1994

Application of expert systems to nitrogen oxide emissions control

Sara Woldehanna
Lehigh University

AUTHOR:

Woldehanna, Sara

TITLE:

Application of Expert Systems to Nitrogen Oxide Emmissions Control

DATE: January 15, 1995

Application of Expert Systems to Nitrogen Oxide Emissions Control

by

Sara Woldehanna

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

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Mechanical Engineering.

December 2, 1994

Date

Thesis Advisor

Chairperson of Department

Acknowledgments

I would like to thank Dr. E. Levy for giving me an opportunity to study under him. I would not be here if not for the love and strong belief my parents had in me. I am also grateful to all my friends that encouraged me on and listened to me at the right times.

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

To comply with the 1990 amendments to the Clean Air Act, power plants take various measures to reduce nitrogen oxide (NO_x) emissions. Experiments on NO_x control have shown that NO_x emissions from coal fired power plants can be reduced by optimizing boiler operations. Optimal operating conditions can be found through parametric testing of the boiler in question.

To aid electric utilities in reducing NO_x emissions through optimal boiler operation, two off-line computer advisors, optimization and diagnostic, were developed. Expert systems, artificial intelligence tools useful in modeling qualitative or uncertain knowledge, were used to develop the software, utilizing knowledge on the operating characteristics of boilers gained through previous projects by the Energy Research Center.

The optimization advisor guides a plant operator or engineer through a series of experiments to achieve a certain objective. The objective might be to achieve the lowest NO_x possible by the boiler or to reach a specified NO_x level while optimizing unit performance. Ensuring safe boiler operation is a major consideration when conducting these experiments.

The diagnostic advisor is targeted towards a boiler that has already been optimized for low NO_x operation. If such a unit were to suddenly or gradually start to produce high levels of NO_x, this software could be used to detect possible causes of the problem and recommend solutions.

At this time, both software packages are configured for the Potomac

Electric Power Company's (PEPCO) Potomac River Station Unit 4, which is a 108 MW, tangentially fired boiler. While the diagnostic software is applicable for the entire load range of the unit, the optimization software is currently configured only for full load operation. At the present, the optimization software has been completed and simulated runs using correlations of data from Potomac River Unit 4 have proved to be successful. The diagnostic advisor's major components have been completed, but it will require revisions and additions before it is ready for use by utility personnel.

INTRODUCTION

Since nitrogen oxides (NO_x) have been shown to cause photo-chemical smog and acid rain, Federal regulations require power plants to bring their stack emission levels below a certain standard. This thesis discusses efforts to reduce NO_x emissions from coal fired power plants.

Nitrogen oxide (NO) makes up about 95% of NO_x emissions from coal fired power plants while nitrogen dioxide (NO_2) constitutes the rest. In emission control studies, NO_x is more commonly identified by the source of nitrogen in the NO_x compound. "Thermal NO_x" is formed from nitrogen in the combustion air and "fuel NO_x" is formed from nitrogen bound in the fuel [1]. Various theoretical and experimental studies have suggested models for the formation of both types of NO_x. This thesis is based on the work done by The Energy Research Center (ERC).

Since concern over emissions were first raised, various techniques have been designed to reduce NO_x. Flue gas treatments, like catalytic decomposition, selective non catalytic reduction, and adsorption with activated charcoal, remove NO_x after it has been formed [1]. Low NO_x burners have been designed to reduce NO_x during the combustion process.

The Energy Research Center (ERC), in collaboration with several electric utilities and funded by Electric Power Research Institute (EPRI), has been engaged in projects that attempt to reduce NO_x emissions through optimization of boiler operation [13,16,17,18,19]. These projects have shown that significant

reductions in NO_x can be obtained by boiler control optimization and improved boiler maintenance practice. The savings in NO_x associated with such practices have led in some cases to the ability to avoid expensive retrofit of low NO_x burners. Boiler optimization has the added advantage of helping to improve boiler efficiency and power plant performance.

The ERC projects in boiler optimization have helped characterize boiler performance and behavior as well as the factors that control NO_x formation. The control parameters that most affect NO_x formation have been identified and studied and the relationship of NO_x formation with boiler performance and safety parameters have been established.

This thesis deals with the development of two computer advisors that model the knowledge described above. The optimization and diagnostic advisors are designed to aid power plants achieve low NO_x emissions in much the same way as an engineer with a NO_x control background. Both advisors are intended for off-line use. Expertise on boiler performance, NO_x formation, safe boiler operating procedures, and experimental and data analysis methods were systematically gathered from ERC engineers and other relevant sources. The knowledge gathered was generally qualitative, which suggested the relevancy of artificial intelligence as a modeling tool.

Expert systems are one type of artificial intelligence tools that mimic human methods of synthesizing solutions. They differ from conventional computer programs in that they do not rely on quantitative algorithms, but they

are able to function under heuristic models and also process qualitative or uncertain information. Expert systems are discussed in more detail in the next chapter. Neural networks, another type of artificial intelligence, are also used in the advisor packages. Neural networks build and synthesize a model of a system using available information in much the same manner as brain neurons.

In the advisor packages, an expert system is used to model the expertise gathered from ERC engineers. It interacts with the user to analyze qualitative and quantitative information and also guides the user through a series of questions, tasks and experiments to obtain more information. A neural network is used to analyze any data that result from these interactions.

This thesis covers the development of the expert system portions of the optimization and diagnostic advisors. The optimization advisor guides the user through a series of experiments on boiler control parameters that are known to affect NO_x formation. The advisor monitors safety parameters constantly and recommends test points that follow safe boiler operating procedures. The objective of these tests is to find the operating conditions that give low NO_x. The data from these experiments is then analyzed by a neural network-optimization package (see reference 2) that finds optimum operating conditions that give a specified low NO_x level with high unit performance.

The diagnostic advisor is targeted towards a boiler that has already been optimized for low NO_x operation. This advisor is used when such a boiler starts producing unexpectedly high NO_x either gradually or suddenly for reasons that

are yet undetermined. The diagnostic advisor was modeled using past experiences of any such occurrences by ERC engineers as well as speculations of possible problem areas using knowledge of boiler operations and NO_x formation. This advisor steps the user through a series of questions and tasks in an effort to point to a possible cause of the high NO_x levels.

Both advisors were designed for pulverized coal, tangentially fired, steam generating plants, and are configured for PEPCO's Potomac River Unit 4. This unit has a full load capacity of 108 MW and operates at subcritical pressures with single superheat and reheat cycles. Suction mills provide pulverized Bituminous coal to four burner levels at four corners. Steam temperatures are controlled by tilting burner mechanisms. While the diagnostic software is applicable throughout the load range of the unit, the optimization software is currently configured only for full load operation.

In this thesis the following topics are covered:

- Overview of expert systems and their use in advisors.
- Discussion of the knowledge base used in the advisors.
- Descriptions of algorithms used to model knowledge and an over-view of the advisor architecture.
- Results of tests conducted on advisors, future plans and recommendations.

EXPERT SYSTEMS

Shortliffe, one of the developers of the earliest medical expert systems, described artificial intelligence (AI) as "the intelligence of any machine that performs a task that a century ago would have been a uniquely human intellectual ability." Around the middle of this century, as faster and more efficient calculating machines were being developed and the first computers were coming out, scientists began to be interested in the possibility of intelligent machines. It was apparent that even though computers were able to do calculations and perform certain tasks hundreds of times faster than human beings, they were not able to compete in areas where "human intelligence" was required. For example, speech recognition, vision, creation, design, telling stories and solving problems without clearly defined algorithms remained outside of computers' capabilities. So, could computers ever become intelligent?

In order to determine if machines could be intelligent, it is first necessary to understand what intelligence means. Webster's New Universal Unabridged Dictionary defines intelligence as:

> " a) the ability to learn or understand from experience; ability to acquire and retain knowledge; mental ability
> b) the ability to respond quickly to a new situation; use of the faculty of reason in solving problems, directing conduct, etc... effectively.
> c) in psychology, measured success in using these abilities to perform certain tasks."

Intelligence is related to utilizing the abilities to learn, understand and reason to develop useful skills. In the second half of the 20th century, scientists from

different fields such as mathematics, biology, psychology and philosophy undertook the task of examining how humans learned, understood or reasoned. Others like mathematician Alan Turing pioneered the field of artificial intelligence by examining the criteria for a machine to be intelligent.

In his 1950 article "Computing Machinery and Intelligence", Turing devised a thought experiment that would determine if a computer is "thinking" independently or just following instructions. The Turing experiment proposed that if a person talking to a machine can't tell that he is not talking to another human being then the machine could be labeled intelligent [3]. This suggested that if a machine can imitate human intelligence without necessarily replicating it, it could be called intelligent. Thus, the intelligence of the computer does not really have to be rooted in reality, and all its knowledge or data base does not have to be associated with objects in the outside world. For example, a ball in the computer's "mind" would just be a concept with the characteristics of roundness, bounciness and certain dimensions but the computer would not associate it with a physical ball the way a human being would.

The study of human intelligence took two different paths: biological and behavioral. In the former, scientists tried to understand the biological basis for human intelligence. In 1943, McCulloch and Pitts of the University of Illinois proposed the first computer model of the human brain with receptors, neurons and the nervous system [4]. Neurons filter out and synthesize the information or stimulus that they receive from the receptors in the eye, ear, nose, skin and

tongue. The result of the processing done by the neurons comes out as an output either in the muscles or in a conclusion reached in the brain. Artificial neural networks are the result of such studies. One of the great advantages of artificial neural networks is their ability to learn about a particular domain on their own by just interacting with their environments. Artificial neural networks have now been successfully applied in machine or process control, image processing (machine vision), signal processing, financial and economic prediction and other areas [5].

The second area of human intelligence research focused on the

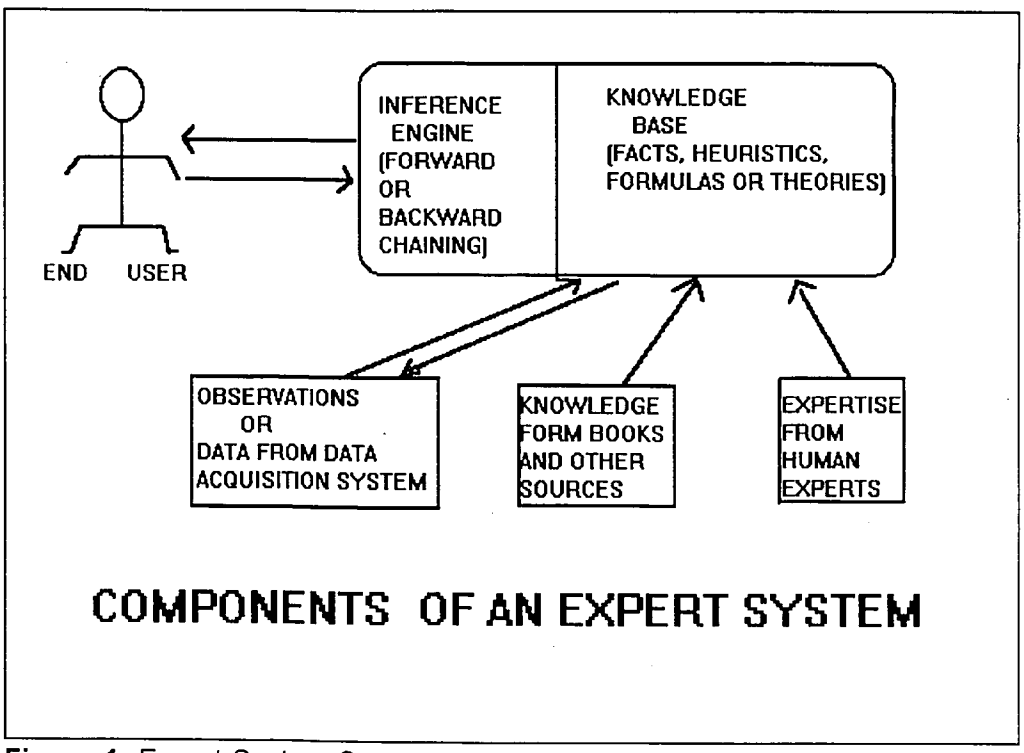

Figure 1 Expert System Components

processes of human reasoning, understanding and learning. Cognitive psychology, which developed at about the same time as AI, contributed a lot to the understanding of these processes. This research area examined uniquely human abilities like language, memory and problem solving.

One of the most explored areas of machine intelligence is the problem solving process. One model sees problem solving as the process of going from an initial state to a final state following any applicable rules; while the problem is the initial state, the solution or goal is the final, desired state. Humans employ different reasoning processes to identify a solution. One method is to explore or search through all the possible and available options. A person explores a particular option until a rule violation (e.g. a physical law) occurs or a solution is found. The earliest expert systems used this approach in games. A number of successful game programs like chess explored all potential moves by looking ahead at possible consequences a few steps further and deciding on the "best" move. This is the same process employed by human chess players and today some of these powerful computer chess games beat even the most accomplished human chess players.

Even though the approach described above was very successful in modeling one aspect of human reasoning method, it became evident that most of the problems that humans excelled at solving required some specific knowledge. This led to the origin of knowledge based systems, otherwise known as expert systems. In knowledge based systems, specific rules gained

either from experience or other learning sources are used to find a solution to a problem. For example, a doctor, in seeing a particular set of symptoms in a patient, does not order all the tests that might be at his/her disposal. Instead, using his/her medical knowledge and his/her past experience with similar symptoms, he/she quickly eliminates a lot of the options and explores only the most "likely" ones.

Thus, an expert in a particular field uses general deduction methods, knowledge specific to the field and heuristic rules in solving a problem quickly and efficiently. Knowledge constitutes what is known about that domain. The knowledge might include formulas, theories, observed facts, and other types of data. Heuristics, on the other hand, are rules of thumb that an expert uses to solve a problem. An expert develops heuristics from experience or by learning from another expert. Thus, an expert in a particular field might not know why a particular action produces results but knows from experience that most of the time that action produces the desired results. While heuristics do not guarantee a result, they are usually a short cut to a solution. It is important to understand that not all the options will be explored in this scenario, and the best solution might not be found all the time. But a "good enough" solution is found most of the time very quickly and efficiently.

In expert systems there are two important components: the knowledge base and the inference engine. The knowledge base consists of the domain specific knowledge and heuristics described above. It represents all that is

known about the problem and the different methods of solving problems in that domain. The knowledge base can be represented in one or more representation schemes like objects, procedural programs, data bases, frames or rules.

One of the most popular and successful methods of representing knowledge in AI is rules. Rule based systems are especially suited for representing heuristics and uncertain or qualitative knowledge. Rules have an if ..., then ... form, and knowledge based systems that represent knowledge in such a way are referred to as rule based. "If it rains, then the ground gets wet" is an example of a simple common sense rule. If the fact that it is raining is known to be true, the above rule would conclude that the ground is wet.

Rule based systems use "pattern matching" to match the premise of a rule to a known fact in the system [7]. A fact is a statement that tells what is known to be currently true about a system. A fact is represented in the knowledge base as a sequential collection of symbols or words. For example "it is raining" is a fact. The expert system searches through its fact base to find a pattern match to the premise of a rules, and if there is a match, the rule is executed. The execution of a rule might change the fact base, thus the state of the system, by either adding facts or deleting them.

Another knowledge representation paradigm that has become very popular in recent years is object oriented programming. Object oriented programming, a representation scheme that imitates the way objects in the real world exist, makes program development modular and easy to modify [9]. An

object is a collection of data with information on how the data behaves, and it reflects a concept, event, or object in the real world. An object called car would contain the following information: make, manufacturing date, engine type, gas milage, passenger room, cargo room, current state, etc. It would also contain information on how this object car behaves in different situations. For example when the gas pedal is pressed, the car moves.

The second component of a knowledge based system is the inference engine. The inference engine acts on the knowledge base to get to the final solution, and it is the basic reasoning process employed behind the problem solving. In rule based systems, two types of reasoning methods are usually used: forward chaining and backward chaining. Forward chaining, or otherwise known as data driven systems, start out with the initial or problem state and use the knowledge base to make deductions until the final solution is found. This would be the case where a detective investigating a case is starting out with a set of observations or clues and tries to figure out what happened.

On the other hand, backward chaining, or goal driven systems, start out with the final or desired state (goal) of the system and try to find observations or facts that will validate the goal. In this case, the detective might have a hypothesis or hunch and goes back looking for facts or clues that might prove or disprove his/her hunch. The reasoning method used by an expert system should best suit the problem at hand.

Within the last 40 years, many applications of knowledge based systems have been successfully implemented. One of the earliest and most popular of these, MYCINE, was a rule based, backward chaining, medical diagnostic expert system designed to assist doctors in diagnosing infectious diseases. The developers of MYCINE evaluated their program by sending the diagnosis of MYCINE and those from nine other doctors of ten patients with meningitis to infectious diseases specialists. The specialists, without knowing that MYCINE was a computer program gave it the highest rating for appropriate diagnosis [6].

Since then a number of expert system application have come into everyday use. Symbolic mathematical packages like MAPLE, grammar correction packages, equipment maintenance packages, flight simulation programs, and many other expert systems have emerged in fields from business to engineering [7].

In the electric power generating industry, expert systems are rapidly finding applications. In the 1992 Expert System Application conference sponsored by the Electric Power Research Institute (EPRI), several expert systems were reported to be in different stages of development or were already in real time use. Some of these expert systems were nuclear safety review advisor (SARA), power system restoration aid (CRAFT), load dispatch management assistant (CCLMA), boiler tube failure prevention management tools, estimator of fly ash particle size and composition in coal combustion

systems (ASHPERT), performance advisor for fossil fuel power plants (HEATXPRT), and evaluator of coal vendor financial viability in fuel purchasing contracts (CVFA). Some of these expert systems were designed for on-line use and monitor data continuously while others were designed for off line use. [8]

However, NO_x emission control through expert system applications still remains a relatively untouched area. NO_xSMART, an on-line advisor developed at TransAlta Utilities, is one known NO_x reduction expert system that is in the latter stages of development. NO_xSmart monitors on-line data and gives recommendations on furnace control parameter adjustments. After an initial trial period, NO_xSMART was reported to reduce NO_x levels of a coal fired boiler by 15% at full load conditions and 9% over all load levels [9].

Other related expert software analyze coal and monitor emissions in an effort to reduce NO_x emission levels. Fireside Advisor, a coal quality expert system for reducing emissions, was reported to be in the planning stages by Karta Technology Inc. at the 1992 EPRI Expert System Conference [8]. Besides advising on coal, Fireside was to be designed to generate unit specific test plans of boiler components. Coal Quality Expert (CQE) is an expert system software funded by the Department of Energy (DOE) and EPRI, and determines the most cost effective coal and emission control strategy for power plant performance and emission levels [10]. Central Station Compliance Advisory System (CSCACS) is an expert system developed by South Coast Air Quality Management District, a regional governmental agency that enforces air quality

regulations in four southern California counties. CSCACS monitors emission levels of 59 utility boilers [11].

Commercial expert system development tools with inference engine but without the knowledge base are used to develop most of the present day expert systems. These tools, referred to as shells, are programmed by the expert system developer with the domain knowledge of the problem. CLIPS, C Language Integrated Production System, an expert system shell written in C, was used for the development of the NO_x advisors in this project. CLIPS is a forward chaining, rule based system designed by a group in NASA laboratory [12].

A rule based system was chosen for the development of the NO_x advisors since the knowledge to be modeled was mostly heuristic or qualitative. A forward chaining (data driven) reasoning model was appropriate because of the enormous amount of initial available data from the plant data acquisition system. The diagnostic advisor could have been modeled satisfactorily using a backward chaining (goal driven) reasoning model but practical issues demanded the choice of one shell to develop both the optimization and diagnostic advisors. CLIPS was chosen in particular because of the number of people on campus that had worked with it extensively and could provide support. The fact that CLIPS was written in C made it a flexible tool that could be modified or integrated with other programs if necessary. CLIPS had an added advantage that it supported object-oriented and procedural programming

as knowledge representing paradigms. The objects in CLIPS 6.0 could also be pattern matched in rules.

On the other hand, CLIPS has two big drawbacks; it has a very crude user interface and its data acquisition and management system is primitive. Despite these disadvantages, CLIPS was still found to be the most attractive shell because of the immediately available support and its flexibility. These disadvantages are currently being overcome at the ERC by the development of a sophisticated user and data interface.

In the following sections, the knowledge base of both the optimization and diagnostic advisors, the method of knowledge representation and overall software architectures are discussed.

OPTIMIZATION ADVISOR

Introduction

The objective of the optimization advisor is to provide guidance to coal fired power plants in their effort to reduce NO_x emission levels. The projected users of this advisor are plant operators or engineers with extensive knowledge in boiler operation but no experience in NO_x emission control or conducting experiments. Thus, the advisor is intended to guide the user in each step of the optimization exercise and process the data. The user is expected to understand power plant terminology and be able to make judgments on issues concerned with boiler operation.

The optimization advisor's task is to reduce NO_x emissions through optimization of boiler operations. Past ERC projects have shown that significant reductions in NO_x can be achieved through boiler control optimization. During an initial six week test of optimum operating parameters at PEPCO's Potomac River Unit 4, average NO_x emission rates were reduced from baseline levels of 0.61 lb/MBtu to 0.37 lb/MBtu over the load range [13].

Boiler optimization is achieved through parametric testing of the variables known to affect NO_x formation. These variables can be identified from theoretical and experimental NO_x formation models as well as from directly testing the boiler under various conditions.

NO_x formation models classify NO_x by the source of nitrogen in the NO_x compound. "Thermal NO_x" is formed from nitrogen in the combustion air and

"fuel NO_x" is formed from nitrogen bound in the fuel. Theoretical models consider the thermodynamics and kinetics of the NO_x producing reactions to determine the variables that control NO_x formation.

The basic equilibrium reactions of thermal NO_x are:

$$O + N_2 \leftrightarrow NO + N \quad (1)$$

$$N + O_2 \leftrightarrow NO + O \quad (2)$$

$$NO + O \rightarrow NO_2 \quad (3)$$

Thermodynamic analysis of these reactions shows a rapid increase in the formation of NO and the decomposition of NO_2 into NO with increasing temperatures. Thus, the overall concentration of NO_x increases as the temperature of the flame is increased [14]. Experiments have also shown that the formation of NO increases dramatically in the flame region and this is assumed to be the result of increased breakdown of oxygen into free radicals which then react with nitrogen [15].

Analysis of the chemical kinetics of the thermal NO_x reactions has shown that the rate of NO formation is slower at lower temperatures. It has been determined that [1,14,15]:

$$\frac{d(NO)}{dt} = f(T, O_2, N_2) \quad (4)$$

where

T = temperature
t = time

Thus, the reaction time determines the concentration of NO. Besides time and temperature, the importance of the concentrations of N_2 and O_2 is also indicated. Since the combustion process favors the oxidation of carbon, O_2 needs to be present above stoichiometric ratios for NO_x to be formed.

While "fuel NO_x" is assumed to be a major contributor to NO_x in nitrogen rich fuels, the process of formation is not yet well understood. Experiments have shown that amount and rate of "fuel NO_x" formation are related to the type of fuel used and the percentage of nitrogen in the fuel. Experiments have also shown that the reaction rate for "fuel NO_x" is much faster than that of "thermal NO_x" [14]. The stoichiometry of the reaction mixture has been found to be an important parameter in the formation of "fuel NO_x" [15].

Previous experiments and analyses suggest methods of NO_x control. Since NO_x levels increase with increasing temperature, decreasing the flame temperature should reduce NO_x levels. Decreasing the amount of excess oxygen is also predicted to reduce NO_x formation. It has been shown that the hottest zone in the flame has the highest NO_x levels and as the combustion gases leave the flame region for cooler combustion zones, further NO_x formation is arrested. So, it has been recommended that having fuel rich flames (with a higher ratio of fuel to air) and later introducing more oxygen to complete the combustion would retard NO_x formation. This process is called staged combustion [1]. The last option for controlling NO_x is the proper choice of the type of fuel burnt.

Experiments on coal fired furnaces support the above recommendations. ERC projects on boiler optimization have shown that coal quality, concentration of oxygen available for combustion, temperature in the furnace and the degree of staged combustion can affect NO_x formation significantly [13,16,17].

Even though general trends on NO_x formation in furnaces have now been established, the optimization of the boiler parameters that control these trends has to be performed for each individual boiler. Optimization of boiler control parameters is conducted through systematic testing of relevant boiler components.

The optimization advisor is configured for full load operation at PEPCO's Potomac River Station Unit 4, which is a pulverized coal, tangentially fired, steam generating system. This unit has a full load capacity of 108 MW and operates at subcritical pressures with single reheat cycle. The parameters determined to control NO_x formation at this unit were economizer oxygen level, burner tilt angle, relative mill coal feeder RPMs and secondary air damper positions [13,16,17].

Other important parameters considered in the optimization advisor are the variables related to performance and safety. Ensuring safe boiler operation is an important consideration when conducting experiments. The optimization advisor monitors safety and system design limits. These parameters are CO level, windbox pressure, furnace oxygen level, steam temperatures, mill suction pressures, mill current, mill exit temperatures, and qualitative assessment of

furnace flame quality.

Some of the steps taken to reduce NO_x emission levels affect boiler efficiency negatively [17]. Loss on ignition, LOI, a parameter that measures the amount of unburned carbon left from combustion, is measured directly and indicates loss of performance, while heat rate is calculated from other measured quantities and also measures power plant performance. LOI is monitored constantly by the advisor while heat rate is used to make final decisions on optimum operating conditions.

The advisor also scrutinizes parameters that are under regulatory limits. Opacity, which is a measure of the visibility of stack emissions, is directly affected by furnace conditions, and as such, is monitored by the advisor during testing.

CLIPS 6.0 was used to develop a model that would guide the user through a series of experiments that provide information on the specific relation of NO_x to boiler operating conditions and heat rate. The model is based on knowledge of the parameters mentioned above, and experimental procedures. The model consists of rules that conduct experiments on the boiler control parameters that affect NO_x formation, and monitor safety and performance parameters. The model uses external curve fit programs to process data for internal decision making and feeds all the data generated from the experiments into a neural network-optimization package for final analysis [2]. This package makes the decision on the optimum operating parameters for the boiler.

Description of the parameters mentioned above, known effects of control parameters on NO_x levels, and safety and performance variables, experimental procedures used in the model, and software architecture are discussed in the following sections.

Knowledge Base

At Potomac River Unit 4, the parameters determined to control NO_x formation are economizer oxygen level, burner tilt angle, mill coal feeder RPMs, and secondary air damper positions.

The economizer, situated downstream from the furnace, is a heat exchanger that extracts heat from the flue gas and heats incoming feed water to the boiler (see Figure 2)[18]. Oxygen sensors located in the economizer measure the concentration of oxygen in the flue gas, which gives an indication of the amount of excess oxygen in the furnace.

Potomac River Unit 4 has a direct fired system and uses four Raymond bowl suction mills that pulverize coal to be fed into the balanced draft furnace (see Figure 3) [21]. The capacity of the mills is measured in terms of revolutions per minute (RPM) of the feeders that bring coal to the mills, and at Potomac River Unit 4, each mill can handle 22,000 lbs/hr of coal [21].

The pulverized coal is delivered to four levels of burners in the furnace via primary air induced by the exhauster (see Figure 4). The tilt mechanism of the burners is designed to control steam temperatures. Burner tilt angle is raised or lowered to adjust the location of the fire ball and the angle ranges from -30° to 30° at Potomac River Unit 4.

The secondary air dampers control the amount of secondary air in the windbox. The secondary air is the balance of air needed in addition to the primary air delivering the coal. The secondary air dampers have incremental

stages of openness. At Potomac River, "one" indicates the most closed state, and "five" indicates the most open state. There are two types of secondary air dampers: fuel air and auxiliary air dampers. Fuel air dampers control the air flow in a narrow passage surrounding the primary air coal nozzle. The auxiliary air dampers control air flow to nozzles located between burners (see Figure 2). Typically, there are the same number of fuel air dampers as burner levels, and one more level of auxiliary air dampers than burners. Accordingly, at Potomac River Unit 4, there are four fuel air dampers and five auxiliary air dampers in each corner of the boiler. Varying the damper positions of the different secondary air dampers controls the air distribution along the furnace, air velocities, and the locations of fuel ignition points. [1]

Experimentation with the control parameters above creates different boiler conditions which in turn affect other variables of interest. In relation to the advisor, these parameters are referred to as safety and performance parameters, and have maximum and/or minimum design, regulatory, or performance limits. These parameters were carefully considered in model design.

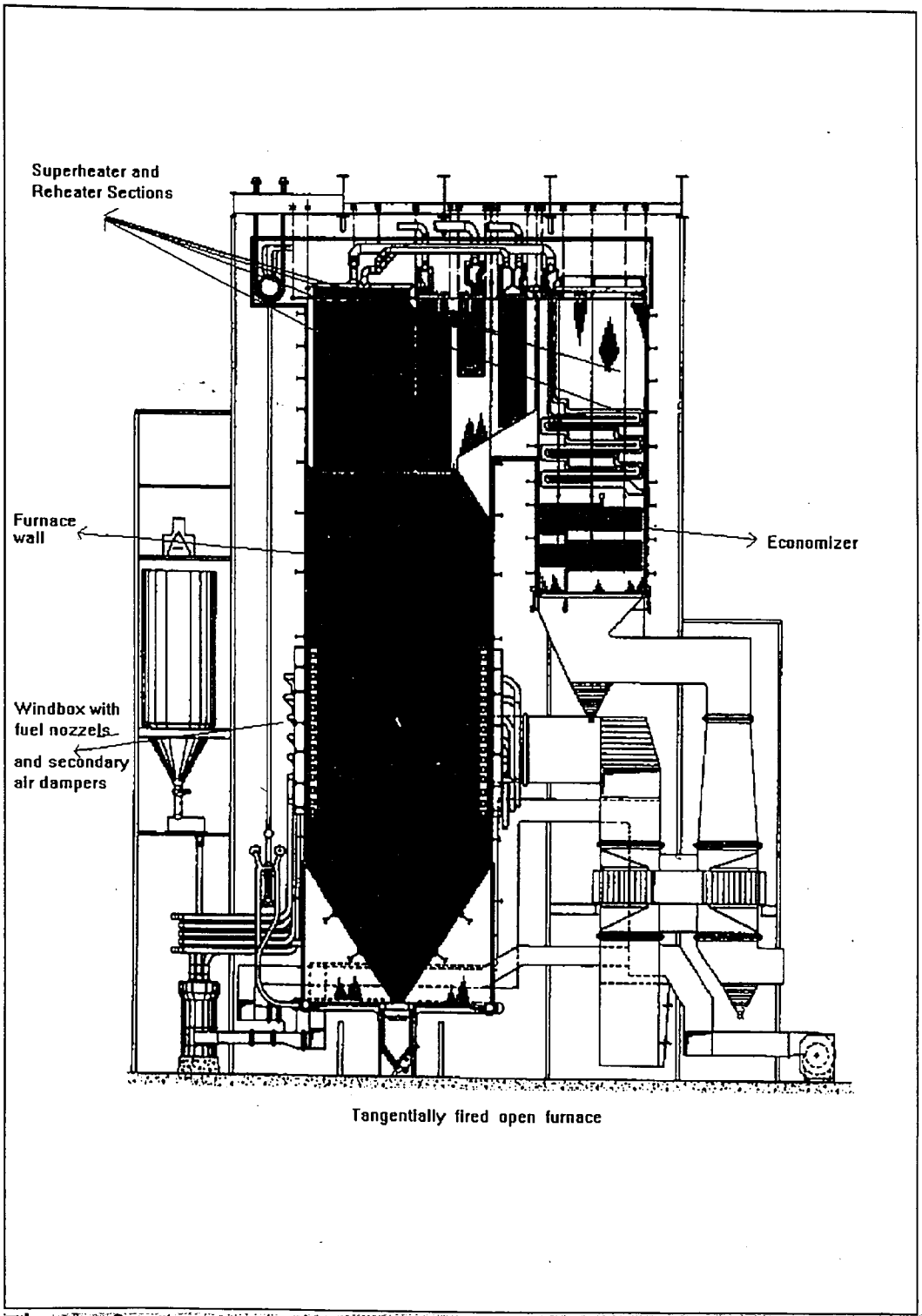

Figure 2 Tangentially Fired Furnace [1]

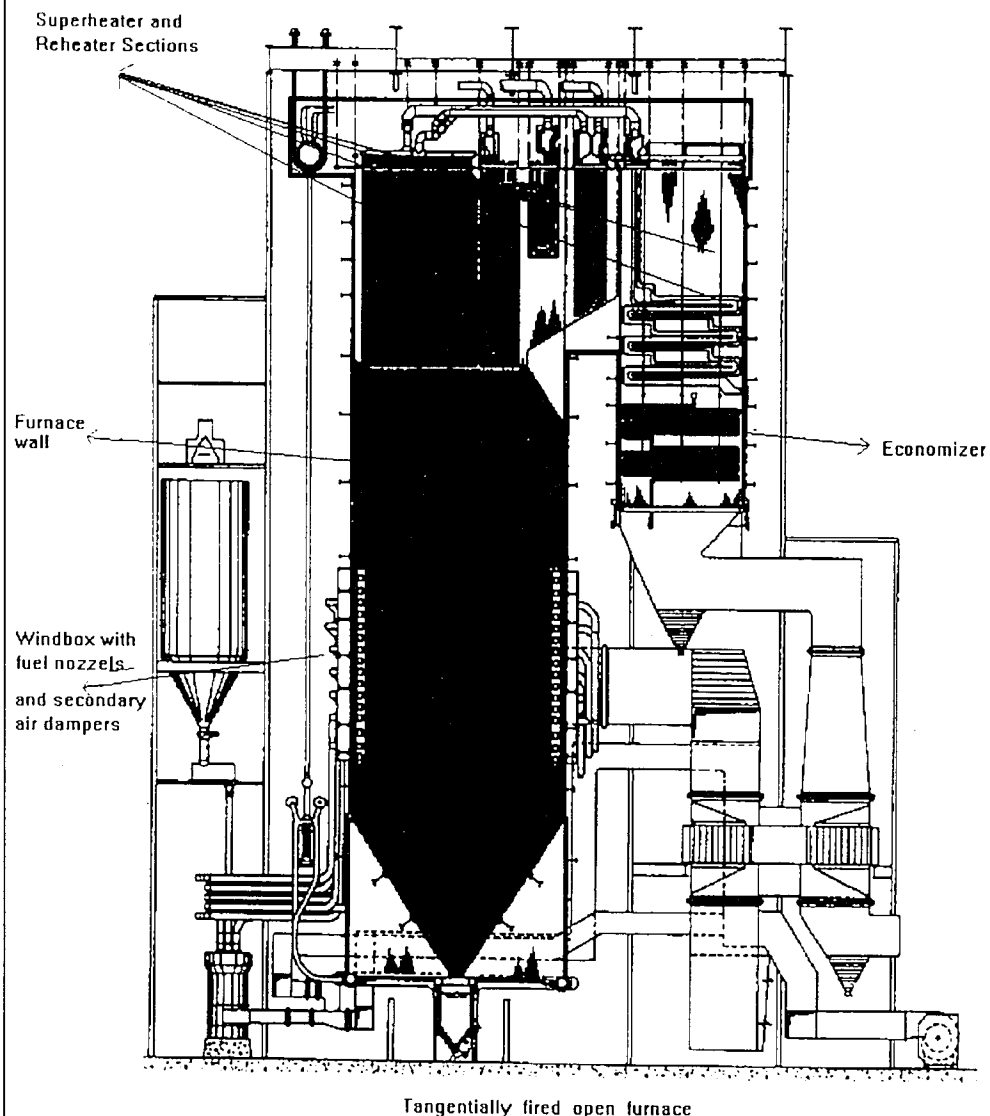

Figure 2 Tangentially Fired Furnace [1]

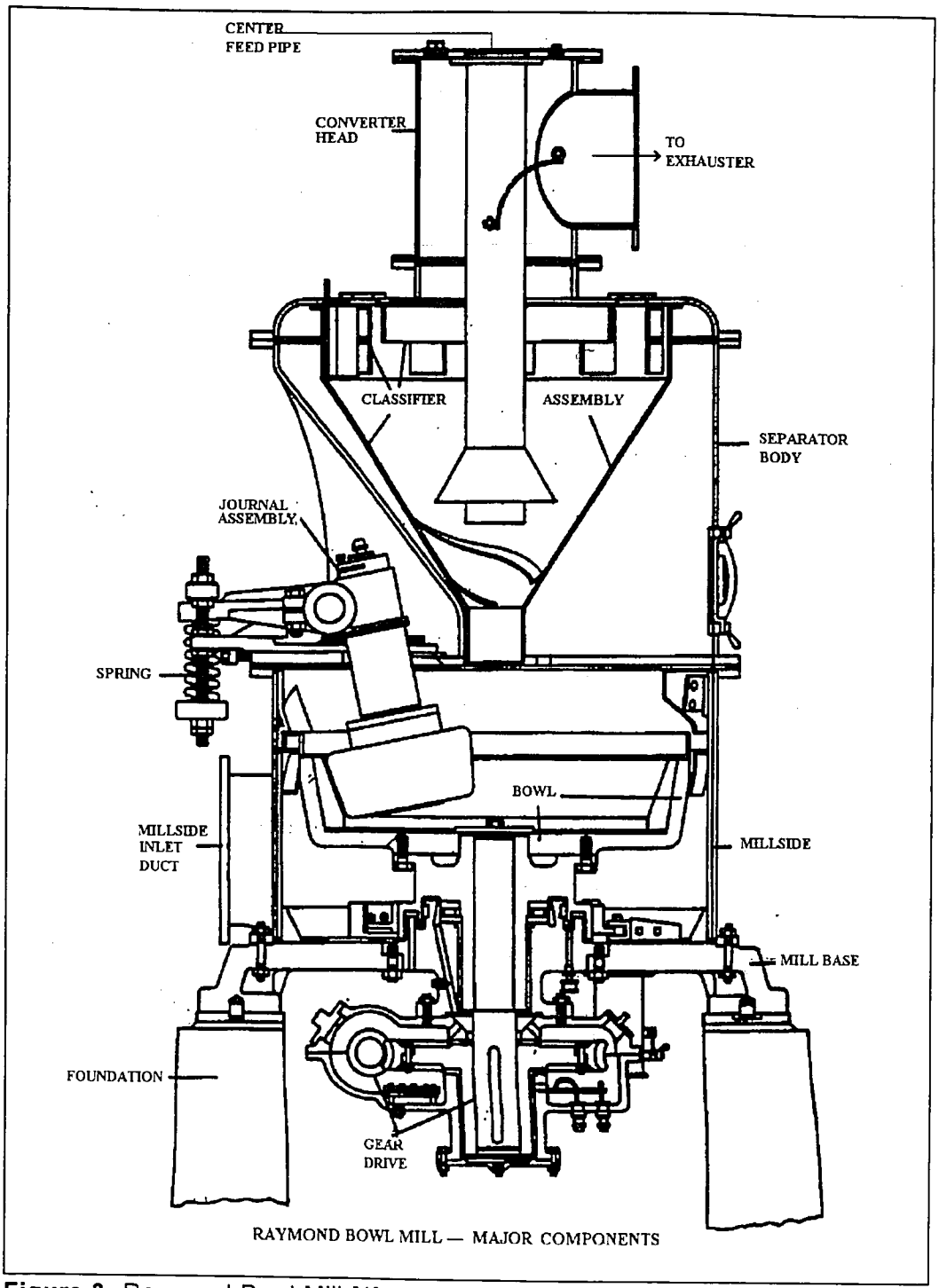

Figure 3 Raymond Bowl Mill [1]

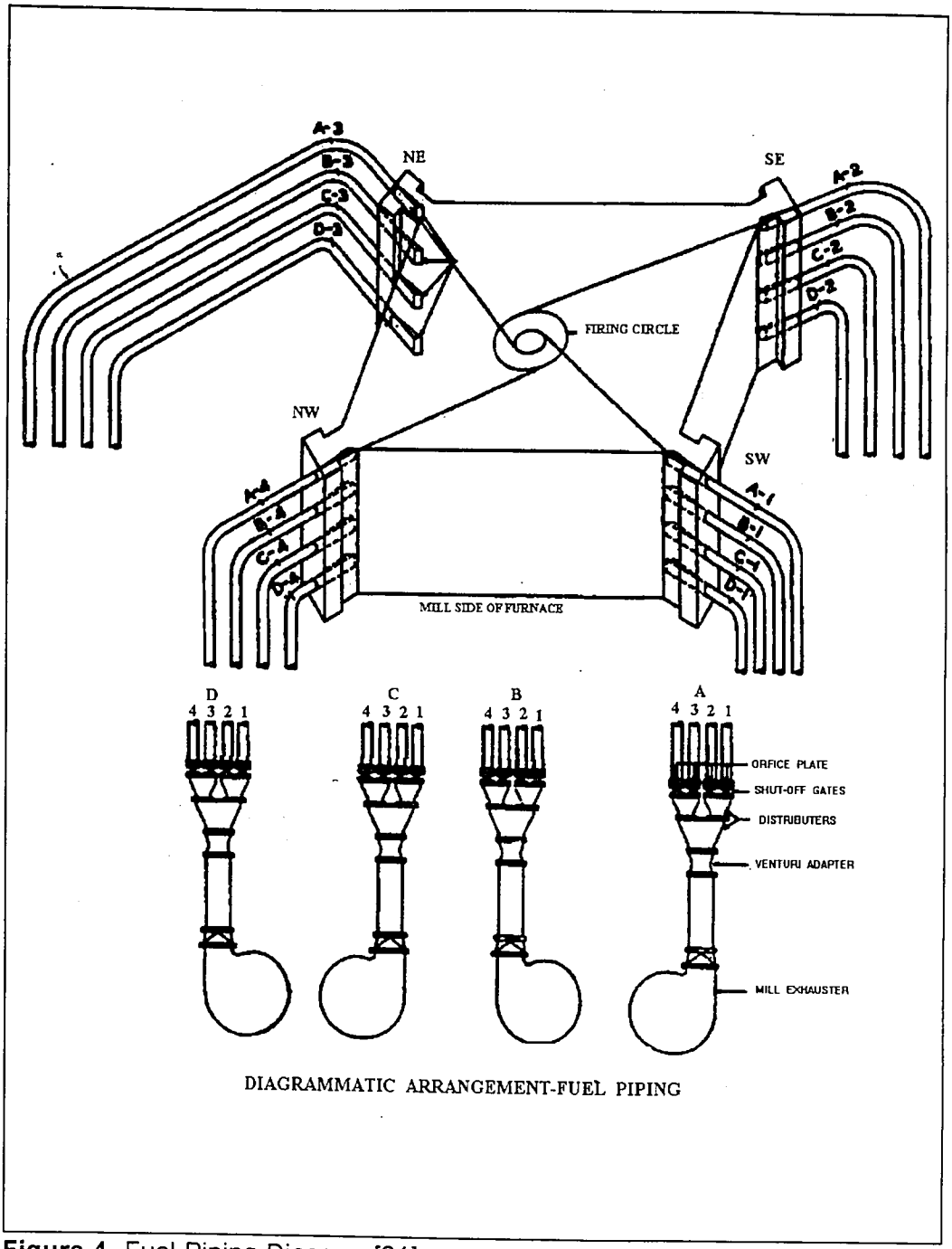

Figure 4 Fuel Piping Diagram [21]

Economizer Oxygen

Typically, furnace oxygen is maintained at above stoichiometric conditions to ensure complete combustion and safe boiler operation, and this excess oxygen is usually measured in the flue gas at the economizer exit. NO_x has been shown to decrease with decreasing excess oxygen levels [13,16,17]. Figure 5 shows parametric data from Potomac River Unit 4 correlating NO_x levels to economizer oxygen levels for different burner tilt angles. At Potomac River, changes in O_2 had the greatest effect on NO_x levels and during parametric tests accounted for changes in NO_x of 0.2 to 0.25 lb/MBtu. Thus, lowering the economizer oxygen level as much as possible is a key part of the strategy used to reduce NO_x emission levels.

The minimum excess O_2 level possible is determined by safety and performance parameters. Lowering economizer oxygen lowers furnace oxygen, which, in turn, affects the amount of unburned carbon, CO, opacity, steam temperatures, windbox pressures, and finally furnace flame quality. It is known through experimentation and plant experience, that lowering the excess oxygen level at Potomac River Unit 4:

- yields higher LOI (see Figure 6);
- does not affect CO, until a certain point when CO increases rapidly (see Figure 7) [20];
- produces lower steam temperatures;
- lowers windbox pressures;

• produces poor quality flames.

Burner Tilt Angle

Burner tilt angles are strongly correlated to NO_x levels and a parabolic behavior was observed in most cases (see Figure 8). At full load at Potomac River, burner tilt angles had the second highest effect on NO_x levels, after O_2, and during parametric tests, these accounted for NO_x reductions of up to 0.09 lb/MBtu [17]. At baseline conditions and during parametric tests at Potomac River, minimum NO_x levels occurred with burners near the horizontal positions, with tilt angles between 0 and 5 degrees [13,16,17]. Thus, the strategy for reducing NO_x emissions involves exploring the tilt angle range to isolate the region of minimum NO_x emissions. Since burner tilt angle is also a mechanism for controlling steam temperatures, manipulating the burner tilt angles affects heat rate, which is a strong function of steam temperatures (see Figure 9).

Mills

The loading pattern among the mills providing coal to the different burner levels has been shown to affect NO_x formation [13,16]. Biasing coal to the bottom burners creates a staged combustion effect. To represent the loading pattern among the mills, a mill bias parameter, β was developed (see Eqn 5) [17]. β ranges from -1 to 1, with positive values indicating bias towards the top mills, and negative values indicating coal biased towards the bottom

$$\beta = \frac{(\Omega_A - \Omega_D) + 1/3 \cdot (\Omega_B - \Omega_C)}{\sum_I \Omega_i} \quad (5)$$

where

Ω_i = *motor speed of the* i^{th} *feeder* (*RPM*)

mills. Figure 10 shows the reduction of NO_x with decreasing mill bias at Potomac River Unit 4. At this unit, during parametric testing at full load conditions, a NO_x reduction of 0.03 lb/MBtu was achieved. To reduce NO_x emissions, the strategy should be to aim towards the most negative β value. Even though at full load, this could be achieved at some units by unloading the top burners completely, at Potomac River all four mills are needed to achieve full load levels. In fact, in some circumstances which depend on coal quality (moisture level, ash level, HHV, HGI, etc.), full capacity of all four mills is needed, and no bias is be possible.

Biasing mills can affect the following parameters. Biasing coal flow to towards the bottom mills lowers steam temperatures, which in turn raises heat rate. When biasing away from the top mill, in order to make up for the load difference, extra coal is fed to the bottom mills. Feeding more coal to a mill increases the mill current, reduces available suction in the mill, and eventually yields reduced mill exit temperatures.

Secondary Air Dampers

Secondary air dampers can be manipulated to create staged combustion by adjusting the vertical distribution of fuel air and auxiliary air. In the case of

auxiliary air dampers, biasing air to the top of the furnace has been shown to reduce NO_x emission levels [13,16,18]. To simplify the hundreds of different combinations of dampers to a variable that represents the air bias along the furnace, an auxiliary air bias parameter, α, was developed (see Eqn. 6) [17].

$$\alpha = \frac{2\,(Aux_1 - Aux_9) + (Aux_3 - Aux_7)}{12} \qquad (6)$$

where

Aux_i = i^{th} *row auxiliary air damper postion*

The parameter α ranges from 0 to 1, with the higher numbers indicating air bias to the top air registers. At Potomac River, during parametric testing, NO_x was reduced by as much as 0.04 lb/MBtu in some ranges of α while no effect was seen at other ranges (see Figure 11).

While the strategy for reducing NO_x is to increase α, different combinations of dampers that give the same α can affect NO_x and other safety parameters differently and, thus, need to be explored. To reduce the number of combinations of dampers explored per one α to those that are more likely to give safe boiler operations, a few rules of thumb were developed by ERC engineers:

- when changing a damper position at one level, dampers should be manipulated only one position at a time;
- the sum of all the auxiliary air damper positions should be between 14 and 18 (for full load conditions at Potomac River unit 4);
- an auxiliary damper should be at the same or greater damper position

as the one a level below it;

- at full load, only one auxiliary air damper can be at the most closed position (which is 1 at Potomac River Unit 4);

Manipulating fuel air dampers varies the fuel to air ratio in the region near the burners, and experimental models (refer to the Introduction of the optimization section) have shown that this affects the formation of NO_x. Thus, closing down on the fuel air dampers should create a fuel rich flame and reduce NO_x formation. It was noted in previous testing of fuel air dampers, that the magnitude of their effect increases with decreasing load levels. At full load testing of Potomac River Unit 4, manipulating the fuel air dampers produced little or no effect on NO_x [17]. Even though major reduction of NO_x by this mechanism is not expected at full load conditions, fuel air dampers are optimized when fine tuning of NO_x levels is needed.

The extent of manipulation of the secondary air dampers is restricted by safety limits. Closing down on dampers can reduce furnace flame quality, and closing down on fuel air dampers shortens ignition points, which can bring the flame too close to the burners. Closing down on dampers increases windbox pressure, while opening dampers decreases it. After maximum auxiliary air bias has been achieved, manipulating fuel air dampers can create more room for further experimentation with auxiliary air dampers. Damper positions can also affect steam temperatures which then affects heat rate, but no trend has been noted so far.

Uncontrolled Parameters

Even though the above parameters have been observed to affect NO_x, and can be controlled in experiments, there are uncontrolled variables which might as equally affect NO_x. For instance, changes in coal quality can affect NO_x significantly (N_2 and volatile matter content of coal, moisture level, HGI, etc.).

Boiler cleanliness is another parameter that has so far been uncontrolled, but for which models are in various stages of development. Slag build-up on boiler tube walls increases steam temperatures and has been shown to increase NO_x emission levels significantly (see Figure 12) [17]. Soot blowing to remove slag from boiler tube walls decreases steam temperatures and NO_x emission levels. Since a good correlation between soot blowing and NO_x formation has not yet been developed, and slag build up increases with time, the advisor employs experimental procedures that correct for the possible effect of boiler cleanliness when testing the control variables discussed above (see Section on experimental procedures).

Experimental Procedures

In designing the experiments conducted by the advisor, little or no interaction of control variables with each other was assumed. This led to a

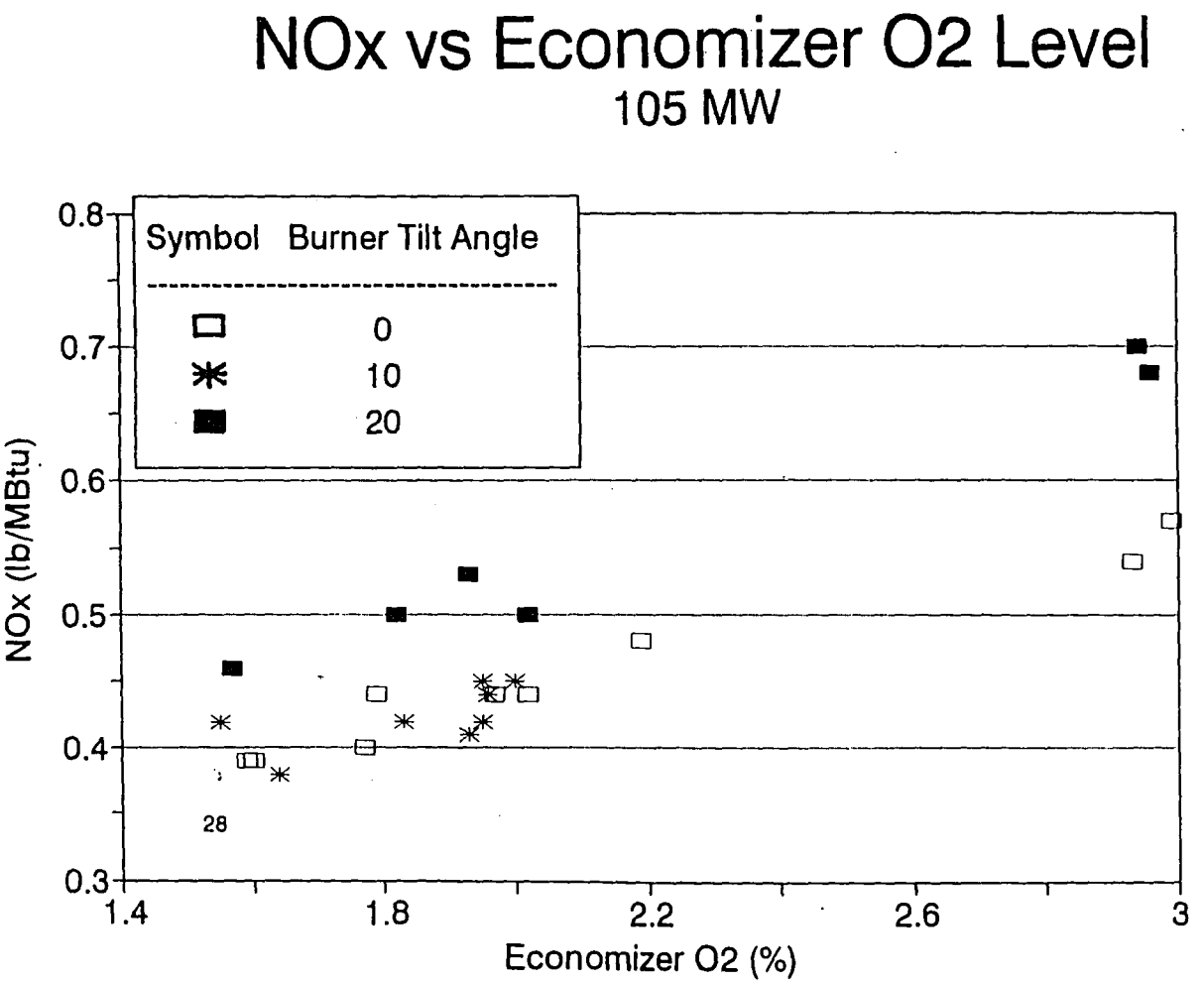
NOx vs Economizer O2 Level
105 MW
Symbol Burner Tilt Angle
0
10
20
NOx (lb/MBtu)
0.8
0.7
0.6
0.5
0.4
0.3
28
1.4
1.8
2.2
2.6
3
Economizer O2 (%)

Figure 5

LOI vs. Economizer O_2

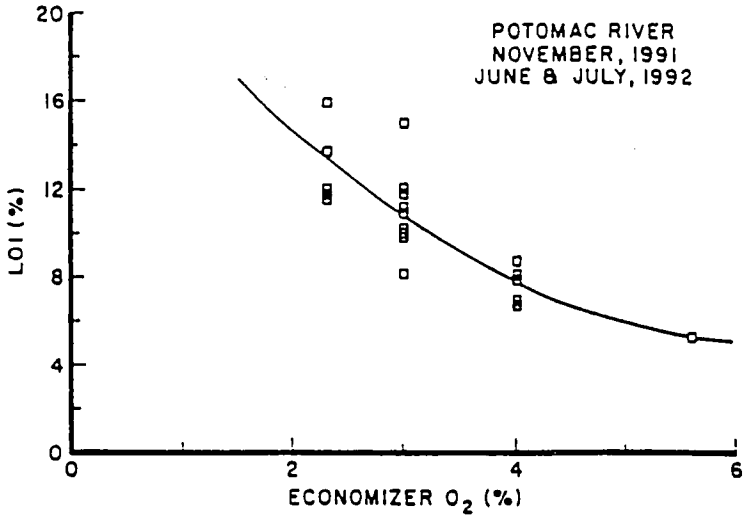

Figure 6

CO vs Economizer O_2

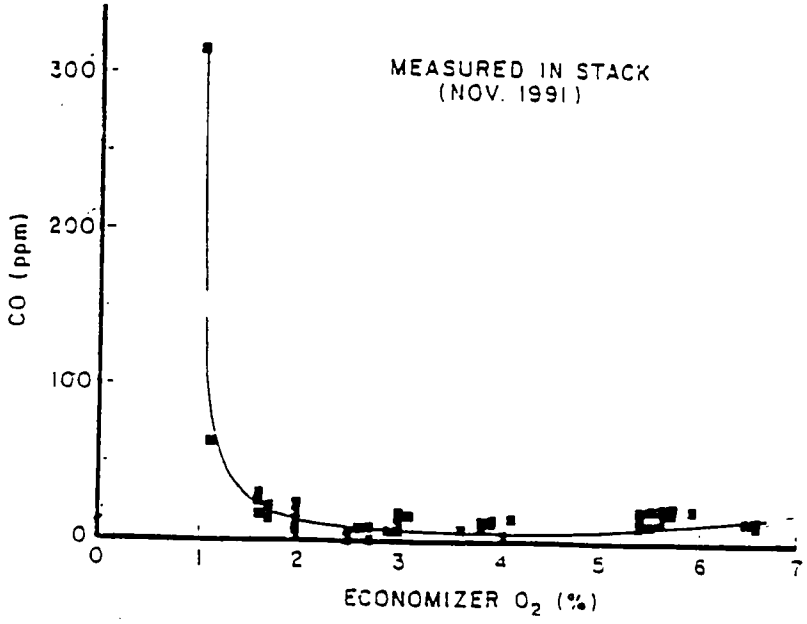

Figure 7

NO_x vs Burner Tilt Angle

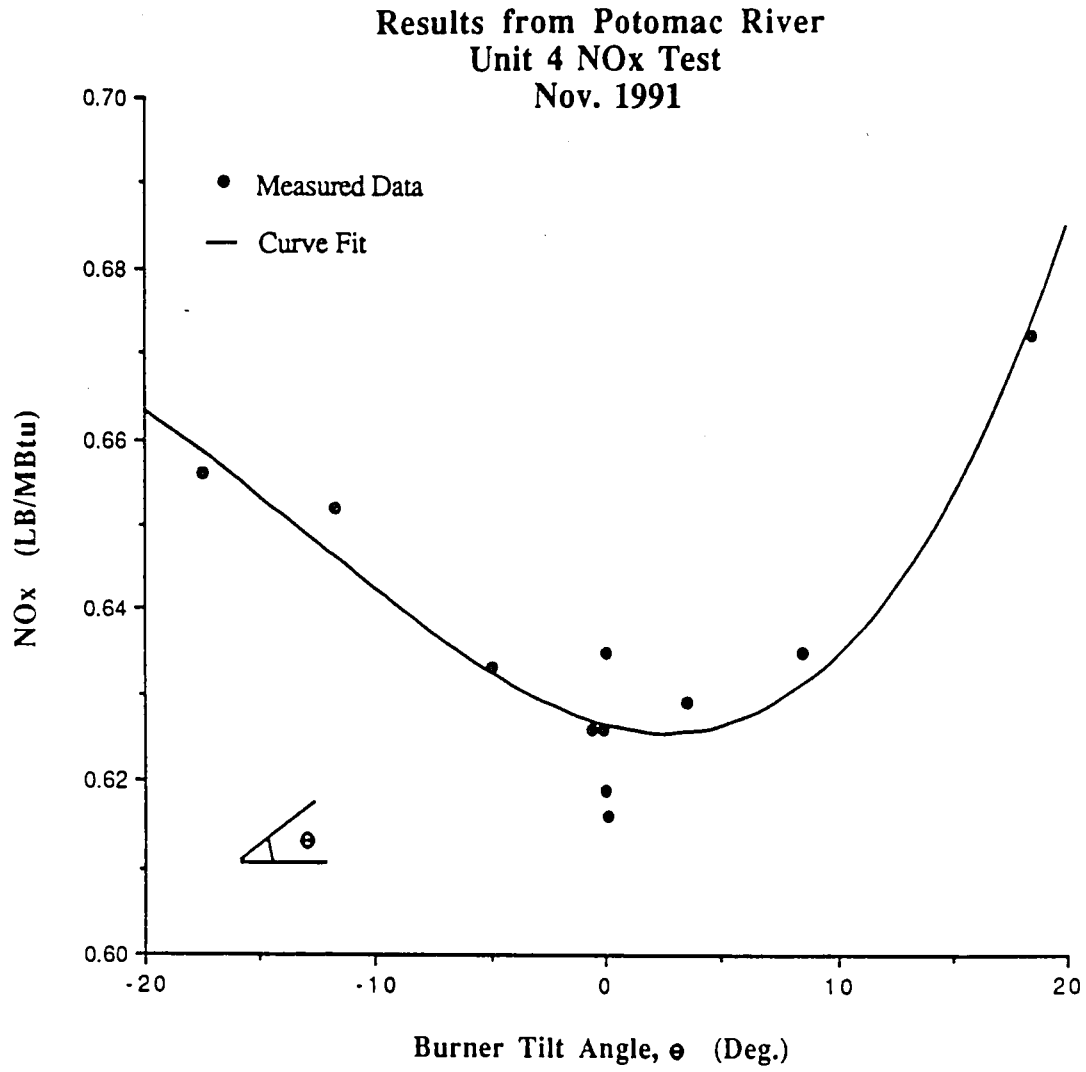

Figure 8

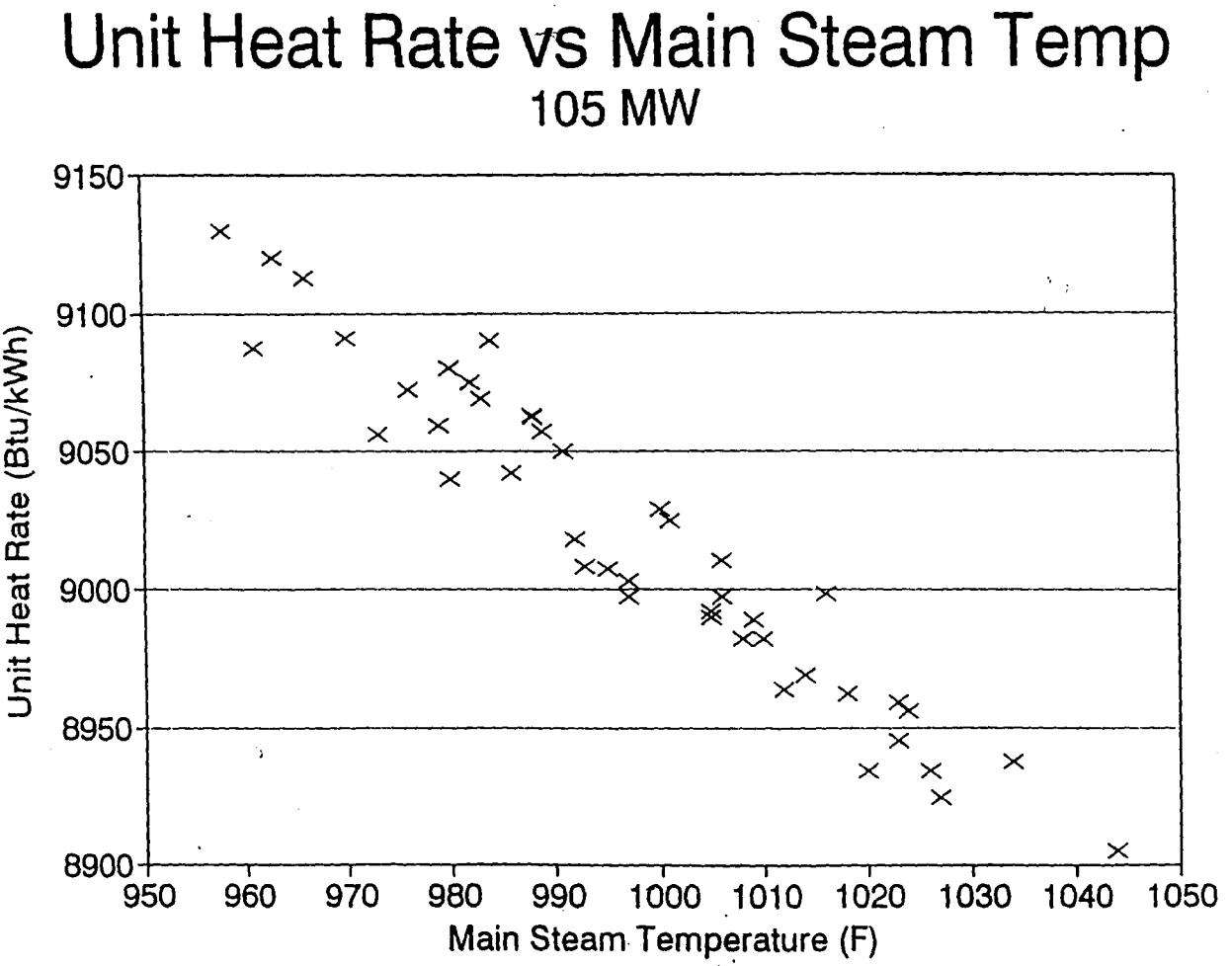
Unit Heat Rate vs Main Steam Temp
105 MW
Unit Heat Rate (Btu/kWh)
9150
9100
9050
9000
8950
8900
950
960
970
980
990
1000
1010
1020
1030
1040
1050
Main Steam Temperature (F)

Figure 9

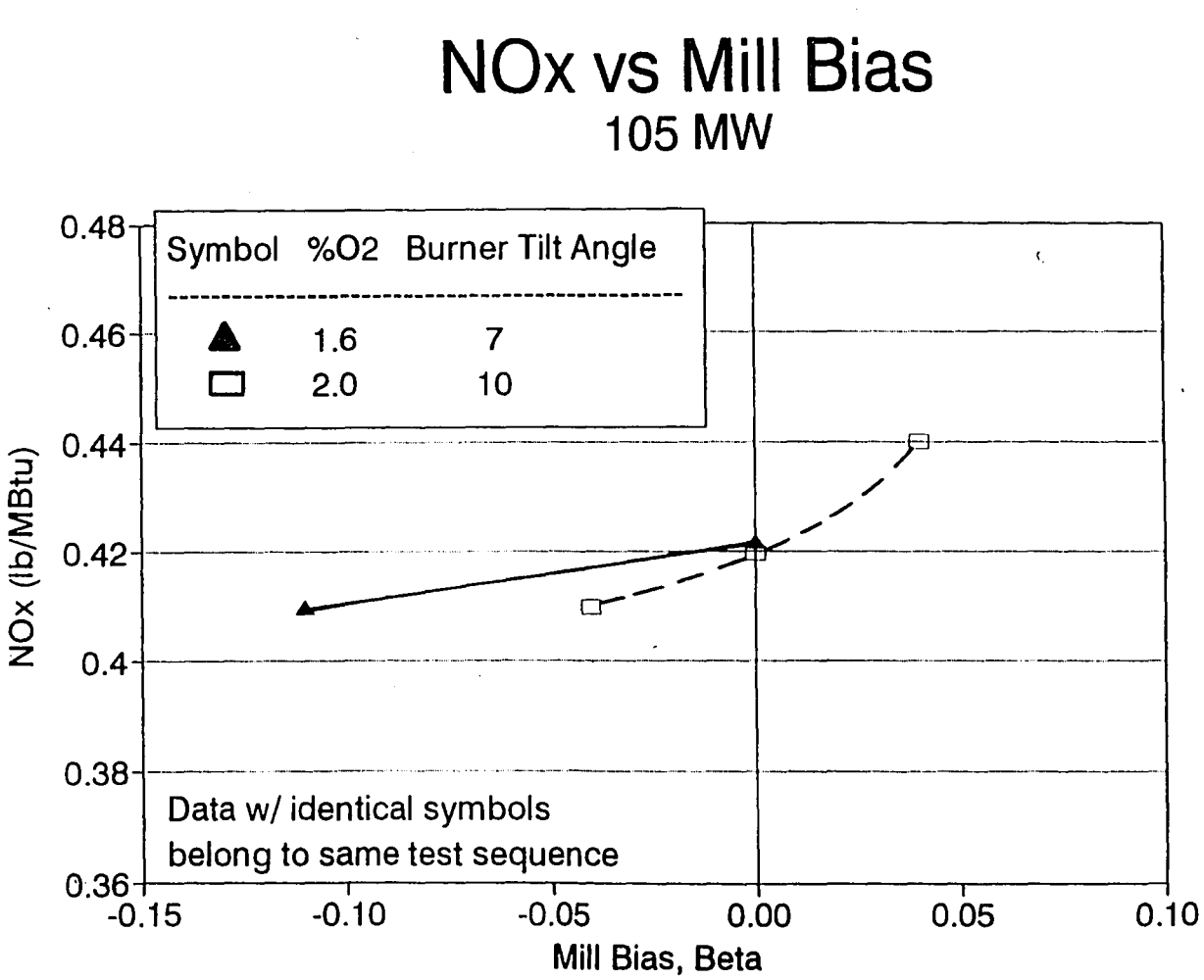
NOx vs Mill Bias
105 MW
Symbol %O2 Burner Tilt Angle
1.6 7
2.0 10
Data w/ identical symbols
belong to same test sequence
NOx (lb/MBtu)
0.48
0.46
0.44
0.42
0.4
0.38
0.36
-0.15
-0.10
-0.05
0.00
0.05
0.10
Mill Bias, Beta

Figure 10

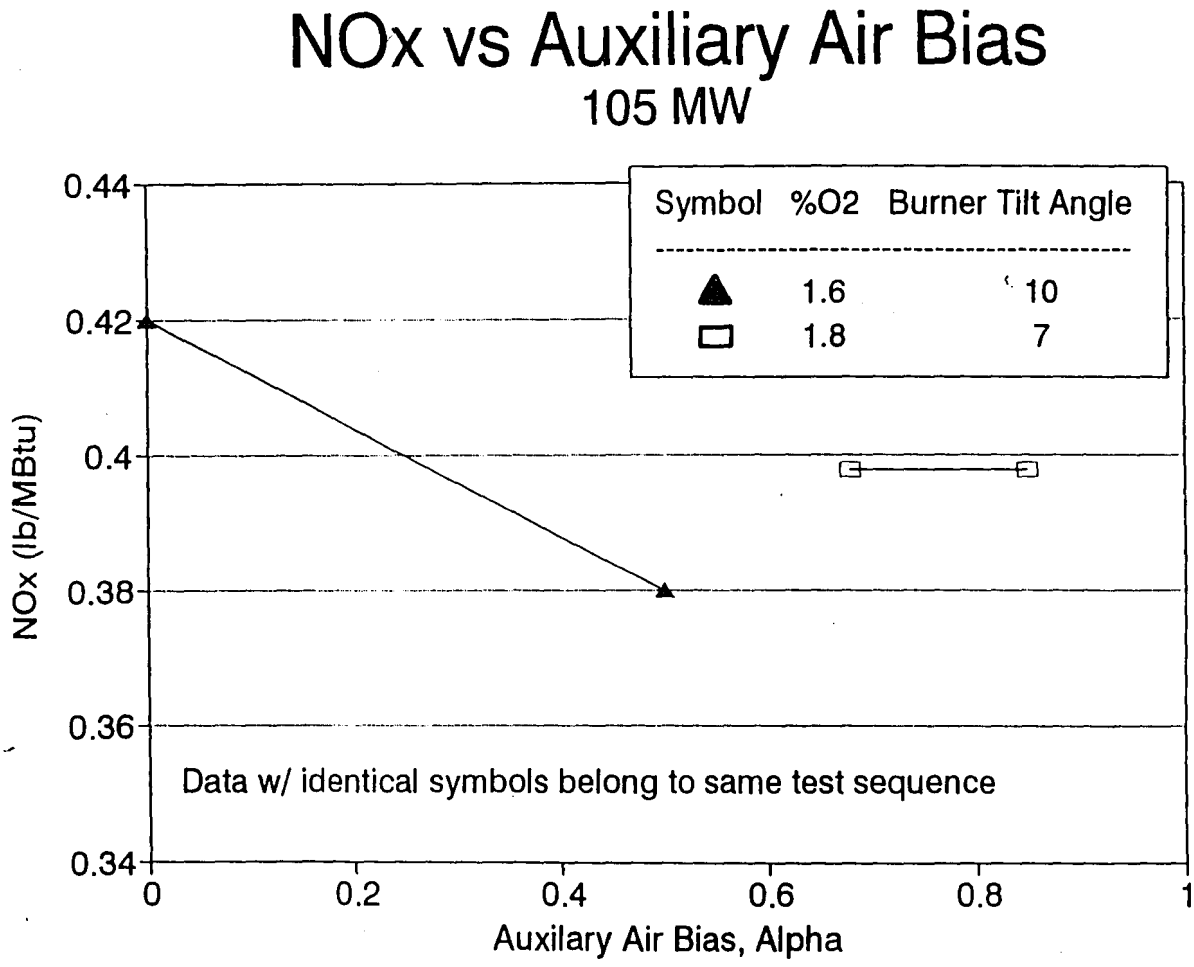
NOx vs Auxiliary Air Bias
105 MW
Symbol %O2 Burner Tilt Angle
1.6 10
1.8 7
0.44
0.42
0.4
0.38
0.36
0.34
NOx (lb/MBtu)
Data w/ identical symbols belong to same test sequence
0
0.2
0.4
0.6
0.8
1
Auxilary Air Bias, Alpha

Figure 11

Effect Of Boiler Slagging on NOx

105 MW

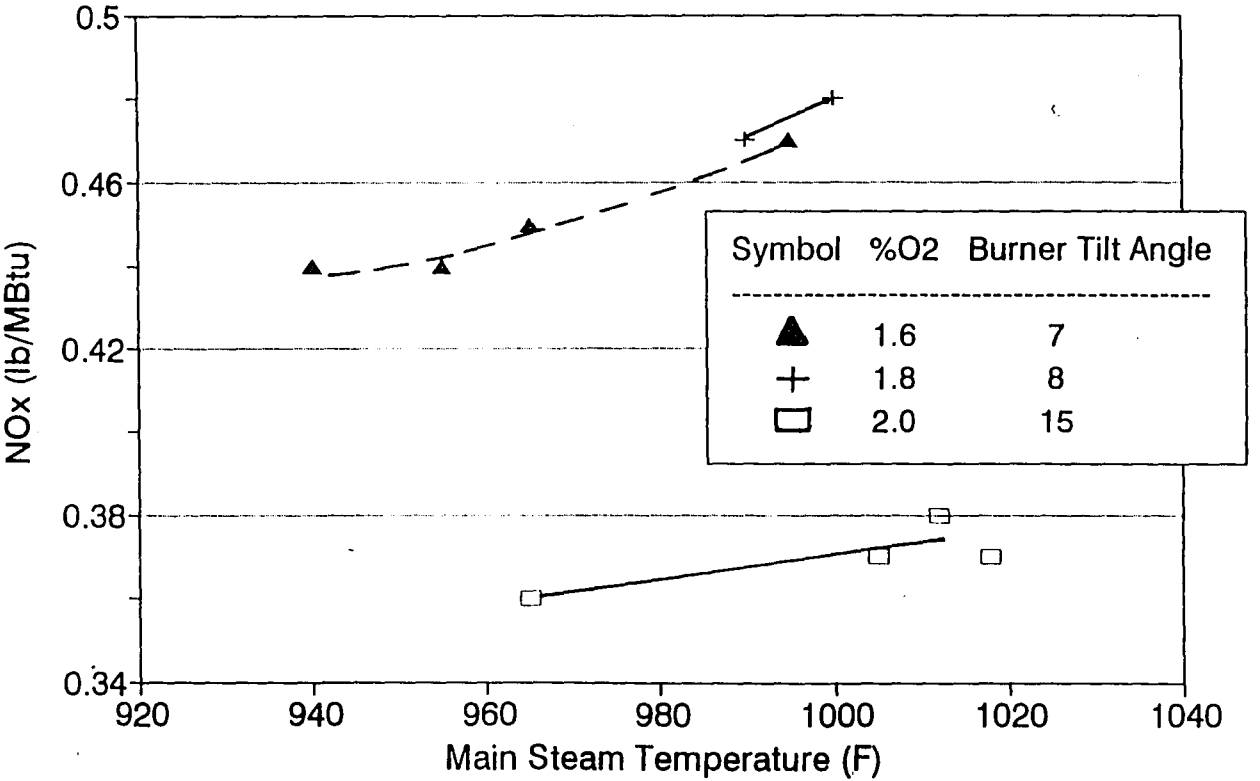

Figure 12

"parametric testing" strategy to be utilized in the advisor. In parametric testing, all other control variables are held constant, while the relationship of one variable with NO_x is explored. The sequence of parametric testing of the control parameters is determined by the magnitude of their effect on NO_x. Thus, at full load, O_2 is tested first, then burner tilt angle, then mills, and finally secondary air dampers. At the end of each parametric testing series, the advisor determines the control setting that produces the desired NO_x level, and that parameter is fixed at that setting for the testing of the rest of the control parameters.

In cases where a slight interaction between control parameters has been observed, and/or the goal of the experimentation was to optimize heat rate subject to a constraint on NO_x (see Section on software description), a factorial design of 2^3 or 2^2 was employed . In this type of experimental design, three distinct points of the two variables are selected, and all possible combinations of these points are tested. In this advisor, a factorial design was used on O_2 and burner tilt angle, and on burner tilt angle and auxiliary air bias. Besides bringing out any coupling effect of the two parameters on NO_x, this experimental procedure is intended to provide the necessary information when simultaneously optimizing two variables like NO_x and heat rate.

For the purposes of recommending test points, continuous functions with a maximum of one minima or maxima (unimodal) of the control variables with NO_x were assumed. NO_x was assumed to be a continuous linear function of O_2,

α, and β, and a parabolic function of burner tilt angle. This assumption is used only in conducting the experiments, since the final data analysis is done by a neural network package that learns the real relationship of the dependent variables with the control parameters.

When making decisions, the advisor curve fits any existing data to the functions mentioned above. Curve fits are used in the advisor to correct for any scatter in data and to predict real trends. Curve fitting also addresses uncontrolled variable effects in a properly designed data set. To minimize the effect of changes in boiler cleanliness over the time period of a test, initial test points are repeated at the end of a test series.

Since time and cost are primary concerns for utilities, minimizing experimentation time was a major consideration in the advisor design. Testing is aimed in the direction of minimum NO_x as suggested by the knowledge of the control parameters described in the previous sections. Thus, testing is conducted towards decreasing O_2, decreasing β, increasing α, and decreasing fuel air damper settings. In this strategy, the minimum NO_x setting is reached relatively rapidly, but the advantage of having a large data set that would capture the effects of uncontrolled variables is lost. Since NO_x has exhibited a parabolic relation to burner tilt angle, when the burners are near horizontal, the direction of burner tilt angles towards decreasing NO_x is not known. Instead, a number of points over the range of angles are tested to find the minimum.

The strategy of testing towards minimum NO_x in one direction also

simplifies the task of monitoring safety and performance parameters. Assuming continuous unimodal functions of safety parameters with control variables, if a safety limit is violated, the advisor makes the decision to discontinue testing of that control variable. For example, when testing excess oxygen level by decreasing the setting steadily, if the CO limit is violated, the advisor discontinues the oxygen test series after obtaining the replicate data that correct for boiler cleanliness.

The advisor also ensures that sufficient data for proper data analysis are gathered. The neural network package needs at least seven to eight data points per independent variable. If, because of limit violations, the range of a particular control parameter is narrow, replicate test points are obtained.

Various features are built into the experimental procedures to ensure safety of boiler operation during testing. In the independent testing of control variables, conservatively small increments are taken. This helps to assure that an unsafe boiler condition is not created unexpectedly. For example, in the case of oxygen, the minimum significant increment is used. When testing fuel air damper settings, which can create dangerous furnace conditions, the user is asked to close one damper at a time and make qualitative judgments on furnace conditions before the next test is conducted.

In the next section, the details of the advisor architecture, the different optimization goals it can address, and other relevant software features are discussed.

Software Description

The advisor is designed to operate under either of two distinct optimization goals:

- find the control settings that give the absolute minimum NO_x, subject only to safe boiler operating conditions;
- find the control settings that give minimum heat rate constrained to a target NO_x value, with safety still being an important consideration.

When operating under the goal of obtaining the minimum NO_x level possible, the advisor recommends only test points that are predicted to achieve low NO_x levels. On the other hand, when optimizing heat rate subject to a constraint on NO_x, additional information on the relation of NO_x and heat rate with each other and with the control variables are needed, and the advisor recommends additional experiments towards this purpose. Thus, the factorial experiments on O_2 and burner tilt angle, and burner tilt angle and auxiliary air bias are performed only when the optimization goal is to find minimum heat rate with a constraint on NO_x.

The advisor was also designed to accommodate different user requirements. While the user has the option to optimize a boiler from a completely unoptimized baseline condition by adjusting all the relevant boiler control parameters, he/she can also choose to fine tune a boiler that has been optimized previously. For instance, major variables such as oxygen and burner tilt angle might have already been optimized, and now the user could be

seeking a further reduction in NO_x by optimizing mills and secondary air dampers. Along the same lines, the advisor is designed to accommodate boilers that might have one or more of the control parameters inoperational. For example, the burners at a particular boiler might be locked in a certain tilt position and can't be adjusted, or the secondary air damper mechanisms might, at the moment, be broken. Thus, the user is given the option of choosing the parameters that he/she wishes to optimize.

Data acquisition is an essential ingredient in the optimization advisor. Data and information gathered before testing begins, and after each test is performed, are used to determine what the next step in the optimization exercise should be. The information required by the advisor can be both quantitative and qualitative. The quantitative data are gathered either from the data acquisition system, or if not available there, are obtained directly from the user. At Potomac River Unit 4, the online plant data acquisition system, Plant Monitoring Workstation (PMW), can provide most of the data required by the advisor after a test is run. The quantitative data from a particular test that are not available online, qualitative data that require judgment from the user, and other system information required by the advisor at the initiation of the optimization exercise, are obtained directly from the user. To aid with these tasks, a user interface and a data communication system for use with PMW are currently under development at the ERC.

Inside the advisor, information is stored either in the form of facts[1] or objects[2] . Rules[3] use pattern matching[4] to manipulate these objects and facts. "Goal of the optimization is minimum NO_x" or "Minimum burner tilt angle is -30°" are examples of facts that might exist at any point in the advisor. All information, except those directly related to test data, are stored as facts. Objects are not used to their full capacity in the advisor. They are used merely for data storage and act as records. An object called ***oper-data***[5] stores the following information on each test point:

- test number
- date
- beginning and end times
- economizer oxygen level (%)
- burner tilt angle (degrees)
- mill loadings (RPM)

---

[1]A fact is a statement that tells what is known to be true about a system (see Section on expert systems for further explanation.)

[2] An 'object' is a collection of data that reflects a concept, event or object in the real world (see Section on expert system for further explanations.)

[3] Rules are knowledge representation schemes in expert systems that use an "If...then...else..." form (see Section on expert systems for further explanation.)

[4]Pattern matching refers to the action performed by rule based programs when they search through all the facts in the system to find the ones that match the premise of a rule. This is simply a matter of comparing the symbols (letters) contained in the fact statement with those in the premise of the rule (refer to Section on expert systems).

[5]***Names associated with any part of the advisor code are shown in this font.***

- mill bias
- auxiliary air dampers
- auxiliary air bias
- fuel air dampers
- furnace oxygen level (%)
- windbox pressure (in/H_2O)
- main steam temperature (F)
- reheat steam temperature (F)
- opacity (%)
- CO (ppm)
- unburned carbon or loss on ignition (LOI) (%)
- mill currents (amps)
- mill suction pressures (in/H_2O)
- mill exit temperatures (F)
- NO_x (lb/MBtu)
- furnace quality (qualitative)
- ignition point (qualitative).

Besides rules, internal functions and external programs are used to process data within the advisor. Functions defined internally in the advisor calculate mill bias, auxiliary air bias, roots of quadratic equations, and perform other routine tasks. An external Fortran code, "*v1least.exe*", that does regression analysis was developed and linked to the advisor. The code uses the least

squares method, and when provided with the degree of the polynomial required and a data set containing one independent variable and one dependent variable, it calculates the value of the coefficients. A second Fortran code, "*v2least.exe*", that handles two independent variables which can be of different degree polynomials, was also developed. "*v2least.exe*" is not used under the current version of the advisor, but is anticipated to be used in future versions.

Since an expert advisor has to develop and be modified constantly as the body of knowledge on the domain grows, it is important to design the advisor in a way that eases such future tasks. Expert systems are especially suited for this, since rules allow modular representation of knowledge. In the best case scenario, most of the rules should be independent of each other so that, if, in the future, a rule is modified, it does not render other rules ineffective. One of the important advantages of rule based systems to procedural programs is that a piece of code is not entwined with the rest of the program, but instead reacts to the current state of the system, represented as facts and objects, independently.

Conducting experiments is sequential and procedural to a certain extent, and complete modularity of rules was not easily achievable. Instead, CLIPS 6.0 supports modularity of subdomains, and this feature was used to develop separate experimental models for each of the control parameters. Modules can be completely independent and blind to other modules, or there can be a hierarchal relationship such that a module is visible to another module, but it

"see" the other module.

The optimization advisor has the following modules: ***start-up, main, economizer-oxygen, burner-tilt-angle, oxygen-burner-tilt, mills, secondary-air-dampers, and burner-tilt-auxiliary-air-dampers.*** The ***start-up*** module is visible to all the other modules, and the ***main*** module is visible to all modules except ***start-up***. The rest of the modules are independent of each other, except for the ***secondary-air-dampers*** module which is visible to the ***burner-tilt-auxiliary-air-dampers*** module (see Figure 13 and Figure 14).

The ***economizer-oxygen, burner-tilt-angle, mills***, and ***secondary-air-dampers*** modules contain rules on parametric experimental procedures of their respective control variables. The ***oxygen-burner-tilt*** and ***burner-tilt-auxiliary-air-dampers*** modules contain rules that specify experiments on different combinations of the two variables suggested in their names. The ***main*** module coordinates the activities of the advisor by interacting with the different modules, the user and the data acquisition system. The ***start-up*** module is initiated before the optimization exercise begins and obtains information on the configuration of the system and any initial information relevant to the optimization exercise.

When the optimization advisor is first started (refer to Figure 15 for flow chart of the execution of the program), the ***start-up*** module has control of the program. The user is asked to choose the optimization goal:

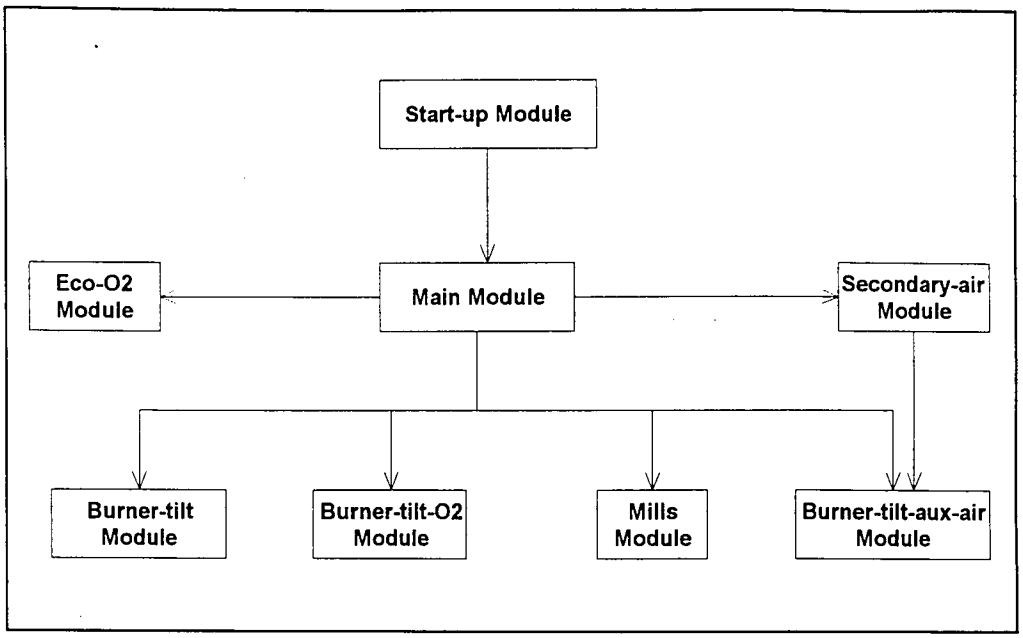

Figure 13 Visibility of Modules

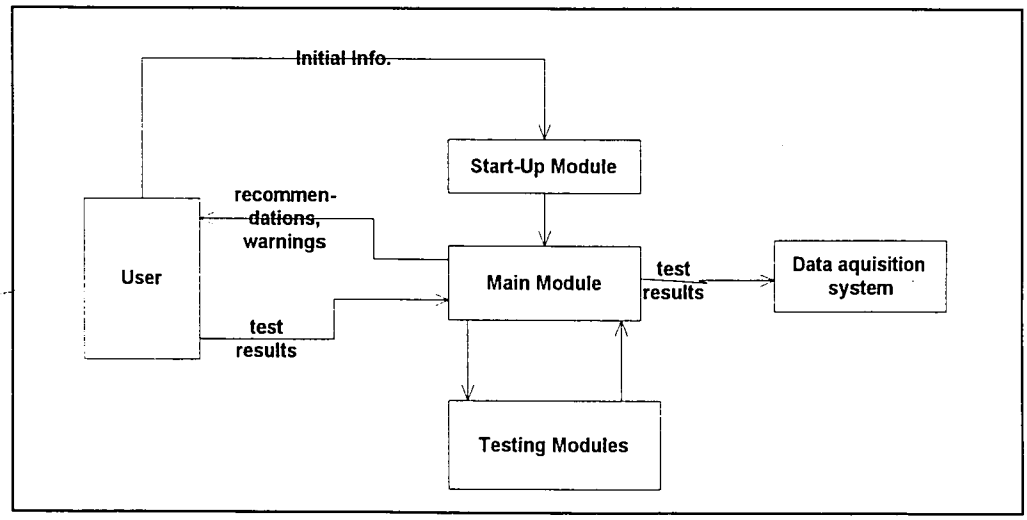

Figure 14 Module and User Interaction

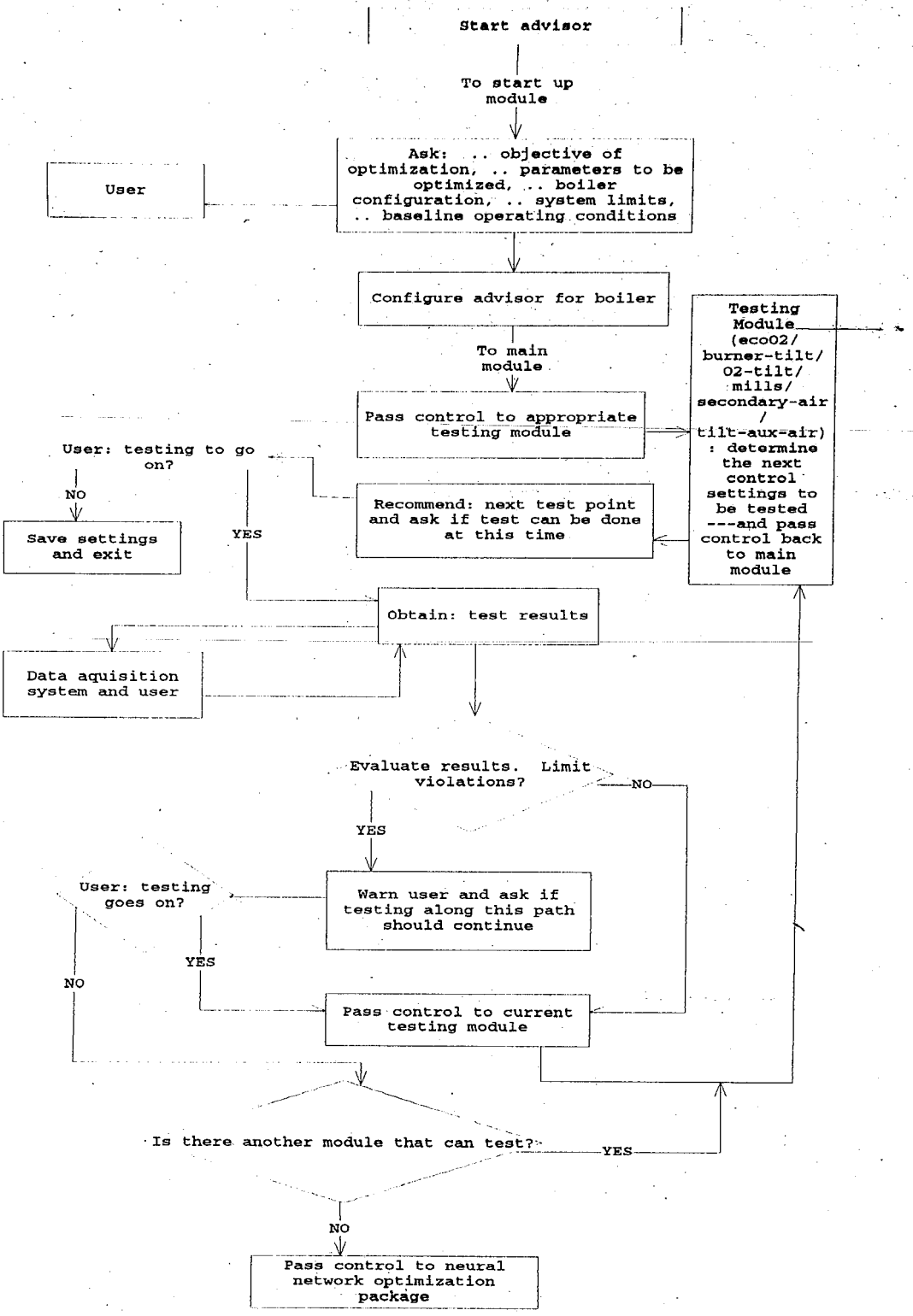

Figure 15 **Execution of Optimization Advisor**

- find the minimum NO_x possible; or
- find the minimum heat rate constraining NO_x to a target value.

If the user chooses the second option, he/she is asked to provide the target NO_x value. Then, the user is asked if any of the control variables have been optimized previously or if any are inoperational. Depending on the optimization goal and the parameters to be optimized, one or more of the modules might be set inactive. The sequence of the optimization has been discussed in the experimental procedures section. To learn the configuration of the system that is to be optimized, the advisor prompts for the following information:

- number of mills
- maximum capacity of mills
- number of auxiliary and fuel air dampers
- open and closed setting of dampers
- minimum and maximum burner tilt angles.

In this initial version of the advisor, the number of mills and number of secondary air dampers are held fixed at the Potomac River Unit 4 configuration (four mills, five auxiliary air dampers, and four fuel air dampers).

Next, the user is asked to provide design and safety limits on the following parameters:

- minimum furnace oxygen
- minimum and maximum windbox pressure

- minimum and maximum main steam temperature
- minimum and maximum hot reheat steam temperature
- maximum opacity
- maximum LOI
- minimum and maximum mill currents
- minimum and maximum mill suction pressures
- minimum and maximum mill suction pressures.

These parameters are monitored after each test to see if any violation has occurred.

Then, the user is asked to provide baseline operating settings for economizer oxygen, burner tilt angle, mill RPMs, and secondary air damper positions. This information is used in the first test point unless the advisor modifies it to settings closer to the low NO_x operating conditions. In accordance with the objective of minimizing the number of experiments, initial baseline settings are sometimes modified within the advisor: if oxygen level is very high, it is reduced to a value known to be safe; if burner tilt angle is set very high, it is reduced to halfway between horizontal and maximum angle; and a slight upward bias is given to auxiliary air dampers. This initial setting is changed back to baseline settings if the first test produced any limit violations.

Next, control of the program is passed to the ***main*** module. The ***main*** module passes the control to the ***economizer-oxygen***, ***burner-tilt-angle***, ***oxygen-burner-tilt***, ***mills***, ***secondary-air-dampers***, or ***burner-tilt-auxiliary-air-dampers*** modules

depending on the goal of the optimization, the parameters that are to be optimized, and the predetermined sequence of testing. The module that has the control then determines what the next test point should be, and passes the information to the ***main*** module.

The ***main*** module communicates this information to the user along with instructions on how to conduct the test. For example, these instructions might include "the unit should be steady before testing begins, and tests should be at least fifteen minutes long." Before conducting the test, the user is given the option of suspending testing for the moment, in which case the current state of the program is saved and the program is interrupted. When the user restarts the program, it starts from the point where it was last suspended. This is an important feature since the experiments can take several days to perform, and an uninterrupted testing routine can not be guaranteed.

After the test is conducted, the advisor obtains the time range of the experiment from the user, extracts all available online data from the plant data acquisition system, and prompts the user for the rest.

The data are then analyzed by the ***main*** module to see if any limit violation has occurred. If a limit violation has occurred, the user is made aware of this via a warning message. The user is then asked to make a judgement on the seriousness of the violation, and whether that particular course of testing should be discontinued. For instance if the opacity limit were violated, the user could elect to temporarily sacrifice a little on opacity to reduce NO_x.

All information on the current test data and any limit violations, are passed along to the current module. Using this information, the module determines if more testing is needed. If more testing is needed, it passes its recommendation on the next test point to the ***main*** module.

Recommendations of test points are based on the current state of the boiler and are determined using the knowledge described in the previous sections. Recommendations may vary for different optimization goals. Test point recommendations are specific for all the control parameters except mills. Since the loading capacity of the mills can change with varying coal supply, the user is only instructed as to the loading pattern of the mills. When the optimization objective is to find the minimum heat rate with a constraint on NO_x, there is a need to obtain mill bias information at a number of β values instead of just at the minimum and no bias situations. So, once the mill capacity during a particular series of test has been established (capacities at no bias and minimum bias are recorded from previous test points during the same test series), the advisor recommends specific RPM's for each of the mill feeders that would give intermediate β values.

If the module that has current control of the program decided that further testing is not needed, then in some cases, it may analyze the data gathered using "***v1least.exe***" to determine if any significant reduction in NO_x has been achieved, and to find a rough estimate of the control setting needed to achieve minimum NO_x. This setting is fixed in subsequent tests of other control

parameters. When appropriate, whether the target NO_x level has been achieved is also determined. The modules that perform the type of data analysis mentioned above are ***economizer oxygen***, ***burner-tilt-angle***, ***mills*** and the auxiliary air section of the ***secondary-air-dampers*** module. The other modules leave any data analysis for the final portion of the optimization package.

If the ***main*** module gets information to conduct more tests, it goes through the same procedure described above. Otherwise, it determines what the next module should be and passes on control. At the beginning of the ***mills*** and ***secondary-air-dampers*** modules, the user is given the option of abandoning further testing if he/she so chooses. This option is provided for the user in case the NO_x reduction achieved up to that point has been sufficient, and the user determines that further testing is not cost effective. Also, before beginning testing of fuel air dampers, the user is cautioned that a person with experience in evaluating ignition points is needed to conduct the tests, and is given the option of abandoning fuel air testing.

If the advisor determines that all the required testing is done, it passes the data base accumulated to the neural network-optimization package [2]. Other important information passed by the advisor to the neural network-optimization package are the optimization goal, target NO_x value if applicable, the parameters to be optimized, and ranges of safe operating conditions for each of these parameters.

The neural network learns the relationships of the dependent variables

with all the relevant independent variables. These relationships are passed to the optimization subroutine which then determines the optimum control settings (see Ref 2 for more information on the neural network-optimization package).

Start-Up Module

The ***start-up*** module has definitions of functions and data bases to be used by the advisor, and rules that obtain initial information and configure the advisor for the boiler that is being optimized.

The functions defined are classified into two categories: input-output functions that interact with the user in the absence of the user interface that is currently under development; and mathematical functions that are used throughout the advisor. The input-output functions include ***Choose_Ask***, ***Type_Number***, ***Type_Integer***, ***Y_N_Ask***, ***good-ok-bad***, and ***run-window-app***. ***Choose_Ask*** prints a specified question with a few choices to the screen and reads the answer back if it corresponds to the choices provided. ***Y_N_Ask*** uses ***Choose_Ask***, but the choices it provides the user are fixed between yes and no, while ***good-ok-bad*** fixes its choices between good, ok, and bad. ***Type_Number*** and ***Type_Integer*** print a question to the screen and read back a real number and integer respectively. ***Run-window-app*** uses ***Y_N_Ask*** to request the user to run an external program and answer yes when the request is performed. All these functions print an error message to the user and print their question again if the user's input is not expected. All the input-output functions are temporary

until a window user interface is developed, and ***run-window-app*** is a temporary function until an internal CLIPS function that runs external programs invisible to the user is created.

Other functions defined in the ***start-up*** module calculate the mill bias given the mill RPMs, and the auxiliary air bias given auxiliary air damper positions. Functions that calculate first and second order polynomials given values of coefficients and the independent variables, and the roots of quadratic equations given coefficients and the dependent variables are also defined.

To store the data generated by the experiments, a class of objects called ***oper-data*** is defined. ***Oper-data*** is defined as "pattern-match reactive" so that rules can access any of its objects by pattern matching just like any other facts (refer to Section on expert systems). All the slots[6] in this class can be written to and read from. A slot for each of the variables that describe a test point is defined, and these slots are declared multifield[7], so that they can store more than one variable. Thus, a multifield slot called ***mill-amps*** would store four mill current values corresponding to four different mills, and a multifield slot called ***aux-air-dampers*** would store values corresponding to the five auxiliary air dampers. Besides the slots that contain information from a particular test, slots called ***status*** and ***par-eval*** are defined. ***Status*** stores information used to identify

[6] Slots are fields that contain a particular piece of information on an object.

[7] A multi field slot is a slot that can contain more than one piece of related information.

what a particular object is being used for: *current*[8] is an object that stores the data for the current test point; *previous* is an object that stores data from previous tests; *optimum* is an object that stores optimum control settings as determined by the expert advisor up to that point; *baseline* stores baseline control settings as specified by the user; *system-baseline* stores advisor modified baseline settings to be used as the first test point; and *recommend* stores advisor recommended control settings for the next test point. Objects with ***status*** *current* eventually become *previous*, and all the objects with ***status*** *previous* make up the database accumulated by boiler testing. There will only be one object each for the ***status*** *baseline*, *system-baseline*, *optimum*, and *recommend*. On the other hand, the slot ***par-eval*** is used to store information on whatever series of testing that particular object is associated with. For example, a value of $ecoO_2$ would mean that the object contains data obtained during economizer oxygen parametric testing, while a value of *tilt-sec-air* would mean that the object contains data obtained during a factorial burner tilt and auxiliary air damper testing series. In the object with ***status*** *recommend*, the slot ***par-eval*** is used for communication between the ***main*** module and the other testing modules (this is described in the ***main*** module section).

Initial facts that set the sequence of the activation[9] of the rules described

---

[8] *Values used or stored by the code are shown in this font.*

[9] Activation in rule based language refers to the triggering of a rule to be possibly used in the near future.

below are first asserted[10]. Thus, depending on what fact is on the top of the list, the following rules are activated and fired[11] (refer to Figure 16 for flow chart). When a rule fires, the actions specified in the consequent part of the rule are performed.

A rule called ***objective*** prompts the user for the optimization objective and asserts this fact into the fact list. If the objective is to minimize heat rate with a constraint on NO_x, this rule causes rule ***targetNO$_x$*** to be activated. Otherwise a fact that says "target NO_x is 0" is asserted into the fact list. ***TargetNO$_x$*** prompts the user for the target value of the NO_x and asserts this fact.

The default objective of the optimization exercise is set by ***objective*** to be minimizing NO_x, no matter what optimization objective the user chose. Once the program is underway, in the case when the optimization objective is to minimize heat rate with a constraint on NO_x, if target NO_x is achieved, the current objective is changed to minimizing heat rate. The strategy under the default objective of minimizing NO_x is to continually assess the control settings that give low NO_x and use these settings for the next series of testing. The strategy under the objective of minimizing heat rate, is to guide testing to obtain enough data for heat rate and NO_x optimization at the completion of the program. These strategies are discussed in greater detail under the modules

---

[10]To "assert a fact" in rule base language refers to storing and saving a particular information containing statement (fact) into the system memory.

[11] When an activated rule is used to perform the actions that it is designed for, the rule is "fired".

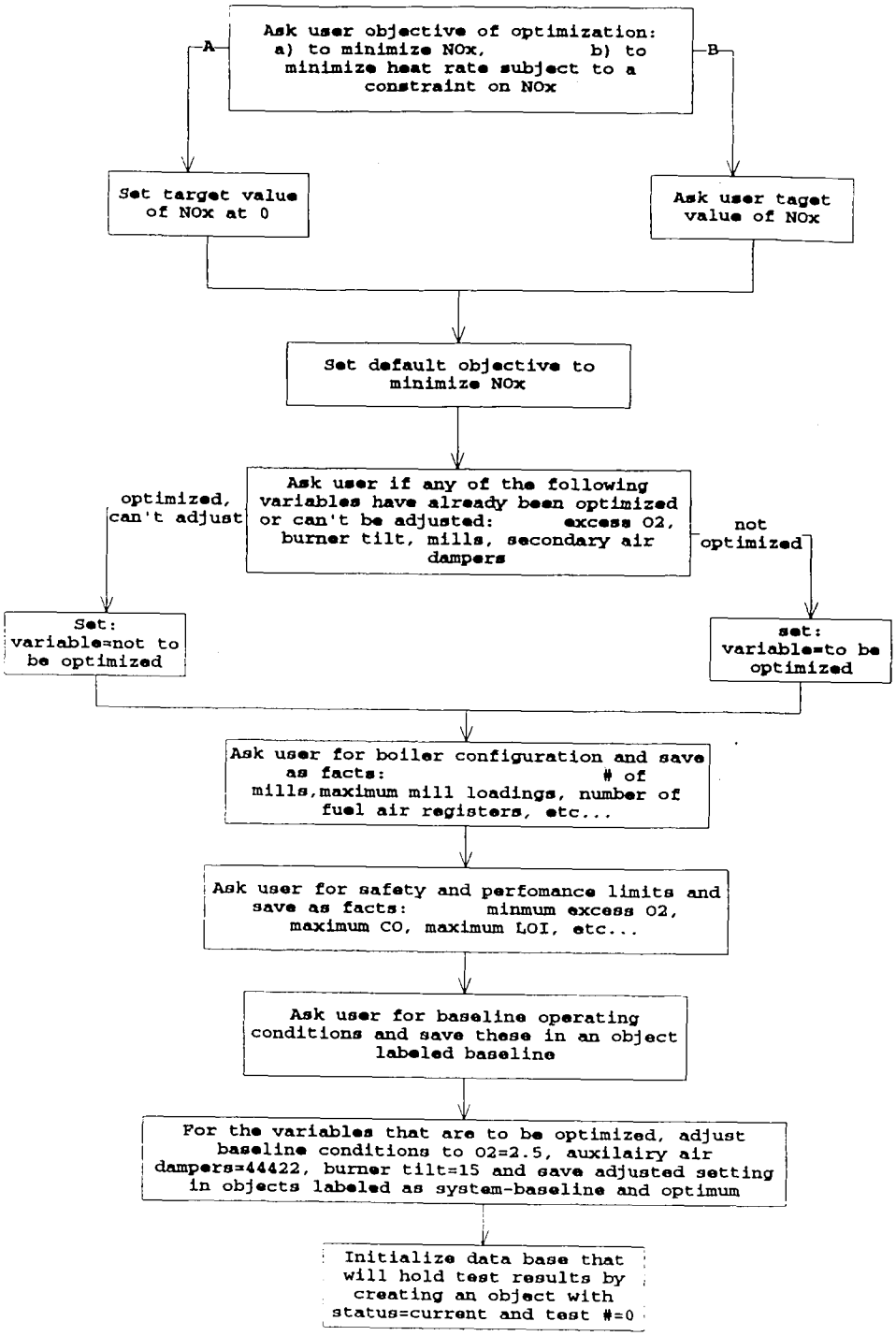

Figure 16 **Start-up Module**

that guide testing for each of the parameters.

Then, a series of rules ask the user which parameters can and should be optimized and determine which modules the advisor will use for testing. Examples of these rules are,

If

user says O_2 has been optimized or
user says burner tilt can't be optimized

then

combination of O_2 and tilt has been optimized

If

optimization objective is to minimize NO_x

then

combination of O_2 and tilt can't be optimized
and combination of tilt and auxiliary air dampers can't be optimized.

The results of these rules determines which one of the modules will be used for testing. If there is a fact that states that a particular parameter or combination of parameters has been or can't be optimized, then the module associated with the parameters is never activated.

A configuration rule prompts the user for values that describe the system and asserts these as facts, and a safety limits rule asks the user for safety and performance limits and asserts these as facts. "Number of auxiliary air dampers is 5" and "minimum main steam temperature is 850" are two examples of such facts.

A rule called ***baseline-operators*** prompts the user for baseline operating control settings and stores them in an ***oper-data*** object with ***status*** *baseline*. Then

a series of rules finds the control settings for the initial test point and stores them in ***oper-data*** objects with ***status*** *system-baseline, optimum* and *recommend.* The control setting for the initial test point is the same as the baseline control setting if that particular parameter has been or can't be optimized. Otherwise, economizer oxygen is set at 2.5%, burner tilt angle is set at 15°, mill bias is set at 0, and auxiliary air dampers are set at positions 4,4,2,2,2 corresponding to a bias of 0.5. These settings are for Potomac River Unit 4, and would need to be readjusted for different boiler configurations.

Control of the program now passes to the ***main*** module.

Main Module

The ***main*** module asserts a few sets of initial facts before execution of the program starts. One set of facts relate variables with the units associated with them. For example, "unit of economizer oxygen is %" etc. These facts are used when communicating information about a particular variable to the user. Another fact states that "significant reduction in NO_x is 0.01 lb/MBtu." This fact is used when analyzing if a particular series of testing produced any significant results.

Another set of facts deals with the optimization sequence of the control parameters: "If nothing is optimized, then goal is economizer oxygen"; "If economizer oxygen is optimized, then goal is burner tilt angle"; "If burner tilt angle is optimized, then goal is O_2-tilt combination"; "If O_2-tilt combination is

a series of rules finds the control settings for the initial test point and stores them in ***oper-data*** objects with ***status*** *system-baseline*, *optimum* and *recommend*. The control setting for the initial test point is the same as the baseline control setting if that particular parameter has been or can't be optimized. Otherwise, economizer oxygen is set at 2.5%, burner tilt angle is set at 15°, mill bias is set at 0, and auxiliary air dampers are set at positions 4,4,2,2,2 corresponding to a bias of 0.5. These settings are for Potomac River Unit 4, and would need to be readjusted for different boiler configurations.

Control of the program now passes to the ***main*** module.

Main Module

The ***main*** module asserts a few sets of initial facts before execution of the program starts. One set of facts relate variables with the units associated with them. For example, "unit of economizer oxygen is %" etc. These facts are used when communicating information about a particular variable to the user. Another fact states that "significant reduction in NO_x is 0.01 lb/MBtu." This fact is used when analyzing if a particular series of testing produced any significant results.

Another set of facts deals with the optimization sequence of the control parameters: "If nothing is optimized, then goal is economizer oxygen"; "If economizer oxygen is optimized, then goal is burner tilt angle"; "If burner tilt angle is optimized, then goal is O_2-tilt combination"; "If O_2-tilt combination is

optimized, then goal is mills"; "If mills are optimized, then goal is secondary air dampers"; "If secondary air dampers are optimized, then goal is burner tilt angle-auxiliary air dampers combination".

A rule called ***perpetuate***, looks through the facts in the system and uses the above facts to determine what the current goal should be (refer to flow chart in Figure 17). For example, if, after coming out of the ***start-up*** module, it is known that O_2 has already been optimized and burner tilt can't be adjusted, this rule would determine that mills should be the current goal in the optimization exercise. Once mills are optimized, this rule would then determine that the current goal is secondary air dampers. Then, a rule called ***change-module*** hands control of the program to the module that deals with the parameter/s specified in the current goal statement.

Before control of the program is handed over to the testing module, the current state of the ***main*** module is saved. At the entrance to the testing module, the current state of that module is also saved. When the program comes back to the ***main*** module, the current state of the ***main*** module is saved again to capture any changes made by the testing module. The state of the advisor is saved as a precaution to any computer or electrical failure that might cause the loss of days of testing. The saved states are also used if the user temporarily suspends programming and restarts the program at a later time.

The program comes back to the ***main*** module with the recommended control settings for the next test stored in the object with ***status*** *recommend*.

REDUCTION RATIO 14:1

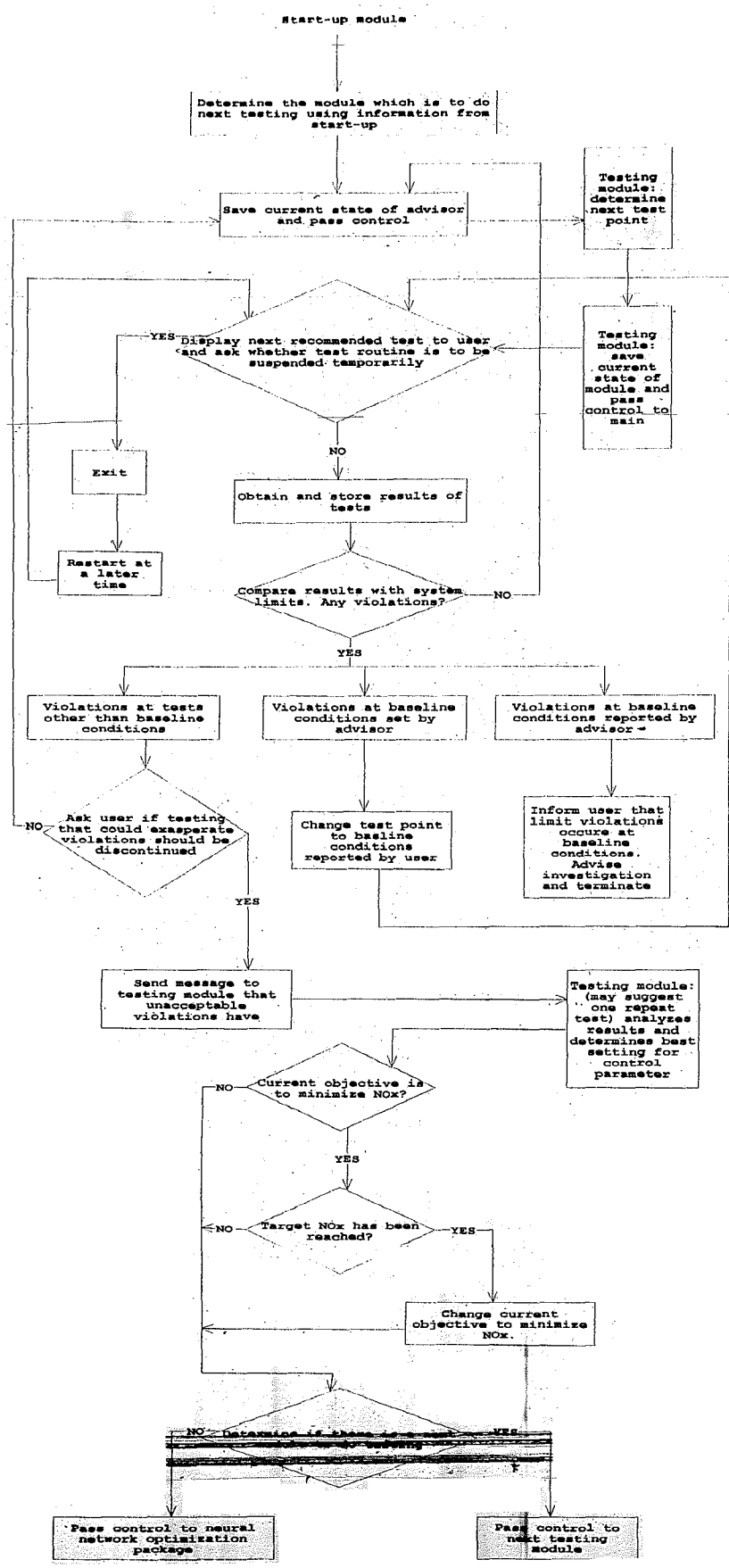

Figure 17 **Main Module**

REDUCTION RATIO 11:1

Then, rules ***do-test*** and ***results*** communicate these recommended values to the user and obtain the results from the current test. The data from the current test are stored in a ***oper-data*** object with ***status*** *current*, while the ***status*** slot of the object containing the data form the last test is changed to *previous*. Rule ***do-test*** gives the user the option of suspending testing for the moment. If the user declines, ***results*** is fired. Otherwise, program is halted to be restarted at a later time with the saved data.

A series of rules compares the current data with the system limits that are in the fact base and alert the user if any violation has occurred. The user is asked to make the decision as to the gravity of the violation. If the user finds the violation unacceptable, then the rules assert a fact stating the violation and at what test point it occurred. An example of such a fact is "opacity violation of 1.6% occurred in O_2 parametric testing at test no 5".

Special rules deal with limit violations at baseline conditions. These rules are activated if a limit violation occurs at the first test point. In this case, the rules would convert the initial test point to user supplied baseline settings stored in the object with ***status*** *baseline*. Rules ***do-test*** and ***results*** then communicate recommended settings to the user and obtain results back.

If this next test point also had limit violations, a rule that advises the user to the fact that limit violation occurred at user supplied baseline conditions is fired. The user is advised to investigate the situation and restart the program at a later time and the program terminates.

Other relevant rules in the ***main*** module assess the different stages of the optimization exercise. Rule ***object-hatred*** changes the current objective to minimizing heat rate, if the optimization objective is minimizing heat rate with a constraint on NO_x, the current objective is to minimize NO_x, and the fact that target NO_x is reached is known (see ***start-up*** module section for description of current objective.) The fact that target NO_x is reached is determined by one of the testing modules.

Rule ***optimized*** is fired if the object with ***status*** *recommend* has *optimized-par* in its ***par-eval*** slot. This occurs if the testing module that has current control of the program has decided that it has finished testing and wants to pass this information to the ***main*** module. The rule then asserts a fact that states that the parameter that was just being tested is optimized and updates the object with the ***status*** *optimum* with the optimum control setting of that parameter. This setting is passed by the testing module to the ***main*** module via the object with ***status*** *recommend.* ***Perpetuate*** is then fired to determine the next goal in the optimization exercise (the next parameter/s to be tested.)

Economizer-Oxygen Module

The ***economizer-oxygen*** module has an initial fact in its knowledge base stating that the economizer O_2 increment is 0.2%. This is the increment used in testing O_2 control settings and stating it as a fact makes any subsequent adjustment of the increment easy.

Rule ***start-test*** is fired only when the ***economizer-oxygen*** module is first called (refer to Figure 18 for flow chart). It modifies[12] the ***par-eval*** slot of the *recommend* object to have the value $ecoO_2$ to indicate the tests that are being done from here on are parametric economizer oxygen tests. The value of the ***par-eval*** slot is copied to all subsequently created objects with ***status*** *current.* Rule ***start-test*** opens a data file "***o2.dat***" where all O_2 set point tested in this module and the corresponding NO_x value are stored. At the end of the testing series, the data in this file are analyzed. Once this rule is fired, control of the program is handed over to the ***main*** module. Since the ***economizer-oxygen*** module has not yet made any changes to the oxygen control setting, the O_2 set point that is going to be tested next is the system baseline.

Prep-experi determines the O_2 set points for the rest of the O_2 parametric testing period. This rule is fired if there has not been any violation during the parametric O_2 testing. This rule writes the values of O_2 and NO_x in the object with ***status*** *current* into "***O_2.dat***", and decreases the O_2 setting in the object with ***status*** *recommend* by the O_2 increment stated earlier. After this rule if fired, control of the program is handed to the ***main*** module.

If a limit violation has occurred, then ***finish-experi*** is fired. This performs the same tasks as ***prep-experi*** except that it sets the O_2 value in the object with the *recommend* ***status*** to the first O_2 value tested. This value is obtained from the

[12] To "modify a slot/object/fact" refers to replacing the old values held by these items by a new value.

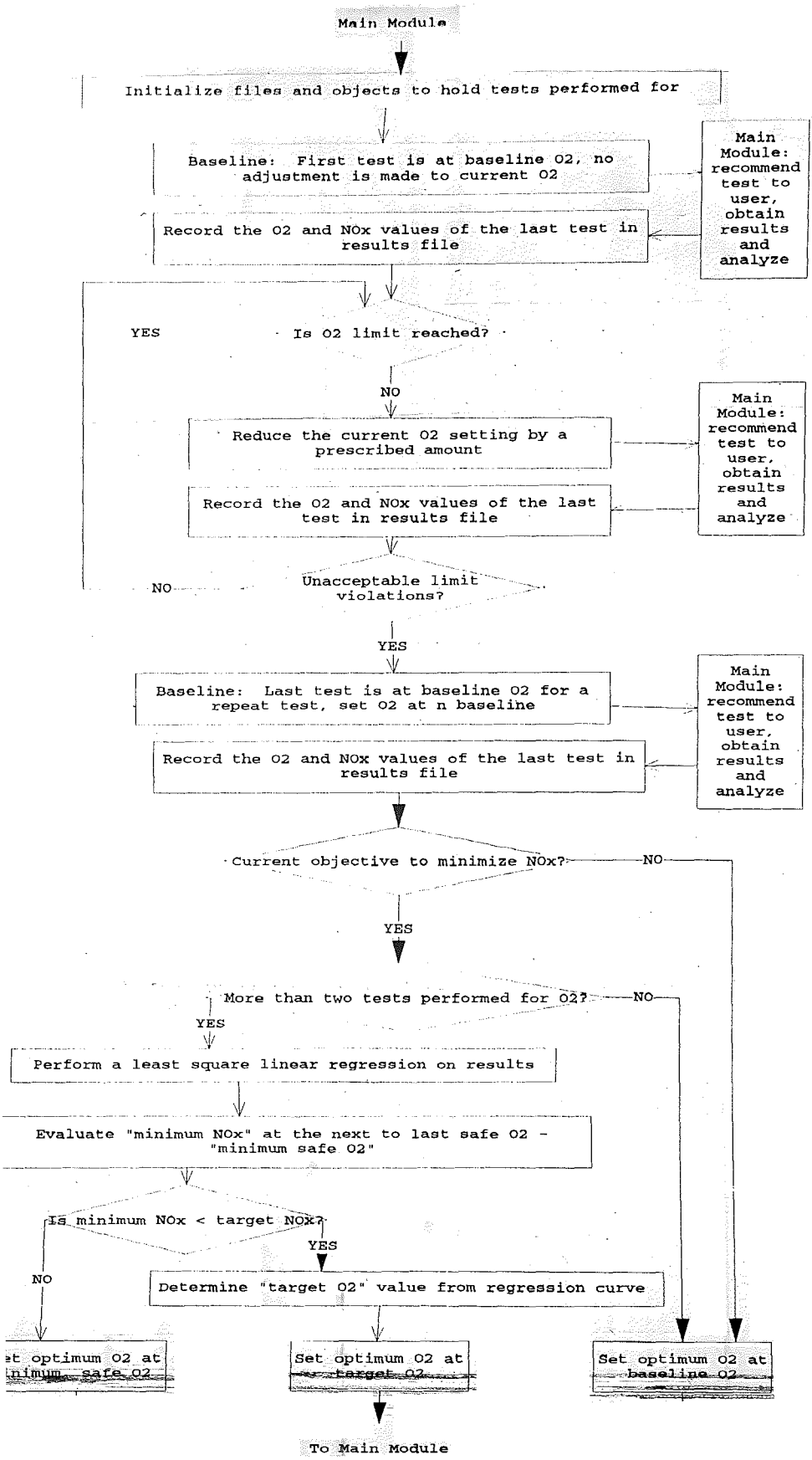

Figure 18 **Eco-O2 Module**

object with *optimum* ***status*** since this object is not modified until the series of testing of a particular parameter is finished. This repeat baseline test is important since slag might have been accumulating on the boiler walls since the first O_2 test was conducted, and since O_2 was tested systematically in one direction the true relationship of O_2 with NO_x would be disguised. The repeat baseline test would reveal the variation of NO_x due to slag build up.

Once the program comes back from this last test, the data from the oxygen parametric tests are analyzed only if the current objective of the program is to find the minimum NO_x. In this case, and if more than 2 data points exist, "***o2.dat***" is renamed "***v1data.dat***" which is the file name expected by the external curve fit program "***v1least.exe***". The first line of "***v1data.dat***" is the number 1 to indicate a linear fit. The coefficients from this curve fit are written into a file called "***v1least.dat***". Rule ***evaluate-target*** uses this information to evaluate if significant reduction in NO_x can be achieved by lowering O_2 to the lowest acceptable level. The lowest acceptable level is an increment above the level where a violation occurred.

If significant reduction occurred, this rule also determines if target NO_x was achieved. If target NO_x was not reached, the O_2 value in the *recommend* object is updated with the lowest acceptable O_2 setting plus one increment. This additional increment is a safety measure.

If target NO_x was achieved, the fact that target NO_x has been reached is asserted, and rule ***find-target*** finds the O_2 setting where this target NO_x is

achieved. Before the program returns control to the ***main*** module, the O_2 value in the object with *recommend* ***status*** is updated with this target O_2 setting. This helps ensure lower heat rate values while operating on target NO_x.

If no significant reduction occurred by lowering O_2 or if the current objective is heat rate, the O_2 value in the *recommend* object is set at the initial setting. The value of O_2 in the *recommend* object when the program finally leaves the ***economizer-oxygen*** module, will be the set point for any subsequent testing where O_2 is not varied. The ***par-eval*** slot in *recommend* object is changed to *optimized-par* to tell the ***main*** module that O_2 parametric testing is finished.

Burner-tilt Module

A statement declaring the burner tilt increment to be eight degrees is defined as an initial fact. This increment is used for adjusting the burner tilt settings during this parametric testing.

The strategy behind the sequence of burner tilt testing is to cover the whole range of burner tilt settings before analyzing the data to find the setting that gives the minimum NO_x level. The reasoning behind this strategy relies on the empirical observation that the burner tilt angle has a quadratic relationship with NO_x. Thus, a regression using only part of the range of tilt settings could give a false curve.

Attention was also given to the fact that steam temperature, which is controlled by burner tilt angle, is also affected by slagging. The latter is an

uncontrolled parameter that could significantly obscure the real relationship of burner tilt angle with NO_x. To control the effect of slagging during the parametric testing, randomizing the sequence of the settings tested was necessary. Since this can create practical problems, such as unstable furnace conditions, if tilts are adjusted randomly from one extreme setting to another, or since much valuable time would be lost in waiting for the unit to steady out, complete randomization of the test points was not considered desirable. Instead, the sequence of the experiments systematically reduces the tilt settings by the increment specified in the initial fact until the lowest tilt possible is reached. The tilt angle is then systematically increased, this time testing points in between of points already tested. This strategy was adopted to correct for any bias due to slag accumulation.

The rules in the ***burner-tilt*** module address two actions; one set of rules is used to recommend test points and another set of rules is used to analyze the data that are generated by the parametric testing. Rules ***start-test, prep-experi-in-range, prep-experi-out-range, finish-experi*** and ***ending*** are used to choose the operating settings that are to be tested.

Rule ***start-test***, which is fired when the ***burner-tilt*** module is first activated, opens a data file, "***tilt.dat***", where all the tilt settings that are tested by this module and the corresponding NO_x values are stored (refer to Figure 19 for flow chart). At the end of the burner tilt testing series, the data in this file are analyzed by the second set of rules in the ***burner-tilt*** module.

REDUCTION RATIO 14:1

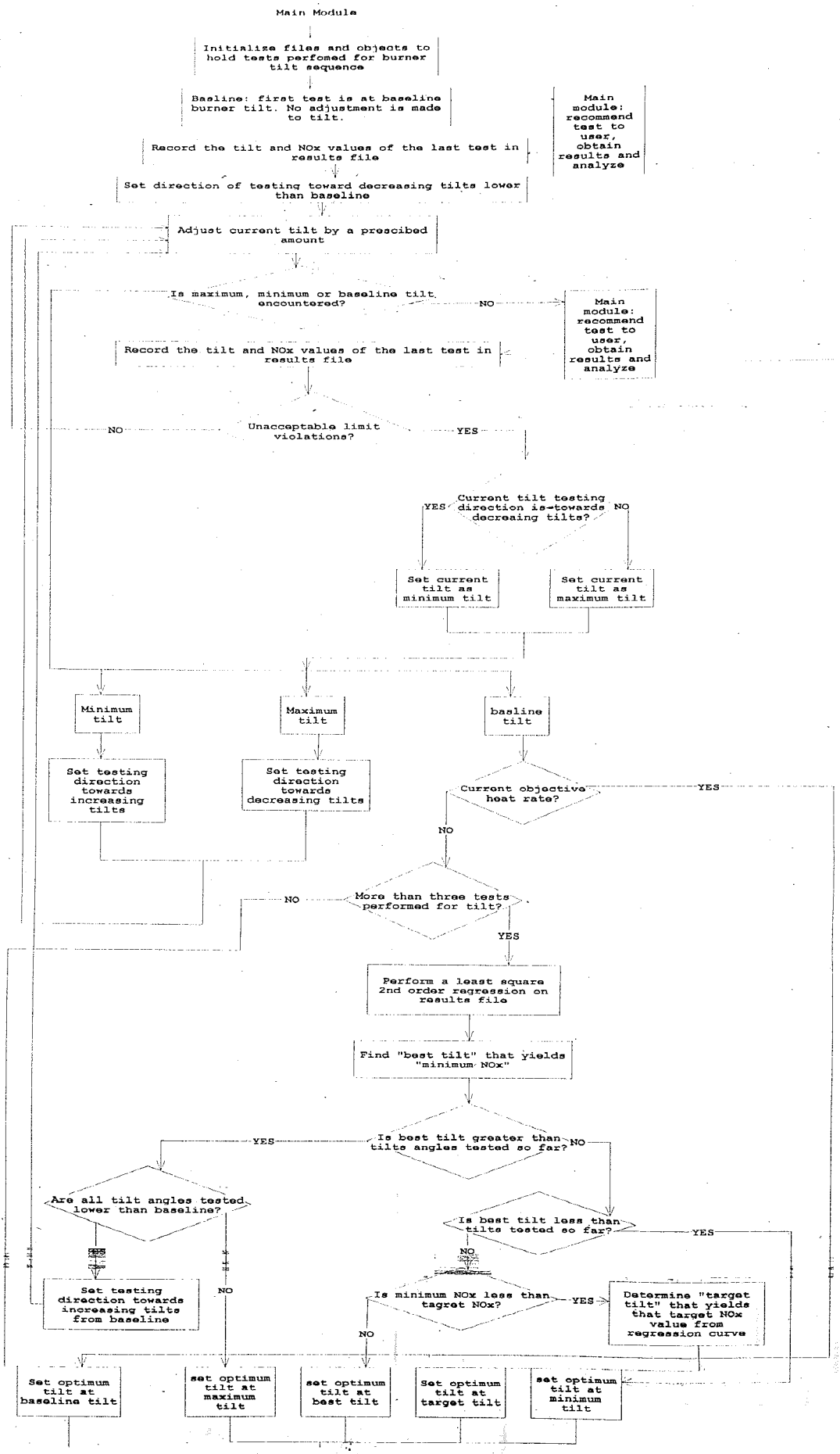

Figure 19 **Burner-tilt Module**

REDUCTION RATIO 11:1

Start-test modifies the ***par-eval*** slot of the *recommend* object to the value *burner-tilt* to indicate that the tests that are conducted from here on are parametric burner tilt tests. The direction of the testing is also set by a fact stating the direction to be towards decreasing tilts. Once this rule is fired, control of the program is transferred over to the ***main*** module, thus making the first point to be tested the one that has been determined to be the optimum up to this point.

Prep-experi-in-range and ***prep-experi-out-range*** select the test points for the rest of the testing series. These rules are fired only if there has been no limit violation and they record the last burner tilt setting tested and the corresponding NO_x value, obtained from the object with the *current* ***status***, in "***tilt.dat***". Depending on what the current testing direction is, these rules either decrease or increase the last burner tilt setting tested by the tilt increment set initially (refer to Figure 20) and store it in the object with ***status*** *recommend*.

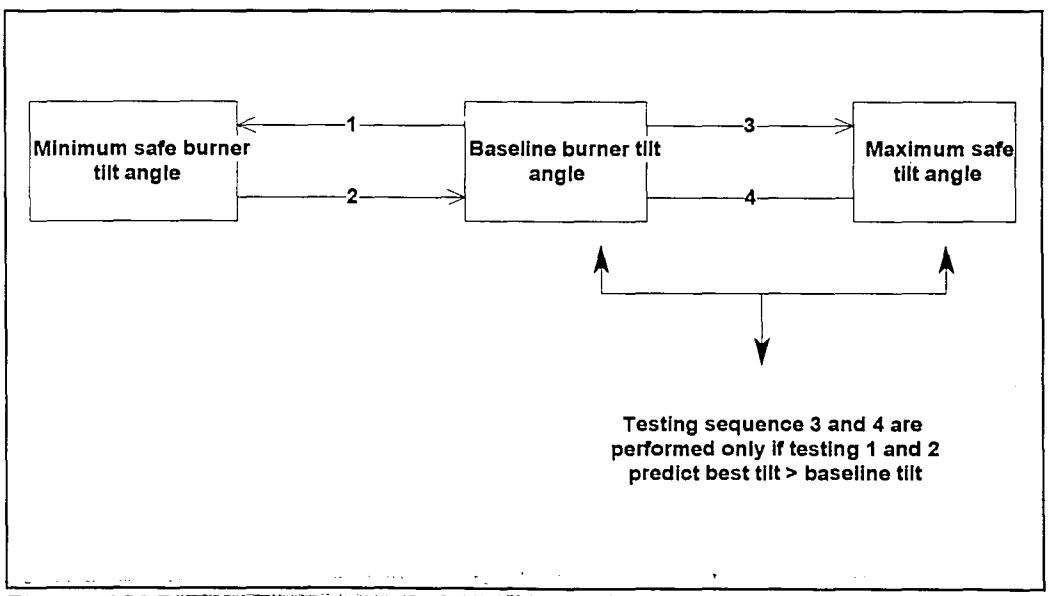

Figure 20 Burner Tilt Testing Sequence

If the last tilt angle tested is near the lowest or the highest possible tilt angles, and if the current testing direction is towards decreasing or increasing tilt angles respectively, then these rules retract[13] the fact stating the current testing direction and instead assert a fact that states a reversed testing direction. The first point to be tested in this new direction is set at a mid point between the last two tilt settings that were tested, thus making the next series of tilt settings to be tested all in between ones that were already tested.

Prep-experi-in-range is the rule that is fired before any tilt data is analyzed. It adjusts tilt settings towards decreasing angles first and then increases tilt angles until the baseline tilt is encountered . Then, if the rules that analyze the data find that the minimum is expected to be found at a tilt setting higher than the baseline setting, the next series of test points are determined by ***prep-experi-out-range*** which tests the tilt settings larger than baseline tilt angle by first increasing tilt angles and then decreasing tilt angles until the baseline setting is encountered.

Rule ***ending*** is fired if ***prep-experi-in-range*** or ***prep-experi-out-range*** can no longer be fired indicating that parametric testing is finished for the moment. If more than two data points have been gathered, this rule causes the rules that analyze data to be activated. Rules ***curve-fit, find-minimum-curve, evaluate-target-no-curve*** and ***evaluate-target-curve*** analyze data that are generated during

[13] To "retract a fact" refers to removing a fact from the system memory, thus making the information contained in the fact unknown from here on.

parametric testing and are only activated if the current objective is to minimize NO_x. If the objective at the moment of burner tilt parametric testing is to minimize heat rate, then the data analysis is left for the neural network-optimization package at the completion of the expert optimization advisor. Rule ***curve-fit*** renames "***tilt.dat***" to "***v1data.dat***" and calls the external curve fit program, "***v1least.exe***". The first line of "***v1data.dat***" contains the number 2 to indicate a quadratic fit. The coefficients from this curve fit are written into a file called "***v1least.dat***". Rule ***find-minimum-curve*** uses this information to find the burner tilt setting that the lowest NO_x level occurs at and determines if the NO_x reduction from baseline tilt settings is significant using the fact in the ***main*** module that states what a significant reduction in NO_x is. If the minimum NO_x level is predicated to occur in the tilt range that is higher than baseline and if this range has not yet been tested, then this rule would activate ***prep-experi-out-range***. Otherwise, the ***burner-tilt*** slot of the object with the ***status*** *recommend* is modified to hold the burner tilt setting that gives minimum NO_x. This will be the value the burner tilt will be set at during the testing of the other control parameters.

Rule ***evaluate-target*** is fired if the current objective of the system is to find the minimum NO_x. ***Evaluate-target*** is used to find if the minimum NO_x value that was found by ***find-minimum-curve*** is lower than the target NO_x value. If minimum NO_x was lower than the target NO_x, this rule finds the burner tilt setting that gives the target NO_x value using the curve that was generated by

the curve fitter. If the burner tilt setting that gives the target NO_x value is higher than the setting that gives the minimum NO_x, the object with the ***status*** *recommend* is modified to hold the target burner tilt setting. This is in accordance with the knowledge that higher burner tilt settings give increased steam temperatures and thus better heat rate.

Tilt-optimized is fired if all the burner tilt testing is truly finished and there are no more relevant rules that can be fired. It hands the control of the program back to the ***main*** module after closing all open files and removing unnecessary facts. At this point the ***par-eval*** slot of the object with the *recommend* ***status*** will have been changed to *optimized-par* by rule ***ending*** after a series of testing has been finished.

O_2-burner-tilt Module

The ***O_2-burner-tilt*** module is activated only if both burner tilt angle and economizer oxygen are parameters to be optimized and the goal of the optimization exercise is to optimize heat rate with a target on NO_x. A 3^2 experimental design is employed to test a narrow range of control settings around the optimum burner tilt angle and oxygen level found in their respective parametric testing.

The rules and facts in the ***O_2-burner-tilt*** module address three tasks; one set of rules determines the nine combinations of burner tilt angle and oxygen levels to be tested; one set of rules and facts chooses the operating settings to

be tested in the next experiment; and the last set of rules evaluates any limit violation and accordingly adjusts the rest of the combinations of tilt and O_2 settings.

Since ***O_2-burner-tilt*** module tests a narrower range of settings than the parametric testing modules, initial facts declare the range of economizer oxygen to be tested to 0.6% and the range of burner tilt angle to be tested to nine degrees. Using these facts and facts stating the highest and lowest burner tilt and oxygen settings tested in the parametric modules, ***generate-combinations*** picks three points each for burner tilt angle and oxygen level around the optimum points determined by the parametric testing modules. ***Form-combinations*** forms nine different paired data sets of O_2 and burner tilt from these points.

Start-test is fired when control is first handed to the ***O_2-burner-tilt*** module (refer to Figure 21 for flow chart), and it initiates the rules that form the paired data sets of oxygen and burner tilt angle. ***Start-test*** also modifies the slot ***par-eval*** of the object with the *recommend* ***status*** to *O_2-tilt* to indicate that the following testing series is for combinations of O_2 and burner tilt angle. ***Prep-experi*** is activated only if there have not been any limit violations or the violations have been dealt with by the third set of rules that evaluate violations. ***Prep-experi*** selects one of the paired data mentioned above and modifies the object with *recommend* ***status*** to hold these values in the ***oxygen*** and ***burner-tilt***

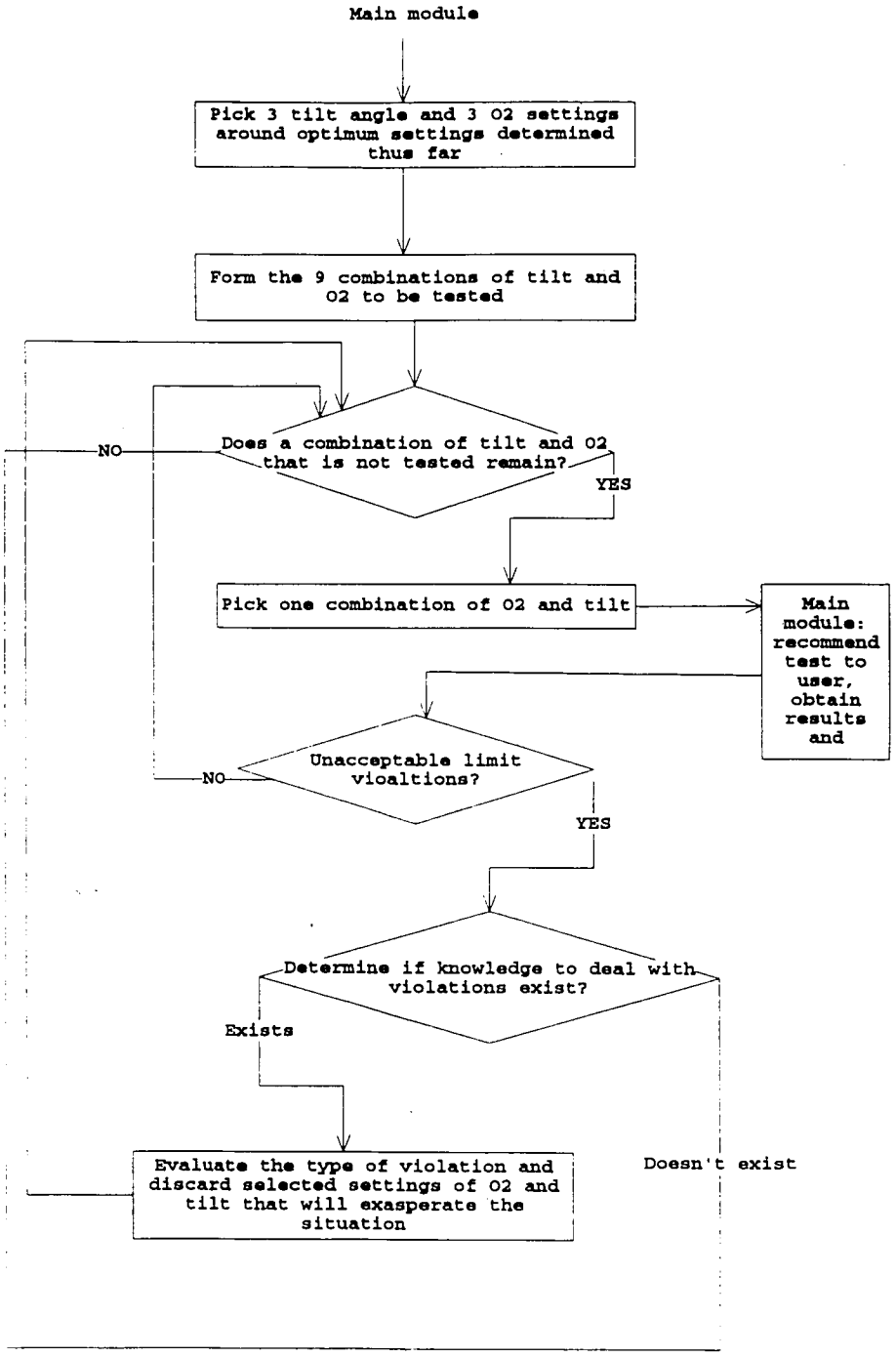

Figure 21 O2-tilt Module

slots. Control of the program is then handed over to the ***main*** module that recommends the testing of the selected points to the user.

The set of rules that deals with limit violations makes use of two kinds of facts. The first kind of facts deals with common sense knowledge and are used by the module to deduce what type of actions to take. These facts are declared in the ***main*** module so that they can be used by other modules if necessary.

These facts are

- if something is too high, lower it,
- if something is too low, raise it,
- opposite of lower is raise
- opposite of raise is lower
- opposite of high is low
- opposite of low is high.

The second kind of facts deal with specific knowledge on boiler variables. These facts are

- to lower NO_x, lower burner tilt angle
- to lower main steam temperature, lower burner tilt angle
- to lower reheat steam temperature, lower burner tilt angle
- to lower NO_x, lower economizer oxygen level
- to lower furnace oxygen, lower economizer oxygen level
- to lower LOI, raise economizer oxygen level
- to lower CO, raise economizer oxygen level

- to lower opacity, raise economizer oxygen level
- to raise main steam temperature, raise economizer oxygen level
- to raise wind box pressure, raise economizer oxygen level
- to improve furnace flame quality, raise economizer oxygen level.

Rules ***action-on-violations***, ***analyze-facts-1*** and ***analyze-facts-2*** use the above facts to deduce what the proper action for a particular violation should be. For instance, if main steam temperatures had gotten too low, then these rules would either conclude that the proper action is to raise the burner tilt angle or to raise oxygen levels.

Rules ***evaluate-violations-O_2-low, evaluate-violations-o2-high, evaluate-violations-tilt-low***, and ***evaluate-violations-tilt-high*** remove paired O_2 and tilt settings from the list selected to be tested depending what the proper action was concluded to be. For instance, if the proper action was found to be to raise the burner tilt setting, then ***evaluate-violation-tilt-low*** removes paired data containing burner tilt settings lower or equal to the one tested last from the paired sets of O_2 and tilt that have not been tested yet.

If there has been a limit violation and the set of rules that deal with violations can't handle this particular problem (knowledge on how to deal with this violation does not exist in the advisor at this time), or all the selected paired O_2-tilt set points have been tested, ***finish-experi*** is fired. ***Finish-experi*** modifies the ***par-eval*** slot of the object with ***status*** *recommend* to contain the values *optimized-par* to indicate to the ***main*** module that testing by this module is done.

Control of the program is then handed back to the ***main*** module.

Mills Module

Facts that set the significant mill RPM change to be twenty and the minimum number of required mill tests to be seven are initially defined. The minimum number of mill tests is determined by the minimum number of tests per variable that the neural network-optimization package needs in order to do a proper analysis. At the initiation of the ***mills*** module, rule ***start-test*** asks the user if he/she wants to go on with mill testing. This is done to give the user the option of reducing the time and effort spent in testing if the NO_x reduction achieved up to that point was deemed sufficient. If the user wants to go on with the optimization, a file, "***mills.dat***", that is to contain the mill bias parameters, β, that would be tested and the corresponding NO_x values is opened.

The type of testing done by the ***mills*** module is determined by the goal of the optimization exercise. If the goal is to find the minimum NO_x regardless of what happens to heat rate, the principle behind the testing is to see if biasing the mills as much as possible produces significant NO_x reduction. Thus, the advisor recommends only two test points to be repeated enough times to satisfy the minimum data requirement. These points are mills loaded with no bias and with maximum bias toward the bottom mills.

The second type of testing is done if the goal of experimentation is to

optimize heat rate while limiting NO_x to a target value. In this case, it is important to provide the neural network-optimization package with enough distinct beta values to generate a relationship between NO_x and β and do proper optimization. Thus, the advisor recommends zero and minimum bias of mills as well as values in between.

Rules ***begin-test-2***, ***no-biasing***, ***calculate-bias***, and ***biasing*** undertake the selection and recommendation of the mill bias parameters to be tested (refer to Figure 22 for flow chart). Rule ***begin-test-2*** selects test points in three stages. The different stages are activated in turn until the minimum number of required tests has been satisfied and each stage handles the selection of the mill bias to be tested next. When the ***mills*** module is first initiated, a rule modifies the ***par-eval*** slot of the object with the ***status*** *recommend* to hold the value *mills* to indicate that the tests that are to follow are for parametric mill testing.

The first stage sets the test point to be done with all the mills loaded equally. Thus, it modifies the ***mill-bias*** slot of the object with the *recommend* ***status*** to zero. When control of the program comes back from doing this test, the second stage changes the slot of the object with the *recommend* ***status*** to hold the value -1. This informs the ***main*** module to recommend to the user to bias back the top mill as much as possible and distribute the extra load equally among the lower mills.

The third stage of the testing sequence is activated if the goal of the

REDUCTION RATIO 14:1

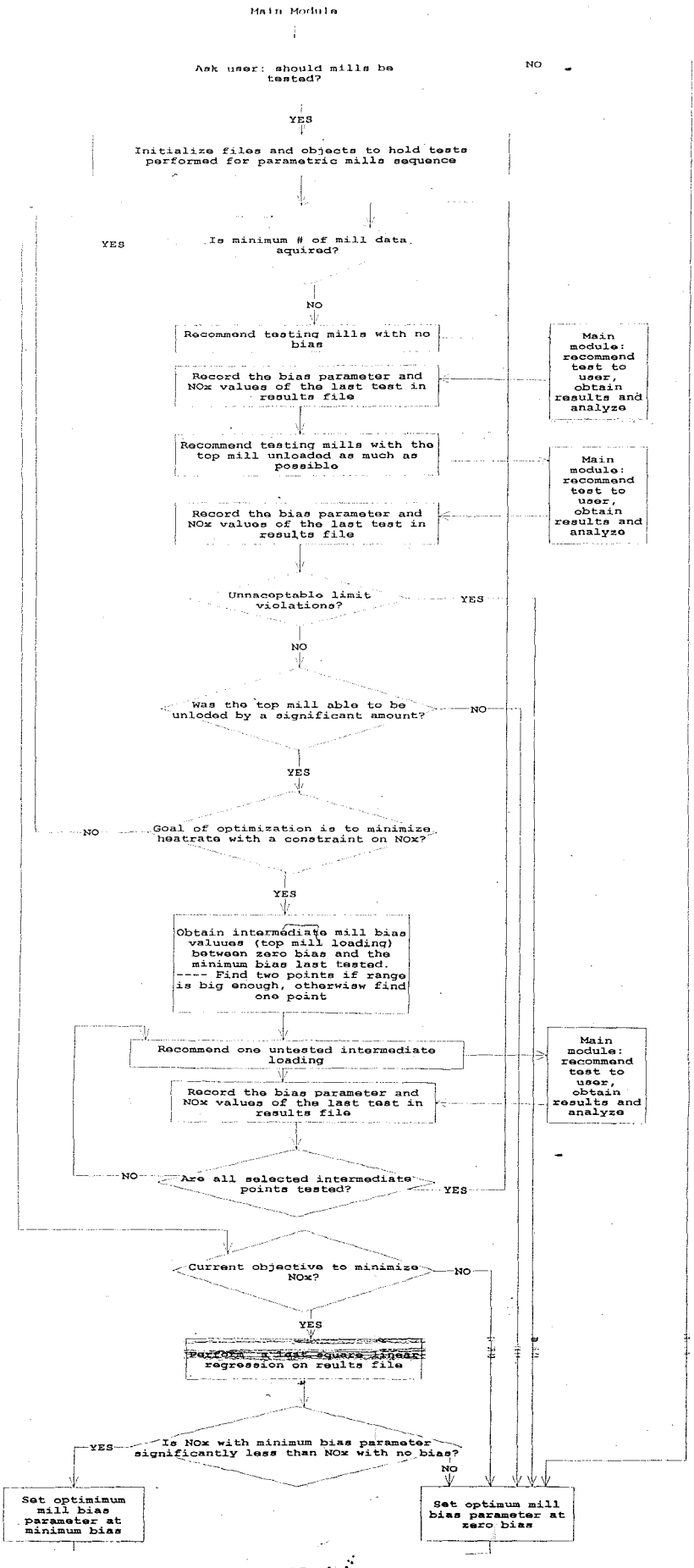

Figure 22 **Mills Module**

REDUCTION RATIO 11:1

experimentation is to optimize heat rate with a constraint on NO_x. If this is not the case, the first stage is activated after the second stage and the next test will be at zero mill bias. The third stage of the testing sequence activates either rule ***no-biasing*** or ***calculate-bias***.

Rule ***no-biasing*** is activated only if the difference between the mill RPM of the top mill when no bias was recommended and when maximum bias was recommended was not significant (less than the significant RPM defined at the beginning of the mills module). In this case, the mills can not really be biased and the rest of the testing is not necessary. End testing is then initiated.

Rule ***calculate-bias*** is activated if significant bias was achieved by the previous two tests. ***Calculate-bias*** divides the difference of the top mill RPM at no bias and minimum bias by three in order to obtain two in-between test points. If the difference between these two new points is insignificant, ***calculate-bias*** obtains only one point in-between the maximum and minimum biases tested.

Rule ***biasing*** then recommends these test points one at a time by modifying the values of the ***mills*** slot in the object with the ***status*** *recommend*. This rule makes certain that the total of all the mill RPM's comes to the total that was recorded when the mills were being run unbiased. The user sees a slightly different recommendation screen for this test point in that instead of general advice to run the mills with no bias or with as much bias as possible, the advisor tells the user at what RPMs to load the mills.

After all the test points found by the third stage have been tested, and if the minimum number of tests has not yet been achieved, the first stage of the testing is reactivated. If the minimum number of tests required have been done, then rule ***ending*** is activated. Rule ***ending*** is also fired if a violation has occurred when biasing the mills. If a violation has occurred or the mills were not able to be biased, the ***mill-bias*** slot of the object with ***status*** *recommend* is modified to zero and the ***par-eval*** slot is modified to *optimized-par*. Then, control of the program is handed over to the ***main*** module.

If mill testing was successful and the current objective of optimization is to minimize NO_x, then ***ending*** activates a series of rules that evaluate the data that were generated. Rule ***curve-fit*** renames "***mills.dat***" to "***v1data.dat***" and calls "***v1least.exe***" to do a linear curve fit on the data. Rule ***evaluate-target***, then, uses the curve that was generated to see if significant reduction in NO_x was achieved by biasing the mills. If biasing mills did not produce any results, then the ***mill-bias*** slot of the *recommend* object is modified to zero. If biasing mills did produce significant reduction in NO_x, then the ***mill-bias*** slot of the *recommend* object is modified to -1. If significant reduction was achieved, then the target NO_x value is compared with the minimum NO_x that was achieved by mill biasing. If the minimum NO_x value was lower than the target NO_x value, then the current objective of minimizing NO_x is removed and an objective of minimizing heat rate is established for the next variable to be tested. At this time, the ***par-eval*** slot of the *recommend* object is modified to *optimized-par* and control of the program is

passed to the ***main*** module.

Secondary-Air Module

The ***secondary-air-damper*** module tests the dampers in two stages: the auxiliary air dampers first, and the fuel air dampers next. At the initiation of this module the user is given the option of quitting if he/she decides that NO_x has been reduced enough. The option of quitting is again offered when testing of the fuel air dampers is about to start if the user is not experienced in evaluating ignition points or if further NO_x reduction is not worth the effort of testing the dampers.

If the user decides to go ahead with auxiliary air damper experimentation, then rule ***aux-combinations*** activates a sequence of rules that generate various combinations of auxiliary air dampers that the advisor may select for testing (refer to Figure 23 for flow chart). Rules ***generate-aux-combinations, wrong-aux-cause-sum, wrong-aux-comb-full-load, remove-insufficient-aux-com*** use the rules that were developed by the ERC (refer to Section on secondary air dampers knowledge base) to develop auxiliary air damper combinations that have less likelihood of yielding unstable furnace conditions. ***Generate-aux-combinations*** generates combinations of auxiliary air damper positions where an auxiliary air damper has at least the same or greater open position as the one a level below it. ***Wrong-aux-comb-sum*** removes any of these combinations that have a sum of damper positions less than fourteen or

REDUCTION RATIO 14:1

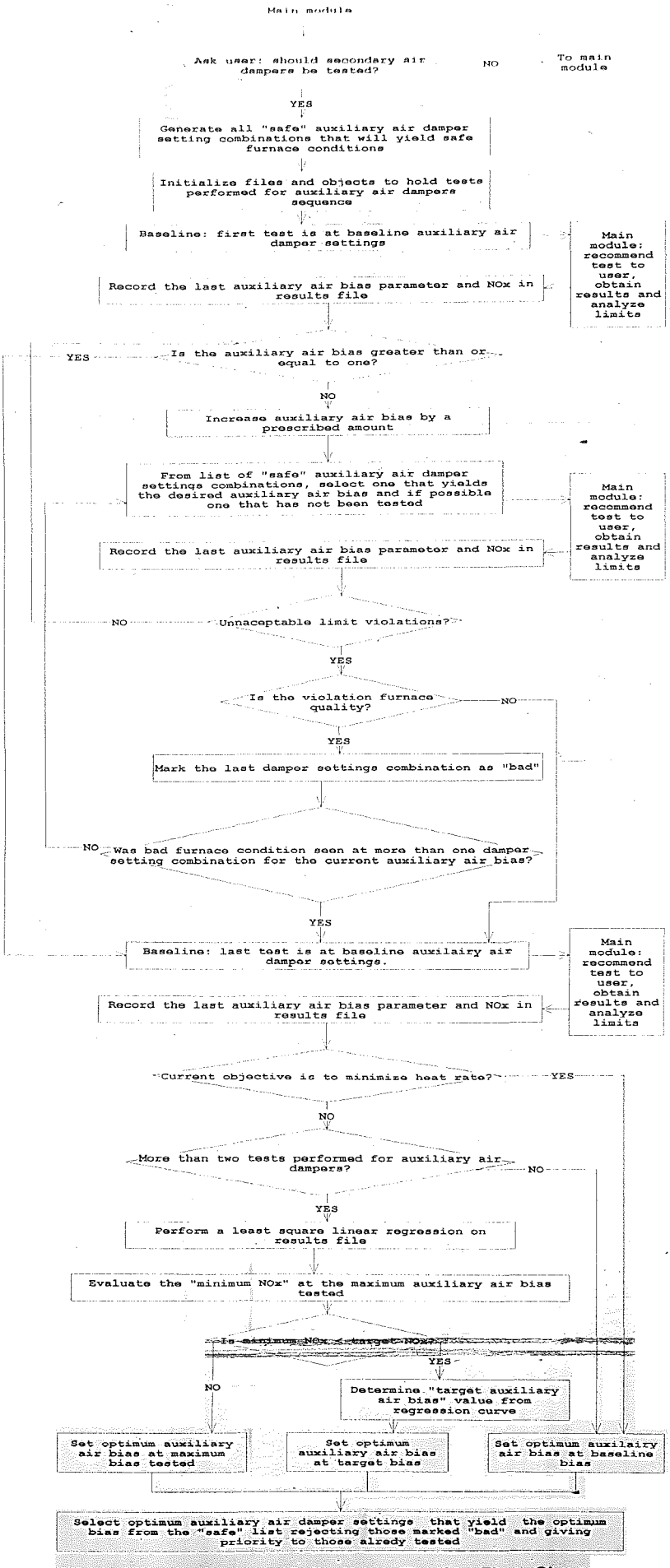

Figure 23 **Auxiliary Air Dampers Testing Flow Chart**

REDUCTION RATIO 11:1

greater than eighteen. ***Wrong-aux-comb-full-load*** removes combinations where more than one damper position is at the most closed state.

After the list of auxiliary air damper positions that can be tested has been compiled, rule ***aux-starting*** modifies the ***par-eval*** slot of the object with the *recommend* ***status*** to *sec-air* to indicate that the coming tests are for secondary air dampers. It also opens a data file, "***auxi.dat***", that is to hold the auxiliary bias parameters, α, that were tested and the corresponding NO_x values. After this rule is fired, control of the program is handed over to the ***main*** module where the baseline damper settings are tested.

When control of the program comes back after doing the baseline test, the rules that select the auxiliary air damper settings to be tested are initiated. If there has not been any limit violation in the last test, ***next-test-aux*** finds the auxiliary air bias of the next test point by calculating the α of the last test and adding one increment to it. If the furnace condition of the last test was bad, the new bias is set at the bias of the last test by ***next-test-aux-bad-furnace***. This would cause the selection of another set of damper settings of the same bias that might possibly give better furnace conditions. ***Next-test-aux-bad-furnace*** also marks the last combination of damper settings as "bad" for future reference. If two damper combinations with the same bias yielded bad furnace conditions, ***next-test-aux-dampers*** activates the end of the auxiliary air dampers testing. If the bias of the last test was the highest auxiliary air bias possible, then both ***next-test-aux*** and ***next-test-aux-dampers*** would activate the rules that deal with the end

of testing.

After the bias for the next test has been selected, a sequence of rules that chooses a combination of auxiliary air damper positions that yield the new auxiliary air bias are initiated. Rule ***possible-dampers*** selects damper settings combinations that yield the desired bias from the list of auxiliary air damper settings compiled previously. In accordance with the rule that damper positions should be manipulated only one position at a time (refer to Section on secondary air damper knowledge base), ***possible-dampers*** selects only the damper setting combinations that satisfy this condition. Thus, a combination of damper settings that yields the desired bias but which has a damper whose position differs from the last setting tested by more than one increment will not be selected.

If more than one combination of damper settings were selected by this method, the following rules are used to select among them. ***Choose-dampers-windbox-high*** removes combinations of dampers which admit less secondary air into the furnace (sum of damper settings is low) if a violation of high windbox pressure was detected. ***Choose-dampers-windbox-low*** removes combinations of dampers which admit more secondary air into the furnace (sum of damper settings is high) if a violation of low windbox pressure was detected. ***Choose-dampers-not-tested*** selects the damper combinations that have not yet been tested. If there are more than one combinations left even after these rules have been fired, ***choose-dampers-arbitrary*** chooses one of the damper combinations

arbitrarily.

Rule ***prep-experi-aux*** modifies the object with ***status*** *recommend* to hold the values of the selected damper positions and writes the α and the corresponding NO_x of the last test to "***auxi.dat***". The dampers that are currently going to be tested are marked "tested" for future reference. Rule ***finish-experi-aux*** is activated if there are no more auxiliary air damper setting combinations selected, which will happen if the highest auxiliary air bias has been already tested, if two damper combinations of the same bias yielded bad furnace conditions, or if the last test created some limit violation that the user found unacceptable. In this case, rule ***finish-experi-aux*** extracts the baseline damper positions from the object with the ***status*** *optimum* and modifies the *recommend* object with these values. This is done to test the baseline conditions at the end of a testing series to counteract the effect of slagging or other time dependent uncontrolled variables.

After control comes back to the ***secondary-air*** module from testing baseline control settings, rule ***end-experi-aux*** modifies the ***par-eval*** slot of the object with ***status*** *recommend* to *optimized-par*. If more than one data point has been collected and the current objective of the system is to minimize NO_x, this rule activates a series of rules that analyzes the data.

Rule ***curve-fit-aux*** renames the file "***auxi.dat***" to "***v1data.dat***" and calls "***v1least.exe***" to do a linear curve fit on the data. Rule ***evaluate-target-aux*** then checks if the lowest NO_x level predicted by this curve gives a significant

reduction in NO_x from baseline auxiliary air bias. If there was significant reduction in NO_x, this rule next checks if the lowest NO_x possible is lower than the target NO_x required. If the lowest NO_x is greater than target NO_x, then the optimum bias is set at the highest bias tested (which will be the bias that will yield the lowest NO_x.) If the lowest NO_x predicted is less than target NO_x value, then ***find-target*** finds the auxiliary air bias that yields the target NO_x using the curve that was generated from the test data and this bias is set as the optimum bias for any series of tests that might follow. In each case, both when the lowest NO_x achieved is lower than or higher than the target NO_x, a series of rules that finds the damper combinations that yield the optimum bias is initiated.

Some of these rules are the same as the ones that select the damper setting combinations of the next test: ***possible-dampers*** and ***choose-dampers-arbitrary***. But instead of using ***choose-dampers-not-tested*** to choose among combinations that were not yet tested, this time the advisor uses rule ***choose-dampers-tested*** to choose among the ones that were already tested. ***Choose-dampers-tested-ok-furnace*** removes combinations of damper settings if bad furnace conditions were exhibited at those particular settings. Rule ***Opti-aux-dampers-found*** then modifies the object with ***status*** *recommend* to hold the optimum damper positions and then control of the module passes on to rule ***change-to-fuel-dampers***.

Change-to-fuel-dampers warns the user that the presence of a person experienced in evaluating ignition points is needed to conduct fuel air damper

testing and gives the user the option of abandoning this series of testing for now. If the user wants to stop, control of the program is handed over to the ***main*** module. If the user wants to go ahead with the testing, then a series of rules that handle fuel air damper testing is activated.

As mentioned in the description of the knowledge base, the most important issue when experimenting with fuel air dampers is to make certain that unstable furnace conditions or flames that are too near the burners are not created. Towards this purpose, rule ***prepare-furnace*** instructs the user to make changes on one damper at a time and check the furnace condition each time before a full test is conducted (refer to Figure 24 for flow chart). The testing sequence is set so that, first, the fuel air dampers are closed as much as possible and then, if further closure of dampers is unsafe, the testing sequence is changed so that fuel air dampers are opened until baseline settings are met. These testing sequences are interchanged until the minimum number of fuel air damper tests is met (this number is declared to be seven at the initiation of the advisor).

Prepare-furnace recommends either the closing or opening of one fuel air damper at a time by one increment and asks the user to check the furnace quality and the ignition point after each adjustment. If the user reports that something bad happened during the last adjustment, the advisor adjusts the damper back to the last position and moves on to the next damper. If bad

REDUCTION RATIO 14:1

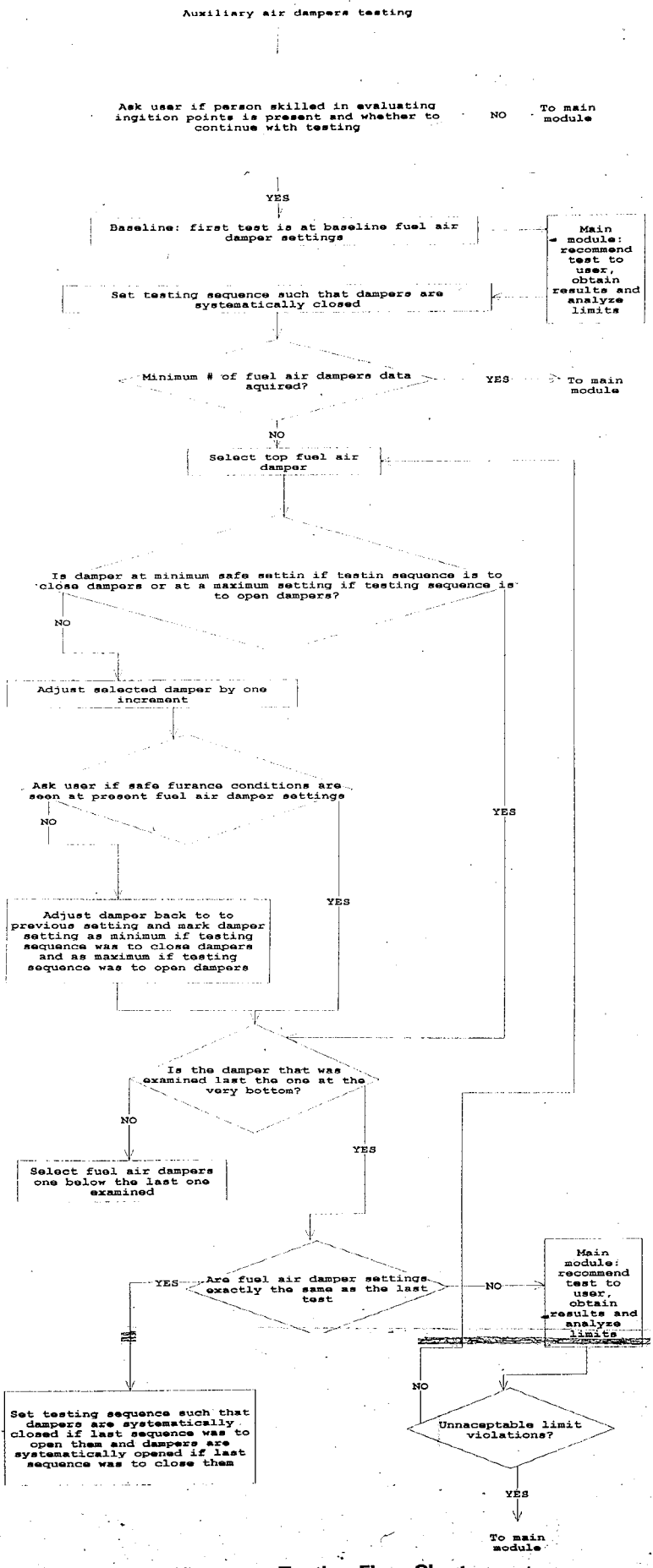

Figure 24 **Fuel Air Dampers Testing Flow Chart**

REDUCTION RATIO 11:1

furnace conditions were seen at a particular burner with a certain damper position, then further closing of that damper will not be recommended in the future. After all the dampers have been adjusted once and checked, rule ***testing-proceed*** modifies the ***fuel-air*** slot of the *recommend* object with the new damper positions that yielded safe furnace conditions and control of the program passes on to the ***main*** module where a real test will be recommended.

A typical damper-checking interaction between the user and the advisor would look like this :

(baseline fuel air dampers are set at 4-4-4-4 starting from the top level)

Advisor: Please adjust fuel air dampers to 3-4-4-4 and observe the furnace.

User: Furnace quality is good.
Ignition points are ok.

Advisor: Please adjust fuel air dampers to 3-3-4-4 and observe the furnace.

User: Furnace quality is good.
Ignition points are ok.

Advisor: Please adjust fuel air dampers to 3-3-3-4 and observe the furnace.

User: Furnace quality is ok.
Ignition points are NOT ok.

Advisor: Please adjust fuel air dampers to 3-3-4-3 and observe the furnace.

User: Furnace quality is bad.
Ignition points are NOT ok.

At this point the advisor would recommend the user to run a test where the fuel

air damper positions are set at 3-3-4-4.

Rule ***testing-stop*** is activated if fuel air dampers can no longer be adjusted in the current testing direction, or if some limit violation is observed. If the reason is the former, and if the minimum number of fuel air tests has not yet been performed, then the testing direction is reversed and testing begins all over again. But, if limit violation has been observed or the minimum number of test points has been acquired, rule ***close-up*** is activated. Rule ***close-up*** modifies the ***par-eval*** slot of the *recommend* object, removes any unnecessary facts and hands control of the program back to the ***main*** module.

Burner-tilt-auxiliary-air-damper Module

The ***burner-tilt-auxiliary-air-damper*** module is activated only if both burner tilt angle and secondary air damper modules are parameters that have already been optimized by the advisor and the goal of the optimization exercise is to minimize heat rate with a target on NO_x. This module makes use of what has happened during the secondary air damper parametric testing and also uses some of the functions in the ***secondary-air*** module. For this reason, the ***secondary-air*** module is visible to the ***burner-tilt-auxiliary-air-damper*** module.

The experimental design employed in the ***burner-tilt-auxiliary-air-damper*** module is to test a narrow range of control settings around the optimum values of burner tilt angle and auxiliary air bias found in previous parametric tests. The rules and facts in the ***burner-tilt-auxiliary-air-damper*** module address three

tasks; one set of rules determines the six combinations of burner tilt angles and oxygen levels to be tested (three tilt settings and two auxiliary air dampers); one set of rules and facts chooses the operating settings to be tested in the next experiment; and the last set of rules evaluates any limit violation and accordingly adjust the rest of the combinations of burner tilt angles and auxiliary air damper settings.

Initial facts in the ***burner-tilt-auxiliary-air-damper*** module declare the range of burner tilt angles to be tested to twelve degrees. Using this fact and facts stating the highest and lowest burner tilt angles tested in the parametric module, ***generate-combinations*** picks three points each for burner tilt around the optimum setting determined by the parametric testing module. The two auxiliary air bias settings to be tested are the lowest and the highest safe biases tested in the ***secondary-air*** module. ***Form-combinations*** forms six different paired data sets of O_2 and burner tilt from these points.

Start-test is fired when control is first handed to the ***burner-tilt-auxiliary-air-damper*** module, and it initiates the rules that form the paired data sets of auxiliary air bias and burner tilt angle (refer to Figure 25 for flow chart). ***Start-test*** also modifies the slot ***par-eval*** of the *recommend* object to ***tilt-sec-air*** to indicate that the following test series is for combinations of auxiliary air bias and burner tilt angle.

Finding-aux then selects one of the combinations of these paired data and activates the ***secondary-air*** module so that damper combinations that yield

the auxiliary air bias specified in the combination of tilt and auxiliary air bias settings to be tested next are selected (refer to Section on the ***secondary-air*** module for detailed explanation). Effort is made to test combinations that were not tested previously if more than two combinations that satisfy the selected bias are found (rule ***choose-dampers-not-tested*** accomplishes this task.)

Rule ***prep-experi*** is activated only if there have not been any limit violations or the violations have been dealt with by the third set of rules that evaluate violations (third set of rules are discussed below). ***Prep-experi*** selects one of the paired data mentioned above and modifies the object with *recommend* ***status*** to hold these values in the ***auxiliary-air-damper*** and ***burner-tilt*** slots. The control is then handed over to the ***main*** module that recommends the testing of the selected points to the user.

The set of rules that deal with limit violations make use of two kinds of facts. The first kind deal with common sense knowledge and were described in detail in the ***O_2-burner-tilt*** module. The second kind deal with specific knowledge on boiler variables and the facts that deal with burner tilt angle are again listed in the ***O_2-burner-tilt*** module. Facts that deal with auxiliary air bias and any of the relevant parameters whose limits might be violated as a result of bias manipulations were not included since these relationships have not yet been well established. For instance, the effect of auxiliary air bias on steam temperatures is not known.

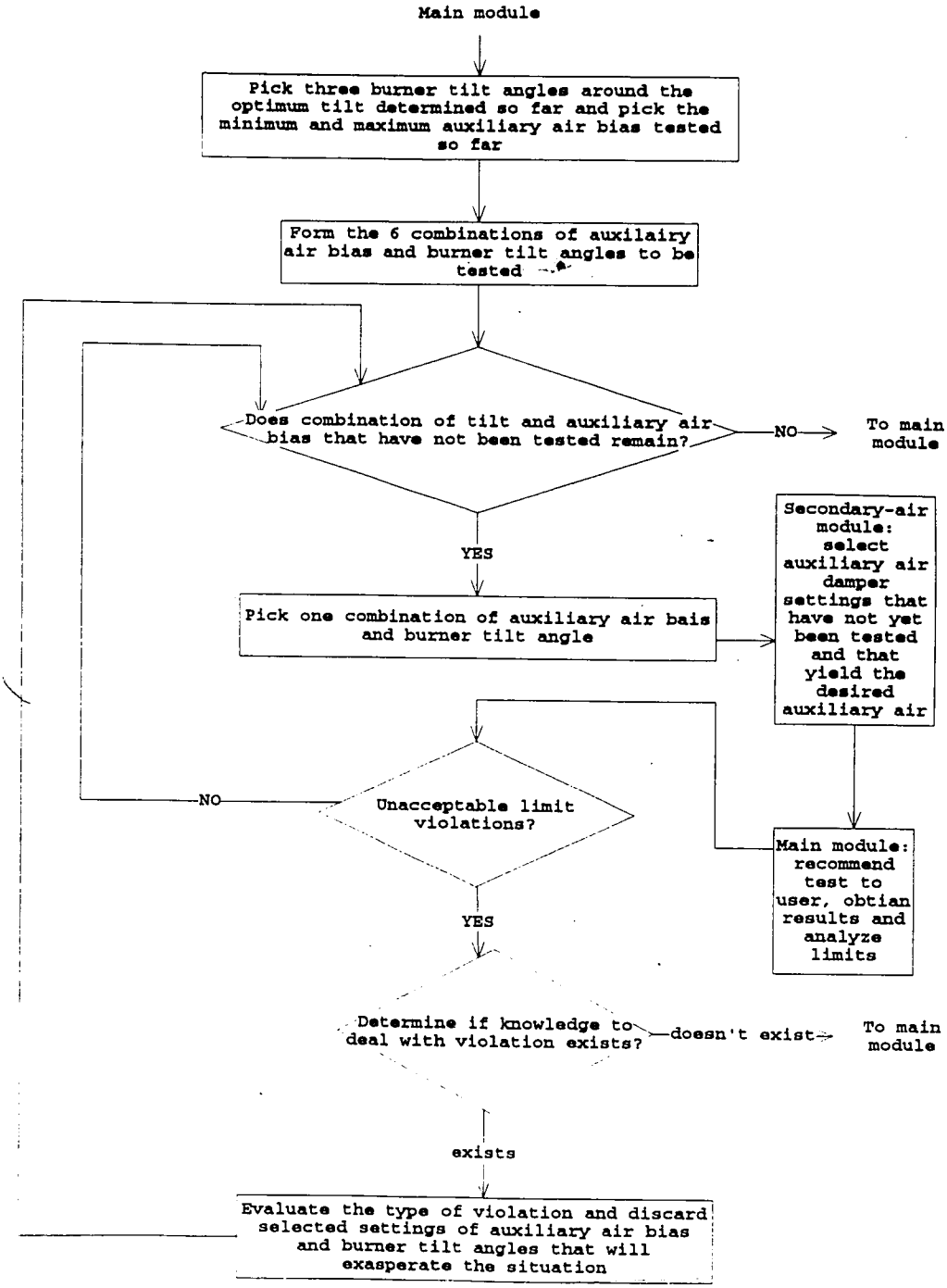

Figure 25 **Burner-tilt-auxiliary-air Module**

Rules ***action-on-violations, analyze-facts-1*** and ***analyze-facts-2*** use the above facts to deduce what the proper action for a particular violation should be. For instance, if main steam temperatures became too low, then these rules conclude that the proper action is to raise the burner tilt angle. Rules ***evaluate-violations-tilt-low***, and ***evaluate-violations-tilt-high*** remove combination of auxiliary air bias and burner tilt settings from the selected list depending on what the proper action was concluded to be. For instance, if the proper action was found to be to raise the burner tilt setting, then ***evaluate-violation-tilt-low*** removes paired data containing burner tilt settings lower or equal to the one tested last from the paired sets that have not been tested yet.

If there has been a limit violation and the set of rules that deal with violations can't handle this particular problem, or all the selected paired tilt and auxiliary air bias set points have been tested, rule ***finish-experi*** is fired. ***Finish-experi*** modifies the ***par-eval*** slot of the object with the ***status*** *recommend* to *optimized-par* to indicate to the ***main*** module that the factorial test of tilt and auxiliary air bias is finished. Control of the program is then handed back to the ***main*** module since all the data analysis of this module is handled by the neural network-optimization package at the conclusion of all testing.

Results

While the expert advisor is not ready to be tested in an environment of its intended use, as an initial check, correlations of data from parametric tests at Potomac River Unit 4, and functions created to approximate behavior of the variables were used to run the program. Equations 7, 8, 9 and 10 show correlations of the parametric data gathered from Potomac River Unit 4 for full load operation [18].

$$NO_x = 0.182 + 0.128.O_2 + 3.44.10^{-4}.tilt^2 - 0.00653.tilt + 0.0192.O_2.tilt \quad (7)$$

$$Main\ steam\ temperature = 1141.43 - 138.662.O_2 + 26.893.O_2^2 + 1.105.O_2.tilt \quad (8)$$

$$Reheat\ steam\ temperature = 1121.07 - 154.639.O_2 + 30.6.O_2^2 + 1.121.O_2.tilt \quad (9)$$

$$LOI = 17.4398\ O_2^{-1.0348} \quad (10)$$

The NO_x equation was artificially adjusted to show influences of β, α and fuel air dampers. Functions that approximated the behavior of CO and

opacity (see Figure 7 and Section on knowledge base) were created and used in trial runs. Furnace oxygen was approximated by adjusting the economizer oxygen setting downward by 1.4 percent to account for convective pass in-leakage. Other parameters were disabled for the trial runs. Figure 26 and 27 show variations of main steam temperature and reheat steam temperatures with burner tilt angle for one trial run. These curves are not smooth since the temperatures vary with O_2 which was varied in the trial run. Figures 28, 29 and 30 show variations of CO, opacity and LOI with O_2 for the same trial run.

The advisor was run for different optimization goals, system limits, and initial conditions. Three trial runs were analyzed: trial run 1 had an optimization goal of minimizing heat rate with a constraint of 0.41 lb/MBtu on NO_x. All the control parameters were tested in trial run 1. Trial run 2 had an optimization goal of minimizing NO_x, with testing performed on all the control parameters. Trial run 3 had an optimization goal of minimizing heat rate with a constraint of 0.41 lb/MBtu on NO_x. Testing was performed only on economizer oxygen, mills and auxiliary air dampers for this case. Recommended trial settings in all the trial runs were found to correspond to those in the knowledge base and to the experimental procedures used in the design of the advisor. The advisor warned the user every time there was a violation in one of the safety and performance parameters. Test runs were reviewed by ERC engineers and approved as tests they would have recommended to achieve the same goal.

Figures 31, 32 and 33 show NO_x versus test numbers for the three trial

runs, and a general trend of experimentation towards decreasing NO_x can be seen. Trial 1 obtained the largest number of data points in accordance with optimizing two dependent variables, and trial 3 had the fewest number of data points because three instead of five parameters were tested.

Figures 34, 35, and 36 show NO_x versus oxygen from data collected during parametric testing of economizer oxygen. The settings where opacity violations occurred are indicated and there are no oxygen data below these points. Note that the user did not accept the opacity violation at a higher oxygen setting in trial 3. Figures 37, 38 and 39 show the oxygen settings as testing proceeded. In all three trial runs, economizer oxygen parametric testing was done first and the first test was repeated once the lowest possible O_2 was tested. Then, economizer oxygen was set at 0.2% above the lowest acceptable O_2 setting for the remainder of the testing (note that the lowest O_2 tested was not acceptable). For the first trial only, O_2 was varied in one more series of testing in a factorial oxygen and burner tilt testing procedure. This procedure was not performed in trial 2 since the optimization objective was to just minimize NO_x, and it was not done in trial 3 since the burner tilt was locked in position.

Figures 40 and 41 show NO_x and burner tilt angle for trial runs 1 and 2, and the angle that gave the minimum NO_x can be seen to be around 4°. Figures 42 and 43 show burner tilt recommendations as testing proceeded. After the parametric burner tilt testing was performed, for tests where burner tilt

was not varied, it was set at the minimum angle of 4°. In trial run 1, where the optimization objective was to minimize heat rate and NO_x, factorial experiments on O_2 and burner tilt, and factorial experiments on burner tilt and auxiliary air dampers were performed.

Figures 44, 45, and 46 show mill biases as testing proceeded. The biases in all three trials were set at 0 before parametric mill testing was done, and then they were set at the minimum values possible once downloading mills was found to produce significant reduction in NO_x (see Figures 47, 48 and 49.) Trials 1 and 3 show a few distinct β values, while trial 2 shows β values just around 0 and at minimum β. These two different patterns occured for the following reasons. When the objective is to optimize both NO_x and heat rate, a few distinct β points are needed to learn the real relation of β with NO_x and heat rate. Conversely, when the objective is to just minimize NO_x, the only information needed is whether downloading mills as much as possible produces a significant reduction in NO_x.

Figures 50, 51 and 52 show auxiliary air bias settings as testing proceeded. The last test in each of the parametric testing series was a repeat of the first test to correct for boiler slagging. After the parametric test in trial run 1, the auxiliary bias setting was returned to the original bias since the target NO_x condition of 0.41 was already achieved (see Figure 53). In trial 2, even though NO_x was at similar levels as in trial 1 (see figure 54), auxiliary bias was set at the highest settings after the parametric test, since the objective is to find

the lowest NO_x possible. In trial 3 (Figure 55), replicate auxiliary air bias settings were obtained since bad furnace quality was reported and the advisor recommended different damper combinations that result in the same bias. Tables 1, 3 and 5 list the different auxiliary air damper setting combinations tested. Tables 2 and 4 show fuel air dampers recommended for trial runs 1 and 2. Damper settings for both the auxiliary air and fuel air dampers are listed starting with the top most damper on the boiler. In trial run 3, the user declined to run fuel air damper tests.

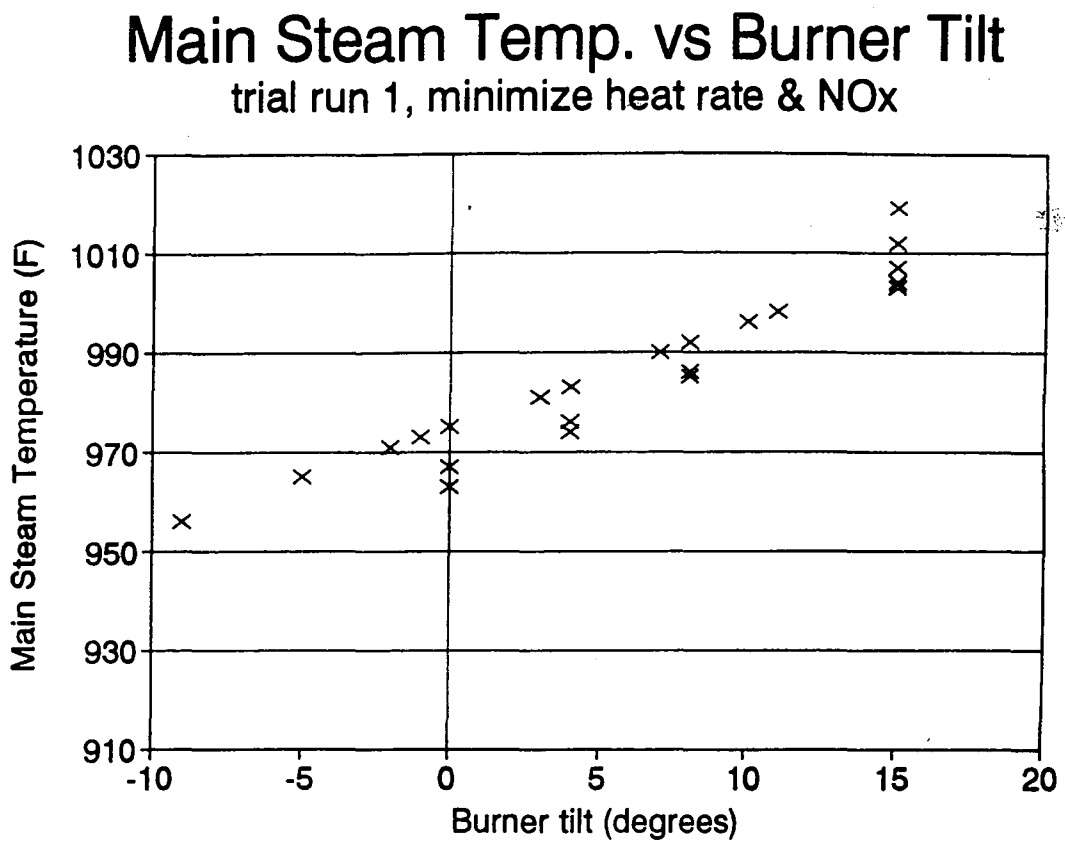
Main Steam Temp. vs Burner Tilt
trial run 1, minimize heat rate & NOx
Main Steam Temperature (F)
1030
1010
990
970
950
930
910
-10
-5
0
5
10
15
20
Burner tilt (degrees)

Figure 26

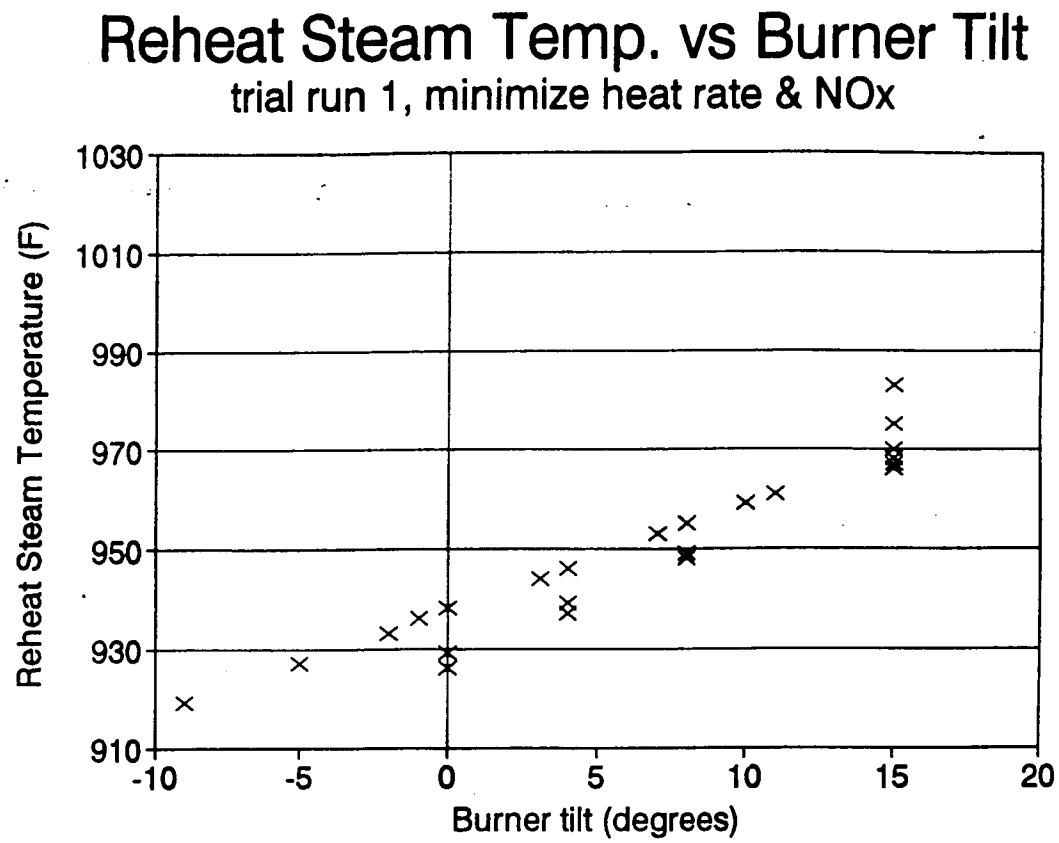
Reheat Steam Temp. vs Burner Tilt
trial run 1, minimize heat rate & NOx
Reheat Steam Temperature (F)
1030
1010
990
970
950
930
910
-10
-5
0
5
10
15
20
Burner tilt (degrees)

Figure 27

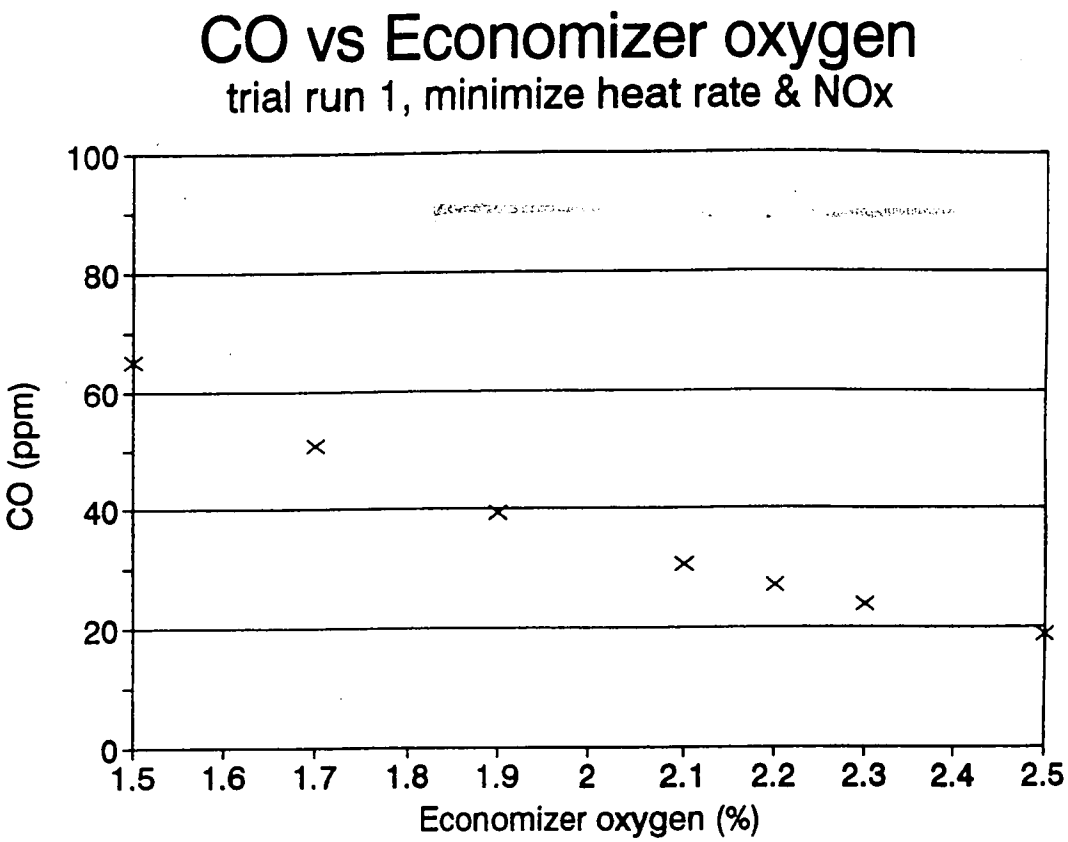
CO vs Economizer oxygen
trial run 1, minimize heat rate & NOx
CO (ppm)
100
80
60
40
20
0
1.5
1.6
1.7
1.8
1.9
2
2.1
2.2
2.3
2.4
2.5
Economizer oxygen (%)

Figure 28

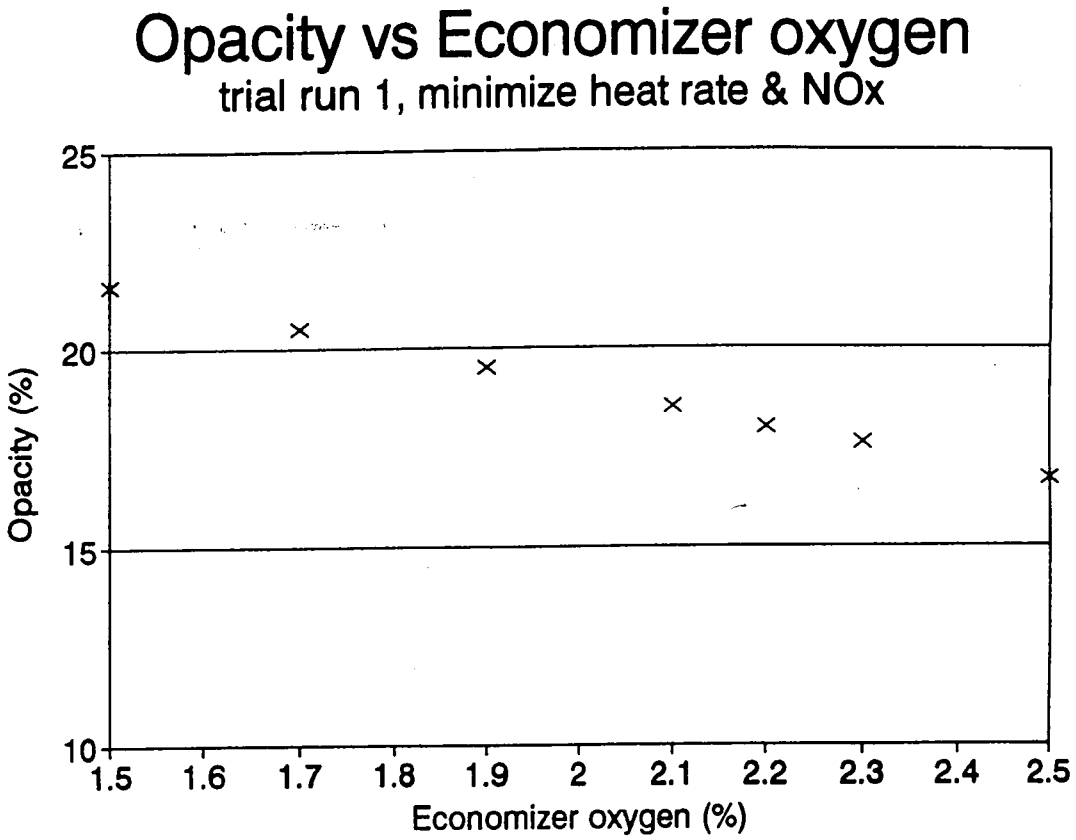
Opacity vs Economizer oxygen
trial run 1, minimize heat rate & NOx
Opacity (%)
25
20
15
10
1.5
1.6
1.7
1.8
1.9
2
2.1
2.2
2.3
2.4
2.5
Economizer oxygen (%)

Figure 29

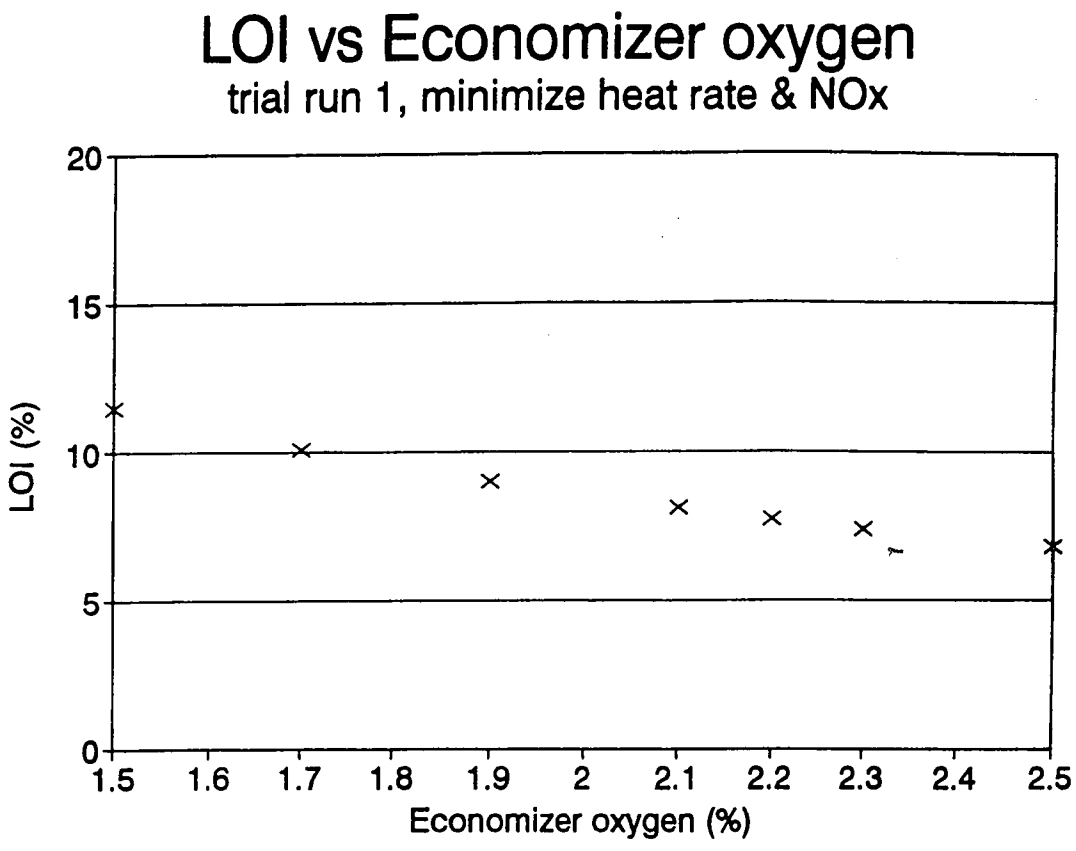
LOI vs Economizer oxygen
trial run 1, minimize heat rate & NOx
LOI (%)
20
15
10
5
0
1.5
1.6
1.7
1.8
1.9
2
2.1
2.2
2.3
2.4
2.5
Economizer oxygen (%)

Figure 30

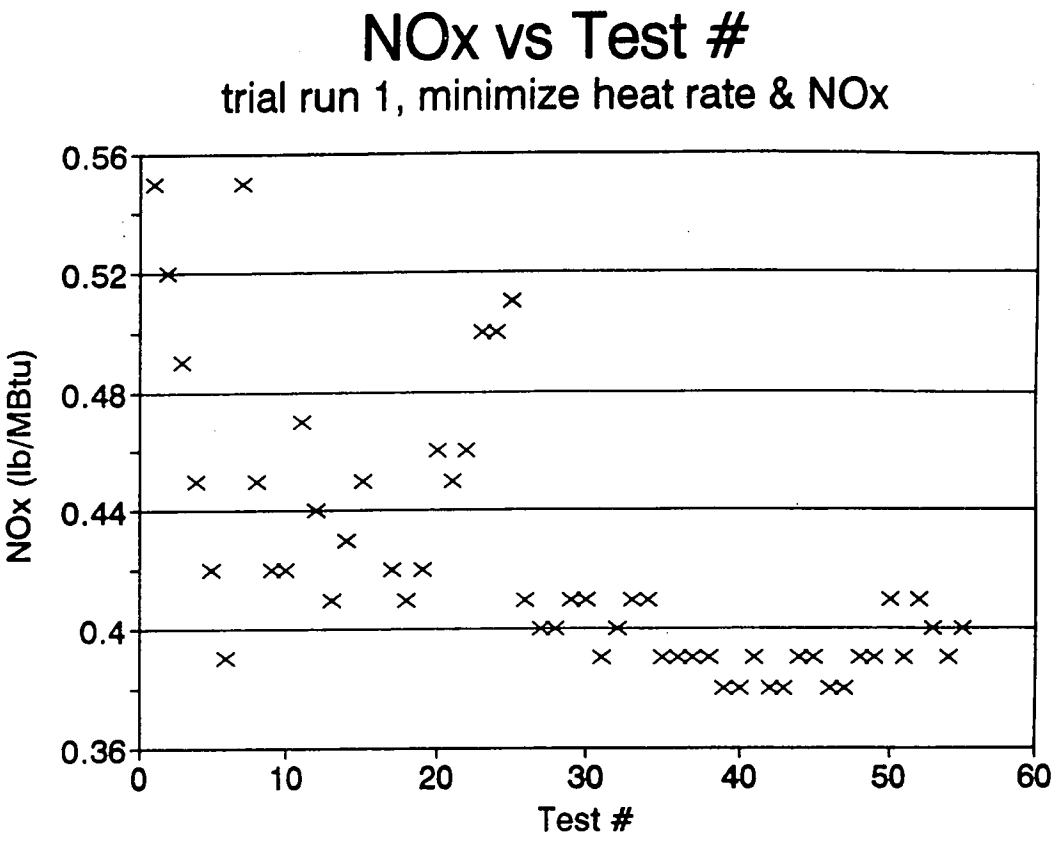
NOx vs Test #
trial run 1, minimize heat rate & NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
0
10
20
30
40
50
60
Test #

Figure 31

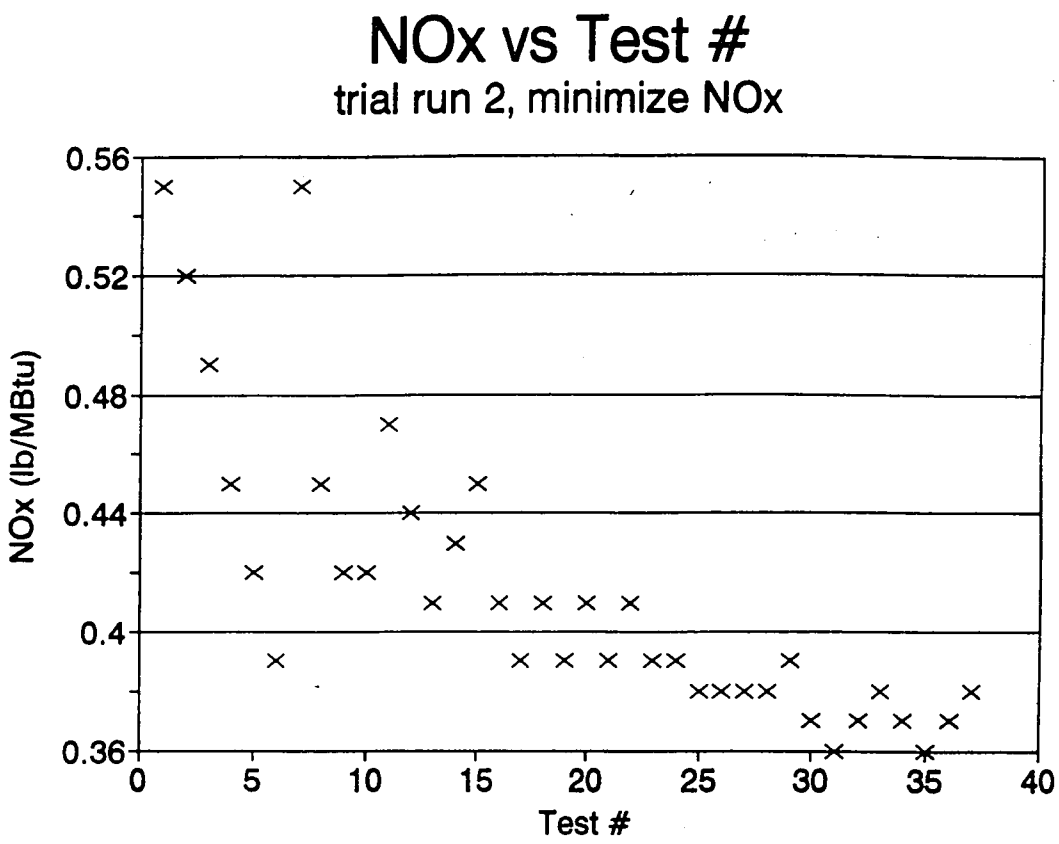
NOx vs Test #
trial run 2, minimize NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
0
5
10
15
20
25
30
35
40
Test #

Figure 32

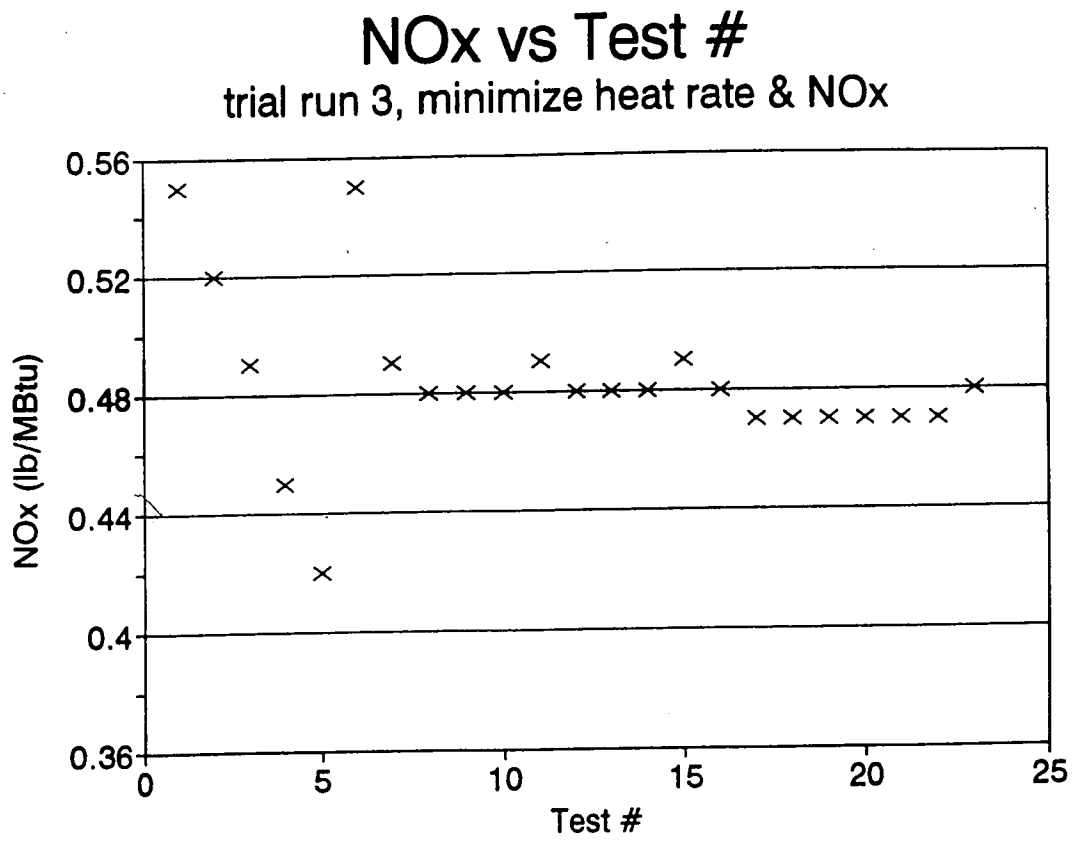
NOx vs Test #
trial run 3, minimize heat rate & NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
0
5
10
15
20
25
Test #

Figure 33

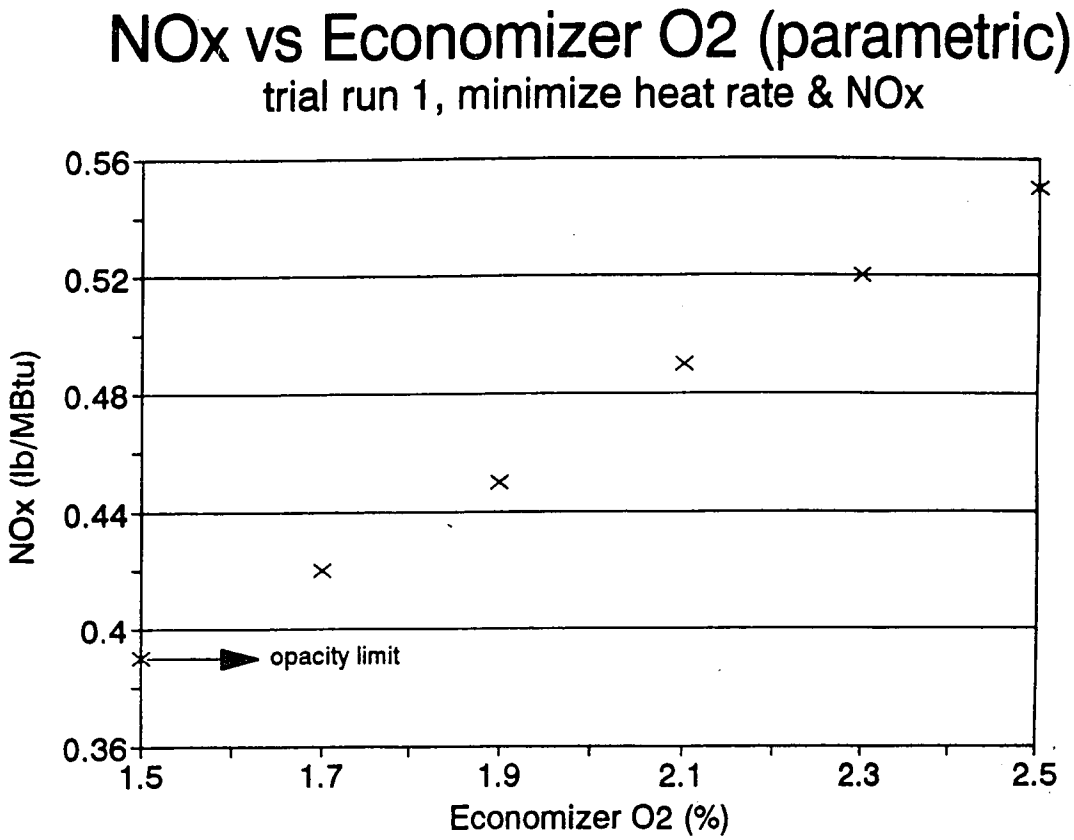
NOx vs Economizer O2 (parametric)
trial run 1, minimize heat rate & NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
opacity limit
1.5
1.7
1.9
2.1
2.3
2.5
Economizer O2 (%)

Figure 34

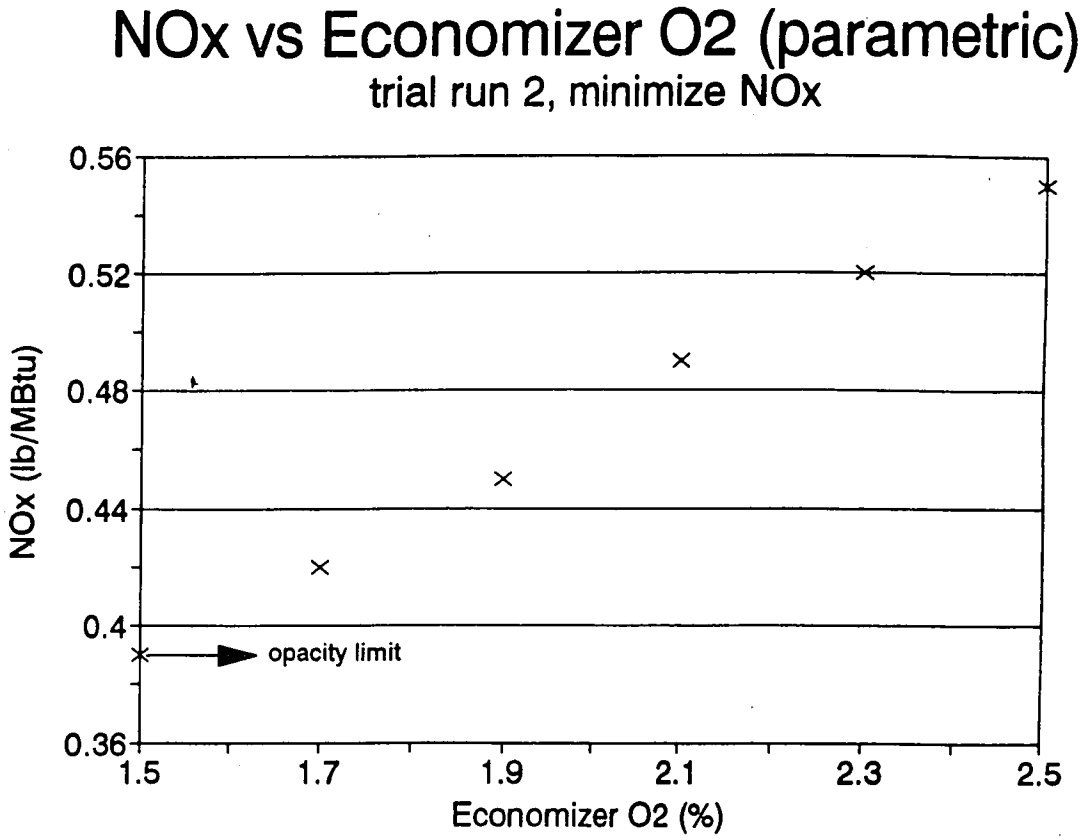
NOx vs Economizer O2 (parametric)
trial run 2, minimize NOx
0.56
0.52
0.48
0.44
0.4
0.36
NOx (lb/MBtu)
opacity limit
1.5
1.7
1.9
2.1
2.3
2.5
Economizer O2 (%)

Figure 35

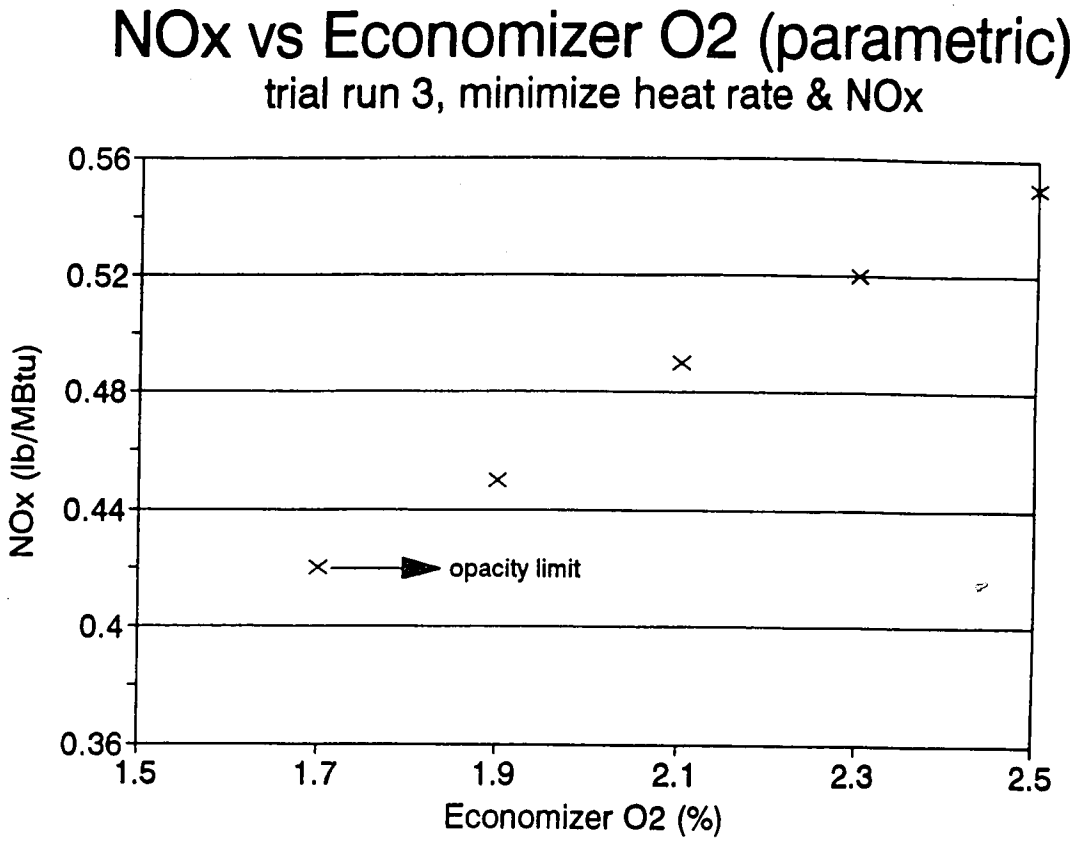
NOx vs Economizer O2 (parametric)
trial run 3, minimize heat rate & NOx
0.56
0.52
0.48
0.44
0.4
0.36
NOx (lb/MBtu)
opacity limit
1.5
1.7
1.9
2.1
2.3
2.5
Economizer O2 (%)

Figure 36

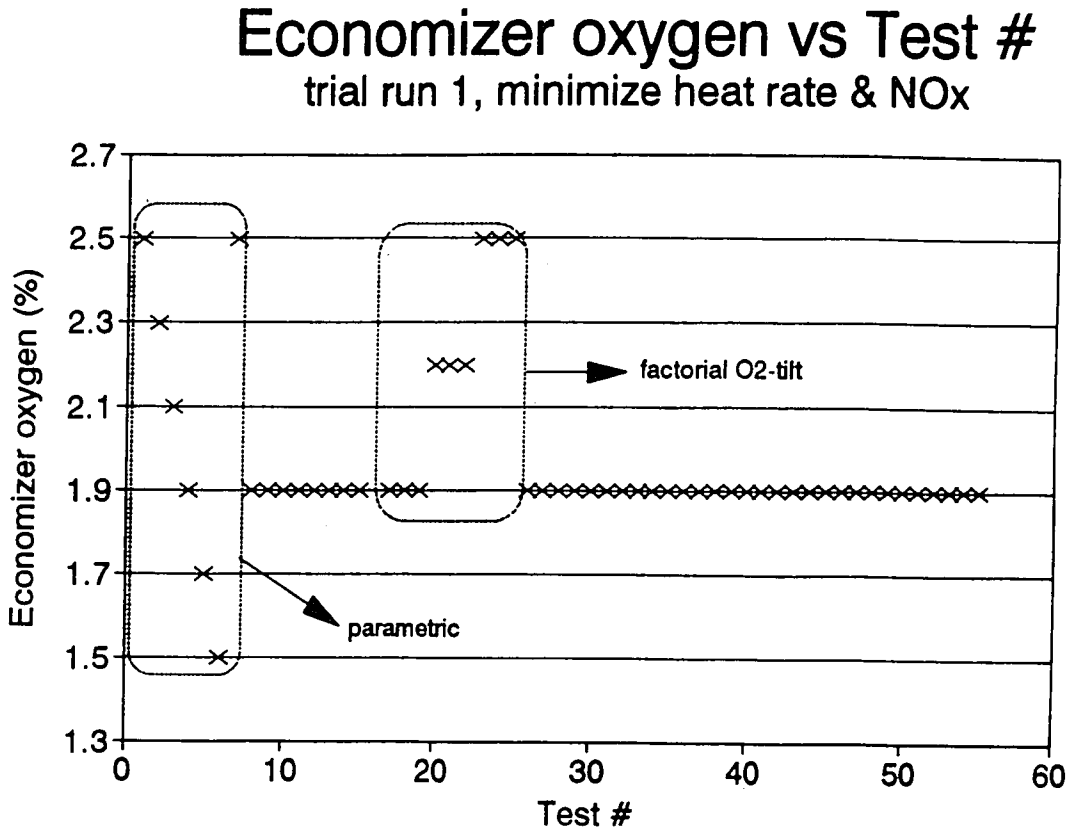
Economizer oxygen vs Test #
trial run 1, minimize heat rate & NOx
Economizer oxygen (%)
2.7
2.5
2.3
2.1
1.9
1.7
1.5
1.3
0
10
20
30
40
50
60
Test #
factorial O2-tilt
parametric

Figure 37

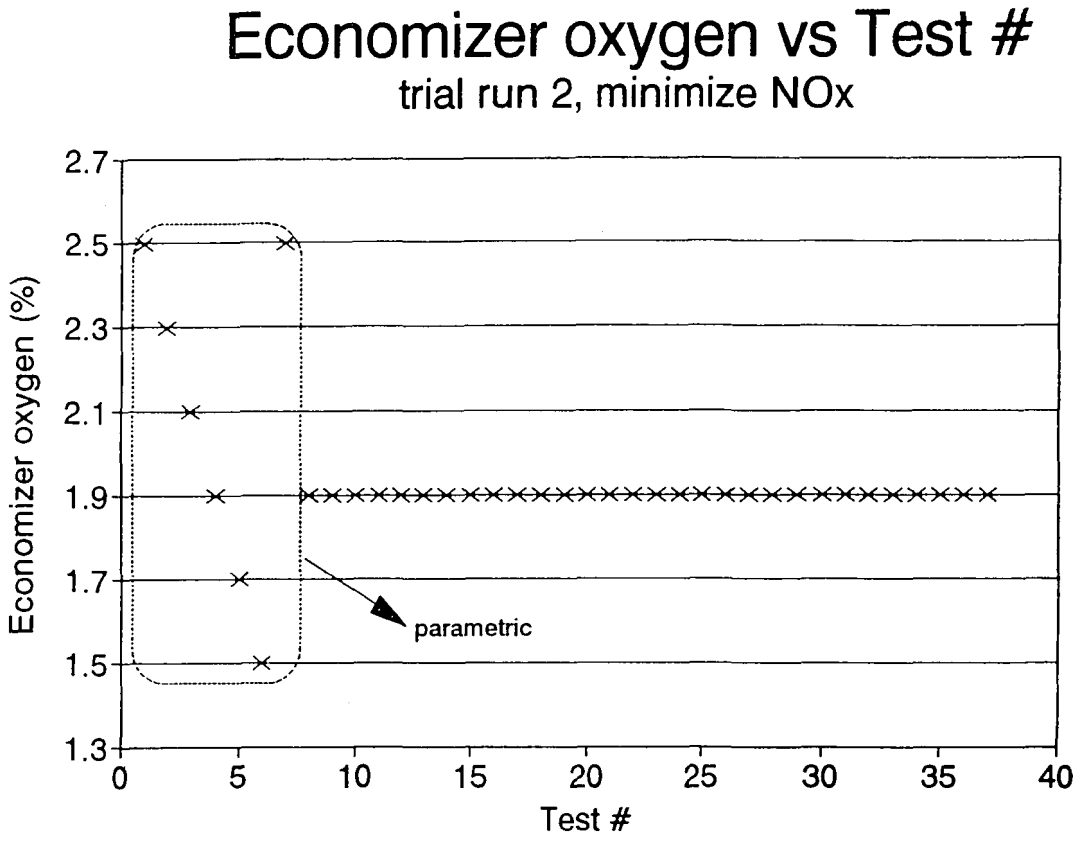
Economizer oxygen vs Test #
trial run 2, minimize NOx
Economizer oxygen (%)
2.7
2.5
2.3
2.1
1.9
1.7
1.5
1.3
0
5
10
15
20
25
30
35
40
Test #
parametric

Figure 38

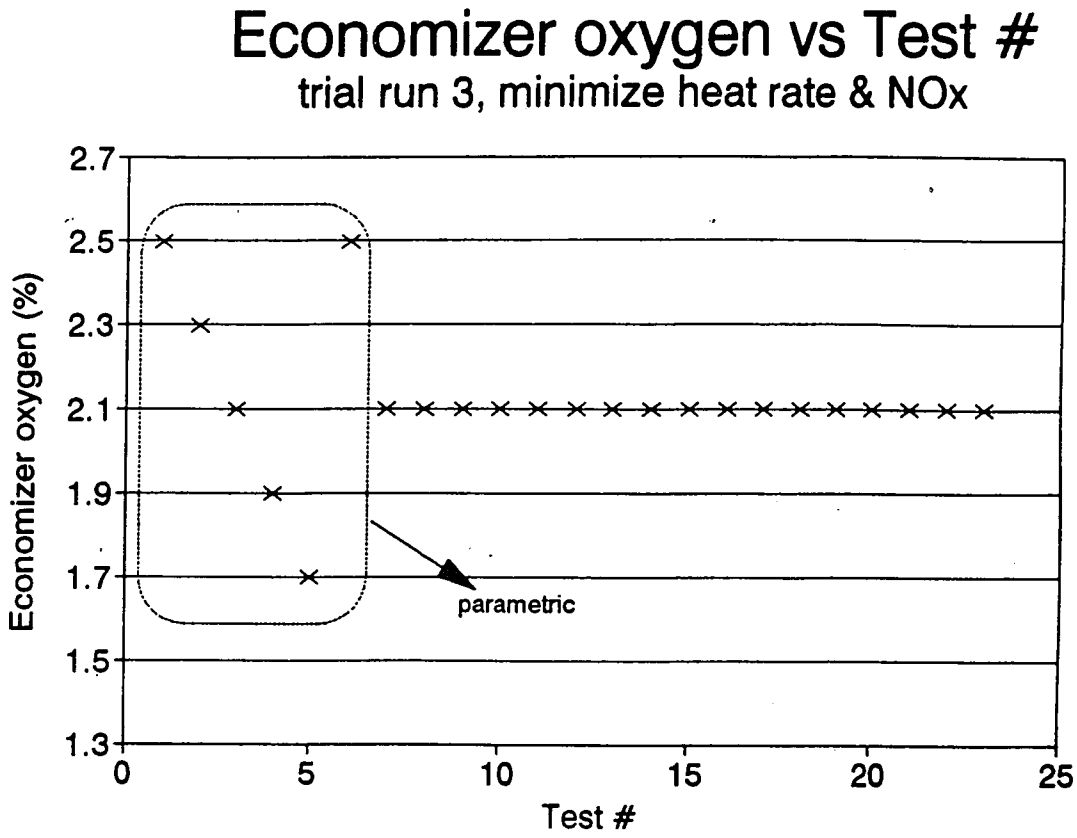
Economizer oxygen vs Test #
trial run 3, minimize heat rate & NOx
Economizer oxygen (%)
2.7
2.5
2.3
2.1
1.9
1.7
1.5
1.3
0
5
10
15
20
25
Test #
parametric

Figure 39

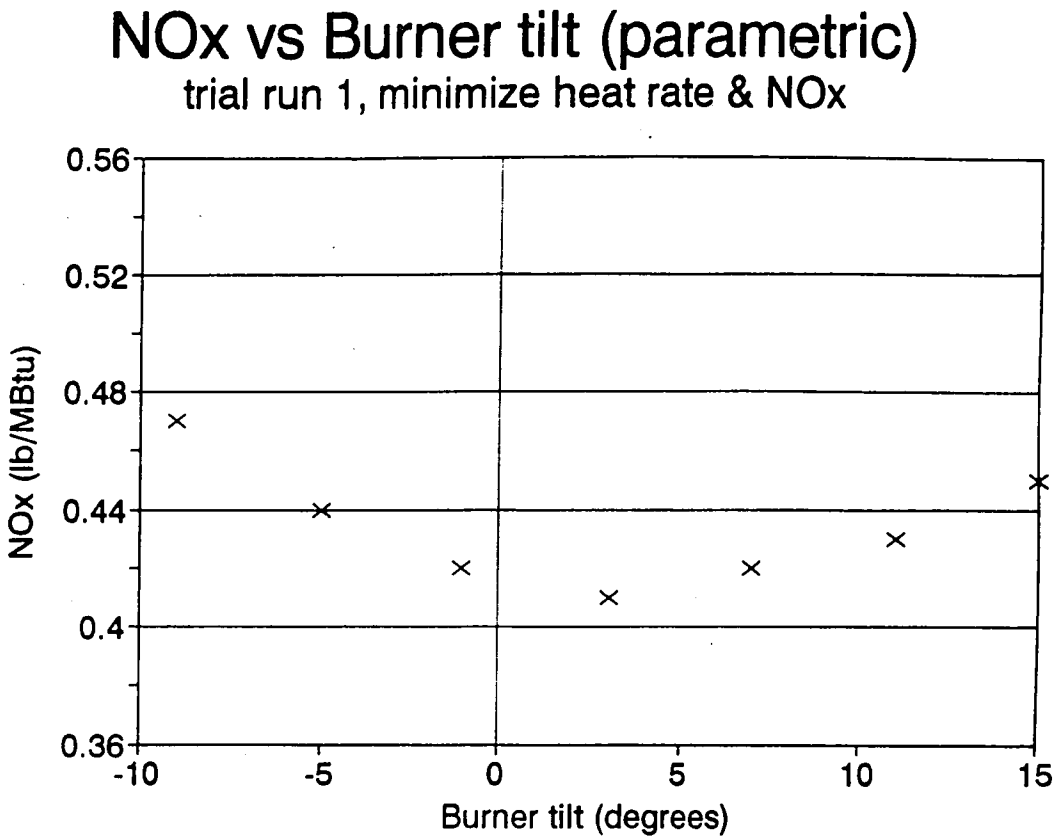
NOx vs Burner tilt (parametric)
trial run 1, minimize heat rate & NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
-10
-5
0
5
10
15
Burner tilt (degrees)

Figure 40

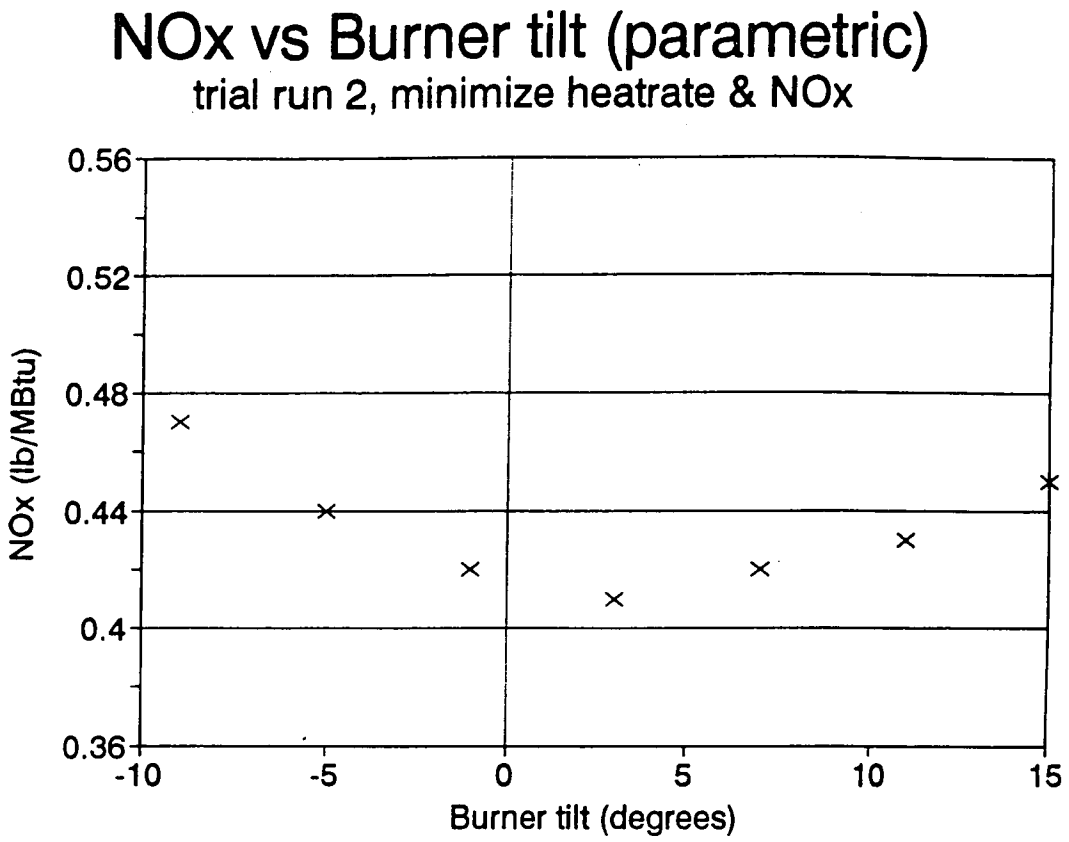
NOx vs Burner tilt (parametric)
trial run 2, minimize heatrate & NOx
0.56
0.52
0.48
0.44
0.4
0.36
NOx (lb/MBtu)
-10
-5
0
5
10
15
Burner tilt (degrees)

Figure 41

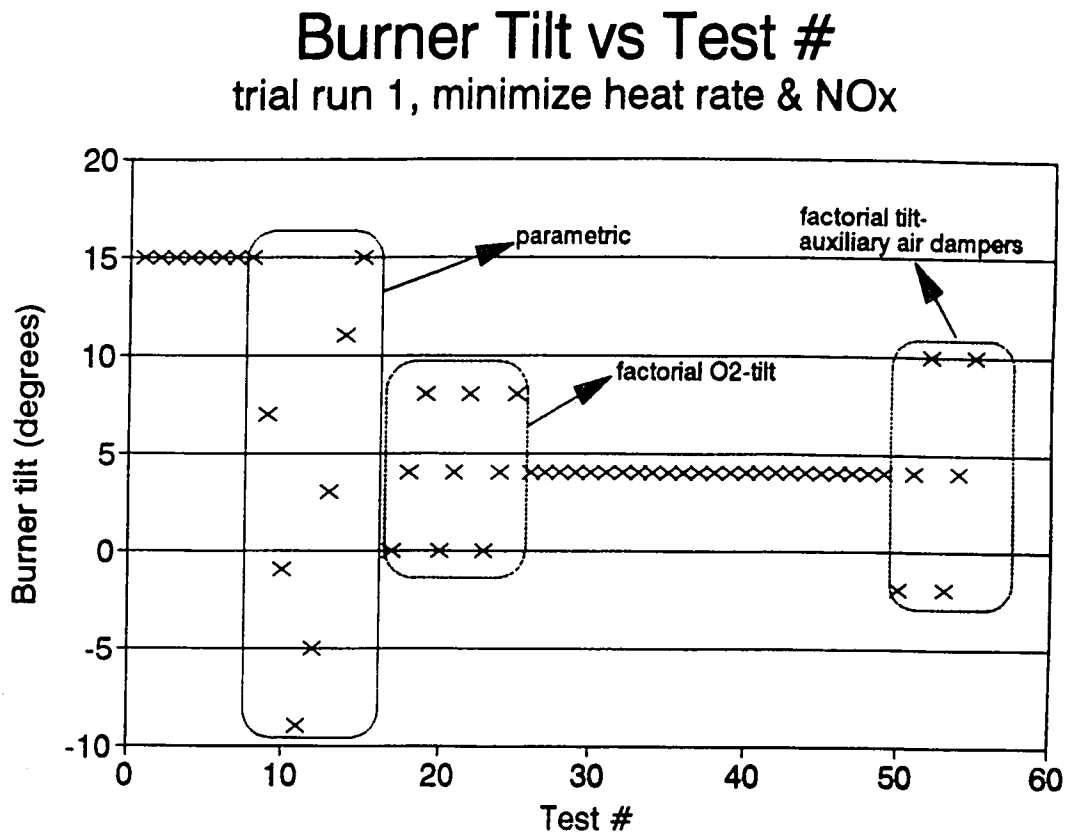
Burner Tilt vs Test #
trial run 1, minimize heat rate & NOx
Burner tilt (degrees)
20
15
10
5
0
-5
-10
0
10
20
30
40
50
60
Test #
parametric
factorial tilt-
auxiliary air dampers
factorial O2-tilt

Figure 42

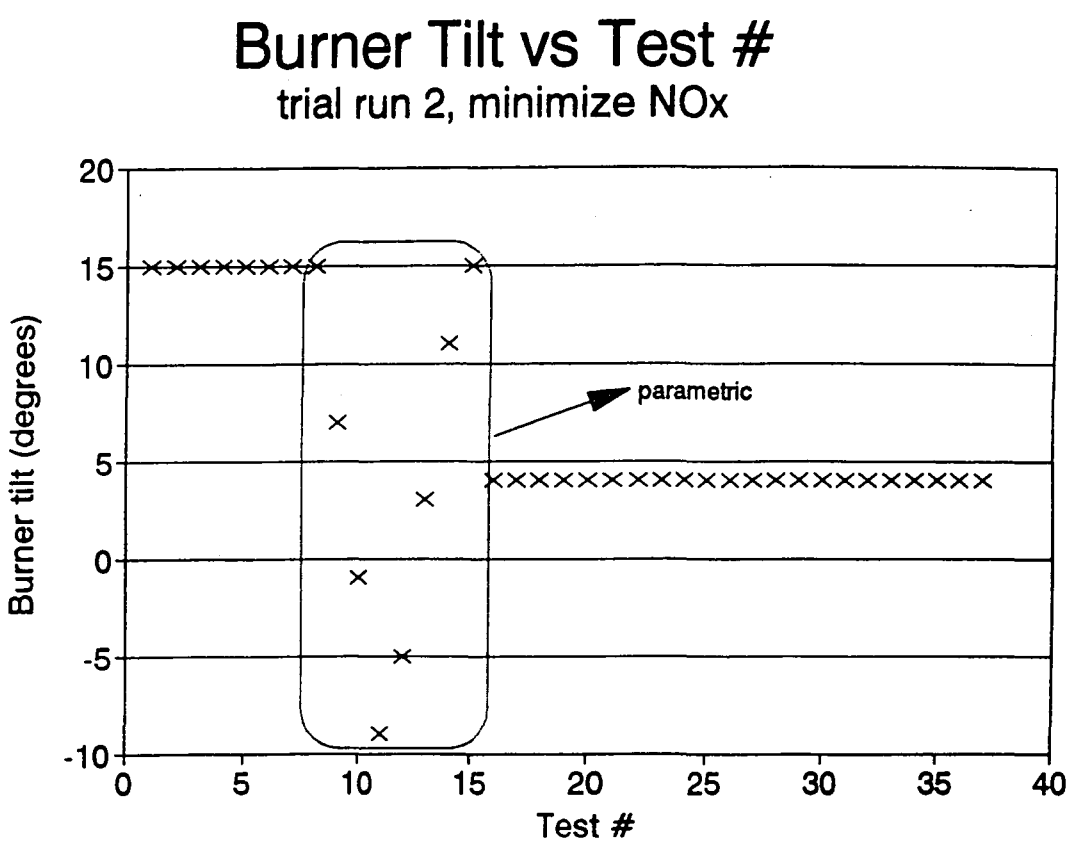
Burner Tilt vs Test #
trial run 2, minimize NOx
Burner tilt (degrees)
20
15
10
5
0
-5
-10
0
5
10
15
20
25
30
35
40
Test #
parametric

Figure 43

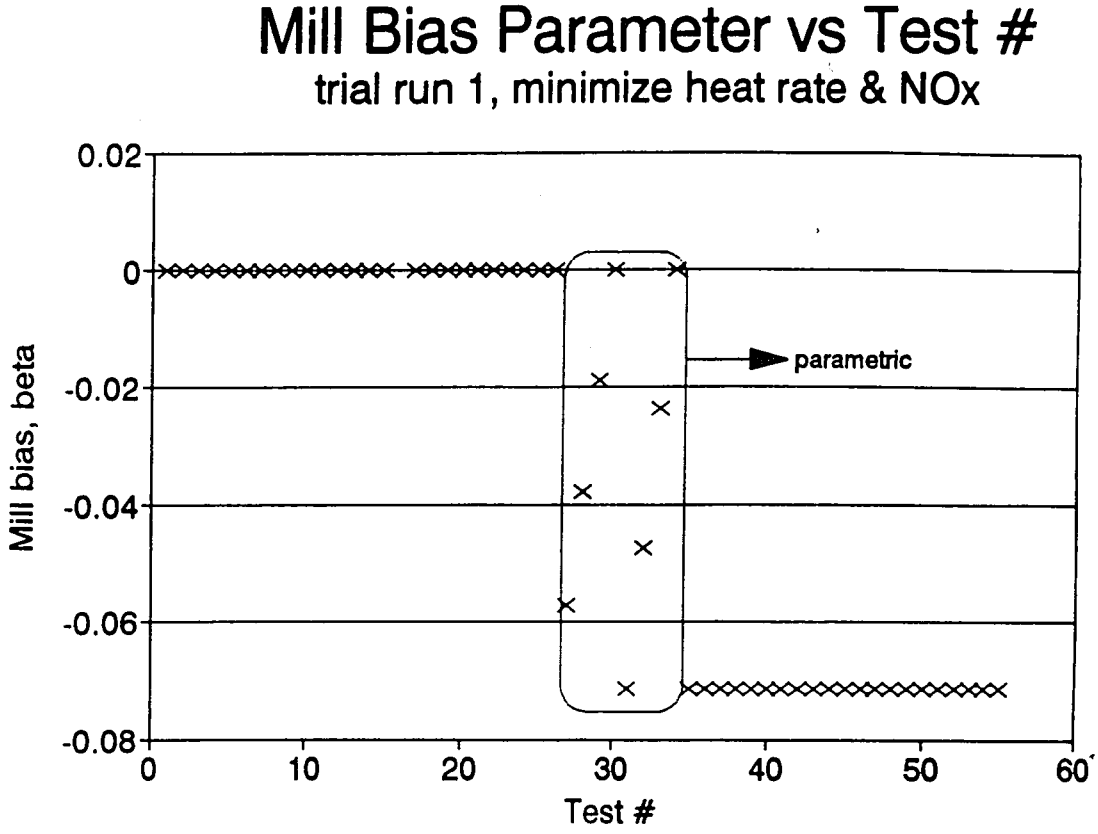
Mill Bias Parameter vs Test #
trial run 1, minimize heat rate & NOx
Mill bias, beta
0.02
0
-0.02
-0.04
-0.06
-0.08
parametric
0
10
20
30
40
50
60
Test #

Figure 44

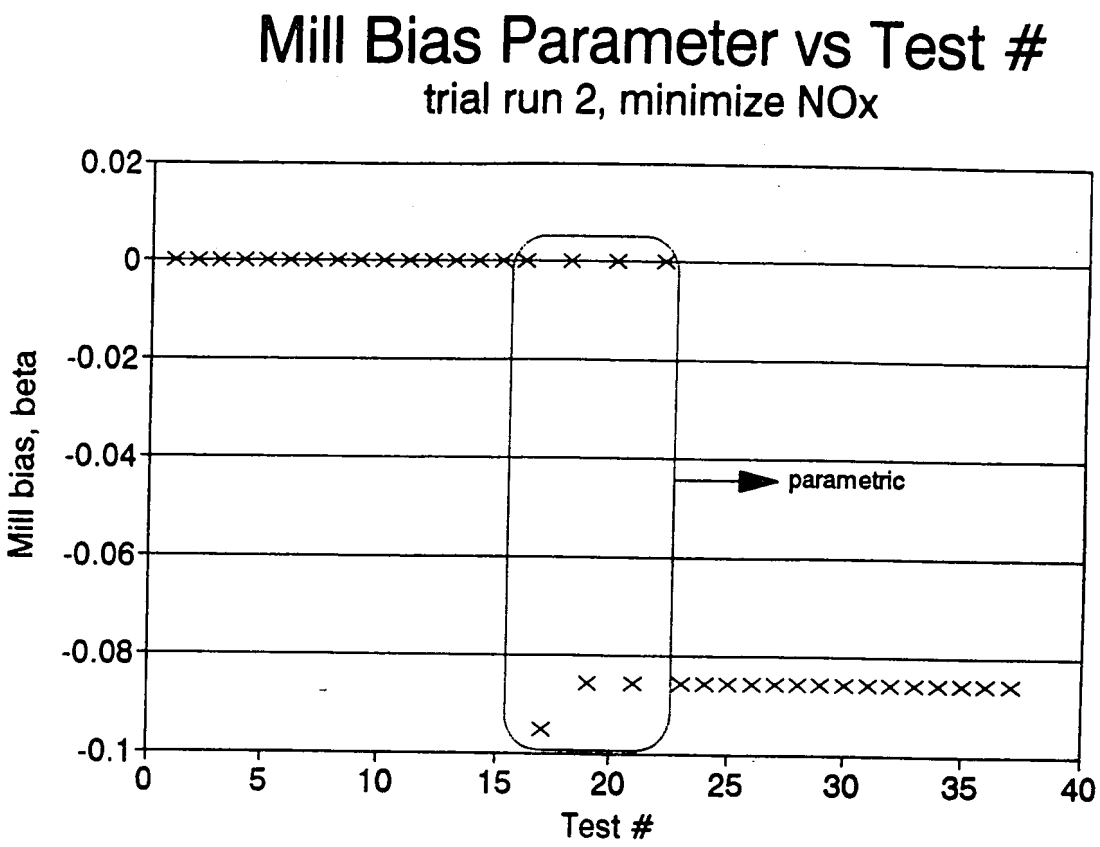
Mill Bias Parameter vs Test #
trial run 2, minimize NOx
Mill bias, beta
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
parametric
0
5
10
15
20
25
30
35
40
Test #

Figure 45

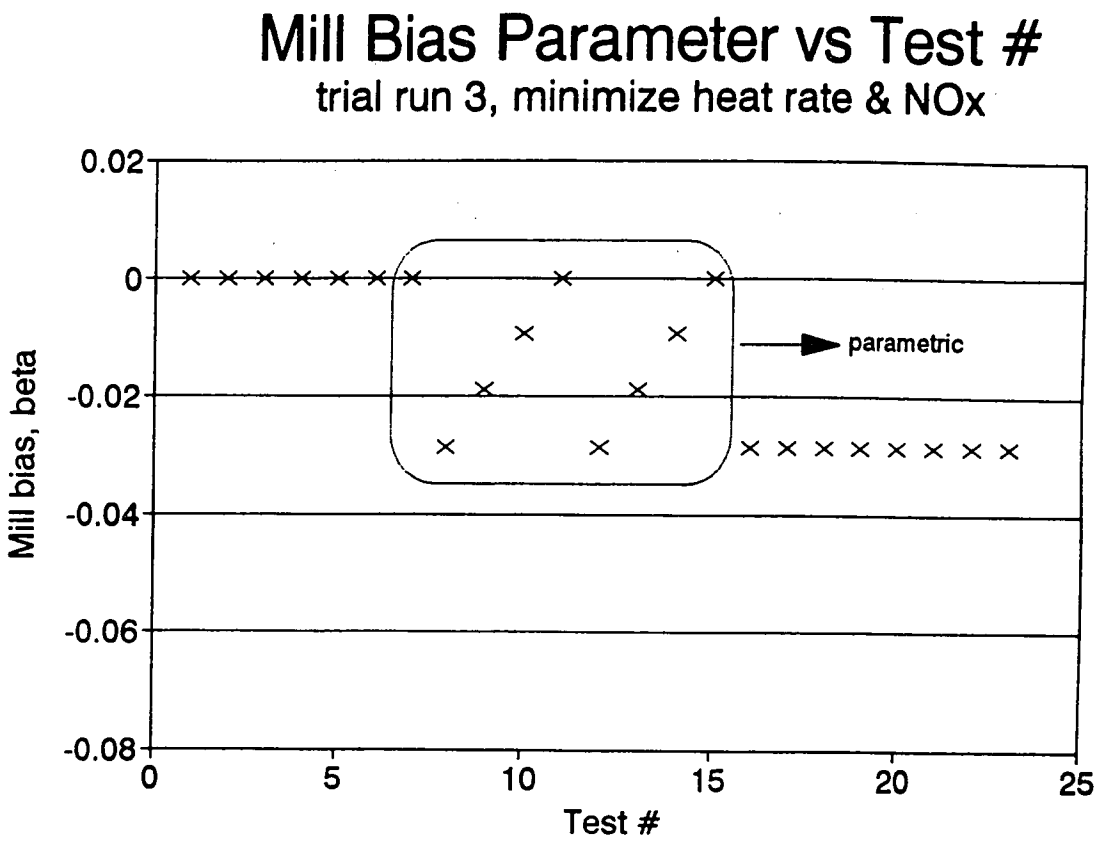
Mill Bias Parameter vs Test #
trial run 3, minimize heat rate & NOx
Mill bias, beta
0.02
0
-0.02
-0.04
-0.06
-0.08
parametric
0
5
10
15
20
25
Test #

Figure 46

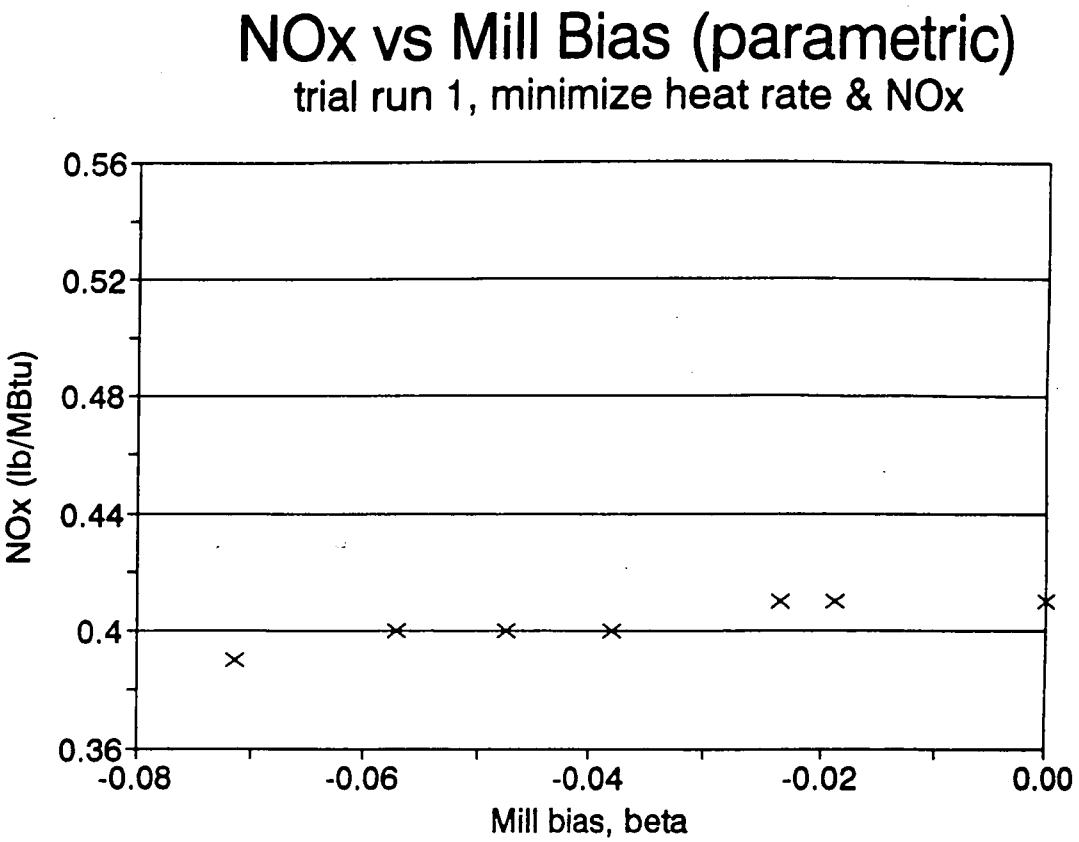
NOx vs Mill Bias (parametric)
trial run 1, minimize heat rate & NOx
0.56
0.52
0.48
0.44
0.4
0.36
NOx (lb/MBtu)
-0.08
-0.06
-0.04
-0.02
0.00
Mill bias, beta

Figure 47

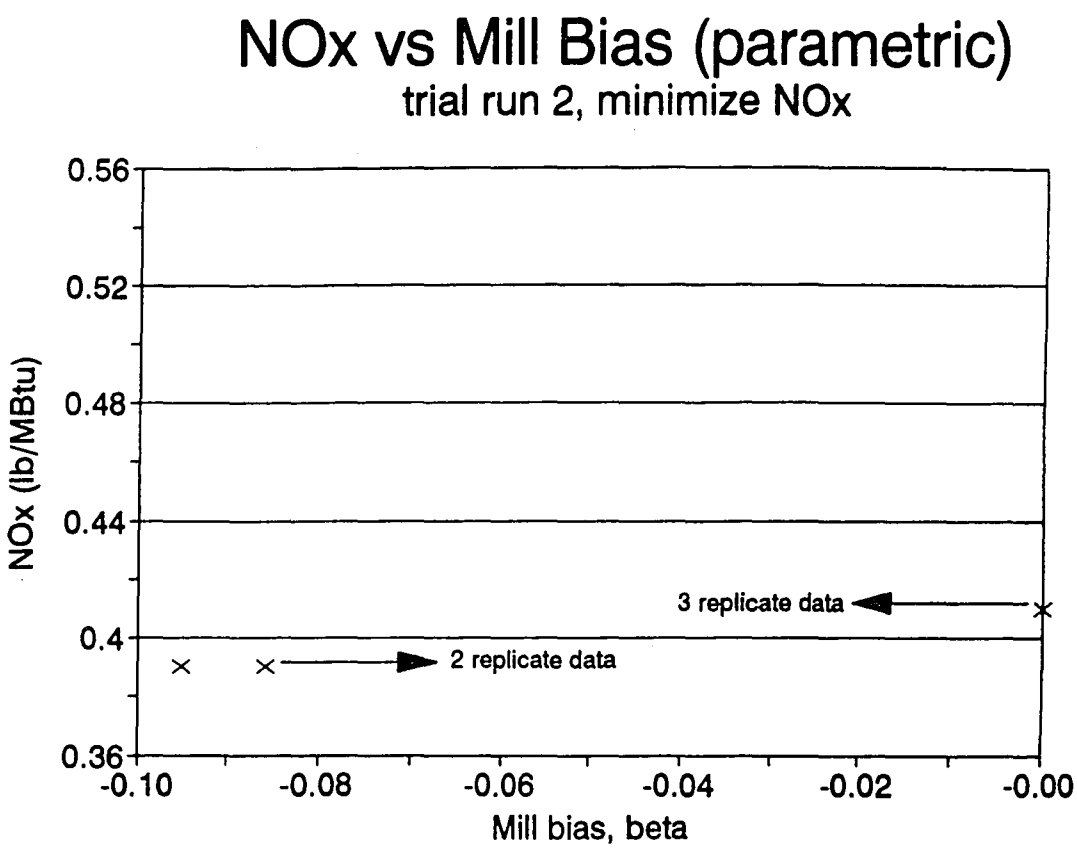
NOx vs Mill Bias (parametric)
trial run 2, minimize NOx
0.56
0.52
0.48
0.44
0.4
0.36
NOx (lb/MBtu)
3 replicate data
2 replicate data
-0.10
-0.08
-0.06
-0.04
-0.02
-0.00
Mill bias, beta

Figure 48

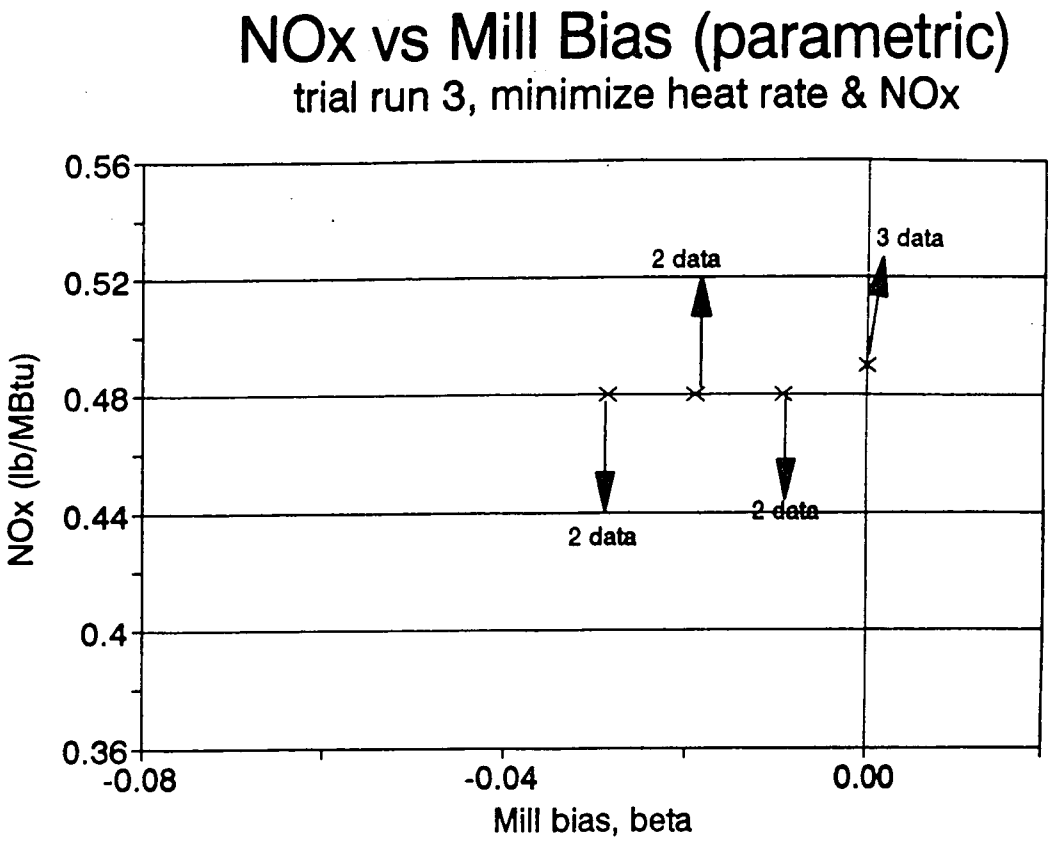
NOx vs Mill Bias (parametric)
trial run 3, minimize heat rate & NOx
0.56
0.52
0.48
0.44
0.4
0.36
NOx (lb/MBtu)
2 data
3 data
2 data
2 data
-0.08
-0.04
0.00
Mill bias, beta

Figure 49

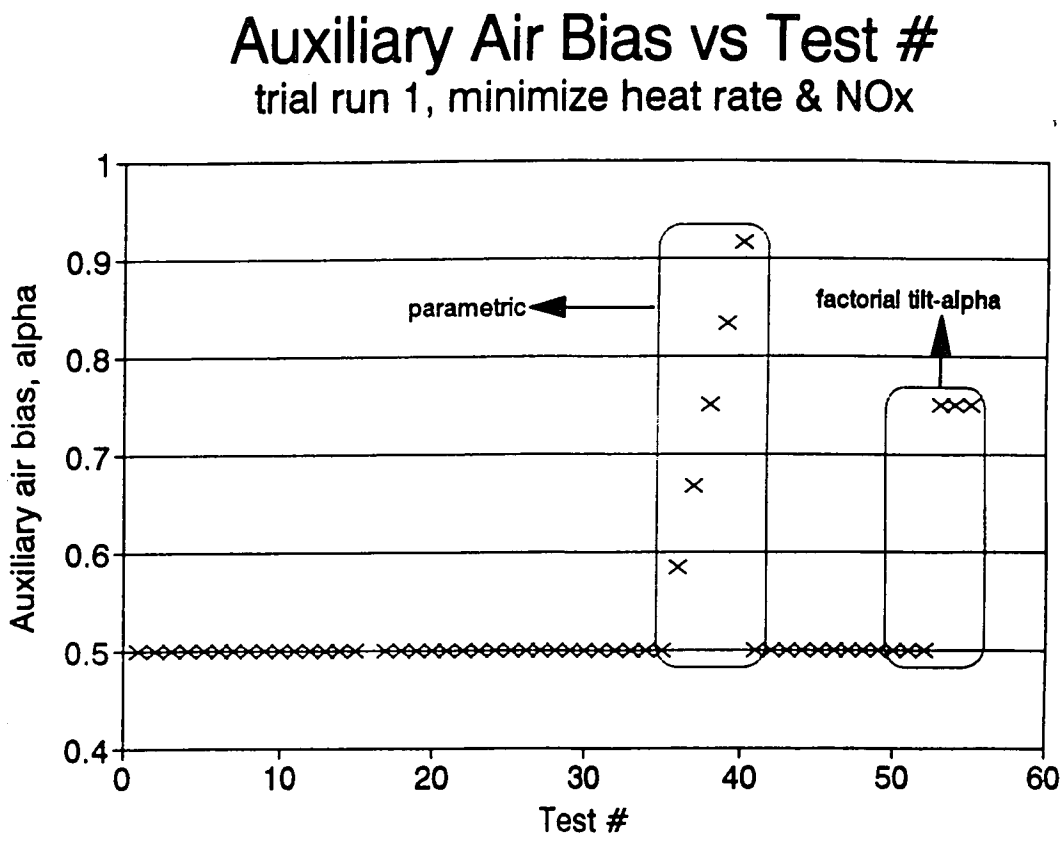
Auxiliary Air Bias vs Test #
trial run 1, minimize heat rate & NOx
Auxiliary air bias, alpha
1
0.9
0.8
0.7
0.6
0.5
0.4
parametric
factorial tilt-alpha
0
10
20
30
40
50
60
Test #

Figure 50

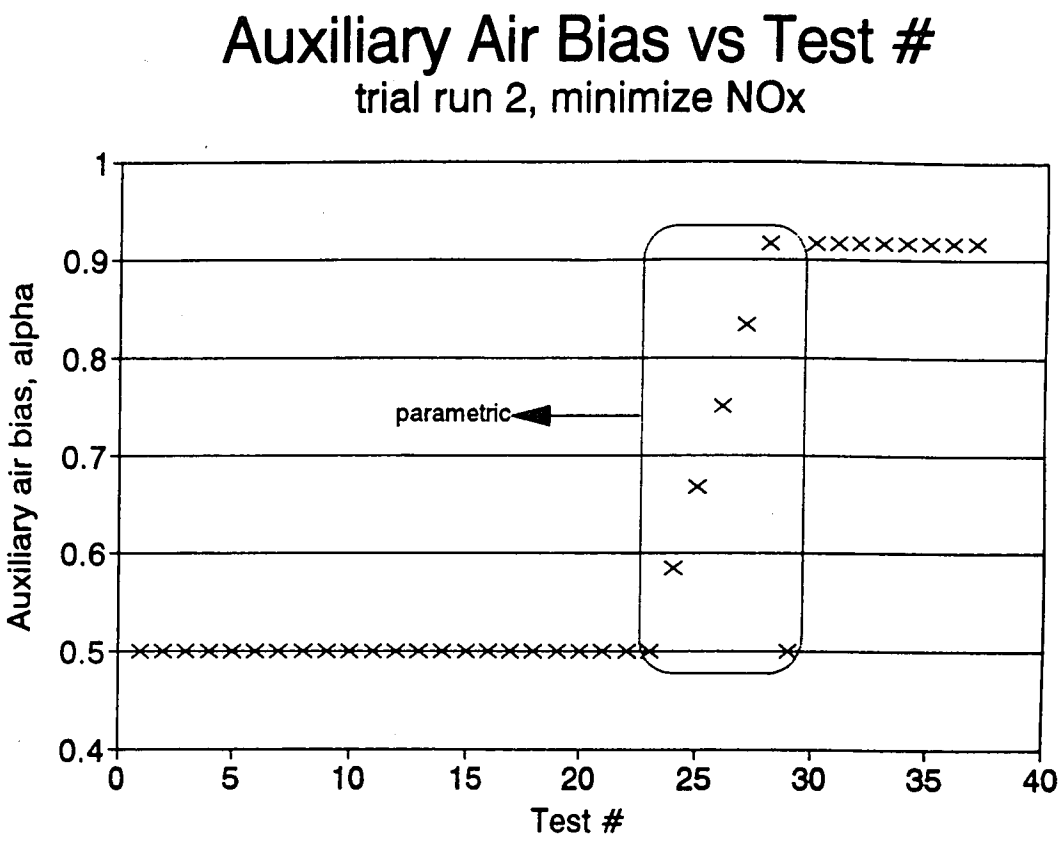
Auxiliary Air Bias vs Test #
trial run 2, minimize NOx
Auxiliary air bias, alpha
1
0.9
0.8
0.7
0.6
0.5
0.4
parametric
0
5
10
15
20
25
30
35
40
Test #

Figure 51

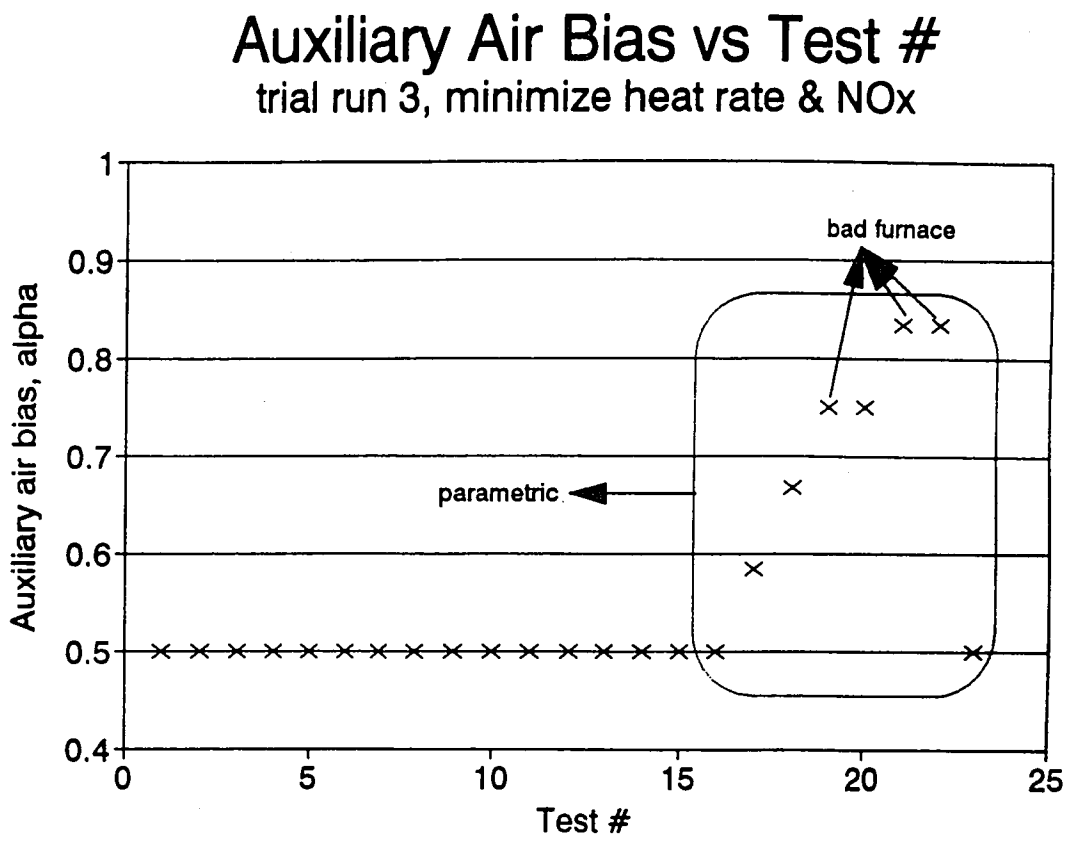
Auxiliary Air Bias vs Test #
trial run 3, minimize heat rate & NOx
Auxiliary air bias, alpha
1
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
Test #
bad furnace
parametric

Figure 52

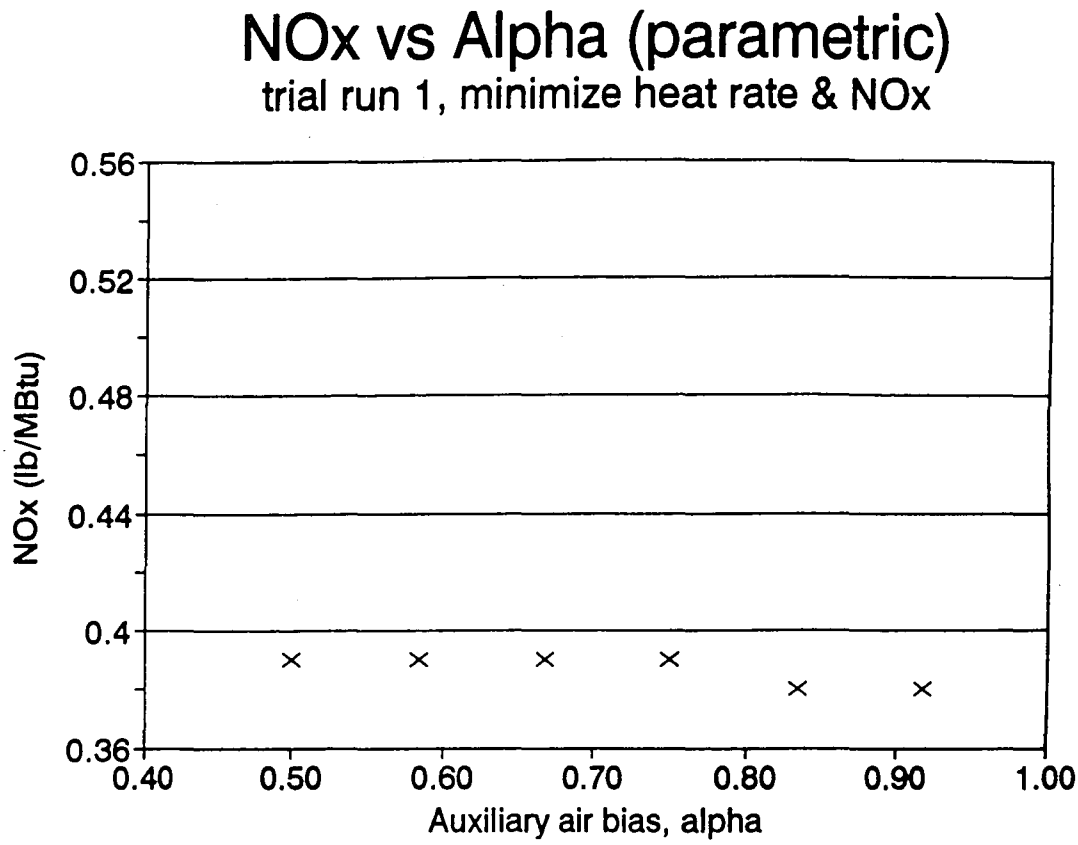
NOx vs Alpha (parametric)
trial run 1, minimize heat rate & NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
0.40
0.50
0.60
0.70
0.80
0.90
1.00
Auxiliary air bias, alpha

Figure 53

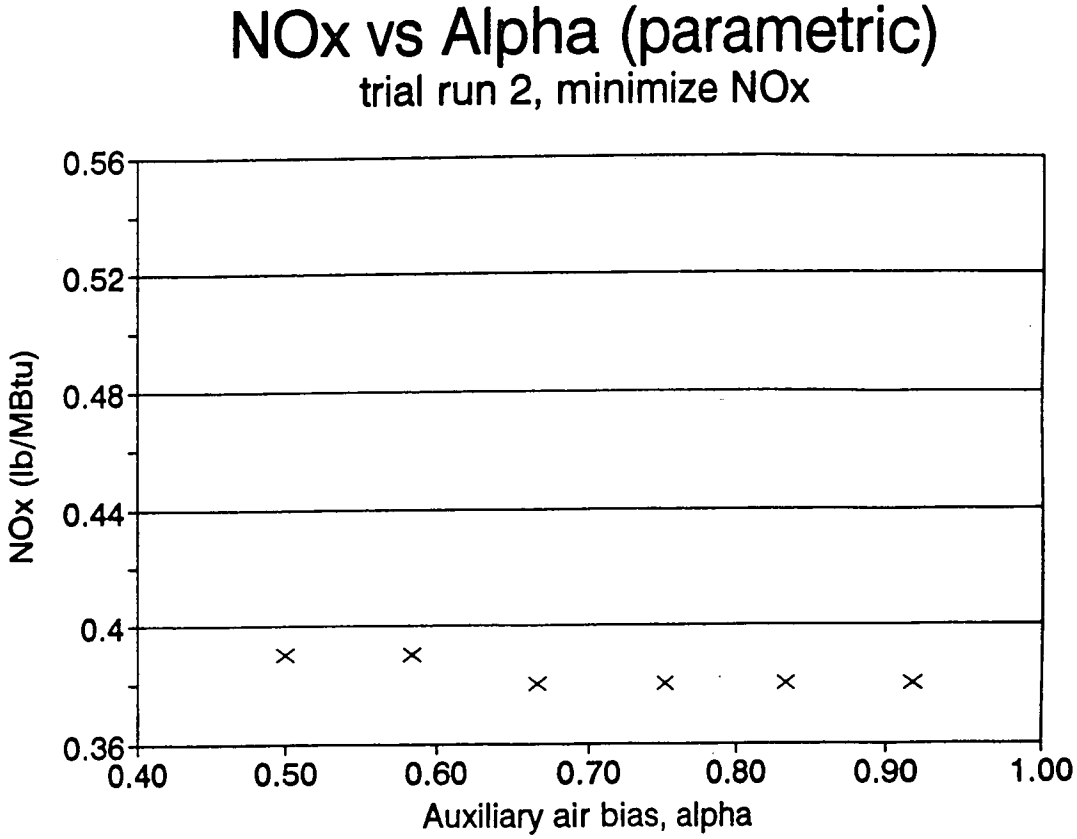
NOx vs Alpha (parametric)
trial run 2, minimize NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
0.40
0.50
0.60
0.70
0.80
0.90
1.00
Auxiliary air bias, alpha

Figure 54

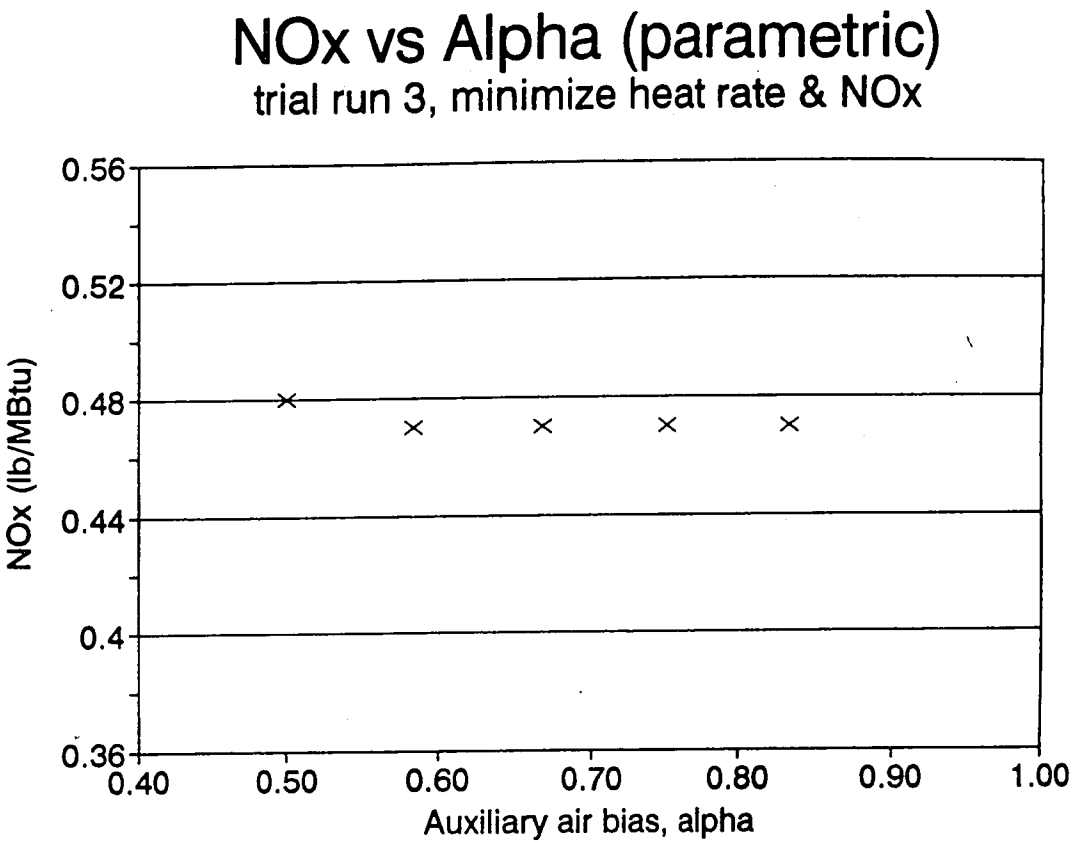
NOx vs Alpha (parametric)
trial run 3, minimize heat rate & NOx
NOx (lb/MBtu)
0.56
0.52
0.48
0.44
0.4
0.36
0.40
0.50
0.60
0.70
0.80
0.90
1.00
Auxiliary air bias, alpha

Figure 55

Table 1 Auxiliary Air Dampers Tested During Parametric Testing (trial 1)

test	O2	tilt	beta	fuel-air	NOx	alpha	aux-air	flame
35	1.9	4	-0.071	4,4,4,4	0.39	0.50	4,4,2,2,2	ok
36	1.9	4	-0.071	4,4,4,4	0.39	0.58	5,4,3,3,2	ok
37	1.9	4	-0.071	4,4,4,4	0.39	0.67	5,4,4,4,1	ok
38	1.9	4	-0.071	4,4,4,4	0.39	0.75	5,4,3,3,1	ok
39	1.9	4	-0.071	4,4,4,4	0.39	0.83	5,5,3,3,1	ok
40	1.9	4	-0.071	4,4,4,4	0.39	0.92	5,5,2,2,1	ok
41	1.9	4	-0.071	4,4,4,4	0.39	0.50	4,4,2,2,1	ok
50	1.9	-2	-0.071	4,4,4,4	0.39	0.50	5,3,3,3,2	ok
51	1.9	4	-0.071	4,4,4,4	0.39	0.50	4,4,4,4,1	ok
52	1.9	10	-0.071	4,4,4,4	0.39	0.50	4,3,3,3,1	ok
53	1.9	-2	-0.071	4,4,4,4	0.39	0.75	5,4,4,3,1	ok
54	1.9	4	-0.071	4,4,4,4	0.39	0.75	5,5,3,2,2	ok
55	1.9	10	-0.071	4,4,4,4	0.39	0.75	5,5,2,2,2	ok

Table 2 Fuel Air Dampers Tested During Parametric Testing (trial 1)

test	O2	tilt	beta	alpha	NOx	fuel-air	flame
42	1.9	4	-0.071	0.50	0.38	3,3,3,3	ok
43	1.9	4	-0.071	0.50	0.38	2,3,3,3	ok
44	1.9	4	-0.071	0.50	0.39	3,4,4,4	ok
45	1.9	4	-0.071	0.50	0.39	4,4,4,4	ok
46	1.9	4	-0.071	0.50	0.39	3,3,3,3	ok
47	1.9	4	-0.071	0.50	0.39	2,3,3,3	ok
48	1.9	4	-0.071	0.50	0.39	3,4,4,4	ok
49	1.9	-2	-0.071	0.50	0.39	4,4,4,4	ok

Table 3 Auxiliary Air Dampers Tested During Parametric Testing (trial 2)

test	O2	tilt	beta	fuel-air	NOx	alpha	aux-air	flame
23	1.9	4	-0.086	4,4,4,4	0.39	0.50	4,4,2,2,2	ok
24	1.9	4	-0.086	4,4,4,4	0.39	0.58	5,4,3,3,2	ok
25	1.9	4	-0.086	4,4,4,4	0.38	0.67	5,4,4,4,1	ok
26	1.9	4	-0.086	4,4,4,4	0.38	0.75	5,4,3,3,1	ok
27	1.9	4	-0.086	4,4,4,4	0.38	0.83	5,5,3,3,1	ok
28	1.9	4	-0.086	4,4,4,4	0.38	0.92	5,5,2,2,1	ok
29	1.9	4	-0.086	4,4,4,4	0.39	0.50	4,4,2,2,1	ok

Table 4 Fuel Air Dampers Tested During Parametric Testing (trial 2)

test	O2	tilt	beta	alpha	NOx	fuel-air	flame
30	1.9	4	-0.086	0.92	0.37	3,3,3,3	ok
31	1.9	4	-0.086	0.92	0.36	2,3,3,3	ok
32	1.9	4	-0.086	0.92	0.37	3,4,4,4	ok
33	1.9	4	-0.086	0.92	0.38	4,4,4,4	ok
34	1.9	4	-0.086	0.92	0.37	3,3,3,3	ok
35	1.9	4	-0.086	0.92	0.36	2,3,3,3	ok
36	1.9	4	-0.086	0.92	0.37	3,4,4,4	ok
37	1.9	-2	-0.086	0.92	0.38	4,4,4,4	ok

Table 5 Auxiliary Air Dampers Tested During Parametric Testing (trial 3)

test	O2	tilt	beta	fuel-air	NOx	alpha	aux-air	flame
16	2.1	15	-0.029	4,4,4,4	0.48	0.50	4,4,2,2,2	ok
17	2.1	15	-0.029	4,4,4,4	0.47	0.58	5,4,3,3,2	ok
18	2.1	15	-0.029	4,4,4,4	0.47	0.67	5,4,4,4,1	ok
19	2.1	15	-0.029	4,4,4,4	0.47	0.75	5,4,3,3,1	bad
20	2.1	15	-0.029	4,4,4,4	0.47	0.75	5,4,4,3,1	ok
21	2.1	15	-0.029	4,4,4,4	0.47	0.83	5,5,3,3,1	bad
22	2.1	15	-0.029	4,4,4,4	0.47	0.83	5,5,4,3,1	bad
23	2.1	15	-0.029	4,4,4,4	0.48	0.50	4,4,2,2,1	ok

DIAGNOSTIC ADVISOR

Introduction

The objective of the diagnostic advisor is to investigate reasons of high NO_x levels in a boiler that has already been optimized for low NO_x operation. Such a boiler would have operated at the targeted NO_x levels for a period of time, before, either gradually or suddenly, it starts producing high NO_x levels. In this case, the advisor is used to find possible causes of high NO_x and guides an engineer through a series tasks and questions towards this purpose. The engineer is assumed to be experienced in running investigative experiments and in making judgements on test data and other boiler operational issues.

In designing the diagnostic advisor, possible causes of high NO_x and the evidence that points to these causes were gathered from ERC engineers. This information was based from past experiences in diagnosing reasons for increases in NO_x in boilers and from speculations based on knowledge of NO_x formation mechanisms and knowledge of boiler hardware.

The knowledge gathered was organized in the form of possible causes of high NO_x levels, and clues that would indicate if these causes were present. These clues could be data obtained from the online data acquisition system, information from tasks or experiments requested by the advisor, historical information on the boiler, or other boiler observations made by the engineer.

These causes and clues, the diagnostic process employed by the advisor, software architecture and present status of the software are discussed

in the next sections.

Knowledge Base

The knowledge used as the basis of the diagnostic advisor was gathered from ERC engineers who used their past experiences in trouble-shooting boilers producing unexpectedly high NO_x, or speculated on what could go wrong using their knowledge of NO_x formation. The first stage of gathering this knowledge included identifying all the possible causes of high NO_x and the evidence that would help identify if a particular cause was present. The second stage in gathering the knowledge focused on identifying and understanding the investigative process employed by the engineers when trying to identify the possible problem areas, and then implementing a working technique for the diagnostic advisor.

Causes of High NO_x Levels

Twelve categories of problems were identified as possible causes of high NO_x levels. These causes affect NO_x formation in varying degrees. Under each cause the evidence that could alert the engineer to the existence of that cause was listed. This evidence is found by performing tasks such as analyzing data for unexpected trends, performing specific experiments that prove or disprove the nature of the problem, checking for any malfunctioning boiler components, and identifying when or over what load range high NO_x levels occurred.

Incorrect Control Settings

The first cause identifies incorrect control settings as a possible reason

of high NO_x. Once a boiler has been optimized for low NO_x operation, if, for some reason, the boiler is set at points other than the recommended settings, it would start producing different NO_x levels that are, very possibly, higher than usual. As discussed in the optimization advisor section, the boiler control parameters used to reduce NO_x are economizer oxygen level, burner tilt angle, the number of mills in operation, the coal bias among burners, auxiliary air damper positions and fuel air dampers positions. The first and only task involved in verifying if control parameter settings could be a cause of high NO_x is to check if the current control settings correspond to recommended control settings.

Besides the parameters listed above, exhauster damper positions and mill classifier settings (see Figure 3) are two parameters that are not considered in the optimization advisor, but could also be optimized for improved boiler performance and reduced emissions. The exhauster damper positions control the flow rate primary air transporting the pulverized coal to the burners and the mill classifier settings control the fineness of the coal that is being pulverized.

A mill that is performing below acceptable levels can indirectly affect NO_x levels by instigating other problems. If boiler performance is suffering because of mill performance problems and LOI values are very high, then plant operators would compensate by increasing the oxygen levels and thus create high NO_x levels. Mill performance problems can also affect the maximum rate of

coal flow and influence the extent the mills can be biased. Thus, in order to be able to adjust the oxygen levels back to recommended settings or to be able to bias mills, the mill performance problem would need to be fixed. LOI values and CO concentrations that are too high for the current oxygen setting, and mill exit temperatures, mill suction pressures, mill currents and mill amps to feeder RPM ratio that are unexpected for the current mill settings are clues to a mill that is not performing to standard. Mill rejects that are unusually high in quantity or contain too much coal, or coal fineness that does not correspond to the mill classifier settings can also indicate a mill that is not performing well.

Oxygen Sensor Malfunction

A problem with economizer O_2 sensors is another possible cause of high NO_x. This problem could exist as a result of incorrect calibration, loss of calibration, sensor malfunction or changes in the O_2 stratification patterns in the economizer. This means that the oxygen readings in the control room do not reflect the true measure of oxygen in the boiler. The presence of this problem could be identified by the existence of one or all of the following: LOI values that do not compare to typical values; calibration reports that show serious drift in calibration points or inaccurate calibrations; calibration lines that have leaks; different sensors producing significantly different outputs or no output; and flue gas flow rate that does not correspond to the indicated economizer oxygen setting. Comparison of independent traverse measurements of oxygen in the economizer with the sensor readings would finally prove or disprove sensor

malfunctions.

Boiler Air Leakage Change

Changes in rates of boiler air leakage to the furnace or convective pass could significantly affect NO_x levels since it affects the amount of oxygen in the boiler, which has been shown to be a major factor in NO_x formation. It could also affect NO_x rate calculations if the rate equation is based on excess oxygen levels. The existence of changes in leakage in the convective pass can be detected by comparing furnace oxygen levels with economizer oxygen levels to see if there is a change in the difference from previous times. Checking boiler maintenance records for any recent work on the boiler or inspecting the boiler envelope for any visible changes are also steps that can be taken to ascertain changes in boiler air leakage. Finally, performing leakage measurements would ascertain changes in rates of leakage.

Malfunctioning Burner Tilt and Air Damper Sensors

Burner tilt angle, secondary air damper, or exhauster damper sensors could be malfunctioning and the indication outside of the boiler could be different from the readings in the control room. This is a problem in the signals that are sent from the boiler components to the control room. Steam temperatures that are not within the typical range for the burner tilt setting could indicate a malfunctioning burner tilt sensor. Unexpected windbox pressures could indicate a problem with the secondary air damper sensors, and unexpected exhauster discharge and mill suction pressures could indicate

problems with the exhauster damper sensors. Performing burner tilt damper surveys, secondary air damper surveys or exhauster damper surveys would tell for certain if any of these problems exist. Surveys are visual checks of the indicators outside the boiler to see if they correspond to control room settings.

Malfunctioning Burner Tilt and Air Damper Mechanisms

Even if the sensors discussed above were not malfunctioning and show the true values, the mechanisms that operate these boiler components could be broken and the indicators outside the boiler might not show what is really happening inside the boiler. For example, if the burner tilt mechanisms were broken, changing the burner tilt settings in the control room would change the burner tilt indicators outside the boiler, but the actual burner tilts would not change. The same scenario could happen for secondary air damper mechanisms and exhauster damper mechanisms. Performing boiler surveys at two different control settings would demonstrate the existence of this type of problem, if the indicators outside the boilers change but nothing else in the state of the boiler changes. Checking if variations in burner tilt setting produced insignificant or unexpected changes in NO_x and steam temperatures would produce clues to a malfunction in the burner tilt mechanisms. Insignificant changes in windbox pressure and NO_x to variations in secondary air damper changes would indicate malfunction of secondary air dampers, and insignificant response of exhauster discharge and mill suction pressures to variations in exhauster damper settings would indicate a malfunction of the exhauster

damper mechanisms. Making visual furnace inspection might also reveal broken tilt mechanisms.

Coal Feeder Malfunctions

Coal feeder malfunctions can contribute to high NO_x problems by not delivering the desired quantities of coal to individual mills. The malfunction could be attributed to a number of different reasons that could be detected by one of the following ways. If the mill amps to feeder RPM ratio is radically different from typical values, if desired mill exit temperatures can no longer be achieved, if mill suction pressures can not be maintained, or if mill output is not steady, then there is reasonable likelihood of a coal feeder malfunction. Performing feeder counts and comparing it to the motor RPM signal would also reveal inconsistencies. Inspecting a feeder for blockages, uncontrolled coal flow, malfunctioning feeder leveling arm or other irregularities is another step that would be taken if coal feeder malfunction is suspected.

Fuel Quality Change

Since "fuel NO_x" is shown to be directly related to fuel quality, changes in fuel quality could be a cause of high NO_x levels. Poor quality coal can result in high LOI values, and coal that does not grind well or is wetter than usual can affect the quantity, fineness and flow of coal delivered to the burners. If, typically, mills can be biased at full load, but at the present, full capacity of mills is needed to achieve load levels, the problem might be a result of changes in fuel quality instead of a mill performance problem as mentioned earlier.

High LOI values and CO concentrations that are different from typical values might be used as an indication of changes in the quality of the coal supply. Coarse coal grind, unexpectedly high mill amps, too much mill rejects or mill rejects that contain too much coal might also point to poor quality coal. Manual and visual inspection can reveal a very wet coal supply or unusually large chunks of coal that should have been ground better before reaching the mills.

Performing proximate and ultimate analysis of the coal would also give more accurate information on the current coal supply. Proximate analysis is performed by heating the coal and gives certain characteristics of the coal: moisture content, volatile matter, fixed carbon and ash. Two additional tests which are routinely performed provide information on heating value and ash-fusion temperature. The Hardgrove Grindability Index, HGI, provides information on the hardness of the coal [1]. The ultimate analysis gives the elemental components of the coal: carbon, hydrogen, nitrogen, oxygen and sulfur.

Error in CEM Calibration

Calibration problems with the continuous emissions monitoring system (CEM) could be the cause of high NO_x readings, when in reality the NO_x levels are unchanged. The CEM measures NO_x and CO_2 levels. Infrequent calibration could result in serious drift in calibration and may be used as a first clue that there might a problem with the CEM. A problem with the calibration gases used in the last calibration could also result in a CEM measurement error and

recalibration using a new batch of calibration gases would fix this problem.

One possibility for detecting CEM calibration problems involves analyzing the fuel factor, F_o. The F_o of a particular type of fuel reflects the combustion characteristics of the fuel and it should fall within a range of values (see Table 6).

$$F_o = \frac{20.9 - \%O_2}{\%CO_2} \qquad (11)$$

If the calculated F_o value (see Eqn. 11) is not within the typical range, it could be the result of incorrect CO_2 or O_2 readings [22].

Table 6 Typical F_o Values for Different Fuels [22]

Fuel Type		F_o range
Coal	Anthracite and lignite	1.016 - 1.130
	Bituminous	1.083 - 1.230
Oil	Distillate	1.260 - 1.413
	Residual	1.210 - 1.370
Gas	Natural	1.600 - 1.836
	Propane	1.434 - 1.586
	Butane	1.405 - 1.553
Wood		1.000 - 1.120
Wood bark		1.003 - 1.130

Unexpected F_o values could also be a result of fuel changes. It is recommended that O_2 and CO_2 be measured at the same location and they both be measured on a dry basis

CEM Mechanical or Electrical Problems

Mechanical or electrical problems with the CEM could cause a problem with the NO_x signal and result in high NO_x readings. Out of range values of NO_x and CO_2 could be a result of this problem. Performing calibrations on NO_x and CO_2, and checking for disagreement with CEM output signals would verify the presence of this problem. Finally, to identify the precise nature of the problem, sampling lines should be checked for leaks, and wiring should be checked for short or open circuits.

Error in NO_x Rate Calculations

Error in NO_x emission rate calculations is another cause of false high NO_x levels. NO_x is calculated according to EPA method 19, and sampling and measurement techniques determine the equation that should be used. The NO_x gas sampling can be done on a dry basis or wet basis, the rate calculation can be based on O_2 or CO_2 (the calculation gas), and the calculation gas sampling can be done on a dry or wet basis. Table 2 shows a compilation of all the equations as listed in EPA method 19 (see Ref 23). The concentration of NO_x is measured in ppm and multiplied by $1.194.10^{-7}$ to convert it to lb/scf and the F factor is in units of scf/MBtu which gives a NO_x rate of lb/MBtu. The F factor of a fuel is found by dividing the volume of the gaseous products of combustion by

the heat content of the fuel. The wet F factor, F_w, includes all products of combustion, the dry F factor, F_d, excludes moisture, and the carbon based F factor utilizes CO_2 measurement. F factors of different fuels ,as listed in Reference 23, are shown in Table 8. B_{wa} is the moisture content of ambient air and usually can be approximated by 0.027 [23]. B_{wg} is the moisture fraction of the flue gas and is measured directly.

NO_x emission rate calculation errors, resulting in change in NO_x level reading, can occur in a number of different ways if sampling locations, methods or computer calculation methods are altered by plant personnel without the appropriate accompanying changes. If NO_x and the calculation gas are not measured in the same location, none of the equations in Table 7 are valid. Errors can also occur by using equations that are not appropriate for the sampling method, or using the wrong parameter for the sampling method and the fuel type (see Tables 7 and 8).

Dirty Boiler

As already mentioned in previous sections, excessively dirty boilers cause increases in NO_x emissions (see Figure 12). First clues to a dirty boiler are very high steam and furnace exit gas temperatures. Since the burner tilt angle is automatically reduced if steam temperatures are too high, very low burner tilt angles could be an indication of a dirty boiler. Checking if sootblowing occurs at acceptable frequencies, whether the appropriate sootblowers are used, and if the sootblowers are malfunctioning would show if

Table 7 NO_x Rate Equations

NO_x sampling method	calculation gas	gas sampling method	NO_x rate equation NO_x(lb/MBtu) =
dry	O_2	dry	$\frac{NO_x.F_d.(20.9)}{(20.9 - \%O_2)}$
wet	O_2	wet	$\frac{NO_x.F_w.(20.9)}{20.9(1 - B_{wa}) - \%O_2}$
			$\frac{NO_x.F_d.(20.9)}{20.9(1 - B_{wg}) - \%O_2}$
wet	O_2	dry	$\frac{NO_x.F_d.(20.9 - \%O_2)}{20.9 . (1 - B_{Wg})}$
dry	O_2	wet	$\frac{NO_x.F_d.(20.9)}{(20.9) - \frac{O_2}{(1 - B_{wg})}}$
dry	CO_2	dry	$\frac{NO_x.F_c.100}{\%CO_2}$
wet	CO_2	wet	$\frac{NO_x.F_c.100}{\%CO_2}$
wet	CO_2	dry	$\frac{NO_x.F_c.100}{\%CO_2.(1 - B_{wg})}$
dry	CO_2	wet	$\frac{NO_x.F_c.100.(1 - B_{wg})}{\%CO_2.}$

the boiler is too dirty or not. Finally, a visual inspection of the furnace might reveal slag build-up.

Improper Ignitor Usage

The last problem identified as a possible cause of high NO_x is ignitor usage. Visual inspection of the furnace by experienced plant personnel could

Table 8 F Factors for Various Fuels [23]

Fuel type		F_d scf/MBtu	F_w scf/MBtu	F_c scf/MBtu
Coal	Anthracite	10,100	10,540	1,970
	Bituminous	9,780	10,640	1,800
	Lignite	9,860	11,950	1,910
Oil		9,190	10,320	1,420
Gas	Natural	8,710	10,610	1,040
	Propane	8,710	10,200	1,190
	Butane	8,710	10,390	1,250
wood		9,240		1,830
wood bark		9,600		1,920
Municipal solid waste		9,570		1,820

reveal improper use of ignitors, malfunctioning ignitors, or improperly aligned ignitors. Ignitors might also need purging.

Since, for each cause, there are a number of different forms of evidence gathering, the engineers identified the tasks that they would give priority in performing. A priority is given to a particular evidence gathering task either because it is the easiest to perform, or because it would provide the most information. For instance an engineer would check LOI values, go over calibration reports, and check for calibration line leaks before ordering traverse measurements of the economizer to identify O_2 sensor problems.

The existence of a piece of evidence could be the result of any one of a

number of causes and varying degrees of weight are associated to the likelihood of each cause. For example , if LOI values are unexpectedly high, one would strongly suspect fuel quality problems, be somewhat suspicious of malfunctioning economizer O_2 sensors and finally be aware of the possibility of mills that are not performing to standard. This is reflected in the different weights given to each one of the problem areas.

Diagnostic Process

To design a technique for the diagnosis of the above causes, it was first necessary to understand the investigative process employed by the NO_x control engineers. Even though different engineers use slightly different techniques and follow different paths, the principle underlying their techniques was established.

When the fact that there is a high NO_x problem at a particular unit is first discovered, general questions that would characterize the boiler, operating conditions, and the nature of the NO_x problem are asked. Using this information as a basis, the more likely causes are identified. In the case where high NO_x levels are observed over the whole load range of the unit, change in the chemical composition of the fuel supply, calibration error of sensors and CEM, and error in the NO_x rate calculations, would be identified as likely causes. In cases where the high NO_x levels occur over only a specified load range, improper control settings could be the cause. If the deviated NO_x levels occur only at low loads, problem with ignitor usage is a likely cause since its effect on

NO_x is minimized at high loads. The character of the change in NO_x from acceptable levels also provides clues to the problem area. Sudden changes in NO_x would be more likely a result of mechanical or electrical problems with the CEM, while a gradual change would be more likely to result from a boiler becoming more heavily slagged with time.

Likely causes can also be identified based on past experience. For example, if a plant engineer who has worked with boilers for several years sees the boiler suddenly exhibit unacceptable trends, he/she might immediately get a "gut feeling" as to where the problem lies. This feeling is developed over a number of years of encountering a particular problem. Since the utility industry does not have extensive experience with trouble shooting boilers for high NO_x problems, this type of apriori likelihoods are not incorporated into the advisor.

Instead, in the absence of any evidence, the engineers might give some causes priority over others based on intelligent guessing, choosing the causes that are easy to verify, choosing causes that would have the greatest effect on NO_x, or just as a matter of style. For instance, checking if the control parameters were on their recommended settings came very high on the list of all the NO_x control engineers interviewed.

Then, some initial data that are easily available are also analyzed to see if a particular cause is more likely than others. For instance, high steam temperatures would point in the direction of a dirty boiler, and, by a smaller degree, to malfunctioning tilt indicators or broken tilt mechanisms. Non typical

mill suction pressures would point to coal feeder malfunction or malfunctioning exhauster damper indicators.

Based on the initial information and data described above, a list of likely causes of the high NO_x levels is formed. Some causes would be higher in the list than others since they would have either stronger or a larger quantity of evidence pointing in their direction. These causes are now investigated one at a time with the checks, experiments, or exploratory tasks described in the previous section. These investigations would either disprove that a particular problem exists, or improve the likelihood of the presence of a problem. If a problem is found to exist, it is corrected, and the investigation of other causes is continued until the level of NO_x is returned to acceptable levels.

To represent the process described above, various models were explored. The challenge lay in characterizing the expressed likelihoods, suspicions, probabilities, and uncertainties. How can a statement like "If LOI is too high, there is a strong likelihood that there is a mill performance problem, some chance of problem with fuel supply, and a small probability of malfunctioning economizer oxygen sensors." be expressed in the expert system? How can mounting suspicion towards a particular problem be represented as more and more evidence is gathered?

Representing uncertainty in expert systems was identified as a very important field in AI, and a number of models have been developed to aid with this task. There are different sources of uncertainty in knowledge based

systems: data or information could be imprecise; knowledge could be heuristic or intuitive, and even if it helps the expert guess well, it is not certain; knowledge could be on the probability of an event; information could be incomplete; and information could be qualitative. Various models were developed or used to deal with these types of uncertainties: probability theory, belief networks, certainty factors, Dempster-Shafer theory, and fuzzy set logic [7,24].

The certainty factors method was chosen for the diagnostic advisor since it is one of the most widely used and easily applied methods, and it has a proven record with diagnostic systems. This method was developed by Sortliffe and Buchanan for MYCINE, the earliest, and one of the most successful medical diagnostic expert systems (see section on expert systems) [6,25].

In the certainty factors method, a statement, A, is associated with a certainty measure C(A), such that C(A) = 1, if A is true, C(A) = -1 if A is false, and C(A) = 0 if nothing is known of A. So, a statement could have a certainty measure that is anywhere between -1 and 1, indicating some degree of certainty on the statement. On the other hand, rules have a certainty factor, CF, that ranges between -1 and 1. The certainty factor is a measure of the reliability of a rule and is expressed in the following manner: If A, then B with a certainty factor = y.

"If you turn on the light switch and the light does not come one, then the bulb is burnt out, with CF = 0.8" is an example of an everyday rule expressed with a

certainty factor.

In order to calculate the certainty measure of the consequence of a rule, Equations 12, 13 and 14 are used. For a rule in the form of "If A, then B with CF", and where A is known with absolute certainty, C(A) = 1, and C(B) is known, then to calculate the new C(B), these equations are used:

$$C(B/A) = C(B) + (1 + C(B))\,CF \quad \text{for } C(B) < 0,\ CF < 0 \tag{13}$$

$$C(B/A) = C(B) + (1 - C(B))\,CF \quad \text{for } C(B) \geq 0,\ CF \geq 0 \tag{12}$$

$$C(B/A) = \frac{(CF + C(B))}{(1 - \min(|C(B)|, |CF|))} \quad \text{for } C(B) \geq 0,\ CF \geq 0 \tag{14}$$

If C(A) < 1, then CF of the rule is modified by multiplying it by C(A). If a rule has multiple conditions, then, the composite certainty measure of the conditions are calculated by the following rules:

$$C(X \text{ or } Y) = \max(C(X), C(Y)) \quad (15)$$

$$C(X \text{ and } Y) = \min(C(X), C(Y)) \quad (16)$$

For the diagnostic NO_x advisor, the experts were asked to quantify their suspicion levels of a cause if they see a particular evidence. This produced certainty factors for all the rules that relate an evidence to a cause. Examples of suspicion levels assigned to a rule are: "If F_o value is not within typical range, then problem could be fuel related (CF = 0.3), or problem could be with CEM calibration (CF = 0.6).", or "If calibration report does not show any errors, then there is less likelihood of a problem with the economizer O_2 sensors (CF = -.2)". Positive values prove the existence of a cause, while negative values disprove it. Most of the evidence does not give a 100% verification of a the presence or absence of a problem.

In this version of the advisor, the antecedents of the rules are assumed to have a certainty measure of 1. This means that the statements that steam temperatures are high, that windbox pressure is too low, or that coal fineness is not typical are known with perfect certainty. So all the evidence is treated as if it is 100% true. This assumption is not necessarily valid, but perhaps it should be reexamined in later versions of the advisor.

The engineers were also asked to prioritize the causes they would explore in the absence of any evidence. Thus the causes were classified into a

few broad categories according to when the engineers thought they should be explored. For example, incorrect control room settings, fuel related problems, or problems with CEM calibration would be explored first, while improper ignitor usage or exhauster damper malfunctions would be explored last. Using these categories as a basis, initial certainty measures were assigned to all the possible causes. As the diagnostic process goes underway and rules are used to make conclusions, the certainty measures of causes are updated by Equations 12 through 16.

The other area where prioritization was used was in the tasks or information that are requested by the advisor when exploring a particular cause of high NO_x. Some information like historical data is easily available and is given priority over the gathering of other types of information. Experiments or other types of tasks that are expensive, time consuming or difficult to perform are not requested until there is high probability of the presence of the problem.

The implementation of the knowledge base and the diagnostic process described above is discussed in the next section.

Software Description

The execution of the diagnostic advisor is accomplished in three stages. In the first stage, the user is asked to provide details on the nature of the NO_x problem and other relevant system information. In the second stage, the advisor analyzes the information obtained in the first stage and compiles an initial list of possible causes of the high NO_x level. In the last stage, using the results of the second stage, the advisor guides the user through questions, tasks, and experiments to determine the real cause of high NO_x levels.

For purposes of discussion, rules, functions and facts in the diagnostic advisor can be categorized into three sections according to their role in the execution of the program. Various features of the advisor that aid in the execution of the program are discussed in the first section. In the second section, the inference engine, or the rules that perpetuate the program are discussed. The last section contains the rules that comprise the specialized knowledge base on the diagnosis of high NO_x and make up the three stages discussed above.

Program Features

The diagnostic advisor uses a set of input-output functions that interact with the user in the absence of a user interface. ***Choose_Ask, Type_Number, Type_Integer, Y_N_Ask, good_ok_bad*** and ***run-window-app*** are input-output functions also used in the optimization advisor and are described in detail in the

the section describing ***start-up*** module (refer to Section on ***start-up*** module). ***Type-String*** prints a question to the screen and reads back an answer if it is a string. ***Open-file*** uses ***Type-String*** to obtain the name of a file containing some specific information from the user and tests if the file actually exists by trying to open it. If the file does not exist, ***Open-file*** informs the user of this and repeats the request for the right file name.

Other functions defined in the diagnostic advisor calculate mathematical values. Functions that calculate mill and auxiliary air bias parameters are discussed in the optimization section. A function called ***NO_x-equation*** calculates the NO_x rate equation given NO_x in ppm, a code identifying the sampling methods, and the value of the calculation gas.

Various facts that would be used in the advisor were predefined. The first set of facts were stored in templates[14] defined for each type of data needed. Each definition of a template has a name and different slots[15] to store various attributes of the data. In the diagnostic advisor, two types of templates were defined: one to store F factors of fuels, and one to store NO_x rate equations. The template called ***f-factors*** has slots for type of coal, wet F factor, dry F factor, carbon based F factor, and the minimum and the maximum of the

---

[14] A template is a customized database structure that stores facts in a specific way. For example a NOx template that stores NOx in ppm and NOx in lb/MBtu could be defined.

[15] The slots of a template are similar to the slots of an object and store one piece of information associated with the data stored in the template.

F_o factors. The template called ***NO_x-calc*** has slots for NO_x gas sampling method, the name of the calculation gas to be used in the rate equation, the sampling method for the calculation gas, the rate equation in a string format that could be displayed to the screen, and a code the could be used to identify the right equation in the NO_x function for the conditions described by the other slots. Thus, the information in Tables 6 and 8 was stored in facts using the template ***f-factors***, and information in Table 7 was stored in facts using the template ***NO_x-calc***. These facts serve as a data base for the diagnostic advisor and are accessed by any rule that needs information on NO_x rate equations or various fuel factors.

Facts that contain current information on the various causes of high NO_x are initially defined. These facts are continually updated by the rules containing the knowledge base on high NO_x diagnosis and have the form (***cause*** <name of cause> <suspicion level> <unique marker>). <Name of cause> contains descriptive names of the causes described in the previous section. These names are consistently used throughout the program to identify a particular cause. Twenty four different names were used to identify all the specific problems that were categorized under twelve sections in the previous section: *eco-O_2-setting, burner-tilt-setting, number-of-mills-operating, mill-bias, aux-air-pattern, fuel-air-pattern, exhauster-damper-settings, mill-classifier-settings, mill-performance, ecoO_2-sensor-malfunction, boiler-leakage-change, burner-tilt-indicators, secondary-air-indicators, exhauster-damper-indicators, burner-tilt-mechanism, secondary-air-*

mechanism, exhauster-damper-mechanism, coal-feeder-malfunction, NO_x-rate-calculation, fuel-related, CEM-calibration, dirty-boiler, CEM-mechanical-electrical and *ignitor-usage.*

<Suspicion level> contains the value of the certainty factor of that cause at that moment. A different initial suspicion level is assigned to each cause before the execution of the program starts to reflect the idea that in the absence of any evidence, an investigator might still be suspicious of one cause more that others either from previous experience or other form of intuition. At this point in the development of the advisor, all the causes were prioritized only in three levels. Thus, initially, a cause will have one of three suspicion levels. Since the value of these initial suspicion levels are very small, in the presence of evidence they become insignificant.

The <unique marker> is a number produced by a random number generator to mark one fact as different from another fact that might by coincidence contain the same information. For very short durations of the program (explained in the next paragraph), two similar ***cause*** facts can exist and the marker becomes important in distinguishing between the two facts.

Examples of ***cause*** facts are: (***cause*** *$ecoO_2$-setting* .03 gen1), (***cause*** *boiler-leakage-change* .02 gen30), (***cause*** *dirty-boiler* .02 gen34), (***cause*** *fuel-related* .01 gen39). Most of the time, only one ***cause*** fact per cause exists in the system. When a rule addressing a particular cause is fired, the rule asserts a new ***cause*** fact with a suspicion level related only to the evidence it has just

evaluated, regardless of the current suspicion of the cause. Thus, at that point there will be two ***cause*** facts that are associated to the same cause and which may or may not have the same suspicion level. Then, a rule called ***combine-certainties***, which has a salience[16] of 1000 and thus has higher priority over all rules of less salience (a rule with no externally defined salience, has an implied salience of 0), is fired. ***Combine-certainties*** is fired only if two ***cause*** facts for the same cause exist, and removes both these facts to assert one ***cause*** fact that assigns that has one suspicion value. ***Combine-certainties*** finds this new suspicion value using Equations 13, 14 and 15.

Besides rules, functions and facts, the diagnostic advisor uses a file, "***cause.lis***", containing detailed text information on the various problems to better communicate with the user. "***Cause.lis***", was developed using a CLIPS capability to build a help file that can be accessed in a tree[17] format. The information contained in this file helps interpret some of the terms used in the advisor in more detail to the user. For example, if the advisor comes up with a conclusion that there is a problem with *NO_x-rate-equation*, accessing information under *NO_x-*

---

[16]A salience refers to the priority given to a rule in comparison to other rules that have the potential to be executed at that point. A higher salience number corresponds to a higher priority. The salience of all rules is 0 unless otherwise defined.

[17]A "tree format" refers to a way of classifying information so that one starts out with the most general information and "branches out" as the information gets more specific. For instance "the animal kingdom" can be represented in a tree format. Animal kingdom is the top of the tree, then the two branches from the top will represent the vertebrates and the invertebrates. Each of these branch out further until all the known animals are specifically categorized.

rate-equation in the "***cause.lis***" file would give a more informative and detailed description as to the nature of the problem. In this case, the description would say "the NO_x rate equation that you are using is incorrect for the calculation gas and the sampling method that you are using." The use of the help file enables the advisor to operate with descriptive variables like *NO_x-rate-equation* internally, but become more informative when communicating with the user.

"***Cause.lis***" is set up to have twenty four different trees corresponding to the names of the twenty four specific high NO_x causes mentioned above. Under each tree, different branches contain information on one aspect of the cause. For example, under *NO_x-rate-calculation*, there are three different branches: *NO_x-rate-equation*, *NO_x-rate-variables*, and *NO_x-rate-location*. The information contained in *NO_x-rate-equation* is described above. The description under *NO_x-rate-location* says "the sampling locations of NO_x and the calculation gas are different creating a possible error in the NO_x rate calculation." The description under *NO_x-rate-variables* would say "The values of the variables that you are using in the NO_x rate calculation are wrong for the type of coal and the gas sampling method you are using."

To aid in accessing the information in "***cause.lis***", facts named ***explanation*** were defined for each of the possible causes. The format of the explanation facts are: (***explanation*** <name of cause> <list of specific information that indicate the possible presence of the cause> <unique marker>). <Name of cause> would contain one of the twenty four names of causes. This corresponds to a

name on the top of a tree in "***cause.lis***". An example of such a name is *NO_x-rate-calculation*. <List of specific information> contains zero or more items and relates to specific information that indicate the possible presence of the cause listed in <name of cause>. Examples of the items that could be listed in <list of specific information> are *NO_x-rate-equation, NO_x-rate-location* and *NO_x-rate-variables*.

When a particular rule concerned with one or more of the causes is fired and finds a problem, it asserts an ***explanation*** fact listing the name of the cause it is evaluating and the name of the problem it has found. A rule called ***combine-explanations***, which has a salience of 1000 is then fired. ***Combine-explanations*** is fired only if more than one ***explanation*** facts for the same cause exist, and removes both facts to assert one ***explanation*** fact with a combined list of specific problems. Thus, at any one time, there exists only one ***explanation*** fact for one cause.

Inference Engine

The rules in this section are used to perpetuate the program through the different stages mentioned in the introduction of this section. Rule ***NO_x-high*** is fired when the program is first initiated and asks the user if there is a high NO_x problem (refer to Figure 56 flow chart of the program). If the user answers in the negative, this rule halts the execution of the program. If the user

REDUCTION RATIO 14:1

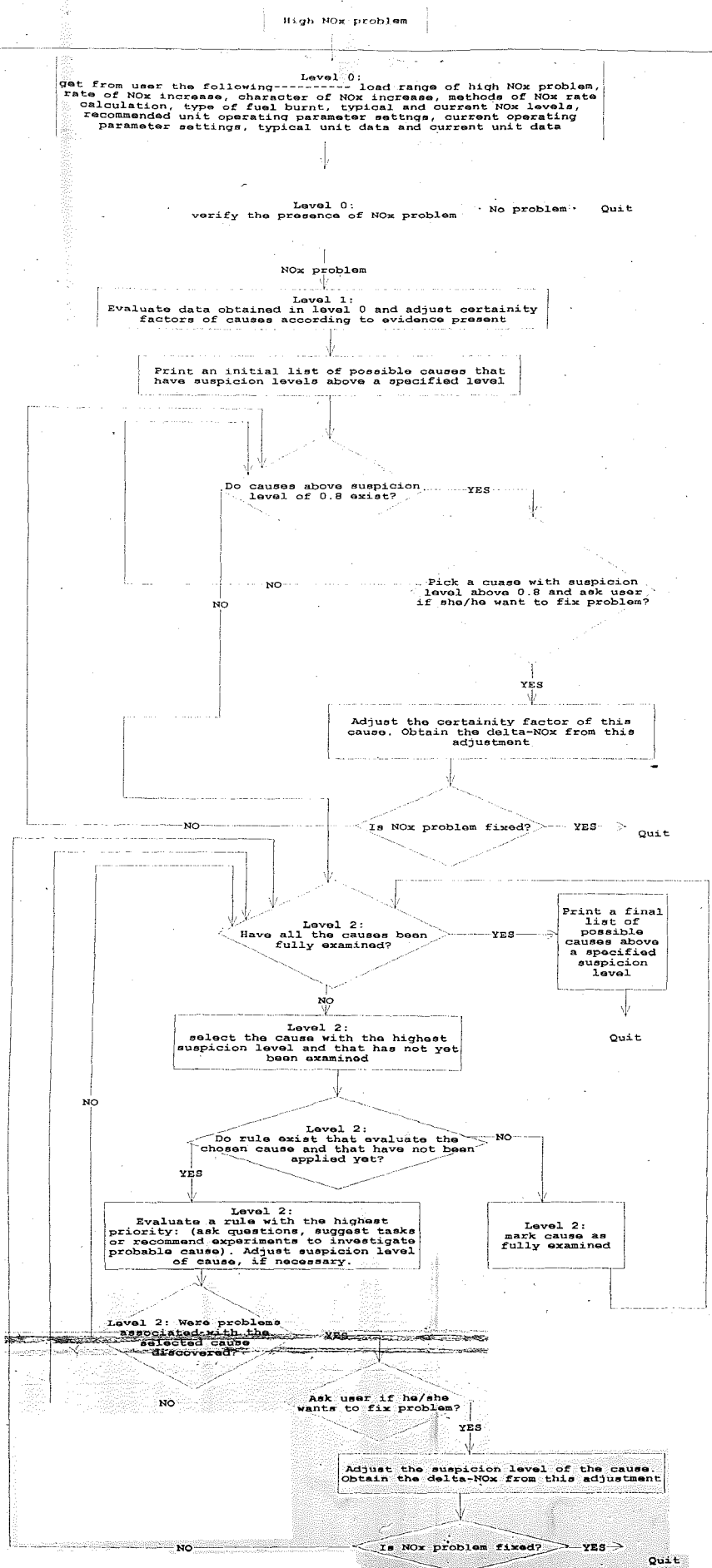

Figure 56 **Diagnostic Advisor**

REDUCTION RATIO 11:1

acknowledges a high NO_x problem, then rule ***level0*** is activated. ***Level0*** has a salience of -1000, which means that it does not get fired until all other relevant rules with higher salience are fired. ***Level0*** initiates the rules that execute the first stage of the program. These rules are grouped under section ***level-0*** inside the program.

The next rule that aids in the execution of the diagnostic advisor is rule ***level1***. ***Level1*** is fired only if ***level0*** has already been fired and it has a salience of -1000 so that it will get fired only after all relevant rules under section ***level-0*** are fired. ***Level1*** passes execution of the program to rules that control the second stage of the program. These rules are grouped under section ***level-1*** inside the program.

Rule ***level1-a*** is activated only after ***level1*** is fired and it has a salience of -1000 preventing it from being fired until all relevant rules under section ***level-1*** are fired. ***Level1-a*** activates a sequence of rules that would print to the screen a list of possible causes of high NO_x and the corresponding suspicion levels in order of declining suspicion levels. An example list would look like:

INITIAL SUSPICIONS

CAUSE	SUSPICION LEVEL
mill-bias-setting	1.0
aux-air-pattern	1.0
boiler-leakage-change	0.89
dirty-boiler	0.80

If causes with suspicion levels above 0.2 do not exist, then a message informing the user that there is no initial guess to the cause of high NO_x is printed.

Rule ***level1-b***, which has a salience of -1000 is fired after the suspicions list is printed out. This rule activates rule ***ask-if-they-want-to-change*** which examines the causes in the initial suspicions list, and if they are above a certain suspicion level (the advisor is reasonably sure that the cause does exist) it activates rules ***adjust*** and ***get-delta-NO_x***.

These two rules are used throughout the execution of the program to communicate to the user the existence of a particular problem and to recommend a fix as soon as possible. Rule ***adjust*** accesses the ***explanation*** fact associated with the cause that is currently under investigation and using the list of specific problems found in this fact, it accesses the text information in "***cause.lis***" to inform the user on the details of the problem. There might be more than one problem listed for one cause such as *NO_x-sampling-location* and *NO_x-rate-variables*. Then the rule asks the user if he/she would like to fix the problem at the present time. If the user decides to correct the problem at this point, rule ***get-delta-NO_x*** is accessed. ***Get-delta-NO_x*** requests from the user the decrease in NO_x that was achieved by fixing the last problem. This decrease in NO_x is then subtracted from the Δ-*NO_x* known at the present. Δ-*NO_x* is the difference between the current value of NO_x and the typical or normal value of NO_x. Rule ***cause-found*** might be fired at this stage of the program if Δ-*NO_x* is within range

of uncertainty of typical NO_x values. If ***cause-found*** is fired, it informs the user that the high NO_x problem has been fixed and then halts execution.

Rule ***level2*** is activated after ***level1-b*** has been fired and it has a salience of -1000. ***Level2*** passes control of the program to the rules that make up the last stage of the program. These rules are organized under section ***level-2*** and direct the user through the different tasks, questions and experiments that detect the cause of high NO_x levels. Rule ***level2-a*** is activated after ***level2*** has been fired and is not fired until all the relevant rules under section ***level-2*** have been fired. ***Level2-a*** activates the same set of rules that rule ***level1-a*** activates. These rules print out a final compilation of problems which have a high probability of being the cause of high NO_x. After this point, the program terminates.

Diagnostic Rules

As discussed in the previous sections, the rules that contain the specialized knowledge are organized under three levels according to what stage of the program they are used at. Rules under section ***level-0*** ask the user to provide details on the nature of the NO_x problem and other relevant system information. Rules under section ***level-1*** analyze this information to obtain different degrees of suspicion levels for the different causes the advisor considers. The rules under section ***level-2*** use the results of section ***level-1*** to guide the user through questions, tasks, and experiments to determine the real

cause of high NO_x levels.

Level-0

Section ***level-0*** has a sequence of rules that extract initial information about the system under investigation and nature of the NO_x problem from the user. All the information obtained is stored in the form of facts to be accessed for later analysis. None of the information is analyzed until control of the program is passed on to section ***level-1***.

Rule ***NO_x-range*** asks for the load range of high NO_x levels, if known: whether it is over all load levels, over low load levels or high load levels. Rule ***NO_x-rate*** asks for the rate of NO_x increase, if known: whether it was sudden or gradual. The information from these question would help prioritize some causes over others. For example, if high NO_x levels were seen only over low load levels, then ignitor usage would come under suspicion (the rule addressing ignitor usage would be found under section ***level-1***). Rule ***load-range*** asks the user what steady state load range to analyze for this session and issues a warning if this range is different from the load level where high NO_x levels are exhibited.

Rule ***calculation-gas*** asks for the gas used in the NO_x rate calculation. Rule ***sampling-method*** asks if the sampling method of NO_x, O_2 and CO_2 is wet or dry. Rule ***sampling-location*** asks for the sampling location of these gases and rule ***fuel-burned*** asks for the type of coal burnt. Using the information from these rules, if the sampling locations of O_2 and CO_2 are the same, rule ***calculate-fo***

calculates the current F_o value for the coal that is being burnt.

Rule ***initial-question-NO_x*** obtains the typical or normal value of NO_x for the load level that is being analyzed, the current NO_x level, and the uncertainties associated with each value. Then Δ-NO_x, which is the difference between the typical and current levels of NO_x, is calculated. Rules ***NO_x-not-high-1*** and ***NO_x-not-high-2*** are fired if Δ-NO_x falls within the uncertainty measures of the typical NO_x values or current NO_x is actually less than typical NO_x values. These rules inform the user that there is no problem and terminate the program.

Rule ***recommended-setting*** obtains the name of the file that contains the recommended low NO_x control settings for the load level that is currently under investigation and reads in these values. At the present, these settings include economizer oxygen levels, burner tilt angle, number of mills under operation, mill bias parameter, auxiliary air dampers, fuel air dampers, exhauster dampers and mill classifiers. This information is stored in facts that have the form (recommended <control parameter name> <list of values>). <List of values> would contain either one setting value or a list of settings corresponding to either different mills or different dampers. Then, rule ***initial-data*** obtains from the user the current values of these control parameters and any measurement uncertainty that might be associated with them. These are stored in facts which have the form (current-setting <control parameter name> <list of values>). Since the values of economizer oxygen level and burner tilt angle can be

associated with uncertainties, they are stored in facts with the form (current <variable name> <value> <uncertainty>).

Then rule ***typical-data*** obtains the name of the file containing typical values and the corresponding uncertainties of all other relevant boiler variable. These are convective pass in-leakage, CO_2, CO, LOI, air flow, exhauster discharge pressure, mill suction pressure, mill amps, coal feeder RPM, mill exit temperature, furnace exit temperature, NO_x in ppm, minimum tilt angle, range of steam temperatures, range of windbox pressures, and ranges of mill amp to mill RPM ratios. The variables that are expressed in terms of a single value with uncertainties are stored in facts that have the form (typical <variable name> <value> <uncertainty>). The variables that are expressed in terms of a range are stored in facts that have the form (range <variable name> <minimum value> <maximum value>). Rule ***current-data*** obtains the name of the file containing the current values of all the variables listed in the typical date file. These data are stored in facts that have the form (current <variable name> <value> <uncertainty>).

Level-1

The rules in section ***level-1*** analyze the data obtained by the rules in section ***level-0*** using some of the NO_x diagnosis knowledge described in previous sections. This analysis causes the initial suspicion levels of all the causes to be adjusted and thus it forms an initial list of possible causes of high

NO_x.

When the NO_x diagnosis knowledge base was described in the section headed "Knowledge Base", it was organized under twelve categories of possible causes of high NO_x and under each category were listed evidences that could suggest the presence of these causes. In describing the process of investigation, it was shown that the NO_x control engineers first analyze data they have easy access to and then form a list of suspicious causes of high NO_x. To represent this process, the NO_x diagnosis knowledge was reorganized in the form of a list of evidences and the problems that they would suggest by their presence. Thus, the presence of a piece of evidence could point at one or a few different causes with varying degrees of certainty. For example, if LOI is much higher than is typically experienced, then cause of high NO_x could, more likely be change in the fuel quality, or it could also be malfunctioning economizer O_2 sensors.

Rules that analyze the initial data obtained in section ***level-0*** and conclude the presence of a particular cause were constructed. These rules use pattern-matching on all the facts that exist in the system and are fired if all their conditions are met. If they are fired, they assert ***cause*** facts for certain causes with associated suspicion levels. These rules also assert ***explanation*** facts that record the reason for the suspicion of the causes they are investigating. ***Combine-certainties*** and ***combine-explanations*** then consolidate these facts with any existing ones as described in previous sections.

Examples of the rules in section ***level-1*** are listed:

Rule: ***eco_2-set***
If
current oxygen level is more than recommended oxygen setting
then
assert these facts:
(***cause*** *eco_2-setting* 1 <marker>)
(***explanation*** *eco_2-setting more-than-recommended* <marker>)

Rule: ***LOI***
If
current LOI is lower than typical LOI
then
assert these facts:
(***cause*** *eco_2-sensor-malfunction* 0.1 <marker>)
(***cause*** *fuel-related* 0.2 <marker>)
(***explanation*** *eco_2-sensor-malfunction low-LOI* <marker>)
(***explanation*** *fuel-related low-LOI* <marker>)

Rule: ***windbox***
If
current windbox pressure is outside the range of typical values
then
assert these facts:
(***cause*** *secondary-air-indicators* 0.2 <marker>)
(***cause*** *secondary-air-mechanism* 0.1 <marker>)
(***explanation*** *secondary-air-indicators out-of-range-windbox* <marker>)
(***explanation*** *secondary-air-mechanism out-of-range-windbox*
<marker>)

Rule: ***low-range***
If
high NO_x levels are seen only at low loads
then
assert these facts:
(***cause*** *ignitor-usage* 0.4 <marker>)
(***explanation*** *ignitor-usage NO_x-low-load* <marker>)

Level-2

Investigative questions, experiments and any other tasks that might

reveal the presence of a particular cause are constructed as rules in this level. Thus, under this level, the approach to the investigation will have taken a different path. Unlike ***level-1*** where all available data are looked at and analyzed to form initial suspicions of possible causes, in this level the advisor looks at one particular cause at a time and explores different avenues that would prove or disprove the presence of that cause.

The cause that would be under current investigation is chosen for having the highest suspicion level of all other causes. If two causes of the same suspicion level exist, then one cause is chosen randomly. Only one investigative task for the chosen cause is performed before the suspicion levels of all relevant causes are updated and the search for the cause of the highest priority is again performed. This means that if an investigation of a particular cause seems to disprove the presence of the cause, then the suspicion level of the cause would have gone down and another cause might have priority for investigation next.

To accomplish the above processes, rules that choose the most suspicious cause for investigation were constructed: i***nitial-most-suspect-cause*** and ***most-suspect-cause***. If no more rules that would be able to investigate the cause found to have the highest suspicion level exist, ***no-rules-available*** is fired to remove the cause from further consideration. ***Combine-certainties*** (described above) update the new suspicion level of the cause that was just under investigation.

If the investigative tasks that explore a cause are prioritized because of level of difficulty or the extent of information they provide, the rules that address these tasks are set up such that a rule of lower precedence will not get activated unless a rule with a higher precedence has already been fired. Thus, one of the conditions for a rule with a lower precedence of investigation would be that an investigative task of higher precedence will already have been performed. For example, one of the conditions that needs to be satisfied for the rule that recommends independent traverse measurements of the economizer to be fired is that calibration lines should have been checked for leaks. This is because the latter task is less expensive and relatively easier to perform.

If a particular rule needs initial information, then accompanying rules that obtain this information were constructed. For example, the rule that would compare oxygen independent traverse measurements of the economizer with oxygen sensor readings has accompanying rules that recommend the traverse measurements to the user, and obtain the files that contain the values of these measurements as well as oxygen sensor readings. If a task is hard to perform like independent traverse measurements, the user is given the option of skipping it and performing it later in his/her convenience.

If any of the rules discovers a problem, it immediately gives the user the option of fixing it by activating rule ***adjust*** which was described earlier. This is unlike *level-1* where all the data are analyzed and a list of suspicious causes are compiled before the user is given the option of fixing the more certain

problems. In this case, ***adjust*** will ask the user if he/she wants to fix the problem and if the answer is positive, it activates ***get-delta-NO_x*** which obtains the reduction in NO_x obtained by the adjustment. If the reduction is sufficient to bring current NO_x levels within range of typical NO_x levels, ***cause-found*** is fired which terminates the program.

Examples of the rules in section ***level-2*** are listed:

Rule: ***wiring-and-lines***
If
the most suspicious cause is *CEM-mechanical-electrical*
then
Ask the user:
"Check sampling lines for leaks and wiring for short or open circuits. Did you find something?"
If
the answer is "YES"
then
activate rule ***adjust***
assert these facts:
(***cause*** *CEM-mechanical-electrical* 1 <marker>)
(***explanation*** *CEM-mechanical-electrical leak-or-short* <marker>)
else
assert this fact:
(***cause*** *CEM-mechanical-electrical* -0.2 <marker>)

Rule: ***independent-NO_x-CO_2***
If
the most suspicious cause is *CEM-mechanical-electrical* AND wiring and sampling lines have already been checked
then
Ask the user:
"Please perform NO_x and CO_2 calibration. Does CEM output signals agree with monitor signals?"
If
the answer is "NO"
then
activate rule ***adjust***
assert these facts:
(***cause*** *CEM-mechanical-electrical* 1 <marker>)

(***explanation*** *CEM-mechanical-electrical independent-measurements-disagree* <marker>)
else
assert this fact:
(***cause*** *CEM-mechanical-electrical* -0.4 <marker>)

Rule: ***soot-blowing-frequency***
If
the most suspicious cause is *dirty-boiler*
then
Ask the user:
"What is the soot blowing frequency?
a) every day
b) every other day
c) once a week "
If
the answer is "once a week"
then
activate rule ***adjust***
assert these facts:
(***cause*** *dirty-boiler* 0.7 <marker>)
(***explanation*** *dirty-boiler infrequent-soot-blowing* <marker>)
If
the answer is "every other day"
then
activate rule ***adjust***
assert these facts:
(***cause*** *dirty-boiler* 0.2 <marker>)
(***explanation*** *dirty-boiler infrequent-soot-blowing* <marker>)
If
the answer is "every day"
then
assert this fact:
(***cause*** *dirty-boiler* -0.2 <marker>)

CONCLUSIONS

Optimization Advisor

The optimization advisor package is not yet ready to be tested at a power plant. The data communication system and the user interface are still being developed. The neural-network-optimization package is undergoing revisions and has not yet been connected to the expert system part of the advisor. The expert advisor is also being modified to accommodate testing at additional load levels before it is made available for use by utility personnel.

To determine if the optimization advisor will successfully optimize a boiler, it has to be tested on real boilers to see how it handles situations that arise outside of the assumptions made to write this program. Some of the assumptions might need to be modified in future versions of the advisor. The assumption that all the relationships between the control parameters with the dependent variables are continuous functions with only one minimum or maximum could be false at some units. For example, a particular component of a boiler might be malfunctioning or working peculiarly and might not behave according to what is generally expected. Thus, NO_x versus burner tilt angle might not be parabolic or there might be a discontinuity in the function of NO_x versus α. The assumption that the coupling between control variables is weak might not be valid at all boilers and a sequential parametric approach to testing might not give the best results. The validity of this assumption can only be verified after testing several boilers. The advisor should be able to handle such

situations.

The advisor would also benefit from being redesigned to make it less procedural and more rule based. As knowledge on NO_x control and behavior of boiler control parameters grows and the principle behind the experimental process employed by the NO_x control engineers is better understood and documented, this task could be more easily achieved.

Future plans for the advisor include development of extensions that will guide experimentation at part loads. Rules to configure the advisor for boilers other than those similar to Potomac River Unit 4 should also be developed.

Diagnostic Advisor

The diagnostic advisor has a few more development stages to pass through before it can be tested. Even though the foundation has been laid and the major framework has been constructed, there are decisions on various features of the diagnostic advisor still to be made and more development work to be done. Some of the decisions made and the final development path taken will determine the required level of expertise of the intended users. Described below are some of the tasks that would need to be accomplished before the diagnostic advisor is ready for use.

A decision should be made on whether the diagnostic advisor will analyze current data to determine if there actually is a high NO_x problem, at what load ranges it exists, and the rate of the NO_x increase. At the present, the

user is assumed to know the answers to these questions and is asked to provide qualitative answers. If more quantitative answers to these questions are deemed necessary, and the intended user does not have the expertise or facilities to provide the answers, the advisor would have to perform rigorous analysis on current and historical data. The development of this analysis tool is a fairly substantial task, and the current state of the program might be considered sufficient for the first version of the advisor.

Whether diagnosis will be performed at one steady state load level at a time or if all load levels will be investigated at once needs to be decided. Analyzing all levels at once could be complicated, while analyzing one load level at a time could be unnecessarily time consuming and might also miss some information that would be obvious when examining the entire load range of the boiler. At the present, the advisor is configured to handle one steady load level at a time.

The gathering and analysis of initial data has to be resolved. These questions involve where and how typical data are gathered and represented and how current data are gathered and over what time period they are averaged. Data spanning full ranges of the various variables required by the diagnostic advisor could be obtained from the existing data base or by performing targeted experiments on the relevant variables at the time of NO_x optimization. Since an existing data base can't be guaranteed, a considerable amount of experiments might need to be performed to fill the gap. The issue of

how, when and by whom typical values of data are compiled is a critical one for the success of the diagnostic advisor. At this point, it is assumed that all the initial data needed by the advisor exist regardless of how they were obtained.

Data representation could be accomplished by building an extensive data base that would have the typical values of all the relevant variables throughout the ranges of all the control settings, developing correlations of the variables with the control parameters they are known to depend on, or by building neural networks [Ref. 2]. Building a data base that could be accessed by the advisor is a conceptually easy but cumbersome task, while a neural network presents an elegant solution that could present some implementation difficulties. The simplest representation scheme might lie with correlation equations of the variables with the control settings they are known to depend on. The solution to this issue would also depend on whether the different load levels are analyzed at once or one steady state level at a time. If all load levels are analyzed at once, then neural networks or correlation equations would present very simple alternatives to developing sophisticated data base functions that could access different parts of the data base simply and efficiently. The neural networks and the correlation equations would have load as one of the independent variables. Currently, the data for the load level that are being analyzed are read from a file requested from the user and then stored internally for the duration of the diagnosis.

The type and extent of the analysis of the data that might be obtained by one of the investigative tasks in the third stage are not yet clarified. The issues with the extent of the analysis lies with the level of experience of the intended user. For example, if the user knows the sensitivity of steam temperatures to burner tilt variations, then the advisor can simply recommend the user vary the burner tilt in a certain way, check the steam temperature and report back whether steam temperatures behaved appropriately. If the user is assumed not to know how to analyze such data, then the advisor would need to obtain the results of the experiments and do the analysis.

If it is decided that the advisor will do the actual analysis of the results of an investigative task, the next step is to determine the type of analysis required. In most cases, simple regression analysis will suffice, and the curve fitting Fortran codes that were developed for the optimization package are also built into the diagnostic advisor for this purpose. Besides the type of analysis that is done on the data, information on how to evaluate the results might become an issue. For instance, the criteria for the evaluation of steam temperature sensitivity to burner tilt angle variation might be hard to define. Currently, even though the framework for obtaining the data exists, the method for data analysis for the listed tasks has not been implemented. This includes economizer O_2 calibration reports, economizer independent traverse measurements, burner tilt survey results, burner tilt sensitivity tests, secondary air damper survey results, secondary air damper sensitivity tests, exhauster

damper survey results, exhauster damper sensitivity tests, proximate and ultimate analysis results, and CEM calibration reports.

The last task that needs to be performed before the advisor is ready to be tested is building the help file, "***cause.lis***." The structure of the file has been developed and some of the subject headings and the information they contain have already been entered. The only task involved in completing "***cause.lis***" is entering all the text information that would be required to explain to the user the different problems and to match the names that access this information with those mentioned in the ***explanation*** facts.

At this point, the advisor will be ready to be tested. It is recommended that testing be accomplished in two stages. The first stage of the testing should involve the NO_x engineers that contributed to the diagnosis knowledge base. These engineers should run the advisor with a data base that could have resulted when a particular problem exists. The engineers should then evaluate the path the advisor takes to get to a solution and if necessary modify the certainty factors of rules to adjust the path. For instance, if the advisor uses a particular investigative rule (at ***level-2***) later than when the engineers think advisable, then they would adjust the certainty factors (refer to Section on diagnostic process employed in the diagnostic advisor) of all relevant rules (at ***level-1***) that deal with the cause under consideration so that the cause will have a higher suspicion level upon the presence of particular evidence. The adjustment of certainty factors of rules to an optimum is a necessary task

since the certainty factors are quantitative measures that try to reflect subjective judgments on the part of the NO_x control engineers.

It is recommended that the second stage of the testing involve testing the advisor at a power plant with real problems. This might have to wait until after the development of a more sophisticated user and data interface. The user and data interface that are being developed for the optimization advisor would probably have to be modified only slightly to be used by the diagnostic advisor. At this point, the advisor will be ready for use by utility personnel.

REFERENCES

1. J. G. Singer, Combustion Fossil Powered Systems. Windsor, CT: Combustion Engineering, Inc., 1981.

2. P. Lee, Application of Neural Network and Mathematical Optimization Techniques for NOx Control. Bethlehem, PA: Lehigh University, Energy Research Center, 1994.

3. E.A. Fiegenbaum Computers and Thought. University of California, Berkeley: McGraw-Hill Book Company, 1963.

4. R. Levine, D.E. Drang and B. Edelson, AI and Expert Systems, A Comprehensive Guide, 2nd edition. United States of America: McGraw-Hill Publishing Company, 1990.

5. E.H. Shortliffe, Computer Based Medical Consultation: MYCINE. New York: American Elsevier Publishing Company, Inc., 1976.

6. B.J. Buchanan and E.H. Shortliffe, Rule Based Expert Systems, the MYCINE experiments of the Stanford Heuristic Programming Project. Addison-Wesley Publishing Company. Inc., 1984.

7. A.J. Gonzales and D.D Dankel, The Engineering of Knowledge-based Systems, Theory and Practice. Englewood Cliffs, NJ: Prentice Hall, 1993.

8. Electric Power Research Institute, Proceedings: Expert System Applications for the Electric Power Industry. Palo Alto, CA: EPRI, June 1992

9. P. Warne, K. Gamble, and M. M. McDonald, NO_xSMART: An Expert System for NO_x Reduction. TransAlta Utilities Report, 1993.

10. Power Magazine, Coming: Expert Systems to Predict Impact of Coal Quality on Plant Operation. McGraw-Hill, Inc., January, 1992.

11. J. Shaw, Breath of Fresh Air; South Coast Air Quality Management District's Emission's Monitoring Expert System; Application Watch. Miller Freeman Publications, 1993.

12. J.C. Giartano, CLIPS User's Guide. NASA, Lyndon B. Johnson Space Center, 1991.

13. D. Eskenazi, M. D'Agostini and P. Lee, Results of Extended Phase I and Phase II Low NO_x Testing. Bethlehem: Energy Research Center, Report 93-500-10-16, 1993.

14. K. Wark and Cecil F. Warner, Air Pollution, Its Origin and Control. New York: Thomas Y. Crowell, 1976.

15. C. T. Bowman "Chemistry of Gaseous Pollutant Formation and destruction" Fossil Fuel Combustion, a Source Book. United States of America: John Wiley & Sons Inc., 1991.

16. D. Eskenazi, M. D'Agostini, E. Levy and Z. Zhang, Summary of NO_x Emissions Pretest at Potomac River Unit 4. Bethlehem: Energy Research Center, Report 92-500-2-5, 1992.

17. M. D'Agostini, S. Woldehanna, D. Eskenazi, T. Schmitt, E. Levy, Analysis of Parametric Low-NO_x Test Data from Potomac River Unit 4.

Bethlehem: Energy Research Center, Report 94-500-2-5, 1994.

18. E. Levy, et al. Application of Boiler Performance Improvement Tools to NO_x Control presented at EPRI Heat Rate Improvement Conference. Birmingham, AL: 1992.

19. D. Eskenazi, et al., Month-Long Baseline NO_x Emissions and Performance At Potomac River Unit 4. Lehigh University, Energy Research Center, Report 93-500-7-12, March 1993.

20. D. Cramer, et al., NO_x Reduction Through Combustion Optimization at PEPCO's Potomac River Station, presented at EPRI 1994 Heat Rate Improvement Conference. Baltimore, MD: May, 1994.

21. Potomac River Unit 4 Boiler Contract Data Sheet. Sales Data Division: May 8, 1953

22. Environmental Protection Agency, Code of Federal Regulations, Title 40 Protection of the Environment, Part 60, Appendix A, Method 3, 1991.

23. Environmental Protection Agency, Code of Federal Regulations, Title 40 Protection of the Environment, Part 60, Appendix A, Method 19, 1991.

24. J.W. Grzymala-Busse, Managing Uncertainty in Expert Systems. Boston: Kluwer Academic Publishers, 1991.

25. H.R. Warner, Computer Assisted Medical Decision Making. New York: Academic press, 1979.

Vita

Sara Woldehanna was born in Addis Ababa, Ethiopia, on May 23, 1968. Her parents, Feleketch Abebe and Woldehanna Haile, reside in Ethiopia. She completed her high school education in Ethiopia and received her Bachelor's degree in physics from Randolph Macon Woman's College in Lynchburg, Virginia in May 1990. Sara is currently employed with Foster Wheeler Energy Corporation in Lebnon, New Jersey.

END OF TITLE