Neural Network Calibration of Moderator Temperature Coefficient Measurements in Pressurized Water Nuclear Reactors

by

Joseph T. Adams

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in
Mechanical Engineering

APPROVED:

James R. Thomas, Jr., Chairman

Dr. Alfred L. Wicks Dr. Felix J. Pierce

May, 1993
Blacksburg, Virginia
c.2

405
V855
1993
A3310
2.2
Neural Network Calibration of Moderator Temperature Coefficient Measurements in Pressurized Water Nuclear Reactors

by

Joseph T. Adams
Committee Chairman: Dr. James R. Thomas, Jr.

Mechanical Engineering

(ABSTRACT)

Neural networks have been shown to be capable of predicting the moderator temperature coefficient in a nuclear reactor by using the frequency response functions between the in-core neutron flux signal and the ex-core thermocouple signal as inputs. In this work, actual data from a nuclear reactor is used by neural networks to estimate the moderator temperature coefficient at different times during a fuel cycle. Along with the conventional method of training neural networks, a new method of training that better models the use of neural networks in predicting the moderator temperature coefficient is also successfully demonstrated. The results show that neural networks are effective at estimating the moderator temperature coefficient if the domain of prediction is within the training domain of the network. The advantage of using the autoregression method to create the frequency response patterns used as inputs to the neural network as opposed to frequency response functions calculated by the Fourier transform method is also shown.
ACKNOWLEDGEMENTS

The author would like to thank Dr. James R. Thomas, Jr. for his support, patience, and help with this thesis. I also want to thank Dr. Alfred Wicks for the helpful conversations concerning signal processing and Dr. Felix Pierce for his review and constructive input to this thesis.

In addition, I want to thank Dr. John D. Herr, Ph.D. for I have drawn heavily from his Doctoral dissertation in preparation of this thesis. I am also indebted to Allen W. Clem for the use of his Master's thesis to prepare this thesis. I want to thank Dr. O. Glockler for his help and suggestions for using neural networks to estimate the moderator temperature coefficient in a pressurized water reactor. I especially want to thank Mark J. Embrechts for his assistance with the neural network software package, METANEURAL.

Finally, I would like to thank my parents for their support during this endeavor and my friends for being there when I needed to take a break from work. Without them, I would not have been able to accomplish this thesis.
# TABLE OF CONTENTS

1 INTRODUCTION ................................................................................. 1

1.1 Moderator Temperature Coefficient in PWR's ........................................ 1

1.2 MTC Measurement: Current Method .................................................... 4

1.3 MTC Measurement: Stochastic Signal Analysis Using Fast Fourier Transform of Data .............................................................................. 5

1.4 MTC Measurement: Stochastic Signal Analysis Using the Multivariate Autoregression Method .......................................................... 7

1.5 MTC Measurement: Neural Networks ..................................................... 8

1.6 Objective of Research .......................................................................... 12

2 LITERATURE REVIEW ........................................................................ 14

2.1 MTC Estimating Methods Using Noise Analysis .................................... 14

2.2 MTC Estimates Using Neural Networks ................................................ 16

3 NEURAL NETWORKS ......................................................................... 18

3.1 Introduction ....................................................................................... 18

3.2 Training a Two-layered Feedforward Network .................................... 21

3.3 Training a Multi-layer Feedforward Network ..................................... 26

3.4 Bias Nodes ....................................................................................... 33
4 NEURAL NETWORK ANALYSIS USING FAST FOURIER TRANSFORM OF DATA ................................................................. 35
  4.1 Introduction ............................................................................................................. 35
  4.2 Data Acquisition ..................................................................................................... 35
  4.3 Frequency Range Selection ....................................................................................... 38
  4.4 Neural Network Program ......................................................................................... 41
  4.5 Analysis Using Method of Upadhyaya and Glockler .............................................. 44
  4.6 Use of Neural Networks to Predict Values (Proposed Method) ......................... 48

5 NEURAL NETWORK ANALYSIS USING MULTIVARIATE AUTOREGRESSION METHOD .................................................. 55
  5.1 Introduction ............................................................................................................. 55
  5.2 Data Acquisition ..................................................................................................... 55
  5.3 Data Analysis .......................................................................................................... 56
  5.4 Frequency Range Selection ....................................................................................... 62
  5.5 Neural Network Program ......................................................................................... 62
  5.6 Analysis Using the Approach of Upadhyaya and Glockler .................................... 64
  5.7 Use of Neural Networks to Predict Values (Proposed Method) ......................... 67

6 CONCLUSIONS, LIMITATIONS, & RECOMMENDATIONS .................. 74
  6.1 Conclusions ............................................................................................................... 74
  6.2 Limitations .............................................................................................................. 76
  6.3 Recommendations ................................................................................................. 77
# LIST OF ILLUSTRATIONS

## FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Two-layer Feedforward Neural Network (the Input Layer, and the Output Layer)</td>
<td>10</td>
</tr>
<tr>
<td>1.2</td>
<td>Three-layer Feedforward Neural Network (the Input Layer, One Hidden Layer, and the Outer Layer)</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Multi-stage Feedforward Neural Network</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Two-layer Feedforward Neural Network</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Three-layer Feedforward Neural Network</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Three-layer Network with Bias Nodes</td>
<td>34</td>
</tr>
<tr>
<td>4.1</td>
<td>$H_1$ Estimator of the FRF Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: McGuire, Unit 2, Cycle 5</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>$H_2$ Estimator of the FRF Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: McGuire, Unit 2, Cycle 5</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>$H_3$ Estimator of the FRF Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: McGuire, Unit 2, Cycle 5</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Coherence Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: Duke McGuire, Unit 2, Cycle 5</td>
<td>42</td>
</tr>
</tbody>
</table>
Figure 4.5 Training Error vs. Amount of Training (Neural Network Using Fourier Transformed Data from Duke McGuire, Unit 2, Cycle 5) ........................................................................................................ 46

Figure 4.6 Training Error vs. Prediction Error (Neural Network Using Fourier Transformed Data from Duke McGuire, Unit 2, Cycle 5) ........................................................................................................ 47

Figure 4.7 Neural Network Results Using Upadhyaya and Glockler's Method for Duke McGuire, Unit 2, Cycle 5 [FFT Processed Data] ........................................................................................................ 49

Figure 4.8 Prediction Error vs. Training Error for the Proposed Method (Neural Network Using Transformed Data from Duke McGuire, Unit 2, Cycle 5) ........................................................................................................ 53

Figure 4.9 Neural Network Results Using Proposed Method for Duke McGuire, Unit 2, Cycle 5 [FFT Processed Data] ........................................................................................................ 54

Figure 5.1 $H_1$ FRF Estimator Results for Duke McGuire, Unit 2, Cycle 5 for AR Model 30 ........................................................................................................ 59

Figure 5.2 $H_2$ FRF Estimator Results for Duke McGuire, Unit 2, Cycle 5 for AR Model 30 ........................................................................................................ 60

Figure 5.3 $H_3$ FRF Estimator Results for Duke McGuire, Unit 2, Cycle 5 for AR Model 30 ........................................................................................................ 61

Figure 5.4 Coherence Between Core-exit Thermocouple and In-core Neutron Flux Detector for Duke McGuire, Unit 2, Cycle 5 for AR Model 30 ........................................................................................................ 63

Figure 5.5 Training Error vs. Amount of Training (Neural Network Using AR Processed Data from Duke McGuire, Unit 2, Cycle 5) ........................................ 65

Figure 5.6 Training Error vs. Prediction Error (Neural Network Using AR Processed Data from Duke McGuire, Unit 2, Cycle 5) ........................................ 66
Figure 5.7 Neural Network Results Using Upadhyaya and Glockler's Method for Duke McGuire, Unit 2, Cycle 5 [AR Processed Data] ................................................................. 68

Figure 5.8 Prediction Error vs. Training Error for the Proposed Method (Neural Network Using AR Processed Data from Duke McGuire, Unit 2, Cycle 5) .............................................................................. 71

Figure 5.9 Neural Network Results Using Proposed Method for Duke McGuire, Unit 2, Cycle 5 [AR Processed Data] ......................................................... 72

TABLES

Table 4.1 Predicted MTC Values at a Training Error of 0.008944 for the Duke McGuire Plant, Unit 2, Cycle 5 [FFT Processed Data] ....................... 51

Table 5.1 Predicted MTC Values at a Training Error of 0.008944 for the Duke McGuire Plant, Unit 2, Cycle 5 [AR Processed Data] ....................... 70
Chapter 1

INTRODUCTION

1.1 Moderator Temperature Coefficient in PWR's

In a nuclear reactor, energy is released through the process of nuclear fission. Fission occurs when a neutron strikes a Uranium-235 nucleus, a fissile isotope used as fuel in a nuclear reactor, causing it to split into smaller nuclei and to release 1 to 5 neutrons and energy [14]. The released neutrons then collide with other $^{235}\text{U}$ nuclei, causing some of them to split. Thus, a fission chain reaction is created.

The multiplication factor, $k$, is a method of quantitatively describing the fission chain reaction. It is defined as the ratio of the fissions in one generation divided by the number of fissions in the preceding generation [1], or,

$$k = \frac{\text{number of fissions in one generation}}{\text{number of fissions in preceding generation}}. \quad (1.1)$$

A critical chain reaction is when the number of fissions from generation to generation is constant, or $k$ equal to 1 [1]. If $k$ is greater than 1, the number of fissions increase from generation to generation, and the chain reaction is supercritical [1]. Finally if $k$ is less than 1, the number of fissions decrease from one generation to the next, and the chain reaction is subcritical [1].

Another method of describing the chain reaction is through the term reactivity. The reactivity, $\rho$, of a reactor is defined as

$$\rho = \frac{k - 1}{k} \quad (1.2)$$
where \( k \) is the multiplication factor \([1]\). If a reactor has positive reactivity, the chain reaction is supercritical. If the reactor has negative reactivity, the chain reaction is subcritical \([1]\).

In a reactor, the reactivity is affected by many parameters, including changes in temperature. The change in reactivity per degree change in temperature is described by the temperature coefficient of reactivity, \( \alpha_r \), and is defined by the relation

\[
\alpha_r = \frac{\rho}{dT},
\]

(1.3)

where \( \rho \) is the reactivity and \( T \) is the temperature \([1]\).

Temperature changes are not uniform throughout a reactor, however, and it is necessary to discuss the specific component that is undergoing a temperature change. The fuel temperature coefficient is defined as the change in reactivity per degree change in fuel temperature, while the moderator temperature coefficient (MTC) is defined as the change in reactivity per degree change in moderator temperature \([1]\).

The moderator in a PWR is light water or ordinary \( \text{H}_2\text{O} \), which also acts as the coolant. The hydrogen atoms of the moderator slow down the fast neutrons emitted during the fission reaction. The fast neutrons strike the hydrogen atoms causing the fast neutrons to slow down. The slowed neutrons, called thermal neutrons, have a better chance of hitting a \(^{235}\text{U} \) nucleus making the fission process more likely.

Because of the core subsystem dynamic behavior, the moderator and fuel temperature coefficients can be separated \([9]\). In a PWR, changes in fuel temperature will cause changes in the moderator temperature through the heat transfer process. Also, changes in the moderator temperature can cause changes in the fuel temperature through the heat transfer process and thus affect the neutron flux. The heat transfer dynamics, however,
have a relatively large time constant (~5 sec.) [9]. Therefore the high frequency components in the fuel temperature fluctuations will be greatly attenuated in the heat transfer process, and the fuel temperature will have a negligible effect on moderator temperature above 0.1 Hz [9]. Thus, the moderator temperature fluctuations are the driving force for reactivity fluctuations in the frequency range above 0.1 Hz. Fuel temperature fluctuations become more important for reactivity fluctuations at frequencies below 0.1 Hz.

The MTC is important because it controls the response of a pressurized water reactor (PWR) to changes in moderator temperature. For a PWR to be stable, the MTC must be negative. The Nuclear Regulatory Commission (NRC) will not license a nuclear reactor unless the MTC is negative at operating temperatures [1]. If the MTC is positive, the PWR is unstable, as can be understood as follows.

In the case of a negative MTC, a decrease in temperature in a critical reactor produces positive reactivity. Since more fissions are occurring, more power is generated, making the temperature increase. An increase in temperature leads to a decrease in the reactivity through the MTC, and reactor power reaches a new equilibrium.

For a positive MTC, a decrease in temperature causes a decrease in reactivity resulting in a decrease in generated power. As the power decreases, the temperature decreases causing the reactivity to decrease. This process continues until the reactor shuts down.

Even though it is desirable for the MTC to be negative, a very large negative MTC can be dangerous in some situations, for example in the event of a hypothetical main steam line rupture. After the rupture, the reactor core would be flooded with much cooler water, reducing the core temperature. With a large negative MTC and a large decrease in
temperature, a large increase in power is generated. This increase in power may be so rapid that the control rods might not be inserted fast enough to prevent core damage. To prevent this accident from occurring, the NRC regulates the magnitude of the MTC in many power reactors [2].

Because the MTC is such an important safety parameter in a nuclear reactor, accurately measuring the MTC is an important task.

1.2 MTC Measurement: Current Method

The current method used by Virginia Power, Duke Power and many other utilities to determine the MTC is the boron-dilution method. The steam throttle is opened and the average temperature of the core coolant is lowered. To offset the reactivity increase, soluble boron, a neutron absorber, is added to the coolant. When the coolant temperature is stabilized, the boron concentration is measured and the core-average temperature is recorded. Based on the change in boron concentration during the test, the reactivity change can be calculated. The calculated change in reactivity divided by the change in the average coolant temperature yields a measure of the MTC [2]. This method is not ideal for the following reasons [2,7,10]:

1. Lowering the average temperature of the core changes the axial power distribution in the core, leading to fission product poison transients which must be monitored for about 24 hours.
2. The longer the test takes to execute, the larger the variance of the results due to induced transients.

3. The boron concentration is small near the end of the fuel cycle, and is difficult to accurately measure. Also, processing the large volume of high purity water needed for the test is costly.

4. Varying the reactor power level disrupts normal reactor plant operations and is costly in terms of reduced generator output.

1.3 MTC Measurement: Stochastic Signal Analysis Using Fast Fourier Transform of Data

The boron-dilution method depends on macroscopic changes in the core. At steady-state, however, there are small stochastic fluctuations that contain information about the system's response to transients. Stochastic signal analysis involves these small microscopic changes in the core parameters. Small fluctuations in the coolant flowrate and inlet coolant temperature can cause small fluctuations in the neutron flux through the MTC. These fluctuations are Fourier transformed into the frequency domain and used to generate an estimator of the true frequency response function (FRF) between the neutron flux (output signal) and the core-exit coolant temperature (input signal).

The $H_1$ estimator of the true FRF is used when noise is on the input signal and is defined as the cross power spectral density function (CPSD) between the neutron flux and the core-exit coolant temperature fluctuation is divided by the power spectral density
function (PSD) of the core-exit coolant temperature fluctuation, or [4,2]

\[ H_1 = \frac{CPSD_{\delta\phi/\phi_0, \delta T_c}(\omega)}{PSD_{\delta T_c}(\omega)}, \]  

(1.4)

where \( \delta\phi / \phi_0 \) is the neutron flux fluctuation normalized to the steady-state neutron flux and \( \delta T_c \) is the coolant temperature fluctuation. Since the thermocouples are inherently noisy, however, the \( H_1 \) estimator is biased.

The \( H_2 \) estimator assumes there is noise on the output and is defined as [4,2]

\[ H_2 = \frac{PSD_{\delta\phi/\phi_0}(\omega)}{CPSD_{\delta T_c, \delta\phi/\phi_0}(\omega)}. \]  

(1.5)

The \( H_2 \) estimator is biased, however, because of noise in the neutron flux signal.

When there is noise in both the input and the output, Mitchell [24] proposed using the \( H_3 \) estimator, which is just the mean of the \( H_1 \) and the \( H_2 \) estimators. The \( H_3 \) estimator, however, does not have a firm theoretical basis in statistics. The true FRF could just as likely be one-third between the \( H_1 \) and the \( H_2 \) estimators or two-thirds between the \( H_1 \) and the \( H_2 \) estimators. In Herr's [2] analysis, it was assumed that half of the error was on the input on half on the output, therefore the \( H_3 \) estimator was used.

The FRF is averaged over the frequency range of highest coherence and then normalized to a known MTC value to predict the MTC for the remaining times during the fuel cycle. The FRF changes with core burnup and is proportional to the MTC [2].

A problem with the Fourier transform analysis is the length of the data sets required. Over three hours of data are required for reliable statistics and signal drift can occur often in that period of time [7].
1.4 MTC Measurement: Stochastic Signal Analysis Using the Multivariate Autoregression Method

The autoregressive (AR) method also relies on the small fluctuations in coolant flowrate and inlet coolant temperature which cause small fluctuations in the neutron flux. Instead of taking three hours of data using frequency-domain methods in the stochastic analysis, the AR method uses only approximately 10 minutes of data using time domain methods [6]. Because the length of the required data sets is greatly reduced, the risk of signal drift of the thermocouples is reduced. The AR method also produces a smoothing effect which makes trends clearer.

In a multivariate autoregression, a time series of \( m \times N \) observations,

\[
X = \{X_1, X_2, \ldots, X_t, \ldots, X_N\} \tag{1.6}
\]

where each \( X_i \) is an \( m \times 1 \) column vector, are recorded [20]. The equally spaced observations of the weakly stationary time series also have zero mean [15]. The autoregression of order \( p \) is then,

\[
X_t = \sum_{i=1}^{p} A_i X_{t-i} + e'_t \tag{1.7}
\]

resulting in \( m \times m \) autoregression equations for the \( A_i \) [7]. Each \( A_i \) is an \( m \times m \) matrix of coefficients and \( e'_t \) is a column vector of the forward autoregression residual. The Nuttall-Strand [12,13] method, a multivariate version of Burg's [11] algorithm, is used to determine the AR parameters.

From the AR coefficients, the power spectral densities are determined from [11,6]

\[
P_{ar} = T [A(f)]^{-1} P' [A(f)]^{-H} \tag{1.8}
\]

where

\[
A(f) = I + \sum_{k=1}^{p} A_k e^{-2\pi f/k} \tag{1.9}
\]
the superscript \(-H\) denotes the Hermitian transpose, and \(j^2 = -1\). Also in Equation (1.8), \(P^f\) is the \(m \times m\) forward prediction matrix of noise variances, and \(T\) is the sampling interval width in seconds [6]. The power spectral densities (PSD) and the cross power spectral densities (CPSD) for the time series can be determined from Equation (1.8), and the time series can be expressed in the frequency domain. The FRF between the core-exit temperature and the in-core neutron flux is then computed from the PSD and the CPSD in the same manner as in the stochastic signal analysis using the Fourier transform method (Sec. 1.3). Finally, the FRF is averaged over the frequency range of highest coherence and normalized to a known MTC value to predict the unknown MTC values [7].

Even though using the AR method produces good results, there are problems due to the calibration method used in the analysis. In the stochastic signal analysis using the Fourier transform method or the AR method, an averaged FRF value is normalized to a known MTC value to predict the unknown MTC values. The known MTC value is determined by using the boron-dilution method. Because of the problems using the boron-dilution method stated in Sec. 1.2, the calibration method used in the stochastic signal analysis is not reliable. A better procedure would be one that does not rely on a calibration method based on the boron-dilution method.

1.5 **MTC Measurement: Neural Networks**

A neural network is a programming technique that mimics the network of biological neurons in the human brain. It consists of various nodes (neurons), the connectivity pattern of these nodes, the response mechanism of the neuron to its input signals, and the learning rule for training such a network [16]. By manipulating the number of nodes,
their connections, and the response mechanism, one can control the behavior of the neural network [16].

Figure 1.1 illustrates a simple two-layer feedforward network. The first layer consists of the input nodes \( x_1 \) and \( x_2 \). The second layer consists of the summation junction and the activation function. The values assigned to the input nodes are multiplied by the weights (constants) \( w_{11}, w_{12}, w_{21}, \) and \( w_{22} \) and summed at the summing junctions. The values from the summing junctions are then fed into the activation function where the final output values are calculated.

After constructing the neural network, it must be "trained" in order to be used as a predicting tool. During the training, the weights, \( w_{ij} \), are determined. A set of inputs and known outputs are used in the neural network, and the weights are changed until the outputs calculated by the neural network are within a certain tolerance of the known outputs. After being trained, the neural network is used to predict the unknown outputs to a new set of input values.

To increase the predicting abilities of the neural network more layers may be added. A neural network with three layers is shown in Fig. 1.2. The extra layer, called a hidden layer, adds more weights to the neural network. This process can be compared to adding more terms to an equation to make a better curve fit. The nodes in the hidden layer, denoted as \( u_1 \) and \( u_2 \) in Fig. 1.2, consist of a summing function and an activation function. The proper selection of the number of hidden layer nodes also improves the performance of the neural network.

Properly "trained" neural networks are beneficial in areas where there are problems in physical or empirical modeling [8]. By using them, parameters can be interpolated or extrapolated for systems that are difficult to model. Because neural networks are
Figure 1.1 Two-layer Feedforward Neural Network (the Input Layer, and the Output Layer)
Figure 1.2  Three-layered Feedforward Neural Network (the Input Layer, One Hidden Layer, and the Outer Layer)
powerful tools in predicting parameters, they have been applied in many areas, including nuclear reactor performance monitoring.

B. R. Upadhyaya and O. Glockler [8] have shown that neural networks can successfully interpolate MTC values at different values of core-burnup. They have also shown that neural networks are poor at extrapolating MTC values well beyond the training domain [8]. Their work with neural networks in predicting MTC values, however, has been limited to using data from a multi-channel model of a PWR core [8], based on design data from the 440 MW(e) Paks PWR in Hungary.

Upadhyaya and Glockler [8] used a three layer neural network (one input layer, one hidden layer, and one output layer) using the FRF of the core-exit thermocouple and the neutron detector at different burnup values as input values to show that the network could interpolate. Twenty-five FRF patterns were used in the analysis. Alternating patterns of the FRF were used to train the neural network, and the remaining FRF patterns were used to interpolate the MTC.

In showing that the neural networks give poor results when extrapolating MTC values, the first group of FRF patterns were used to train the neural network, and the remaining set of FRF patterns were then used to extrapolate the MTC. The further the extrapolating FRF pattern was from the group of training FRF patterns, the worse the results [8].

1.6 Objective of Research

The objective of the work described in this thesis is to test the neural network approach with data from an actual nuclear plant. Data from the Duke McGuire Nuclear
Station is used. By using real data, the effectiveness of using the neural network method of predicting the MTC can be determined.

Two methods were implemented using neural networks. The first method is the method used by Upadhyaya and Glockler [8]. Alternating FRF patterns were used to train the neural network, and the remaining FRF patterns were used to predict the MTC. It will be shown that the method can be used with actual reactor data.

The second method more closely models the use of a neural network in predicting MTC values at different times during a fuel cycle. In this method, all but one of the FRF patterns is used to train the network and the remaining FRF pattern is used to predict the MTC. The FRF pattern used to predict the MTC is systematically chosen until all the FRF patterns are chosen to predict their respective MTC. It will be shown that by using this method the trained neural networks can be better used as predictors of the MTC.

Even though the results of a neural network model are good, the model is still limited by the method used to determine the FRF inputs of the neural network [8]. Errors in the inputs of the neural network limit the accuracy of the neural network model. Therefore in this thesis, different methods of determining the FRF to be used as inputs to the neural network will also be investigated. The first method involves a FFT of the data in order to determine the FRF, and the second method uses the multivariate autoregression method to determine the FRF. The best method is then determined.
Chapter 2

LITERATURE REVIEW

2.1 MTC Estimating Methods Using Noise Analysis

The concept of using the coolant temperature and neutron flux fluctuations to estimate the MTC was first introduced by Thie [17]. Thie proposed using the RMS values of the ex-core neutron flux and the core-exit temperature signals measured in the frequency range $0.1 \leq f \leq 0.5$ Hz to determine the moderator coefficient of reactivity $\alpha_T$ through the following relationship:

$$\frac{RMS[\delta \phi / \phi_0]}{RMS[\delta T_c]} = G_0(f) \alpha_T,$$

(2.1)

where $G_0$ is the local transfer function. This technique to determine the MTC was first applied by Türkcan [18] to the Borssele reactor in the Netherlands. Pór [19] later showed that the method is biased when the coherence between the neutron noise and coolant temperature fluctuations is not near unity [19].

Shieh et. al [9] studied a method of predicting the sign of the MTC by using the low frequency coherence between the core-exit coolant temperature and in-core neutron flux. A model of the Loss-Of-Fluid Test (LOFT) reactor was developed to determine the effect of changing the sign of the MTC on this coherence. The results showed that the CPSD phase extrapolates to -180 degrees for a negative MTC and 0 degrees for a positive MTC as the frequency approaches 0 Hz when the perturbation source is the inlet coolant temperature, coolant velocity, or random heat transfer coefficient fluctuations [9]. The
results from the model were then compared to the data analysis from the LOFT reactor, a commercial PWR [1140 MW(e)], and the Borssele PWR [477 MW(e)] to substantiate the results [9]. The study also concluded that the dominant perturbation causing fluctuations in neutron power and core-exit coolant temperature signals in these reactors is the reactor flow fluctuations [9].

After finding Thie's method [9] to be biased, Aguilar and Pór [20] used a new formula for the estimation of the MTC based on the earlier work by Thie [9] and Pór [19]. By analyzing Unit 2 of Paks Nuclear Power Station, they confirmed that the MTC depends linearly on the boron concentration, and a linear equation relating the MTC and boron concentration was determined. The results also showed that the PSD of the temperature fluctuations remains approximately constant during the fuel cycle, while the neutron flux PSD increases.

Knowing that the PSD of the temperature fluctuations remains constant during a cycle and the neutron flux PSD increases, Herr and Thomas [21] developed a method to estimate both magnitude and sign of the MTC by using the FRF between the in-core neutron flux signal and the core-exit thermocouple signal. These signals were Fourier transformed into the frequency domain and were used to determine the FRF. The FRF was then averaged over the frequency range 0.1 to 0.5 Hz, and normalized to a known MTC value to predict the MTC at succeeding times in the fuel cycle. The method was used with data from Virginia Power's North Anna Power Station and Duke Power's McGuire Nuclear Station. The results of the analysis agreed well with core design and measured MTC values.

Clem and Thomas [6,7] improved the method developed by Herr and Thomas [21] to estimate the MTC. Instead of Fourier transforming the core-exit thermocouple and
neutron flux signals to generate the FRF used in the analysis, the AR method was used to obtain the FRF. The Fourier process required over 3 hours of data, leading to problems of signal drift and equipment malfunction [6]. The AR method requires only about 10 minutes of plant data to perform the analysis [6]. Time series data from the ex-core thermocouple and the in-core neutron detector is modeled by the autoregressive parameters and then the PSD and CPSD are formed. The PSD and the CPSD are used to form the FRF, and the FRF is then averaged over the frequency range of highest coherence. The averaged FRF is again normalized to a known MTC value to estimate the MTC values over the remainder of the fuel cycle. The results of this method compared very well with those from Herr and Thomas [21].

2.2 MTC Estimates Using Neural Networks

Upadhyaya and Glockler [8] introduced the use of neural networks to estimate the MTC. Since the relationship [7, 20, 21] used to estimate the MTC is nonlinear, it was proposed that a neural network could better map the FRF to the MTC [8]. A multi-channel model of a PWR core, using the design data from the 440 MW(e) Paks PWR was used to generate the neutron-coolant temperature CPSD/coolant temperature PSD ratio [8]. The CPSD/PSD ratio was used as input to the neural network and the MTC values made up the output. It was concluded that the neural network could be used for on-line estimation of the MTC provided the training domain covers the domain in which the MTC is to be estimated. It was also shown that the neural network performs poorly when the testing domain deviates far from the training domain. Finally, the neural network gives the best results when the signal from an in-core, as opposed to an ex-core, neutron
detector is used. Since their computer model is a highly idealized representation of a real nuclear reactor, it was suggested in their paper that the method should be tested using actual noise data from a commercial reactor.
Chapter 3

NEURAL NETWORKS

3.1 INTRODUCTION

A feedforward neural network, shown in Fig. 3.1, is made up of three types of layers: the input layer, the hidden layers, and the output layer. The input layer is the first layer of the neural network and consists of input nodes. The input values, $x_1$ and $x_2$, of the system to be modeled are assigned to the input nodes and can be arranged as a column vector, $\mathbf{X}$. The next set of layers are called the hidden layers. The hidden layers also consist of nodes, but the nodes are different from the input nodes. The hidden layer nodes, $u_1$ and $u_2$, are a combination of a summing junction and an activation function. The last layer in a neural network is the output layer and is made up of output nodes. The output nodes are also made up of a summing junction and an activation function. The values assigned to the output nodes are the output values calculated by the neural network and can also be arranged as a column vector, $\mathbf{Y}$.

The nodes of the layers are connected by weight connections. The $w^s_{ij}$ values in Fig. 3.1 are the weights associated with the input originating from the source neuron and are organized in a weight matrix, $[\mathbf{W}_s]$. The subscript $j$ denotes the destination node and the $i$ subscript denotes the source node. The superscript $s$ corresponds to the number of the stage in the neural network. For example, the weight $w^1_{21}$ connects the input node $x_1$ to the hidden node $u_2^1$ in the first hidden layer of the first stage.
Figure 3.1 Multi-stage Feedforward Neural Network
After the values from the previous nodes are multiplied by their respective weights, they are summed at the summing junctions. The summed value is then fed into the activation function block where an output value is calculated. The activation function is a continuous function that is differentiable everywhere and is monotonically increasing [16]. Some examples of activation functions are the sigmoid function [16],

\[ f(x) = \frac{1}{1+e^{-x}}, \]

(3.1)

and the hyperbolic tangent function. The purpose of the activation function is to model the firing of biological neurons. If a biological neuron reaches a threshold energy, the cell fires [16].

The neural network has three main stages of development: the training phase, the testing phase, and the user's phase [16]. The training phase is when the values for the weights are determined. Input data with known outputs are used in the neural network as training data. The configured neural network generates output values that are compared with the known outputs of the system. The weights are adjusted until the output values generated by the neural network are within a certain tolerance of the known outputs, according to a learning rule [16]. After the network is trained, the network is checked to see if it can generalize in the testing phase. Patterns that were not used to train the network are used as inputs to the network and the generated outputs are checked for accuracy. If the network does not generalize well, the network has to be retrained or the structure of the neural network must be changed [16]. Once the network is trained to predict values well, the network can be used in the user's phase on new patterns to predict values.
To better understand the training method of a neural network, a two-layered neural network will be considered first because of its simplicity. Later in this chapter, the more useful multi-layered neural network will be described.

### 3.2 Training a Two-layered Feedforward Network

Figure 3.2 shows a feedforward two-layered neural network. The inputs, $x_1, x_2, x_3, \ldots, x_n$, are connected to the output layer through the weights $w'_{ji}$. In this example, the activation function $f$ in the output layer is assumed to be the identity function, therefore it does not affect the outputs, $y_1, y_2, \ldots, y_k$. Also since there is one stage of weights, the superscript 1 will be dropped to simplify the following equations. The operations performed in the network can be described by the following equations [16]:

$$
y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + \cdots + w_{1n}x_n
$$

$$
y_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + \cdots + w_{2n}x_n
$$

$$
\vdots
$$

$$
y_k = w_{k1}x_1 + w_{k2}x_2 + w_{k3}x_3 + \cdots + w_{kn}x_n
$$

or in matrix form,

$$
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_k
\end{bmatrix} =
\begin{bmatrix}
w_{11} & w_{12} & w_{13} & \cdots & w_{1n} \\
w_{21} & w_{22} & w_{23} & \cdots & w_{2n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
w_{k1} & w_{k2} & w_{k3} & \cdots & w_{kn}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix}
$$

or,

$$
\tilde{Y} = \tilde{W}\tilde{X}
$$

(3.4)

where $\tilde{Y}$ is the output matrix, $\tilde{W}$ is the weight matrix, and $\tilde{X}$ is the input matrix.
Figure 3.2 Two-layer Feedforward Neural Network
Alternatively, the output, $y_k$, can be expressed as a linear function of the input nodes according to [16]

$$y_j^p = \sum_{i=1}^{n} w_{ji} x_i^p$$

(3.5)

where $n$ is the number of input nodes, $w_{ji}$ is the weight connecting input node $i$ to output node $j$, $x_i^p$ is the vector coordinate representing a pattern $\tilde{X}_p$, and $y_j^p$ is the vector coordinate representing an output pattern $\tilde{Y}_p$. The index $p$ connotes the individual patterns.

The weights are determined so that the neural network will generate output values $\tilde{Y}_p$ that match the target values $\bar{T}_p$ when it is fed the input values $\bar{X}_p$ [16]. The weights are determined iteratively starting with a randomly chosen weight matrix. The iterative procedure is governed by a learning rule that is based on minimizing the error $E_p$ [16].

The error is defined as the sum of the squares of the deviations between the target pattern $\bar{T}_p$ and the actual output $\tilde{Y}_p$, or,

$$E_p = \frac{1}{2} \sum_{j=1}^{k} (t_j^p - y_j^p)^2$$

(3.6)

for each of the possible patterns $p$ [16]. The variable $t_j^p$ is the vector coordinate of $\bar{T}_p$, the variable $y_j^p$ is the vector coordinate of $\tilde{Y}_p$, and $k$ is the number of output nodes in the network. Substituting Eq. (3.5) into Eq. (3.6), the following equation is created:

$$E_p = \frac{1}{2} \sum_{j=1}^{k} \left[ t_j^p - \sum_{i=1}^{n} w_{ji} x_i^p \right]^2 .$$

(3.7)

After a pattern is shown to the network, all weights of the network are updated by [16]

$$w_{ji}^{(N+1)} = w_{ji}^{(N)} + \Delta w_{ji}$$

(3.8)
where the superscript \((N+1)\) denotes the \(N+1\) iteration and \(\Delta w_{ji}^\mu\) is given by [16]
\[
\Delta w_{ji}^\mu \propto \frac{\partial E_j^\mu}{\partial w_{ji}^\mu}.
\] (3.9)

To take the proportionality into account in Eq. (3.9), an arbitrarily small coefficient \(\eta\) is used according to
\[
\Delta w_{ji} = -\eta \frac{\partial E_j^\mu}{\partial w_{ji}^\mu}.
\] (3.10)

where the negative sign takes into account the fact that the partial derivatives will be negative in general and \(\eta\) is the learning rate [16]. By applying the chain rule, Eq. (3.10) becomes
\[
\Delta w_{ji} = -\eta \frac{\partial E_j^\mu}{\partial y_j^\mu} \frac{\partial y_j^\mu}{\partial w_{ji}^\mu}.
\] (3.11)

By using Eq. (3.5) and Eq. (3.6) in Eq. (3.11), the following equation results:
\[
\Delta w_{ji} = \eta (t_j^p - y_j^p)x_i^p.
\] (3.12)

This equation is called the linear delta rule, or Widrow-Hoff algorithm [16]. The \(\delta_j^p\), introduced by Widrow is defined as [16]
\[
\delta_j^p \equiv t_j^p - y_j^p.
\] (3.13)

changing Eq. (3.12) to
\[
\Delta w_{ji} = \eta \delta_j^p x_i^p.
\] (3.14)

Finally, the equation for updating the weights becomes
\[
w_{ji}^{(N+1)} = w_{ji}^{(N)} + \eta \delta_j^p x_i^p
\] (3.15)

In practice the weights are updated after all the \(q\) possible training patterns have been shown to the network. After \(q\) patterns have been shown the equation is
\[
w_{ji}^{(N+1)} = w_{ji}^{(N)} + \eta \sum_{p=1}^q (t_j^p - y_j^p)x_i^p,
\] (3.16)
and is called the batch training method [16]. Equation (3.16) can be vectorized in the following manner. The vector $\vec{\delta}_j$ is defined as

$$
\vec{\delta}_j = \begin{bmatrix}
    t_j^1 - y_j^1 \\
    \vdots \\
    t_j^p - y_j^p \\
    \vdots \\
    t_j^q - y_j^q
\end{bmatrix}
$$

(3.17)

where $q$ is the total number of patterns [16]. Similarly, the vector $\vec{x}_i$ is defined as [16]

$$
\vec{x}_i = \begin{bmatrix}
    x_i^1 \\
    \vdots \\
    x_i^p \\
    \vdots \\
    x_i^q
\end{bmatrix}.
$$

(3.18)

Therefore the delta or Widrow rule for multiple patterns can be written for batch training process as

$$
w_{i}^{(N+1)} = w_{i}^{(N)} + \eta \vec{\delta}_j \vec{x}_i,
$$

(3.19)

where the superscript $T$ represents the transpose of the vector [16]. The set of patterns, $q$, are fed into the neural network over and over, and the Delta rule is used to change the weights until convergence occurs.

By modifying the Delta rule, the network can be made to converge faster during the training phase. A momentum term, $\alpha$, is added according to [16]

$$
\Delta w_{i}^{(N+1)} = \eta \vec{\delta}_j \vec{x}_i + \alpha \Delta w_{i}^{(N)}
$$

(3.20)
to speed the changing of the weights. The momentum term reflects the previous changes in the weights and allows the network to bypass a local minimum in the error plane. Increasing $\alpha$ increases the change in the weights and makes the network converge faster.

The choice of the learning rate $\eta$ affects the rate at which the network converges. A large learning rate causes a larger change in the weights and a faster convergence. Increasing $\eta$ too much may cause an oscillatory condition, however. In the oscillatory condition, the error term does not decrease in magnitude and oscillates between error values.

A two-layer neural network with an input layer and an output layer is useful in simulating linear systems. However, many systems are nonlinear in behavior. The neural network can be used to mimic the behavior of a nonlinear system but hidden layers with activation functions must be used. In the next section, the operation of a multi-layer neural network is described.

3.3 Training a Multi-layer Feedforward Network

To correctly model some systems, hidden layers must be added to create a multi-layer neural network. By adding hidden layers, nonlinear activation functions are incorporated into the network, giving the network the ability to behave in a nonlinear manner that better matches the behavior of some systems.

Figure 3.3 shows a three-layer feedforward neural network. It has the input layer, the output layer, and one hidden layer. Remember also that each stage has a weight matrix $[\hat{W}_j]$ associated with it. By using Fig. 3.3, the operations in a multi-layer neural network are described as follows.
Figure 3.3 Three-layer Feedforward Neural Network
The output vector $\bar{Y}$ is obtained by letting the result of the second stage of the network operate on the result of the first stage of the network according to [16]

$$\bar{Y} = \bar{W}_2[\bar{W}_1\bar{X}]$$

(3.21)

By assuming all the activation functions are linear, Eq. (3.21) can reduced to [16]

$$\bar{Y} = \bar{W}_2[\bar{W}_1\bar{X}] = [\bar{W}_2\bar{W}_1]\bar{X} = \bar{W}_3\bar{X}$$

(3.22)

by using the associative property of matrix multiplication. Thus if all stages have linear activation functions, the multi-layered network would reduce to the two-layered network whose weight matrix, $\bar{W}_3$, is a linear combination of the original ones [16]. This leads to the idea that activation functions must be nonlinear in order for them to be useful. In fact, the activation function must meet the following criteria to be used in a neural network [16]:

1. The activation function is a monotonically increasing function, which prevents representations for which the output signal exhibits an extremum for a linear increase in the input.

2. The activation function has only one argument, $\text{net}_j$, which results from a linear combination of the input variables, $x_i$. This prevents the network from discriminating the activity of the individual input nodes.

Two commonly used activation functions are the sigmoid activation function and the hyperbolic tangent function. The output of a neuron $j$ produced by the sigmoid function
is expressed as [16]

\[ y_j = f(n_{et_j}) = \frac{1}{1 + e^{-n_{et_j}}}, \quad (3.23) \]

where

\[ n_{et_j} = \sum w_jx_i. \quad (3.24) \]

The output of a neuron \( j \) produced by the hyperbolic tangent function is described as

\[ y_j = f(n_{et_j}) = \tanh(n_{et_j}). \quad (3.25) \]

To train the weights of the hidden layer the back-propagation algorithm is used. The back-propagation method is a supervised training method that relies on input-output vector pairs called training pairs. The input training vector is applied to the network and the resulting output is compared to the desired target output vector. The weights are changed based on the error between the output calculated by the network and the desired output. The credit for developing the back-propagation method is mostly given to Parker, le Cun, Paul Werbos, or David Rumelhart [16].

The backpropagation algorithm for multistage feedforward neural networks can be formulated as an extension of the Widrow-Hoff delta rule illustrated in section 3.2. The following equations are for a three-layer network, shown in Fig. 3.3 [16]:

\[ E_p = \frac{1}{2} \sum_{j=1}^{k} (t_j - y^p_j)^2, \quad (3.26) \]

for the output layer we have

\[ y^p_j = f(n_{et_j}^p), \quad (3.27) \]

\[ n_{et_j}^p = \sum_{h=1}^{m} w_{jh}y^p_h. \quad (3.28) \]
For the hidden layer we have

\[ y_h^p = f(\text{net}_h^p), \quad (3.29) \]

\[ \text{net}_h^p = \sum_{i=1}^{n} w_{ih} x_i^p. \quad (3.30) \]

The index used for the outer layer nodes is \( j \), while the index used for the hidden layer nodes is \( h \), and the index for the input layer nodes is \( i \). The variable \( p \) denotes the value for a specific pattern. Notice also that there are \( k \) output neurons, \( m \) hidden neurons and \( n \) input nodes.

The weights in the network can be changed by using the following rule:

\[ w^{(N+1)}_{\mu} = w^{(N)}_{\mu} + \Delta w_{\mu} \quad (3.31) \]

The value \( \Delta w_{\mu} \) must be determined for the different stages of weights, however.

The \( \Delta w_{jh} \) of the weights connecting the hidden layer with the output layer can be determined by applying the chain rule according to [16]

\[ \Delta w_{jh}^p = -\eta \frac{\partial E_p}{\partial w_{jh}} \]

\[ = -\eta \frac{\partial E_p}{\partial y_j^p} \frac{\partial y_j^p}{\partial w_{jh}} \]

\[ = -\eta \frac{\partial E_p}{\partial y_j^p} \frac{\partial y_j^p}{\partial \text{net}_j^p} \frac{\partial \text{net}_j^p}{\partial w_{jh}} \]

\[ = +\eta (t_j^p - y_j^p) f''(\text{net}_j^p) y_h^p \]

\[ = \eta \delta_j^p y_h^p. \quad (3.32) \]

We define \( \delta_j^p \) as [16]

\[ \delta_j^p = (t_j^p - y_j^p) f''(\text{net}_j^p). \quad (3.33) \]
The updating of the weights is done after showing all the patterns to the network. For multiple patterns we introduce vectors for the Deltas and the outputs of the hidden node \( h \), so that [16]

\[
\Delta w_{jh} = \eta \delta_j^p \tilde{y}_h \tag{3.34}
\]

The vector notation is illustrated in section 3.2 [Eqs. (3.17) and (3.18)].

A similar method is applied for updating the weights that connect the input layer to the hidden layer. The chain rule is again applied [16] as follows

\[
\Delta w_{hi}^p = -\eta \frac{\partial E_p}{\partial w_{hi}},
\]

\[
= -\eta \frac{\partial}{\partial w_{hi}} \frac{1}{2} \sum_{j=1}^{k} (t_j^p - y_j^p)^2,
\]

\[
= -\frac{1}{2} \eta \left[\sum_{j=1}^{k} \frac{\partial (t_j^p - y_j^p)^2}{\partial y_j^p} \frac{\partial y_j^p}{\partial w_{hi}}\right],
\]

\[
= -\frac{1}{2} \eta \left[\sum_{j=1}^{k} \frac{\partial (t_j^p - y_j^p)^2}{\partial net_j^p} \frac{\partial net_j^p}{\partial w_{hi}}\right],
\]

\[
= -\frac{1}{2} \eta \left[\sum_{j=1}^{k} \frac{\partial (t_j^p - y_j^p)^2}{\partial net_j^p} \frac{\partial f'(net_j^p)}{\partial net_j^p} \frac{\partial \sum_{h=1}^{n} w_{jh} y_h^p}{\partial w_{hi}}\right],
\]

\[
= -\frac{1}{2} \eta \left[\sum_{j=1}^{k} \frac{\partial (t_j^p - y_j^p)^2}{\partial net_j^p} w_{jh} f'(net_j^p) x_i^p\right],
\]

\[
= -\frac{1}{2} \eta \left[\sum_{j=1}^{k} \frac{\partial (t_j^p - y_j^p)^2}{\partial net_j^p} w_{jh} f'(net_j^p) y_h^p\right],
\]

\[
= \eta \left[\sum_{j=1}^{k} (t_j^p - y_j^p) f'(net_j^p) w_{jh}\right] f'(net_j^p) x_i^p.
\]
\[ = \eta \left[ \sum_{j=1}^{k} \delta_j^p w_{jh} \right] f'(net_h^p)x_i^p, \]

\[ = \eta \delta_h^p x_i^p. \quad (3.35) \]

The weights of the network are updated after all the input patterns are applied. This method of updating the weights is called the batch updating process [16]. First the weights connecting the hidden layer and the outer layer are updated according to the following Delta rule [16]

\[ \Delta w_{jh}^p = \eta \delta_j^p y_h^p, \quad (3.36) \]

where the deltas for the output layer can be calculated by

\[ \delta_j^p = (t_j^p - y_j^p)f'(net_j^p). \quad (3.37) \]

We then update the weights of the previous layer (as in our case the input layer) according to [16]

\[ \Delta w_{hi}^p = \eta \delta_h^p x_i^p, \quad (3.38) \]

where the

\[ \delta_h^p = \left[ \sum_{j=1}^{k} \delta_j^p w_{jh} \right] f'(net_h^p). \quad (3.39) \]

It is important to notice that the \( \delta \)'s calculated in the previous step for the outer layer are used here in Eq. (3.39) [16].

To summarize, the \( \delta \)'s of the last layer are updated first, then the \( \delta \)'s of the layer ahead of the last layer are updated based on the \( \delta \)'s from the previous layer [16]. This process is continued until the weights connecting the input layer to the first hidden layer are changed. The weights are changed after all the patterns are shown to the network.

Sometimes it is hard for the backpropagation method to converge, however. To
remedy this problem, a momentum term, \( \alpha \), is added according to [16]

\[
\Delta w^{(N+1)}_{\mu} = \eta \delta_j^T \tilde{x}_i + \alpha \Delta w^{(N)}_{\mu}.
\]

(3.40)

It has been shown that proper combinations of the learning parameter, \( \eta \), and the momentum parameter, \( \alpha \), will train the network. According to Embrechts [22], one should start with \( \eta = 0.1 \) for the learning parameter and \( \alpha = 0.5 \) for the momentum factor. To increase the speed of learning, the learning parameter is increased without causing an oscillatory condition. The momentum factor can then be increased to gain even more speed in training. If an oscillatory condition does occur, the training parameters must be reduced. In training, however, there are no exact rules on determining the values of \( \eta \) and \( \alpha \) [16], and in some cases experimentation is the best method of determining the training parameters.

### 3.4 Bias Nodes

In some systems it is necessary to add a bias node to the network in order for them to behave correctly. A bias node, shown in Fig. 3.4, is a node that is always assigned the value of 1. It is connected to the network through a weight, just as the other nodes. The purpose of the bias node is to add a constant value to the network. The bias node can be thought of as adding a constant to a curve fit.
Figure 3.4 Three-layer Network with Bias Nodes
Chapter 4

NEURAL NETWORK ANALYSIS USING FAST FOURIER TRANSFORM OF DATA

4.1 Introduction

In this chapter, the method of Upadhyaya and Glockler [8] and the proposed method will both be applied to data acquired from the Duke McGuire Unit 2 Nuclear Plant. The data used for this analysis is from Herr's Dissertation [2]. In Herr's work [2], the FRF patterns are calculated by the Fast Fourier Transform (FFT) Method and are used in the present work as inputs to the neural network, while the experimental MTC values determined by Herr [2] are used as target values in the neural network analysis.

4.2 Data Acquisition

The data used in this neural network analysis was taken from the Duke McGuire Nuclear Plant, Unit 2. The data was sampled, recorded on an FM tape recorder, and later analyzed independently by Duke Power as per J.D. Herr's directions [2]. The data was digitized on Duke's PDP-11 system and the spectra were determined [2].

From the spectra, an estimate of the FRF between the neutron flux and the core-exit coolant temperature is determined. A single-input/single-output model of the FRF was formulated because only the input (coolant temperature) and the output (neutron flux) signals are measurable in the reactor [2]. The true FRF can be estimated by the $H_1$ or the $H_2$ estimator.
The $H_1$ estimator is defined as [23,2]

$$H_1 = \frac{\text{CPSD}_{\delta\phi/\phi_o, \delta T_c} (\omega)}{\text{PSD}_{\delta T_c} (\omega)},$$

(4.1)

where $\delta\phi/\phi_o$ is the neutron flux fluctuation normalized to the steady-state neutron flux and $\delta T_c$ is the coolant temperature fluctuation. The $H_1$ estimator is equal to the CPSD between the core-exit temperature and the neutron flux at some point within the core divided by the PSD of the core-exit temperature. By using the $H_1$ estimator it is assumed that there is no noise on the input. In reality, noise does exist in the thermocouple measurements causing the $H_1$ estimator to be biased low. Because of this bias, the $H_1$ estimator is not used.

A plot of the $H_1$ estimates of the FRF for a nearly a year of plant data for McGuire Unit 2 is shown in Fig. 4.1. The magnitudes of the curves do increase with burnup, but many curves cross and they are not separated well. Because of the noise in the thermocouple signals, the $H_1$ FRF estimator gives poor results.

The $H_2$ estimator is defined as [23,2]

$$H_2 = \frac{\text{PSD}_{\delta\phi/\phi_o} (\omega)}{\text{CPSD}_{\delta T_c, \delta\phi/\phi_o} (\omega)}.$$  

(4.2)

The PSD of the neutron flux is divided by the CPSD between the core-exit temperature and the neutron flux at a point within the core. The $H_2$ estimator assumes there is no noise on the output. The neutron flux PSD (the output) should have less noise than the thermocouples, but it is not completely free from noise. Since there is noise on the output, the $H_2$ estimator is biased high, and may not be a good choice for the FRF estimator.
Figure 4.1 $H_1$ Estimator of the FRF Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: Duke McGuire, Unit 2, Cycle 5
Figure 4.2 shows the $H_2$ FRF estimator results at several times during a cycle. The curves are larger in magnitude at each burnup just as those of the $H_1$ estimator, and the curves are separated better than with the $H_1$ estimator. The curves do cross at the higher frequencies, however.

When noise exists in both the input and the output signals, Mitchell [24] proposed using the $H_3$ estimator, defined as the average of the $H_1$ and the $H_2$ estimators. The $H_3$ estimator is does not have a firm theoretical basis in statistics, however. The true FRF is between the $H_1$ and the $H_2$ estimators, but it is unlikely that it is exactly the mean of the $H_1$ and the $H_2$ estimators. It is just as likely that the true FRF is one-third between the $H_1$ and the $H_2$ estimators or two-thirds between the $H_1$ and the $H_2$ estimators. However, it is assumed that half of the noise is on the input and half of the noise is on the output for the data in this analysis, making the $H_3$ estimator the best choice for these data sets. By using the $H_3$ estimator for these data sets, FRF's are separated better than using the $H_1$ or the $H_2$ estimators, and the trends can also be seen better. Therefore, the $H_3$ estimator is used in this analysis.

Figure 4.3 shows the $H_3$ estimator plotted as a function of frequency. A more nearly monotonic increase between FRF curves at successive burnups is evident in the $H_3$ estimator. The FRF's are separated better than the two other estimators, because the amount of cross-over of the curves is much less.

4.3 Frequency Range Selection

For each individual FRF, $H_3$ estimator values at each frequency are used as input to the neural network. For the neural network model, only data above 0.1 Hz is used, because any data below 0.1 Hz may also contain information about the fuel temperature.
Figure 4.2  $H_2$ Estimator of the FRF Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: Duke McGuire, Unit 2, Cycle 5
Figure 4.3 \( H_3 \) Estimator of the FRF Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: Duke McGurre, Unit 2, Cycle 5
coefficient [9]. In addition to this restriction, only FRF values with the highest coherence are used. Therefore a more accurate neural network model can be created.

The coherence, shown in Fig. 4.4, is highest from 0.1 Hz to 0.5 Hz. Since the coherence is highest in this frequency range, only the $H_3$ estimator values in the frequency range from 0.1 to 0.5 Hz are used.

### 4.4 Neural Network Program

The neural network program used in this analysis of the Fourier transformed data was METANEURAL [22]. The program was originally conceived and developed at Rensselaer Polytechnic Institute in Troy, New York [22]. The present version of the program was completed by Jeffrey Sax, Bert Robben, Marc Pollefeyts, and Peter Puellinckx at the Catholic University of Leuven, Belgium [22]. Mark J. Embrechts of Troy, New York is another author of the program [22].

METANEURAL is a DOS shareware program that is easy to use on any IBM PC or compatible [22]. It is highly recommended that a PC with a fast processor (286 processor or higher) and a math coprocessor be used with METANEURAL [22]. The program is more enjoyable to use with a fast processor. The program also requires 512k of available memory [22]. A separate text editor such as KEDIT is also needed to create the training pattern files and the testing (prediction) pattern files.

A PC compatible with a 486 processor running at 33 MHz was used to do the analysis. The 486 processor contains an i487 math coprocessor built onto the chip. Using METANEURAL [22] took between 15 seconds and 15 minutes to train the networks, depending on the amount of reduction in the training error required. More time
Figure 4.4  Coherence Between the In-core Neutron Flux and the Core-exit Coolant Temperature [FFT Processed Data]: Duke McGuire, Unit 2, Cycle 5
is required to reduce the training error to a smaller value. It cannot be stressed enough that a fast computer is required when using METANEURAL [22].

METANEURAL [22] contains two main programs: META.EXE and NETMAKER.EXE. NETMAKER.EXE is used to create a text file that contains the information about the neural network such as the number of layers, the number of nodes in each layer, the momentum parameter (α), the learning parameter (η), and the activation function. The program META.EXE is used to train the neural network, and to predict values with the trained neural network.

The information in the text file made with NETMAKER.EXE is used to set up the neural network that META.EXE trains. META.EXE reads the training patterns into the network from a text file created with a text editor. After the network is trained to a specific training error value, the program saves the weights in a text file.

Once the network is trained, it is used to predict MTC values for patterns not used in the training file. NETMAKER.EXE is again used to create a text file that contains information for the predicting neural network. The predicting network has exactly the same components as the network used in training, i.e. the number of layers, the number of nodes in each layer, etc. The predicting network is not used to train the network further, but is used only to predict values. META.EXE uses the information in a text file on the prediction network to create the predicting network. It then uses the weights determined in the training phase to predict the MTC for the new patterns in the predicting text file. The text file containing the patterns used in predicting values is also created with a text editor.
4.5 Analysis Using the Method of Upadhyaya and Glockler

In Upadhyaya and Glockler's approach [8], alternating patterns are used to train the neural network and the remaining patterns are used for predicting their respective MTC values.

A three-layer neural network with one input layer, one hidden layer, and one output layer was used in the present analysis. The input layer contains 10 input nodes corresponding to the 10 $H_3$ estimator values at distinct frequencies between 0.1 and 0.5 Hz. The hidden layer contains 15 nodes, while the output layer contains one node that corresponds to the MTC value.

The network was trained using the FRF's as input patterns and their respective experimentally determined MTC values. Because the backpropagation algorithm requires that the training data be between 0 and 1, the FRF and the MTC values were normalized. The FRF and the MTC values were adjusted to be between 0.1 and 0.9 for consistency with Upadhyaya and Glockler [8].

The sigmoid function was used as the activation function in the neural network. When using the sigmoid function the training values should be between 0 and 1, because the output of the sigmoid function is from 0 to 1. If the training data is from -1 to +1, the hyperbolic tangent function should be used, because its value lies in the range -1 to +1.

After training the neural network, it is ready to be used to predict values. The method proposed by Upadhyaya and Glockler [8] consists of training the network on alternating patterns and using the remaining patterns for predicting. The network is optimized when the square root of the sum of the squared errors (the difference between the network calculated value and the target value) for the predicting patterns is minimized.
Figure 4.5 shows that as the amount of training increases the training error decreases. The training error is defined as the square root of the sum of the squares of the errors for all the training patterns [22], or

$$\text{training error} = \sqrt{\sum_{p=1}^{q} \left\{ \sum_{j=1}^{k} (t_j^p - y_j^p)^2 \right\}} \quad (4.3)$$

where $j$ is the index of the output layer nodes, $t$ is the target value of the training patterns, $y$ is the value from the output node, and $p$ denotes the specific pattern being applied to the network for training. There are $q$ training patterns and $k$ output layer nodes.

When testing the neural network, the prediction error decreases as the training error decreases until it reaches a minimum, as shown in Fig. 4.6. The prediction error is defined as

$$\text{prediction error} = \sqrt{\sum_{p=1}^{r} \left\{ \sum_{j=1}^{k} \frac{(t_j^p - y_j^p)^2}{(t_j^p)^2} \right\}} \quad (4.4)$$

where $j$ is the index of the output layer nodes, $t$ is the target value of the testing patterns, $y$ is the value from the output node, and $p$ denotes the specific testing pattern being applied to the network for training. There are $r$ testing patterns and $k$ output layer nodes. After the minimum is reached, the predicting error increases as the training error is decreased. After a certain amount of training, the network will begin to "memorize" the results and its predicting performance will decrease. This phenomena is similar to a student that memorizes the answers to questions without understanding the material. The student can answer all the questions that he memorized, but incorrectly answers questions that are slightly different than those memorized. Similarly, the predicting performance of the
Figure 4.5  Training Error vs. Amount of Training (Neural Network Using Fourier Transformed Data from Duke McGuire, Unit 2, Cycle 5)
Figure 4.6  Prediction Error vs. Training Error (Neural Network Using Fourier Transformed Data from Duke McGuire, Unit 2, Cycle 5)
neural network will suffer if it is over-trained. When the predicting error reaches a minimum, the network is considered to be properly trained.

The results of this analysis are shown in Fig. 4.7. The plot contains MTC values as a function of burnup. The burnup is in units of megawatt days per metric tonne uranium (MWD/MTU). In the plot, the core design MTC curve is a straight line because only two core-design MTC values were available to create it [2,7]. The MTC values used to train the network were experimentally determined by Herr [2]. It can be seen that the neural network model does predict the MTC values well when using this training method when the FRF patterns used for predictions are within the training domain. The MTC value at 11669 MWD/MTU predicted by the neural network is worse than the other predicted values because the neural network is extrapolating this value. In other words, the FRF pattern used to predict the MTC at 11669 MWD/MTU is outside the training domain of the neural network. By doing this analysis, it shows that neural networks using actual plant data are useful to interpolate MTC values and that Upadhyaya's Method can be used effectively with actual reactor data.

4.6 Use of Neural Networks to Predict Values (Proposed Method)

Even though the method used by Upadhyaya and Glockler [8] produces good results, a new method is proposed to better model the use of a neural network in predicting MTC values at different times during a fuel cycle. In a real-life scenario, a nuclear plant will need to determine the MTC at a time during a fuel cycle when the MTC is unknown.

Upadhyaya and Glockler's method determines the MTC using alternate FRF patterns to predict the MTC at the remaining times in the fuel cycle. This method does not model
Figure 4.7  Neural Network Results Using Upadhyaya and Glockler's Method for Duke McGuire, Unit 2, Cycle 5 [FFT Processed Data]
the actual use of the neural network in a nuclear plant. In reality, all the FRF patterns should be used to train the neural network to predict the MTC at a time during the fuel cycle when the MTC is not known. Ideally, many patterns with known MTC values would be available from past fuel cycles to use in training the network.

In the proposed method all but one of the patterns are used to train the network, and the MTC is then predicted for the pattern not used in training the network. A new pattern is systematically chosen to be left out of the training of the network until the MTC has been predicted for each of the patterns.

Eight neural networks were used in this analysis. All of them were constructed the same with 10 input layer nodes, 15 hidden layer nodes, and 1 output layer node. The sigmoid function was again used as the activation function, because the values used in the network are in the interval (0.1, 0.9).

Each neural network was trained with all but one of the patterns and was used to predict the MTC for the FRF pattern that was not included in the training patterns. Each of the eight networks was trained to various training errors and was then used to predict its respective MTC value. Each of the networks was trained to the same training error values, also. For example, at a training error of 0.008944 the MTC values in Table 4.1 were predicted for the Duke McGuire Plant. The prediction error in the proposed method is defined as the square root of the sum of the squares of the errors for all the predicting patterns for each of the eight neural networks, or

\[
\text{prediction error}_{\text{proposed method}} = \sqrt{\frac{\sum_{i=1}^{t} (t_i - y_i)^2}{(t_i)^2}} \tag{4.5}
\]
**Table 4.1** Predicted MTC Values at a Training Error of 0.008944 for the Duke McGuire Plant [FFT Processed Data]

<table>
<thead>
<tr>
<th>Pattern (Burnup)</th>
<th>Predicted MTC</th>
<th>Actual Value</th>
<th>Normalized Square of the Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRF 1 (759 MWD/MTU)</td>
<td>-8.1282</td>
<td>-8.42</td>
<td>0.001201</td>
</tr>
<tr>
<td>FRF 2 (1753 MWD/MTU)</td>
<td>-11.0732</td>
<td>-10.93</td>
<td>0.000172</td>
</tr>
<tr>
<td>FRF 3 (2988 MWD/MTU)</td>
<td>-12.7999</td>
<td>-12.4</td>
<td>0.00104</td>
</tr>
<tr>
<td>FRF 4 (5610 MWD/MTU)</td>
<td>-16.616</td>
<td>-16.88</td>
<td>0.000245</td>
</tr>
<tr>
<td>FRF 5 (7020 MWD/MTU)</td>
<td>-18.5566</td>
<td>-18.89</td>
<td>0.000312</td>
</tr>
<tr>
<td>FRF 6 (7779 MWD/MTU)</td>
<td>-19.7284</td>
<td>-19.74</td>
<td>$3.45 \times 10^{-7}$</td>
</tr>
<tr>
<td>FRF 7 (9747 MWD/MTU)</td>
<td>-24.4931</td>
<td>-23.45</td>
<td>0.001979</td>
</tr>
<tr>
<td>FRF 8 (11669 MWD/MTU)</td>
<td>-25.7393</td>
<td>-27.13</td>
<td>0.002628</td>
</tr>
</tbody>
</table>

Prediction Error = 0.087037
where $s$ is the index of the number of neural networks, $t$ is the target value, $y$ is the value from the output node, and $z$ is the total number of neural networks.

By plotting the prediction error as a function of training error, a minimum can be seen as in Fig. 4.8. At the minimum prediction error, the neural network is considered to be properly trained.

The results are shown in Fig. 4.9. The predicted MTC results are plotted along with core design values for the Duke McGuire reactor. There are only two core design points, however, and a straight line is drawn through them to create the design curve [2,7]. The experimentally determined MTC values are also plotted. It can be seen that the predicted values are close to the experimental values. By using the proposed method, the MTC values can be predicted consistently.
Figure 4.8  Prediction Error vs. Training Error for the Proposed Method (Neural Network Using Fourier Transformed Data from Duke McGuire, Unit 2, Cycle 5)
Figure 4.9  Neural Network Results Using Proposed Method for Duke McGuire, Unit 2, Cycle 5 [FFT Processed Data]
Chapter 5

NEURAL NETWORK ANALYSIS USING MULTIVARIATE AUTOREGRESSION METHOD

5.1 Introduction

In this chapter, the FRF patterns used as inputs to the neural network are computed by the AR method [7] instead of using the FFT method as in Herr's Dissertation [2]. The data was acquired from the Duke McGuire Unit 2 Nuclear Plant and is from the thesis of A.W. Clem [7]. The FRF patterns calculated by the AR Method are used as inputs to the neural network, while the experimental MTC values determined by Clem [7] are used as the target values in the neural network analysis. In this analysis, the method of Upadhyaya and Glockler [8] and the proposed method will both be applied to the data.

5.2 Data Acquisition

The data in this neural network analysis was taken from the Duke McGuire, Unit 2 Nuclear Station. Approximately ten minutes of data were copied from the Duke analog tape at each of the eight data points during the fuel cycle [7]. Data for an ex-core thermocouple and in-core neutron detector in the same fuel assembly was available for an entire fuel cycle [7].

55
The data from the dubbed analog tape was transferred to digital format using the data collection system developed by Herr [2] and consisted of an IBM PC equipped with an analog-to-digital converter with a triggering circuit for synchronous sampling of the data channels [7]. An anti-aliasing filter with a 15 Hz low-pass capability is used prior to digitization [7]. The data was sampled at 45 samples per second and then simultaneously written to disk so that contiguous time series are available [7]. More details on the data acquisition can be found in the work of Clem [7] and Herr [2].

Once the time series data was obtained, the FRF's can be determined by the AR method. Before analyzing the data, however, the fluctuating neutron flux signal, $\delta \phi$, is normalized to the steady DC flux voltage to make the resulting neutron signal representative of $\delta \phi / \phi_0$ and the thermocouple voltage is converted to degrees F [7].

### 5.3 Data Analysis

The AR method was used to calculate the FRF's from the time series data. In the AR method, the recorded time series of $m \times N$ observations,

$$X = \{X_1, X_2, \ldots, X_t, \ldots, X_N\}$$  \hspace{1cm} (5.1)

where each $X_t$ is an $m \times 1$ column vector, can be modeled as an autoregression of order $p$,

$$X_t = \sum_{t-1}^{p} A_i X_{t-i} + e'_t,$$  \hspace{1cm} (5.2)

resulting in $m \times m$ autoregression equations for each of the $m \times 1$ vectors, $X_t$ [7]. Each $A_i$ is an $m \times m$ matrix of coefficients and $e'_t$ is a column vector of the forward autoregression residual.
The Nuttal-Strand [12,13] method, a multivariate version of Burg's [11] algorithm, was used to determine the AR parameters, $A_r$ [7].

From the AR coefficients, the power spectral densities are determined from [11,6]
\[
P_{ar} = T[A(f)]^{-1}P' [A(f)]^{-H}, \tag{5.3}
\]
where
\[
A(f) = 1 + \sum_{k=1}^{p} A_k e^{-2\pi j f k T},
\]
the superscript $-H$ denotes the Hermitian transpose, and $j^2 = -1$. Also in Eq. (5.3), $P'$ is the $m \times m$ forward prediction matrix of noise variances, and $T$ is the sampling interval width in seconds [6]. The power spectral densities (PSD) and the cross power spectral densities (CPSD) for the time series can be determined from Equation (5.3), and the time series is thus transformed to the frequency domain.

A computer program was developed by Clem [7], using subroutines from the text by Marple [11], to solve for the AR coefficients, $A_r$, and calculate the PSD's. The ten minutes of plant data was broken up into 25 23-second (1024 pts) data sets. The AR coefficients were then averaged to calculate the PSD's [7]. For more information on the algorithm, including a listing of the computer program, see Clem's thesis [7].

The model order, $p$, used in the analysis was determined by the Akaike Information Criterion (AIC) [25, 7]. For a multi-channel AR model the form of the AIC is [11]:
\[
AIC_p = N \ln(\text{det } P') + 2m^2 p, \tag{5.4}
\]
where $N$ is the number of points, $P'$ is the $m \times m$ forward prediction matrix of noise variances, $m$ is the number of data channels, and $p$ is the order of the autoregression [7].
The model order that gives the smallest AIC value is the best model order [7]. It was found that a model order of 30 was the best for this set of data [7].

The FRF between the core-exit temperature and the neutron flux is computed from the PSD and the CPSD in the same manner as in the Fourier transform method (Chapter 4).

The $H_1$ estimator of the true FRF is plotted in Fig. 5.1 and it can be seen that the curves are smoother than those generated by the FFT method. The curves do cross at some frequency points, however. Since there is noise in the thermocouple signals, the $H_1$ estimator is a biased estimate as discussed in Chapter 4.

The $H_2$ estimator of the true FRF is plotted in Fig. 5.2. The amount of crossing of the FRF curves is less than that for the $H_1$ estimator, and the trends are clearer. As the burnup increases, the magnitudes of the FRF curves increase more or less uniformly. Even though there is less noise in the neutron flux signal than in the thermocouple signal, there is still noise in the neutron flux signal, making the $H_2$ estimator biased.

As mentioned before in Chapter 4, for the case when noise is present in both the input and the output, Mitchell [24] proposed using the $H_3$ estimator, the mean between the $H_1$ and $H_2$ estimators. It should be noted again that the $H_3$ estimator is not statistically rigorous, since it is unlikely that the true FRF is exactly the mean of the $H_1$ and $H_2$ estimators. For this set of data it is assumed that half of the noise is in the thermocouple signal and half of the noise is in the neutron detector signal. Therefore, the $H_3$ estimator is used in this analysis.

Figure 5.3 contains the plots of the $H_3$ FRF estimator. The curves are separated better than those of the $H_1$ and $H_2$ estimators, and the amount of crossings in the $H_3$ estimator curves is smaller than the other two estimators.
Figure 5.1  $H_1$ FRF Estimator Results for Duke McGuire, Unit 2, Cycle 5 for AR Model 30
Figure 5.2 \( H_2 \) FRF Estimator Results for Duke McGuire, Unit 2, Cycle 5 for AR Model 30
Figure 5.3  $H_3$ FRF Estimator Results for Duke McGuire, Unit 2, Cycle 5 for AR Model 30
5.4 Frequency Range Selection

$H_3$ estimator values at each frequency are used as inputs in the neural network for each of the FRF's. $H_3$ estimator values between 0.1 and 0.5 Hz are used because the coherence between the core-exit thermocouple and in-core neutron flux detector is highest in this range, as shown in Fig. 5.4. It should also be noted that any data below 0.1 Hz may contain information about the fuel temperature coefficient, and not exclusive information about the moderator temperature coefficient [9]. By choosing this frequency range, the neural network better models the relationship between the FRF and its respective MTC value, making it a better predictor of MTC values.

5.5 Neural Network Program

The neural network program used in this analysis of the AR processed data was METANEURAL [22] as described in Chapter 4. METANEURAL is a DOS shareware program that runs on an IBM PC or compatible with 512k available memory [22]. A PC with at least a 286 processor or higher with a math coprocessor be used with METANEURAL [22]. A separate text editor such as KEDIT was also used to create the training pattern files and the testing (prediction) files.

A PC compatible with a 486 processor running at 33 MHz was used in this analysis. The 486 processor contained an i487 math coprocessor on the same chip. Using this computer, it took between 15 seconds to 20 minutes to train the network, depending on the amount the training error was to be reduced. More training time was required to reduce the training error more.
Figure 5.4  Coherence Between Core-exit Thermocouple and In-core Neutron Flux Detector for Duke McGuire, Unit 2, Cycle 5 for AR Model 30
Once the network was trained, METANEURAL was used to predict the MTC for patterns not used in the training pattern. The neural network predictions were almost instantaneous on the computer.

5.6 Analysis Using the Approach of Upadhyaya and Glockler

In this approach, alternating FRF patterns are used to train the neural network and the remaining patterns are used for predicting their respective MTC values. The experimentally determined MTC values from Clem's work [7] were used as the target values used in training the network.

A three-layer neural network was used. The input layer consisted of nine nodes corresponding to the nine frequencies components from 0 to 0.5 Hz in each of the FRF curves. The hidden layer contain 15 nodes and the output layer contained one node corresponding to the respective MTC value for the specific FRF curve.

The FRF input values and their respective MTC output values were mapped to the interval (0.1, 0.9). This normalization speeds the training phase as previously discussed.

The sigmoid activation function was used because the input and output values used in the network are in the interval (0, 1). After constructing the network, it was trained until the prediction error, defined by Equation (4.4) in Chapter 4, was minimized. Figure 5.5 shows that as the amount of training increases, the training error, defined by Equation (4.3) in Chapter 4, decreases. In Fig. 5.6, the prediction error decreases until the training error reaches 0.0. At this point further training cannot reduce the prediction error any more, and the network is considered optimally trained.
Figure 5.5  Training Error vs. Amount of Training (Neural Network Using AR Processed Data from Duke McGuire, Unit 2, Cycle 5)
Figure 5.6  Prediction Error vs. Training Error (Neural Network Using AR Processed Data from Duke McGuire, Unit 2, Cycle 5)
Figure 5.7 shows the results of the neural network analysis using the method of Upadhyaya and Gockler [8]. The predictions of the network are plotted along with the training points and a line corresponding to the core design MTC values. Since there are only two core design MTC values, a straight line is drawn through them as a best estimate. The experimental points predicted using the AR Method are also plotted. It can be seen again that the neural network predicts the first three MTC values well, because the FRF patterns are within the training domain of the neural network. The MTC value predicted at 11669 MWD/MTU is worse than the other predictions because the network is extrapolating this value. It should be noted, however, that the prediction of the MTC at 11669 MWD/MTU is better using the AR processed data than the Fourier transformed data in Chapter 4. These results show that the method of Upadhyaya and Gockler [8] can be applied to data analyzed using the AR method.

5.7 Use of Neural Networks to Predict Values (Proposed Method)

The proposed method is again used to better model the actual use of a neural network to determine the MTC at some time in the fuel cycle of a nuclear reactor. Instead of using alternating patterns to train the network, the proposed method uses all the patterns except the one that is used to predict the unknown MTC. Ideally, FRF data and their corresponding known MTC values for several fuel cycles will be used to determine the unknown MTC at some time during the fuel cycle. To determine the unknown MTC, one would use all the previous FRF patterns and known MTC values to train the network, not just alternating FRF patterns and known MTC values.
Figure 5.7 Neural Network Results Using Upadhyaya and Glockler’s Method for Duke McGuire, Unit 2, Cycle 5 [AR Processed Data]
In this analysis, eight neural networks were used to predict the eight MTC values at different times in the fuel cycle. Each of the networks were constructed the same with 9 input nodes, 15 hidden layer nodes, and one output node. The 9 input nodes corresponded to the 9 frequency values between 0.1 and 0.5 Hz of the $H_3$ estimator curves. The output of the node in the output layer corresponded to the predicted MTC value by the network.

In each neural network, all the $H_3$ estimator curves except the one that was used to predict its corresponding MTC was used to train the neural network and then used to predict the MTC at that training error value. The networks were all trained to the same training error and contained all the same components: the number of inputs, the number of hidden layers, the number of output layers, and the activation function. The only thing different in the networks was the $H_3$ estimator curves used to train the network.

At each of the training errors, the neural networks were used to predict its respective MTC. The prediction error was then determined by using Eq. (4.4). In Table 5.1 contains the MTC values and the prediction error at a training error of 0.008944.

As the training error is reduced, the prediction error is reduced also until a point where the prediction error is minimized, as seen in Fig. 5.8. At the minimum prediction error, the network is considered optimally trained. After the minimum prediction error is reached, the neural network's predicting abilities become worse, because it is memorizing the patterns as discussed in Chapter 4.

The results are shown in Fig. 5.9. The figure contains the core design MTC values plotted as a straight line because there are only two core design MTC values. The experimental values determined by Clem [7] using the AR method are also included as well as the MTC values predicted by the trained neural network. The predicted MTC
Table 5.1  Predicted MTC Values at a Training Error of 0.008944 for the Duke McGuire Plant [AR Processed Data]

<table>
<thead>
<tr>
<th>Pattern (Burnup)</th>
<th>Predicted MTC</th>
<th>Actual Value</th>
<th>Normalized Square of the Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRF 1 (759 MWD/MTU)</td>
<td>-8.224</td>
<td>-8.92</td>
<td>0.006088</td>
</tr>
<tr>
<td>FRF 2 (1753 MWD/MTU)</td>
<td>-10.7392</td>
<td>-10.75</td>
<td>1.01×10^{-6}</td>
</tr>
<tr>
<td>FRF 3 (2988 MWD/MTU)</td>
<td>-12.4032</td>
<td>-12.4</td>
<td>6.66×10^{-8}</td>
</tr>
<tr>
<td>FRF 4 (5610 MWD/MTU)</td>
<td>-16.8352</td>
<td>-16.29</td>
<td>0.00112</td>
</tr>
<tr>
<td>FRF 5 (7020 MWD/MTU)</td>
<td>-18.7712</td>
<td>-18.77</td>
<td>4.09×10^{-9}</td>
</tr>
<tr>
<td>FRF 6 (7779 MWD/MTU)</td>
<td>-21.2448</td>
<td>-20.87</td>
<td>0.000323</td>
</tr>
<tr>
<td>FRF 7 (9747 MWD/MTU)</td>
<td>-24.5568</td>
<td>-24.59</td>
<td>1.82×10^{-6}</td>
</tr>
<tr>
<td>FRF 8 (11669 MWD/MTU)</td>
<td>-26.6912</td>
<td>-28.31</td>
<td>0.00327</td>
</tr>
</tbody>
</table>

Prediction Error = 0.10394
Figure 5.8 Prediction Error vs. Training Error for the Proposed Method (Neural Network Using AR Processed Data from Duke McGuire, Unit 2, Cycle 5)
Figure 5.9  Neural Network Results Using Proposed Method for Duke McGuire, Unit 2, Cycle 5 [AR Processed Data]
values are very close to the experimental values and shows that this method is effective at predicting MTC values at various times in the fuel cycle. It should be noted that the results of this analysis are better than the similar analysis using the Fourier transformed data. The MTC value at 11669 MWD/MTU is still the worse predicted value, because the network is extrapolating this value. The predicted MTC value is still better than the MTC predicted by the network using Fourier transformed data. Ideally, data from several fuel cycles will be used with this method, allowing the network to interpolate the data instead of extrapolating the MTC value.
Chapter 6

CONCLUSIONS, LIMITATIONS, & RECOMMENDATIONS

6.1 Conclusions

In this work, neural networks were implemented to predict the MTC at different times during a fuel cycle in a pressurized water reactor. Actual data from the Duke McGuire Nuclear Power Station was implemented to show that neural networks can be used to predict MTC values in an actual nuclear reactor.

Upadhyaya and Glockler's method [8] was first implemented to demonstrate that this method can be used with actual data from a reactor. According to the method, alternating FRF patterns were used to train the network, and the remaining FRF patterns were used to predict the respective MTC values. The $H_3$ estimator of the FRF was used as inputs to the neural network while the neural network predicted the corresponding MTC value. Because noise existed in both the input (thermocouple signal) and the output (neutron flux signal), the $H_3$ estimator of the true FRF, the mean of the $H_1$ and the $H_2$ estimators, was used. It should be noted that it is just as likely that the true FRF is one-third or two-thirds between the $H_1$ and $H_2$ estimators, and not the mean of the $H_1$ and the $H_2$ estimators. By using the $H_3$ estimator it is assumed that half of the noise is in the input signal and half of the noise is in the output signal. Also in training the network, experimentally determined MTC values were used as the target values. The trained network was shown to predict MTC values well as long as the predicted MTC values
were within the training domain. Predicting MTC values outside the training domain were less successful.

A second method (the proposed method) was then used to better model the actual use of a neural network to predict a MTC value during a fuel cycle. Instead of using alternating FRF patterns to train the network, all of the FRF patterns, except the pattern corresponding to the unknown MTC value, were used to train the network. Again, the $H_3$ estimator of the true FRF was used as the input to the neural network for the aforementioned reasons, and experimentally determined MTC values were used as the target values when training the neural network. The optimally trained network was shown to predict MTC values close to the experimentally determined values. By using this method, neural networks were shown to effectively determine MTC values at different times during the fuel cycle. It is especially effective using AR processed data.

It was also shown that the method by which the FRF curves were determined affected the accuracy of the neural network. The FRF curves calculated by the AR method were smoother and showed trends better than the FRF curves determined by the Fourier transform method. Since the FRF curves are smoother using the AR method, the neural network model predicts MTC values better using FRF curves calculated by the AR method.

Another benefit of using the AR method of producing the FRF curves instead of the Fourier transform method is that the AR method requires less data. The AR method requires only 5% of the data that the FFT method requires and produces equal, if not better, results [7].
6.2 Limitations

The use of neural networks are advantageous in many ways, but they also have limitations. One limitation is due to the method of training the network. In training, experimental MTC values are used as target values. Errors in the experimentally determined target values used in training cause errors in the neural network model, which in turn, cause errors in the predictions of the trained neural network.

Another problem is the lack of data. Since data for only one fuel cycle was used in these tests, the neural network had to be used to extrapolate the MTC value for the last burnup value. Ideally there would be data for more than one fuel cycle to be used in training the neural network, allowing the network to predict MTC values within the training domain. As long as the network predicts MTC values within the training domain, the results are satisfactory.

Limitations also occur in the methods of determining the FRF curves used as inputs to the neural network. The Fourier transform method requires a long record length to reduce the random error of the spectral estimates [2]. During this long recording time, signal drift becomes a problem. Signal drift in the thermocouple signal causes signal saturation, causing losses in the data [2].

The AR method of calculating the FRF curves improves on the Fourier transform method, but it also has limitations. Because the AR method requires much less data, it was assumed that the problem of signal drift during the acquisition of data would be reduced. Even though the AR method does use much less data, signal drift still exists in the data. The AR method assumes that the data is weakly stationary, which means that the data has to have a constant mean and variance. It was shown by Clem [7] that the time series data from the Duke McGuire Nuclear Power Plant was not weakly stationary,
however. By looking at the mean of the thermocouple and neutron flux signal, a low-frequency signal drift was detected in the data that reduced the statistical confidence that the 25 data sets were from the same process [7]. The variance of the signals looked satisfactory [7]. Because both the AR and FFT methods have used the same data set length, the apparent nonstationarity is a problem with both techniques [7]. Clem [7] hypothesized that the low-frequency signal drift causing the nonstationarity in the data set mean does not affect the signal fluctuations in the frequency range of interest (0.1 to 0.5 Hz), thus making the data stationary in this frequency range and making the analysis valid. Clem [7] did not prove his hypothesis, however.

6.3 Recommendations

The results of using neural networks to determine the MTC at various times in the fuel cycle are promising. To improve the performance of the neural networks, more data is required, however. It is recommended that data from an actual reactor be taken for several fuel cycles and used in the neural network. By establishing a data base of FRF patterns and their corresponding known MTC values for several fuel cycles, the neural network will not have to extrapolate MTC values and would be more effective at determining MTC values.

Research could also be done to determine if the data in the frequency range of interest (0.1 to 0.5 Hz) is stationary when using the AR method, making the method a more valid way to create the FRF curves. Ultimately, a method of eliminating the low-frequency signal drift when taking data needs to be established since both the Fourier transform and the AR method suffer from this problem.
REFERENCES


VITA

The author was born on December 18, 1967 in Lynchburg, Virginia. After graduating in 1986 from Gretna Senior High School in Gretna, Virginia, he continued his education at Virginia Polytechnic Institute in engineering. While working on his B.S. in mechanical engineering, he worked at B&W Nuclear Service Company in Lynchburg, Virginia as a Co-op student. He graduated in 1991 with a B.S. in mechanical engineering and decided to further his education by going to graduate school at VPI. He received his M.S. degree in May, 1993.

Joseph J. Adams