1999

In -cylinder combustion -based virtual emissions sensing

Michael L. Traver
West Virginia University

In-Cylinder Combustion-Based Virtual Emissions Sensing

by

Michael L. Traver

A DISSERTATION

Submitted to the College of Engineering and Mineral Resources
at
West Virginia University

in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
1999

Keywords: Virtual Sensor, Neural Network, Diesel, Emissions, Combustion

Acknowledgements

The author would like to thank the following people for their unique and important contributions: Julle for her love and support during the writing of this dissertation and for taking the time to correct my grammatical errors; Richard Atkinson for his invaluable help with the data acquisition and DSP systems and their associated electronics; Talus Park for laboriously installing the pressure transducers in the engine among other things; Tom McDaniel for his expertise in the operation of the engine; Dr. Chris Atkinson for giving me the chance to put this project together and for his expert help; and my committee members for their time and effort.

Table of Contents

List of Figures

List of Tables

Nomenclature

A/D	Analog to Digital
A/F	Air to Fuel ratio
ADC	Analog to Digital Converter
C	celsius
CA	Crank Angle
CI	Compression Ignition
CLT	Coolant
CMAC	Cerebellar Model Articulation Controller
CO	Carbon Monoxide
CO_2	Carbon Dioxide
D2	Diesel specification number 2
DI	Direct Injection
DOHC	Dual OverHead Camshaft
DSP	Digital Signal Processor
EBP	Error Back-Propagation
ECT	Engine Coolant Temperature
EGO	Exhaust Gas Oxygen
EPROM	Erasable PROgrammable Memory
EVM	EValuation Module
FIPW	Fuel Injection PulseWidth
FTP	Federal Test Procedure
HC	HydroCarbon
HN	Hidden Nodes
Hz	hertz
IAT	Intake Air Temperature
$IMEP_g$	Integrated Mean Effective Pressure (gross)
kPa	kilopascal
L	Liter
LMFB	Location of Mass Fraction Burned
LPP	Location of Peak Pressure
MAP	Manifold Air Pressure
MAT	Manifold Air Temperature
MFLOP	Million FLoating point OPerations per second
MS-DOS	MicroSoft Disk Operating System
NO	Nitric Oxide
NO_2	Nitrogen Dioxide
NO_X	Oxides of Nitrogen
OBDII	On-Board Diagnostics, 2nd revision
PC	Personal Computer
PM	Particulate Matter
P-V	Pressure-Volume
R^2	A Statistical Indicator Used in Regression Analysis
RBF	Radial Basis Function
RPM	Revolutions Per Minute
SI	Spark Ignition
TDC	Top Dead Center
TPS	Throttle Position Sensor
UEGO	Universal Exhaust Gas Oxygen

Abstract

The development of a real-time, on-board measurement of exhaust emissions from heavy-duty engines would offer tremendous advantages in on-board diagnostics and engine control. In the absence of suitable measurement hardware, an alternative approach is the development of software-based predictive approaches. This study demonstrates the feasibility of using in-cylinder pressure-based variables as the inputs to predictive neural networks that are then used to predict engine-out exhaust gas emissions. Specifically, a large steady-state engine operation data matrix provides the necessary information for training a successful predictive network while at the same time eliminating errors produced by the dispersive and time-delay effects of the emissions measurement system which includes the exhaust system, the dilution tunnel, and the emissions analyzers. The steady-state training conditions allow for the correlation of time-averaged in-cylinder combustion variables to the engine-out gaseous emissions. A back-propagation neural network is then capable of learning the relationships between these variables and the measured gaseous emissions with the ability to interpolate between steady-state points in the matrix. The networks were then validated using the transient Federal Test Procedure cycle and in-cylinder combustion parameters gathered in real time through the use of an acquisition system based on a digital signal processor. The predictive networks for NO_X and CO_2 proved highly successful while those for HC and CO were not as effective. Problems with the HC and CO networks included very low measured levels and validation data that fell beyond the training matrix boundary during transient engine operation.

1. Introduction

1.1 General Background

Internal combustion engines have been subject to emission control techniques since the passage of the regulatory Clean Air Act in1966. Successive amendments reducing the allowable levels of emissions emanating from new engines were also extended to cover particulate emissions from diesel engines. The trend towards lower allowable emissions levels appears to be continuing with particular emphasis on diesels. To this end a variety of test procedures has been developed for the measurement of emissions during typical driving cycles. These tests must be performed on chassis dynamometers due to the complex and large scale nature of the required emissions analysis equipment. Emissions measurement is used not only to assess regulatory compliance, but in engine and engine controller development as well.

Neural network architectures have gained popularity in recent years due to their excellent pattern recognition and prediction capabilities. They have been applied to fields as varied as optical character recognition and prediction of stock market performance [Haykin, 1994; Ward Systems Group, Inc., 1996]. Emissions formation in internal combustion engines lends itself well to the use of these networks as the processes involved are nonlinear in nature, multidimensional, highly transient, and difficult to characterize in a series of predictive equations. Furthermore, the nature of neural network training does not necessitate the detailed knowledge of combustion kinetics available only to research laboratories with extremely expensive and intrusive equipment.

The prediction of diesel engine emissions could potentially provide several benefits. Currently, emissions inventories for typical urban driving cycles are determined by approximation through dynamometers either with engine-only tests or full vehicle chassis testing. Emissions prediction could provide real-time and real-world levels in an actual urban driving cycle, eliminating the guesswork and providing more accurate information. Engine control schemes might also be able to take advantage of emissions prediction. Feedback-based controllers require a signal to indicate air-to-fuel (A/F) ratios and thus dictate operating conditions to minimize eventual emissions production. A predictive network could provide a virtual emissions signal which a controller could use directly for feedback purposes. Similarly, a direct emissions level indicator would reduce testing times for both post production compliance and engine controller development. Furthermore, the potential for diagnostic capabilities cannot be discounted. Not only can predictive networks supply emissions prediction levels, but they could also provide valuable information concerning failing equipment in an engine. Wildly fluctuating emissions values can indicate a failure somewhere in the engine system; something a typical neural network could also be trained to recognize, allowing an alert to be sent to the operator.

This study aims to prove that detailed knowledge of the emissions from diesel engines can be easily obtained through the application of neural networks using information from readily available engine sensors and established methods such as in-cylinder pressure measurement using flush-mounted pressure transducers. This thesis proceeds with an explanation of basic diesel combustion, continues with a discussion of emissions formation, reviews the literature on neural network applications and architectures, includes a review of pressure sensing techniques, and details the process of developing an in-cylinder pressure-based virtual sensing system.

1.2 Problem Statement

Neural networks are mathematical approximations that mimic the human brain's ability to learn relationships between several variables within an experimental dataset. They have been utilized in a variety of applications including control processes, pattern recognition, and trend prediction. NeuroDyne, Inc., in collaboration with West Virginia University, has demonstrated virtual emissions sensing using training data acquired at 20 Hz from a Saturn 1.9 L spark ignition engine. The virtual emissions sensing system demonstrated in this work was trained on data acquired on a continuous basis and proved quite capable of predicting levels of emissions in previously "unseen" data. The sensing system, however, relies upon the correct time shifting of the emissions analyzer signals to correspond with the engine parameter signals. Since the time difference between these two signals is not constant, due to delays in response time of the analyzers and the dispersive effect of the dilution tunnel, the use of a constant sized buffer to time-shift the emissions signals suffers from some inherent inaccuracy. Further, the use of only extra-cylinder engine operating parameters (such as the manifold air pressure, engine speed, throttle position, etc.), although providing excellent direct applications for industry, provides little information on the processes affecting the combustion inside the cylinder. Thus a separate approach based on distinct steady-state engine operating points and the fast real-time acquisition of in-cylinder combustion information would seem to be a more robust and informative method. Further, it is postulated that with sufficient training on in-cylinder combustion, transient conditions will pose no problems for the neural network's predictive capabilities. Consequently, the main goal of this research is to apply digital signal processing (DSP) technology to interpret the real-time signals generated by an in-cylinder pressure transducer and to feed this information to a trained neural network to provide instant information on the level of exhaust gas emissions produced by the engine.

2. Combustion Fundamentals

2.1 Diesel Combustion

Unlike spark-ignited (SI) engines which rely on throttles, the torque output of a diesel (or compression ignition (CI)) engine is controlled by the amount of fuel introduced into the combustion chamber. Air is inducted into the cylinder during the intake stroke and after significant compression has occurred during the compression stroke, fuel is injected under high pressure (a direct injection diesel engine is assumed). A side effect of this technique is the absence of pumping losses from a throttle, increasing the inherent efficiency of the diesel engine (typically a maximum of about 45% compared to approximately 33% for an SI engine). The conditions inside the cylinder at the time of injection are such that they exceed the autoignition threshold of the fuel; initiating combustion following a brief ignition delay period. As a result, cylinder-to-cylinder variations are minimized as the same effective volume of air is inducted into each cylinder [Heisler, 1995]. Also, the conditions at the time of fuel injection will remain nominally the same regardless of engine speed as opposed to a spark-ignition engine which must compress a fuel-air mixture. The in-cylinder conditions at the end of the compression stroke are primarily a function of compression ratio, intake temperature, and the engine speed [Heisler, 1995]. The first two factors are rather obvious, but the last, speed, becomes important when the time components of heat transfer, rates of combustion, and ring blowby are considered. Additionally, due to the method of combustion, detrimental autoignition known as “knock” is not a problem; compression ratios can thus be much higher, increasing thermal efficiency. Turbocharging also becomes attractive for diesel engines as the fuel delivery method allows increased intake pressures to boost the density of oxygen available.

Combustion in the diesel engine is controlled by the kinetics of diffusion flame reactions. Due to the prevailing conditions inside the cylinder, fuel injected is combusted more through the action of turbulent eddies and impinging geometry than anything else. Fuel spray enters the cylinder under high pressure, typically 20+ MPa [Heywood, 1988] (135 to 200 MPa in future engines), in an effort to atomize the droplets into easily combustible particles and provide enough energy to propel the fuel across the combustion chamber in the short period of time available. These particles then evaporate rapidly, aided in part by the heat released by the oxidation of other fuel particles. Finer atomization of the fuel particles aids in the process by increasing the surface area for evaporation. Once the heat release rate through oxidation exceeds the heat lost by evaporation, convection, and conduction, flame ignition occurs.

The time between the injection of the fuel and the occurrence of the self-sustained oxidation-ignition is known as the ignition delay. The start of injection of the fuel can be determined by recording the time at which the injector needle lifts off the seat. Typically, the end of the delay can be identified as the point where there is a discernible change in the slope of the heat release rate [Heywood, 1988].

Once the fuel has been injected and evaporated, oxygen must be supplied to the flame front in order to sustain combustion. This is accomplished through in-cylinder turbulence induced by the engine intake geometry and combustion chambers designed to increase swirl, the circular motion of the charge around a vertical axis centered on the piston. This swirl action washes the fuel mixture with fresh oxygen unless the fuel droplets are too small, in which case the swirl only drags the fuel along, never mixing it. Oversized droplets do not suffer this problem, but their large size slows down the rate of combustion and can result in elevated soot emissions, thus leaving an optimal droplet size for a given swirl/engine design.

Atomization of the fuel is only one factor affecting the ignition delay. Further effects include the conditions at the time of injection, mainly the pressure and temperature inside the cylinder. If these conditions remain constant over a range of speed, the ignition delay will not vary, but generally, other parameters such as cylinder wall temperature and time for heat transfer will alter the temperature and pressure of the air charge at injection, thus affecting the delay. The higher the temperature and pressure, the shorter the delay. As has already been noted, cylinder swirl has a significant effect on combustion, and the same is true for the delay. Finally, the cetane number, a measure of a fuel's tendency to vaporize and autoignite with higher numbers signifying faster autoignition, will, based on its very definition, affect the ignition delay. There are a number of associated parameters that are directly affected by the length of the ignition delay, among them misfire, smoke emissions, fuel conversion efficiency, and smoothness of operation [Heywood, 1988].

At the end of the ignition delay, the fuel encounters rapid oxidation and the cylinder experiences a comparatively rapid pressure rise. The location of the maximum pressure rise correlates roughly with the smallest volume and slowest piston speed near the top dead center position. The peak in-cylinder pressure is highly dependent on the ignition delay phase as that determines the pressure-volume phasing of the piston. Although it would seem that engine speed would have an effect on the combustion duration, increasing engine speed in fact increases the turbulent swirl of the charge at a rate that keeps this phase roughly constant in crank angle degrees for a range of speed [Heisler, 1995]. The rapid oxidation phase is followed by a period of steady combustion as the oxygen concentration becomes diluted and the injectors continue to supply a steady stream of fuel. These two phases provide the majority of heat released during the power stroke. The amount of energy transferred to the piston depends upon additional conditions inside the cylinder such as the temperature of the cylinder walls and the proper phasing of the combustion pressure with the piston position. As the piston is driven down through the power stroke, the pressure inside the cylinder drops, causing combustion to tail off. This reduces the various pollutant formation kinetics and freezes their respective concentrations, causing incomplete oxidation of some fuel molecules. Additionally, overly rich fuel regions produce black smoke from this incomplete combustion; as a result, "most diesel engines are rated for full-load operation just below the point at which the exhaust smoke visibly darkens" [Heisler, 1995].

Consequently, the diesel engine does not combust at a designated air-to-fuel ratio (A/F), but rather at a steady progression of fuel lean conditions [Edwards, 1974]. This situation creates an inherently stable flame as only a complete removal of fuel or air will interrupt the combustion process, with the exception of the chamber surfaces acting to quench the flame. Figure 1 shows a schematic of the A/F distribution in a typical diesel flame.

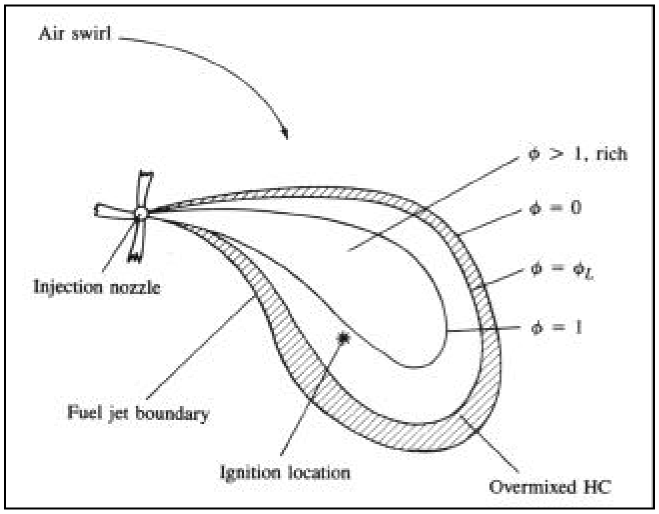


Figure 1: Diesel engine fuel spray showing equivalence ratio contours at time of ignition. Φ_L is the equivalence ratio at the lean combustion limit (≈ 0.3). [Heywood, 1988, p. 622]

Although the local A/F in the vicinity of the injected fuel spray remains reasonably close to stoichiometric, the overall charge A/F may vary quite considerably depending upon the operating conditions of the engine. In order to assure nearly complete combustion without the production of large amounts of particulates and smoke, an excess air amount of 20% above stoichiometric is typically maintained as the lower limit [Heisler, 1995]. This leads to a maximum A/F of around 18:1 at full load conditions and a minimum of around 100:1 at idle conditions [Heywood, 1988]. Figure 2 shows typical A/F distributions for both diesel and spark-ignited engines.

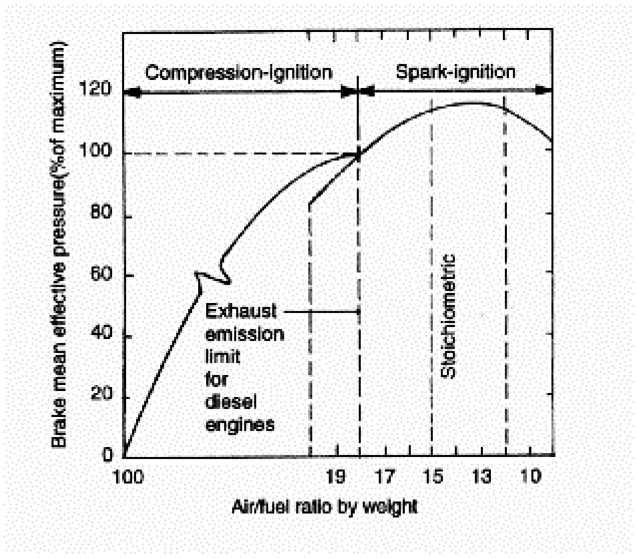


Figure 2: Typical A/F operating regions. [Heisler, 1995, p. 228]

2.2 Emissions Formation

2.2.1 Hydrocarbons

Hydrocarbons (HC) represent the family of emissions composed of hydrogen and carbon with a variety of bonds originating in partially or completely unburnt fuel. These range from simple non-reactive methane molecules (CH_4) to more complex chemical rings and chains like benzene (C_6H_6) and the aldehydes (e.g. acetaldehyde: CH_3CHO). Hydrocarbons are formed when fuel is not adequately oxidized, or burned. In diesels, incomplete combustion of the fuel results in soot formation, visible as large clouds of black smoke, containing up to 0.5% of the fuel mass. During startup and subsequent misfire, unburnt fuel may condense and produce clouds of white smoke [Degobert, 1995]. Overall, the level of HC emitted as a pollutant is strongly dependent upon the fuel distribution, the in-cylinder temperature, and the resulting combustion process inside the cylinder.

Hydrocarbon emissions can be split into two major groups: non-reactive and reactive. This grouping stems from the chemical reactivity of the molecules with respect to the indirect formation of smog (visible rust-colored air pollution created by the interaction of hydrocarbon chains, oxides of nitrogen, oxygen, and sunlight). Hydrocarbons play a secondary role in ozone formation by accelerating the formation of NO_2, which reacts with O_2 to produce ozone, the basic component of photochemical smog. The reactive components include all hydrocarbon chains except methane, which is highly stable and which gives rise to the term “non-methane organic gases”, describing all non-methane hydrocarbons and oxygenates. In addition to participating in smog formation, many oxygenates are also irritants to the eyes and lungs. Furthermore, many of these molecular chains are not found in the fuel prior to combustion, thus demonstrating the complex chemical kinetics that occur inside a combustion chamber.

One of the factors in the production of hydrocarbon emissions is the quenching of the flame front as it approaches the relatively colder surfaces of the cylinder walls and piston. These surfaces absorb heat energy to such an extent that combustion cannot be sustained within the fuel-air mixture. Crevices and gaps such as those seen between the cylinder walls and piston dominate this mechanism while hydrocarbons quenched at the walls are readily oxidized later in the cycle [Heywood, 1988]. One source of emissions unique to direct injection diesels comes from the fuel injector tips. Fuel left over in the nozzle tips after injection has ceased slowly evaporates and seeps into the combustion chamber where it may or may not be oxidized. However, the major source of HC emissions is the localized rich or lean conditions found within the combustion zones. Again, Figure 1 shows typical fuel distribution zones in an injected spray of diesel fuel. As the spray is injected, the air mixes with the outer edges of the fuel, producing very lean zones that oxidize in a non-self-sustaining manner and seldom to completion. As the spray continues to mix with the air, these lean zones expand outward, leaving more combustible mixtures behind in the center of the chamber. The amount of HC left unburned is then a function of the mixing rate (or turbulent swirl) of the engine, the cylinder conditions, and because of its association with the prior two, the ignition delay. According to Heywood, there is a non-linear relationship between the ignition delay and the amount of HC produced. Leanness, however, is not the sole condition aiding in the formation of hydrocarbon emissions. Overly rich mixtures will also result in incomplete combustion, a condition that can be caused by insufficient mixing of the oxygen in the air with the fuel spray. This is especially the case just after the injector nozzles have ceased spraying, as the pressure forcing the fuel out has dropped and the remaining fuel enters the combustion chamber at low speed. The low velocity of the fuel causes undermixing of the fuel and air to occur, which generates an overly rich region. Desorption of HC from the layer of lubricating oil that coats the cylinder walls adds to the overall level found in exhaust gas and is controlled by the characteristics of the fuel being used and its ability to be absorbed by the oil layer.

Engine operating conditions play a role in HC emissions mainly as a function of the load on the engine. Idle and light load conditions can generate overall A/F of around 100:1, causing an excess of over-lean regions in the injected fuel spray. Consequently, light load and idle produce substantially more HC emissions than full load on a mass-per-unit-time specific basis (g/s) [Heywood, 1988]. On the other end of the spectrum, overfueling of the engine at high loads will produce excessive HC due to an insufficient supply of oxygen.

The timing of the injection has an effect on HC as well. If the timing is advanced away from top dead center and away from the optimum timing, the ignition delay lengthens, allowing a higher percentage of the total fuel injected to mix with the air and impinge on the cylinder walls. This also produces more areas of lean mixtures, hindering efficient combustion and raising the amount of unburned HC [Degobert, 1995]. On the other hand, retarding the timing produces overly rich regions with insufficient time to combust, with the end result being visible smoke. In a similar vein, lengthening the time that the injectors are open and spraying fuel into the cylinders reduces HC at low load, but at high load leads to an increase in smoke and particulates [Degobert, 1995].

2.2.2 Particulate Matter

The distinction between particulate matter (PM) and hydrocarbon emissions is a matter of condensation temperature. Generally, heated probes in a dilution tunnel are maintained at 190°C and any hydrocarbon chain gathered by the probe that condenses is filtered out and lumped with the soot and ash accumulations as particulate matter, which is gathered by filtering the diluted exhaust stream at a constant 52°C.

Particulate formation is a major concern in diesel engine combustion and consists mainly of carbonaceous conglomerations. These clumps are formed mostly through incomplete combustion of fuel with small contributions from the lubricating oil [Heywood, 1988]. As the fuel in the advancing flame plume combusts, pyrolytic reactions crack the hydrocarbons that have yet to pass through the flame. As these reactions occur, particulate masses form and pass through the flame. A side effect of this process is the radiation heat transfer that is given off by the heated particulates, increasing the pyrolytic reactions in the unburned fuel. If the fuel mixing is poor within the cylinder, large quantities of particulates can form [Edwards, 1974]. Typically, above temperatures of 500 °C, the particles are composed solely of clusters of carbon, while at temperatures below this, higher molecular weight hydrocarbons condense onto the clumps. As the particulates travel through the flame front and into the more heavily oxygen-populated areas, the clumps tend to oxidize reducing their concentrations in the leaner regions of combustion. Additionally, inorganic compounds in the fuel can form small clumps of material known as ash. The sulfur content of the fuel may have an effect upon the amount of PM measured as the sulfur leaves either as sulfur oxides or as sulfates condensed onto the PM. A high sulfur fuel (e.g., one with sulfur content approaching 3.5% by weight) will demonstrate distinctly higher PM emissions in the same engine than a low- or zero-sulfur fuel as a result.

2.2.3 Oxides of Nitrogen

The main source of nitrogen in the chemical formation of oxides of nitrogen (NO_X) is atmospheric, although a very small portion is caused by nitrogen compounds found in some fuels. The fuel source is a more pronounced contribution in diesel combustion, however. The basic kinetic equations for the transformation of atmospheric nitrogen in the presence of oxygen are known as the Zeldovich mechanism. Two of these equations have been rigorously tested, with a third equation generally accepted to contribute significantly; as such, the three are sometimes referred to as the “extended” Zeldovich mechanism.

$$\mathrm{O} + \mathrm{N_2} \Leftrightarrow \mathrm{NO} + \mathrm{N} - 75\frac{\mathrm{kcal}}{\mathrm{mol}} \quad \text{(Eq. 2.1)}$$

$$\mathrm{N} + \mathrm{O_2} \Leftrightarrow \mathrm{NO} + \mathrm{O} + 31.8\frac{\mathrm{kcal}}{\mathrm{mol}} \quad \text{(Eq. 2.2)}$$

$$\mathrm{N} + \mathrm{OH} \Leftrightarrow \mathrm{NO} + \mathrm{H} + 39.4\frac{\mathrm{kcal}}{\mathrm{mol}} \quad \text{(Eq. 2.3)}$$

The third reaction is usually found in rich mixtures where OH is readily available. As the burned gas region behind the flame front absorbs energy from the combusting mixture, the pressure and temperature of the gas in the cylinder both rise significantly. It is this region's high temperature which spurs the formation of nitric oxide (NO) and in most cases, the flame front production is simply ignored. The flame front does, however, play two significant roles by providing the thermal energy required to dissociate the N_2 into N radicals and by providing the reactions which lead to the NO producing chains. The main controlling factors are the amounts of oxygen and nitrogen radicals available and the temperature of the mixture. This can be demonstrated through a derivation starting with the rate of formation of NO based on the Zeldovich equations:

$$\frac{d[\mathrm{NO}]}{dt} = k_1^+[\mathrm{O}][\mathrm{N}_2] + k_2^+[\mathrm{N}][\mathrm{O}_2] + k_3^+[\mathrm{N}][\mathrm{OH}] - k_1^-[\mathrm{NO}][\mathrm{N}] - k_2^-[\mathrm{NO}][\mathrm{O}] - k_3^-[\mathrm{NO}][\mathrm{H}]$$

(Eq. 2.4)

where the brackets denote species concentration in moles per volume. Heywood supplies values taken from experimental studies for the rate constants k_i. A similar equation is then formulated for monatomic nitrogen and by way of a steady state approximation, the rate *d[N]/dt* is set equal to zero. This supplies a method of eliminating the concentration of N from the NO rate formation equation; combining this result with the equilibrium oxygen atom concentration [Heywood, 1988] produces the following NO formation rate relation (where $[\ ]_e$ denotes equilibrium concentration) :

$$\frac{d[\mathrm{NO}]}{dt} = \frac{6 \times 10^{16}}{T^{1/2}} \exp\left(\frac{-69090}{T}\right)[\mathrm{O}_2]_e^{1/2}[\mathrm{N}_2]_e \quad \mathrm{mol/(cm^3 \cdot s)} \qquad \text{(Eq. 2.5)}$$

This equation can be used as a rough estimate of the NO formed during the combustion cycle provided that the in-cylinder temperature and concentrations of oxygen and nitrogen are known throughout the cycle. Unfortunately, while in-cylinder pressure is roughly uniform across the cylinder at any given time, the temperature within the cylinder varies significantly depending upon the position of the flame front and whether the point of "measurement" consists of unburned or burned gases or a mix of the two. Nonetheless, to gauge approximately the relative influence of the oxygen concentration and the temperature, a range of potential A/F ratios (to supply the molar concentration of oxygen at equilibrium conditions) and temperatures were placed into this equation with Figure 3 resulting. The temperature of the mixture is especially important as there is a non-linear relation between it and the rate of formation of NO. It is this relation that gives rise to the NO_X-efficiency trade-off. Thermodynamic principles state that the higher the temperature achieved within the cylinder, the more efficient the engine will be. However, higher temperatures result in much higher levels of NO_X, leading to the trade-off.

The formation kinetics of NO “freeze” below a given temperature inside the cylinder as the piston continues downward on the expansion stroke. It is also this kinetic freeze which causes diesels to produce a significant amount of nitrogen dioxide (NO_2). At light load, there is a considerable portion of the cylinder charge containing unused and relatively cool amounts of air mixing with the burning fuel. NO_2 is primarily formed in the flame front and can only be conserved by kinetic freezing, a process made easy by the generous amounts of cooler air at light load. Thus, concentrations of NO_2 can approach 10-30% of the

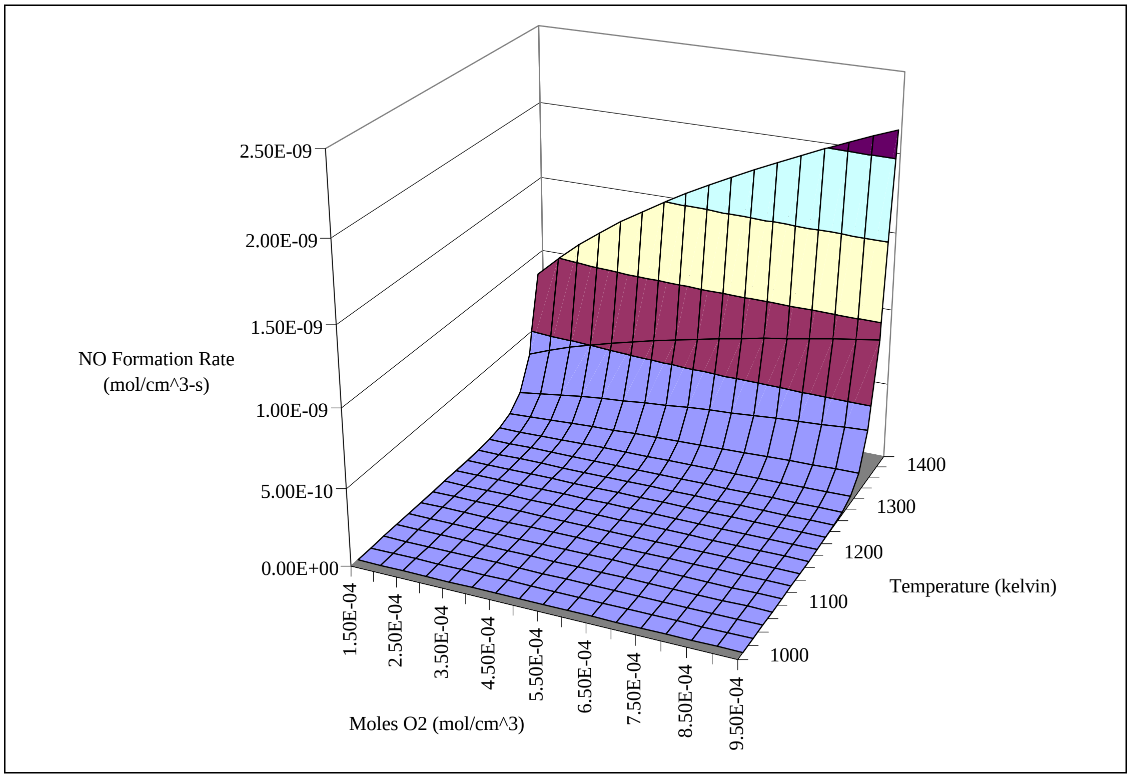


Figure 3: NO formation rate as a function of the in-cylinder temperature and molar concentration of oxygen. Based on experimental rate formation equation [Heywood, 1988, p. 575]. Moles of O_2 represent a lambda ratio of 1.2 to 20 (not represented linearly) at standard conditions.

overall oxides of nitrogen in a diesel at light load [Heywood, 1988]. Speed also plays a small role in NO_2 formation as lower speeds increase the residence time of NO with O_2 [Degobert, 1995]. All of these effects also demonstrate why NO_X formation can be correlated to the ignition delay. The length of the ignition delay will affect the pressure-volume phasing at top dead center producing a significant effect upon the temperature of the burned-gas region where NO_X forms. However, tests conducted with rapid sampling techniques indicate that the majority of NO has formed before half of the fuel has been fully combusted [Heywood, 1988, p. 590], relegating the combustion duration (the time between which 10% and 90% of the mass fraction of fuel has been burned) to a secondary factor in NO correlation.

A/F also plays a significant role in the production of NO_X, with the peak formation rate occurring at a point just lean of stoichiometric. This peak can be explained by the still fairly high combustion temperatures coinciding with the high availability of nitrogen and oxygen, which is why the peak does not occur at a point slightly rich of stoichiometric (where combustion temperatures are highest). As the combustion in an engine strays farther and farther into the lean region, the combustion temperatures decrease significantly and it is this effect which dominates the kinetics of NO_X formation. Typically, combustion temperatures in diesel engines can reach in the neighborhood of 2500+ kelvin.

Efforts to reduce the temperatures inside the cylinder to reduce NO_X formation can have unintended consequences when particulate matter is taken into account. Particulates form more readily at lower combustion temperatures and at points below about 500° C, "the particles become coated with adsorbed and condensed high molecular weight organic compounds" [Heywood, 1988, p. 627]. These counterproductive trends produce the so-called NO_X-particulate tradeoff.

2.2.4 Carbon Monoxide and Carbon Dioxide

Carbon monoxide primarily forms in fuel-rich environments where the oxygen required for complete combustion is insufficient. In a diffusion flame, these regions are common, and so CO forms readily in the early stages of combustion. However, because of the abundance of oxygen at typical diesel loads, most CO present in the cylinder will be readily oxidized as the cycle continues [Schafer and van Basshuysen, 1993]. This reduces the exhaust concentration to almost negligible amounts. There are conditions that can produce significant concentrations of CO, mainly occurring under full load. Primarily, the percentage of areas inside the cylinder that experience fuel-rich conditions is much higher under full load.

Carbon dioxide (CO_2) is one of the primary results of hydrocarbon combustion. CO_2 and water would be the only gaseous results of combustion in a perfectly stoichiometric reaction. Because of this, it can be used to gauge the efficiency of a combustion process. There are currently no guidelines covering the emission of CO_2, discounting indirect measures such as the corporate average fuel economy regulation. Concerns stemming from the controversial issue of global warming may prompt more attention towards the reduction of CO_2 emissions into the atmosphere.

3. Predictive Neural Networks and Combustion Modeling

3.1 Neural Network Architectures

Artificial neural networks derive their name from the biological system upon which they are based, namely the human brain. Conventional theory posits that the brain is composed of millions of neurons in a complex network with the capability of learning patterns to which it is exposed. An example of this is the human capability to identify distinct facial structures on fellow humans. Artificial neural networks attempt to mimic this pattern recognition ability. Additionally, they have the capacity to learn the traits of given inputs with respect to desired outputs provided a sufficient amount and scope of training data are supplied. They operate by learning the associations between input parameters, assuming that such associations exist.

There are many different types of neural network architectures based on different targeted results, but the style this study shall focus on is termed a feedforward network. Specifically, it will focus on a subset of feedforward networks called Error Back-Propagation (EBP) networks. These nets consist of several layers of neurons connected together as in Figure 4.

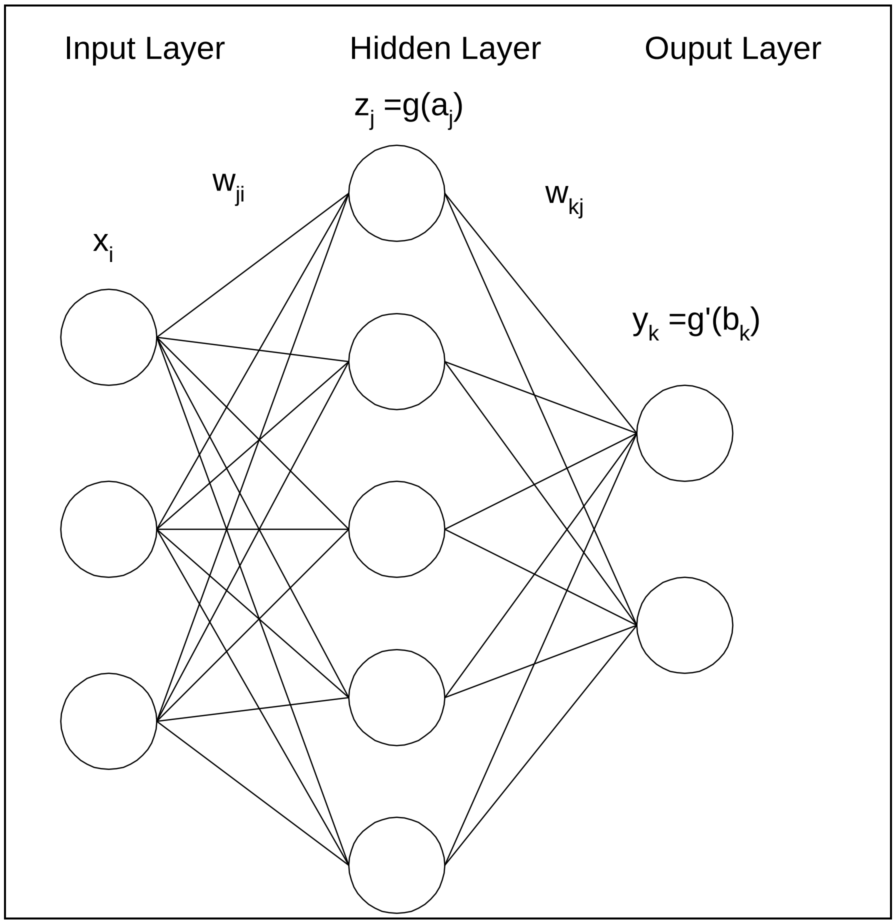


Figure 4: Schematic of typical back propagation neural network architecture.

The information from the first layer, the input, x, is multiplied by a weight, w, corresponding to each neuron and distinct for each link to the hidden layer, and then summed to provide an intermediate value. This value is then sent through a non-linear activation function to act as the hidden layer variable. The hidden layer is likewise multiplied, summed and fed through an activation function to provide the values of the output [Haykin, 1994]. Thus the two layers are written as:

$$a_j = \sum_{i=1}^{d} w_{ji} x_i + w_{j0} \quad \text{and} \quad z_j = g(a_j) \quad \text{(Eqs. 3.1 and 3.2)}$$

and

$$b_k = \sum_{j=1}^{m} w_{kj} z_j + w_{k0} \quad \text{and} \quad y_k = g'(b_k) \quad \text{(Eqs. 3.3 and 3.4)}$$

where d and m represent the number of neurons in the input and hidden layers respectively. The terms with the 0 in the subscript represent offset biases or thresholds used for better differentiation between classes. These classes are the categories in which the network places the output variables. For some networks, the classes are large (there would be two for a simple binary "yes-or-no" style output), but for the EBP, there can be very many according to the dimensional levels of the data set. The term g represents the activation function which can be based on a logistic or hyperbolic tangent function or any other non-linear function between the values of -1 and 1. As a result, data that is entered into the input layer is generally scaled to -1 and 1 with each extreme representing the minimum and maximum values expected to be encountered within each variable in the data set.

Once the network has been initialized through random weight seeding (all w in the equations above), the process of training begins. For EBP networks this involves comparing the actual measured output values to those predicted by the network (y_k) and then correcting the weights in all layers accordingly to improve the prediction. After one pass through the network, output values are compared to actual values to produce an error value. The amount of adjustment that each weight in the previous layer receives depends upon the initial error calculation and a learning rate parameter (typically on the order of 5% of the input scale) [Haykin, 1994]. Each layer has its corresponding weight matrix updated in this manner from the output layer backwards, thus giving rise to the EBP moniker. Once all values in the dataset have been calculated and compared against the actual values (one complete cycle through the data is known as an epoch), further training can continue until a predetermined level of accuracy has been achieved based on the training dataset.

After the network has been sufficiently trained, a test set of data must be introduced to confirm the network’s accuracy. This test set should include data that has not been previously “seen” or used by the network during training. Published results should generally contain comparisons from this test set as a means of gauging the accuracy of prediction using the network.

3.2 Applications of Non-Linear Approaches to Internal Combustion Engine Control

The automotive industry has embraced the revolution in electronics and digital technology that has been occurring over the past thirty years. Advancements have included microprocessor control of electronic fuel-ignition systems and the corresponding electronic circuitry found in the sensors that provide information to them. Aside from stricter efficiency regulations and competitive design meant to attract the consumer, the main impetus for these controls has been ever-tighter regulation of allowable pollutants stemming from engine combustion. These regulations have spurred increasingly complex methods of exactly controlling the A/F ratio to reach the goals that have been set. Lately, there have been forays into the realm of nonlinear control techniques to achieve these goals. This section will review the many approaches that have been made in this area as well as some other applications of neural networks to the control of various aspects of the internal combustion engine. The majority of the research in this area concerns the dynamics of fueling in the spark-ignition engine.

One approach to improve control of the A/F ratio involves careful modeling of the fuel injection process. Due to the fact that injection takes place outside the cylinder, the spark-ignition engine suffers from complex fueling and air intake dynamics that affect the overall A/F ratio that eventually combusts. A number of researchers have tackled this problem and applied various techniques to determine the correct ratio. Mainly, this is done to better control combustion during fast throttle openings where the exhaust gas oxygen sensor has too long a delay period to accurately reflect the true A/F ratio. Additionally, typical hot wire anemometers have long time constants that prevent immediate application in determining the A/F ratio.

Grizzle et al. [1994] developed a method of predicting the cylinder air charge by applying nonlinear differential equation modeling to the air intake dynamics and using a least squares fit line between two previous values and the current value to predict the next time step value. The cylinder air charge mass was calculated using values including the mass air flow, intake temperature, and engine speed. They reported better results using the predictor to adjust fueling levels over the standard method used on the test engine leading to approximately 10% reduction in HC emissions on the Federal Test Procedure driving cycle.

Shiraishi et al. [1995] applied a cerebellar model articulation controller (CMAC) to mass air flow determination in an effort to control the A/F ratio more accurately. The CMAC maps the input data into a binary indicator field which in turn activates certain weights in the next layer according to their binary values. The outputs are then the algebraic sum of the corresponding activated weights. The researchers argue that the normal back propagation network is too computationally intense and converges too slowly for their application. The CMAC is used to determine the mass flow rate of air based on the two inputs of manifold pressure and engine speed. Results of the study indicate performance equivalent to, if not better than, stock electronic control modules.

A popular approach to the problem involves intensive modeling of both the air intake system and the fuel injection process, especially with regard to deposition and evaporation rates of fuel on the manifold surface. Hendricks et al. [1992] tackled this problem and noted that most modern control systems fail to “compensate for the nonlinear dynamics of the fuel film” or to “estimate correctly the air mass flow at the location of the injector.” They built up a set of nonlinear equations based on the dynamics of these areas, then applied conventional control techniques to account for the fuel film and an unconventional nonlinear observer to predict the engine's volumetric efficiency. Jones et al. [1995] applied a nonlinear least squares algorithm to modified forms of discretized system model equations to predict the values of fuel and air rates one time step ahead. The suggested method requires the storage of values indicating the amount of fuel entering the cylinder as a fraction of that injected versus engine load and speed. With those values as the reference, the recursive nonlinear algorithm performed well. The authors conclude that “the results… illustrate the excellent air-fuel ratio regulation which may be achieved by the closed-loop AFR control system when the model parameter values are accurately known.” Similarly, Bidan et al. [1995] use complex throttle dynamics equations to predict the amount of fuel inducted from all sources (injectors, evaporation from manifold walls, scavenged from previous cycle, etc.) for the upcoming cycle. The prediction is based upon flow behavior and the derivative of the pressure history at the current time step.

Lenz and Schroeder [1997] implement a three-state model to describe the system. The three states consist of one for the air pressure in the intake manifold, one to account for the unvaporized fuel in the manifold, and the last based on the engine speed. The authors develop relations for each state and produce potential inputs to a Radial Basis Function (RBF) neural network, without actually implementing them due to the highly parallel processing nature of the network. They do, however, offer prospective research topics based on their work.

In an earlier work, Lenz and Schroeder [1996] detailed a method for breaking down the relationship between throttle angle, air mass, and fuel mass inducted into the engine to produce a given torque at a proper A/F ratio. Their method involved predicting the amount of air that should be inducted in the next engine cycle based on the driver's manipulation of the throttle. Upon gaining knowledge of the future air mass, the correct fuel mass could be injected onto the back of the intake valves in anticipation. The authors used General Regression Neural Networks (GRNN) to learn the various interdependencies with neuronal populations ranging from 15 to 400.

Another approach to nonlinear control of the A/F ratio utilizes a technique called the sliding mode observer. The observer in this case is an arbitrary variable set equal to the difference between the mass air flow and the product of fuel flow and the stoichiometric value. The ideal case for the variable is then the value zero and the "sliding" portion of the name is derived from a differential with respect to time that adheres to Lyapunov stability which states that a slight disturbance to the system will not cause chaotic perturbations, but rather that the system will return to a stable condition. Carnevale et al. [1995] demonstrated the effectiveness of this technique by maintaining the A/F ratio within 2% of stoichiometric. Prior to them, Kaidantzis et al. [1993] used this method with success to estimate and compensate for the transient fuel and air flow dynamics at the intake.

A different approach was formulated by Takahashi [1996], who presented an algorithm for determining the correct amount of fuel to inject based on a data table produced by applying the δ-rule, a basic learning law applied to neural networks. Essentially, the table values are compared against target values during operation and updated through a partial differentiation scheme. The table values are then used to predict the amount of fuel to be injected to achieve the desired A/F ratio.

Shayler et al. [1996] decided to partially eliminate the complex differential equations describing the amount of fuel delivered to the cylinders by employing a multi-layer neural network. The network works toward predicting the amount of fuel actually inducted into the cylinder based on three input conditions: air mass flow, coolant temperature, and the fuel film mass on the manifold walls. The last input is determined by considering the fuel injected during the last cycle and solving the differential equations. The authors used a set of 9 transient data sets, placed 5 nodes in the hidden layer (for a 3:5:1 model), and trained the network over approximately 260 epochs. Results indicated acceptable performance with the benefit of vastly simpler calibration procedures, the elimination of a number of look-up tables, and fast development time.

Due to the necessity of open-loop fuel management, cold starting of an engine offers an opportunity for neural network application. Asik et al. [1997] utilized a multi-layer network with back-propagation containing eight inputs and two hidden layers of eight nodes each to produce one output signal, the A/F ratio. Besides typical input parameters like engine speed and load, the authors used a metric derived from the crankshaft fluctuations induced by rapidly altering the fuel ignition pulse width. In this manner, data sets were gathered for three different states: lean, rich, and stoichiometric fueling conditions. The results for open-loop management indicated good tracking of the network compared to actual values supplied by a universal exhaust gas oxygen (UEGO) sensor, but with an underprediction in the absolute values of the A/F ratio.

Crucial to the previous modeling projects is the correct identification of the volumetric efficiency of an engine. De Nicolao et al. [1996] set out to do just that, comparing different techniques of producing the volumetric efficiency including neural networks and polynomial fitting. The authors used speed and manifold pressure as the input variables for prediction purposes. They determined that the various models used, including a multilayer back-propagation neural network, were quite useful in accurately modeling volumetric efficiency. A caveat offered by the authors concerning the back-propagation network detailed that the final results of the volumetric efficiency map were sensitive in one area to the initial conditions used to train the network. In other words, the network could get trapped in local minima if the initial neuron weights were chosen incorrectly. They suggest training the network with a variety of initial weights to avoid such a problem and ensure higher accuracy.

Ramli and Morris [1993] applied a back-propagation network to the problem of non-linearities in fuel injector and throttle characteristics. They also noted that a more complicated network could treat the entire engine as a black box function provided that all pertinent information was supplied (which was not the case for their study. Their main thesis was that complex modeling of nonlinear dynamics suffered from the need for great accuracy and that only slight misrepresentations could alter the effectiveness negatively. They showed that a simple three-layer network could provide acceptable accuracy for fuel injector throttle dynamics without extensive analytical techniques, provided enough training was performed. Additionally, they pointed out that neural controllers have great robustness and high fault tolerance in case of minor internal failure by virtue of the spreading out of information through the interconnectedness of the neurons.

Idle engine speed control presents another problem potentially solvable through the application of neural networks. Salam and Gharbi [1996] attempted to do so by developing a static neural network to determine throttle and spark advance settings, and then feeding those signals into a recurrent, or time dependent, network to produce targeted settings of engine speed and manifold pressure. To ensure stable control environments, the recurrent network was augmented with basic filters, producing large degree of success. Idle engine speed control was also an indirect goal of work by Feldkamp and Puskorius [1994]. Their research focused on increasing the performance of neural controllers through a process they call multi-streaming. Essentially, they trained the controller on different data streams provided by separate source plants while keeping the trained weights. As each data stream came to an end, the weights were remembered and the data stream was restarted from a different source. In this manner, the controller can be trained on different plants exhibiting different operational states, thus avoiding an overly plant-specific trained network.

The spark-ignition engine is not the only platform that can benefit from neural network application. Diesel engine injection and the compression-ignition engine also can benefit from the application of nonlinear control. Kao and Moskwa [1995] applied nonlinear observers to estimate cylinder pressure values from speed signals taken from the crankshaft. Their method involved utilizing a cylinder pressure dynamic equation to provide the basis for the observer. Obtaining this, the authors were able to reproduce the pressure profile within acceptable limits from which they then could infer fuel burning rates.

In an effort to eliminate thermal drift within piezoelectric pressure sensors, Leonhardt et al. [1995] used what they call the “difference pressure”, defined as the resultant pressure differential when the compression pressure is subtracted from the measured pressure inside a combusting cylinder. They used variables derived from this pressure along with the engine speed as inputs to a neural network in order to predict mass of fuel injected and the crank angle for start of injection for a direct-injection diesel engine. They found that both CMAC and RBF networks were comparable in performance with a usual divergence from the test signal of no more than 10%, although in a later work the RBF was chosen as superior [Leonhardt et al., 1995]. They also used this information to determine misfire and to identify the associated cylinder. Their work is the closest that could be found during the literature review to the work done in this study. In another study designed to identify faults in diesel engine turbochargers, Ludwig and Ayoubi [1995] employed a multi-layer network with engine speed and fuel injection mass as inputs to predict the loading pressure of the turbocharger. The network performed with enough accuracy that deviations from the estimated pressure could be further classified to denote turbocharger faults. At the time of printing such classifications had not been available for inclusion in their findings.

In a paper on general misfire in automotive engines, Ribbens et al. [1994] used a three-layer back-propagation network to predict a Fourier component of the crankshaft angular speed. By computing certain statistical components from this and also defining a threshold level, misfiring cylinders could be identified. The network was set up with nine inputs, one output, and 25 neurons in the hidden layer. The authors claim error rates (i.e. misdiagnosed misfiring cylinders) of 0.006%, potentially qualifying the method of identifying misfire for stringent OBDII requirements.

Recently published research shows that neural networks are becoming increasingly useful tools for many and varied applications. Their strength lies in greatly simplifying complex non-linear dynamic relationships fairly quickly. The only drawback appears to be the typically lengthy time required for adequate training. With the performance of high speed digital computers increasing constantly, there is reason to believe that neural networks will and should find a place in high speed engine control, emissions, and performance prediction and diagnostics.

Furthermore, a review of the available literature indicates that there are no examples of in-cylinder pressure-based emissions prediction neural networks. The only publications that exist on the topic of emissions prediction have been written by members of NeuroDyne, Inc. and West Virginia University [Atkinson, et al., 1998; Hanzevack, et al., 1997].

3.3 Combustion Modeling

There are two basic types of numerical modeling techniques for the diesel engine: simple thermodynamic and multidimensional [Ramos, 1989]. The thermodynamic techniques can be further split into single-zone and multi-zone models. Both rely on simple thermodynamic assumptions and equations and treat the contents inside the cylinder as one or several separate control volumes. The multidimensional models are based on intensive numerical calculations in both the spatial and temporal dimensions.

The benefits of the simple thermodynamic models are the relatively quick solution times involved and the straightforward nature of the equations. Unfortunately, these models aren't quite as accurate as the more involved multidimensional models. Increased accuracy comes at a price, however, as these models are computationally intensive.

In a comparison of numerical techniques applied to a Saturn SI engine, Yang et al. [1994] demonstrated that a multidimensional model (KIVA-II) could successfully calculate levels of gaseous emissions for an engine operating at steady state within reasonable limits of accuracy. A zero-dimensional model, however, didn't fare as well, as it erratically predicted gaseous emissions for four different steady-state conditions. Additionally, both techniques were dependent upon the successful approximation of unknown quantities such as the cylinder wall temperature.

Similarly, Tauzia, et al. [1997] applied a two-zone model to data from a direct injection diesel engine. They found that the model could predict values for NO_X within 20% or so of the experimentally obtained data for a series of steady-state conditions as well as a few transient conditions.

However, Muller and Zillmer [1998] evaluated in-cylinder data from a diesel engine to adjust the parameters in a two-zone model to achieve realistic combustion temperatures and consequently predict nitric oxide and soot formation rates at different points in the combustion cycle.

Rutland, et al. [1995] detailed efforts to apply multi-dimensional modeling code (KIVA-II and KIVA-III) to achieve more accurate modeling of the diesel combustion process. They admitted that true modeling would not be obtained in the foreseeable future due to the vast amount of computing power required (grids with 10^{12} points would suffice, but current supercomputer technology can only practically compute 10^5 points). Consequently, they used a wide range of sub-models to achieve good results comparing their predictions of NO_X and soot levels with those obtained experimentally.

Overall, numerical modeling of the combustion process in the diesel engine produces excellent results for multi-dimensional simulation code. Thermodynamic techniques also produce good results, although not as accurately as the more computationally intense multi-dimensional models. Both rely on approximations and experimental data to fine-tune the equations and methods involved. No mention has been made in the literature as to the amount of time needed for these calculations. It is quite possible that computer power and the modeling techniques have not advanced to a stage that permits real-time prediction of gaseous emissions. However, the usefulness of numerical modeling for engine development and other off-line applications cannot be understated.

3.4 Virtual Emissions Sensing at West Virginia University

NeuroDyne, Inc. has developed a virtual emissions sensing system, a software-based neural network that has the ability to predict levels of emissions generated by a spark-ignition engine. The efficacy of this system in predicting engine-out emissions has been demonstrated by West Virginia University during work conducted at the Engine Research Center on a Saturn automobile engine. Additionally, there has been previous work performed on a Lycoming aircraft engine [Hanzevack et al., 1997]. The network employed a three layer recurrent EBP scheme with nine inputs, fifteen neurons in the hidden layer and four outputs. The outputs consisted of the levels of emission of four gases: hydrocarbons, carbon monoxide, carbon dioxide, and oxides of nitrogen (both nitric oxide and nitrogen dioxide). The nine inputs consisted of a range of widely monitored parameters such as the manifold air pressure and temperature and specially instrumented parameters like the exhaust temperature. After training of several hours was performed off-line by Dr. Theresa Long, the network was capable of accurately predicting the levels of emissions (Figures 5-8). The speed and load that the engine was taken through to generate the data for Figures 5-8 are presented in Figure 9. A schematic of the system is provided in Figure 10.

Testing was accomplished with a 1992 Saturn 4-cylinder in-line 4-stroke spark-ignition 16 valve DOHC engine of 1.9 liter displacement. Load was applied with a water brake dynamometer (Superflow SF-901) and measured with a load cell mounted on the dynamometer housing. The dynamometer load cell was calibrated using a lever arm and a standard weight. The cooling of the engine was accomplished with a radiator and a fan controlled to operate at temperatures above 103 °C. The stock thermostat was employed, and the cooling system was pressurized to 16 psig (110 kPa gauge). A list of engine parameters recorded and the instruments used are provided in Table 1. Table 2 lists the stock specifications for the test engine.

The laboratory facility at West Virginia University was used for the study and many hands were involved in the construction and setup. A general overview of the equipment used follows.

The stock exhaust system was employed without the catalytic converter. The exhaust was routed to a dilution tunnel where the exhaust gases were diluted with air from the room and emissions of HC, CO_2, CO, and NO_X were measured (Table 3). The exhaust gas and dilution air were pulled by blowers through a 400 scfm metering venturi operating at a "choked" condition. Ambient temperature and pressure were monitored continuously at the venturi inlet to allow calculation of the true mass flowrate of the gases in the tunnel. Background data were taken to subtract the amount of measured gases already in the dilution air from the total measured amount, yielding the true quantity of emissions in the exhaust. The air to the engine and the dilution air were not conditioned, but measurements of ambient temperature, pressure, and relative humidity were taken before each test.

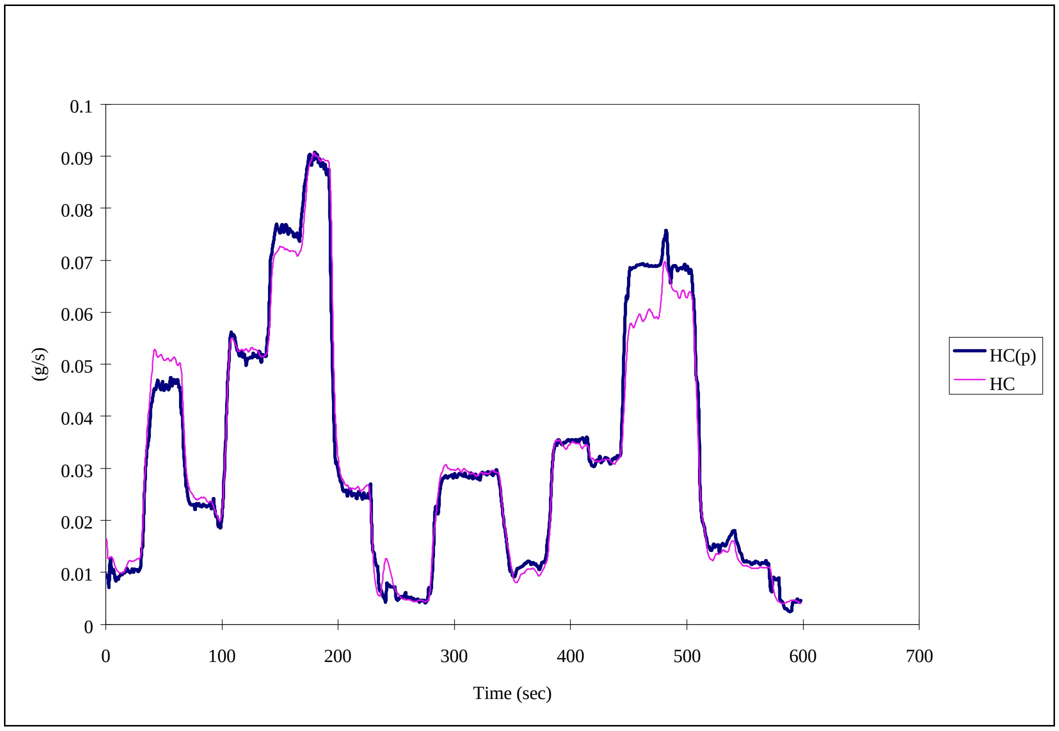


Figure 5: HC emissions, predicted vs. actual, Saturn 1.9L engine operating on gasoline [Hanzevack, et al. 1997].

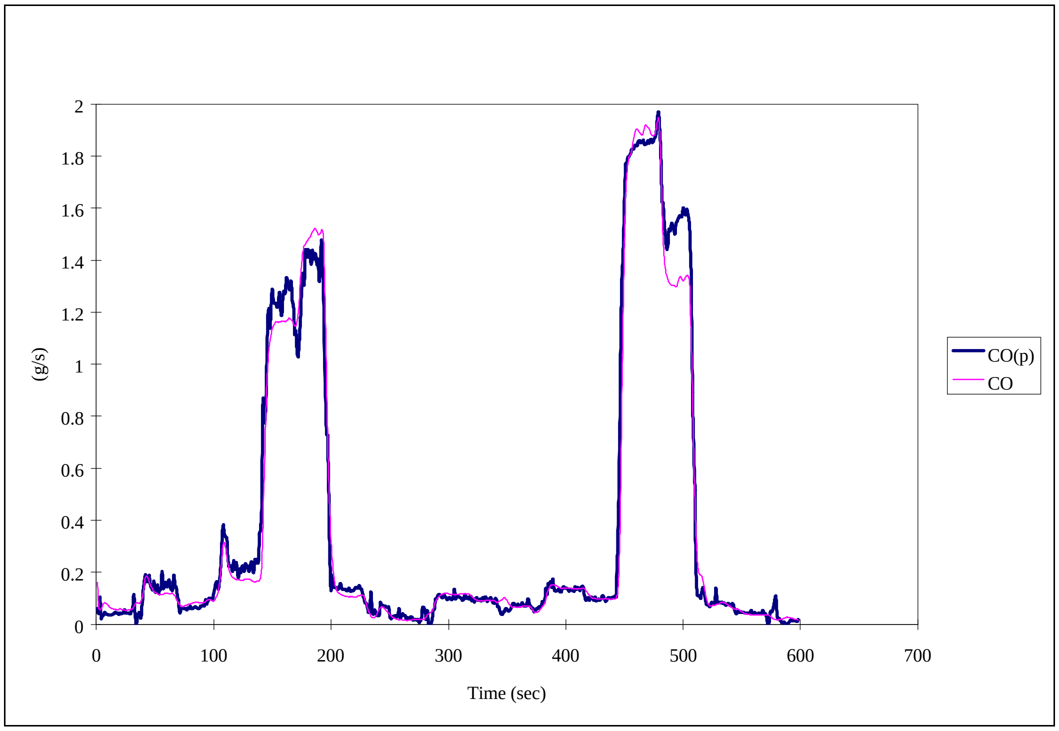


Figure 6: CO emissions, predicted vs. actual, Saturn 1.9L engine operating on gasoline [Hanzevack, et al. 1997].

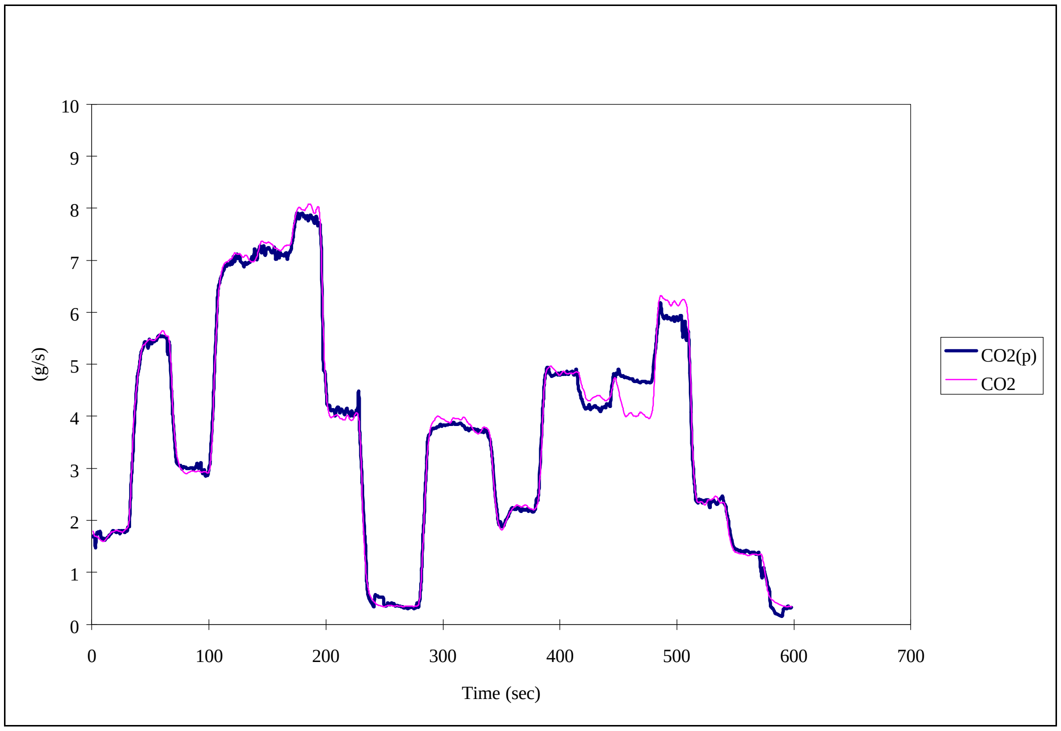


Figure 7: CO_2 emissions, predicted vs. actual, Saturn 1.9L engine operating on gasoline [Hanzevack, et al. 1997].

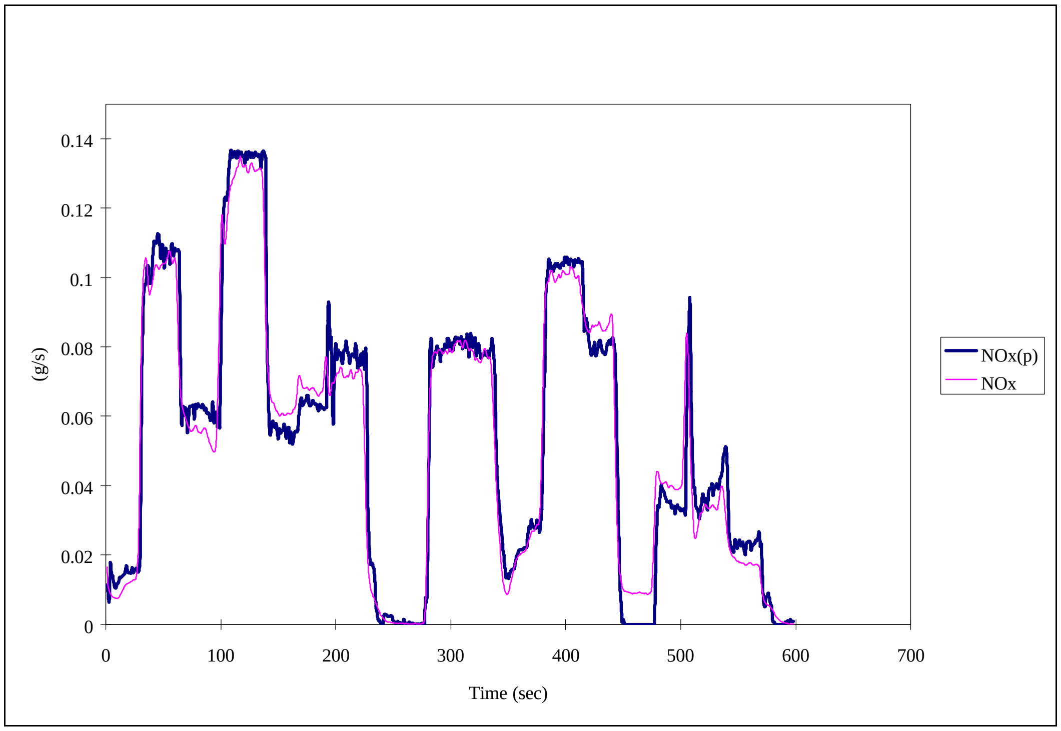


Figure 8: NO_X emissions, predicted vs. actual, Saturn 1.9L engine operating on gasoline [Hanzevack, et al. 1997].

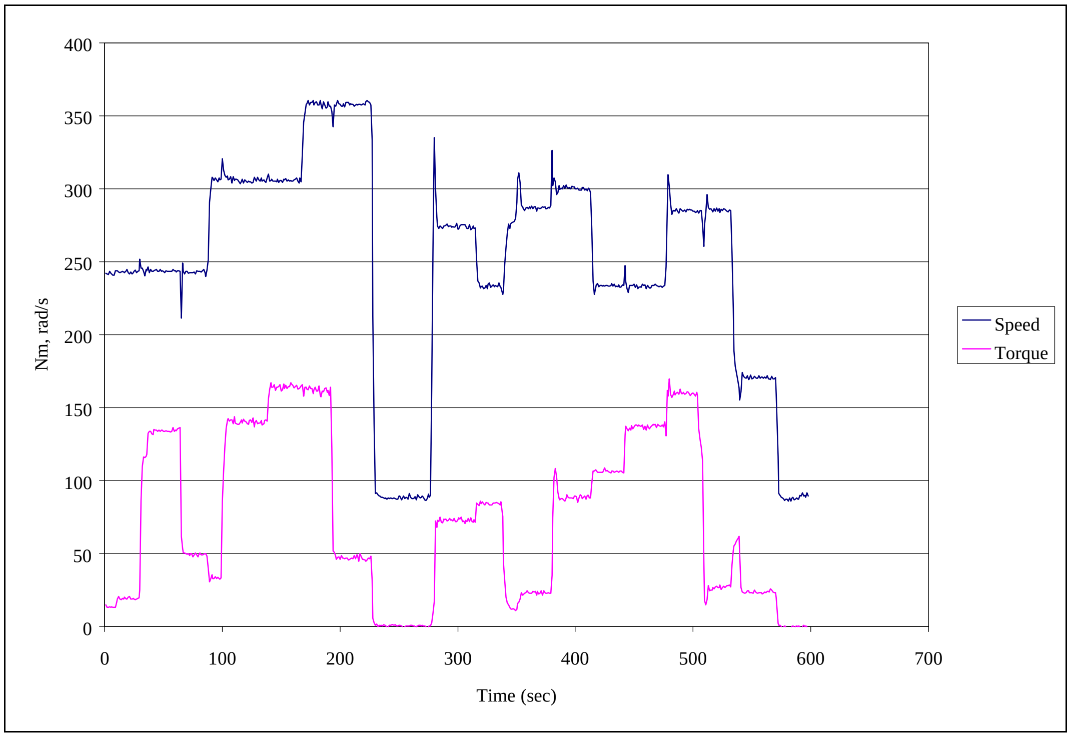


Figure 9: Speed and load map for the neural network predictions in figures 5-8, Saturn 1.9L engine operating on gasoline.

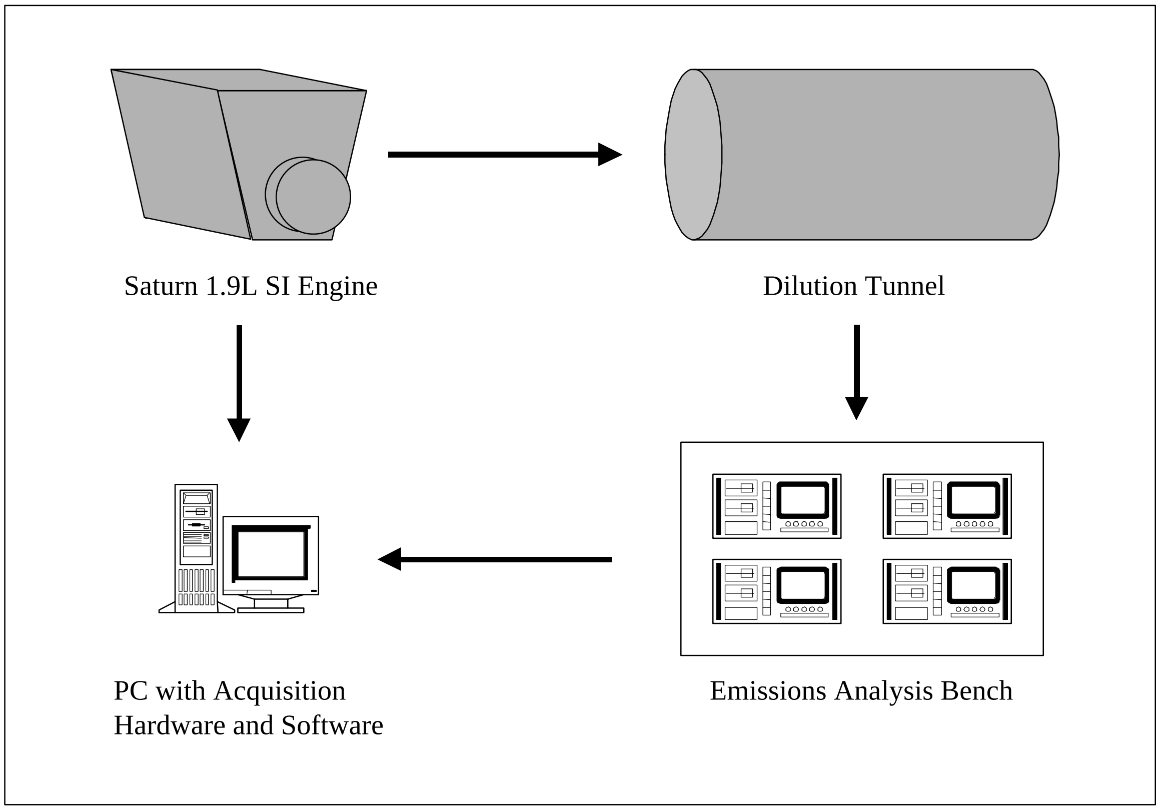


Figure 10: Schematic of emissions measurement system for Saturn 1.9L engine [Hanzevack, et al. 1997].

Table 1 : Saturn Engine Instrumentation [Tennant, et al. 1994]

Instrument	Model
Exhaust Thermocouples	Omega K-Type, ungrounded #TJ36-CASS-18(U)-12
Manifold Air Pressure Sensor	GM Part # 21020103
Manifold Air Temperature Sensor	GM Part # 21020104
Universal Exhaust Gas Oxygen Sensor	NTK:NGK Spark Plug Co. Part # TL-7113-A1
Throttle Position Sensor	GM Part # 21020101
Coolant Temperature Sensor	GM Part # 21020104
Opticoder (Ignition Advance)	Sumtak Opticoder LEI-292-1024
Tachometer	Superflow Magnetic Pickup
Fuel Injectors	AC Rochester

Table 2 : Stock Specifications for 1992 Saturn (Serial # 12NILE337976) [Tennant, et al. 1994]

Type:	Water cooled-4 cycle
Cylinder Arrangement:	4 cylinder Inline, Transverse
Bore and Stroke:	82.0 mm x 90.0 mm (3.2 in x 3.5in)
Compression Ratio	9.5:1
Displacement	1.9 liter (116 cu. in.)
Valve Train	4 Valves per cylinder - DOHC
Lubrication System	Pressure Feed
Fuel	Chevron Unleaded Gasoline, 89 Octane (R+M)/2
Engine Wet Weight (No Accessories)	99.8 kg (220 lb)

Table 3 : Emissions Bench Instrumentation [Tennant, et al. 1994]

Instrument	Model
HC Analyzer	Rosemount Analytical 402 FID
NO_X Analyzer	Rosemount Analytical 955 Chemiluminescent
CO Analyzer	Beckman Industrial 868 NDIR
CO_2 Analyzer	Beckman Industrial 868 NDIR

A data acquisition system, incorporating a Keithley DAS-16 data acquisition board with a 12 bit A/D converter, was used to record the emission values, engine speed, manifold air temperature (MAT), throttle position (TPS), coolant temperature (CLT), exhaust temperature, fuel injection pulsewidth (FIPW), and λ (the fuel to air ratio normalized to stoichiometric conditions) via a universal exhaust gas oxygen (UEGO) sensor. The UEGO was necessary because it has a much wider authority range than the conventional exhaust gas oxygen (EGO) sensor used by the stock controller. Engine speed, ignition advance and fuel ignition pulse width were acquired by a Keithley Metrabyte CTM-10 10-channel 16-bit counter-timer board. All values were acquired at the rate of 20 Hz. The data were acquired and stored with an IBM-compatible 133MHz Pentium computer.

Prior work for unrelated projects had already produced working data acquisition code in the BASIC language, written by Richard J. Atkinson and Remco J. deJong. For compatibility reasons this code had to be altered and rewritten in C++. Additionally, major portions of the data acquisition program had to be added, including a direct unit conversion section to provide real unit data and a buffer section to account for delays in the emission analyzers. Because the analyzers do not have instantaneous response and because of the presence of approximately 50 feet of exhaust piping and dilution tunnel between them and the engine, the emissions values reported have to be adjusted in time to correspond to relevant engine behavior. This adjustment was made by feeding all other engine parameters into a large memory matrix that effectively buffered the data by varying amounts. The only variable that the neural code encounters in real time is NO_X since it has the longest analyzer delay time (Table 4). A flowchart demonstrating the buffer is shown in Figure 11.

Table 4 : Emission Delays

Hydrocarbons	9.25 seconds
Carbon Monoxide	11.75 seconds
Carbon Dioxide	11.75 seconds
Oxides of Nitrogen	14.00 seconds

Unfortunately, the main code for the virtual sensing system had been written in C on a Unix workstation and a direct port to the PC was not possible. NeuroDyne, Inc. had the main code converted into a MS-DOS compatible format. This code was then combined with the C++ version of the acquisition code. Although the results of the NeuroDyne study were very good, there were some problems with the method selected. During training of the network, the inputs are weighted, summed, sent through an activation function, and then compared against the actual values that are to be predicted. In the case of the measured emissions, the actual value was somewhat compromised because of the exhaust system/dilution tunnel lag. This was particularly evident during transient operation of the engine. Consequently, for

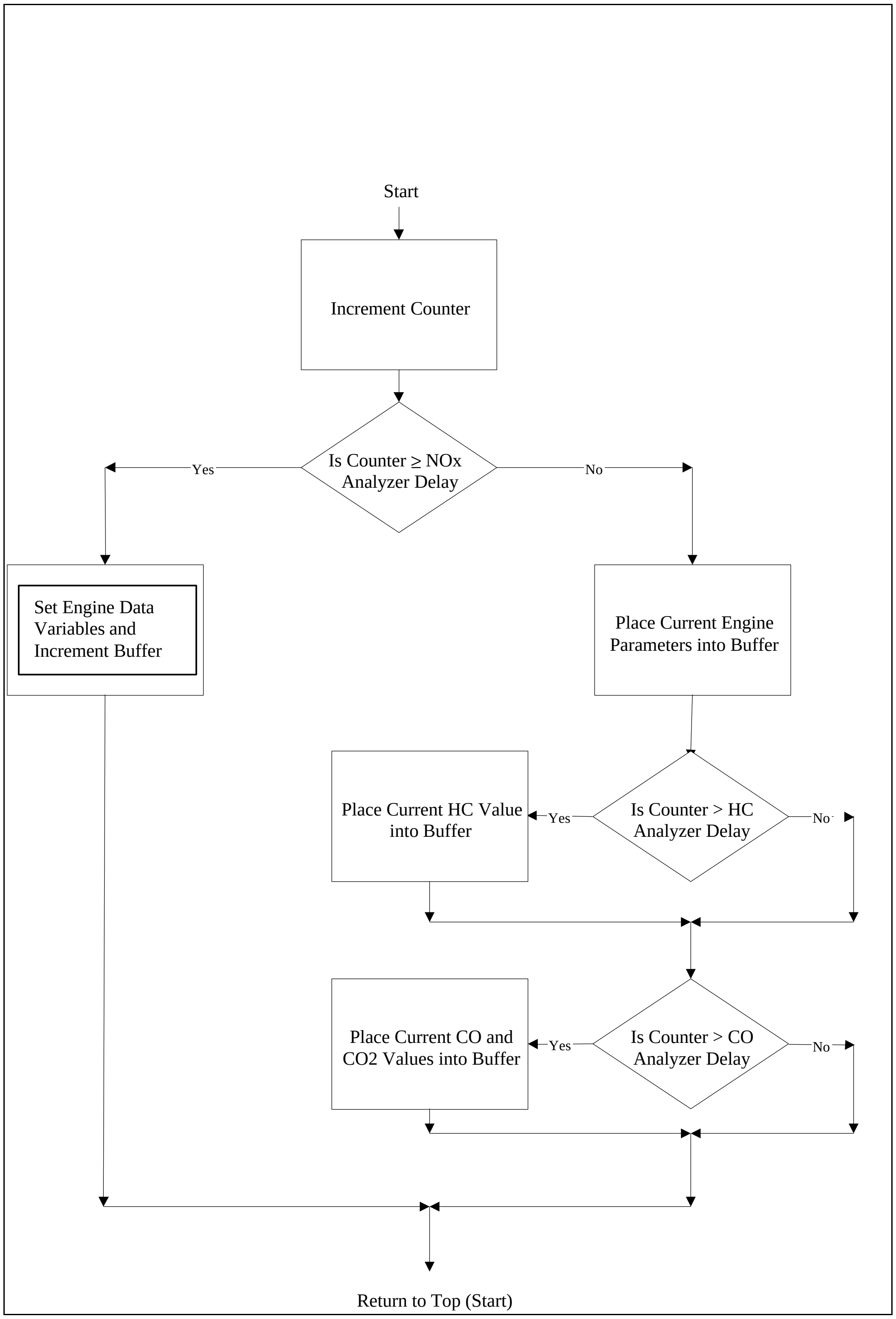

Start
Increment Counter
Is Counter ≥ NOx Analyzer Delay
Yes
No
Set Engine Data Variables and Increment Buffer
Place Current Engine Parameters into Buffer
Place Current HC Value into Buffer
Yes
Is Counter > HC Analyzer Delay
No
Place Current CO and CO2 Values into Buffer
Yes
Is Counter > CO Analyzer Delay
No
Return to Top (Start)

Figure 11 : Flow chart diagram for the analyzer signal buffer algorithm.

training purposes, the data had to be time-shifted backwards in order for the input variables to match the output variables. This shift was determined by gauging the "lift-off" point for torque or manifold air pressure and then comparing that point to the corresponding "lift-off" of the gaseous emission in question. There was no guarantee that this constant shift was accurate over the entire data set. The impact of the dilution tunnel system is that the networks ultimately treat the engine, exhaust system, dilution tunnel, and analyzers as one large "black box". The recurrent nature of the networks and the fact that they use trends in time to aid prediction tempered this problem somewhat, but the time shift issue persists.

The choice of typical engine controller related parameters as the primary inputs was a logical one, but also suffered from a time-related problem. Basically, extra-cylinder parameters that involve temperatures will themselves have a small time constant effect due to the heat transfer characteristics of the engine materials. These parameters will still be related to the gaseous emissions produced, but not in a direct manner. The combustion in the cylinder will govern the emissions levels, while the extra-cylinder parameters will simply reflect the combustion inside the cylinder.

The end result is that the predictions will only be as good as the input parameters' relationship to emissions formation and the analyzer/tunnel system response to emissions production. This may be adequate for overall prediction of a dilution tunnel system, but may fall just short of on-board diagnostic capability to determine misfire, for instance. For these reasons, for this work an in-cylinder based pressure-measurement system employing a steady-state test matrix was proposed as a better method of predicting engine emissions.

4. In-Cylinder Pressure Sensing

4.1 General Overview

Generally speaking and in this author's opinion, the most informative signal readily available from an internal combustion engine comes from the in-cylinder pressure profile. Since this signal reflects the actual conditions of the engine while it is running, it can potentially provide a plethora of pertinent information concerning the characteristics, behavior, and overall health of an engine. The main drawbacks to gathering this information are the invasive nature of the required sensor and the relative lack of robustness of most such sensors. Therefore widespread adoption of the in-cylinder pressure transducer for engine control and diagnostics has not yet occurred. As a research tool, however, these drawbacks are secondary and negligible compared to the usefulness of the information provided.

The most common method of measuring the pressure inside the cylinder utilizes a piezoelectric crystal-based pressure transducer. These devices generate a signal depending upon the pressure force imposed upon the crystal in a relative sense. In other words, the signal will not be in absolute terms, so a baseline pressure must be established in order to attain engineering units. Transducer installation typically involves drilling a hole through the head of the engine and placing the transducer with its sensing tip flush with the surrounding combustion chamber surface. Non-intrusive devices are available that take advantage of the spark plug position in spark-ignited engines, but because of the sampling tube that extends down the side of the plug these devices suffer from adverse pressure wave effects due to their distance between the pressure source and the sensing element. For some applications where magnitude is of minor importance but phasing is critical, such as in peak pressure timing control, a force washer positioned under the spark plug provides adequate information. For the application considered for this research, however, an in-cylinder transducer is the ideal solution.

One of the problems with these transducers is thermal drift: a tendency for the sensor signal to slowly drift upwards or downwards with changes in temperature. It can cause trouble if the pressure signal is “pegged” to a constant offset for the purpose of translating it into absolute pressure units. Methods for accounting for thermal drift have been outlined by Brunt and Pond [1997], who concluded that referencing the pressure signals to the inlet manifold pressure was adequate, although it did suffer some drawbacks. Their other alternative used a polytropic index based upon the compression characteristics, but suffered from sensitivity to signal noise. Due to the testing conditions, which inject a moderate amount of noise into the signal, and the need for low computational load (a subject which shall be discussed later in this work), the inlet manifold referencing was used here. Other considerations for working with pressure transducers include the resolution of the digitized signal and proper phasing between the cylinder volume and pressure signal to ensure the proper location of top dead center of the piston.

For the current study, the pressure transducers were flush mounted in cylinders 3 and 5 in the spaces formerly occupied by the glow plugs. The work of physically installing the transducers was performed by Talus Park. The TDC position of cylinder 1 was then determined using standard methods with motoring traces recorded for closer analysis. The optical encoder mounted on the front of the crankshaft was then adjusted until top dead center had been found to within 1 degree of crank angle. The motoring curves were analyzed for phasing problems and mechanical deficiencies; when none were found, work proceeded. The diagnostic capabilities of the motoring P-V curve can be found in the paper of Lancaster et al. [1975] and in a previous work by this author using a Saturn 1.9 L engine [Traver, 1994].

4.2 In-Cylinder Pressure Signal Derived Parameters

By far the most direct and convenient variables available from an in-cylinder pressure profile are the in-cylinder peak pressure and its location. Studies have shown the relation between the location of the peak pressure and engine efficiency [Tennant et al., 1994], and controllers have been developed to take advantage of this fact as well as to reduce the amount of emissions produced [Atkinson et al., 1994; Pestana, 1989; Watanabe et al., 1996]. Algorithms for finding these parameters from a digitized pressure signal are quite straightforward with a simple comparative if-then statement.

The indicated mean effective pressure (IMEP) is a measure of the work done inside the cylinder normalized to the cylinder volume. The amount of work that the piston cylinder assembly actually undergoes is the same as a similar system would undergo at a constant pressure (the IMEP). IMEP is calculated by integrating the work performed on the piston numerically and dividing by the displacement volume. Because the exhaust cycle is omitted, the subscript “g” is often placed onto IMEP to indicate the gross amount of work done. This is the designation that shall be used in this paper. The main drawback to calculating IMEP is that the instantaneous volume must be known throughout a complete cycle and inaccurate geometric measurements can create problems.

Two parameters that are calculated similarly to one another are the combustion duration and ignition delay. Ignition delay provides a measure of the time between the fuel injection into the cylinder and the time at which discernible combustion starts within the cylinder. Discernible combustion has been defined in a number of ways (varying from the point at which the pressure history deviates from a theoretical motoring curve to the point at which either 1% or 10% of the mass fraction has burned) depending upon researcher preference. Algorithms for this determination rely upon the First Law of Thermodynamics or a polytropic compression-and-expansion process to calculate the instantaneous heat release rate at each crank angle [Traver, 1994; Brunt and Emtage, 1997]. For this study, the determination of the ignition delay period was chosen as approximately the time from start of injection to the 1% mass fraction burned point. The value is not actually 1% as the computer algorithm searches for a significant summation of positive heat release from the following First Law derivation and treats the first digital point of that summation as the end of the delay. Additionally, recommendations for adequate statistical burn-rate analysis include resolutions of at least 1 degree and populations of at least 150 combustion events [Brunt and Emtage, 1997]. For this study, combustion event populations numbered 128 due to hardware limits and the crank angle resolution was approximately 1/3 of a degree.

The combustion duration measures the time, usually in crank angle degrees, during which combustion is taking place within the cylinder. Generally, and due to measurement and accuracy constraints, the combustion duration is defined between two set values of the percentage of the mass fraction burned. For the purposes of this study, the range shall be between 10% and 90% of the mass fraction burned. The algorithm used for this parameter is also based on the First Law of Thermodynamics, and is the same as that used for determining the ignition delay. In practice, the incremental heat added is summed over a range of data points to provide an overall energy quantity and then looped through again to determine the points at which 10% and 90% of combustion have occurred.

The basic algorithm to determine heat release assumes a quasi-static condition within the combustion chamber, or that the combustion occurs at a uniform temperature and pressure. Unfortunately, the diesel direct-injection process presents problems with this assumption and consequently any method of analysis based on it can only give approximate answers. Problems in applying the assumption include the presence of crevice volumes that do not behave quasi-statically, the burned gas composition (which is unknown), non-uniform air-fuel ratios during combustion, and inaccuracies involved in the prediction of heat transfer to the walls of the cylinder [Heywood, 1988].

The First Law of Thermodynamics can be written as:

$$\frac{dQ}{dt} - p\frac{dV}{dt} + \sum_i \dot{m}_i h_i = \frac{dU}{dt} \qquad \text{(Eq. 4.1)}$$

where Q is the heat transfer parameter, p is the in-cylinder pressure, V is the volume, m_i is the mass of fuel injected, h is the enthalpy, and U is the internal energy. During the compression and expansion phases of the cycle the only mass crossing the system boundary is the fuel injected and so the mass-enthalpy term reduces to a "mass of fuel" enthalpy term. Assuming that the enthalpy and internal energy are sensible terms (using a baseline of 298 kelvin; denoted as *e*) and that the net heat released is defined as the difference between the energy released through combustion and the energy lost to heat transfer from the system (denoted as *n*), the equation can be rewritten as:

$$\frac{dQ_n}{dt} = p\frac{dV}{dt} + \frac{dU_s}{dt} \qquad \text{(Eq. 4.2)}$$

The heat loss through the system boundary presents a problem only at the end of combustion where temperatures have risen significantly. Studies have shown that the energy lost due to wall heat transfer does not affect the ignition delay parameter significantly under normal operating conditions [Brunt and Emtage, 1997; Heywood, 1988]. The fuel enthalpy difference is sufficiently small so as to be negligible due to the small mass of fuel compared to the working fluid and the small temperature difference between the fuel and the baseline 298 kelvin. Next, the system is assumed to behave as an ideal gas, giving:

$$\frac{dQ_n}{dt} = p\frac{dV}{dt} + mc_v\frac{dT}{dt} \qquad \text{(Eq. 4.3)}$$

where c_v is the specific heat at constant volume. Differentiation of the perfect gas law with R, the universal gas constant, provides a means of eliminating the temperature term which is generally unavailable in pressure analysis:

$$\frac{dQ_n}{dt} = \left(1 + \frac{c_v}{R}\right) p \frac{dV}{dt} + \frac{c_v}{R} V \frac{dp}{dt} \qquad \text{(Eq. 4.4)}$$

Substituting the specific heat ratio, γ, provides the final equation used in the present analysis with the result being equally valid when substituting the independent variable θ, or crank angle, to represent time, t:

$$\frac{dQ_n}{dq} = \frac{g}{g-1} p \frac{dV}{dq} + \frac{1}{g-1} V \frac{dp}{dq} \qquad \text{(Eq. 4.5)}$$

Another method for determining the combustion duration and the ignition delay through analysis of the incremental heat addition was introduced by Rassweiler and Withrow [1938] and examined by Brunt and Emtage[1997]. Brunt and Emtage compared several algorithms for producing these parameters and concluded that the Rassweiler and Withrow model, which used estimations for the expected end of combustion and polytropic indices to more accurately determine the mass fraction burned, was the most effective for producing accurate results when used on a spark ignition engine. However, due to processing power constraints associated with calculating the heat-release parameters on a cycle-by-cycle basis and the limited increase in accuracy, the basic algorithm described above was used to generate the data for this study. Overall accuracy, although important, is not as crucial for the training of the neural network as consistency, given that the network trains on repeatable results and employs what has been learned for future predictions. In other words, absolute and complete description of the combustion dynamics involved is not necessary for a successful predictive neural network.

Additionally, several other parameters were calculated from the pressure histories. The location of 50% of the mass fraction burned of fuel, or LMFB50, was based on the same algorithm as the combustion duration.

The maximum burning rate was calculated through a five-point central difference differentiation algorithm. For each point, the estimated compression pressure was first subtracted from the total pressure to provide the pressure rise due only to combustion. The compression pressure was estimated through the polytropic relation based on the conditions during the closing of the intake valve. Similarly, the maximum instantaneous torque generated and its location are determined for each combustion cycle. The instantaneous torque value is calculated from the geometric parameters of the engine that provide a moment arm which is then multiplied by the total downward force impinging on the piston determined from the pressure recorded at that point.

Finally, the incremental heat-addition calculations provide three additional parameters: the mixing burn, defined as the time in crank angle degrees for the mass fraction burned to increase from 50% to 99%; the maximum heat added, defined as the highest calculated incremental heat value that occurs in the cycle; and the location of the maximum heat added. These are all byproducts of the combustion duration calculations already being performed. In a typical diesel combustion process, there are two distinct burning phases that can be seen in the heat release information: the initial heat release spike and the longer oxidation time. The mixing burn length attempts to approximate the oxidation burning sequence. Figures 12 and 13 and Table 8 detail the in-cylinder pressure parameters that were taken during the study.

Geometric features of the pressure signal can also be used to define combustion characteristics. Leonhardt et al. [1995] used the center of gravity and secant lengths to characterize pressure signals for the purpose of neural network supervision of the fuel-injection process. Gassenfelt and Powell [1989] also used geometric descriptors such as moments and centroids to generate algorithms for determining A/F ratio from pressure signals. Consequently, effective descriptors for combustion characteristics and their effects upon engine performance and behavior are not limited to hard engineering variables, but extend to geometric parameters as well. However, it was decided that the combustion based variables would offer adequate information for neural network training.

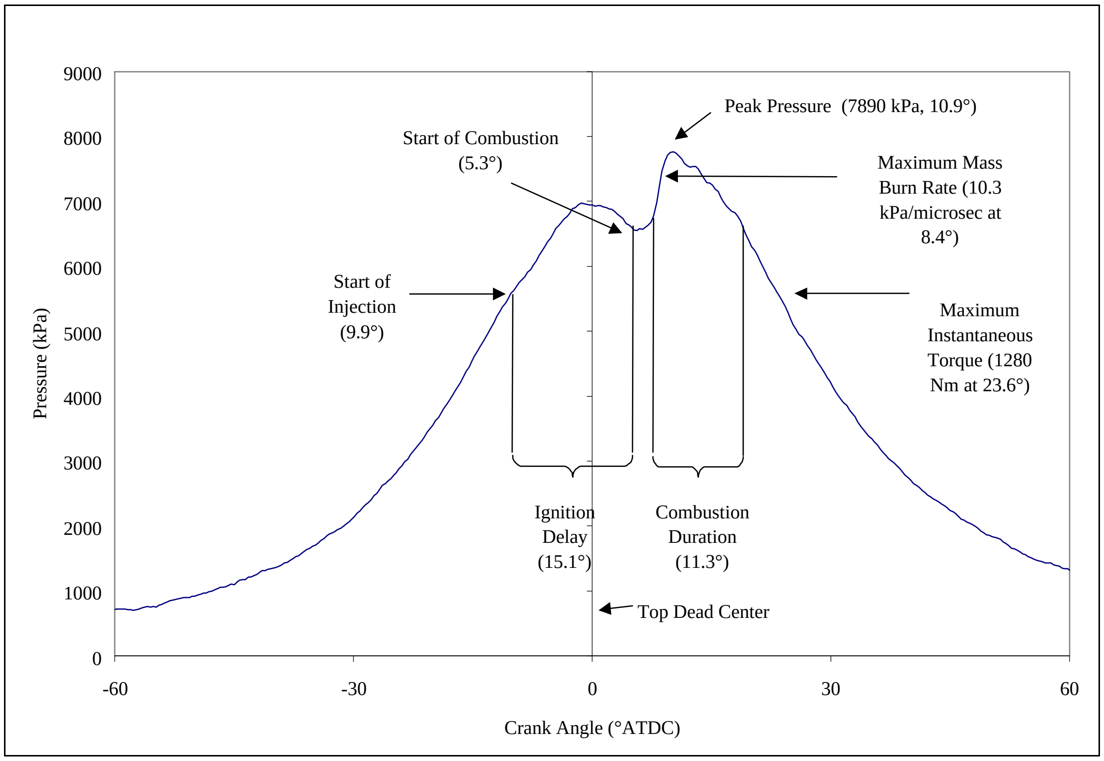


Figure 12: Typical in-cylinder pressure parameters taken from Navistar T444 DI diesel engine; 2200 RPM, 203 N-m, IMEPg = 577 kPa, diesel #2.

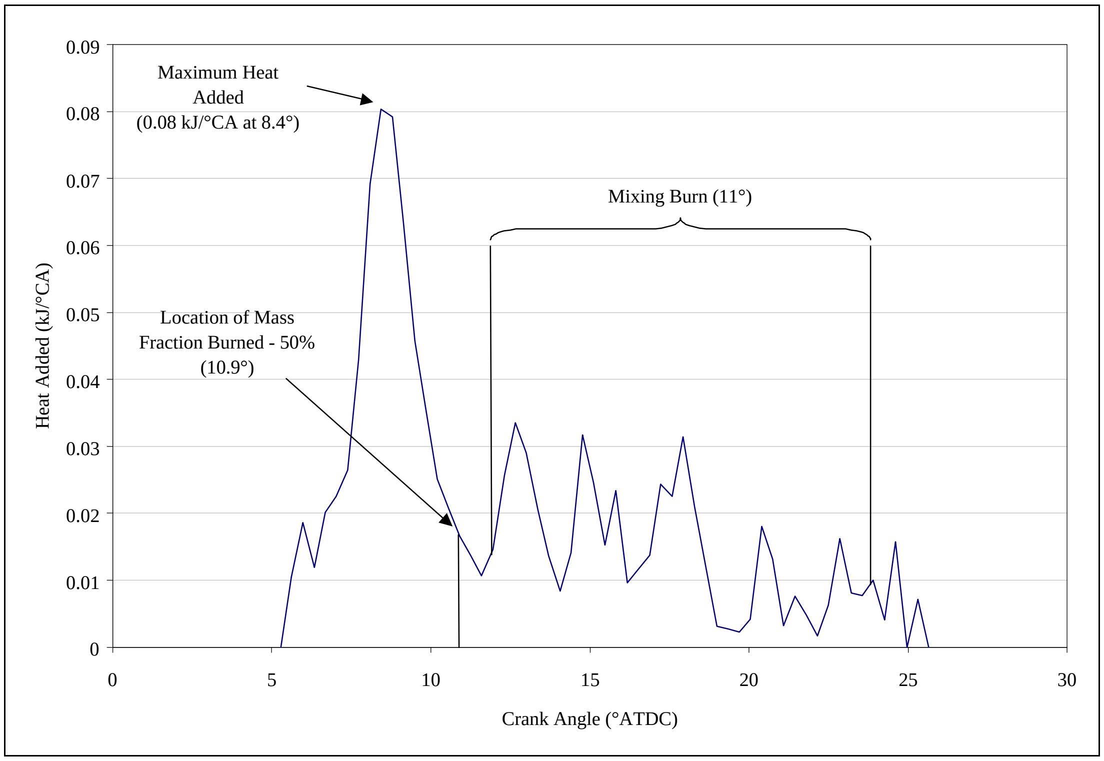


Figure 13: Typical heat release profile and corresponding burnt mass fraction information for Navistar T444 DI diesel engine; 2200 RPM, 203 N-m, IMEPg = 577 kPa, diesel #2.

5. Experimental Setup

A 1994-specification Navistar direct-injection diesel engine (Tables 5 and 6) fueled by D2-specification diesel provided the data for this study. An initial power map provided the information needed to develop a representative test matrix. The engine map was divided into 64 test points with increments of approximately 300 RPM and 50 lb-ft. (Figure 12). For each data set point, the engine was brought to the appropriate speed and load and allowed to reach steady-state in order to prevent transience in operating conditions and to provide steady emissions production. Once all parameters of interest had achieved a steady-state condition, data acquisition was initiated. Every revolution of the engine generated a trigger signal for the acquisition of the designated “slow-speed” variables such as manifold air pressure (MAP), intake air temperature (IAT), engine coolant temperature (ECT), start of injection (SOI), engine speed, emissions (Table 7), etc. After a set number of revolutions another trigger initiated the “high-speed” acquisition using a DAS-58 high speed sample-and-hold ADC board. With the pressure signals from two cylinders and the top dead center (TDC) phasing signal being recorded, the DAS-58 was capable of acquiring complete pressure histories for 128 combustion events. Consequently, 130 megabytes of data were eventually acquired on April 16, 1998 for network training purposes.

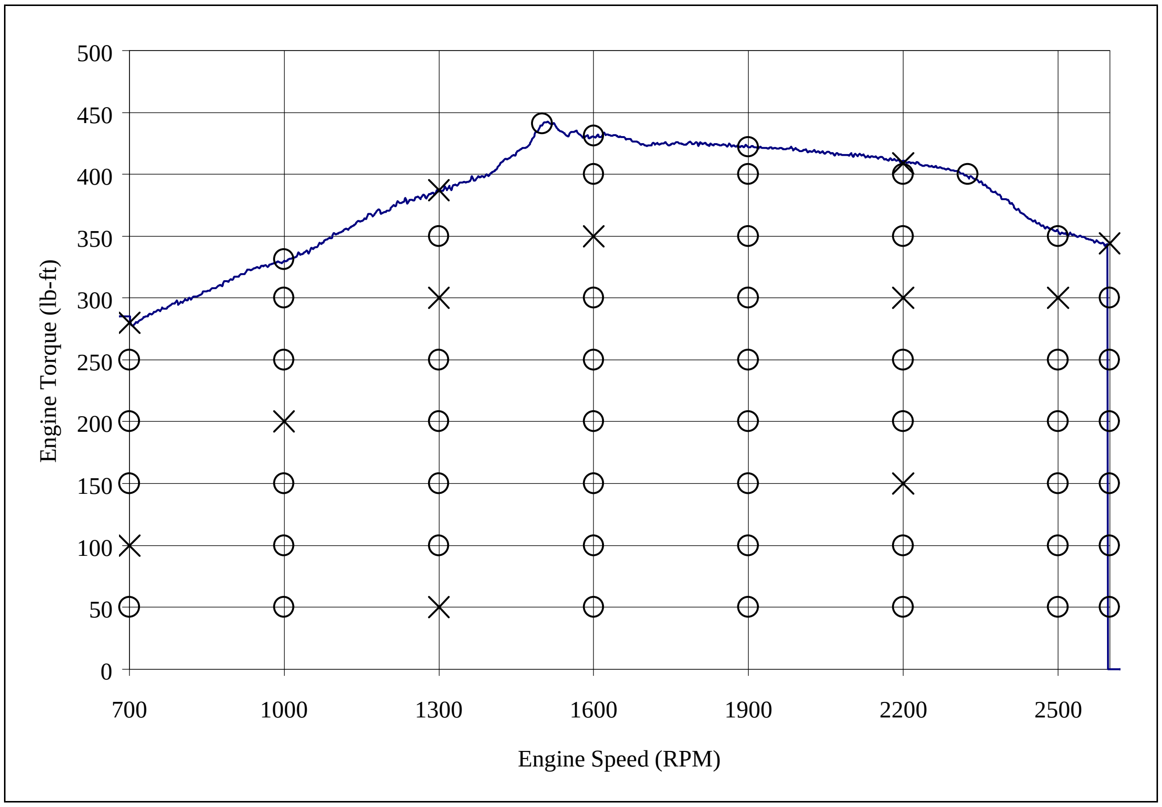


Figure 14: Speed and load map for Navistar T444 operating on diesel #2 (O indicates training point, X indicates test point).

Table 5 : Navistar Engine Instrumentation

Instrument	Model
Exhaust Thermocouples	Omega K-Type, ungrounded #TJ36-CASS-18(U)-12
Manifold Air Pressure Sensor	Motorola Part # 1807249C1
Manifold Air Temperature Sensor	Motorcraft Part # F2VY-12A648-A
Oil Temperature Sensor	Motorcraft Part # F2VY-12A648-A
Coolant Temperature Sensor	Motorcraft Part # F2VY-12A648-A
Injection Control Pressure	Motorcraft Part # 1607J29
Opticoder (Event Timing)	Sumtak Opticoder LEI-292-1024

Table 6 : Navistar T444E Engine Specifications

Type:	Water Cooled Turbocharged Direct Injection Diesel
Cylinder Arrangement:	8 Cylinder V-type
Bore and Stroke:	104.39 mm x 106.20 mm (4.11 in x 4.18 in)
Compression Ratio	17.5:1
Displacement	7.3 liter (444 cu. in.)
Horsepower	175HP @ 2600 RPM
Fuel	BP Diesel #2 Specification

Table 7 : Emissions Analyzers

Instrument	Model
HC Analyzer	Rosemount Analytical 402 FID
NO_X Analyzer	Rosemount Analytical 955 Chemiluminescent
CO Analyzer	Rosemount Analytical 880A NDIR
CO_2 Analyzer	Beckman Industrial 868 NDIR

6. Neural Network Selection and Training

6.1 Training Data Pre-Processing

The data were processed using a custom reduction program that incorporated both slow- and high-speed data streams into one comprehensive summary file. Each combustion event was analyzed and after TDC was confirmed using the phasing signal, various combustion related parameters were derived from the pressure curves. Each test point provided 128 combustion events and 270 revolutions of slow-speed data. All parameters were averaged for each test point and the result used to represent that point on the test matrix.

Once the comprehensive data file had been generated and converted to spreadsheet format, the NeuroShell software [Ward Systems Group, 1996] was able to read the information. Once imported, the minimum and maximum values for all data columns were calculated. These were used to scale the inputs and outputs from –1 to +1 to ensure that there were no unequal and unwanted contributions from any one data column. For instance, without scaling, the relative importance of engine speed could be overinflated during network training if the other inputs' values were an order of magnitude lower (e.g. engine speed of 1500 RPM vs. a manifold air pressure of 100 kPa vs. a combustion duration of 10°).

Ideally, the data gathered would represent the entire range of data that the engine is expected to encounter. After scaling, the opportunity was given to extract a representative set of data from the training set to be used as a testing or calibrating reference set. The software provided a means of generating the test set through a random extraction method, but it was decided that a better method would be to manually choose 12 test points covering a wide range of conditions to ensure that at least one full power set point

Table 8 : Combustion-Derived Parameters

Shorthand Abbreviation	Explanation	Definition
Peak	Peak Pressure Value	Maximum pressure encountered in an individual cycle (kPa).
LPP	Location of Peak Pressure	Location of the maximum pressure encountered (CA°).
IMEPg	Indicated Mean Effective Pressure (gross)	Integrated work for the compression and expansion cycles, divided by the displacement of a single cylinder (kPa)
IgnDel	Ignition Delay	Time in crank angle degrees from fuel injected to the discernable start of the mass fraction burned (CA°).
CombDur	Combustion Duration	Time in crank angle degrees from 10% to 90% of the mass fraction burned (CA°).
LMFB50	Location of Mass Fraction Burned - 50%	Location of 50% of the integrated mass fraction burned curve (CA°).
MaxBurn	Maximum Burn Rate	Maximum rate of pressure rise calculated from a 5-pt. central difference (kPa/microsecond).
MaxBurnLoc	Maximum Burn Rate Location	Location of the maximum burn rate (CA°).
MaxTorque	Maximum Torque	Maximum instantaneous torque based on geometric calculation of moment arm and pressure (Nm).
MaxTorqueAngle	Maximum Torque Angle	Location of the maximum torque (CA°).
MixingBurn[*]	Mixing Burn	Measure in crank angle degrees of the time from 50% to 99% of the mass fraction burned (CA°).
MaxQ	Maximum Heat Release (Q)	Maximum value of the instantaneous heat release used to calculate the mass fraction burned (kJ/CA°).
MaxQLoc	Maximum Heat Release (Q) Location	Location of the maximum heat release (CA°).

[*] During verification of the data generated by the DSP-based acquisition system, an error was discovered in the off-line reduction software that effectively added the start-of-burn value to the Mixing Burn parameter. Unfortunately, the neural networks had already been trained using these values, but since the networks are capable of successful prediction so long as the data are consistent, it was decided to continue regardless. Consequently, the values from the DSP-based system were adjusted by the start-of-burn for consistency.

was represented (as shown in Figure 12). The purpose of the test data was to provide the neural network with a data set that the network had not previously "seen" and which was therefore valuable as a gauge to test for accuracy. After extraction, the number of training data points was fixed at 52, covering a wide range of engine conditions.

Before training commenced, the data were analyzed to determine the best candidates for each of the recorded emissions. Linear data correlation techniques provided an indication of the predictive capabilities of the candidate variables with scatter plot charts providing further insight (Appendix A). The data correlation was done through Microsoft's Excel spreadsheet program and is used to measure the relationship between two sets of data points. Specifically, the "population correlation calculation returns the covariance of two data sets divided by the product of their standard deviations."[Microsoft, 1996] These correlations offered a first step towards choosing the appropriate input variables. Obviously, relationships that were not close to linear would not have high correlations and so further investigation was necessary to ensure equally important variables were not left out. To this end, the scatter plot charts were particularly informative for ignition delay and MaxBurn. As can be seen in the charts in Appendix A, the ignition delay shows a definite effect on NO_X production, but that relationship appears to be split into separate engine-speed bands, leaving a statistical correlation number of 0.126 across the whole speed and load range. This corresponds well with the evidence gathered in the past concerning the speed scaling of the ignition delay and the long history of timing advance devices that take advantage of it. Further, MaxBurn, and to an extent LPP, show a slight bifurcation effect that the correlation numbers do not explain. These somewhat non-linear relationships are well suited for neural network training as they can still be determined with adequate training while combination with other variables may describe a more straightforward relation with the gaseous emissions. More intricate multi-dimensional relationships between these variables and the gaseous emissions were not possible since at the time the data were acquired, control of independent variables such as fueling and start of ignition was not available. Nonetheless, the relationships that could be determined were adequate for choosing a set of potential input variables for successful predictive networks.

Correlation values between the emissions and all data variables are supplied in Table 9 (values above 0.5 and below –0.5 have been boldfaced) where each value is the average for each engine set point. Negative values represent an inverse relationship. From these sources and biasing the selection in favor of theoretically strong indicators, seven variables were selected for each emission gas to serve as inputs. The number of variables, seven, was chosen due to the rather large amount of network processing required for all combinations of inputs. The rationale will be explained below.

Table 9 : Linear Correlation Values for the Data Matrix (Cylinder 3 Only)

		HC	CO	CO2	NOx
HC	HydroCarbons	1			
CO	Carbon Monoxide	-0.21306	1		
CO2	Carbon Dioxide	-0.31780	-0.01426	1	
NOx	Oxides of Nitrogen	-0.40886	0.01986	**0.96890**	1
Speed	Engine Speed	0.43751	-0.21311	**0.61527**	**0.50003**
FIPW	Fuel Injector Pulse Width	**-0.77224**	0.31948	0.47451	**0.57593**
SOI	Start of Injection	**0.70669**	0.06152	0.16552	0.08341
ExTemp	Exhaust Temperature	**-0.51188**	0.22817	**0.89532**	**0.93233**
IAT	Intake Air Temperature	0.12988	-0.35696	**0.74043**	**0.62214**
MAP	Manifold Air Pressure	-0.17083	-0.10780	**0.97094**	**0.89665**
ECT	Engine Coolant Temperature	-0.11371	0.19725	**0.69235**	**0.69743**
APS	Accelerator Pedal Position	0.151934	0.25141	**0.69803**	**0.64697**
Torque	Engine Torque	**-0.68372**	0.23919	**0.79250**	**0.86961**
EOT	Engine Oil Temperature	0.22454	-0.11398	**0.78579**	**0.72506**
ICP	Injection Control Pressure	-0.39150	0.20646	**0.94363**	**0.94182**
Smoke	Bosch Smoke Meter	-0.00721	0.17513	**0.61855**	**0.58751**
Peak	Peak Pressure	**-0.50044**	0.45322	**0.84224**	**0.87798**
COV(Peak)	Coefficient of Variation of Peak Pressure	**0.77690**	-0.13656	**-0.55399**	**-0.59470**
LPP	Location of Peak Pressure	-0.28463	-0.13806	0.33880	0.47287
COV(LPP)	COV of LPP	-0.05834	-0.05060	0.39979	0.27505
IMEPg	Indicated Mean Effective Pressure (gross)	-0.47408	0.12468	**0.95383**	**0.97816**
COV(IMEPg)	COV of IMEPg	0.10285	-0.05545	-0.35835	-0.37824
IgnDel	Ignition Delay	**0.73036**	-0.25122	0.23863	0.12634
CombDur	Combustion Duration	-0.36937	-0.03626	**0.95619**	**0.95842**
LMFB50	Location of Mass Fraction Burned 50%	-0.20946	-0.37399	**0.89292**	**0.83780**
MaxBurn	Maximum Burn Rate	0.02063	0.37815	-0.20617	-0.12689
MaxBurnLoc	Location of Maximum Burn Rate	-0.14946	**0.69667**	-0.30248	-0.21578
MaxTorque	Maximum Instantaneous Torque	-0.44461	0.12026	**0.96696**	**0.98130**
COV(Torq)	COV of Maximum Inst. Torque	**0.81725**	-0.12123	-0.38685	-0.44670
MaxTAngle	Location of Maximum Inst. Torque	-0.25686	-0.19218	**0.95591**	**0.89339**
MixingBurn	Mixing Burn Length	**-0.56789**	0.37016	**0.65451**	**0.75617**
MaxQ	Maximum Instantaneous Heat Addition	**-0.53731**	0.41577	-0.35195	-0.27452
MaxQLoc	Location of Max. Inst. Heat Addition	-0.15513	-0.39886	**0.80901**	**0.70906**

Finally, theoretical considerations were given to the selection of the inputs that were to be used for training. Some inputs were chosen for their representation as direct measures of the formation rates of the gaseous emissions while others, although possessing a linear relationship with the emissions, were only loosely related to the formation rates. For instance, the peak pressure is a more or less direct measure of the in-cylinder temperatures and thus the NO_X formation rate, while the IMEPg, because it reflects the amount of work being done, indicates the amount of fuel being combusted and gives a more general indication of the NO_X formed.

Hydrocarbon formation in the diesel engine is heavily influenced by the amount of oxygen available and the rate of combustion of the fuel injected. For the latter reason, MaxQ, combustion duration, and the peak pressure were chosen as inputs to the training networks. A misfire condition or a delayed start of combustion could be measured by a low value for the maximum heat release rate as compared to more normally high values associated with good combustion. Similarly, a very long or abnormally short combustion duration could indicate poor performance, especially in combination with the peak pressure. In either case, complete combustion is questionable and the hydrocarbon emissions should show the effect accordingly. Along those lines, the phasing of the pressure and volume of the cylinder can have an effect on combustion and so the location of peak pressure, the location of 50% of the mass fraction burned, and the ignition delay may give an indication of combustion quality. Finally, the direct measures of possible hydrocarbon formation were considered. Since post oxidation of the fuel heavily influences the amount of hydrocarbons escaping down the exhaust pipe, the mixing burn variable was chosen to help account for it.

The processes behind carbon monoxide formation are similar to the processes that produce hydrocarbons. As a result, the choices for inputs are similar. Once again, the post-oxidation phase of combustion late in the cycle has a notable effect on the final amount of CO found in the exhaust gas. For that reason the mixing burn parameter was chosen in the hopes of producing a highly relevant network input.

Carbon dioxide is one of the two major molecular products of hydrocarbon combustion (water is the other). The amount of fuel and how well that fuel is converted to energy determine the overall quantity of CO_2 that leaves the cylinder in the exhaust gas. Consequently, the geometric qualities of the pressure trace offer considerable information for potential network inputs. Foremost among them is the IMEPg, which provides a numerical value for the work produced per unit volume. The work produced is very much dependent upon the pressure in the cylinder and the time in the cycle that that pressure occurs. For this reason, the peak pressure contributes valuable information as well. Other variables describing pressure-volume and combustion phasing such as the combustion duration, location of mass fraction burned-50%, the ignition delay, and the location of the maximum burn rate can also be quite useful as inputs. Finally, the maximum burn rate, which describes the derivative of the combustion pressure, infers the qualities of the fuel and in conjunction with other variables could offer information on the combustion efficiency, which has a high correlation to carbon dioxide production.

NO_X formation is accelerated by three major factors: the availability of nitrogen and oxygen and the temperature of the combusting mixture. Since there are no direct measurements of the in-cylinder gas temperature available, other means of establishing those influences had to be substituted. For a perfect gas, pressure and temperature rise together (for the current application) and so the choice of peak pressure as an input is obvious. The higher the pressure reached inside the cylinder, the higher the spatially and temporally averaged temperature of combustion. Along those same lines, the maximum heat release rate, MaxQ, and the maximum rate of change of the combustion pressure, MaxBurn, were chosen as useful input variables. Theoretically, a large heat addition will raise the temperature inside the cylinder accordingly and the maximum combustion pressure change will reflect that also. Consequently, three of the inputs are qualitative records of high-temperature behavior. The phasing of the pressure and volume can also have an influence on the eventual in-cylinder temperature and so the ignition delay and LMFB50 were chosen to account for that potential. If the location of mass fraction burned-50% occurs at a very late stage in the cycle, the volume of the cylinder will be steadily growing and temperature effects due to combustion will be reduced. Similarly, the ignition delay will indicate how relatively late or early in the cycle combustion is occurring and give a measure of the volatility of the fuel. The final two variables chosen for the NO_X prediction networks were IMEPg and the combustion duration. The IMEPg is a simple measure of the amount of work being done and will thus have a strong overall correlation to NO_X

while combustion duration can indicate the total amount of fuel combusted. Table 10 shows the variables chosen for each respective emission level.

Table 10 : Variable Combinations for Network Inputs

Emission Gas	Input Variables
HC	Peak, LPP, CombDur, LMFB50, IgnDel, MixBurn, MaxQ
CO	Peak, LPP, LMFB50, MaxBurn, MaxBurnLoc, MixBurn, MaxQ
CO_2	Peak, CombDur, IMEPg, LMFB50, IgnDel, MaxBurn, MaxBurnLoc
NO_X	Peak, CombDur, IMEPg, LMFB50, IgnDel, MaxBurn, MaxQ

Finally, to ensure that the networks train properly, uniqueness within the data set should be confirmed. If two speed and load conditions could produce the same pressure-derived variable set, then the network would have trouble distinguishing between the two, leading to a reduced ability to predict successfully unseen data. To this end, the data from the 64 point steady state test matrix were plotted in three dimensions to aid in the determination of uniqueness (Appendix B) . Even a quick glance will confirm that some variables experience no repetition within the data set (e.g., peak pressure, IMEPg, ignition delay). Others, however, do show small constant sections and areas where the same variable displays the same value, but when combined with other inputs, the effect of this problem is reduced considerably. Neural networks use the combined effects of many variables to determine the relationships and a simple multiplicative function would eliminate the problem of a repetitive value in a single variable. Consequently, the data matrix used to train the networks has unique variables for each point in the grid.

6.2 Applied Neural Network Architectures

NeuroShell 2 [Ward Systems Group, Inc., 1996] provided a variety of network architectures suited to the problem at hand. Additionally, a batch processor supplied with the software enabled the user to train a number of different networks sequentially. This proved useful as the number of hidden layer neurons cannot be predicted for the best network and must be determined heuristically since there is no accepted theory in neural network literature as to a satisfactory method of selecting the number. Three architectures were chosen based on their use of 3-layer error back-propagation. As discussed previously, the EBP network is quite good at interpolation and this feature fits the data matrix assembled. The first architecture was a straight 3 layer EBP with a hyperbolic tangent activation function in the hidden layer and a linear activation function for the output layer. The number of hidden layer neurons ranged from 5 to 20 within the NeuroShell 2 batch processor. The second architecture employed a second hidden layer in parallel to the first. This second layer uses a different activation function in an attempt to find patterns in the data not uncovered by the first. The activation functions for these hidden layers in the second architecture were the Gaussian and Gaussian complement while the output activation function remained linear. Finally, the last architecture added a third hidden layer. This setup was basically the same as the second with the third hidden layer using a hyperbolic tangent activation function. Both architectures used a range of hidden neurons from 4 to 19[*]. Architectures 2 and 3 are referred to by the names Ward #1 and Ward #2 (Ward Systems Group, Inc. is the publisher of NeuroShell 2) and architecture 1 is known as the 3-Layer (Figure 15).

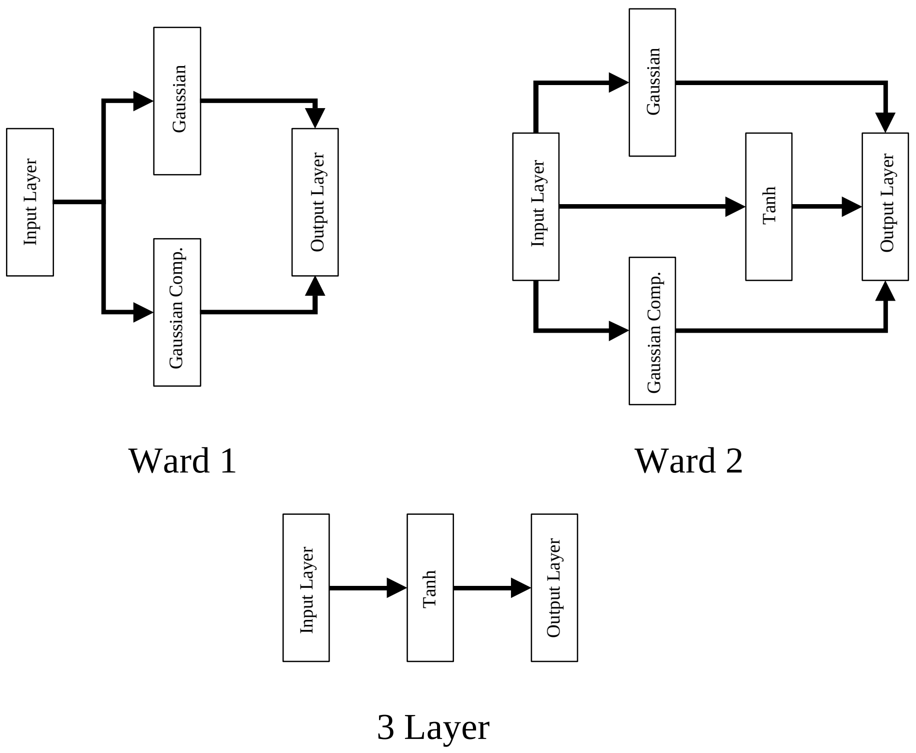


Figure 15: The three different NeuroShell network architectures.

[*]The Ward #1 and Ward #2 networks employed secondary and tertiary hidden layers in the architecture, effectively adjusting the number of hidden nodes by 2x and 3x, respectively. All future notations of the number of neurons in the hidden layer refer to each layer, and not the aggregate.

The batch processor was then set up to train networks continuously. Each of the gaseous emissions predictions (NO_X, HC, CO, CO_2) was trained with the three networks over a range of the number of hidden layer neurons for each combination of inputs possible from the initial seven. Combinations of 3, 4, 5, 6 and 7 inputs resulted in a total of 99 different architectures. Considering that there are 4 emissions, 3 network architectures and 16 hidden neuron levels, the amount of training time required to train all 19,000 networks has the potential to be quite large. This was the primary factor in limiting the selection of potential input variables to seven. An 8 variable pool would have resulted in 219 combinations of inputs (56 for 3 and 5 inputs, 70 for 4, 28 for 6, 8 for 7 and 1 for 8 and 42,000 total networks). With four Pentium class computers operating whenever possible, the seven-input combination took two months to complete. A similar process with an eight-input pool would have taken twice as long.

6.3 Training Results

After each batch processing session, the results for each hidden node sweep for each architecture were tabulated. NeuroShell supplies the option of applying a multiple regression analysis to a data set in order to return the statistical indicator R^2, defined as:

$$R^2 = 1 - \frac{SSE}{SS_{YY}} \qquad \text{(Eq. 6.1)}$$

where

$$SSE = \text{Sum of Squares of Errors} = \sum (y - \hat{y})^2 \text{ and } SS_{YY} = \sum (y - \bar{y})^2$$

and where y is the actual value, $\hat{y}$ is the predicted value of y and $\bar{y}$ is the mean of all the y values [Ward Systems Group, 1996, p. 194]. An R^2 value was determined for both the training set and the test set data. A good network should give high results (approaching 1.0) for both sets. These results were analyzed for the best R^2 value averaged between the test set and training set; the corresponding inputs, number of hidden-layer nodes, and architecture were recorded. Typically, the R^2 values reached a relative maximum for a given number of nodes in the hidden layer. The reason for the hidden node sweep is that there is no general theory governing the correct number of nodes necessary for good neural network prediction. Figure 16 demonstrates this by showing the local maxima within the hidden node sweep. The end result was a table for each gaseous emission showing the best networks with the highest potential for correctly predicting transient emissions data (Appendix C).

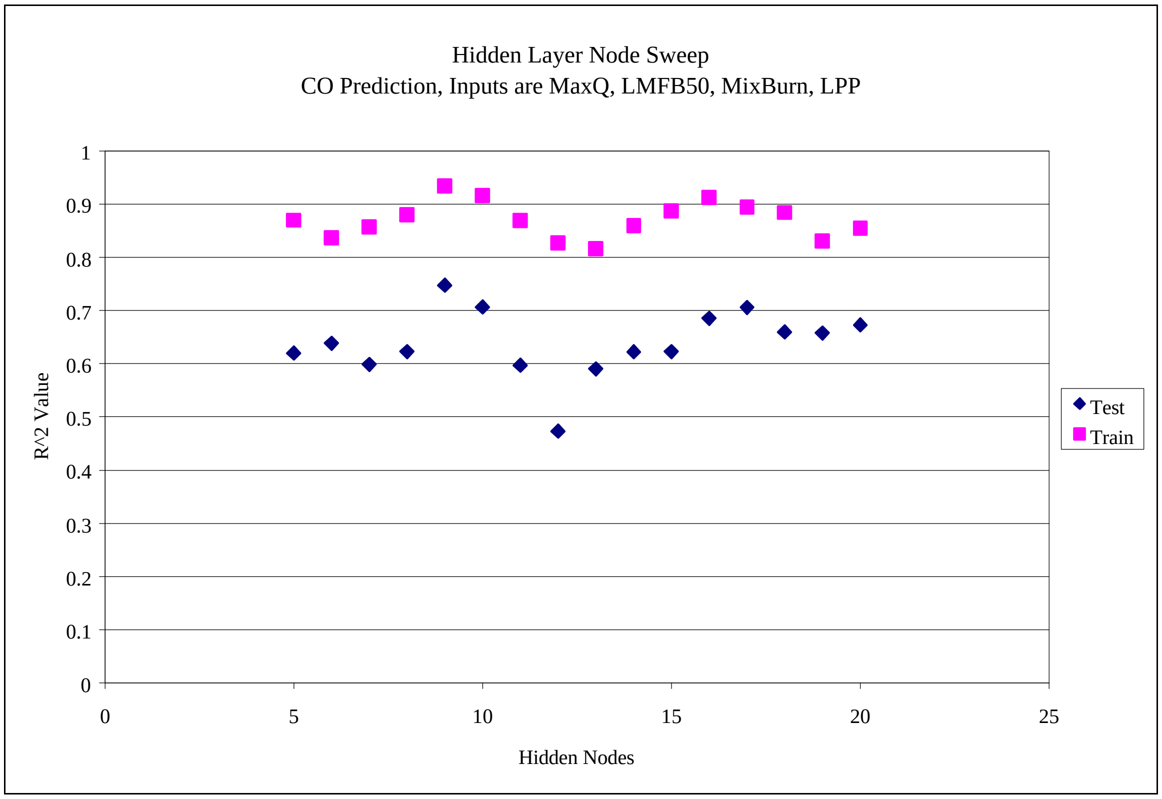


Figure 16: Typical hidden layer node sweep results (4 inputs and 1 output).

The tables of data generated by the extensive network training offer a small amount of useful information about the relative importance of certain inputs over others. Because of the heuristic nature of the training and because of the difficulty of extracting the influence of one variable from the combined influence of several, the information is not decisive. Examining the lists of the best 20 networks (Appendix B) for each gaseous emission does allow for some interpretation. If one eliminates from the lists those networks containing the same inputs but different architectures and then tabulates the relative occurrences of the individual input variables, one can get an idea of the influence of those variables. Examination of the weighting factors was considered, but the values are not particularly informative and the Ward Systems Group warns against using them for anything but the most general application, especially as the number of inputs rises. Table 11 shows the percentage of the total number of input combinations of which each input variable is a member. For instance, ignition delay occurs in 70.6% of the unique networks in the top 20 networks for hydrocarbon prediction. The conclusion that can be made from this table is that certain inputs seem to be far more important than others for improved predictive abilities. NO_X appears to be heavily affected by IMEPg, LMFB50, and ignition delay as these appear in all of the top 20 networks. Indeed, one of the top 20 predictive neural networks for NO_X has the three-input architecture containing just those variables.

Table 11 : Percentage of Variables Occurring in the Top 20 Networks Trained to Predict Each Gaseous Emission (Total is the number of unique input combinations in the top 20)

	HC						
Total	IgnDel	CombDur	MaxQ	LMFB50	LPP	MixBurn	Peak
17	70.6	52.9	76.5	70.6	58.8	35.3	88.2
	CO						
Total	Peak	MaxQ	LMFB50	MixBurn	MaxBurn	LPP	MaxBLoc
15	100	100	80	40	53.3	46.6	60
	CO2						
Total	Peak	CombDur	LMFB50	MaxBLoc	IgnDel	IMEPg	MaxBurn
11	100	54.5	63.6	72.7	100	45.5	72.7
	NOx						
Total	CombDur	IMEPg	LMFB50	IgnDel	MaxQ	Peak	MaxBurn
9	55.5	100	100	100	55.5	11.1	44.4

Once the networks had been trained, they were converted to C++ source code files and prepared for integration into a data analysis program. The final step involved the acquisition of high speed in-cylinder pressure data and emissions analyzer signals over the entire FTP cycle.

As a final test of the network variables chosen, periodically sampled pressure points from the digital pressure traces were used as inputs, in order to demonstrate whether engineering-based variables derived from pressure profiles were more or less efficient for use in a neural network. Sixty-four, thirty-two, sixteen, eight, and four pressure points taken from set intervals from the average pressure trace from each speed and load were used as inputs (Figure 17), with results given in table 12. HC and CO clearly do not benefit from a sampled pressure-based network scheme. The average R^2 values are below those obtainable from the engineering variable-based networks. CO_2 and NO_X, however, seem to respond well to the sampling method. This is probably due to the heavy dependence of their formation rates on geometrically influenced parameters. They both have high correlation to both IMEPg and peak pressure, which are geometrically based parameters. Also, they both seem to be affected by such variables as the ignition delay and the MaxBurn, which can be discerned from a raw pressure signal. When the number of

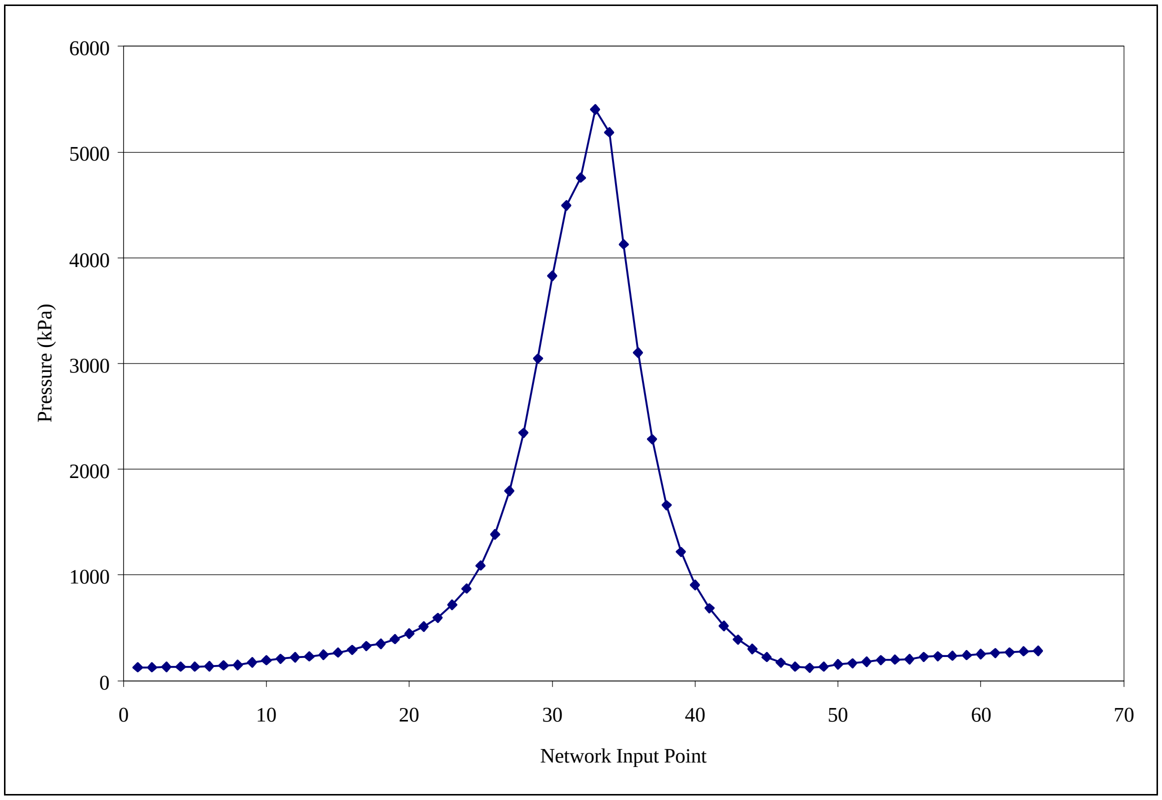


Figure 17: Representation of a typical in-cylinder pressure cycle reduced to a network input matrix (64 inputs).

inputs is a factor, though, the engineering-based variables show an advantage. The pressure-sampled data do not produce very high R^2 values until 16 or more inputs are used. This is in direct contrast to the engineering variables, which have high R^2 values for as few as three inputs. The conclusion that can be drawn from this is that real-world engineering-based variables are more efficient for prediction purposes than periodically sampled pressure curves, despite the fact that enough samples can in fact produce good results for NO_X and CO_2. The reason being that the engineering variables are derived from the larger 1024 data point set with a considerable amount of post-processing. The lower numbers of points used as direct inputs are simply lower resolution images of the full 1024 point set.

Table 12 : Network Results from Sets of Pressure Points from Sampled Average Pressure Traces for Each Speed and Load Set-Point with Comparison to Four-Engineering-Variable Networks

		Sampled Pressure Trace			In-Cylinder Pressure Variables			
Number of Inputs	Hidden Neurons	R^2 Test Set	R^2 Train Set	R^2 Average	Hidden Neurons	R^2 Test Set	R^2 Train Set	R^2 Average
HC								
64	40	0.785850	0.933887	0.859868				
32	24	0.735079	0.965853	0.850466				
16	16	0.779130	0.868464	0.823797				
8	12	0.575277	0.831664	0.703471				
4	10	0.680254	0.888052	0.784153	15	0.916069	0.986431	0.951250
CO								
64	40	0.608614	0.895498	0.752056				
32	24	0.723613	0.924028	0.823820				
16	16	0.578273	0.859865	0.719069				
8	12	0.618232	0.864459	0.741346				
4	10	0.629076	0.863229	0.746153	6	0.997674	0.998376	0.998025
CO_2								
64	40	0.996524	0.999005	0.997764				
32	24	0.997050	0.998041	0.997545				
16	16	0.997938	0.998911	0.998424				
8	12	0.997082	0.998434	0.997758				
4	10	0.997541	0.996863	0.997202	9	0.997932	0.998829	0.998380
NO_X								
64	40	0.989771	0.997465	0.993618				
32	24	0.993306	0.997218	0.995262				
16	16	0.979816	0.973019	0.976417				
8	12	0.980584	0.974955	0.977770				
4	10	0.972753	0.960654	0.966703	20	0.998166	0.997204	0.997685

7. Digital Signal Processing

7.1 General Overview

Recent advances in microprocessor design and manufacture have enabled greater and greater power to be harnessed for high-speed data analysis. The Digital Signal Processor (DSP) is a microprocessor specifically designed for high efficiency performance during repetitive looping calculations encountered in signal processing. Many algorithms used for traditional pressure variable calculations utilize software loops to a high degree, such as those used for determining IMEPg and combustion duration. Because of this, the DSP is highly suited for application in combustion pressure analysis. Additionally, advances in DSP design now enable extremely high-speed processing capabilities, well in excess of those required for real time analysis.

Combustion analysis using a DSP has been used in previous research endeavors largely for control purposes. Lee et al. [1995] constructed a DSP analysis system, wrote accompanying software for ease of use, and drew on the setup to provide information on engine variables in real time for the purpose of modifying engine control. Leisenring, et al. [1995] used a DSP to calculate the heat-release rate inside the cylinder and correlated that information to the A/F ratio in a SI engine. The information was then applied to better control the engine during cold-start operation before the EGO sensor was warm enough to provide feedback control.

7.2 DSP Application Development at WVU

A Texas Instruments TMS320C30 Evaluation Module was installed in a 133-MHz Pentium-based IBM-compatible personal computer for the purpose of analyzing pressure profiles on a cycle-by-cycle basis in real time. The Evaluation Module (EVM) consists of an 8-bit PC-compatible half-length card containing a 33-MFLOP (million floating point operations per second) DSP with 64k words of on-board memory. In addition, the half-length card features a bidirectional host interface capable of approximately 200 kbytes-per-second throughput.

For preliminary development purposes, a small electronic engine simulator was constructed by Richard Atkinson to deliver previously recorded and stored pressure profiles to the DSP Evaluation Module. The pressure-like profiles were first scaled and converted to hexadecimal before being programmed onto an EPROM chip. A 10-channel switch was then wired to the chip in order to send out 2048 distinguishable digital samples depending upon operator choice. These signals were converted to analog to effectively represent an equivalent electronic signal from an in-cylinder pressure transducer. The conversion was performed by a Harris 8-bit digital-to-analog converter (DAC) with a 20 ns settling time for a very effective high-speed rate.

The analog signals from the simulator were then sampled and converted by a Maxim MAX121 analog-to-digital converter (ADC) at 14-bit resolution and with a maximum throughput rate of 308 kilo-samples per second. The MAX121 was specially designed to work with various DSP chips, including the TMS320. Software included in the literature received with the MAX121 was modified for the specific task of sampling engine data (Appendix A). The converted 14-bit samples were then grabbed by the TMS320 through a 16-bit serial port after two trailing zeros had been added. The resulting sample then had to be right-shifted two bits to return to the original value converted by the MAX121.

The simulator was designed to operate between speeds equivalent to idle and full throttle on the SI engine testbed, a 1992 Saturn DOHC 1.9L spark-ignited gasoline engine. Consequently, equivalent signal updates simulated 800 to 4000 revolutions per minute with an additional signal simulating an engine pulse occurring at top dead center of cylinder #1. The end result of this is a square wave with an effective wavelength of two engine revolutions. To simulate the opticoder found on the engine driveshaft, a third signal had to be generated. The opticoder has 1024 teeth and serves as the sampling clock for the high speed ADC boards connected to it. A Harris CD4040BE 12-stage binary counter chip performed this function and clocked the sampled pressure data from the MAX121 to the TMS320.

The simulated pressure traces were successfully supplied to the EVM and the DSP was tested for computational ability and throughput for simulated speeds of 800 to 4000 rpm. The EVM proved capable of analyzing 6 combustion parameters per cycle at 4000 rpm and supplying them to the host computer running a port communication program. Although the simulation proved the basic concept, it needed significant modification for its real-world application.

7.3 DSP Installation

Once proof of concept had been obtained with the simulator, the EVM and PC host were moved to the engine control room for real-world application. The MAX121 circuit system was assembled and mounted in the data acquisition box, where easy access to power and the required signals was readily available. A negative 12-volt supply line had to be constructed by Richard Atkinson in order for the circuit to behave properly and this was accomplished through the use of a MAX765 circuit.

The operating signals were then hardwired into the circuit. The pressure signal from the transducer in cylinder 3 was carried from the engine through a charge amplifier and then through a power supply with an amplification of 1X (both units supplied by PCB, Piezotronics, Inc.). From there, the signal was sent into the MAX121 circuit for digitizing and synchronization with the DSP-based EVM. At the same time a signal indicating the power stroke of cylinder #3 was generated by a custom engine-control system designed and built by Richard Atkinson and Talus Park for an unrelated project. The optical encoder signal was also used to pace the digital conversion of the MAX121 and to ensure that the pressure signal was sampled at 1024 per revolution (Figure 18), once every second engine revolution.

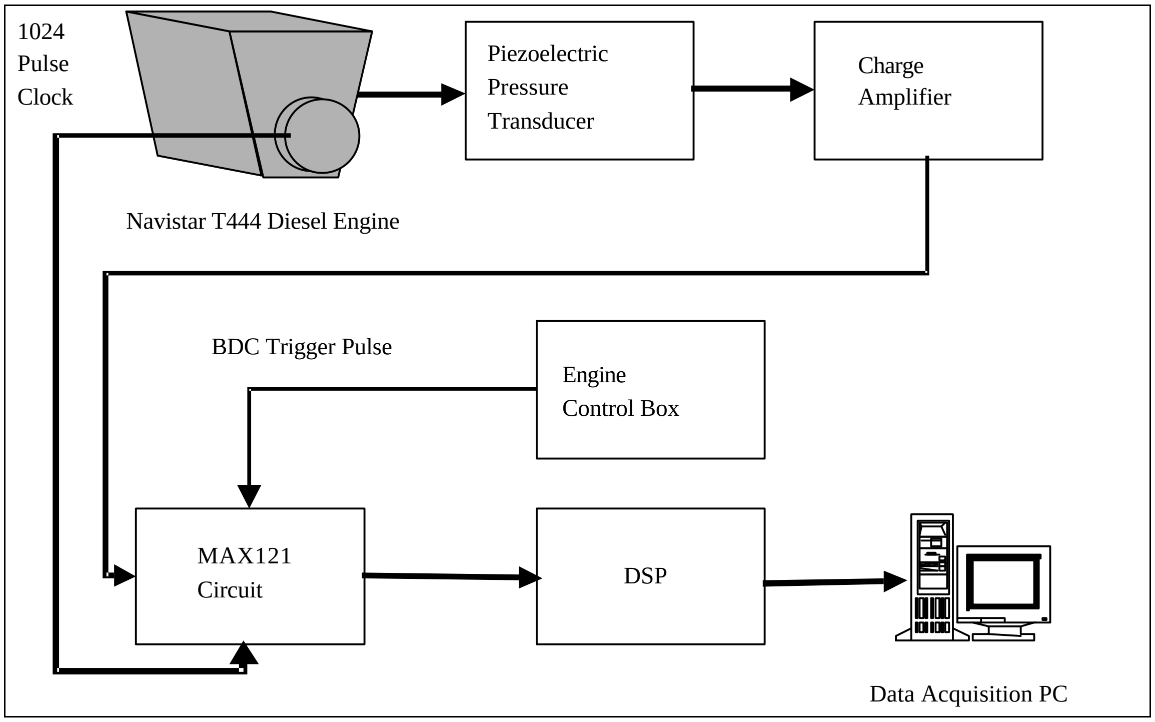


Figure 18: Schematic showing in-cylinder pressure signal progression.

The reason for this was the way in which the DSP sampled and reduced the pressure signal. The DSP sampled the data during the power stroke and performed a simple pressure peak search during that time, but the time allowed between sampling points was insufficient to provide the other variables. Consequently, the rest of the in-cylinder parameters were calculated during the pumping stroke. Due to the processing-power constraints of the DSP chip itself, the algorithms had to be tailored to perform during only the more crucial periods in the cycle. The mass-fraction-burned algorithms, for instance, were only calculated from -25° to 45° after top dead center. This allowed the DSP to calculate the variables in the required amount of time and still leave enough time to transfer the data to the host PC for permanent storage on the hard disk drive. Because the host PC and the DSP evaluation module card required confirmation of reads and writes to the communications port, there was no effective way to place the in-cylinder variables into a readable-at-any-time memory buffer, resulting in one sample per every second revolution. Although this complicated the results verification phase of the study, it had to suffice.

The MAX121 circuit introduced a voltage divider which had to be accounted for during the pressure conversion, and later test results indicated that there was significant high-frequency noise on the signal, leading to the addition of a 224 pico-farad capacitor between the signal and ground to act as a filter.

The EVM had to be programmed correctly in the C language in order to operate properly. For the most part, the code for the pressure reduction algorithms was converted from the BASIC code used in the off-line program. This had to be integrated into the register level programming code that instructs the EVM to communicate with the PC host and acquire the data synchronously from the MAX121 chip. After some debugging, the program was brought to an acceptable operational level. The code can be found in Appendix D.

8. Network Validation

8.1 Data Acquisition and Processing

In order to validate the theory that a neural network trained on non-transient steady data with in-cylinder pressure variables could adequately predict gaseous emissions over a transient cycle, the corresponding transient in-cylinder pressure variables had to be acquired in real time. This was accomplished through the use of the DSP board (described previously) acquiring data at a rate of every other engine revolution over the complete FTP cycle.

Once the data had been assembled into one large file, a couple of preliminary steps needed to be taken in order to improve the results. First, each pressure variable was sent through a 7-point central point average to smooth out the continuous pressure "signals" (Figure 19). Seven points were chosen for the moving average based on examination of the smoothing effects of the filter. Moving averages with fewer points cleaned the signal to a certain extent, but in the author's judgement, seven points performed the best without adversely affecting the natural trends of the data.

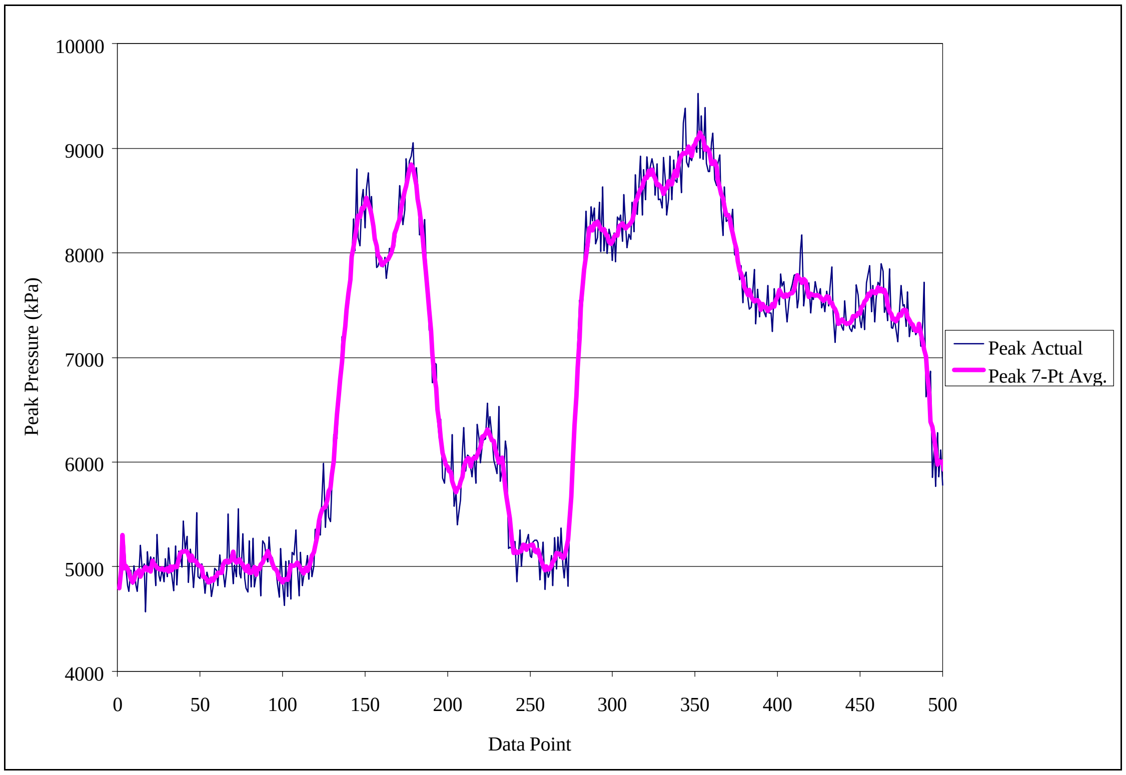


Figure 19: Comparison of measured peak pressure values vs. 7-pt. central point average smoothing.

Next, some of the variables were filtered to remove physically impossible results, caused by the reduction algorithms which could not adequately cope with engine idle conditions. When there is very little combustion occurring in the engine or when the engine is being motored by the dynamometer, the algorithms cannot successfully determine whether or not there is a mass-fraction-burned curve. Consequently, values for the location of maximum burn and the start of injection could occur, inaccurately, at 25° before top dead center. Obviously, the algorithm was defaulting to the first value calculated. A simple band-pass filter was applied in Microsoft's Excel spreadsheet program to eliminate these physical impossibilities.

As a result of the sampling system, the emissions data have slight anomalies. Primarily, the speed at which they were acquired varies over the cycle. The effective sampling rate varies from 6 to 22 Hz depending upon the point in the cycle at which it was taken. Unfortunately, the delays in the exhaust system, the dilution tunnel, and the analyzers themselves conspire to shift this information in time. The delays mean that when the engine changes speeds, the emissions are not sampled at the same rate that the in-cylinder pressure parameters are sampled. However, the neural networks predict emissions values based on the in-cylinder pressure data for a specific time step that do not match the emissions taken at the same time step. This effectively introduces a varying lag in both the time and data sample domains. Additionally, the dispersive effects of the dilution tunnel complicate matters as emissions generated by the engine do not behave as solid blocks on the journey to the sampling probes. Changing volumetric flowrates from the changes in engine speed enhance these dispersive effects. For these reasons, the results presented have not been shifted in time.

The networks for each emission were selected from its top 20 trained networks and NeuroShell converted the weight and summation calculations into C++ source code. This source code was called from a simple file reading program (Appendix F) and the results were written to a separate file.

The length of the FTP cycle is 20 minutes, but due to the changing data-acquisition sampling rate, the final tally was 13000+ data points. Each data point had 10 pressure-related parameters associated with it as well as the emissions data. The networks attempted to predict the values of the emissions data with the instantaneous-mass-flow-rate value. Interestingly, the best network found in the top 20 trained networks did not necessarily produce the best results.

8.2 NO_X Prediction Results

Results for the network with the top training results are shown in Figures 20 through 22. Contrast these with the results shown for the best network the author could find after trial and error and with some insight into the best combination of inputs (Figures 23 through 25). Note: The R^2 values mentioned in the captions refer to the training-set values and not to the predicted vs. actual values.

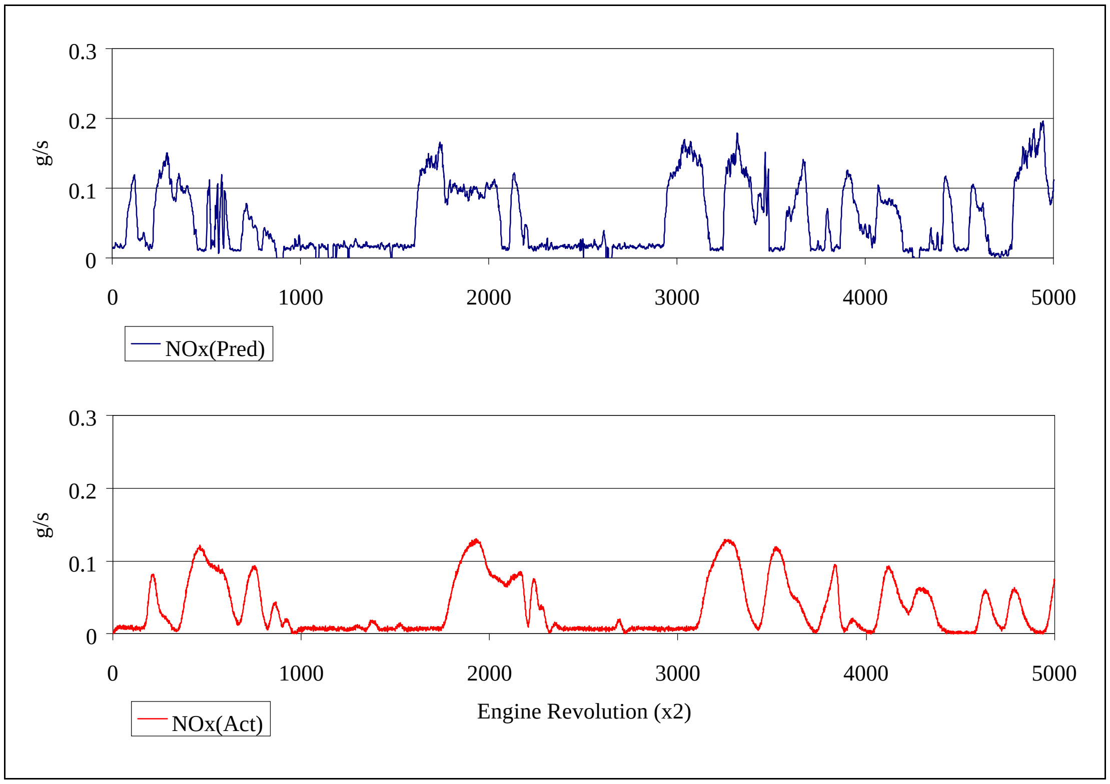


Figure 20: NO$_X$ prediction comparison, top network, first 5000 points of FTP (CombDur, IMEPg, LMFB50, IgnDel, MaxQ; 3 Layer; 18 HN; .998 avg. R^2).

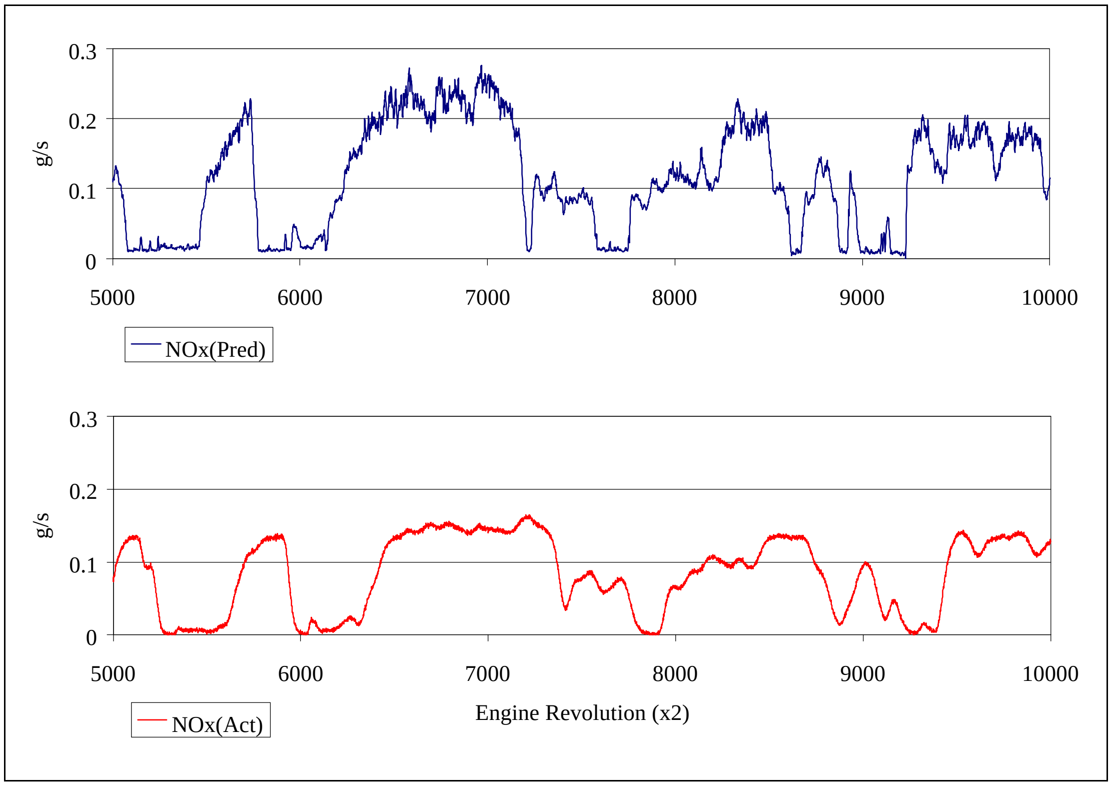


Figure 21: NO$_X$ prediction comparison, top network, second 5000 points of FTP (CombDur, IMEPg, LMFB50, IgnDel, MaxQ; 3 Layer; 18 HN; .998 avg. R^2).

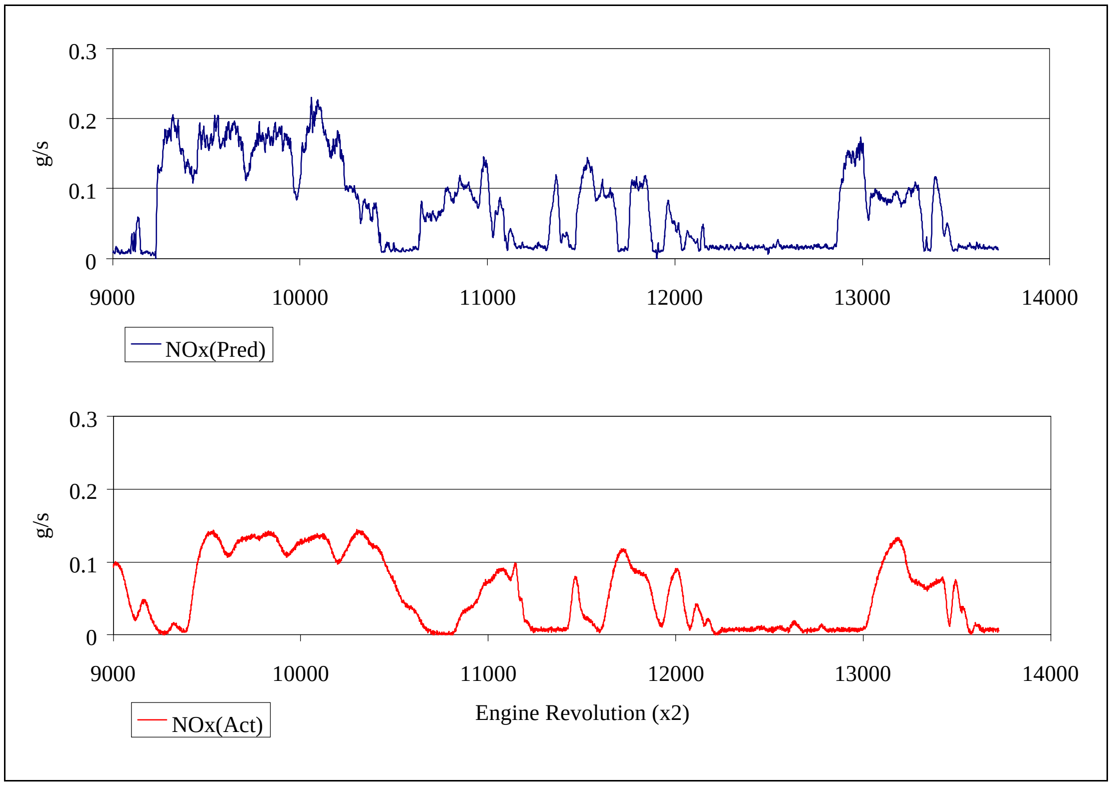


Figure 22: NO$_X$ prediction comparison, top network, final 5000 points of FTP (CombDur, IMEPg, LMFB50, IgnDel, MaxQ; 3 Layer; 18 HN; .998 avg. R^2).

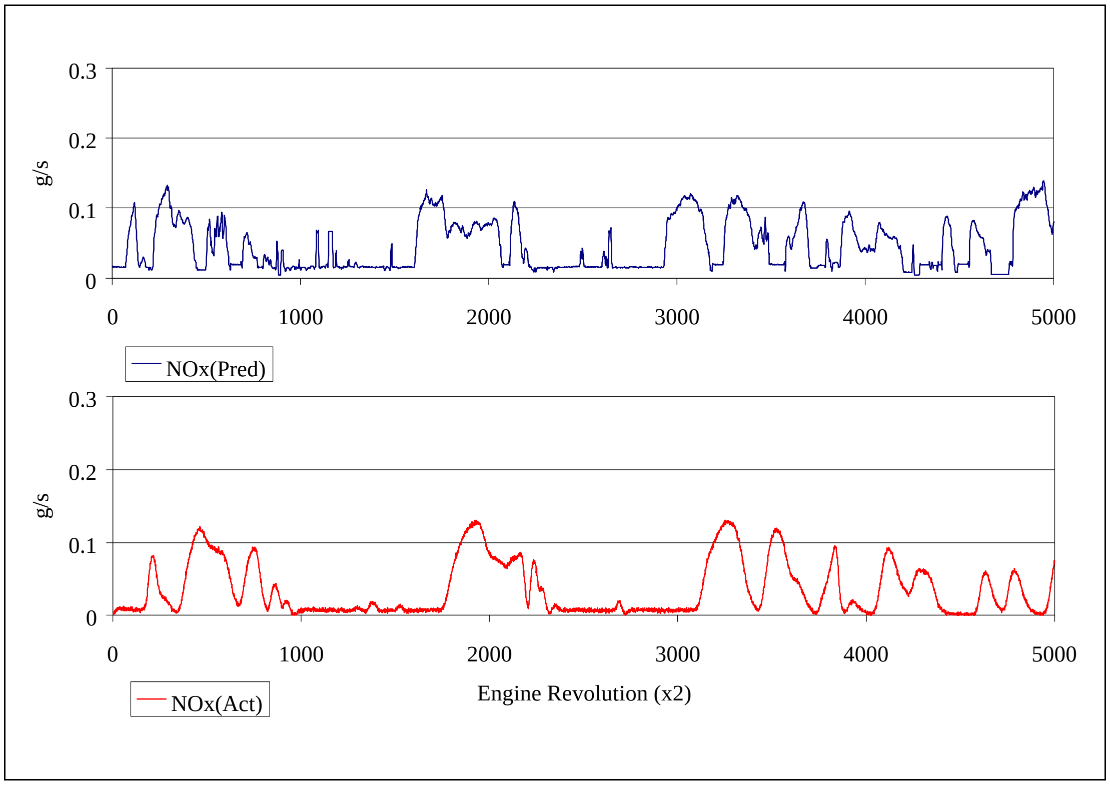


Figure 23: NO$_X$ prediction comparison, best result, first 5000 points of FTP (Peak, CombDur, IMEPg, LMFB50, IgnDel; Ward 1; 10 HN; .996 avg. R^2).

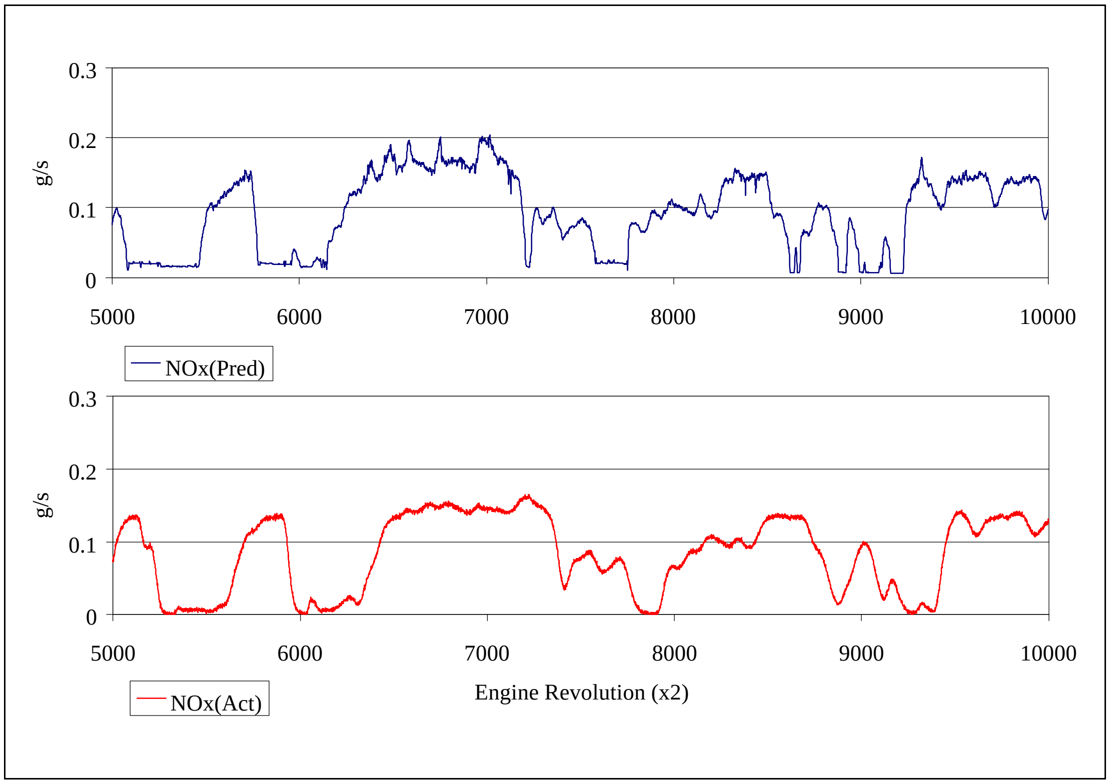


Figure 24: NO$_X$ prediction comparison, best result, second 5000 points of FTP (Peak, CombDur, IMEPg, LMFB50, IgnDel; Ward 1; 10 HN; .996 avg. R^2).

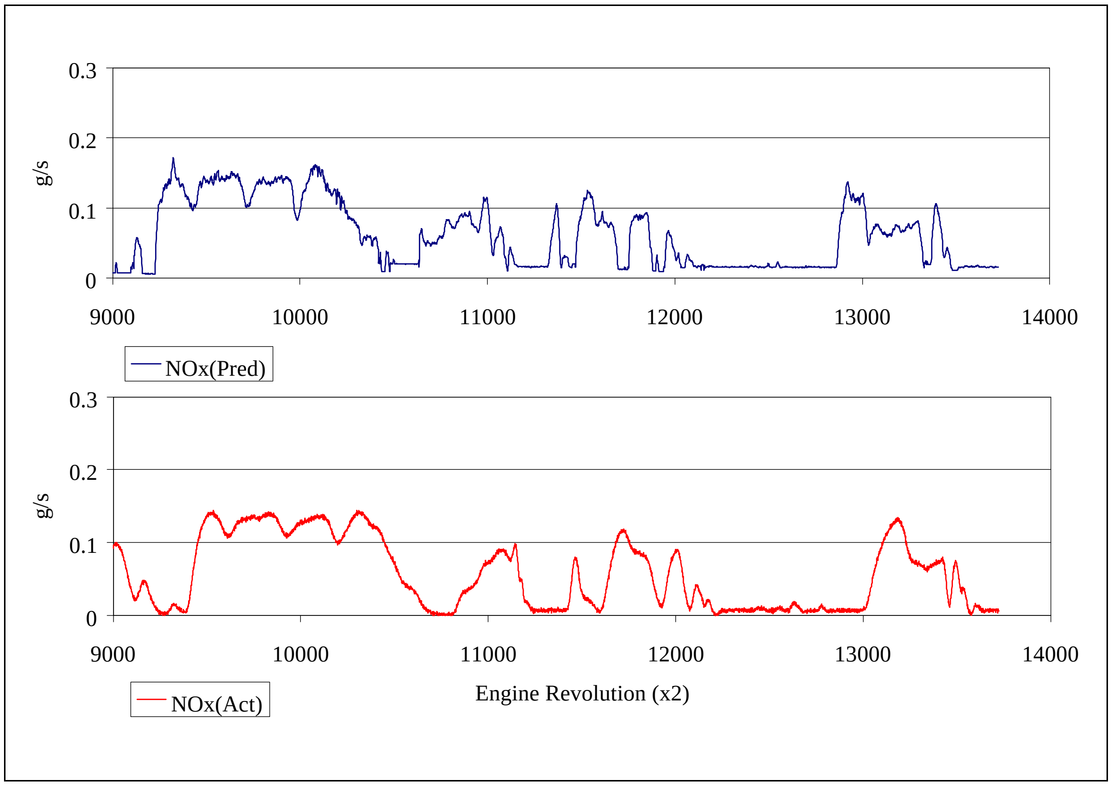


Figure 25: NO$_X$ prediction comparison, best result, final 5000 points of FTP (Peak, CombDur, IMEPg, LMFB50, IgnDel; Ward 1; 10 HN; .996 avg. R^2).

Obviously, the results indicate that the best network results from the training regimen do not accurately reflect the best results for this particular application. As can be seen from the charts, the top-rated trained network grossly overestimated the actual values during the middle section of the FTP. The only major difference between that network and the one with more accurate results is the replacement of the maximum heat release with the peak pressure. This comes as little surprise as the maximum-heat-release value was the only one of the parameters that did not translate well into the new acquisition system. The DSP-based system sampled a signal slightly smaller than one volt with a 14-bit ADC chip. Consequently, noise in the high-frequency range would show up well in the least significant bits of the digitized values. The DAS-58 system, in contrast, used a signal that was amplified 6 times and which was sampled with a 12-bit conversion. Furthermore, the DAS-58 has a high-frequency filter built into the board to avoid such errors, while the DSP-based system had a small 224 picofarad capacitor hardwired in to try to alleviate the problem. Analysis of the data showed that the noise was still significant. For these reasons, the DSP-based system suffered from noise problems not found in the training data but readily apparent in the testing data. Parameters derived from the maximum heat release rate did not suffer, however. This is due to the integral value of the mass-fraction-burned amount. Oscillations due to the high frequency noise would cause maximum values to be significantly higher for any given instant, but the cumulative effect of both high and low values caused the integrated value to conform quite well to the values found in the DAS-58 setup. This is why only the MaxQ variable seems to be at fault. Unfortunately, a large number of the trained networks relied upon the MaxQ variable as an important input. An improved data acquisition system that takes into account these high-frequency noise issues might produce better results. Smoothing of the MaxQ data would alter the information too drastically as the resolution is small enough that the peak values would be reduced by a significant amount. At any rate, the network trained without MaxQ performed quite well despite the migration to a different acquisition system. This demonstrates the robustness of the prediction technique.

The results produced by the better NO_X network (5 inputs, 10 hidden nodes, and a Ward 1 architecture) follow the actual values very closely despite the viewable time delay involved. There is no good way to tell if the high instantaneous spikes predicted are accurate or not since the analyzers, tunnel, and exhaust system all work to damp out and disperse (or "smear") them in the actual values. This network also avoids the MaxQ problem as it is not one of the inputs used. Several networks were applied to the data and the networks that did not incorporate MaxQ performed better on a consistent basis. For the most part, those networks that used MaxQ overpredicted the values of NO_X. One can see this effect in the top network example, while the best network found, has a prediction level in line with the actual values. These overpredictions are more than likely linked to the high frequency noise problem in the MaxQ calculations that consistently overstated the MaxQ value compared to the training data.

To get a better representation of how the predicted and actual emissions values follow each other, the data files were translated into the time domain. The engine speed had been recorded and since sampling was performed at every other revolution, an equivalent time series could be calculated. The actual emissions data series was then shifted backwards in time to approximate the delays involved. Because the sampling rate is dependent upon the engine speed, the shift was not constant for the entire test period. However, there is a period during the middle of the FTP where engine speed does not vary so widely that a reasonable time shift couldn't be done. Still, a sharp eye can catch the slight distortions due to the sampling rate variation in the data. Figures 26 and 27 show the results for NO_X from 300 to 900 seconds into the test.

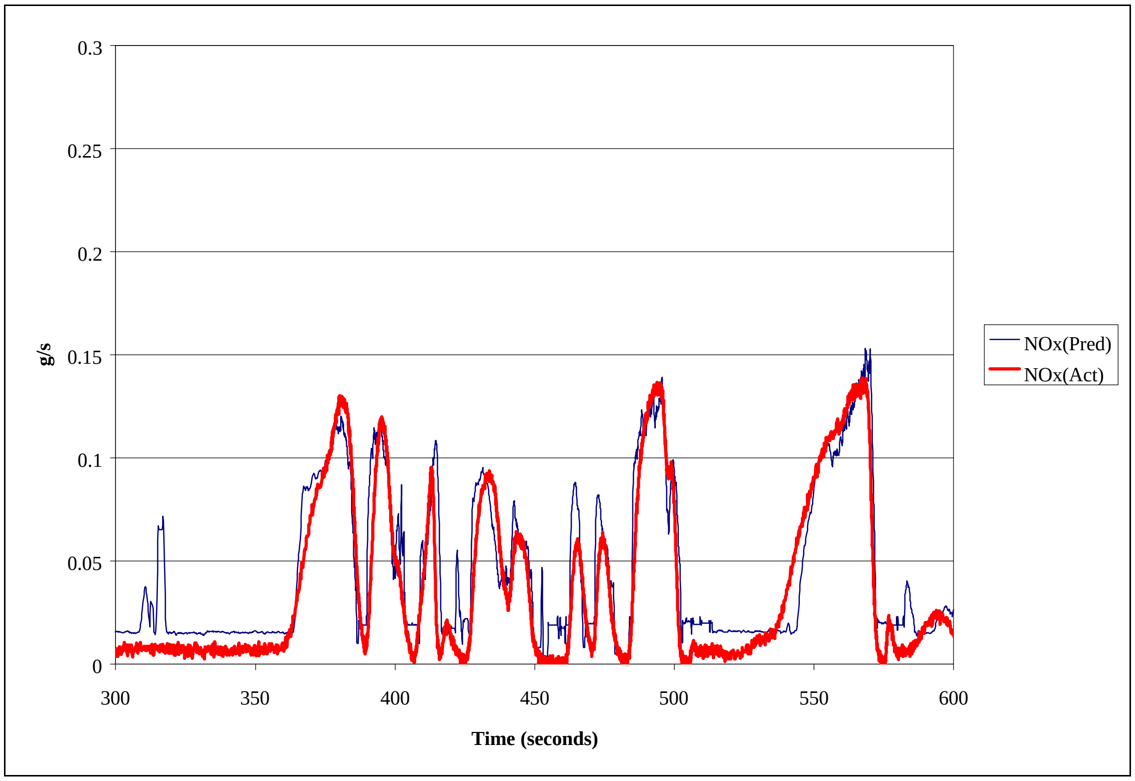


Figure 26: NO_X prediction comparison, time series-based, best result, 300 to 600 seconds in FTP (Peak, CombDur, IMEPg, LMFB50, IgnDel; Ward 1; 10 HN; .996 avg. R^2).

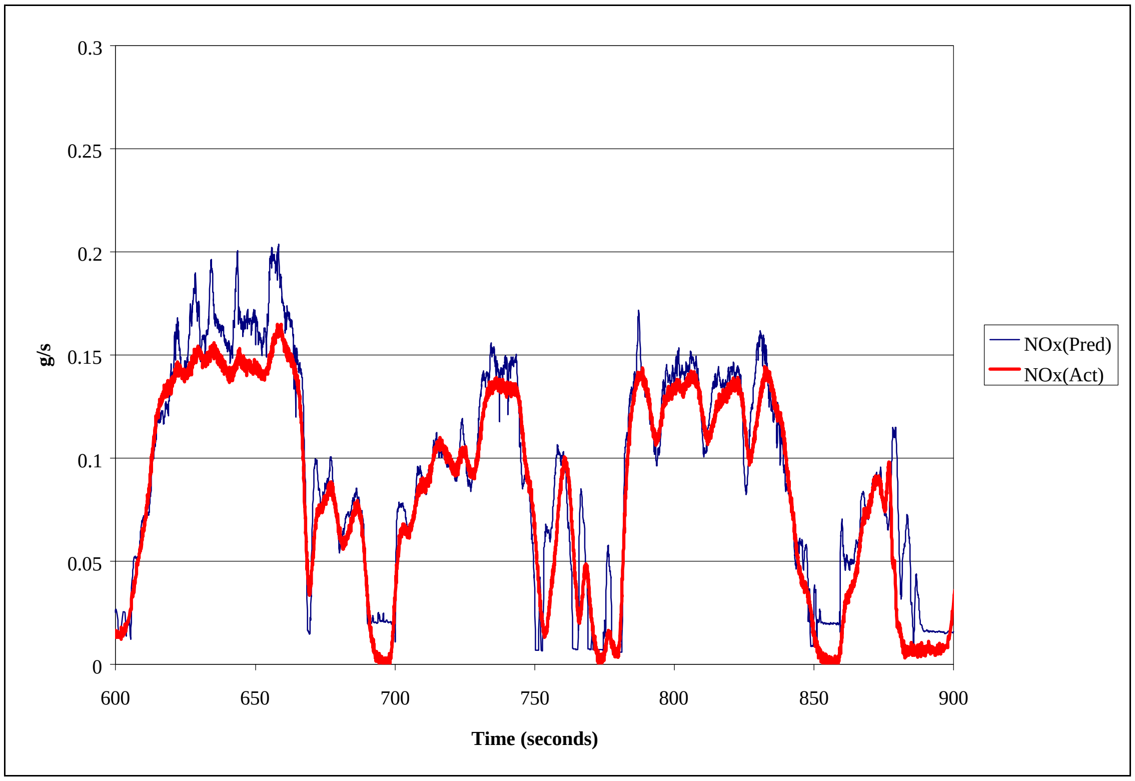


Figure 27: NO_X prediction comparison, time series-based, best result, 600 to 900 seconds in FTP (Peak, CombDur, IMEPg, LMFB50, IgnDel; Ward 1; 10 HN; .996 avg. R^2).

8.3 CO_2 Prediction Results

Results for the CO_2 networks are presented in Figures 28 through 33. The top trained network is presented first, while a network that produced better results is second. Again, the R^2 values refer to the training data and not the actual-vs.-predicted results.

As the charts indicate, there is not a large difference between the top trained network and the best network found. This is largely due to the lack of dependence on the variable MaxQ for the CO_2 networks. Consequently, the CO_2 networks gave more or less the same predictions at the same level of relative accuracy.

The success of the NO_X and CO_2 predictive networks can be attributed to the strong relationship between the geometrically-related variables such as peak pressure and IMEPg and the formation of both of these gaseous emissions. It shows that the emissions with a strong dependency on readily measurable in-cylinder parameters can be quite successfully predicted given the right architecture and combination of inputs. HC and CO present a more challenging problem as the formation of these gases are not strongly related to the measured variables selected.

The CO_2 predictive network was also shifted in time for the section of the FTP covering 300 to 900 seconds. The same technique was used as the NO_X predictive networks and the results for CO_2 are shown in Figures 34 and 35.

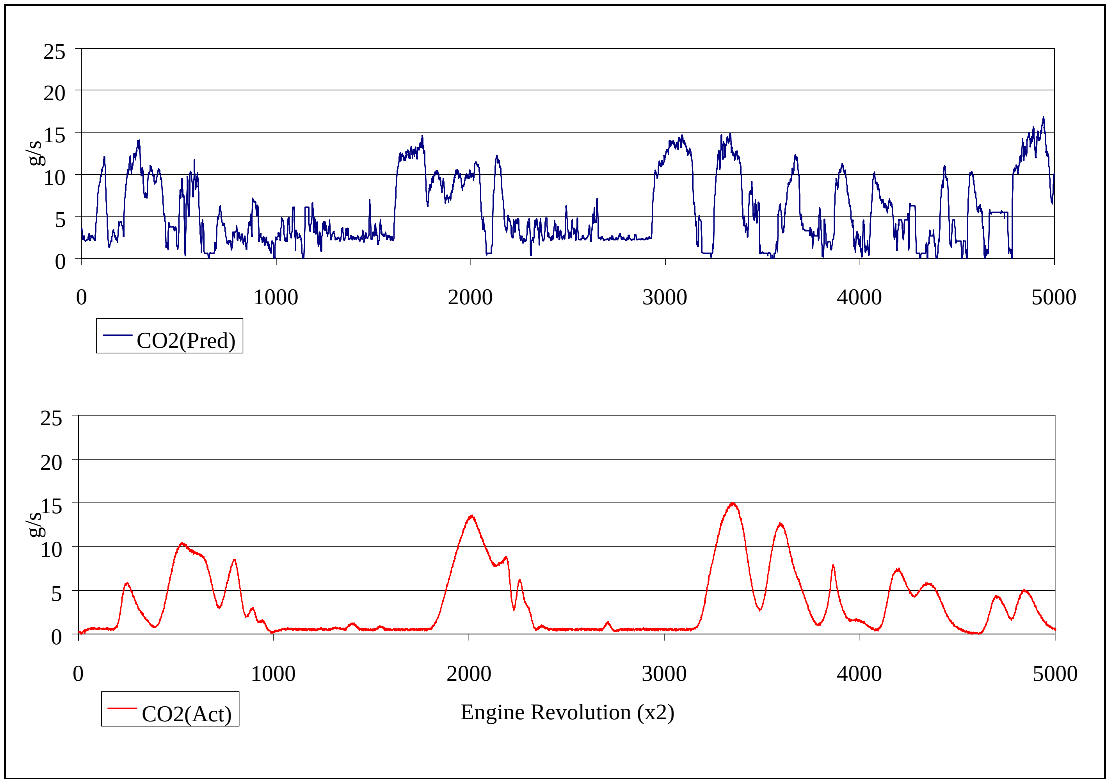


Figure 28: CO_2 prediction comparison, top network, first 5000 points of FTP (Peak, CombDur, LMFB50, MaxBurnLoc, IgnDel; 3 Layer; 16 HN; .999 avg. R^2).

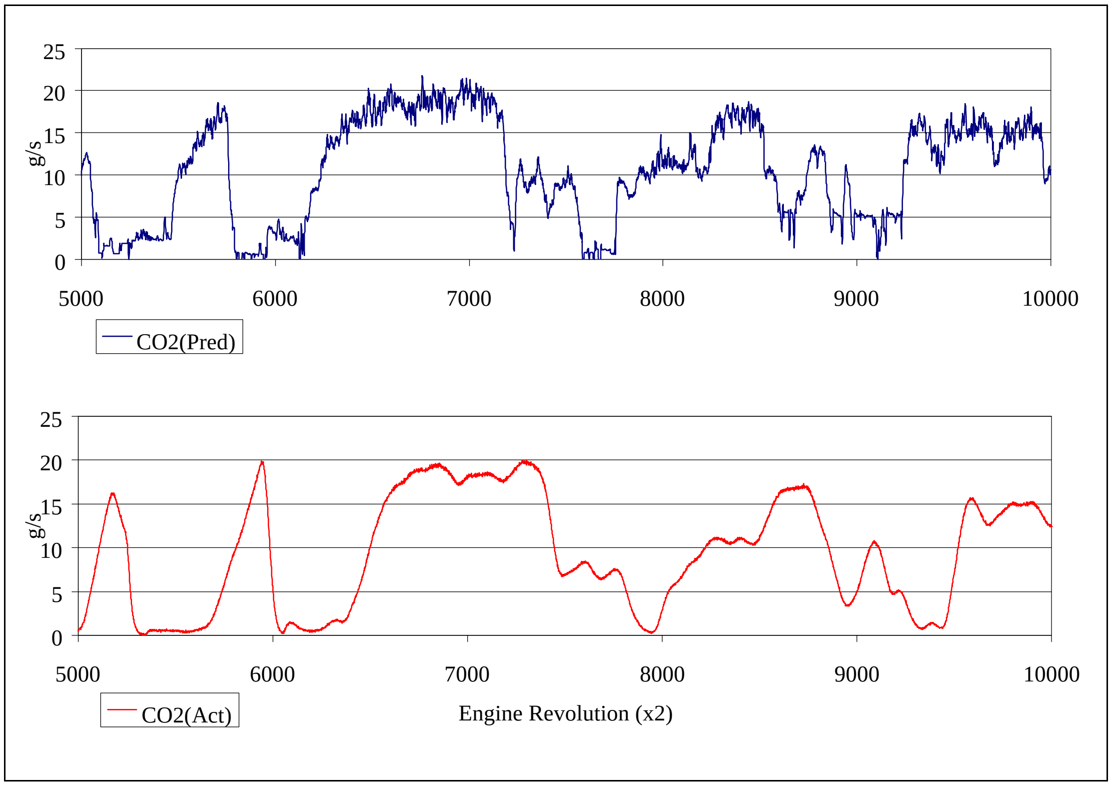


Figure 29: CO_2 prediction comparison, top network, second 5000 points of FTP (Peak, CombDur, LMFB50, MaxBurnLoc, IgnDel; 3 Layer; 16 HN; .999 avg. R^2).

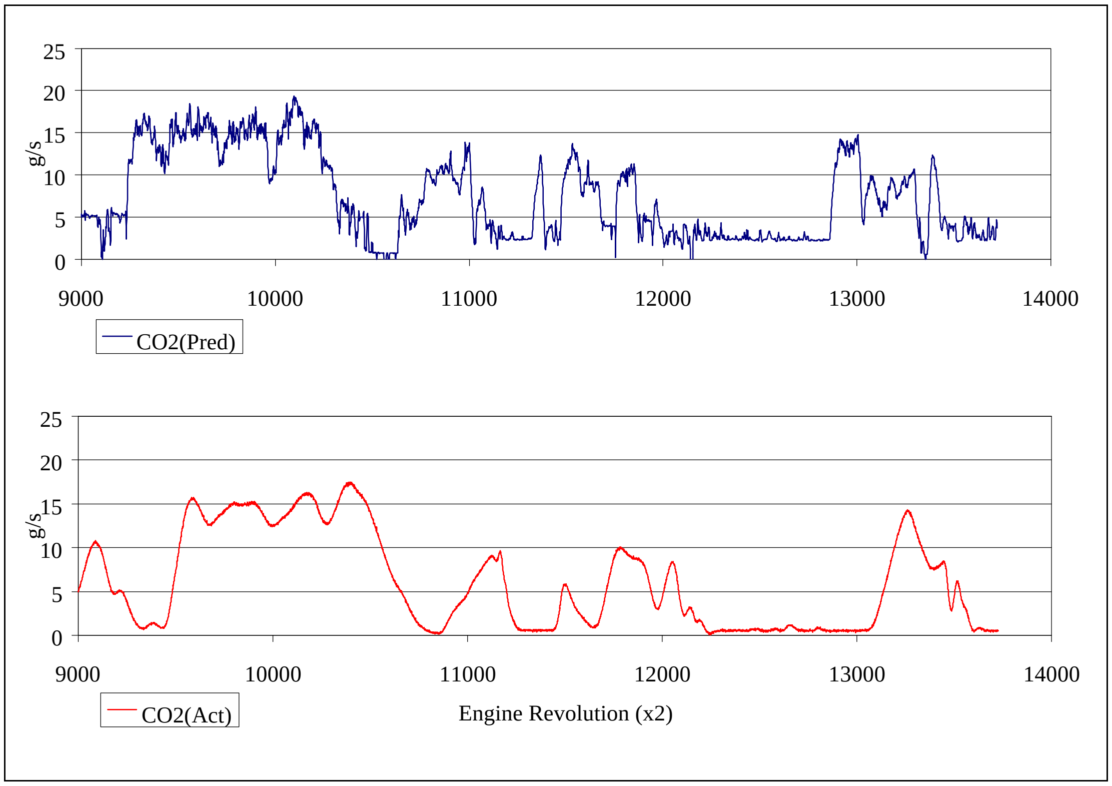


Figure 30: CO_2 prediction comparison, top network, final 5000 points of FTP (Peak, CombDur, LMFB50, MaxBurnLoc, IgnDel; 3 Layer; 16 HN; .999 avg. R^2).

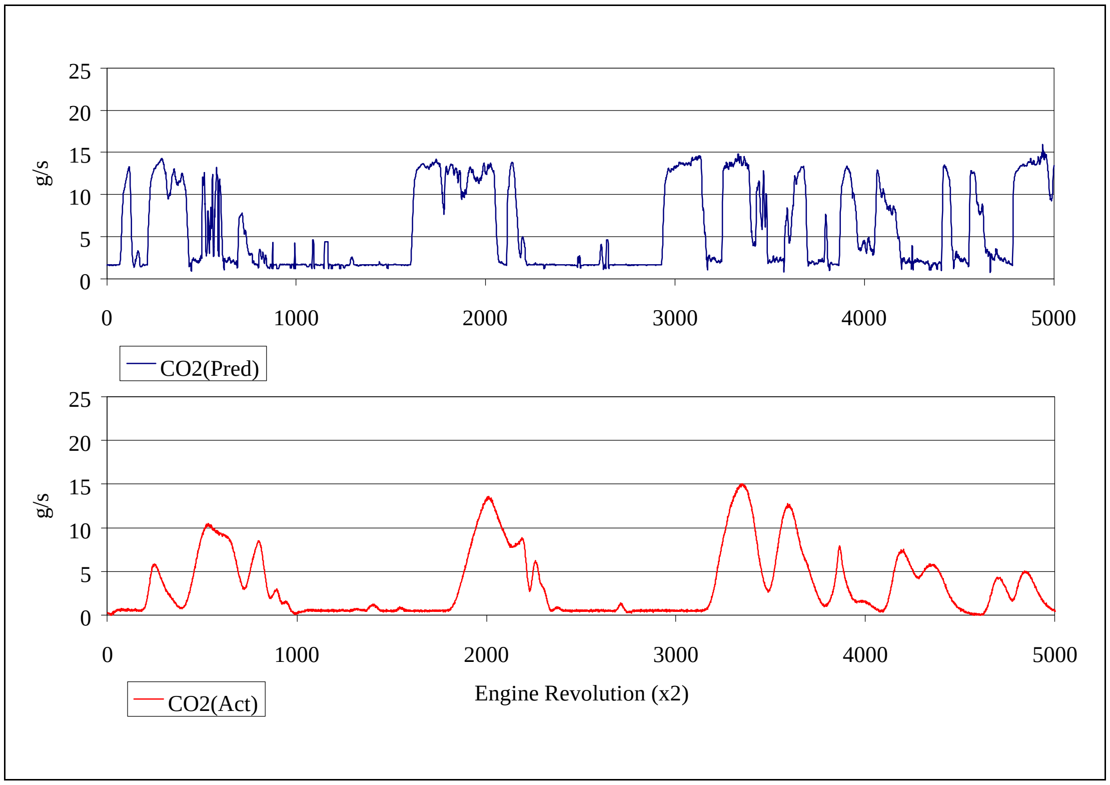


Figure 31: CO_2 prediction comparison, best network, first 5000 points of FTP (Peak, IMEPg, MaxBurn, IgnDel; Ward 1; 9 HN; .998 avg. R^2).

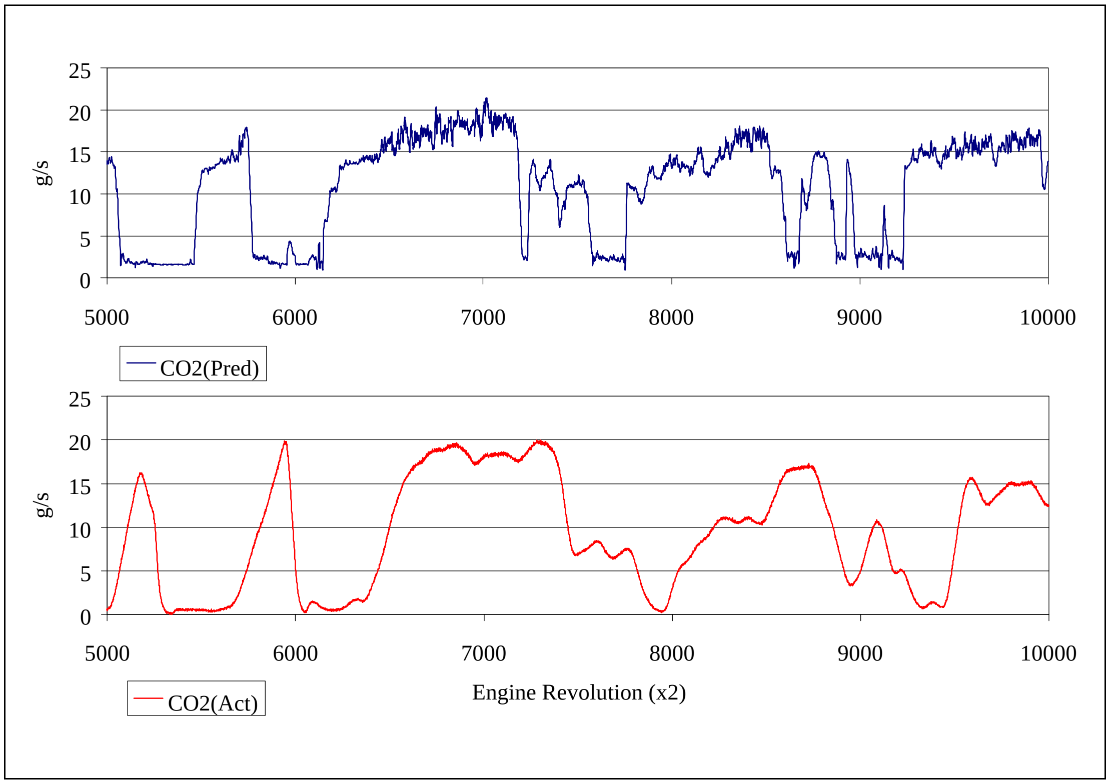


Figure 32: CO_2 prediction comparison, best network, second 5000 points of FTP (Peak, IMEPg, MaxBurn, IgnDel; Ward 1; 9 HN; .998 avg. R^2).

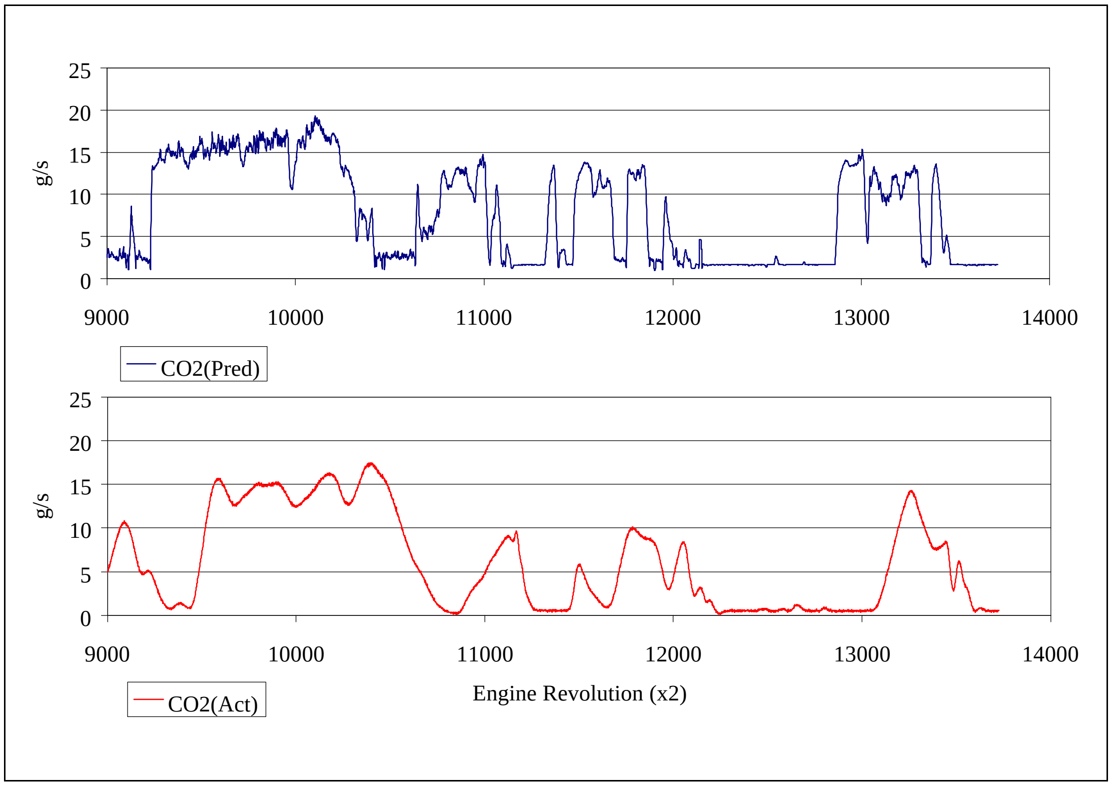


Figure 33: CO_2 prediction comparison, best network, final 5000 points of FTP (Peak, IMEPg, MaxBurn, IgnDel; Ward 1; 9 HN; .998 avg. R^2).

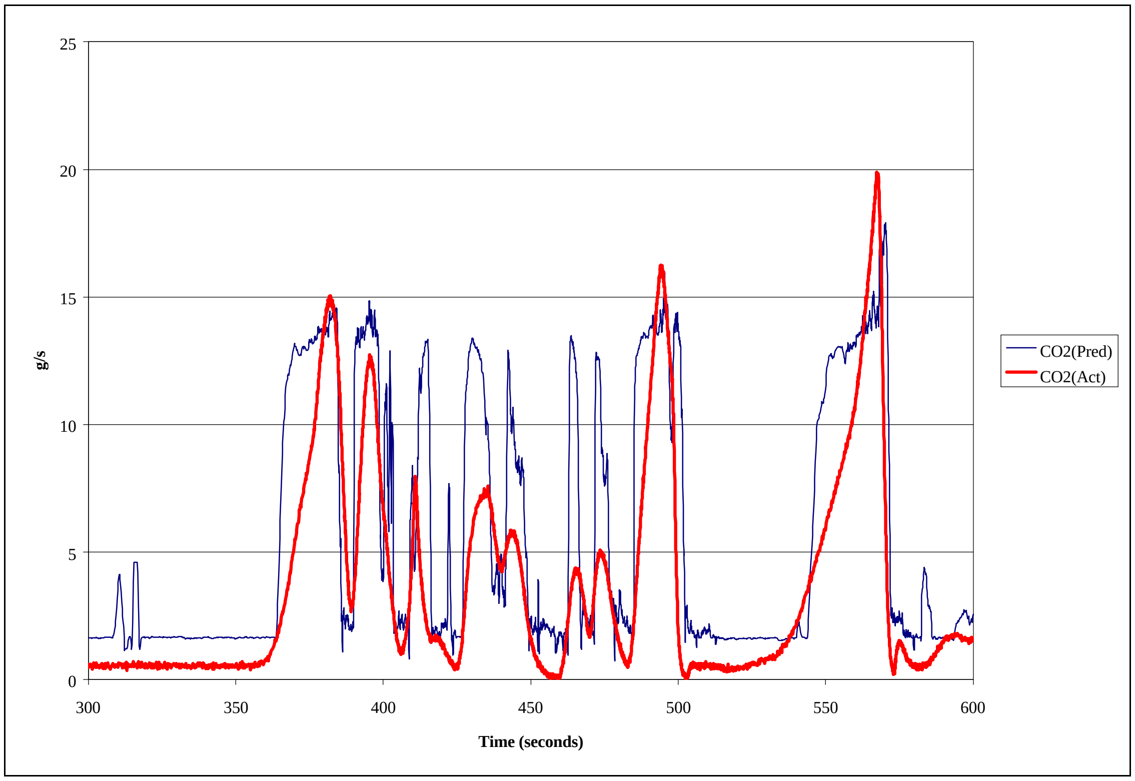


Figure 34: CO_2 prediction comparison, time series-based, best network, 300 to 600 seconds in FTP (Peak, IMEPg, MaxBurn, IgnDel; Ward 1; 9 HN; .998 avg. R^2).

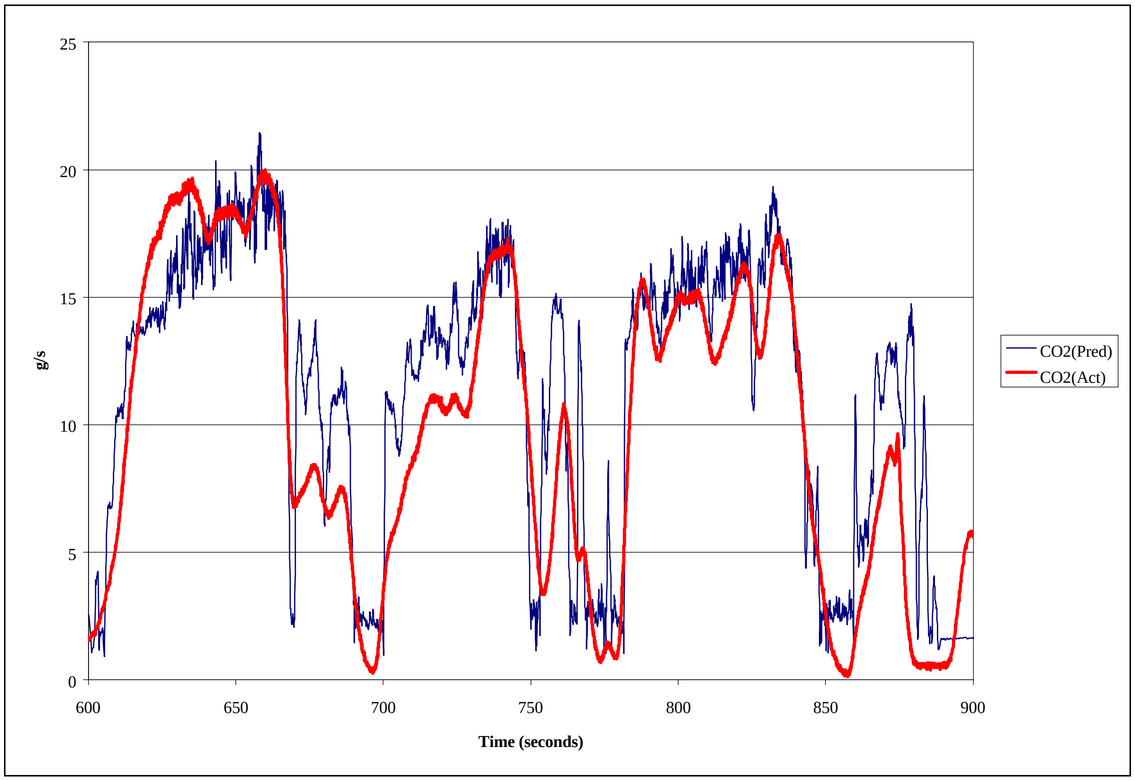


Figure 35: CO_2 prediction comparison, time series-based, best network, 600 to 900 seconds in FTP (Peak, IMEPg, MaxBurn, IgnDel; Ward 1; 9 HN; .998 avg. R^2).

8.4 CO and HC Prediction Results

Despite promising results from the training sessions, successful predictive networks for hydrocarbons and carbon monoxide proved difficult to achieve. Although only the first row networks from the top 20 are presented (Figures 36 and 37), they are typical of several different network input/architecture combinations that were tried. Furthermore, only the first 5000 points of the FTP are shown, as the rest of the run is similar.

As the charts demonstrate, the HC and CO nets were not at all good at predicting the actual gaseous emissions. In fact, the predicted values look only slightly less chaotic than a signal devoted only to noise. There are potentially several reasons for this, primarily the prevalence of MaxQ as an input value to many of the networks for HC and CO. MaxQ seems to have a large correlation to these emissions as it appears in all of the top 20 trained networks for CO, and 16 of the 20 for HC. Trials with other networks that had been trained without MaxQ as an input did not fare any better in their predictive capabilities, although the average R^2 values were generally lower as well. On the positive side, the problem means that a very helpful variable has been found for predictive networks to use based on in-cylinder pressure. On the negative side, however, the switch between acquisition systems has sabotaged any hope of finding a good predictive network.

Furthermore, both HC and CO are not produced in large absolute quantities by the engine in question. Therefore, there is not a large variation in the amounts generated leading to more difficulty in determining the relationships between inputs and the actual values of the emissions. In the case of CO_2 and NO_X, the engine produces a wide range of gaseous emissions levels depending upon the engine load and speed setting. This makes the connections in a network easier to determine than would a small variation in a data set. The absolute values of the various emissions gases also vary by several orders of magnitude, making the assumption that several in-cylinder pressure parameters would serve as successful predictors for all emissions gases somewhat suspect. Macro-predictors such as IMEPg may have a high correlation to the large amount of CO_2 produced, but the connection to the very much smaller amounts of CO may not be sufficient for successful predictive capabilities.

Additionally, the transient sections of the FTP produce levels of HC that lie outside of the training matrix. Since the steady-state matrix couldn't supply such high values, the network could not effectively train for them and so the predictive capabilities were insufficient. The neural networks are quite good at interpolation between the steady-state data points, but the transient data introduced values that would have to be extrapolated to. The formation mechanics of HC may also play a role in the difficulty of prediction as they may be dominated by non-in-cylinder-pressure related phenomena such as crevice volume influences and desorption from the lubricating oil layer. Background values for the gases were not a concern as they were recorded to be very near zero.

Analyzer error does not appear to be a huge factor for HC predictive network accuracy, but results for CO indicate that it could have been a primary factor in the poor performance of the predictive network. Figures 38 and 39 show the analyzer error ranges for both HC and CO for a highly transient section of the FTP. The average percentage error for the HC results was 26.9% while for CO it was 62.9%. This indicates that the CO analyzer span gas range was too high for most of the test. The high concentration gas was necessary for some sections of the FTP, but for the majority of time, it resulted in very coarse resolution on the low end. This could partially explain the difficulties that were met when trying to predict the CO emissions.

In the end, the dependence of CO and HC on the one variable that did not survive the acquisition-system shift, the failure of the training matrix to cover all conditions for HC, the small variance in emissions levels over the engine operating range, analyzer error, and quite possibly the failure to choose appropriate in-cylinder pressure parameters led to the failure of the HC and CO networks.

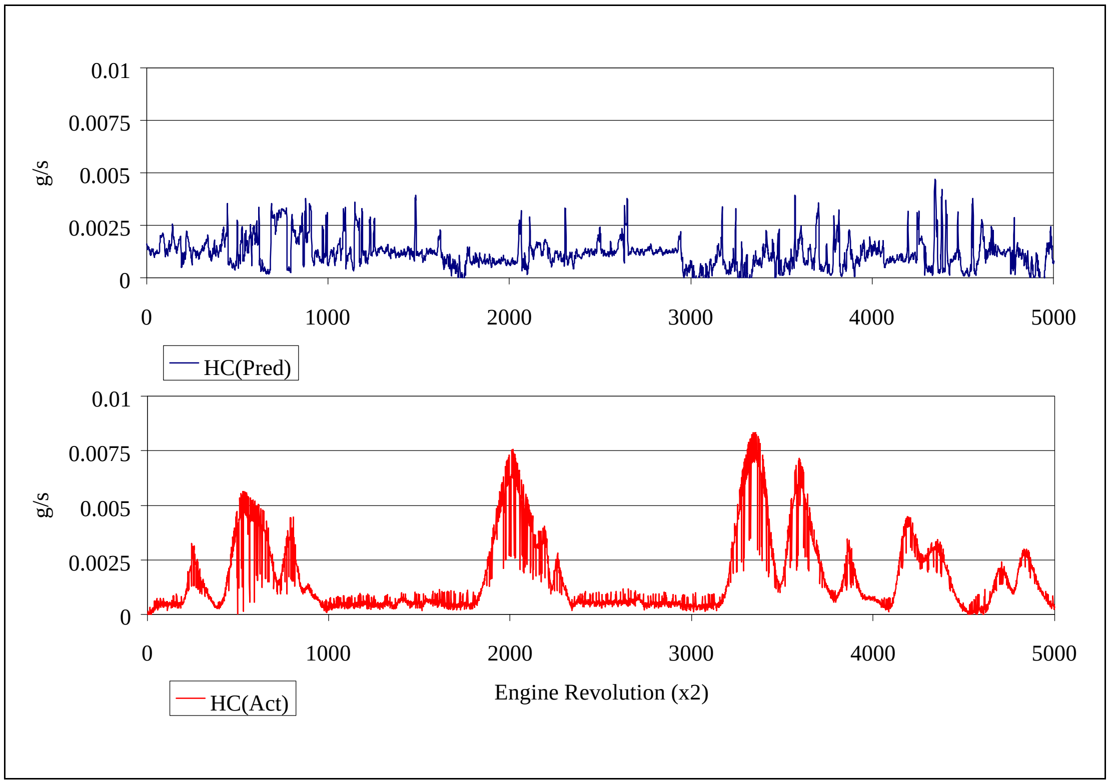


Figure 36: HC prediction comparison, top network, first 5000 points of FTP (IgnDel, CombDur, MaxQ, LMFB50, LPP;3 Layer ; 13 HN; .975 avg. R^2).

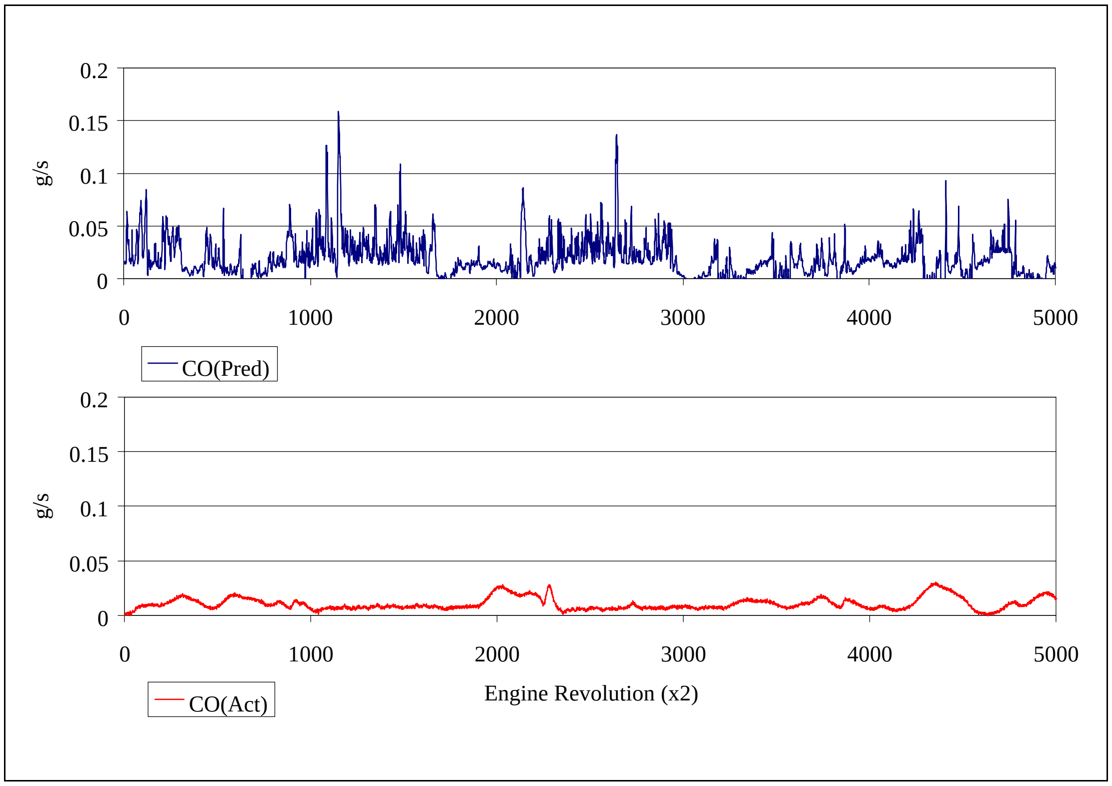


Figure 37: CO prediction comparison, top network, first 5000 points of FTP (Peak, MaxQ, LMFB50; Ward 1 ; 13 HN; .999 avg. R^2).

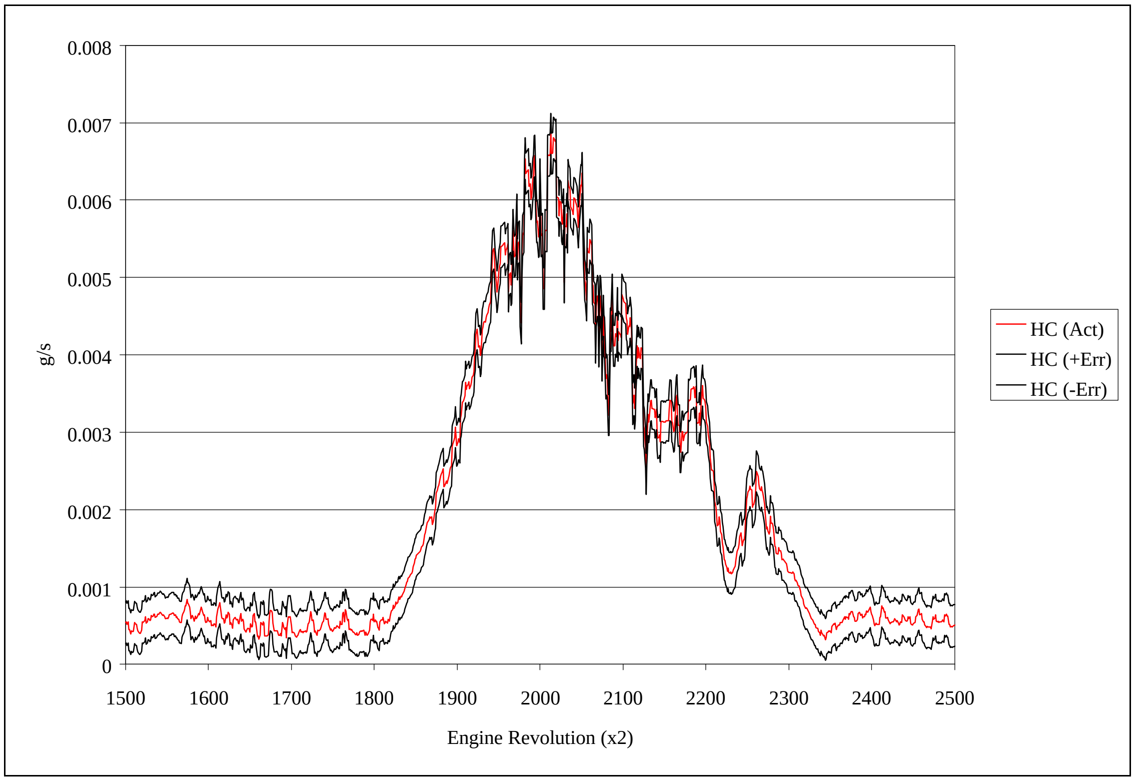


Figure 38 : Actual HC measured with analyzer error range.

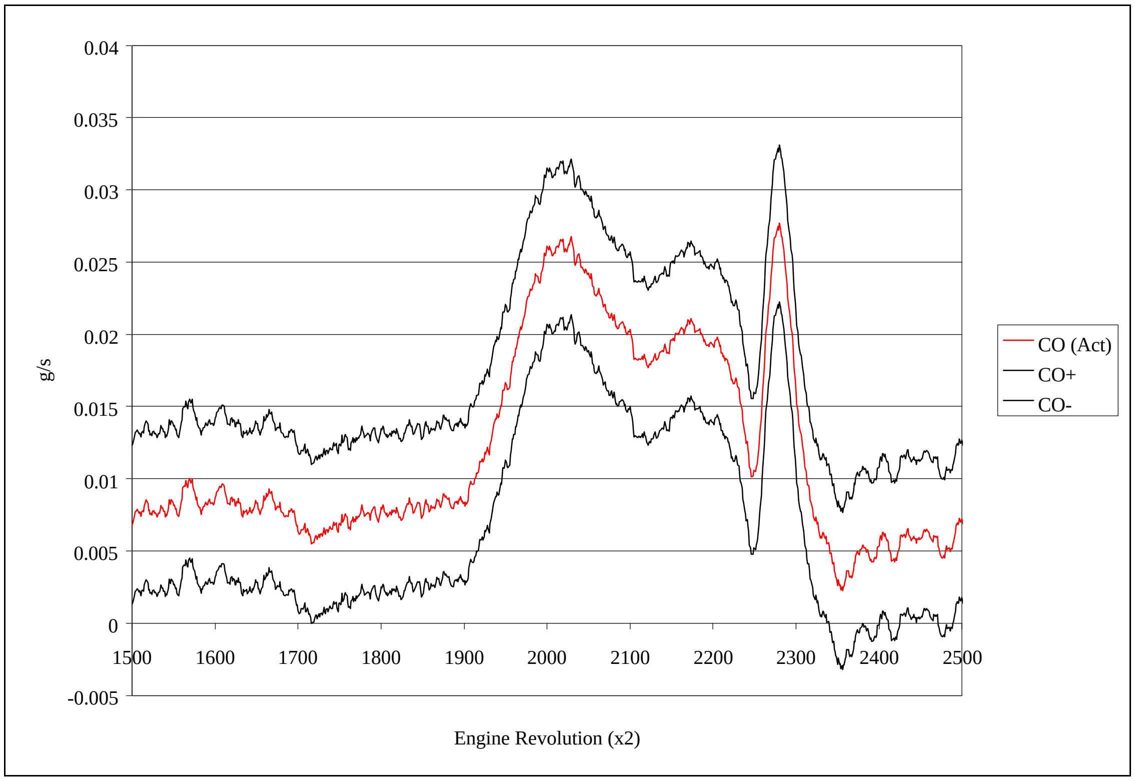


Figure 39 : Actual CO measured with analyzer error range.

9. Conclusions and Recommendations

The success of the NO_X and CO_2 predictive networks demonstrates the validity of the theory proposed at the beginning of this study: that in-cylinder pressure information taken during quasi-steady-state operation of a running engine could be used successfully in a neural network to predict transient gaseous emissions data over a long cycle. Since emissions formation is largely a function of the in-cylinder combustion processes and since these processes can be interpreted through the judicious use of in-cylinder pressure, variables calculated from in-cylinder pressure have a high correlation to the gaseous emissions levels generated. Consequently, these variables are well-suited to neural network applications that predict emissions levels produced by an engine. The alternative to the use of predictive neural networks is computationally intensive physical models that attempt to solve for every potential kinetic rate reaction that a molecule of fuel may encounter during oxidation [Moses, et al., 1996]. From a theoretical standpoint, the modeling will be superior, but from a practical standpoint, the application of these models do not lend themselves well to real world studies until mobile computing sources are significantly faster, cheaper, and lighter.

Errors introduced by the exhaust system, dilution tunnel, and analyzers were reduced in the training phase of the study due to the steady-state nature of the measurements, but emerged again during the test-analysis phase. The use of a steady-state matrix of engine operating points proved effective in training a predictive network for a transient emissions response, but only to the extent that the results could be compared by the human eye. A quantitative interpretation of the results was not possible due to the methodology employed; a time-based measurement vs. an engine speed-based prediction. Future studies should employ a data-transfer method that allows for a steady clock sampling speed. The current system was based on engine speed because the DSP technology required it to provide meaningful in-cylinder pressure information. If the in-cylinder pressure parameters could be stored in a memory buffer between the DSP module and the PC host, software running at a higher sampling speed could read the information constantly, only changing after the power stroke of the engine. This might produce repeated values when the sampling rate and the engine speed diverge, but the ability to compare results against the real values generated by the analyzers would more than compensate for this drawback.

The study has shown that variables based on in-cylinder pressure are adequate for use in predicting the emissions levels of some gases. Specifically, NO_X and CO_2 are prime candidates for predictive neural networks owing to their close correlation between formation rates and variables available from the in-cylinder pressure. Although the study failed to produce networks capable of predicting HC and CO, the high R^2 values found during training suggest that there may be hope for finding them. The failure of the HC and CO networks points out the fallibility of migrating to a separate data acquisition system between the training and testing phases and the necessity to train networks on the widest possible extremes of data from typical operation. The difference in the analog-to-digital conversion between the DAS-58 and the DSP combined with the pressure signal resolution to produce results that did not lend themselves well to the neural networks. A more robust scheme involving both training and testing on the DSP-based system may produce the HC and CO predictive networks sought after. Additionally, the ability to adjust the pertinent engine control parameters might aid in producing steady-state conditions that mimic transient conditions. For example, a set point that used higher than usual amounts of fuel might successfully imitate a transient condition where the engine speed is increasing rapidly. The inability to do so led to the inadequate training matrix for CO and HC. The tendency of the engine to produce low levels of these gaseous emissions also contributed to the difficulties involved in prediction.

Furthermore, the study showed the feasibility of using a DSP-based system to quickly sample and display results from in-cylinder combustion in real time. The possibilities for real-time control of a diesel engine using that information, as well as the potential for real world emissions prediction, promise a bright future for the use of this technology.

In order to achieve better results, future projects might employ the use of fast-reponse emissions analysis equipment located as near as possible to the exhaust manifold to train and validate networks. This would eliminate the dispersive effects of the dilution tunnel and remove the majority of the delay time involved. Further, the samples of gaseous emissions could be directly related to the in-cylinder pressure variables occurring in the cycle previously. This would cut down significantly on the amount of data required to train the network as well as eliminate the necessity of averaging the pressure variables and the emissions values. Cycle-to-cycle variations of emissions would also be eliminated in such a setup leading to an increase in system accuracy.

Particulate matter is of primary concern and every effort should be made to include this in the list of predicted output variables. Predictive capabilities of a particulate matter neural network would probably be on the same order as HC and CO. The same difficulties in training the network as those found with HC and CO would also probably be found. Consequently, if the problems associated with HC and CO can be overcome, particulate matter prediction should also be possible.

This study has only proven the feasibility of using in-cylinder pressure based variables as inputs to a predictive neural network. There is much work to be done to refine the methodology, but it does hold out hope that relevant emissions levels can be found from a real-world driving cycle by installing the necessary hardware into a typical in-use diesel-powered vehicle. Also, the future potential for use in engine management and possibly engine control merit serious consideration.

References

Asik, J., Peters, J., Meyer, G. and Tang, D., “Transient A/F Estimation and Control Using a Neural Network”, SAE Paper 970619, 1997.

Atkinson, C., Traver, M., Tennant, C., Atkinson, R. and Clark, N., "Exhaust Emissions and Combustion Stability in a Bi-Fuel Spark Ignition Engine", SAE Paper 950468, 1995.

Atkinson, C., Long, T., and Hanzevack, E., "Virtual Sensing: an Emissions Prediction System for On-Board Diagnostics and Engine Control", SAE Paper 980516, 1998.

Atkinson, R., Tennant, C., Traver, M., Atkinson, C. and Clark, N., “A Controller for a Spark Ignition Engine with Bi-Fuel Capability”, SAE Paper 942004, 1994.

Bidan, P., Boverie, S., and Chaumerliac, V., “Nonlinear Control of a Spark-Ignition Engine”, *IEEE Transactions on Control Systems Technology*, Vol. 3, No. 1, pp. 4-13, 1995.

Brunt, M. and Pond, C., “Evaluation of Techniques for Absolute Cylinder Pressure Correction”, SAE Paper 970036, 1997.

Brunt, M. and Emtage, A., “Evaluation of Burn Rate Routines and Analysis Errors”, SAE Paper 970037, 1997.

Carnevale, C., Coin, D., Secco, M. and Tubetti, P., “A/F Ratio Control with Sliding Mode Technique”, SAE Paper 950838, 1995.

De Nicolao, G., Scattolini, R. and Siviero, C., “Modelling the Volumetric Efficiency of IC Engines: Parametric, Non-Parametric and Neural Techniques”, *Control Engineering Practices*, Vol. 4, No. 10, pp. 1405-1415, 1996.

Degobert, P., Automobiles and Pollution, Society of Automotive Engineers, Warrendale, Penn., 1995.

Edwards, J., Combustion : Formation and Emission of Trace Species, Ann Arbor Science Publishers, Inc., Ann Arbor, Mich., 1974.

Feldkamp, L. and Puskorius, G., “Training of Robust Neural Controllers”, *Proceedings of the 33rd Conference on Decision and Control*, Lake Buena Vista, FL, pp. 2754-2759, 1994.

Gassenfelt, E. and Powell, J., “Algorithms for Air-Fuel Ratio Estimation Using Internal Combustion Engine Cylinder Pressure”, SAE Paper 890300, 1989.

Grizzle, J. W., Cook, J. A. and Milam, W. P., “Improved Cylinder Air Charge Estimation for Transient Air Fuel Ratio Control”, *Proceedings of the American Controls Conference*, Baltimore, MD, pp. 1568-1572, 1994.

Hanzevack, E., Long, T., Atkinson, C. and Traver, M., “Virtual Sensors for Spark Ignition Engines Using Neural Networks”, *Proceedings of the American Controls Conference*, Albuquerque, NM, June, 1997.

Haykin, S., Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, Inc., Englewood Cliffs, NJ, 1994.

Heisler, H., Advanced Engine Technology, Society of Automotive Engineers, Warrendale, Penn., 1995.

Hendricks, E., Vesterholm, T. and Sorenson, S., “Nonlinear, Closed Loop, SI Engine Control Observers”, SAE Paper No. 920237, 1992.

Heywood, J., Internal Combustion Engine Fundamentals, McGraw Hill, Inc., New York, 1988.

Jones, V. K., Ault, B. A., Franklin, G. F., and Powell, J. D., “Identification and Air-Fuel Ratio Control of a Spark Ignition Engine”, *IEEE Transactions on Control Systems Technology*, Vol. 3, No. 1, pp. 14-21, 1995.

Kaidantzis, P., Rasmussen, P., Jensen, M., Vesterholm, T. and Hendricks, E., “Advanced Nonlinear Observer Control for SI Engines”, SAE Paper 930768, 1993.

Kao, M. and Moskwa, J. J., “Nonlinear Diesel Engine Control and Cylinder Pressure Observation”, *Journal of Dynamic Systems, Measurement and Control*, Vol. 117, pp. 183-192, 1995.

Lancaster, D. R., Krieger, R. B. and Lienesch, J. H., "Measurement and Analysis of Engine Pressure Data", SAE Paper No. 750026, 1975.

Lee, K., Lim, S., Kim, S., Kim, J., Kim, T. and Son, J., “A DSP-Based Fast Data Acquisition System for Analysis of Engine Performance”, SAE Paper 950843, 1995.

Leisenring, K., Rizzoni, G. and Samimy, B., “Methods for Internal Combustion Engine Feedback Control During Cold-Start”, SAE Paper 950842, 1995.

Lenz, U. and Schroeder, D., “Artificial Intelligence for Combustion Engine Control”, SAE Paper 960328, 1996.

Lenz, U. and Schroeder, D., “Transient Air-Fuel Ratio Control Using Artificial Intelligence”, SAE Paper 970618, 1997.

Leonhardt, S., Gao, H. and Kecman, V., “Real Time Supervision of Diesel Engine Injection with RBF-based Neural Networks”, *Proceedings of the American Controls Conference*, Seattle, WA, pp. 2128-2132, 1995.

Leonhardt, S., Ludwig, C. and Schwarz, R., “Real-Time Supervision for Diesel Engine Injection”, *Control Engineering Practice*, Vol. 3, No. 7, pp. 1003-1010, 1995.

Leonhardt, S., Schwarz, R. and Isermann, R., “Real-Time Supervision of the Diesel Engine Injection Process”, SAE Paper 970535, 1997.

Ludwig, C. and Ayoubi, M., “Fault Detection Schemes for a Diesel Engine Turbocharger”, *Proceedings of the American Controls Conference*, Seattle, WA, pp. 2118-2122, 1995.

Microsoft Corporation, Microsoft Excel 97, Office 97 Standard Edition, Redmond, WA, 1996.

Moses, E., Rao, K., and Winterbone, D., "3D Modelling and Photographic Investigation of Combustion in Hydra DI Diesel Engine", SAE Paper 960836, 1996.

Müller, E. and Zillmer, M., "Modeling of Nitric Oxide and Soot Formation in Diesel Engine Combustion", SAE Paper 982457, 1998.

Pestana, G., “Engine Control Methods Using Combustion Pressure Feedback”, SAE Paper 890758, 1989.

Ramli, M. and Morris, A., “Application of Neural Networks to the Automotive Engine Problem”, *Int. J. of Vehicle Design*, Vol. 14, nos. 2/3, pp. 184-193, 1993.

Ramos, J. I., Internal Combustion Engine Modeling, Hemisphere Publishing Corporation, New York, NY, 1989.

Rassweiler, G. and Withrow, L., "Motion Pictures of Engine Flames Correlated with Pressure Cards", SAE Paper 380139, 1938.

Ribbens, W., Park, J. and Kim, D., “Application of Neural Networks to Detecting Misfire in Automotive Engines”, *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech and Signal Processing,* Adelaide, Aust., pp. 593-596, 1994.

Rutland, C., Ayoub, N., Han. Z., Hampson, G., Kong, S., Mather, D., Musculus, M., Patterson, M., Ricart, L., Stephenson, P., and Reitz, R., "Progress Towards Diesel Combustion Modeling", SAE Paper 952429, 1995.

Salam, F. and Gharbi, A., “Temporal Neuro-Control of Idle Engine Speed”, *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, Dearborn, MI, pp. 396 – 401, 1996.

Schafer, F. and van Basshuysen, R., Reduced Emissions and Fuel Consumption in Automobile Engines, Springer-Verlag, Vienna, 1993.

Shayler, P., Goodman, M. and Ma, T., “Transient Air/Fuel Ratio Control of an S.I. Engine Using Neural Networks”, SAE Paper 960326, 1996.

Shiraishi, H., Ipri, S., and Cho, D., "CMAC Neural Network Controller for Fuel-Injection Systems", IEEE Transactions on Control Systems Technology, Vol. 3, No. 1, March, 1995.

Takahashi, S., “A Method for Learning the Varying Parameters of Gasoline Engine Control System Based on δ - Rule”, *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, pp. 2763-2768, 1996.

Tauzia, X., Hetet, J., Chesse, P., Roy, P., and Inozu, B., "A Simulation Study for Medium Speed Diesel Engine Pollutant Emissions and Their Reduction", ASME ICE-Vol. 29-3, pp. 41-48, Fall Technical Conference, Madison, WI, 1997.

Tennant, C., deJong, R., Atkinson, R., Traver, M., Atkinson, C., Vincent, C., Clark, N. and Lyons, D., “Performance of a High Speed Engine with Dual Fuel Capability”, SAE Paper 940517, 1994.

Traver, M., “An Analysis of In-Cylinder Pressure Histories in a Bi-Fuel Engine”, Master’s Thesis, West Virginia University, 1994.

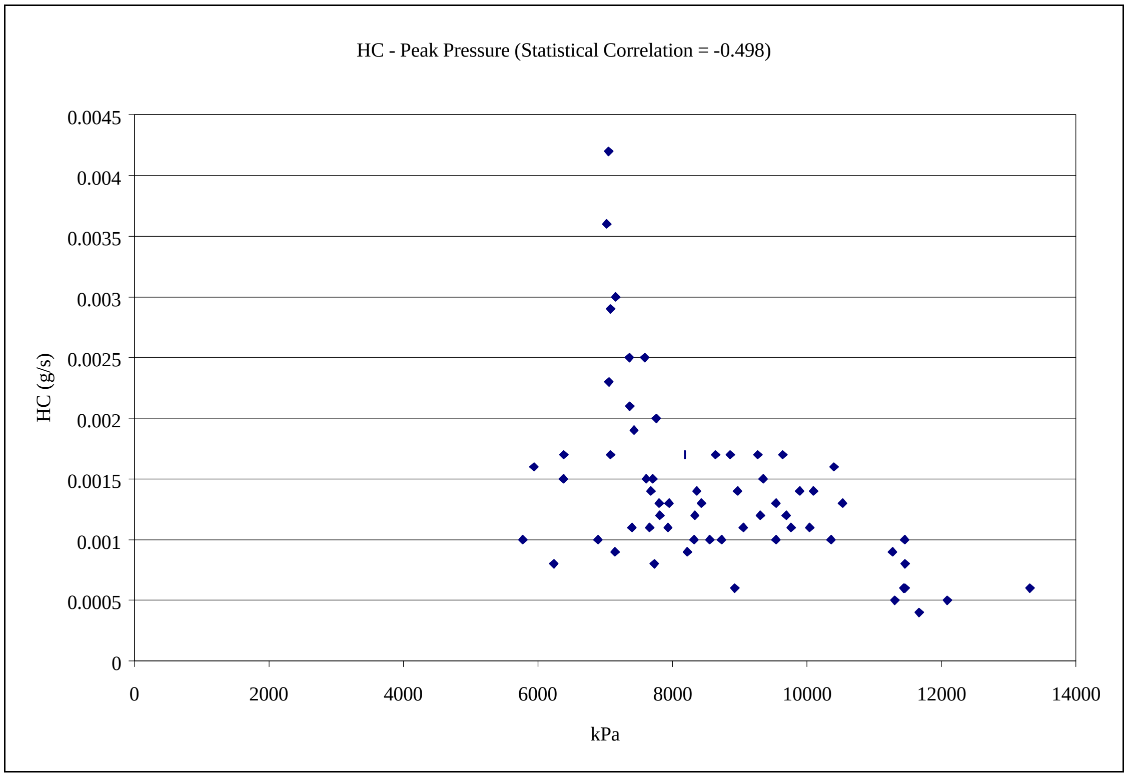
Ward Systems Group, Inc., NeuroShell 2, Release 3.0, Frederick, Maryland, 1996.

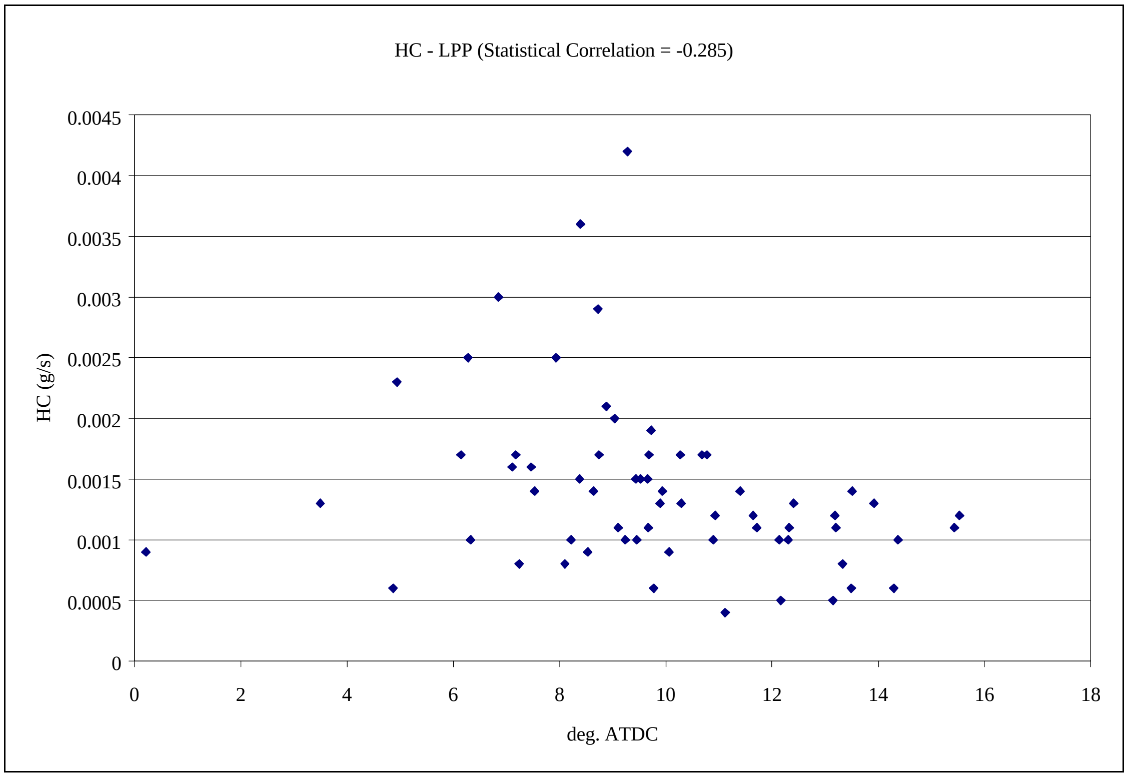
Ward Systems Group, Inc., NeuroShell 2 Manual, Frederick, Maryland, Fourth Edition, 1996.

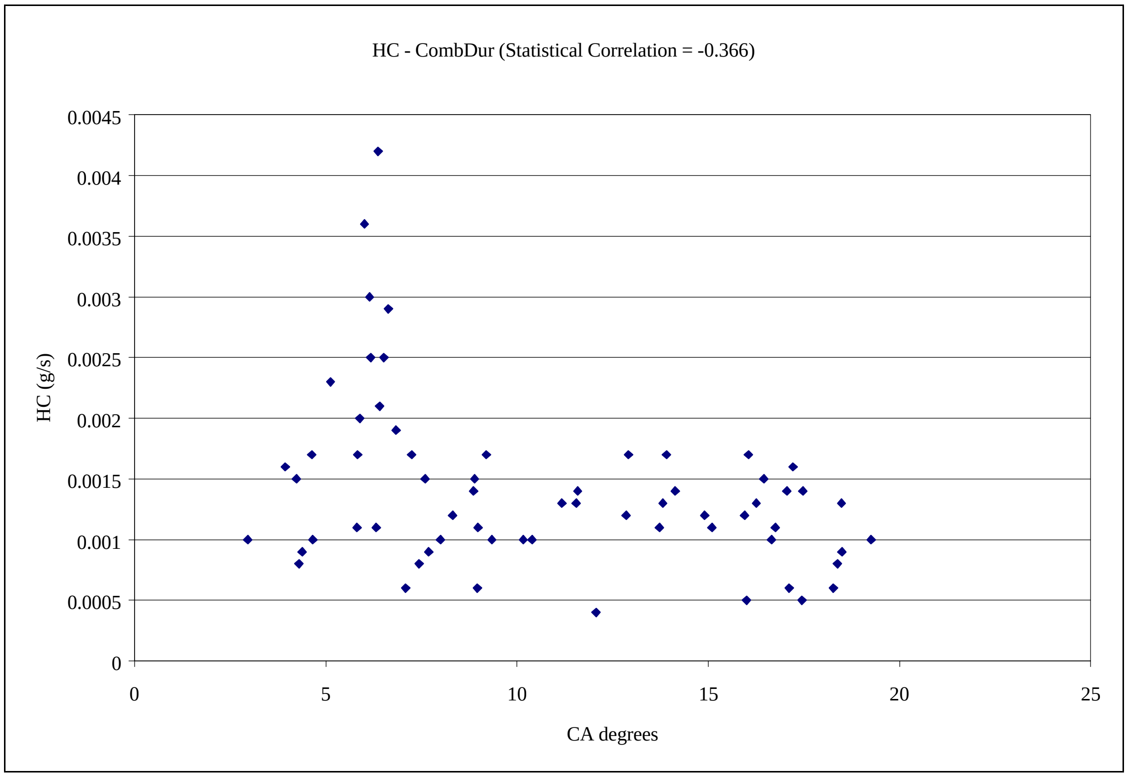
Watanabe, S., Machida, K., Iijima, K. and Tomisawa, N., “A Sophisticated Engine Control System Using Combustion Pressure Detection”, SAE Paper 960042, 1996.

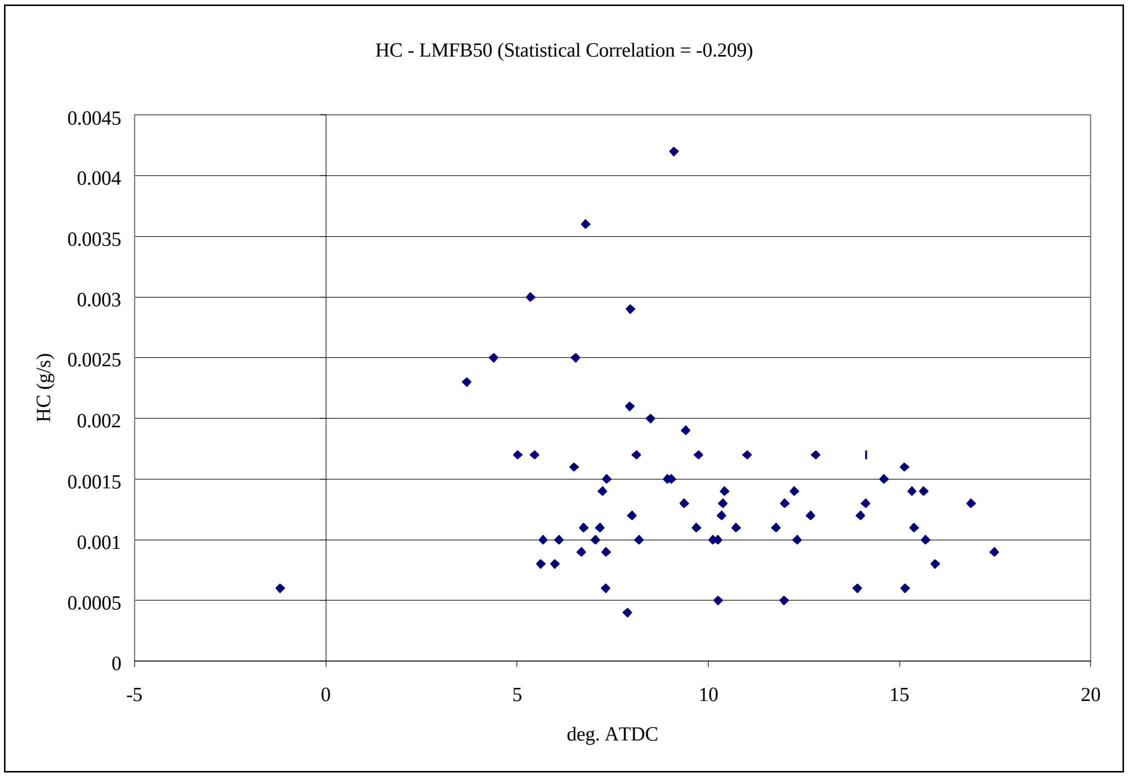
Yang, Y., Smith. J., Celik, I., Lyons, D., and Crawford, B., "Analysis of Emissions from a Four-Cylinder Saturn Engine Using a Combination of Zero- and Multi-Dimensional Modeling", ASME-ICE Fall Technical Conference, Lafayette, IN, 1994.

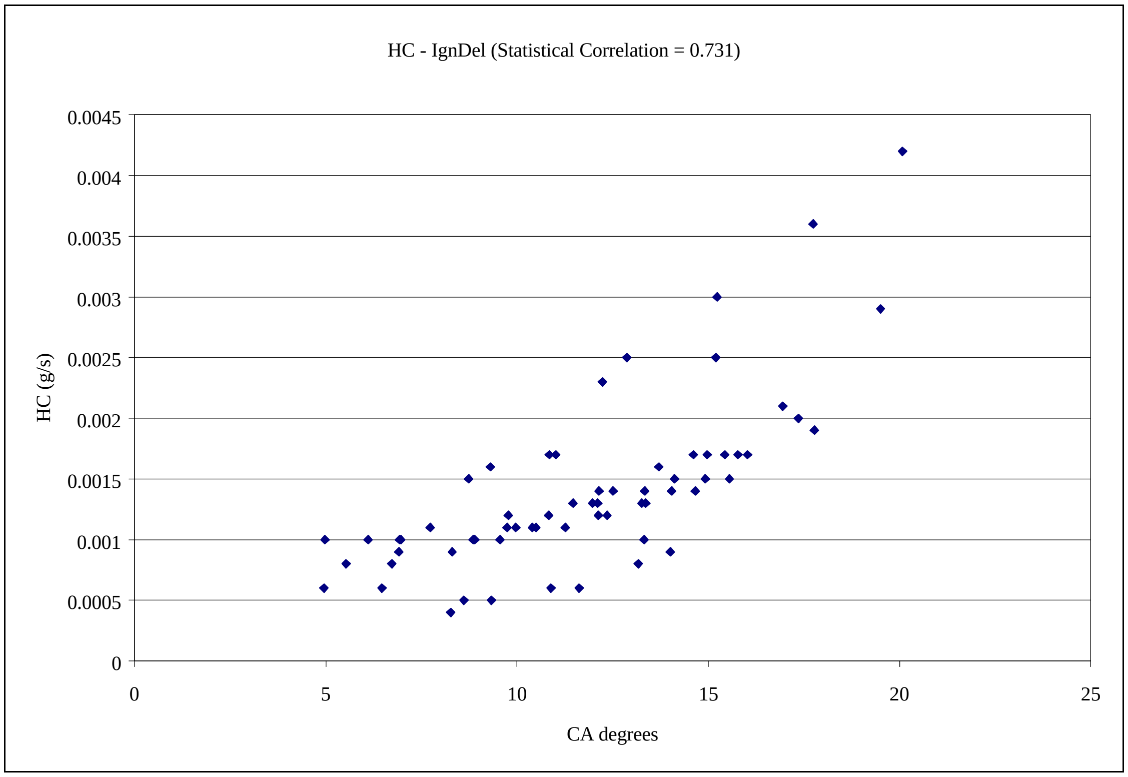
Appendix A: Emissions and In-Cylinder Variable Relationships Charts for Selected Network Inputs with Statistical Correlation

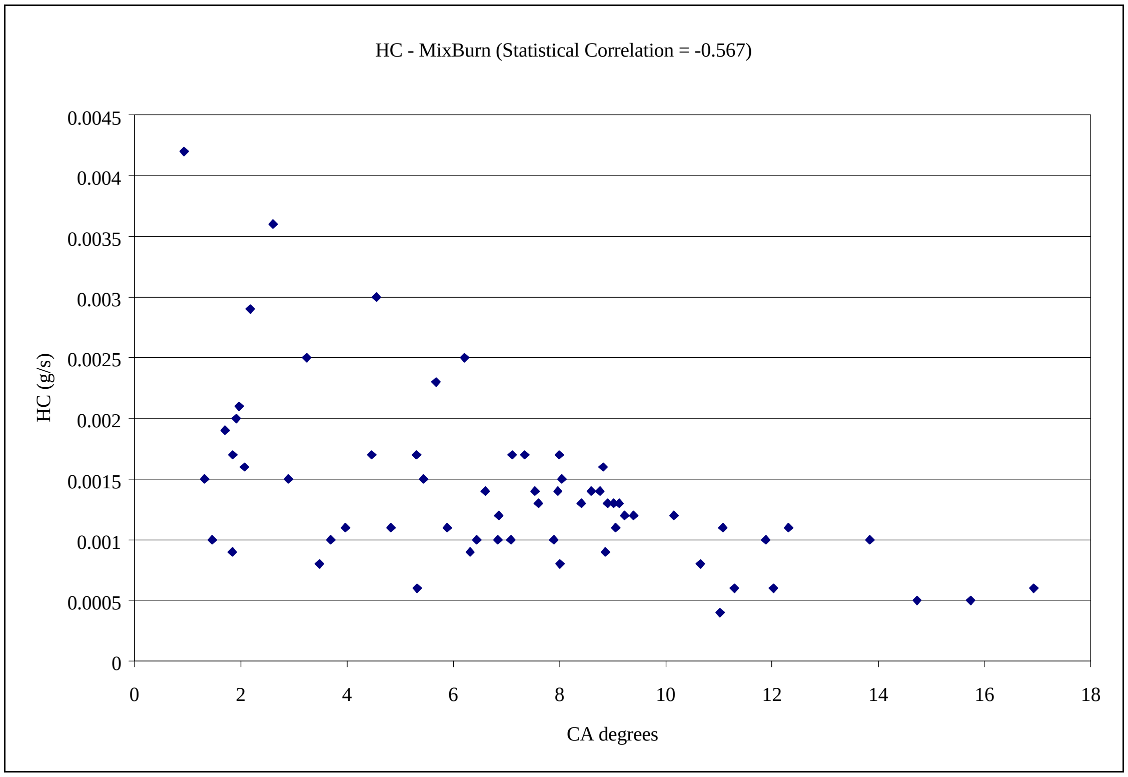

HC - Peak Pressure (Statistical Correlation = -0.498)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
2000
4000
6000
8000
10000
12000
14000
kPa

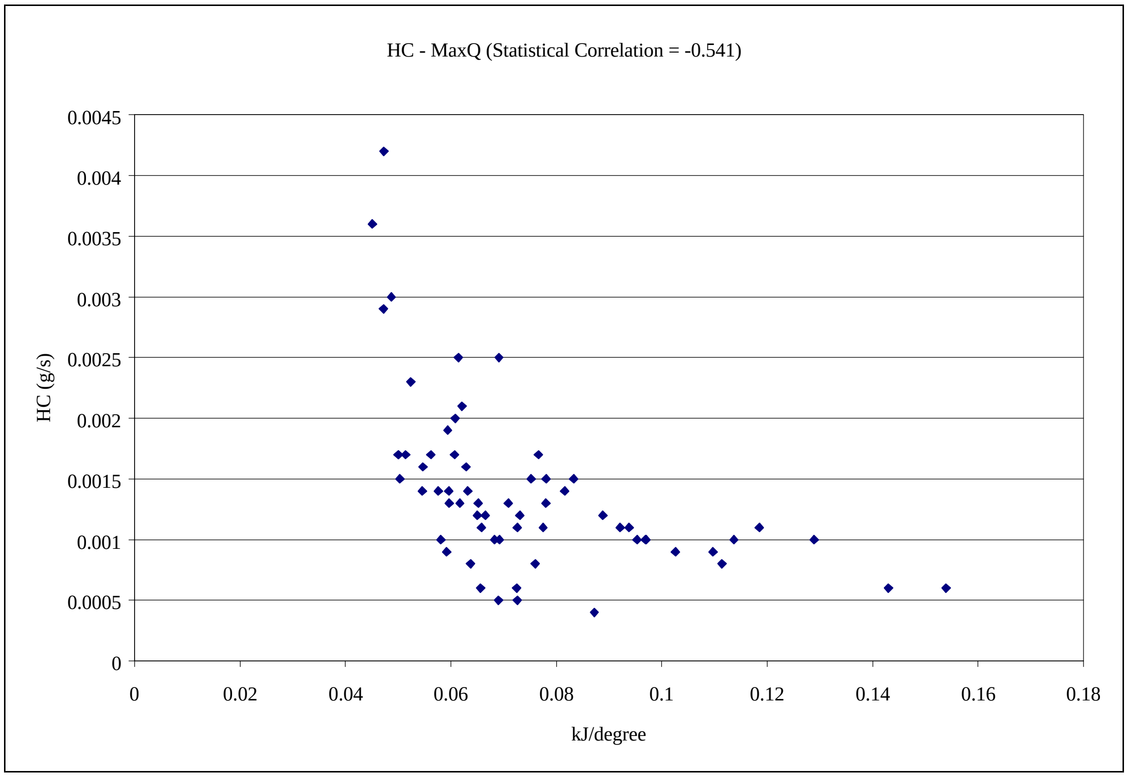

HC - LPP (Statistical Correlation = -0.285)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
2
4
6
8
10
12
14
16
18
deg. ATDC

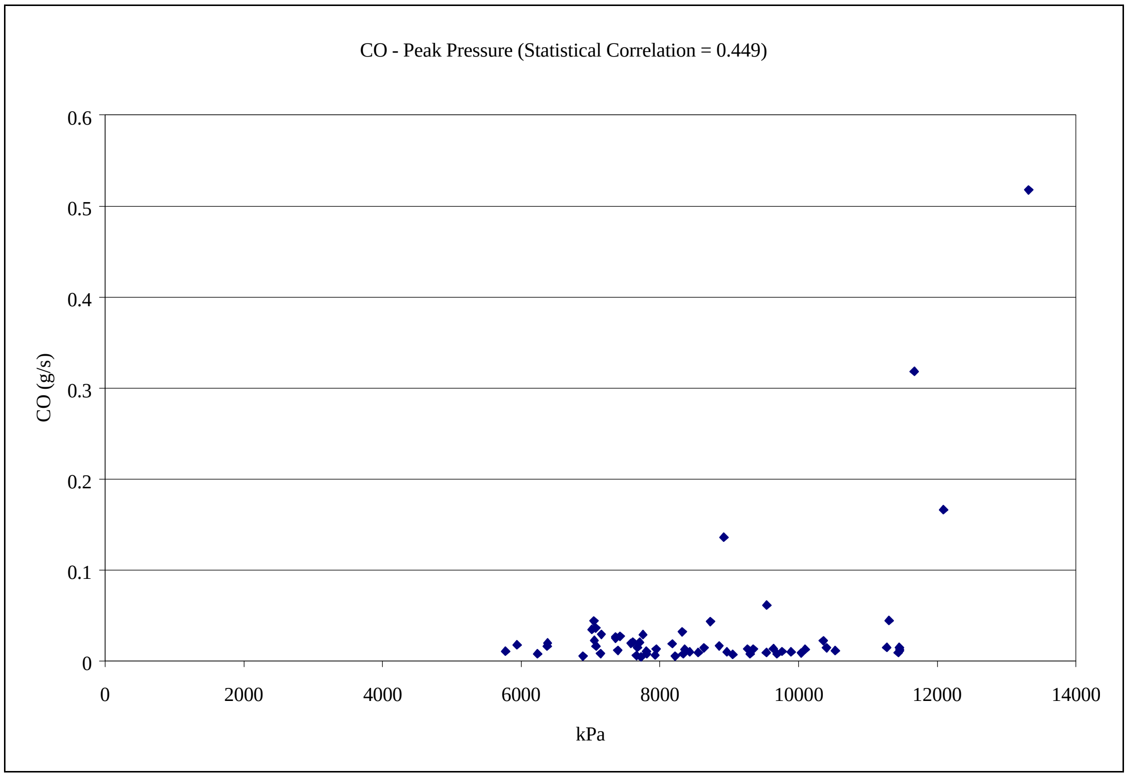

HC - CombDur (Statistical Correlation = -0.366)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
5
10
15
20
25
CA degrees

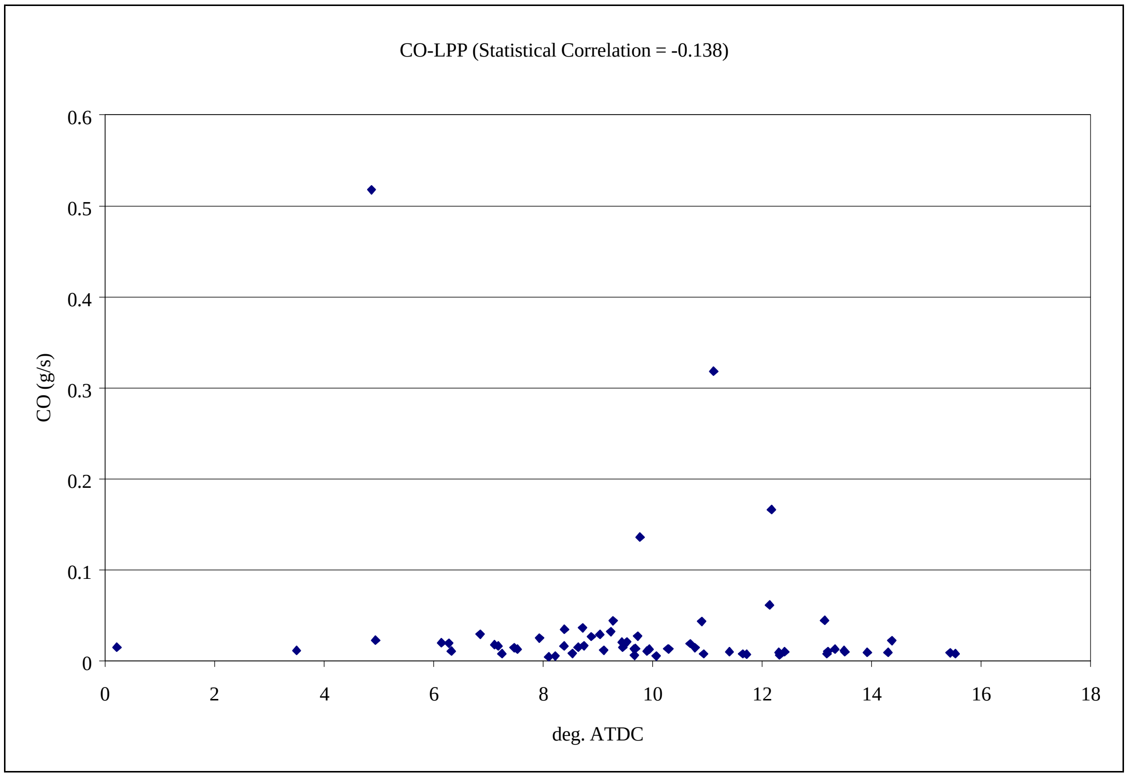

HC - LMFB50 (Statistical Correlation = -0.209)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
-5
0
5
10
15
20
deg. ATDC

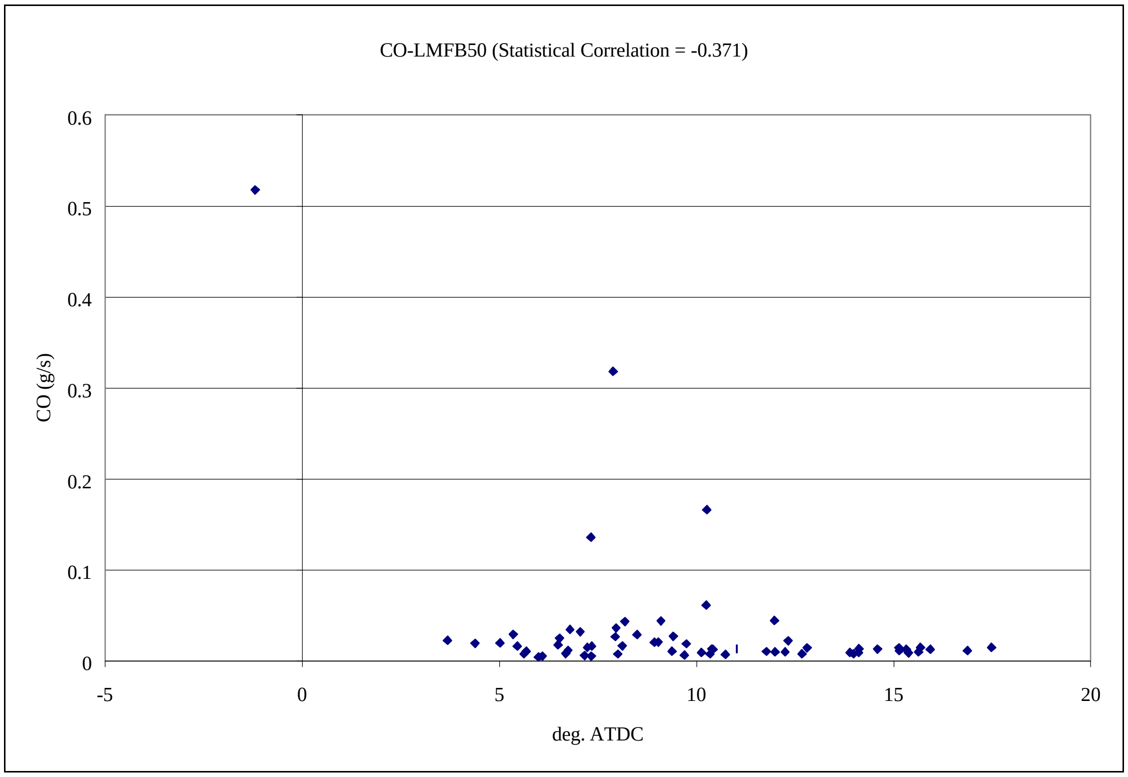

HC - IgnDel (Statistical Correlation = 0.731)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
5
10
15
20
25
CA degrees

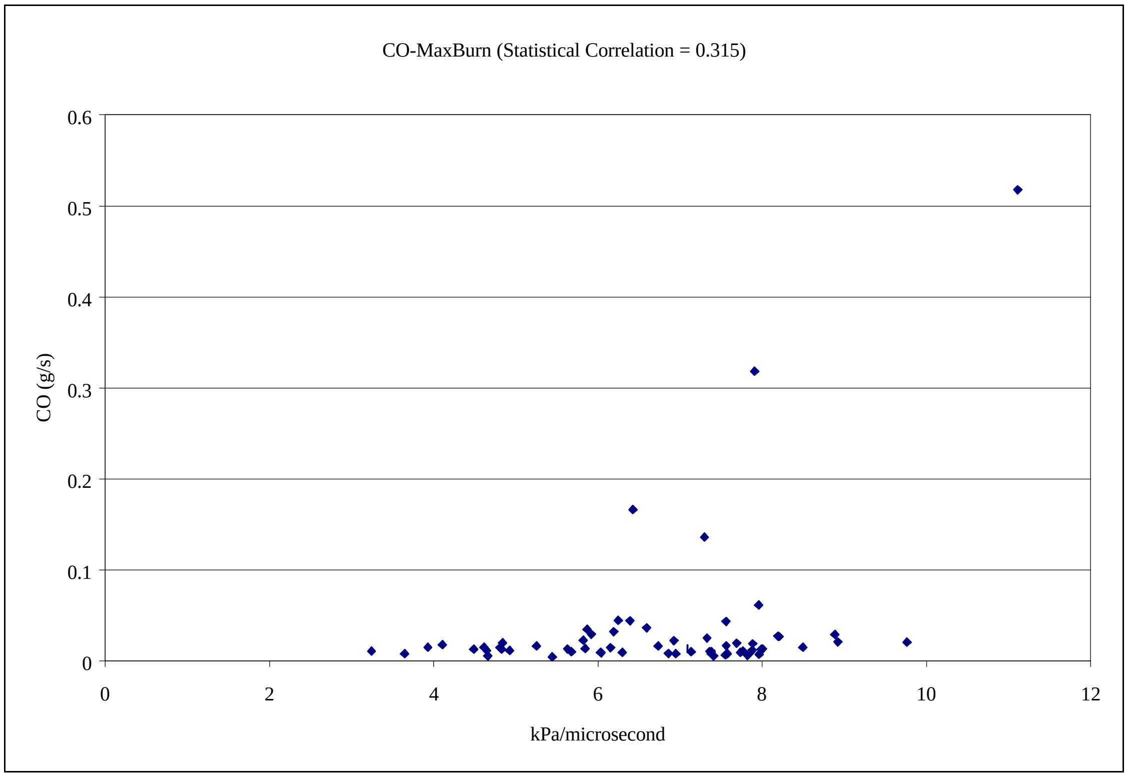

HC - MixBurn (Statistical Correlation = -0.567)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
2
4
6
8
10
12
14
16
18
CA degrees

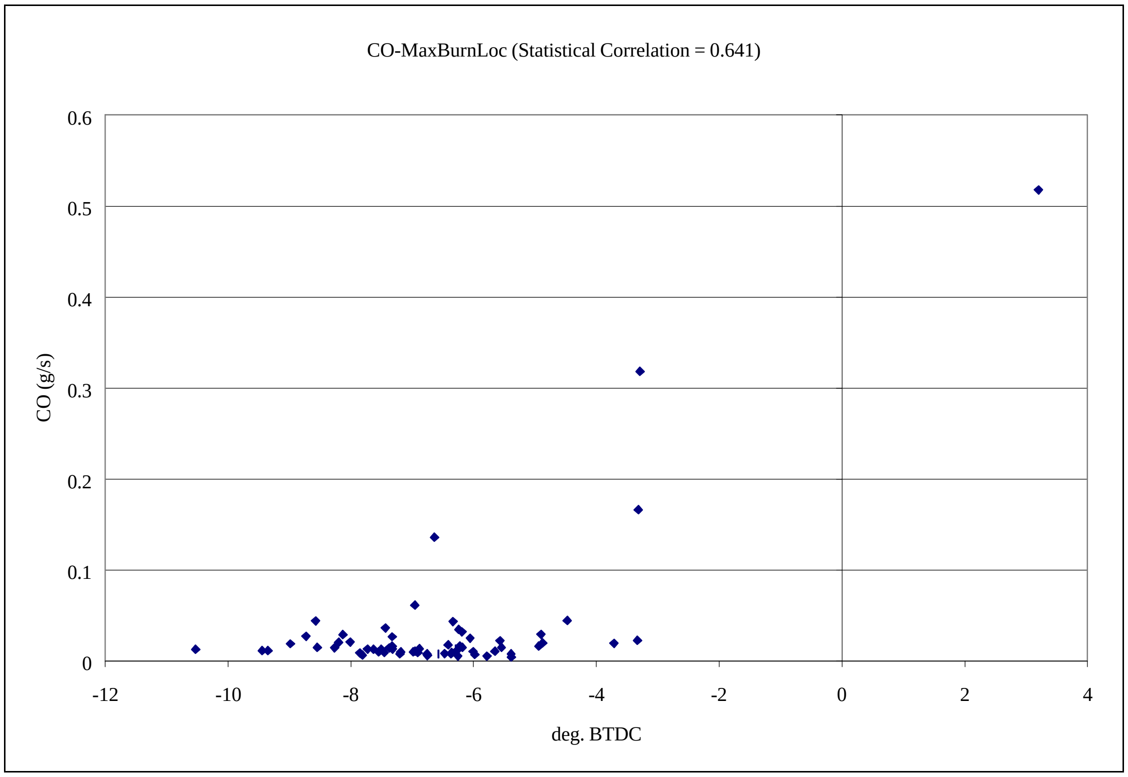

HC - MaxQ (Statistical Correlation = -0.541)
HC (g/s)
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
kJ/degree

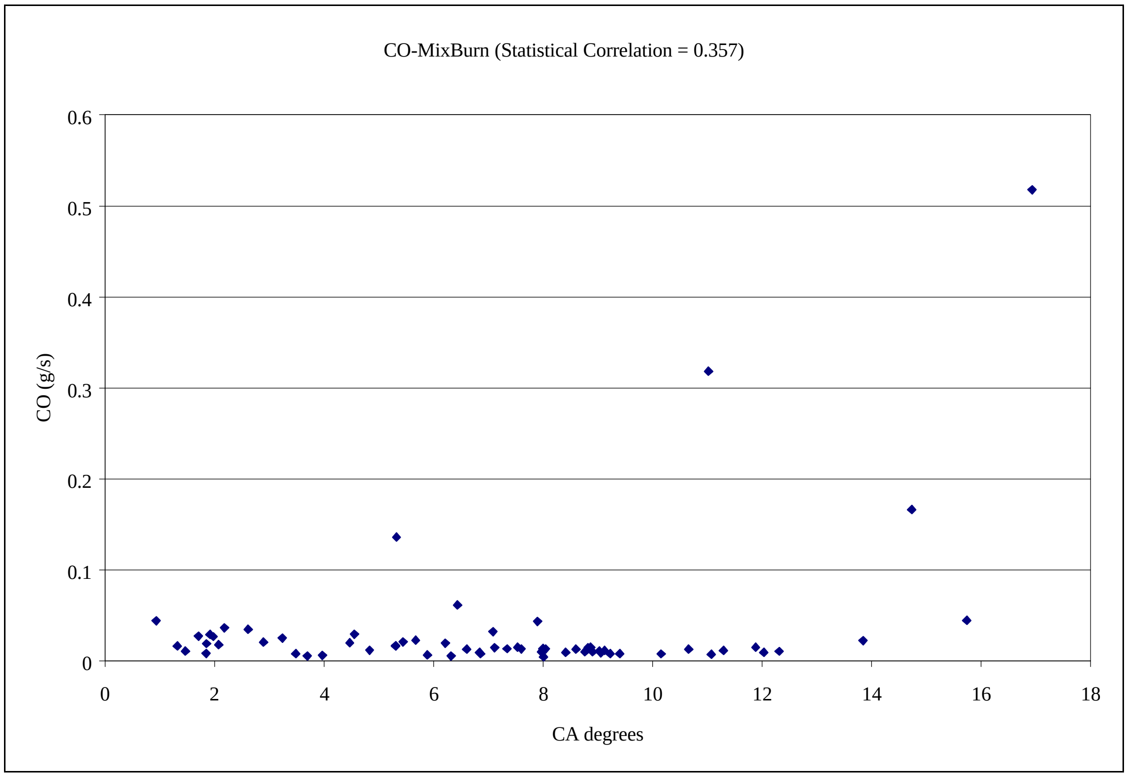

CO - Peak Pressure (Statistical Correlation = 0.449)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
12000
14000
kPa

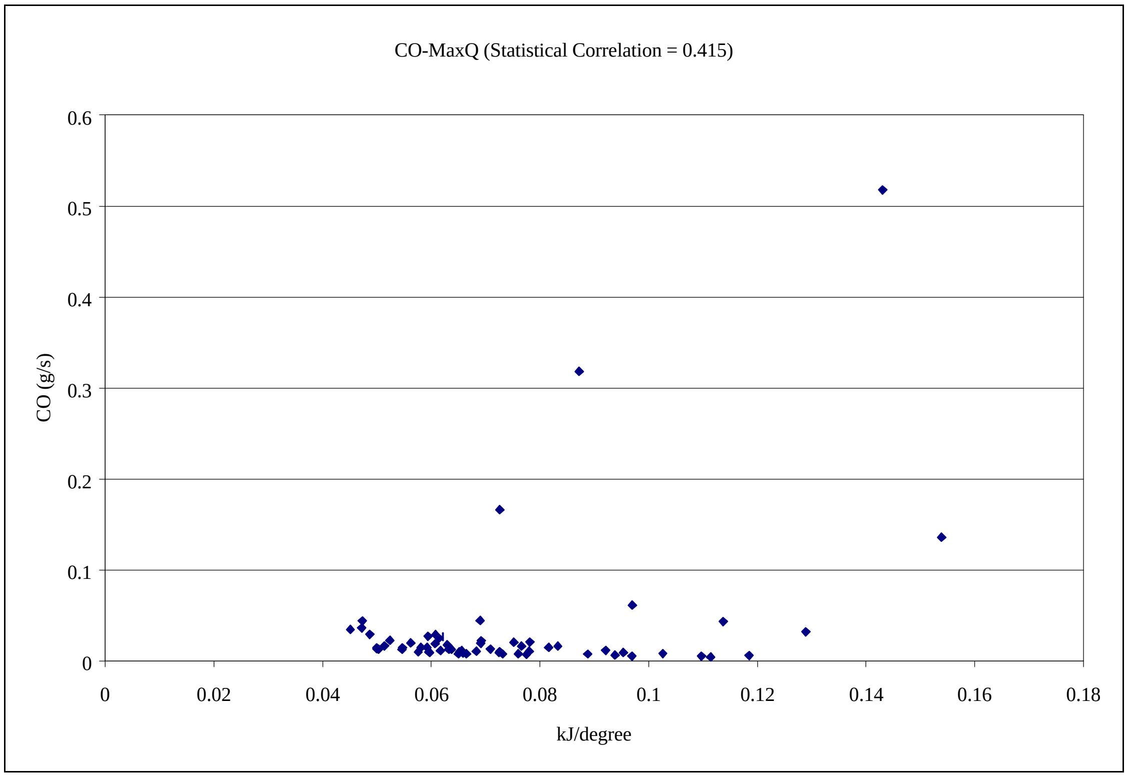

CO-LPP (Statistical Correlation = -0.138)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
14
16
18
deg. ATDC

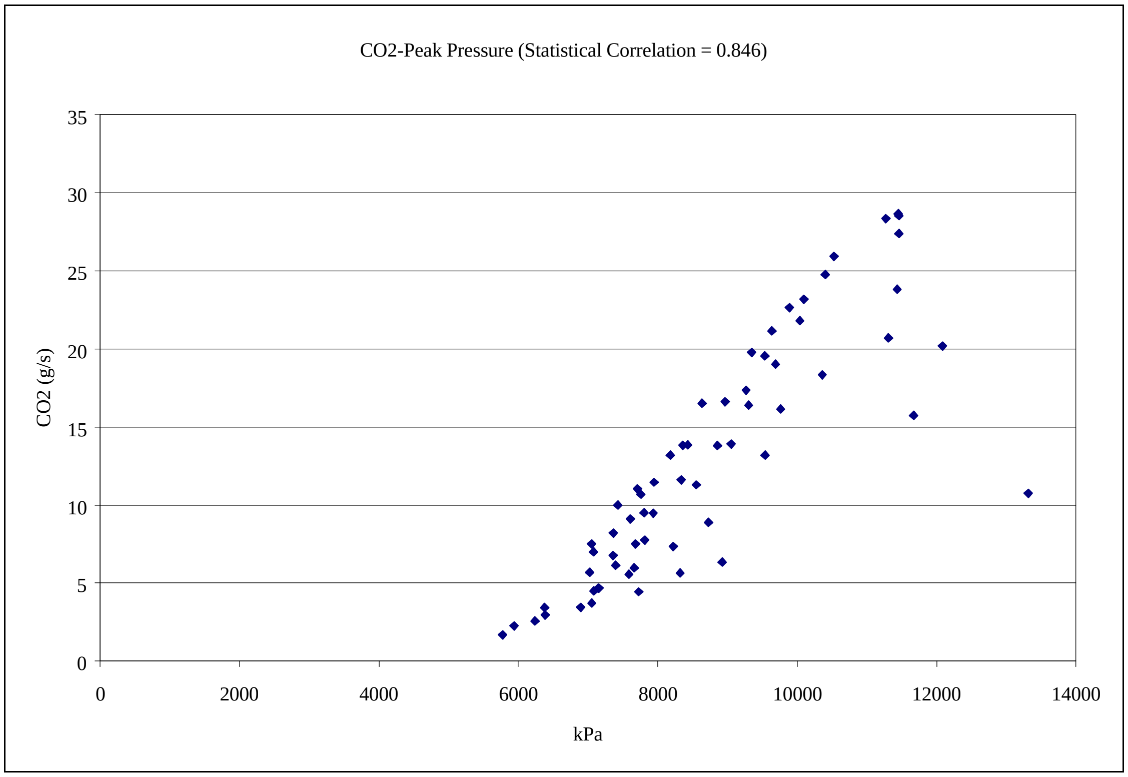

CO-LMFB50 (Statistical Correlation = -0.371)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
-5
0
5
10
15
20
deg. ATDC

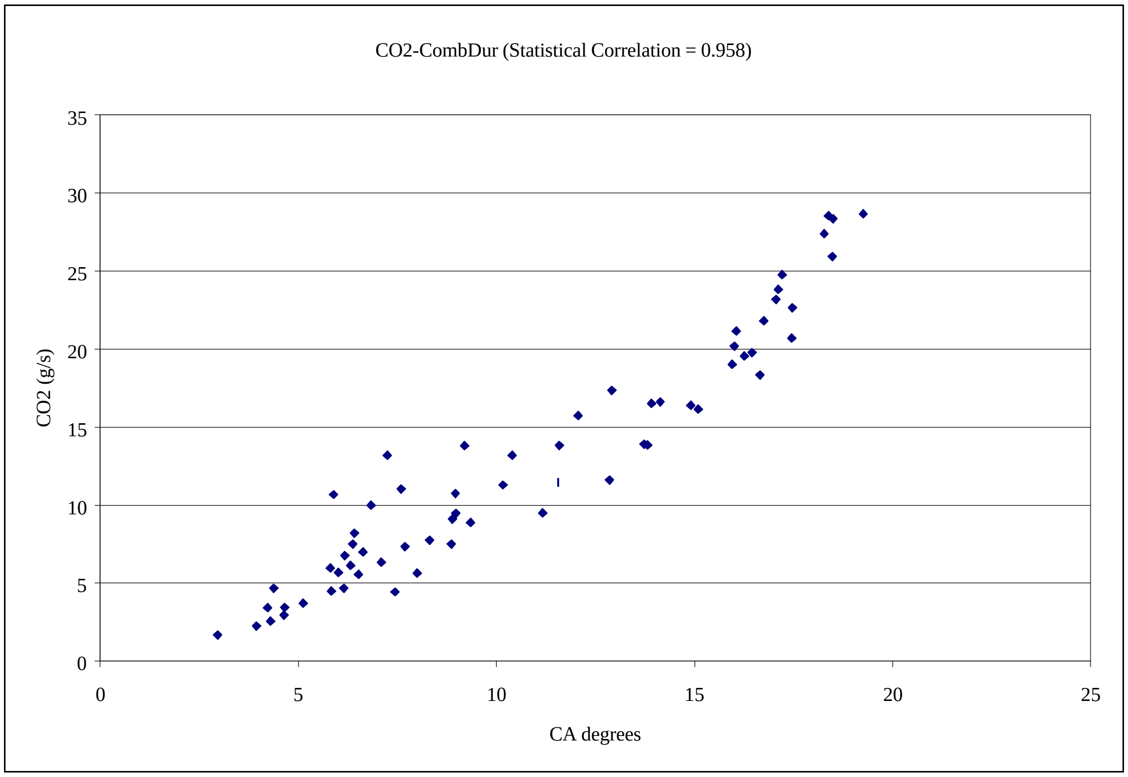

CO-MaxBurn (Statistical Correlation = 0.315)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
kPa/microsecond

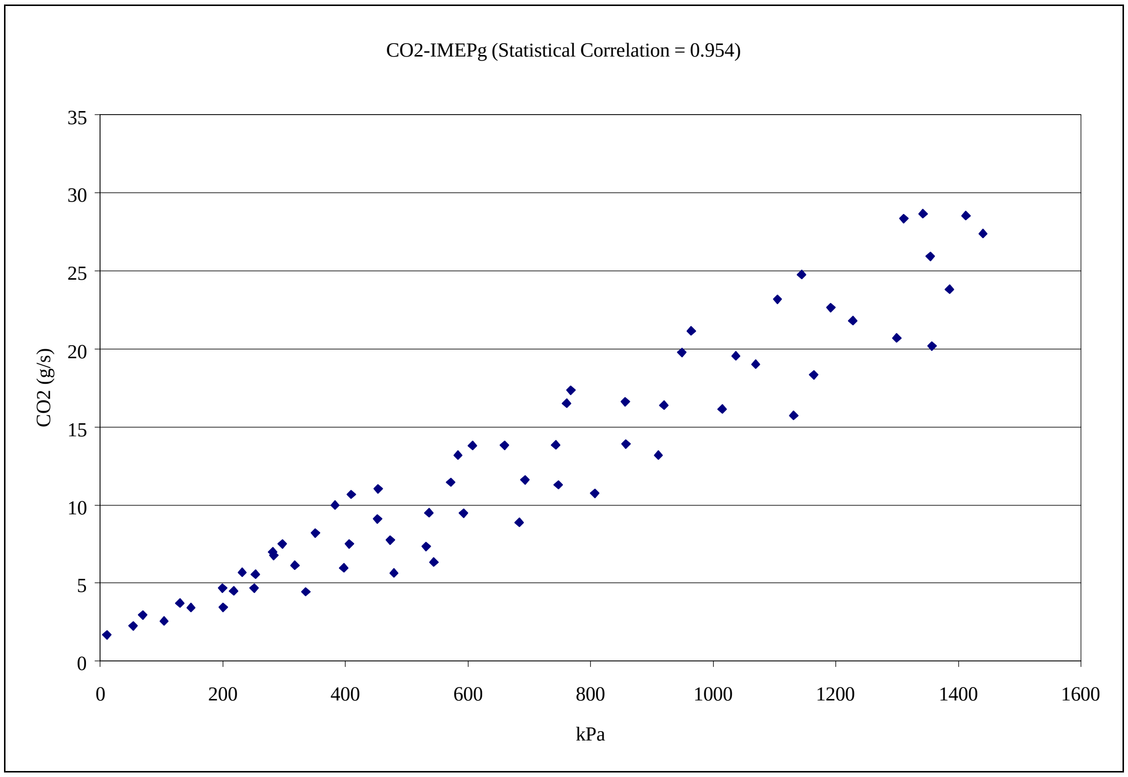

CO-MaxBurnLoc (Statistical Correlation = 0.641)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
-12
-10
-8
-6
-4
-2
0
2
4
deg. BTDC

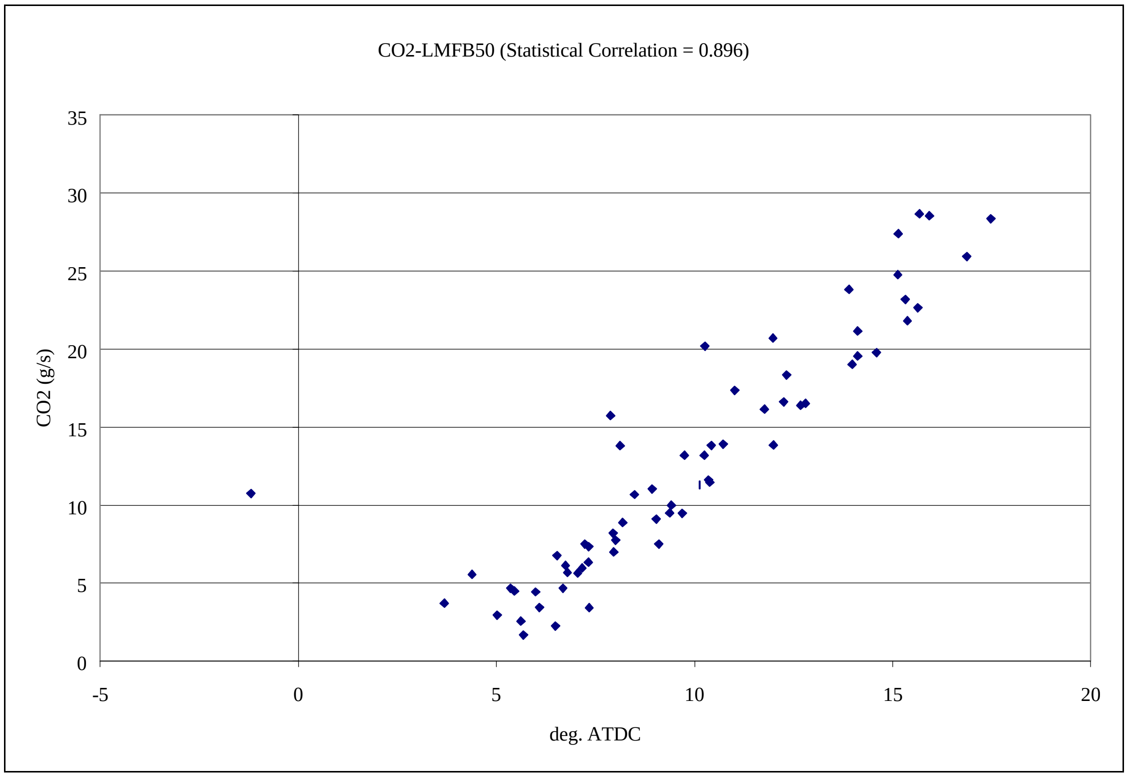

CO-MixBurn (Statistical Correlation = 0.357)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
14
16
18
CA degrees

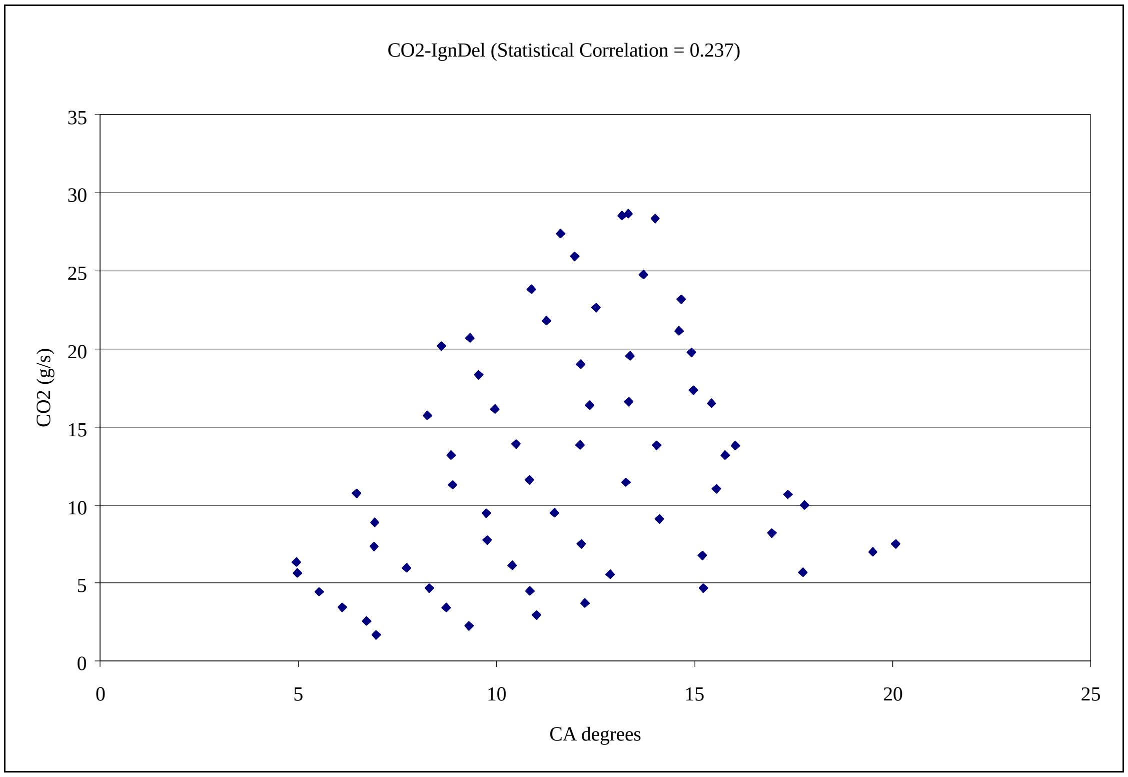

CO-MaxQ (Statistical Correlation = 0.415)
CO (g/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
kJ/degree

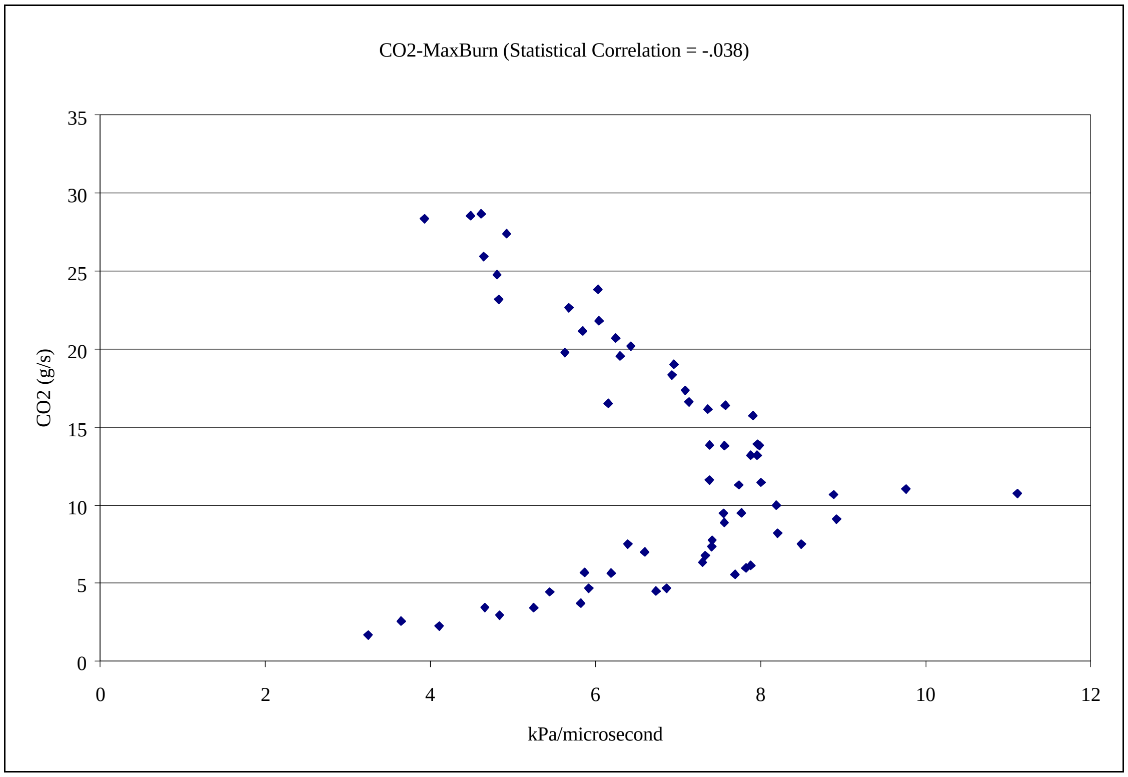

CO2-Peak Pressure (Statistical Correlation = 0.846)
CO2 (g/s)
0
5
10
15
20
25
30
35
0
2000
4000
6000
8000
10000
12000
14000
kPa

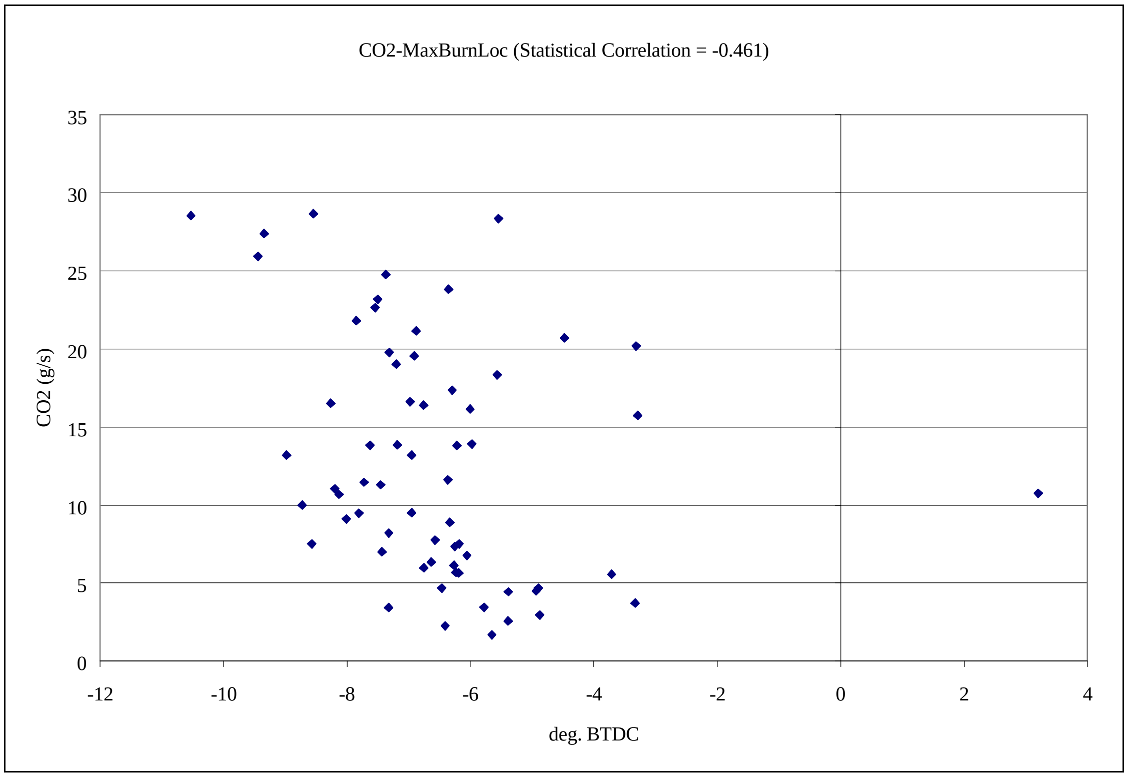

CO2-CombDur (Statistical Correlation = 0.958)
CO2 (g/s)
0
5
10
15
20
25
30
35
0
5
10
15
20
25
CA degrees

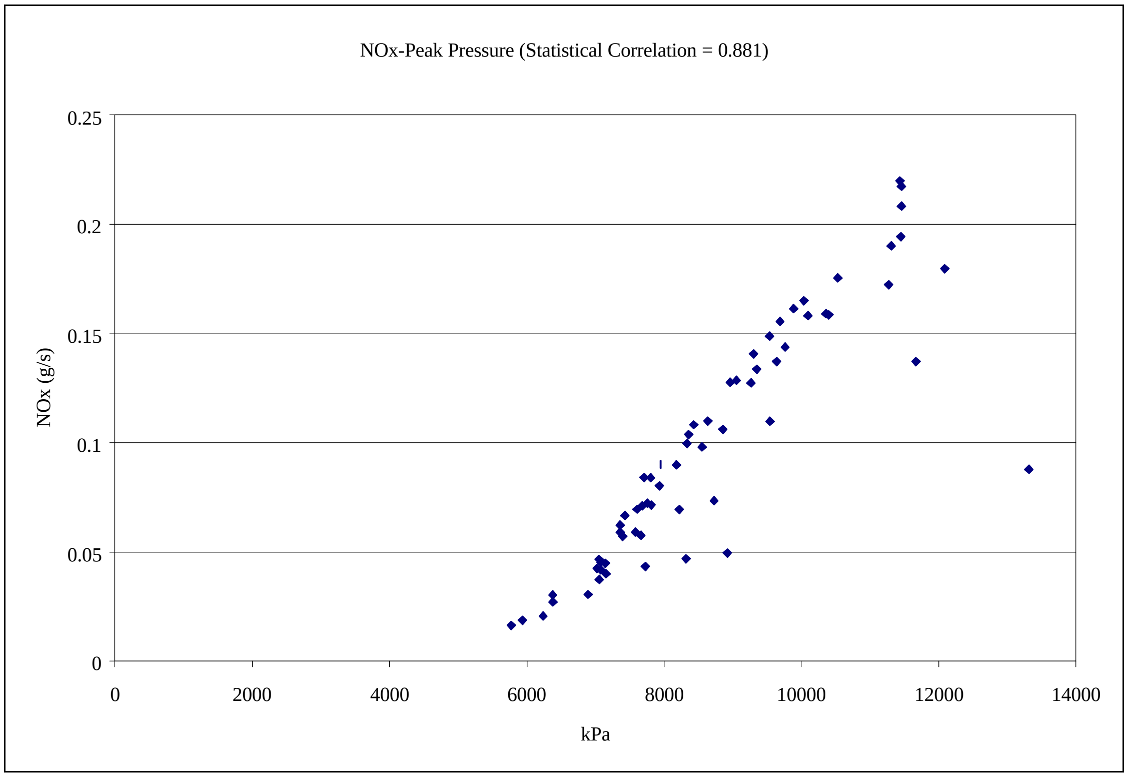

CO2-IMEPg (Statistical Correlation = 0.954)
CO2 (g/s)
kPa
0
5
10
15
20
25
30
35
0
200
400
600
800
1000
1200
1400
1600

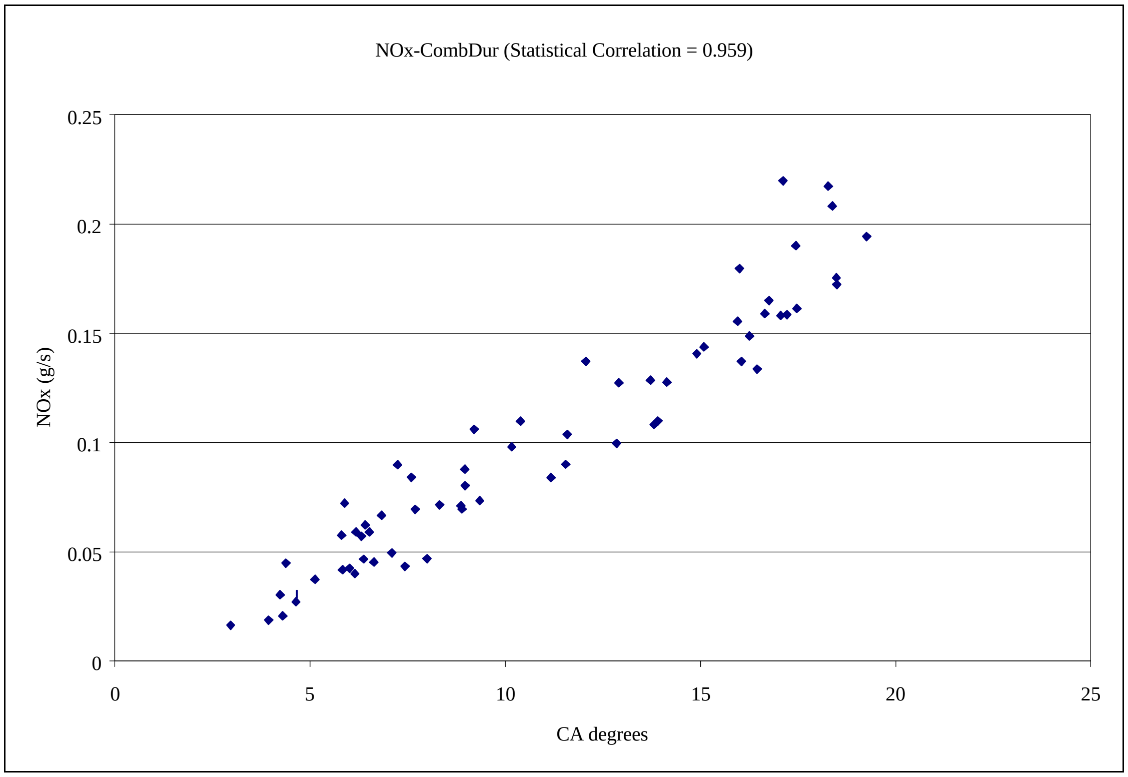

CO2-LMFB50 (Statistical Correlation = 0.896)
CO2 (g/s)
deg. ATDC
0
5
10
15
20
25
30
35
-5
0
5
10
15
20

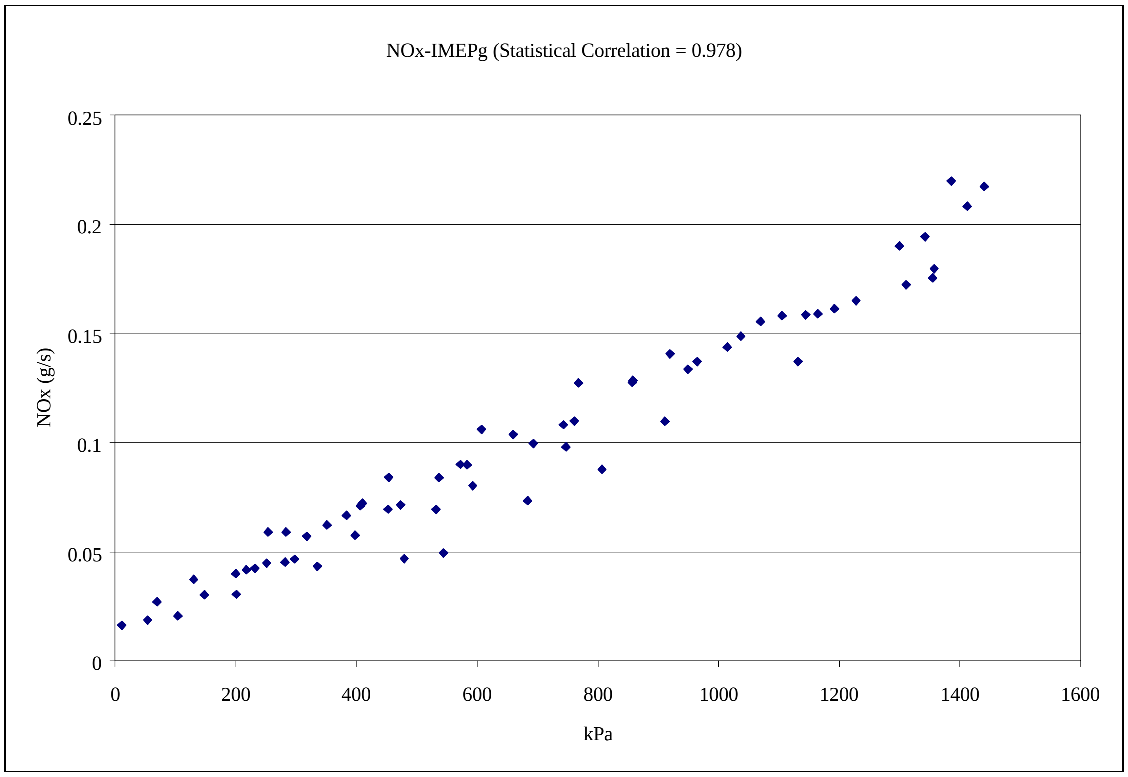

CO2-IgnDel (Statistical Correlation = 0.237)
CO2 (g/s)
35
30
25
20
15
10
5
0
0
5
10
15
20
25
CA degrees

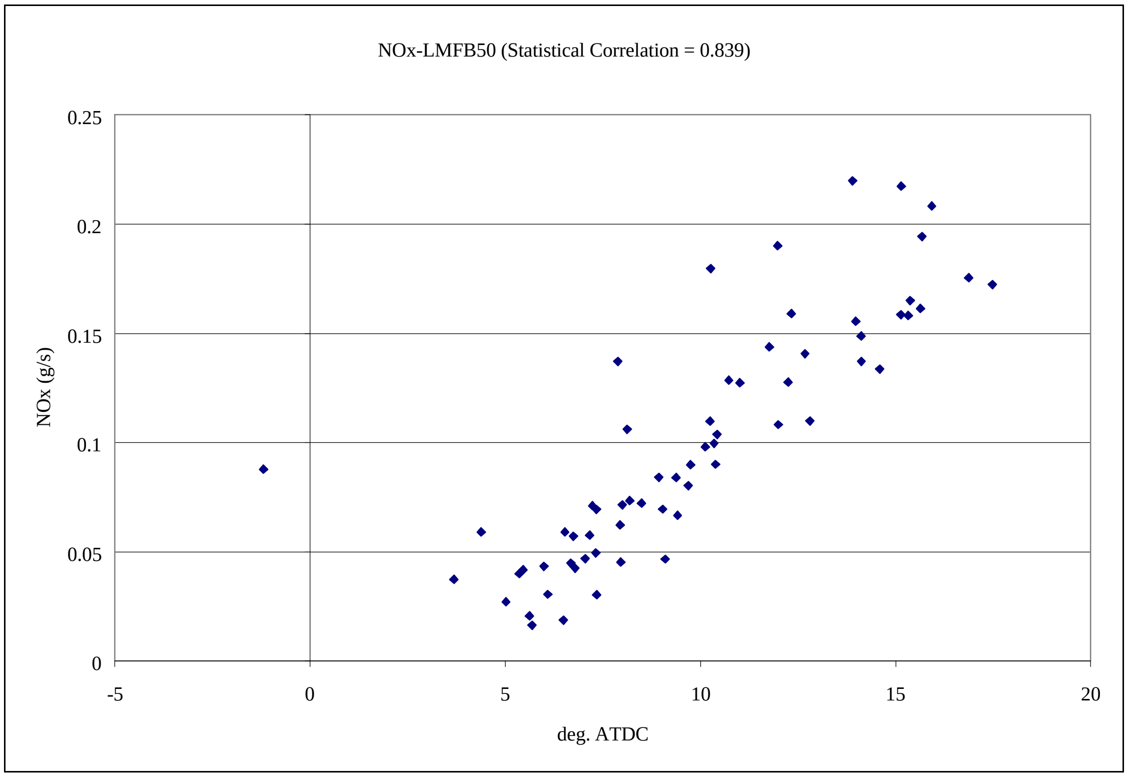

CO2-MaxBurn (Statistical Correlation = -.038)
CO2 (g/s)
35
30
25
20
15
10
5
0
0
2
4
6
8
10
12
kPa/microsecond

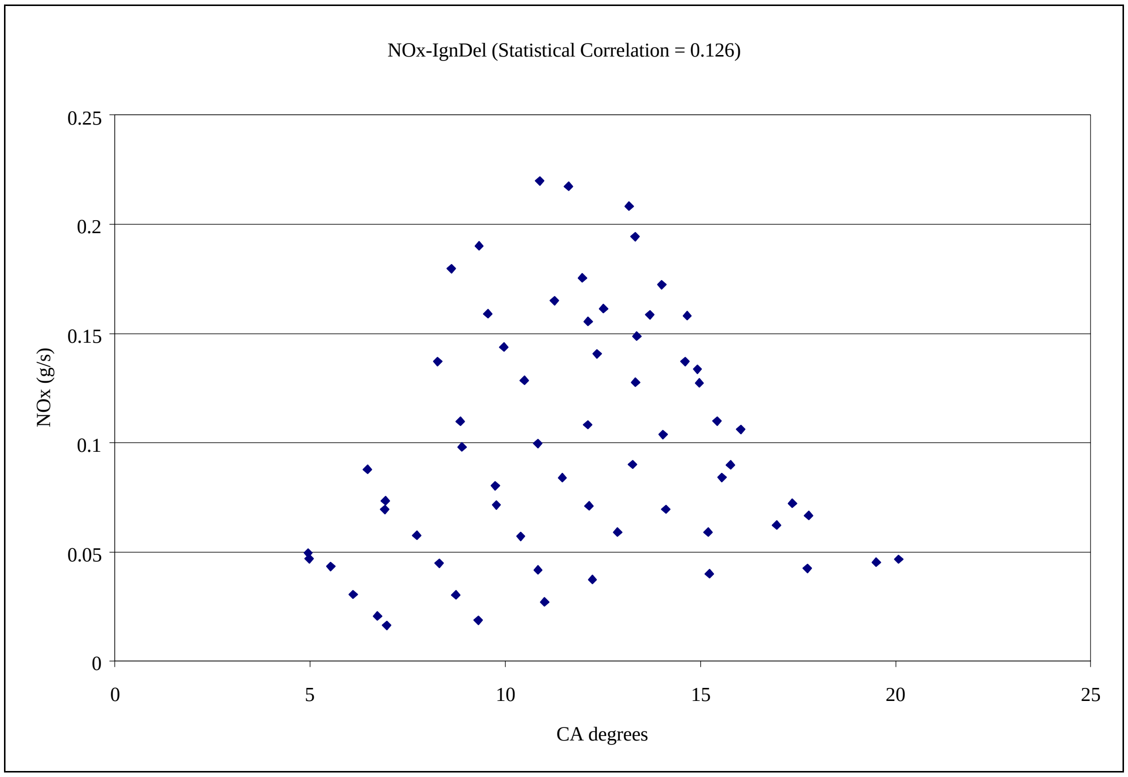

CO2-MaxBurnLoc (Statistical Correlation = -0.461)
CO2 (g/s)
deg. BTDC
35
30
25
20
15
10
5
0
-12
-10
-8
-6
-4
-2
0
2
4

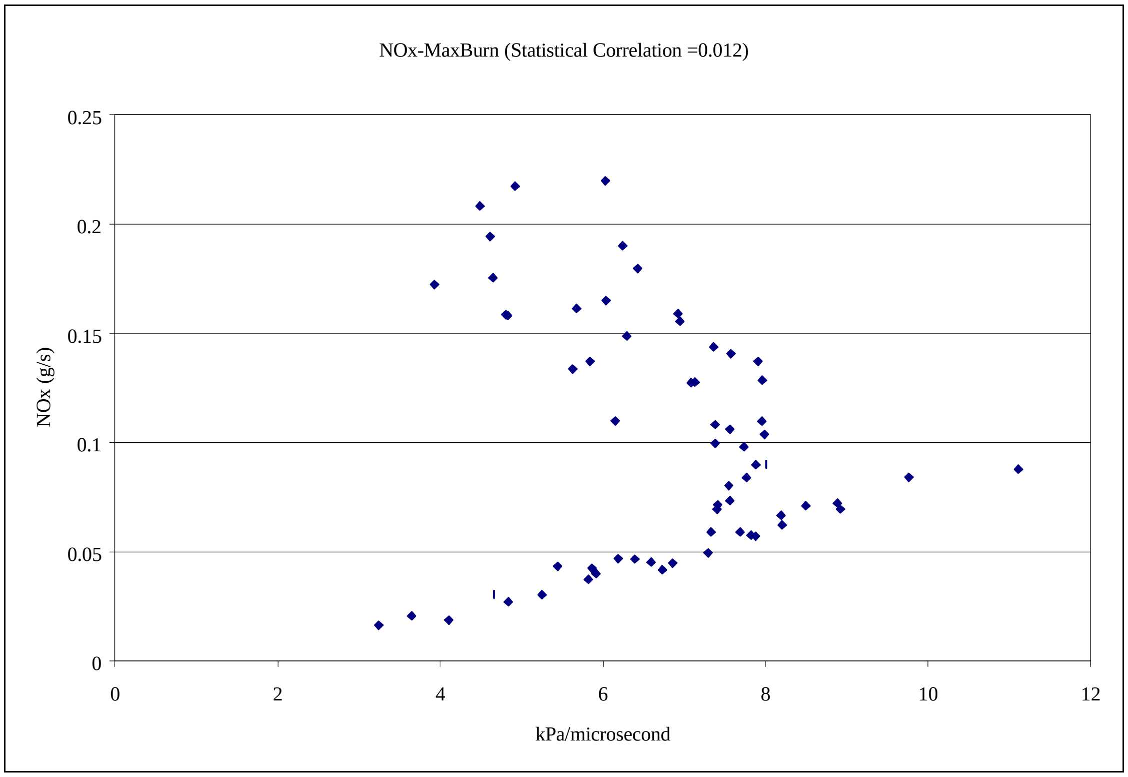

NOx-Peak Pressure (Statistical Correlation = 0.881)
NOx (g/s)
kPa
0.25
0.2
0.15
0.1
0.05
0
0
2000
4000
6000
8000
10000
12000
14000

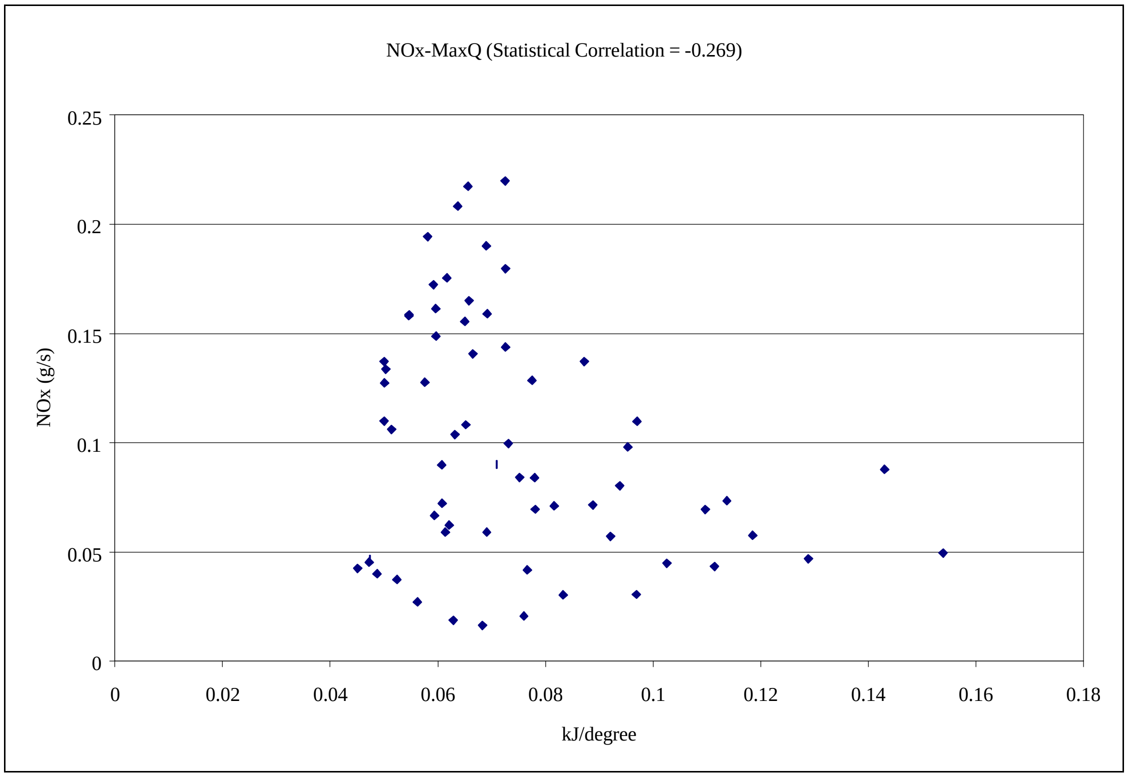

NOx-CombDur (Statistical Correlation = 0.959)
NOx (g/s)
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
CA degrees


NOx-IMEPg (Statistical Correlation = 0.978)
NOx (g/s)
0.25
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000
1200
1400
1600
kPa

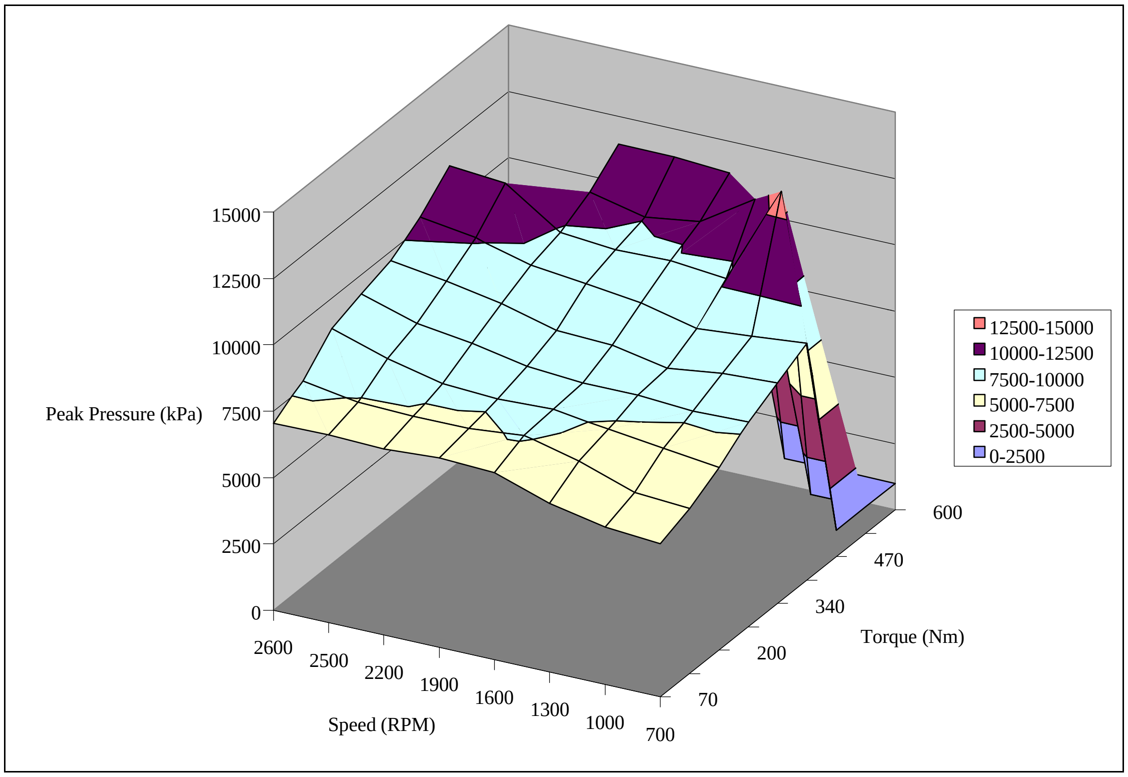

NOx-LMFB50 (Statistical Correlation = 0.839)
0.25
0.2
0.15
0.1
0.05
0
NOx (g/s)
-5
0
5
10
15
20
deg. ATDC

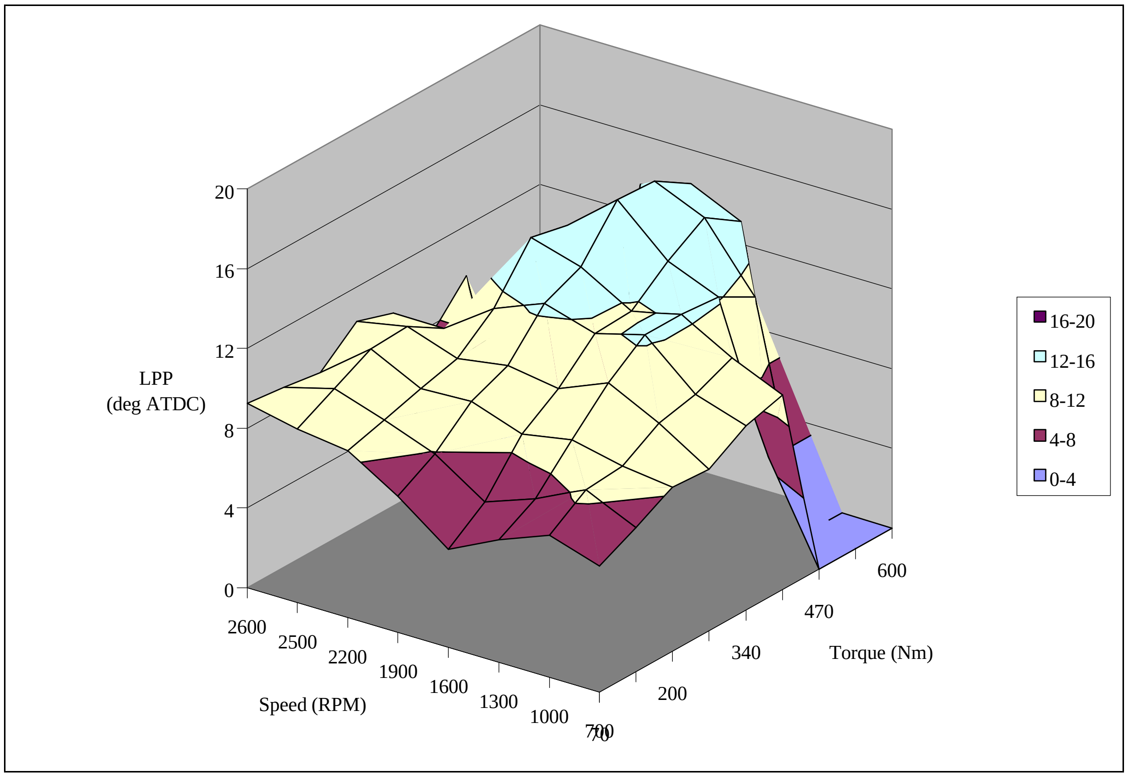

NOx-IgnDel (Statistical Correlation = 0.126)
0.25
0.2
0.15
0.1
0.05
0
NOx (g/s)
0
5
10
15
20
25
CA degrees

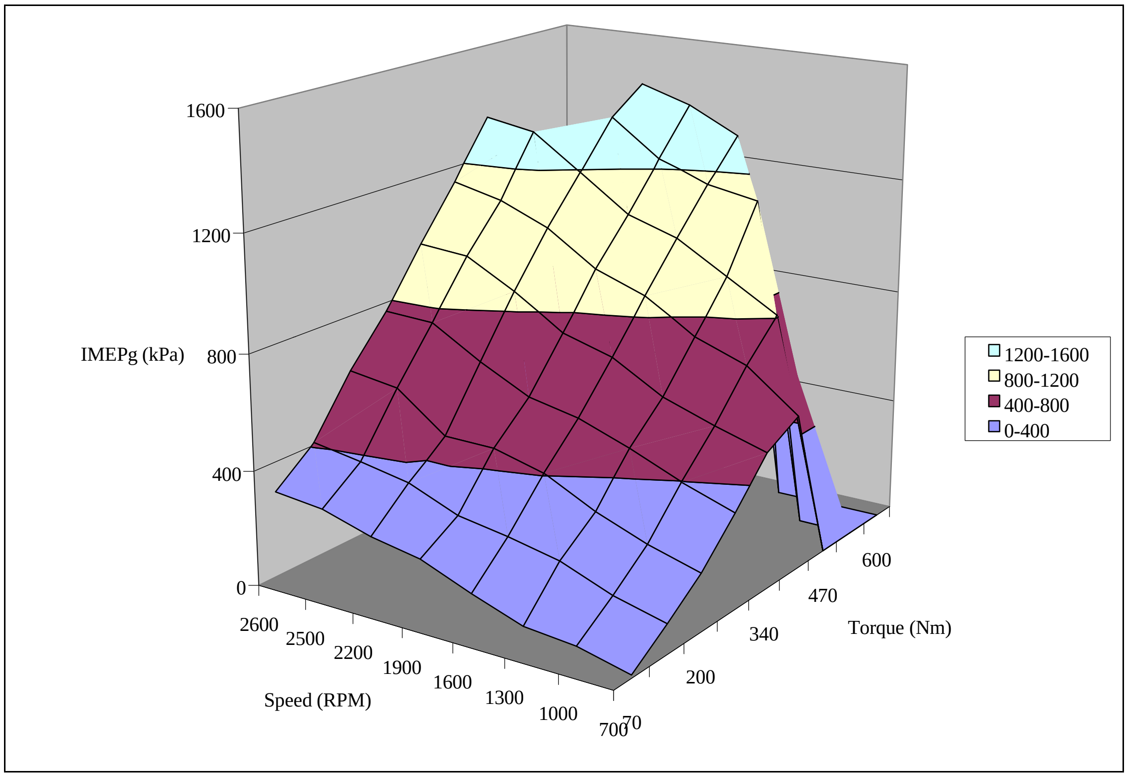

NOx-MaxBurn (Statistical Correlation =0.012)
NOx (g/s)
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
12
kPa/microsecond

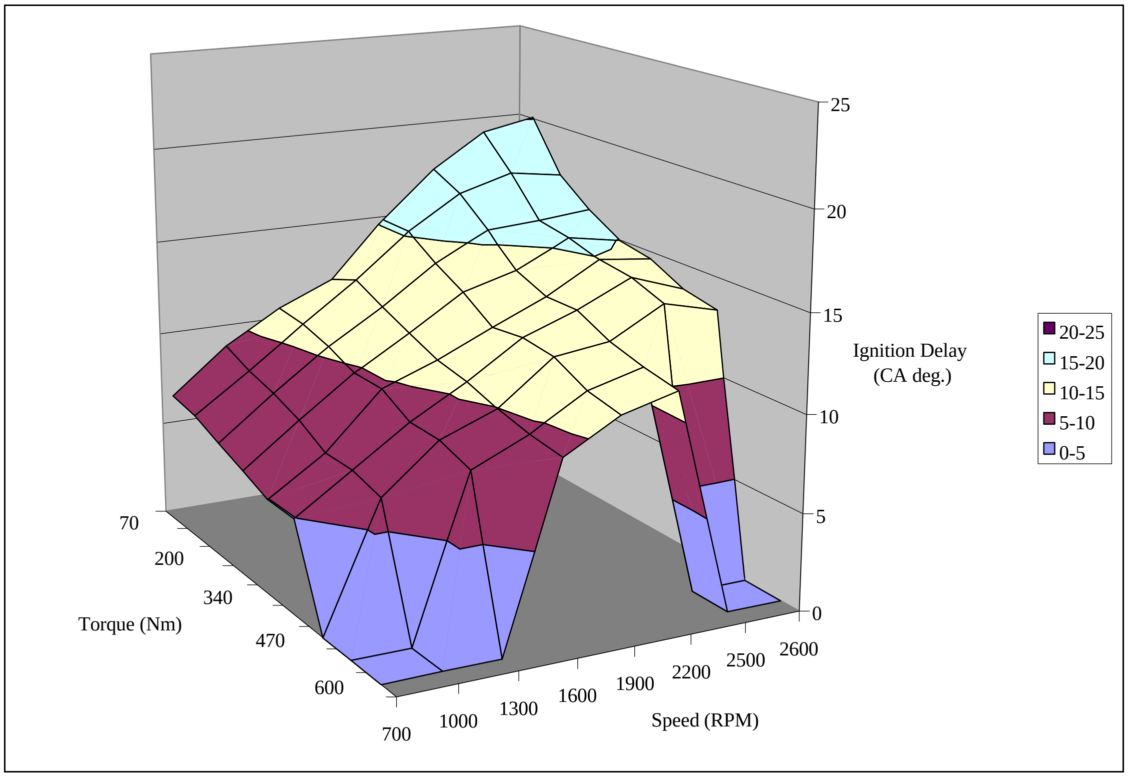

NOx-MaxQ (Statistical Correlation = -0.269)
NOx (g/s)
0.25
0.2
0.15
0.1
0.05
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
kJ/degree

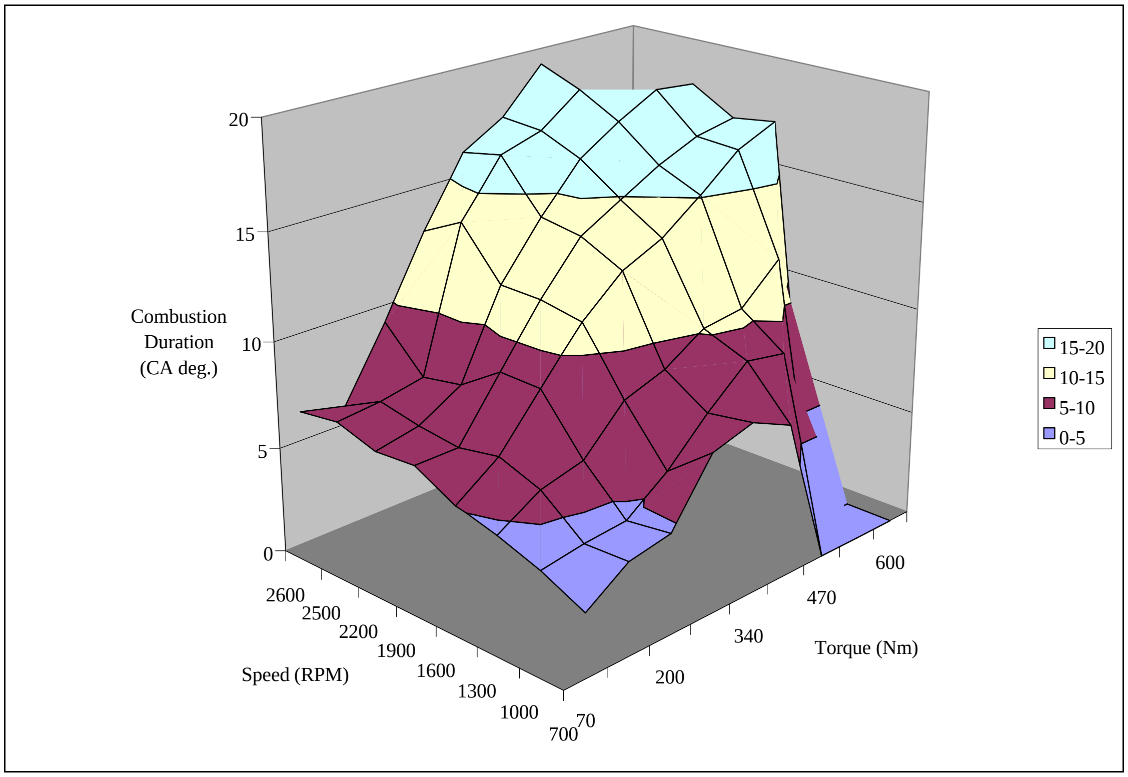
Appendix B: 3-D Charts Showing Uniqueness of the 64 Point Data Grid

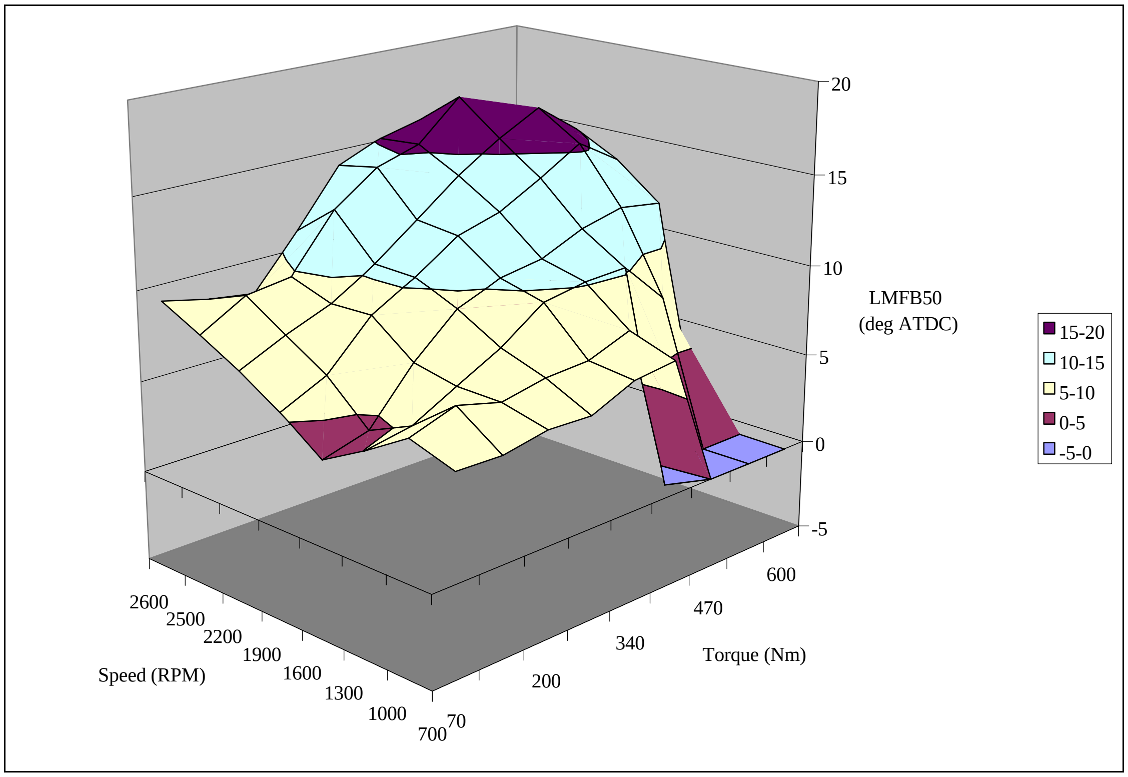

Peak Pressure (kPa)
15000
12500
10000
7500
5000
2500
0
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)
12500-15000
10000-12500
7500-10000
5000-7500
2500-5000
0-2500

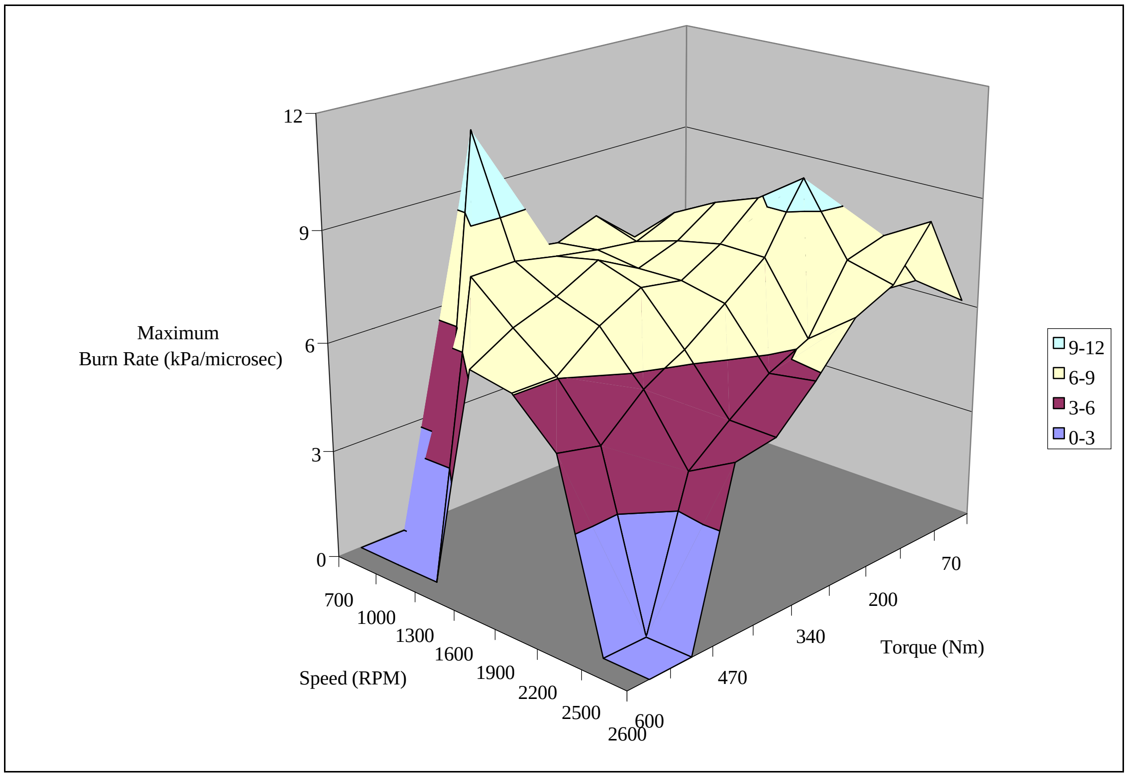

LPP
(deg ATDC)
20
16
12
8
4
0
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)
16-20
12-16
8-12
4-8
0-4

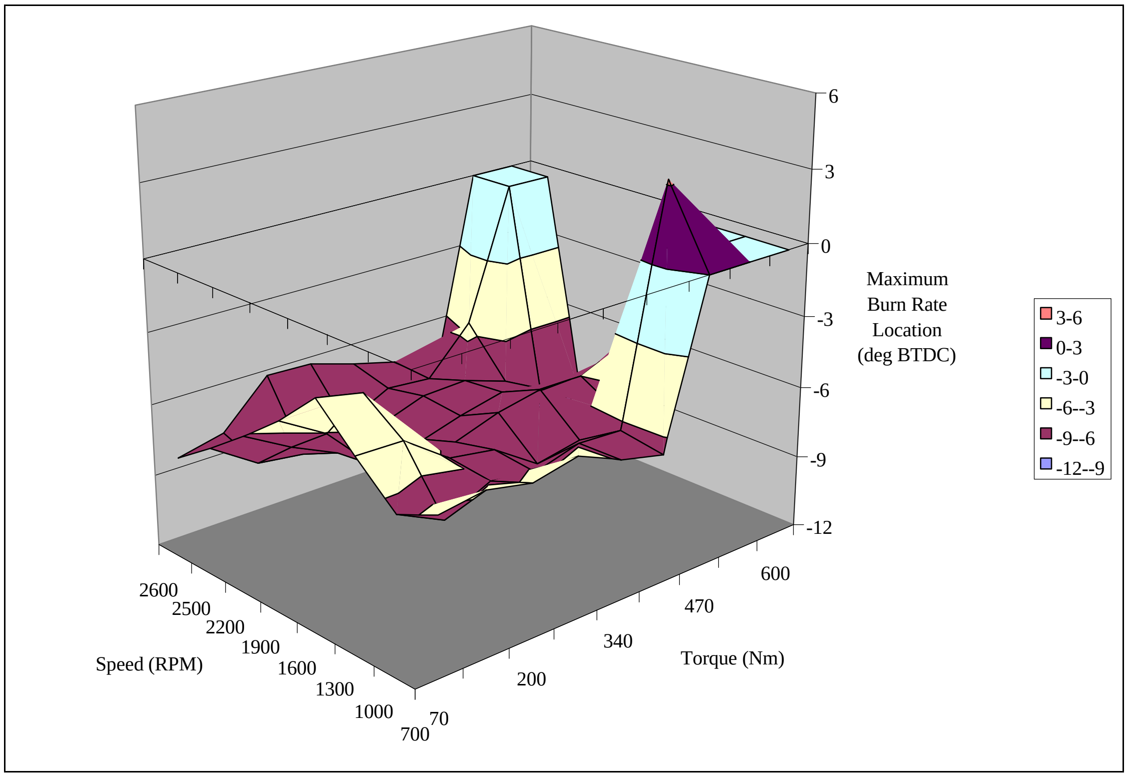

1600
1200
800
400
0
IMEPg (kPa)
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)
1200-1600
800-1200
400-800
0-400

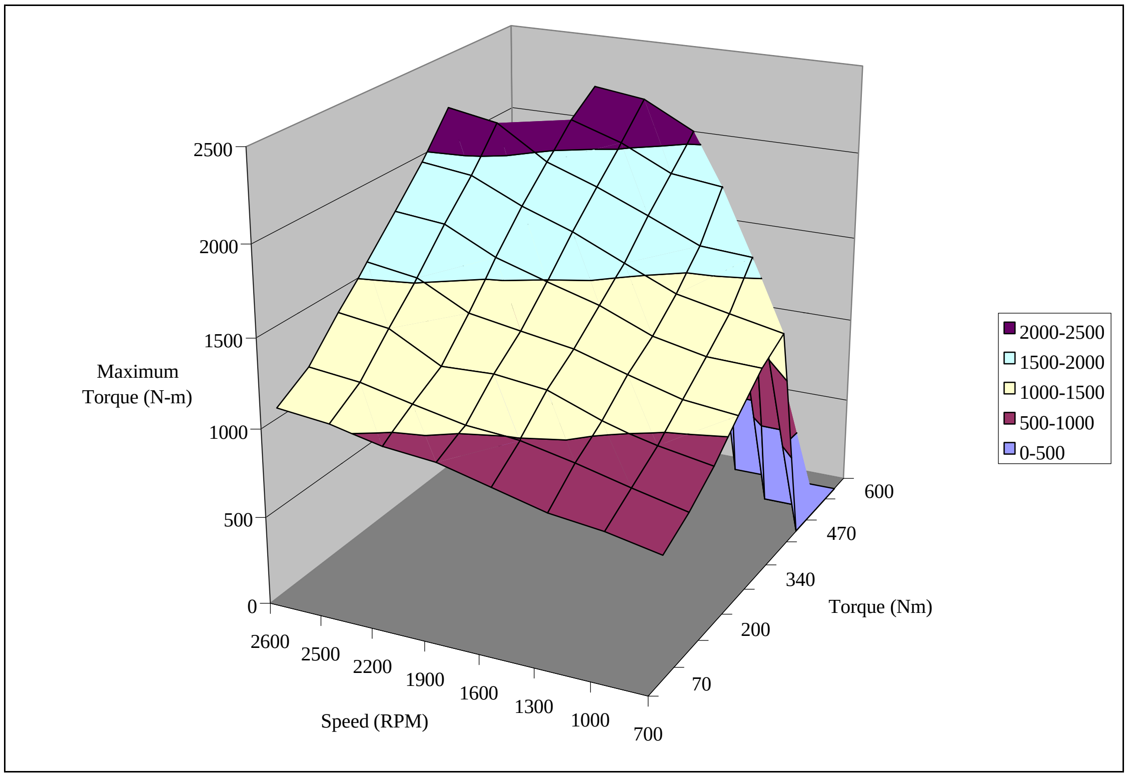

25
20
15
10
5
0
Ignition Delay
(CA deg.)
70
200
340
470
600
Torque (Nm)
700
1000
1300
1600
1900
2200
2500
2600
Speed (RPM)
20-25
15-20
10-15
5-10
0-5


Combustion Duration (CA deg.)
20
15
10
5
0
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)
15-20
10-15
5-10
0-5

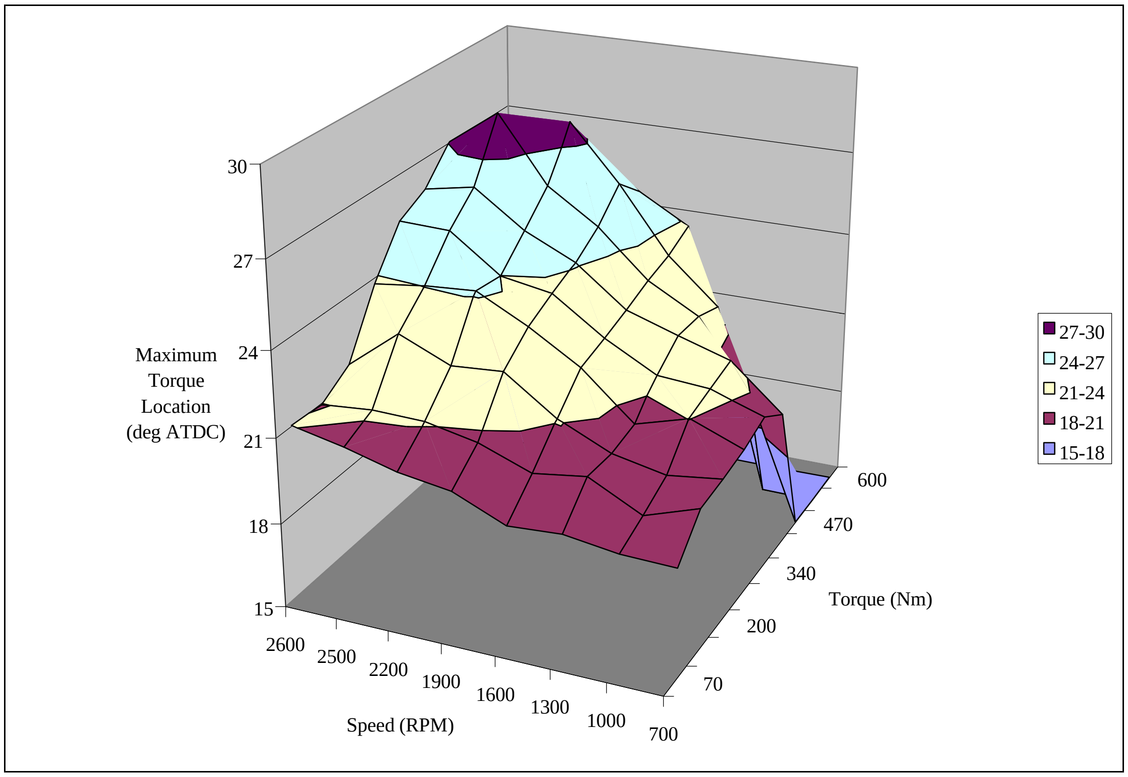

LMFB50 (deg ATDC)
20
15
10
5
0
-5
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)
15-20
10-15
5-10
0-5
-5-0


12
9
6
3
0
Maximum
Burn Rate (kPa/microsec)
700
1000
1300
1600
1900
2200
2500
2600
Speed (RPM)
70
200
340
470
600
Torque (Nm)
9-12
6-9
3-6
0-3

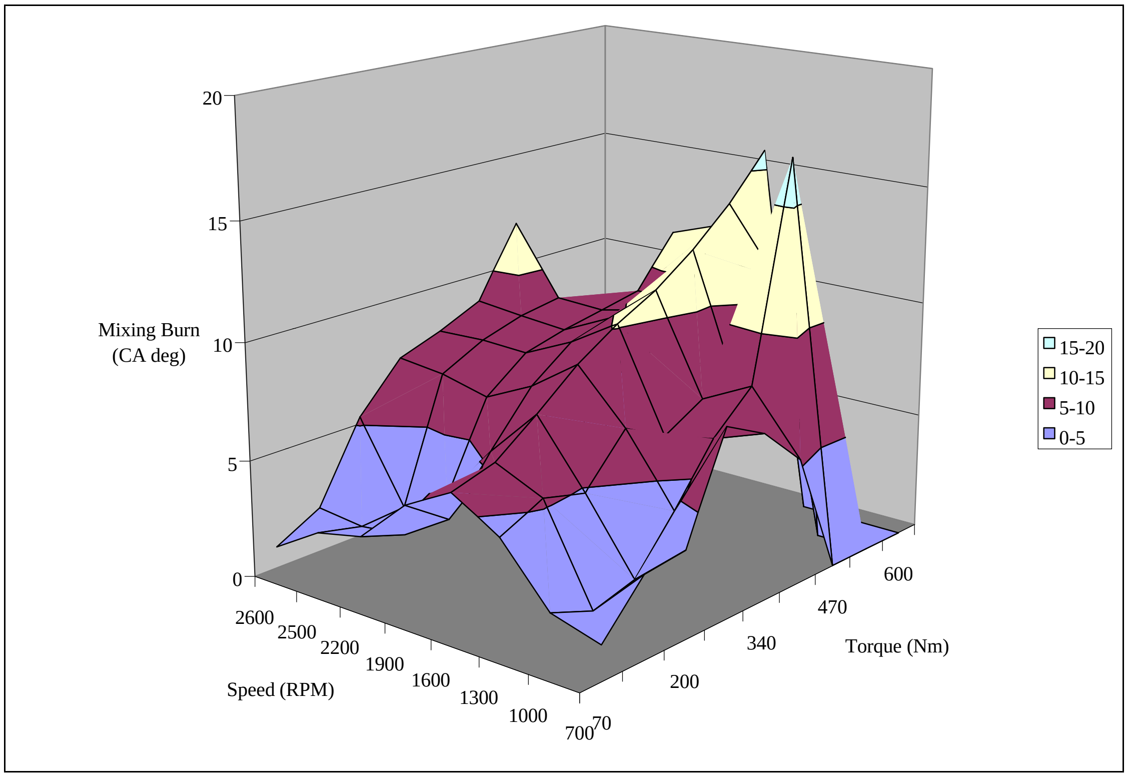

6
3
0
-3
-6
-9
-12
Maximum
Burn Rate
Location
(deg BTDC)
3-6
0-3
-3-0
-6--3
-9--6
-12--9
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)

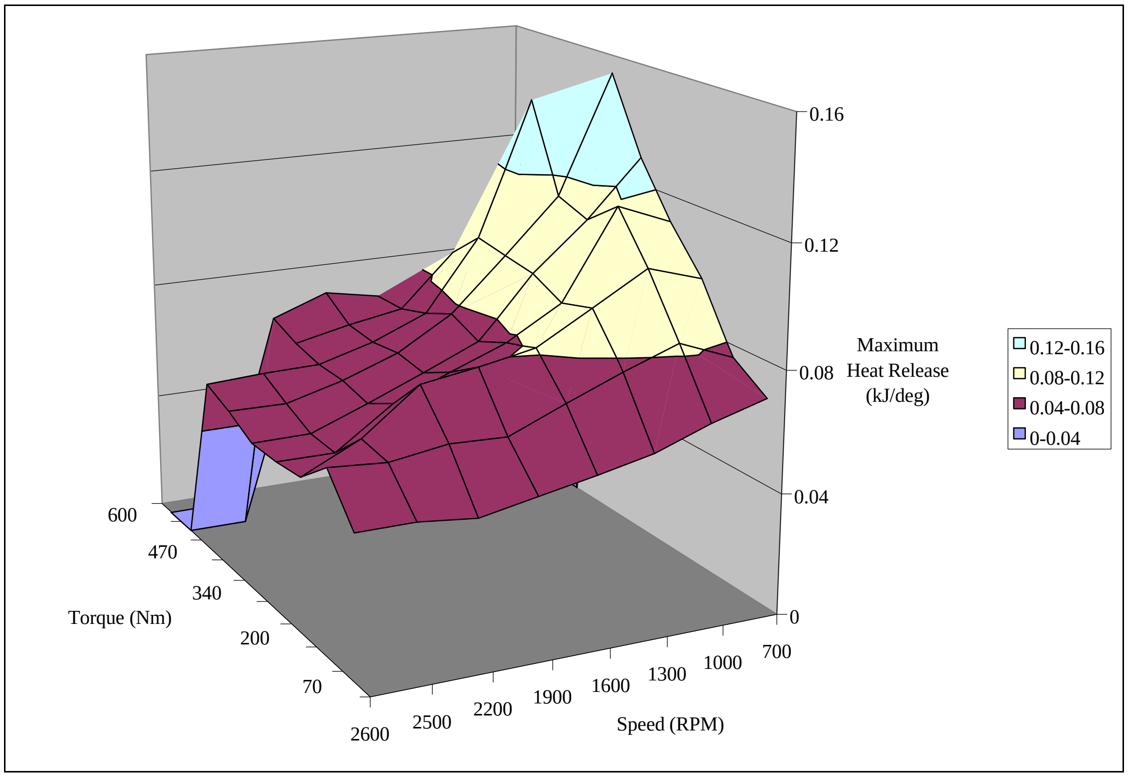


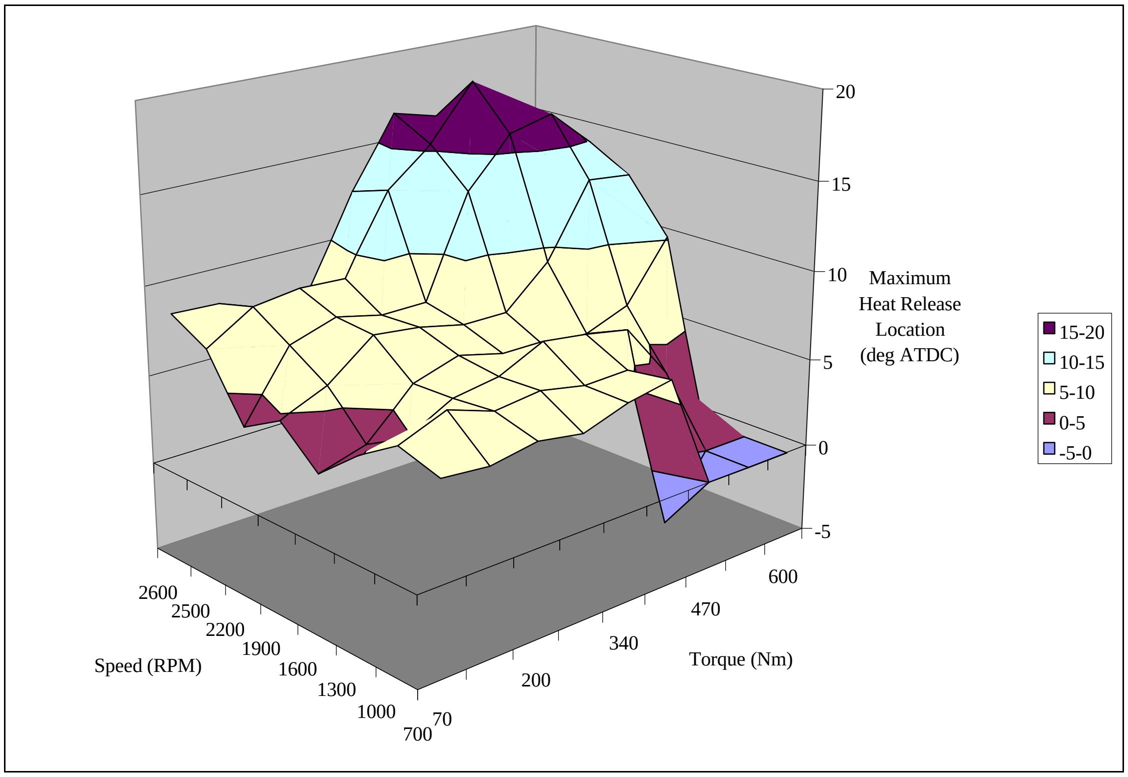
Maximum Torque (N-m)
0
500
1000
1500
2000
2500
Speed (RPM)
2600
2500
2200
1900
1600
1300
1000
700
Torque (Nm)
70
200
340
470
600
2000-2500
1500-2000
1000-1500
500-1000
0-500



Maximum Torque Location (deg ATDC)
15
18
21
24
27
30
Speed (RPM)
2600
2500
2200
1900
1600
1300
1000
700
Torque (Nm)
70
200
340
470
600
27-30
24-27
21-24
18-21
15-18


Mixing Burn (CA deg)
0
5
10
15
20
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)
15-20
10-15
5-10
0-5


Maximum Heat Release (kJ/deg)
0
0.04
0.08
0.12
0.16
600
470
340
200
70
Torque (Nm)
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
0.12-0.16
0.08-0.12
0.04-0.08
0-0.04



20
15
10
5
0
-5
Maximum Heat Release Location (deg ATDC)
15-20
10-15
5-10
0-5
-5-0
2600
2500
2200
1900
1600
1300
1000
700
Speed (RPM)
70
200
340
470
600
Torque (Nm)

Appendix C: Neural Network Training Results: Best 20 for Each Gaseous Emission

		HC				
Avg	Variables	Net Type	In	HN[*]	Tst R^2	Trn R^2
0.9746	IgnDel, CombDur, MaxQ, LMFB50, LPP	3Layer	5	13	0.9599	0.9892
0.9644	Peak, IgnDel, MixBurn, MaxQ, LPP	3Layer	5	16	0.9493	0.9795
0.9553	Peak, IgnDel, CombDur, MaxQ, LMFB50, LPP	3Layer	6	17	0.9766	0.9341
0.9513	Peak, MixBurn, CombDur, MaxQ	3Layer	4	15	0.9161	0.9864
0.9485	Peak, CombDur, MaxQ, LMFB50	3Layer	4	8	0.9311	0.966
0.9422	Peak, IgnDel, CombDur, LMFB50, LPP	3Layer	5	16	0.976	0.9084
0.9414	Peak, MixBurn, MaxQ, LMFB50	Ward1	4	7	0.9281	0.9546
0.9411	Peak, MixBurn, CombDur, MaxQ	Ward1	4	7	0.9166	0.9657
0.9404	Peak, IgnDel, CombDur, MaxQ, LPP	3Layer	5	15	0.9246	0.9562
0.938	Peak, IgnDel, MixBurn, CombDur, LMFB50, LPP	3Layer	6	10	0.9118	0.9642
0.9379	Peak, MaxQ, LMFB50, LPP	Ward2	4	4	0.9395	0.9364
0.9294	Peak, IgnDel, MaxQ, LMFB50, LPP	3Layer	5	5	0.9192	0.9395
0.927	Peak, IgnDel, MixBurn, CombDur, MaxQ, LPP	3Layer	6	11	0.8906	0.9634
0.925	Peak, IgnDel, MixBurn, MaxQ, LMFB50, LPP	Ward2	6	5	0.9223	0.9278
0.9243	IgnDel, CombDur, LMFB50	Ward1	3	4	0.9149	0.9337
0.9243	Peak, IgnDel, MaxQ	Ward1	3	9	0.947	0.9016
0.9175	Peak, IgnDel, CombDur, MaxQ, LMFB50, LPP	Ward2	6	6	0.9124	0.9225
0.9144	Peak, IgnDel, MaxQ	3Layer	3	5	0.9416	0.8871
0.9123	Peak, IgnDel, LMFB50	3Layer	3	17	0.9341	0.8904
0.9079	Peak, MaxQ, LMFB50	Ward1	3	4	0.923	0.8928

[*] Please see footnote on page 59.

		CO				
Avg	Variables	Net Type	In	HN	Tst R^2	Trn R^2
0.9988	Peak, MaxQ, LMFB50	Ward1	3	13	0.9992	0.9984
0.9985	Peak, MaxQ, LMFB50, MixBurn, MaxBurn, LPP	Ward1	6	13	0.9989	0.998
0.9983	Peak, MaxBurnLoc, MaxQ, LMFB50, MaxBurn, LPP	Ward1	6	12	0.9984	0.9982
0.9982	Peak, MaxBurnLoc, MaxQ, LMFB50, MaxBurn, LPP	Ward2	6	8	0.998	0.9984
0.9982	Peak, MaxBurnLoc, MaxQ, LMFB50, MixBurn, MaxBurn	Ward1	6	4	0.9981	0.9982
0.998	Peak, MaxBurnLoc, MaxQ, LMFB50	Ward2	4	6	0.9977	0.9984
0.998	Peak, MaxBurnLoc, MaxQ	Ward1	3	8	0.9981	0.9979
0.998	Peak, MaxQ, LMFB50	Ward2	3	5	0.9981	0.9979
0.998	Peak, MaxQ, LMFB50, MaxBurn, LPP	Ward1	5	14	0.9988	0.9972
0.9979	Peak, MaxBurnLoc, MaxQ, LMFB50, MixBurn, LPP	Ward2	6	4	0.9982	0.9977
0.9979	Peak, MaxBurnLoc, MaxQ, LMFB50	Ward1	4	13	0.9974	0.9984
0.9978	Peak, MaxQ, LMFB50, MaxBurn, LPP	Ward2	5	6	0.9981	0.9975
0.9976	Peak, MaxBurnLoc, MaxQ, LMFB50, LPP	Ward1	5	7	0.9971	0.9981
0.9976	Peak, MaxBurnLoc, MaxQ, MaxBurn, LPP	Ward1	5	5	0.9989	0.9963
0.9976	Peak, MaxQ, LMFB50, MaxBurn	Ward1	4	7	0.9989	0.9963
0.9976	Peak, MaxQ, LMFB50, MixBurn	Ward2	4	5	0.9981	0.9971
0.9976	Peak, MaxBurnLoc, MaxQ, MixBurn, LPP	Ward1	5	11	0.9965	0.9986
0.9975	Peak, MaxBurnLoc, MaxQ, LMFB50, MaxBurn	Ward1	5	10	0.9977	0.9972
0.9974	Peak, MaxQ, LMFB50, MixBurn, MaxBurn	Ward2	5	9	0.9972	0.9976
0.9971	Peak, MaxQ, LMFB50, MixBurn	Ward1	4	7	0.9959	0.9983

		CO2				
Avg	Variables	Net Type	In	HN	Tst R^2	Trn R^2
0.9988	Peak, CombDur, LMFB50, MaxBurnLoc, IgnDel	3Layer	5	16	0.9991	0.9986
0.9988	Peak, CombDur, IMEP, MaxBurnLoc, IgnDel	3Layer	5	9	0.9991	0.9985
0.9986	Peak, CombDur, MaxBurn, MaxBurnLoc, IgnDel	Ward2	5	11	0.9988	0.9984
0.9986	Peak, CombDur, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	3Layer	6	9	0.9982	0.999
0.9986	Peak, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	3Layer	5	6	0.9982	0.999
0.9986	Peak, IMEP, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	Ward2	6	5	0.9981	0.999
0.9985	Peak, CombDur, MaxBurn, MaxBurnLoc, IgnDel	Ward1	5	14	0.9986	0.9985
0.9985	Peak, CombDur, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	Ward2	6	11	0.9985	0.9986
0.9984	Peak, IMEP, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	Ward1	6	14	0.9979	0.9989
0.9984	Peak, IMEP, LMFB50, MaxBurn, IgnDel	Ward2	5	9	0.9981	0.9987
0.9984	All Seven	Ward1	7	6	0.9981	0.9986
0.9984	Peak, IMEP, LMFB50, MaxBurn, IgnDel	3Layer	5	9	0.9981	0.9987
0.9984	Peak, IMEP, MaxBurn, IgnDel	Ward1	4	9	0.9979	0.9988
0.9983	Peak, IMEP, LMFB50, MaxBurn, IgnDel	Ward1	5	9	0.9977	0.9989
0.9983	Peak, CombDur, MaxBurn, IgnDel	3Layer	4	10	0.9987	0.9978
0.9982	Peak, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	Ward1	5	14	0.9979	0.9986
0.9982	Peak, IMEP, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	3Layer	6	19	0.9977	0.9987
0.9982	Peak, IMEP, MaxBurn, IgnDel	Ward2	4	4	0.9977	0.9986
0.9982	Peak, LMFB50, MaxBurn, MaxBurnLoc, IgnDel	Ward2	5	5	0.9976	0.9988
0.9982	Peak, LMFB50, MaxBurnLoc, IgnDel	3Layer	4	19	0.998	0.9983

		NOx				
Avg	Variables	Net Type	In	HN	Tst R^2	Trn R^2
0.998	CombDur, IMEP, LMFB50, IgnDel, MaxQ	3Layer	5	18	0.9983	0.9976
0.9978	Peak, CombDur, IMEP, LMFB50, IgnDel, MaxQ	Ward2	6	4	0.9988	0.9968
0.9978	Peak, CombDur, IMEP, LMFB50, IgnDel, MaxQ	3Layer	6	11	0.9982	0.9974
0.9978	IMEP, LMFB50, IgnDel, MaxBurn, MaxQ	Ward2	5	10	0.9989	0.9966
0.9977	IMEP, LMFB50, IgnDel, MaxQ	3Layer	4	20	0.9982	0.9972
0.9977	CombDur, IMEP, LMFB50, IgnDel, MaxBurn, MaxQ	3Layer	6	15	0.9981	0.9973
0.9977	IMEP, LMFB50, IgnDel, MaxBurn, MaxQ	3Layer	5	20	0.9988	0.9965
0.9976	IMEP, LMFB50, IgnDel	Ward1	3	8	0.9981	0.9971
0.9976	CombDur, IMEP, LMFB50, IgnDel, MaxQ	Ward2	5	4	0.9985	0.9967
0.9976	Peak, CombDur, IMEP, LMFB50, IgnDel, MaxQ	Ward1	6	4	0.9977	0.9974
0.9975	IMEP, LMFB50, IgnDel, MaxBurn, MaxQ	Ward1	5	4	0.9986	0.9964
0.9975	IMEP, LMFB50, IgnDel, MaxBurn	Ward1	4	4	0.9988	0.9962
0.9975	CombDur, IMEP, LMFB50, IgnDel, MaxQ	Ward1	5	7	0.9979	0.9971
0.9975	CombDur, IMEP, LMFB50, IgnDel, MaxBurn, MaxQ	Ward1	6	5	0.9986	0.9963
0.9975	IMEP, LMFB50, IgnDel	3Layer	3	16	0.9984	0.9966
0.9974	IMEP, LMFB50, IgnDel, MaxQ	Ward2	4	4	0.9976	0.9971
0.9973	IMEP, LMFB50, IgnDel, MaxBurn	Ward2	4	7	0.9981	0.9965
0.9973	CombDur, IMEP, LMFB50, IgnDel, MaxBurn	Ward1	5	6	0.9978	0.9968
0.9973	IMEP, LMFB50, IgnDel, MaxBurn	3Layer	4	17	0.9979	0.9966
0.9972	CombDur, IMEP, LMFB50, IgnDel	Ward2	4	5	0.9977	0.9968

Appendix D: Texas Instruments DSP Evaluation Module Source Code

```
#include "stdlib.h"
#include "math.h"
#include "c30_1wvu.h"
#include "trav_c30.h"

/**********  Register and serial port constants
*************************/

long IOF_AMASK = 0x000000E;
long IOF_SET_XF1 = 0x0000060;
long IOF_RESET_XF1 = 0x0000020;
long CTRL = 0x0808000;
long SERGLOB1 = 0x8100080;  /*8120280 for continuous mode, 810 for fixed
burst mode*/
long SERPRTX1 = 0x0000020;
long BDX1_0 = 0x0000020;    /* Set DX1 to 0  */
long BDX1_1 = 0x0000060;    /* Set DX1 to 1  */
long SERPRTR1 = 0x0000111;
long SERTIM1 = 0x00003C0;
long SERTIM1VAL = 0x00020000;
long HOST_DATA = 0x00804000;

/**********  Program variables
*********************************************/

int ioftemp;
int iftemp;
static float pressure[1026];
static float volume[1026];
static float delQ[770];
static float pressmooth[1026];
static float Pr[6];

void init_ser_port(void);
float calc_volume( int i);
extern unsigned int ieeeflt();

void main(void)
{

/** transduc_const is derived from the transducer, charge amplifier
pair.
0.145 = psi to kPa conversion, .897 = pC/psi, .997 = mV/pC (Ch. Amp.),
1000 = mV/V
1.0 = PCB Box Gain, 0.8408 = Voltage Divider in DSP Circuit, 16383 = 14
bit ADC, 10 = EVM range) */

float transduc_const = (.145 * 0.897 * 0.997 / 1000.0 * 1.0 * 0.8408 *
16383 / 10.0);

int index;
float max;
int maxloc;
int i,j;
int looplength;
int input_temp;
int LDPISumTotal;
int LDPISum;
int LDPI50;
unsigned int *dump;
unsigned int junkhigh, junklow, junktemp;
int mapval;
unsigned int mapxfer;
long int mapsum;
float map;
float imepg,intarea,PolyConst;
int MaxMassBurnLoc, counter;
float MassBurn,MaxMassBurn, VolumeChange;
float PVolume;
float gamma = 1.25;
int Flag10, Flag50, Flag90, Flag99;
int Start10, Start50, End90, End99;
int Start, MFBStart, MFBLength;
int CombDur, MixBurn, MFBCounter;
float MaxQ, MFBMax;
float MFBSum;

init_evm();
init_host();
init_ser_port();

/* Initialize map and mapval.  Note: first calculation will be
inaccurate due to this*/
map = 101.325;
mapval = 7200;

for (i=0; i<=1024; i++){
       volume[i] = calc_volume(i);
       pressure[i] = 0.0;
}

ioftemp = 0x80;
looplength=1024;

asm(" LDI       IOF,R1");    /* Set XF1 as input pin */
asm(" AND       0Fh,R1");
asm(" LDI       R1,IOF");



do{
        index = 0;
        max = 0.0;
        maxloc = 0;
        mapsum = 0;

                   /************ Wait until signal is high
******************************/
        do{
                          serial_port[1][X_PORT] = BDX1_0;
                    asm(" LDI      IOF,R1");
              asm(" STI      R1,@_ioftemp");
                        ioftemp = ioftemp & 0x80;

                   }while(ioftemp==0);

        /*********** Wait for signal to be low
*******************************/

                   do{
                          serial_port[1][X_PORT] = BDX1_1;
              asm(" LDI      IOF,R1");
              asm(" STI      R1,@_ioftemp");
                   }while((ioftemp & 0x80));




                   /********** Toggle DX1 *************/

                   serial_port[1][X_PORT] = BDX1_1;
                   serial_port[1][X_PORT] = BDX1_0;


        do{

              do{
              }while(!(serial_port[1][GLOBAL] & 0x1));

              input_temp = serial_port[1][R_DATA] & 0xFFFF;
              input_temp = (input_temp ^ 0x8000) >> 2;

              pressure[index] = map + (input_temp -
mapval)/transduc_const;

              if ( pressure[index] > max) {
                                                  max = pressure[index];
                                                  maxloc=index;
              }
              if (index <= 20){
                        mapsum+=input_temp;
              }
              index++;


        }while(index <= looplength);  /********* End of acquisition loop
***/

        /**********  Now process data
******************************************/

        mapval = mapsum/21;

        intarea = 0.0;
        for(i=1;i<=1024;i++){
                  intarea += pressure[i] * (volume[i]-volume[i-1]);
        }
        imepg = intarea*1000.0/0.9125;

        pressmooth[439] = (pressure[437] + pressure[438] + pressure[439]
+ pressure[440] + pressure[441])/5.0;
        PolyConst = map * 1.206488e-4;   /**** MAP *  (Displacement +
Clearance Vol. ^ gamma) ***/
        MaxMassBurn = 0;
        MaxMassBurnLoc = 0;
        counter = 0;
        MFBMax = 0;
        MaxQ = 0;
        MFBSum = 0;
        MFBCounter = 0;
        for(j=0;j<=4;j++){
                  Pr[j] = 0.0;
        }


        for(i=440;i<=640;i++){
                  pressmooth[i] = (pressure[i-2] + pressure[i-1] +
pressure[i] + pressure[i+1] + pressure[i+2])/5.0;
                  delQ[i] = (pressmooth[i]+(pressmooth[i]/(gamma -
1.0)))*(volume[i]-volume[i-1])+(volume[i]/(gamma-1.0))*(pressmooth[i]-
pressmooth[i-1]);

                  if (delQ[i] > MaxQ) {
                        MaxQ = delQ[i];
                  }
                  if(delQ[i] < 0){
                        MFBSum = 0;
                        Start = i;
                        MFBCounter = 0;
                  }
                  MFBSum += delQ[i];
                  MFBCounter++;
                  if (MFBSum > MFBMax){
                        MFBMax = MFBSum;
                        MFBStart = Start;
                        MFBLength = MFBCounter;
                  }
                  VolumeChange = pow((volume[i-1]/volume[i]),gamma)-1;
                  PVolume = PolyConst / pow(volume[i-1], gamma);
                  counter++;

                  switch (counter){
                        case 1 : Pr[0] = pressmooth[i] - PVolume;
                                      goto xxxx;
                        case 2 : Pr[1] = pressmooth[i] - PVolume;
                                      goto xxxx;
                        case 3 : Pr[2] = pressmooth[i] - PVolume;
                                      goto xxxx;
                        case 4 : Pr[3] = pressmooth[i] - PVolume;
                  }
                  Pr[4] = pressmooth[i] - PVolume;

                  MassBurn = ((-Pr[4] + 8*Pr[3] - 8*Pr[1] + Pr[0])/12.0)
* (2.8444);
                  if (MassBurn > MaxMassBurn) {
                        MaxMassBurn = MassBurn;
                        MaxMassBurnLoc = i;    /*  This is a relic
mistake from Offline program.  Should be i-2  */
                  }
                  Pr[0] = Pr[1];
                  Pr[1] = Pr[2];
                  Pr[2] = Pr[3];
                  Pr[3] = Pr[4];
xxxx:;

        }
        MFBSum = 0;
        Flag10 = 0;
        Flag50 = 0;
        Flag90 = 0;
        Flag99 = 0;
        for (j=MFBStart+1; j<=MFBStart+MFBLength;j++){
                  MFBSum += delQ[j];
                  if ((MFBSum >= (0.1*MFBMax)) && (Flag10 == 0)){
                    Start10 = j;
                    Flag10 = 1;
                  }
                  if ((MFBSum >= (0.5*MFBMax)) && (Flag50 == 0)){
                    Start50 = j;
                    Flag50 = 1;
                  }
                  if ((MFBSum >= (0.9*MFBMax)) && (Flag90 == 0)){
                    End90 = j;
                    Flag90 = 1;
                  }
                  if ((MFBSum >= (0.99*MFBMax)) && (Flag99 == 0)){
                    End99 = j;
                    Flag99 = 1;
                  }
        }


        CombDur = End90 - Start10;
        MixBurn = End99 - Start50;


        LDPISumTotal = 0;
/*          for(i=512; i<=1023; i++){
              LDPISumTotal += pressure[i] - pressure[1023-i];
        }
        LDPISum = 0;
        i=511;
        do{
              i++;
              LDPISum += pressure[i] - pressure[1023-i];
              LDPI50 = i;
                   }while(LDPISum <= (LDPISumTotal/2));*/

        *host = MixBurn;   /*  MixBurn  */

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");


        *host = CombDur;   /*CombDur */

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");


        *host = MaxMassBurnLoc;

        do{
              asm(" LDI      IF,R1");
```

```
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        *host = maxloc;

        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        *host = MFBStart;

        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");


            *host = Start50;

        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

/*******   Pass peak pressure to Host
*************************************/

        junktemp = ieeeflt(max);
        junklow = junktemp & 0xFFFF;
        junkhigh = junktemp >> 16;

        *host = junkhigh;

        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        *host = junklow;

        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

/******* Pass IMEPg to Host
*********************************************/

        junktemp = ieeeflt(imepg);
        junklow = junktemp & 0xFFFF;
        junkhigh = junktemp >> 16;
        *host = junkhigh;

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        *host = junklow;

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        /******* Pass MaxQ to Host
**********************************************/

        junktemp = ieeeflt(MaxQ);
        junklow = junktemp & 0xFFFF;
        junkhigh = junktemp >> 16;


        *host = junkhigh;

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        *host = junklow;

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");


        /******* Pass MaxMassBurn to Host
**********************************************/

        junktemp = ieeeflt(MaxMassBurn);
        junklow = junktemp & 0xFFFF;
        junkhigh = junktemp >> 16;


        *host = junkhigh;

        do{
              asm(" LDI      IF,R1");
              asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");

        *host = junklow;

        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x4)==0 );
        asm(" LDI     0h,IF");


/****  Read MAP Integer Value from the Host  ****/
        do{
            asm(" LDI      IF,R1");
            asm(" STI      R1,@_iftemp");
        }while((iftemp & 0x2)==0 );
        mapxfer = *host;
        mapxfer &= 0xFFFF;

        map = (mapxfer*300.0/65535)+90.0;

        asm(" LDI     0h,IF");


}while(1);    /************ End of endless loop, hahaha
*******************/


}

void init_ser_port(void)
{

serial_port[1][R_TPER] = SERTIM1VAL;
serial_port[1][GLOBAL] = SERGLOB1;
serial_port[1][X_PORT] = SERPRTX1;
serial_port[1][R_PORT] = SERPRTR1;
serial_port[1][R_TCON] = SERTIM1;

}

float calc_volume(int j)
{
float crankangle;
float pistonpos;
float pi;
float answer;


pi = 4.0 * atan(1.0);
crankangle = -180.0 + j*360.0/1024;
pistonpos = 0.0531* cos(crankangle * pi /180.0) + sqrt(3.279721E-2 -
2.81961E-3 * pow(sin(crankangle * pi /180.0), 2));
answer = 5.530303e-5 + (pi*2.724318e-3)*(0.2342-pistonpos);
return answer;

}
```

Appendix E: Data Acquisition Source Code

```
/*  DSP Based Acquisition program for the Navistar 444 diesel engine.
        1st Iteration - Ground up construction 11/13/97 - MLT

        First adjustments made.  CTM-10 and DAS-16 information
        changed to reflect new acquisition setup for Navistar engine - 3/13/98

        Convert updated, several modules updated and imported and general
        housecleaning to represent Navistar setup - 4/9/98

        General programming improvements added and some corrections made - 7/9/98

        Eliminated the variable Statadd as it was superfluous.  7/14/98

        Completed update, finished Menu logic and added MAP conversion.  7/16/98

        Adjusted DSP Reading module and eliminated unnecessary sections.  10/5/98

*/

#include<stdio.h>
#include<graphics.h>
#include <dos.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

void setupadc(int datadd, int cmdadd);
void screen(char **ChannelTitle, double *ChannelValue, float *PressureInfo);
void conditns(float *AmbCond);
void backgrnd(char **ChannelTitle, int datadd, int cmdadd);
void filename(float *AmbCond, float *AnalyzerBackground, char *inffilename);
void getcoeff(float *AnalyzerBackground, char **ChannelTitle);
void getadc(long int *ChannelADC);
void convert(float *AnalyzerBackground, long int *ChannelADC, double *ChannelValue, float
*AmbCond);
void readdsp(float *PressureInfo, float MAP);
void savedisk(double *ChannelValue, float *PressureInfo, char *inffilename);

double AnalyzCoeff[4][4];
int gdriver = VGA;
int gmode = VGAHI;



main (void)
{
 int i;
 int datadd=0x364;
 int cmdadd=0x365;
 long int ChannelADC[24];
 float AmbCond[3];
 float AnalyzerBackground[4];
 double ChannelValue[36];
 char inffilename[8];
 float PressureInfo[12];
 float MAP, oldMAP;
 char choice, choicedummy;
 int writeflag, synchroflag, choiceflag;
 char tempstring[25] = "C:\\DATA\\" ;
 char dirstring[25];
 FILE *dsp_file;
 char *ChannelTitle[]={
            "MAP", "Adv", "FIPW", "Speed", "n/a", "HC", "lCO", "CO2", "NOx", "VnT",
            "VnP", "TRQ", "ECT", "EOT", "ExT", "IAT", "n/a", "APS", "ICP", "n/a", "n/a" };


 /* initialize graphics and local variables */
/* initgraph(&gdriver, &gmode, "D:\\bc45\\bgi");
 setcolor(1);*/
 oldMAP = 101.325;
 writeflag = 0;
 synchroflag = 0;
 conditns(AmbCond);
 backgrnd(ChannelTitle, datadd, cmdadd);
 getcoeff(AnalyzerBackground, ChannelTitle);
 setupadc(datadd, cmdadd);

 do{
            filename(AmbCond, AnalyzerBackground, inffilename);
            strcpy(dirstring, tempstring);
            strcat(dirstring, inffilename);
            strcat(dirstring, ".dsp");

            dsp_file = fopen( dirstring, "a+");

            do{
                        if (synchroflag == 1){
                                    printf("Press Button to Start\n");
                                          do{
                                    }while(!inp(0x2FE));
                                    printf("Data capture initiated.\n");
                                    synchroflag = 0;
                        }
                        outp(0x328, 0);
                        outp(0x329,2);    /*Start Das-16 on Trigger, Pin 25 */


                        while (!(inp(0x328) & 0x10)){
                        };

                        getadc(ChannelADC);

                        MAP = -.0046*pow((1e6/ChannelADC[0]),2)+3.2644*(1e6/ChannelADC[0])-
203.29;
                        if ((MAP < 75) || (MAP > (1.25*oldMAP))){
                                   MAP = oldMAP;
                        }
```

```
                    oldMAP = MAP;
                    readdsp(PressureInfo, MAP);

                    if (writeflag == 0){
                              convert(AnalyzerBackground, ChannelADC, ChannelValue,
AmbCond);
                              screen(ChannelTitle, ChannelValue, PressureInfo);
                              printf("\nMAP = %3.1f", MAP);
                    }
                    if (writeflag == 1){
                               for (i=0;i<=20;i++){
                                         fprintf(dsp_file, "%d ,", ChannelADC[i]);
                               }
                               for (i=0;i<=8;i++){
                                         fprintf(dsp_file, "%5.5f ,", PressureInfo[i]);
                               }
                               fprintf(dsp_file, "%5.5f \n", PressureInfo[9]);

                    }
                    /*outp(cmdadd, 0xB8);  /* Hold counters 4 and 5 */


          }while(!kbhit());


          choicedummy = toupper(getch());
          sleep(1);
          choiceflag = 0;
          fclose(dsp_file);

          do{
                    printf("Enter Choice of Operation: <M>onitor, <S>ynchronize,\n");
                    printf("<W>rite to Disk, E<X>it.\n");
                    printf("To synchronize, install COM port y-connector switch.\n");
                    choice = toupper(getch());

                    if (choice == 'M')
                               choiceflag = 1;
                    else if (choice == 'S'){
                               choiceflag = 1;
                               printf("Synchronizing Switch Set, Press Button to Take Data");
                    }
                    else if (choice == 'W'){
                               choiceflag = 1;
                               printf("Writing Data to File, Press a Key to Interrupt");
                    }
                    else if (choice == 'X')
                               choiceflag = 1;
                    else choiceflag = 0;

          }while(choiceflag == 0);

          switch (choice){
                case 'M': writeflag = 0;
```

```
                                        synchroflag = 0;
                                        break;
                         case 'S': writeflag = 1;
                                        synchroflag = 1;
                                        break;
                         case 'W': writeflag = 1;
                                        synchroflag = 0;
                                        break;
                         case 'X': break;
             }
 sleep(1);

 }while(choice != 'X');

 closegraph();

 return 0;
}

#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <string.h>


void savedisk(double *ChannelValue, float *PressureInfo, char *inffilename)
{
 char tempstring[25] = "C:\\DATA\\" ;
 char dirstring[25];
 int i;
 FILE *dsp_file;

 strcpy(dirstring, tempstring);
 strcat(dirstring, inffilename);
 strcat(dirstring, ".dsp");

 dsp_file = fopen( dirstring, "a+");

 for (i=0;i<=20;i++){
                         fprintf(dsp_file, "%5.5f ,", ChannelValue[i]);
 }
 for (i=0;i<=9;i++){
                         fprintf(dsp_file, "%5.5f ,", PressureInfo[i]);
 }
 fprintf(dsp_file, "%5.5f \n", PressureInfo[10]);

 fclose(dsp_file);

}

/* This subprogram reads the values from backgrnd.dat and creates an information
file for reduction purposes.  */

#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <conio.h>

extern double AnalyzCoeff[4][4];
getcoeff(float *AnalyzerBackground, char **ChannelTitle)
{
  int AnalyzADC[4];
  int i;
  int j;
  int jj;
  int p;
  int k;
  char trash[25];
  char coeffstring[80];
  char tempstring[25] = "C:\\DATA\\CAL\\";
  char dirstring[25];
  FILE *file_ptr;
  FILE *coefffile;


  for (i=5;i<=8;i++){
                        strcpy(dirstring, tempstring);
                        strcat(dirstring, ChannelTitle[i]);
                        strcat(dirstring, ".cal");
                        coefffile = fopen(dirstring, "r");

                        for (j=1;j<=2;j++){
                                    fgets(trash, 25, coefffile);
                                    /*printf("%d  %s",j, trash);*/
                        }


                        for (jj=0;jj<=3;jj++){
                                    fgets(coeffstring, 40, coefffile);
                                    p = strlen(coeffstring);

                                    for (k=1;k<=p;k++){
                                                if (coeffstring[k] == 'D') {
                                                            coeffstring[k] = 'E';
                                                }
                                    }

                                    AnalyzCoeff[i-5][jj] = atof(coeffstring);
                          /*        printf("%d %e %d %s\n",jj,AnalyzCoeff[i-7][jj],p, coeffstring);*/

                        }

                        /*printf("\n File source is : %s \n", dirstring);*/
                        fclose(coefffile);

  }
  file_ptr = fopen( "C:\\DATA\\CAL\\Backgrnd.dat", "r");
  for (i=0;i<=3;i++){
              fscanf(file_ptr, "%d", &AnalyzADC[i]);
              AnalyzerBackground[i]=
AnalyzCoeff[i][0]+AnalyzCoeff[i][1]*AnalyzADC[i]+AnalyzCoeff[i][2]*pow(AnalyzADC[i],2)+AnalyzC
oeff[i][3]*pow(AnalyzADC[i],3);

  }
  fclose(file_ptr);


  return 0;
}

/* This Subprogram will gather the data from the boards  */
#include <dos.h>
#include <stdio.h>
#include <conio.h>

getadc(long int *ChannelADC)
{
             int i;
             long int hibyte;
             long int lobyte;
             int datadd = 0x364;
             int cmdadd = 0x365;

             /* This section sets up the CTM-10 board for various important parameters */
             outp(cmdadd, 0x11);    /* Read Counter 1 Hold : MAP */
             lobyte = inp(datadd);
             hibyte = inp(datadd);
             ChannelADC[0] = lobyte+256*hibyte;
             if (ChannelADC[0] == 0){
                          ChannelADC[0] = 8500;
             }
/*           printf("cmdadd = %x,   datadd = %x \n", cmdadd, datadd);*/
/*           printf("ChannelADC[0] = %d\n", ChannelADC[0]);*/

             outp(cmdadd, 0x12); /* Read Counter 2 Hold : Timing */
             lobyte = inp(datadd);
             hibyte = inp(datadd);
             ChannelADC[1] = lobyte+256*hibyte;
/*           printf("ChannelADC[1] = %d\n", ChannelADC[1]);*/

             outp(cmdadd, 0x13); /* Read Counter 3 Hold : FIPW */
             lobyte = inp(datadd);
             hibyte = inp(datadd);
             ChannelADC[2] = lobyte+256*hibyte;
/*           printf("ChannelADC[2] = %d\n", ChannelADC[2]);*/

             outp(cmdadd, 0x14); /* Read Counter 4 Hold: Speed */
             lobyte = inp(datadd);
             hibyte = inp(datadd);
             ChannelADC[3] = lobyte+256*hibyte;
/*           printf("ChannelADC[3] = %d\n", ChannelADC[3]);*/
```

```
        outp(cmdadd, 0x15); /* Read Counter 5 Hold : n/a */
        lobyte = inp(datadd);
        hibyte = inp(datadd);
        ChannelADC[4] = lobyte+256*hibyte;

        outp(0x322, 0xF0);  /*Read DAS-16 from channels 0 to 15(F) */
        for (i=5; i<=20;i++) {
                outp(0x320, 0);
                while (inp(0x328) & 0x80){
                                }
                ChannelADC[i] = inp(0x320)/16 + inp(0x321)*16;
        }

        outp(0x328, 0);

        return 0;
}

#include <time.h>
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <string.h>

extern double AnalyzCoeff[4][4];

void filename(float *AmbCond, float *AnalyzerBackground, char *inffilename)
{
        struct  time t;
        time_t tj;
        char tempstring[25] = "C:\\DATA\\" ;
        char dirstring[25];
        char timestring[8];
        char secondhalf[6];
        char firsthalf[6];
        char datebuf[9];
        int p;
        int i, j;
        FILE *inf_file;

        tj = time(NULL);
        ltoa(tj, timestring, 10);
        p=strlen(timestring);

        for (i=0;i<=4;i++){
                secondhalf[i] = timestring[p-4+i];
        }

  _strdate(datebuf);
        for(i=0;i<=4;i++){
                if (i==0 || i==1){
                        firsthalf[i]=datebuf[i];
                }
                if (i==3 || i==4){
                        firsthalf[i-1]=datebuf[i];
                }
        }

        firsthalf[4] = 0;

        strcpy(inffilename, firsthalf);
        strcat(inffilename, secondhalf);

        printf("Date is %s \n", datebuf);
        printf("Filename is : %s \n", inffilename);

        gettime(&t);
        strcpy(dirstring, tempstring);
        strcat(dirstring, inffilename);
        strcat(dirstring, ".inf");

        inf_file = fopen( dirstring, "w");
        fprintf(inf_file, "%s \n", datebuf);
        fprintf(inf_file, "%2d:%02d:%02d \n",t.ti_hour, t.ti_min, t.ti_sec);
        fprintf(inf_file, "Ambient Temp : %2.1f \n", AmbCond[0]);
        fprintf(inf_file, "Ambient RH   : %2.1f \n", AmbCond[1]);
        fprintf(inf_file, "Ambient Pres : %3.1f \n", AmbCond[2]);
        for (i=0;i<=3;i++){
                fprintf(inf_file, "Analyzer[%d] background is \n %5.1f \n", i, AnalyzerBackground[i]);
                fprintf(inf_file, "Coefficients are:\n");
                for (j=0;j<=3;j++){
                        fprintf(inf_file,"%.15e \n", AnalyzCoeff[i][j]);
                }
        }
        fclose(inf_file);


}

/*  This subprogram converts the ADC code data into real engineering unit
data */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

extern double AnalyzCoeff[4][4];

convert(float *AnalyzerBackground, long int *ChannelADC, double *ChannelValue, float *AmbCond)
{
  int i;
  float TempConvert;
  float rhoHC = 0.5746;         /* Densities are based on Diesel #2 Specs.  */
  float rhoNOx = 1.913;
  float rhoCO = 1.164;
  float rhoCO2 = 1.83;
  float H, Kh, Qeng;
  float Q, Prespsf, TempRank;
  float SatVapPres;


  for (i=0;i<=20;i++){

                        switch(i)
                        {

                        /*  Case 0 is the MAP taken from the CTM-10 board  (kPa)*/
                                case 0: if(ChannelADC[i] != 0)
                                                        ChannelValue[i] = -
.0046*pow((1e6/ChannelADC[i]),2)+3.2644*(1e6/ChannelADC[i])-203.29;
                                                else
                                                        ChannelValue[i] = 0;
                                                break;

                        /*  Case 1 is the Ignition timing, in microseconds, CTM-10 board */
                                case 1: ChannelValue[i] = ChannelADC[i];
                                                break;

                        /*  Case 2 is the fuel ignition pulse width in microseconds */
                                case 2: ChannelValue[i] = ChannelADC[i];
                                                break;

                        /*  Case 3 is the engine speed taken from the CTM-10 board */
                                case 3: if (ChannelADC[3] != 0)
                                                    ChannelValue[i] = 60.0
/(ChannelADC[i]*0.00001);
                                                else {ChannelValue[i] = 1;
                                                        ChannelADC[3] = 1;
                                                }
                ChannelValue[1] = ((float) ChannelADC[1]/ ChannelADC[3]*36.0);
                                                break;

                        /*  Case 4 is not used */
                                case 4: ChannelValue[i] = ChannelADC[i];
                                                break;

                        /* Case 5 is HC, 6:CO, 7:CO2, 8:NOx  */
                                case 5: TempConvert = ChannelADC[i];
                                                    ChannelValue[i] = (AnalyzCoeff[0][0] +
AnalyzCoeff[0][1]*TempConvert+AnalyzCoeff[0][2]*pow(TempConvert,2)+AnalyzCoeff[0][3]*pow(Te
mpConvert,3))-AnalyzerBackground[0];
                                                    break;

                                case 6: TempConvert = ChannelADC[i];
                                                    ChannelValue[i] = (AnalyzCoeff[1][0] +
AnalyzCoeff[1][1]*TempConvert+AnalyzCoeff[1][2]*pow(TempConvert,2)+AnalyzCoeff[1][3]*pow(Te
mpConvert,3))-AnalyzerBackground[1];
                                                    break;

                                case 7: TempConvert = ChannelADC[i];
```

```
                                                ChannelValue[i] = (AnalyzCoeff[2][0] +
AnalyzCoeff[2][1]*TempConvert+AnalyzCoeff[2][2]*pow(TempConvert,2)+AnalyzCoeff[2][3]*pow(Te
mpConvert,3))-AnalyzerBackground[2];
                                                break;

                        case 8: TempConvert = ChannelADC[i];
                                                ChannelValue[i] = (AnalyzCoeff[3][0] +
AnalyzCoeff[3][1]*TempConvert+AnalyzCoeff[3][2]*pow(TempConvert,2)+AnalyzCoeff[3][3]*pow(Te
mpConvert,3))-AnalyzerBackground[3];
                                                break;

                /* Case 9 is the Venturi Temperature K */
                        case 9: ChannelValue[i] = (ChannelADC[i]-819.2) * 0.8 * 0.292969 + 273;
                                        break;

                /* Case 10 is the Venturi Pressure kPa */

                        case 10: ChannelValue[i] = (ChannelADC[i]-819.2) * 0.8 * 1.054688 / 144 /
0.145;
                                        break;

                /* Case 11 is Engine Brake Torque  Nm */
                        case 11: ChannelValue[i] = (ChannelADC[i] * 0.5568 - 585.05) / 0.7376;
                                        break;

                /* Case 12 is the Engine Coolant Temperature in deg. C */
                        case 12: ChannelValue[i] = pow((ChannelADC[i]/819.2),3) * -2.5206
                                                + pow((ChannelADC[i]/819.2),2) * 20.535 +
ChannelADC[i]/819.2 * -73.587 + 131.29;
                                        break;

                /* Case 13 is the Engine Oil Temperature in deg. C */
                        case 13: ChannelValue[i] = pow((ChannelADC[i]/819.2),3) * -1.9667
                                                + pow((ChannelADC[i]/819.2),2) * 16.497 +
ChannelADC[i]/819.2 * -66.835 + 148.29;
                                        break;

                /* Case 14 is Exhaust Temperature in deg. C */
                        case 14: ChannelValue[i] = ChannelADC[i] * 0.2436 - .7545;
                                        break;

                /* Case 15 Intake Air Temperature */
                        case 15: ChannelValue[i] = ChannelADC[i] * 0.1231 - 4.592;
                                                break;
                        /*  Channel 16 is unused  */
                        case 16:
                                                ChannelValue[i] = ChannelADC[i] /819.2;
                                                break;

                        /* Case 17 is the Accelerator Pedal Sensor (%)  */
                        case 17: ChannelValue[i] = (ChannelADC[i] - 503) / (3620.0-503.0) * 100;
                                                break;

                        /* case 18 is the fuel rail pressure (ICD) */
```

```
                                    case 18: ChannelValue[i] = ChannelADC[i];
                                                        break;
                                    case 19: ChannelValue[i] = ChannelADC[i];
                                                        break;
                                    case 20: ChannelValue[i] = ChannelADC[i];
                                                        break;


                              }  /* end switch */
               }   /* end "for" loop */


/* Calculate Intake conditions for emissions adjustment Kh is NOx humidity
               ajdustment  */

SatVapPres = (1.084627e-6 * pow(AmbCond[0],3) - 0.0000300890207*pow(AmbCond[0],2) +
0.00121605333*AmbCond[0] + 0.00295389169)*100;
H = 6.211 * AmbCond[1] * SatVapPres / (AmbCond[2]*0.1333224 - (SatVapPres* AmbCond[1]/100));
Kh = 1/(1-0.0182*(H-10.71));

/* Determine Tunnel Flowrate  10.791 in Qeng calculation is the calibrated
coefficient for the stationary lab venturi #1.  Ventury #2 is 10.766. */

TempRank = ChannelValue[9] * 9/5;           /* K to Rankine*/
Prespsf = ChannelValue[10]/ 0.04788026;     /* kPa to lbs/ft^2 conversion */
Qeng = 10.791 * Prespsf/(sqrt(TempRank));   /* ft^3/min */
Q = Qeng * 0.00047195;                      /* m^3/sec */

/* Convert ppm values for emissions into g/s */

  ChannelValue[5] = ChannelValue[5] / 1E6 * Q * rhoHC * 1000.0;   /*1000 is for kg to g */
  ChannelValue[6] = ChannelValue[6] / 1E6 * Q * rhoCO * 1000.0;
  ChannelValue[7] = ChannelValue[7] / 1E6 * Q * rhoCO2 * 1000.0;
  ChannelValue[8] = Kh * ChannelValue[8] / 1E6 * Q * rhoNOx * 1000.0;

  return 0;
}

/* Subprogram to input ambient conditions before a test */

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

/* extern int NOXDelay, CODelay, HCDelay;*/

void conditns(float *AmbCond)
{
 char choice;
 do{
               clrscr();

               puts("\r\n");
```

```
            puts("Emissions and Engine Data Acquisitions\r\n");

            printf("Enter the ambient temperature (C) : ");
            scanf("%f", &AmbCond[0]);
            printf("Enter the relative humidity (%) : ");
            scanf("%f", &AmbCond[1]);
            printf("Enter the ambient pressure (mm Hg) : ");
            scanf("%f", &AmbCond[2]);
            printf("\n\n\n");
            printf("Ambient Temperature: %2.1f \n", AmbCond[0]);
            printf("Relative Humidity  : %2.1f \n", AmbCond[1]);
            printf("Ambient Pressure   : %3.1f \n", AmbCond[2]);
            printf("\n Are these values correct? <Y> or <N> ");
            choice = toupper(getch());


  }while(choice != 'Y');
}

/* This subprogram acquires the background data from the tunnel before the
engine is started */

#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <ctype.h>

void setupadc(int datadd, int cmdadd);

backgrnd(char **ChannelTitle, int datadd, int cmdadd)
{
            char choice;
            int Channum;
            int counter;
            int chan;
            int DataPoint;
            int i;
            long int total;
            long int AnalyzerADC[8];
            FILE *file_ptr;

            choice = 'x';
            printf("\n\n\n\n\r");
            printf("To take Background Emission level data, enter <Y> now.\n\r");
            printf("\n\r");
            puts("Press any other key to continue...");
            choice = toupper(getch());
            if (choice == 'Y') {
                        setupadc(datadd, cmdadd);
                        do{
                                    clrscr();
                                    printf("Press SpaceBar After Steady State");

                        }while(!kbhit());
```

```
                clrscr();
                printf("\n\r Taking background data...");
                counter = 0;
        /* DAS-16 MUX reads one channel at a time.  Reads HEX00 through HEX33*/
                for (chan = 0; chan <= 51; chan += 17){
                                outport(0x322, chan);
                                Channum=chan/17;
                                printf("\nChannel %d, %s", Channum, ChannelTitle[Channum+5]);
                                total=0;

                                for (i=1; i<=100; i++){
                                        outp(0x320, 0);
                                        while(inp(0x328) & 0x80){
                                 }
                                        DataPoint= inp(0x320)/16 + inp(0x321)*16;
                                        total = total + DataPoint;
                                }

                                AnalyzerADC[counter] = total/100;
                                printf("  :  %d",AnalyzerADC[counter]);
                                counter++;

                }
                file_ptr = fopen("C:\\DATA\\CAL\\Backgrnd.dat", "w");
                for (i=0; i<=3; i++){
                                fprintf(file_ptr, "%d \r\n",AnalyzerADC[i]);
                }
                fclose(file_ptr);

        }
        return 0;
}

/* This subprogram sets up the boards for acquisition */
#include <dos.h>

setupadc(int datadd, int cmdadd)
{
        outp(0x329, 0);  /*Sets DAS-1602 to internal software trigger. */
        outp(0x32B, 1);  /*Sets DAS-1602 to -5 to +5 Volts */


        /* reset STC and set to 8-bit bus */

        outp(cmdadd, 0xFF);
        outp(cmdadd, 0xE7);

        /* set MM register to decade */

        outp(cmdadd, 0x17);
        outp(datadd, 0xB0);
        outp(datadd, 0x8A);
```

```
/* counter 1 measures MAP pulse period */

outp(cmdadd, 0x1);  /*set counter 1 to mode Q*/
outp(datadd, 0xA8);
outp(datadd, 0xCB);

/* counter 2 measures Advance */

outp(cmdadd, 0x2); /*set counter 2 to mode Q */
outp(datadd, 0xA8);
outp(datadd, 0x8B); /* set gate to 1 microsecond intervals */

/* counter 3 measures FIPW via the IDM Feedback signal */

outp(cmdadd,0x3);   /*set counter 3 to mode Q*/
outp(datadd,0xA8);
outp(datadd,0x8B);  /*Sourced from F1, gated with Gate3, 1 microsecond intervals */

/* counter 4 measures engine speed via the CYL ID signal*/

outp(cmdadd, 0x4);  /* set counter 4 to mode D */
outp(datadd, 0xA8);
outp(datadd, 0x8C);  /* set gate to 10 microsecond intervals */

/* counter 5 - NOT USED */

outp(cmdadd, 0x5);  /*set counter 5 to mode D */
outp(datadd, 0xA8);
outp(datadd, 0xCB); /* sourced from SRC3 */

/* set all load registers to zero */

outp(cmdadd, 0x9);
outp(datadd, 0x0);
outp(datadd, 0x0);

outp(cmdadd, 0xA);
outp(datadd, 0x0);
outp(datadd, 0x0);

outp(cmdadd, 0xB);
outp(datadd, 0x0);
outp(datadd, 0x0);

outp(cmdadd, 0xC);
outp(datadd, 0x0);
outp(datadd, 0x0);

outp(cmdadd, 0xD);
outp(datadd, 0x0);
outp(datadd, 0x0);

/* Load and arm counters */
```

```
            outp(cmdadd, 0x7F);

            return 0;
}

/*  This subprogram prints the text values of each channel to the screen. */
#include <conio.h>
#include <stdio.h>

screen(char **ChannelTitle, double *ChannelValue, float *PressureInfo)
{

 clrscr();

 printf("%s   = %3.1f  kPa\n", ChannelTitle[0],ChannelValue[0]);
 printf("%s   = %2.1f  BTDC\n", ChannelTitle[1],ChannelValue[1]);
 printf("%s   = %4.0f  ms\n", ChannelTitle[2],ChannelValue[2]);
 printf("%s   = %4.1f  RPM\n", ChannelTitle[3],ChannelValue[3]);
 printf("%s   = %4.4f  g/s\n", ChannelTitle[5],ChannelValue[5]);
 printf("%s   = %4.4f  g/s\n", ChannelTitle[6],ChannelValue[6]);
 printf("%s   = %4.4f  g/s\n", ChannelTitle[7],ChannelValue[7]);
 printf("%s   = %4.4f  g/s\n", ChannelTitle[8],ChannelValue[8]);
 printf("%s   = %3.1f  K\n", ChannelTitle[9],ChannelValue[9]);
 printf("%s   = %3.1f  kPa\n", ChannelTitle[10],ChannelValue[10]);
 printf("%s   = %3.1f  Nm\n", ChannelTitle[11],ChannelValue[11]);
 printf("%s   = %3.1f  C\n", ChannelTitle[12],ChannelValue[12]);
 printf("%s   = %3.1f  C\n", ChannelTitle[13],ChannelValue[13]);
 printf("%s   = %4.1f  C\n", ChannelTitle[14],ChannelValue[14]);
 printf("%s   = %3.1f  C\n", ChannelTitle[15],ChannelValue[15]);
 printf("%s   = %2.2f  n/a\n", ChannelTitle[16],ChannelValue[16]);
 printf("%s   = %2.2f  %\n", ChannelTitle[17],ChannelValue[17]);
 printf("%s   = %4.0f  ADC\n", ChannelTitle[18],ChannelValue[18]);
 printf("%s   = %4.0f  ADC\n", ChannelTitle[19],ChannelValue[19]);
 printf("%s   = %4.0f  ADC\n", ChannelTitle[20],ChannelValue[20]);
 printf("\nPeak Pressure = %5.2f   kPa\n", PressureInfo[6]);
 printf("IMEPg = %4.2f   kPa\r\n", PressureInfo[7]);


return 0;
}

#include <stdio.h>
#include <conio.h>

void readdsp(float *PressureInfo, float MAP)
{
            unsigned int value;
            unsigned long totaljunk;
            float *realjunk;
            unsigned int mapint;



/*          outport(0x240+0x0014, 0x0002);  /*Initialize host Read/Write registers*/
/*          outport(0x0240+0x0014, 0x0004);*/

mapint = (MAP - 90)*65535/300;

                    do{
                                outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    PressureInfo[0] = ((int) value ) *(360.0/1024.0);


                    do{
                                outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    PressureInfo[1] = ((int) value )*(360.0/1024.0);


                    do{
                                outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    PressureInfo[2] = ((int) value - 512)*(360.0/1024.0);

                    do{
                                outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    PressureInfo[3] = ((int) value - 512 )*(360.0/1024.0);

                    do{
                                outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    PressureInfo[4] = ((int) value - 512 )*(360.0/1024.0);

                    do{
                                outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    PressureInfo[5] = ((int) value - 512 )*(360.0/1024.0);

          /*  Read and display a floating point number  */
                    do{
                                          outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
```

```
totaljunk = value;
totaljunk <<= 16;
totaljunk &= 0xFFFF0000;

do{
                outport(0x0240+0x0014, 0x6044);
}while((inport(0x0240+0x0400)& 0x2)==0);
outport(0x240+0x0014, 0x0002);
value = inport(0x0240+0x0808);
totaljunk += value;
realjunk = (float * ) &totaljunk;
PressureInfo[6] = *realjunk;


do{
                outport(0x0240+0x0014, 0x6044);
}while((inport(0x0240+0x0400)& 0x2)==0);
outport(0x240+0x0014, 0x0002);
value = inport(0x0240+0x0808);
totaljunk = value;
totaljunk <<= 16;
totaljunk &= 0xFFFF0000;
do{
                outport(0x0240+0x0014, 0x6044);
}while((inport(0x0240+0x0400)& 0x2)==0);
outport(0x240+0x0014, 0x0002);
value = inport(0x0240+0x0808);
totaljunk += value;
realjunk = (float * ) &totaljunk;
PressureInfo[7] = *realjunk;


do{
                outport(0x0240+0x0014, 0x6044);
}while((inport(0x0240+0x0400)& 0x2)==0);
outport(0x240+0x0014, 0x0002);
value = inport(0x0240+0x0808);
totaljunk = value;
totaljunk <<= 16;
totaljunk &= 0xFFFF0000;
do{
                outport(0x0240+0x0014, 0x6044);
}while((inport(0x0240+0x0400)& 0x2)==0);
outport(0x240+0x0014, 0x0002);
value = inport(0x0240+0x0808);
totaljunk += value;
realjunk = (float * ) &totaljunk;
PressureInfo[8] = *realjunk;


do{
                outport(0x0240+0x0014, 0x6044);
}while((inport(0x0240+0x0400)& 0x2)==0);
outport(0x240+0x0014, 0x0002);
```

```
                    value = inport(0x0240+0x0808);
                    totaljunk = value;
                    totaljunk <<= 16;
                    totaljunk &= 0xFFFF0000;
                    do{
                                        outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400)& 0x2)==0);
                    outport(0x240+0x0014, 0x0002);
                    value = inport(0x0240+0x0808);
                    totaljunk += value;
                    realjunk = (float * ) &totaljunk;
                    PressureInfo[9] = *realjunk;


/*  The following 5 lines ensure a synchronized write to the COM_DATA port */
                    outport(0x0240+0x0014, 0x0004);
                    outport(0x240+0x0808, mapint);
                    do{
                              outport(0x0240+0x0014, 0x6044);
                    }while((inport(0x0240+0x0400) & 0x0004) == 0);



          return;
}
```

Appendix F: Network Application Program Source Code

```
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <math.h>
#include <io.h>
#include <conio.h>

void fire_hc(double *Net_Pass, double *HC);
void fire_nox(double *Net_Pass, double *NOx);
void fire_co(double *Net_Pass, double *CO);
void fire_co2(double *Net_Pass, double *CO2);

void main(void){

  struct dsp_struct {
              char inf[14], rel[14];
  } dsp_file[8];

  int i, j,total_files;
  int k, m;
  struct find_t ffblk;
  double HC[1], NOx[1];
  double CO[1], CO2[1];
  int done;
  char savestring[8];
  char trash[80];
  char dirstring[25] = "D:\\dspdata\\1106_dsp\\";
  float ChannelValue[21];
  float Press_Data[10];
  double Net_Pass[8];

  FILE *data_file;
  FILE *save_file;

  i=0;
  done = _dos_findfirst("d:\\dspdata\\1106_dsp\\*.rel",_A_NORMAL,&ffblk);
  strcpy(dsp_file[i].rel, ffblk.name);

  while (!done) {
              i++;
              done = _dos_findnext(&ffblk);
              strcpy(dsp_file[i].rel, ffblk.name);
  }


  total_files = i;

  for(i=0;i<total_files;i++){


              strcpy(dirstring,"D:\\dspdata\\1106_dsp\\");
              strcat(dirstring, dsp_file[i].rel);
              data_file = fopen(dirstring, "r");
              printf("Data_File = %s\n", dirstring);
```

```
            strncpy(savestring, dsp_file[i].rel, 8);
            savestring[8] = '\0';
            strcpy(dirstring,"D:\\dspdata\\1106_dsp\\");
            strcat(dirstring,savestring);
            strcat(dirstring,".net");
            save_file = fopen(dirstring, "w");
            printf("Applying nets to  : %s\n", dirstring);
            fprintf(save_file, "HC(Pred), HC(Act), CO(Pred), CO(Act), CO2(Pred), CO2(Act), NOx(Pred),
NOx(Act)\n");
            for(k=0;k<=1;k++){
                        for (m=0;m<=30;m++){
                              fscanf(data_file, "%s", &trash);
                        }
            }

            do{

                  for (j=0;j<=20;j++){
                              fscanf(data_file,"%f",&ChannelValue[j]);
                              fscanf(data_file,"%1s",&trash);
                  }
                  for (j=0;j<=8;j++){
                              fscanf(data_file,"%f",&Press_Data[j]);
                              fscanf(data_file,"%1s",&trash);
                  }
                  fscanf(data_file,"%f",&Press_Data[9]);

if (Press_Data[7] >= -75){

 /* outarray[0] is HC */
/* inarray[0] is Peak-1 */
/* inarray[1] is IgnDel-1 */
/* inarray[2] is LMFB50-1 */

                  Net_Pass[0] = (double) Press_Data[6];
                  Net_Pass[1] = (double) Press_Data[4];
                  Net_Pass[2] = (double) Press_Data[5];
/*                Net_Pass[3] = (double) Press_Data[5];
/*                Net_Pass[4] = (double) Press_Data[5];
                  Net_Pass[5] = (double) Press_Data[0];*/
                  fire_hc(Net_Pass, HC);

 /* outarray[0] is NOx */
/* inarray[0] is Peak-1 */
/* inarray[1] is IMEPg-1 */
/* inarray[2] is IgnDel-1 */
/* inarray[3] is CombDur-1 */
/* inarray[4] is LMFB50-1 */

                  Net_Pass[0] = (double) Press_Data[6];
                  Net_Pass[1] = (double) Press_Data[7];
                  Net_Pass[2] = (double) Press_Data[4];
```

```
                        Net_Pass[3] = (double) Press_Data[1];
                        Net_Pass[4] = (double) Press_Data[5];
/*                      Net_Pass[5] = (double) Press_Data[8];*/
                        fire_nox(Net_Pass, NOx);

/* outarray[0] is CO */
/* inarray[0] is Peak-1 */
/* inarray[1] is LMFB50-1 */
/* inarray[2] is MaxBurnLoc-1 */

                        Net_Pass[0] = (double) Press_Data[6];
                        Net_Pass[1] = (double) Press_Data[5];
                        Net_Pass[2] = (double) Press_Data[2];
/*                      Net_Pass[3] = (double) Press_Data[2];
/*                      Net_Pass[4] = (double) Press_Data[0];
                        Net_Pass[5] = (double) Press_Data[8];*/
                        fire_co(Net_Pass, CO);

/* outarray[0] is CO2 */
/* inarray[0] is Peak-1 */
/* inarray[1] is IMEPg-1 */
/* inarray[2] is IgnDel-1 */
/* inarray[3] is MaxBurn-1 */
                        Net_Pass[0] = (double) Press_Data[6];
                        Net_Pass[1] = (double) Press_Data[7];
                        Net_Pass[2] = (double) Press_Data[4];
                        Net_Pass[3] = (double) Press_Data[9];
/*                      Net_Pass[4] = (double) Press_Data[9];
/*                      Net_Pass[5] = (double) Press_Data[2];*/
                        fire_co2(Net_Pass, CO2);
}
else {
                       HC[0] = 0.0;
                       CO[0] = 0.0;
                       CO2[0] = 0.5;
                       NOx[0] = 0.007;
}
fprintf(save_file, "%5.5f , %5.5f, %5.5f, %5.5f, %5.5f, %5.5f, %5.5f, %5.5f\n", HC[0], ChannelValue[5],
CO[0], ChannelValue[6], CO2[0], ChannelValue[7], NOx[0], ChannelValue[8]  );

          } while (!feof(data_file));

          fclose(data_file);
          fclose(save_file);
  }


xxxx:
}
```