much larger than this estimate so that the system is busy swapping memory to the hard disk, which is evidence that the algorithm requires more memory than the available physical memory. In such instances, the virtual memory system of SunOS™ on Sparcstation provides enough memory with reliability to guarantee the completion.

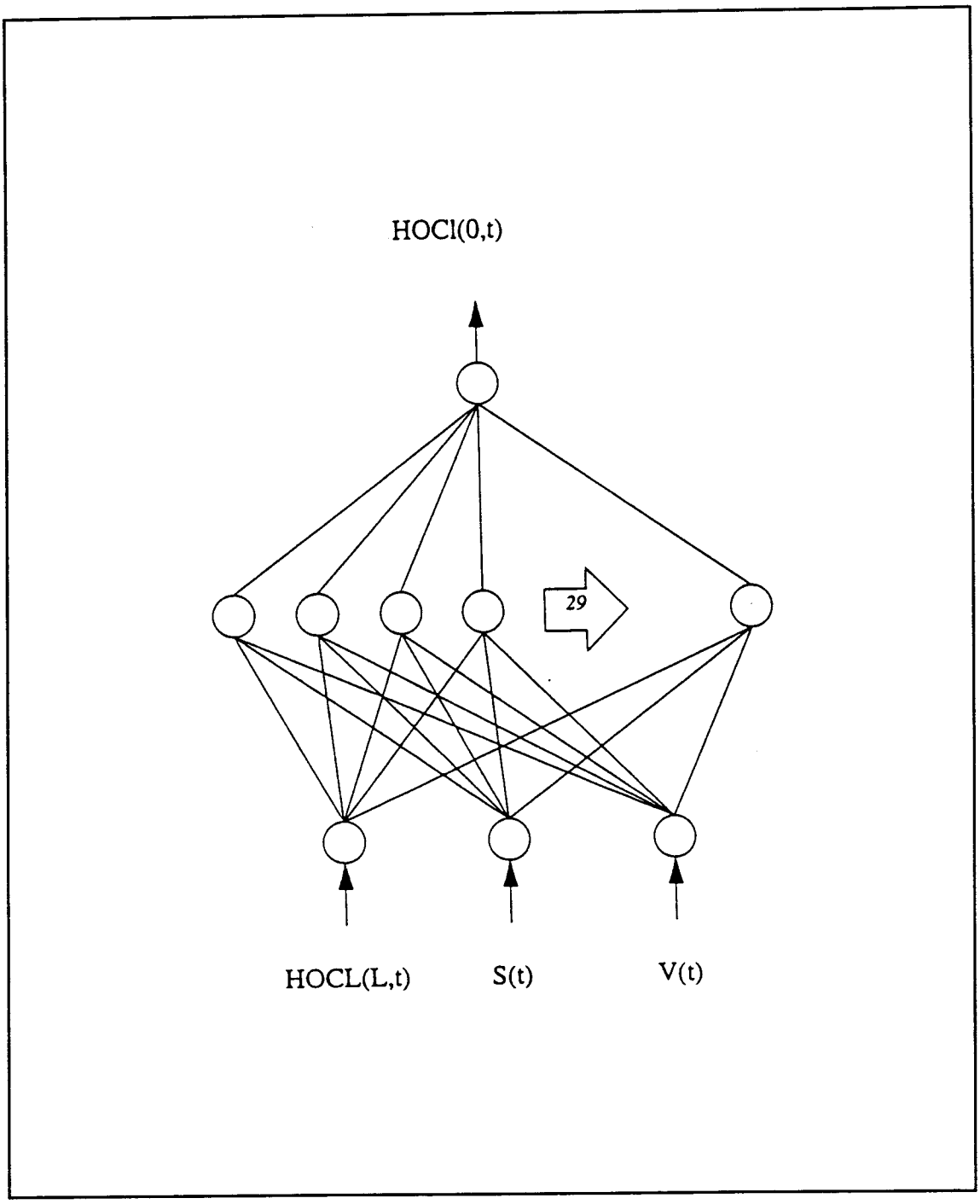


Figure 4. 2 The Architecture of a Neural Network for Chlorination Process Control [Nelson 1991]

The training parameters were initially determined as follows:

Maximum epochs	:	500
Error goal	:	0.01
Learning Rate	:	0.001
Momentum	:	0.5
Increment factor	:	5
Decrement factor	:	0.2

If the training can not reach the error goal within the maximum number of epochs, it can mean one of two things:

1. The data set does not provide enough relation for neural network to reach the error goal, or
2. It requires more training epochs to reach the error goal.

In the former case, one needs to check the resulting network with the training set and the testing set. If the performance is satisfactory, one can say the neural network is trained and ready to use. Otherwise, one needs to find a better way to train the network to improve the sum of squared errors. It is possible to change the number of neurons, the initial values of weights and biases, and the learning algorithm parameters,

such as learning rate, momentum, error goal, and increment/decrement factors. It is easy to change the parameters in the learning algorithm because it does not require restarting the algorithm. Other parameter changes require the training to be restarted.

In the latter case when an insufficient number of epochs were initially allowed, additional epochs can be allowed to continue the training until the training reaches the error goal, or the continued training fails to improve error. One can observe progress by inspecting the changes in sum of squared errors. In case 1, the error will not show improvement with additional epochs so that it will not reach the error goal no matter how many epochs are allowed. In case 2, each epoch shows reduction in the error.

5. Neural Network Training

Using the data and neural network described in the previous sections, the neural network was trained and used in a chlorination process control simulation. Two goals were established:

1. To examine the ability of neural network to find the necessary relation between input and output.
2. To examine the ability of neural network to handle the data with errors.

To achieve the second goal, I needed to introduce error to the training set. Uniformly distributed random noise (white noise) was introduced into the data set. The amount of error was adjusted to reflect different levels of data collection precision.

Appendix A shows the procedure used for the neural network training. The maximum error levels examined are 0.0%, 2.5%, 5.0%, 7.5%, and 10.0%. The procedure in Appendix A was modified to reflect each of these error levels.

Table 5.1 shows the summary of the neural network training. Each row represents a different noise level in the training set. The network training started with maximum epochs of 500. When the first attempt of network training reached the maximum

epochs, the resulting network was examined to use the training set and test set. If the performance of the network was satisfactory, the network was considered to be trained. If not, the training paramenters were changed and training was continued until the network produced satisfactory results. Table 5.1 contains the values of each training parameter at the conclusion of training for each data set. The momentum, increment, and decrement are same for all cases, and were the initial choices. The learning rates and maximum epochs were changed for the efficient training. It was found that the initial learning rate of 0.001 was most efficient. Once the initial training of 500 epochs failed, the pattern of the changes in the sum of squared errors was examined. If it was improving at the end of the training period, the training of the network using the same training parameters was continued. If the training was not improving, the learning rate was reduced by a factor of 5 or 10 to make the training progress. The training was considered to be progressing when the sum of squared errors level was continuously decreasing. Unlike other learning algorithms, the sum of squared errors monotonically decreases for the Levenberg-Marquardt learning algorithm. Therefore, it was concluded that network training was not improving when the sum of squared errors was not decreasing over a number of epochs. This indicated that the training should be changed. It was necessary to repeat the training process over again when the training was not successful because the initial values for weights and baises were important and randomly chosen at the beginning of the training.

Noise Level	Learning Rate	Momentum	Increment	Decrement	Epochs	SSE
0.00%	1.00E-03	0.5	5	0.2	700	14.3
2.50%	1.00E-04	0.5	5	0.2	750	1.4
5.00%	5.00E-05	0.5	5	0.2	625	9.5
7.50%	1.00E-04	0.5	5	0.2	1500	1557
10.00%	1.00E-04	0.5	5	0.2	625	29.2

Table 5.1 Neural Network Training Parameters

The initial network training did not achieve the error goal after 500 epochs. The pattern of change in the sum of squared errors was also not showing improvement. At this point the resulting neural network was examined to determine the desirability of further training. By checking the resulting neural network with the training set and the test set, it was concluded that further training was warranted. By trial-and-error approach, it was decided to reduce the learning rate by a factor of 5 or 10 and continue the training until the sum of squared errors reached less than 50.0. Through this procedure, neural networks with satisfactory performance could be obtained. The neural network performance was deemed satisfactory when it generated less than 10% error on the training set and the test set. This strategy worked for most of cases, except for the 7.5% error case. For some unknown reason, when the error level in training data was 7.5%, the sum of squared errors in the neural network training was not reduced below 1500; however, the resulting neural network showed satisfactory performance, and I decided to use the resulting neural network as it was.

By applying the test set to the resulting neural networks, the performance of the n networks was examined. Appendix B shows the procedure to generate the test set examine a neural network using the data.

In next chapter, the result of this experiment are discussed in detail.

6. Discussion

The results of the training can be examined by comparing the outputs generated by the neural network to the outputs obtained by applying the model equation, Equation 4.2.3. The upper chart of Figure 6.1 shows the mean and standard deviation of the percentage differences between the neural network generated values and the model generated values. The lower chart of Figure 6.1 shows the values of sum of squared errors at the end of the network training. The x-axes of the both graphs are the amount of noise introduced in the training set. Figure 6.1 shows the neural network errors which are within 10% limit, which was arbitrary selected as acceptable. Figure 6.2 shows the expected output for the first test set. In this test set, the required results of sine function were supplied. From the model equation, the value of the initial hypochlorous acid was obtained. Figures 6.3 through 6.7 show the neural network generated results. The values represent the error in the results when applying neural network generated values of the initial hypochlorous acid in to the model equation. This is equivalent to using the neural network as a feed forward controller. The errors are computed using the following equation:

$$Error = \frac{(TestHOCL - HOCL)}{HOCL} \tag{6.1}$$

where

TestHOCl is the neural network generated output and

HOCl is the desired results.

Figures 6.2 through 6.7 show that the neural network generated values stay well within 10% error range except in some extreme cases. Therefore, it is concluded that the neural network can tolerate 10.0% noise in the training set for chlorination process control. However, I can also see that lower error in training set resulted in lower error in the results generated by the neural network, except for the case with 2.5% error in the training data. The amount of noise in the training set can affect the quality of the network. As one can see in the upper-subgraph of Figure 6.1, smaller noise in the training set resulted in a neural network generating less error in the results. Therefore obtaining the most accurate input data set is still important to ensure the quality of the resulting neural network.

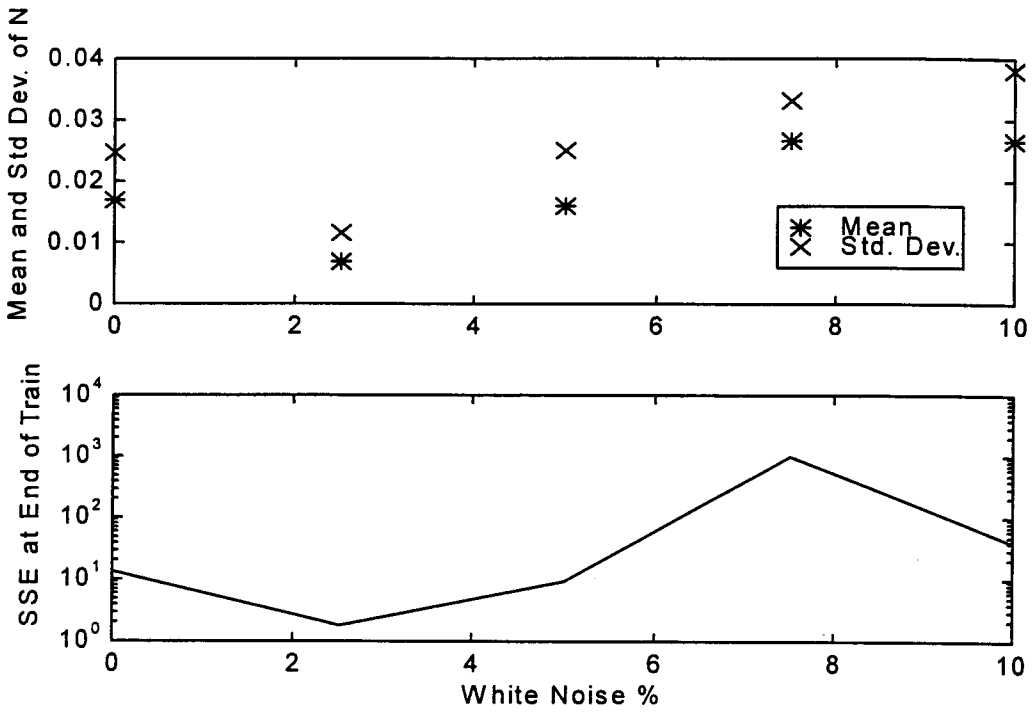


Figure 6. 1 Mean and Standard Deviation of Neural Network Generated Data on Inpu in Training Set with Sum of Squared Errors at the End of Training

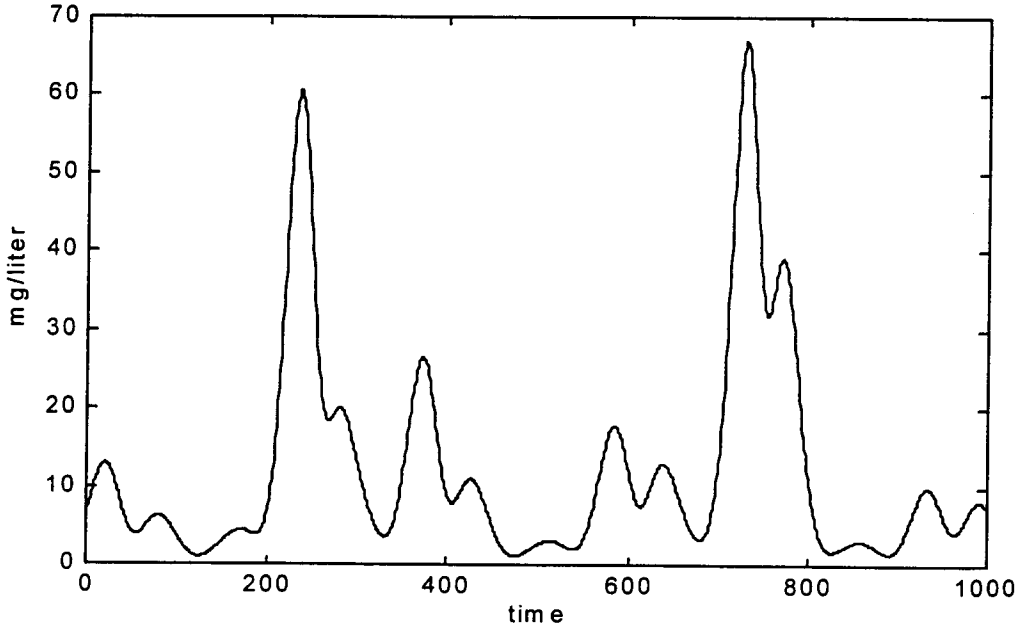


Figure 6. 2 The Expected Output: HOCl(0, t)

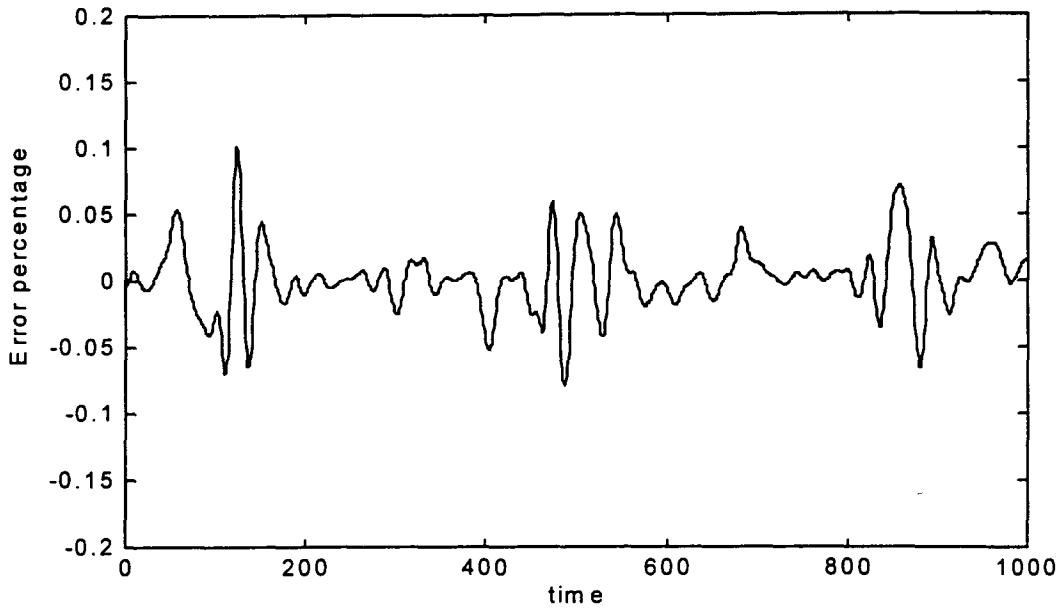


Figure 6. 3 Error in Neural Network Controlled Results (0.0 % Noise in Training Set)

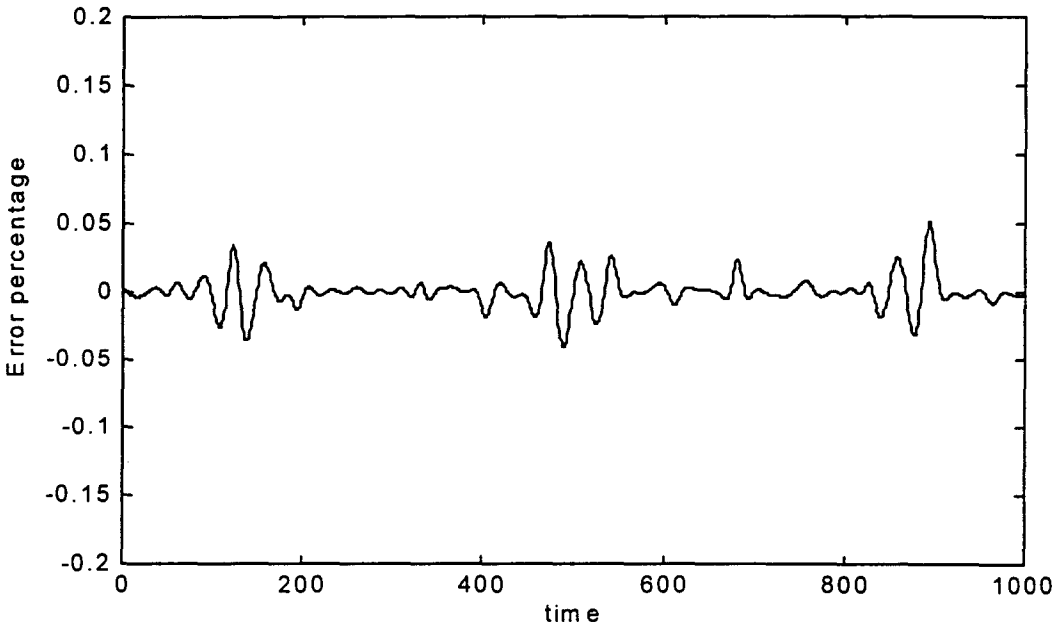


Figure 6. 4 Error in Neural Network Controlled Results (2.5 % Noise in Training Set)

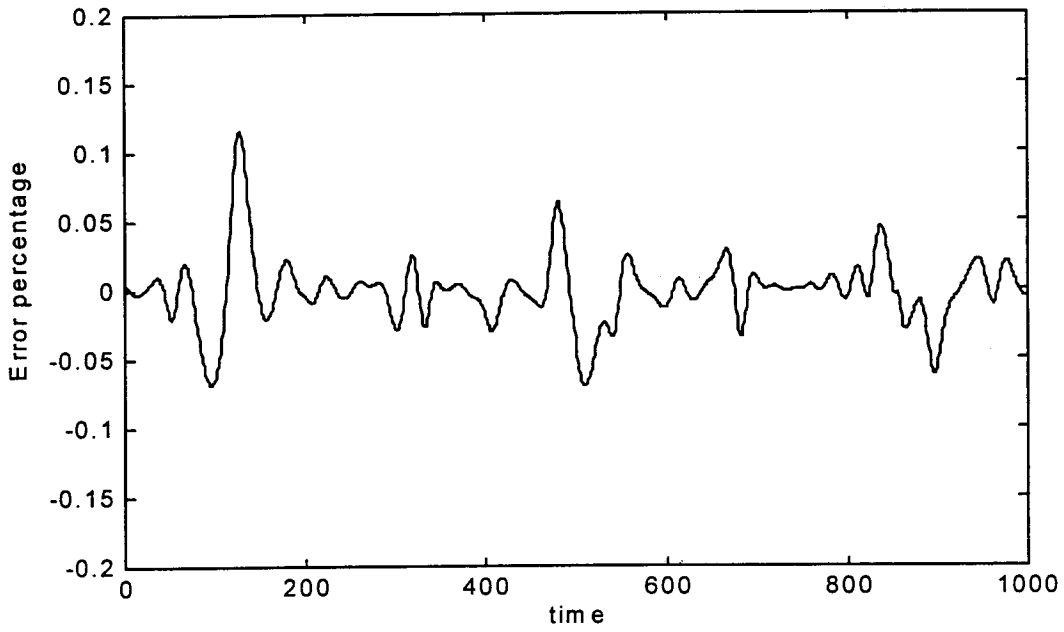


Figure 6. 5 Error in Neural Network Controlled Results (5.0 % noise in Training Set)

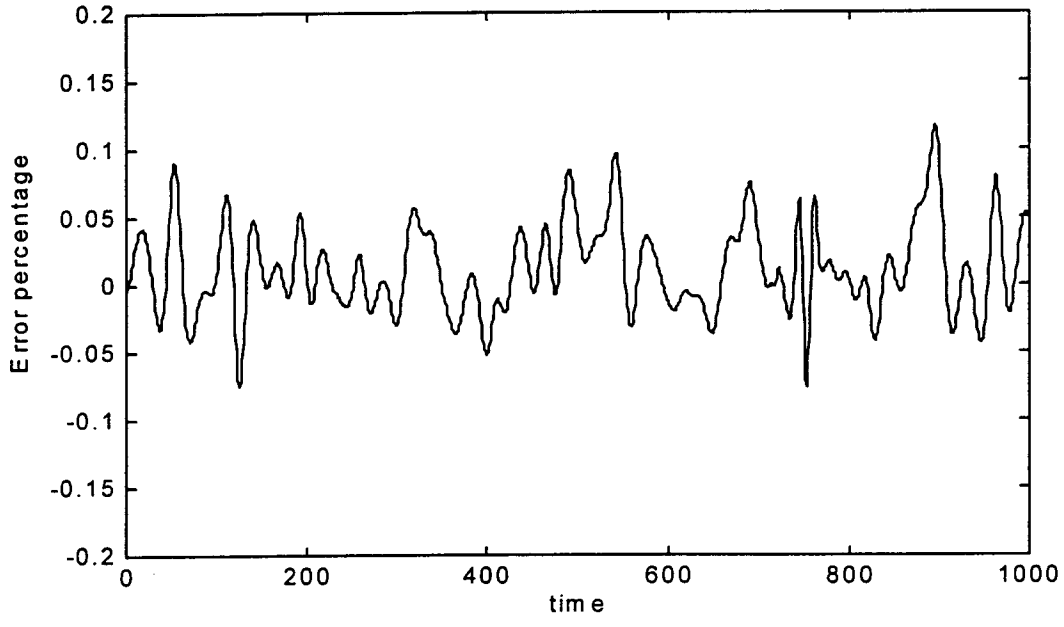


Figure 6. 6 Error in Neural Network Controlled Results (7.5 % Noise in Training Set)

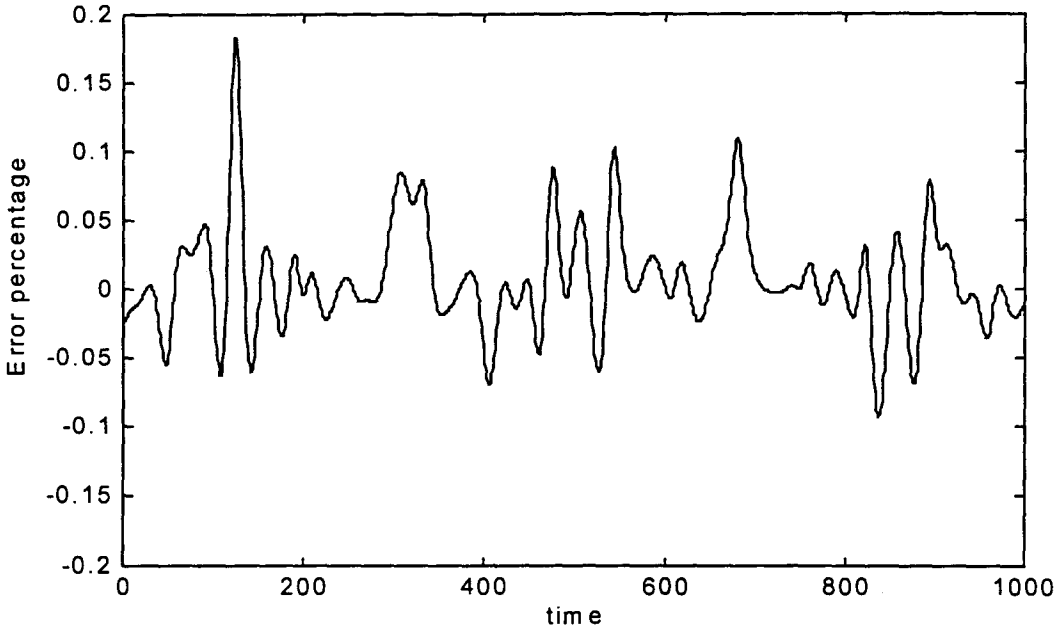


Figure 6. 7 Error in Neural Network Controlled Results (10.0 % Noise in Training Set)

The neural network generated results have the lowest sum of squared errors when the amount of noise in the training set is 2.5%, and not 0.0%. For this training set, the neural network reaches an unusually low sum of squared errors at the end of training, which is 1.4363. For the other cases, the sums of squared errors are higher than 9.0 (Table 5.1). This result suggests that reaching a lower sum of squared errors at the end of training will construct a better quality neural network. However, this is not true in general because of a phenomenon called overtraining.

Overtraining occurs when a network has learned not just the basic mapping associated with the input and output data presented to it, but also the subtle nuances and even the errors specific to the training set. An overtrained network performs very well on the training set, but performs poorly on data values not included in the training set [Freedman 1995]. To examine the neural network for overtraining, one needs to provide a test set which contains some variation of data values. At the end of network training, the test set is applied to the network to see how well it performs on unseen data values. Therefore, the training of a neural network is terminated when the network performs well on both the training and test sets.

The neural network under investigation does not seem to suffer from over-training because the data in the training set well cover the ranges for each input variable. Conceptually, the neural network is finding a mapping between the input data and

output data throughout the course of training. Therefore, when one provides the training set that well covers the ranges of the input parameters, the resulting neural network is not overtrained. For the cases for which one can not provide a data set to cover the data ranges, one needs to provide many test sets with different combinations and carefully examines the neural network during the training for different sum of squared errors.

Figure 6.8 through 6.12 show the results of applying a second test set to the neural networks. In this test set, the hypochlorous acid concentration at the end of the chlorination process is constant, 1.0 (mg/liter). The neural network generated values are applied to the model equation, Equation 4.2.3. As shown in the figures, the neural networks generate satisfactory results. All neural networks keep the hypochlorous acid concentrations within 10% of the desired values.

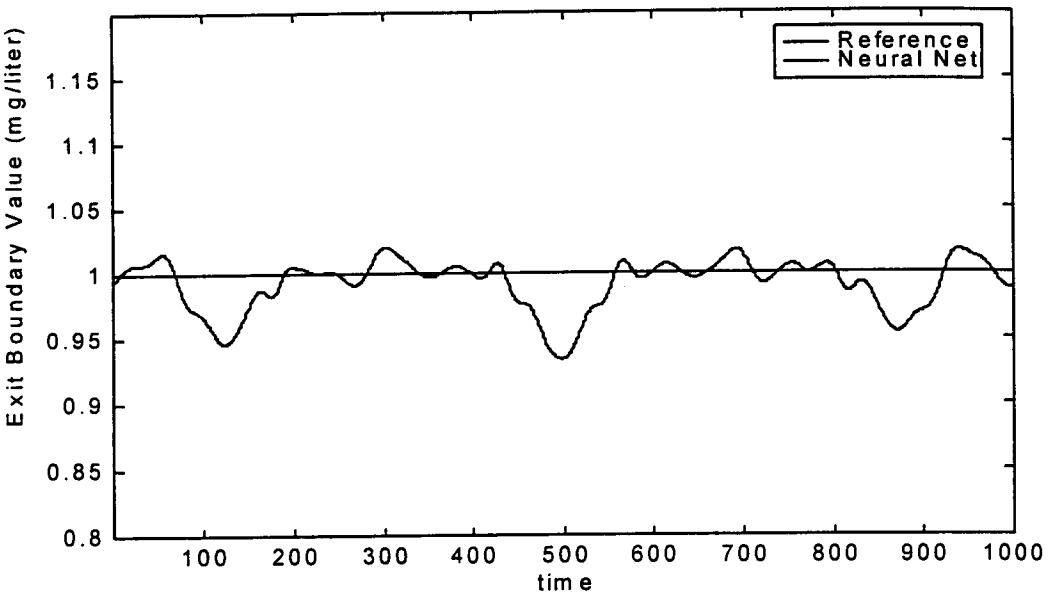


Figure 6. 8 Comparison of Expected Output and Neural Network Generated Values (% Noise in Training Set)

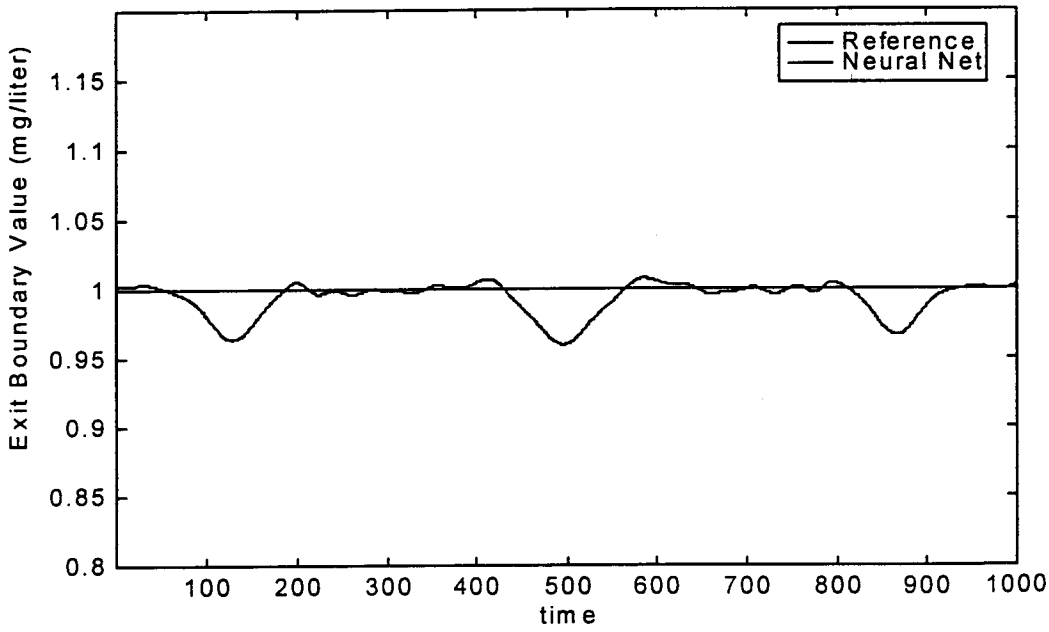


Figure 6. 9 Comparison of Expected Output and Neural Network Generated Valu (2.5% Noise in Training Set)

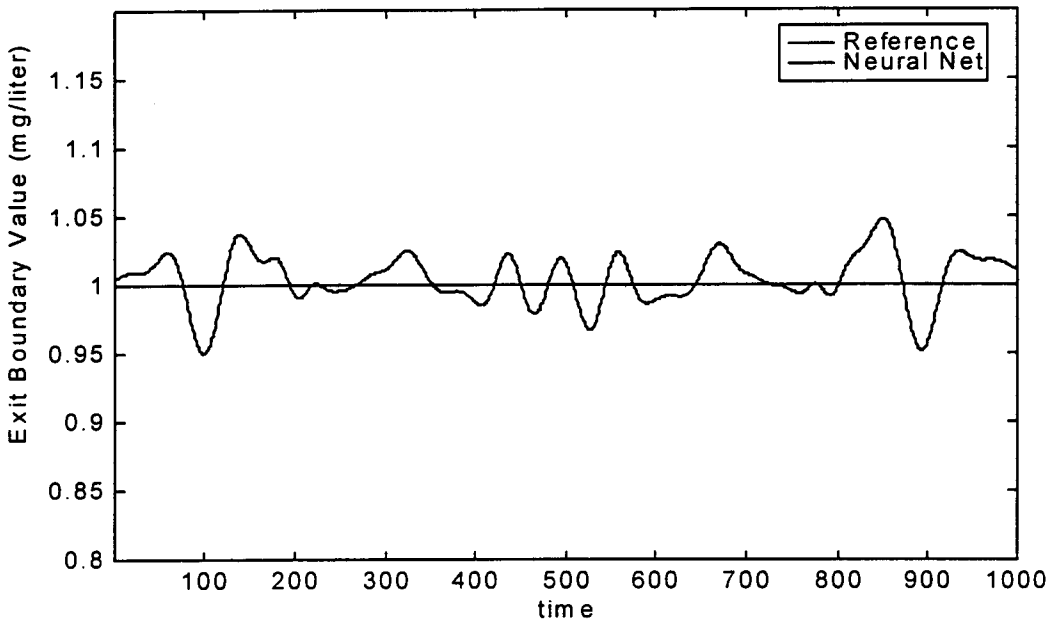


Figure 6. 10 Comparison of Expected Output and Neural Network Generated Values (5.0 % Noise in Training Set)

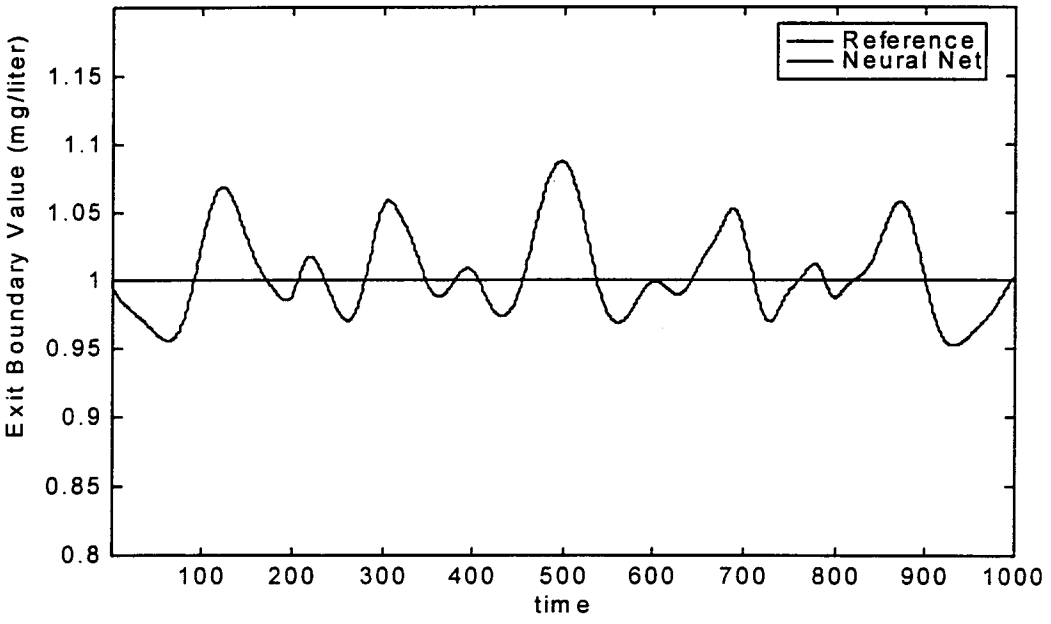


Figure 6. 11 Comparison of Expected Output and Neural Network Generated Values (7.5 % Noise in Training Set)

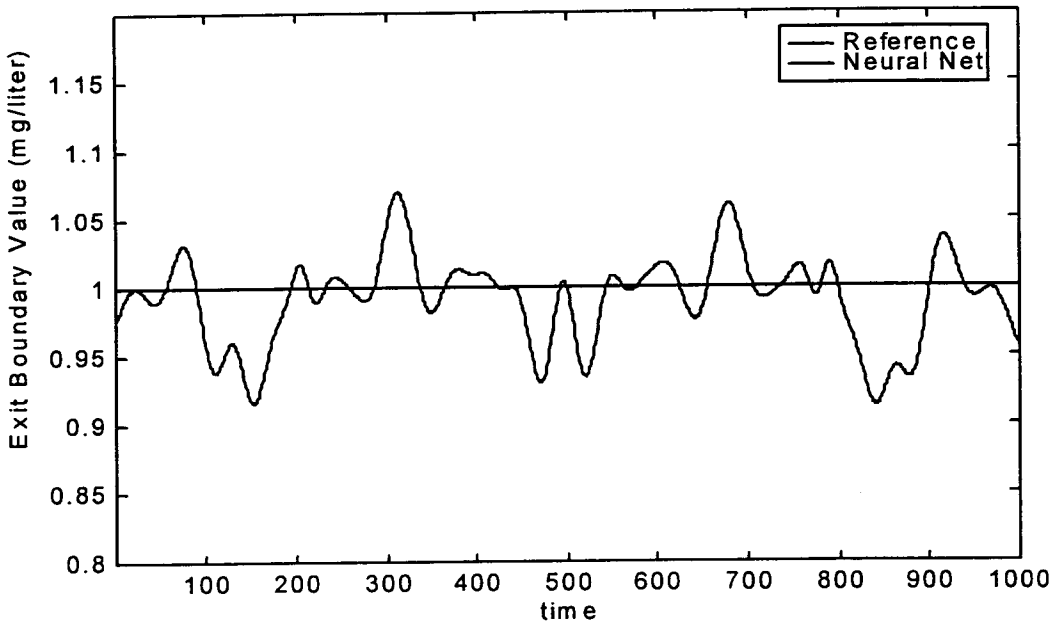


Figure 6. 12 Comparison of Expected Output and Neural Network Generated Values (10 % Noise in Training Set)

7. Conclusion

This thesis showed that a neural network could determine influent chlorine concentration for a simplified ideal chlorine reactor to provide the desired effluent chlorine concentration for varying flow velocities and chlorine demand concentration. For this study, I used a simplified model for the chlorine reactor assuming that chlorine reacts only with a single, chlorine demanding substance to generate data sets necessary to build and examine the neural network. To replicate the real situation, I introduced different levels of noise which reflect the different accuracy of measurements.

The neural network consisted of 3 layers with 29 neurons in the hidden layer. The neural network was trained using 5 different training sets, each containing diffrent noise levels ranging from 0% to 10%. The training was completed in 625 to 1500 epochs depending on the noise level in the training set. The resulting neural network was able to predict the influent chlorine concentration within the error tolerance of 10% regardless the noise level of the data in the training set.

From the result, an efficient chlorination process control system could be built using neural network technology with a set of measured data. After the completion of training using the data set, the neural network found necessary correlation of input data values

(the desired effluent chlorine concentration, the chlorine demanding compound concentration, and the flow velocity) to the desired output value.

This result suggests that neural network based control system is a promising method for chlorination process control, and further research and development are warranted.

8. Limitations and Further Research

For this project, I have made a great deal of assumptions for simplicity as listed Chapter 4. These assumptions are reasonable for a first attempt in developing a ne network control system for a chlorination, but impose severe limitations on resulting neural network. To eliminate these assumptions, one needs to include m parameters in the input data set. Additional values of HOCl(x, t), V(t), and S(t) r need to be included to introduce non-stationary behavior into the chlorination mo One will also need to include other parameters such as pH and temperature.

In this project, I assumed a strong correlation between the HOCl concentration and level of disinfection. This relationship has been confirmed by many research however, it may be better to have a measure of disinfection such as the concentratio coliform bacteria. At present, real time indicators of disinfection efficiency measure indicator organisms or pathogen concentration are not available; neverthe such indicators would improve the control system.

In the process of modeling, it is better to include more mechanisms and relati which will increase the size of neural network training program. The increas computer technology will facilitate the development of more advanced neural netw

It provides a new opportunity for the investigation of the optimization chlorination process.

Appendix A

Neural Network Training Procedure without Error

```
% Determine the error level introduced into the training set
Noise = 0.0;

% Determine the number of steps within the range of data values
n = 9;

% Input data generation
for i=1:n,
 for j=1:n,
  for k=1:n,
   index      =(i-1)*n*n+(j-1)*n+k;
   HOCl(index) = 0.5+(i-1)*1/(n-1);
   S(index)    = 5 + (j-1)*10/(n-1);
   V(index)    = 0.0175+(k-1)*0.035/(n-1);
  end
 end
end

K = 1.66e-4;
L = 40;


% Desired output given the input generated earlier, it includes the
% error defined earlier.
HOClB = HOCl.*exp(K*L*S./V);
nt1 = HOClB.*(Noise*rand(1,n*n*n)+1);

np1(1,:) = HOCl;
np1(2,:) = S;
np1(3,:) = V;

% Number of neurons in hidden layer
NoNeurons = 29;
```

```
% Define and set the training parameters
P = [min(np1')*0.99; max(np1')*1.01]';
disp_freq = 1;
max_epoch = 250;
err_goal = 0.01;
lr = 0.001;
momentum = 0.5;
inc = 5;
dec = 0.2;
[w1,b1,w2,b2] = initff(P, NoNeurons, 'tansig', HOClB, 'purelin');
tp = [disp_freq max_epoch err_goal lr momentum inc dec];

% Actual training procedure
[w1,b1,w2,b2,te,tr] = trainlm(w1,b1,'tansig',w2,b2,'purelin',np1,nt1,tp);
```

Appendix B

Neural Network Testing Procedure

```
% This program is a test program for neural network for chlorination

% process control. It is to beused for generating resut for report.

% number of steps in the testing set
time = 1:1000;

% Generating test input data using sine functions
TestHOCl = 1 + 0.5*sin(2*pi*time/70);
TestS = 10 + 5*sin(2*pi*time/180);
TestV = 0.035 + 0.0175*sin(2*pi*time/400);

% Some parameters
K = 1.66e-4;
L = 40;

% Computing the desired output for the test input data
TestHOClB = TestHOCl.*exp(K*L*TestS./TestV);

% Form test input set for neural network
testp(1,:) = TestHOCl;
testp(2,:) = TestS;
testp(3,:) = TestV;

% Applying test input to the neural network
TestHOClB = simuff(testp, w1, b1, 'tansig', w2, b2, 'purelin');

% Applying the result into the model equation
TestResult = TestHOClB.*exp(-K*L*TestS./TestV);

% Comparing the result with expected output
TestResult2 = (TestResult-TestHOCl)./TestHOCl;

% Computing mean and standard deviation of error
[mean(abs(TestResult2)), std(TestResult2)]
```

References

Bryant, J. O. (1972) Contonuous Time Simuation of the Conventional Activated Sludge Wastewater Renovation System, Ph. D. Dissertation, Clemson University, Clemson, SC.

Caudill, M. and Butler, C. (1994) Understanding Neural Networks, The MIT Press, Cambridge, MA.

Chapin, R. M. (1931) "The Influence of pH upon the Formation and Decomposition of the Chloro Derivatives of Ammonia.", Journal of Chemical Society, v.53, pp. 913-920.

Demuth, H. and Beale, M. (1994) Neural Network Toolbox, The Math Works, Inc., Natick, MA.

Feben, D. and Taras, M. J. (1950) "Chlorine Demand Constants of Detroit's Water Supply.", Jornal of American Water Works Association, v.42, pp. 453-461.

Feben, D. and Taras, M. J. (1951) "Studies on Chlorine Demand Constants.", Journal of American Water Works Association, v.43, pp. 922-931.

Freedman, R. S., Klein, R. A., and Lederman, J. (1995) Artificial Intelligence in the Capital Market, Probus Pub., Chicago, IL, pp. 87-133.

Griffin, A. E. (1940) "Observation on Break Point Chlorination.", Journal of American Water Works Association, v.32, pp. 1187-1190.

Griffin, A. E. and Chamberlin, N. S. (1945) "Exploring the Effect of Heavy Sc Chlorine in Sewage.", Journal of Sewage Works, v.17, pp. 730-742.

Kokoropoulos, P. and Manos, G. P. (1973) "Kinetics as Design Criteria for Chlorination.", Journal of Environmental Engineering Divisions, ASCE, v.99, p 83.

Montgomery, J.M. (1985) Water Treatment Principles and Design, John Wi Sons, Inc., New York, NY.

Morris, J. C. (1965) "Kinetic Reactions between Aqueous Chlorine and Nit Compounds.", Proceedings of the Fourth Rudolphs Research Conference, F Rudolphs Research Conference, Rutgers University.

Morris, J. C. (1966) "The Acid Ionization Constant of HOCl from 5 to 35°C.", Jc of Physical Chemistry, v.70, pp. 3798.

Morris, J. C. and Wei, I. (1969) "Chlorine-Ammonia Breakpoint Reactions: M Mechanisms and Computer Simulation.", Paper presented at the American Che Society, Minneapolis, MN.

Nelson, M.M. and Illingworth, W.T. (1991) A Practical Guide to Neural Addison-Wesley Publishing Company, Inc., Reading, MA.

Reynold, T.D. (1982) Unit Operations and Processes in Environmental Engineе PWS-KENT Publishing Company, Boston, MA.

Rossum, R. J. (1943) "A Proposed Mechanism for Breakpoint Chlorination.", Journal American Water Works Association, v.35, pp. 1446-1449.

Stenstrom, M.K. (1976) A Dynamic model and Computer Compatible Control Strategies for Wastewater Treatment Plants, Ph.D Dissertation, Clemson University, Clemson, SC.

Stenstrom, M.K. (1980) "Optimization of Chlorination of Activated Sludge Plant Effluent by Ammonia Control.", Paper Presented at Joint Automatic Control Conference, San Francisco, CA.

Taras, M. J. (1950) "Preliminary Studies on the Chlorine Demand of Specific Chemical Compounds.", Journal of American Water Works Association, v.42, pp. 462-474.

Taras, M. J. (1953) "Effect of Free Residual Chlorination on Nitrogen Compounds in Water.", Journal of American Water Works Association, v. 45, pp. 47-61.

White, G.C. (1992) Handbook of Chlorination and Alternative Disinfectants, Van Nostrand Reinhold, New York, NY.

Zupan, J. and Gasteiger, J. (1993) Neural Network for Chemists, VCH Publishers, New York, NY.