
An Intelligent Automobile Diagnostic System

by

Marcus Kramer

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

February 7, 1996

Copyright 1996 Marcus Kramer. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author _____
Department of Electrical Engineering and Computer Science
January 24, 1996

Certified by _____
STEPHEN R. DUNN
Technical Director of the Biomedical Engineering Center

Accepted by _____

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

JUN 11 1996

Barker Eng

LIBRARIES

(Intentionally left blank)

An Intelligent Automobile Diagnostic System

by
Marcus Kramer

Submitted to the
Department of Electrical Engineering and Computer Science

February 7, 1996

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science.

Abstract

The Intelligent Automobile Diagnostic System is designed to be an inexpensive, flexible, diagnostic tool for automobile control systems. The diagnostic tool consists of a portable PC, an interface module, and specialized system software utilizing fuzzy logic. The interface module connects between the parallel data port of the laptop PC and the vehicle's existing ECM cable harness. The software is designed to control the data acquisition process and to perform the diagnostic analysis of the collected data. While monitoring all of the desired sensors in the vehicle during a test-drive, the diagnostic system determines if any of the automobile's electrical or mechanical systems are malfunctioning. The system performs this diagnosis by utilizing a fuzzy expert system to first model all of the monitored sensors in the vehicle. The fuzzy expert system then diagnoses possible problems with the vehicle by comparing the actual recorded data to the system's modeled data. Discrepancies between the two data sets, along with the status of the current operating conditions of the vehicle, form the basis of the system's diagnostic process. The final output of the diagnostic tool includes a list of the top few possible problems with the vehicle, along with a calculated certainty level associated with each possibility.

Thesis Advisor: Stephen K. Burns

Title: Technical Director of the Biomedical Engineering Center

Acknowledgments

Thanks to Mom and Dad for supporting my education, and thanks to Professor Burns for his valuable insight and suggestions.

Table of Contents

	Page #
1 Introduction	8
1.1 Motivation	9
1.2 Objectives for the Project	10
1.3 Background	12
1.3.1 Other Fuzzy Logic Applications	12
1.3.2 Existing Automobile Diagnostic Systems	13
1.4 Thesis Organization	15
2 Automobile Background	16
2.1 Basic Vehicle Operation	16
2.1.1 Internal Combustion Engine	16
2.1.2 Transmission	18
2.2 Electronic Control Module	19
2.3 Description of the Electronic Sensors	19
2.3.1 Exhaust Oxygen Sensor	20
2.3.2 Engine Coolant Temperature Sensor	20
2.3.3 Throttle Position Sensor	21
2.3.4 Manifold Air Temperature Sensor	21
2.3.5 Manifold Absolute Pressure Sensor	21
2.3.6 Engine Speed Sensor	22
2.3.7 Vehicle Speed Sensor	22
3 Hardware	23
3.1 The Diagnostic Connection	23
3.2 Design of the Interface Circuitry	24
3.2.1 The Hardware Components	24
3.2.2 Hardware Cost Issues	26
3.3 The PC Parallel Port	27
4 Software	29
4.1 The Data Acquisition Module	29
4.1.1 Low-Level Parallel Port Issues	29
4.1.2 High-Level Acquisition Control	30
4.2 Collect Sample Data Sets	31
5 Fuzzy Logic	32
5.1 Fuzzy Logic Background	32
5.1.1 Fuzzification	33
5.1.2 Inference	38

5.1.3 Composition	41
5.1.4 Defuzzification	42
5.2 Writing Fuzzy Logic Rules	44
5.3 Fuzzy Logic Implementation	45
5.3.1 Phase 1-2: Parameter Preparation	46
5.3.2 Phase 3: Difference	48
5.3.3 Phase 4: Diagnostic	50
6 Results and Observations	54
6.1 Data Recording	54
6.2 Three Diagnostic Examples	55
6.2.1 Modified MAP Test	57
6.2.2 Engine RPM Example	60
6.2.3 Engine Temperature Test	63
7 Conclusion	67
7.1 Meeting the Objectives	67
7.2 Suggested Improvements	69
7.3 Summary	71
8 References	72

List Of Figures

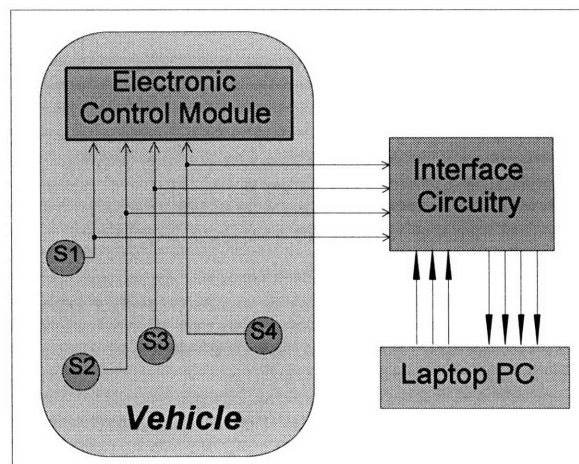
	Page #
Figure 1 Diagnostic system block diagram	8
Figure 3.a Diagnostic system block diagram	24
Figure 3.b Schematic diagram for the interface module	26
Figure 3.c Parallel port register assignments	27
Figure 5.a Linear representation of a temperature scale	34
Figure 5.b Linear temperature scale with a 212°F input	34
Figure 5.c The <i>MEDIUM</i> fuzzy subset	35
Figure 5.d The <i>MEDIUM</i> fuzzy subset with a 212°F input	35
Figure 5.e A complete fuzzy set for the engine temperature range	36
Figure 5.f A Non-uniform fuzzy set	37
Figure 5.g The fuzzy temperature set with an input of 205°F	37
Figure 5.h Rule#17 applied to the RPM fuzzy set	40
Figure 5.i Rule#17 and rule#18 applied to the RPM fuzzy set	41
Figure 5.j Defuzzification centroid calculation	42
Figure 5.k The three parameters used in the centroid calculation	43
Figure 5.l Defuzzification calculation of rules#17 and #18	43
Figure 5.m Two dimensional fuzzy rule chart	44
Figure 5.n Fuzzy logic implementation block diagram	45
Figure 5.o Two sample fuzzy sets defined in the rule base	46
Figure 5.p Difference fuzzy sets for the oxygen and MAP sensors	49
Figure 5.q The Diagnostic Certainty fuzzy set	52
Figure 5.r Sample system diagnostic output summary	53
Figure 6.a Graph of a sample data set recorded over a 60 second interval	57
Figure 6.b Data graph with a modified MAP signal	58
Figure 6.c Diagnostic output for the MAP test	59
Figure 6.d Graph of the second sample data set	60
Figure 6.e Graph of the second data set with a modified RPM signal	61
Figure 6.f Diagnostic output for the RPM example	61
Figure 6.g Graph of engine temperature over a 5 minute interval	64
Figure 6.h Diagnostic output for the engine temperature analysis	65

Chapter 1 Introduction

Traditionally, automobile manufacturers have denied the average car-owner the ability to easily understand what is going on inside their car. The only trouble indicator that the driver is provided with, is a single dummy light that can be activated by any one of many different possible problems. Extracting even slightly more detailed information from the vehicle is a fairly involved process for the average car owner. This lack of available information for the driver was one of the motivations for developing the *Intelligent Automobile Diagnostic System (IADS)*. IADS is a diagnostic tool for automobile control systems that is designed to be easy to use, comprehensive, inexpensive, and flexible. The diagnostic tool consists of a laptop PC¹, an interface module, and specialized system software utilizing fuzzy logic. IADS can be easily connected to the vehicle's computer. It simply utilizes a 'T' connector to tap into all of the desired signals among those that enter the vehicle's *electronic control module (ECM)*. This cable is then fed into the interface module, which is connected to the laptop PC's parallel port as shown in **figure(1)**.

FIGURE 1

Simplified system block diagram showing the connection of the diagnostic tool to a test vehicle.



The system software is designed to control the data acquisition process and perform the diagnostic analysis of the collected data. While

¹ The diagnostic system can use any existing IBM compatible PC (80x86).

monitoring all of the desired sensors in the vehicle, the diagnostic system determines if any of the automobile's electrical or mechanical systems are malfunctioning. The system performs this diagnosis by using a fuzzy expert system to create a model of what the sensors should be reading under various driving conditions. The fuzzy expert system then performs a diagnosis of the vehicle by comparing the actual data recorded to the system's modeled data. Discrepancies between the two data sets, as well as the current operating conditions of the vehicle, are the basis of the IADS's final diagnostic process. The final output of the system includes a list of the vehicle's possible problems along with a calculated certainty level associated with each suggested possibility.

1.1 Motivation

The primary motivation for this project came from my frustration with trying to figure out what was going on inside my car when it wasn't performing properly. In some cases, if the problem is severe enough so that the vehicle's computer (ECM) can detect a fault, then it activates a dummy-light² on the instrument panel. However, the same dummy-light is activated regardless of the specific problem. This leaves most car owners at the mercy of automobile mechanics, since the light means virtually nothing to the driver (other than that it is time to take the car to someone who can somehow figure out what it means).

It is common for people to simply ignore the dummy-light, since their car is still at least somewhat functional, and they can't afford repairs that aren't critical. Another reason to disregard the dummy-light is that people don't want to bother taking their car in for repair without even having a vague idea of what caused the dummy light to trigger. With some degree of effort and knowledge, the driver can locate the ALDL connector³ on their vehicle so that they can use it to enable the vehicle's

² The dummy light is usually labeled "Service Engine Soon."

³ The ALDL (Assembly Line Diagnostic Link) is a electrical connector used for vehicle diagnostics while the car is still being assembled. It also connects to various after-sales diagnostic and service equipment. The link provides serial data from the vehicle's ECM, and is usually hidden somewhere under the dashboard.

Field Service Mode⁴. The ECM then flashes the dummy light in a specific pattern that corresponds to one of about 40 possible system trouble codes. If the driver owns a repair manual for their vehicle, they can decipher the code to see which system caused the ECM to report an error. It is probably reasonable to assume that most drivers have no idea how to perform this crude system check, so, again, they are left at the mercy of an auto mechanic to tell them what they think is wrong with the vehicle.

There are some test accessories that are available which allow vehicle owners to connect small electronic units to the ALDL. These devices are usually called Scan tools, and they literally “scan” the current state of all of the vehicle’s sensors. However, this tool is not capable of performing any sort of diagnostic function; it simply displays the instantaneous values of each of the sensors--one at a time.

In order to perform any sort of diagnostic procedure on a vehicle, the owner would have to take his car to a mechanic that owns one of the large, expensive diagnostic systems. Most people are probably not willing to take their car in for an expensive diagnosis if they are not sure if there is a serious problem with their car. So, one of the main objectives for this project was to develop a comprehensive diagnostic system that is easy for non-mechanics to use, and that could be run off of anyone’s existing laptop PC.

1.2 Objectives for the Project

The general objective of this project was to develop an inexpensive, comprehensive, flexible automobile diagnostic system and to demonstrate the system’s ability to accurately diagnose the vehicle’s problems by simply analyzing data from the existing sensors in the vehicle. The system was designed to use detailed knowledge about the

⁴ The Field Service Mode is enabled by shorting a designated test terminal to ground on the ALDL connector (for GM cars).

proper operating conditions for an automobile's control systems in order to determine if one of the systems was not operating properly.

The specific objectives for the project were to:

- Create a diagnostic system that utilizes as many existing parts as possible (e.g. any IBM compatible laptop PC, and the existing vehicle sensors).
- Provide the user with a real-time graphical display that shows the behavior of all of the vehicle's sensors over specific time intervals.
- Provide the user with an inexpensive, informative tool for monitoring their vehicle's performance.
- Make the diagnostic tool relatively easy to operate.
- Enable the system to perform all of the diagnostic functions non-invasively⁵.
- Demonstrate the system's ability to perform an accurate diagnosis using only the existing vehicle sensors.
- Demonstrate the benefits of using fuzzy logic for both the modeling and model based reasoning (diagnosis) in the system.
- Demonstrate the system's ability to perform accurate diagnostics of slowly changing parameters that would normally be very difficult for technicians to monitor.
- Demonstrate that the system offers many benefits over existing diagnostic technologies.

An analysis of how well these objectives were met in the project is presented in the conclusion of the paper (Chapter 7).

⁵ In this case, non-invasively means that no additional transducers were connected to the vehicle's engine. Most of the existing diagnostic systems used by mechanics are equipped with a wide variety of specialized probes and transducers that are meant to monitor very specific subsystems.

1.3 Background

The heart of this project was the challenge of providing the system with the capability of making valid diagnostic decisions based on multiple, interdependent analog inputs. This type of diagnosis is a common challenge in many modern complex electro-mechanical systems. The ability to deal with multiple, time-variant inputs is also extremely useful in complex closed loop control systems. Several fuzzy expert systems have already been utilized in industrial control applications where they needed to make control decisions based on several different analog sensors. Existing technologies often simply monitor specified threshold levels in order to determine if a subsystem is faulty. Other, more sophisticated, systems utilize complex mathematical models to monitor a system's performance. These types of systems are very difficult to implement, since accurate mathematical models have to be created, which is not an easy task for a complex system.

It would be desirable to have a relatively simple, reliable, and flexible method for implementing a monitoring or diagnostic system. Fuzzy logic is an ideal choice for this type of application. Once the fuzzy logic engine is in place, it is fairly simple to write rules for a fuzzy expert system to model and diagnose the system using natural language.

1.3.1 Other Fuzzy Logic Applications

Some of the projects where fuzzy logic has been successfully applied include, anti-lock braking systems for cars and trucks, auto-focus cameras, efficient air conditioning control systems, financial modeling, and active suspension systems for automobiles.

Liebert Corporation of Columbus, Ohio (specialists in environmental control systems) designed a fuzzy logic control unit for air conditioning systems. The system offers many improvements over conventional air conditioning control systems. The system can perform precision control of temperature and humidity, while optimizing many other factors involved with the air conditioning system, such as: 1) minimizing the cycling times, which reduces wear and increases reliability, and 2)

intelligently utilizing the outside air to assist the cooling or heating process depending on the time of day. Fuzzy logic closed-loop control systems are probably the most common implementation of fuzzy logic in industry.

Japanese research groups are avidly investigating many different types of applications for fuzzy logic. One such application is a foreign exchange trading system, that incorporates influencing factors from a variety of sources. This system predicts the exchange rate of the yen against the US dollar by not only using information about current market conditions, but by also including information about major political events, economic events, and national political policy. [McNeil, Freiburger]

An industrial application that Japanese researchers are working on is a fuzzy logic based power plant management system. The system is designed to detect early warning signs that could indicate possible accidents at the plant--especially warning signs and patterns that might not be noticeable by human monitors. [McNeil, Freiburger] This type of system can not only utilize a knowledge base that is provided by an expert, but it can also reference detailed data from past experiences that may assist it in detecting a related failure.

This particular thesis project, however, is meant to demonstrate that fuzzy logic utilizing an expert system can be a useful tool for making *diagnostic* decisions based on analog sensor data.

1.3.2 Existing Automobile Diagnostic Systems

Most diagnostic systems that are currently available are either common PC based systems, or specialized adaptations of PC systems. The diagnostic systems are usually highly specialized since they have to be used in a harsh environment, so they are ruggedized in several ways. Very few of the existing diagnostic tools are portable; most of them are built into large, industrial wheeled carts, and are designed to be used directly in an auto shop. There are some systems that have a satellite unit that can plug into the vehicle to record data while the car is being

driven out of the shop. The satellite units usually have no diagnostic capabilities, so their data has to be downloaded to the main diagnostic system back at the shop.

The majority of these diagnostic systems connect to the ECM via the ALDL serial connector. Since the vehicle's ECM broadcasts the current values of each of the car's sensors over the serial link, the diagnostic equipment can read in all of this available data as it is presented. Although this serves as a relatively simple interface to the ECM, it does not necessarily provide an extremely reliable diagnosis of the system. This type of diagnostic connection assumes that all of the data that the ECM broadcasts is correct, however, if the ECM is itself malfunctioning or unable to detect a subsystem failure, the data may be inaccurate. In this case, the diagnostic system may not be able to detect a faulty sensor, since the diagnostic system is relying on the ECM's output instead of the sensor output.

There are, however, some diagnostic systems that connect to all of the data and control I/O signals of the ECM by tapping into the connectors that go directly into the ECM. This type of diagnostic tool can make its own readings of all of the sensors, and it can easily monitor the control behavior of the ECM to see if it is malfunctioning. For this thesis, the system connects directly to some of the ECM inputs. The diagnostic system can then take its own readings of the vehicle sensors at very high rates without relying on the accuracy of the ECM.

Another type of diagnostic system that does not connect to the vehicle is an interactive expert system coupled with a detailed data base of specific vehicles. This type of system engages a technician in a question-answer dialogue in order to narrow down the possible problem with the faulty vehicle. The system asks several questions about the vehicle's symptoms, such as:

- "Does the problem occur when the engine is hot?"
- "Is this an intermittent problem?"
- "Has the vehicle ever had this problem repaired in the past?"

This type of diagnostic approach can learn new information whenever each vehicle is repaired, by updating its expert system with all of the initial symptoms and with the outcome of the repair.

Yet another implementation of vehicle diagnostics is an assembly line diagnostic system that is designed to test each of the electronic sensors and subsystems as they are installed in the vehicle. This can correct immediate problems, and can also detect minor malfunctions that may cause future problems.

1.4 Thesis Organization

The thesis paper is divided into six sections.

- The first section, Chapter 2, provides background information about automobiles that is important to know in order to completely understand how the diagnostic system operates.
- Chapter 3 deals with the interface hardware and software issues.
- Chapter 4 explains the fundamentals of fuzzy logic.
- Then Chapter 5 describes how fuzzy logic was applied in both the modeling and diagnostic phases of this particular project.
- Chapter 6 provides several example diagnostics, and explains their results in detail.
- Chapter 7 discusses how well the objectives were met, and provides suggestions for improving the system.

Chapter 2 Automobile Background

In order to completely understand the examples and results that are discussed later in the paper, it is necessary to have a general understanding of how most vehicle's operate.

2.1 Basic Vehicle Operation

Modern automobiles are controlled by a combination of mechanical and electrical inputs. The driver's main inputs are the throttle, brakes, gear selection (in vehicles with manual transmission), and steering. Based primarily on the driver's demand on the throttle, the vehicle's electronic control module (ECM) creates all of the remaining necessary signals to control the vehicle's subsystems.

The two major mechanical components of a vehicle are the engine and drivetrain⁶. The engine produces the power for propelling the vehicle and for many of the vehicle's subsystems⁷, while the drivetrain is primarily responsible for transferring that power to the vehicle's wheels. The majority of the sensors in an automobile monitor different engine functions, so the diagnostic system is primarily designed to detect engine malfunctions--although some engine-related symptoms actually indicate problems with some of the vehicle's major subsystems.

2.1.1 Internal Combustion Engine

The engine in a vehicle is clearly the most important and complex component of the vehicle. This is why there are so many systems responsible for continuously monitoring various engine parameters.

⁶ The drive-train includes the transmission, differential, and the remaining couplings that transmit the power from the engine to the vehicle's wheels.

⁷ The engine is also used to power the air conditioning compressor, power steering pump, power brakes, coolant fluid pump, etc. Most of these subsystems are coupled to the engine's output shaft by belts, gears, or chains.

Automobile engines are known as internal-combustion engines. They literally utilize chemical explosions to produce mechanical power. The explosions occur inside cylinders in the engine. Each cylinder has a piston, and each piston is linked to the engine's output shaft. An explosion inside one of the engine's cylinders forces the piston out, which in-turn causes the output shaft to rotate. When each of the cylinders⁸ produce explosions in a specific sequence, the output shaft rotates continuously with relatively constant force.

The explosions in the engine cylinders are created by combining air and fuel in a well controlled mixture. Generally, a ratio of 14.7:1 for air:fuel mixture is the ideal ratio for efficient combustion⁹. Different types of engines have different methods of mixing the air and fuel, but all *four-stroke* engines use the same general principal¹⁰. The four strokes are known as the intake, compression, power, and exhaust strokes. On the first stroke, the downward movement of the piston sucks air and vaporized fuel into the cylinder chamber through an opened valve. Then once the valve closes, the second stroke begins: the piston moves up to compress the air-fuel mixture at a ratio of approximately 9:1. Once the piston reaches the top of its travel stroke, the spark plug in the cylinder is activated to ignite the fuel, which causes the gas to rapidly expand, forcing the piston back down on its third stroke. Once the piston reaches the bottom of its stroke, a second valve is opened in the top of the cylinder so that the upward movement of the piston on its fourth stroke expels all of the exhaust gas created by the combustion. Now the cycle starts all over again with the downward movement of the piston sucking in more air and fuel for the intake stroke.

⁸ There are typically 4 or 6 cylinders in most common automobiles.

⁹ Under special circumstances it is desirable to slightly alter this ratio for relatively short periods of time.

¹⁰ Carburetor equipped engines suck the fuel-air mixtures into the cylinders; Throttle Body Injection systems have a single fuel injector that vaporizes fuel in a central chamber so that the fuel-air mixture can be sucked into any of the cylinders; and Multi-Point Fuel Injection systems have one injector per cylinder, and they inject fuel either into the combustion chamber, or near the intake valve area for the cylinder.

The engine includes sensors that monitor:

- The temperature of the intake air
- The mass of the intake air
- The absolute air pressure in the intake manifold
- The engine coolant temperature
- The amount of oxygen in the exhaust gas
- The engine speed
- The engine shaft's rotational position

2.1.2 Transmission

The engine's output shaft is coupled to the vehicle's drive-wheels through either an automatic or manual transmission. An automatic transmission uses a type of oil (automatic transmission fluid) to actually couple the engine shaft to final drive shaft. The engine shaft has a rotary pump at its end, and the drive shaft has a turbine at its shaft end. These are both housed in the same sealed chamber with the transmission fluid. The coupling is essentially achieved through the rapid swirling of the transmission fluid by the engine's shaft end. The fluid's movement then creates a rotational force on the transmission shaft's turbine, which causes it to rotate as well. The fluid acts as a damped coupling between the two shafts. This coupling allows the transmission to remain engaged, while the engine is running, and the vehicle is standing still. The engine shaft is still swirling the fluid in the coupling chamber, but the vehicle's braking system essentially holds the drive shaft still.

In a manual transmission, the coupling between the engine and the driveshaft is directly achieved through the gears. When a vehicle with manual transmission comes to a stop, the clutch has to be depressed in order to disengage the transmission's coupling to the engine, so that the engine can continue to rotate while the drive-shaft stops.

2.2 Electronic Control Module

Most vehicles built since the early 1980's have included some type of electronic control module (ECM) that is in charge of controlling specific engine functions primarily to minimize the vehicle's emissions while optimizing performance. The ECM actually controls the fuel delivery based on several parameters such as the driver demand input via the throttle position sensor (TPS). Engine temperature and intake air temperature also significantly influence the amount of fuel delivered by the ECM, while other less-significant factors are also considered. Federal emission regulations actually brought about the introduction of an ECM into all vehicles in the early 1980's. These standards become more stringent every few years, which requires enhancement of the electronic control and sensing capabilities of the vehicle's ECM. Some of the latest required enhancements to the emission control system include: additional monitoring of the exhaust gas *after* it passes through the vehicle's catalytic converter¹¹, and a fuel system pressure check prior to ignition, which is designed to detect small leaks in the fuel-vapor system. In order to both minimize emissions and maintain high performance, automobile engines are equipped with several different types of transducers that monitor critical engine functions.

2.3 Description of the Electronic Sensors

While some of a vehicle's sensors are passive, and some are active, almost all of them output voltages between 0-5vDC. Some of the sensors also only have one wire connecting them to the ECM¹², since they use the vehicle's chassis or engine block as their ground lead. This technique helps to reduce the enormous amount of wire installed in modern vehicles. However, some of the more delicate sensors have a separate ground wire that connects directly to the ECM ground in order to minimize noise.

¹¹ The catalytic converter is a muffler-like device that catalyzes a chemical reaction in the exhaust fumes which converts some air pollutants into less harmful substances.

¹² This reason for this single wire design is to minimize the amount of wire in a vehicle. When vehicle's are assembled, the installation of the wiring takes a significant portion of the overall assembly time.

2.3.1 Exhaust Oxygen Sensor

The exhaust oxygen sensor is mounted in the exhaust manifold of the vehicle's engine. The sensor protrudes through the manifold and extends into the stream of exhaust generated by the vehicle. The oxygen sensor is responsible for measuring the amount of oxygen remaining in the exhaust gas which is a direct indication of the air-fuel mixture ratio produced by the fuel injectors. The ideal ratio of the air-fuel mixture is 14.7 to 1. The feedback loop containing the oxygen sensor, throttle position sensor, engine temperature sensor etc. is designed specifically to maintain this ideal ratio. Too much oxygen remaining in the exhaust gas, indicates insufficient fuel supplied by the injectors; this condition is known as a lean mixture. When the remaining oxygen in the exhaust falls too low, the injectors introduced too much fuel, and the condition is then called a rich mixture.

The oxygen sensor produces a voltage output of 0.010 volt for a lean mixture, and a voltage of 0.90 volt for a rich air-fuel mixture. [Ingersol] Since the oxygen sensor is a very sensitive device, and since the exhaust gases are pulsating past the sensor corresponding to the engine's piston movements, the output of the oxygen sensor usually oscillates by a few tenths of a volt. The exact value of the sensor's output is not crucial, as long as the sensor's output doesn't wander too far from the middle reading for an ideal mixture.

2.3.2 Engine Coolant Temperature Sensor

The engine coolant temperature sensor is located in the path of the coolant stream that circulates around and through the engine. The sensor is a thermistor which simply changes its resistance as the temperature of its environment changes. At very high temperatures, the sensor has a very low resistance of approximately 70Ω at 266°F , while at very low temperatures, the resistance rises to over $100,000\Omega$ at -40°F . [Ingersol] When a five volt signal is applied to the thermistor in series with a fixed resistor, the thermistor's change in resistance produces a voltage drop (0-5volts) through the simple voltage divider network.

The engine coolant temperature sensor influences several factors in the engine control system. The most noticeable of which is when the vehicle is first started in a cold climate. The ECM senses that the engine temperature is far below normal operating conditions, so it temporarily increases the normal idle speed in order to quickly warm up the engine to a reasonable operating temperature.

2.3.3 Throttle Position Sensor

The throttle position sensor (TPS) consists of a small lever connected to a simple potentiometer. The lever is also connected to the throttle linkage cable, so that when the accelerator pedal is pressed, the lever is pulled which opens the throttle valve and changes the potentiometer's setting. When a 5volt reference signal is applied to the potentiometer, it outputs a voltage between 0.5volts and 5.0volts. While a 0.5volt output corresponds to an idle condition, a 5.0volt output indicates a wide-open throttle (WOT) condition. The ECM monitors the throttle valve's angle in order to determine an appropriate amount of fuel delivery for the driver's demanded acceleration.

2.3.4 Manifold Air Temperature Sensor

The manifold air temperature sensor (MAT) is also a thermistor similar to the engine coolant sensor. The MAT is mounted in the intake air manifold so that it can measure the temperature of the air that is on its way into the engine. When supplied with at 5.0volt reference signal by the ECM, the MAT also outputs a voltage between 0-5.0volts depending on the air temperature. The temperature of the intake air influences the amount of fuel delivery since warmer air requires less fuel to burn efficiently.

2.3.5 Manifold Absolute Pressure Sensor

The manifold absolute pressure sensor (MAP) measures changes in the intake manifold air pressure. Variations in pressure are primarily caused by load on the engine and changes in engine speed. The MAP sensor is

connected to the intake manifold with a thin hose which filters out high frequency fluctuations in air pressure, so that an average or absolute reading can be made by the sensor. This is an active sensor that usually consists of a small diaphragm strain gauge and some signal amplification circuitry. This device also outputs a voltage of 0-5.0volts that is fed to the ECM. A high voltage indicates low vacuum and high pressure, while a low voltage indicates a high vacuum and low pressure. This manifold pressure also influences the amount of fuel delivered to the engine. A high pressure requires more fuel, because the engine is under higher load and requires increased power.

2.3.6 Engine Speed (RPM) Sensor

Engine RPM can be determined in several different ways. Some systems have a dedicated sensor that monitors the position of the engine's output shaft, while other control systems simply monitor the rate of the ignition pulses. In this implementation, the engine RPM is determined by monitoring the pulse train output from the ignition module. Since each pulse corresponds to a spark in one of the cylinders, and since the vehicle has a 4 cylinder, 4-stroke engine: two pulses represent one engine revolution. When the pulses are counted and timed, the engine's RPM can then be easily determined by dividing the number of pulses by the measured time interval in which they occurred. The ECM primarily uses the RPM information for ignition and fuel injection timing, and to maintain a proper idle speed.

2.3.7 Vehicle Speed Sensor

Vehicle speed sensors (VSS) vary slightly between different types of vehicles, but they primarily consist of a small permanent magnet generator module that is linked to the transmission. The generator module outputs a pulse train which corresponds to 4000pulses per mile. Just as with the RPM sensor, when these pulses are counted and timed, the vehicle speed is easily determined. The ECM usually uses vehicle speed in order to control special transmission related issues.

Chapter 3 Hardware

The automobile that was used to develop and test IADS was a 1987 Buick Skyhawk¹³. The laptop PC used in this project was an IBM-compatible 486 / 33MHz. This type of computer is sufficient for sampling all of the sensors at rates well over 1000Hz, while still displaying real-time graphs of each parameter. The actual diagnostic process that is performed on the collected data, however, is a little slow on this type of machine. It takes the 486/33 about 40seconds to perform the complete diagnostic of a typical set of data recorded over a 1 minute interval. Although a Pentium 100mhz processor can complete the diagnostics for the same data set in only about 4 seconds, the 486/33 processor is still sufficient to perform all of the systems diagnostic functions within a relatively reasonable amount of time.

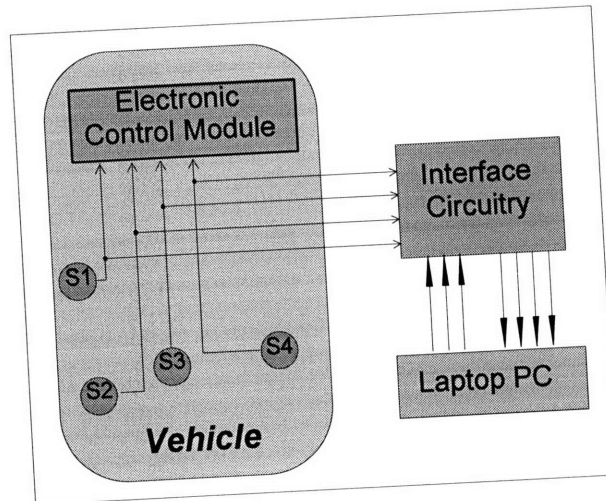
3.1 The Diagnostic Connection

The connection of this system is achieved by tapping into the ECM wire connector harness with a multi-pin T connector. Each sensor is directly wired back to the vehicle's ECM connector, so it is fairly easy to monitor the same signals that the ECM is reading by simply taping into this connector. This method is also consistent with the non-invasive objective of the diagnostic system. The entire system can be connected to the vehicle in only a few minutes, and there are no special sensor probes or additional wire connectors to install for the diagnosis. **Figure (3.a)** illustrates the connection of the diagnostic interface to the vehicle's ECM harness and to the laptop PC.

¹³ The Buick Skyhawk has a 4 cylinder engine, automatic transmission, and most of the common options available on vehicles in its class.

FIGURE 3.a

System block diagram showing the connection of the diagnostic tool to a test vehicle.



The signals from each of the vehicle's sensors are read at specified rates by the interface circuitry. There are also some bi-directional signals between the PC and the interface circuit that are responsible for transferring the converted analog data as well as controlling the data sampling process. Most of the sensors in the vehicle output a voltage between 0-5volts that directly corresponds to the particular parameter that they are designed to read.

3.2 Design of the Interface Circuitry

Analog interface circuitry is required to allow the computer to read the analog signals generated by the vehicle's sensors. So the first stage in the project was to design and construct an interface circuit board that could be easily connected to the vehicle's signal harness.

3.2.1 The Hardware Components

The heart of the interface board consists of a 10-bit analog to digital converter. The remaining circuitry was primarily responsible for signal conditioning and optical isolation in order to protect the vehicle's computer, the vehicle's sensors, and the laptop PC. Because the RPM and Vehicle Speed Sensors generate pulse trains instead of analog

signals, the interface also required two ripple counters to read the signals from these sensors. The specific contents of the interface circuitry consists of the following components:

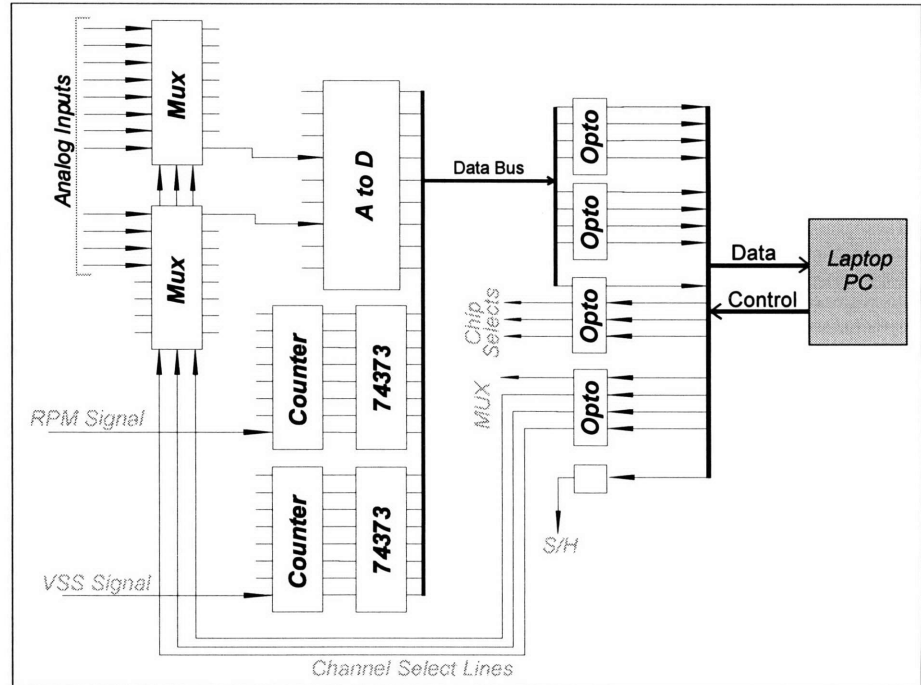
- A 10-bit, four channel analog to digital converter capable of sampling at well-over 100KHz. This ADC was chosen for its low cost, its high speed, and its relatively simple control requirements.
- Two 8-channel analog multiplexers that extend the sampling capability of the ADC to a total of 19 possible analog channels.
- Optical isolation circuitry consisting of an opto-isolator module for each of the 17 total input and output signals from the PC. This isolation is intended to protect both the PC and the vehicle's computer from harmful voltages.
- Two 8-bit ripple counters used to count engine rotations per minute (RPM) and the vehicle speed pulses from the Vehicle Speed Sensor.
- Tri-state bus driving circuitry that enables the PC to read data not only from the A/D converter, but also from the two 8-bit ripple counters, and from the eight digital inputs channels.
- Control logic circuitry that operates the coded chip-select signals and multiplexer addresses.
- An op amp operating as a DC amp to simply amplify one of the low voltage analog input signals from the Oxygen Sensor.
- Two 5volt power supply regulators. Power for the interface board is taken from the vehicle's 12V battery, so the board needs a five-volt regulator for both the digital and analog circuitry on the board.

Since the board is powered by the vehicle's 12V battery, the only other connections that the interface module requires are the parallel port communication connector to the PC, and an analog data connector from the vehicle's computer harness.

A simplified schematic diagram of the IADS interface module is shown in figure(3.b) below.

FIGURE 3.b

Simplified schematic diagram of the system's analog interface module.



3.2.2 Hardware Cost Issues

Part of the objective of this project was to keep the price for this system as low as possible. The relatively simple design of the interface circuit board reflects this cost objective. The total cost of the materials required for the interface module is less than \$100. Because the signals that the system acquires are based on mechanical systems, there is no need to sample these signals at extremely high rates. This specification allows the laptop PC to have complete control over the entire acquisition process without the need for an external microprocessor-based acquisition system. The PC is still capable of sampling each sensor at well over 1000Hz, which is still much faster than necessary for monitoring all of the vehicle's signals. Utilizing a common laptop PC to run the system is another cost effective solution. There is no need to

“reinvent the wheel” by designing specialized diagnostic computer hardware for this application,.

3.3 The PC Parallel Port

There are a few relatively new variations of the traditional parallel port interface that is available on some newer PCs. One of these is known as an Enhanced Parallel Port (EPP). The major enhancement of this new design is that it allows the eight data bits of the port to be bi-directional. However, since almost all PCs are compatible with the traditional Centronics parallel port configuration, and since the laptop PC used in this project was not equipped with an EPP, the control software was limited to dealing with a relatively limited number of input and output signals. The three registers that control the parallel port are illustrated in figure (3.c).

FIGURE 3.c

The three parallel port registers that control the data flow between the PC and the interface module.

0	Chip Select C	0	Data Bus 4	0	X
1	Chip Select B	1	Data Bus 3	1	X
2	Chip Select A	2	Data Bus 2	2	X
3	Sample / Hold	3	Data Bus 1	3	X
4	Mux Channel D	4	Data Bus 0	4	Data Bus 8
5	Mux Channel C	5	X	5	Data Bus 7
6	Mux Channel B	6	X	6	Data Bus 6
7	Mux Channel A	7	X	7	Data Bus 5

Register Addr = 0x378 Register Addr = 0x379 Register Addr = 0x37A

The first data register controls the eight bit data bus, which can only be used for output on a standard Centronics parallel interface. The second register for the interface is known as the status register. This register has five read-only bits and three unused bits. The third register is known as the control register; it contains four bi-directional bits, an interrupt bit, and three unused bits. [Messmer] So the total number of signals that the standard Centronics parallel interface can control, are 8-12 outputs, and 5-9 inputs. For this implementation, the port registers are configured to utilize nine inputs and eight outputs.

Since there are only nine input connections available, not all of the 10-bits from the analog to digital converter can be read at once. So two successive reads would have to be performed in order to capture all 10 bits from the ADC chip. However, since 9 bits still provide very good accuracy for this application (approximately 10mV resolution), and since the accuracy of the converter is 10bits +/- 1/2 bit, the least significant bit can be discarded without significant loss of accuracy. The output signals consist of three chip select lines, four multiplexer data select lines, and a read/sample-hold line. The three chip select lines can access up to eight peripheral chips, and the four multiplexer lines are used to select one of 16 possible analog inputs from the MUX chips. The combination read, and sample-and-hold line is responsible for controlling the ADC. When this signal goes low, the ADC samples the signal on its input and begins to perform the conversion. After approximately 600ns, the chip initiates an interrupt, signaling that the output conversion is complete, and that the data presented at its output is valid.

Chapter 4 Software

All of the software for this project was written using Borland C/C++ v3.1 for MS-Windows. The main reason for using the Windows environment was because of its ability to produce sophisticated graphics fairly easily.

4.1 The Data Acquisition Module

After the interface hardware was completed, the second phase of the project was to program data acquisition control software. Since the interface circuit board is designed to be simple and inexpensive, the PC is responsible for issuing all of the control signals. The data acquisition module responsible for controlling both the low-level port communication, as well as the sampling rate and sampling duration.

4.1.1 Low-Level Parallel Port Software Issues

The communication between the PC and the interface module utilizes a fairly simple protocol. Communication is accomplished by first sending a specific byte to the output register that controls the parallel port. Then, as soon as the data on the interface module is valid, the remaining two port input registers can be read to transfer the data from the module to the PC.

In order to control the chip-selects and the multiplex lines, the proper bits need to be set in the output register. The parallel interface then makes the appropriate voltage transitions on each pin the corresponds to a specific bit of the port register. When reading data from the parallel interface, the two input data registers correspond to the digital voltages that are present on their associated input pins. The status of the registers can then be read by simply accessing their addresses. Since some of the signals are inverted by the parallel interface¹⁴, some of the register bits need to have their logic corrected by the software. Also, since the data

¹⁴ Certain bits of the port are inverted by default in order to interface properly with standard printers.

is being read from two registers, the final 9-bit result must be assembled by shifting the data from the third register up by five bits and then combining the two registers together.

EXAMPLE:

When the PC wants to read data from a specific sensor, it simply selects the channel by writing to the MUX bits of the output register. Then it initiates the sample-and-hold, read mode, and chip-select of the ADC by simply lowering the S/H bit. Then after a short delay of less than 1 μ s, the conversion is complete and the data is available to be read from the two input registers of the parallel interface. When the two registers are read, some of the bits have to be inverted and shifted to properly assemble the received data. Finally, the PC once again raises the S/H line to deselect the ADC and complete the read cycle.

4.1.2 High-Level Acquisition Control

The software allows the user to select the sampling rate, sampling duration, and which sensors to read. A typical and sufficient sampling rate for this implementation is 5-10 samples per second for each sensor. Since the sensors are monitoring relatively slowly varying mechanical systems, there is no need to sample faster than this unless a very subtle feature needs to be detected. The sampling duration depends mainly on the type of malfunction that the user is trying to detect. For most of the features that IADS can detect, a sampling duration of approximately 1-2 minutes produces good results. However, when attempting to detect a very slowly varying parameter, such as engine temperature, a sampling duration of over five minutes is often necessary. The failures that occur over such long time intervals are usually difficult for automobile technicians to detect, but the software is able to monitor these sensors very precisely over extended time intervals and can then make accurate decisions about the parameter fairly easily.

The general control portion of the software is also responsible for displaying real-time graphs of each sensor as the data is being collected.

This feature allow major errors or possibly bad connections to be easily noticed by the system user. This feature alone will enable amateur mechanics to recognize significant problems with their vehicle. Each data set that is captured is also stored in a data file that can be retrieved later for viewing, printing, or diagnostic processing by the fuzzy expert system.

4.2 Collect Sample Data Sets

After the low-level software tool was completed, the system was used to collect various data sets from various driving conditions. This process was necessary in order to observe the exact behavior of each of the sensors under different conditions. Although automotive technicians probably have a reasonably good idea about the general acceptable range of values for each sensor, they would still probably need to study some sample data sets before they were able to write accurate rules about the system. Studying the sample sets also enables the programmer or the 'expert' to more accurately define the five data ranges (very low, low, medium, high, very high) that are used in the fuzzy logic portion of the diagnostic software. Also, since automobile technical manuals don't publish extremely detailed information about the proper operating levels of the electronic control system under different driving conditions, the collection of actual data sets was necessary to determine valid ranges for each sensor.

Chapter 5 Fuzzy Logic

While “equations can model simple systems like a pendulum, and statistics can describe huge disorganized systems like gas molecules in a jar.” mathematics staggers with complex systems, biological systems, and humanistic systems. [McNeil, Freberger] Complex electro-mechanical systems often defy human comprehension; they are extremely difficult to completely understand or model. These issues formed the primary motivation for Dr. Lotfi Zadeh’s introduction of a different kind of mathematics, known as *fuzzy logic*.

5.1 Fuzzy Logic Background

Fuzzy logic is an extension of conventional Boolean logic that is capable of dealing with concepts of partial truths. Fuzzy logic was introduced by Dr. Lotfi Zadeh in the mid 1960’s as a tool to model the uncertainty of natural language. [McNeil, Thro]

For the IADS, fuzzy logic is utilized in a fuzzy expert system in order to both create a model for the system, and perform the final diagnosis of the symptoms based on the model. A fuzzy expert system is simply an expert system that utilizes fuzzy logic instead of traditional Boolean logic. Because of their non-discrete nature, fuzzy expert systems are better suited for numerical processing than conventional symbolic reasoning engines. [Cox]

It would be ideal to be able to write a set of rules in plain language, that would enable the IADS to perform the analysis of the vehicle’s signal data. The first step in this process is to translate the inputs into a form that an expert system can deal with. Fuzzy logic can be thought of as the translator for the reasoning process in this application. The second step in this process is to make a model of the predicted behavior for all of the system’s parameters. IADS can much more easily make decisions about the actual input data when it is compared to the modeled data. This type of reasoning is known as *model-based-reasoning*. Instead of using traditional modeling techniques for this implementation, the fuzzy

expert system is responsible for generating the model for each parameter. This is a much easier task to achieve by using fuzzy logic, than it would be if mathematical equations had to be derived to model each subsystem's behavior. Once the model is created, fuzzy logic is utilized a second time to make a final diagnosis about the state of the entire system based on that model.

There are four basic steps in a fuzzy expert system. These steps are known as:

1. Fuzzification
2. Inference
3. Composition
4. Defuzzification

The fuzzification step simply translates each input value into a form that can be dealt with by the rule base. In the inference phase, the validity of each rule is computed, so that, in the composition step, all of the valid rule premises that apply to the same output parameter are combined. Finally, the defuzzification process translates the fuzzy conclusion back into a real parameter value such as a temperature.

5.1.1 Fuzzification

The fuzzification process translates inputs into a format that an expert system utilize. In this application, the fuzzification process translates the input sensor values into a representation called *membership levels*.

Rules used in a fuzzy expert system usually have a context such as:

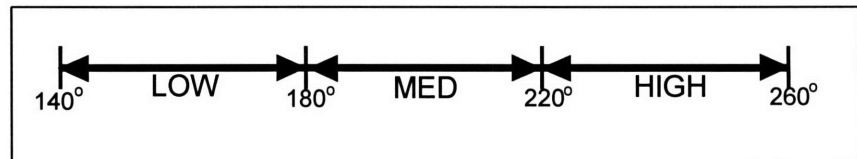
[if *RPM* is *HIGH*, and *SPEED* is *LOW*, then *TEMP* is *HIGH*]

In this case, *RPM* and *SPEED* are the *input variables*, while *TEMP* is the *output variable*. The values of the input variables are read from the vehicle's sensors. The premise of the rule describes to what degree the rule is applicable, while the conclusion of the rule assigns a *membership function* to the output variable.

Before any of the rules can be written, there needs to be some way to quantify the input signal data. In most expert systems, an expert uses heuristics¹⁵ to create rules and make judgments about a system. Suppose that an expert states that engine temperatures between 140°F and 180°F are *LOW*; temperatures between 180°F and 220°F are normal or *MEDIUM*; and temperatures between 220°F and 260°F are *HIGH*. **Figure(5.a)** shows a graphical representation of these three temperature ranges.

FIGURE 5.a

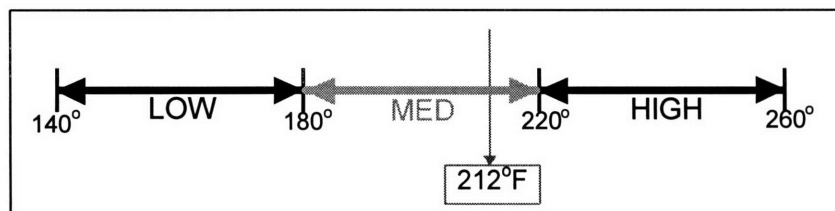
A linear representation of the engine temperature scale.



Now suppose, that the engine temperature sensor is currently reading a value of 212°F. In the linear temperature scale representation, this value would appear somewhere in the *MEDIUM* range as shown in **figure(5.b)**.

FIGURE 5.b

Linear temperature scale, with an input value of 212°F noted in the medium range.



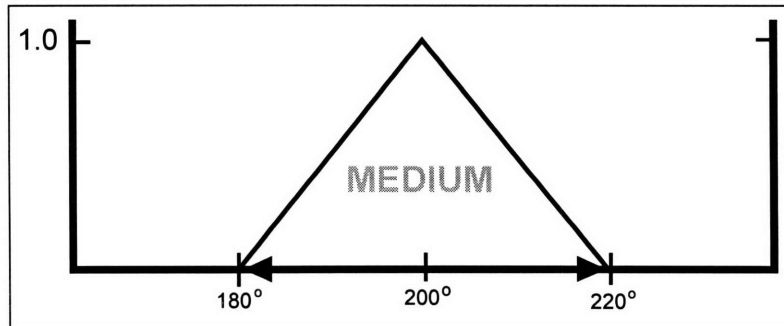
However, from this representation it is unclear how this temperature can be distinctly related to a rule. It is only obvious that it is a *MEDIUM* temperature, but this representation has no clear distinction between 212°F and, say, 195°F; they are both *MEDIUM* temperatures. So there is a need to improve this representation in order to be able to distinguish between two different temperature values within the same range.

¹⁵ A heuristic is best defined as a rule-of-thumb.

Figure(5.c) shows what is known as a *fuzzy subset* or *membership function*.

FIGURE 5.c

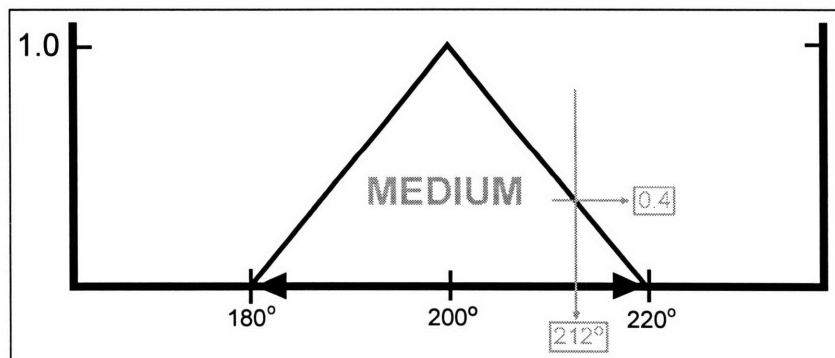
The *MEDIUM* fuzzy subset.



This graph illustrates a function that uniquely determines the level of membership for any temperature within the entire range of *MEDIUM* temperatures. The values between 0-1 can be thought of as certainties or *membership levels* in the *MEDIUM* range. Now, if the input value is 212°F, there is a clear distinction between other temperature readings that also fall into the *MEDIUM* range. **Figure(5.d)** shows that the temperature 212°F has a membership level (or certainty) of 0.4 in the *MEDIUM* set, which translates linguistically into a phrase such as “sort-of-medium.”

FIGURE 5.d

The *MEDIUM* fuzzy subset with an input value of 212°F has a membership level of 0.4.

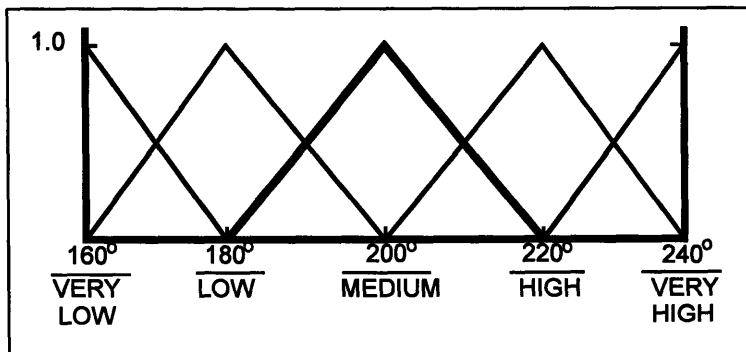


This process of translating an input value into a membership level is known as *fuzzification*. It essentially translates the data that is read from the sensors into a representation that the knowledge base is capable of recognizing.

To complete the representation of the fuzzy sets over the entire possible temperature range, the remaining *fuzzy subsets* have to be defined. There are many different ways of defining each range. In some applications, trapezoidal *membership functions* are used, and in very sophisticated applications, non-linear membership functions may also be used. For this implementation, it is sufficient to use the triangular membership functions in order to define each fuzzy subset. This system utilizes five ranges for each sensor. In very simple applications, it may be desirable to only define three ranges, while in applications where accuracy is essential¹⁶, more than five ranges may be necessary. But the number of ranges defined directly affects the number of rules that have to be written to define an output for all possible combinations of input ranges. **Figure(5.e)** shows a typical complete set of ranges defined for engine temperature.

FIGURE 5.e

A typical complete membership set defined for the entire engine temperature range.



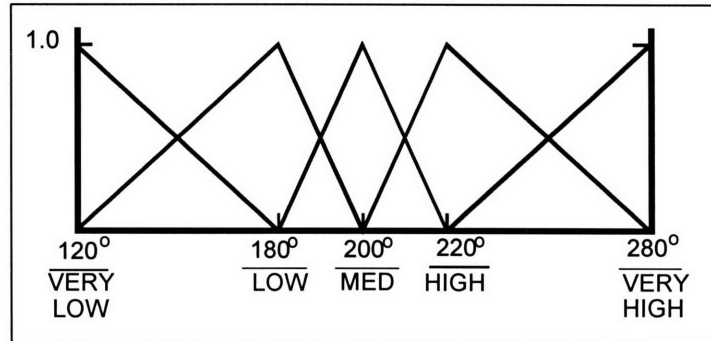
Just as there are many different ways to represent each individual subset, there are many different methods for defining how the subsets span the entire range of values. **Figure(5.e)** shows completely symmetric subsets defined over the entire temperature range; each increment between ranges is a uniform 20°F. However, it is often desirable to define these subsets differently. In some cases where there is a large valid temperature range, but the temperatures near the middle of the span are

¹⁶ The number of ranges can be thought of the resolution of the set. If many small ranges are defined in each set, then the expert system programmer has more control when writing rules that attempt to recognize very subtle variations in system parameters (but the number of rules that must be defined increases significantly with each additional range defined in the fuzzy set).

the most critical to distinguish, it is more useful to define the subsets as shown in **figure(5.f)**.

FIGURE 5.f

Fuzzy set defined for non-uniform temperature ranges.

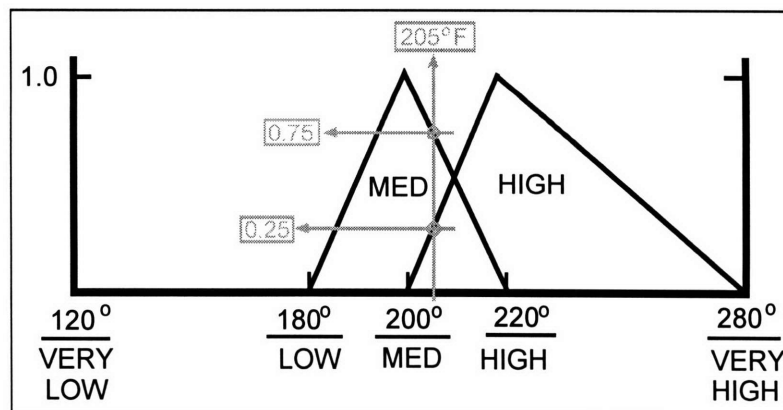


Although the slopes and widths of some of the sets changed, the membership levels for any given input over the entire range always add to 1.0 certainty¹⁷. This is not a requirement for these types of sets, but it is a fairly common practice. This overlapping technique also helps to uniformly smooth transitions between neighboring subsets.

In this particular implementation, each time that a sensor value is read and recorded, the associated membership levels for that sensor are calculated using the fuzzification process. For instance, if a temperature of say, 205°F was read from the engine temperature sensor, it would yield a membership level of (0.75) for the *MED* range, and a level of (0.25) for the *HIGH* range as shown in **figure(5.g)**.

FIGURE 5.g

An input temperature of 205°F yields a membership level of 0.75 in the *MEDIUM* range, and a membership level of 0.25 in the *HIGH* range.



¹⁷ This means that if there is an input that lies in the overlap between two subsets, the sum of the membership levels associated with each of the subsets will add to 1.0.

The membership values for the other three ranges would all be zero for this particular input.

Temperature Membership Levels
(for a 205°F reading):

<i>VLOW:</i>	0.0
<i>LOW:</i>	0.0
<i>MED:</i>	0.75
<i>HIGH:</i>	0.25
<i>VHIGH:</i>	0.0

Now, further suppose that the following five rules were defined in the knowledge base.

1. [if *TEMP* is *VERYLOW* then ..]
2. [if *TEMP* is *LOW* then ..]
3. [if *TEMP* is *MED* then ..]
4. [if *TEMP* is *HIGH* then ..]
5. [if *TEMP* is *VERHIGH* then ..]

Since an input of 205°F translated into a 0.75 *MEDIUM* membership level, and a 0.25 *HIGH* membership level, rules 3 and 4 will fire¹⁸ in the rule base. Rule 3 will fire with 0.75 certainty, and rule 4 will fire with 0.25 certainty. The next step in the process deals with all of the resulting membership levels.

5.1.2 Inference

In the fuzzification process, every rule's premise is evaluated to determine if it has a non-zero certainty level. If there are two input variables in a given rule, then both of them have to evaluate to non-zero values since they are combined with an *And* operation. The inference process is responsible for applying the validity of the input variables to

¹⁸ When the premise for a rule has a nonzero membership value, it is said to 'fire,' or apply.

the output variables for each rule that fires. For example, once the fuzzification process is finished computing the membership levels of all of the input variables, then the inference process takes over in applying these certainty levels to the rule base. Suppose the system computed the following membership levels for the temperature and pressure input variables corresponding to sensor values of say 245°F and 30KPa:

	Temperature	Pressure
<i>VLOW</i>	0.0	0.0
<i>LOW</i>	0.0	0.8
<i>MED</i>	0.0	0.2
<i>HIGH</i>	0.4	0.0
<i>VHIGH</i>	0.6	0.0

Now, further suppose that temperature and pressure are the two parameters responsible for modeling engine RPM. The rule base would then consist of a list of 25 rules correlating every possible combination of temperature and pressure with a resulting estimated engine RPM. The rules would be stated as follows:

1. [if *TEMP VLOW* and *PRESS VLOW* then *RPM LOW*]
2. [if *TEMP VLOW* and *PRESS LOW* then *RPM LOW*]
3. [if *TEMP VLOW* and *PRESS MED* then *RPM MED*]
4. [if *TEMP VLOW* and *PRESS HIGH* then *RPM HIGH*]
5. [if *TEMP VLOW* and *PRESS VHIGH* then *RPM HIGH*]

6. [if *TEMP LOW* and *PRESS VLOW* then *RPM LOW*]
7. [if *TEMP LOW* and *PRESS LOW* then *RPM MED*]
- ...

17. [if *TEMP HIGH* and *PRESS LOW* then *RPM MED*]
18. [if *TEMP HIGH* and *PRESS MED* then *RPM HIGH*]
- ...

22. [if *TEMP VHIGH* and *PRESS LOW* then *RPM HIGH*]
23. [if *TEMP VHIGH* and *PRESS MED* then *RPM VHIGH*]
- ...

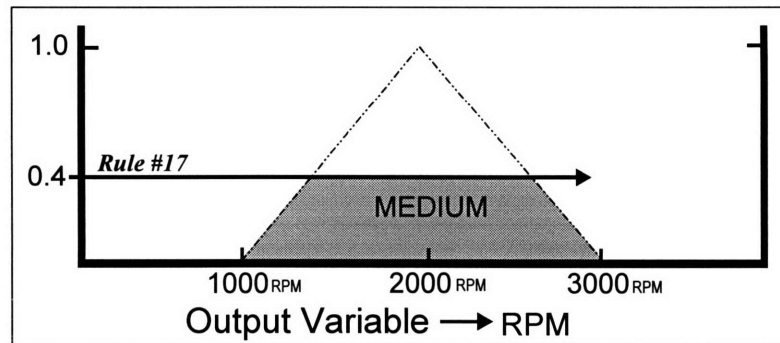
For this particular example, rules 17, 18, 22, 23 are the only rules to fire, since the entire premise of each of these rules (both input variables) evaluate to a non-zero certainty level. The fuzzy logic operators that correspond to the conventional logic operators AND, OR, and NOT, are MIN, MAX, and NOT. [Yester] So for each of the rule premises that have two input variables combined with an AND operator, the MIN of the two membership levels is taken to evaluate the degree of truth (certainty level) for the entire rule. The resulting certainties for the rules that fire are computed below:

- 17. [if *TEMP HIGH* and *PRESS LOW* then *RPM MED*]
 $\text{MIN}(0.4 \text{ and } 0.8) \implies 0.4$
- 18. [if *TEMP HIGH* and *PRESS MED* then *RPM HIGH*]
 $\text{MIN}(0.4 \text{ and } 0.2) \implies 0.2$
- 22. [if *TEMP VHIGH* and *PRESS LOW* then *RPM HIGH*]
 $\text{MIN}(0.6 \text{ and } 0.8) \implies 0.6$
- 23. [if *TEMP VHIGH* and *PRESS MED* then *RPM VHIGH*]
 $\text{MIN}(0.6 \text{ and } 0.2) \implies 0.2$

These resulting membership values are then applied to each output variable. For example, rule #17 fires with a certainty of 0.4, which is applied to the *RPM MED* fuzzy subset. **Figure(5.h)** shows the graphical representation of the application of this rule to the output set.

FIGURE 5.h

Rule #17 fires with a certainty of 0.4. This certainty is applied to the rule's output variable: *MEDIUM* RPM.



Since the rule only fired with a certainty of 0.4, the output variable's fuzzy subset is said to be *clipped*. If the rule fired with a certainty of 1.0, then the rule's result would include the entire *MEDIUM* subset, but

since it only fired with partial certainty, only a fraction of the *MEDIUM* subset is applied to the output.

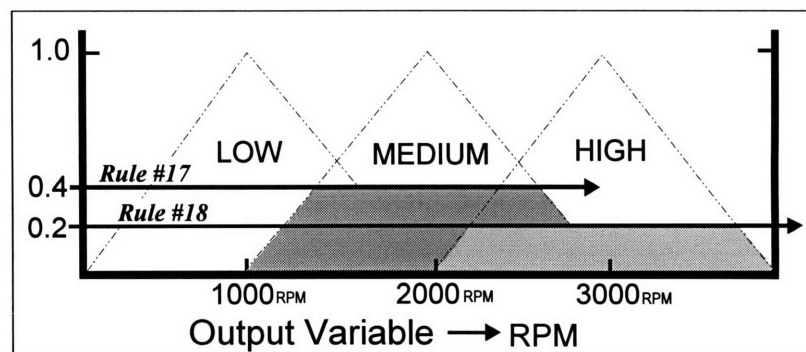
5.1.3 Composition

During the composition phase, all of the fuzzy subsets that are assigned to each output variable are combined to form a single fuzzy subset for each output. There are two primary methods of combining the subsets. One method is known as *max* composition, where the pointwise maximum of each of the subsets applied to the variable is taken to determine the final output fuzzy subset. The other well-known method is called *sum* composition. In sum composition, the pointwise sum of all of the fuzzy subsets that contribute to the output variable is computed to determine the final output subset.

For example, assume that *only* rule #17 and rule #18 fired in the previous example. Rule #17 fired with a certainty of 0.4 applied to the RPM *MED* fuzzy subset, and rule #18 fired with a certainty of 0.2 applied to the RPM *HIGH* subset. An illustration of the sum composition process for these two rules is shown in **figure(5.i)**.

FIGURE 5.i

Rule #17 fires with a certainty of 0.4 applied to RPM *MED*, and rule #18 fires with certainty 0.2 applied to RPM *HIGH*. The resulting fuzzy subsets for each rule are shown.



Since sum composition can yield fuzzy subsets with values greater than 1.0, it is usually followed by, or combined with, a defuzzification method that resolves these cases. In this fuzzy logic implementation, the composition and defuzzification processes are combined into a single step using a technique known as the *centroid* method.

5.1.4 Defuzzification

Although in some applications it is useful to keep the final form of the fuzzy logic process as a fuzzy subset, in most cases, a final ‘crisp’ number is the preferred output. Just as the fuzzification process can be thought of as translating sensor output values into fuzzy logic representations, the defuzzification process can be thought of as the procedure that translates the fuzzy representations back into real parameter values.

In this fuzzy logic implementation, the centroid defuzzification method was utilized. **Figure(5.j)** shows the centroid equation: μ represents the certainty with which each rule fires, **Area** is the area of the remaining fuzzy subset after it has been ‘clipped’ by the certainty level, and **Center** is the value of the center of the corresponding fuzzy subset (the x-coordinate value where the certainty level is 1.0).

FIGURE 5.j

The defuzzification centroid calculation.

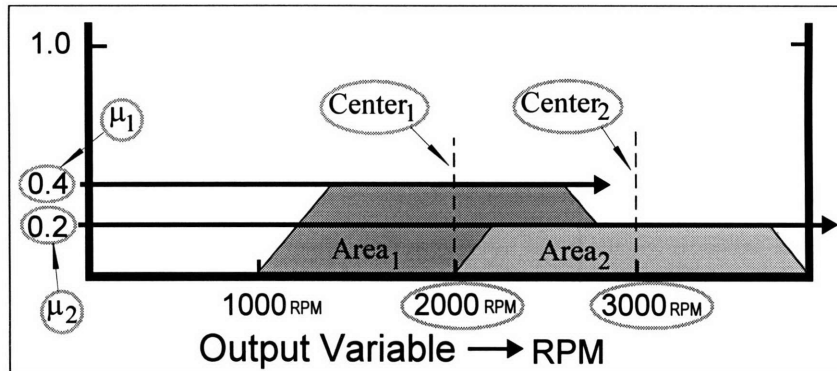
$$\text{Output} = \frac{\sum_{j=1} (\mu_j \text{Area}_j \text{Center}_j)}{\sum_{j=1} (\mu_j \text{Area}_j)}$$

- μ_j is the certainty level of the fuzzy subset j
- Area_j** is the area of fuzzy subset j
- Center_j** is the center value (peak value) of fuzzy subset j

The number of elements in the summation is determined by the number of rules that apply to the same output variable. For the current example, all of the rules that fire and have RPM as their output variable would each contribute one element to the summation. **Figure(5.k)** highlights the variables used in the centroid equation assuming that only rules 17 and 18 fire, as in the previous example.

FIGURE 5.k

The illustration shows the six parameters associated with rule #17 and rule #18.



Simply by observing the relative sizes of the two fuzzy subsets in **figure(5.k)**, it is easy to see that the resulting defuzzified output will be weighted more closely to the 2000RPM (*MED*) value than the 3000RPM (*HIGH*) value, since the *MED* fuzzy subset has a higher certainty (which corresponds to a greater weight in the defuzzification process). When the specific values for each element are substituted into the centroid equation, the resulting defuzzified output is 2220RPM. **Figure(5.l)** shows the details of the calculation.

FIGURE 5.l

Defuzzification of rules 17 and 18 using the **centroid** method.

$$\begin{aligned} \text{Area}_1 &= 1/2(2000)(1.0) - 1/2(1200)(1-0.4) = 640 \\ \text{Area}_2 &= 1/2(2000)(1.0) - 1/2(1600)(1-0.2) = 360 \end{aligned}$$

$$\frac{(0.4)(640)(2000) + (0.2)(360)(3000)}{(0.4)(640) + (0.2)(360)} = 2220$$

There can be any number of rules that apply to one particular output variable. Each rule that fires and applies to a particular output, can be thought of as adding its own certainty, or weight, to the output of the variable. This is how smooth transitions between the various fuzzy sets are accomplished. Several rules applying to a specific output fire, each with different certainty levels, and some with different output subsets (ranges). When all of their resulting weights are combined using the

centroid calculation, the output is a distinct number that lies somewhere between all of the output fuzzy subsets that applied. This defuzzification process completes the fuzzy logic operation.

5.2 Writing Fuzzy Logic Rules

Rules with single input variables are fairly easy to write. There must be five rules defined in a rule set in order to completely cover all possible input ranges of the input variable. For example, if only *one* rule was written that stated:

[if *RPM HIGH*, then *VSS MED*]

then, if the *RPM* input falls out of the *RPM HIGH* range, the *VSS* model will be undefined. So, five rules that cover all of the possible input ranges have to be written in order to ensure that the model for *VSS* is always defined.

When there are two input variables, 25 rules have to be written to complete the rule set for a particular output variable. Two input variables yields 25 different input combinations, since there are 5 subsets for each input variable. In this situation, it is helpful to construct a chart when writing the rules in order to help visualize the situation. A sample chart for a set of two-input rules is shown in **figure(5.m)**.

FIGURE 5.m

Two dimensional sample rule chart used for writing two-input fuzzy logic rules.

		<i>Integral of TPS</i>				
		VLOW	LOW	MED	HIGH	VHIGH
<i>Integral of RPM</i>	VLOW	VSS= VLOW	VSS= VLOW	VSS= VLOW	VSS= VLOW	VSS= LOW
	LOW	VSS= VLOW	VSS= LOW	VSS= LOW	VSS= LOW	VSS= MED
	MED	VSS= MED	VSS= MED	VSS= MED	VSS= MED	VSS= HIGH
	HIGH	VSS= HIGH	VSS= HIGH	VSS= HIGH	VSS= VHIGH	VSS= VHIGH
	VHIGH	VSS= HIGH	VSS= VHIGH	VSS= VHIGH	VSS= VHIGH	VSS= VHIGH

The two input variables are *averageRPM* and *averageTPS*, while the output variable is *VSS*. Once the chart is created, it is easy to write out the 25 rules.

In situations where three input variables are required to model a parameter, there have to be 125 rules written to complete the rule set for that output variable. However, in these cases it is common for one of the input parameters to have little influence throughout most of its range, which simplifies the creation of so many rules. For example, if the third input variable is engine temperature, and it only influences the output of the rule if the temperature is *VERYHIGH*, then only 25 rules are affected by this special situation. So the same two-dimensional rule grid can be used for engine temperature ranges of *VLOW*, *LOW*, *MED*, and *HIGH*. Only the rule grid corresponding to *VHIGH* temperature has to be modified.

5.3 The Fuzzy Logic Implementation

In this project, the fuzzy logic process is applied in two distinct modules of the system. In the first stage, fuzzy logic is called upon to create a model of the system's parameters. The second stage of the process utilizes this model along with all of the other input parameters available to the system in order to perform the final diagnosis of the vehicle. **Figure(5.n)** shows a block diagram representing the system divided up into four distinct phases.

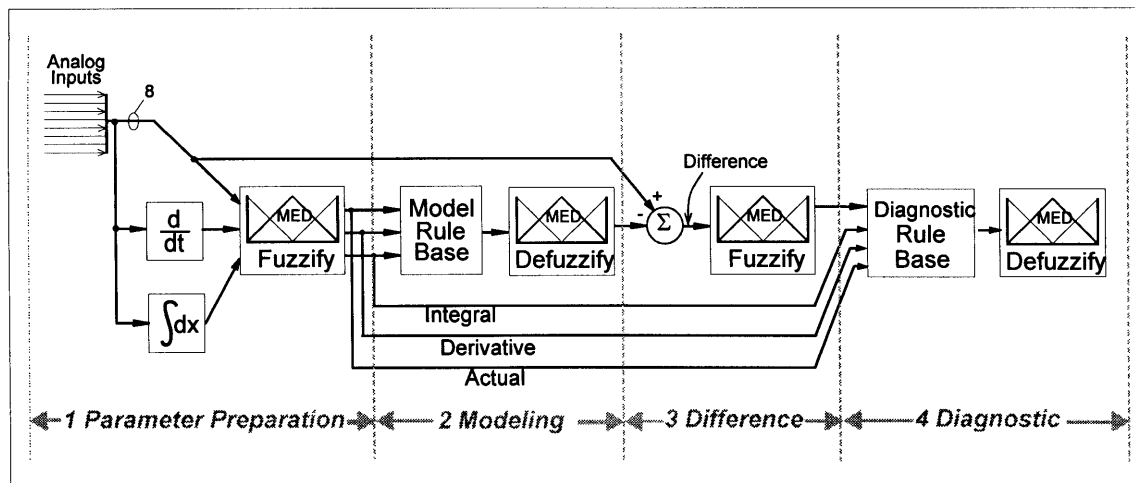


FIGURE 5.n Fuzzy logic implementation block diagram, illustrating the four phases of the diagnosis.

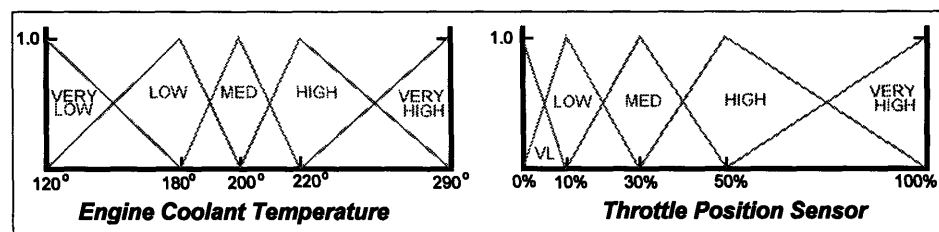
5.3.1 Phase 1-2: Parameter Preparation and Modeling

The first three phases of the entire process are responsible for preparing all of the parameters that the *Diagnostic Phase* will utilize for the final output. In general, the modeling process is accomplished by writing rules that can predict the value of each of the vehicle's sensors, based on the current and past state of all of the rest of the vehicle's sensors. For example, inputs **A** and **B** can be used to predict the value of input **C**, and perhaps inputs **C** and **D** can in-turn predict a value for parameter **A**. As a more specific example: the Throttle Position Sensor (*TPS*) and the *RPM* sensor can together be used to model the value that the Vehicle Speed Sensor (*VSS*) should be reading. So there is a distinct set of rules contained in the *Modeling Rule Base* that are responsible for modeling values for each of the 8 sensors. Phase 1--the *Parameter Preparation Phase*-- illustrated in figure(5.n), performs the fuzzification of all of the eight input parameters that are referenced in the Model Rule Base by the expert system. Phase 1 also computes the derivative and integral values of each of the sensor inputs, so that the expert system can also utilize these values in creating the model.

During the modeling process, each parameter has a fuzzy set associated with it. Each fuzzy set contains five different ranges (*fuzzy subsets*) for the parameter. Two actual examples of fuzzy sets taken from the system's program are shown in figure(5.o).

FIGURE 5.o

Two sample fuzzy sets from the system rule base.



Simplified Example:

Input:

If the engine temperature is 220°F, then the fuzzification process computes a membership of 1.0 in the **HIGH** subset as shown in figure(5.o).

Process:

Now, assume that there is only one simple model rule that fires in this case; the rule states [if *TEMP HIGH* then *THROTTLE HIGH*]. To compute the output, the application of this single rule needs to be defuzzified.

Output:

The fuzzy set in **figure(5.0)** for the throttle position sensor would once again be used to compute the model TPS output value, which in this case (*THROTTLE* is *HIGH* with certainty 1.0) would yield a 50% open throttle.

The same fuzzy sets are used for each parameter in both the fuzzify stage and the defuzzify stage of the logic. Specifically, when the analog input value for, say, the *TPS* sensor is fuzzified, the *TPS* fuzzy set is utilized. Also, after all of the rules that apply to modeling the *TPS* parameter, the same *TPS* fuzzy set is utilized in the defuzzification process¹⁹.

After the *Parameter Preparation Phase* is complete, then the rule base in Phase 2--the *Modeling Phase*--is called upon to make judgments about the current and past states of the inputs in order to model each parameter. After the entire Modeling Rule Base has been searched for all applicable rules, the defuzzification process completes the *modeling phase* by translating all of the resulting fuzzy subsets back into distinct parameter values.

This modeling process can not only be used to generate modeled parameters for comparison purposes, but it can also be used to model parameters that are not available as inputs to the system. For instance, engine torque would be a useful parameter for the system to use in the rule base, but there is no torque sensor in a car. However, by monitoring the Throttle Position Sensor, the RPM sensor, and the Vehicle Speed Sensor, the expert system can create a model of the torque output from the engine.

¹⁹ In other words, each fuzzy set associated with a particular parameter is utilized as both the input variable set and the output variable set.

This modeling technique is a useful strategy in simplifying and clarifying rules that will be used in the next stage of the diagnosis. For example, if the system needed a rule that predicted an output based primarily on the engine torque, then the rule could simply use the calculated torque variable instead of having to reference all three of the input variables that in-turn predict the torque. This technique makes it much easier to read and understand the rules in the rule base, because there are fewer variables referenced in each rule, and there are fewer redundant rules. Since each input and output variable has five possible ranges (*VLOW*, *LOW*, *MED*, *HIGH*, *VHIGH*), each additional parameter referenced in a rule multiplies the number of rules for that particular output variable by five. An output variable with a single input variable needs only five rules; an output variable with two input variables needs 25 rules; an output variable with three input variables needs 125 rules, etc.

5.3.2 Phase 3: Difference

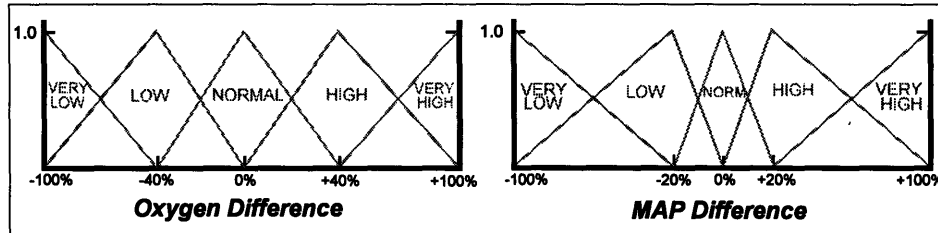
The next step in the diagnostic process, after all of the desired parameters for the system have been modeled, is to calculate the difference between each actual parameter reading and its modeled value. In order to calculate the difference values, the modeled input parameters have to be defuzzified first so that their numeric values can be subtracted from the actual sensor values. After the difference value is calculated for each parameter, it needs to be translated back into a fuzzy logic representation so that the final *Diagnostic Phase* can utilize a rule base in order to make decisions based on these differences²⁰. So, once again, the *fuzzify* stage of the fuzzy logic tool is called upon to categorize each difference value. A new group of fuzzy sets must be defined in order to categorize the difference values for each parameter. The differences are defined as being *VERYLOW*, *LOW*, *NORMAL*, *HIGH*, or *VERYHIGH*. Because the behavior of each of the sensors are very different from each other, there needs to be a unique *difference fuzzy subset* assigned to each parameter. Simply assigning the same percentage of tolerance to all of the sensors would not yield accurate results. For example, since the

²⁰ This type of reasoning is commonly known as Model Based Reasoning.

oxygen sensor fluctuates greatly, it is assigned a fairly large tolerance of about +/- 40% to be considered *HIGH* or *LOW*, but the MAP sensor is much more stable and can be predicted more accurately, so its tolerance is set at only +/- 20% for the *HIGH* or *LOW* range. The difference fuzzy sets for these two parameters are shown in figure(5.p).

FIGURE 5.p

Difference fuzzy sets for the oxygen and MAP sensors.



The fuzzified difference values for each of the eight input parameters are stored in the variables: *DIFFOXYGEN*, *DIFFTPS*, *DIFFMAP*, *DIFFTEMP*, *DIFFMAT*, *DIFFBATTERY*, *DIFFVSS*, and *DIFFRPM*. So, if the output variable *DIFFTEMP* was considered *HIGH*, it would mean that the actual input for engine temperature was *hotter* than the model predicted. And if *DIFFTEMP* was considered *VERYLOW*, it means that the actual engine temperature was *much colder* than the model predicted.

This difference value makes the final diagnostic process much easier to create, and is another example of the simplifying technique explained in section 5.3.1. Rules can now be written that simply look at one input (the difference between the model and the actual value) that represents influences from a variety of the system's input variables. The primary function of the *Parameter Preparation Phase* is to provide the *Diagnostic Phase* with all of the tools that it will need to perform its function. Providing these prepared parameters makes the diagnostic rule base much easier to create, edit, and understand.

For example, a rule that utilizes a difference value as an input can be stated as follows:

[if *DIFFVSS HIGH* and *TEMP LOW* then . .]

This rule translates into: if the actual *VSS* (vehicle speed) is high compared to the modeled *VSS* value, and the *TEMP* (engine temperature) is currently *LOW*, then . .

The equivalent rule that does not utilize this difference value (*DIFFVSS*), would have to be stated in the following form:

[if *RPM HIGH* and *TPS LOW* and *VSS MED* and *TEMP LOW* then . .]

It is very difficult to read and understand a rule in this format. The first three input variables are required to determine that the *VSS* is higher than it should be. This same function was accomplished in the previous rule by simply referencing only one input variable's state--*DIFFVSS HIGH*. Any other rule in the diagnostic rule base that needs information about the validity of the vehicle's speed only needs to reference a single variable, instead of repeatedly computing it using on the same three input variables every time. This technique also helps to reduce the total size of the rule base, since redundant rules premises and computations are avoided. Each time the *DIFFVSS* parameter is referenced, it only adds five rules for itself; but if the three original parameters for *DIFFVSS* are referenced each time, they would add 125 rules for every instance when they are reference.

5.3.3 Phase 4: The Diagnostic Phase

The entire set of parameters that the first three phases of the process provide to the *Diagnostic Phase* consists of:

- Fuzzy values of the actual sensor inputs
- Fuzzy values of the modeled input variables
- Fuzzy difference values between each modeled and actual input
- Fuzzy integral of each actual sensor input
- Fuzzy derivative of each actual sensor input

The integral and derivative values are particularly useful in the diagnostic process, since the majority of the symptoms that the system can accurately detect are based on the time-history of each sensor.

Imagine trying to figure out what the vehicle is doing by simply looking at the instantaneous value of each sensor without remembering the previous values of the sensors. Even a human expert could not perform any sort of diagnosis with only that type of instantaneous information. However, when viewing a graph of the time-history for each sensor, it is much easier to understand the operation of all of the subsystems. This is why the integral and derivative inputs to the system are so crucial.

After all of the data is prepared in the first three phases, phase 4 is ready to perform the final diagnostic reasoning. The Diagnostic Rule Base contains about 600 rules that make diagnostic decisions based on all of the parameters that the preparation phases provide. IADS is capable of diagnosing over 60 possible problems, but some of the problems have the same symptoms, so wherever it was reasonable, the problems were combined in order to decrease the number of redundant rules and the overall size of the rule base. For example, the system can detect several different problems with the fuel system, like a clogged fuel filter, faulty fuel pump, clogged fuel injectors, etc.; so these problems were all lumped together into the category of a *clogged fuel system*. So the number of possible problems that the system can suggest was reduced to about 40. Some of the 40 problems that IADS is capable of identifying are:

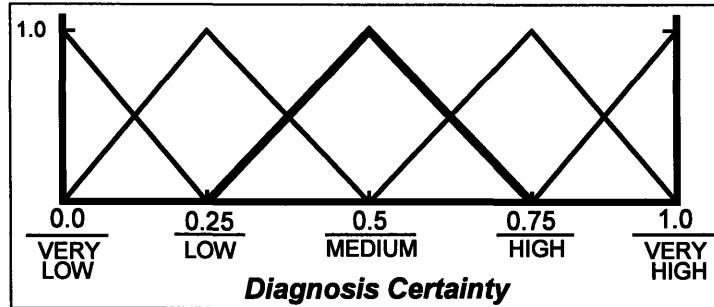
- Low Oil Level
- Low Coolant Level
- Radiator Fan Broken
- Air Filter Clogged
- Transmission Fluid Low
- Temperature Sensor Faulty

The final output of the system is a list of the top several possible problems along with their associated certainty. The certainty level is a number between 0-1.0, where 1.0 is defined as complete certainty about a particular problem. In order for the rule base to assign certainties to each possible problem, one more fuzzy set is necessary to translate the output variables of the diagnostic rules into a value between 0-1.0. This

fuzzy set is called the *Diagnosis Certainty* set, and it is defined over the certainty range [0,1.0] as shown in **figure(5.q)**.

FIGURE 5.q

Fuzzy Set for the output diagnosis certainty.



In this case, the same fuzzy set is used for each of the 40 possible output variables, because the set represents the same parameter in this case: degree of certainty.

The diagnostic rules are written in the same form as the modeling rules. For example, if a rule stated: “if the *RPM* sensor reading is *HIGH* compared to the model, then there is *HIGH* certainty that the problem is *TRANSMISSION_FLUID_LOW*,” its syntax would be transcribed as follows:

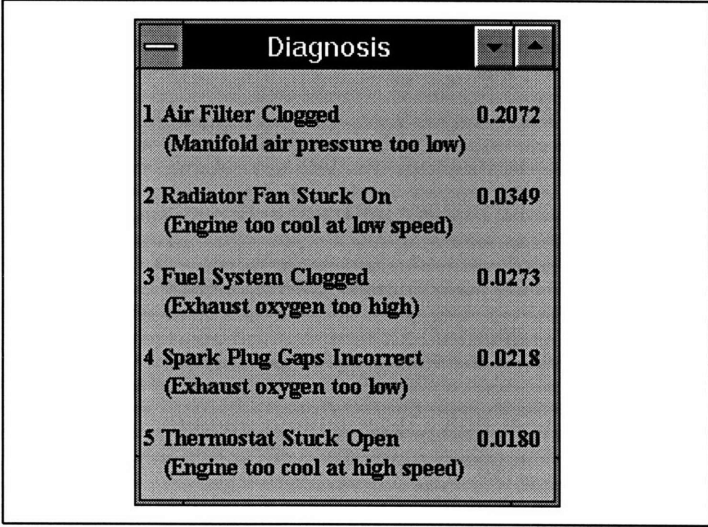
[if *DIFFRPM HIGH* then *TRANSMISSION_FLUID_LOW HIGH*]

The output variable (in this case: *transmission_fluid_low*) for each of the diagnostic rules is one of the possible 40 problems, while the output value is always calculated using the same *Diagnosis Certainty* fuzzy set that is shown in **figure(5.q)**. Just as in the *Modeling Phase* of the system, the *Diagnostic Phase* has to call upon the inference engine to search through all of the rules that apply for the each of the possible problems, and to evaluate the validity all of the associated input variables for each case. After all of the certainties have been applied to each output variable, the resulting fuzzy subsets are defuzzified to obtain the final numerical degree of certainty.

So at each data sample point, the certainty of all of the possible problems is computed using the data set for that particular sample point. In order to perform an accurate diagnosis, the decision can't be based on a single data point in the set, so the certainty levels that are computed at each point are summed and normalized over the entire sampling duration. The final result is a normalized certainty value between 0-1.0 throughout the entire data set. This is done to filter out minor errors in the signals that only occur for a short period of time. After the normalized certainties are calculated for all of the possible problems, IADS ranks and displays the problems that resulted in the highest certainties as shown in **figure(5.r)**.

FIGURE 5.r

Sample system
diagnostic output
summary.



Diagnosis	
1 Air Filter Clogged (Manifold air pressure too low)	0.2072
2 Radiator Fan Stuck On (Engine too cool at low speed)	0.0349
3 Fuel System Clogged (Exhaust oxygen too high)	0.0273
4 Spark Plug Gaps Incorrect (Exhaust oxygen too low)	0.0218
5 Thermostat Stuck Open (Engine too cool at high speed)	0.0180

This particular sample output indicates that the most certain problem is a clogged air filter. Each suggested problem also shows a short explanation of what symptoms influence the diagnosis of that particular problem. The highest certainty level for this sample output is 0.2, while the remaining certainties are all below 0.035. Because it would be very difficult to make a model of the system that predicted each input variable with extreme accuracy, there will always be some small differences between the modeled parameters and the actual input values. These small differences will trigger some of the diagnostic rules to assign small certainties to problems that probably don't really exist, or at least are not very serious. Any output certainty below 0.1 should be considered very low certainty and can usually be ignored. It is, however, useful to see a ranking of the degrees of certainty, so that the mechanic will know which problem to investigate first.

Chapter 6 Results and Observations

The final phase of the project involved testing the system's diagnostic capabilities using various data sets that were collected from the vehicle. The time intervals in which the data sets are recorded should be tailored to the type of analysis that is desired.

6.1 Data Recording

The majority of the data sets were recorded over one-minute intervals, which is the most ideal time scale for monitoring almost all of the sensors--other than the two temperature sensors. Most of the significant changes in a vehicle's subsystems occur over intervals of several seconds. So recording a 60second interval shows a few of these features throughout the data set. A one minute time interval is still short enough to produce fairly good resolution of the subtle features in sensor outputs. A few data sets were also collected in the 5-10 minute range in order to monitor the slow temperature fluctuations of the engine, and the manifold air temperature changes.

In order to accurately monitor the fuel level sensor, time intervals of well over 20 minutes must be recorded. However, because of the poor quality of the output from the fuel level sensor, the system was not able to perform accurate diagnostics based on fuel consumption. As the car moved and drove over bumps, the gas in the tank would splash around causing the sensor to produce an unstable signal. Even filtering the data would not create accurate results, since many other factors, such as road incline, also significantly affect the sensor output. Also, because of the physical design of the fuel-level sensor, throughout most of the fuel range, there was only a variation of a few millivolts between each gallon increment, so, accurate readings of fuel consumption were not possible. Monitoring fuel consumption was going to be a key feature in the system, because fuel consumption associated with various driving conditions is a fairly difficult parameter for technicians to monitor or estimate. If a different type of sensor (that is immune to fluctuations resulting from the vehicle's movement) was installed to monitor the fuel level, then the system could have detected several more problems with

the car, such as low tire pressure, poor oil quality, etc. But since the sensor is only used to display the approximate fuel level to the driver, and is not used in any of the ECM's functions, it is not designed to be an extremely accurate or stable sensor.

Another alternative method of monitoring the fuel consumption would be to monitor the duration of the pulse commands to each fuel injector. Assuming relatively constant fuel pressure, summing up the time that the fuel injectors were activated could yield a fairly accurate estimate for fuel consumption. However, the assumption of constant fuel pressure is probably not valid, and more sophisticated circuitry would have been necessary in order to accurately measure the pulse duration. The injector pulses range between approximately 0.5-3.0 milliseconds. [Ingersol]

6.2 Three Diagnostic Examples

Because it would be difficult and perhaps dangerous to physically alter systems in an automobile in order to test the performance of IADS, some of the normal sample data sets that were collected from the vehicle were slightly modified through software in order to simulate a variety of mechanical and electrical failures. Three of the scenarios that were tested are described below. In each of these three scenarios only one parameter's input was modified by a constant scale factor for a limited duration of the entire sampling period. Individual failure is probably the most common failure in a vehicle, however, some possible faults can significantly affect the values of several of the vehicle's remaining sensors. Also, if an important subsystem in the vehicle fails, it will most likely affect the output of at least one sensor significantly, and will probably moderately alter several other sensor outputs. Perhaps the only time that several of the vehicle's outputs would be significantly affected would be in extreme failure conditions, where the problem is so obvious that a diagnostic system is not even needed to tell the technician what has failed.

Since the Modeling Phase of the diagnostic process utilizes some of the actual input values in order to model other input values, the model may

be inaccurate when it utilizes one of the *modified* input variables in its reasoning process. For example, assume that sensor **A** is faulty. Then the model created for parameter **A** by inputs **B** and **C** will indicate that the value for **A** is incorrect. This is the desirable condition for the system. However, when the faulty **A** input is used in rules to model another parameter, **D**, then that parameter model may not be correct (depending on all of the actual sensors involved in creating the model).

A real example illustrates this situation even more clearly. Assume the **TPS** (Throttle Position Sensor) is broken, so it outputs a constant value of 0.5volts, which translates into 0% throttle. So for the first situation, the model will predict non-zero values for the **TPS** based on the fact that the vehicle is moving and accelerating (**VSS** and **RPM** are the inputs that model the **TPS**). This is fine, because the system will see that the actual **TPS** is not following the model, so it will suggest problems with the **TPS** sensor. However, when the actual (broken) **TPS** value is used in rules that model other parameters, such as **RPM**, the model will once again not coincide with the actual (appropriate) **RPM** value. So in this case, the system may diagnose a faulty **RPM** sensor, or low transmission fluid, or a high idle condition (none of which are true). But since the system doesn't initially know whether it is the **TPS** sensor that is broken or the **RPM** sensor that is faulty, it proceeds with its diagnosis and suggests all of the possible problems that the entire set of inputs indicate.

This situation isn't as bad as it seems, because there are almost always at least two inputs used to model each parameter; in the rules that model the **RPM** value, both the (broken) **TPS** value and the good **VSS** value are used. Since the **VSS** value is still a correct input, the resulting model for **RPM** will not be that far off. The modeled value for the **TPS**, however, will be significantly different from the constant 0% output, so many of the diagnostic rules will trigger with high certainty that the **TPS** is faulty. Since the system doesn't know for certain that the **TPS** is broken, it has to suggest all of the possible problems that the input values indicate. For instance, perhaps the **TPS** is actually 0% for a long period of time, but the car's idle condition is abnormally high. In this situation, the vehicle's acceleration and speed would indicate that the throttle is being depressed when it actually is not. So in general, the system shouldn't make decisions about which inputs are not valid, and only use the valid

inputs for the modeling process. This would not allow the system to suggest all of the possible problems that may be affecting the sensors.

6.2.1 Modified MAP Test

In the first diagnostic example, the MAP sensor was modified by a factor of 0.7 throughout a portion of the 60 second sampling interval in order to observe which problems IADS would suggest. The real (unmodified) data set²¹ is shown in **figure(6.a)** below. The black lines are the actual sensor values, and the white lines are the values that the model computed for each parameter.

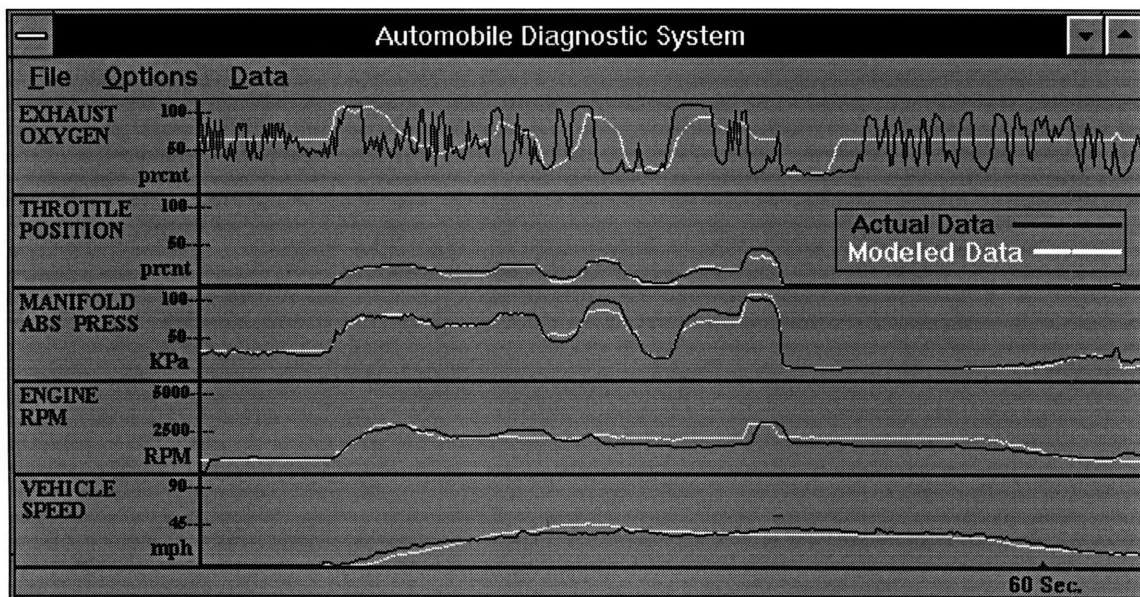


FIGURE 6.a An actual sample data set recorded over a 60 second interval.

The wild oscillations in the oxygen sensor cannot be modeled by the system, but there is no need to follow these sensors fluctuations precisely. The only time that the oxygen sensor is of interest is when it saturates at either a lean or rich fuel-air mixture. The system is capable of modeling the sensor when it saturates, primarily by monitoring the changes in the driver's acceleration demand (derivative of *TPS*). The

²¹ This data set does not include the following parameters: *MAT*, engine temp, battery voltage, and fuel level. These were omitted to make the graph fit the page well, and because they were not crucial for this example.

oscillations of the sensor output are partially due to the pulsation of exhaust gas corresponding to the pumping of each engine cylinder, and they can also be attributed to the closed-loop behavior of the ECM. The ECM continuously modifies the air-fuel mixture based on the acceleration demand. As long as the sensor fluctuates around the middle value of the exhaust oxygen level, the system is working properly.

This sample data set shows that IADS accurately models all of the input parameters. Now, the *modified* data set is shown in **figure(6.b)**. The map sensor data was factored by 0.7 between the 10 second mark and 45 second mark in the data set.

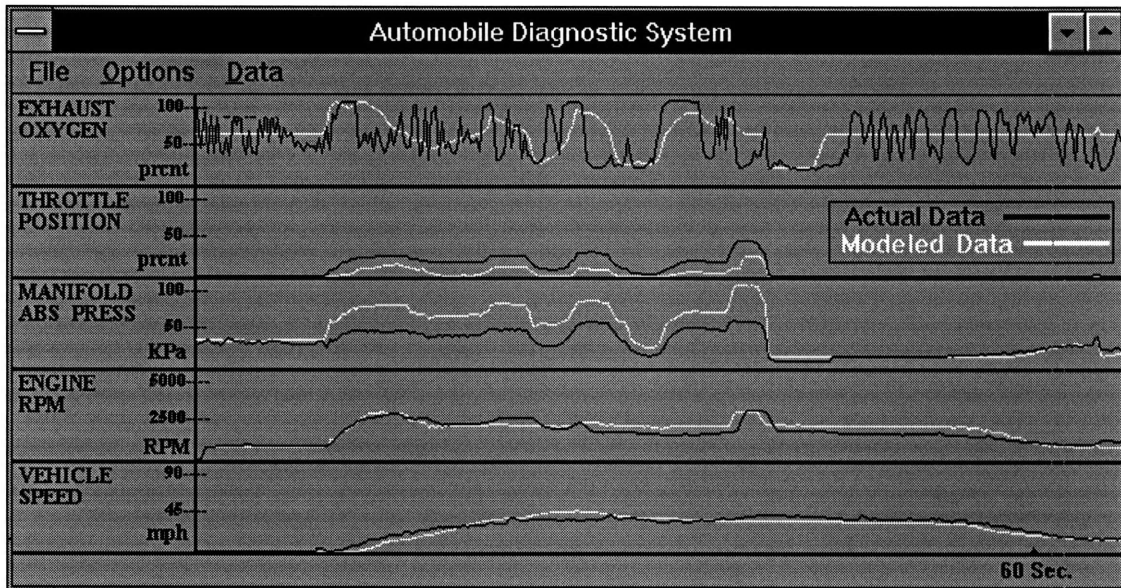


FIGURE 6.b Sample data set where the *MAP* signal was modified by a factor of 0.7 over a 35 second interval.

This data set shows that the system still models the *MAP* sensor correctly, but it also shows that the *TPS* model is lower than it should be. This is because the system uses the *modified MAP* value to model the *TPS*. Again, this is not a bad situation, since the system doesn't know if the *MAP* sensor is reading too low, or if the *TPS* is reading too high--both of these situations are definitely possible.

The final diagnostic outputs of the system for both the original data set, and the modified data set are shown below in **figure(6.c)**.

FIGURE 6.c

The final diagnostic results for the *MAP* example's original and modified data sets.

Diagnosis 1		Diagnosis 2	
1 Radiator Fan Stuck On (Engine too cool at low speed)	0.0349	1 Air Filter Clogged (Manifold air pressure too low)	0.2072
2 Fuel System Clogged (Exhaust oxygen too high)	0.0273	2 Radiator Fan Stuck On (Engine too cool at low speed)	0.0349
3 Spark Plug Gaps Incorrect (Exhaust oxygen too low)	0.0218	3 Fuel System Clogged (Exhaust oxygen too high)	0.0273
4 Idle Air Valve Stuck Open (Manifold air pressure too high)	0.0194	4 Spark Plug Gaps Incorrect (Exhaust oxygen too low)	0.0218
5 Thermostat Stuck Open (Engine too cool at high speed)	0.0180	5 Thermostat Stuck Open (Engine too cool at high speed)	0.0180

Diagnosis of Original Data Set Diagnosis of Modified Data Set

The only significant difference between the two output summaries is the clogged air filter shown in the second diagnosis. A certainty of 0.2 is fairly significant. The reason that it wasn't even higher is because the *MAP* signal was only modified for a portion of the data set--approximately 35 seconds. Since the *MAP* value was *correct* for the remaining 25 seconds in the set, and since the system normalizes its diagnosis for the entire sampling duration, the calculated certainty level wasn't extremely high. Even if the map sensor was significantly modified throughout the entire data set, the certainty for a clogged air filter still may not reach 1.0. A very low *MAP* reading does not necessarily indicate a clogged air filter, and the rules in the Diagnostic Rule Base reflect this. Perhaps the highest certainty that the rule base assigns to the possibility that the air filter is clogged is only *HIGH* certainty. For example, a simplified rule from the rule base states:

[if *MAPDIFF VERYLOW* and . . . , then *CLOGGED_AIR_FILTER HIGH*]

This rule translates as follows: if the actual *MAP* reading is much lower than the model predicts, then the certainty that the air filter is clogged is *HIGH*--not *VERYHIGH*. This assigns a certainty of 0.75 to this possible problem, so that even if this rule fired for every sample point throughout the entire data set, the final normalized certainty that the air filter is clogged would still only be 0.75. The discretion for assigning accurate certainties to each of the problem belongs to the expert who programs

the rules for the system. There certainly are several problems that have distinct symptoms which suggest the problem with *VERYHIGH* certainty.

For this diagnostic test, only one of the problems in the rule base was significantly affected by modifying the *MAP* input. The following example illustrates a situation where *several* problems are suggested by modifying a single input parameter.

6.2.2 Modified RPM Example

This second example utilizes another data set that was modified by increasing the *RPM* by a factor of 1.4 between the 10 second mark and the 60 second mark. The unmodified data is graphed below in **figure(6.d)**.

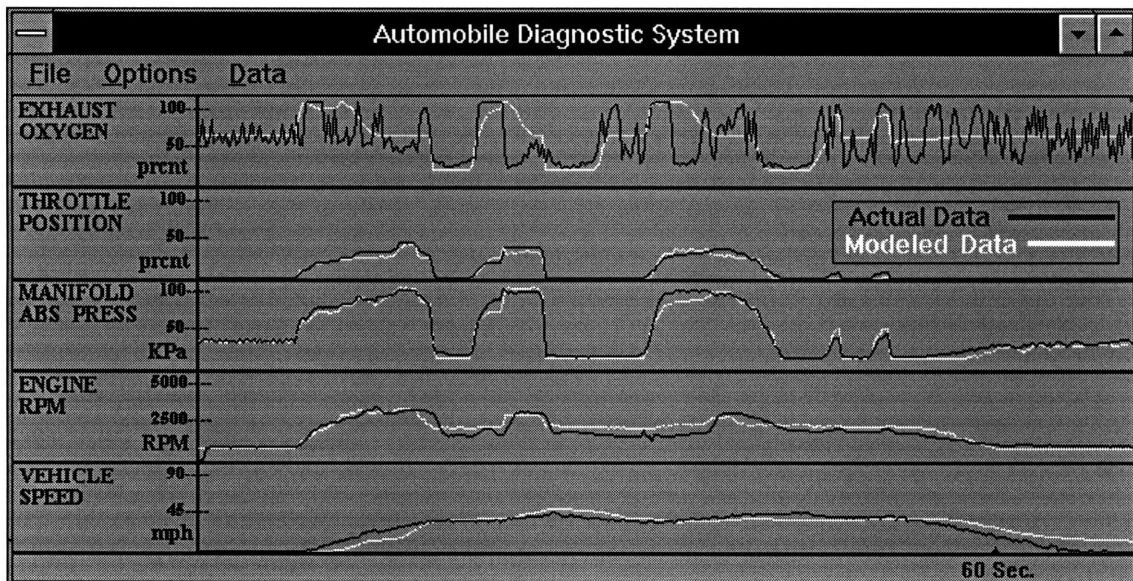


FIGURE 6.d Original sample data set used for the *RPM* example diagnostic.

Again, the diagnostic system modeled each of the parameters fairly accurately. The *modified* version of this data set is shown in **figure(6.e)**.

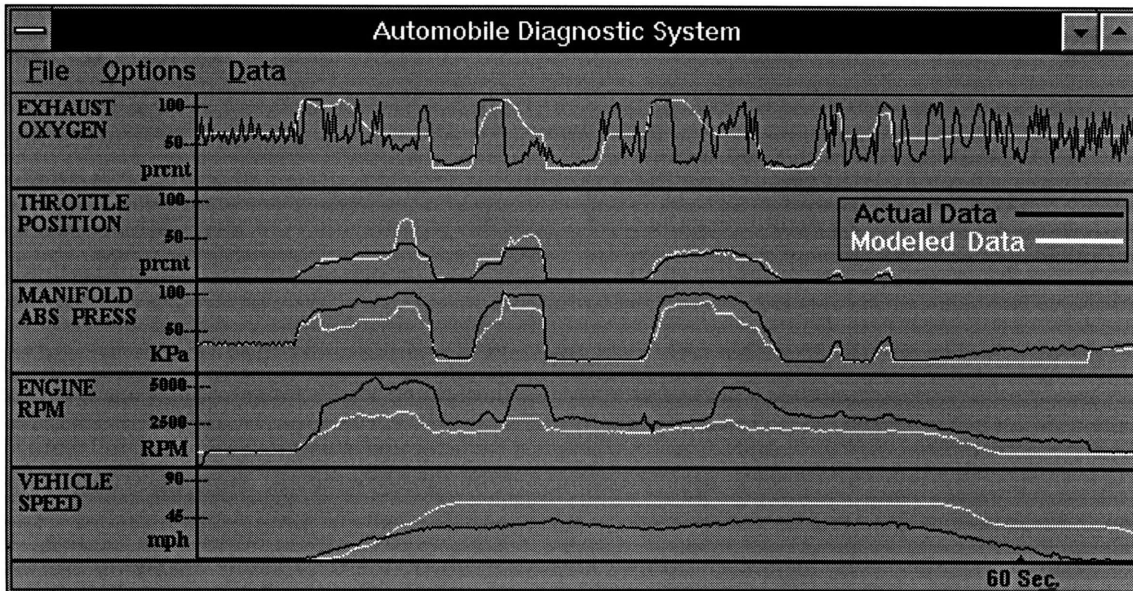
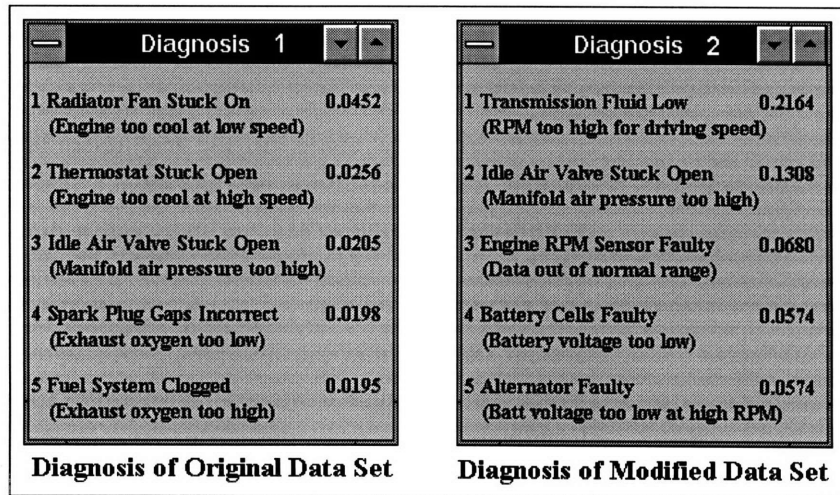


FIGURE 6.e Graph of the second example's modified data set where the *RPM* value was factored by 1.4.

This data set shows that the *RPM* value is still modeled accurately, but the model for the *VSS* is significantly corrupted, while the models for the *MAP* and *TPS* were moderately corrupted by the modified *RPM* input. Since several of the parameters were affected by this modified input, many of the diagnostic rules will be triggered in this example resulting in the system suggesting several possible problems. The diagnostic results for both the original and modified data sets for this test are shown in **figure(6.f)**.

FIGURE 6.f

Diagnostic results for the *RPM* test's original data set and modified data set.



In the *original* data set, the two highest possible problems indicated are a stuck radiator fan, (certainty of 0.045), and a thermostat valve that is stuck open (certainty of 0.026). Both of these are still fairly low levels of certainty, so these problem can safely be ignored. They were both triggered due to small discrepancies between the modeled engine temperature value and the actual engine temperature. Apparently the actual engine temperature was a few degrees cooler than the model expected. This overcooling condition can be caused in several ways, but the two that IADS suggests are: 1) if the car is moving slowly, and the radiator fan is constantly on, the engine will be overcooled, and 2) if the car is moving quickly, and the thermostat valve is constantly open allowing coolant to always flow through the radiator, then the engine will be overcooled by the rapid flow of air through the radiator. But, again, the certainty levels of both of these possible problems are low enough that they can be safely disregarded.

In the diagnostic results for the *modified* data set, several problems accumulated moderate certainty levels. The suggested problem with the highest certainty is low transmission fluid level. This situation would cause the engine to have to rotate faster than normal in order to achieve any driving speed. The transmission fluid acts as a coupling between the engine and the drive-train, and if the fluid is low, the coupling will not perform efficiently. Since the *RPM* signal was significantly increased in this scenario while keeping all of the other factors constant, this suggested problem is the most likely cause of the symptom.

The second suggested problem, an idle air valve that is stuck open, is a result of the modified *RPM* parameter's affect on the model for the *MAP* sensor. Since the model for the *MAP* relies heavily on the engine *RPM* (which determines how much vacuum is present in the air intake manifold), the high *RPM* level indicated a high vacuum (low pressure) condition, so the *MAP* model was lower than the actual *MAP* sensor value. Since the actual map value was higher than IADS expected, the system thought that the idle air control valve (IAC) was stuck open, which would cause a higher-than-normal *MAP* reading. The IAC is usually only open when the car is first started and the engine is warming up. Under normal driving conditions, the valve is almost always closed.

Although this suggested problem does not seem to be correct for this situation (since it is already known that the *RPM*--not the *MAP*--is the corrupted signal), if the engine *RPM* was in fact that high, then the suggestion would be an accurate explanation of why the *MAP* reading was too high for that condition.

The third suggested problem is a faulty *RPM* sensor. In general, the rules are written such that if a sensor has abnormally high or abnormally low values (that are inconsistent with the model) over a significant time interval, then the sensor itself may be responsible for the faulty readings. In this test case, a faulty sensor is also a fairly good explanation of the symptoms, since the *RPM* input is the only one that is significantly different from its model for a long time-interval.

The final two suggestions have to do with the output of the alternator. Since the *RPM* is so high, the system expects the voltage output of the alternator to be slightly higher than it would be under normal *RPM* conditions. This is highly dependent on the voltage regulation system in the vehicle. Some vehicle's have a noticeable fluctuation in alternator output voltage due to large changes in *RPM*, and if the *RPM* in this vehicle actually did reach the modified levels that are shown, the alternator output may have been slightly higher than the regulator is designed to output.

6.2.3 Engine Temperature Test

The final example is a long-term interval (five minutes), that is used to closely monitor slow engine temperature fluctuations. The engine temperature was modified by a factor of 0.8 in order to simulate an over-cooling condition. The graphs for both the normal temperature data set and the modified data set are shown in **figure(6.g)**.

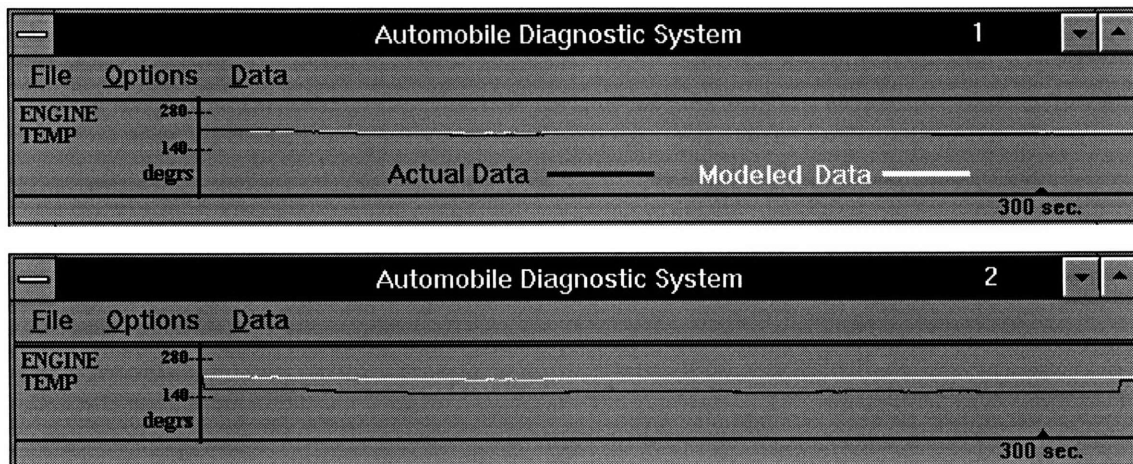


FIGURE 6.g Long-term data set collected from the vehicle to diagnose faulty temperature conditions.

Normal engine operating temperature is usually around 200°F. The first graph indicates that the temperature started at about 210°F, and then slowly cooled down to about 200°F. The reason for the slight cooling is because the car was at a stand-still at the beginning of the data capture interval, and then it accelerated to a moderate driving speed. When a car is sitting still, the engine slowly warms up until it reaches a defined temperature where the radiator fan turns on to cool the engine back down to a normal operating temperature. When a car is moving over approximately 30mph, the air-flow through the radiator is sufficient to keep the engine at its proper temperature. So, for this data set, the car was sitting still for a few minutes which caused the engine to warm up slightly, but then when the car reached a constant driving speed, the air flow through the radiator cooled the engine back to normal. The model follows along the actual temperature curve fairly well, since it is primarily based on the driving speed, engine *RPM*, and the intake-air temperature.

The modified data set shows that the engine temperature reaches a steady state at approximately 160°F. The resulting diagnosis for this scenario is shown in **figure(6.h)**.

FIGURE 6.h

Engine temperature diagnostic result.

Diagnosis 1		Diagnosis 2	
1 Fuel System Clogged (Exhaust oxygen too high)	0.0320	1 Thermostat Stuck Open (Engine too cool at high speed)	0.3953
2 Alternator Faulty (Batt voltage too low at high RPM)	0.0132	2 Engine Temp Sensor Faulty (Data out of normal range)	0.2118
3 Battery Cells Faulty (Battery voltage too low)	0.0132	3 Fuel System Clogged (Exhaust oxygen too high)	0.0320
4 Air Filter Clogged (Manifold air pressure too low)	0.0120	4 Radiator Fan Stuck On (Engine too cool at low speed)	0.0267
5 Thermostat Stuck Open (Engine too cool at high speed)	0.0106	5 Alternator Faulty (Batt voltage too low at high RPM)	0.0132

Diagnosis of Original Data Set Diagnosis of Modified Data Set

The diagnosis of the *original* data set has very low certainty levels for all of the problems that it suggests, so they can all be safely disregarded. The results from the diagnosis of the *modified* data set show that the most likely problems are either a clogged thermostat valve, or a faulty engine temperature sensor. As explained in section 6.1.2, when the thermostat is stuck open, the coolant continuously circulates to the radiator and is overcooled by the rapid air flow through the radiator core. The thermostat valve is designed to close when the engine temperature falls below 180°F, since the optimum engine temperature is around 200°F. Closing the thermostat valve prevents the coolant from circulating through the radiator, which is the main element of the engine's cooling system. Since the vehicle was traveling at high speed (which facilitates rapid radiator cooling), and since the system detected that the coolant temperature was lower than normal, it concluded that the thermostat valve must be stuck open.

The other main factor associated with over-cooling is when the radiator fan is stuck in the 'on' position. But in this case, since the vehicle was traveling at highway speeds for most of the sampling interval, the radiator fan does not significantly affect engine cooling, so it was only assigned a very small certainty level (0.027). The reason that the certainty was even that high is because the certainty accumulated during the short time interval when the car was at a standstill at the beginning

of the sampling period before it reached cruising speed. Again, since the certainty levels are normalized over the entire data set, and since the radiator fan rule only applied for the first few seconds when the car wasn't traveling quickly, the thermostat suggestion received a much higher certainty level than the radiator fan suggestion.

The problem that received the second highest certainty level is the faulty temperature sensor condition. Since the temperature value was abnormally low over a long time interval, the rule concerning a faulty sensor accumulated a significant certainty. As explained in section 6.1.2, when a parameter is either abnormally high or abnormally low for a relatively long time, then a faulty sensor is usually one of the most likely problems. The rest of the suggested problems in this diagnosis all have insignificant certainty levels.

Chapter 7 Conclusion

The test results in chapter 6 indicate that IADS did in fact meet most of its design objectives.

7.1 Meeting the Objectives

The general objective of this project was to develop an inexpensive, comprehensive, flexible automobile diagnostic system and to demonstrate the system's ability to accurately diagnose the vehicle's problems by analyzing data collected from the existing vehicle sensors.

Discussion of how well each main objective was met:

1) Create a diagnostic system that utilizes as many existing parts as possible (e.g. any IBM compatible laptop PC, and the existing vehicle sensors).

IADS did, in fact, primarily utilize commonly available components: There are no highly specialized ICs in the interface module, and the only other hardware required is a PC and some cable and connectors. The system also strictly utilized the existing sensors on the vehicle to perform all of its diagnostic functions. Although additional sensors would greatly enhance the diagnostic capability, the results indicate that IADS can make an accurate diagnosis based only on the existing sensors.

2) Provide the user with a real-time graphical display that shows the behavior of all of the vehicle's sensors over specific time intervals.

The PC was programmed for the Windows environment so that it could display a graph for each of the eight sensor signals. The graphing feature can also simultaneously display the modeled data, the integral data, the derivative data, and the model-difference data.

3) Provide the user with an inexpensive, informative tool for monitoring their vehicle's performance.

The cost of the entire system is extremely low, compared to the industrial systems that many auto shops own. If the car-owner

already has a PC, the material cost for the rest of the IADS system is very low.

4) Make the diagnostic tool relatively easy to operate.

Because the software runs in the Windows environment, the graphical user interface for the system is very simple to use. There are only a few parameters to set, and only a few commands needed to run the system.

5) Enable the system to perform all of the diagnostic functions non-invasively.

Since the system only monitors the existing vehicle sensors, and the only connection that is required is the T connector at the ECM input, IADS is definitely a non-invasive diagnostic system.

6) Demonstrate the system's ability to perform an accurate diagnosis using only the existing vehicle sensors.

The results of the three tests described in chapter 6 indicate that the system is very capable of performing accurate diagnostics using only the existing sensors.

7) Demonstrate the benefits of using fuzzy logic for both the modeling and model based reasoning (diagnosis) in the system.

The relative simplicity with which the diagnostic and modeling systems were implemented, indicates that fuzzy logic was an excellent design choice for IADS. The alternative option would have involved deriving complex mathematical equations to model the system, which would have been much more complex, much less flexible for future modifications, and possibly less accurate than the fuzzy logic implementation.

8) Demonstrate the system's ability to perform accurate diagnostics of slowly changing parameters that would normally be very difficult for technicians to monitor.

The test example that monitored engine temperature demonstrates that IADS is capable of accurately modeling and diagnosing slowly changing parameters. However, the objective was to enable the system to perform a much larger variety of diagnostics, especially using the information from the fuel level

sensor. IADS could not completely meet this objective because the existing fuel level sensor produced a very unstable output.

9) Demonstrate that the system offers many benefits over existing diagnostic technologies.

Although the highly specialized diagnostic systems that automotive technicians utilize have many advanced features and capabilities that IADS does not have, IADS does offer many benefits to these expensive, complex machines. The cost of IADS is much lower, it is a completely portable diagnostic system, it is very easy to update the rule base, and it is attainable and useable by almost anyone.

7.2 Suggested Improvements

There are many enhancements that can be made to the system in order to greatly increase its ability to produce accurate diagnostic results.

Additional sensors would provide greatly needed information for the knowledge bases to use for both modeling and diagnosis. There is a wide variety of invasive sensors that current diagnostic systems use to monitor specific engine functions. Some of these sensors include, engine compression testers, fuel pressure monitors, engine knock sensors, mass air flow sensors, etc.

Simply monitoring more of the existing systems in the vehicle would also improve the system's diagnostic capability. There are several potentially useful status signals that IADS does not currently monitor, such as the air conditioner compressor signal, that indicates if the A/C compressor is active (which causes a significant load on the engine).

Enhanced interface circuitry would allow the system to monitor some of the rapidly changing signals produced by the ECM. The current interface is not capable of accurately measuring the time duration of the injector pulse widths, so IADS does not know how much fuel is being delivered to the engine.

Connecting IADS to the ECM's serial interface instead of tapping directly into the sensor wires offers both benefits and drawbacks. The sampling rate of IADS would be limited by the serial communication rate, so the system would not notice subtle variations in the sensor output. However, connecting to the serial link would enable IADS to gather more status information concerning the operation of the ECM.

Combining IADS with an interactive expert system that engages a technician in a short dialog, would significantly enhance the system's ability to make accurate diagnostic decisions. The technician can input information about other major symptoms that IADS is not capable of noticing, such as a scratching sound from the brakes, or the unusually loud rumble of the engine. Other valuable information would include the age of the car, the mileage on the car, the repair history of the car, etc.

Implementing a learning ability into the system would greatly improve the accuracy of diagnostics, since the system could base its decision on similar prior symptoms that commonly indicate a specific problem. The system could also modify its diagnosis based on feedback from the technician. If the system suggested a problem, and it turned out to be correct, the system can store this as a valid result. But if the suggested problem was incorrect, then the system may adjust its certainty level for that problem in the next diagnosis.

If the fuzzy logic portion of the system was *self* learning, then it could tune its fuzzy sets to optimize its modeling ability based on data collected from various vehicle's under many different driving conditions.

Increasing the number of ranges (subsets) allowed in each fuzzy set could also improve the accuracy of the modeling phase. This would, however, greatly increase the size and complexity of the system's rule base.

Most of these suggestions for improving IADS would require significant enhancements and would greatly increase the complexity of the system. But a few of the suggestions could be relatively easily incorporated into

the current design (such as improving the hardware so that the system can monitor more of the existing ECM signals).

7.3 Summary

The Intelligent Automobile Diagnostic System was designed to be an inexpensive, flexible, comprehensive diagnostic tool for automobile control systems. Although the system was not tested on a vehicle with an actual known faults, the examples using software simulated faults demonstrated that IADS achieved almost all of its design goals. IADS is easy to use, relatively inexpensive, easily modified, and demonstrates an extremely useful application of fuzzy logic. Without utilizing fuzzy logic, this project would have been much more complex, would have required a significant increase in development time, and would not be as flexible as the current system. If the operation of a system can be described linguistically, then it can be controlled with fuzzy logic. When a system needs to be developed that can monitor, model, or control a complex system, fuzzy logic will most likely be the best overall tool for the job.

Chapter 8 References

- [1] de Baat, G.A. *Self-Test Features on an Engine Management Microprocessor*. Fourth International Conference on Automotive Electronics, Nov. 1983. Institution of Electrical Engineers, London 1983.
- [2] Bromley, P.J. *Advanced Automotive Electronics*. IFS Publications, U.K. 1989.
- [3] Cox, Earl. *The Fuzzy Systems Handbook*. Academic Press, Inc. 1994.
- [4] Fisher, Bill, Helen Fisher, Howard Fisher. *Electronic Fuel Injection Troubleshooting Guide*. Mitchell International 1991.
- [5] Haynes, John, Larry Warren. *GM J-Cars: Automotive Repair Manual*. Haynes Publications 1990.
- [6] Horowitz, Paul, Winfield Hill. *The Art of Electronics*. Cambridge University Press. 1989.
- [7] Idogawa, Yoshiyuki, Kazuo Funabashi, Yukio Maehashi. *A New Microcomputer for Engine Control*. Electronic Engine Management and Driveline Controls. Society of Automotive Engineers, Pennsylvania 1982.
- [8] Ingersol, Gary. *Chilton Repair Manual*. Chilton Book Company 1988.
- [9] Kandel, Abraham. *Fuzzy Expert Systems*. CRC Press. 1992.
- [10] Kobayashi, Kazuo, Naoki Hiwa, Kazuo Nil, Tatsuji Ikeda. *Diagnostics for Vehicle Electronics--Present and Future*. Proceedings of the 1992 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1992.

- [11] McElroy, R.C. *Automotive Engine Electronics*. Accuracy Publishing Co. 1991.
- [12] McNeill, F. Martin, Ellen Thro. *Fuzzy Logic, A Practical Approach*. Academic Press Inc. 1994.
- [13] McNeil, Daniel, Paul Freiberger. *Fuzzy Logic*. Touchstone. 1993.
- [14] Messmer, Hans-Peter. *The Indispensable PC Hardware Book*. Addison Wesley 1994.
- [15] Onesti, Robert J., Robert D. Dannenberg. *A Strategy for Diagnosing Modern Truck Electrical Systems*. Proceedings of the 1990 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1990
- [16] Pepper, Jeff, Dennis R. Mullins. *Artificial Intelligence Applied to Audio Systems Diagnosis*. Proceedings of the 1986 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1986.
- [17] Ribbens, William B., Norman P. Mansour. *Understanding Automotive Electronics*. Howard W. Sams & Co. 1984.
- [18] Rodda, William J., John L. Kastura, Stephen T. Mahan. *Automotive Electronic Implications of OBD II*. Proceedings of the 1992 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1992.
- [19] Tedesco, L.S. *Service Bay Diagnostic System*. Proceedings of the 1986 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1986.
- [20] Tennant, Harry. *Onboard Knowledge Systems in Vehicles*. Proceedings of the 1990 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1990.

- [21] Walters, William L., Michael J. Thielan. *Integrated Vehicle Systems Diagnostics--Powertrain and Chassis*. Proceedings of the 1986 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1986.
- [22] Weathers, Tom Jr. *Automotive Computers and Control Systems*. Prentice-Hall, Inc. New Jersey 1984.
- [23] Weishaupt, W. *Technical Aspects of the BMW On-Board and Service Diagnostic System*. Proceedings of the 1984 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1984.
- [24] Yester, J.L., R.H. McFall. *Fuzzy Logic Controller for Active Suspension*. Proceedings of the 1992 International Congress on Transportation Electronics. Society of Automotive Engineers, Pennsylvania 1992.
- [25] Yu, T., G. Rizzoni. *A Floating Point Co-Processor for Real-Time Fault Detection and Isolation in Electronically Controlled IC Engines*. Eighth International Conference on Automotive Electronics. Institution of Electrical Engineers, London 1991.

7131-58