1994

Application of neural networks and mathematical optimization techniques for NOx control

Preston Charles Lee
Lehigh University

AUTHOR:

Lee, Preston Charles

TITLE:

Application of Neural Networks and Mathematical Optimization Techniques of NOx Control

DATE: October 9, 1994

Application of Neural Networks and Mathematical Optimization Techniques for NOx Control

by

Preston Charles Lee

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Mechanical Engineering

Lehigh University

1994

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

Date: August 19, 1994

Thesis Advisor: ____

Dr. Edward K. Levy

Chairman of Department:

Dr. Robert P. Wei

ACKNOWLEDGMENTS

My sincerest appreciation and gratitude to my family for providing love, support, and understanding throughout this, and every other, endeavor I have undertaken. I owe a great deal of thanks to Drs. Edward K. Levy and Laura I. Burke for their guidance on this project, along with Debbie Eskenazi, Mark D'Agostini, and John Salmento for their time and technical expertise.

The work in this thesis was made possible as part of a continuing series of projects funded by the Potomac Electric Power Company and the Electric Power Research Institute. Their assistance in facilitating my graduate education is very much appreciated.

Finally, thanks to all of my friends who offered encouragement along the way and to Jimmy Buffett, whose music helped me keep things in perspective.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

Page

Page

ABSTRACT

The Potomac River Generating Station is mandated by state and federal regulations to reduce NOx emissions to 0.38 lb./MBtu, at all power output levels. This thesis addresses optimizing the unit heat rate of a pulverized coal boiler while maintaining this NOx emissions limit. The modeling of the boiler was achieved through neural networks, a form of artificial intelligence, while the optimization was accomplished through mathematical optimization algorithms. This process was applied to boiler operation at full load (108 MW) and at the 45 MW load level.

Parametric data, gathered from previous boiler testing, were used to train several different neural networks for NOx and unit heat rate. Numerous computer experiments were conducted to determine the best means of maximizing the neural network's ability to predict boiler output responses when given different input data. Work on the numerical optimization process consisted of testing different pre-packaged and self-written computer algorithms to determine which provided the necessary accuracy, flexibility, and dependability this particular application.

The NOx and heat rate neural networks produced accurate results in predicting boiler response values at full load conditions. The average difference between the model predicted values and the actual values were 0.01 lb/MBtu and 26.5 Btu/kWh, respectively. The optimization algorithm produced results that were within $\pm$ 2.5% of the optimal values that had been calculated using statistical and curve fitting methods. The 45 MW neural network models produced results of the same quality as the full load models. The average difference for these models was 0.016 lb/MBtu for NOx and 19.4 Btu/kWh for unit heat rate. However, when the equations comprising the 45 MW models were numerically optimized and graphed to show operational trends, they failed to coincide with either observed physical behavior or to previous statistical results.

The fundamental parameter that caused the full load models to succeed and the 45 MW models to fail was the quality of the data used to train the neural networks. The influence of this parameter pervades throughout this thesis and its effects on the experiments conducted provided significant insight to future utilization of this modeling and optimization methodology.

CHAPTER 1

INTRODUCTION

Title IV of the 1990 amendments to the Clean Air Act require that, depending on the type of boiler, the emission of oxides of nitrogen (NOx) be reduced to 0.45 or 0.50 pound per million Btu of fuel input to the boiler. The Potomac River Station in the State of Virginia is required to achieve further reductions to an emission rate of 0.38 lb./MBtu.

Unfortunately, as a boiler is operated in a low NOx emission mode, main and reheat steam temperatures often decrease, resulting in lower turbine cycle efficiency. Power plant performance is measured in terms of the unit heat rate, which is the reciprocal of the efficiency, and it has units of (Btu/kWh). As unit heat rate increases, more coal must be consumed to produce the same power output. Maintaining reasonable heat rate levels while reducing and controlling NOx emissions via boiler tuning is the goal of an on-going study at the Potomac Electric Power Company's (PEPCO) Potomac River Station. Previous work at Potomac River showed that unit heat rate and NOx production depend strongly upon boiler operating conditions. With numerous dependent and independent operating parameters and the complex interaction of these parameters, formulating phenomenological model(s) of power plant operation is difficult, if not impossible.

This study addresses the problem from a mathematical and computational approach. Working with field data obtained during plant operation, neural networks, a form of artificial intelligence, were applied to create a model of furnace operation, NOx, and heat rate. The neural network generates a mapping of operational inputs to the subsequent outputs using objective numerical values, without requiring a quantitative knowledge of the underlying physics. The resulting equations from the neural network model are analyzed by an optimization algorithm which determines the operating conditions that minimize heat rate subject to maintaining NOx regulatory limits. The emphasis of this investigation was to develop a methodology for using neural networks and optimization algorithms to determine optimal furnace operating conditions. An underlying premise of this thesis is the development of a series of computer algorithms that derives the input to output mapping equations, feeds the equations into the optimization algorithm, and reports the calculated optimal boiler operating conditions back to the plant engineer or operator. Therefore, the subsequent computational solution methods must be compatible so as to facilitate the passing of data between algorithms, simple to operate and understand, and easy to implement into a larger boiler control/optimization software package. Additional considerations included determining the amount of data required to achieve adequate mappings and the degree of uncertainty present in such a methodology.

CHAPTER 2

POWER PLANT OPERATION-EMISSIONS AND PERFORMANCE

In order to gain proper perspective of how neural networks were implemented into modeling boiler operation, the nature of the problem and previous research efforts must be discussed. The physical problem is described in this chapter along with the goals and objectives of the study. Finally, an overview of the solution procedure is provided, showing how the physical problem was analyzed, and solved.

BOILER OPERATION AND EMISSIONS MONITORING

The Potomac Electric Power Company (PEPCO) has five units at the Potomac River Station, each equipped with a Combustion Engineering, tangentially fired, drum design boiler. The nominal generating capacity for each unit is 108 MW, while the minimum load is 35 MW. Typically, the station operates 46% of the time at full load (>90 MW) and 48% of the time at lower loads (35 to 50 MW). The corner-fired unit burns eastern bituminous coal, pulverized in four Raymond bowl mills. The four mills, A through D, supply four different burner levels within the boiler, each level having four burners. All four mills must be operated when generating 90 to 108 MW, three mills are needed in the mid-load range (60 to 90 MW), and only the center two mills operate in the low load range (35 to 60 MW).

Air for combustion is provided by auxiliary air and fuel-air nozzles. At each of the 16 burners a fuel-air nozzle is located. These burners and nozzles can be vertically tilted up and down with a range of -30° to +30°, from the horizontal. The unit's steam cycle is sub-critical and the main and single reheat steam temperatures are controlled by the degree of burner tilt. In addition to the fuel-air nozzles located at each of the burners, an auxiliary air nozzle is located between each burner, above the top row, and below the bottom row of burners. The total air introduced into the boiler is quantified by the economizer oxygen level while the air flow rate to each of the 36 nozzles is controlled by dampers. All of the dampers can be adjusted from fully open (position 5) to fully closed (position 1). Windboxes, located at the four corners of the boiler, supply air to all of the air nozzles. The amount of economizer oxygen supplied by the fans and the damper settings controls the pressure differential between the windbox and furnace [1,2,3,4]. The location of the described boiler hardware is detailed in Figure 2.1.

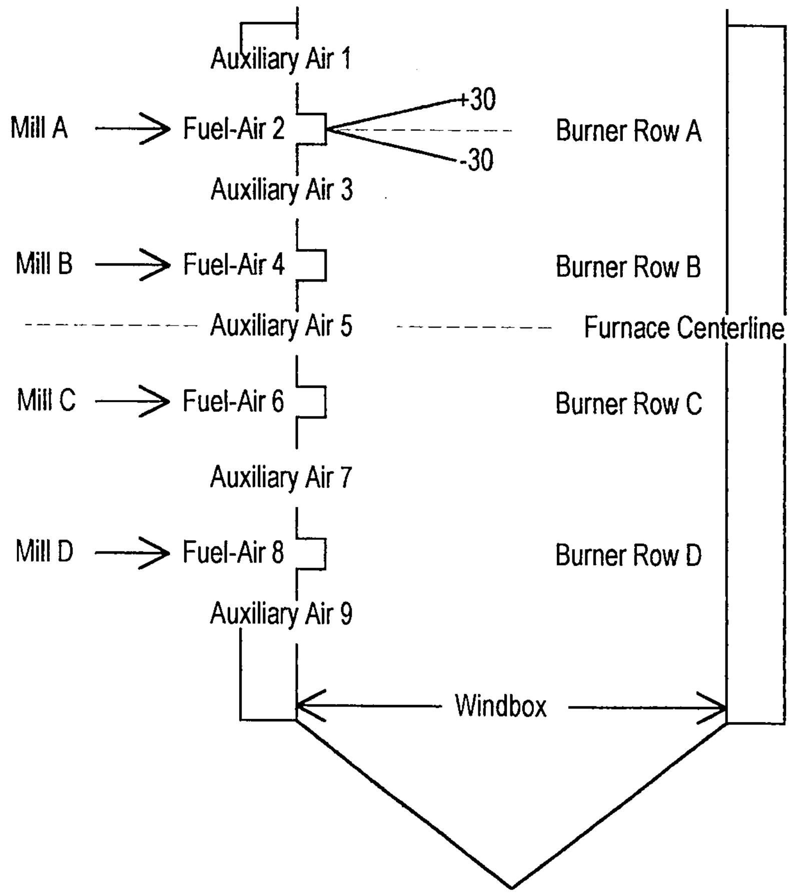

Figure 2.1

Potomac River Unit 4 Boiler Corner Configuration

Data collection is achieved using the Electric Power Research Institute (EPRI) Plant Monitoring Workstation (PMW). The recorded data includes main and hot reheat steam temperatures, economizer O_2, unit load, burner tilt angle, mill feeder speeds, windbox pressure, air damper positions, air preheater inlet gas temperature, and flue gas temperature at the economizer exit. Data are downloaded from the PMW as two-minute average values and stored as archived data. These data can be manipulated and analyzed in several different ways to fine-tune operation. Additional software produced by Lehigh University for EPRI allows the calculation of unit heat rate, boiler efficiency, coal, gas and air flow rates, and other performance parameters associated with coal-fired boilers.

Environmental Protection Agency (EPA) regulations require the Potomac River Station to perform Continuous Emissions Monitoring (CEM) for NOx, SO_2, and opacity [5]. NOx emissions were measured in the stack using a Monitor Labs Inc. Model 8840 Analyzer, while the wet O_2 present in the stack was determined with a Teledyne meter. The NOx emissions rate (lb./MBtu) was calculated using EPA Method 19 for dry NOx and wet O_2 measurements. To convert the NOx rate from parts per million (ppm.) to lb./MBtu, Equation 2.1 was used [2]:

$$\mathrm{NOx}(\mathrm{lb/MBtu}) = \frac{(1.194\mathrm{x}10^{-7})*9780*20.9*\mathrm{NOx}(\mathrm{ppm})}{20.9 - \dfrac{O_{2,\mathrm{stack}}}{(1.0-0.027)}} \qquad \mathbf{(2.1)}$$

In adapting Eq. 2.1 to this boiler, the following parameter values were used:

F-factor = 9780 dscf/MBtu

moisture fraction = 0.027

$O_{2,\mathrm{stack}}$ = economizer O_2+1.8%

MINIMIZATION GOALS

This study focused on optimizing boiler operation for two fundamental output parameters, NOx and heat rate. Although several other important considerations exist, these two are the most important due to EPA regulations and PEPCO's desire to maximize overall power production efficiency. The consummate goal is to minimize heat rate, which is synonymous with maximizing efficiency, while maintaining NOx emission rates below the levels required by federal and state regulations. Knowing what comprises the two optimization goals is prerequisite for undertaking the task of optimizing these outputs.

NOx Formation

During the combustion of pulverized coal, NOx is formed from the nitrogen contained in both the coal particles and the excess air. The majority of the oxides formed is nitric oxide, NO, and controlling its formation during combustion is imperative in maintaining compliance with the Clean Air Act Amendments. NO formed from fuel-bound nitrogen (fuel NO) accounts for 60% to 80% of the total NO emissions from pulverized coal boilers [6]. NO formation occurs during both the coal devolatilization and char burn out phases of coal combustion. The fuel NO generated during devolatilization proceeds primarily from the formation of hydrogen cyanide, HCN and its subsequent decomposition [6]. The amount of fuel NO formed via this mechanism is extremely sensitive to burner aerodynamics and local air to fuel ratios in the near-burner region. NO increases with increasing local air-fuel ratio, which is dependent upon the fuel and secondary air distribution. This property is generally considered the most significant of those relevant to pulverized coal combustion.

The nitric oxide formed from molecular nitrogen in the combustion air is termed "thermal NO". The rate of thermal NO formation is primarily a function of gas temperature with temperatures beneath 1600-1800°K yielding relatively small concentrations of thermal NO. Another factor in thermal NO production is the amount of oxygen present in the furnace. Therefore, economizer oxygen levels, especially at high temperatures found at full load, become critical to controlling thermal NO formation. Overall, the amount of oxygen present in the furnace, the manner it is distributed, and the near burner mixing patterns during combustion dictate the amount of fuel and thermal NO created, as well as other NOx species. For these reasons, the manner in which boiler parameters such as coal flow, air flow, and burner tilt angle are set has a significant effect on the combustion reaction process and the rate of NOx formation. These parameters regulate the combustion process and when properly balanced, can be used to reduce NOx emissions.

Unit Heat Rate

Unit heat rate is a limiting factor on the amount of NOx emissions that it is practical to achieve. This is a "cost" function for power production with the units (Btu/kWh). The higher the heat rate value, the more fuel is required to produce a given amount of power. As with the creation of NOx, heat rate is a function of how the boiler is operated. The fuel flow, air flow and distribution and mixing patterns control how effectively the fireball heats the steam within the boiler waterwalls. The higher the steam temperature, the more efficient the turbine cycle is in producing power. At Potomac River Unit 4 the governing control for steam temperatures is the burner tilt angle. The greater the tilt above the horizontal, the higher the steam temperatures. In addition, at Unit 4, as the power output decreases, steam temperatures decrease, despite any efforts and changes made to the boiler operating conditions. The main steam temperature stays relatively constant from full load at 108 MW to 75 MW, then decreases to an average of 28°F lower at 45 MW [7].

The net unit heat rate was obtained using EPRI's HEATRT Code [8] and the following assumptions were made to run the code [2]:

- Baseline values of turbine cycle heat rate vary with main steam flow rate and were obtained from the heat balances in the unit's thermal kit.
- Economizer O_2, economizer exit gas temperature, and steam temperatures vary as functions of load.

- Convective pass leakage and mill exit temperature were constant.
- Unburned carbon varied as a function of economizer O_2.
- Forced draft fan air inlet temperature was constant and the air inlet relative humidity was 50%.
- Exhauster back pressure was constant at its design value of 1" Hg.

The fourth assumption entails unburned carbon which requires a brief description. The amount of unburned carbon, also referred to as Loss On Ignition (LOI), accounts for the fuel efficiency during boiler operation. As the amount of economizer O_2 is decreased, less air is available for the combustion process, resulting in incomplete burning of fuel. The percent of unburned carbon increases which, in turn, increases the amount of fuel required to produce a given power output.

BOILER OPERATION AND NOX MINIMIZATION

As described in the previous section, there are several operating parameters that can be adjusted and tuned to achieve lower NOx emissions without sacrificing unit efficiency. After several months of parametric testing and analysis by D'Agostini et al. [2, 3, 4], boiler output responses and trends to different combinations of operating conditions were established. This study focused on the two load levels, (105 and 45 MW), at which Unit 4 operates most of the time; but the entire database and analysis were important for the neural network and optimization program development as well as for checking the validity of results.

Operating Parameters

As was mentioned in the boiler configuration section, four types of operator controllable parameters were varied during the testing. Economizer O_2 and burner tilt angle can be varied directly and act as independent control parameters. However, numerous possible variations of coal and air flow distributions exist and a complete coverage of these would require an unrealistic amount of testing. Instead, three bias parameters were formulated to capture the effects of these different distributions. These discrete quantities are mill bias, β, fuel-air bias, ϕ, and auxiliary air bias, α. These represent the vertical distribution of coal flow, fuel-air flow, and auxiliary air flow, respectfully. The following derivations of these bias parameters are based on the boiler configuration shown in Figure 2.1, and were developed by D'Agostini, et al. [4].

Mill Bias. The vertical distribution of coal to the four elevations of burners is specified based on the distance to the burner above or below the point midway between the B and C burner rows (see Figure 2.1). Coal streams entering above this centerline are deemed positive while those below are negative. This expression is symbolically expressed as:

$$\beta = \frac{1}{\dot{m}_{c,tot}} \cdot \sum_i \dot{m}_{c,i} \cdot \frac{x_i}{L} \tag{2.2}$$

where

m_c = coal mass flow rate
x = distance from injection point to furnace centerline
L = distance between furnace centerline and the top burner row
i = index denoting each mill
tot = sum of all mills

The coal flow rate to each mill is :

$$\dot{m}_c = k_f \cdot \Omega \tag{2.3}$$

with

k_f = feeder constant (assumed to be the same for all feeders)
Ω = feeder motor speed

The ratio of x_i/L for each burner row is:

Burner Row	Ratio x_i/L
A	+1
B	+1/3
C	-1/3
D	-1

Substituting this information and Eq. 2.3 into Eq. 2.2, the final result is:

Mill Bias: $$\beta = \frac{(\Omega_A - \Omega_D) + \frac{1}{3} \cdot (\Omega_B - \Omega_C)}{\sum_i \Omega_i} \qquad (2.4)$$

where: Ω_i = motor speed of the ith mill feeder (rpm)

Auxiliary Air Bias. The same methodology for the formulation of mill bias applies to the auxiliary air bias parameter. The secondary air flows entering the furnace from each of the five rows of auxiliary air nozzles are multiplied by the distances from the nozzle locations to the burner row centerline, which occurs at auxiliary air nozzle #5. Unfortunately, Potomac River Unit 4 has no quantifiable measure of the air flow rate through these nozzles. The flow rate is determined by the degree of openness that the damper is set at (1≈fully closed, 5≈fully open). The assumption was made that the air flow rate is proportional to the degree of openness so that a damper set at #4 has twice as much air flow than a damper set at #2. The distances between nozzles is normalized using:

$$N = \sum_i Aux_i \cdot \frac{x_i}{L} \qquad (2.5)$$

where

N = parameter given by equation 2.5
Aux = auxiliary air damper setting
i = index for each row of auxiliary air dampers (i=1,3,5,7,9)
x = distance between the air nozzle level and furnace centerline
L = distance between the nozzle rows (not the same as for mill bias parameter)

The x_i/L ratio for each nozzle row is:

Nozzle Row	Ratio x_i/L
1	+2
3	+1
5	0
7	-1
9	-2

The maximum possible value for N occurs by setting nozzle rows 1 and 3 at position #5 and rows 7 and 9 at position #1. This results in N equal to 12 and therefore a normalization factor of 1/12. From this, Eq. 2.5 becomes:

$$\alpha = \frac{1}{12} \cdot \sum_i \mathrm{Aux}_i \cdot \frac{x_i}{L} \qquad \textbf{(2.6)}$$

Expanding this summation results in the final form of the auxiliary air bias parameter:

Auxiliary Air Bias: $$\alpha = \frac{2 \cdot (\mathrm{Aux}_1 - \mathrm{Aux}_9) + (\mathrm{Aux}_3 - \mathrm{Aux}_9)}{12} \qquad \textbf{(2.7)}$$

where Aux_i = damper position for the i^{th} row of auxiliary air nozzles.

Fuel Air Bias. This parameter represents the vertical distribution of air through the fuel-air nozzles. At different unit loads, either two, three, or four burner rows will be operational. This value of the fuel-air to fuel flow ratio averaged over each active burner row equates to:

$$\phi = \sum_i w_i \cdot \left(\frac{\dot{m}_{fa}}{\dot{m}_c} \right)_i \qquad \textbf{(2.8)}$$

where

w = weighting factor
m_{fa} = fuel air flow rate
m_c = coal flow rate
i = index for fuel air damper rows (i=2,4,6,8)

The weighting factor, w_i, is given as:

$$w_i = \frac{\dot{m}_{c,i}}{\dot{m}_{c,tot}} \tag{2.9}$$

The coal flow rate is the same as for mill bias:

$$\dot{m}_c = k_f \cdot \Omega \tag{2.10}$$

This reduces Eq. 2.9 to

$$w_i = \frac{\Omega_i}{\Omega_{tot}} = \frac{\Omega_i}{n \cdot \Omega_{avg}} \tag{2.11}$$

where:

avg = average motor speed among all operating feeders
n = the number of operating mills

The problem of direct measurement of air flow through the fuel air nozzles exists as it also does for the auxiliary air nozzles. The assumption was thus made that air flow through the fuel air damper is proportional to the damper setting (degree of openness) multiplied by the square root of the windbox to furnace pressure differential. This is required because the fuel air damper openings are smaller than the auxiliary air dampers and so the air flow through the fuel air dampers is more restricted. This makes the fuel air dampers more sensitive to the furnace back pressure, or windbox pressure, and this effect is analagous to flow through a channel. This is symbolically stated as:

$$\dot{m}_{fa,i} = k_{fa} \cdot FA_i \cdot \sqrt{\Delta P_{wb}} \tag{2.12}$$

where: k_{fa} = damper constant
FA_i = fuel air damper position
ΔP_{wb} = windbox to furnace pressure differential ("H_2O)

Substituting Eqs. 2.9, 2.10, 2.11, and 2.12 into Eq. 2.8 the expression becomes:

$$\phi = \sum_i \frac{1}{n \cdot \Omega_{avg}} \cdot \frac{k_{fa}}{k_f} \cdot FA_i \cdot \sqrt{\Delta P_{wb}} \qquad \textbf{(2.13)}$$

It was assumed that the ratio of the fuel air damper constant, k_{fa}, and coal flow constant, k_f, equals a constant, K, when the fuel air bias, (ϕ), equals unity at a reference condition. This assumption simplifies the equation to:

$$\phi = \frac{1}{n \cdot \Omega_{avg}} \cdot K \cdot \sqrt{\Delta P_{wb}} \cdot \sum_i FA_i \qquad \textbf{(2.14)}$$

To satisfy the $\phi = 1$ at a reference condition, K is:

$$K = \frac{\Omega_{ref}}{FA_{ref} \cdot \sqrt{\Delta P_{wb,ref}}} \qquad \textbf{(2.15)}$$

where:

$\Omega_{ref} = 700$ rpm
$FA_{ref} = \#3$
$\Delta P_{wb,ref} = 2.5$ ("H_2O)

Substituting Eq. 2.15 into 2.14 and rewriting, the final form is:

Fuel Air Bias: $$\phi = \frac{1}{n} \cdot \frac{\Omega_{ref}}{\Omega_{avg}} \cdot \sqrt{\frac{\Delta P_{wb}}{\Delta P_{wb,ref}}} \cdot \sum_i \frac{FA_i}{FA_{ref}} \qquad \textbf{(2.16)}$$

where:

n = number of mills operating
ΔP_{wb} = windbox to furnace differential pressure ("H_2O)
FA_i = position (amount dampers are open) for ith row of fuel air dampers
Ω_{avg} = the average of all mill feeder speeds in operation (rpm)

Note that Eq. 2.16 requires only the mill speeds and fuel air damper positions corresponding to active burner rows.

Trends and Effects at 105 Megawatts

With the furnace operating parameters defined, the effects these parameters have on boiler operation, NOx emission, and heat rate are now examined at the rated maximum output for Potomac River Unit 4. At the 105 MW load, 50 parametric tests were conducted and all of the analysis and results were summarized in [4]. The following sections highlight the key parameters and their causal relationships with NOx and heat rate.

Economizer Oxygen (O_2). The previous discussion on NOx formation indicated the sensitivity of NOx to the amount of economizer, or excess, oxygen present for combustion, especially at high temperatures. A strong linearly increasing dependence of NOx on economizer oxygen is seen in Figure 2.2. It is readily apparent that to maintain a NOx emission level of 0.40 (lb./MBtu), economizer oxygen cannot be greater than two percent. However, a reduced level of oxygen for combustion results in more incomplete burning of coal particles, higher CO emissions, and waterwall tube wastage. The effect of economizer oxygen on unit heat rate, as determined by the HEATRT code, is shown in Figure 2.3. The scatter in the heat rate data is relatively large, thus making it difficult to establish any trends within the data.

Burner Tilt Angle The burner tilt angle directly affects the air/fuel mixing pattern and fireball location within the furnace. Poor mixing results in incomplete burning and the fireball location dictates steam temperatures (and thus heat rate), residence time, and the creation of thermal NOx. Figure 2.4 demonstrates this tradeoff by the parabolic shape of the NOx vs. tilt angle relation. A minimum NOx occurs around the +5° to +10° burner tilt range. Figures 2.5 and 2.6 show that economizer O_2 and burner tilt are relatively independent of each other in their effects on NOx. A negative burner tilt angle lowers the fireball within the furnace, but it does not cause a

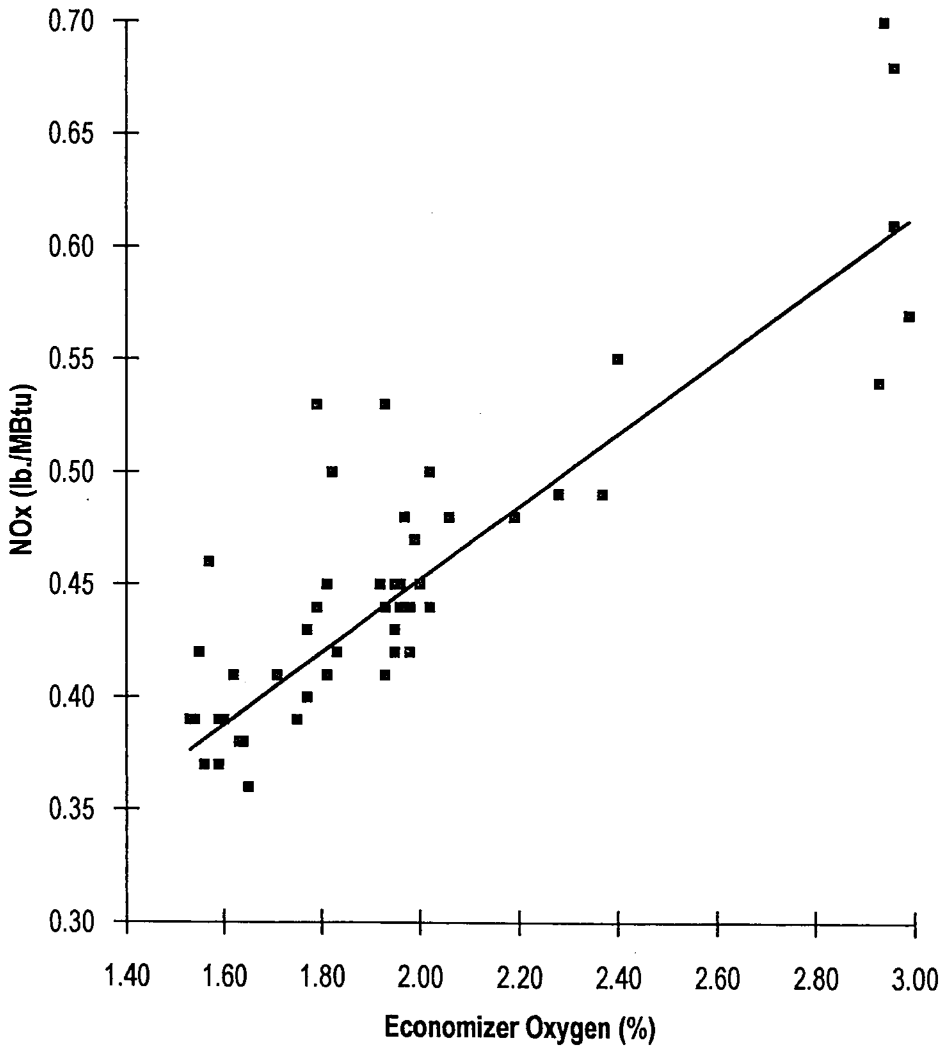

Figure 2.2

Full Load NOx vs. Economizer Oxygen

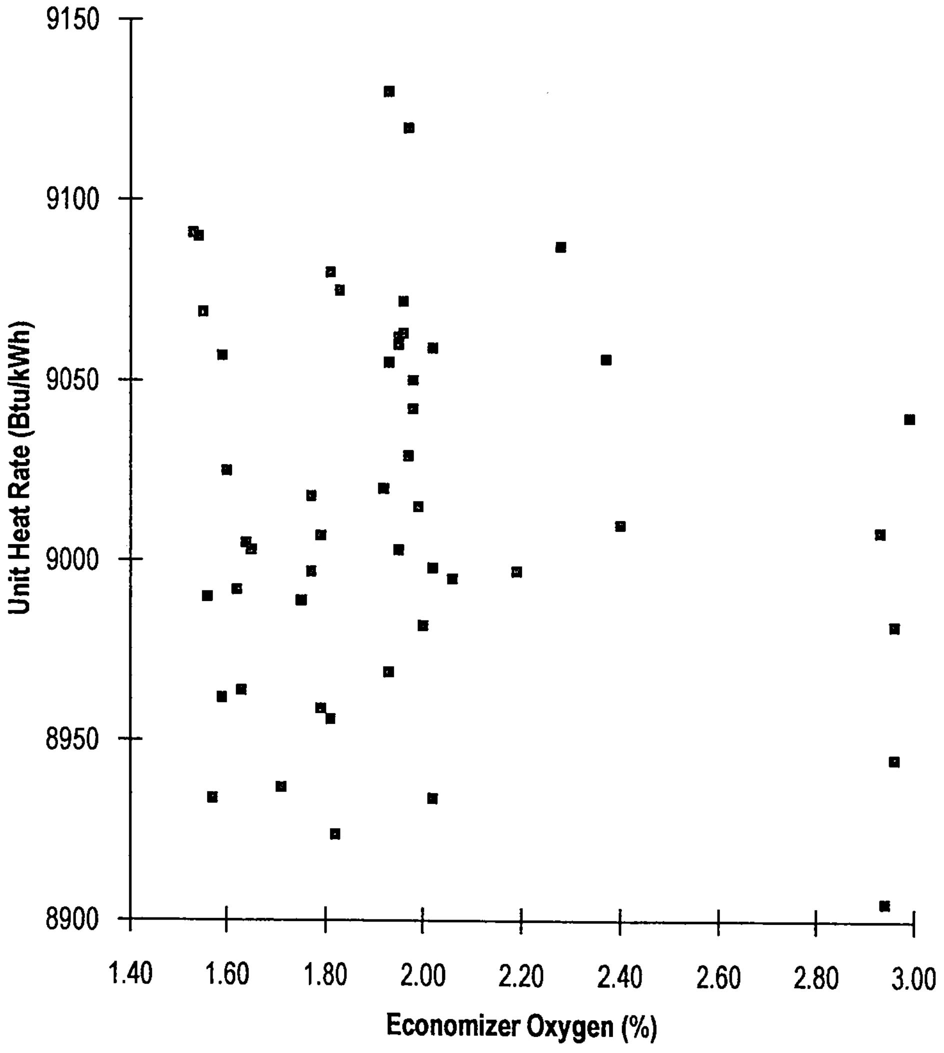

Figure 2.3

Full Load Unit Heat Rate vs. Economizer Oxygen

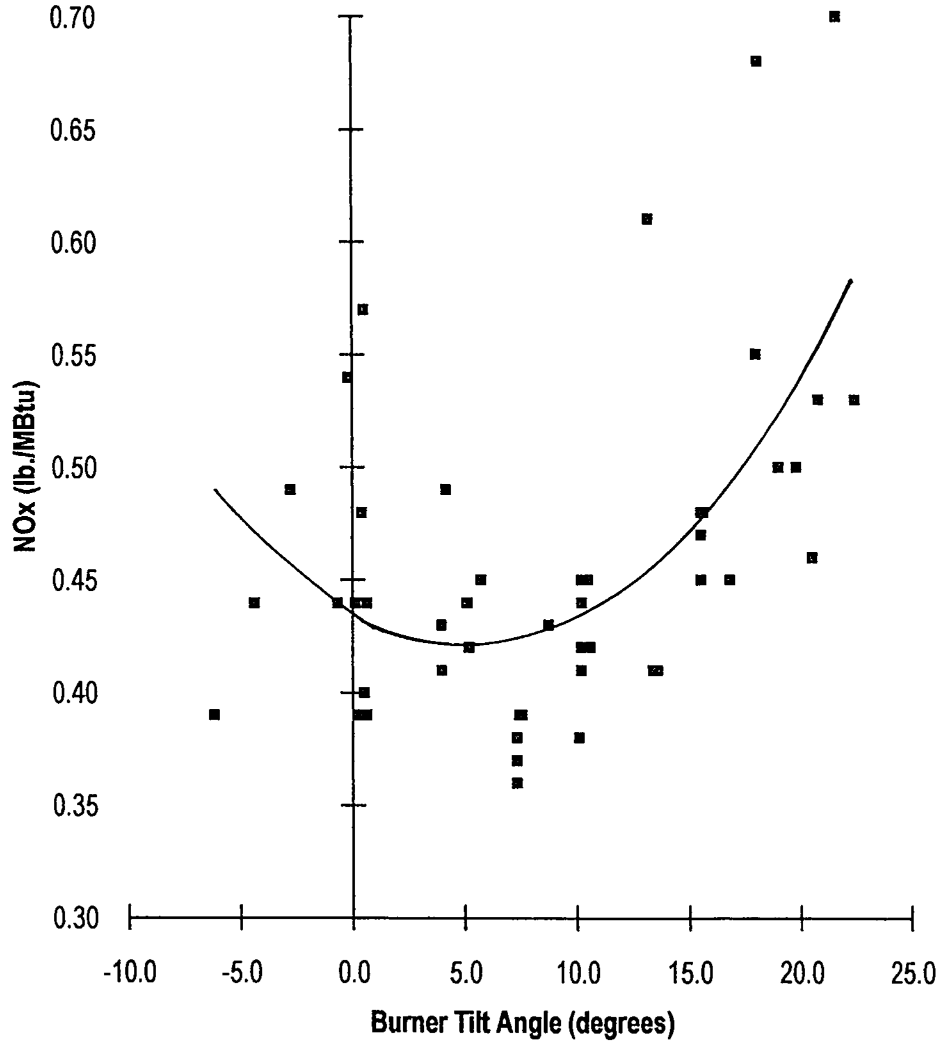

Figure 2.4

Full Load NOx vs. Burner Tilt Angle

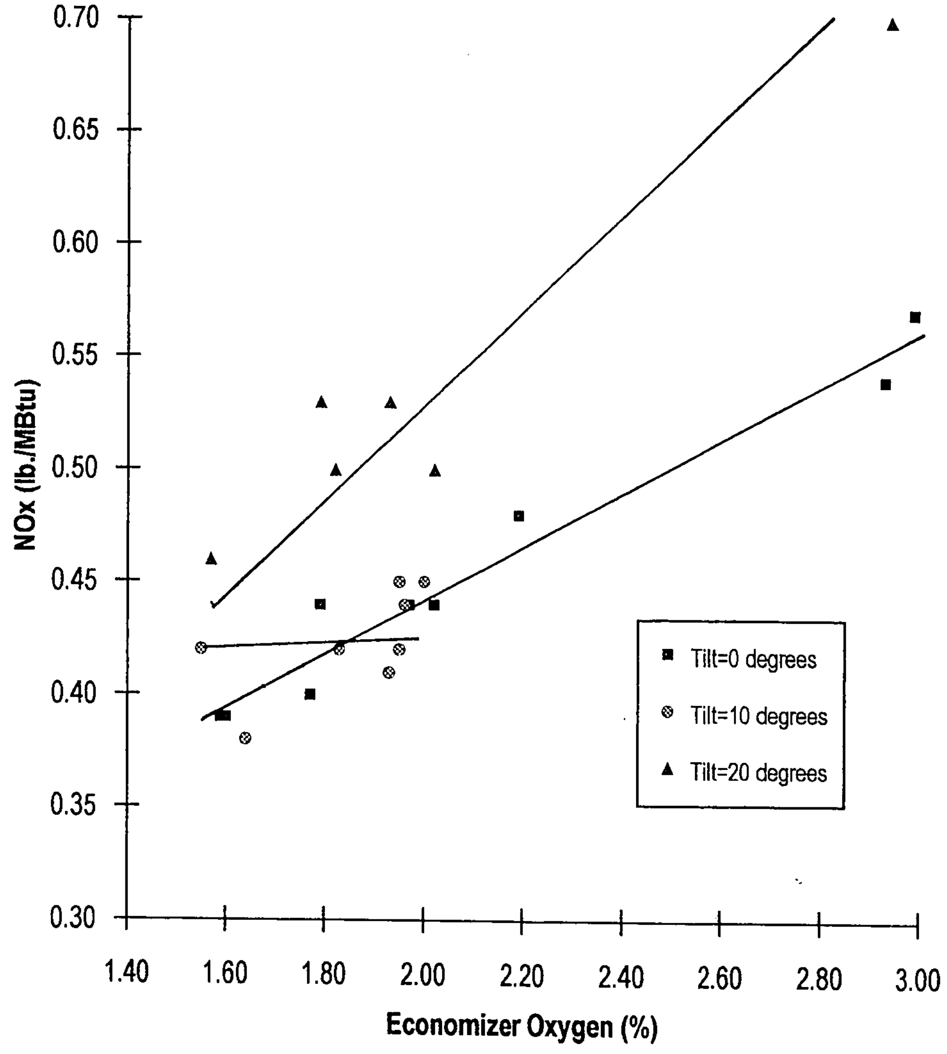

Figure 2.5

Full Load NOx vs. Economizer Oxygen at
Constant Burner Tilt Angle

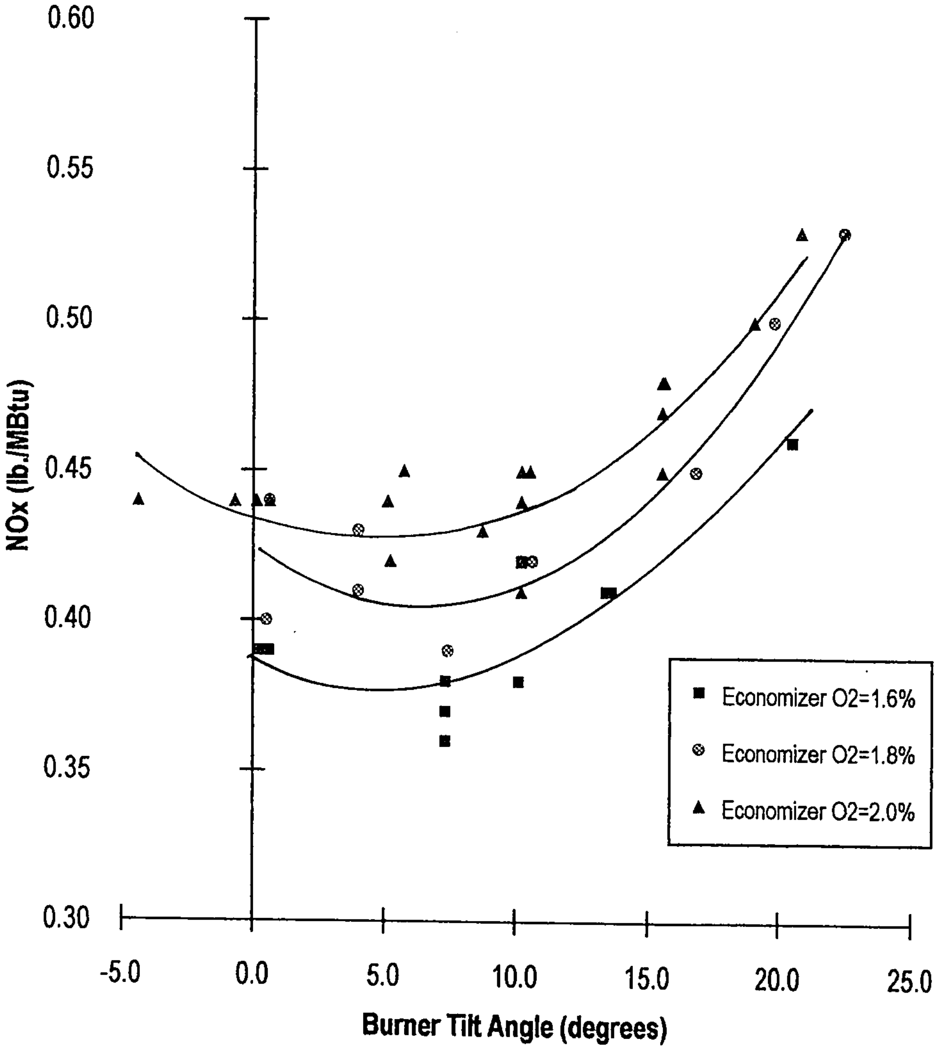

Figure 2.6

Full Load NOx vs. Burner Tilt Angle at Constant Economizer Oxygen on NOx. A negative burner tilt angle lowers the fireball within the furnace, but it does not cause a significant change in NOx. But as seen in Figure 2.7, negative tilts cause the main steam temperature (and also hot reheat) to drop below the temperature set points for the drum boiler design. This causes heat rate to increase, and therefore, the best operating region for burner tilt angle is between +5 and +10 degrees above the horizontal.

Mill Biasing. At Potomac River Unit 4, full load operation requires all four mills to be running near or at their maximum coal loading capacity. The quality of the coal strongly influences whether any mill biasing can occur. High moisture or ash content along with a low higher heating value reduces the possibility of biasing the mills. However, if these factors do not apply, then NOx reductions up to 0.03 (lb./MBtu) can be achieved. Mill biasing entails reducing coal flow to burner row A, located at the top of the furnace, while the other mills remain fully loaded. One caveat of this technique is that more stringent fuel requirements would be needed which could negatively affect the utility's fuel costs.

Auxiliary Air Biasing. An effect called overfire air is known to decrease NOx emission during normal boiler operation. Overfire air is achieved by keeping the upper rows of auxiliary air dampers (#'s 1, 3, and 5) more open than the #7 and #9 dampers. The greater the bias value, the more air enters the furnace around the upper burner rows which creates a staged combustion effect. Only two test sequences were conducted at 105 MW that isolated the air bias effect and they are plotted in Figure 2.8. The majority of the tests were conducted at auxiliary air biases greater than 0.5 where no change was readily seen. More data on auxiliary air biasing are needed at full load to firmly establish the relationship between this parameter and NOx.

Fuel Air Biasing. Fuel air damper biasing was tested only in a limited way at full load. In all tests, all four dampers were changed uniformly between the #5 position and the #3 position.

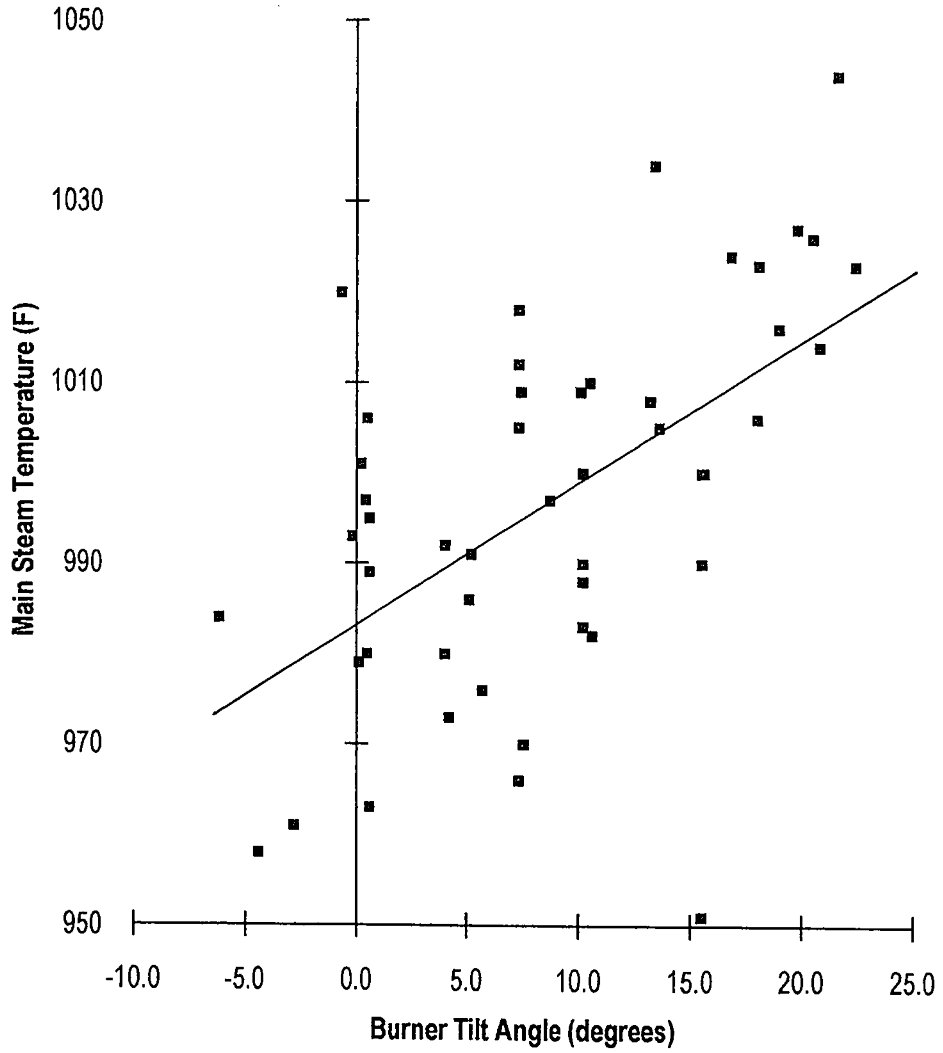

Figure 2.7

Full Load Main Steam Temperature vs. Burner Tilt Angle

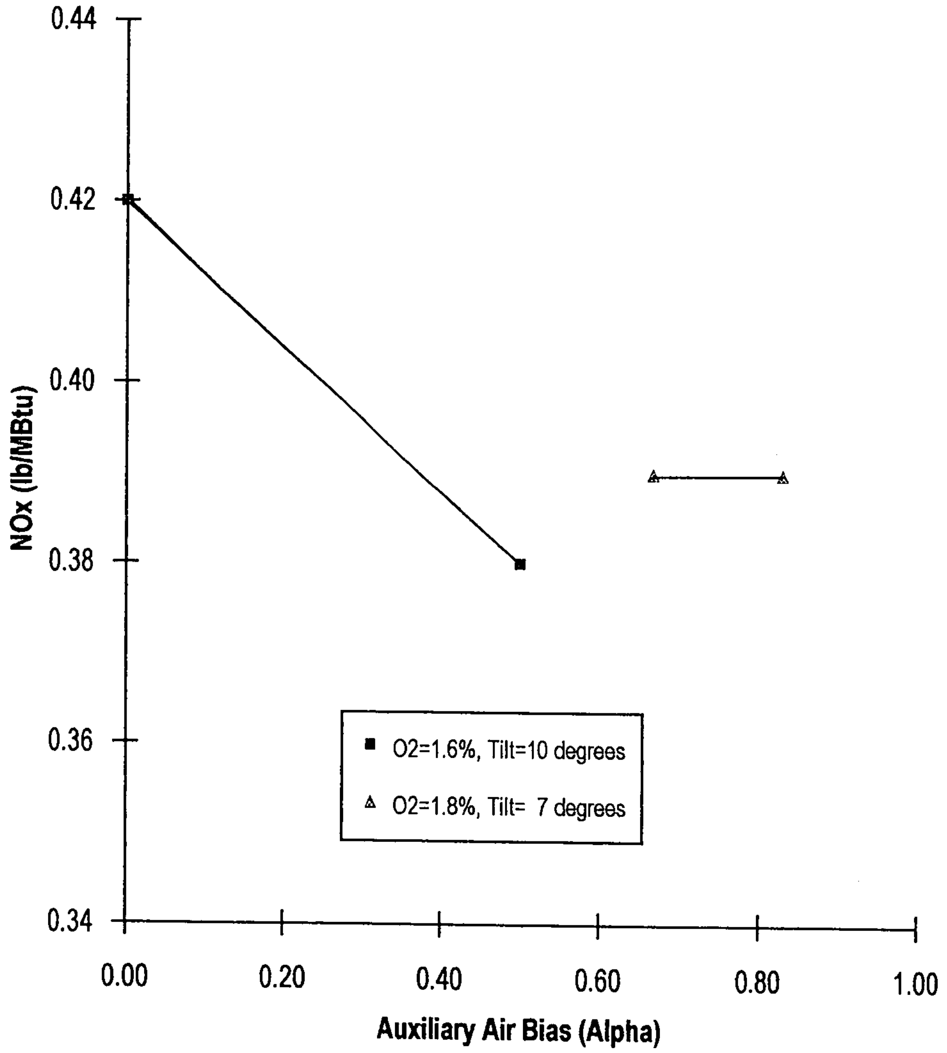

Figure 2.8

Full Load NOx vs. Auxiliary Air Bias at Constant Economizer Oxygen and Burner Tilt Angle

Closing the dampers any further was deemed too dangerous by the plant engineers. In addition, no tests were run in which the damper positions were varied with respect to the others. No observable impact on NOx due to the systematic changes in the fuel-air damper settings was observed during testing. As with auxiliary air biasing, more testing is required to fully understand the causal relationship between fuel-air biasing and the formation of NOx.

Trends and Effects at 45 Megawatts

The second most important load range for Potomac River Unit 4 operation is 35 MW to 50 MW. Testing at 45 MW was the most extensively conducted at this lower range and it provided valuable insight for low NOx operation and heat rate. Only two mills are needed at 45 MW, thus much greater flexibility in mill biasing occurs at this load level. The main challenge was to determine the combination of mills which provided low NOx emissions and sufficiently high levels of main and hot reheat steam temperatures. A total of 60 test points was gathered during testing at 45 MW, but heat rate was calculated for only 17 of these tests.

Economizer O_2. As seen previously at the maximum load range, economizer oxygen has a strong, linear effect on NOx production at 45 MW. Figure 2.9 details this trend, although with a large amount of scatter within the data. These data incorporate two different mill loading strategies, and when segregated the effect of mill loading on NOx becomes obvious. Figure 2.10 shows that using the two middle burner rows and the B and C coal feeders causes lower NOx emissions. Unlike full load operation, economizer oxygen cannot be less than four percent due to poor flame stability.

Burner Tilt Angle. Figure 2.11 shows the effect of burner tilt angle on NOx for two different combinations of mills. The results show that burner tilt angle has an effect upon NOx emissions at 45 MW that is similar to the full load behavior. The overall optimal tilt angle range is from 0° to

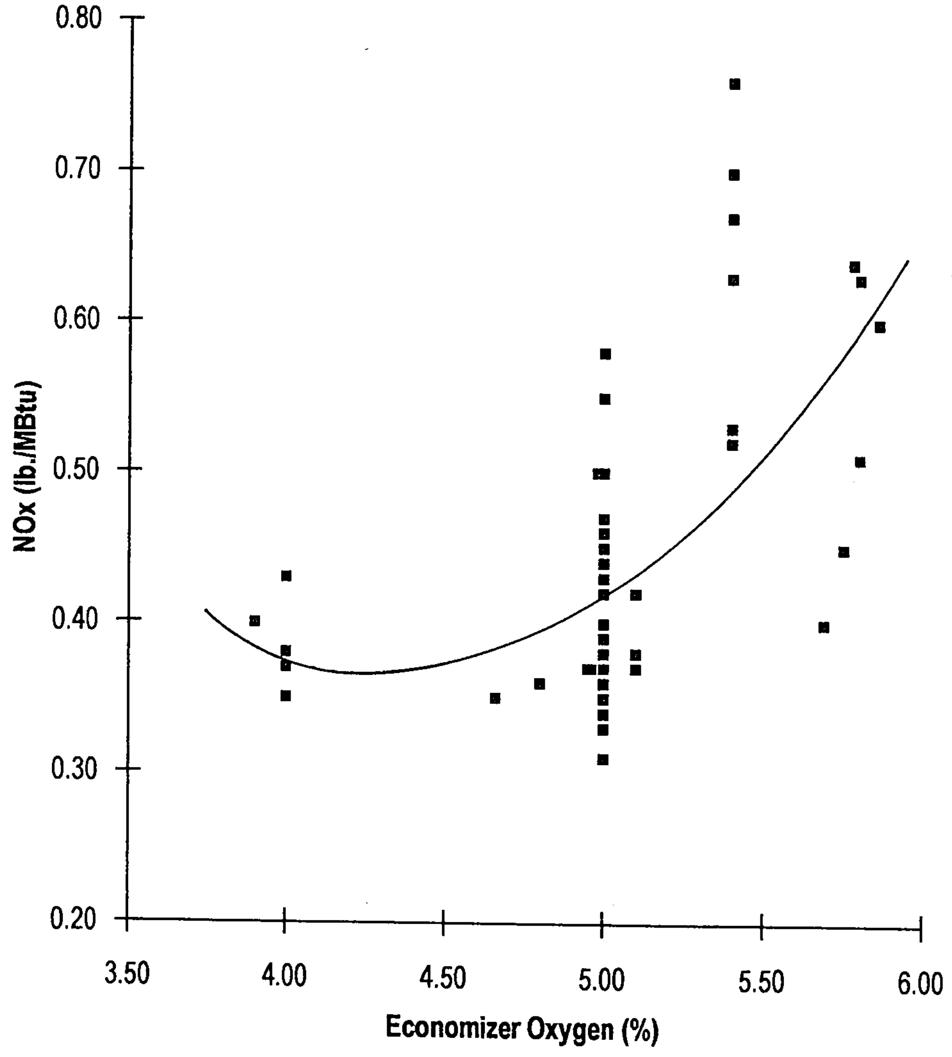

Figure 2.9

45 MW NOx vs. Economizer Oxygen

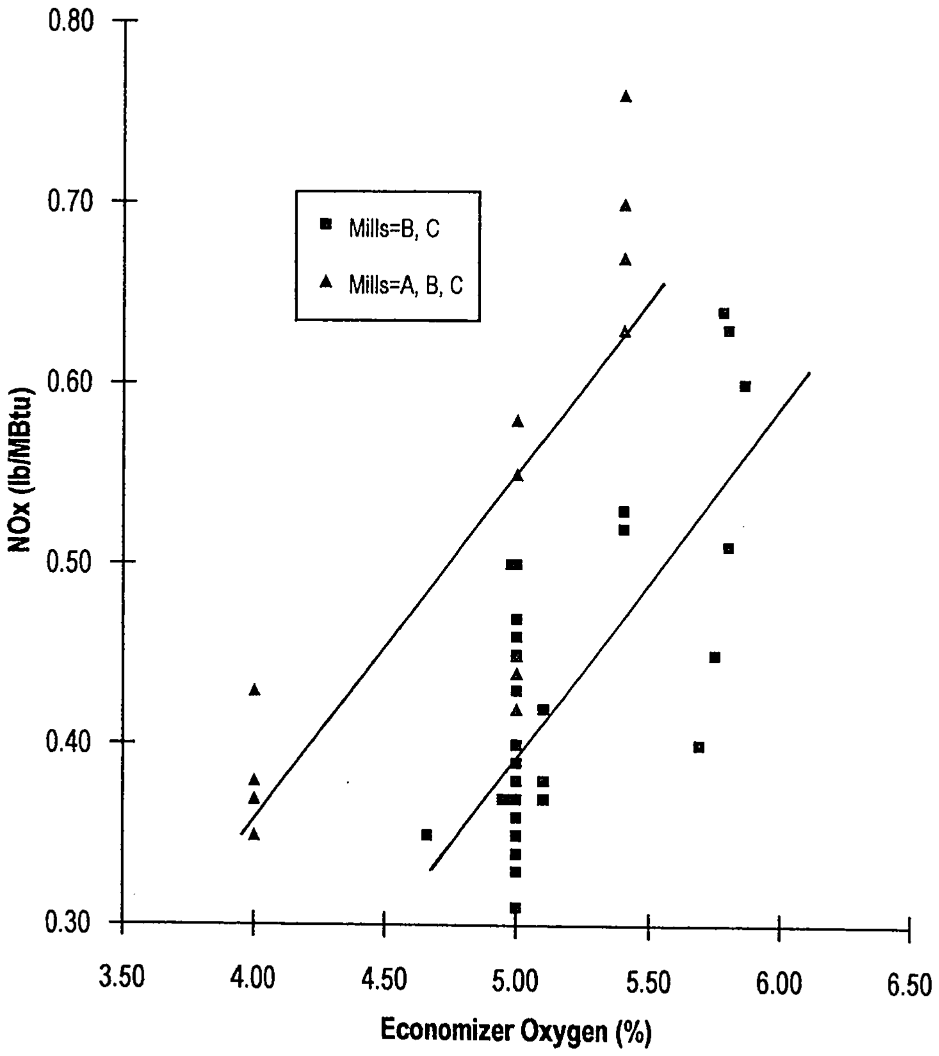

Figure 2.10

45 MW NOx vs. Economizer Oxygen
at Constant Mill Loading Patterns

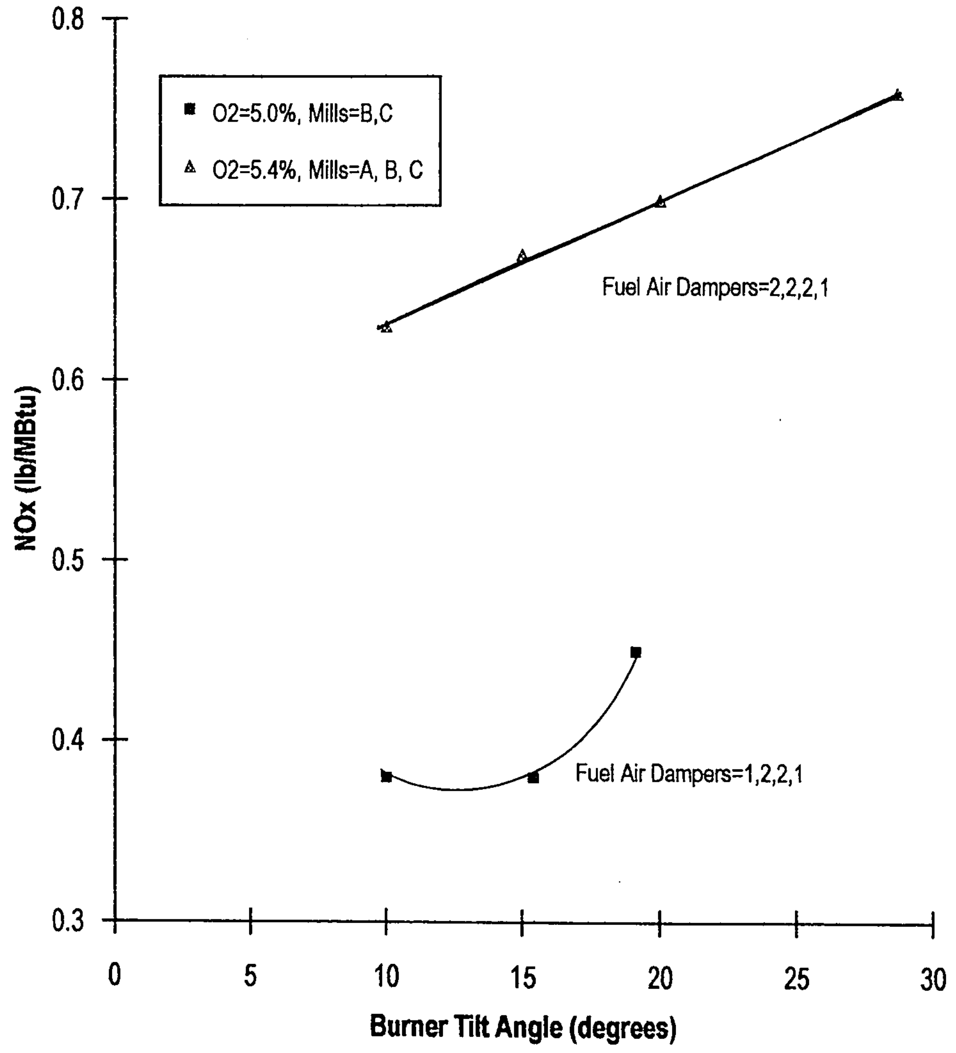

Figure 2.11

45 MW NOx vs. Burner Tilt Angle at Constant Economizer Oxygen, Fuel Air Damper Settings, and Mill Loading Patterns

+10° above the horizontal. Additional factors that affect NOx formation which are not specified in this graph are the degree of boiler cleanliness and secondary air distribution.

Mill Biasing. The three different mill loading schemes that were tested incorporated two and three mill operation. The two mill operation had the B and C mills feeding the center two burner rows. Mills A, B, and C or mills B, C, and D comprised three mill operation at 45 MW. The third of these mill biasing schemes resulted in extremely low steam temperatures on the first test and further testing/operation at this mill configuration was eliminated. Almost 72% of the testing was conducted with two mills in operation with the remainder using the A, B, and C mill loading configuration. No other two-mill combinations were tested, although this type of testing is needed to account for situations where either of the two middle mills becomes unserviceable. Although two mill operation does not allow for much biasing between the two mills due to the fuel demand required by the furnace, Figure 2.12 demonstrates that the lowest NOx levels were achieved when using only the two center burner rows.

Auxiliary Air Biasing. The role of the auxiliary air damper bias at 45 MW is slightly different than at higher loads. An increase in the damper bias causes more air to enter the upper levels of the windbox, thus increasing the separation of fuel and air in the furnace. As more air was introduced around the top row of burners (increasing α), the NOx level decreased regardless of the other operating conditions. This trend is detailed in Figure 2.13.

Fuel Air Biasing. At lower loads, the fireball intensity is much less than at higher load levels. This allows for more local effects around the burners to impact upon the creation of NOx. The fuel-air bias becomes more important at lower loads because of these local effects. Figure 2.14 shows that as the fuel air bias parameter, ϕ, decreases, NOx decreases.

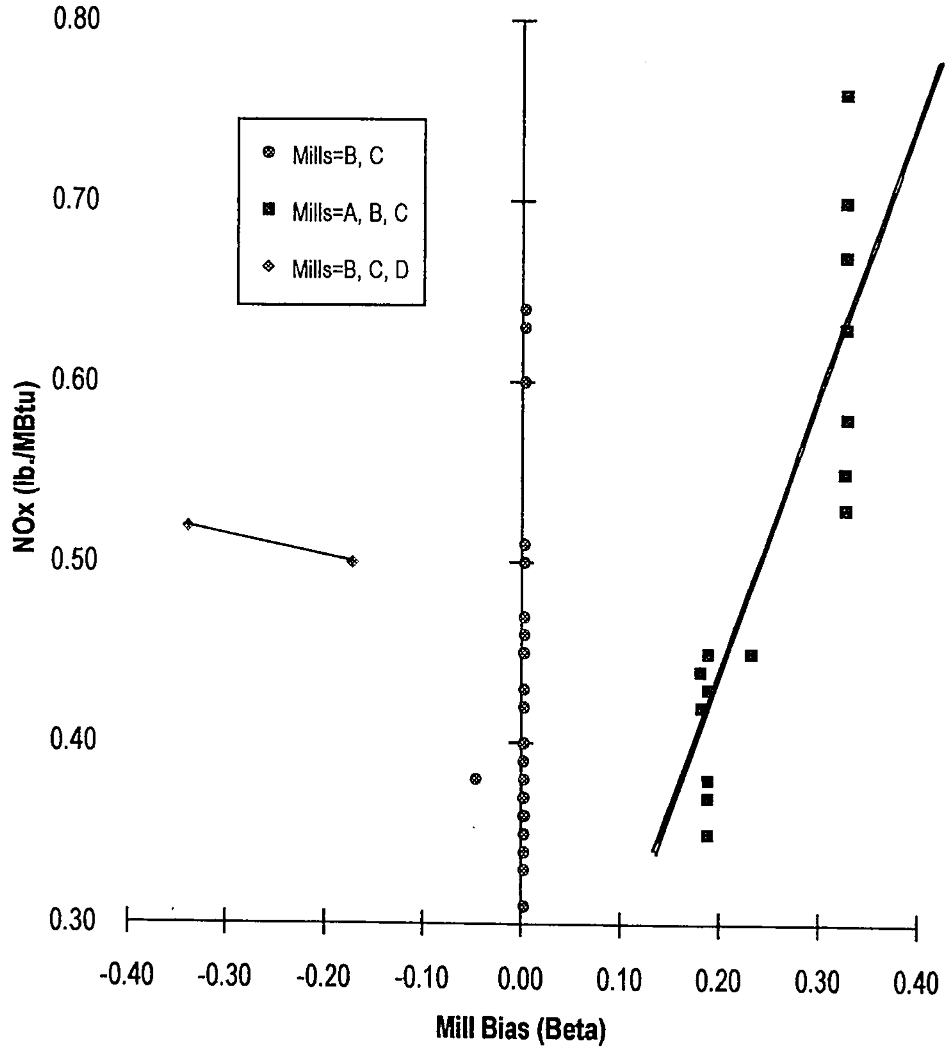

Figure 2.12

45 MW NOx vs. Mill Bias at Constant Mill Loading Patterns

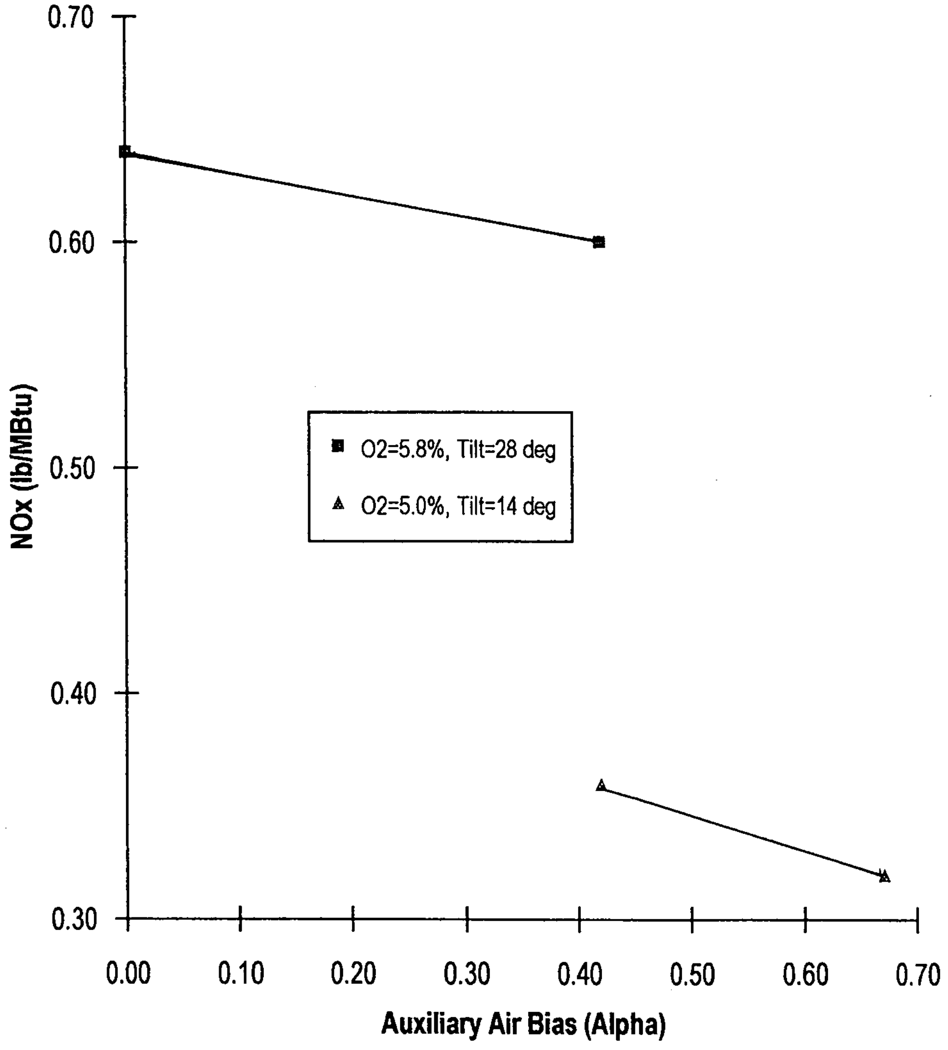

Figure 2.13

45 MW NOx vs. Auxiliary Air Bias at Constant Economizer Oxygen and Burner Tilt Angle

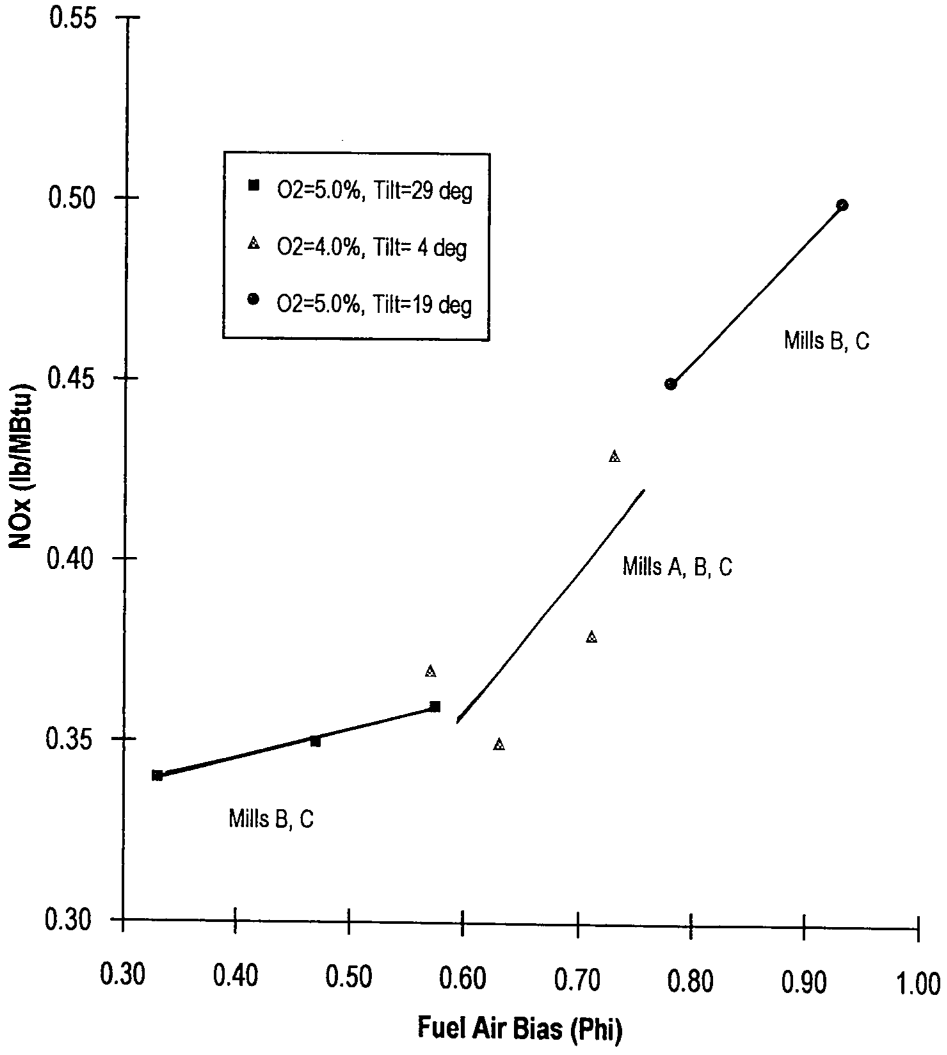

Figure 2.14

45 MW NOx vs. Fuel Air Bias at Constant Economizer Oxygen, Burner Tilt Angle, and Mill Loading Pattern

OBSTACLES TO OPTIMIZATION

A primary obstacle in reducing NOx emissions while maintaining acceptable heat rate values is that these two parameters are inversely related. Simply stated, a low heat rate operating mode results in higher NOx emissions and as boiler parameters are adjusted to lower NOx, heat rate increases. The causes and effects of this behavior are reviewed in this section.

NOx and Unit Heat Rate Interaction

Full load, low NOx furnace operation impacts unit performance through its effects on unburned carbon (also referred to as loss-on-ignition, or LOI), sensible enthalpy loss out of the stack, and steam temperatures. LOI and enthalpy loss are directly affected by the amount of oxygen present in the boiler. Data collected during testing indicated that a minimum heat rate occurs at 2.6% economizer O_2 but it is not very sensitive to changes in oxygen levels between 2% and 3%. However, sharp increases in LOI and stack losses occur at oxygen levels below 2%, along with decreased steam temperatures resulting in higher heat rate values. Increased LOI reflects poor combustion which can cause high rates of waterwall tube wastage, an increased risk of fires in ash hoppers, and poor electrostatic precipitator performance [4]. As the unit heat rate increased at lower O_2 levels, the NOx emission levels decreased. This decrease in NOx is linear with respect to economizer oxygen, as described earlier, therefore, the lower the O_2, the easier it is to comply with NOx emission regulations. Steam temperatures are the controlling factor with unit heat rate at Potomac River, with the burner tilt angle and mill loading patterns having the largest impact on steam temperatures. Higher steam temperatures are achieved by raising the tilt angle high above the horizontal, but NOx levels rise as well when the tilt goes above +10° and below 0°.

At 45 MW operation,. the economizer oxygen level is sufficiently high so LOI was never greater than six percent during testing. Steam temperatures again dictated the unit's heat rate performance and therefore, the burner tilt angle played a greater role in the heat rate values generated. The distribution of secondary air became important at the lower load levels and two modes of low NOx operation were identified. The first was relatively low burner tilt angle with a small degree of overfire air (α). The second mode involved raising the burner tilt angle and increasing the overfire air (increasing α). Both methods achieved low NOx compliance but the latter method resulted in better unit thermal performance [4].

Boiler Cleanliness

One parameter that was present in all tests but is not adequately accounted for is boiler cleanliness. As combustion continues over a period of time, residual particulate matter adheres to the waterwall tubes of the boiler, causing deposits of slag. The amount and distribution of slag is dependent upon the type of coal, boiler design, and the operating conditions during the combustion process. The boiler is cleaned of slag by blowing high pressure air or steam through certain sections of the boiler (soot blowing) at various times during operation.

As slagging occurs, NOx emissions increases due to the reduced radiative heat transfer from the fireball to the waterwall tubes, causing the flame temperature to rise which creates more thermal NOx. This makes NOx compliance difficult to achieve and maintain at higher loads where furnace gas temperatures are already in the 1600-1800°K range. For this reason, slagging is an important consideration for long term operation. The effect of slagging on unit heat rate is more complex. Boiler type is a critical factor, such as whether the boiler is a drum or supercritical design.

Observations made at Potomac River Unit 4 indicate that as slagging increases, steam temperatures rise along with NOx. Scatter that appeared in the data for certain parameters is attributed to the amount of slag present in the boiler and its effect on steam temperatures rather than to any systematic variation in controllable parameters. This is illustrated in Figure 2.15 where at low NOx conditions, heat rate has a high degree of scatter with little or no change in controllable parameters. Slagging effects were not incorporated into this study and research on quantifying and modeling the buildup of slag within utility boilers is currently being conducted. Once slagging and

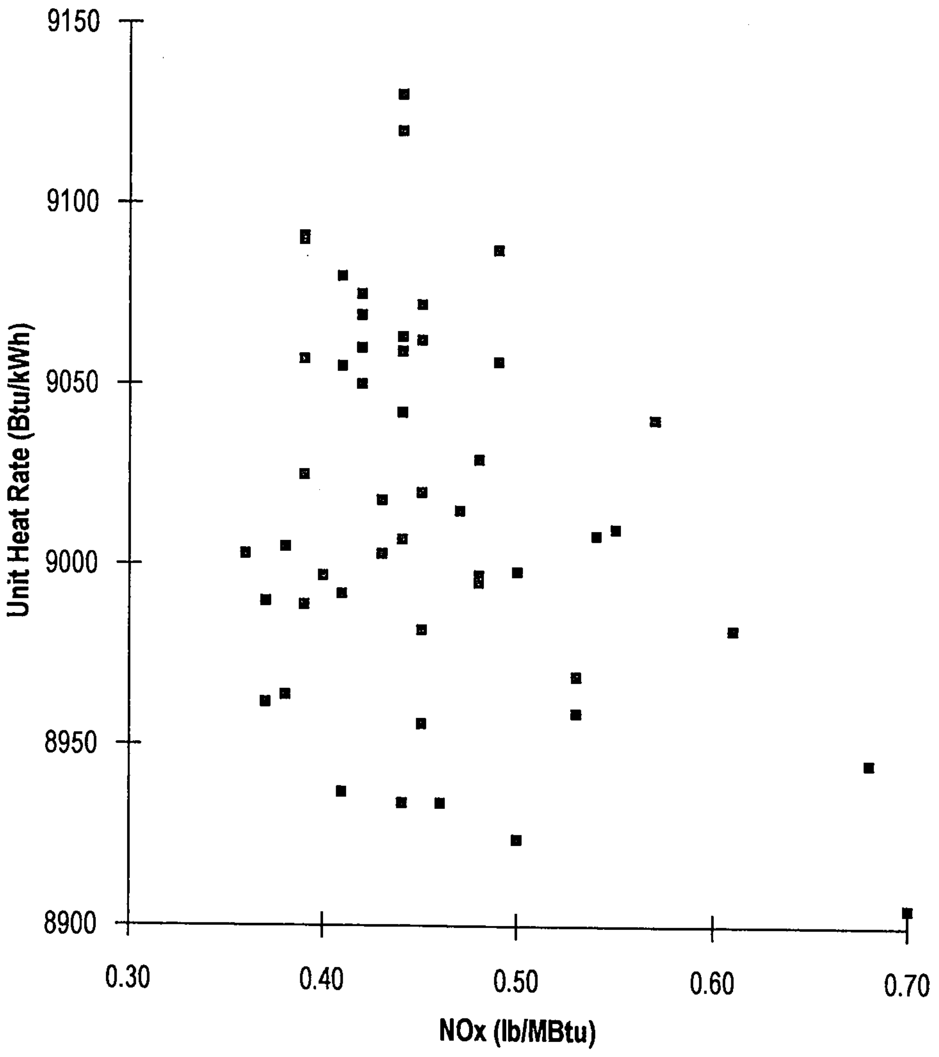

Figure 2.15

Full Load Unit Heat Rate vs. NOx

its effects are properly understood, its properties will greatly clarify the NOx/heat rate optimization process resulting in a much improved predictive models.

The NOx vs. Heat Rate Optimization Problem

As mentioned previously, at Potomac River Unit 4, it is easier to achieve compliance with Title IV of the Clean Air Act at lower loads because of the greater flexibility in operation. The full load case is the most difficult to achieve the regulatory 0.38 (lb./MBtu) NOx emissions rate without substantial heat rate penalties. In order to meet the regulations, unit operations will need to be optimized over the entire load range and during load transitions.

As the data in this section show, finding the best set of operating conditions is not easily accomplished. The methods used by D'Agostini et al. [4] included parametric tests, data collection, and statistical analysis to find the best operating conditions. The statistical analysis is very time consuming as different non-linear functions must be tested to find a mathematical model that predicts NOx emissions at different load levels. Table 2.1 details the results of the work done by D'Agostini et al. for the 105 MW and 45 MW load levels. Table 2.2 lists the resulting correlations for NOx for both load levels. This analysis did not consider heat rate and therefore additional testing would be required to improve heat rate within the low NOx operating spectrum. Additionally, if any substantial changes are made to the boiler hardware, the entire process must be repeated.

Table 2.1

Low-NOx Operating Parameters for 105 MWand 45 MW

Parameter	Low NOx 45 MW	Low-NOx 105 MW
Economizer 02 (%)	5.0	1.6
Burner Tilt Angle (degrees)	19.0	7.0
Mills in Operation	B and C	A, B, C, and D
Coal Loading to Mills	Even Loading	Even Loading
Auxiliary Air Dampers (#1-9)	4, 3, 2, 1, 1 (Bias = 0.667)	4, 4, 2, 2, 2 (Bias = 0.50)
Fuel-Air Dampers (#2-8)	1, 2, 2, 1	4, 4, 4, 4
Expected NOx (lb./MBtu)	0.35	0.39

Potomac River needs to reach NOx compliance by the CAAA Phase I deadline of January 1995. Their options are either to install costly low NOx burners or to use boiler tuning and advanced control methods to achieve their goals. They have opted for the boiler tuning method and the Energy Research Center is advancing this approach by designing, testing, and implementing a boiler tuning and optimization software package. The software will use a combination of expert system and neural network artificial intelligence methods. This thesis details the development of the neural network and optimization portion of this study. An outline of the solution process follows, and subsequent chapters detail the fundamental parts of the study. Finally, the actual design, testing, and analysis of this neural network optimization approach is discussed along with recommendations for future use and improvements.

Table 2.2

Summary of NOx Correlations for 105 MW and 45 MW

Parameter	Symbol	Units
Economizer Oxygen Level	O_2	%
Burner Tilt Angle	θ	degrees
Mill Bias	β	None
Auxiliary Air Bias	α	None
Fuel Air Bias	ϕ	None

Unit Load	NOx Correlation (lb./MBtu)	Std. Error
45 MW	$NOx = 0.416+(0.00732*O_2^2)+(0.679*\alpha*\beta)-(0.0642*O_2)-(0.000104*\theta^2)+(0.870*\beta^2)-(0.236*\alpha)+(0.1403*\phi)+(0.001920*\theta*O_2)$	0.027
105 MW	$NOx = 0.182+(0.128*O_2)+(3.44\times10^{-4}*\theta^2)-(0.00653*\theta)+(0.00192*O_2*\theta)$	0.015

OVERVIEW OF THE SOFTWARE SOLUTION

The need for a fast and accurate method to achieve CAAA compliance during on line operation is obvious. The correlations in Table 2.2, however, require many hours of statistical trial and error curve fitting to find a model with a reasonable standard error. A solution to this dilemma involves the use of artificial intelligence in the form of neural networks to learn the boiler operating characteristics. Once the boiler model is derived, mathematical optimization can be applied to the resulting equations from the neural network model. This method is much faster than statistical analysis and the entire operation can occur without user interaction. This approach is one part of an overall NOx and performance software program being designed by the Lehigh University Energy Research Center (ERC). The other main component was an expert system. A brief overview of the software follows, along with a description of how the expert system and the neural network/optimization algorithms interact.

Two situations are of concern to the power plant engineer. The first arises when something within the boiler causes the NOx emissions to increase substantially. Such an event might be caused by a mechanical malfunction, a drift in instrument calibration, a change in coal quality, or a change in the control settings. When this occurs, someone at the plant will need to determine the reason for the increase in NOx so that corrective measures can be undertaken. The second occurs when substantial operating or boiler hardware changes occur and the boiler must be retuned back to a low NOx operating region. This thesis deals primarily with the second scenario although some references to the first are inferred.

With the approach developed by the ERC, when a boiler must be retuned, the plant engineers and operators follow a test plan dictated by the expert system. Similar to a doctor asking a patient questions about certain symptoms in determining an illness, the expert system algorithm steps through the details of how a system is operating and what the system responses are to certain changes in operating conditions. For Potomac River, each change in operating parameters and the system response is stored as input for the neural network. After all tests are complete, the neural network learns the inter-relations of the input parameters to the system response. After sufficient time, the neural network can predict the system response to arbitrary inputs within a very small standard error tolerance. How the neural network accomplishes this task is detailed in Chapter 3.

After learning and deducing a final model for the new boiler operation, the equations generated by the neural networks are passed into a computerized optimization routine. Here the equations undergo a minimization technique (covered in Chapter 4) to find the best heat rate that the boiler can attain while still in compliance with NOx emissions. These results are passed back to the user as guidance for how the boiler parameters can be set for low-NOx operation. What it is the user wants to optimize and how many parameters are to be varied dictate the amount of time required to gather the data for training and optimizing the neural network. Data requirements and expert system interaction are described in Chapter 3. The simplified flow chart in Figure 2.16 shows the progression of how this method achieves its goal of NOx compliance without substantial losses in unit heat rate.

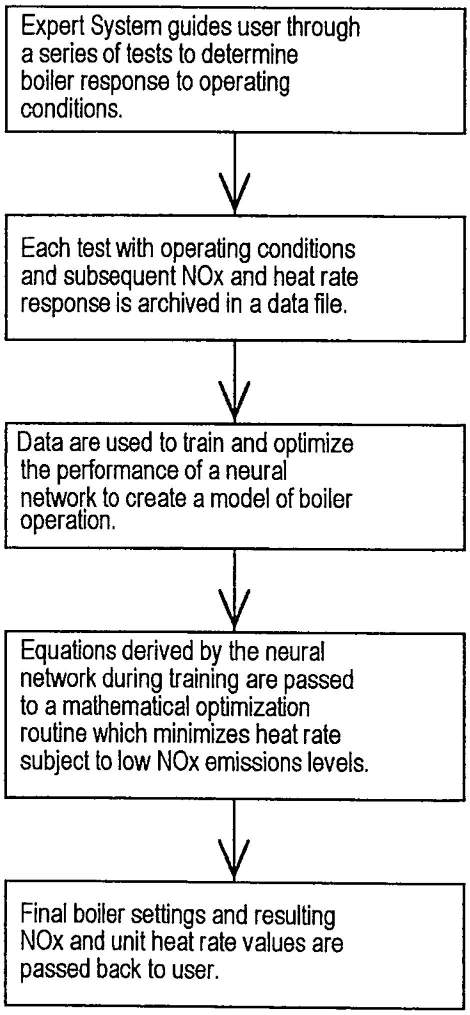
Expert System guides user through a series of tests to determine boiler response to operating conditions.
Each test with operating conditions and subsequent NOx and heat rate response is archived in a data file.
Data are used to train and optimize the performance of a neural network to create a model of boiler operation.
Equations derived by the neural network during training are passed to a mathematical optimization routine which minimizes heat rate subject to low NOx emissions levels.
Final boiler settings and resulting NOx and unit heat rate values are passed back to user.

Figure (2.16)

Flowchart of NOx Control Software Operation

CHAPTER 3

NEURAL NETWORKS AND BACKPROPAGATION

Neural networks are a form of artificial intelligence that attempt to model the functional capabilities of the human brain. The ability to process and respond to millions of different types of data simultaneously makes the brain a benchmark for computational sciences. When broken down to its simplest form, the human brain and nervous system are merely a mass of cellular units called neurons. Each neuron is a microprocessor which receives and combines input signals from other neurons through its dendrites. If the combined signal is strong enough, the neuron will activate and fire an output response signal to other neurons via the axon. The response signal is a measurable small electrical impulse generated by chemical reactions within the neuron and its strength depends upon the input signals. If the same combination of input signals is received over a period of time, the same response will occur and this comprises learning and memory. By combining the actions of several neurons together, exponential gains in processing and learning input data can occur, even without prior knowledge of what the data are or represent. This is the inherent power of modeling the human brain, composed of millions of densely packed neurons, into a computer processing methodology.

DEVELOPMENT AND APPLICATIONS OF NEURAL NETWORKS

Warren McCullough and Walter Pitts initiated the realm of artificial intelligence with the watershed paper "A Logical Calculus of Ideas Imminent in Nervous Activity" in 1943. This spawned the development of expert systems and neural networks. Several attempts to recreate the function of the human brain evolved through the 1950s and 1960s, each achieved notoriety but still had some form of computational shortcoming. Research centered around the artificial neuron's structure, learning process, and ability to deal with noisy input data. In 1982, after a lengthy lull in research funding and efforts, John Hopfield designed a model comprised of several interconnected processing elements (artificial neurons) that sought an energy minimum in response to input data. The model represented neuron function as a thresholding operation and neuron memory as information stored in the interconnections between the artificial neurons. The model's success spurred new efforts in neural network research. Currently there are dozens of derivatives of Hopfield's model and as many learning paradigms, each with particular real world applications. Some of the applications are: language processing, data compression, image and character recognition, combinatorial problems, pattern recognition, financial and economic modeling, functional synthesis, and signal processing. The last item is of interest to the boiler tuning problem as prediction, system modeling, and noise filtering are subsets of this group.

The foremost difference between neural networks and traditional computing is the neural network's ability to learn by example. Rather than search for a mathematical function that fits the underlying relations within a set of data, neural networks generate their own functions by processing data samples. No physical characteristics of the data are required, just objective numerical or visual data. Neural networks require little human expertise; the same type of neural network algorithm will work for many different systems. The network, comprised of processing elements that are interconnected, similar to biological neurons interconnected by axons, is shown a set of inputs and the known output response to those inputs. The importance, or "weight", of these connections are changed as the network is presented more data samples. The amount of change to the connection weight is governed by the difference between the network-predicted response and the known output response. This difference is processed by a "learning rule" which makes an adjustment to all of the connection weights within the network. The change in the weight values is such that the next time that particular set of data is shown to the network, the predicted response is closer to the known response. As the network is continually shown these examples, it configures itself to achieve a desired input to output mapping that incorporates all of the data samples presented to it.

Neural networks also outperform traditional computing methods in the way they store information about the network model's response to inputs and outputs. Neural networks use distributed associative memory found in the processing element connection weights. The value of the weights quantify the network's current state of knowledge. An item of knowledge, represented by a learning example, is shared across all connection weights along with the knowledge acquired from previous learning. This method of knowledge retention allows the network to respond to an example it has never seen through generalizing the knowledge it has accumulated previously. Once a network has been trained, it can be used to simulate the actual system; that is, for any set of inputs, it can produce a set of outputs similar to those that the actual system would have produced.

Neural networks do not require outside rules or expertise as in a rule-based expert system. Under certain output conditions, expert systems may have an overwhelming number of underlying rules to distinguish system behavior. Furthermore, where an expert can be diverted from proper analysis by noise in the data being analyzed, a neural network is less subject to such behavior. Finally, whereas traditional computing systems are rendered useless by any type of memory loss, neural networks are fault tolerant. For example, if an input signal fails or is registering noisy data, the neural network can continue operation, albeit not as effectively. This is because not all of that input's data response is stored in one location but rather across several in lesser magnitude. Such graceful degradation of performance is optimal for on-line, real world applications where input signals can malfunction and sudden shutdown of the model is cataclysmic [20].

BACKPROPAGATION: FUNDAMENTAL EQUATIONS AND OPERATION

A backpropagation network is named for the means by which it accounts for errors between the network's predicted output value(s) and the actual known output value(s). It is within the family of feed-forward networks which have the following operating characteristics: 1) data is read into an input layer; 2) these data are processed through a single or a series (up to three) of middle layers called hidden layers; 3) an output layer uses the information passed into it to predict what the output response to the inputs should be. A backpropagation network assumes that all processing elements are somewhat to blame for an erroneous response and this error is spread evenly, or propagated, back through the network from the output layer to the input layer.

The type of backpropagation model used in this study, and the most typical, consists of three fully interconnected layers. These layers are composed of processing elements, called nodes, which perform the mathematical functions that result in an output prediction. The input layer is primarily for feeding the data into the hidden layer and has one node for each input into the network. The hidden layer can be comprised of any number of nodes and therefore computational experiments must be conducted to find the optimal number of nodes for the hidden layer. It is the hidden layer that provides the network with the capability to analyze non-linear function mappings. If the input layer were connected directly to the output layer then only linear relationships between input values and subsequent output values would be determined. In some instances this is all that a problem requires. Figure 3.1 details a typical backpropagation network similar to the ones used in this study.

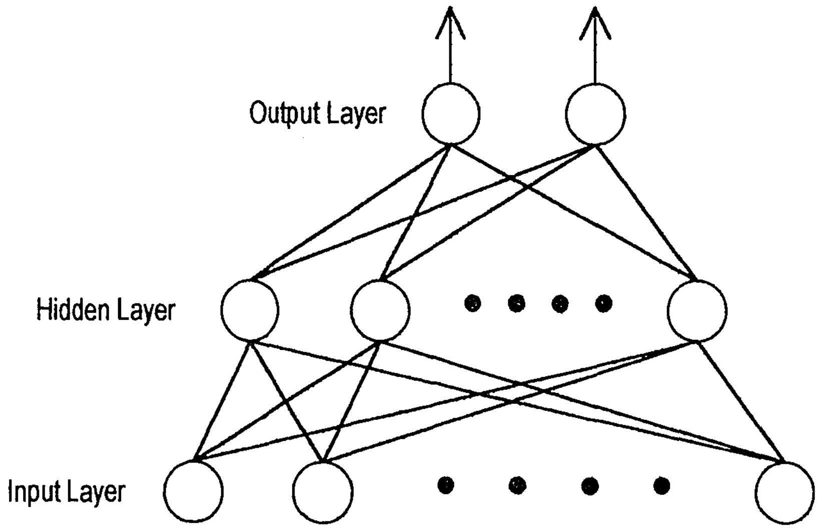

Figure 3.1

Typical Backpropagation Network Architecture

To understand how the neural network learns the relationships between a system's inputs and corresponding outputs, the transfer of numerical information between the network processing elements (PEs) must be understood. A backpropagation network PE transfers its input as:

$$x_j^{[s]} = f\left(I_j^{[s]}\right) \qquad \textbf{(3.1)}$$

and

$$I_j^{[s]} = \left(\sum_i \left(w_{ji}^{[s]} * x_i^{[s-1]}\right)\right) \qquad \textbf{(3.2)}$$

where: $x_j^{[s]}$ = the current output value of the jth neuron in the current layer [s]

$I_j^{[s]}$ = the summation of weighted inputs to the jth neuron in layer [s]

$w_{ji}^{[s]}$ = the connection weight to the jth neuron in layer [s] from the ith neuron in the previous [s-1] layer

f = the transfer function which scales the value of $I_j[s]$ to keep it proportional to the other PE output values

To fully understand these equations, a closer look at a typical processing element (PE) is required and Figure 3.2 provides this detail.

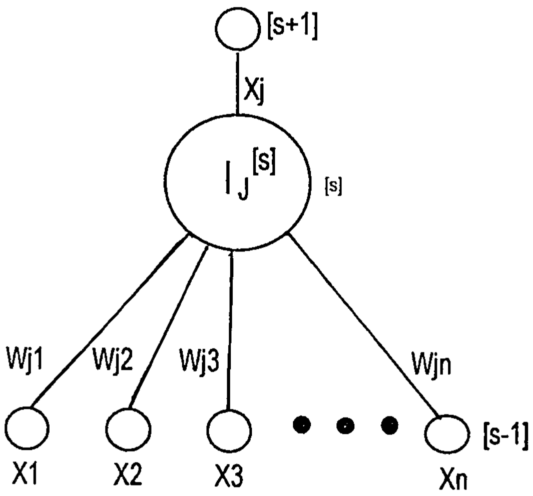

Figure 3.2

Typical Processing Element (PE)

Referencing Figure 3.1, an input data vector, **i**, is introduced at the input layer of the network and the desired (actual) output vector, **d** is supplied by the user. These vectors are composed of input and k output data components which represent a single data point from a set of parametric data. Let **o** denote the predicted output vector produced by the network with its current set of weights. For each output layer processing element (PE), there is a difference between the predicted output value (o) and the known, actual output value (d). This difference for each output PE is called a local error. The sum of all of these output PE local errors represents the total error present within the output layer. This total error of the neural network is termed the global error, E. This global error for the output layer is mathematically defined by different differentiable functions which are discussed later in this chapter. For now, the global error function is assumed to be:

$$E = 1/2 * \sum_{k} \left((d_k - o_k)^2 \right) \tag{3.3}$$

where: d_k = the desired, or actual, output value for the kth output PE
o_k = the observed, or predicted, output value for the kth output PE

This is the standard error function and is quite common to backpropagation neural networks. Therefore, the local errors of the output PEs must be minimized to minimize the global error. The fastest means to accomplish this is by applying a negative gradient to the global error:

$$e_j^{[s]} = -\frac{\partial E}{\partial I_j^{[s]}} \tag{3.4}$$

where: $e_j^{[s]}$ = the current local error at jth PE within the current [s] layer
$\partial I_j^{[s]}$ = the partial derivative of the sum of the product of the inputs and connection weights entering the current [s] layer (Eq. 3.2 and Fig. 3.2))

By minimizing the local error at each of the output PEs, the global error for the output layer, and thus the network, is minimized. To minimize the local error of the output PEs, the local error of all preceding PEs within the hidden layer(s) must be minimized as well. The only way to minimize all of these local errors is to change, or update, the variable connection weights between these PEs. This is a fundamental concept of backpropagation.

A network has a global error comprised of local errors within the output layer. To minimize the output local errors, the preceding hidden PE local errors must be minimized. Thus, the global error is split among the output PEs which, in turn, is split among the PEs of the hidden layer(s), each in succession from the output layer to the input layer. This is how the current global error is propagated back through the network in discrete values of local errors. These local errors are reduced only by the updating (changing) of the connection weights, so backpropagation simply stands for connection weight updating. Over the course of training, these weights are changed such that the local errors decrease for all PEs which causes the global error to decrease. This exemplifies the neural network learning process.

The question of how the local error is minimized at the hidden layer PEs still remains. The network layer notation in Figure 3.1 shows that the current layer being examined is defined as [s]. The layers above and below this layer are denoted as [s+1] and [s-1], respectively. The input layer is always the bottom layer and therefore it can have no [s-1] layer beneath it. Conversely, the output layer is always the final layer and cannot have an [s+1] layer above it. If a network has two hidden layers and the first of those layers is being examined, it is the [s] layer. The input layer would be the [s-1] layer and the second hidden layer would be the [s+1] layer. The same logic applies to using the second hidden layer as the [s] layer. The output layer is the [s+1] layer and the first hidden layer is the [s-1] layer.

The first step to minimizing the network global error is to calculate the output layer PE local errors. Starting with Eq. 3.4, each output PE's local error is found by:

$$e_k^{(o)} = -\frac{\partial E}{\partial I_k^{(o)}}$$

$$e_k^{(o)} = \frac{-\partial E}{\partial o_k} * \frac{\partial o_k}{\partial I_k} \qquad \textbf{(3.5)}$$

$$e_k^{(o)} = (d_k - o_k) * f'(I_k)$$

where: k = the output layer PE index
d_k = the desired (actual) output value
o_k = the observed (predicted) output value
I_k = the summed products of inputs and connection weights (Eq. 3.2)

The f' is the derivative of the transfer function which scales the local error at the kth PE in the output layer so that all local errors within the output layer stay proportional to one another. These transfer functions are detailed later in this chapter.

Next, the hidden layer PE local errors must be calculated. Substituting Eqs. 3.2 and 3.3 into Eq. 3.4 and applying the chain rule twice, the general mathematical form for calculating local errors at any of the hidden layer PEs is:

$$e_j^{[S]} = f'\left(I_j^{[S]}\right) * \sum_k \left(e_k^{[S+1]} * w_{kj}^{[S+1]}\right) \qquad \textbf{(3.6)}$$

where: $e_j^{[s]}$ = the local error of the jth PE in layer [s]
$I_j^{[s]}$ = the sum of the weighted inputs into the jth PE in layer [s] (Eq, 3.2)
$e_k^{[s+1]}$ = the local error values of the PEs in the above [s+1] layer
$w_{kj}^{[s+1]}$ = the connection weights to the kth PE in layer [s+1] from the jth PE in layer [s]

The reader is reminded that this equation is applied from the output layer down towards the input layer. The hidden layer immediately below the output layer has its PE local errors calculated next, using the output layer local errors and connection weights. This continues until the hidden layer immediately above the input layer has its PE local errors calculated last. In this instance, the f' is the derivative of the transfer function which scales the local error at the jth PE in the current [s] layer so that all local errors within the [s] layer stay proportional to one another.

With the local errors for all of the PEs above the input layer calculated, the next step is to change the connection weights entering each PE so as to reduce the amount of local error for that PE. The input data components are constant and since each subsequent layer's input is dependent upon the previous layer's output, the only variable to adjust is the connection weight. Given a current set of network weights, $w_{ij}^{[s]}$, the network training algorithm must know whether to increase or decrease the weight values in order to decrease the local error. This is accomplished by using:

$$\Delta w_{ji}^{[S]} = -\mathrm{lcoef} * \left(\frac{\partial E}{\partial w_{ji}^{[S]}} \right) \tag{3.7}$$

where: w_{ji} = the connection weight to the jth PE in layer [s] from the ith PE in layer [s-1]
lcoef = a user-defined, scalar learning coefficient which is detailed later in this chapter

Basically, the backpropagation algorithm changes each weight according to the magnitude and direction of the negative gradient on the global error surface. The partial derivative of the global error with respect to any connection weight within the network is found by:

$$\frac{\partial E}{\partial w_{ji}^{[S]}} = \left(\frac{\partial E}{\partial I_j^{[S]}} \right) * \left(\frac{\partial I_j^{[S]}}{\partial w_{ji}^{[S]}} \right) \tag{3.8}$$

This is merely the product of the local error equation, Eq. 3.4, and partial change to the ith PE input (Eq. 3.2) to the jth PE in layer [s] with respect to the connection weight. Rewriting this in simpler notation:

$$\frac{\partial E}{\partial w_{ji}^{[s]}} = -e_j^{[s]} * x_i^{[s-1]} \tag{3.9}$$

where $x_i^{[s-1]}$ = the input from the ith PE in the [s-1] layer below the current [s] layer

Combining Eqs. 3.7 and 3.9, the weight update is mathematically defined as:

$$\Delta w_{ji}^{[S]} = lcoef * e_j^{[S]} * x_i^{[S-1]} \tag{3.10}$$

Therefore, the amount of change for any connection weight entering the jth PE in layer [s] is simply the product of that PE's local error, the input value from the ith PE in the [s-1] layer below it, and a constant learning coefficient that scales the magnitude of change so as to keep all connection weight changes proportional. Each time a new data vector is presented to the network, the backpropagation algorithm calculates the change in each of the connection weights. This change is added to the current weight value to reduce each PE's local error in an effort to minimize the network global error. This is how the neural network learns the numerical relationships between a system's causal inputs and corresponding outputs.

Before discussing the various user-determined components of the backpropagation neural network, a short summary of the algorithm is provided. Given an input vector, $\underline{\mathbf{i}}$, and a desired output vector, $\underline{\mathbf{d}}$, the following steps occur:

1. Present $\underline{\mathbf{i}}$ to the input layer of the network and propagate it through to the output layer to obtain an output vector, $\underline{\mathbf{o}}$. As this information propagates through the network it will also set all of the summed inputs, I_j and output states X_j for each processing element in the network.

2. For each processing element in the output layer, calculate the scaled local error as given in (3.5) and calculate the change in weight using (3.10).

3. For each layer, [s], starting at the layer below the output layer and ending above the input layer (or simply the hidden layer if only one hidden layer is used), and for each node within the layer, [s], calculate the scaled local error as given in (3.6) and then calculate the change in weight as determined by (3.10).

4. Update all weights in the network by adding the current weight changes to the corresponding previous weights.

This process is repeated for every input/output data vector supplied by the user and this is how the neural network learns the interrelationships between the input and the output values. This methodology is tailored to any specific problem or system being modeled by using the following neural network operational considerations.

OPERATIONAL CONSIDERATIONS FOR THE BACKPROPAGATION NETWORK

As described in Chapter 2, a neural network software package, NeuralWare Explorer, was used in this study. There are several parameters that can be selected within the software that control the learning process for the backpropagation network. Although the complete list is rather extensive, only the fundamental parameters that were central to the development of the current boiler model will be discussed. A brief description of what each parameter does within the network and under what conditions it would be used is provided along with its mathematical definition.

Learning Coefficient (Rates)

Used earlier in the error backpropagation equations (**lcoef**), this value controls the step size for changing the connection weights after processing the data vector. If this value is too large, the network error may diverge and increase. If too small, the network error will take too long to converge to within the set tolerance. A "rule-of-thumb" is that the learning rate should be between zero and one, and the hidden layer should have a learning rate twice that of the output layer [14]. Optimal learning rates, which result in maximum network performance, cause the network to have a smoothly decaying output error function. Furthermore, the connection weights for each layer must change from their initial values to optimal values in an even and orderly fashion. If the weights for one layer increase faster than another layer, the network assumes a fixed state and will not achieve its optimal condition from the learning process. Either problem can be alleviated by proportionally decreasing the learning coefficient values for all layers, obtaining data that are more random for the training set, or checking the scaling of the input values. This remedy was based on thousands of tests conducted by the creators of the neural network software used in this study [14].

Momentum Term

When using a gradient descent algorithm, the learning coefficient controls how fast learning will occur. To prevent excessive learning time due to a small learning coefficient, a momentum term, α, is introduced to decrease learning time without the risk of divergence. Equation (3.10) is modified so that a portion of the previous weight change is fed through to the current weight change via:

$$\Delta w_{ji}^{[S]} = \text{lcoef} * e_j^{[S]} * x_i^{[S-1]} + (\alpha * \Delta w_{ji}^{[S]}) \qquad \textbf{(3.11)}$$

This momentum term causes general trends to be reinforced and oscillatory behavior to be dampened out. This allows for faster learning with a low learning coefficient by forcing the network out of local minima, without causing extreme changes in the weight updates. The momentum term is usually larger than the respective layer learning rate, but the proportionality rules regarding increasing and decreasing their values are the same as the learning coefficients.

Cumulative Weight Updates

Another method employed to decrease learning time is to update the connection weights only after a set number of data vector presentations. This is referred to as cumulative backpropagation since the weight changes are accumulated until a certain number of data vectors is presented. This given number of presentations is referred to as the epoch size and can correspond to either one vector (incremental learning), all vectors within the data set (batch learning), or any number in between. In general, the epoch size should be set to accommodate a batch of training examples large enough to represent the input data population [16]. Ultimately, once the best epoch size is determined, it represents the fundamental "frequency" of certain dominant components of the underlying noise. The epoch size can also be considered as a composite error function rather than individual pattern dependent error functions. A composite error function is the sum of individual error functions; consequently its partial derivative with respect to any connection weight is the sum of the partial derivatives of the corresponding individual error functions. The weight changes are dictated by this partial derivative so accumulating the delta weights over the entire set of data patterns is the same as using backpropagation with a composite error function.

If the epoch size is not too large, cumulative weight updates lead to faster convergence than individual updates which reduce the error function only for a particular data vector. The incremental update method may increase other component error functions whereas a global update will always work towards reducing the overall error function. Unfortunately, when using a cumulative approach to weight updates, many more calculations are required to achieve a single update and the benefit of using an overall error function may be lost if the epoch is too large.

Error Functions

The error function used in (3.3) is proportional to the square of the Euclidean distance between the desired output and the actual predicted output for a particular set of inputs. Any error function whose derivatives can be calculated at the output layer can be substituted for this standard error function. The two other common error functions for backpropagation are:

Cubic Error: $$E = \frac{1}{3} * \sum_k \left|(d_k - o_k)\right|^3 \quad \textbf{(3.12)}$$

and

Quadratic Error: $$E = \frac{1}{4} * \sum_k \left|(d_k - o_k)\right|^4 \quad \textbf{(3.13)}$$

These reduce to local errors of:

$$e_k^{(0)} = (d_k - o_k)^2 * f'(I_k) \quad \textbf{(3.14)}$$

$$e_k^{(0)} = (d_k - o_k)^3 * f'(I_k) \quad \textbf{(3.15)}$$

Transfer Functions

The transfer function performs a given mathematical operation on the summed products of connection weights and input values entering the processing element as seen in Figure (3.2). Any smooth and differentiable function can be used as a transfer function. The best functions provide a filtering effect on the data being passed through it by limiting the range of possible output responses. The choice of the transfer function is determined by the nature of the data and what the network is trying to learn. A transfer function is applied at each hidden layer and at the output layer. All nodes within any given layer have the same transfer function. The most common transfer functions of backpropagation are:

Sigmoid: $$f(z) = \frac{1}{1+e^{-z}} \tag{3.16}$$

Hyperbolic Tangent: $$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.17}$$

Sine: $$f(z) = \sin(z) \tag{3.18}$$

Linear: $$f(z) = z \tag{3.19}$$

The reader is reminded that: $z = \left(I_j^{[S]}\right)$, as depicted in Figure 3.2 and Eq. 3.2.

The sigmoid generates a smooth version of a step function between zero and one while the hyperbolic tangent is a bipolar version of the sigmoid with a range of (-1,+1). If the problem at hand is learning "average" system behavior, the sigmoid transfer is best. If finding system deviations from average behavior is desired, then the Tanh is the better choice. The sine function gives a smooth oscillation between -1 and +1 instead of a step function but its use requires knowledge about underlying relationships within the data. The derivatives of these functions are easily calculated and the resulting error propagation equations that result are:

Sigmoid: $$e_j^{[S]} = x_j^{[S]} * \left(1.0 - x_j^{[S]}\right) * \sum_k \left(e_k^{[S+1]} * w_{kj}^{[S+1]}\right) \quad \textbf{(3.20)}$$

Tanh: $$e_j^{[S]} = \left(1.0 + x_j^{[S]}\right) * \left(1.0 - x_j^{[S]}\right) * \sum_k \left(e_k^{[S+1]} * w_{kj}^{[S+1]}\right) \quad \textbf{(3.21)}$$

Sinc: $$e_j^{[S]} = \cos(I_j^{[S]}) * \sum_k \left(e_k^{[S+1]} * w_{kj}^{[S+1]}\right) \quad \textbf{(3.22)}$$

Linear: $$e_j^{[S]} = \sum_k (d_k - o_k) \quad \textbf{(3.23)}$$

The linear transfer is used primarily for the scaling of output layer PEs. If the neural network being used contains hidden layers, the linear transfer function is rarely used to scale the output of the hidden PEs. This is because the use of a linear transfer annuls the non-linear benefits of using a hidden layer. Unless the network has no hidden layers and a purely linear relation between inputs and outputs exists, then the linear transfer function is applied outside of the output PE layer.

The learning procedure which uses Sigmoid or Tanh transfer functions can be thought of as synthesizing a continuous function y = g(x), by showing it a discrete set of ($\underline{x}$, y) pairs. After being trained on data vectors in this form, when the network is presented a previously unseen pattern it will perform a non-linear interpolation and produce a reasonable function value. In the case of using a sine function, the learning procedure seems to perform a mode decomposition such that it finds the important frequency components of the function described by the inputs and outputs. This underlying mode decomposition has been termed a "Generalized Fourier Analysis" [10]. The best first choice, for backpropagation, is the Tanh transfer function. [14].

Learning Rule

Learning is the process of changing the connection weights to achieve desired responses to input data. The learning rule for each layer in the network applies to all nodes within that layer. The learning rules available for backpropagation networks are as follows:

Delta Rule. The error between the actual and predicted output is transformed by the derivative of the transfer function. The actual weight update equations for this rule are :

$$\begin{aligned} w_{ij}' &= w_{ij} + C_1 * e_i * x_{ij} + C_2 * m_{ij} \quad \text{and} \\ m_{ij}' &= w_{ij}' - w_{ij} \end{aligned} \tag{3.24}$$

where: C1 = learning coefficient
C2 = momentum coefficient
w′= the updated weight vector from jth input to the ith node in the layer
x = inputs to the connection of the jth input to the ith node in the layer
m = the memory of the last change in weight value
m′= the difference between the old weight and the updated weight

The weights are changed in proportion to the error (e), and the input to that connection (x) while the momentum term is used to smooth out the weight changes to prevent large oscillations. If the training data are very ordered or structured, then the delta rule is not a good choice as it will lead to difficulties in achieving convergence.

Cumulative Delta Rule. Working with epoch sizes greater than one requires accumulated weight changes which are applied all at once. This learning rule accumulates weight changes over several examples and works well when the data are very structured and/or ordered. However, this technique has difficulty linking the learning coefficient which is calculated by dividing the learning coefficient by the square root of the epoch size. If data are noisy, then the epoch size must be tuned, but for a first guess, an epoch of 16 is suggested [14].

Summation Functions

Two general summations are prevalent in neural networks, the sum and the cumulative sum. A normalized cumulative sum exists but there are very few applications that require its use.

Sum. This is the standard summation of the products of connection weights and processing element input values represented as:

$$I_j = \sum_j w_{ij} * x_j \tag{3.25}$$

Cumulative Sum. The cumulative sum of the effective inputs is added to the prior sum of these inputs. This is symbolically written as:

$$I_j = I_{iold} + \sum_j w_{ij} * x_j \tag{3.26}$$

SUMMARY

The parameters reviewed above are the ones typically used. There are many additional techniques that can be applied to the backpropagation paradigm to fine-tune the network operation. However, as the study was conducted the functions described above were found to have the most impact on network performance.

Before discussing the subjective considerations of the neural network approach, a brief overview of the software used in this study is necessary. The neural network computing software was provided through Lehigh University under educational contract with Neural Ware, Incorporated, located in Pittsburgh, Pennsylvania. The software, NeuralWare Explorer, is a student oriented introduction to the capabilities of neural network computing. All facets of this study dealing with the neural network model were explored using this software. Although limited in capabilities compared to the advanced package, the parameters reviewed above, along with several others, were examined and tested for their effects on the boiler model. The results of these tests are covered in Chapter 5, Solution Methodology and Results. The information provided about the neural network parameters was taken in part from the reference guides that accompany the software [9, 11].

Problem Specific Considerations for Neural Network Modeling

Beyond the standard backpropagation functional components are several subjective issues which must be accounted for when implementing a neural network. The purpose of this section is to describe these issues, discuss the different approaches to each issue, and detail how the boiler optimization problem warranted the solution decisions that were made. There are several schools of thought regarding these different areas. A great deal of research by the originators of the neural network paradigm and their initial applications [10,18,21,24] has gone into validating the methods described within. However, for the sake of brevity, only the most pertinent approaches to these subjective considerations will be detailed.

Neural Network Architecture and Size.

When modeling a real world application with a neural network, a great deal of thought and experimentation are required to determine the pertinent input parameters needed to calculate appropriate output response. This can be accomplished either by having done past research on the system in question such that one knows what inputs comprise the necessary responses, or by an experimental method. This experimental method requires grouping different system inputs together, optimizing a network based on these inputs, and comparing the prediction performance of each network against each other. This was the methodology used in this study since boiler operation was such a complex system to understand and required more than basic knowledge of operating parameter interaction. One caveat which is addressed in the next section is the data requirements for different sized networks. One can easily be lured into trying to use all possible inputs to a system to get the most precise model. However, data requirements for training and testing networks increase quite rapidly with every input added to the model. This was demonstrated in this study since quality data were very limited and caused the network with all input parameters to perform much worse than the network with fewer, more well chosen input parameters.

Another architectural consideration is the number of processing elements to use in each hidden layer. The hidden layer does not correspond to any part of the actual system operation and therefore has no physical link or significance between inputs and outputs. This intermediate layer only permits the backpropagation algorithm to model non-linear functions of greater complexity, and choosing the number of nodes within the hidden layer(s) almost always involves experimentation [12]. As with the number of inputs, the number of hidden nodes affects the amount of data required to achieve an acceptable level of model performance. The governing factor in determining data requirements is the total number of connection weights that are within the network. The more weights in the network, the more data are required. Three procedures exist for determining the number of hidden units. If one does not have adequate time for experimentation, then a moderate degree of complexity can be introduced by setting the number of hidden units equal to the geometric mean of the inputs and outputs [13]:

$$n_{hid} = (n_{in} * n_{out})^{\frac{1}{2}} \tag{3.27}$$

The remaining two methods involve experimentation and these are the constructive and destructive approaches. The constructive approach starts with a network containing no hidden units. The network is trained and tested until optimal performance is achieved, which is subsequently recorded. One processing element (node) is then added to the hidden layer and the process is repeated. Once there is no change in performance between the N node network and the N+1 node network, the network with N nodes is the best choice for the network architecture. The reason is that the N+1 network is starting to "memorize" the data instead of creating a generalized model. In statistical terms, this memorization action is synonymous to overfitting a curve to a set of data.

The destructive approach starts with a network containing many hidden nodes. The hidden nodes are pruned out of the architecture in one of three ways. The first way is the most complicated and consists of many experiments. The network is trained for a given duration and then tested using both the training and testing data. Then for each unit in the hidden layer: 1) disable the unit, setting its output to zero; 2) retest the network on both the training and test sets; 3) record the results. If disabling the unit improves both the performance on the training and test sets, then leave it disabled. Use the new results for the standard to measure the effect of each succeeding unit and then continue the training process [14].

The second method is simpler but can result in reduced accuracy from the final network performance [14]. If, after training, a connection weight value is below a designated threshold then the connection is suspended but the processing element remains. This way the network complexity remains while improving the speed of learning by eliminating insignificant connections. One must be cautious as to what the tolerance should be to prevent eliminating subtle, yet significant, network interactions. Finally, a basic troubleshooting methodology can be used when evaluating network performance by using the training and testing data sets. If the test data prediction response error is much greater than the training prediction response error, then decrease the number of hidden units, diversify the training data set, or decrease the training time [15].

The object of creating a mathematical model is to balance the tradeoff between accuracy and the ability to generalize and extract reasonable responses to new data. The complexity of the neural network in relating inputs to outputs determines this. A neural network maps inputs to outputs via a sequence of weighted summations and transfer functions. This mapping is parameterized by the weights of the connections among processing elements. The learning process finds the best values for these weights. Reducing the number of connections reduces the number of parameters in the mapping, which, in turn, limits the number of relationships between input and output data. The network will then attempt to learn only the most consistent relationships, not the spurious ones (which won't carry over from training to testing). The number of hidden nodes and inputs determine the number of connections, but assuming a set number of inputs, an empirical method for selecting the optimal number of nodes in the hidden layer is applied. This method is very time and data intensive, although quite easy to use. This empirical method uses the R-Square statistic that scales network prediction accuracy from zero (worst) to one (best). This and other performance statistics are detailed in the following section of this chapter.

For a hypothetical neural network, the following results from a series of computational experiments were obtained:

Table 3.1

Sample Network Results for Optimal Performance Selection [16]

Number of Hidden Processing Elements	Training Set R-Square (0 = worst, 1 = best)	Test Set R-Square (0 = worst, 1 = best)
9	0.9381	0.7248
5	0.9135	0.7291
4	0.9037	0.7204
3	0.9107	0.7695
2	0.8821	0.7588
1	0.8500	0.6656

The neural network that used three processing nodes in the hidden layer gave the best performance combination for both the training data set and the testing data set. This result implies that the connection weights are such that the network has learned enough about the input/output relationships that an accurate prediction can be made for data it has already seen. Furthermore, the network gives a very reasonable prediction on data it has not been trained on (the test set data) and thus the network can generalize system operation. One does not want to select the architecture that performs the best in either category because that may imply that either network is overtrained (best training set performance) or may lack accuracy (best test set performance). As detailed in a following section, the difference in data requirements for each of these architectures is significant and so finding the optimal architecture is imperative, especially when vast quantities of data are not available.

Measuring Neural Network Performance

Measuring how well a system is performing is relatively straightforward. One simply calculates the percentage of correct answers obtained in a testing or operational situation and compares them to specifications determined beforehand or against a tolerance value. Neural networks require more than this simple technique and, in some instances, several performance measures are required to accurately judge and compare network accuracy. Unfortunately, the majority of literature on neural network usage does not adequately describe the methods used to determine the optimal network performance. Therefore, in this study, no straight guidelines are given, only descriptions of the best measures for this problem and why they were chosen.

The backpropagation network was the method chosen to map the boiler operating conditions to the heat rate and NOx. Like any mathematical model being compared to actual observable data, certain statistical parameters are used to gauge performance. The ones used in this study were the average difference, the average percent error, the R-Squared, and Adjusted R-Squared values.

The average difference and average percent error are two commonly used performance measures for any mathematical mapping. The average difference is also called the average error between the actual and predicted values. In equation form, this value is:

$$E_x = x - \tilde{x} \tag{3.28}$$

where x = the actual PE output value
x = the predicted PE output value

The average error is simply the sum of these differences divided by the total number of data vectors used within the test data set. The average percent error (or average relative error) over n test cases is:

$$R_x = \frac{\sum_{1}^{n}\left(\frac{x-\tilde{x}}{x}\right)*100}{n} \tag{3.29}$$

The difference between using E_X and R_X depends upon the magnitude of the actual and predicted values. As the absolute value of the actual value moves away from unity, the average relative error is a better indicator of accuracy of the approximation [17]. Since NOx values are generally between zero and one, equation (3.26) was used primarily for measuring NOx network performance and (3.27) was used for the heat rate network since heat rate magnitude is in the range of 10^3. For the overall analysis, both parameters were calculated for all networks so as to provide a final "bottom line" evaluation and to equate the predicted outputs with actual output response values.

The R-Squared and Adjusted R-Squared parameters are the most commonly used measures for comparing network performance during the process of finding the optimal settings for the best prediction capabilities. They are combinations of the average sum-squared error, the normalized sum-squared error, and the least mean squared error. Recall that the goal of training a backpropagation network is to minimize the average sum-squared error between the predicted and actual output values. The different error functions were given as squared, cubic, and quadratic and are equations (3.3), (3.12), and (3.13), respectively. Focusing on networks using the squared error function, there are different opinions on the best means of determining performance. The least mean squared approach outlined by Widrow [18] is:

$$E = \frac{\sum_{k=1}^{N}(x-\tilde{x})^2}{N} \tag{3.30}$$

whereas Rumelhart and McClellan's method, as described in [19] is:

$$E = \frac{\frac{1}{2}\sum_{k=1}^{N}(x-\tilde{x})^2}{N} \tag{3.31}$$

The use of the 1/2 multiplier is heavily debated but most references submit to the fact that using the multiplier is dependent upon the neural network tool being used and how the error term is implemented in the backpropagation scheme. Experiments conducted for this thesis revealed that the 1/2 term had little or no effect on the training of different networks, therefore, equation (3.28) was used throughout this study. An important caveat is that (3.30) and (3.31) are for single output networks only. When comparing different networks with two or more outputs, these performance parameters will give skewed results. One inherent problem with either error value is that it is dependent upon the variances in the desired (actual) output values. An error parameters that is independent of these variances is the normalized error. Variance is defined as the average of the squared deviations from the mean. It is also called the mean square and can be either a population variance or a sample variance. In neural network terms, there is little difference in which version is used, so the population variance is the choice for this derivation [19]. The population (target) variance for a single output network is represented as σ^2, the mean of the actual output values is μ, and the actual output value is α. For a test set of N vectors, the variance is:

$$\sigma^2 = \frac{\sum_{N}(\alpha-\mu)^2}{N} \tag{3.32}$$

The standard deviation is simply the square root of the variance, also known as the root mean square. An error measure that removes the effects of the target value variance and yields an error value between zero and one for all networks regardless of configuration is the R-Square Coefficient [19]:

$$R^2 = 1 - \left[\frac{\sum_N (\alpha - \rho)^2}{\sum_N (\alpha - \mu)^2} \right] \qquad \textbf{(3.33)}$$

where: α = the actual output value

ρ = the predicted output value

μ = the mean output value of all test patterns

N = the total number patterns in the test set

This parameter is particularly useful for backpropagation because it is independent of the network topology and application. It concentrates solely upon the ability of the network to learn the average relationship between the input values and the corresponding outputs. When the error approaches zero, the network is more or less "guessing" the output and has not even determined the average behavior of the system. As the error approaches one, not only has the average system behavior been found, the higher order of intricacies between inputs and outputs has been determined. This normalized error reflects the proportion of the output variance that is due to error rather than the network architecture (including the initial random weight values) of the network itself. Overall, it is the most useful performance measure for backpropagation [19].

During this project, different combinations of boiler operating inputs were used to determine either a NOx or heat rate output. Some networks performed quite well with high values of R-Square coefficients. The Adjusted R-Square coefficient is used to compare the networks. This Adjusted R-Square coefficient equation is found by modifying the R-Square coefficient:

$$\text{Adjusted } R^2 = 1 - \left[\frac{(N-p) * \sum_N (\alpha - \rho)^2}{(N-p-1) \sum_N (\alpha - \mu)^2} \right] \qquad \textbf{(3.34)}$$

where p = the total number of parameters used to predict the output, which is the same as the total number of connection weights within the network.

Therefore, a network using economizer oxygen, burner tilt, and auxiliary air damper settings (7 inputs) can be compared to one with economizer oxygen, burner tilt, and auxiliary air bias, and fuel-air bias (4 inputs). With these two error parameters, the optimal backpropagation network can be devised for any system. The comparison of different networks from this study is covered in Chapter 5.

Neural Network Data: Quantity vs. Quality

The consideration that is paramount to successful system modeling using the backpropagation network is the quality of the data and how the data are used to train and test the network. This area of neural network use is one of the most important but receives very little emphasis in most literature. The type of data, amount of data, and how the data are used usually determines whether the network will properly map the system inputs to the outputs. Several experiments were conducted to determine the data requirements and the results are in Chapter 5. The common rules-of-thumb regarding data requirements are reviewed in this section with respect to how they applied to this study.

Referring back to the section on network architecture, the size of the network directly affects the amount of data needed for training to achieve an accurate mapping of inputs to outputs. The number of connection weights dictated by the architecture and the internal noise within the data can drastically increase or decrease the amount of data required to achieve a successful mapping. The prevalent rule in determining data quantity is outlined by Klimasauskas [14], where at least five and up to ten data vectors are required for each connection weight. The number of adjustable parameters in the backpropagation model is equal to the number of weights (n_W):

$$n_w = (n_i + 1)(n_h) + (n_h + 1)(n_o) \tag{3.35}$$

with subscripts i, h, and o referring to the number of input, hidden, and output nodes, respectively. Therefore, the number of data vectors for effective training is in the range of:

$$5n_w <= n_d <= 10n_w \qquad (3.36)$$

This method is adequate when the user has vast quantities of data available, provided the data are not excessively noisy or repetitive. The question of data quality comes to light under these conditions. Experiments by Chitra [15] determined that one should avoid data with little variation in the input values but drastic variations in the output response. He continued to emphasize avoiding repetitious data since this causes the network to memorize certain pattern relationships instead of generalizing system interaction. Memorization is signified by only certain processing elements responding to inputs while others remain dormant. The computer science adage "garbage in = garbage out" applies to the neural network data paradigm. If one has vast quantities of bad data, training a neural network with that data will not provide the user with either accuracy or good generalization.

One approach to decreasing network architecture size and data requirements involves grouping input parameters into single input values. The parameters used must reflect physical process changes accurately so that the neural network results have qualitative meaning. This approach reduces the number of required inputs, the amount of data required, and the computation time for network optimization. The bias parameters derived in Chapter 2 are of this type and proved extremely useful in the solution procedure.

Another approach is to use well defined and organized data obtained from special tests instead of data obtained from normal, continuous operation of the system. This method supplies the neural network with quality data that meets the requirements for optimal training and testing data sets. The data in the training set are chosen so that the likelihood of each possible outcome is represented and the test set is chosen to represent the entire population of possible outcomes [14]. Data from these parametric tests meet these constraints much better than normal operating data which can contain gaps in the overall operating ranges for the system. Furthermore, if the amount of available data is limited due to time or operating constraints, it is imperative that parametric data be used to train and test the network. This was the situation for this study. Parametric testing at Potomac River is costly and time intensive. Data collection was limited with only 50 data points collected at full load and 41 data points for 45 MW operation. The data collected had certain advantages and disadvantages which affected the quality of the backpropagation network mapping. These causal relationships are detailed in Chapter 5.

When high quality data are accessible, a different methodology is used to determine the amount of data required for a successful mapping. The DANA model approach, termed as such by Owens and Mocella [13], dictates the network architecture and hence the number of connection weights. The number of hidden units is dictated by equation (3.27) and the number of connection weights is found using equation (3.35). If the number of experimental samples is n_e, then the number of observed output response values (n_t) is:

$$n_t = (n_e)(n_o) \tag{3.37}$$

Since this study used a one output network, the amount of output response values is equal to the number of experimental samples (data vectors). In order to have a DANA modeled system, more observations than connection weights are required so that $n_t > n_w$. As a design criterion, the author suggests that, for the boiler problem, the number of experimental data points exceeds the number of weights by 15%, or a minimum of 10 data vectors. For example, there are 50 experimental data points (n_e) available to train a four input (n_i), one output (n_o) network. By applying the governing equations:

$$n_h = (4*1)^{1/2} = 2 \tag{3.27}$$

$$n_w = (4+1)*2+(2+1)*1 = 13 \tag{3.35}$$

$$n_t = 50*1 \tag{3.38}$$

$$\therefore \; n_t \geq 13 + (0.15) * 13 \text{ and/or } n_t = n_w + 10 \qquad (3.39)$$

Therefore, this meets the DANA model criteria since the number of data points exceeds the number of connection weights. Even if subsequent network experimentation increases the number of hidden processing nodes, the model remains within DANA design criteria until $n_h = 5$. Compared to equation (3.36), this method greatly reduces the amount of data required to train and test a network and emphasizes the benefits of using well defined and organized data. However, for cases where the amount of data available is extremely limited, Weigend [21] experimented with two methods to obtain a mapping. The first method involves providing a large number of parameters for the network, but stopping training before the network has made use of its many degrees of freedom (the oversized network). This results in a network with good generalizing properties but lower accuracy than if allowed to train longer. The second method involves a learning procedure seeking a minimal network capable of accounting for the input data. The user starts with a nominal network configuration. As training continues, connection weights that drop below a certain threshold are deleted. This reduces the network to a minimal configuration and increases accuracy but also reduces the network's generalizing capabilities. In either case, the user must balance the desired network prediction accuracy and the network's ability to generalize system response to new data.

Training and Testing the Backpropagation Neural Network

The final subjective topic in deriving the optimal neural network for modeling the physical system is training and testing the network. With the requirements for data quantity and quality met, one must decide on how to divide the data into training and testing sets as well as which training paradigm will result in optimal network performance. Again, there are a variety of opinions regarding how to best train and test a network and this paper will only highlight the ones considered for this project.

As mentioned earlier, the training set should equally represent the likelihood of all possible outcomes from system operation while the test set should represent the entire population of the operating data. Authors usually agree that between 10% and 20% of all available data should be reserved for testing purposes only. Thus, the network is trained (repetitive processing of the data) on 80% to 90% of all available data. The training data are then passed through the trained network once to determine the level of accuracy achieved from training. The test data are also passed through the network once to see how well the network generalizes system response to unseen data. The tradeoff between accuracy and generalizing ability is of concern. If the network accurately predicts the training data responses but cannot generalize and predict the output response for the test set data, then the network has been overtrained. Overtraining is a function of the number of hidden nodes, the number of connection weights, and the duration of training. A case of the number of hidden nodes (and therefore connection weights) was described earlier in determining the optimal architecture. The training process is analogous to solving for the coefficients in a polynomial regression problem. Too much training can cause the network to become an over-constrained polynomial curve fit for the data which, in turn, prevents acceptable generalization for data the network has not seen.

One of the challenges in using polynomials for fitting experimental data is that they are intrinsically unbounded and they oscillate. In contrast, the effective complexity of polynomial approximation is related to the number of terms in the polynomial. This infers that stable, believable solutions can be achieved without critical concern for the effects of too few or too many parameters. If one has a large and diverse set of training patterns compared to the number of weights, overtraining is not too probable. However, if there are too few weights, the network function will not map well onto the data points presented during training. If the network contains a large number of connection weights, the consensus is not to train the network for a long duration [22,23]. This opinion, however, is quite subjective and the best rule-of-thumb is to cease training when the test set performance starts to decrease or the accuracy of predicted responses plateaus.

The remaining question is which training and testing paradigm will provide optimal input to output mapping of the available data. Two methods govern this decision and they are the single and double test set approaches. The single test set is fairly standard [24,25]. Initially the network is trained using only the training data set (80% of all available data) and checked with the test set until the optimal number of hidden nodes is determined. Then the training and testing set are combined (all available data) and the backpropagation network is trained using all data to finalize the optimal values for the connection weights. Owens and Mocella [13], found this method to be the fastest and most reliable training approach, as well as easy to implement. Therefore, this study used the single test set paradigm.

The double test set paradigm is based on the belief that using a test set to select a best architecture for the network invalidates its use as a test set. Therefore an additional test set is reserved until after the network architecture has been determined. This approach is called cross-validation of which several types were examined [26]. Ultimately these methods were discarded in this thesis due to a small amount of data, computational time constraints, and the need to develop software that would be totally automated. The single test set approach outlined above was the one implemented into the final solution procedure for determining the optimal backpropagation network.

Even with the numerous choices available to the user, optimizing neural network performance for a particular application is not overly difficult or time consuming. Experimentation is required, but a few initial trials will reduce the number of viable network operational choices significantly, after examining how the data interact with the network topology. Each system that is modeled is unique and therefore only a few network variables will influence the network's ability to accurately generalize system operation. However, until these trials are conducted, no neural network functional or operational assumptions can be made in regards to the system being modeled.

CHAPTER 4

NUMERICAL OPTIMIZATION OF NEURAL NETWORK EQUATIONS

The method of computing the governing neural network equations for a given system with operating inputs and subsequent outputs was described in the previous chapter. The second fundamental objective of this study was to optimize the equations such that a minimum heat rate and NOx could be determined along with the corresponding input values that achieve these minima. Several optimization methods and their algorithms were examined to determine which provided the required accuracy, computation time, and ease of incorporating it into a control software package. This chapter provides the basic concepts of optimization and a description of the different approaches that were examined. The equations generated by the neural network training process are described and the criteria for choosing the optimization algorithm is reviewed. The method selected is then discussed in detail along with how the network equations were programmed into the optimization algorithm.

FUNDAMENTAL CONCEPTS OF OPTIMIZATION

The ultimate goal of optimization is to take a single function, f(x), which depends on one or more independent variables, and find the values of those variables such that f(x) is a minimum or a maximum. Numerical methods and thus computer computation for locating these extrema can be adapted for this situation. The computational effort lies in the evaluation of f(x) and its partial derivatives with respect to every variable (depending on the algorithm chosen). An extremum can be either global, the absolute highest or lowest function value, or local where the extremum is within a finite range, not including the range boundaries. Virtually nothing is known about finding the global extrema in general. There are two approaches that are standard for this field of research. The first is to find local extrema starting from widely varying initial values of the independent variables and pick the most extreme of the lot. The second is to perturb a local extremum by taking a finite amplitude step away from it and observe whether the algorithm returns to a better point or "always" to the same point [27].

The goal of multivariable unconstrained optimization is:

$$\text{Minimize } f(x) \quad x \in R^N \tag{4.1}$$

where x is an element ($\in$) within the range (R) of a N-dimensional vector composed of independent variables (x_i with i=1,2,3,...N). The function, f(x), is a scalar objective function and we assume that x_i can take any value, even though the value is usually from a discrete set. It is also convenient to assume that f(x) and its derivatives exist and are continuous everywhere, although optima may occur at discontinuous points of f(x), or its gradient:

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N} \right]^T \tag{4.2}$$

which appears here as a transposed (T) Nx1 matrix. One must remember that f(x) may have a minimum at a point where f(x) and $\nabla f(x)$ are discontinuous or do not exist, but for now we assume that they exist and are continuous.

If the initial input vector, $x^{(0)}$, does not produce the optimal solution, the next step is to determine the subsequent vector, $x^{(1)}$, that is required to reach that solution. This is how the different algorithms distinguish themselves. The following from Ragsdell, et al. [28], develops conditions that allow one to characterize and classify parameter points in the N-dimensional function solution space (design space). Optimality criteria are examined so solutions can be recognized and the most useful method for finding the optimum is identified. Consider the Taylor expansion of a function of several variables:

$$f(x) = f(\bar{x}) + \nabla f(\bar{x})^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 f(\bar{x}) \Delta x + O_3(\Delta x) \tag{4.3}$$

where x = the current or expansion point in R^N

Δx = x-x, the change in x

$\nabla f(x)$ = the N-component column vector of the first derivatives of f(x) evaluated at x

$\nabla^2 f(x) = \mathbf{H}_f(x)$ = the NxN symmetric matrix of second partial derivatives of f(x) evaluated at x, often called the Hessian matrix. The element in the *i*th and the *j*th column is $\partial^2 f/\partial x_i \, \partial x_j$.

$O_3(\Delta x)$ = all terms of order greater than 2 in Δx

Ignoring the higher order terms and examining the change in the magnitude of the objective f(x) corresponding to arbitrary changes in x.

$$\Delta f(x) = f(x) - f(\overline{x}) = \nabla f(\overline{x})^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 f(\overline{x}) \Delta x \qquad \textbf{(4.4)}$$

By definition, a minimum is a point such that all other points in the "neighborhood" surrounding the point produce a greater function value, or:

$$\Delta f = f(x) - f(\overline{x}) \geq 0 \qquad \textbf{(4.5)}$$

The point x is a global minimum if (4.5) holds true throughout the N-dimensional solution space equated by (4.1) and this global minimum is designated by x**. Equation (4.5) yields a local minimum when, for some neighborhood δ where δ>0, || x-x || ≤δ; whereas removing the equality sign in (4.5) gives a strict minimum point. When Δf(x) is either positive, negative, or zero depending on the choice of neighboring points in the area δ, then x is a saddle point.

Assume that f(x) and its first and second order gradients exist and are continuous. In order to determine the sign of Δf(x) from (4.3), the arbitrary values of Δx and ∇f(x) must be zero in (4.4). This means x must be a stationary point. Otherwise, Δf(x) could be positive or negative depending on the signs of Δx and ∇f(x). Therefore, x must satisfy the stationary conditions:

$$\nabla f(\overline{x}) = 0 \qquad \textbf{(4.6)}$$

so equation (4.4) becomes

$$\Delta f(x) = \frac{1}{2}\Delta x^T \nabla^2 f(\bar{x})\Delta x \qquad \textbf{(4.7)}$$

and thus the sign of $\Delta f(x)$ depends on the nature of the quadratic form;

$$Q(x) = \Delta x^T \nabla^2 f(\bar{x})\Delta x \qquad \textbf{(4.8)}$$

rewritten as:

$$Q(z) = z^T A z \qquad \textbf{(4.9)}$$

Basic calculus dictates the following rules:

A is a relative minimum if	$Q(z)>0$ for all z
A is an absolute minimum if	$Q(z)\geq 0$ for all z
A is a relative maximum if	$Q(z)<0$ for all z
A is an absolute maximum if	$Q(z)\leq 0$ for all z
A is a critical point if	$Q(z)>0$ for some z
and	$Q(z)<0$ for other z

With these rules, the stationary point x is a:

minimum, if $\nabla^2 f(x)$ is concave upward

maximum, if $\nabla^2 f(x)$ is concave downward

saddle point, if $\nabla^2 f(x) \leq 0$ or ≥ 0 (an inflection point)

If a descent direction cannot be found, x is determined to be a local (x*) minimum and corresponds to the case where $\nabla^2 f(x)$ is concave upward. This allows the necessary and sufficient conditions to be identified for declaring x*a local minimum.

<u>Necessary Conditions</u>:

$$\text{If } \nabla f(x^*) = 0 \text{ and } \nabla^2 f(x^*) \text{ is an absolute minimum} \qquad \textbf{(4.10)}$$

Sufficient Conditions:

$$\text{If } \nabla f(x^*) = 0 \text{ and } \nabla^2 f(x^*) \text{ is a relative minimum} \qquad (4.11)$$

then x* is an isolated local minimum of f(x). Usually one must be satisfied with finding a local minimum, but if it can be proven that

$$\Delta x^T \nabla^2 f(x) x \geq 0 \qquad (4.12)$$

for all x, then f(x) is a convex function and a local minimum is a global minimum.

OPTIMIZATION ALGORITHMS

With the mathematical objective defined, the question of how to best achieve this goal comes to the forefront. Geometrically, an objective function constitutes an N-dimensional surface within N+1 dimensional space consisting of the independent variables x_1, x_2, ...x_n and the dependent variable, z. Direct multivariable optimization techniques are depicted as a progression of test points on each of the planes defined by the independent parameters. At each point, a function evaluation is performed and a decision is made whether to continue or terminate the search for the optimum.

The simplest progression of test points occurs when only one variable is changed at a time while the remaining N-1 variables are held constant. This results in a one dimensional search routine known as sectioning [29]. Once a minimum value is found using this single parameter value, a new independent variable is selected to start searching while the old parameter retains the value that provided the smallest function response. This method is repeated for all N parameters until no further decreases in the objective function are observed. The major drawback of this method is that it can be inefficient and even ineffective when the objective function becomes complex and the independent parameters interact with each other either as products or quotients.

To alleviate this drawback, algorithms were developed around the sectioning method to improve flexibility and robustness without introducing significant complications. These algorithms are called direct search methods and each has similar traits but distinct characteristics for the type of application they are used for. These methods require only the function values at each iteration and they do not require any partial derivatives to be calculated. The function f(x) is assumed continuous and since gradients are not required, this method can be applied to functions with complex, undefined, non-continuous, or non-existent derivatives. Since the functions being evaluated are multivariable and different combinations of the independent parameters can result in the same minimum value (multi-modal), the user must be satisfied with local minima results only.

Direct Search Algorithms

Multivariable methods that employ only function values to guide the search for the optimum fall under two categories, heuristic techniques and theoretical techniques. Heuristic techniques are search methods constructed from geometric intuition for which no performance guarantees other than empirical results can be stated. Theoretically based techniques are mathematically founded and allow for performance guarantees, such as convergence, to be established, at least under restricted conditions [28]. Three direct search techniques were examined in this study: the S^2, or simplex search, the Hooke-Jeeves pattern search, and Powell's conjugate direction method. The first two are heuristic techniques with fundamentally different strategies. The S^2 method employs a regular pattern of sequential points in the design space, whereas Hookes-Jeeves uses a fixed set of directions (the coordinate directions) in a recursive manner. Powell's conjugate method is theoretically based and it was devised assuming a quadratic objective, which means functions will converge in a finite number of iterations. All three are computationally uncomplicated. However, they are slower than derivative based methods in terms of the speed of moving to the next iteration. Since the amount of calculations is small compared to those required for the gradient descent techniques, these time requirements are, for the most part, balanced across the spectrum of these optimization techniques [29]. A brief description of each optimization method examined is now given, along with the benefits and drawbacks of each method as it applied to this problem.

S^2 or Simplex Search Method. A regular simplex is a geometric shape composed of N plus one vertices, where N is the number of independent variables in the function. These vertices are equidistant and create an N+1 geometric polyhedron. For example, an equilateral triangle is a simplex in two dimensions, a tetrahedron is a simplex in three dimensions, and so on. The main property of a simplex as it is used in optimization routines is that a new simplex can be generated on any face of the old one. This is accomplished by projecting any chosen vertex a suitable distance through the centroid defined by the remaining vertices of the old simplex. A new simplex is thus formed by replacing the old vertex by this newly projected point. In this way, a new simplex is generated (re-dimensioned) with a single evaluation of the objective function.

The method begins by setting up a regular simplex in the space bounded by the independent variables and evaluating f(x) at each vertex. The vertex with the highest functional value is identified as the "worst" vertex and that point is stretched outward or shrunk inward through the centroid of the simplex (a reflection). This creates a new point which is now used to re-dimension the simplex and recalculate the function values at each vertex. This process continues as long as the function response decreases smoothly and stops when either the minimum is straddled or the iterations begin to cycle between two or more simplexes. This algorithm was examined using the Nelder and Meade Downhill Simplex Method in Multidimensions which appeared in [27]. It was programmed on a personal computer and tested using the resulting equations from the neural network analysis described in Chapter 3.

The Complex Method. This method is a derivative of the Simplex method to account for the generation of the initial simplex points. A parameter within the simplex algorithm, α, governs how much the worst point is retracted or expanded through the centroid of the simplex during an iteration's re-dimensioning phase. This parameter can cause some points of the simplex to be infeasible once the move is completed. Each point of the simplex must be tested for feasibility and, if deemed infeasible, suitably adjusted. This adjustment can cause an undesired twisting of the simplex which inaccurately defines the solution space of the function. Therefore, the new vertices must be defined sequentially rather than simultaneously but are still calculated using the regular simplex algorithm. These considerations led to the modifications to the simplex method and resulted in the complex method proposed by Box [30].

Box proposed that the set of P initial trial points be generated randomly and sequentially. Given the upper and lower bounds for the parameter vector $\underline{x}$, the pseudo-random variables, uniformly distributed on the interval (0,1), are sampled. From this sampling, the point coordinates in N+1 dimensional space are determined. N samples are required for an N-dimensional point. Each newly generated point is tested for feasibility, and if deemed infeasible, it is moved inwards to the centroid of the previously calculated points until it becomes feasible. The total number of points to be used, P, should be no less than N+1, but it can be larger. The algorithm now follows the simplex method outlined above. The search is terminated when the pattern of points has shrunk so that the points are sufficiently close together and/or when the differences between the function values at the points becomes smaller than a preset tolerance limit.

Box conducted several experiments with his method [30] and recommended using P = 2N points to define any given simplex. The justification for this many additional points is to prevent the complex from reducing one of its dimensions too far and causing a vertex of the polyhedron to become aligned with another dimension. If a relatively large number of points is used to initially dimension the complex. then a sufficient amount of them should remain after the feasibility criteria is met.

This method was tested by the author using the same neural network equations as for the simplex method and utilizing the IMSL mathematical computer routine, BCPOL [31]. The algorithm uses P = 2N points but otherwise conducts itself like the simplex method in P = N+1 dimensions. The results from this method were the same as the simplex method, but IMSL routines appear courtesy of Lehigh University's computer network and the actual coding is not available due to copyright laws.

The Hooke-Jeeves Pattern Search Method. The two previous search routines were based on the systematic disposition and manipulation of a pattern of trial points. Despite the emphasis placed upon the geometric configuration in which the trial points are located, the main role of the set of points is to generate a direction of search. The point locations influence how sensitive the generated search direction is to the local variations in function topology. The entire set of points is condensed into a single vector difference that defines a direction. The remaining search logic is concerned primarily with adjusting the step sizes so that reasonable improvement is attained after each iteration. Since determining a direction to search in is the major concern, a reasonable improvement to the search technique would be to provide a set of direction vectors in which to guide the search. In a simple sense, a fixed set of directions could be selected and searched recursively for improvement. The search speed could be increased if one or more of the search directions were modified after each iteration so as to better align these directions with the overall function topology. To ensure that the entire solution space is examined, one must insist that the search directions be independent and span the entire domain of f(x). Furthermore, at least one direction must be provided for each parameter resulting in N independent directions for N parameters within the objective function.

The simplest version of this technique is the sectioning approach discussed earlier. Hooke and Jeeves improved upon this method by periodically searching in a direction dictated by the past sequence of iterations and by using the history of the search to determine a new point. Although a simple concept, this decreased convergence time significantly. Basically, the Hooke-Jeeves procedure is a combination of exploratory moves of the single variable (sectioning) search with a change in the current pattern (vector) of the N points or an acceleration move regulated by some basic rules. The exploratory moves examine the local behavior of the function and seek the direction of any sloping valleys that may be present. The pattern moves utilize the information generated during exploration to step rapidly through these valleys. Exploratory moves can have a specified step size for each coordinate direction and can change during the search. For each iteration, a temporary minimum point (base point) is established after all N coordinates have been investigated. A downward gradient (pattern) move, consisting of a single step from the present base point along the line joining the previous base point to the current base point, is then made. If continued exploration were to reveal similar success in moving the base point, each succeeding pattern move would be larger than its predecessor which accounts for the acceleration through valleys [29]. The search is terminated when the step size between base points becomes sufficiently small.

The numerous variations in exploratory moves to establish base points and pattern moves along the gradient make this algorithm difficult to experiment with and accurately compare it to the previously discussed methods. The increased flexibility of the Hooke-Jeeves method also makes it difficult to implement into a software package requiring little or no user interaction. Therefore, after only a few experiments, this method was abandoned for this study. If suitable generalized rules could be implemented such that using the method is simplified without losing efficiency or robustness, then this method has potential for future use.

Powell's Conjugate Direction Method. This algorithm uses the history of the iterations to build up directions for acceleration and at the same time avoids degenerating to a sequence of coordinate searches. A quadratic model is used since it is the simplest nonlinear function having an optimum (linear functions have no interior minima) and, near the optimum, all nonlinear functions can be approximated by a quadratic. Hence, the behavior of the algorithm along the quadratic function will give some indication of how the algorithm will converge for general functions. If a quadratic function in N variables can be transformed so that it is just the sum of perfect squares, then the optimum can be found after exactly N single-variable searches, one with respect to each of the variables [28]. Powell's method focuses on minimizing the line connecting each iteration's minimum function value, or base point, instead of calculating gradients.

A fundamental flaw with Powell's method is that its usefulness is extremely function dependent. If trapped in a long valley with several bends in it, a quadratically convergent method will try to extrapolate to the minimum of a parabola that is not present [27]. Since gradients are not calculated, this decreases the method's efficiency. However, a method with quadratic convergence can save several times N^2 extra line minimizations because quadratic convergence doubles the number of significant figures at each iteration. This quadratic convergence also accounts for the much improved accuracy over traditional search methods without having to calculate any gradients.

The algorithm for Powell's method appears in [27] and requires a start point P which is a vector of length N and an initial direction matrix. The algorithm also requires the function to be implemented within two different programs. One program calculates the move and the other minimizes the line connecting successive moves. The nature of the power plant data which creates the function mapping of boiler operating conditions to NOx and heat rate provides evidence that the objective function contained several long, shallow valleys. Furthermore, it was difficult to determine how or why this method would provide the best performance; and since extreme accuracy was not necessary for this study, the benefits of using this algorithm were few. These reasons led to the decision to not select Powell's Method for further examination.

Gradient Based Algorithms

The direct methods described above are capable of handling many practical engineering problems and are the best choice when reliable information about the governing equation is limited. Although these methods will almost always find a solution, they can require an excessive number of function evaluations to find that solution. This, combined with a desire to find stationary points and fulfill the first necessary condition of optimization in equation (4.10), warrants the use of gradient information.

Only two gradient based algorithms were investigated for this study and neither provided enough improvement in optimizing the equations to warrant the additional effort to program them. The first was the mathematical library algorithm, BCONF [31], that minimized a function of N variables, with bounds, using a quasi-Newton method and a finite difference gradient. The second was the Fletcher-Reeves-Polak-Ribiere (FRPR) method [27], which calculates gradients and line minimizations in route to the optimum. The routine required parameter start points, parameter boundaries, a variable scaling matrix, a function scaling matrix, and 14 additional pieces of data that detailed step sizes and tolerances. Fortunately, all of these had default values which were used in testing this routine. Programming the FRPR routine required the same types of information but with no guidance on selecting the best values or how to determine them for a particular application. Therefore, implementing this algorithm into a stand alone, non-interaction program was not possible. No earnest attempt was made at this method since the quasi-Newton method validated the results achieved using the much simpler direct search algorithms without any significant improvements in speed.

SAMPLE NEURAL NETWORK EQUATIONS

The above discussion made several references to the dependence of selecting an optimization routine upon the type of equation being optimized. Before revealing the algorithm chosen for this study, a brief overview of the type of equations a backpropagation network is composed of is necessary. Because a backpropagation network is a feed-forward type of neural network, successive layers are dependent upon the output of the previous layers. These interactions provide the neural network with its ability to model a non-linear system but they also make the optimization process more complicated.

After a neural network has been trained to satisfaction and has achieved the desired performance, training ceases and the network becomes purely feed-forward. As outlined in Chapter 3, the summation of weighted inputs and the transfer functions can be varied. Although these summations and transfer functions are individually differentiable, optimization can be done only on one non-linear, multivariable equation at a time. If all of the neural network functions were linear, then one could simultaneously solve this set of equations through matrix manipulation. This function for a backpropagation network consists of a series of nested non-linear functions in which taking any derivative becomes infeasible. A sample network, similar to the ones designed in this study, is used in this section to show the intricacies of this equation.

Using a neural network with 4 inputs, 3 hidden processing units, and one output, the following trained network characteristics are given:

- the weighted inputs are simply summed as they enter the hidden processing nodes
- the 3 hidden nodes each use a Tanh transfer function to scale its output between -1 and 1
- the outputs from the hidden nodes are weighted and simply summed as they enter the output
- the output node uses a simple, linear output transfer function

The first statement is represented as:

$$H_{1,in} = \sum_{i=1}^{4} Bias_1 + (w_{1,i} * input_i)$$
$$H_{2,in} = \sum_{i=1}^{4} Bias_2 + (w_{2,i} * input_i) \qquad \textbf{(4.13 a, b, c)}$$
$$H_{3,In} = \sum_{i=1}^{4} Bias_3 + (w_{3,i} * input_i)$$

where $w_{j,i}$ = connection weight value into node j from input i

These three hidden unit values undergo the transfer function process of:

$$H_{j,out} = \mathrm{Tanh}(H_{j,in}) = \frac{e^{H_{j,in}} - e^{-H_{j,in}}}{e^{H_{j,in}} + e^{-H_{j,in}}} \tag{4.14}$$

Now, these three values are weighted and summed as they enter the output node:

$$O_{in} = \sum_{j=1}^{3} \mathrm{Bias}_j + (w_{out,j} * H_{j,out}) \tag{4.15}$$

Finally, this value is linearly transferred as the output value such that:

$$O_{out} = O_{in} = f(x_i) \tag{4.16}$$

Evaluating this series of functions in the logical progression is quite simple. However, taking a partial derivative with respect to x_i requires partial derivatives within the Tanh functions as well as within the summations resulting in several chain rule derivative calculations. This process is even more complicated if a Tanh or Sigmoid function is incorporated as the output transfer function or if a cumulative sum or normalized cumulative summation is used. As more hidden units or inputs are introduced, the complexity of the partial derivatives would increase substantially thus reducing the chances of achieving a fast and accurate solution to the optimization process.

One last consideration must be made concerning the optimization process as it applies to this study before discussing the algorithm chosen to accomplish the task at hand. The boiler optimization problem deals with minimizing heat rate while keeping NOx emissions below regulatory limits. However, all optimization algorithms allow for only one dependent variable and so simultaneously optimizing both parameters is not possible. For this reason, individual networks were designed for NOx and heat rate response to boiler operating inputs. Their resulting equations were integrated into the optimization process by using the concept of the penalty function.

Penalty function methods can be applied to the independent parameters, the dependent variable, or both during the optimization process. This approach is another form of constrained optimization and can be implemented in various ways. Since the neural network equations are generated using input values scaled between -1 and 1, the independent parameters have their boundaries automatically built into the equations. Therefore, only a comparison of the iteration's current minimum to the allowable limit need be done. The goal of minimizing heat rate subject to the NOx limit dictates that as each iteration to minimize the heat rate equations is performed, the NOx value is calculated under the current iteration conditions. If the NOx limit is exceeded, then that point for the heat rate minimization process must be sufficiently penalized such that the optimization algorithm will not get trapped out in a solution space of infeasible points.

Penalties are mathematical functions applied in one of several forms dependent upon the nature of the equation being optimized. Parabolic, logarithmic, linear inequality, and step functions are the most prevalent in engineering applications [28]. Since the backpropagation equations are nested and difficult to describe as a single geometric form, the step function was applied in this study with positive results. After each iteration of optimizing the heat rate equations, the NOx level was calculated using its own network equations at the current iteration conditions. If the NOx was above 0.38 lb./MBtu, a significant penalty of 1000 Btu/kWh was added to the current value of heat rate at that iteration. When the algorithm searched for the "worst" point at the beginning of the next iteration, the penalized value was picked and the algorithm would try to minimize heat rate again starting from that point. Although simple in concept, this method proved to be quite successful without complicating the optimization process or increasing computation time.

CONSIDERATIONS FOR ALGORITHM IMPLEMENTATION

Because the neural network equations are a model of boiler operation with each parameter being restricted to a specific operating range, attempting global optimization of all possible boiler configurations is not feasible. Therefore, constrained optimization was required so as to keep the algorithm functioning with certain physical and safety operating constraints. Ultimately, there is no ideal algorithm for any application, only some that are better suited for a particular application than others. Since our model is multidimensional, the following guidelines were taken into consideration for selecting the optimization algorithm:

- Computer memory requirements for optimization algorithms require memory storage of order N or N^2, where N is the number of parameters (dimensions) in the model. Moderate values of N and reasonable memory size allow this factor to be overlooked. However, if the complexity of the function increases, this can become an important factor.
- The algorithm should make no special assumptions about the function being optimized; such as gradients, continuity, smoothness, or solution surface topography.
- The ease of implementation is important such that the algorithm is concise, self contained, and relatively straightforward and easy to understand.
- The ability to calculate and evaluate the partial derivatives of the function dictate complexity and the amount of required calculations and run-time. Conjugate gradient methods and Quasi-Newton (variable metric) methods require derivative calculations and sub-minimization algorithms.
- Accuracy in the final solution does not have to be to within an extremely tight tolerance. The data collection methods and the process of optimizing a function equated to a generalized mapping of boiler operating conditions to subsequent outputs has a moderate degree of uncertainty. This inherent system uncertainty does not allow for highly accurate results, only results with the same degree of uncertainty.
- If gradient-based techniques are employed, difference approximations must be done at each iteration. This requires considerable experimentation to determine step sizes that strike a proper balance between roundoff and truncation errors. Such lengthy analysis consumes too much time and would be difficult to automate into the stand alone "black-box" approach which is an objective for this study.

The considerations above reduce down to the concept of keeping the method simple, as long as it's successful. After the algorithms detailed previously were tested and compared to each other, they were all found to produce very similar results, well within the uncertainty tolerances for this problem. Therefore, the simplest method, the downhill simplex, was chosen. A detailed description of this algorithm follows along with how the neural network equations were programmed into a function subroutine.

THE SIMPLEX METHOD REVISITED

This method originated as a means to optimize the performance of existing, operating plants, when there was error present in the measured response of the plant to imposed process-variable changes. The simplex search method is based upon the observation that the first-order experimental design requiring the fewest number of points is the regular simplex. The evaluation and selection of the worst vertex point has been discussed. Now the means of reducing that worst point through expansions(reflections) and contractions are discussed to provide greater understanding of the algorithm. Earlier forms of the simplex method did not alter the regularity of the initial simplex shape. Only inward reflections (hence reductions in size) were performed which was laborious and time consuming in producing a solution. Nelder and Meade allowed for expansion and contraction during the course of moving the current "worst" point to a better point. Their modifications require consideration of the following points:

$x^{(h)}$ = the point with the highest current function value
$x^{(g)}$ = the next highest point
$x^{(l)}$ = the lowest current point
$f^{(h)}, f^{(g)}, f^{(l)}$ = the corresponding function values at these points

A reflection step is described by the line:

$$x = x^{(h)} + \lambda(x_c - x^{(h)}) \tag{4.17}$$

or

$$x = x^{(h)} + (1+\theta)(x_c - x^{(h)}) \tag{4.17a}$$

If θ=1, a normal simplex reflection results such that x_{new} is located a distance $\| x_c - x^{(j)} \|$ from x_c. When $-1 \leq \theta < +1$, a shortened reflection, or contraction, is produced while the choice of $\theta > +1$ will generate a lengthened reflection step, or expansion, of the simplex. These conditions are depicted in Figure 4.1.

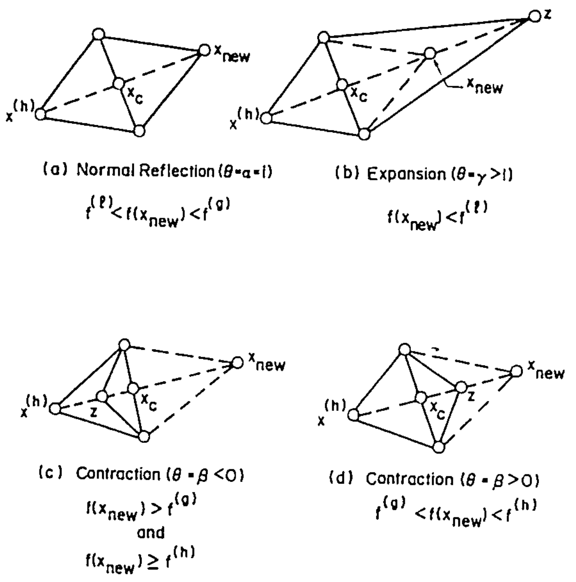
x_{new}
$x^{(h)}$
x_c
(a) Normal Reflection ($\theta = \alpha = 1$)
$f^{(\ell)} < f(x_{new}) < f^{(g)}$
z
x_{new}
$x^{(h)}$
x_c
(b) Expansion ($\theta = \gamma > 1$)
$f(x_{new}) < f^{(\ell)}$
x_{new}
$x^{(h)}$
z
x_c
(c) Contraction ($\theta = \beta < 0$)
$f(x_{new}) > f^{(g)}$
and
$f(x_{new}) \geq f^{(h)}$
x_{new}
$x^{(h)}$
x_c
z
(d) Contraction ($\theta = \beta > 0$)
$f^{(g)} < f(x_{new}) < f^{(h)}$

Figure 4.1

Expansion and Contraction of a Simplex

The three values of θ used for normal reflection, contraction, and expansion are denoted respectively α, β, and γ. The method proceeds from the initial simplex by determination of $x^{(h)}$, $x^{(g)}$, $x^{(l)}$, and x_c. The function values for each vertex of the simplex are checked to see if termination is appropriate. If not, a normal reflection, expansion, or contraction is taken, using the tests outlined in Figure (4.1). The iterations continue until the simplex function values do not vary significantly. Nelder and Meade suggest the values of $\alpha = 1.0$, $\beta = 0.5$, and $\gamma = 2.0$ for the best results.

Some limited numerical comparisons indicate that this method is very reliable in the presence of noise or error in the objective function and is reasonably efficient. Additional experimentation with the α, β, and γ parameters along with the construction of the initial simplex was done by Parkinson, et al., in 1972 [32]. It was determined that the shape of the initial simplex was not important but its orientation was. They suggested $\alpha = 2.0$, $\beta = 0.25$, and $\gamma = 2.5$ which worked well if successive repeated expansions were allowed. Ultimately, the method expands the simplex in these different directions until it reaches a "valley floor" and it continues along the valley as long as the function value keeps decreasing. If there is a situation where the simplex must try to collapse in on itself too tightly, it will contract itself around the current lowest (best) vertex point.

Termination of the algorithm occurs when the vector distance moved in that step is fractionally smaller in magnitude than some preset tolerance (TOL in the computer coding). Alternatively, the decrease in the function value can be used to terminate the algorithm if the decrease between iterations is fractionally smaller than a preset tolerance (FTOL in the coding). Press [27] suggests that TOL should not be smaller than the square root of the machine precision but FTOL may be equal to the machine precision. For this study, the uncertainty in the data and neural network models allow these tolerances be less binding without a loss in accuracy. It is imperative that the results from this optimization be examined carefully to ensure that these tolerances were not fooled by a step that failed to move the simplex in any direction. If the user suspects that the best local minimum has not yet been reached, the multidimensional minimization should be restarted from the point that has been determined to be the minimum. At this restart point, the user should reinitialize any ancillary input quantities such that N of the N+1 vertices of the simplex are reinitialized using any scaled value added to the current point while the N+1 vertex assumes the point of the claimed minimum.

CHAPTER 5

SOLUTION METHODOLOGY AND RESULTS

NEURAL NETWORK OPTIMIZATION

The methodology ultimately selected for optimizing the neural networks used in this study was the result of many computational experiments. These experiments consisted of parametric testing of the NeuralWare software with boiler data, where all of the boiler operating conditions (as outlined in Chapter 3) were held constant while one parameter was changed and the subsequent results recorded. Analysis of these results determined which of the neural network operating characteristics had the greatest effect upon the training of the networks used in this study. After these characteristics were identified, a suitable approach for applying these parameters was determined. This approach became the primary training methodology for all of the networks examined in this thesis. The approach was never altered and this made it possible to compare networks without having to account for any bias created by different training techniques.

Data Preparation

Prior to constructing, training, and optimizing a neural network, one must ensure that the data to be used in this process are adequate enough to achieve the best results. Several different networks, using the boiler operating parameters as inputs, were examined. Each network required a specific arrangement of the parameter values within the data vectors. A data vector consists of all of the input parameter values for a single parametric test. These values are the data vector components and a series of tests (vectors) comprise a data set. The primary concerns for handling the data vector components were:

1. taking the raw data (actual operating conditions) and computing the bias parameters
2. arranging the required input and output parameters so they corresponded to the neural network architecture in question
3. scaling the input and output values

4. splitting the available data into training and testing sets.

These four steps are required for any neural network except for the first step. This step is eliminated when the use of input bias parameters are not warranted. This occurs when only a small number of inputs is required to sufficiently describe the output(s). During the earlier stages of this study, this preparation process was carried out by using a spreadsheet software package [33]. As development of the NOx control software package for PEPCO continued, this process was automated and linked to the expert system program. The following passage describes how this automated process works and how it fits into the overall NOx control software package.

After a substantial change in boiler hardware or operating conditions occurs, the user will desire to get the boiler back into an efficient, low NOx operating mode. He will execute the expert system parametric testing routine to guide him through different steps to bring NOx back into regulatory limits. As the expert system guides the user through a series of tests, the data for each test point is recorded in a data base (as a data vector) for later use. The parameters (vector components) are stored in the following order for each test data vector:

Economizer Oxygen/Burner Tilt/Auxiliary Air Damper Positions (1,3,5,7,9)/Fuel Air Damper Positions (2,4,6,8)/Coal Mill Feeder Speeds (A,B,C,D)/Windbox Pressure/Unit Heat Rate/NOx.

The first fifteen parameters are control variables for boiler operation (outlined in Chapter 2) while the last three are the boiler responses to these control variables. Using the formulas for the mill feeder speed, auxiliary air, and fuel air bias parameters (Eqs. 2.4, 2.7, and 2.16), these biases are calculated for each test point and this information is stored in the same data file as the original raw data values. The user then decides which inputs (either raw or bias data values) he wants to utilize to describe a particular boiler output. Once selected, these components of the individual test point data vectors are extracted, formatted, and written to a temporary file. This file is designated as the "available data" file and consists of all parametric data collected during the expert system testing procedure. Two other types of files are used throughout the neural network optimization process, the training data file and testing data file. The available data file is divided into these two files, as described below, and each has a specific purpose in the neural network optimization process. The data file terminology is important for understanding the neural network optimization process and the terms are not interchangeable.

The third section of the data preparation program is the scaling of the neural network input and output values. Neural networks require all input and output data to be scaled within user-defined upper and lower bounds. The bounds for backpropagation networks are usually -1 and +1 for inputs and -0.8 and +0.8 for outputs. The NeuralWare software uses these bounds as default values with an option to be externally defined. Furthermore, if NeuralWare software is used, this data scaling process is done automatically for the user. However, if this particular neural network software package is not incorporated into the NOx control package, then the scaling of data must be done internally. In either situation, the data scaling process is the same. The scaling algorithm finds the maximum and minimum value of each boiler input and output parameter within all of the available data. Using these values and the scaling bounds (default or user defined), the algorithm calculates a scaling coefficient and an offset value using the following equations:

$$\text{Scale} = \frac{\text{Hi} - \text{Low}}{\text{Max} - \text{Min}} \tag{5.1}$$

and

$$\text{Offset} = \frac{(\text{Max} * \text{Low}) - (\text{Min} * \text{Hi})}{(\text{Max} - \text{Min})} \tag{5.2}$$

where: Max = the parameter's maximum value within all of the available data
Min = the parameter's minimum value within all of the available data
Hi = the upper scaling bound (commonly +1 for inputs and +0.8 for outputs)
Low = the lower scaling bound (commonly -1 for inputs and -0.8 for outputs)

and the final scaled parameter value is calculated by:

$$\text{Scaled Data Value} = (\text{Scale} * \text{Actual Data Value}) + \text{Offset} \tag{5.3}$$

The desired inputs and output have now been selected, scaled, and they are written to another temporary data file. The next step is to select and extract a set of data (≈10%) from the available data set. These extracted data are the neural network testing data and they are written to a separate file while the remaining data are the network training data. As reviewed in Chapter 3, the training data should equally represent the likelihood of each possible outcome while the testing data are chosen to represent the entire population of the available data. This is accomplished in part by the expert system and this data preparation algorithm. It was determined, and proven later in this chapter, that, for this study, a minimum of six to eight data points are required for each input that is used in any neural network. Therefore, if five inputs are used to model an output, then 30 to 40 data vectors will be required from the expert system program to generate a decent mapping of inputs to outputs. One important condition is that the components of the data vectors are spread apart in each component's range of data. This prevents the data vectors from being clumped together in certain regions of the data distribution. The expert system accomplishes this task by making the user conduct tests at conditions such that subsequent test points are a finite, pre-determined distance apart from each other within their operating range. This ensures an equal distribution of data within the range of each parameter as well as between different parameters.

The automated selection of particular data vectors for the testing data set required the calculation of a pseudo-random number based on the computer's internal clock. The time value of the clock is assigned as an initial value within an algorithm that calculates a number between one and the total number of data points available. This number is used to designate which data vector will be selected as a neural network testing data vector. This vector is extracted from the available data and written to a separate testing data file. Since the internal clock will have changed during this process, a different random variable will be generated each time. This process is repeated until ten percent of the available data has been selected as testing data. The remaining data that were not selected for testing are written to a different data file to be used for training the neural networks.

Neural Network Optimization

Once the data have been pre-processed, they are ready for use in training and testing the neural network en route to optimizing the network performance. The method used to optimize network operation was a compendium of different existing approaches [13, 14, 23] along with experimental results. The resulting network optimization method is a fast and reliable means of achieving an accurate mapping of boiler inputs to outputs without loss of generalization and prediction capability.

As discussed in Chapter 3, there are numerous user-controlled variables for fine-tuning neural network operation and thus improving network performance. The following discussion outlines, in order, the neural network performance optimization process used in this study. To simplify this explanation, it is assumed that a neural network, consisting of four inputs, is used to model and predict the response of a single output. There is only one hidden layer between the input layer and the output for this network. The reader is reminded that the following approach yielded the best results for this study, but it may not work as well in different applications that utilize neural networks to model other physical problems.

1. Optimize the Hidden Layer. With the basic network architecture of four inputs and one output already known, the number of processing elements (PEs) in the hidden layer must be determined before any other tuning techniques are applied. The hidden layer size is critical for providing non-linearity to the mapping so therefore it is the priority issue in network optimization. This step requires the use of both the training and the test data sets to find the optimal number of PEs in the hidden layer. As described by Owens and Mocella [13], an initial guess at the number of hidden PEs is calculated by Eq. 3.25 which, for this example, results in three hidden PE's. The network is then trained for a user-defined number of cycles through the training data (learn count), which, for

this study, was 10,000 [14]. This learn count value is used for each network optimization process until the learn count parameter was optimized. This allows for unbiased network configuration comparison, by having all networks trained for the same amount of time and letting the network parameters dictate performance.

After the training session is complete, the training data are passed through the network once in a test mode. The test mode means that no changes in the connection weights will occur as these data pass through and the network functions only in a prediction capacity. This checks the network's accuracy by showing how well it learned the relationships between inputs and outputs. The test set data are then passed through the network to check how well the network can generalize and predict an output for input data it has not yet seen. The results from these two tests are written to output files so that they can be analyzed using the R-Square and Adjusted R-Square performance measures described by Eqs. 3.31 and 3.32, respectively. The test that uses the training data is analyzed with the Adjusted R-Square parameter and the test data set results are checked with the R-Square parameter. These values are recorded for later comparison to networks with greater and fewer hidden PEs.

The user now increments the number of hidden PEs by one and this process is repeated. The author determined that architectures with up to five additional hidden PEs above the initial guess should be trained and analyzed using this method. Also, network architectures with two fewer hidden PEs than the initial guess should be examined. Therefore, if the initial number of hidden PEs is three, architectures with 1, 2, 4, 5, 6, 7, and 8 hidden PEs should be analyzed. Each of these networks will have both the Adjusted R-Square and R-Square parameters calculated for them and these values are tabulated and compared, as in Table 3.2. The network architecture that has the best combined performance in accuracy and generalization is chosen.

Occasionally the situation arises where the best architecture may have several more hidden PEs than the initial guess. This calls for a check on whether the network satisfies the criterion that the network be overdetermined. This entails comparing the number of connection weights to the number of available data vectors by applying Eqs. 3.33, 3.35, and 3.36. If the best network architecture results in a model with too many connection weights and an insufficient mapping is more likely to occur, the next best architecture is chosen.

Once the optimal number of hidden PEs is determined, that network architecture is used for the remaining optimization steps. The training data and the test data are then re-combined into one data set which contains all available data. Training and testing the network with this single data set is so that this network model sees all of the data and can generate a better mapping than by holding back data for testing only [13]. The total data set is used until the final optimization step, finding the optimal learn count, is reached. At that point the data are again separated into the original training and testing data sets.

2. Optimize the Transfer Functions. Backpropagation networks perform best when using combinations of the Sigmoid, Tanh, Sine, and Linear functions (Eqs. 3.14 - 3.17). Since most backpropagation network applications require only one hidden layer, optimizing these transfer functions is not difficult. The author determined that throughout the study, the following combinations of transfer functions yielded the best results:

Table 5.1

Optimal Network Layer Transfer Function Combinations

Transfer Function Combination #	Hidden Layer Transfer Function	Output Layer Transfer Function
1	Tanh	Tanh
2	Tanh	Linear
3	Sigmoid	Linear
4	Sine	Sine
5	Sine	Linear

The most common combinations throughout this study were the Tanh-Tanh (combination #1) and Tanh-Linear (combination #2) transfer functions. This is expected since the nature of the backpropagation network, in this study, is to find deviations from average behavior and the hyperbolic tangent function is the most suited for this purpose [14]. As before, the network is configured individually with each of these combinations, trained for 10,000 cycles, tested using the same data, and the R-Square parameter is calculated and recorded. The best R-Square value out of all combinations determines which transfer function combination will be used.

3. Optimize the Epoch Size. With the network architecture and layer transfer functions determined, the smaller refinements are now optimized. Recalling that the epoch size determines how often the network connection weights are updated, this parameter finds the underlying frequencies within the data. By optimizing this parameter, the network is allowed to learn the data at an optimal pace, analogous to a student being able to comprehend a certain quantity of information in a given time period.

Optimizing the epoch size is a two part operation. First the epoch size is set equal to two, the lowest value without causing incremental updates (epoch=1). Using all of the available data to train and test the network, the same procedure is conducted as before. Each subsequent trial has the epoch size increased by two so that even values of epoch size are used. It was determined by the author that increments of one did not yield significant enough differences in network performance to base optimization decisions upon so the value of two was picked to simplify the process. This approach continues up to an epoch equal to 20, the author's upper limit for this parameter. This limit was determined by adding 25% to the NeuralWare default value for epoch size (16). No significant improvement in network performance was ever observed with epoch sizes greater than this limit.

After all ten epoch sizes, from two to twenty, have been examined, the three epoch sizes that yield the best R-Square performance are selected for a second round of analysis. To ensure that the epoch sizes would repeatedly result in the best learning capability, each of the three epoch sizes are trained and tested three more times apiece, using the same method, and their R-Square values recorded. The epoch size that yields the highest average R-Square value is chosen for continuing the network optimization.

4. Optimize the Learning Rate and Momentum Coefficient. While the above parameters are optimized, the user can observe the network's error curve being generated as training continues. This NeuralWare graphic is the root mean square (RMS) error curve and it scales, between 0 and 1, the error between actual and predicted output values. If this curve does not smoothly decrease but instead has large fluctuations, then changes to the learning rate and momentum coefficient must be made. The rule of thumb is to proportionally lower both values while maintaining the output layer's values at 1/2 the hidden layer's values. This approach appears in the literature [9, 11, 15] and it worked well for the author while experimenting with the neural network parameters. No cases of extreme error fluctuation were observed during this study and the majority of the networks functioned nicely using the NeuralWare default values for learning rate and momentum [11].

5. Optimize the Learn Count. Up to this point, the number of network data presentation cycles has been set at 10,000. In this last phase of network optimization, this parameter is varied so that the best combination of accuracy and generalizing capabilities are acquired. Neural network learning is similar to calculating the coefficients of a non-linear, higher order polynomial that can trend the interaction of independent variables to a dependent variable. One does not wish to fit a curve that passes through each data point (overtraining) because if the function is given input values that were not used in determining its coefficients, it will fail to predict an output with reasonable accuracy. Therefore, a balance must be struck between accuracy and prediction capability which is the reason for optimizing this parameter.

The data are again split into the same training and testing sets used previously to optimize the number of hidden layer PEs. Using all of the previously optimized network parameters, the network is allowed to learn for any number of cycles. The training and testing set are passed through in a prediction-only mode and the same performance statistics are calculated and recorded. The user then selects a different learn count and the process is repeated. Any increment in the learn count can be used but the author determined that a minimum of 500 is required to achieve significant changes in performance. Again, the learn count that yields the best combination of R-Square and Adjusted R-Square values for the testing and training set, respectively, is the optimal learning count.

Once the optimal learn count is identified, the training and testing sets are again combined and the network is trained using all of the available data for the optimal number of learning cycles. The data are then passed through the network in a testing mode to get a final evaluation on the network's accuracy. The results from this final test run are used in evaluating the overall network performance, using the methods described in Chapter 3. Also, the final connection weights between processing elements are recorded for determining the percent contribution of each input on the network output response. These two topics are examined later in this chapter.

<u>6. Optimize Other Parameters</u>. There are many other parameters that the user can adjust to more finely tune the network operation. However, after many experiments, only the parameters described above had enough impact on the network optimization process to be worth incorporating into the NOx control software package. The standard default functions that are most common to backpropagation networks were used during the optimization process. These functions are:

Standard Error Function (Eq. 3.7)
Norm Cumulative Delta Learning Rule
Normal Summation Function (Eq. 3.23)

These complete the variable parameters that govern backpropagation network operation as described in Chapter 3. If time is of the essence, all default values within the <u>NeuralWare</u> software package usually provide an acceptable "rough" mapping of the selected inputs to outputs.

Otherwise, the above steps will get an accurate network operating to its best potential in a minimal amount of time.

NEURAL NETWORK PERFORMANCE ANALYSIS

The neural network has now been configured such that it gives the best mapping of inputs to outputs based upon the R-Square performance parameter. However, this does not provide much quantifiable information about what this mapping means in physical process terms. The user requires means to identify just what the optimized network represents and what it can tell him about the system being modeled. Several different methods were used in this study to accomplish these goals, and each of the neural networks created were subjected to this analysis. The analysis methods are described below and the individual network results are detailed later in this chapter.

Performance Parameters

The performance parameters used in this study were chosen for their ability to capture the essence of the physical meaning of the optimized neural network. The R-Square and Adjusted R-Square were used to evaluate the network during different stages of network optimization. With the optimization complete, two additional parameters were used so that different optimal network configurations could be compared to one another. These parameters are the average difference and the average percent error, mathematically defined in Eqs. 3.26 and 3.27. For the NOx networks, where the actual and predicted output values were less than one, the author used the average percent error for network comparisons. For heat rate networks, with output values in the thousands, the author relied more upon the average difference.

When a network is optimized and tested, the actual output value and the network-predicted value (for each vector within the data set) are written to an output file. These output files were then evaluated through use of Microsoft Excel 4.0, a spreadsheet software package [33]. This software allowed for many different analyses of the network which are subsequently covered in this chapter. In the network results sections that follow, the R-Square, Adjusted R-Square, average difference, and average percent error are all presented for the full load and 45 MW load levels. These performance parameters are also used to measure the changes in network performance when less data and clustered data were used to train and optimize networks.

Neural Network Connection Weight Analysis

As described earlier, once a network is optimized, it performs only in a predictive mode and no further changes to the PE connection weights occur. The following analysis, taken from Garson [34] and Chitra [15], allows the user to determine the amount that each input contributes to the network output. It is important to note that only the absolute values of the weights are used and that the network bias connection weights are not included. The reader is reminded that the network bias PE is connected to the hidden layer PEs and the output PE. The bias simply provides input values into these PEs in case the actual input values to these PEs are equal to zero. If the PE inputs are equal to zero for several cycles, learning does not progress. Therefore the bias PE acts as an "electrical ground" for the network so that numerical information continues to flow through the network and learning is never suspended by bad or missing data vector components.

The connection weights (CWs) between the input layer and the hidden layer are extracted from the optimized network and arranged in a matrix. Setting NV = the number of input PEs and NH = the number of hidden PEs, the matrix has NH rows and NV columns. The weights are assigned positions based upon the notation in Table 3.1, where the first subscript (j) is the destination PE and the second subscript (i) is the origin PE. Table 5.2 displays this matrix configuration.

Table 5.2

Input Layer to Hidden Layer Connection Weight Matrix

	Input #1	Input #2	•	•	Input #NV
Hidden PE #1	W_{11}	W_{12}			$W_{1\,NV}$
Hidden PE #2	W_{21}	W_{22}			
•			$W_{j\,i}$		
•					
Hidden PE #NH	$W_{NH\,1}$				$W_{NH\,NV}$

The indicial notation indicates that i = the row index and j = the column index Taking the sum of all weights in each of the hidden PE (i) rows, across all of the input PE (j) columns, or::

$$\sum_{j} W_{ij} \tag{5.4}$$

each weight value within each hidden PE row is divided by this sum of the weights within the row. This calculates the weight fraction of the ith hidden PE from the jth input PE, or:

$$F_{ij} = \frac{W_{ij}}{\sum_{j} W_{ij}} \tag{5.5}$$

Now, instead of PE connection weights, the weight fractions are the components of the matrix.

At this point, the connection weights between the hidden layer and the output layer are introduced. Since this study used only single output networks, this connection weight matrix is an NH x 1 column matrix. The notation O_i represents the connection weight from the ith hidden PE to the output PE. The input to hidden layer weight fraction matrix and the hidden layer to output connection weight matrix are thus written as:

Table 5.3

Hidden Layer to Output PE Weight Fraction Matrix

F_{11}	F_{12}	•	•	$F_{1\,NV}$	O_1
F_{21}	F_{22}			$F_{2\,NV}$	O_2
•		F_{ii}		•	•
•				•	•
$F_{NH\,1}$	•	•	•	$F_{NH\,NV}$	O_{NH}
C_1	C_2	•	•	C_{NV}	

The bottom row is a 1 x NV row matrix which quantifies the contribution of the jth input to the output. Each of these elements is calculated by:

$$C_j = \sum_i F_{ij} * O_i \tag{5.6}$$

With the input contributions calculated, the percent contribution of each input is found by:

$$P_j = \frac{C_j}{\sum_j C_j} * 100 \tag{5.7}$$

and the sum of these contributions should equal 100 percent.

Analyzing the connection weights is extremely useful when various inputs are being combined differently to determine which have a significant effect upon the output response. If the user has background knowledge of the physical system, this analysis serves as a check to see if the network is operating in a similar capacity. For example, in this study, the creation of NOx at full load is extremely dependent upon the level of economizer oxygen, as seen in Figure 2.2. If the connection weight analysis resulted in the economizer oxygen percent contribution being significantly smaller than parameters known to be less important, than the neural network has not correctly learned the underlying input interactions.

Graphing Neural Network Trends

If the connection weight analysis yields results that are compatible to actual system operation and response, the next check is to graph the various trends that the neural network has learned through the training process. This requires programming the network equations within a spreadsheet [33]. The final optimal network settings for data scaling, transfer functions, summation and connection weights are incorporated into a spreadsheet, similar to the way the equations appear in Chapter 4 (Eqs. 4.13 - 4.16). Once the functions are embedded in the spreadsheet, all of the available data is copied into these equations. If the output responses from the spreadsheet equations are the same as the predicted output values from the neural network test run, the network equations were successfully transferred. Although this seems trivial, the equation transposing process from the NeuralWare software to the spreadsheet software was very susceptible to error.

Once the equations are programmed, incremental changes to the input parameters are made in the spreadsheet and the subsequent network output response is calculated. These responses are graphed under various constant parameter conditions and then they are compared to the boiler operating graphs which appeared in Chapter 2. The same trends should appear within the neural network equations as in the observed physical trends, with the exception of slight offsets in the curves to account for the relative error within the network equations. This error, although present, should not be significant to the point of exceeding the amount of scatter in the original physical data.

This is the final check on the neural network to ensure that the physical characteristics of boiler operation at the Potomac River Station were modeled accurately but not too rigidly. Although these different network checks are time consuming and involved, they are extremely important to the overall success of the entire optimization process. If a poor mathematical model is passed to the numerical optimization routine, then the user will be supplied with unreliable and, or, unfeasible results.

Reduced and Clustered Data Tests

Once the full load and 45 MW load level neural networks for NOx and unit heat rate were optimized and functioning properly, experiments regarding data quantity and quality were examined. This was important for future boiler optimization problems since the amount of available data for training and testing neural networks is limited by time and cost constraints at the plant. If an accurate and reliable network could be generated with fewer data vectors, then the lower limit of this data requirement had to be determined.

The full load and part load data bases were analyzed first to observe the distribution of data for each input. Then, the data available for training was reduced by 25%, 50%, and 75%. A neural network was trained, optimized, and tested for each of these reduced data cases. Each performance statistic was calculated and graphed with each statistic being a function of the amount of available data. A "break-even" point is observed in each instance where accuracy drops below an acceptable level. These graphs appear in each of the network's results section.

The effect of using data that was clustered within a small range of a single parameter was the second experiment. This was done to check the effects of poorly distributed data. The clustering of data resulted in training data set size equivalent to the 50% data reduction case, but now the data were unevenly distributed. Guidelines developed from this study were incorporated into the expert system algorithm so that as the user retunes boiler operation to achieve efficient, low-NOx performance, the data points have the necessary distance between them. Having adequate distance between data points prevents the effects of clustered data. The results for each load level will be discussed after the NOx and unit heat rate neural network results sections.

FULL LOAD NETWORK RESULTS

Achieving NOx compliance at the full load level without sacrificing unit heat rate is a major objective at Potomac River. Although the lower loads are important, NOx compliance is easier to achieve at lower loads. For this reason, the full load networks received the greatest amount of scrutiny in this investigation.

NOx Neural Network

This network was the forerunner of all subsequent networks and its available data set consisted of 50 vectors. The final decisions on many of the neural network designs and optimization considerations were made using this network. The comparison between which inputs were finally selected to model boiler operation is reviewed first; followed by the optimal network settings and the connection weight analysis. The network performance results are compared to parametric test data obtained by D'Agostini, et al. [4]. Finally, a comparison between the neural network boiler trends and the observed boiler physical trends is included as a final performance analysis.

Input Selection. Fifteen independent control variables for boiler operation were identified at Potomac River by Energy Research Center (ERC) engineers. Since it was not practical to perform field testing for each variable, the bias parameters derived in Chapter 2 were used to characterize the mills and air damper settings. The full load NOx network had several different input configurations combining physical boiler settings and bias parameters. A statistical performance comparison of these networks follows, along with the final network configuration that was used throughout the remainder of this study.

Nine different networks were investigated to determine which combination of inputs gave accurate results and were reliable for numerical optimization. The following list details the different combinations of inputs used to predict NOx:

Net #1 = NOx = f(O2)
Net #2 = NOx = f(Tilt)
Net #3 = NOx = f(O2, Tilt)
Net #4 = NOx = f(O2, Tilt, Alpha, Phi)
Net #5 = NOx = f(O2, Tilt, FA2, FA4, FA6, FA8, Alpha)
Net #6 = NOx = f(O2, Tilt, AA1, AA3, AA5, AA7, AA9, Phi)
Net #7 = NOx = f(O2, Tilt, Mill A, Mill B, Mill C, Mill D, AA1, AA3, AA5, AA7, AA9, Phi)
Net #8 = NOx = f(O2, Tilt, Mill A, Mill B, Mill C, Mill D, FA2, FA4, FA6, FA8, Wind, Alpha)
Net #9 = NOx = f(O2, Tilt, Mill A, Mill B, Mill C, Mill D, FA2, FA4, FA6, FA8, AA1 - AA9)

where: O2 =	economizer oxygen (%)
Tilt =	degree of burner tilt from the horizontal
Alpha =	auxiliary air bias, Eq. 2.7

Phi =	fuel air bias, Eq. 2.16
FAi =	fuel air damper position (degree of openness)
AAi =	auxiliary air damper position (degree of openness)
Mill i =	mill feeder speed (rpms)
Wind =	windbox pressure

The following table reflects the performance of each of these networks. The reader is reminded that the closer the R-Square and Adjusted R-Square values are to one, the more accurate the network prediction. The R-Square parameter reflects the network prediction accuracy while the Adjusted R-Square accounts for how well the different network architectures predict output, subject to the number of inputs used in the network. The Adjusted R-Square is the parameter to use for comparing the different network architectures.

Table 5.4

Comparison of Experimental Full Load NOx Networks

	R-Square	Adjusted R-Square	Average % Error
Net#1 (1 input)	0.7515	0.7464	3.47
Net#2 (1 input)	0.3690	0.3558	5.21
Net#3 (2 inputs)	0.9094	0.9060	2.79
Net#4 (4 inputs)	0.9187	0.9081	2.59
Net#5 (7 inputs)	0.9270	0.9094	1.89
Net#6 (8 inputs)	0.9484	0.9104	1.61
Net#7 (12 inputs)	0.9502	0.9211	1.43
Net#8 (12 inputs)	0.9573	0.9402	1.31
Net#9 (15 inputs)	0.9695	0.9507	1.26

Several factors had to be considered to decide which network was the best choice for the NOx prediction model. First, for every independent variable (or input), there must be a sufficient range of values that variable can have and during parametric testing that range must be adequately covered. One group of variables, the fuel air damper positions, were identically set and were always evenly adjusted. This lack of diversity within the fuel air damper setting data makes these damper settings a poor choice for input parameters. Furthermore, with every additional input, the amount of connection weights increases based upon Eqs. 3.25 and 3.33. This requires substantially more data vectors, which must be obtained by testing, which may not be feasible. Finally, when the networks with more than seven inputs were optimized using IMSL subroutines [31], the optimization algorithm tended to extrapolate to unreasonable answers. This was due to the paucity of available data to train these larger networks. Overall, if there is no significant improvement in the network performance, then additional inputs are not worth the added requirements to provide data for them.

In light of these factors, the best choice of inputs is Net #4, which has NOx as a function of economizer 02, burner tilt, auxiliary air bias, and fuel air bias. At full load, all of the mills are running at near maximum rpm, with very little difference between their speeds. Therefore, networks with mill feeder speeds have too many parameters with too little variation among and between each variable. Similarly, the individual damper positions are not varied enough throughout the data set. By using bias values instead of the actual settings, the network has discrete data values for learning.

By using a four input network, up to six hidden PEs can be incorporated into the network and it would still be an over determined model [13]. This allows for a wider selection of network optimization choices. Networks with more inputs would either have too few hidden PEs or too few data points to adequately train the network. Finally, the success achieved using the four input NOx network made it the architecture of choice for all other networks investigated in this study.

Network Settings and Performance Results. After performing the network optimization steps outlined previously in this chapter, the final settings and performance results for the full load NOx network are as follows:

Table 5.5

Full Load NOx Neural Network Optimal Settings

Number of Hidden Layer PEs:	3
Transfer Function: Hidden Layer/ Output Layer	Tanh / Linear
Epoch Size:	2
Learn Count:	35,000
Learn Rate: Hidden Layer/ Output Layer	0.3 / 0.1
Momentum: Hidden Layer/ Output Layer	0.4 / 0.4
Error Function:	Standard
Learning Rule:	Norm-Cum-Delta
Summation Function:	Normal Sum

Table 5.6

Full Load NOx Neural Network Performance Results

Performance Measure	Input Percent Contribution
R-Square = 0.9187	Economizer O2 = 23.4%
Adj. R-Square = 0.9081	Burner Tilt Angle = 45.2%
Avg. % Error = 2.59 %	Auxiliary Air Bias = 22.7%
Avg. Difference = 0.01 (lb/MBtu)	Fuel Air Bias = 8.7%

<u>NOx Neural Network Trends vs. Observed Physical Trends.</u> The following series of figures is presented for comparison to the physical boiler trends shown in Chapter 2. These figures detail the prediction accuracy of the NOx neural network model using the training data, as well as network model trends using hypothetical inputs. The trend graphs show how well the neural network learned the underlying interactions of the boiler input data instead of only learning the training data.

In order to understand any discrepancies between the actual boiler behavior and the neural network model, the distribution of the training data must be discussed. If a network is trained on incomplete or inconsistent data, the network's ability to accurately generalize boiler behavior will be limited. Figures 5.1 through 5.4 show the number of data points within specified ranges for each input parameter. One can see that there are data available throughout the data ranges for each parameter, although some of the distributions are one-tailed. Economizer oxygen (Fig 5.1) and auxiliary air bias (Fig 5.3) are more skewed than the other parameters and the burner tilt data (Fig 5.2) has the best distribution of all parameters. The distribution of these data dictated the values that were assigned to the fixed parameters for the trended data graphs that follow. If the distribution was spread across a certain discrete range, then the average value within that range was used for fixed parameter values. If the data was skewed, then the mode of all data for that parameter was used for the fixed parameter value. For example, Figure 5.3 shows that auxiliary air bias values of 0.50 and 0.83 were most common throughout the full load data. Therefore, these values were used as the fixed values (constants) for network trend plots such as NOx versus economizer oxygen at constant auxiliary air bias (Figure 5.7).

The distribution of data for each parameter acts as a check for the network reliability. If the network equations give poor correlations and trends, examining the data distribution usually reveals the reason for these results. If the network equations show trends and relationships where there were little or no data to support such results, then the trend is suspect.

Figure 5.5 depicts a scatter plot of the network predicted NOx and the actual NOx values versus economizer oxygen at full load. Included are the linear curve fits for each set of data. Not only is the accuracy of the predicted NOx very good, the trend for each set of data, depicted by the linear curve fits, is identical. This linear correlation plays a significant part in the optimization process as the minimum economizer oxygen level produces the lowest attainable NOx.

The effect of burner tilt angle on the predicted NOx is similar to that of the actual NOx and is shown in Figure 5.6. A parabolic curve fit through the predicted NOx scatter plot has the same shape and intercept as the actual NOx plot in Figure 2.4. It is readily observable that the minimum NOx occurs within the burner tilt range of five to ten degrees above the horizontal. As Table 5.6 shows, economizer oxygen and burner tilt contributed the most towards the production of NOx. Auxiliary air bas is also a contributor, but more in the aspect of being coupled with the oxygen level and burner tilt position.

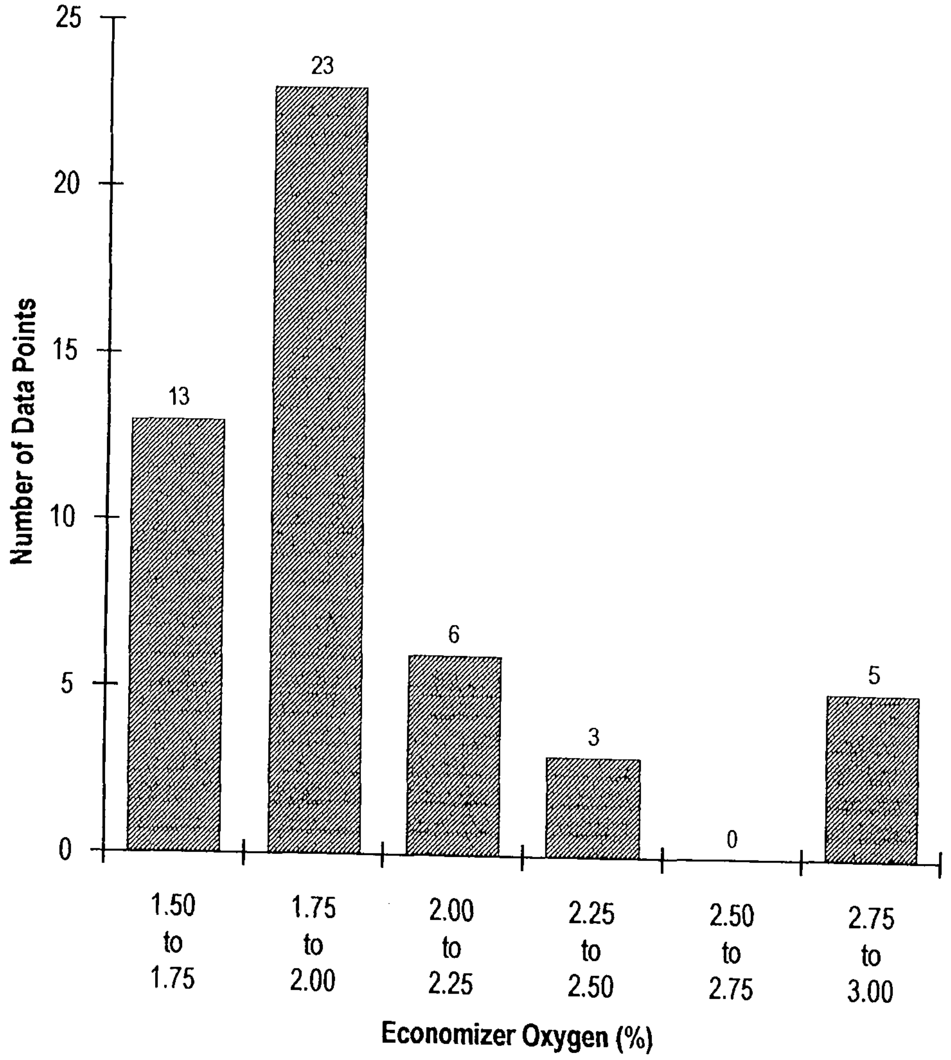

Figure 5.1

Full Load Available Neural Network Data
Distribution for Economizer Oxygen

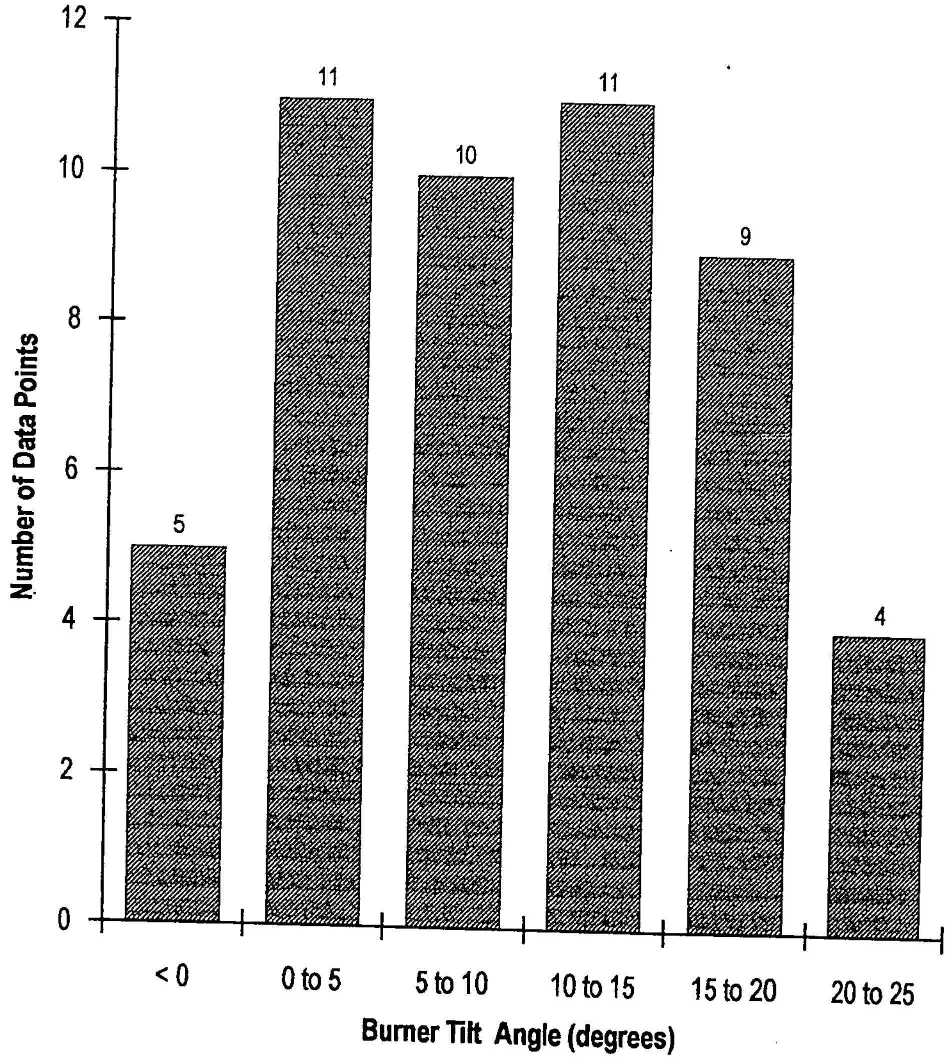

Figure 5.2

Full Load Available Neural Network Data
Distribution for Burner Tilt Angle

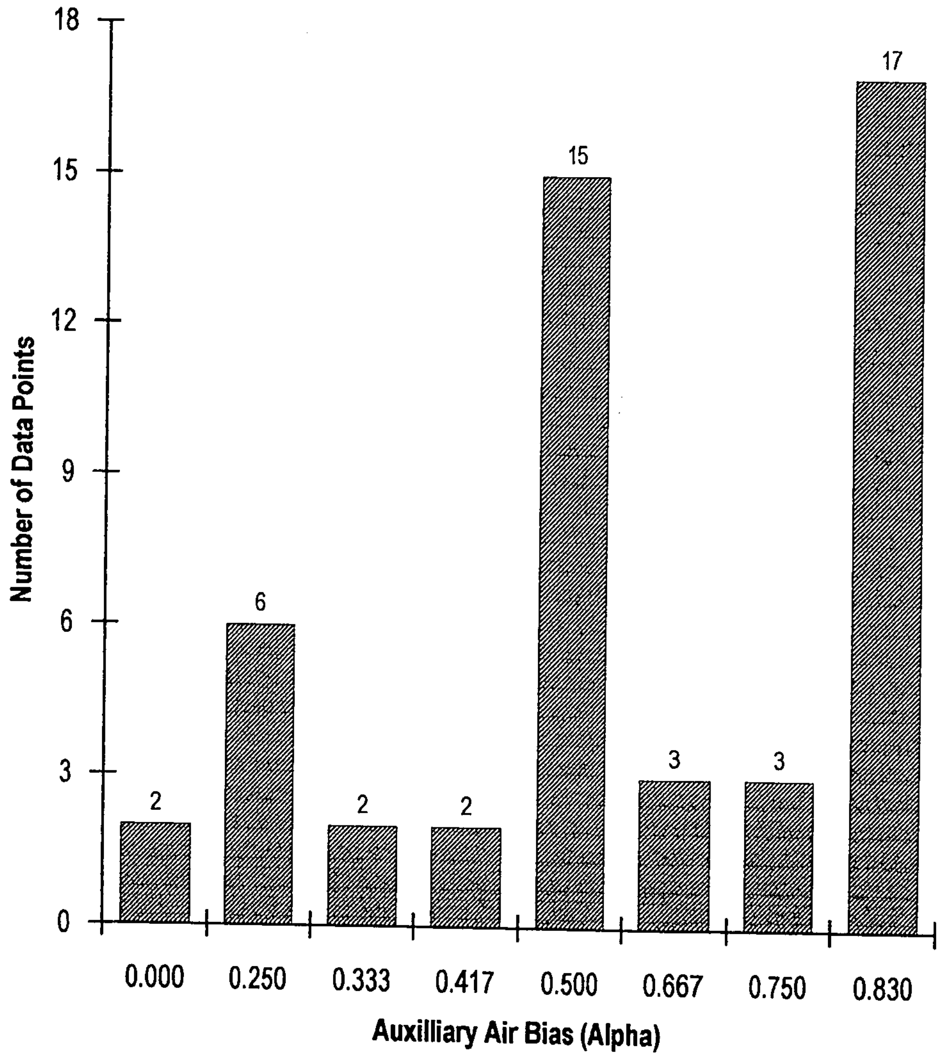

Figure 5.3

Full Load Available Neural Network Data
Distribution for Auxiliary Air Bias

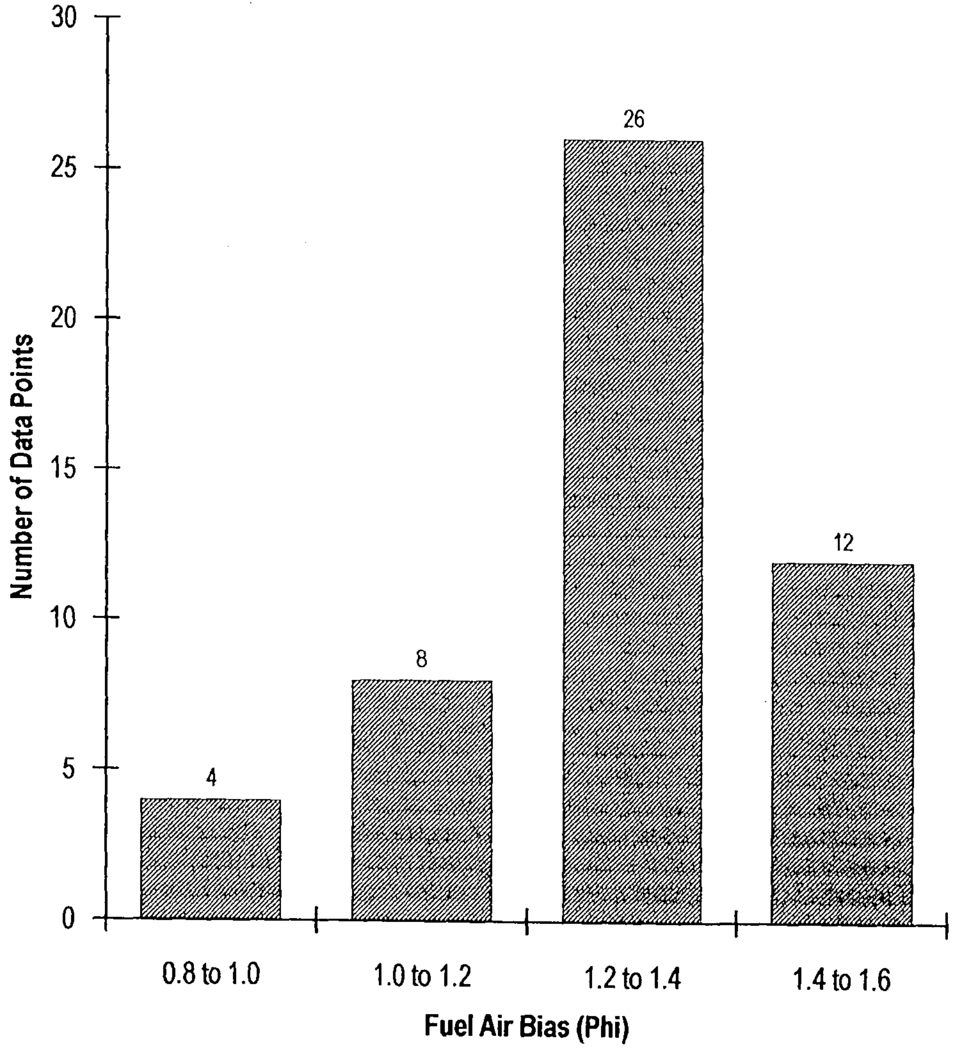

Figure 5.4

Full Load Available Neural Network Data
Distribution for Fuel Air Bias

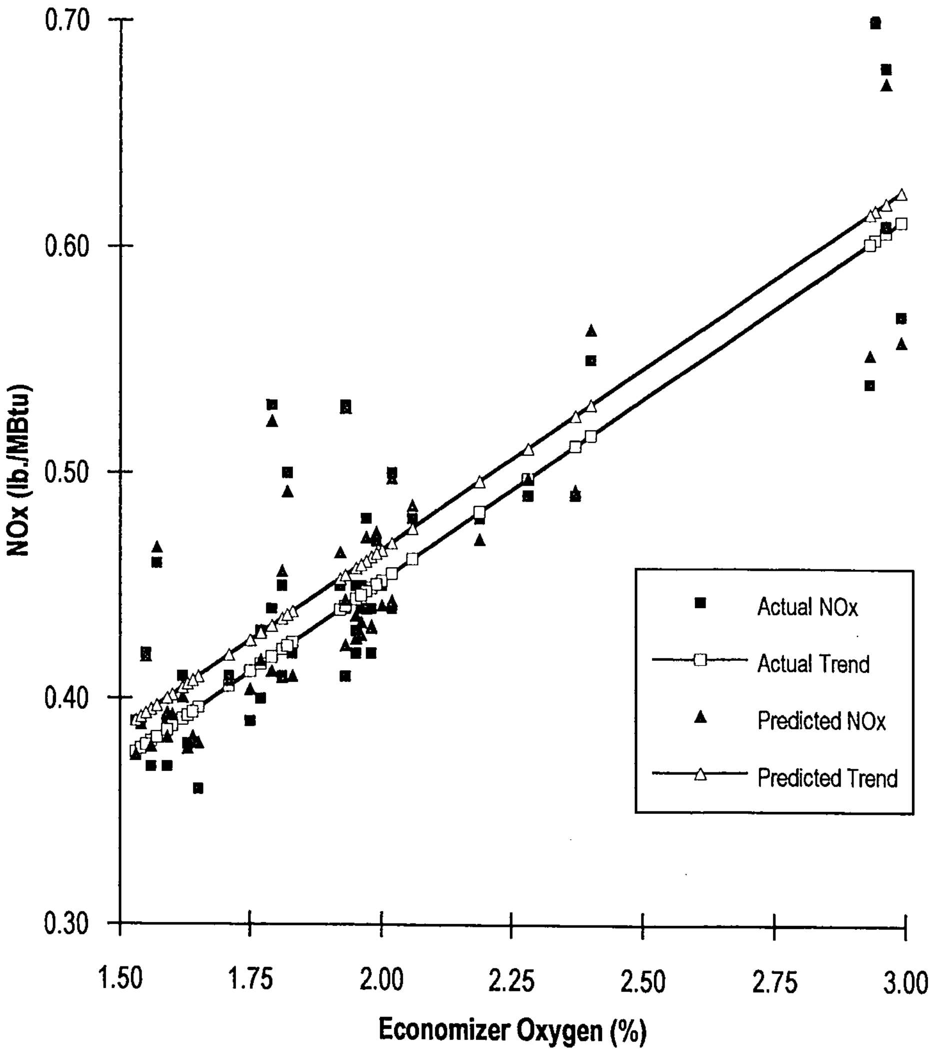

Figure 5.5

Full Load Actual and Predicted NOx
vs. Economizer Oxygen

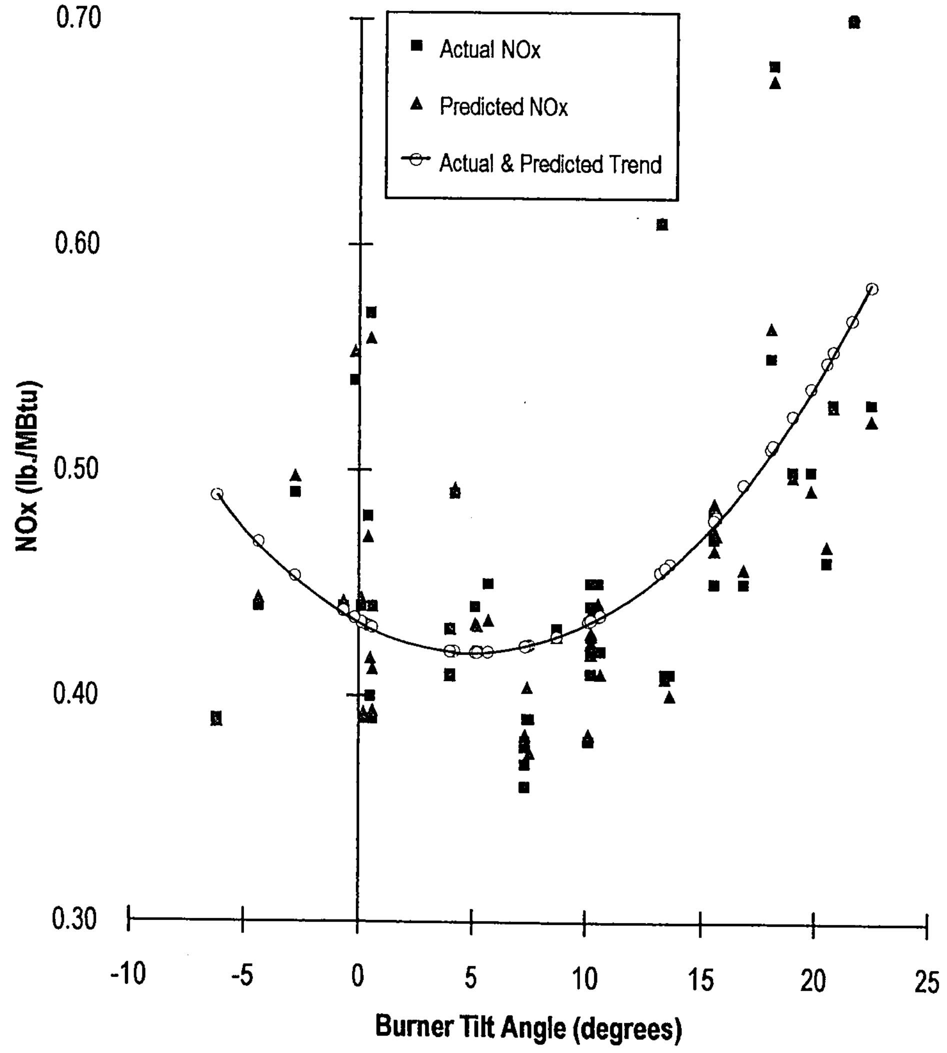

Figure 5.6

Full Load Actual and Predicted NOx
vs. Burner Tilt Angle

The following graphs use the neural network equations to trend data other than the actual parametric data used to train the network. The data for these trends were generated by incrementing through the different independent parameter data ranges. These data were then combined with fixed values for certain independent parameters, as described earlier, and plotted. The plots depict how NOx varies with changes in the independent parameters to show how well the neural network learned boiler operation. These trends are compared to observed boiler operation detailed in D'Agostini's work [4].

Figures 5.7, 5.8 and 5.9 show the interaction of economizer oxygen and burner tilt angle. Each of these figures was created with a constant auxiliary air bias of 0.5 (moderate overfire air) and a fuel air bias = 1.2. These values were used because they appeared often in the available data (Figures 5.3 and 5.4) for training the network and they were the values selected for full load operation by D'Agostini, et al. [4]. The predicted NOx versus economizer oxygen with all other parameters held constant reveals the same linear relationship as when all of the other parameters were allowed to vary. This is consistent with Figure 5.5, although in that figure the auxiliary air bias and fuel air bias parameters are varied and more scatter is within the trend. Similarly, Figure 5.8 has the same parabolic curve trends as seen earlier in Figures 5.6. The tails of the curve do not steepen as much in the constant parameter graphs, but the minimum NOx for each curve is in the same burner tilt range of five to ten degrees above the horizontal.

Figure 5.9 depicts the economizer oxygen and tilt interaction in the NOx prediction model at full load. It has the combined trends of the two previous plots with a parabolic curve that linearly shifts upward in NOx as the economizer oxygen level increases. The minimum NOx still occurs where it did previously, at the minimum oxygen level and between the positive five to ten degree burner tilt angle. This three dimensional solution surface reinforces the important effects of oxygen and burner tilt angle on NOx production at full load.

The third most important input parameter was auxiliary air bias and therefore its effect upon the oxygen and burner tilt interaction needed to be explored. Figure 5.10 shows this effect at

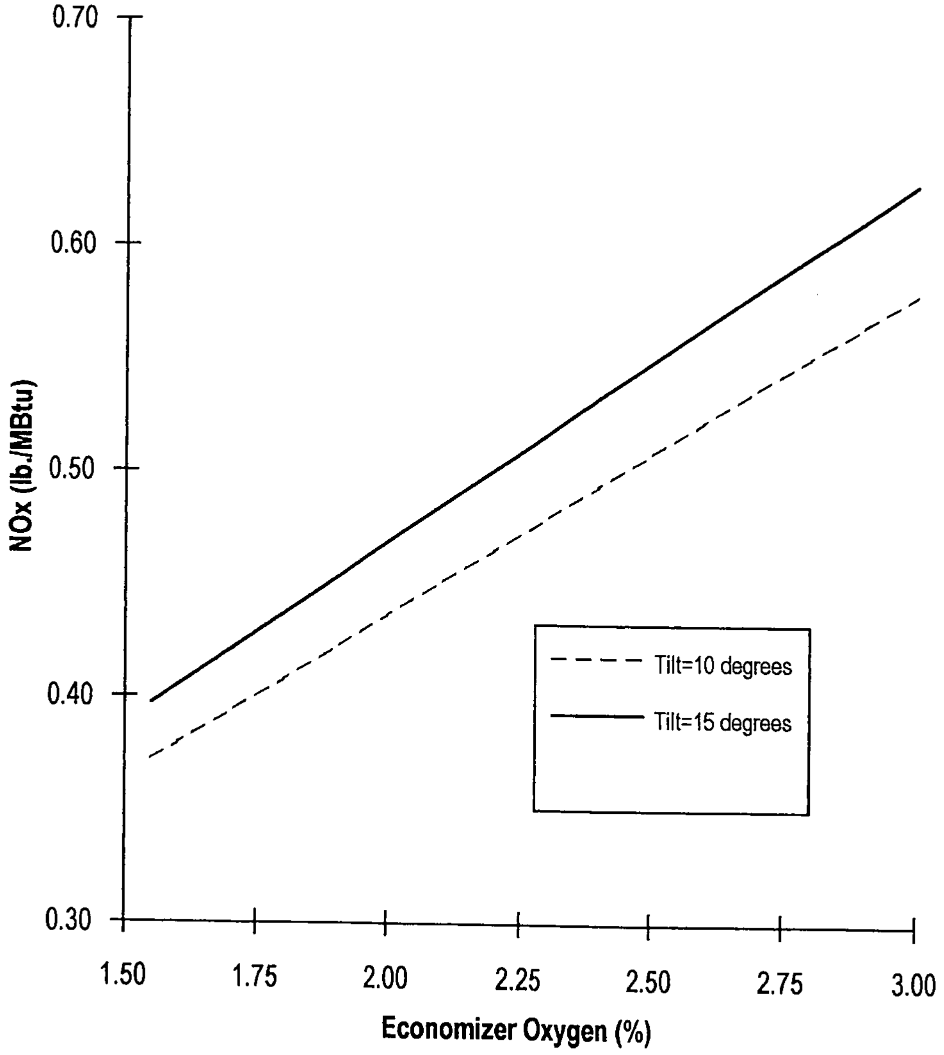

Figure 5.7

Full Load Predicted NOx vs. Economizer Oxygen at Constant Burner Tilt Angle, Auxiliary Air Bias=0.5, and Fuel Air Bias=1.2

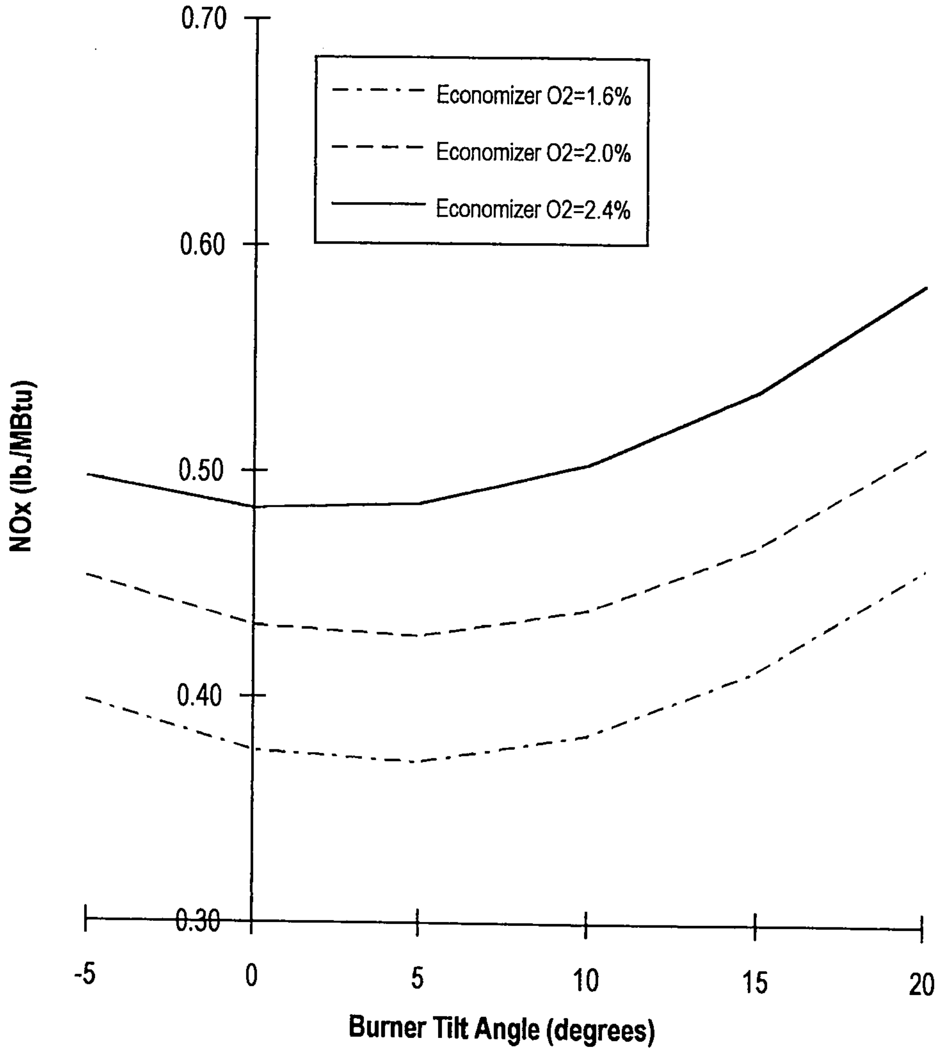

Figure 5.8

Full Load Predicted NOx vs. Burner Tilt Angle at Constant Economizer Oxygen, Auxiliary Air Bias=0.5, and Fuel Air Bias=1.2

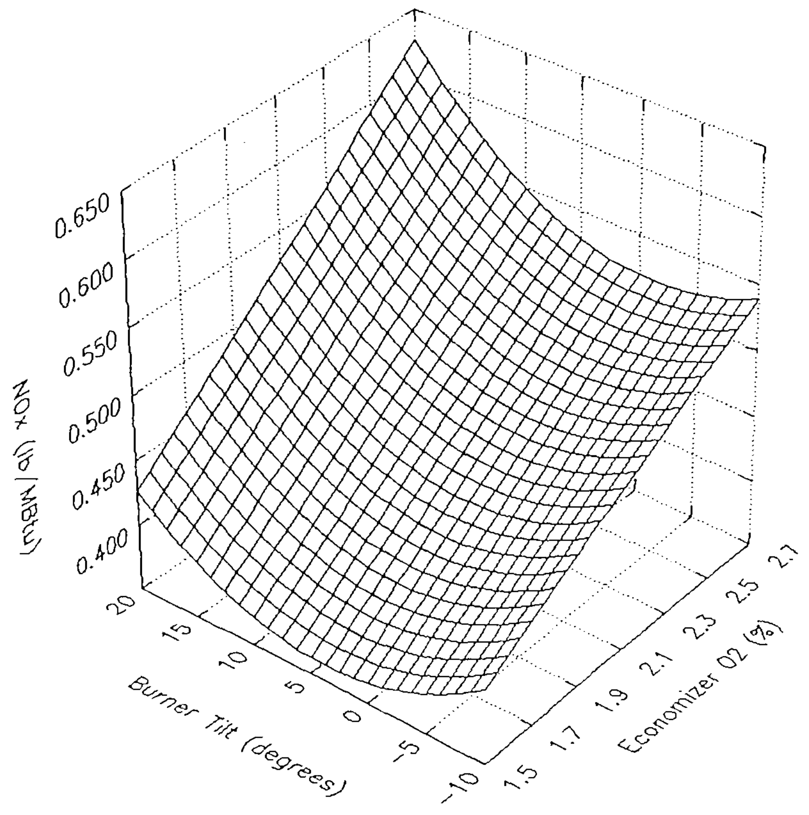

Figure 5.9

Full Load Predicted NOx vs. Economizer Oxygen and Burner Tilt Angle at Auxiliary Air Bias=0.5, and Fuel Air Bias=1.2

different constant parameter values along with a fuel air bias of 1.2. The minimum NOx is still dictated by the lowest oxygen level and the ten degree burner tilt angle. When the tilt is held constant and the oxygen level is raised, the curve shifts upward, denoting an increase in NOx. Furthermore, at constant tilt, the two curves have identical parabolic shapes. Overall, it is apparent that a moderate degree of overfire air at these different conditions produces the lowest NOx. The minimum NOx occurs at an auxiliary air bias of 0.583, which corresponds to auxiliary air damper positions listed in Appendix D.

The final figure in this series, Figure 5.11, shows a three dimensional plot of the burner tilt and auxiliary air bias interaction in producing NOx. This surface shows the two dimensional parabolic curves of burner tilt and auxiliary air bias forming a three dimensional, concave, parabolic solution surface. The minimum NOx occurs at a burner tilt of approximately +7 degrees and an auxiliary air bias between 0.5 and 0.6. If a fourth dimension could be viewed, one would see that this parabolic surface would linearly shift up or down depending upon the economizer oxygen level. Therefore, the optimal conditions for full load boiler operation, based upon these plots, are the above settings for tilt and auxiliary air bias at the minimum safe economizer oxygen level possible. This corresponds exactly with the findings of D'Agostini, et al. [4]. The fuel air bias did not play a significant role at this load level in neither the neural network model nor in D'Agostini's findings.

Heat Rate Neural Network

Modeling the boiler heat rate with neural networks was not as successful, in terms of accuracy, as modeling the creation of NOx. However, enough training data was available for the network to create a fairly well-generalized mapping of boiler settings to unit heat rate. The same inputs were used in the full load heat rate network as in the full load NOx network. This was done so the two networks could be compared more accurately when using all available training data and under reduced and clustered data conditions. The biggest obstacle to achieving a more accurate

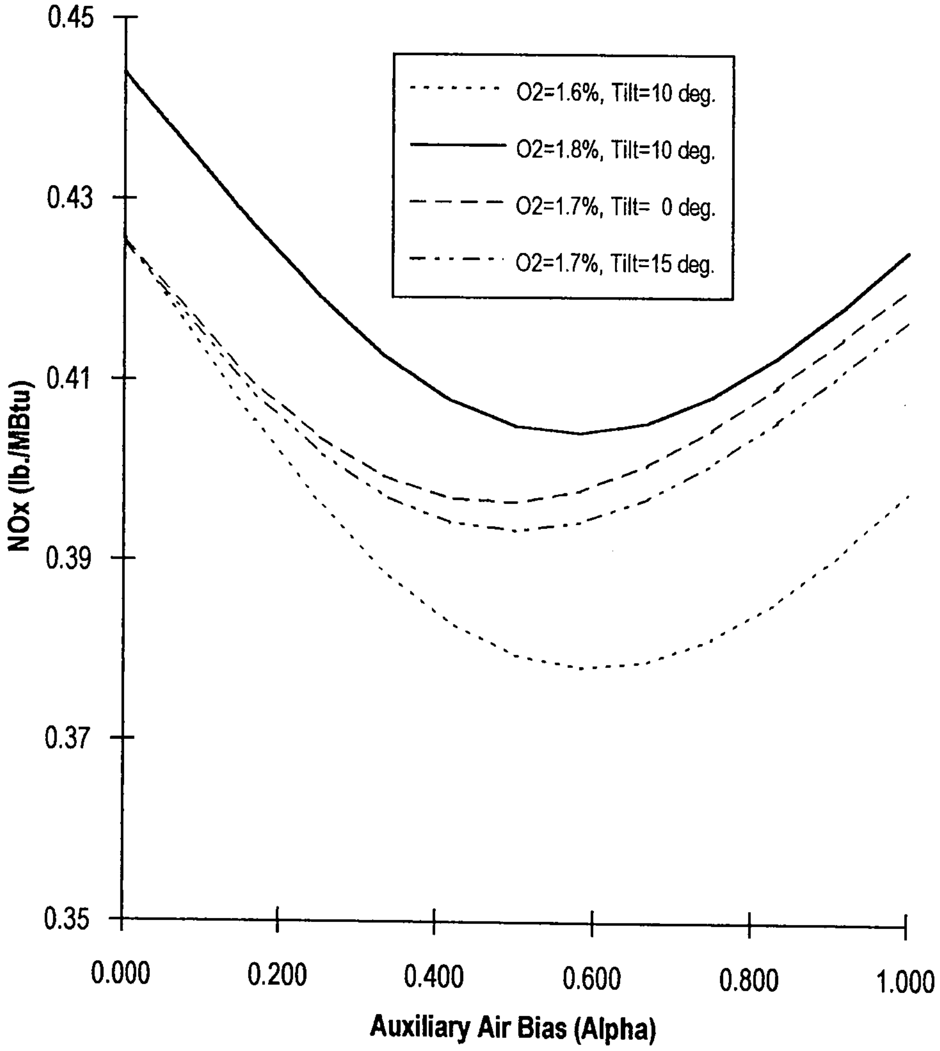

Figure 5.10

Full Load Predicted NOx vs. Auxiliary Air Bias at Constant Economizer Oxygen, Burner Tilt Angle, and Fuel Air Bias=1.2

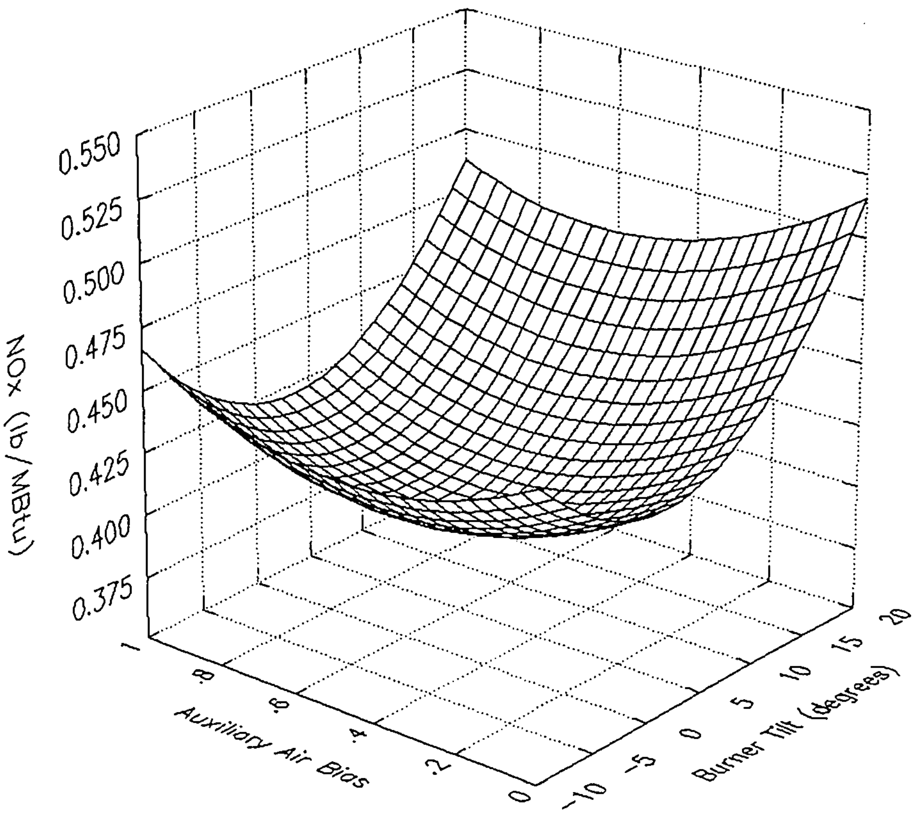

Figure 5.11

Full Load Predicted NOx vs. Burner Tilt Angle and Auxiliary Air Bias at Economizer Oxygen=1.8%, and Fuel Air Bias=1.2

mapping is the lack of a boiler cleanliness factor. As discussed in Chapter 2, the heat transfer within the boiler is affected by slag deposits upon the waterwall tubes. This, in turn, affects the steam temperatures and thus unit heat rate. Since no cleanliness factor was available, the training data contained cases of similar boiler input values which resulted in very different unit heat rate values. This confused the neural network during training and therefore the model's accuracy is not as good as it could be if there were a distinguishing input parameter to characterize boiler cleanliness.

The reader may wonder why a network using both NOx and heat rate as outputs was not attempted. This option was attempted, but a few factors dictated the use of separate networks. The first was the lack of training data. A combined output network resulted in significantly more connection weights and thus required more data to properly adjust these weights to optimal operating conditions. Second, the lack of a boiler cleanliness factor would confuse the network and decrease its mapping potential, as just described. It was deemed better to have only one output affected by this missing parameter instead of two. Finally, the numerical optimization algorithm only allows for one function to be optimized at a time. For this reason, the heat rate neural network output equation was optimized while the NOx network equation was used as a penalty function, as discussed in Chapter 4.

Network Settings and Performance Results.

Table 5.7

Full Load Unit Heat Rate Neural Network Optimal Settings

Number of Hidden PEs:	3
Transfer Function: Hidden Layer / Output Layer	Tanh / Linear
Epoch Size:	12
Learn Count:	70,000
Learn Rate: Hidden Layer / Output Layer	0.3 / 0.1
Momentum: Hidden Layer / Output Layer	0.4 / 0.4
Error Function:	Standard
Learning Rule:	Norm-Cum-Delta
Summation Function:	Normal Sum

Table 5.8

Full Load Unit Heat Rate Neural Network Performance Results

Performance Measure	Input Percent Contribution
R-Square = 0.5890	Economizer O2 = 16.5%
Adj. R-Square = 0.5524	Burner Tilt Angle = 33.9%
Avg. % Error = 0.2941	Auxiliary Air Bias = 27.9%
Avg. Difference =26.5 (MBtu/kWh)	Fuel Air Bias = 21.7%

Table 5.8 highlights the decreased accuracy of the heat rate neural network compared to the NOx network, at full load. However, considering the amount of scatter in the training data and the lack of a boiler cleanliness factor, an average difference of ≈27 Btu/kWh is surprisingly low. The unit heat rate also has lesser dependence on economizer oxygen levels than the NOx network but a greater dependence upon the auxiliary and fuel-air biases. The parameter that links the NOx and heat rate networks together is the burner tilt angle, which is detailed in the following section.

Unit Heat Rate Neural Network Trends vs. Observed Physical Trends. The same approach as in the NOx network case was taken to graph the trends within the heat rate network equations. The first, Figure 5.12, shows the predicted heat rate as a function of economizer oxygen for all of the test conditions used in training the neural network. These results show that while heat rate depends on economizer oxygen, it is also strongly affected by other operating parameters. Indeed, Figure 5.13, shows a strong dependency of heat rate on burner tilt angle. The curve fit applied is parabolic but the plot could also be represented by a straight line. In either case, the unit heat rate shows improvement with increasing tilt.

Physical trends at constant parameter conditions were graphed starting with economizer oxygen in Figure 5.14. This graph shows that as the burner tilt increases, the dependency of heat rate on economizer oxygen decreases. This is indicated by the lessening degree of slope in the linear fits as the tilt increases. As with NOx, the unit heat rate increases with increasing levels of economizer oxygen and the minimum heat rate occurs at the minimum safe oxygen level. This graph was plotted using the same constant auxiliary air bias value of 0.50 and fuel-air bias parameter of 1.2.

Figure 5.15 and 5.16 depict the interaction of auxiliary air bias and burner tilt angle. At constant levels of economizer oxygen, the trend of decreasing heat rate with increasing burner tilt is reinforced. Figure 5.15 shows this relation at a moderate degree of overfire air. Figure 5.16 shows the same conditions except that the auxiliary air bias is increased to 0.83, a higher degree of overfire air. At an economizer oxygen level of 1.6%, this increase in overfire shifts the curve down ≈90 Btu/kWh, a significant decrease. However, at an economizer oxygen level of 2.0%, this increase in the amount of overfire air results in a maximum heat rate increase of ≈50 Btu/kWh. These results suggest that if the minimum safe economizer oxygen level is below 2.0%, then increasing the amount of overfire air should decrease the unit heat rate. The more oxygen present in the system, the less overfire air should be applied to the boiler operating conditions.

The dependency of unit heat rate on economizer oxygen and tilt is shown in Figure 5.17. It combines the linear dependency of heat rate on oxygen and the parabolic dependency on burner tilt angle. The flattened portion of the surface in the upper right corner is due to the upper limit of the dependent axis scaling value (set at 9100 Btu/kWh for graphical presentation purposes), not because of the neural network model. Again, it is surmised that a minimum economizer oxygen gives a minimum heat rate. Burner tilt angle, however, can be anywhere within the range of zero to twenty degrees above the horizontal without substantial increases in unit heat rate. This is important for the numerical optimization process since the degree of burner tilt impacts strongly on the amount of NOx produced.

Figure 5.18 shows the unit heat rate dependency on auxiliary air bias in better detail. At lower oxygen levels and higher burner tilt angle, one expects the heat rate to be lower. In both curves it is also seen that the amount of overfire air has a point where adding more overfire air to the boiler increases the unit heat rate. This point occurs within the auxiliary air bias range of 0.583

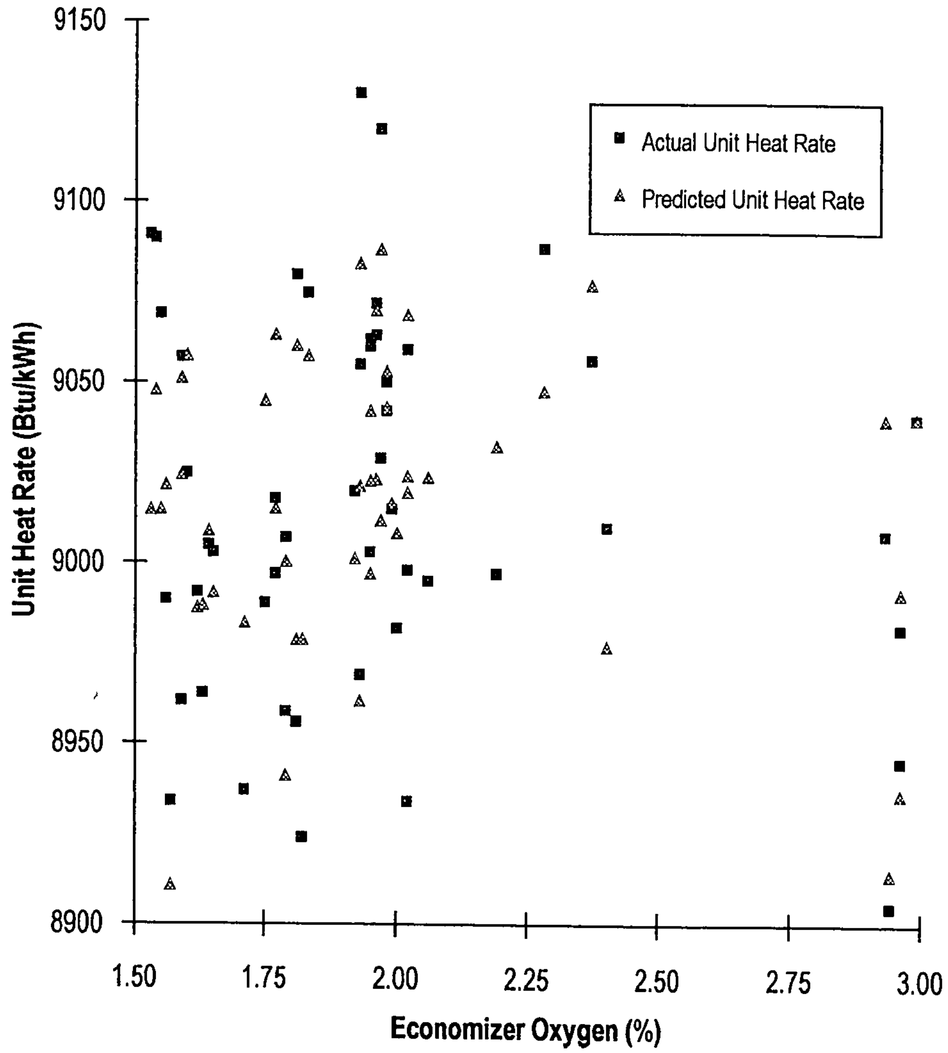

Figure 5.12

Full Load Actual and Predicted Unit Heat Rate vs. Economizer Oxygen

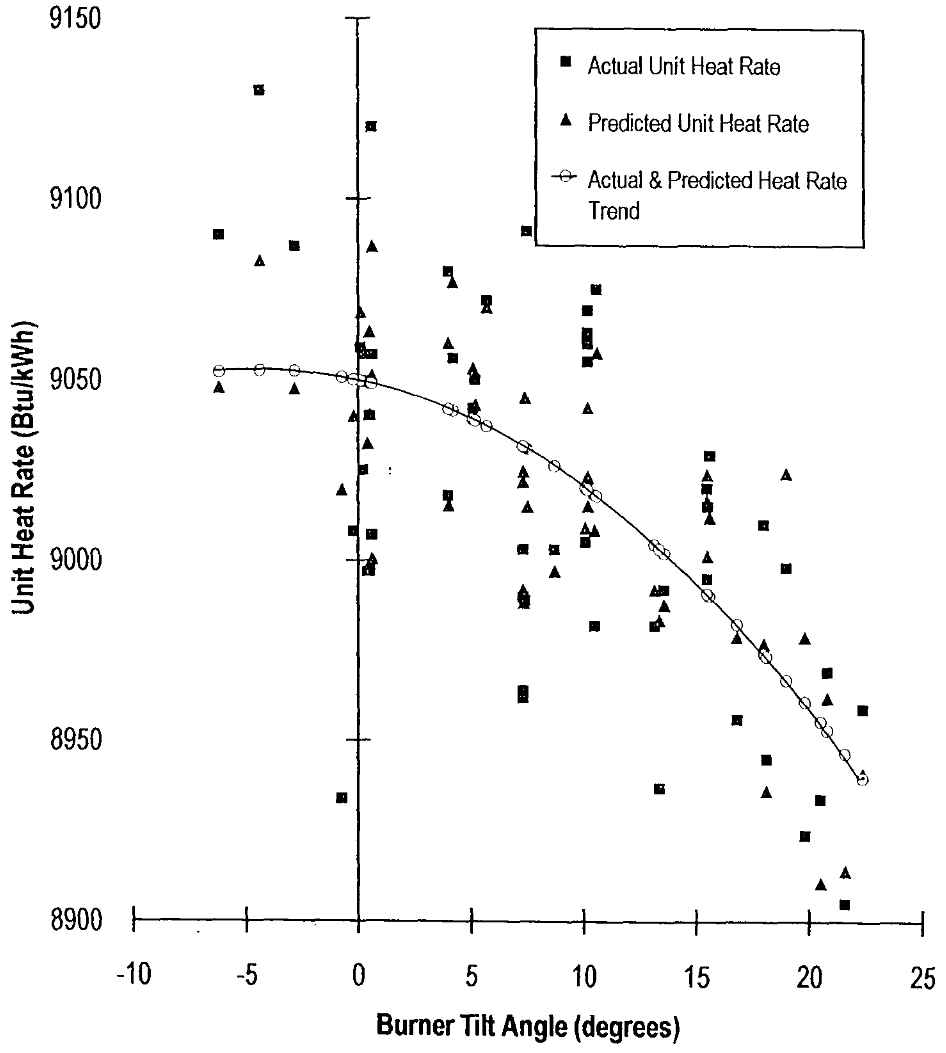

Figure 5.13

Full Load Actual and Predicted Unit Heat Rate vs. Burner Tilt Angle

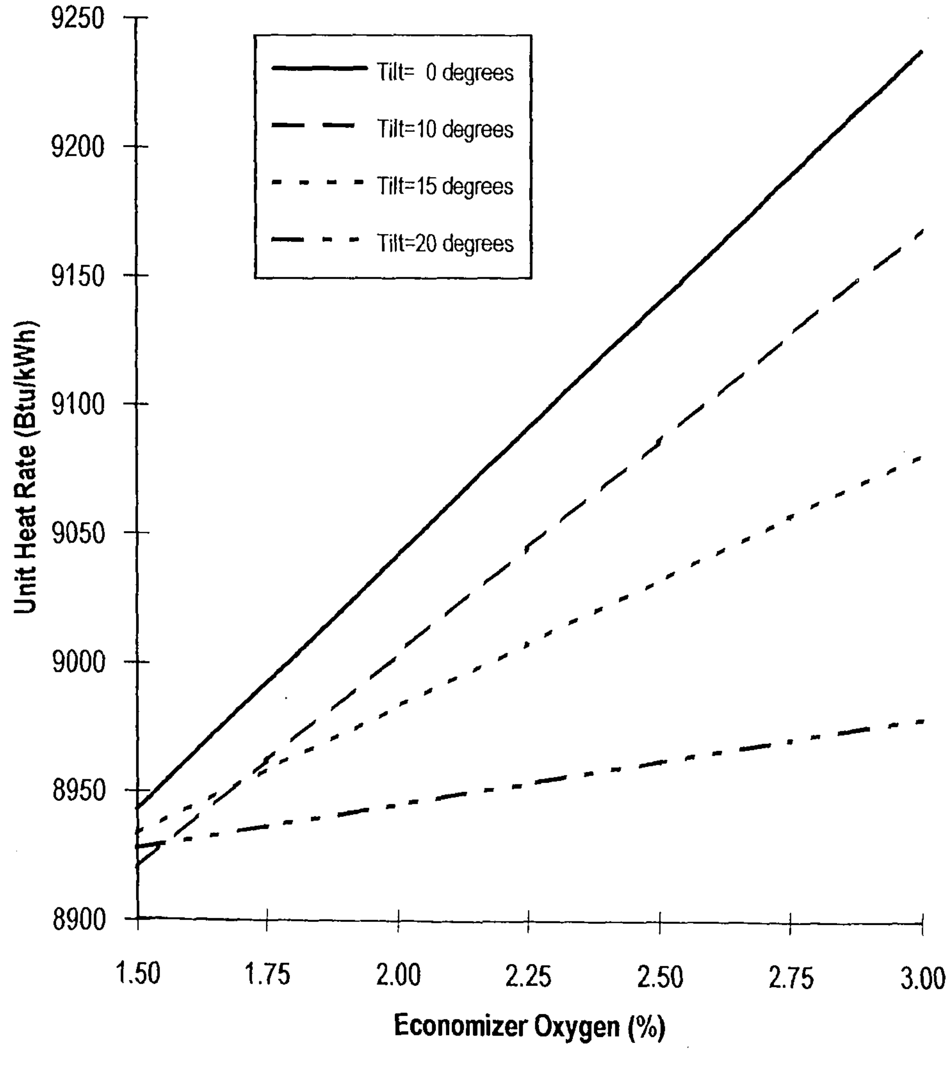

Figure 5.14

Full Load Predicted Unit Heat Rate vs. Economizer Oxygen at Constant Burner Tilt Angle, Auxiliary Air Bias=0.5, and Fuel Air Bias=1.2

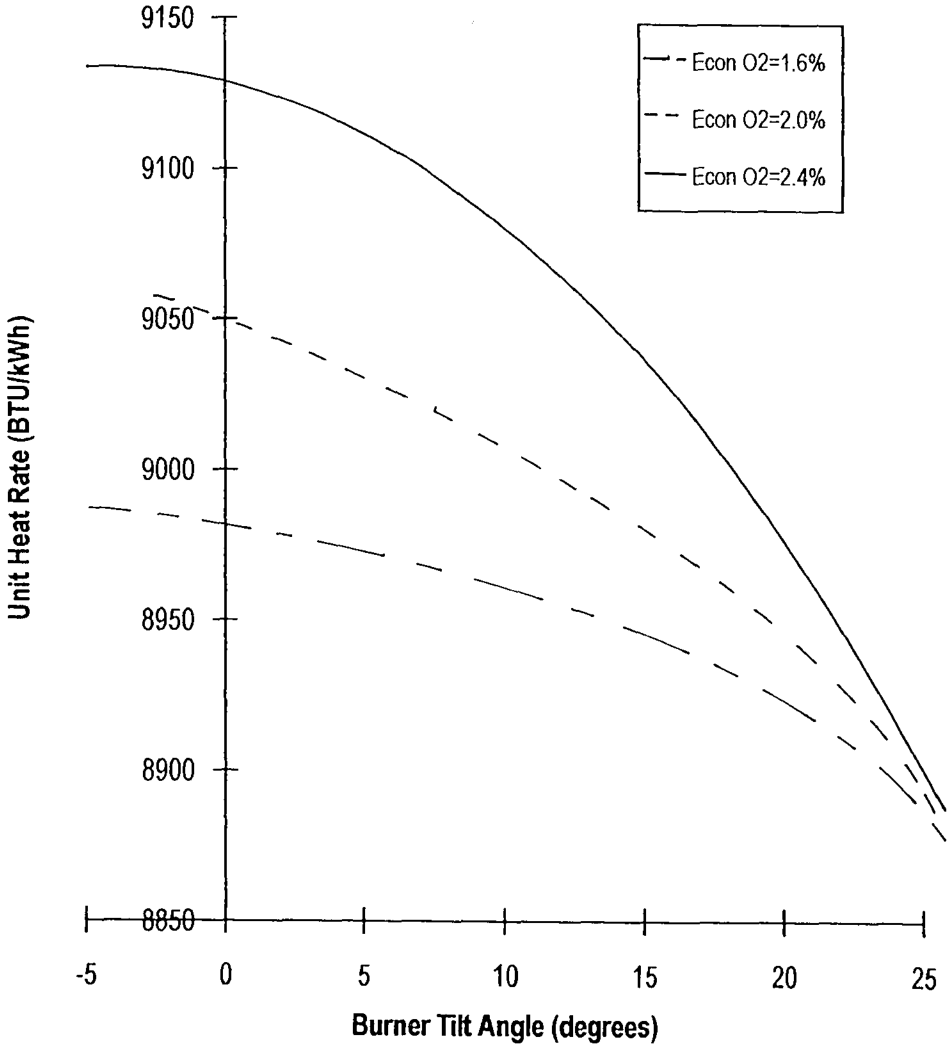

Figure 5.15

Full Load Predicted Unit Heat Rate vs. Burner Tilt Angle at Constant Economizer Oxygen, Auxiliary Air Bias=0.5, and Fuel Air Bias=1.2

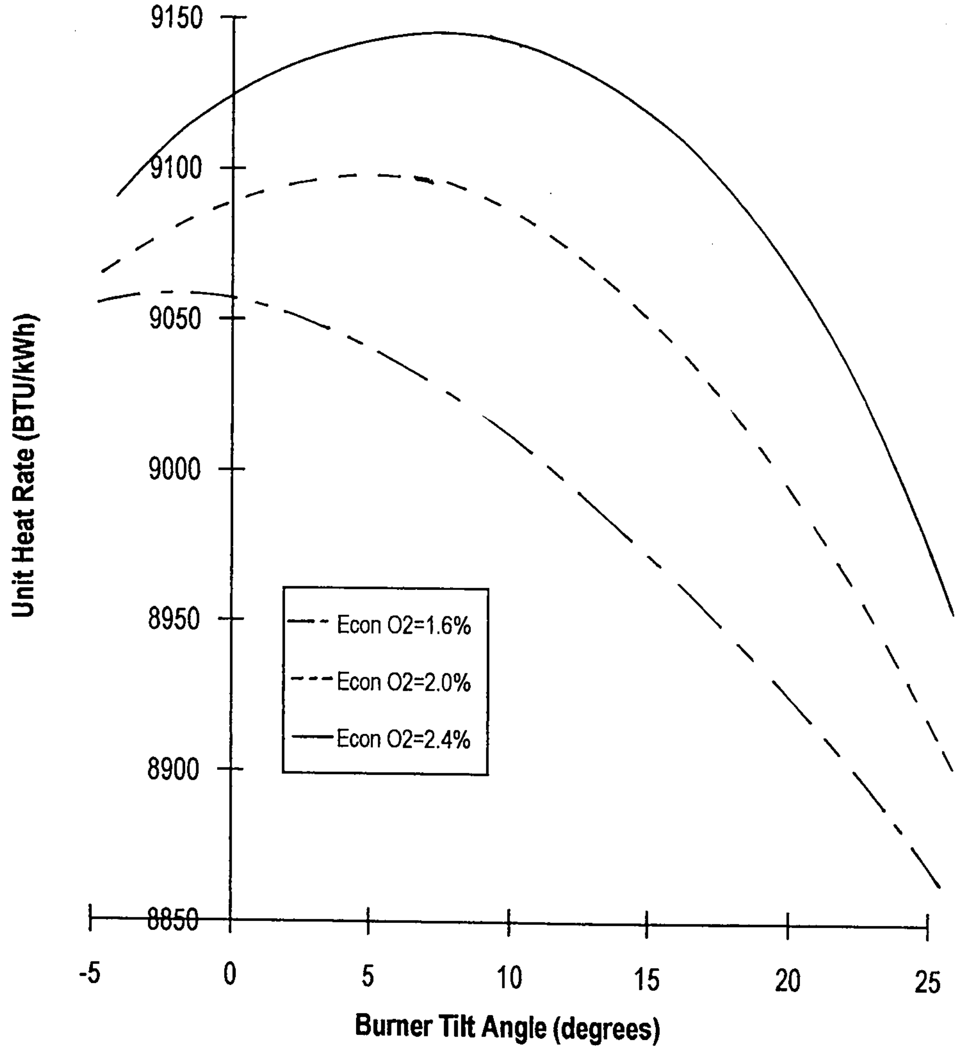

Figure 5.16

Full Load Predicted Unit Heat Rate vs. Burner Tilt Angle at Constant Economizer Oxygen, Auxiliary Air Bias=0.83, and Fuel Air Bias=1.2

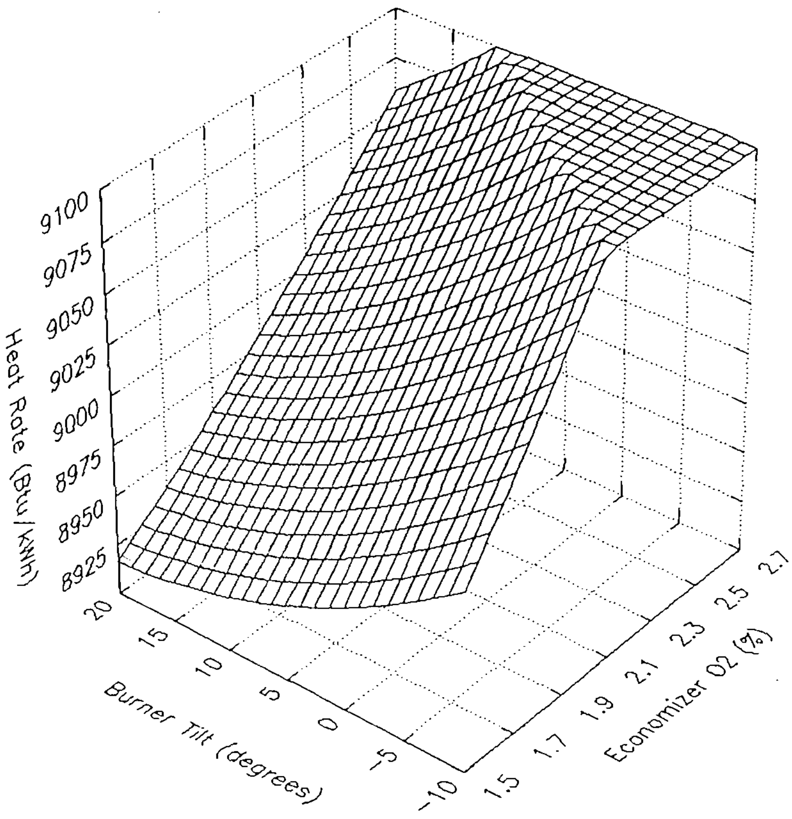

Figure 5.17

Full Load Predicted Unit Heat Rate vs. Economizer Oxygen and Burner Tilt Angle at Auxiliary Air Bias=0.5, and Fuel Air Bias=1.2

to 0.667. The damper positions for these biases are listed in Appendix A and correspond to a moderate amount of overfire air.

The combined effects of burner tilt angle and auxiliary air bias are plotted in Figure 5.19. As with NOx, this solution surface would shift up and down the heat rate (z) axis depending upon the amount of economizer oxygen. Furthermore, as more oxygen is introduced, the auxiliary air bias curve would steepen, as seen in the previous figure, thus causing an increase in unit heat rate.

Finally, in Figure 5.20, the predicted unit heat rate and NOx are plotted against each other as in Figure 2.15. The scatter plots are similar and the basic trend inferred from both plots is that as the NOx level increases, the unit heat rate decreases. This final check was done to see how well the output of both neural network models compared to observed data, since each network was generated independently of the other.

It is evident that the boiler optimization problem has been successfully transposed. The full load neural networks have created a multivarible curve fit consisting of four independent parameters. Now, instead of optimizing actual boiler operation, the user now optimizes a mathematical model of the boiler. The correlations derived by D'Agostini [4] in Table 2.2 used multivariable regression techniques which required a great deal of user knowledge, interaction, and time to formulate them.

Full Load Reduced and Clustered Data Experiment Results

The last topic, in the full load neural network study, was to examine the effects of smaller training data sets and clustered training data on neural network performance. This was done to verify the sensitivity of neural network training on data quantity and quality. Originally, fifty data vectors comprised all of the available full load data. These fifty points were reduced to training data sets of 37, 25, and 13 data vectors reflecting a 25%, 50%, and 75% reduction in the amount of data, respectively. For each reduced data set, a NOx and unit heat rate neural network, with the

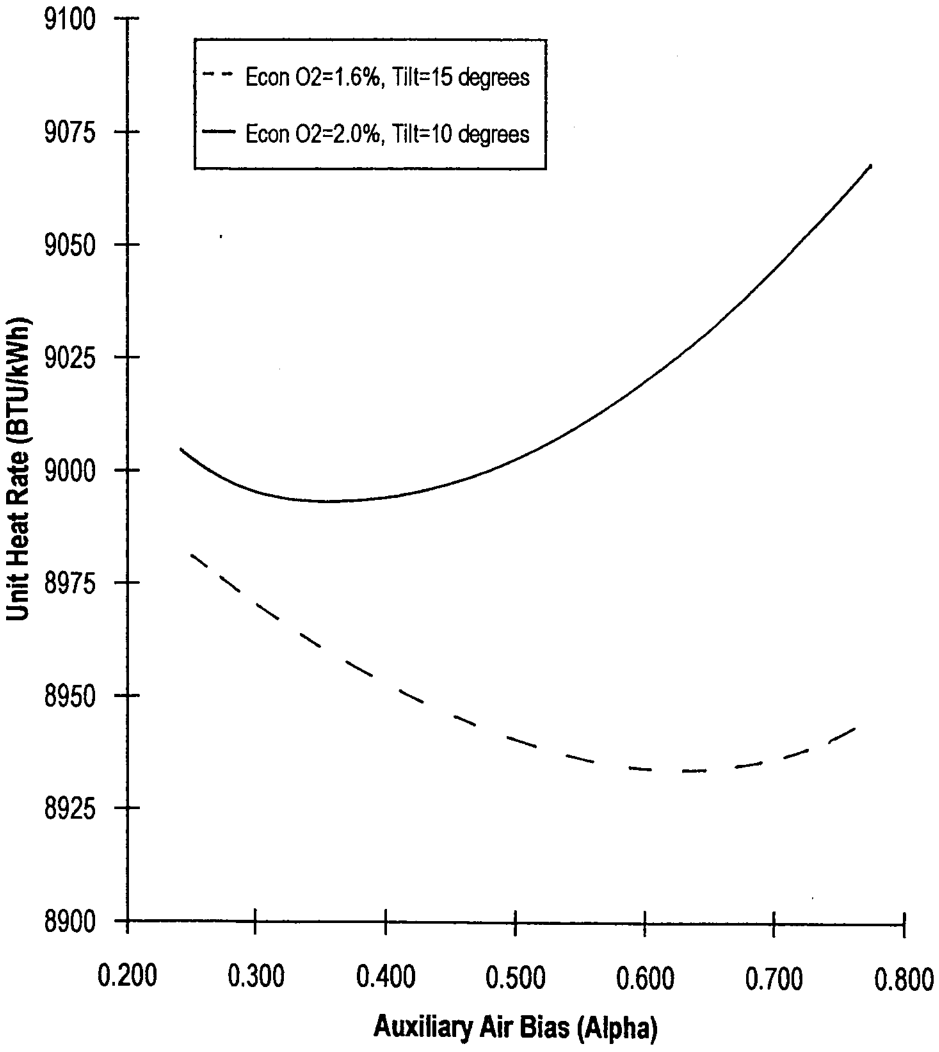

Figure 5.18

Full Load Predicted Unit Heat Rate vs. Auxiliary Air Bias at Constant Economizer Oxygen, Burner Tilt Angle, and Fuel Air Bias=1.2

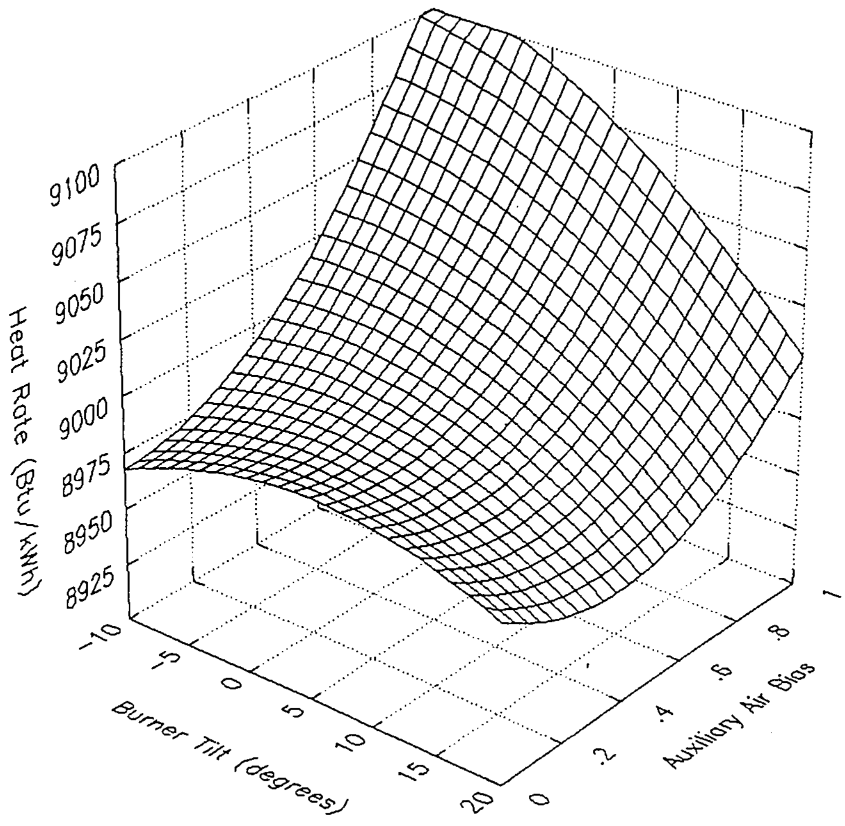

Figure 5.19

Full Load Predicted Unit Heat Rate vs. Burner Tilt Angle and Auxiliary Air Bias at Economizer Oxygen=1.8% and Fuel Air Bias=1.2

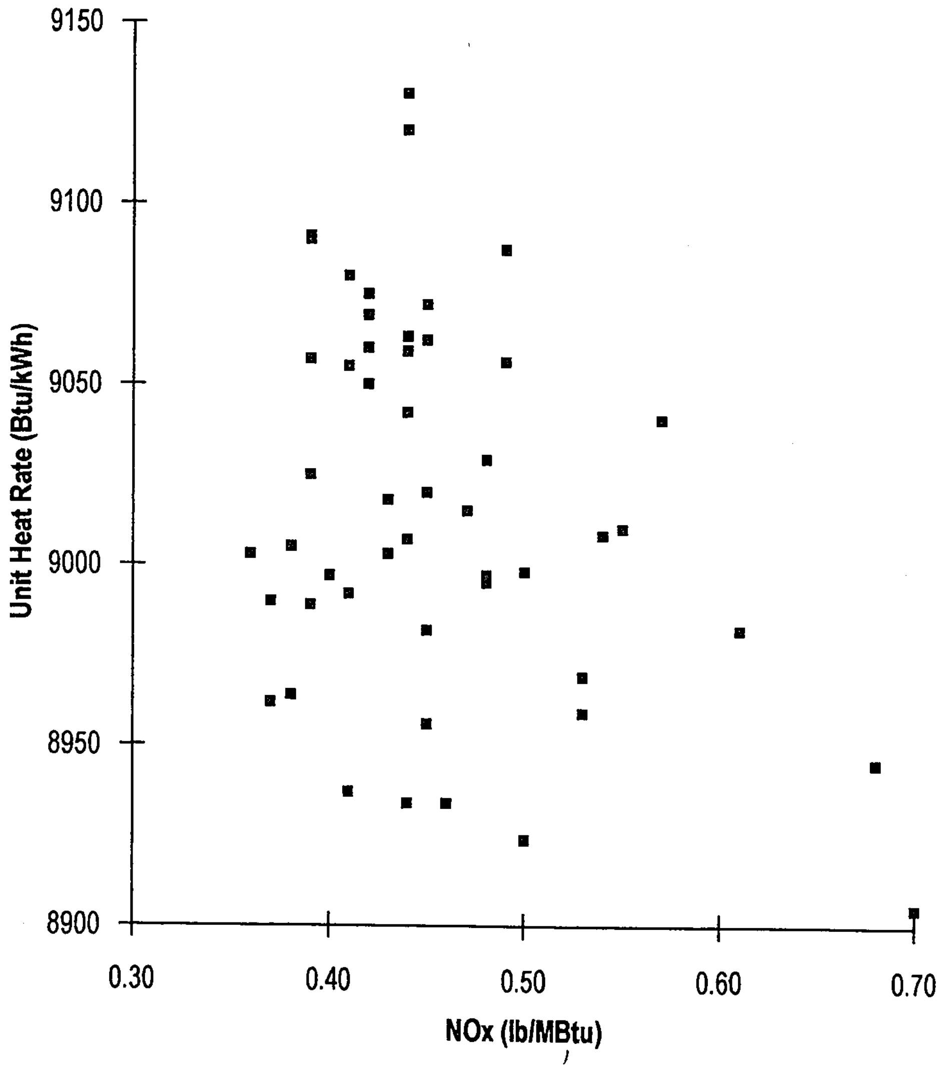

Figure 5.20

Full Load Predicted Unit Heat Rate vs. Predicted NOx same architectures used previously, were trained and optimized. This was accomplished using the same strategy described earlier in this chapter.

After the networks were optimized, they were tested using 10 out of the 13 data vectors that were not included in any of the reduced training data sets. These ten vectors were selected such that all data vector components were within the same ranges as the training data vector components. This prevented any of the networks from being tested with data that were not within the training data boundaries, a situation that would skew the network performance. After optimizing the network's operation and conducting the performance test runs for the training and testing data sets, the usual performance parameters were calculated. Each parameter was plotted against the percent of available data used to train the network. This showed the effects of reducing the training data set size on the networks' performance capability.

The clustered data experiment was done to examine the effects of using data clustered around a small range of one input parameter. Due to the small amount of available data, this could only be done with the economizer oxygen input parameter. This data vector component was the only one which had a sufficient amount of values within a reduced range. This range consisted of 24 data vectors which had an economizer oxygen value between 1.75% and 2.0%. This clustered data set was used to train a NOx and heat rate neural network using the same network architecture and optimization process as in previous computational experiments. The final performance parameters were calculated for the training and testing data sets and these values were tabulated to show the differences in network performance.

Figure 5.21 shows the effects of reduced and clustered data on the NOx neural network's R-Square performance statistic. The R-Square is the statistic that ranks the network's accuracy, between zero and one, to predict output responses to input data it has seen (training data) and has not seen (testing data). If a network is trained on a smaller data set, it more easily learns the fewer number of numerical relationships between the inputs and their corresponding outputs. Thus, the R-Square for the training data improved as fewer data were used for training. However, the network learns less about the overall system behavior since the training data do not sufficiently represent the "big picture" and overtraining commonly occurs. Overtraining is quickly identified by the network's performance in predicting outputs for the test set inputs. As the amount of training data is reduced, the network's ability to learn the general relations and interactions of the inputs and output decreases. This explains why the network performs well on predicting outputs for the training data but poorly on the test data.

Figure 5.22 and 5.23 depict the average difference and the average percent error versus the amount of training data, respectively. The average difference and average error of the training set decrease with smaller data set sizes while the test set results show an increase in the prediction error. These are caused by the same reasons just described. In each instance, the NOx network has a performance "breakpoint" around 75% of the available data for the R-Square and 50% for the average difference and average percent error. This means that, out of 50 available data vectors from Potomac River testing, between 50% and 75%, or 25 to 35 data vectors, are necessary for achieving an acceptable neural network mapping of boiler inputs to NOx output. The author concluded that six to eight data vectors are thus required for each input for the full load NOx network.

Training a network on clustered data gave a similar response to that of the reduced data experiment. The full load data were not conducive to training a network with data vectors clustered within a small range. Only economizer oxygen had a sufficient amount of data points (24) that were within a small range (1.75% to 2.0%) for examining the effects of clustered data on neural network training. Since economizer oxygen was the only parameter that could be investigated, the author believes that the clustered data experiment was skewed due to the inability to cluster the data about other input parameters. Furthermore, because the NOx network equations were strongly linked to economizer oxygen, the final performance results for the training and testing data may be better than the results of trials conducted using different data clusters. This may lead one to suspect that the full load clustered data results are better than should be expected. The results of clustering

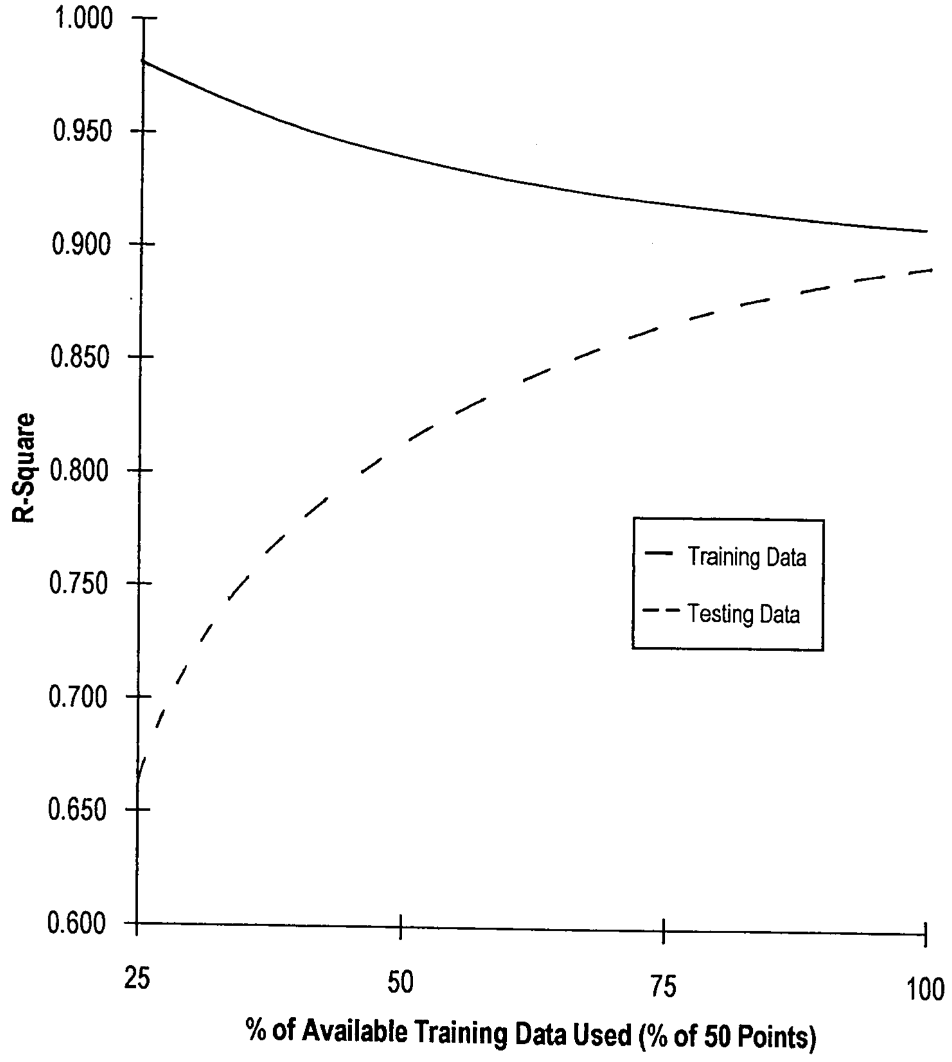

Figure 5.21

Full Load NOx Neural Network R-Square vs.
Percent of Available Training Data Used

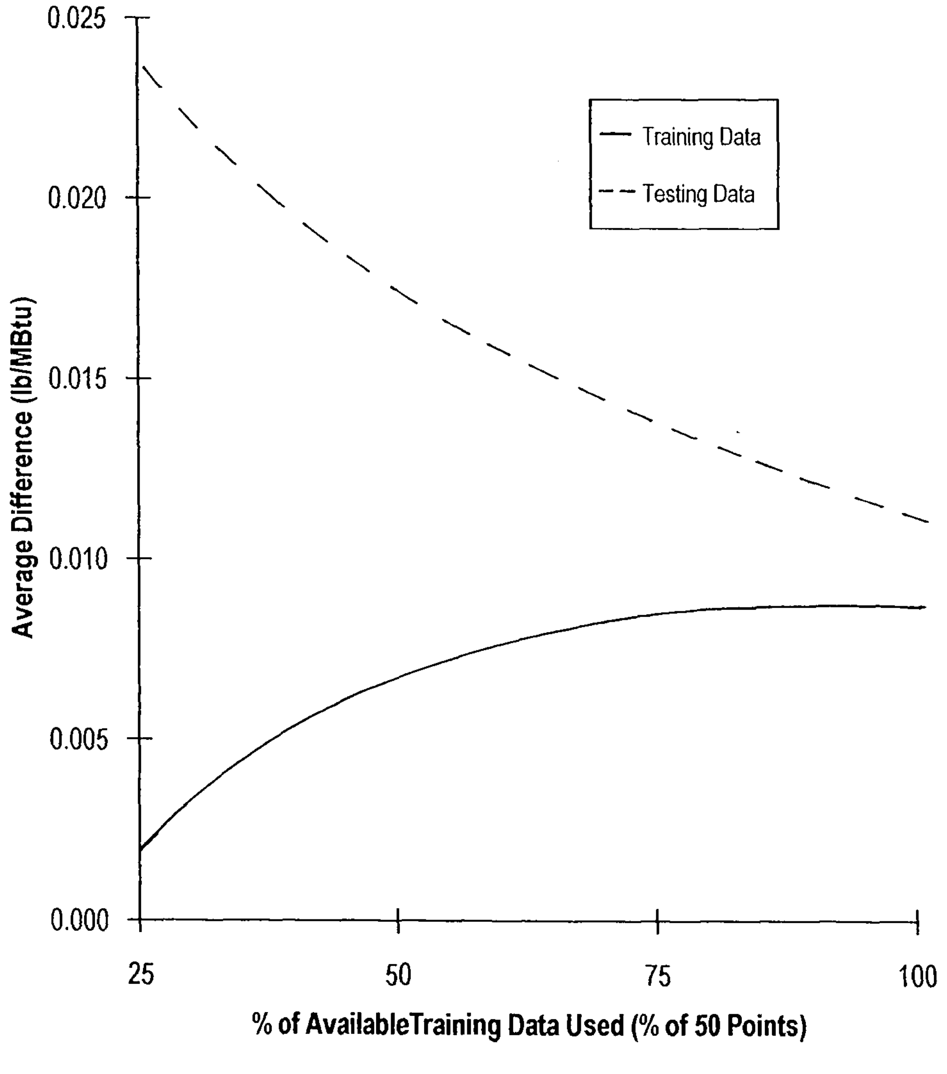

Figure 5.22

Full Load NOx Neural Network Average Difference vs.
Percent of Available Training Data Used

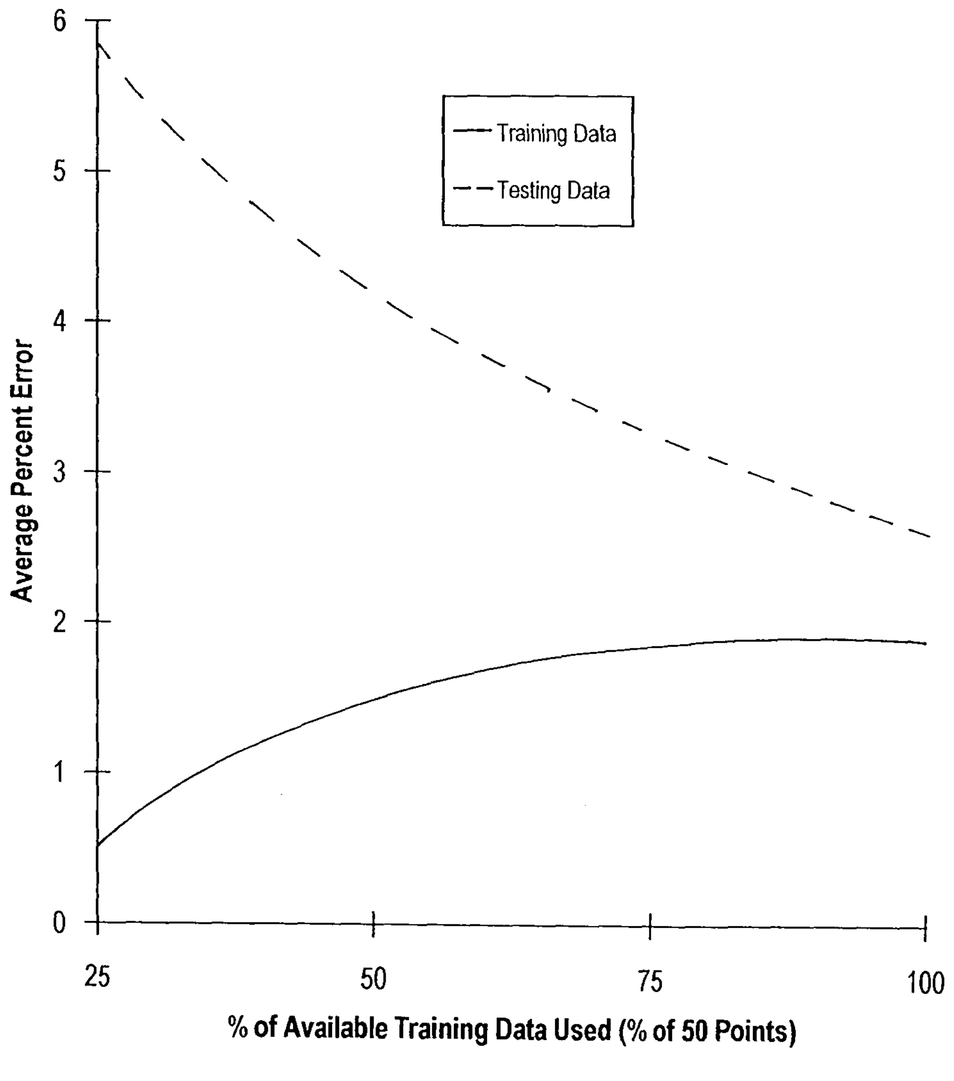

Figure 5.23

Full Load NOx Neural Network Average Percent Error vs. Percent of Available Training Data Used closely match those of the 50% reduced training data experiment because the clustered data set had 24 data points while the 50% reduced training set had 25 data points.

The same analysis was done for the unit heat rate network using reduced training data set sizes. These results are depicted in Figures 5.24, 5.25, and 5.26. In these cases, the "breakpoint" occurs closer to 75% of all training data available to achieve an accurate yet generalized mapping of inputs to outputs. This translates to a range of 32 to 40 data vectors, or 9 to 10 per input parameter. The same clustered data results for the NOx network were observed in the heat rate network, even though the effect of economizer oxygen was not as important to heat rate as it was for NOx. More experiments of this nature are required in the future when more parametric data become available from tests conducted at Potomac River. The clustered data network results are detailed in the following table:

Table 5.9

Comparison of Full Load NOx and Heat Rate
Reduced Data and Clustered Data Networks

NOx	50% Data	Clustered	Heat Rate	50% Data	Clustered
R-Square Train:	0.945	0.965	R-Square Train:	0.756	0.781
R-Square Test:	0.819	0.807	R-Square Test:	0.450	0.384
Avg. Diff Train:	0.007	0.006	Avg. Diff Train:	15.8	13.47
Avg. Diff Test:	0.018	0.020	Avg. Diff Test:	45.0	49.63
Avg.%Err Test:	1.424	1.242	Avg.%Err Test:	0.20	0.150
Avg.%Err Test:	4.190	4.381	Avg.%Err Test:	1.00	1.112

The author determined that, as a minimum, an available data set with 30 to 40 vectors, is sufficient to train and optimize both the NOx and unit heat rate networks at full load operation. This translates to having 7 to 10 data points for every input used in a neural network for this load level. This data set size can be reduced if the data quality for each input improves. This is accomplished by equal data distribution and more and/or better quantifying input variables, such as boiler cleanliness, for modeling boiler operation.

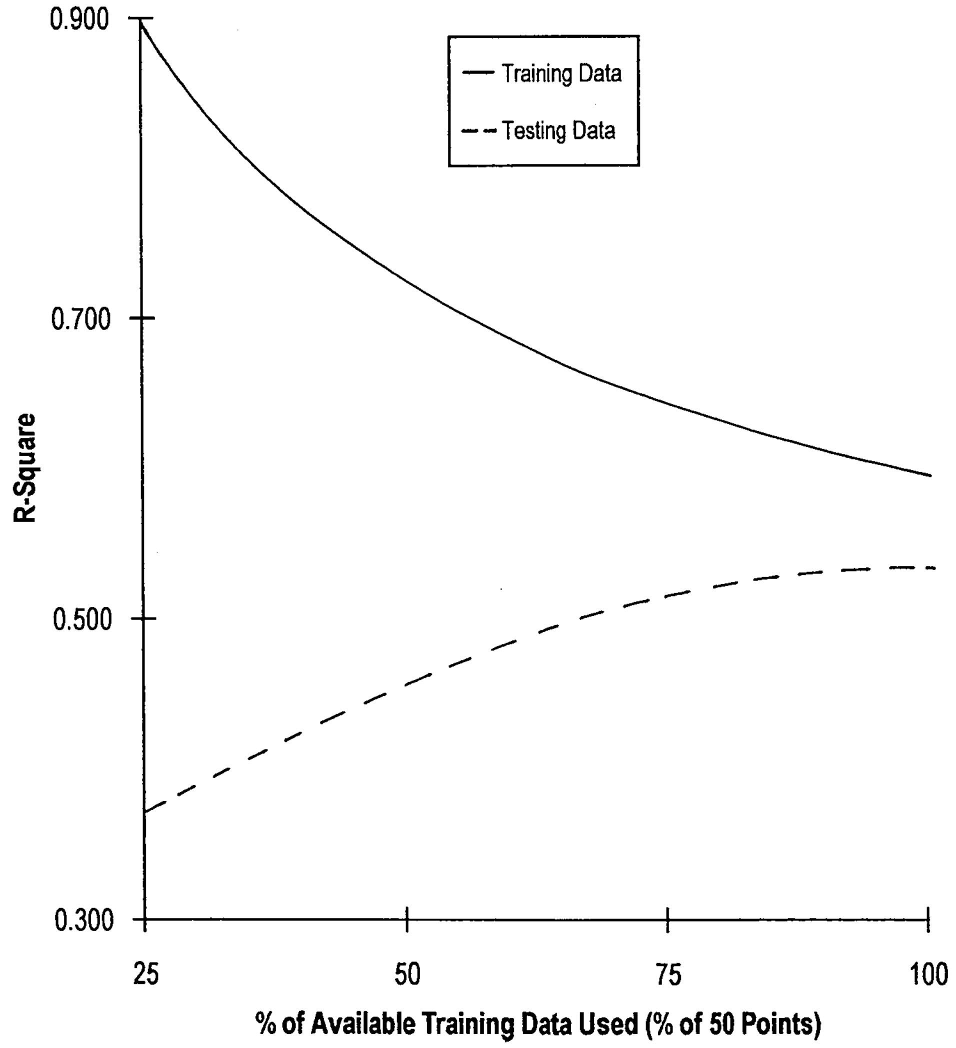

Figure 5.24

Full Load Unit Heat Rate Neural Network R-Square vs.
Percent of Available Training Data Used

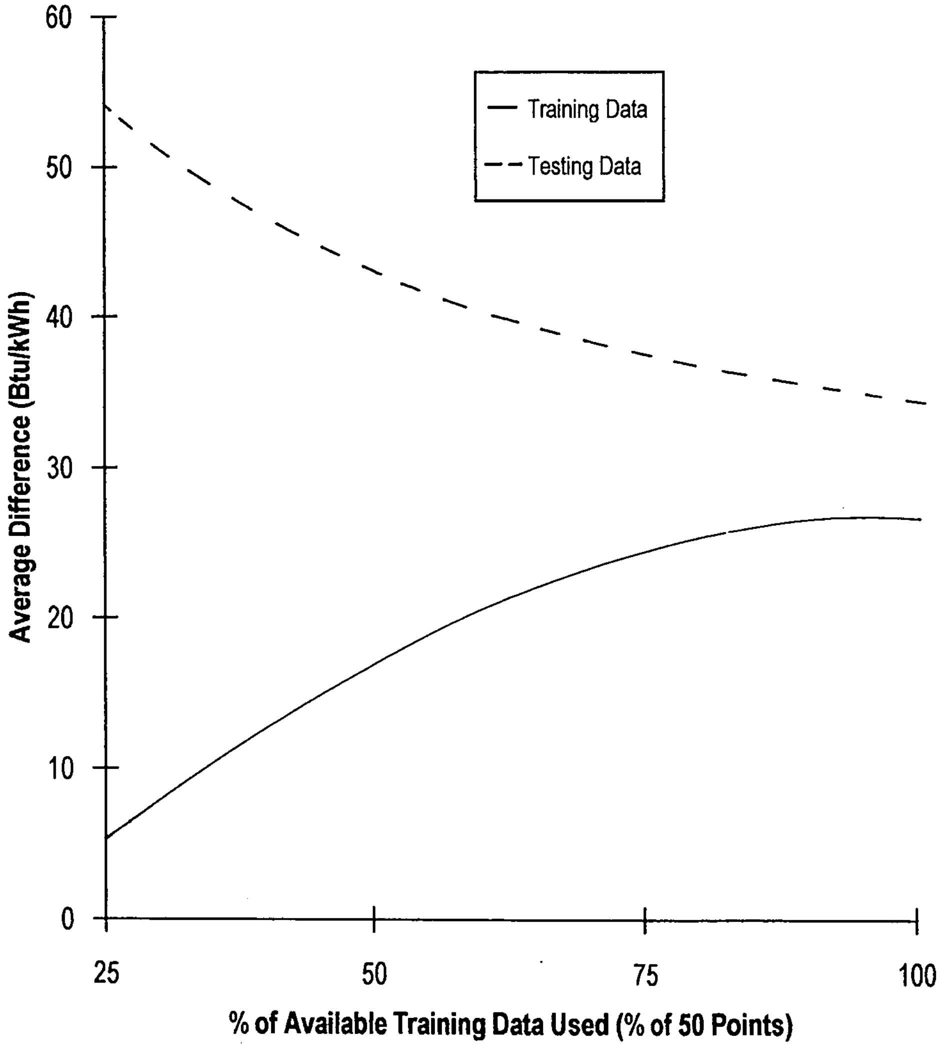

Figure 5.25

Full Load Unit Heat Rate Neural Network Average Difference vs. Percent of Available Training Data Used

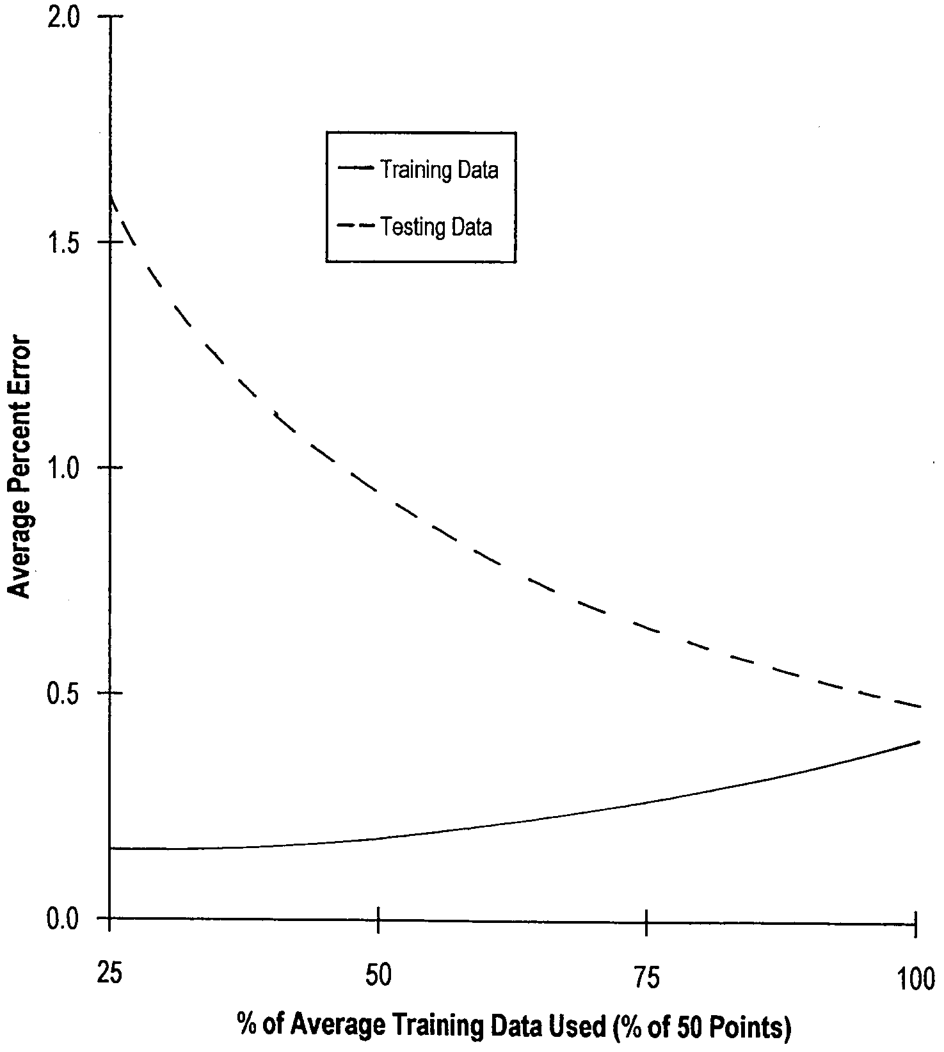

Figure 5.26

Full Load Unit Heat Rate Neural Network Average Percent Error vs. Percent of Available Training Data Used

45 MW NEURAL NETWORKS

At Potomac River the NOx emission level of 0.38 lb./MBtu is not difficult to achieve at the 45 MW load level. The main goal of testing at this load level was to maintain this low NOx mode and find the operating conditions that resulted in a minimum unit heat rate. Although the training, optimization, and analysis of the 45 MW neural networks was identical to the full load case, several differences had to be accounted for.

The major shortcoming at this load level was the lack of heat rate data. A total of 60 data vectors was available for training and testing neural networks at this load. However, the quality of the data was not nearly as good as in the full load case and, in addition, there were only 17 calculated heat rate values. This required the author to match the operating conditions of data vectors without heat rate values to those with heat rate values. The main and hot reheat steam temperatures were then compared and rough estimates for the missing unit heat rates were made. If the data vector that was missing a heat rate value had higher steam temperatures than its counterpart, then a lower heat rate value was assumed for it; otherwise, a higher heat rate value was assumed. Although this approach made it possible to complete the data set and train a heat rate neural network, the network performance was tainted by the lack of quality output data. This is seen in the results section for the 45 MW neural networks.

A significant difference in boiler operating conditions at 45 MW from full load is the mill loading pattern. At full load, all four mills must operate near or at capacity, whereas at 45 MW, the operator has a variety of mill loading options. The primary loading pattern is to use the two middle burner rows, B and C, for combustion. This mode of operation comprised 72% of the available data. The available data contained 16 data vectors, or 27%, where mill loading consisted of the A, B, and C burner rows. Only one test was conducted using the B, C, and D burner rows for combustion. Because of very low steam temperatures, this mode of operation was deemed unsafe by plant operators and engineers and therefore no further tests in this mode of operation were allowed.

One benefit of the 45 MW data was that several tests were conducted with all but one of the operating parameters held constant and with that one parameter changing incrementally. This occurred primarily for the auxiliary and fuel air damper positions. Unfortunately, with the different mill loading strategies being applied throughout the parametric testing, it is difficult to determine which parameter change, mill loading or air distribution, made the biggest impact upon the resulting NOx and heat rate networks.

NOx Neural Network

As in the full load case, the NOx network was used to make decisions governing the training and optimizing of subsequent networks. The majority of the decisions concerning the network optimization procedure were previously established during the full load NOx network portion of the study. The remaining decisions at the 45 MW load dealt primarily with input data selection and dividing the data into training and testing sets. The analysis for the 45 MW neural networks was the same as for full load networks and these results follow this section.

Input Selection. With the option of different mill loading strategies, the initial input selection consisted of economizer oxygen, burner tilt angle, auxiliary air bias (Eq. 2.7), fuel air bias (Eq. 2.16), and mill loading bias (Eq. 2.4). Separate neural networks were created and optimized for NOx and unit heat rate using these inputs. The performance results for both networks were very similar to their counterparts at full load, which was encouraging. However, the optimization of these networks proved much more difficult at 45 MW than at full load.. Using the IMSL optimization routine BCONF [31], the two networks yielded far different optimal conditions to achieve their individual minimums. These results were checked using spreadsheet calculations a different IMSL optimization routine, BCONG [31]. Both checks resulted in similar optimal conditions so further analysis was required. The processing element (PE) connection weights were used to calculate the input parameter contribution to the output values as described earlier in this chapter. The following table details the results of this analysis:

Table 5.10

45 MW NOx and Unit Heat Rate Network Comparison

	NOx Neural Network		Heat Rate Neural Network	
	Input Contribution	Optimal Settings	Input Contribution	Optimal Settings
02	8 %	3.9 %	20 %	4.81 %
Tilt	12 %	4.4 degrees	15 %	9.2 degrees
AA Bias	12 %	0.667	24 %	0.000
FA Bias	12 %	0.013	26 %	2.060
Mill Bias	56 %	0.25	15 %	0.333
NOx		0.28 (lb/MBtu)		0.79 (lb/MBtu)
Heat Rate		9811 (Btu/kWh)		9662 (Btu/kWh)

This reveals that mill bias is the controlling factor in the NOx network while heat rate depends nearly equally on all five independent parameters. A closer look at the available training data revealed that when the mill loading pattern was A, B, and C, changes in operating conditions had a more significant impact on NOx compared to the same changes made at the B and C mill loading pattern. Also, the mill bias parameter (Eq. 2.4) assigns a value with a small range to describe the physical impact of a change in mill loading which has a much greater physical range. This difference has a vital effect on the training of the neural networks and thus the resulting network equations. It is easily seen in the above table that the two networks' optimal conditions are quite different.

After using the three different methods (BCONF, BCONG, and spreadsheet calculations) to calculate the optimal conditions for each network separately, the equations were optimized together using the Simplex method [27]. The heat rate network equations could not be optimized to within the specified NOx limit of 0.38 lb./MBtu without unrealistic results caused by function extrapolation.

The testing conducted by D'Agostini, et al. [4], showed that the minimum NOx attained during parametric testing at 45 MW occurred when only the B and C mills were used (Figure 2.12). Furthermore, the fuel-air bias parameter (Eq. 2.16) already accounts for the mill feeder speeds and loading pattern. Finally, the heat rate values used were, for most data vectors, rough approximations which incurred substantial uncertainty to the network operation. Based upon this information, the author decided to forego using the mill bias parameter as an input for the 45 MW neural networks. The only way to eliminate the need for a mill bias parameter was to only use data vectors that had a mill bias equal to zero ($\beta = 0$). This was accomplished by discarding data vectors that contained feeder speeds for A or D mill. This action reduced the number of available data vectors for training and testing the 45 MW networks from 60 to 43. With an adequate amount of data vectors and each vector having a mill bias of zero, the need for a mill bias parameter for training and testing the 45 MW networks was removed. Thus, the independent variables for the 45 MW load networks would consist of the same parameters used at full load. An advantage to this approach is that the networks from both load levels could be compared on equal terms. Although not used in this study, the mill bias parameter is an important parameter for modeling NOx and heat rate at the lower load levels. Future testing and analysis is needed to improve the use of this parameter or develop a new parameter that better captures the effect of changing the mill loading scheme.

The choice of using data vectors which had only the B and C mill loading patterns reduced the amount of available training data vectors to 41. Of these 41 vectors, 13 had unit heat rate values calculated from the HEATRT code [8] with the remaining values being approximated. The available data was separated into a 6 data vector test set and a 35 data vector training set.

As done for the full load case, the data distribution for the 45 MW networks must be reviewed. This provides insight as to what data trends of boiler operation the neural networks learned. Compared to the full load case, the 45 MW data is poorly distributed. Figures 5.27 through 5.30 detail this and the only parameter with a fair data distribution is the fuel air bias parameter (Figure 5.30). This poor data distribution affects the heat rate neural network performance, especially when combined with the uncertainty in the unit heat rate values.

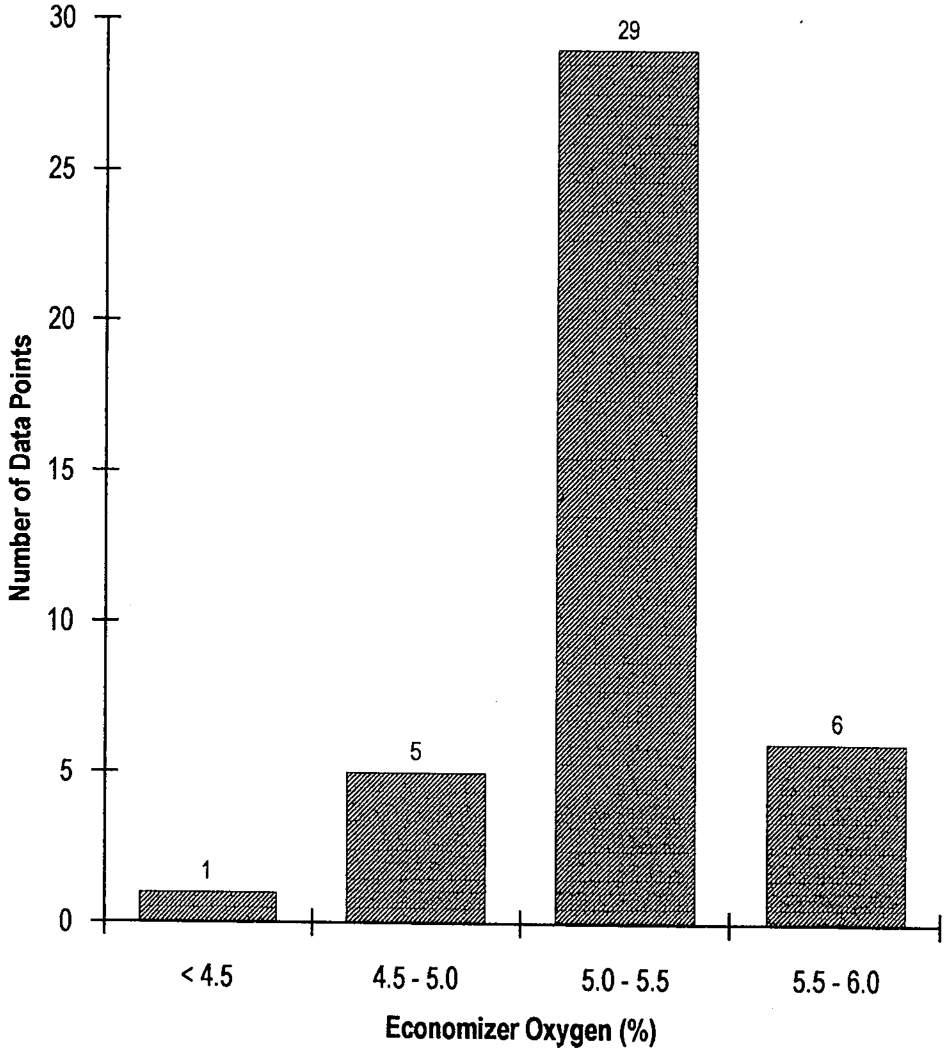

Figure 5.27

45 MW Available Neural Network Data Distribution
for Economizer Oxygen

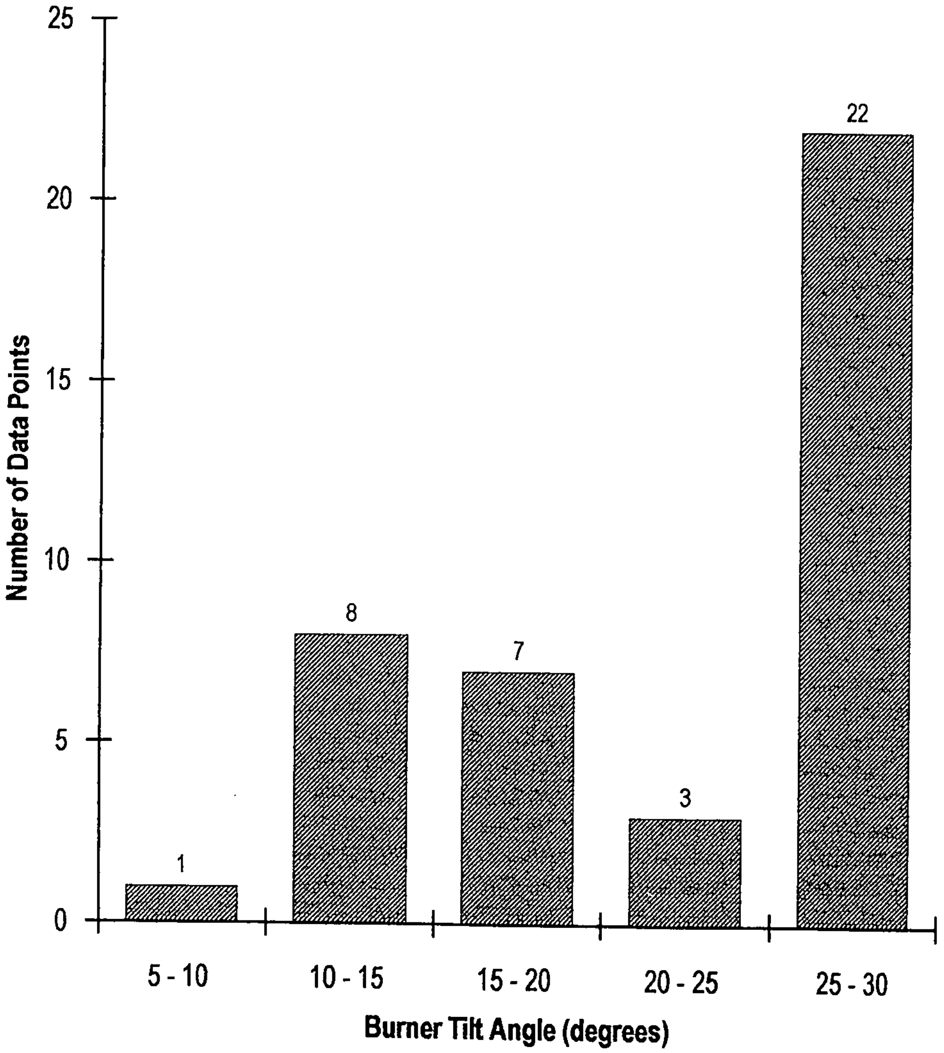

Figure 5.28

45 MW Available Neural Network Data Distribution
for Burner Tilt Angle

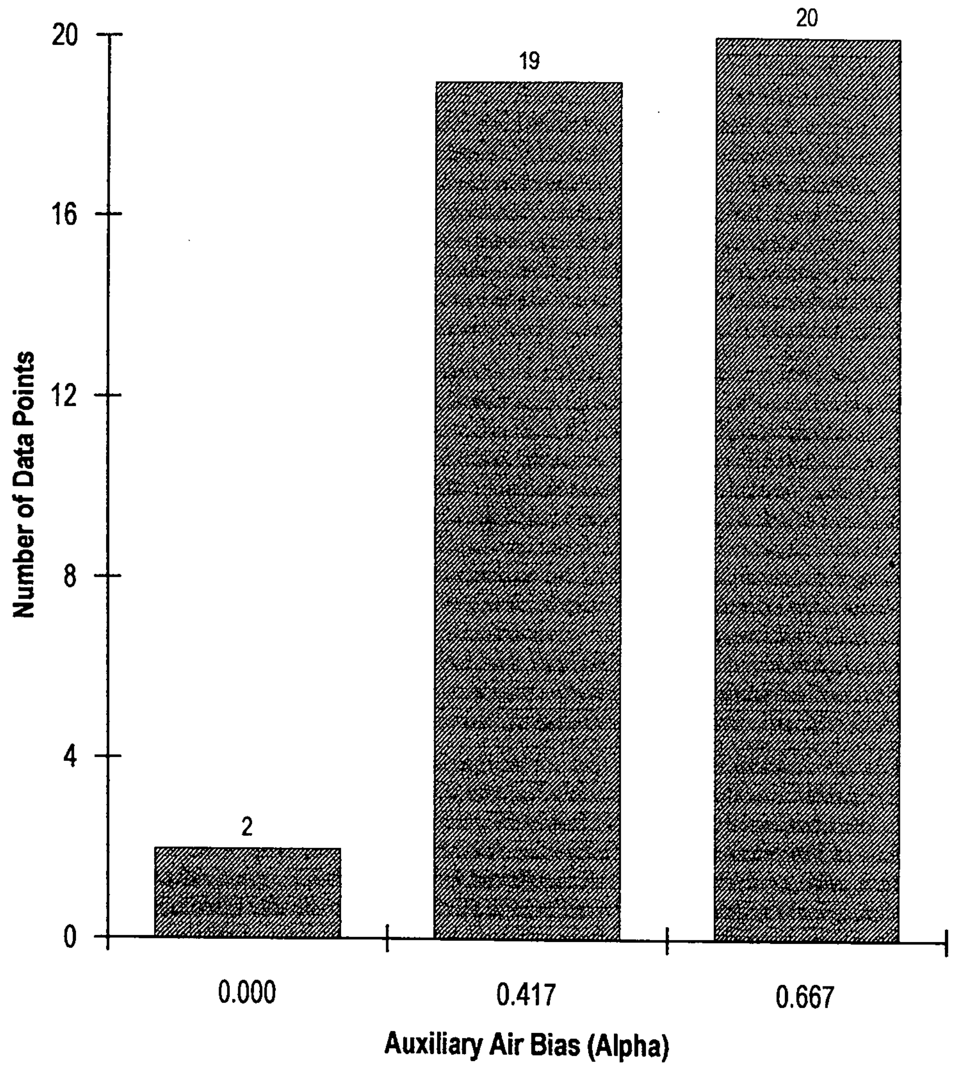

Figure 5.29

45 MW Available Neural Network Data Distribution
for Auxiliary Air Bias

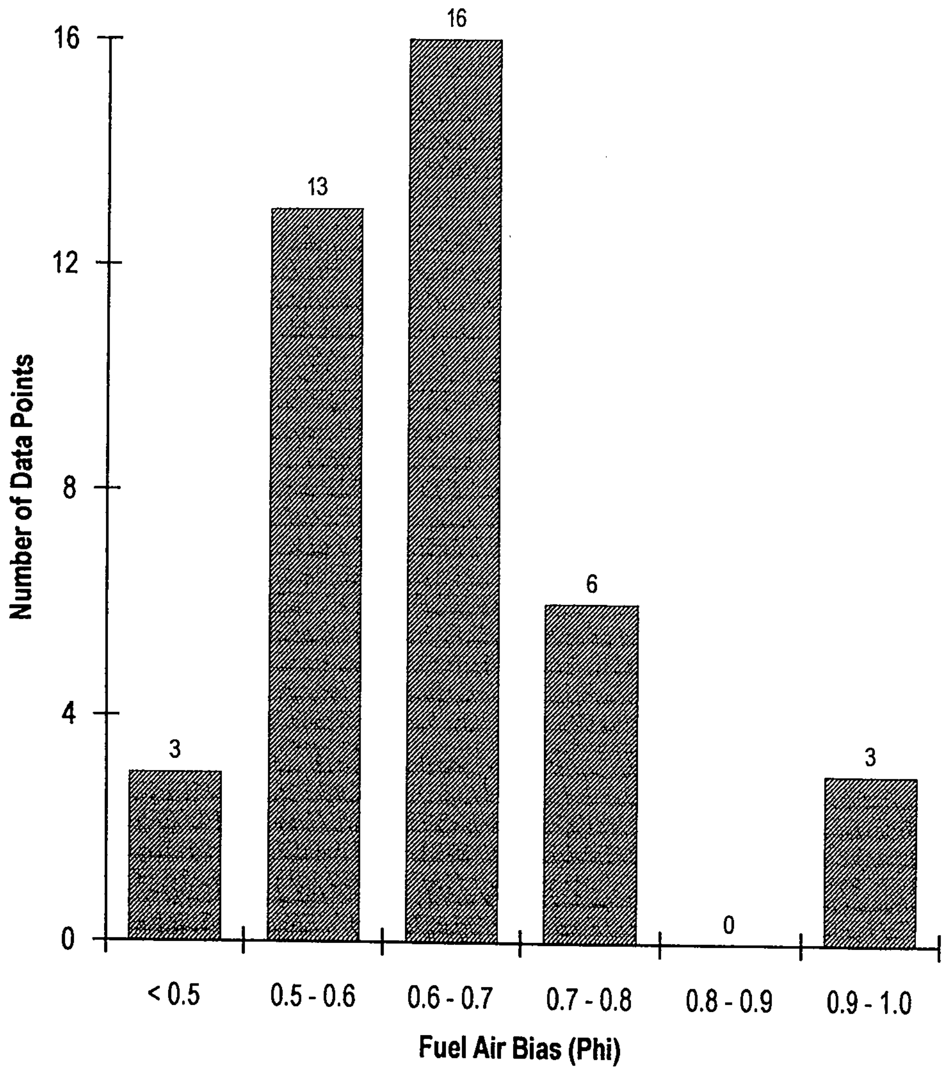

Figure 5.30

45 MW Available Neural Network Data Distribution
for Fuel Air Bias

Network Settings and Performance Results.

Table 5.11

45 MW NOx Neural Network Optimal Settings

Number of Hidden PEs:	3
Transfer Function: Hidden Layer / Output Layer	Tanh / Tanh
Epoch Size:	20
Learn Count:	35,000
Learn Rate: Hidden Layer / Output Layer	0.3 / 0.1
Momentum: Hidden Layer / Output Layer	0.4 / 0.4
Error Function:	Standard
Learning Rule:	Norm-Cum-Delta
Summation Function:	Normal Sum

Table 5.12

45 MW NOx Neural Network Performance Results

Performance Measure	Input Percent Contribution
R-Square = 0.9146	Economizer O2 = 13.7%
Adj. R-Square = 0.9051	Burner Tilt Angle = 21.1%
Avg % Error = 4.21 %	Auxiliary Air Bias = 28.7%
Avg. Difference = 0.016 (lb/MBtu)	Fuel - Air Bias = 36.5%

NOx Neural Network Trends vs. Observed Physical Trends.

Figure 5.31 and 5.32 depict the predicted NOx versus the economizer oxygen and burner tilt angle, respectively, for all available data vectors. NOx is linearly related to economizer oxygen, the same behavior as seen at full load. However, NOx is now linearly related to burner tilt angle at 45 MW whereas at full load it was parabolic. There is a high degree of scatter within Figure 5.32 and so this linear relationship may not necessarily hold true throughout all combinations of operating conditions.

The linear relationship of NOx and economizer oxygen is reinforced in Figure 5.33 when plotted at constant tilt, auxiliary air bias, and fuel air bias. The two different burner tilt angles have different trend slopes but each similar tilt angle has the same slope regardless of auxiliary air bias. Also, the NOx level increases with an increase in burner tilt angle, as shown by the upward vertical shift in the data plots. This coincides with observed data and therefore this neural network relationship appears valid.

Similar to Figure 5.33, Figure 5.34 shows that under constant parameter conditions, the linear relationship of NOx to burner tilt angle remains. In this instance, an increase in economizer oxygen causes a vertical shift upwards, denoting an increase in NOx. Meanwhile, a higher auxiliary air bias causes a decrease in NOx at a constant economizer oxygen level.

With only two different auxiliary air bias parameters being used during parametric testing at Potomac River, NOx could not be plotted against this parameter without a large uncertainty being present in the results. Figure 5.35 shows the affect that fuel air bias has upon predicted NOx emissions. Similar to the previous burner tilt angle figure, the trend plots with identical auxiliary air bias values have the same slope and an increase in burner tilt results in an increase in NOx. This behavior was seen in the work of D'Agostini et al. [4], and is consistent with observed boiler behavior.

Heat Rate Neural Network

Network Settings and Performance Results.

Table 5.13

45 MW Load Unit Heat Rate Neural Network Optimal Settings

Number of Hidden PEs:	3
Transfer Function: Hidden Layer / Output Layer	Tanh / Linear
Epoch Size:	4
Learn Count:	55,000
Learn Rate: Hidden Layer / Output Layer	0.3 / 0.1
Momentum: Hidden Layer / Output Layer	0.4 / 0.4
Error Function:	Standard
Learning Rule:	Norm-Cum-Delta
Summation Function:	Normal Sum

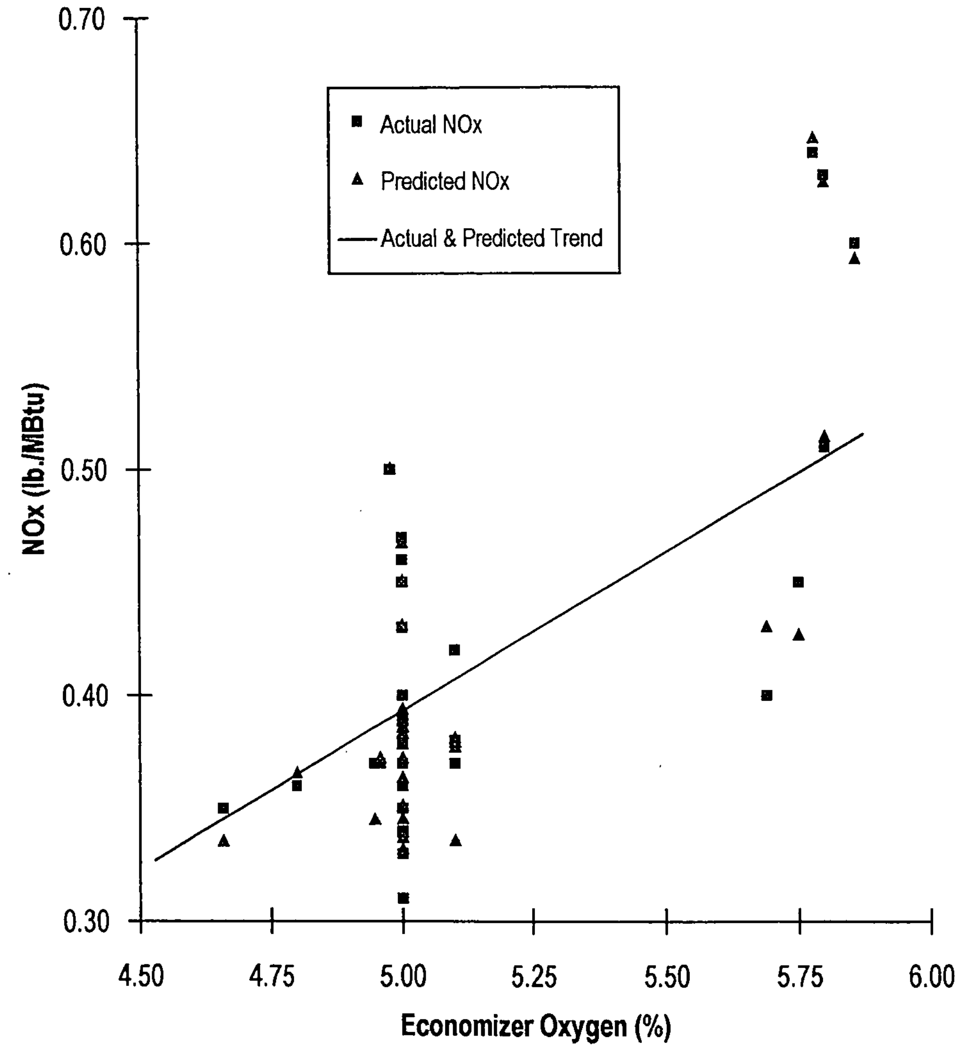

Figure 5.31

45 MW Actual and Predicted NOx
vs. Economizer Oxygen

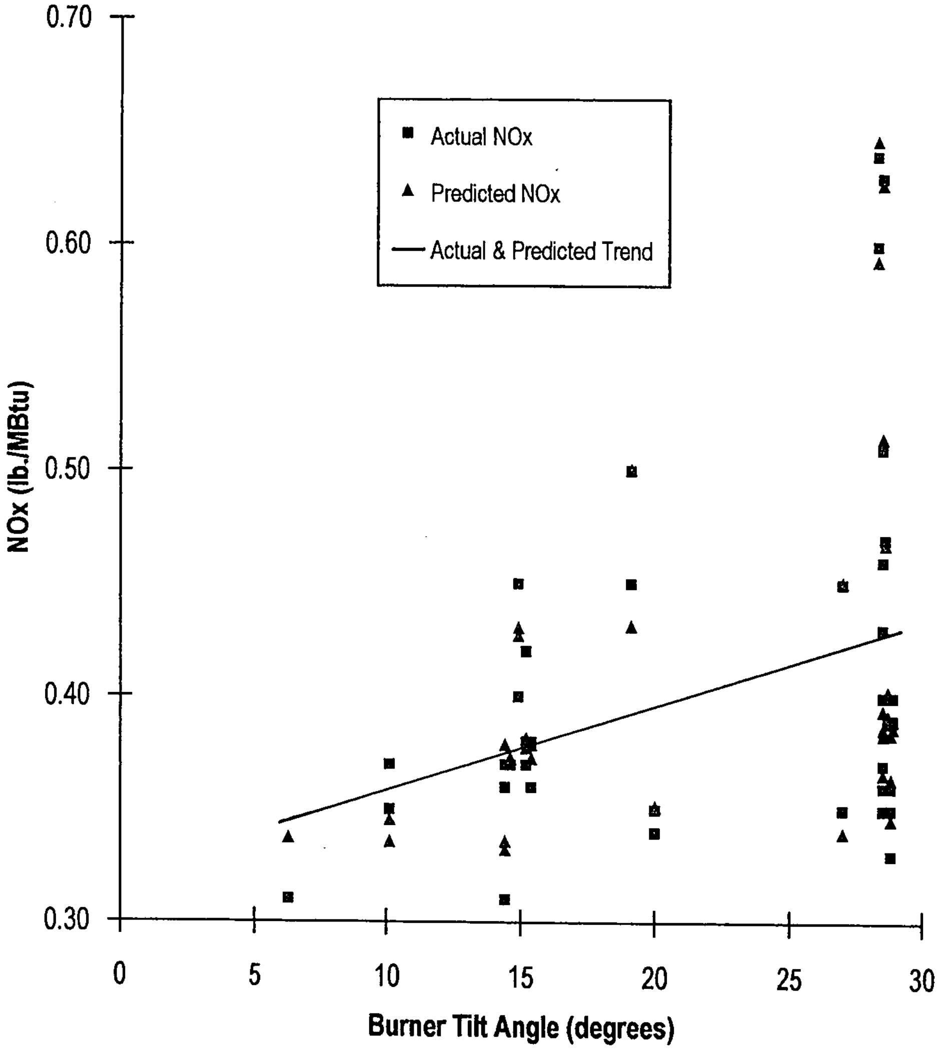

Figure 5.32

45 MW Actual and Predicted NOx
vs. Burner Tilt Angle

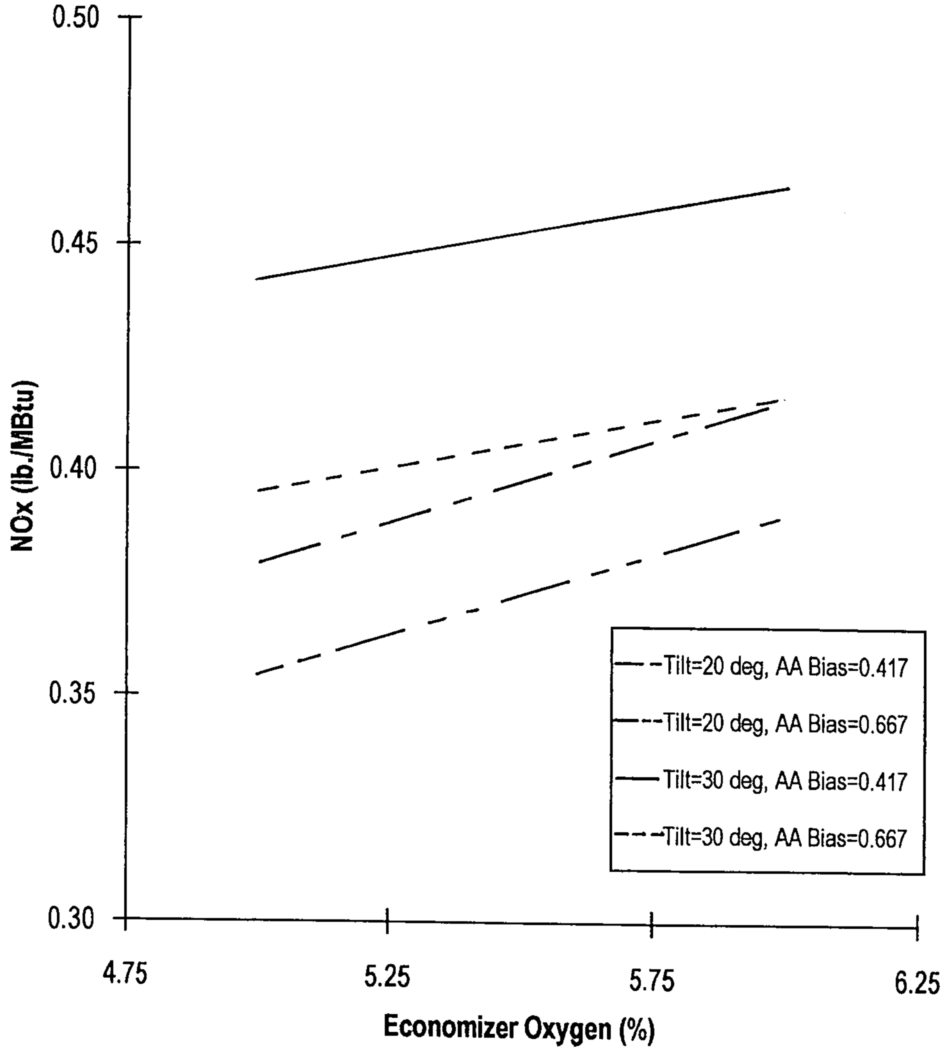

Figure 5.33

45 MW Predicted NOx vs. Economizer Oxygen at Constant Burner Tilt Angle, Auxiliary Air Bias, and Fuel Air Bias=0.60

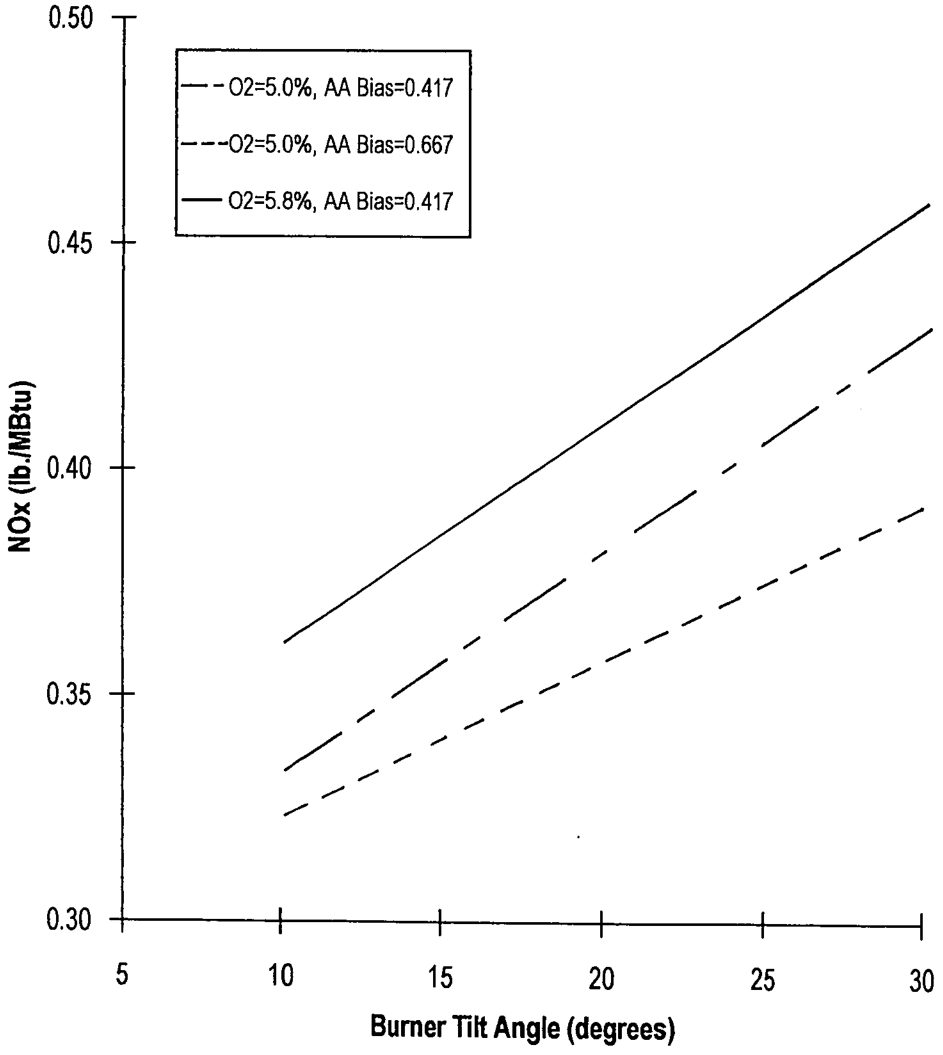

Figure 5.34

45 MW Predicted NOx vs. Burner Tilt Angle at Constant Economizer Oxygen, Auxiliary Air Bias, and Fuel Air Bias=0.60

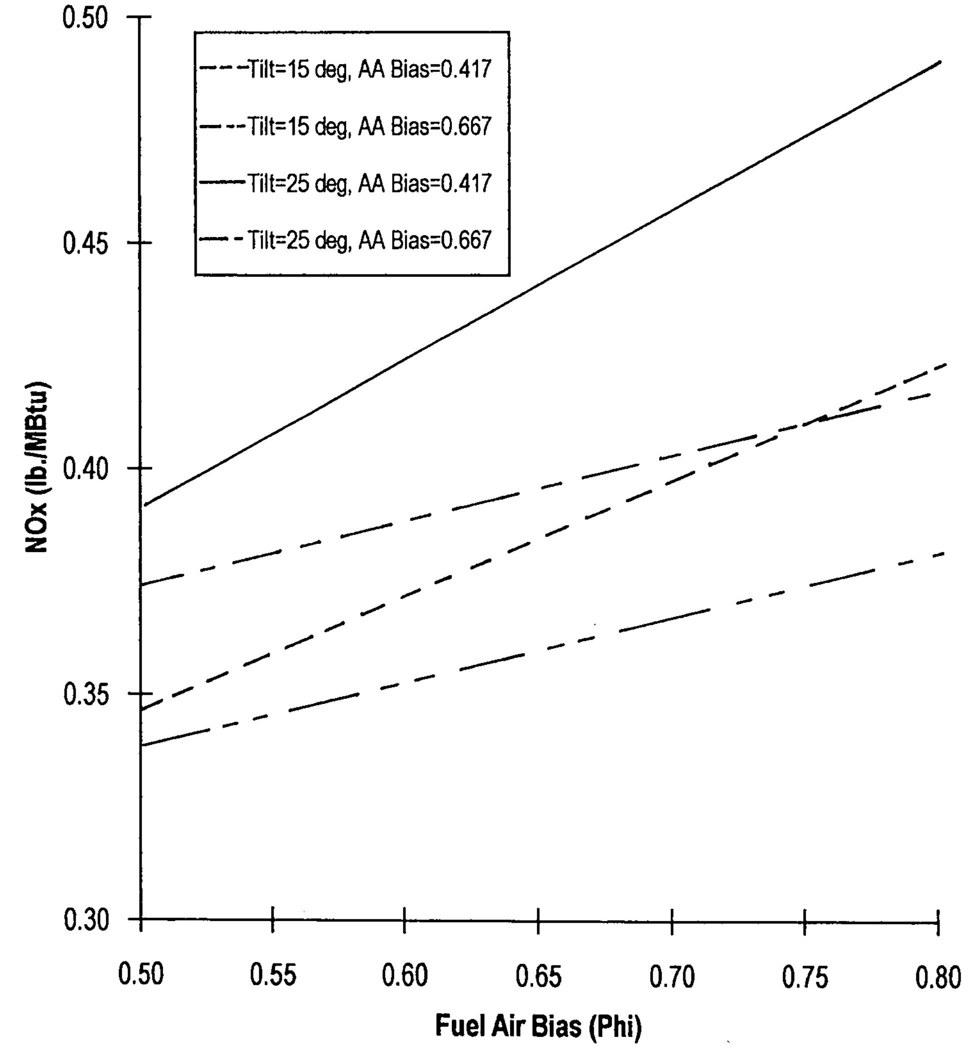

Figure 5.35

45 MW Predicted NOx vs. Fuel Air Bias at Constant
Burner Tilt Angle, Auxiliary Air Bias, and Economizer Oxygen=5.4%

Table 5.14

45 MW Load Unit Heat Rate Neural Network Performance Results

Performance Measure	Input Percent Contribution
R-Square = 0.7066	Economizer O2 = 30.0%
Adj. R-Square = 0.6740	Burner Tilt Angle = 11.3%
Avg % Error = 0.198%	Auxiliary Air Bias = 16.9%
Avg. Difference =19.4 (MBtu/kWh)	Fuel Air Bias = 41.8%

<u>Unit Heat Rate Neural Network Trends vs. Observed Physical Trends</u>.

With the previous discussion of the poor distribution within the 45 MW available data and the uncertainty in the heat rate values used, this network had the worst performance of all used in this study. There is a lack of available knowledge about the cause and effects of boiler operating conditions on the resulting heat rate at this load level. Furthermore, some plotted trends resulted in obvious disparities between network model operation and observed boiler characteristics.

Figures 5.36 and 5.37 show the large degree of scatter within the economizer oxygen and burner tilt angle, respectively, and the subsequent predicted unit heat rate. This scatter and poor heat rate values were carried through to the plots using constant input parameter values. However, Figure 5.38 shows that heat rate improves when the burner tilt angle is raised and it has already been proven in this thesis that such a relationship is true. Figure 5.39 also shows heat rate improving with increasing burner tilt angle and decreasing amount of overfire air. The change in overfire air is shown to be more significant than a change in economizer oxygen which causes only a vertical shift in the data trend. Thus, even without quality data to train with, the heat rate neural network still learned some of the basic boiler operation at 45 MW.

Overall, the heat rate neural network had somewhat acceptable performance measure results for predicting the unit heat rate (Table 5.14). Ultimately, however, the network did not completely learn the underlying interactions between independent parameters and heat rate. This is due to the deficiencies within the data that were discussed earlier. When the 45 MW unit heat rate and NOx network equations were used in the numerical optimization process, this "lack of understanding" within the equations affected the validity of the optimization output.

Finally, Figure 5.40 plots predicted unit heat rate versus predicted NOx for the 45 MW load level. The scatter within the data does not allow for any definite trend to be identified which is the same result D'Agostini et al. [4] achieved. However, D'Agostini did conclude that two modes of boiler operation will achieve acceptable heat rate values at low NOx conditions. The first is specified as a low tilt, low air bias configuration while the second calls for a high tilt, high air bias configuration. The NOx neural network is in agreement with these configurations (Figure 5.34) but the heat rate network is not (Figure 5.39). The heat rate neural network leads one to believe that a more moderate degree of auxiliary air distribution would result in better thermal performance while maintaining NOx levels. An additional problem with the heat rate network may be the boiler's identified ability to achieve minimum heat rate values at these two very different operating modes. The data that represent these conditions, along with the poor data distribution, may have further confused the network during training. A future possibility is to split the lower load operations into different boiler operating modes so that only one minimum exists for each mode.

With the reasons already discussed and the continued lack of a boiler cleanliness parameter, the results obtained at 45 MW are not a reason for disregarding the neural network approach. If anything, the 45 MW neural networks only reinforced what this author had already stated - the effectiveness of neural networks is directly correlated to the quality and quantity of the available data to train it.

45 MW Reduced and Clustered Data Experiment Results

The same procedure used at full load for analyzing the effects of reduced and clustered data was used at 45 MW. However, due to the uncertainty in the heat rate data and the poor data distribution, only the NOx neural network was used for the reduced data network experiments. Clustering was not possible and, judging by the prior discussion, would not yield reliable results.

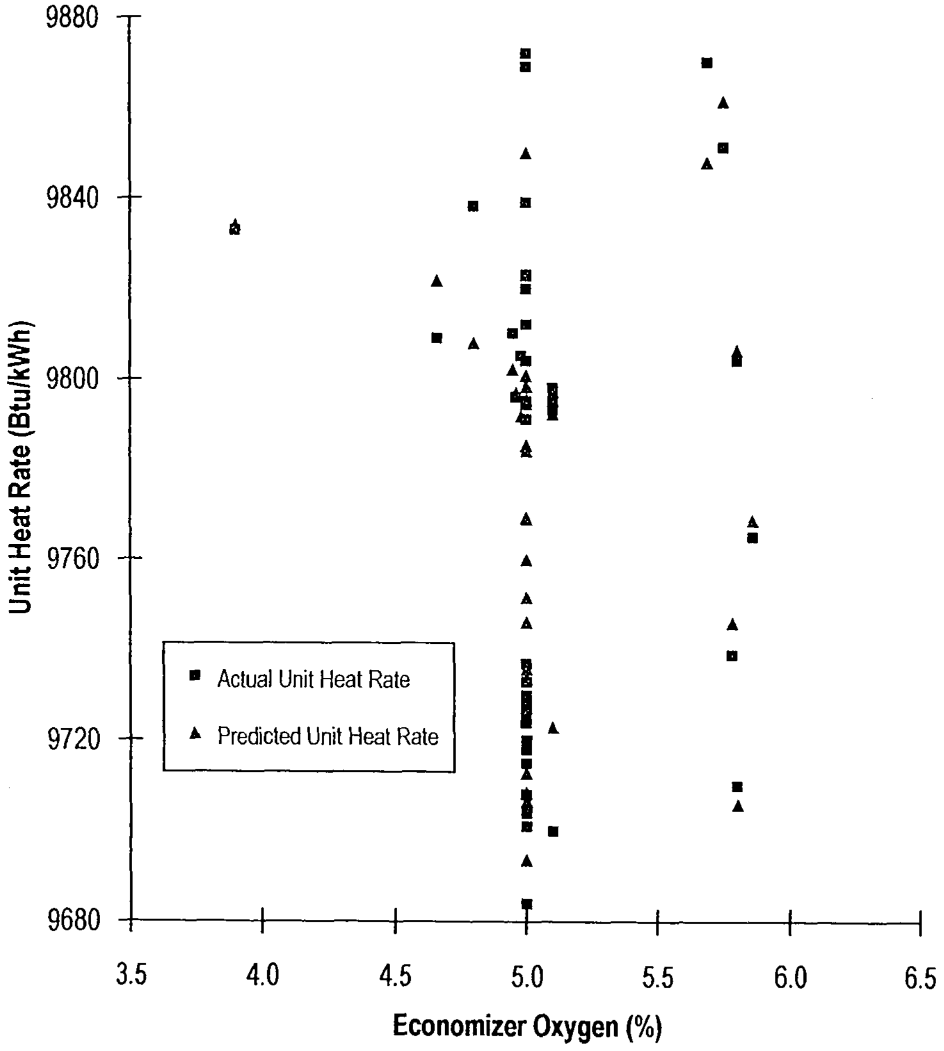

Figure 5.36

45 MW Actual and Predicted Unit Heat Rate vs. Economizer Oxygen

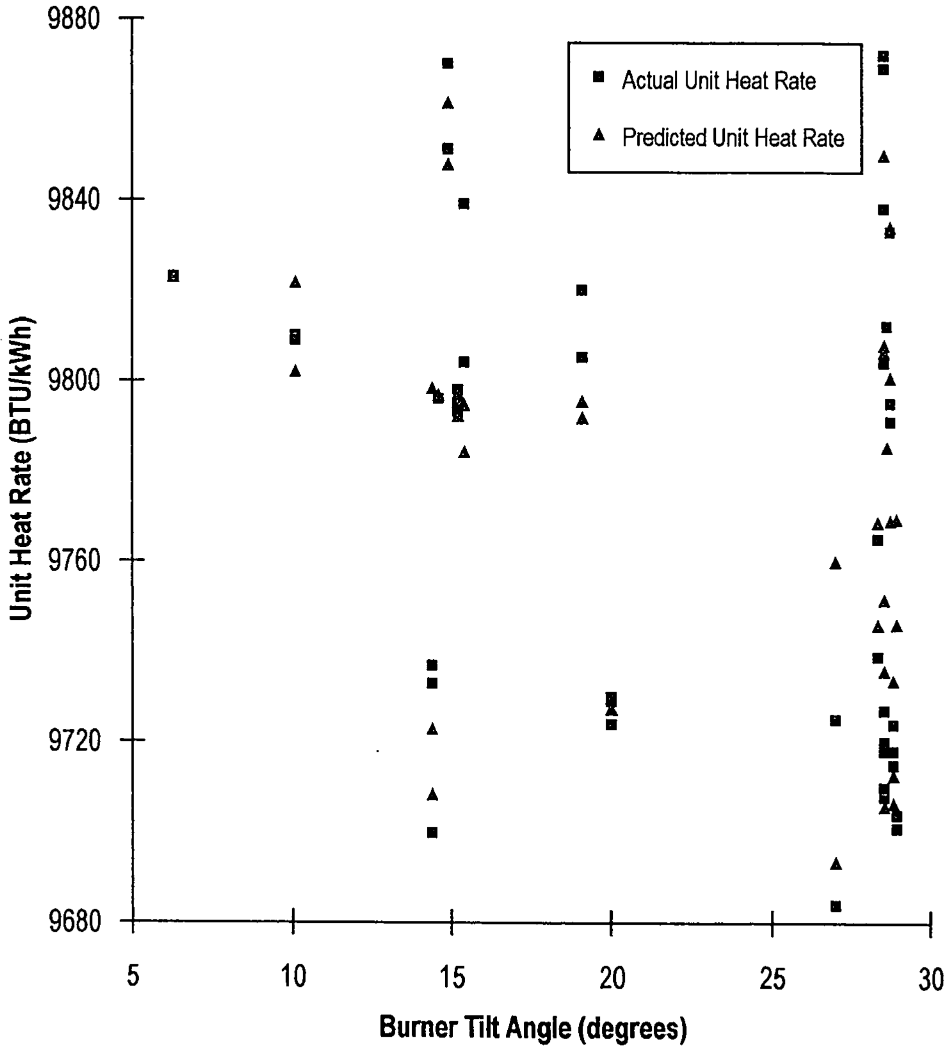

Figure 5.37

45 MW Actual and Predicted Unit Heat Rate vs. Burner Tilt Angle

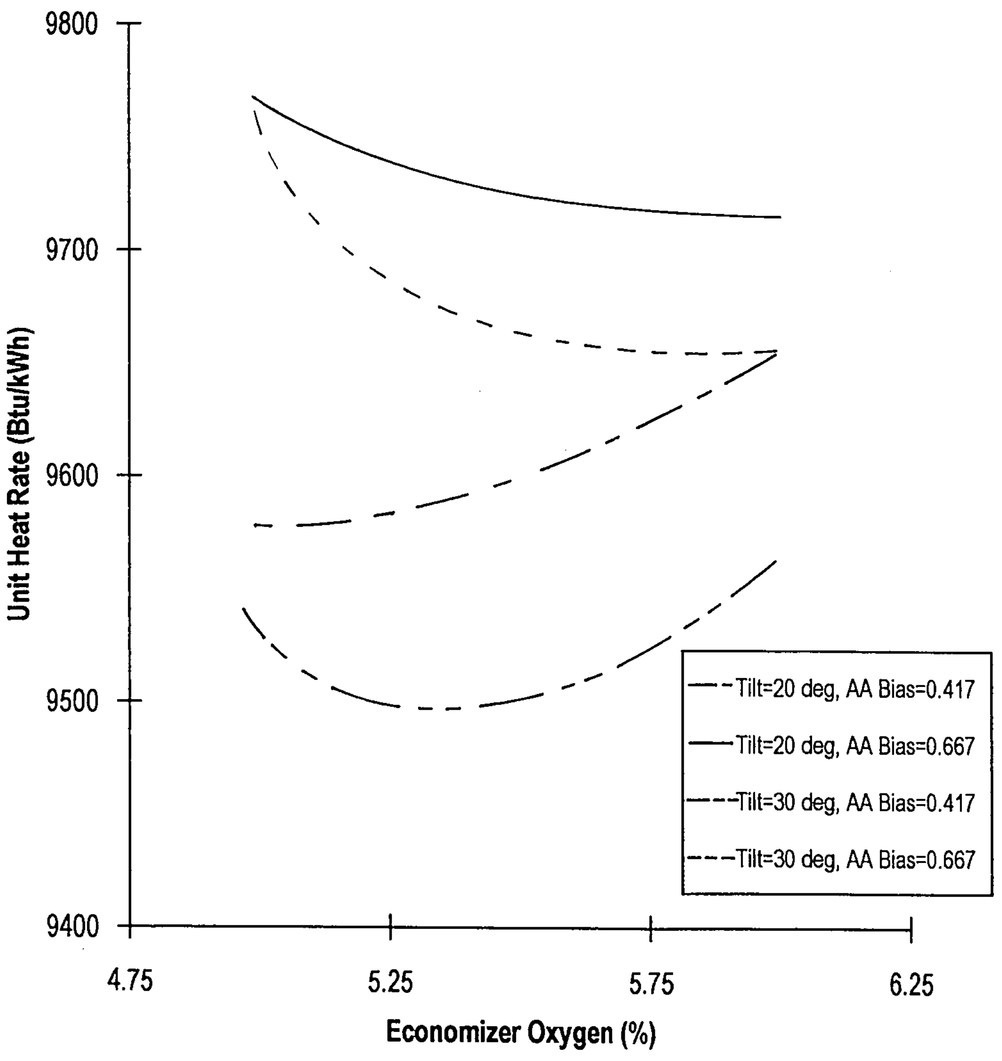

Figure 5.38

45 MW Predicted Unit Heat Rate vs. Economizer Oxygen at Constant Burner Tilt Angle, Auxiliary Air Bias, and Fuel Air Bias=0.60

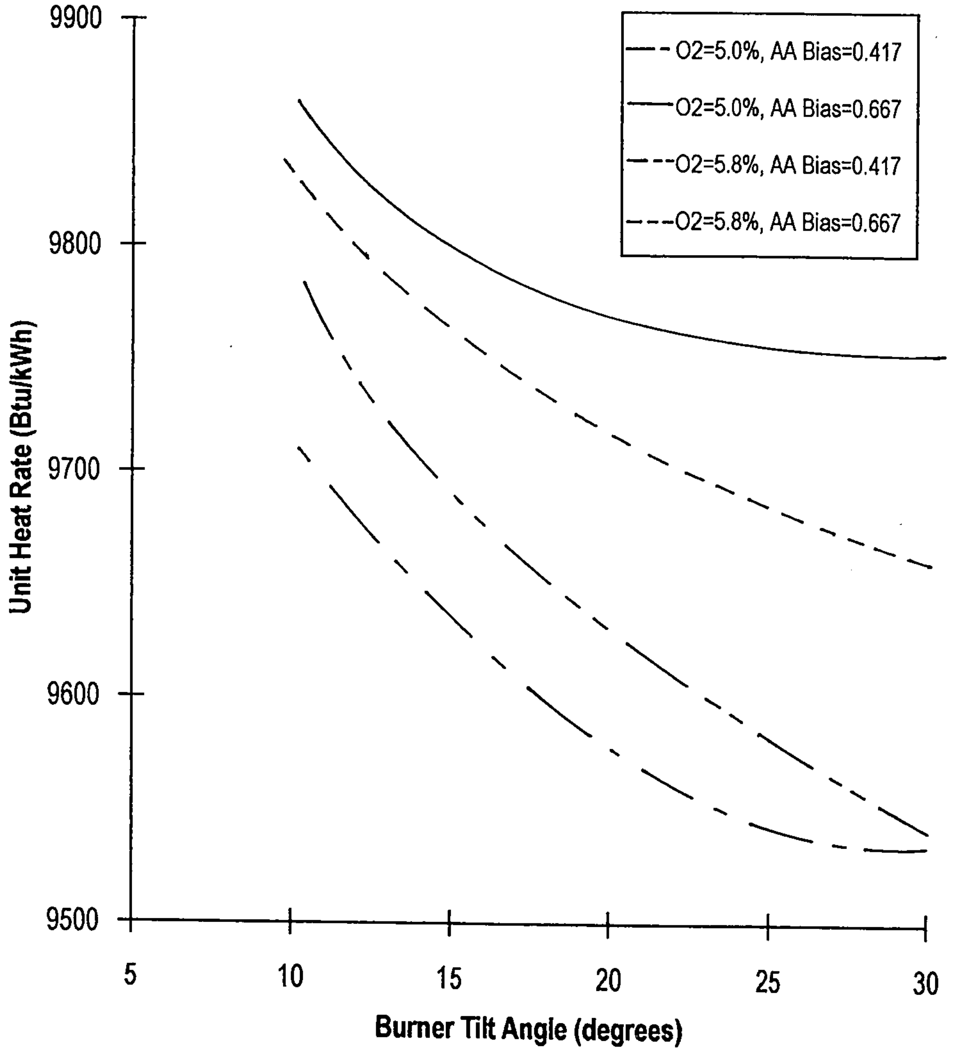

Figure 5.39

45 MW Predicted Unit Heat Rate vs. Burner Tilt Angle at Constant Economizer Oxygen, Auxiliary Air Bias, and Fuel Air Bias=0.60

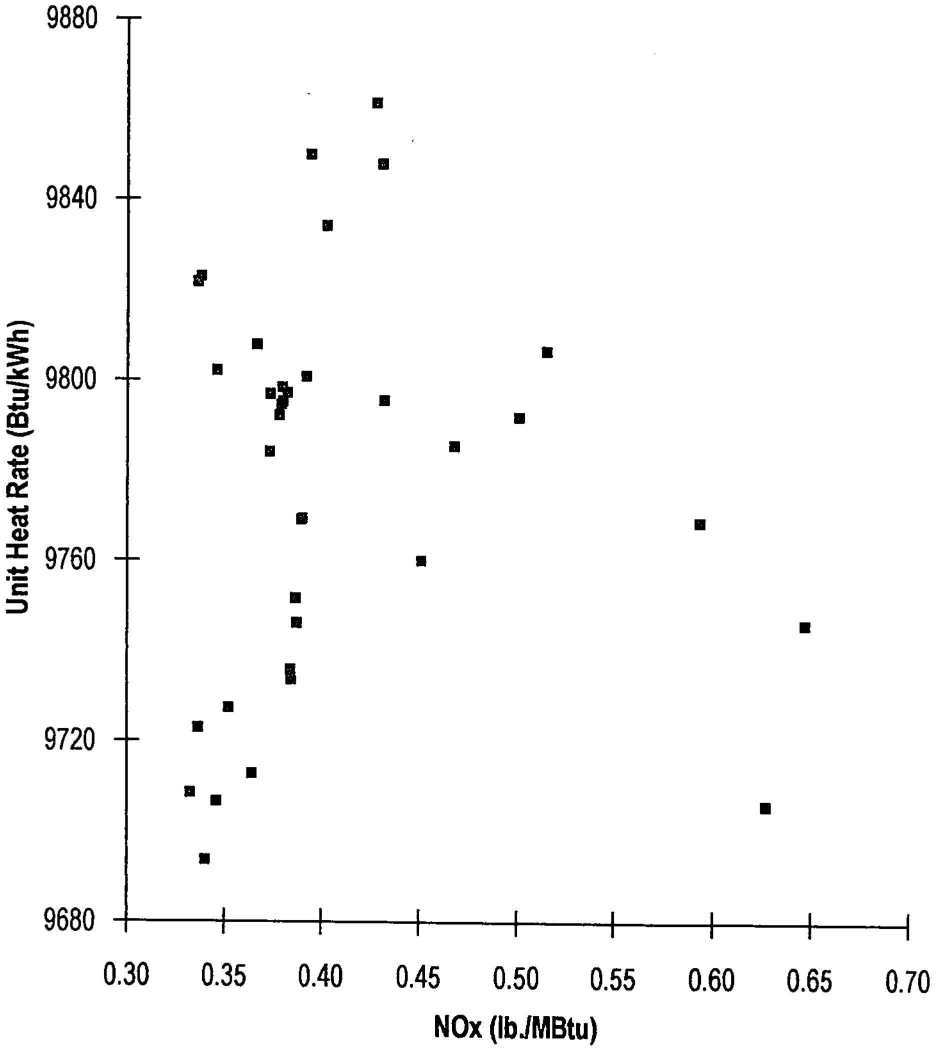

Figure 5.40

45 MW Predicted Unit Heat Rate vs. Predicted NOx

The 45 MW reduced data networks for NOx had similar outcomes as the full load cases. The graphs of R-Squared, average difference, and average percent error are presented in Figures 5.41, 5.42, and 5.43, respectively.

It can be inferred from these graphs that the performance "breakpoint" occurs at $\approx$75% of available data. This translates into $\approx$ 31 training data vectors are required to maintain sufficient levels of accuracy and generalizing ability. This means that between seven and eight data vectors are required for each input used in a neural network at this load, which is similar to what was found for the full load case. This number is based upon poorly distributed data and could most likely be reduced by improving the available data quality.

RESULTS OF NEURAL NETWORK OPTIMIZATION

With the optimal neural networks for each load established, the final objective was to use the neural networks to determine the optimal boiler operating conditions. The Nelder and Meade Simplex Algorithm was described in Chapter 4; and this optimization methodology was applied to the neural network equations to find the conditions for optimal boiler operation. The final optimization procedure used for the combined unit heat rate and NOx network equations was the result of developing smaller optimization programs and combining their operation. Once this combined network equation optimization program was working, tests were conducted to determine the best way to initialize and execute the algorithm. After these initial values were decided upon, their implementation into the overall NOx control software package was researched.

After the full load case was completed, the 45 MW load was examined for the purpose of feasibility rather than actual implementation. As discussed previously, the 45 MW data quality and neural network behavior made these networks difficult to accurately analyze and compare to field results. A working solution was discovered but testing of the 45 MW optimization algorithm to find the best operating conditions was not conducted. However, the ability to calculate a physically

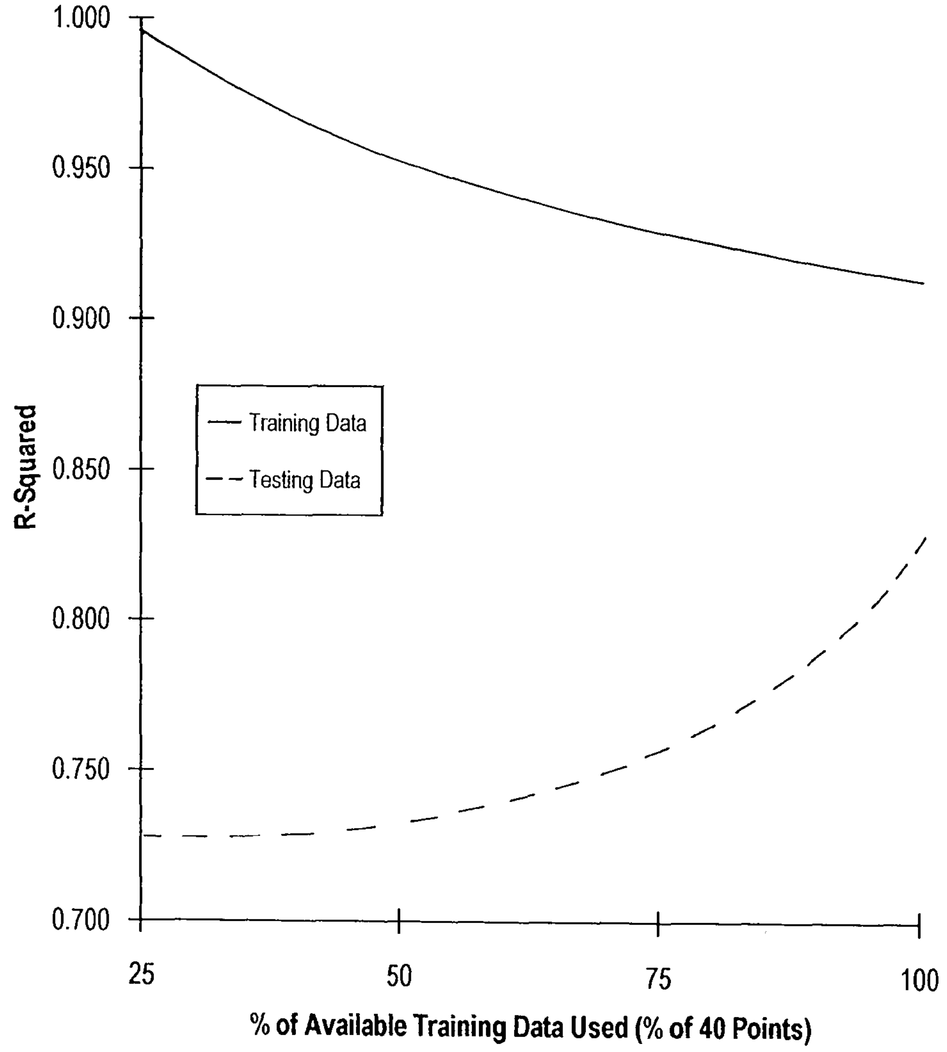

Figure 5.41

45 MW NOx Neural Network R-Square vs.
Percent of Available Training Data Used

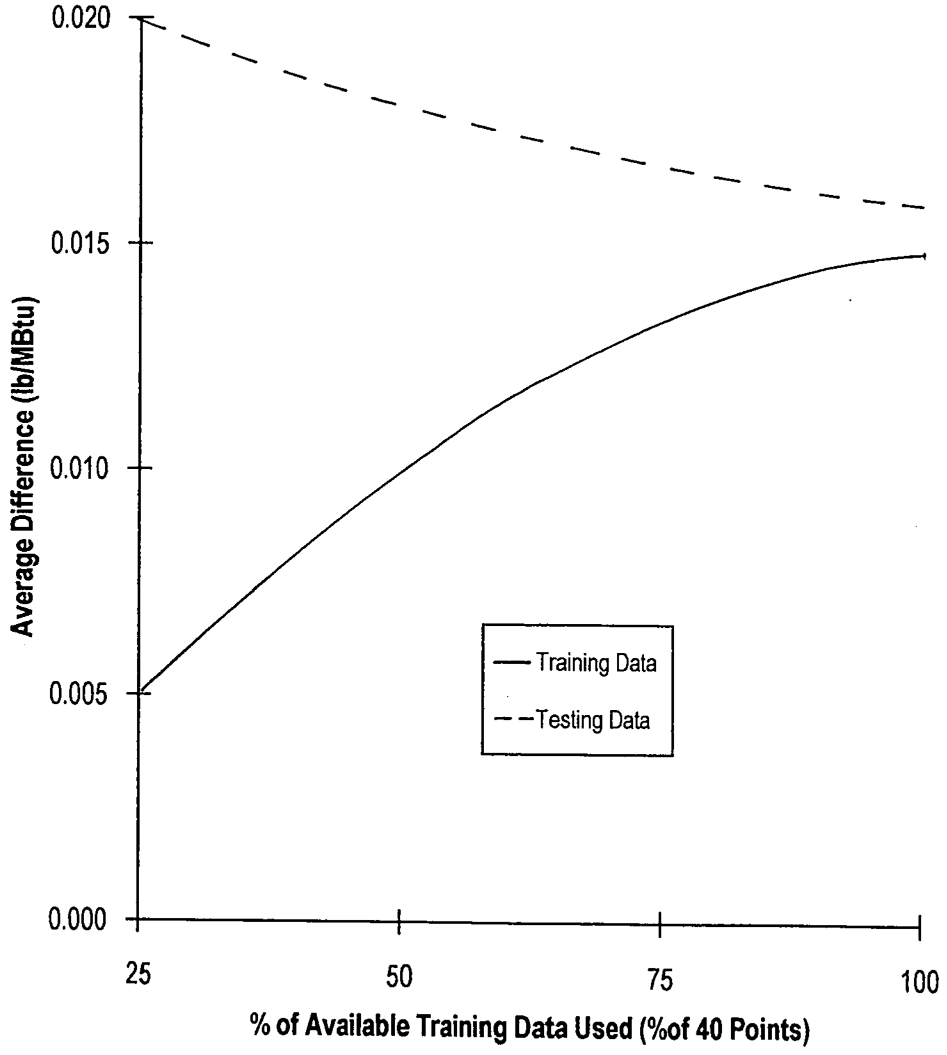

Figure 5.42

45 MW NOx Neural Network Average Difference vs.
Percent of Available Training Data Used

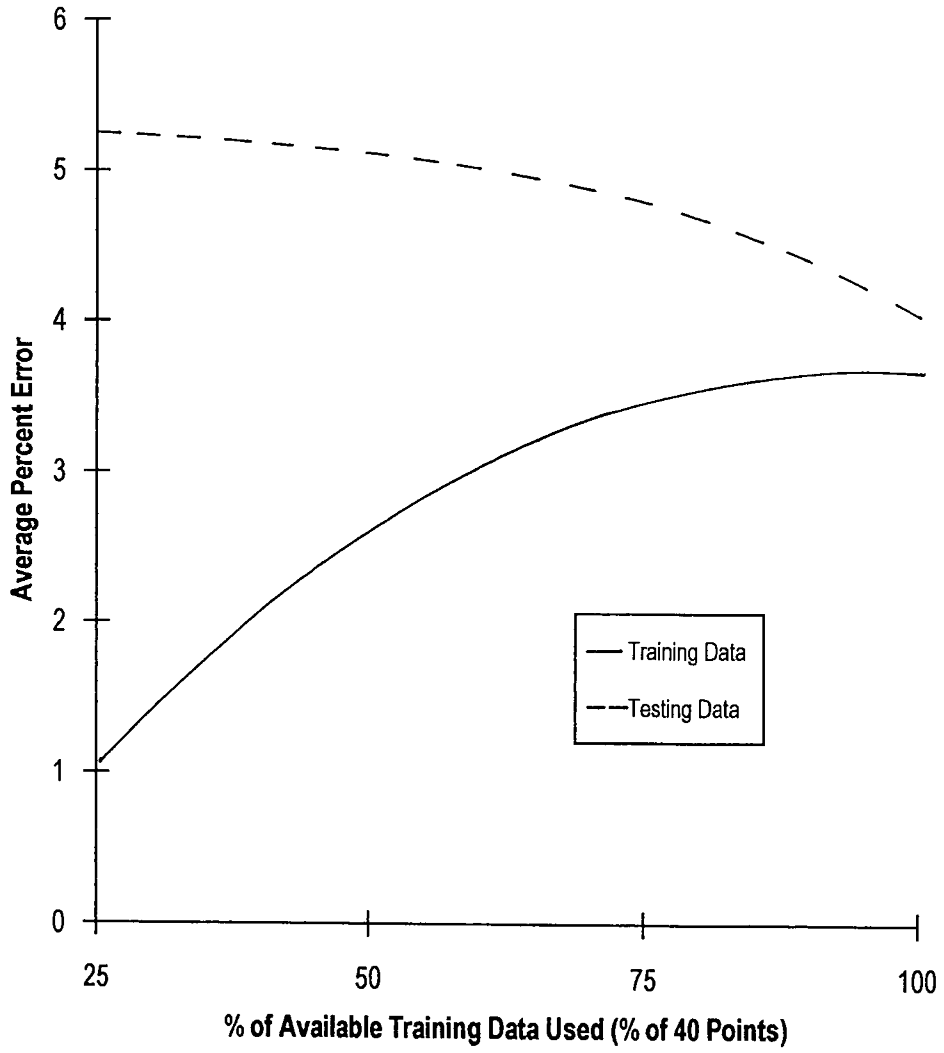

Figure 5.43

45 MW NOx Neural Network Average Percent Error vs.
Percent of Available Training Data Used

feasible solution leads the author to believe that once better data and networks are generated, optimization at the 45 MW load level will follow in the success of the full load case.

The final topic in this chapter is the post-processing of the optimization algorithm results. The Simplex Algorithm returns the optimal operating conditions as a set of final simplex vertex positions, each containing boiler bias parameters. These biases must be transformed back to their equivalent physical settings so that the user can adjust the boiler operating mode. Several approaches have been attempted to best accomplish this task, keeping in mind that no user interaction is desired in the final version of this software package. Although research continues as this thesis is written, the current version of the post-processing algorithm will be reviewed.

Full Load Neural Network Equation Optimization

The first step was to get the optimization program running properly for the full load NOx neural network equations. Once successful, the full load heat rate equations were optimized with the full load NOx network equations being implemented as a penalty function which was detailed in Chapter 4. This makes the Simplex algorithm act as a constrained optimization routine and the unit heat rate is optimized while maintaining the desired NOx emissions limit.

Full Load NOx Network Equation Optimization.

The full load case was the cornerstone for developing future neural network equation optimization programs. Until this point, the author had used the IMSL [31] optimization routines to check the feasibility of optimizing neural network equations. The availability of a reliable, stand-alone optimization program that required no user interaction was paramount to the success of the NOx control software package.

The full load NOx neural network equations were programmed into the function subroutine of the Nelder and Meade Simplex algorithm [27]. As mentioned in Chapter 4, for four independent parameters, the simplex will have five vertices. When the optimization is complete, the algorithm output consists of the final location of these five vertices, which are in terms of the scaled neural network values. These outputs were then scaled from the neural network values into the real world values for economizer oxygen, burner tilt angle, and the auxiliary air and fuel air biases. For comparing these results to the work of D'Agostini, these 5 closely grouped output values for each parameter are averaged for a final answer.

The following table depicts the results from optimizing the full load NOx neural network equations only, compared to the results of D'Agostini et al. [4]:

Table 5.15

Comparison of Full Load NOx Neural Network Equation Optimization Results to Parametric Testing Analysis Results

	Simplex Optimization Results	Potomac River Test Results
Economizer O2 (%)	1.53	1.60
Burner Tilt (degrees)	5.6	7.0
Auxiliary Air Bias	0.50	0.50
Fuel Air Bias	1.41	1.40
NOx (lb/MBtu)	0.37	0.38

The simplex optimization results were checked using spreadsheet calculations and two IMSL optimization routines, BCPOL and BCONF [31]. These revealed very similar results, with the maximum relative error among all routines being under 2%, for all parameters. Therefore, it was believed that the simplex optimization was effective for this application and the heat rate network equations were introduced.

Full Load Unit Heat Rate Network Equation Optimization. Several alterations to the optimization algorithm had to be made to accommodate the heat rate and NOx neural network equations. After the simplex was initialized with the starting vertices, each subsequent iteration required a calculation of the heat rate and NOx at the new vertex locations. After NOx was calculated at each vertex, it was compared to the user-defined allowable NOx limit. If the current vertex NOx exceeded the NOx limit, a user-defined value was added to that vertex's heat rate value as a "penalty" for exceeding the NOx limit. By adding this penalty to the current heat rate value, it made that specific vertex the "worst" out of all the simplex vertices because it had the highest heat rate value. This forced the algorithm to choose this particular vertex as the worst point and therefore this vertex was the first one to undergo a change in its location during the next iteration. This change is such that the new heat rate value was less than the previous value and the vertex's NOx value was closer to the compliance level. This process continued until the minimum heat rate for each vertex was found and each vertex had a NOx value equal to or lower than the NOx limit.

This approach proved successful from its inception. However, the algorithm now had four function constants to contend with during the optimization process. These function constants are described below along with the tests that were conducted to determine the interaction between these and other algorithm parameters.

<u>Initial Simplex Orientation</u>. As previously described in Chapter 4, the simplex for this 4 independent parameter optimization has five vertices. The dimensions of the simplex are not critical to the optimization process, but the initial orientation of the simplex within the solution space is critical to the success of the algorithm. This was examined by incrementally changing the start point of a single vertex for a single parameter. The amount of change was incremented as was the number of parameter changes and vertex start points. One of three results always occurred. The algorithm either successfully converged to a feasible answer, extrapolated to an unfeasible answer, or never converged. The author and ERC engineers devised a method to ensure that the lowest sensible heat rate conditions (that meet NOx limitations) is found during the optimization portion of the NOx control package.

The physical trends for the full load case showed a strong dependence on economizer oxygen and burner tilt. If either of these were combined incorrectly or had a slightly offset initial orientation, the algorithm failed. However, the auxiliary air bias and fuel air bias parameters were varied quite drastically and still had a successful convergence rate of over 70%.

Function Tolerance. The function tolerance acts as a limit within the optimization algorithm. If the values of heat rate at each of the simplex vertices differ by an amount equal to or less than the function tolerance, then convergence is assumed and the optimization process stops. The function tolerance is a scaled value for unit heat rate and, since the typical error, for the full load heat rate neural network, was approximately 27 Btu/kWh, the function tolerance was set at 0.10, or 13.5 Btu/kWh. If the function tolerance is set lower, than the user is attempting to calculate an optimal heat rate with less uncertainty than the data used to generate the equations. If the function tolerance is raised, then the final location of the vertices within the solution space will be further apart. This results in too broad of a range of optimal operating conditions and the user may be misled as to what the actual best conditions for boiler operation are.

NOx Function Limit. Similar to the function tolerance, the NOx function limit is the scaled value of the allowable NOx limit for heat rate optimization. When the algorithm checks the simplex's heat rate and NOx values at its vertices, this limit dictates whether the penalty value will be assigned to the vertex heat rate value. Following the inverse trends between heat rate and NOx, as the NOx limit is allowed to rise, the optimal achievable heat rate decreases. Conversely, if the NOx limit is lowered, then the minimum attainable unit heat rate will rise.

To test this, all simplex initial conditions were kept constant except for the NOx limit. Starting with a NOx limit of 0.36 lb/MBtu, the algorithm was run and the resulting heat rate and optimal operating parameters were recorded. The NOx limit was then incremented by 0.01 lb/MBtu and the process repeated. The results of this test are shown graphically in Figures 5.44-5.47. The NOx/heat rate interaction within the neural network equations coincides with observed boiler behavior and the independent operating parameters support this interaction which is discussed later in this section.

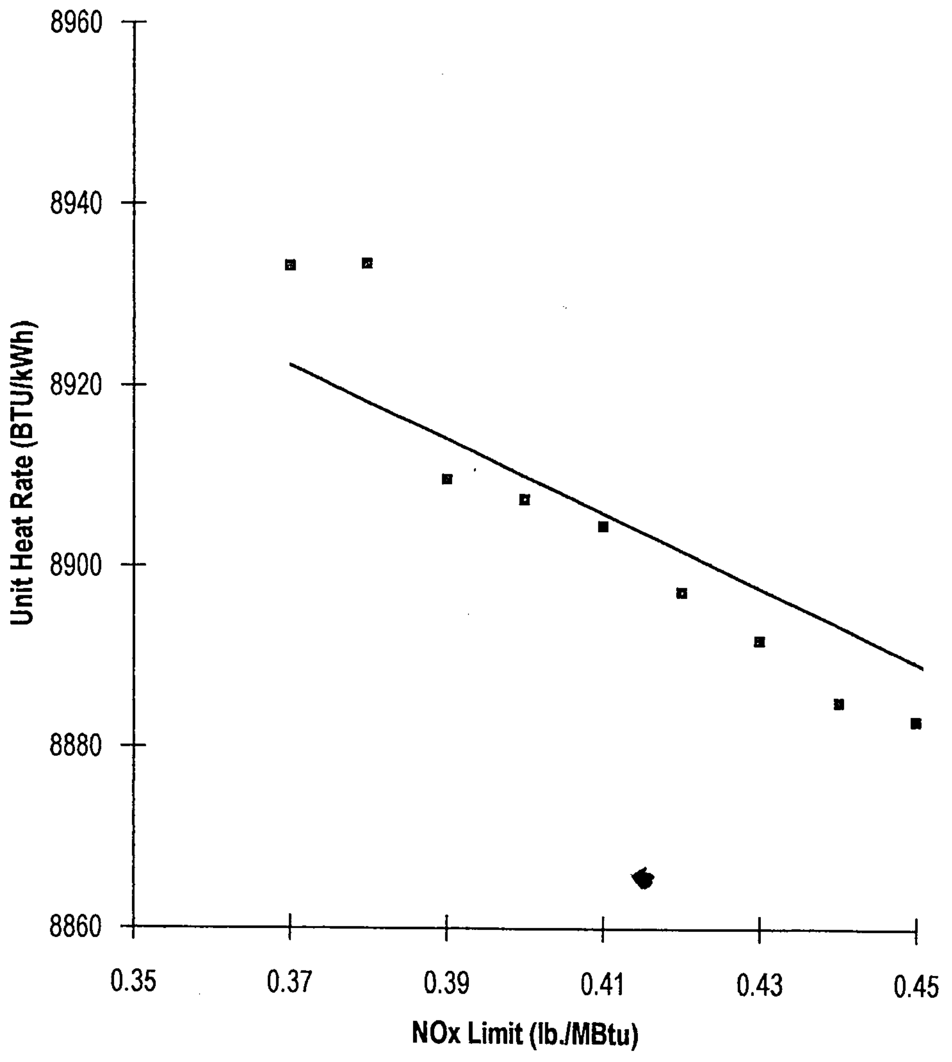

Figure 5.44

Full Load Averaged Optimal Unit Heat Rate vs.
Optimization Algorithm NOx Limit

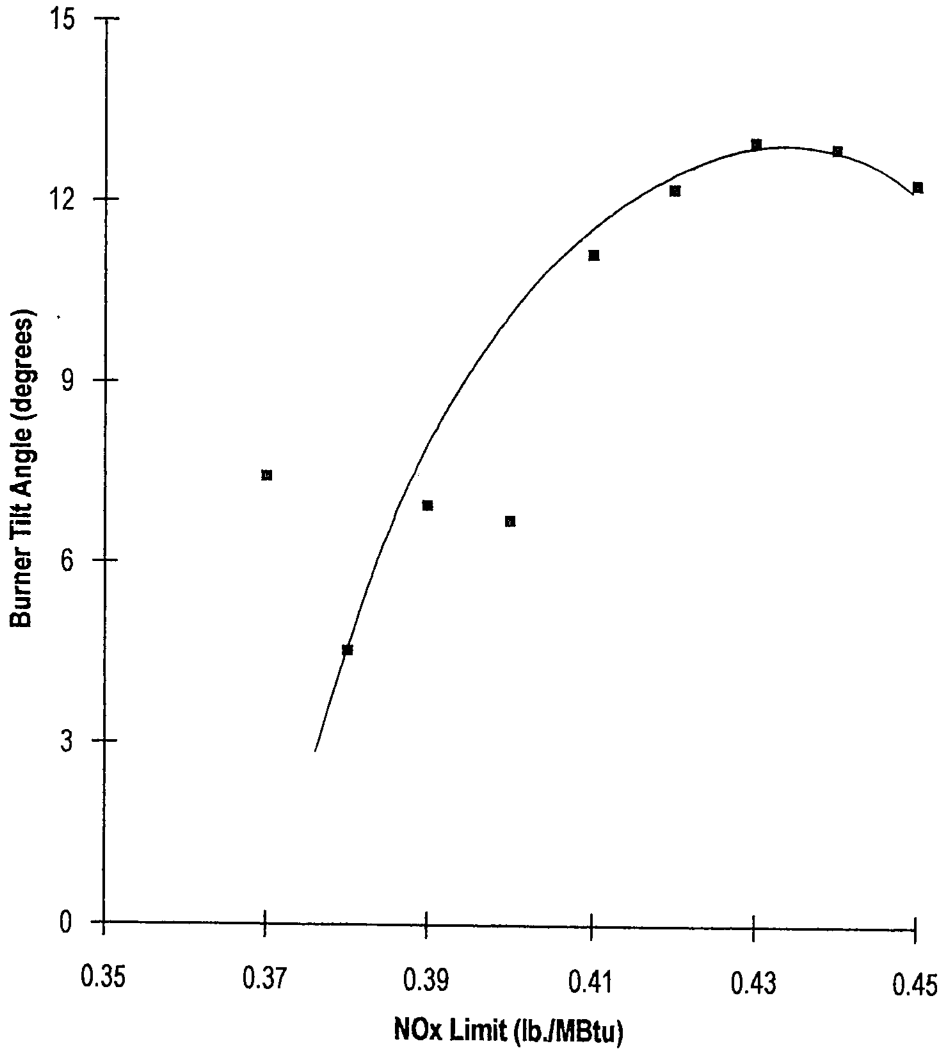

Figure 5.45

Full Load Averaged Optimal Burner Tilt Angle vs.
Optimization Algorithm NOx Limit

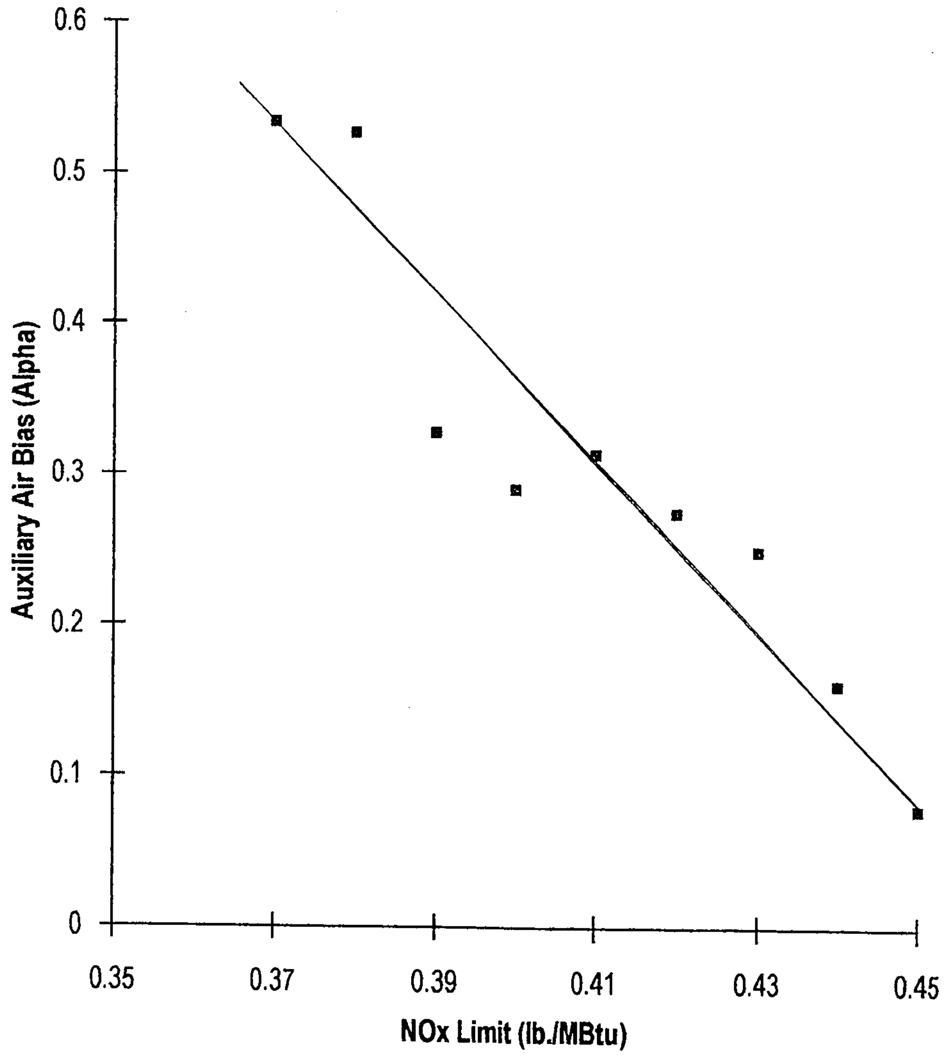

Figure 5.46

Full Load Averaged Optimal Auxiliary Air Bias vs.
Optimization Algorithm NOx Limit

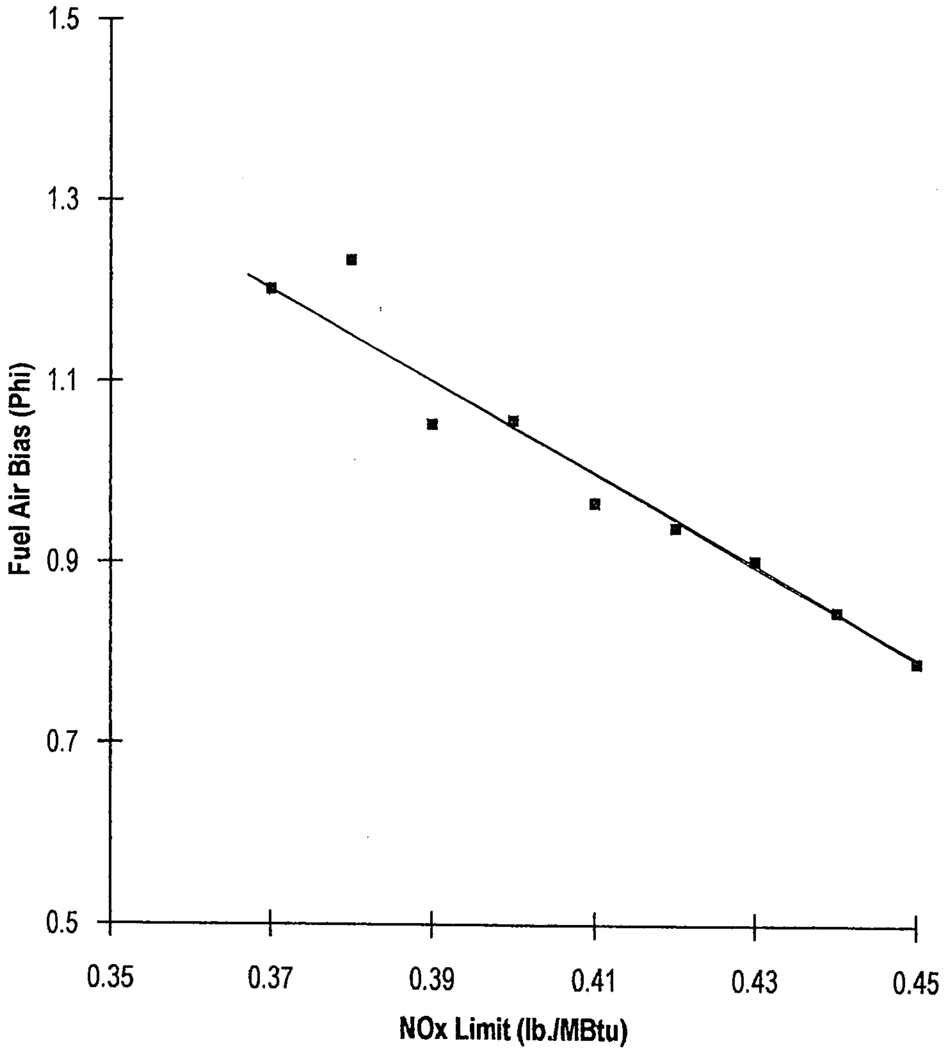

Figure 5.47

Full Load Averaged Optimal Fuel Air Bias vs.
Optimization Algorithm NOx Limit

Heat Rate Penalty Value. This is the quantity that is added to a vertex's heat rate value at the current iteration if the NOx level at that vertex exceeds the NOx limit. This parameter is simple in concept and the only precaution is making it too big or too small. For example, suppose that the scaled heat rate at a vertex is equal to -0.15 and the current NOx value at that vertex exceeds the specified NOx limit. The penalty value must be substantial enough so that when it is added to the current scaled heat rate value, it forces the algorithm to select that vertex as the current "worst point". This will, in turn, force the algorithm to move that vertex towards a smaller heat rate value that is closer to the NOx limit. If the penalty value exceeds the range of actual (unscaled) heat rate data used to train the neural network, the algorithm may not converge, although this situation never occurred during this study.

Conversely, a penalty value that is too small, especially if within the order of the function tolerance, will not be significant enough to influence the algorithm's next move when added to the current vertex heat rate. If the vertex's NOx exceeds the NOx limit but the penalty value is not sufficient enough to make that vertex the current "worst point", the benefit of the penalty function has been negated. Overall, this parameter only requires a reasonable balance between having the desired effect without jeopardizing the algorithm operation. The author has determined that the most dependable method for determining the heat rate penalty value is obtained by taking the difference between the minimum and maximum heat rate values (unscaled) within the training data and dividing by four. This quantity must then be scaled using Eqs. 5.1 -5.3 so as to maintain numerical compatibility within the algorithm.

These parameters affect the neural network equation optimization algorithm for heat rate using NOx as a constraint. Whenever a significant change to the neural network equations occur, these parameters should be checked to ensure optimal algorithm operation.

Results of the Full Load Heat Rate Equation Optimization.

Using a NOx limit of 0.38 lb./MBtu and several random initial simplex orientations, the following table details the best optimization result:

Table 5.16

Optimal Operating Conditions for Full Load
Unit Heat Rate and NOx Neural Network Equations

Parameter	Averaged Simplex Output
Economizer Oxygen:	1.53 %
Burner Tilt Angle:	5.1 degrees
Auxiliary Air Bias:	0.417
Fuel Air Bias:	1.12
Unit Heat Rate:	8935 Btu/kWh
NOx:	0.38 lb./MBtu

These operating conditions are very similar to the NOx neural network optimization results in Table 5.14. The main differences are the auxiliary air and fuel air distribution into the boiler. Unfortunately, there was no additional analysis involving heat rate within D'Agostini's report [4] and therefore checking the validity of these results is more difficult than the NOx-only scenario. However, ERC engineers reviewed these results and are in general agreement that they are feasible. Tests are needed in which the boiler at Potomac River Unit #4 is set at these conditions to see how close these results are to observed data. With the addition of a boiler cleanliness parameter and more parametric test data, the neural networks and thus the optimization results should be more closely aligned with actual boiler operation.

The optimization experiments with the full load networks answered several questions with regard to the best method of initializing the simplex algorithm without user interaction. The optimization algorithm is extremely sensitive to the initial simplex orientation. Also, the results from different initial simplex orientations can vary by as much as 10 to 20 Btu/kWh. These conditions dictated the need to try several different initial simplex orientations without user interaction.

It was decided among the ERC engineers and the author that the simplex will be initialized using the same random number generating routine, described earlier in this chapter ,and then the equations will be optimized. After the optimization routine has converged to an answer, two validity tests must be performed. First, the results must be checked to ensure that the optimal parameter values are within the range of the network training data. The output will be passed through a data filter that will compare the final results with the minimum and maximum values for each independent and dependent parameter. If the output contains values that are outside of the training data range, that answer will be discarded.

The second check compares the optimal parameter values, that have cleared the data filter, to any safety restrictions placed on independent parameters. If the algorithm determines the optimal economizer oxygen to be 1.7% but the user-defined safe O_2 operating limit is 1.8%, this algorithm result will be discarded. If the algorithm results pass the validity checks, the results are stored as the temporary "best solution". The simplex is re-initialized and this optimization process is repeated. It was determined by the ERC engineers and the author that once ten sets of acceptable results have been obtained, the best result from these ten will be outputted to the user.

The overall performance of the optimization algorithm parallels observed boiler behavior. Figures 5.44 to 5.47 show that as NOx is allowed to increase, the unit heat rate can decrease. This decrease in heat rate is caused by an increase in burner tilt angle (Figure 5.45). Subsequently, as NOx is allowed to rise, less overfire air is required and therefore the auxiliary air bias decreases (Figure 5.46) which improves heat rate. The fuel air bias (Figure 5.47) is shown to decrease when the NOx limit is raised. ERC engineers have not analyzed the impact of this bias parameter on overall boiler behavior and therefore any definite conclusions cannot be made. The algorithm always resulted in the minimum economizer oxygen possible. This is probably due to NOx having a large dependency upon O_2 but heat rate having a lesser dependence. Thus, to achieve the NOx limit, the O_2 should be minimized which doesn't affect the resulting heat rate very much.

45 MW Load Neural Network Equation Optimization

The 45 MW neural network equations failed to accurately model boiler operation and this subsequently affected the optimization routine's ability to provide correct results. The 45 MW network equations proved to be unstable and therefore the experimental analysis done for the full load case could not be repeated at this load. Furthermore, the 45 MW load received less emphasis in D'Agostini's report [4] than full load so verifying different results was not possible. However, a few sets of results were generated and are highlighted in the following table:

Table 5.17

Optimal Operating Conditions for 45 MW Load
Unit Heat Rate and NOx Neural Network Equations

Parameter	Averaged Simplex Output
Economizer Oxygen:	3.90 %
Burner Tilt Angle:	17.7 degrees
Auxiliary Air Bias:	0.000
Fuel Air Bias	0.321
Unit Heat Rate :	9671 Btu/kWh
NOx:	0.36 lb./MBtu

The heat rate value is rather high but the reader is reminded again that approximately 70% of the unit heat rate values used in the training data were estimated and were not calculated using the HEATRT code [8]. A significant difference in these results from the parametric field data is that the measured optimal heat rate occurred at high levels of auxiliary air bias whereas the optimization algorithm dictates an auxiliary air bias of zero. Such results reinforce the hypothesis that a controlling factor of this entire approach to heat rate and NOx optimization lies in the quality of the data.

Post-Processing of the Optimization Output Data

When the simplex algorithm has completed the optimization process, the result is the final position of the simplex vertices and the values for unit heat rate and NOx at those locations. These simplex vertices are simply combinations of the different independent parameters which for this study were economizer oxygen, burner tilt, auxiliary air and fuel air bias. The scaling of these parameters from neural network ranges to real world values is done within the optimization routine. The next step is to take these different combinations of parameters and calculate a final set of boiler operating conditions that the operator can identify. This is the purpose of the post-processor.

The underlying logic of the post-processor is to successively take each parameter and transform it back to a final, succinct answer. The first two, economizer oxygen and burner tilt angle, are straightforward. The algorithm averages the optimization routine results for each parameter and stores them in memory. These are used later for translating the air bias parameters back to damper settings and windbox pressure.

Translating the auxiliary air bias parameter back to damper settings requires the use of rules devised by ERC engineers. These rules were applied to all of the auxiliary air damper combinations possible at Potomac River Unit #4. There are five auxiliary air nozzles at each corner of the boiler (Figure 2.1), each with 5 degrees of openness. this results in 5^5 or 3125 different combinations. The rules applied are:

- the sum of all of the damper positions must be between 14 and 18 (full load) and between 8 and 11 for 45 MW
- the damper positions must be such that AA1 $\geq$ AA3 $\geq$ AA5 $\geq$ AA7 $\geq$ AA9
- no more than one damper may be set at a #1 position

These rules reduced the number of possible auxiliary air damper position combinations to 54 for full load and 20 for the 45 MW load. These different combinations and the auxiliary air bias value for each combination appears in Appendix A, along with the calculated auxiliary air bias values and the sum of the damper positions. These damper combinations and their respective bias parameter values are used to select the optimal auxiliary air bias values, determined by the optimization algorithm.

The optimal auxiliary air bias values, calculated by the simplex algorithm, are averaged together. This averaged value is compared to the set of auxiliary air bias parameters that are within Appendix A, which are termed the "allowable" bias parameters and damper combinations. Whichever of the "allowable" auxiliary air bias values that the averaged simplex value is closest to, that "allowable" bias value becomes the optimal auxiliary air bias parameter. Each "allowable" auxiliary air bias parameter has a specific set of "allowable" damper combinations, as dictated by the rules above. The post-processor routine goes back into the available training data set to find a data vector that has the same, or is closest to, combination of auxiliary air damper settings as those in the "allowable" damper settings. This forces the user to implement a combination of damper settings that was used to train the neural network which more closely links the neural network model to the observed boiler operation.

The fuel air bias is the most complicated parameter to translate back into real world operating conditions due to the complexity of the bias equation:

Fuel Air Bias: $$\phi = \frac{1}{n} \cdot \frac{\Omega_{ref}}{\Omega_{avg}} \cdot \sqrt{\frac{\Delta P_{wb}}{\Delta P_{wb,ref}}} \cdot \sum_i \frac{FA_i}{FA_{ref}} \tag{5.8}$$

where:

n = number of mills operating

ΔP_{wb} = windbox to furnace differential pressure ("H_2O)

FA_i = position (amount dampers are open) for ith row of fuel air dampers

Ω = mill feeder motor speed (rpm)

avg = average of all mill feeder speeds in operation (rpm)

ref = reference conditions: Ω_{ref} = 700 rpm

ΔP_{wb} = 2.5 "H_2O

FA_{ref} = 3

For full load operation, all four mills are in use and $n = 4$. The available data are again accessed by the routine and the average mill feeder motor speed is calculated, using all data vectors. Therefore, with the parameters n, Ω_{avg}, Ω_{ref}, $\Delta P_{wb,\,ref}$, and FA_{ref} known, these parameter values can be combined and mathematically reduced to a constant or:

$$\phi = \text{Constant} * \sqrt{\Delta P_{wb}} * \sum_i FA_i \tag{5.9}$$

The main difficulty lies in the relationship between the windbox pressure and sum of the fuel air damper positions. Since the degree of openness of these dampers affects the windbox pressure, one must be solved in terms of the other, or:

$$\sum_i FA_i = \frac{\phi}{\text{Constant} * \sqrt{\Delta P_{wb}}} \tag{5.10}$$

This equation requires an iterative solution, but first a few rules devised by the ERC engineers must be reviewed for the fuel air dampers:

- all four dampers must be set on positions between 3 and 5
- all dampers in a combination cannot differ by more than one position
- the sum of all dampers must be between 12 and 20

These rules force this variable to have a discrete set of solution possibilities (Appendix A), but the question that remains is, how does one calculate the windbox pressure value, ΔP_{wb}? ERC engineers determined that windbox pressure is a function of economizer oxygen and the degree of openness of all air dampers. However, the exact interactions of the O_2 and damper positions at different operating conditions are unknown. Finding these interactions required a small, separate neural network. The data used for training this network came from the same parametric data used for the full load NOx and heat rate network. However, instead of using the damper positions to calculate bias parameters, individual damper positions were combined for both the auxiliary air and fuel air registers. The auxiliary air dampers were summed together, as were the fuel air damper positions, for each data vector within the training data set. This added two new data components to every training data vector, the sum of auxiliary air damper positions and the sum of fuel air damper positions.

The windbox pressure network consisted of three inputs (economizer oxygen, sum of the auxiliary air damper positions, sum of the fuel air damper positions) and one output (windbox pressure). The resulting equations were substituted into Eq. 5.10 as the windbox pressure variable, ΔP_{wb}. Equation 5.10 is then incorporated into an iterative solution procedure and the two variables are incrementally changed until convergence is reached. Once converged, the windbox pressure and its corresponding sum of fuel air dampers is now known.

Using the sum of the fuel air dampers found in Eq. 5.10, this value is used in the same manner as the designated auxiliary air bias value described above. Only certain combinations of fuel air dampers will result in the sum found in Eq. 5.10. Therefore, the fuel air damper sum from Eq. 5.10 is matched to the fuel air damper sums and combinations dictated by the ERC rules described above (Appendix A). Once the fuel air damper sum is known, the corresponding damper position combinations that comprise that sum are used to search the training data base for an exact or similar match of fuel air damper settings. This search procedure is identical to the auxiliary air damper position search described above. The final combination of fuel air damper positions is then outputted to the user for implementation.

All of the operator controlled boiler parameters are now known for the optimal conditions found through the neural networks and simplex optimization algorithm. These values are relayed to the user so that the boiler parameters can be adjusted so as to achieve better heat rate performance while maintaining the desired NOx emissions level.

SUMMARY

This chapter concludes with a listing of the best selections for the neural network modeling and optimization process for the Potomac River Unit #4 boiler.

Table 5.18

Recommended Settings for Full Load and 45 MW Neural Networks and the Optimization Algorithm for the Potomac River Unit #4 Boiler Study

Number of Inputs:	4
Number of Hidden Layer PEs:	3
Hidden Layer Transfer Function:	Tanh
Output Layer Transfer Function:	Tanh or Linear
Epoch Size (NOx networks):	8
Epoch Size (Heat Rate networks):	16
Learn Count (NOx networks):	30,000
Learn Count (Heat Rate networks):	65,000
Hidden Layer/Output Layer Learning Rate:	0.3 / 0.1
Hidden Layer/Output Layer Momentum :	0.4 / 0.4
Error Function:	Standard Error
Learning Rule:	Norm-Cum-Delta
Summation Function:	Normal Summation
Optimization Algorithm Function Tolerance:	0.10 (13.5 Btu/kWh)
Optimization Algorithm NOx Limit:	-0.70 (0.38 lb./MBtu)
Optimization Algorithm Penalty Value:	1.00 (9157 Btu/kWh)

The suggested values in the above table are based upon the research conducted during this study and apply to this particular problem. These values are guidelines for continued research in this area and provide the user with a good starting point. If time or computational limits exist, a backpropagation network for any application can be optimized using the approach below. The order of parameter optimization is based upon network sensitivity to these parameters and if no limits for optimization exist, then the user should follow the outline given earlier in this chapter. Ultimately, only the first three topics in the following list must be examined to get a neural network close to optimal operating conditions while the latter considerations are for fine-tuning the network.

1. The crucial element is the learn count. The learn count must be optimized experimentally or by using a "save best" execution scheme. This entails setting an upper limit on the learn count, such as 100,000, and then calculating the R-Square parameter at pre-determined intervals, such as 10,000. The best combination of R-Squares for the training and testing data sets determines the optimal learn count, similar to the previous discussion on optimizing the number of hidden layer PEs. This technique can be automated into the NOx control package and, therefore, it is currently being pursued.

2. The epoch size finds underlying frequencies within the training data and therefore must be optimized. The user may use a default value of 16 initially, but this study found that epoch values of 4, 8, and 12 should be examined as well for optimal network operation. If these values cannot be checked, then the default epoch size of 16 suffices without too great a decrease in accuracy.
3. The number of hidden layer PEs is important for adding the correct degree of non-linearity to the network mapping. The user may start with the initial value found from Eq. 3.27, but a network with one less PE must be tested along with networks containing up to three additional PEs greater than this initial value. As with epoch size, if the default value is used, some accuracy will be lost and the actual system behavior may not be mapped correctly.
4. The transfer functions for the hidden and output layers are not as critical since backpropagation usually works best with the hidden layer having a Tanh transfer function and the output layer having either a linear or Tanh transfer function. Both situations provide very similar results and no additional experiments are necessary. Therefore, these functions can be set as default values for any backpropagation network application.
5. The backpropagation network performs best with the normal summation function (Eq. 3.25), the standard error function (Eq. 3.3), and the Norm-Cum-Delta learning method [11]. Each of these are within the NeuralWare software.
6. The learning rate and momentum coefficient are the final considerations. The default values for the hidden layer (learning rate = 0.3 and momentum = 0.4), and the output layer (learning rate = 0.1 and momentum = 0.4), are usually sufficient. However, if the root-mean-square error curve for the difference between predicted and actual output values does not decrease smoothly, then these values must be adjusted as described earlier in this chapter.

The optimization algorithm is most sensitive to the initial simplex orientation and resolution of this sensitivity was discussed earlier. The algorithm is further controlled by the function tolerance, NOx limit, and penalty value, in this order. Selecting these values was discussed earlier in this chapter and programming them into the software package is not difficult. Overall, the parameters in the above table are easy to implement into any automated process. Further work is needed to determine whether these are specific only to the Potomac River Unit #4 boiler or whether these values can be substituted into models for other, completely different, boiler configurations.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

Results presented in this thesis indicate that boiler operation and NOx emissions at Potomac River Unit 4 can be mathematically modeled and optimized using neural networks. The feasibility of developing an on-line boiler control software for this unit has been proven. However, many practical issues concerning implementation and interaction of the recommended solutions remain to be addressed. Many of these issues are currently being researched as this NOx control software package matures towards its final form.

NEURAL NETWORKS

The use of neural networks for simulation and control of power plant operation was established well before this thesis [35]. Many applications have used only the predictive modeling capabilities of neural networks whereas this thesis took this process one step further and mathematically optimized the resulting network equations. The following discussion highlights the benefits and weaknesses of using neural networks for boiler modeling along with recommendations for continued research.

Strengths

Neural networks were proven valuable on several counts. First, they are adaptive. Because they learn by using observed behavior, they don't require a theoretical understanding of boiler operation. They infer solutions from the data presented to them and capture subtle relationships that exist within the operational data. A neural network can use the relationships learned from the training data to predict boiler responses to data it has never seen before. If sufficient amounts of data exist, imperfect and incomplete data may be used for training which provides a degree of fault tolerance.

The non-linearity provided by the hidden layer captures complex interactions among the input variables. To express these interactions in mathematical form, it would normally require extensive statistical analysis and mathematical function fitting of the data. Neural networks are highly parallel and their numerous identical, independent operations can be executed simultaneously. This increase in speed and power allows for more experimentation and hypothetical modeling of boiler operation whereas statistical analysis takes more time and effort to achieve the same results.

Weaknesses

The primary weakness of neural networks is their dependence upon the quantity and quality of data. The success of the full load networks and the failure of the 45 MW networks proved this point quite well. As physical or operational characteristics of the boiler change, new data must be collected so as to obtain the most accurate and reliable neural network models. The expert system portion of the NOx control software package accomplishes this, resulting in a sufficient number of well dispersed data components within their operating ranges.

Neural network results are often difficult to interpret unless an understanding of the physical process exists, or the data used to train the network are such that these relationships are readily seen. Human knowledge, as well as statistics, are very useful elements in analyzing network performance. This study had the benefit of comparing its results to those of D'Agostini et al. [4] and other ERC research findings. If the research had based decisions solely upon the neural network performance statistics, the probability for failure would have increased dramatically. Therefore, the need for understanding the physical process in this application was paramount to its success.

The absence of definite rules for training and optimizing a neural network complicates the implementation of this control package. The fundamental parameters, within the NeuralWare software, that affected network operation were identified. Several rules of thumb were used throughout this thesis to simplify the network optimization process. However, if different software packages or neural networks other than backpropagation are used, this process of identifying key operating parameters must be repeated.

Recommendations

Improving the methods of configuring, training, and optimizing the neural networks without user interaction is a fundamental concern for future success of this research. Unfortunately, any process will still require some form of manual fine tuning but minimizing that quantity is a necessity. Determining how to implement these rules and determining when they should be applied will be dependent upon how the neural network training process is programmed into the overall software package.

Further studies on the effects of data quality upon network operation are required. These include using fixed values for certain data components throughout training, training a network using filtered and unfiltered on-line data, and improving the descriptive capabilities of the bias parameters. Each of these areas will affect network operation and thus the numerical optimization process. Deciding when to use the neural network for optimization purposes is a concern as is the need to determine whether to update an existing network or initiate an entirely new model. Additionally, the decision as to what data to use to update or retrain a network must be addressed.

Finally, simpler and more dynamic neural network types should be explored. An alternate form of artificial intelligence, fuzzy logic, is another possibility for controlled process operation that has already been used elsewhere [36]. Other types of neural networks that are available are radial basis function networks and counter propagation networks [9, 11]. While these approaches may not replace the existing backpropagation model, perhaps they can be applied to enhance its performance.

NUMERICAL OPTIMIZATION

The ultimate goal for the optimization process was to minimize unit heat rate subject to NOx staying at or under 0.38 lb./MBtu. This was achieved at full load but, for reasons discussed previously, not at the 45 MW load level. The underlying goal of the optimization process is to evaluate the function being optimized as few times as possible and achieve a valid solution. The Nelder and Meade Simplex Algorithm [27] proved to be accurate and robust enough for this study, considering it was very subject to the effects of the quality of network training data.

Strengths

The alterations made to the original algorithm, (randomized starting points and penalty function), added more complexity to the optimization process but enhanced its reliability. These additions gave the algorithm characteristics of the next higher order of optimization processes, the conjugate direction methods, detailed in Chapter 4. These alterations provided the required power to optimize the function quickly as well as provide insight as to the best way to implement and operate the optimization routine.

The simplicity of the algorithm allows for easy implementation and experimentation of the optimization process. Parametric changes in the algorithm operating constants and the subsequent effects on the solution allowed for guidelines to be established for future use. This would be more difficult if algorithms of greater complexity were used; and the small increase in accuracy or speed from such algorithms did not justify introducing such complexity.

Weaknesses

The algorithm's sensitivity to the initial simplex orientation causes difficulty in calculating a reliable result. This problem was overcome by the reinitialization process described in Chapter 5. However, if network models of greater complexity are used, this sensitivity may become difficult to circumvent. Furthermore, finding the best algorithm initial conditions may prove to be more difficult, even with the rules of thumb developed in this study.

The algorithm's ability to find the optimal operating condition is entirely dependent on the network equations it is optimizing. These equations are, in turn, dependent primarily upon the data used for training the network. If bad data are used from the outset, then the entire neural network and optimization process must be repeated. Also, the optimization algorithm cannot be updated with the addition of new data. If the network connection weights change, then the optimization process must be entirely redone. Fortunately, the computation time for this algorithm is not very great and so repeating the optimization process does not pose a burden upon the NOx control software package.

Recommendations

Further research on other optimization algorithms may result in a combination of different approaches resulting in a hybrid optimization process. This could negate the dependency of the simplex on its initial orientation and possibly provide greter accuracy and stability to the optimization process. Additionally, this could improve the speed of the optimization process as well as the reliability of its solutions. However, the difficulty in automating a hybrid algorithm may outweigh its usefulness. Regardless, there is potential for improvement and so alternate optimization methods should be examined in greater detail.

A better method of ensuring that the optimization process will result in outputs that are deemed safe by plant operating personnel is required. Currently, the optimal operating conditions are calculated and then checked against the safe operating limits which are defined by the user prior to performing the optimization. If the optimal solution includes unsafe operating conditions, then the results are discarded and the optimization is repeated. A potentially better method is to filter the training data and remove (or simply do not collect data) within the defined unsafe operating regions. This results in the neural network being trained with data that are within safe operating zones and prevents the optimal solution from containing unsafe conditions. This approach is currently being researched for incorporation into the NOx control software package.

REFERENCES

1. E. Levy, et al. "Application of Boiler Performance Improvement Tools To NOx Control." Proceedings of the 1992 EPRI Heat Rate Improvement Conference, Birmingham, Alabama, November, 1992.

2. D. Eskenazi, et al. "Month-Long Baseline NOx Emissions and Performance at Potomac River Unit 4," Lehigh University, Energy Research Center Report 93-500-712, March, 1993.

3. D. Eskenazi, et al. "Results of Extended Phase I and Phase II Low NOx Testing at Potomac River Unit 4," Lehigh University, Energy Research Center Report 93-500-10-16, April, 1993.

4. M. D'Agostini, et al. "Analysis of Parametric Low NOx Test Data From Potomac River Unit 4," Lehigh University, Energy Research Center Report 94-500-01-01, January, 1994.

5. E. Levy, et al. "Boiler Performance Monitoring And Continuous Emissions Monitoring, And Emissions Control: An Opportunity For Linkage," Proceedings of the 1992 EPRI Heat Rate Improvement Conference, Birmingham, Alabama, November, 1992.

6. M. D'Agostini. "Modeling NOx Formation in Pulverized Coal Combustors" A General Examination Report Prepared for Doctoral Degree Requirements at Lehigh University, Bethlehem, Pennsylvania, March, 1994.

7. D. Eskenazi, et al. "Summary of NOx Emissions Pretest at Potomac River Unit 4," Lehigh University, Energy Research Center Report 92-500-2-5, February, 1992.

8. D. Eskenazi, et al. "User's Guide For HEATRT Code," Lehigh University, Energy Research Center Report 92-500-3-13, March, 1992.

9. NeuralWare, Inc. Neural Computing, NeuralWare, Inc., Pittsburgh, 1991.

10. A. Lapedes and R. Farber. "Non-Linear Signal Processing Using Neural Networks: Prediction and System Modeling," Los Alamos Laboratory Report LA-UR-87-2662, June, 1987.

11. NeuralWare, Inc. Reference Guide, NeuralWare, Inc., Pittsburgh, 1991.

12. D. Hammerstrom. "Neural Networks at Work," IEEE Spectrum, June, 1993, p. 30.

13. A. J. Owens and M. T. Mocella. "An Experimental Design Advisor and Neural Network Analysis Package," Proceedings of the 1991 International Workshop on Adaptive Neural Networks, September, 1991.

14. C. Klimasauskas."Applying Neural Networks, Part III: Training a Neural Network," PCAI, May/June, 1991, pp. 20-24.

15. S. P. Chitra. "Use Neural Networks for Problem Solving," Chemical Engieering Progress, April, 1993, pp. 44-52.

16. J. P. Guiver and C. Klimasauskas. "Applying Neural Networks Part IV: Improving Performance," PCAI, July/August, 1991, pp. 34-41.

17. J. H. Mathews. Numerical Methods for Mathematics, Science, and Engineering 2nd Edition. Prentice Hall, Englewood, NJ: 1992, pp. 28-29.

18. D. H. Nguyen and B. Widrow. "Neural Networks for Self Learning and Control Systems," IEEE Control Systems Magazine, April, 1990, pp. 18-23.

19. R. C. Eberhart and R. Dobbins. Neural Network PC Tools. Academic Press, Inc., San Diego: 1990. pp.161-173.

20. J. D. Keeler. "Prediction and Control of Chaotic Chemical Reactions Via Neural Network Models," Proceedings of the 1993 Conference on Artificial Intelligence in Petroleum Exploration and Production, Plano, TX, May, 1993.

21. Weigend, Hubermand, and Rumelhart. "Predicting the Future: A Connectionist Approach," International Journal of Neural Systems, March, 1990.

22. C. Klimasauskas, "An Introduction to Neural Networks with Applications to An Adaptive PID Controller," Proceedings of the 1991 Society of Manufacturing Engineers Conference on Computer and Automated Systems, Chicago, November, 1991.

23. C. Klimasauskas, "Neural Networks: An Engineering Perspective," IEEE Communications Magazine, September 1992, pp. 50-53.

24. J. Hertz. A. Krogh, and R. Palmer. Introduction to the Theory of Neural Computation. New York: Addison-Wesley, 1991.

25. M. Piovoso and A. Owens. "Sensor Data Analysis Using Artificial Neural Networks," 4th International Conference on Chemical Process Control, South Padre Island, TX., May, 1991.

26. J. Leonard et al. "A Neural Network Architecture that Computes Its Own Reliability," Computers and Chemical Engineering, September, 1992.

27. W. Press, et al. Numerical Recipes in FORTRAN, 2nd Edition.Cambridge, U. K.: Cambridge University Press, 1992.

28. T. Ragsdell et al. Engineering Optimization Methods and Applications. New York: Wiley and Sons, 1983.

29. R. L. Zahradnik. Theory and Techniques of Optimization for the Practicing Engineer, New York: Barnes and Noble, 1971.

30. M. J. Box, "A New Method of Constrained Optimization and a Comparison with Other Methods," Computer Journal, August, 1965, pp. 42-52.

31. "IMSL: FORTRAN Subroutines for Mathematical Applications, Vol. 3, Version 2.0", IMSL Inc., Houston, TX, 1991.

32. W. E. Biles. "Optimization of Multiple Response Simulation Models," University of Notre Dame Final Report, ONR-Contract N00014-76-C-1021, 1978.

33. "Microsoft Excel 4.0" Microsoft Corporation, Bellvue, WA, 1992.

34. D. Garson. "Interpreting Neural Network Connection Weights," AI Expert, April, 1991.

35. K. F. Reinschmidt. "Neural Networks: Next Step for Simulation and Control," Power Engineering, November, 1991, pp. 41-45.

36. S. Fraleigh. "Fuzzy Logic and Neural Networks, Practical Tools of Process Management," PC AI, May/June, 1994.

APPENDIX A

Full Load and 45 MW Load Auxiliary Air and Fuel Air Damper Position Combinations and Corresponding Bias Parameter Values

45 MW Auxiliary Air Damper Combinations and Corresponding Bias Parameter Values

Aux Air Damper 1	Aux Air Damper 3	Aux Air Damper 5	Aux Air Damper 7	Aux Air Damper 9	Auxiliary Air Bias	SUM
2	2	2	2	2	0.000	10
2	2	2	2	1	0.167	9
3	2	2	2	2	0.167	11
2	2	2	1	1	0.250	8
3	2	2	2	1	0.333	10
3	2	1	1	1	0.417	8
3	2	2	1	1	0.417	9
3	3	2	2	1	0.417	11
3	3	1	1	1	0.500	9
3	3	2	1	1	0.500	10
3	3	3	1	1	0.500	11
4	2	2	2	1	0.500	11
4	2	1	1	1	0.583	9
4	2	2	1	1	0.583	10
4	3	1	1	1	0.667	10
4	3	2	1	1	0.667	11
4	4	1	1	1	0.750	11
5	2	1	1	1	0.750	10
5	2	2	1	1	0.750	11
5	3	1	1	1	0.833	11

Full Load Auxiliary Air Damper Combinations and Corresponding Bias Parameter Values

Aux Air Damper 1	Aux Air Damper 3	Aux Air Damper 5	Aux Air Damper 7	Aux Air Damper 9	Auxiliary Air Bias	SUM
3	3	3	3	3	0.000	15
3	3	3	3	2	0.167	14
4	3	3	3	3	0.167	16
4	4	3	3	3	0.250	17
4	4	4	3	3	0.250	18
4	3	3	3	2	0.333	15
4	4	4	4	2	0.333	18
5	3	3	3	3	0.333	17
4	3	3	2	2	0.417	14
4	4	3	3	2	0.417	16
4	4	4	3	2	0.417	17
5	4	3	3	3	0.417	18
4	3	3	3	1	0.500	14
4	4	2	2	2	0.500	14
4	4	3	2	2	0.500	15
4	4	4	2	2	0.500	16
5	3	3	3	2	0.500	16
4	4	4	4	1	0.500	17
4	4	3	3	1	0.583	15
4	4	4	3	1	0.583	16
5	3	2	2	2	0.583	14
5	3	3	2	2	0.583	15
5	4	3	3	2	0.583	17
5	4	4	3	2	0.583	18
4	4	3	2	1	0.667	14
4	4	4	2	1	0.667	15
5	3	3	3	1	0.667	15
5	4	2	2	2	0.667	15
5	4	4	2	2	0.667	17
5	4	4	4	1	0.667	18
5	5	3	3	2	0.667	18
5	4	3	2	2	0.667	16
5	3	3	2	1	0.750	14
5	4	3	3	1	0.750	16
5	4	4	3	1	0.750	17
5	5	2	2	2	0.750	16
5	5	3	2	2	0.750	17
5	5	4	2	2	0.750	18
5	4	2	2	1	0.833	14
5	4	3	2	1	0.833	15
5	4	4	2	1	0.833	16
5	5	3	3	1	0.833	17
5	5	4	3	1	0.833	18
5	5	2	2	1	0.917	15
5	5	3	2	1	0.917	16
5	5	4	2	1	0.917	17
5	5	5	2	1	0.917	18

Full Load and 45 MW Fuel Air Damper Combinations
and Corresponding Bias Parameter Values

Fuel Air Damper 2	Fuel Air Damper 4	Fuel Air Damper 6	Fuel Air Damper 8	SUM
3	3	3	3	12
3	3	3	4	13
3	3	4	3	13
3	4	3	3	13
4	3	3	3	13
3	3	4	4	14
3	4	3	4	14
3	4	4	3	14
4	3	3	4	14
4	3	4	3	14
4	4	3	3	14
3	4	4	4	15
4	3	4	4	15
4	4	3	4	15
4	4	4	3	15
4	4	4	4	16
4	4	4	5	17
4	4	5	4	17
4	5	4	4	17
5	4	4	4	17
4	4	5	5	18
4	5	4	5	18
4	5	5	4	18
5	4	4	5	18
5	4	5	4	18
5	5	4	4	18
4	5	5	5	19
5	4	5	5	19
5	5	4	5	19
5	5	5	4	19
5	5	5	5	20

VITA

Preston Charles Lee was born to Eloise and Windsor Lee in Allentown, Pennsylvania, on March 11, 1968. He graduated as salutatorian from Valley Forge Military Academy in 1985 and attended the United States Military Academy, at West Point, New York, from 1985 to 1989. Upon graduating on May 24, 1989 with a Bachelor of Science Degree in Mechanical Engineering, he served as a lieutenant in the United States Army Corps of Engineers at Fort Lewis, Washington, and the Kingdom of Saudi Arabia. After 3 years he left the Army to attend Lehigh University as a graduate student in the Mechanical Engineering Department. He currently works in the Energy Research Center, under the direction of his advisor, Dr. Edward K. Levy.