

CR-194228

SBIR-06.06-5550
rel date 7/23/92

Final Report For NASA Phase II SBIR NAS9-17995
July 1990

1/N-16-CR
107
p. 250

SPACE TRANSPORTATION ANALYSIS
AND
INTELLIGENT SPACE SYSTEMS

(NASA-CR-194228) SPACE
TRANSPORTATION ANALYSIS AND
INTELLIGENT SPACE SYSTEMS Final
Report (Netrologic) 250 p

N94-70972

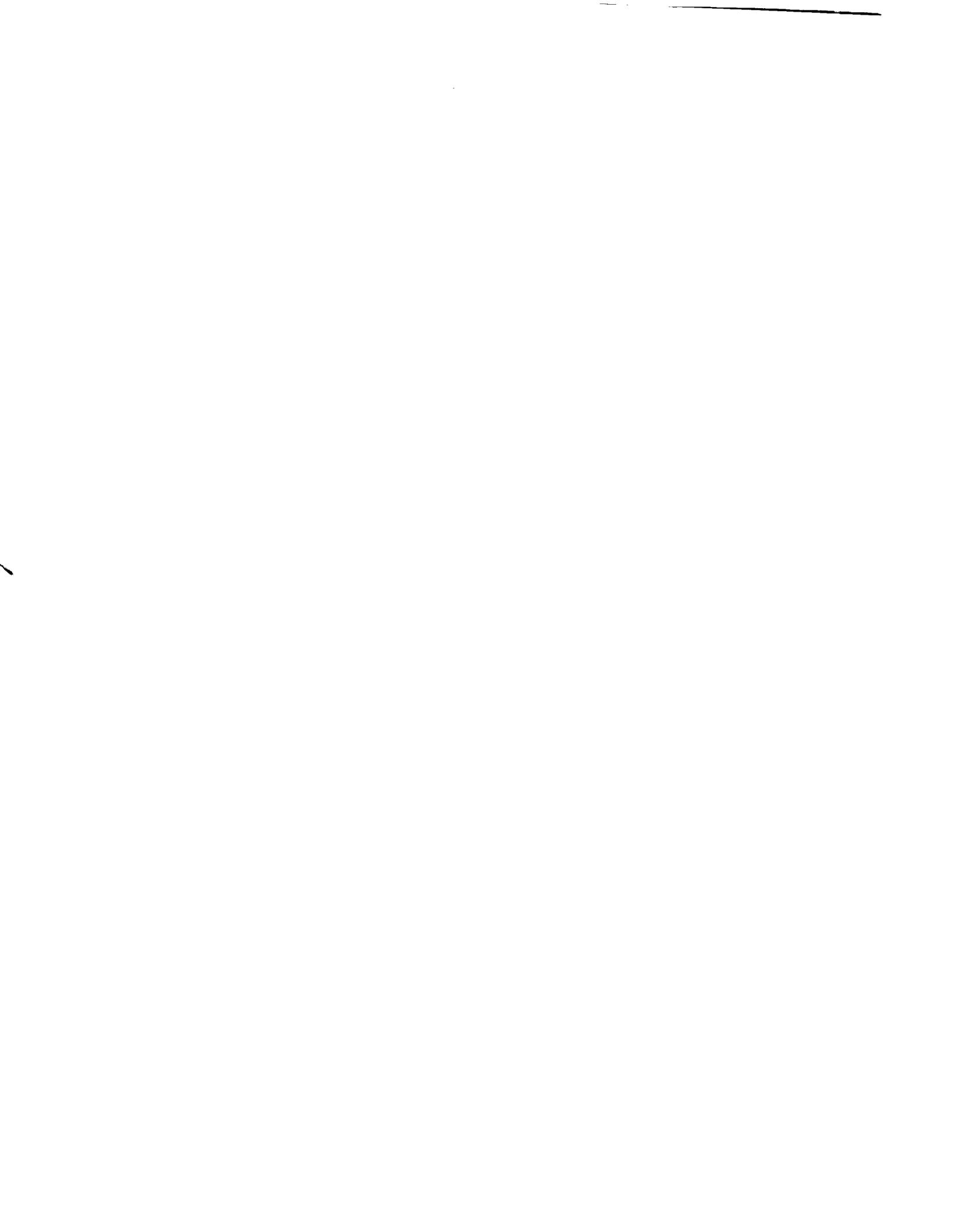
Unclas

Z9/16 0183168

Submitted To:

NASA
Lyndon B. Johnson Space Center
Houston, TX 77058

NETROLOGIC, Inc. • 5080 Shoreham Place • Suite 201 • San Diego, California 92122



ABSTRACT

This report describes work carried out in support of a NASA sponsored two year SBIR program to investigate the applicability of neural networks to spare systems. NETROLOGIC and our subcontractors identified four different applications where neural networks offered significant computational advantages. These applications were rocket engine health monitoring, rocket control valve identification, cryogenic fuel mass estimation and industrial weld estimation.

The first work described addresses the problem of on-line anomaly detection and fault typing for the space shuttle main engine (SSME). The basic method involves sensor data compression through neural nets and uses features representing time-variance of and relationships between SSME sensor values. Actual test-stand test case data was employed for training and testing.

Recognition of valve signatures in the ATLAS rocket (which contains some 150 valves of various types), was accomplished using feedforward neural networks. In this case, data was collected in the form of electric current transient signatures from ATLAS rocket valves installed on a pneumatic test bench. The data was then used as training and test data for two neural networks, one which was trained to distinguish between signatures for individual valves falling into three separate types of valves during a valve-open state change (rising current), and the other which was trained to distinguish between signatures for the same three types of valves during a valve-close state change (falling current).

Another effort described involved the use of feedforward neural networks to gauge cryogenic fuel mass by estimating the amount of mass in the tank from spectral resonance measurements. Two types of networks were investigated, the first employing a compact representation of the input space, the second retaining all the information in the resonance measurements.

Finally, neural nets were used to determine the quality of an industrial inertia weld by examining data accumulated during the welding operation.

The techniques employed to solve the problems above from different areas of health monitoring for space propulsion systems can also be applied to many similar problems in space transportation analysis.

Acknowledgement

The staff of NETROLOGIC who have been most active on the SSME project are Dan Greenwood, who served as principle investigator, Fareed Stevenson, Thomas Bishop, James Johnson, Spencer Menlove and Dr. Sue Walker Toledo. In addition, Claudia Meyer and Larry Cooper of NASA Lewis have provided invaluable assistance in obtaining SSME data and explaining the subtleties of interpreting it, Dr. Chris Bowman of Ball Corporation helped with the original data analysis and the original configuration of the neural networks, and James Villareal of NASA Johnson Space Center provided programmatic support.

The principal investigators on the work concerning the classification of industrial inertial welds were James Johnson and Jeffrey Sumner of NETROLOGIC.

NETROLOGIC had several subcontractors during the Phase II program. General Dynamics performed the major effort on the rocket valve analysis under the direction of Dr. Gerry Szatkowski, Ball Corporation assisted in all of the program under the direction of Dr. Chris Bowman, and HNC assisted in the cryogenic fuel mass estimation project. Dr. Terence Smith and Mr. Gil Pitney of the University of California at Santa Barbara assisted in evaluating network approaches early in the program.

Table of Contents

ABSTRACT	ii
Acknowledgement	iii
SECTION 1: SUMMARY	1-1
SECTION 2: SPACE SHUTTLE MAIN ENGINE FAILURE DETECTION	2-1
2.1 Problem Statement	2-1
2.2 Basic Network Structure	2-8
2.3 Data Source and Description	2-8
2.4 Pre-Processing of Data	2-13
2.5 Assignment of roles to output units	2-19
2.6 Fault-declare Times and Net-Structuring Issues	2-21
2.7 Fault-Typing	2-26
2.8 Training Method and Results	2-27
2.9 Other SSME Failure Detection Systems Netrologic Has Been Investigating	2-53
2.10 Recommended SSME Neural Net Investigation for the Near Future	2-54
SECTION 3: VALVE SIGNATURE RECOGNITION	3-1
3.1 Problem Statement	3-1
3.2 Technical Background	3-2
3.3 Test Design	3-6
3.3.1 Valve Types	3-6
3.3.1.2 <i>Wright (REM) Valve</i>	3-6
3.3.1.3 <i>Marotta Valve</i>	3-7
3.3.2 Signature Characteristics	3-9
3.3.3 Signatures for the Valve Types	3-10

3.3.4	Smart BIT Data Collection and Format	3-15
3.3.5	Neural Network Design	3-17
3.4	Test Results	3-18
3.4.1	Training Procedure	3-18
3.4.2	Test Procedure and Results	3-19
3.5	Future Research Directions	3-22
3.5.1	Failure Determination	3-22
3.5.2	Failure Prediction	3-27
3.5.3	Temperature Effects	3-28
3.5.4	Pressure Effects	3-29
3.5.5	Multiple Valve Signatures	3-30
3.5.6	Gravity Effects	3-31
3.5.7	Valve Consistency	3-32
3.5.8	Conclusions	3-32
SECTION 4: RF GAUGING OF CRYOGENIC FUEL TANKS USING NEURAL NETWORKS		4-1
4.1	Problem Statement	4-1
4.2	Experimental Data	4-2
4.4	Approach 2 - Mode Averaging Network	4-3
4.5	Discussion of Approach 2	4-8
4.6	Approach 3 - Mode Assignment Network	4-9
4.7	Discussion of Approach 3	4-10
4.8	Conclusions	4-11
SECTION 5: CLASSIFICATION OF INDUSTRIAL INERTIAL WELDS USING NEURAL NETWORKS		5-1
5.1	Problem Statement	5-1
5.2	Data Format	5-1
5.3	Implementation of the Neural Network	5-2
5.4	Testing the Neural Network	5-3
5.5	Results of the Neural Networks	5-4

SECTION 6: RECOMMENDATIONS FOR FUTURE WORK	6-1
6.1 SSME Health Monitoring	6-1
6.2 Valve Signature Identification	6-2
6.3 Cryogenic Fuel Tank Mass Gauging	6-3
6.4 Industrial Weld Quality Gauging	6-3
SECTION 7: BIBLIOGRAPHY	7-1
APPENDIX A	A-1
APPENDIX B	B-1
1 Data File Formats, Valve Signature Investigations	B-1
B.2 Neural Network Data Files	B-12
B.3 List of Figures in Section 3	B-13
B.4 List of Tables in Section 3	B-16

SECTION 1

SUMMARY

This report describes work carried out by NETROLOGIC and its subcontractors in the area of health monitoring for space propulsion systems, rocket engine control valve identification, cryogenic fuel mass extraction and fuel mass estimation.

The general problem of determining when an engine/machine failure has occurred received the largest amount of effort. The methods developed, although general, were applied specifically to the problem of on-line anomaly detection and fault typing for the space shuttle main engine (SSME).

The problem was to decide when to shut down an SSME based on data from a set of sensors monitoring basic operating modes of the engine. The method under investigation involved the computation of features representing the time-variance of and relationships between SSME sensor values, using actual test-stand test case data. The choice of appropriate features requires detailed expert analysis of the expected behavior of the sensors both when the engine is functioning normally and when a failure occurs. In the investigations that have been carried out so far, networks have been trained, using backpropagation, to recognize a set of samples, corresponding to both fault and nominal conditions, using features calculated from the sensor data recorded in test firings. After such training, the network was able to diagnose nominal and fault samples from cases withheld from the training data set with a high degree of accuracy.

The second effort addressed rocket valve signature recognition. In this case, data was collected in the form of electric current signatures from ATLAS rocket valves installed on a pneumatic test bench. The data was then used as training and test data for two neural networks, one being trained to distinguish between signatures for a valve-open state change (rising current), and the other to distinguish between signatures for a valve-close state change (falling current). No errors were made on any of the training or test patterns by either network in determining which valves were generating the signatures when noise was not present, and both networks did very well when noise was limited to 10% or less. When the amount of noise was very high, the networks could still distinguish between which of three types of valves was generating the signal, but began to lose the capacity to distinguish between individual valves. The success of neural nets to solve these relatively difficult problems (potentially valuable for the reduction in hardware circuitry that could result), as well as careful analysis of valve signatures in different valve states, makes it clear that neural nets could distinguish between the states of valve opening, valve closing, valve stuck closed attempting to open, valve stuck open attempting to close, etc.

Another investigation involved the use of feedforward neural networks to gauge cryogenic fuel tanks by estimating the amount of mass in the tank from input spectral resonance measurements. Two types of networks were applied, the first employing a compact representation of the input space, the second retaining all the information in the resonance measurements. It turned out that the second method required a larger training set than the first. Both types of networks were able to learn to estimate masses, with differing degrees of accuracy.

The last piece of work described in this report concerns work on detection of welding failure during the construction of a jet engine. Its goal was to develop a neural network to determine the quality of an industrial inertial weld by examining data accumulated during the welding operation. Some of the networks tested showed 100% accuracy on the data of unknown status that was presented to them.

Note: All of the investigations mentioned in this report used feedforward neural networks trained with back-propagation. Feedforward networks can be trained to associate arbitrary input patterns with arbitrary output patterns, and have the ability to *categorize* and *generalize*, so that similar inputs can be expected to map to similar outputs, and new input patterns, different from those on which the network has been trained, will be mapped to outputs based on their similarity to training patterns.

In the basic operation of feedforward networks, connections are one-way, going from inputs to the hidden layer to outputs (hence the name feedforward). Units in neural nets are analogous to neurons. The connections between them are analogous to synapses. Each of the connections in a neural network is characterized by a weight at every point of the process, which is considered to be the strength of the connection. Each unit attains a level of activation, which is a function of the weighted sum of its inputs.

The choice of how many hidden units to have in each layer is dictated by two opposing factors. It is generally easier for a network to learn how to map inputs to outputs if there are more hidden units, but if there are too many hidden units, the network is liable over-specialize itself to in the particulars of training data employed, with the result that it may be less successful at generalizing with new data.

Backpropagation is a very powerful method to use for supervised learning, and requires allowing a network to train itself with the correct characterizations of the samples in a training data set. During training, a comparison is made between the indicated correct characterization and the characterization given by the network for every input and output target pair in the training data set. Based on the difference between the target vector given to the net and the results arrived at by the net, the neural network alters relevant connection strengths so that its results for each training sample are closer to being correct than before.

In other words, after each pass through the training data set, this so-called "back-propagation of error" usually enables the network to get closer to the correct results on more and more of its samples. Eventually, the network's results for every case should be close enough (as determined by a training tolerance) to the correct results so that the network converges and training can be terminated. When the network results satisfy the training tolerance decided upon, the network is said to have converged. For more details about such neural networks, the reader may consult [6], or any of the many basic books currently addressing the subject.

SECTION 2

SPACE SHUTTLE MAIN ENGINE FAILURE DETECTION

2.1 Problem Statement and Summary of Investigation and Results

The problem of deciding to shut down a space shuttle main engine (SSME) when a malfunction has apparently occurred is a very important one. The cost of a single SSME is now given as 45 million dollars. The loss of a complete shuttle amounts to several billion dollars in hardware alone, and the lives of human beings are involved as well. A shut-down without need is also expensive on the ground. Cost estimates are harder to calculate regarding this situation, but the whole test run must be repeated, and the cost of a single test run has been loosely placed at a million dollars. Furthermore, in space, shutting down one of the three SSMEs on the shuttle is considered to increase the general risk involved in the flight.

In order to recognize danger signs early enough to shut down the rocket engine and eliminate or minimize damage, an SSME fault detection system must be faster and more accurate than existing systems. It appears that this should be possible, for with the current test-stand failure response systems, which utilize automatic redlining, redundant sensors and controller voting logic, along with human monitoring, post-test analysis of sensor data often showed an indication of anomalous engine behavior well before a shutdown sequence was initiated. Neurologic now has ample evidence that neural networks and related deterministic data fusion methods can provide improved SSME test-stand fault detection in ways that have natural extensions to in-flight monitoring.

A fast SSME diagnostic method is essential since a large number of simultaneous sensor measurements (over 200 are available) are input to a test shutdown decision module at a high sampling rate. Sensor data fusion and evaluation are complicated processes since clues to engine performance may involve subtle combinations of sensor measurements varying through time. Practical considerations dictate that a detection system cannot alter the current engine or control system and should utilize existing data. Since the SSME's major components are line replaceable units, ideally a fault detection system should be independent of engine-to-engine performance variation.

Neural networks contribute to effective failure detection since they are:

- 1) fast, especially if implemented on parallel hardware;
- 2) capable of discovering subtle patterns in input data without being explicitly taught what combinations are significant;
- 3) capable of generalizing based on previously learned examples of actual test data;
- 4) robust --- relatively insensitive to noisy data; and
- 5) easily adapted to accommodate new information as it becomes available.

It is not a simple thing to evaluate the success of an approach to the SSME shutdown problem. At the very least, a method should not have any false alarms in firings of engines that encountered no major difficulties (in the usual data set, case 457 and 463), and it should find all faults that manifest themselves in data meeting the prerequisites of the method.

But if methods find the fault in a case at times, possibly different than those pointed to in the failure investigation reports (as when sensors began changing due to the fault), then it is difficult for those who are not rocket engineers, who know about all the properties of the first-generation engine that encountered the fault to know which of the times should be considered correct. There are many small irregularities in the data of even very healthy engines, and when a method detects one irregularity in a fault case, especially one of those where earlier fault detection than indicated in the failure investigation reports could save an engine, it is natural for the investigator to assume that the method is correct in declaring a fault. Evidence of this effect in the fault-declare time taken for case 307 is given below. Investigators were influenced by the outputs of earlier networks, and then by irregularities they noticed in the raw sensor data. They decided to act on the assumption that their method was correct when these tendencies were encouraged by conversations with others who felt they or their methods had also detected the fault earlier in case 307 earlier than the failure investigation reports.

Thus, for investigators who are trying to develop computational methods that might detect irregularities in the data earlier than is easily done manually, it is difficult to determine when they have succeeded if the results are not in some way confirmed by further work by an analyst. A carefully designed fault simulator would be invaluable.

For the present, it is reasonable to expect the following of an SSME failure-detection system.

- a) If the SSME fails in a new way, faults will still be detected.
- b) The system uses more than one sensor's values for a given point in time, and considers relationships between them.
- c) It uses relationships between data at different points in time, from different situations (power level, venting/repressurizing, start-up, transient or steady-state situation), involving single and multiple sensors.
- d) It relates data from different cases.
- e) It takes into consideration the parameters of a particular engine, and of a particular firing.

A complete SSME failure-detection system can consist of several different algorithms/nets that simultaneously give their estimates of the health of the engine, which are then integrated by another algorithm. For instance, algorithms with different aims might be running simultaneously - ones with emphasis on incurring no false alarms running alongside ones that try to find a fault as early as possible (looking for any irregularity that might arise). Thus not all the criteria listed above need be satisfied by any one part of the system, but all should be satisfied by at least one.

And a different set of algorithms might be in force in different situations. For instance, when an engine is being tested on the test-stand for the first time, algorithms should not omit comparing sensor values produced by the engine to what might be expected of the sensor values, based on experience with other engines. After the first time an engine is fired, the failure-detection system might focus on comparing the engine with its own past runs, or with its self earlier in the current run, once it has shown itself to be healthy at start-up and settled down well into its first steady-state period.

The following very different kinds of neural networks can all provide valuable information relevant to the SSME shutdown issue. Each has its strengths and its weaknesses.

The Kohonen Novelty Detector.

This type of network is trained using data of a certain kind, and indicates when data of a different kind is encountered.

For the SSME shut-down decision, it would be appropriate to train a Kohonen Novelty detector network on a very large amount of unquestionably nominal data, i.e. data which represents the whole space of nominal data. This data should include data in which there were minor faults not requiring engine shut-down, in order to avoid shutting down the engine without need. It is inappropriate to train a net with the apparently nominal data sampled in the fault cases before the failure investigation reports indicate that the fault is showing in the sensors, since it is desirable to have as large a possible set of cases to test the trained network on. All these cases should be tested on a network that has not seen them. Moreover the main information investigators need from such a network is where, in the only fault cases that exist, the data first begins to look different from data in engines not requiring shut-down. This decision should not be biased by including data in the training set that possibly corresponds to a situation in which engine shut-down should already have been initiated.

To test this kind of network, a broad spectrum of first generation nominal data is required, taken from cases that did not experience major faults. After the behavior of the network is ascertained in that case, the same kind of network can be trained on a broad spectrum of second generation nominal data. It is likely to be meaningful to test such a network on the first-generation fault cases, for reasons that will be gone into in some detail below.

The Kohonen Feature Mapping Network or ART2.

These networks, like the first kind of network mentioned, employ what is called "unsupervised learning". In the case of supervised learning a network is told how to separate its input space into categories relevant to the problem, while in unsupervised learning, the network itself divides up the space.

The use of this kind of network was postponed since it would be more significant to use such networks with a broader set of nominal data, in particular, second generation nominal data. The value of this kind of network is that it will divide the input space up into as many categories as specified, assuming it can find a way to do so. But once these categories are created, investigators will have to have informed engineering input to determine whether the categories are important for the problem, and most shuttle engineers now work only with second-generation engines, and the development of these networks would involve many months. Such networks, when presented with data from a fault case and asked to divide the space into two categories, did divide it at approximately the fault-declare time given in the failure investigation report (cf. [11]).

Recurrent Networks and Networks with Data Windows.

Recurrent networks and networks employing data windows supply a segment of the data stream to the network as a single input, i.e. a certain number of usually consecutive samples are input to the network simultaneously. This is important when networks are being constructed to explore the data provided to them (i.e. find correlations, patterns) as much as possible, and is always worth exploring if there is sufficient time.

There are many ways of encoding information involved in the stream of data that comes into an algorithm in sequence. For example, it is possible to calculate "features" from the data stream involving relationships between data at different points in time that investigators feel would be valuable for the fault-detection method to consider. This latter method allows more control over what a network is likely to draw its conclusions from, and makes it more efficient for networks to determine categories naturally found through consideration of the properties of the data reflected in the features. Calculating relevant features is also more efficient in terms of space and time during the testing period.

Since, based on the promising SAFD results, it was determined that a set of five to ten simple features is sufficient to handle the SSME shutdown problem, the path of exploring features rather than data windows was pursued.

Feed-Forward Network Employing Feature Data, and Trained by Back-Propagation; failure investigation report decisions taken as the guideline for the time of fault occurrence.

As just mentioned, this SSME fault-monitoring method is the kind of network that was chosen for the principle investigations of the project.

This method has the advantage of being very effective at avoiding false alarms, and thus might be more suited for test-stand monitoring, while a more sensitive method could be favored in-flight (e.g. Kohonen Novelty Detector). As indicated above, methods of both these kinds could be run in the two situations, and integrator algorithms could integrate the results differently in the two different situations.

Since the fault-declare time given in the failure investigation reports is a time at which the fault has unquestionably begun to occur, training networks to aim at finding this time should not be dangerous in terms of false alarms, if done with care. To minimize false alarms the investigator can often not employ all the fault data. However, when choosing samples with which to teach networks about the nature of fault data. For some of the fault samples after the analyst-determined time of the fault look like other data that is considered to be nominal. For this (human) decision regarding when test case fault data begins to look unlike any nominal data that can ever be expected to occur, a richer set of unquestionably nominal data than was provided by the two nominal cases is important.

The networks of this type that were created found fault declare times very close to the time selected by the analysts who prepared the fault declare times for the training data sets. In order to avoid false alarms, these times were generally chosen to be a little bit later than the earliest ones given in the failure investigation reports. Most importantly, the networks detected the faults in the gradually developing cases in time to shut down the engine, and usually at a point in time very close to the times indicated in the failure investigation reports as the point at which the failure began to show in the PID values. In the rapidly developing cases, they detected almost every fault sample that was considered by analysts to be unquestionably faulty. Unfortunately, this does not enable them to detect the fault in time to save the engine. Even at the time of the first fault sample it detected in these cases, it is too late to shut the engine down without experiencing major damage.

These networks, thus, worked well on the data set that was provided to them. The question arises whether they could be expected to do well on the test-stand on second-generation engines. Here a far broader range of nominal data in the training data set is required, as well as the possible addition of more sensors, and of a few more features. But the question remains whether or not those additions could be expected to be sufficient. Does not the fact that only a few kinds of faults have occurred, and all those in first-generation engines, totally preclude the value of this approach?

An argument can be given that this is not the case, and that such networks could, in fact, be very effective without significant modification. The reasoning upon which such a conclusion can be based also says something about the value of most other algorithms that have used the information encoded in first-generation fault data, or established their accuracy through tests on first-generation fault data.

The main assumptions that would guarantee the validity of the neural network approach described in this report are:

- 1) The subset of sensors chosen can always be expected to be affected by any major failure, i.e. by any failure requiring engine shut-down.
- 2) These sensors measure major parameters of the engine (pressure in the main combustion chamber, etc.), while the changes in the engines over the years have not affected the basic engine design, and as a result have not significantly effected the expected values of the parameters measured by these sensors.
- 3) The kinds of ways these parameters are likely to be affected as a major engine failure develops, i.e. as the engine nears the point where major damage is about to occur, will be similar over a large range of components whose failure may cause the major breakdown, and examples of the ends of most breakdown scenarios have already occurred. Thus, although a detection method that learns from the faults that have occurred may not detect a fault in time to prevent all damage, it may be able to prevent major damage.

Results of the current investigations with neural networks presented some evidence for the third assumption, the hypothesis that failures in different components generally affect the PIDs in similar ways. For the way the neural networks were checked was by training on data from all the cases except one (case 173 was also always left out of training data sets). The case the network had never seen was then tested. When the case withheld was a fault case and the fault was detected, the network would also indicate how it had detected the fault. Its output units would indicate which other cases's fault data the fault data in the withheld was similar. The data corresponding to a fault occurring in one part of an engine was often determined to be similar to data corresponding to a fault that had occurred in a completely different part of the SSME engine.

Although each of the hypotheses listed above can be expected to have some degree of validity, *the quantification of the extent to which they are true* must be established by SSME engine analysts.

The excellent results obtained, however, indicate that the choice of doing an in-depth investigation of this very well-understood and reliable network type first was a very good choice. In summary, a neural network is an effective component of an SSME health-monitoring system involving other components, since neural nets often find patterns that analysts overlooked, and can be designed to inform other components of a fault-detection system about specific parts of the shuttle engine whose performance is anomalous.

2.2 Basic Network Structure

Feedforward neural networks consisting of a layer of 24-36 input units, one layer of 8-12 hidden units, and a layer of 1-10 output units were used. More layers of hidden units could have easily been added, but the networks that were constructed learned all training data essentially perfectly with only one hidden layer. In these networks, each of the input units is connected to each of the units in the hidden layer, and each of the units in the hidden layer is connected to each of the output units. Other parameters of the networks that have been used recently are: output and hidden alpha of .05, output and hidden beta of .04, range of initial weights of -.2 to .2.

2.3 Data Source and Description

The data available for SSME fault-detection investigations was a very large quantity of sensor data from eleven actual SSME test-stand engine firings conducted between 1981 and 1989 (see Figure 2.1; in referring to cases in this report, the first three digits of the case number as given in this figure will be omitted). For each case, the data included the values of a large number of sensors, twenty-five values per second, recorded from the time the shuttle engine was first fired and lasting, in the nine fault cases, until the time the engine was turned off. In fact, such a command was issued in each fault case on the basis of one or more of the engine sensors having exceeded a set limit, but it was never issued in time to prevent severe damage to, the multi-million dollar engine involved.

If the number of cases is relatively small, but just these few cases generated many millions of bytes of data that had to be run through a neural net from twenty-five to fifty times for the network to be able to "learn" the data thoroughly. There was a total of about 45,000 data samples in the data selected for processing out of the test cases provided, each sample being a set of sensor readings at a particular point in time during a firing. Thus this data came very close to exceeding the space/time limitations of the computer hardware available for the project. It turned out that the neural nets employed could in fact generalize correctly to new data even given such a restricted training set, so that the data set was adequate for these investigations. The extensions of the methods to actual test-stand or space flight use should, of course, incorporate as much of the available data as possible.

Figure 2.1 SSME (Space Shuttle Main Engine) TEST CASES

Nine Fault Cases

<i>Case 901-173</i>	March 31, 1978
LOX Post Fractures, Erosion MCC	
<i>Case 901-225</i>	December 27, 1978
Main Oxidizer Valve Malfunction	
<i>Case 902-249</i>	September 21, 1981
Power Transfer Failure, Turbine Blades	
<i>Case 750-259</i>	March 27, 1985
MCC Outlet Manifold Neck, Fuel Leak	
<i>Case 901-307</i>	January 28, 1980
LOX Post Fractures, Erosion FPS	
<i>Case 901-331</i>	July 15, 1981
LOX Post Fractures, Erosion MCC	
<i>Case 901-340</i>	October 15, 1981
Turn Around Duct Cracked/Torn	
<i>Case 901-364</i>	April 7, 1982
Hot Gas Intrusion to Rotor Cooling	
<i>Case 901-436</i>	February 14, 1984
Coolant Liner Buckle	

Two Nominal Cases

<i>Case 902-457</i>	November 1988
<i>Case 902-463</i>	February 1989

Only time periods during which the SSME was operating at full power for a given power-level were considered, since steady-state fault diagnosis is a difficult and important problem and transient analysis is much more difficult. The full power case is also the most critical problem to solve because most major failures occurred during steady-state situations. Transient anomaly detection is complex since the sensor data can be expected to change quickly, and in complicated ways, when the power level is changing. Moreover, the patterns of change may depend on the exact nature of the transient (start and finish power levels, rate of throttling, etc.).

Steady-state data, on the other hand, has certain properties that can be exploited for health monitoring. For instance, sensor values often remain reasonably steady so that values that change considerably indicate something abnormal. The same criteria for engine health should apply regardless of the amount of time elapsed in the steady-state period.

Netrologic recommends an investigation of the use of neural nets for failure detection during the transient phase. Since neural nets can recognize distinctive time series such as valve signature transients discussed below, they can be expected to be useful for rocket engine transient analysis. Recurrent neural networks, which have been successfully applied to sonar signal recognition problems, are a promising approach to this problem.

Most of the data used in this study came from recordings of cases in which faulty engine performance occurred. The nine fault cases that were used represent failures of various types, caused by malfunctions in different hardware components. Although this provides a variety of data for training and testing, it also means that there is insufficient fault data for a neural net to generalize in any detail about particular failure types.

Use has been made in the investigations described here of only a restricted subset of the available sensor measurements, referred to as Parameter Identification Numbers (PIDs) in NASA terminology and in this report. The selection of the twelve PIDs that have been used (see Figure 2.2) was based on three factors:

1) Availability for almost all cases under investigation. Not all sensors were installed and functioning in every test firing. Since a fundamental objective is to combine data from a number of test cases, and generalize to other cases, data must have very much the same format for all cases. Therefore, a PID was chosen only if it were available for nearly all of the cases used in these studies. However, this is not an absolute restriction: if a particular PID is missing from a particular test case, it is possible to use zero values (values that essentially tell the network that the PID is not changing) for that PID in that case. In fact, judging from the data received, it is essential for a method to be able to accommodate missing, faulty, or "dead" sensors.

2) Significance for diagnosis. Analysis of fault case profiles shows that, for each case, some sensors show strong early symptoms of faulty operation, while other sensors appear to have less value for the diagnosis. Naturally, the PIDs which were chosen were significant in the cases under investigation, and most of the PIDs chosen were significant in more than one case. A few examples of PID values are presented in the graphs in Figure 2.4 and in Appendix A, Section A.1.

3) Current and expected future use. Only SSME PIDs which are currently in use, so that there is data on their ordinary functioning, and which are expected to be in use in the future would be valuable for the construction of a fault-detection system for actual future use on the test-stand or in space.

It has been assumed that most of the data from the fault cases is actually indicative of healthy functioning of the engine, for the vast majority of test data comes from before the point in time when shuttle analysts have judged, after analysis of all the sensor data, that the failure began to develop in the engine.

In fact, there was a very limited amount of fault data in five cases, because the interval between this "fault-declare time" and the time of the last sensor measurements was very short (as short as 0.12 and 0.16 seconds, or 3 and 4 fault samples).

Figure 2.2. PIDs (Parameter ID's) for SSME (Space Shuttle Main Engine)

1. 18 (566) MCC CLNT DS T
Main Combustion Chamber Coolant Discharge Temperature B
2. 24 (371) MCC FU INJ PR (MCC HG IN PR)
Main Combustion Chamber Hot Gas Injector Pressure A
3. 40 OPOV ACT POS
Oxidizer-Preburner Oxidizer Valve Actuator Position A
4. 42 FPOV ACT POS
Fuel Preburner Oxidizer Valve Actuator Position A
5. 52 (459) HPFP DS PR
High Pressure Fuel Pump Discharge Pressure A
6. 63 MCC PC
Main Combustion Chamber Pressure Average
7. 209 (302) LPOP DS PR A
Low Pressure Oxidizer Pump Discharge Pressure A
8. 231 (663) HPFT DS T1 A
High Pressure Fuel Turbine Discharge Temperature A
9. 232 (664) HPFT DS T1 B
High Pressure Fuel Turbine Discharge Temperature B
10. 233 HPOT DS T1
High Pressure Oxidizer Turbine Discharge Temperature A
11. 234 HPOT DS T2
High Pressure Oxidizer Turbine Discharge Temperature B
12. 261 (764) HPFP SPEED
High Pressure Fuel Turbopump Shaft Speed

These are all CADS sensor measurements (data from sensors mounted on the engine, sampled 25 times per second). Numbers in parentheses are corresponding facility measurements, when applicable.

2.4 Pre-Processing of Data

The inputs to the networks are derived from PID values. Each sample fed into the network corresponds to a particular point in time. However, the input values are not simply the raw values for each PID at that time. The nature of the variation in PID values over time may be more indicative of faulty performance than the value of the PIDs at any isolated moment. Therefore, for each point in time, three features were originally calculated for each PID, which took into account the medium, long, and short-term history of that PID leading up to that time (for the formulas used to calculate these features, see Figure 2.3). These features resemble calculations used in Rocketdyne's SAFD algorithm. To save time and disk space, in the most recent runs the feature reflecting the short-term history (the last .16 seconds) has been dropped, after it was found that the second feature could usually be used to detect a fault just as quickly.

The use of averages in the features is for the purpose of smoothing out "noise" in the sensor data. Thus the second feature is essentially the smoothed PID, with the smoothed value at the beginning of the steady-state subtracted off to factor out differences across individual engines and across different power levels for the same engine. The division by the standard deviation in the raw PID value is done to put all features on the same scale, in similar ranges. Both neural nets and humans often find it easier to work with data having this property.

The first feature is an approximation to the local rate of change of the PID, and is also more indicative of an abrupt change in the value of the PID than the second feature is. Moreover, each of the three resulting features is a very crude measure of how much the sensor value has changed over the time period being considered, compared to how much it might be expected to change. This would be more precisely the case if the differences were divided by the standard deviations in averaged PID values. But the features currently in use employ averages taken over different periods of time. In any case, at least for diagnosis using neural nets, division by constants are irrelevant unless information is lost.

Before features values are fed into the neural nets, their range is restricted to the interval $[-7.5, 7.5]$ by truncation. This truncation limit truncates only fault values and allows a small gap, a little more than 1, between the extreme range of nominal data, which is around 5 and the upper bound, 7.5. Fault values for the features approach 1000 in our data, so that it was felt that not truncating them might make it difficult for the network to detect subtle relationships in the nominal data. Figures 2.4, 2.5, 2.6 and 2.7 contain graphs of the raw PID values of PID 42 from the second steady-state time slice of case 436, and of the corresponding three features calculated from them. Further examples of graphs of raw values and the features corresponding to them occur in Appendix A, Sections 1 and 2.

Figure 2.3 Features Computed for each PID for each Sample

- MEDIUM TERM SMOOTHED CHANGE

$$\frac{Avg50(t) - Avg3(t)}{\sigma}$$

- LONG TERM SMOOTHED CHANGE

$$\frac{Avg50(t) - Avg50(t_s)}{\sigma}$$

- VERY SHORT TERM CHANGE

$$\frac{X(t) - Avg3(t - .08)}{\sigma}$$

where

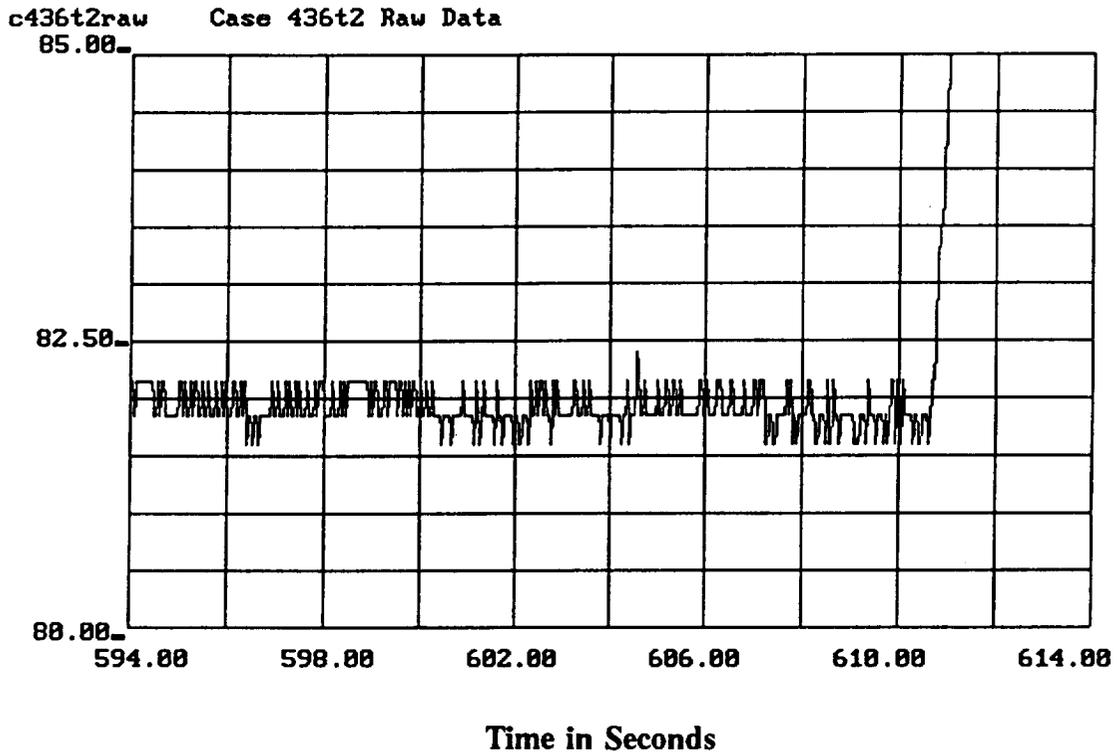
$X(t)$ = PID value at time t

$Avg3(t)$ = mean PID value for last 0.12 seconds (3 samples)

$Avg50(t)$ = mean PID value for last 2 seconds (50 samples)

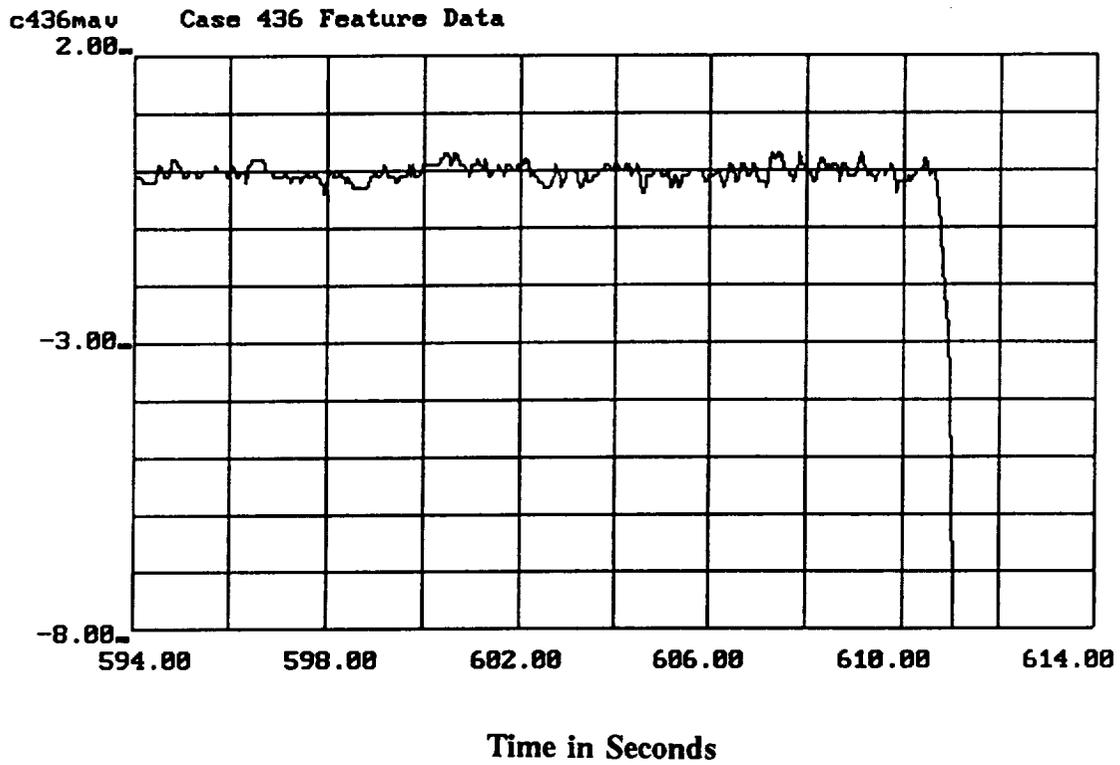
σ = standard deviation of nominal PID value
(derived from steady-state data from all available firings)

t_s = time 50 samples after start of current steady-state interval



Graph of Raw Data Values, Case 436t2, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A

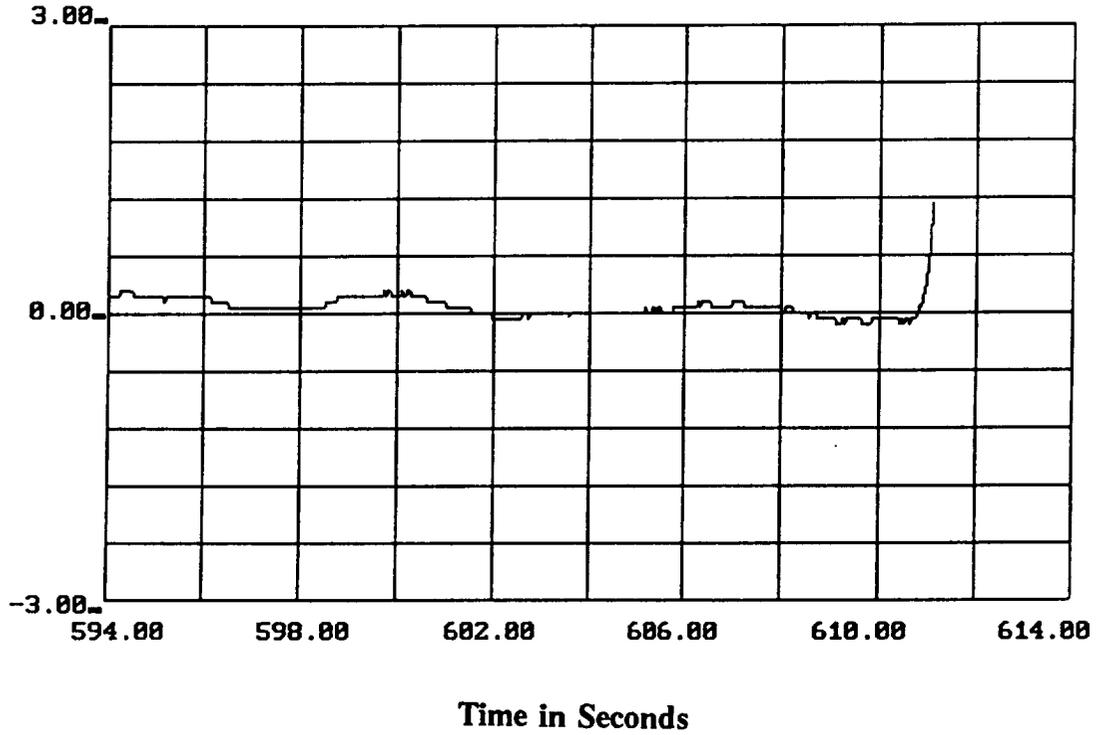
Figure 2.4



-14-

Graph of Feature 1 Values, Case 436t2, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A
Figure 2.5

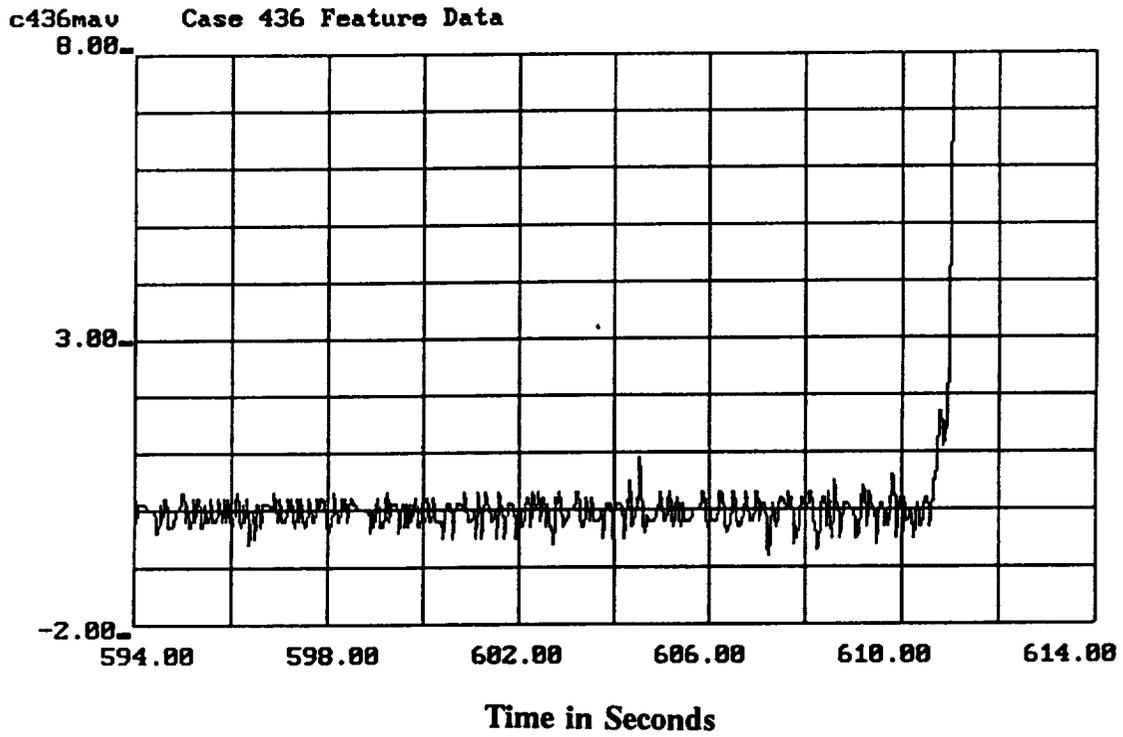
c436mav Case 436 Feature Data



-15-

Graph of Feature 2 Values, Case 436t2, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A

Figure 2.6



-16-

Graph of Feature 3 Values, Case 436t2, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A

Figure 2.7

As two or three features were used in the networks constructed, the total number of simultaneous inputs to the network for each point in time was either two or three times the number of PIDs. Since twelve PIDs are now being used, this means that the networks currently employed have 24 or 36 inputs.

In future studies, more features will be computed for each sample, to provide more detailed information on the time-variation of PIDs, or to explicitly input features which code relationships between PIDs or other features. In theory, the network is capable of performing any computation on the inputs, so all features calculable from the PID values in a single data sample would be superfluous. In practice, however, inputting certain compound features explicitly often causes the network to learn in a way that will lead to better generalization. And unless recurrent neural nets are used, relationships between PIDS measured at different times must be calculated and used to train a network.

The two features currently used, are sufficient for the detection of major engine failures. However, additional features may allow slightly earlier detection of the faults that have occurred, and perhaps allow the detection of additional types of faults.

2.5 Assignment of roles to output units

The output of the network represents its evaluation of the input data. The values of the output units are all floating-point numbers between zero and one. Ten output units are currently used, each of which represents a different diagnosis category.

The first output unit is designed to give the network's determination as to whether the sample is anomalous (i.e. "fault": engine shutdown should be considered) or nominal (engine performance is nominal). The remainder of the output units are used to give the network's determination as to whether the sample it is evaluating seems to be like a fault occurring in any of the fault cases in the training data. Thus there is an output unit for each of the fault cases.

To train the neural net, the target value for each of these output units is set at 0.9 for "yes" and at 0.1 for "no". Thus a fault sample for the third fault case would have as target values for the ten output units:

0.9 0.1 0.1 0.9 0.1 0.1 0.1 0.1 0.1 0.1
i.e.,
yes no no yes no no no no no

So here the first 0.9 (the first "yes") just says "this is a fault sample", and the second 0.9 (the second "yes") says "this is a fault sample from case 249".

As has been already mentioned, the fault-declare time initially taken for each of the fault cases was based on NASA failure investigation reports which give times when NASA analysts believed that sensors had started to indicate signs of problems or faulty performance. Sensor samples taken before the fault-declare time were considered nominal, and samples taken after that time were considered fault data.

Note that a nominal sample of the third fault case is not given the output unit:

0.1 0.1 0.1 0.9 0.1 0.1 0.1 0.1 0.1 0.1 0.1,

but rather one in which the third "case" value (fourth target value) is set to 0.1 instead of 0.9. This is most nominal data is likely to be quite similar across cases. For the network to distinguish between the different cases of nominal data would be a very difficult task (it might take a very long time for training to converge), and forcing a network to use particulars of the training data could make it more difficult for it to generalize in the way it must to detect fault data it has never seen. Moreover, training a network to distinguish between different nominal cases would usually have little value in terms of fault detection.

After a network is trained, it is then tested both on data contained in its training set, and on new data. Values for output units close to 0.9 or 0.1 indicate whether the sample is determined by the network to be fault or nominal, or similar to a fault case included in the net's training data.

Thus a sample with the output values

0.85 0.15 0.09 0.05 0.06 0.08 0.05 0.10 0.10 0.91

would indicate a sample that was considered anomalous, and similar only to fault data in the last fault case (case 436). In determining a trained network's evaluation of the nature of a sample presented to it, the indication of "faultiness" currently used is a value greater than .5 in the first output unit or a value greater than .5 on one of the "fault data in a fault case" indicators.

2.6 Fault-declare Times and Net-Structuring Issues

Reasons for pre-processing of data can only be discussed after the structure and working of the networks is understood. It should be fairly clear that the decision as to which samples should be included in the training set as fault samples and which should be included as nominal will be critical to the nature of the trained network that will result.

But even after careful analysis of the data the appropriate selection of a "fault-declare time" is not always a simple thing for an analyst to decide. Especially when it is a question of what fault-declare time to use for neural net training and evaluation.

For instance, consider a sensor whose values one can see will soon be moving out of nominal ranges. What point between where it first begins a continuous value change and the extreme value it takes before the engine is destroyed or shut down should one label as "fault-declare time"? And how many and which PIDs need to be faulty?

A fault appears to show itself at the beginning of the rise or fall of the first PID whose values are changed by the fault, so that the point when any PID begins a change indicative of the fault would be the appropriate "fault-declare time". And this choice is reinforced by the fact that one would really like to place the fault-declare time as early as possible, for an engine must be shut down as quickly as possible after it begins to fail. It is this kind of fault-declare time that NASA analysts discuss in the failure investigation reports on fault cases.

But observations of nominal data often show that sensors go up or down unpredictably for varying periods of time during nominal runs. Thus, a more suitable fault-declare time to be given to a network for training might be the point at which some PID's values first go outside the range of all values ever taken on during a completely nominal run. In fact, both these criteria are now in use simultaneously for indicating fault-declare times to the neural nets during training, the failure investigation report indication of the time the fault first started being indicated by sensors, and the time at which sensor values begin to look different from any clearly nominal data.

What led to this approach was the following. A first set of runs used only the "early fault-declare time", the time the fault begins to show in sensors. But this led to many "false alarms", nominal samples considered to be anomalous, being produced when trained nets were tested on data from cases they had not seen. This was due to early fault data in fault cases in the training data set looking very much like nominal data in the new cases.

In the most recent runs fault samples are put into the training data as faulty only if they are strongly faulty, i.e. do not look like any clearly nominal data.

Some data that is considered to be nominal, but which is "border-line", was also taken out of the training data set. The data that is excluded from all training sets has been called "intermediate data" in this report.

The structure and number of samples in the data files that were used in the last complete set of runs is given in Figure 2.8. The intermediate data excluded from all training sets is indicated in this table and the number of samples in that category is enclosed in parentheses in the nominal and fault columns. The analyst-determined fault-declare times inherent in the data at the beginning of the runs is given in Figure 2.8 in the line under the times for the samples under consideration. All times are in seconds from start-up of the engine. When there are numbers to the left and right of a slash on that line, the number to the left is the early fault-declare time, the time at which the fault begins to show up unquestionably in the sensor data, while the number to the right is the time at which the feature values clearly begin to look different from all nominal data. When a number in parentheses occurs before these other two numbers, it indicates a time in the middle of what has been determined by analysts to be nominal data. All nominal data past that time has also been called intermediate data, on the grounds that in these cases there is a continuous slow movement from nominal-looking data to obviously faulty data, so that a line of demarcation is actually impossible to decide upon. Thus it is left to the network to decide where to place the fault-declare time, and any time within the intermediate range will be considered correct.

Note that all data in case 173 was put into the intermediate category, because five of the twelve PIDs are missing or in invalid ranges, so that its data was considered too inconsistent in nature with the rest of the data to be included in training data sets, or to judge network performance by. The curly brackets on its case number in Figure 2.8 are meant to indicate this. In fact none of the networks were able to detect that case 173 was a fault case, all apparently having been trained to detect faults only if more PIDs gave strong fault indications than was possible, given the PID situation in case 173. It is also not clear whether case 173 should be considered a gradually or rapidly occurring fault. There are signs of faultiness early on, but a very large change in many PID values occurs just before the engine is shut down.

A set of runs before the most recent ones just mentioned was also made in which *every* fault sample was included in the training data sets for those fault cases with a very limited number of fault samples, this being done in hopes of catching the faults in these cases as early as possible. Thus the two fault-declare times as described in the last paragraph were used in this set of runs only for the gradually developing fault cases. But there were still too many false alarms.

Figure 2.8 Basic Data Files, July 1990

		times	samples	nominal	fault	Intermediate
{1.}	Case 173	112.00 - 201.08 165.68	2228	(1342)	(886)	2228 1342+886
2a.	Case 225t1	10.00 - 32.96	575	575	0	
2b.	Case 225t2	37.00 - 60.96	600	600	0	
2c.	Case 225t3	65.00 - 255.60 255.52	4766	4763	3	
3.	Case 249	263.96 - 450.56 320.32/330.00	4666	1409	3257 (242)+3015	242
4.	Case 259	43.96 - 101.50 101.36	1439	1435	4	
5.	Case 307	17.00 - 75.00 18.92/24.00	1451	48	1403 (127)+1276	127
6.	Case 331	154.00 - 233.12 232.32/232.40	1979	1958	21 (2)+19	2
7.	Case 340	24.00 - 405.48 278.92/290.12	9538	6373	3165 (280)+2885	280
8a.	Case 364t1					
8b.	Case 364t2	14.00 - 39.96	650	650	0	
8c.	Case 364t3	48.00 - 65.96	450	450	0	
8d.	Case 364t4	74.00 - 392.12 (121)204.32/217.00	7954	3258 1175+(2083)	4696 (317)+4379	2400 2083+317
9a.	Case 436t1	153.96 - 220.00	1652	1652	0	
9b.	Case 436t2	553.96 - 611.08 (610.56)610.72/610.76	1429	1419 1415+(4)	10 (1)+9	5 4+1
10a.	Case 457t1	104.00 - 166.96	1575	1575	0	
10b.	Case 457t2	176.00 - 219.96	1100	1100	0	
11.	Case 463	103.96 - 238.12	3355	3355	0	

Moreover, the desired benefits of leaving the "weakly faulty" data in for the rapidly developing faults did not materialize: networks trained on such data sets were worse at detecting the faults even in the "rapidly developing" cases (and in the other cases as well) with this approach than with the one now used that takes all "weak fault" data out of the training set.

But in the two cases in which there were very many false alarms during these first two sets of runs, cases 307 and 364, it was first concluded that it was very likely that the networks had been more correct than the determination originally made in the failure analysis reports. This judgement is still considered likely about case 307, which has high feature values almost immediately, but the rising PIDs starting around 204 seconds in case 364 involved factors other than the fault. Such factors which make the fault-declare time very difficult to determine. The early fault-declare times had presumably often been determined by analysts by looking at graphs of the raw PID values for the cases individually. Thus, the networks had shown that across-case comparisons, the kind networks make, are important in determining both the early and late fault-declare times for cases.

The development described in the previous two paragraphs demonstrate that working with neural nets is a dynamic interactive process. It demands quite a bit of thought concerning how to construct the net (in particular what kind of feature data it should be asked to operate on and what kind of output units to employ), as well as eventually in the final analysis of the often very informative network outputs.

Various different SSME neural net approaches, including the unsupervised learning networks mentioned above, may make the "fine-tuning" involved in the fault-declare time considerations just discussed irrelevant if they are as successful as the current approach. They could eliminate the problem that exists deriving from the fact that it is not actually known how much, if any, of the data in the fault cases should really be called nominal.

As mentioned earlier it could turn out that with only slight modifications the feed-forward back-propagation neural network approach that has been extensively investigated may turn out to be one of the most fruitful neural network approaches to this problem.

Two additional variations on that technique are important. One approach to training a network would be to use the second-generation nominal data trained against synthetic deviant data. Note that this approach in some sense eliminates difficulties involved in the decision about fault-declare time for cases in the training data. The problem reemerges again, however, when one tries to evaluate this method by testing it on first-generation fault cases. If the network says the fault occurs at a different time than the failure investigation reports, again: how is one to decide when the network is correct other than by having engineer/analysts rethink the issues involved?

The second approach would be to train networks with both genuine and synthetic fault data, in order to "fill out" the space of fault data presented to the network.

Various methods of generating synthetic deviant data could be used, including generating random values for PIDs. Since random data would have none of the correlations that exist between various PIDs, and because several PIDs will be generally be outside nominal ranges, almost all randomly generated data would be quite different from nominal data. Early experimentation with randomly generated synthetic data resulted in large numbers of false alarms, in spite of the fact that almost all data generated had several PID values outside the range of most nominal data. Additional features (especially ones specifically encoding correlations between PIDs and more properties of time sequences) may help with characterizing nominal data to a network of this type.

It will be effective to restrict the synthetic data set in various ways. However, the danger with introducing a restricted synthetic data set is that the end effect can be to have the neural net learn the theory embodied in the choice of the synthetic data. For example, if "deviant" data were restricted to data whose Euclidean distance from the origin was larger than five, say, the network might diagnose as fault precisely those samples whose distance from the origin was greater than five. Thus, the network could ignore all the information embodied in the real test-case data in its training set, with the exception of the fact that most of it had a Euclidean distance from the origin of less than five.

Generating restricted deviant data is promising. In particular, synthetic deviant data might be generated by creating samples with n features, $n = 3, 4, 5$, outside the range of any nominal data in the test cases in the training data set for a holdout case, the remaining features could be given random values between -1 and 1. If this results in false alarms in the holdout case as in the past, an attempt could be made to "fill in" the nominal range by generating midpoints between nominal samples in different test cases. Note that synthetic data could be labeled as to which PIDs are out of nominal ranges in the sample. Then a sample found similar to data with PID N , and PID N alone, out of range, would probably be assumed to be indicative of a sensor failure associated with PID N . While in cases with several PIDs out of range, the nature of the fault could be identified much more precisely by knowing which PIDs were leading the network to assume that a fault had occurred.

A related approach requires the existence of a fault-simulation program. Developing such an approach would be a major undertaking even though there is currently a simulator that simulates healthy engine performance. A network could be trained with a large amount of genuine nominal data from recent engines along with synthetic fault data generated by the fault simulator. In any case, such simulated fault data would be valuable for the testing of any fault-detection system involving situations where there is insufficient actual fault data.

It should be pointed out in this training data/net structuring context that *the problem of failed sensors* that has been identified as a serious one, especially when there is little sensor redundancy (more than one sensor measuring the same thing) as on the SSME. It is very important that a fault detection system, especially one designed to make a decision as whether or not to shut down a main engine, be able to tell the difference between a failed sensor and a failed engine whenever possible. The approach we employed is to just leave the apparently erroneous PID data in the training data set (missing PIDs were given the value of zero), so that the networks can learn to ignore such failed sensors, combined with a technique of switching to a different network that overlooks a particular sensor if it is found that the original network no longer declares a fault if a single sensor is omitted (see below for an example based on case 457). Networks performed well for the occasional missing or apparently failed sensor.

Fortunately, in all the fault cases so far examined, eventually several PIDs simultaneously indicate the fault, so that missing PIDs, if they are not too numerous, have not posed a serious problem. The problem of having a failed sensor cause a false alarms was solved by feeding into the original network the same feature values except for having zeros replace the true values of the PID in question. This approach to the problem of failed sensors will of course detect faults only after they are being indicated by two or more PIDs simultaneously.

2.7 Fault-Typing

It would be very useful for a network to be able to distinguish between different failure types. For instance, this will be true if different shutdown or safety procedures are employed depending on the type of the failure involved, or if the neural network forms a part of a larger fault detection system that could benefit from any information the network can provide. Indeed, fault detection should optimally involve the notification of a failure, the isolation of the type of failure, and the estimation of its severity. Work so far has emphasized detection of a possible failure warranting a shutdown sequence, and isolation both through attempting to detect similarity to past failures and through detecting which PIDs are being used by the network to indicate the fault. The severity of the possible fault was not addressed, but some estimate of severity could be derived by looking at the values of the PIDs indicating the fault (both their values and their rate of change).

Further study of fault isolation and severity estimation should be pursued. As already mentioned, one approach would be to include output units to indicate which PID features are considered abnormal. It is expected that further developments in these areas will be made on the basis of in-depth understanding of the functioning of the shuttle engine, including a detailed analysis of how each specific kind of failure can be expected to be reflected in the sensor values. This kind of knowledge would naturally lead to a failure simulator, which would provide a categorization of errors and an evaluation of their severity.

2.8 Training Method and Results

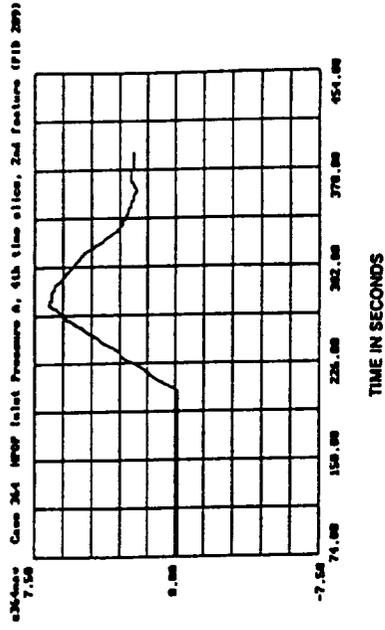
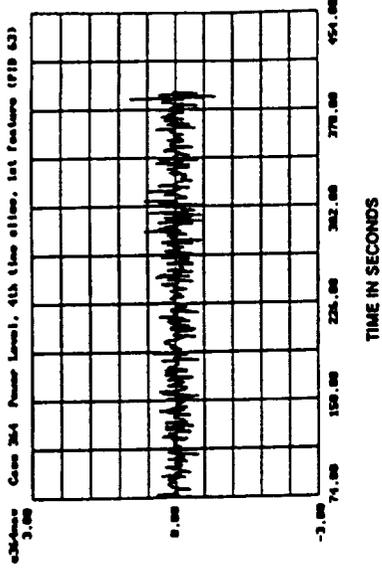
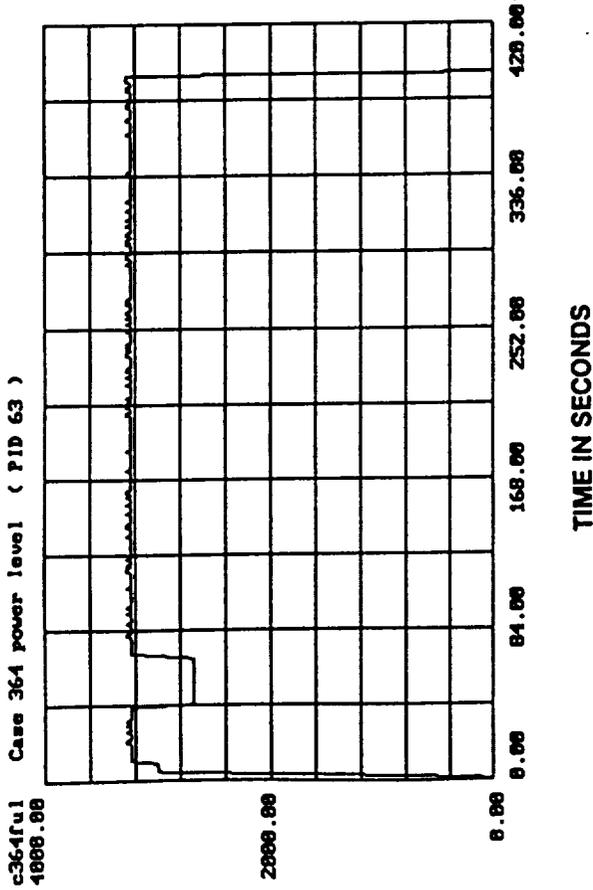
The method used in the most recent sets of runs is to train a network on data from all the SSME test cases except one, with all the "intermediate" data discussed above left out of the training data set. It has been mentioned that all of case 173 data was placed in the intermediate data set. Thus, a training data set contains data from nine test cases, except when case 173 is the case being tested, in which case the training data set has ten test cases.

For the five cases that contained very low proportions of fault data, the fault samples were duplicated in each training data set a hundred times in the first set of runs, and two hundred times for the later sets of runs. This was done in order to train the network more quickly and more accurately. Without this duplication, the kind of data occurring in the rapidly developing faults would be seen so infrequently by the neural net in training that it would have a very hard time learning it. Moreover, after the first set of runs the last 728 samples of the fault data in each of the four cases with large amounts of fault data were duplicated once, in order to bring the amount of fault and nominal data into a better balance. The training data file was shuffled thoroughly before the network was trained.

Networks were trained using 25-50 passes through the training data set. After training, the trained networks were tested on the training cases. The first networks, which worked only with the initial fault declare time, i.e. had no intermediate data withheld from the training data set, so that they contained similar nominal and fault data required more passes to learn the data set, about 2,500,000 pattern presentations, or on the order of 50 passes through the data set.

When the training data sets excluded all intermediate data, networks that trained on 1,500,000 pattern presentations, or about 30 passes through the training data set, still learned the data with very high, often perfect, accuracy i.e. after training for the amount of time specified, they would output "nominal" when fed nominal data, and "fault" when fed fault data from the training data set. When learning was not quite perfect, the incorrect outputs always occurred for data immediately before or after the fault-declare time. This not only showed the difficulty of handling similar data in the two categories but also the related fact that the transition period around the fault-declare time was the most difficult to learn. In fact, the training in the last sets of nets, where intermediate data was taken from cases with both large and small amounts of fault data and two of the early fault-declare times and intermediate data sets were adjusted to reflect what the first sets of networks had pointed out, was too perfect. Reducing the number of pattern presentations even further, or reducing the number of hidden units, would possibly produce better results than the ones now obtained.

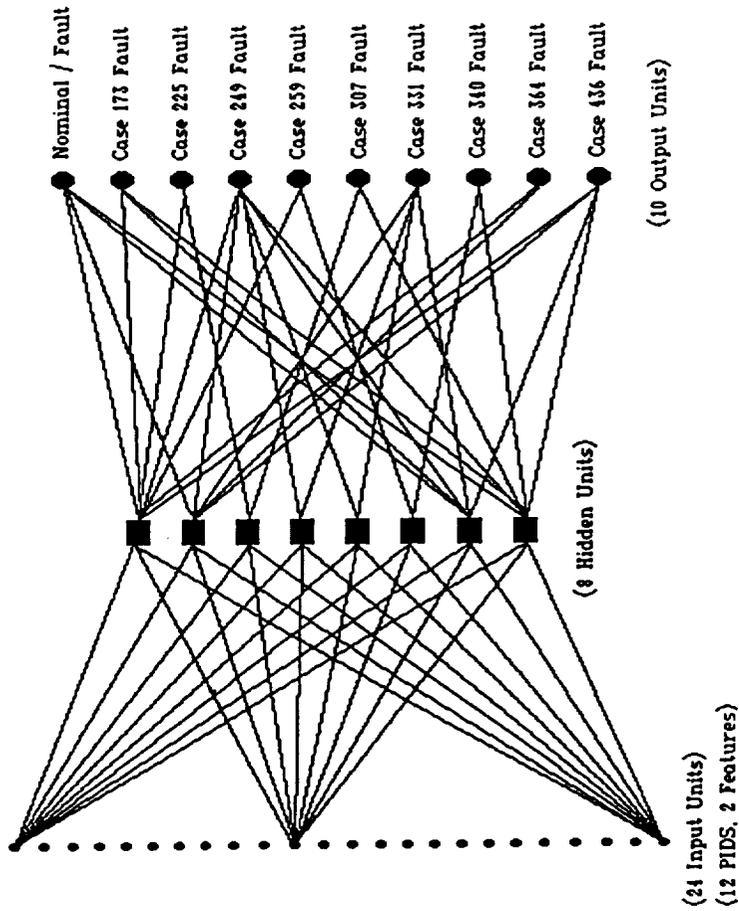
Figures 9 and 10 summarize the procedures and structure of the neural networks employed.



1. EXTRACT STEADY STATE TIME SLICES BY EXAMINING PID 63 (CASE 364 HAS 4 STEADY STATE TIME SLICES). (24 SAMPLES PER SECOND)
2. FOR EACH SET OF SENSOR READINGS IN THE TIME SLICE, CALCULATE BOTH FEATURES

Neural Network Procedure

Figure 9



3.

A. TRAIN ON SHUFFLED DATA WITH CASE FOR TESTING EXCLUDED B. TEST ON CASE WITHHELD FROM TRAINING SET

Neural Network Procedure (Continued)

Figure 10

The following table summarizes typical training results for a set of hold-out runs using intermediate data (the data is taken from runs made with the data structured as described in Figure 2.8).

Holdout Case	Trained Samples	Correct Samples
173	50,000	49,998 = 99.996%
225	43,462	43,460 = 99.995%
249	44,848	44,846 = 99.995%
259	47,765	47,763 = 99.995%
307	47,948	47,946 = 99.995%
331	44,242	44,240 = 99.995%
340	40,014	40,012 = 99.995%
364	42,618	42,618 = 100.000%
436	45,133	45,129 = 99.991%
457	47,325	47,325 = 100.000%
463	46,645	46,643 = 99.995%

The real test for neural networks comes when a trained network is asked to make determinations on data it had never seen, which in the investigations being described means on cases it has never seen. Very early runs with networks trained on data from three or four cases indicated that networks might fail to find any faults without synthetic data being added to expand the fault data set. But none of the recent runs using all cases except one or two in the training data (and no synthetic data) failed to detect fault data, showing that adding just a few more cases gives sufficient "richness" to both the nominal and fault data sets to allow the networks to discriminate. The complete set of runs that was made with the refinements to the structure of the training data sets that have been discussed above (the resulting structure of the data files as given in Figure 2.8) were very successful. The results of these networks is given in the following table:

Case	Correct Nominal	Incorrect Nominal	Correct Fault	Incorrect Fault
173	1342	0	0	886
225	5938	0	2	1
249	1409	0	3050	207
259	1435	0	4	0
307	48	0	1318	85
331	1958	0	19	2
340	6373	0	2928	237
364	2275	0	5852	927
436	3071	0	7	3
457	1791	884	0	0
463	3355	0	0	0

Table 2.1 Neural Net Determination of Samples as Nominal or Fault
Number of Samples Evaluated Correctly

The correct nominal column gives the number of samples in the case in question that were labeled nominal by an analyst and which had firings of less than .5 on all the neural net output units (i.e. was determined to be nominal by the neural network according to this criterion). The incorrect nominal column gives the number of samples from the case that were labeled nominal by the human analyst, but for which the neural net fired greater than or equal to .5 on at least one of its output units. The second two columns contain similar numbers for samples labeled anomalous (i.e. "fault") by the human analyst.

Note that each line corresponds to a different neural net; a neural net trained on a different set of training samples than used for the other nets. For instance the first line is for the run during which all of case 173 data was withheld from the training data set so that the training data set contained data from the remaining ten cases, with the exclusion of intermediate data. And the second line corresponds to the network for which case 225 data was withheld from the training data set so that the training data set contained data from cases 249 through 463 (case 173 data is excluded from every training data set).

The information given in Table 2.1 is perhaps more relevant when given in terms of percentages:

Case	Correct Nominal	Incorrect Nominal	Correct Fault	Incorrect Fault
173	100%	0%	0%	100%
225	100%	0%	66.7%	33.3%
249	100%	0%	93.6%	6.4%
259	100%	0%	100%	0%
307	100%	0%	93.9%	6.1%
331	100%	0%	90.5%	9.5%
340	100%	0%	92.5%	7.5%
364	100%	0%	86.3%	13.7%
436	100%	0%	70%	30%
457	67%	33%	NA	NA
463	100%	0%	NA	NA

Table 2.2 Neural Net Determination of Samples as Nominal or Fault Percentages of Samples Evaluated Correctly

All these percentages depend on decisions made about fault-declare times. In cases 173, 307 and 364 (fourth time slice), the true time of the fault is very much in question (possibly occurring at the very beginning of the time slice in question). It is best to focus on the fact that whenever the fault developed slowly enough to allow a safe engine shutdown to be possible, the networks detected faults (often very different from those they had been trained on) long before the time required to perform an engine shutdown.

The test sets contained all the samples around the fault in the holdout case, i.e. they included all the intermediate data as well, while most of this data was withheld from training data sets specifically because it was on the borderline between nominal and anomalous data. Since these quantitative network evaluations are the only way to give some kind of measure of the effectiveness of the networks, it would be more appropriate to evaluate the network's performance on the *nonintermediate samples* alone as well. The following two tables provide the same information as in the ones just given for the samples in cases that were not put into the intermediate data category.

The intermediate data has been determined by trained networks to consist of approximately half nominal data and half fault data.

Case	Correct Nominal	Incorrect Nominal	Correct Fault	Incorrect Fault
173	1342	0	0	886
225	5938	0	2	1
249	1409	0	3015	0
259	1435	0	4	0
307	48	0	1272	4
331	1958	0	19	0
340	6373	0	2885	0
364	2275	0	4343	36
436	3067	0	7	2
457	1791	884	0	0
463	3355	0	0	0

Table 2.3 Neural Net Determination of Samples as Nominal or Fault Numbers of Samples Evaluated Correctly, No Intermediate Data

The table giving percentage values is again perhaps more valuable, at least for the gradually developing cases:

Case	Correct Nominal	Incorrect Nominal	Correct Fault	Incorrect Fault
173	100%	0%	0%	100%
225	100%	0%	66.7%	33.3%
249	100%	0%	100%	0%
259	100%	0%	100%	0%
307	100%	0%	99.7%	0.3%
331	100%	0%	100%	0%
340	100%	0%	100%	0%
364	100%	0%	99.2%	0.8%
436	100%	0%	77.8%	22.2%
457	67%	33%	NA	NA
463	100%	0%	NA	NA

Table 2.4 Neural Net Determination of Samples as Nominal or Fault Percentages of Samples Evaluated Correctly, No Intermediate Data

It is noteworthy that for some samples the network would fire under .5 on the first output unit (the nominal/fault output unit), while firing above .5 on one of the cases which meant the network concluded that the sample looked like fault data in that case. The results given in the above tables are those obtained by declaring a sample to be determined to be anomalous by the net if any of the ten output units fires above .5. This shows an unexpected benefit of having asked the net to distinguish between the fault cases. A network that had been trained without the "case" output units, with only the nominal/fault output unit, would not necessarily give the same results as seen in the single nominal/fault output unit of a network trained with many output units.

The following table presents the "fault-declare" times determined by the last set of runs. Here the "first" net-determined fault-declare time corresponds to the first fault sample detected, the "early" net-determined fault-declare time is the first time the network found ten consecutive samples faulty, and the "late" net-determined fault-declare time is the first point at which all samples from that time on were considered faulty. The time is the number in parentheses in the table following the analyst-determined early fault-declare time are for the end point of intermediate data for the case.

Fault Declare Times

Case	Net, First	Net, Early	Net, Late	Analyst
225	255.56	NA	255.56	255.52
249	328.24	329.04	329.04	320.32(330)
259	101.36	NA	101.36	101.36
307	22.16	22.16	48.00	18.92(24)
331	232.16	232.40	232.40	232.32(232.40)
340	287.48	289.52	289.52	278.92(290.12)
364	138.60	138.88	221.72	204.32(217)
436	610.84	NA	610.84	610.72(610.76)

More relevant to the saving of the engine is the length of time between the time the fault was detected and the time engine shutdown occurred.

Seconds from Fault Declare Time To Shut-Down

Case	Net, First	Net, Early	Net, Late	Analyst
225	.08	NA	.08	.12
249	122.32	121.52	121.52	130.24
259	.16	NA	.16	.16
307	52.84	52.84	27.00	56.08
331	.76	.76	.76	.84
340	118.00	115.96	115.96	126.56
364	253.52	253.24	170.40	187.80
436	.28	NA	.28	.40

It can be seen from these tables that "false alarms", nominal samples being declared anomalous by the net, occur only in case 457 when the net has finished categorizing data. With the exception of this case, the networks have clearly satisfied what was required of them, namely to detect faults in time to shut down the engine when possible, while not indicating a shutdown when it is not required.

Note that in the 5 cases with less than 30 fault samples, it would not be possible to shut the engine down in time to prevent damage, assuming there is in fact no way to detect the fault more quickly from the PID values in these cases, for it takes several seconds to shut a space shuttle main engine down. Nevertheless, the networks did detect all but the first sample or two in each of these cases.

The false alarms in case 457, it turns out, are due to PID values different from nominal values in all other training case due to something called "venting" and "repressurization" being carried out in this case.

Conversations with NASA concerning this provided the additional information that this situation had also occurred in cases 364 and 463 and also in other cases, but long before the time slice selected for inclusion in this data. It makes case 364 particularly confusing, because the early fault-declare time originally used, the time of 201.4, is actually the time of the beginning of repressurization, so that the actual occurrence of the fault could be anywhere in the time slice. It is a similarity to case 364 that is causing case 457 to be perceived by the net as faulty. The nature of the venting and repressurization operations carried out in case 463 had produced less extreme PID values than in case 457, so that having case 463 nominal data in the training set was unable to prevent having case 457 be seen as like case 364. With the test cases now used, it was impossible to make the false alarms in case 457 completely disappear, even by calling all of case 364 data nominal up to second 248.

This problem of the kind of false alarms that occur in case 457 would almost certainly disappear with a larger nominal data set, i.e. with a data set including more nominal cases with data like that seen in case 457. Having such data would not make case 364 appear nominal, because the training set that case 364 was tested against had been trained with case 457 nominal data. Since networks have never encountered any difficulty training themselves to consider all of the case 457 data as nominal, it is clear there is a "gap" between case 457 data and the fault data in the fault cases (when there is overlap between two categories, networks have difficulty converging during training). Networks, like any other fault-detection method that bases itself on what has actually been seen to occur in the past, can always fail when nominal values more extreme than any nominal data it has been given is presented to it. Further ways of dealing with such false alarms should be investigated. For instance, training a network to also evaluate the severity of fault data (as indicated by the norm of the feature vector, for instance), and only declaring a fault when the fault is determined by the network to be of more than minimal severity.

Other very interesting approaches to this problem will be discussed at the end of this section, but a further way of understanding the results obtained by the networks should be seen first.

The graphs of the neural network output units will be given in this section and a larger number of examples may be found in Appendix A, Section 3. The first output unit, "output unit 0" on the graphs (the number of the output unit occurs to the right of the graph), is the nominal/fault output unit. The networks are trained to output .1 for nominal samples and .9 for fault samples. The remaining output units are trained to have an output of .9 mean "like a fault sample in case ...". Since case 173 has been dropped from training data sets, the correspondence is for the "case" output units is:

Output Unit	Case
2	225
3	249
4	259
5	307
6	331
7	340
8	364
9	436

Section 3 of Appendix A ends with the graphs of the two completely nominal cases, case 457 and case 463, so that the reader can get a feeling for what being seen as nominal by a network looks like by looking at them first. Figure 11 contains a graph of the nominal/fault output unit for case 463, and since case 463 appears the most nominal of all cases to the networks, it is suggested that first graphs to be examined in Appendix A be for that case. It can be seen that almost all output units are close to .1, as they do in Figure 11.

Nevertheless, output unit 4 rises at about 132 seconds into the firing; apparently part of case 463 has a certain resemblance to some fault data from case 249. This is one of the slowly developing cases, so that it may still contain some fault data that is not very faulty, even though most of the less faulty data was put into the "intermediate" category. After looking at case 463, it can be seen that the network considers case 457 to be faulty.

For the remainder of cases, the fault cases, graphs of "case" output units are only included in Appendix A, Section 3, if they are at least slightly indicative of the fault involved.

As an example of neural network output for a fault case, Figures 12, 13, 14, 15, and 16 contain graphs of the output units from the network trained on all cases except cases 364 and 173, when tested on the fourth time slice of case 364. Consider first the first output unit, the one that should fire around .1 for nominal data and around .9 for fault data.

The first output unit (the nominal/fault unit) for case 364 rises gradually from .1 at the beginning of the fourth steady-state time slice, 74 seconds into the firing, to settle at .9 at around 244 seconds. It is above .5 most of the time after 170 seconds, however. Looking at the graph of the first or fourth output unit (output units number 0 and 3 on the graphs) might suggest the fault was developing from the beginning of the time slice. In any case, the decision to have all the samples between seconds 121 and 217 in the intermediate data shows the appropriate caution about case 364 and the possibility that the data before second 121 might actually be faulty not too dangerous, for it does not appear to be very strongly faulty, even as evaluated by the network.

For the graphic illustration to accompany the rest of this discussion of neural net outputs, the reader should consult Appendix A.

The graphs of the first output unit become high in case 225, 249, 259, 331, and 436 (all the rapidly developing cases, as well as case 249) at almost exactly the analyst-determined fault-declare time.

Case 173 never reaches .5 in either the first output unit nor in any of the case units. But output unit 8 rose very suddenly and approached .4 just before the engine was shut down and other output units rose very suddenly at the same point.

It is interesting that in case 225 the first output unit is not indicative of the fault. Note that only output unit 6 fired high.

Case 307 was originally assigned a fairly late fault-declare time: the failure analysis investigations determined that symptoms began showing regarding the fault only after 31 seconds into the firing (according to that analysis, various PIDs starting indicating the fault at 31, 38, 44, 47, 49, 55, and 61 seconds). Note that the graphs of output unit 2 (the output unit indicating similarity to fault data in case 225) and output unit 0 increase sharply at both sometime between second 21 and second 23 and again sometime between second 61 and second 63. But investigators have already noticed earlier indications of this fault. Early fault-declare time has recently been taken as 18.92 seconds (late fault-declare time as 24 seconds) based on an in-depth examination of feature values across cases (suggested by the net-determined earlier fault-declare time obtained from preliminary networks for this case).

Case 340 rises dramatically from around .1 at 284 seconds to around .9 at 290 seconds and begins to move upwards a little earlier. This is essentially the period spanned by the intermediate data for case 340.

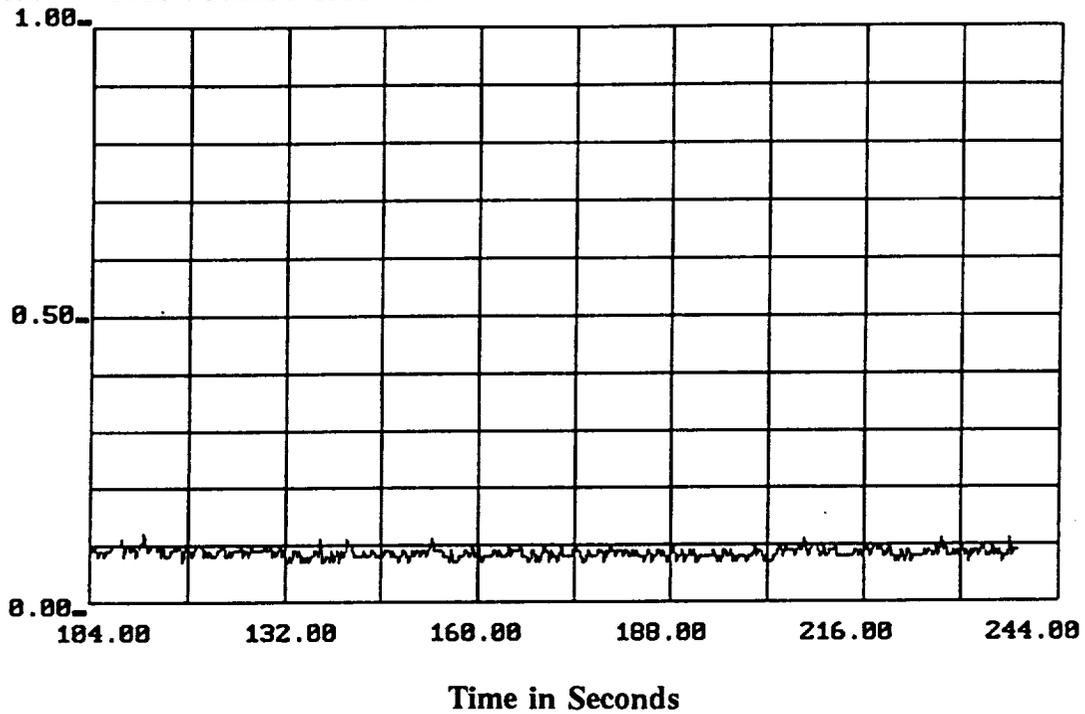
Analysis of these graphs raises the question of how to actually decide to shut down an SSME based on neural net output. Should it be done as soon as the neural net output exceeds a threshold for a certain number consecutive.

Or, should the engine be shut down immediately whenever neural net output exceeds .9? Should the SSME be shut down immediately if there is a very large jump upwards in the neural net output between two or three time samples (say a sudden jump of .3 or .4), independently of what the output was at the end of the jump?

Such issues would of course have to be resolved by any failure-detection system trying to reach the same decision.

Furthermore the question of the kind of fault-typing such networks can provide should be raised. Since every fault sample in every training data set is associated with a case, it is not surprising that almost every sample determined to be a fault sample by a holdout network generally fires on both the first output unit and one or more of the case units. This is not the case unit that occurs for that case that was not included in the training data set.

no463 No463 Holdout Case 463



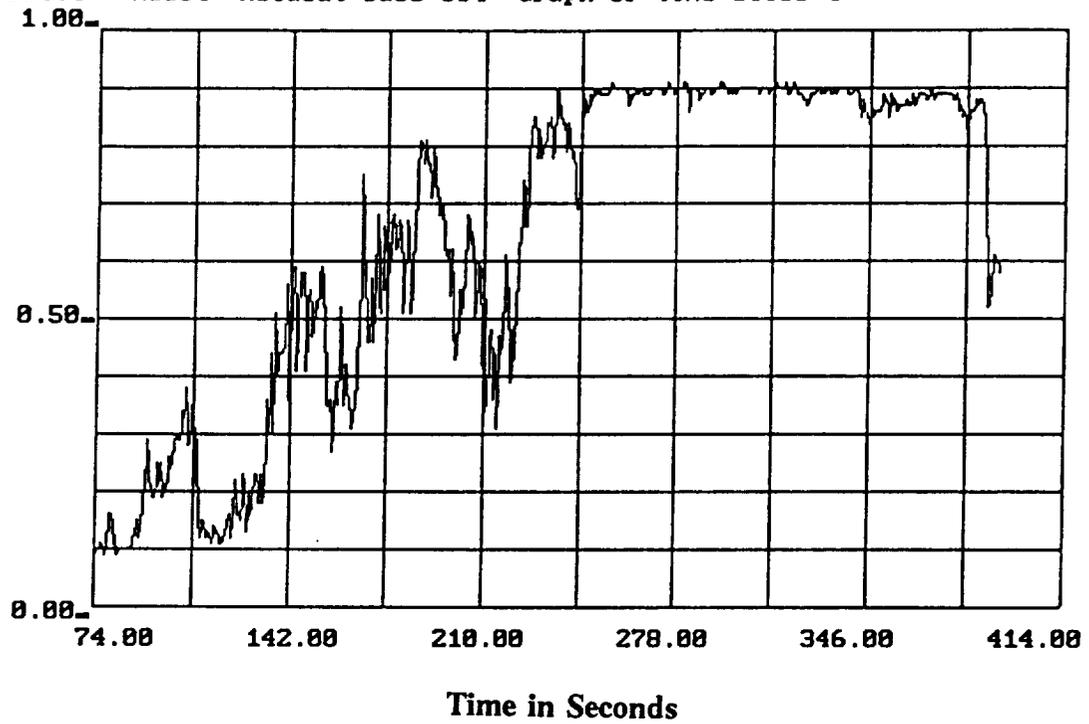
-8-

Graph of Neural Net Output Unit 0, Holdout Case 463

Nominal/Fault Unit

Figure 2.11

no364t4 No364 Holdout case 364 Graph of time slice 4

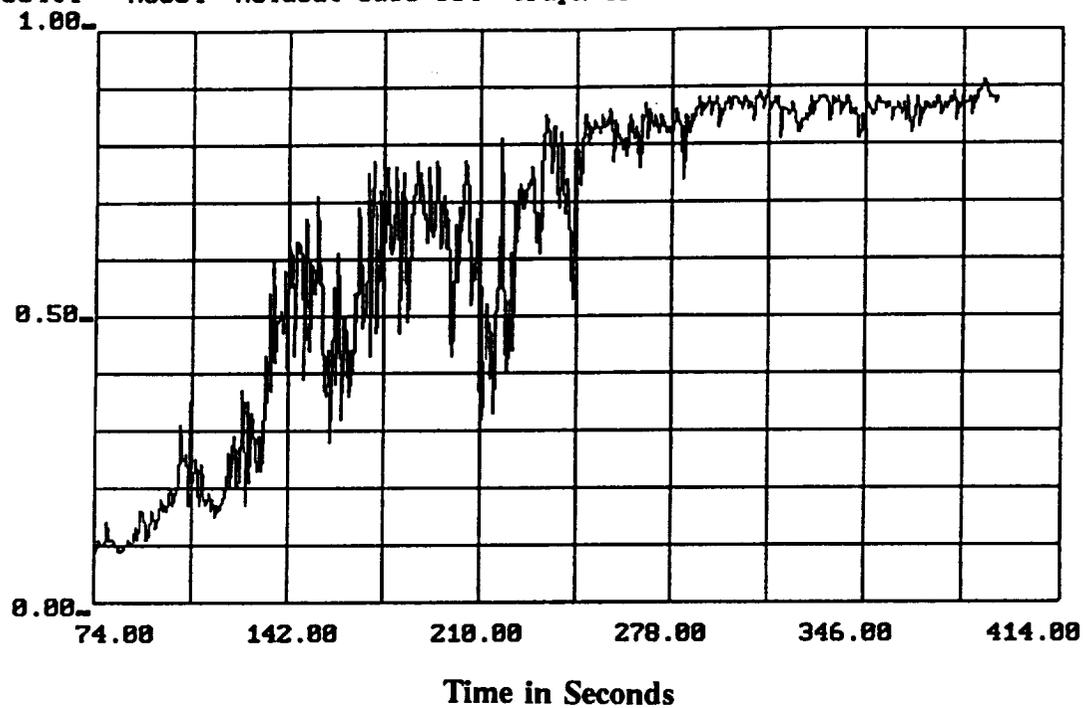


Graph of Neural Net Output Unit 0, Holdout Case 364

Nominal/Fault Unit

Figure 2.12

no364t4 No364 Holdout case 364 Graph of time slice 4

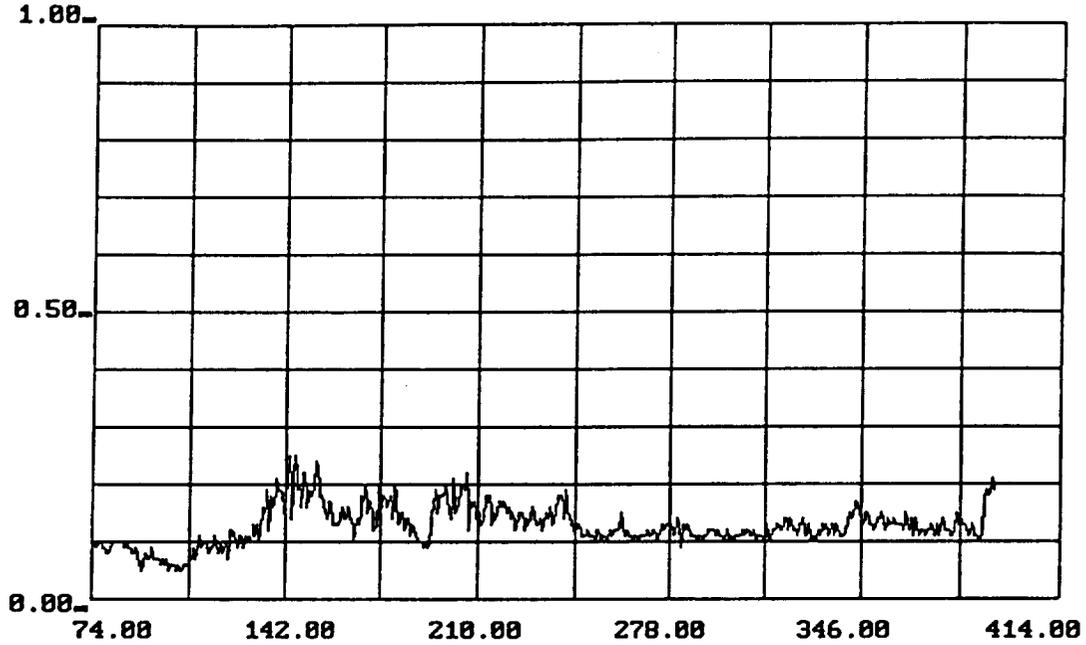


Graph of Neural Net Output Unit 3, Holdout Case 364

Case 249 Unit

Figure 2.13

no364t4 No364 Holdout case 364 Graph of time slice 4



-4-

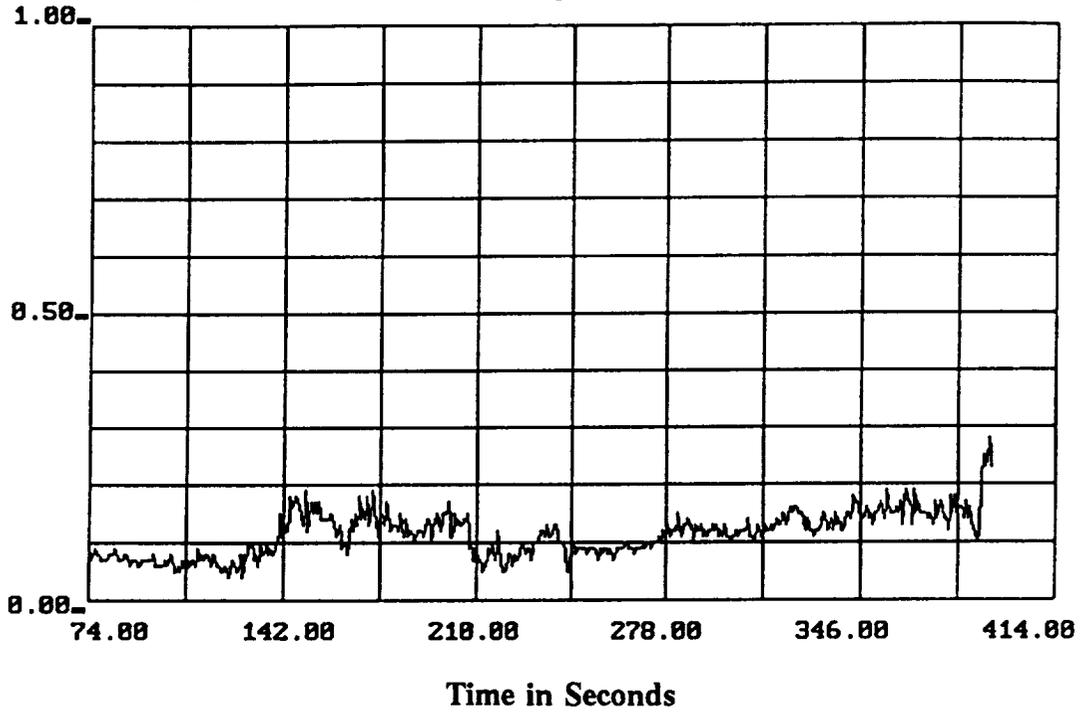
Time in Seconds

Graph of Neural Net Output Unit 4, Holdout Case 364

Case 259 Unit

Figure 2.14

no364t4 No364 Holdout case 364 Graph of time slice 4



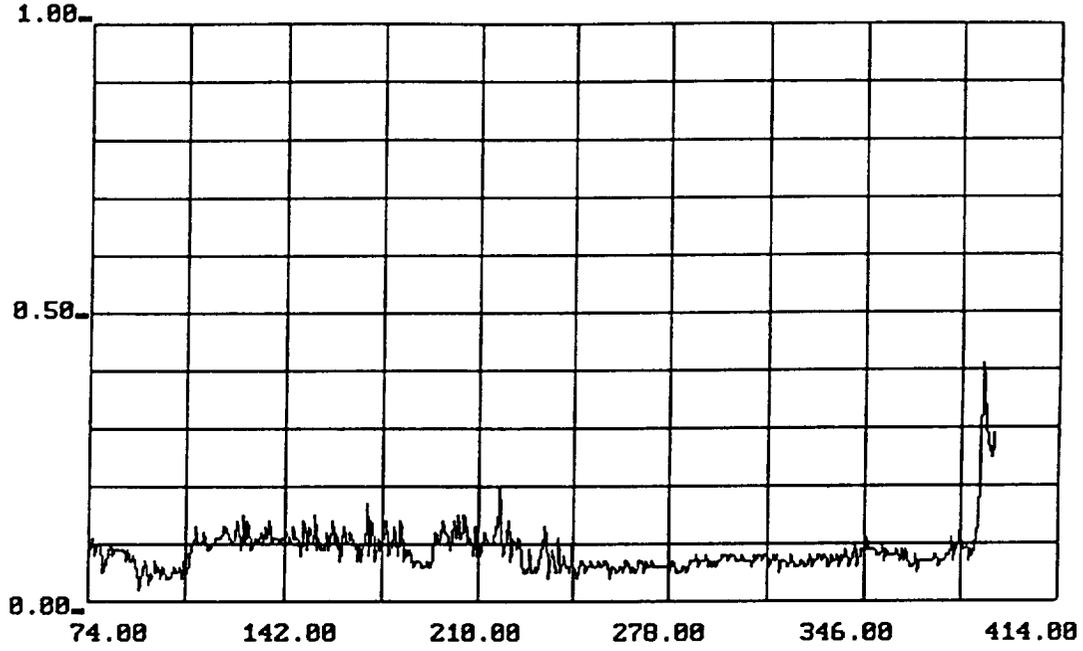
-7-

Graph of Neural Net Output Unit 7, Holdout Case 364

Case 340 Unit

Figure 2.15

no364t4 No364 Holdout case 364 Graph of time slice 4



Graph of Neural Net Output Unit 9, Holdout Case 364

Case 436 Unit

Figure 2.16

By examining the graphs one can discover that the holdout networks found the similarities between cases given in the following table. Here the fault samples in the left hand column case have been found similar to fault samples in the right-hand column cases, in the order of importance shown.

Case	Similar Cases
173	364
225	331
249	364, 436
259	340
307	225, 259
331	436, 249
340	249, 225, 364
364	249, 436
436	331,249
457	364,307
463	249

According to a broad characterization of failures into six different categories that occurs in the SAFD report produced by Rocketdyne, cases 173, 331, and 307 are all LOX Post Fractures, case 259 is a failure is an MCC outlet manifold, case 225 is a valve failure, and cases 340, 436, 364 and 249 are all High Pressure Fuel Turbopump Failures.

To the extent that cases 249, 364 and 436 have been detected as similar, the categorization by the networks reflects this division. And cases 225 and 331 both involve the oxygen side of the engine. One would have to feed into a network as training data a finer characterization than is provided now in order to get a finer characterization out of the network.

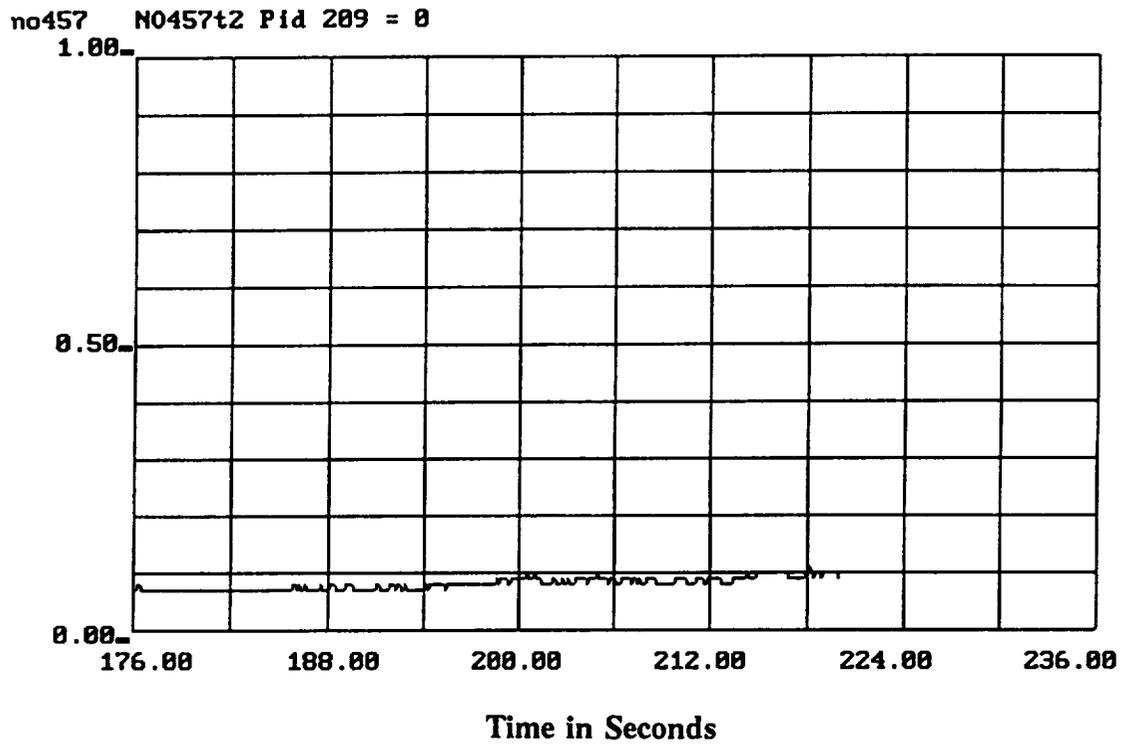
The discussion of fault-typing leads indirectly back to the problems involved in case 457. In the discussion of fault-typing given earlier it was mentioned that two approaches to this problem had been investigated, the case to case similarity just mentioned, and the elimination of PIDs by replacing their features values with zero, to see if the PID values in question had been necessary for the network to detect the fault.

The second kind of fault-typing will not be illustrated in general, but will be demonstrated in a special situation, one that provides a way of handling the venting/repressuring situation of case 457. For PID 209 is the one most strongly affected by the venting/repressurizing process. Figure 2.17 presents the result of testing the same network on case 457 data with the features corresponding to PID 209 replaced with zeros. The fault completely disappears. So a second, and simple solution to the venting/repressurization problem presented by the current "fault" in case 457 is the elimination of PID 209 from the networks' sensor set whenever venting/repressurizing is occurring. The first effective solution, mentioned above, which is also a simple one, was the use of a representative set of nominal data, which would presumably include data like that seen in case 457.

Figure 2.18 contains the graph of the nominal/fault output unit for the holdout case 364 neural network output tested on case 364 with PID 209 "zeroed-out". Note that the firing on the case 249 output unit is almost identical, the firings on case 7 and 9 slightly higher, and the firing on the nominal/fault output unit a little lower, but nevertheless very similar.

Other graphs from these tests occur in Section 4 of Appendix A, so that the effects of separating out the repressurizing influence as seen through PID 209 are evident.

In the context of the problem presented by case 457 (by venting/repressurizing factors), however, another approach to sensor fusion can be incorporated in. It turns out that there is a very simple deterministic sensor fusion method capable of catching all the faults, for which case 457 presents no difficulties. It has already been pointed out in passing that one way to combine all the features in a data sample is calculate their length, or norm (the norm of a sample is its length as a vector in 24-dimensional Euclidean space, the square root of the sum of the squares of all its feature values). Section A.5 of Appendix A contains the very interesting results of doing this for all the cases and graphing the results. The norms of the vectors in case 457 never go above 6.44. If one takes having a norm of more than 6.5 as fault-determination, the following fault-declare times are obtained (and there are no false alarms):



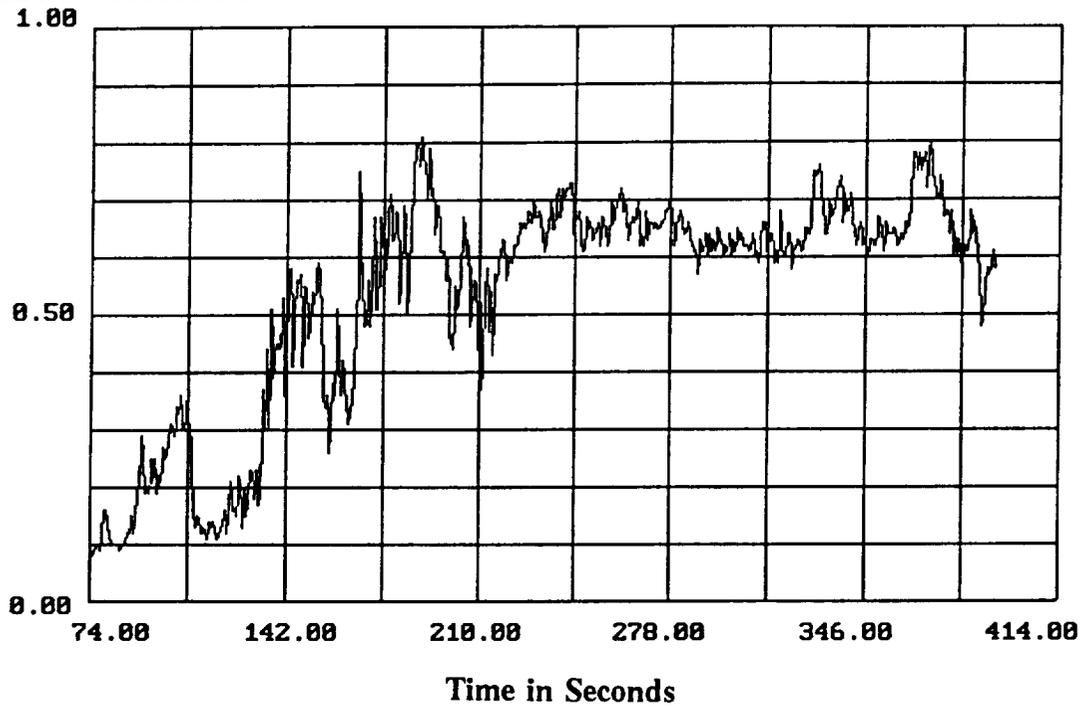
Graph of Neural Net Output Unit 0, Holdout Case 457

PID 209 Features Replaced with Zeros

Nominal/Fault Unit

Figure 2.17

no364 No364t4 Pid 209 = 0



Graph of Neural Net Output Unit 0, Holdout Case 364

PID 209 Features Replaced with Zeros

Nominal/Fault Unit

Figure 2.18

Fault Declare Times

Case	First ≥ 6.5	Analyst
225	255.52	255.52
249	337.16	320.32(330)
259	101.36	101.36
307	34.48	18.92(24)
331	232.44	232.32(232.40)
340	289.80	278.92(290.12)
364	225.48	204.32(217)
436	610.84	610.72(610.76)

Note that even case 173 is caught by this net. All its samples now seen as clearly fault, three samples, would have been caught with a limit of 6.7 on nominal samples). For the rest of the cases, a much higher limit would have caught the fault. In fact, in the gradually developing cases, a limit of 10 would have caught the fault many seconds before the time the engine was shut down on the test stand. The maximum norms for the cases, given in the usual order, were the following: 8.28, 9.09, 21.27, 23.51, 11.31, 19.89, 13.17, 18.79, 17.26, 6.44, 3.61.

These graphs of norms (which are called "absolute values" by the graphing program) have been very valuable in trying to *choose* fault-declare times. The norm is clearly a powerful sensor-fusion tool, a tool that might well always detect the faults that the networks with this current set of features can detect. But there might be other faults that these features and the norms calculated from them might not detect; faults revealing themselves through an increase in the noise level, for instance. Norms would clearly not be so helpful with any set of features including some for which it would not be a large number that would be indicative of the fault. The example of case 436 shows that if one took into consideration not the norm itself, but the rate of change in the norm, one would notice the fault even earlier.

There are many things one can deduce by computations involving the raw sensor data, and many ways to combine and use the information derived. The same features (i.e. properties of data) that can be used in deterministic methods can be used in neural networks or expert systems.

There is some art to using the norm as a sensor-fusion tool. To illustrate it, consider the fact that the following set of features might be more appropriate as inputs into the norm fusion:

1. Smoothed raw PID value, cross-time-slice and cross-engine differences factored out:

Let $Avg50(t)$ be the average of the PID value over the last 50 samples including the sample at time t . Let

$$Avg50Diff(t) = Avg50(t) - Avg50(t_s),$$

where $Avg50(t_s)$ is the average of the first 50 samples of the steady-state time slice.

Then let

$$FeaSmooth(t) = Avg50Diff(t) / SD(Avg50Diff),$$

where $SD(Avg50Diff)$ is the standard deviation of $Avg50Diff$ around its mean.

2. Rate of change feature.

Let $Avg25(t)$ be defined like $Avg50(t)$. Let

$$AvgSlope25(t) = Avg25(t) - Avg25(t-1),$$

where $t - 1$ means the time one second ago (twenty-five samples ago).

Let

$$FeaSlope(t) = AvgSlope25(t) / SD(AvgSlope25),$$

where the SD function is as before.

3. Immediate change feature.

Let $Diff(t) = |PID(t) - PID(t - .04)|$. This is the absolute value of the difference between the last two PID values. Then let

$$FeaDiff(t) = Diff(t) / SD(Diff).$$

For use in norm fusion when a decision is made on value staying above a certain range for a specified number of samples, it might make more sense to use the feature

$$FeaDiff(t) = \max\{Diff(t) / SD(Diff), FeaDiff(t_{smaller})\},$$
$$t-2 < t_{smaller} < t$$

since the feature defined the first way might be expected to become smaller almost immediately.

Notice that by always dividing by the standard deviation that has the same kind of relation to the basic object being measured, all features being fed into the norm are in some sense comparable. If one wants them assigned different levels of importance in the norm, they may be weighted in the desired way, very much as a neural network weights inputs. It should be realized, however, that when a system is created for actual use, incorporating large amounts of real data into the training data sets, the standard deviations that should be used here - and elsewhere, for instance in any feature calculation, whether for norms or otherwise - should be calculated separately for the different power levels of the engine.

2.9 Other SSME Failure Detection Systems Netrologic Has Been Investigating

Concurrently with the work on neural nets that was described above, initial investigations have been made into other approaches to fault detection:

- preliminary investigations into the norm (Euclidean distance from the origin) of the feature vector - or of the change in the length or rate of change of the length of this vector, as indicative of a fault (see graphs 2.12.5); the use of this data-fusion feature was discussed in detail at the end of Section 2.8 in the context of the help it could give concerning the evaluation of case 457 data;
- finding the "nearest neighbor" in terms of Euclidean distance from a point to be evaluated; this method has had some success in discriminating between fault and nominal data;
- finding the vector that makes the smallest angle with the sample vector; this method, which requires a great deal of refinement, as does the one just mentioned, has also had a reasonable amount of success even at its current stage of development;
- counting the number of PIDs whose features are outside the range of values observed in nominal data, or the number of features in each of several subset ranges outside the range of nominal data. This is a "pseudo-redline" approach; this is related to the SAFD approach;
- use of a probabilistic neural net; the general approach is similar to what could be called the "nearest neighbor" method;
- determining an approximation to the convex hull of the nominal data perhaps with the use of synthetic fault data, and calling nominal anything that occurred inside it, deviant anything outside it.

2.10 Recommended SSME Neural Net Investigation for the Near Future

The two items considered to be of highest priority are the investigation of the value of additional features for fault detection and the approach to neural net design for engines lacking failure data or which have not yet failed. For example, evaluating data from the second generation SSMEs, of the in-flight data from these SSMEs, for instance, using nominal/synthetic training sets is a promising approach. It is believed that the second depends on the first, so the work on both should be carried out simultaneously.

A new approach to fault-typing can be made part of this work by having part of the synthetic fault data generated by taking nominal samples and forcing one or more of the features outside nominal range or out of synchronization with other PIDs, with the inclusion of output units to record "PID X out of range" or "PIDs X and Y out of synchronization".

Another approach depends even more strongly on new data, and, in particular, data that has been very carefully analyzed. It is closely related to the approach mentioned of dividing the current data into three categories, nominal, intermediate and fault. But this approach could be taken further. The problem posed by the lack of fault data has been mentioned frequently. Nevertheless, there is some data that is not being used that could be called real fault data.

After every successful test firing, NASA analysts spend many hours examining all the sensor data recorded during the run to find out whether the engine actually developed a problem during the firing that was not noticed during the short period of time the firing lasted. Sometimes problems of differing degrees of severity are found. If a continuous scale for "engine problems" could be developed, with 10 or 15 categories, say, and a large amount of data was classified according to this scale, and synthetic data were added to fill out the top end of the scale, the network trained on such data would be evaluating the severity of the fault it thinks it might be detecting, and giving its own fault-declare time, say when 8 is reached on a scale of 1 to 10, or 12 on a scale of 1 to 15.

The NASA analysts whose help would be needed for the classification of such data could also be helpful with the first item mentioned, determining the effect of different feature sets on the discriminatory powers of the nets. For some of the first features that should be looked into are the ones these analysts are using for their own evaluations of the behavior of an engine during a firing. Some that Neurologic analysts consider valuable are:

(an approximation to) the second derivative of the PID,
the number of oscillations, i.e. changes in direction of the motion of the PID value,
the standard deviation of the PID values over a group of consecutive samples,
the difference in PID values between the last two samples,
the difference between the maximum and minimum PID value over a group of consecutive samples,
the difference between correlated PID values,
the norm of the sample vector, and properties of this quantity.

SECTION 3

VALVE SIGNATURE RECOGNITION

3.1 Problem Statement

An ATLAS rocket includes some 150 valves of various types in its pneumatic system. Each of these valves is computer controlled, and, in order to confirm correct operation, is computer monitored as well. This means that each valve must be accompanied by a remote sensor, which must itself be wired back to the controller. The sensors add significantly to the cost and time required during manufacture of the rocket because the valves-plus-sensors are substantially more expensive than the valves alone would be, and because the extra wiring needed to return the sensor data to the controller requires approximately 1.5 man-hours to install for each valve, plus several additional miles of wire to connect all the sensors. Furthermore, the sensors are not reliable. The vast majority of valve-failure readings are due to a failure in the sensor for the valve, rather than the valve itself. At this time there is no facility available to double-check the sensor's operation, and thus determine if a valve-failure signal is a true failure or a false reading. The only means of handling such problems is to manually check the system, resulting in substantial downtime while the check is being performed.

The goal of this investigation was to determine if neural network technology could assist in some or all of the above problems. Specifically, the major thrust of the effort was expended in determining if a neural network could monitor current transients along the power-bus to identify valve operations. If this were possible, then, all the individual wirings from sensor to controller could be replaced by a much smaller number of power buses. It was hoped that neural nets, which have the advantage of being highly robust both in their recognition of noisy and incomplete patterns, as well as in their ability to operate correctly even given significant hardware damage to themselves, might be able to help with this difficult problem in rocket-monitoring, where high reliability is of special importance.

The operation of a neural network in such a situation might be something like the following. A command would be sent by the controller to a valve, and the current transient that came back along the power bus would be evaluated by the network. The network would have to be able to distinguish between at least the following kinds of signatures:

- valve is opening,
- valve is closing,
- valve attempting to open is stuck closed,
- valve attempting to close is stuck open,
- valve has weak spring,
- valve has spring that is too strong.

In addition, it would be helpful if the neural net could handle the task of telling which valve was sending the current transient (for example, in case one valve started operations without commands from the controller, or if an expected current transient came slightly out of the order of commands, perhaps due to one not having arrived at all). And the weak spring and strong spring valve problems might be hard for the net to determine without having learned the normal spring signatures for an individual valve (it was found that valve signatures varied a great deal from valve to valve, even among valve of the same manufacturer).

Note that a neural net could be expected to be able to help with distinguishing between a failed valve and a failed sensor, because the current transients received along the power-bus could be expected to be very different in the two cases (for instance, if the above list included all the kinds of signatures that could be expected to come back when a sensor was functioning, nothing returning along the power bus, or something returning that was different than one of the expected healthy operation or failed valve signatures could be assumed to indicate a failed sensor).

The investigations described in this report were carried out as follows. First, data was collected in the form of current signatures from ATLAS-rocket valves installed on a pneumatic test bench. This data was analyzed in terms of the expected difficulties neural nets might have in making the required distinctions, and an appropriate neural net structure for the kind of task was developed. To test the conclusions that had been drawn, neural nets were constructed to carry out what was estimated to be one of the more difficult tasks, the identification of which individual valve was sending a signal. For this first effort, the networks were trained only to identify the valve-open or valve-closed signature of a valve.

This data was then used as training and test data for two neural networks (one to distinguish between valves sending a valve-open signature, the other to distinguish between valves sending a valve-closed signature). The neural networks were then evaluated for item recognition with respect to valve identification (and also with respect to type of valve).

Based on the understanding resulting from these investigations, this section of the report presents an assessment of neural network technology applicability and commercial potential for this and similar tasks.

3.2 Technical Background

The "Smart Built-In-Test", or Smart BIT, pneumatic bench provided invaluable valve current signature data for these investigations. This complex hardware/software construct has a mockup of a pneumatic system including pressure control valves, pressure control regulators, relief valves, pneumatic actuated valves, purges, and a pressure storage vessel. A common electric shunt is provided to monitor all the valves on the bench. Figure 3.1 shows the schematic representation of the prototype test bench.

Figure 3.2 shows a photograph of the test bench and the computer and control and monitoring electronic hardware. The control software which activates all the proper valves and instrumentation for a particular test and provides real-time on-screen color graphics plotting the pressure, temperature, current, voltage, frequency, or other data of interest vs time as well as a display of pertinent engineering data (maxima, minima, deltas, etc.) was written in the ASYST control language. A sample of the program code used to collect the current signature data is given in Appendix B, Section B.6.1.1.

Early work with this bench had involved the capturing of signatures of different kinds from three types of solenoid valves, the Marotta, Circle Seal, and Wright Component valves. The most significant findings were that the kind of signatures mentioned in the previous paragraph were easily differentiated by the human eye, but that the quality of the signature was representative of the quality of the valve. More explicitly, the low-cost valves were very inconsistent in their behavior from cycle-to-cycle, while high quality flight type valves such as those used in the Centaur RCS (Wright Component valves) were extremely consistent.

The variables found to be the most repeatable were actuation time (time for the valve poppet to shuttle), and inductance (induced voltage generated by the poppet passing through the solenoid coil). Inductance is recognized by the dips seen in the current signature. To see if faults could be readily identified, a Circle Seal valve was locked up. As expected, the familiar dip in the current trace was not present. Sticking/friction was simulated using the Wright Component valves by way of increasing pressure on the pressure assisted seat. (Mechanical faults to the valves were not developed beyond this point as Wright Component valves are an all welded construction, and the Marotta valve was too erratic for developing a data base.) A potential concern at the beginning of valve signature investigations with the pneumatic bench was the possible effect of valve temperature. Testing concluded that temperature has little effect except when a valve was first actuated after a long period of being off.

Based on these results from the Smart BIT IRAD, the valve data was determined to be distinct enough that a neural network should be able to determine the state of a valve, and possibly which valve opened or closed. Thus the early work with the Smart BIT IRAD that showed a larger variation among individual valves and between valves of different manufacturers than expected, while perhaps discouraging to those who were hoping for more uniformity to make replacement of valves involve less complications, is advantageous for the purpose of distinguishing between valves in the context that has just been discussed of using a neural network to identify specific valves and their actions. And neural nets can easily be retrained to recognize the new signature when a valve is replaced by another.

SMART-BIT PROTOTYPE BENCH SCHEMATIC

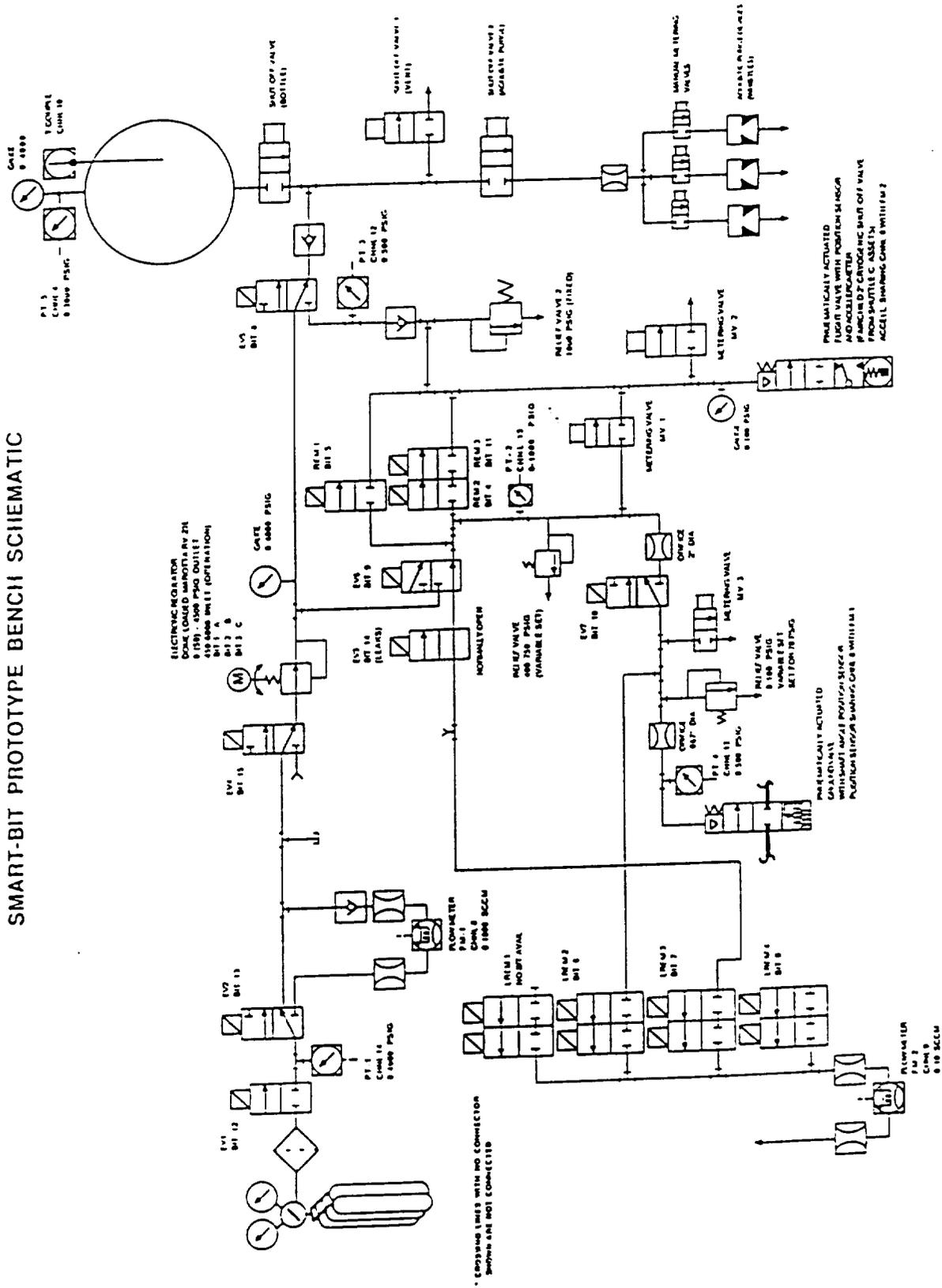


Figure 3.1. Schematic Representation of Smart Built-In-Test Pneumatic Test Bench

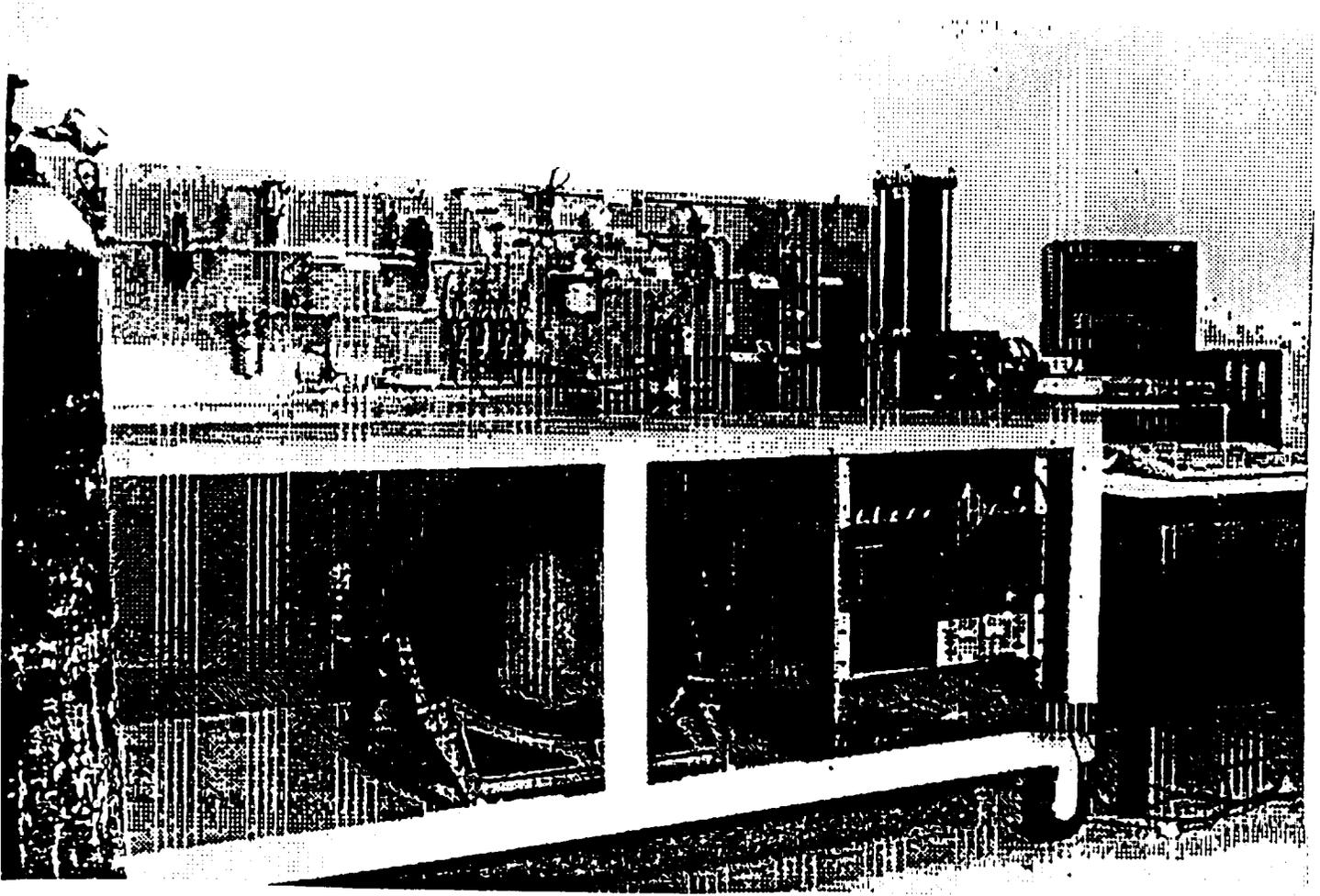


Figure 3.2. Photograph of Smart Built-In-Test Pneumatic Test Bench

3.3 Test Design

To understand the test design, it is necessary to understand the three types of valves on the Smart BIT test bed. Each of these valve types also has several parameters that can be changed to modify their response and artificially induce failures. The three valves are illustrated in Figures 3.3, 3.4, and 3.5 below.

3.3.1 Valve Types

3.3.1.1 Circle Seal Valve

Five Circle Seal valves were used during the test. These valves, arbitrarily designated as "type 1," have a return spring and an adjustment screw that controls the tension of the spring.

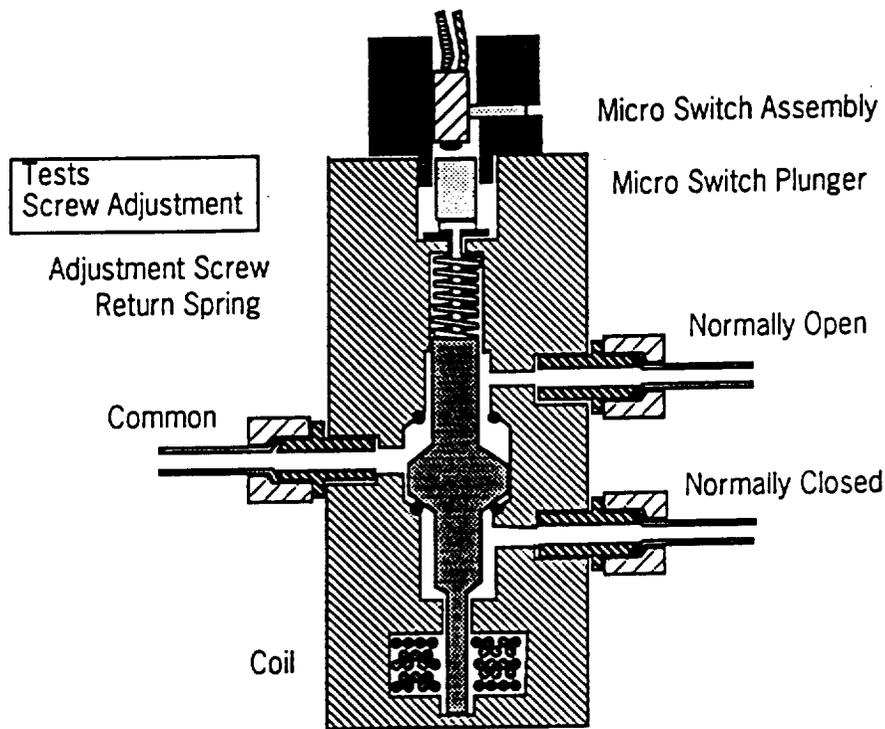
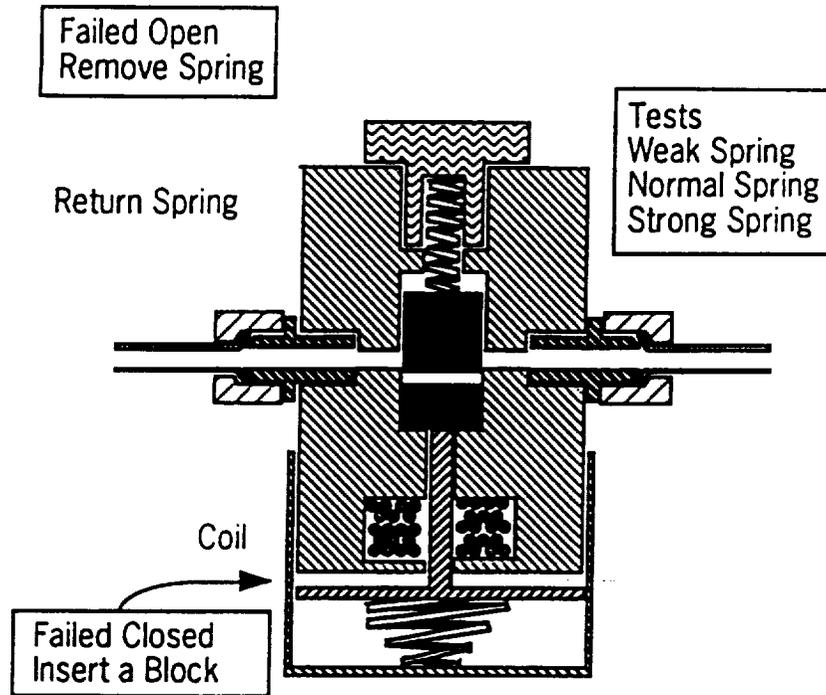


Figure 3.3. A Circle Seal valve.

3.3.1.2 Wright (REM) Valve

Three Wright (REM) valves were used during the test. These valves were arbitrarily designated "type 2." They are actual flight valves, and are in a sealed, contained unit. None of these valves were disassembled, and no interior detail design was available to illustrate them. The Wright valves are exceptionally fast valves, with a response time that is considerably faster than either the Circle Seal or Marotta valves.



Normally Closed

Figure 3.4. The Marotta Valve.

3.3.1.3 Marotta Valve

There were two Marotta valves on the test bed, designated as "type 3" valves. These valves have a return spring that can be varied by substituting the spring with one with a weaker or stronger spring constant. In addition, the valve action can be artificially blocked by placing a physical barrier that prevents the valve from closing completely. Such a condition is called a "failed closed" because the valve failed to close properly. A "failed open" state can also be induced in the valve by removing the return spring entirely, thus preventing the valve from opening on command.

The electromagnetic operation of all of these valves is shown in detail in Figures 3.5 and 3.6. In essence, the change in current in a coil winding around a magnetic coil reverses the direction of the induced magnetic field, thus causing the core to "pop up" out of the central winding (Figure 3.5). This causes the return spring to compress, so that when the electromagnetic conditions are changed by a "close" command, the poppet core is forced back down into the windings (Figure 3.6).

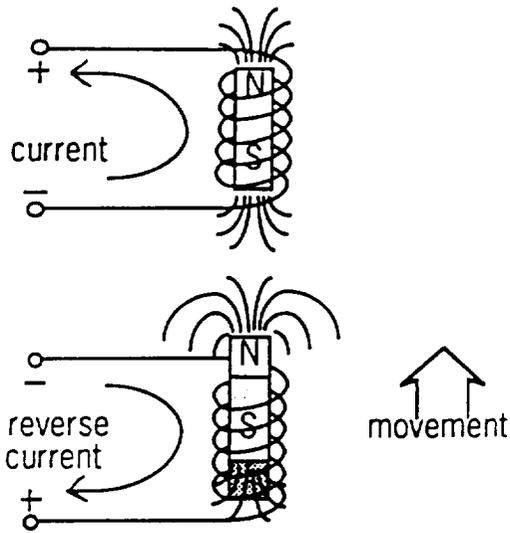


Figure 3.5. Applying a current through the coil windings causes the poppet to move, which generates a reverse current in the coil.

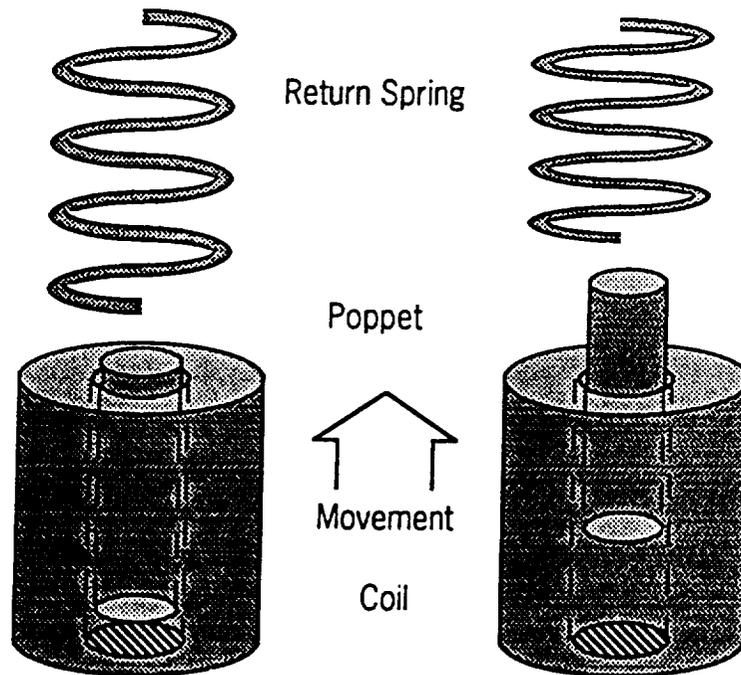


Figure 3.6. The compression of the return spring when the poppet core pops up provides the tension necessary to force the poppet back down into the coil once the electromagnetic conditions permit.

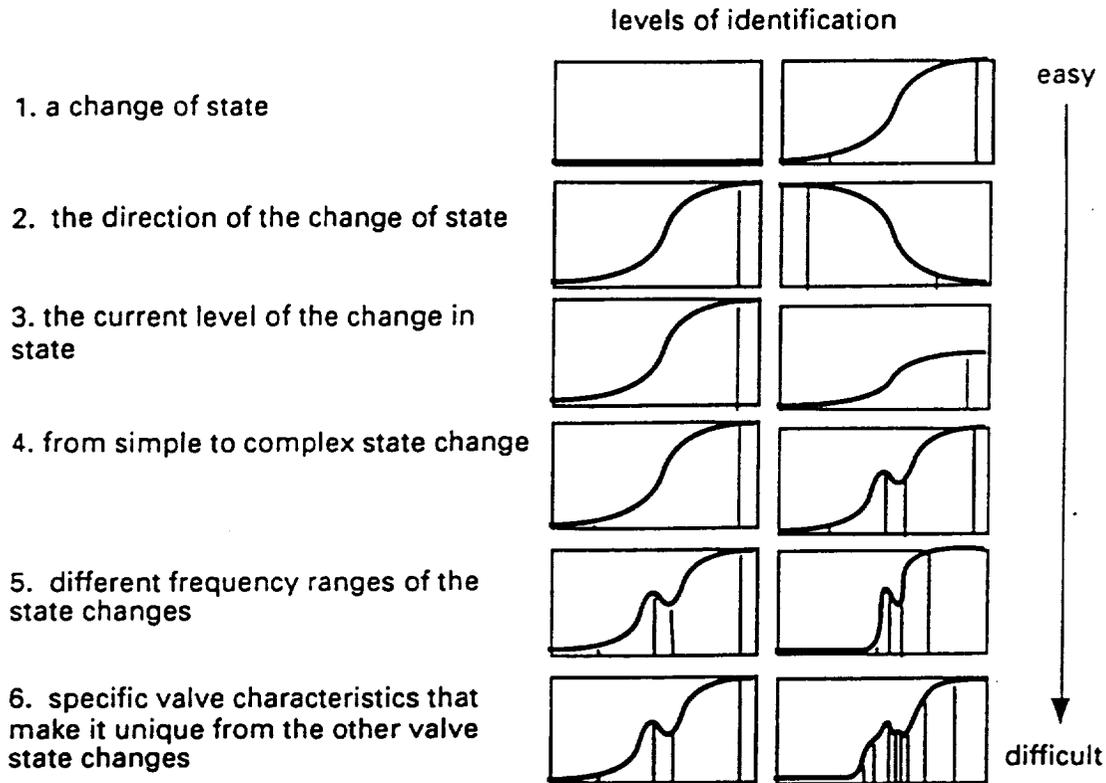


Figure 3.7. Sample valve signatures showing relative ease of identification.

3.3.2 Signature Characteristics

Given the above valve types and inducible failures, the signature curves themselves become of interest. Figure 3.7 illustrates several key signature features that can potentially be used to determine the state of the valves on the test bed.

The first determination to be made is whether or not a valve has changed state. A typical signature changes from a flat curve to rising curve as shown in the figure. This signature would occur if a valve opened. When a valve closes, the curve is a shorter, falling curve from the open-state current level to a zero current level. Generally, it takes less time for the valve to close than it does to open.

The second determination is to decide what happened (i.e., whether the valve opened or closed) by observing the slope of the signature. A rising curve indicates a valve opened; a falling curve indicates the valve closed.

The third determination to be made is to find the current levels for the given state change. This is the first indication that can be used to distinguish valve types. Typically, a Circle Seal valve requires less than 500 milliamps to open; a Wright (REM) valve requires nearly 900 milliamps, and a Marotta requires more than 900 milliamps.

The fourth signature characteristic to be determined is whether the signature curve is simple or complex. During the time needed for a valve to actually open or close, the action is not always smooth. For example, the current necessary to start the poppet moving (i.e., overcome its initial inertia) may be fairly high, but the current may drop sharply once it has begun to move. Similarly, other effects such as the strength of the spring, spring "bounce", friction, and other factors, may all change the signature curve from a smooth sigmoidal shape to a complex curve. Ideally, these characteristics can be used to distinguish between valve types or even between valves of the same type.

A fifth characteristic is the speed at which the curve characteristics occur. If the signature is taken over a fixed time period of, say, 75 milliseconds, a Wright valve can be expected to act within 20 milliseconds, while a Circle Seal may not fully open or close for nearly 70 milliseconds. The Marotta valves typically take an intermediate amount of time to act.

Finally, all the previous characteristics combined provide a unique valve signature that distinguish each valve from any other.

Of the signature types noted, it was determined that the first two were of such triviality that they were inappropriate to use as training material. Instead, the focus of the effort was to determine how many of the remaining four effects could be reliably detected by a neural network.

Figure 3.8 illustrates the general characteristics of a typical current signature for each of a valve-open and valve-close command, with characteristic features of each marked.

3.3.3 Signatures for the Valve Types

Data was collected for each of the above three valve types, to determine if their characteristic signatures were sufficiently distinct to be used as a determining characteristic. Figures 3.9, 3.10, and 3.11 show typical signatures for the Circle Seal, Wright, and Marotta valves respectively. Even a casual comparison of the curves reveals the differences in signature between the valve types.

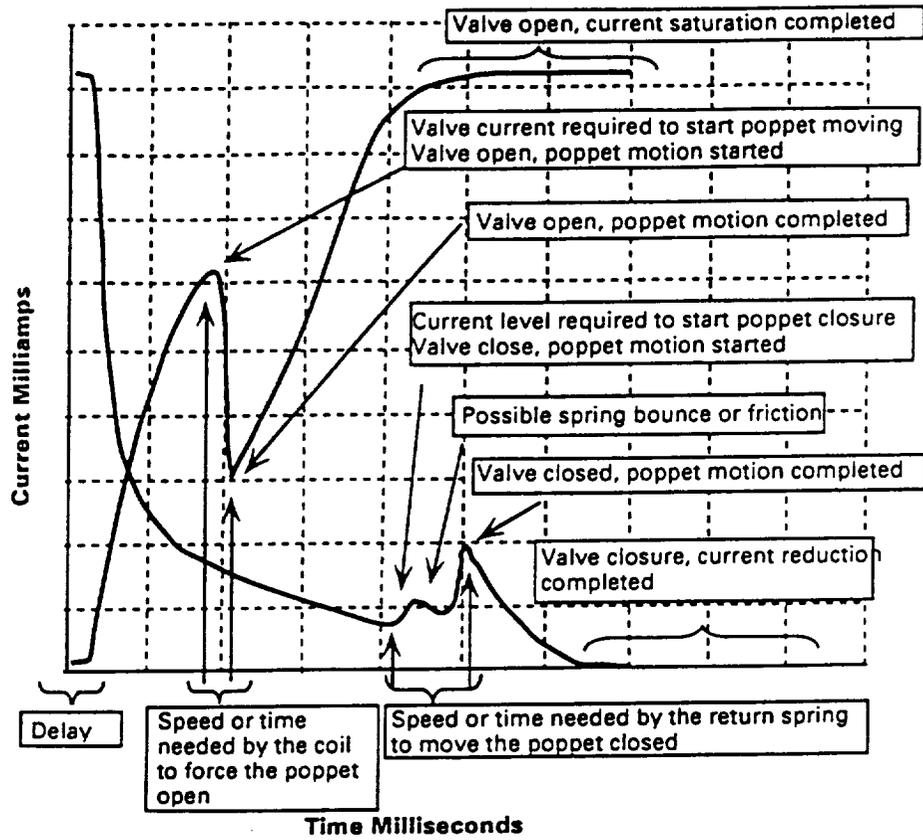


Figure 3.8. A typical current signature with features marked.

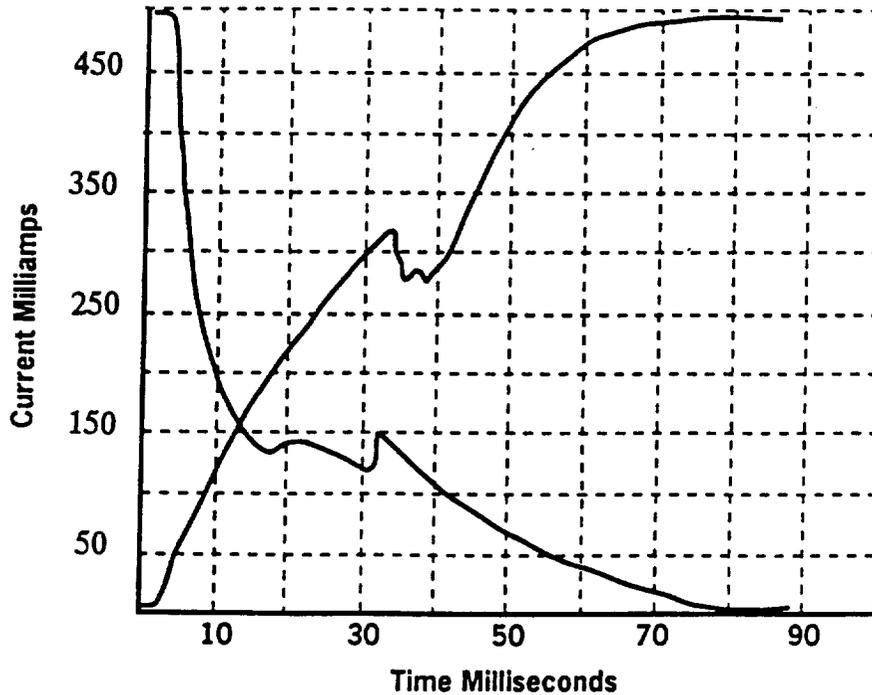


Figure 3.9. Typical Circle Seal signatures for valve-open (increasing current) and valve-close (decreasing current) state changes.

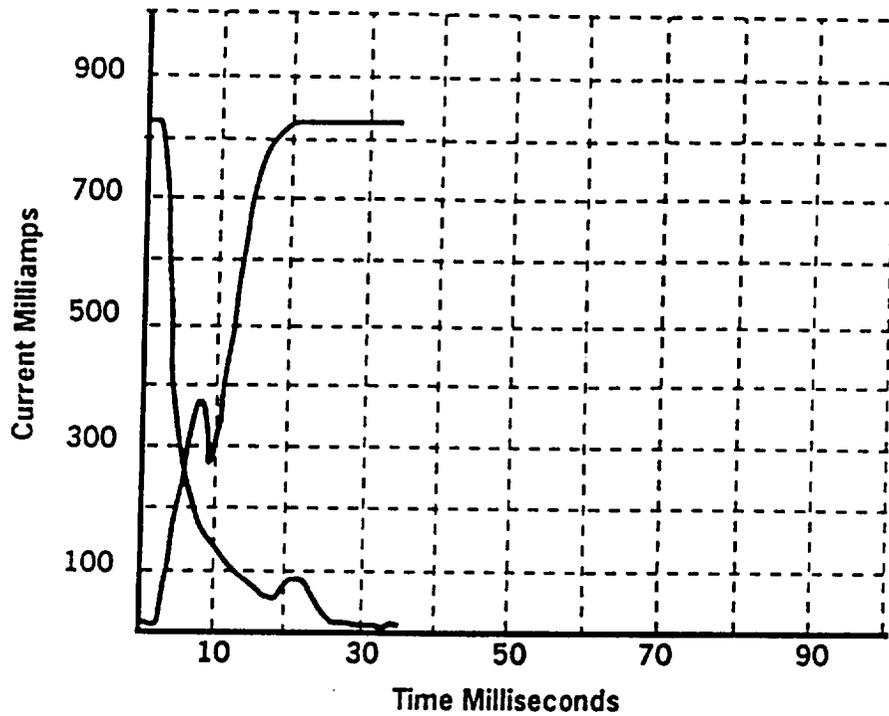


Figure 3.10. Typical Wright valve signature curves. Note the higher current and shorter response time compared to the Circle Seal valves.

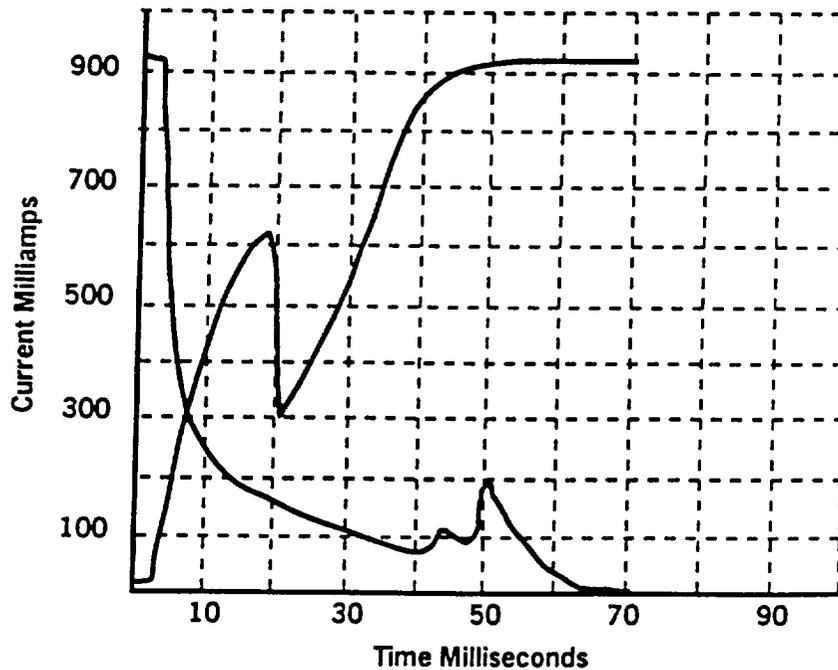


Figure 3.11. A typical set of signatures for a Marotta valve. Notice that these valves have both the high current of the Wright and the long response time of the Circle Seal valves.

In figure 3.12, all three valve types are superimposed on a single graph so that the differences are clearly delineated.

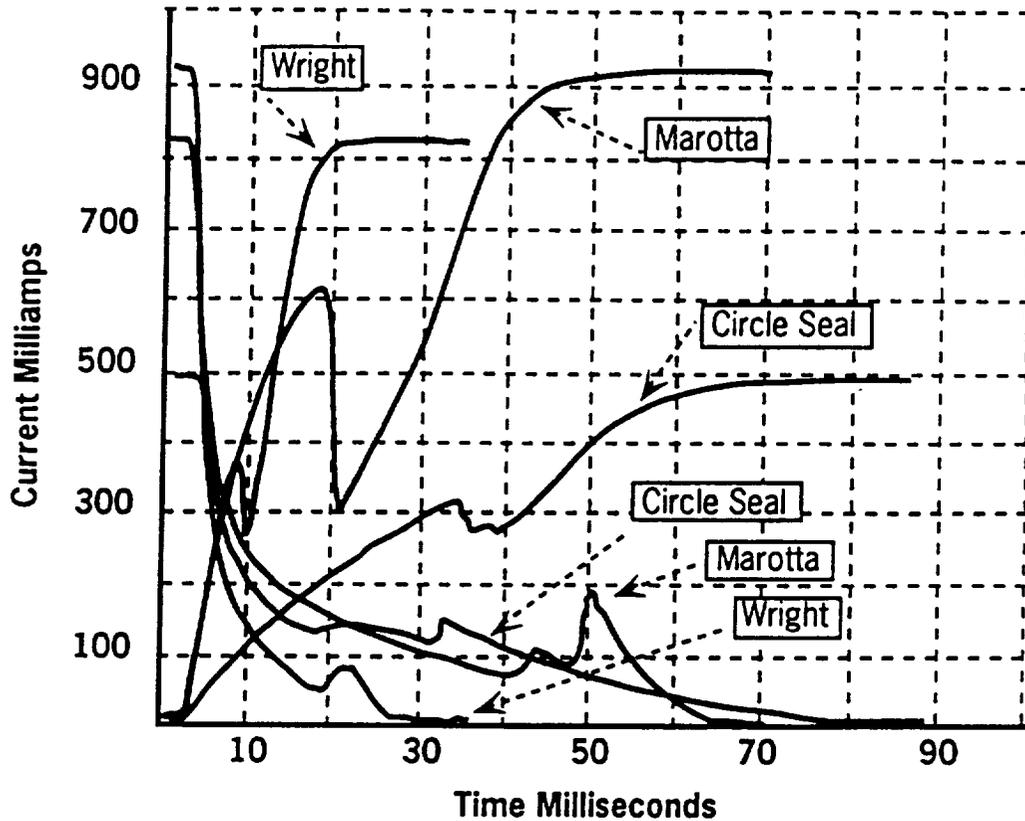


Figure 3.12. Signatures from all three valve types are superimposed to delineate the differences between them.

It should be noted here that one of the Circle Seal valves had a signature that varied significantly from the other valves of the same type. This valve's signature is shown in Figure 3.13. It is not clear why this valve is so different from the other Circle Seal valves; no failures or misadjustments were associated with this valve's signature. To see more clearly just how different this valve's signature is from the more typical Circle Seal signature, Figure 3.14 graphs the signatures from both the anomalous valve (#15) and a more typical valve (#12).

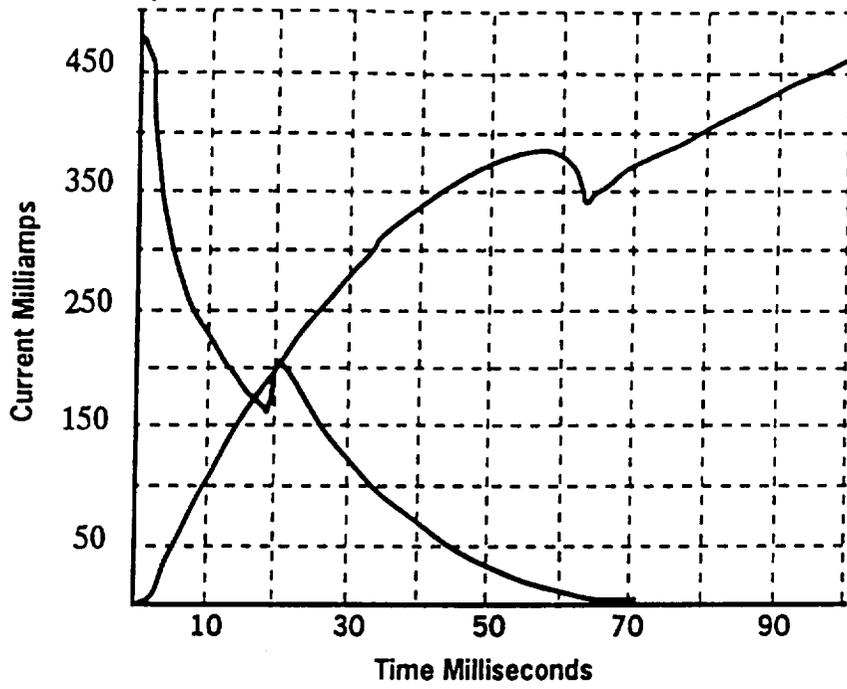


Figure 3.13. One Circle Seal valve had an anomalous signature.

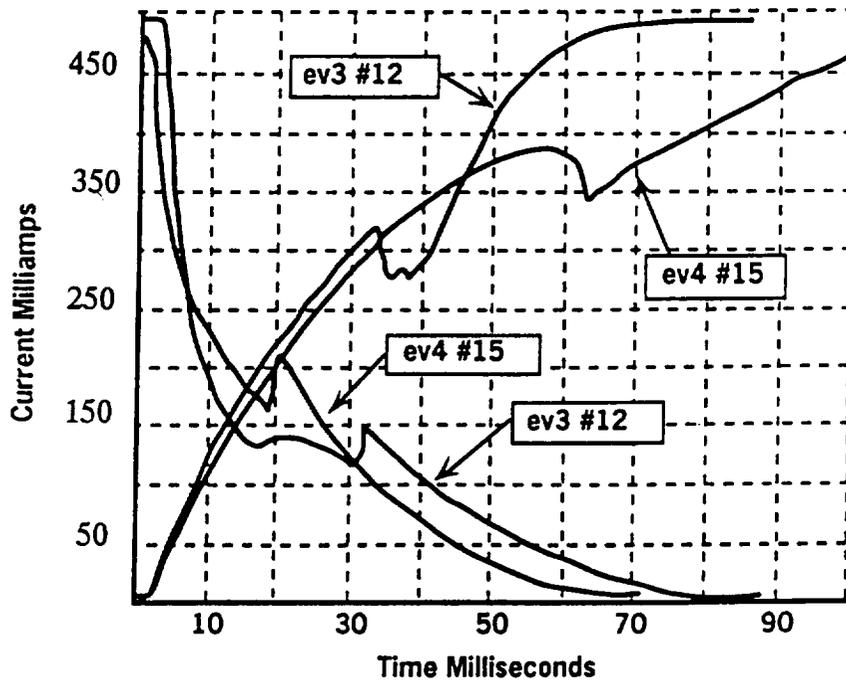


Figure 3.14. A comparison between the signatures of the anomalous Circle Seal valve (#15) and a more typical one (#12).

Table 3.1 Data Run Header Format

Item	Size	Type	Explanation
1.	128 byte	string	time, date, comment about the test
2.	2 byte	int	failure code
3.	2 byte	int	valve number
4.	2 byte	int	open time (100 msec normally)
5.	2 byte	int	open samples (400 normally)
6.	2 byte	int	close time (100 msec normally)
7.	2 byte	int	close samples (400 normally)
8.	2 byte	int	number of invisible (unsaved) runs
9.	4 byte	real[array]	open data of size [open samples] in length
10.	4 byte	real[array]	close data of size [close samples] in length

Table 3.2 Valve Number and Failure Code Clarification

Num	Name	Type	Comments
12	ev1	Circle Seal	solenoid on the bottom
13	ev2	Circle Seal	solenoid on the bottom
15	ev4	Circle Seal	solenoid on the top
9	ev6	Circle Seal	solenoid on the top
8	ev5	Circle Seal	solenoid on the top
5	rem1	Wright	two valves tied together
4	rem2	Wright	two valves tied together
11	rem3	Wright	two valves tied together
14	ev3	Marotta	slightly different model than the next one
13	ev2m	Marotta	the wires are removed from ev2 and attached to ev2m (the ev2m valve is not in the system but is just laid on top of the bench - this is also the valve that was taken apart and used in evaluation of the effects of failures on the current signatures)

3.3.4 Smart BIT Data Collection and Format

The current signature samples were written to a series of files with both the open and close data in the same file. The data collection program (detailed in Appendix B, Section B.6.1) stores each of the collected data runs in a separate file with the number of the run and the number of the valve as part of the file name. The data files are formatted with the information listed in Table 3.1.

Within the valve's current signature data files explanations in Table 3.1, two data items still need more clarification. The two items are the valve number and the failure code that was used. The valve number is the hardware bit number that is used to control the activation of the valve on the test bench, and so a cross correlation of the valves in Figure 3.1 with the type and class of valve needs addressed. This data is listed in Table 3.2.

The failure code is an indication of the deviation from the normal state of the valve. This change in status is indicated with a value other than zero in the failure code of the file. Such a change may be that the valve has been failed open (i.e., failed to open) or failed closed (i.e., failed to close), the spring has been replaced by a weaker or stronger spring than normal, or that there has been pressure introduced into the system, or even that the data has been stored in a different format than originally stored. The list of failure codes that were assigned are listed in Table 3.3.

Table 3.3 Failure Code Meanings

Num	Description
0	none (no failure - normal operation)
1	failed open (no return spring)
2	failed closed (shim or block inserted to stop valve from operating)
3	weak spring (high friction on return)
9	raw data (no failure)
10	100 psig
11	150 psig
12	200 psig
13	250 psig
14	300 psig
15	350 psig
16	400 psig
17	450 psig
18	500 psig

In order to ensure that the valve data were usable, several test runs were made before the collection of data was started. During these test runs it was noted that the current signature drifted as each of the consecutive tests were executed. The data collection program was set up to make "invisible runs," which were not stored into data files. It was found that the drift did indeed stop after such a sequence of runs, stabilizing the valve's signature. The invisible runs are performed in the same way as the rest of the runs except that their signature data were not stored.

3.3.5 Neural Network Design

The networks used for this project were backpropagation networks, which have been demonstrated to have superior pattern recognition capabilities over a wide variety of applications. To reduce programming effort a commercial simulator, SAIC's ANSim ("Artificial Neural Systems Simulator"), was used for training and testing. A C-language program was written to convert the raw data into a format that ANSim can process; the listing for this program is in Appendix B, Section B.6.2 of this report. The backpropagation networks were three-layer networks with 300 neurodes in the input layer, 9 neurodes in the output layer, and 25 neurodes in the middle layer.

Preprocessing of the input data consisted primarily of minor format changes. Each of the signatures was initially in a separate binary data file, with header information identifying the valve number, run information, and so on, as described in Appendix B, Section B.6.1. For ANSim to train the network, a collection of runs had to be gathered into a single file, with a special header placed at the top; ANSim is also rather particular about the format of the data in the training file. Data was collected for 400 measurements over 100 milliseconds in all cases; however, to reduce the dimensionality of the input layer of the neural network, only the first 300 measurements (75 milliseconds of data) were actually included in the input patterns used for training and testing. Additional preprocessing was performed to scale the input patterns to a -0.5 to +0.5 range. Since the raw data had an actual range of approximately 0 to 100 ma., this was done by dividing each current value by 100.00 to give a number between 0.0 and 1.0, and then subtracting 0.5 to get a number in the final range. In the case of the valve-close data, an additional normalization operation, performed by the data normalization routine within ANSim, was also done. This normalizes (removes the mean from) and scales the input values across the entire file. The normalization values were saved in a special data file, and all test pattern files were normalized using these same values.

The output format of the data was arbitrary, and was a 9-element binary pattern. The first three (leftmost) elements designated the valve type; the next four designated the valve number in binary; and the last two indicated whether the pattern was a valve-open or valve-close state change. (Since two separate networks were ultimately used for valve-open and valve-close, these last two elements were the same for all patterns presented to each network. In other words, each network either always saw a "1 0" pattern or a "0 1" pattern for these two elements.)

Table 3.4. The output pattern element definition (all elements are either 0 or 1)

CircSeal	Wright	Marotta	Valve #	Valve #	Valve #	Valve #	Open	Closed
Example: Circle Seal, valve #12, Valve-Open pattern								
1	0	0	1	1	0	0	1	0

3.4 Test Results

Two neural networks were trained to recognize the valve signatures from the data collected from the test bed. One network was trained to distinguish between signatures for a valve-open state change (rising current) and another was trained to distinguish between signatures for a valve-close state change (falling current).

3.4.1 Training Procedure

Training data consisted of the first and last data collection runs from each valve used on the Smart BIT test bed. For some valves, 20 test runs were collected; for others, 40 test runs were collected. In either case, the first and last test runs for each of the ten valves were included in the training set. The rationale for this approach was that the first "cold" run and the last "warm" run for a given valve ought to span the expected behavior of that valve. (This expectation was confirmed by informal observations of the collected data.) Test pattern sets were derived from the intermediate runs. For example, if a given valve had 20 runs made on it, the training data for that valve consisted of runs 1 and 20; test sets for that valve included runs 5 and 15.

Training was continued until the total RMS error for all twenty patterns in the training set was less than or equal to 0.10. Since each of the 9 output elements has an output range of 1.0 (-0.5 to +0.5), and since there were 20 training patterns for each network, this error level implies a total error of approximately 0.05%. The valve-open network trained in approximately 250 passes through the training set to achieve this performance level. The valve-closed network required approximately 2800 passes to train to the same level of competence. The difference in training requirements arises from the relatively subtle distinguishing features between the valves for the latter case. In essence, as can be seen by Figure 3.12, there is much more variation between valve types for the valve-open situation than there is for the valve-close situation, making the latter much more difficult for the network to learn.

The close-valve network was trained with noise on. This means that the network was presented with slightly noisy versions of the training patterns every pass through the training set, rather than the exact patterns themselves. In effect, this means that the network literally never saw exactly the same pattern twice, since in each pass the noise is randomly and dynamically added to the input patterns.

Both networks rapidly learned to distinguish between the three valve types; more than half of the training time was used to distinguish the specific valve number for each pattern. The Wright valves were the most difficult for the networks to learn in both cases. This is most likely due to their very fast response time. Because they responded so quickly, the "granularity" of the signature data is much greater for these valves, resulting in a loss of fine detail in the signature curves. This could be corrected by taking readings more frequently, particularly in the first 20 milliseconds of the curve. As a result, these patterns are extremely difficult for the network to learn.

3.4.2 Test Procedure and Results

Both neural networks performed exceptionally well during testing. To assess the networks' responses, the training data set and the test data set for each was processed through the networks at a variety of noise levels, from 0% noise to 50% noise. "10% noise" in this context means that a random number between 0.10 was added to each element of the input pattern before processing it through the network. Since the dynamic range of the input data is restricted to the range -0.50 to +0.50, or a total of 1.0, this is equivalent to adding error bars to each pattern element. In the case of 10% noise, these error bars would be the equivalent of plus or minus 100 ma for each reading of the pattern. Notice that as shown in the signatures of Figure 3.12, many of the features of the curves had a total magnitude of approximately 50 to 100 ma. Thus, even 10% noise imposed a significant penalty on the network trying to distinguish the curves from each other.

Each network's responses were evaluated as "correct" "incorrect" or "not sure" for each pattern processed. A "correct" response was assigned only if every element of the network's output matched the desired response in color. In other words a desired +0.5 response required an output that was at least +0.15 or greater, and a desired -0.5 response required an output of at least -0.15 or less. On the ANSim display screen, responses more positive or more negative than this are displayed with a varying color scale that shows definitely red (for a positive output) or blue (for a negative output). When the network's output was between -0.15 and +0.15, ANSim produces a black-and-white indication; these were marked as "not sure" if only one such output pattern element had the black-and-white indication and if all other pattern elements were correctly designated by color. All network responses that were not "correct" or "not sure" were marked as "incorrect."

Tables 3.5, 3.6, 3.7, and 3.8 summarize the networks' responses. Tables 3.5 and 3.6 present the Open Network's test results for the training and test data respectively. Tables 3.7 and 3.8 present the Close Network's test results for the training and test data respectively. The first column of each table lists the noise level for that pass. The second column lists the number of times (out of 20 patterns in each file) the network produced the incorrect valve type (type 1=Circle Seal, type 2=Wright, type 3=Marotta); the third column lists the number of times the network produced the incorrect binary valve number. The last column lists the file name on the disk provided (Disk #3) that contains the details of the network's responses for that pass. When the network had one or more "not sure" responses, those are listed in parentheses beside the number of errors. Thus, in Table 3.5 at the 5% noise level, the Open Network has a Valve No. Errors entry of "0 (1)"; this means that the network made no errors, but had one pattern in which the valve number had one pattern element that was black-and-white. Two or more black-and-white pattern elements counted as an "incorrect" response.

Table 3.5
Open Network's Performance with Training Data

Noise Level	Valve Type Errors	Valve No. Errors	Results File Name
0%	0	0	O00TRN.RES
5%	0	0 (1)	O05TRN.RES
10%	0	1 (1)	O10TRN.RES
15%	0	2	O15TRN.RES
20%	0	6 (1)	O20TRN.RES
25%	0	6	O25TRN.RES
30%	1	9	O30TRN.RES
50%	1	10	O50TRN.RES

Table 3.6
Open Network's Performance with Test Data

Noise Level	Valve Type Errors	Valve No. Errors	Results File Name
0%	0	0	O00TST.RES
5%	0	0	O05TST.RES
10%	0	2 (1)	O10TST.RES
15%	0	3 (1)	O15TST.RES
20%	0	6 (1)	O20TST.RES
25%	0	8	O25TST.RES
30%	0	7	O30TST.RES
50%	2	13	O50TST.RES

Table 3.7
Closed Network's Performance with Training Data

Noise Level	Valve Type Errors	Valve No. Errors	Results File Name
0%	0	0	C00TRN.RES
5%	0	0 (1)	C05TRN.RES
10%	0	0	C10TRN.RES
15%	0	6 (1)	C15TRN.RES
20%	0	9 (2)	C20TRN.RES
25%	2	6 (1)	C25TRN.RES
30%	3	9 (1)	C30TRN.RES

Table 3.8
Closed Network's Performance with Test Data

Noise Level	Valve Type Errors	Valve No. Errors	Results File Name
0%	0	0	C00TST.RES
5%	0	0 (2)	C05TST.RES
10%	0	4 (1)	C10TST.RES
15%	0	6 (1)	C15TST.RES
20%	0	5 (1)	C20TST.RES
25%	0 (1)	7	C25TST.RES
30%	2	11 (1)	C30TRN.RES

No errors were made by either network on any of the training or test patterns when noise was not present. In fact, both networks performed admirably on both the training and the test data sets when noise was limited to 10% or less. Recalling that this level of noise is the same as placing error bars on each measurement of 100 ma of current, it is astounding that the network is able to correctly determine both the type and number of the valves so well with this level of variation. Even more surprisingly, the networks were generally able to determine the type of valve with noise levels of 25% to 30% or more. The Open Network was even able to correctly distinguish the type of valve with noise levels of 50% (500 ma uncertainty!) more than nine times out of ten. Given that the scale of the features in the signature curves are on the order of 50 to 100 ma in size, this is an exceptional feat.

The included network disk (Disk #3) contains the result files for all these tests. The format for these files is described in Appendix B, Section B.6.2 of this report.

Overall, the results of this project have been extremely heartening. It is clear that neural networks can indeed learn to "read" the power bus transients and determine what happened, what kind of valve it happened to, and which specific valve it happened to. This has great implications for reducing the cost of ATLAS rocket valve monitoring systems. At the very least, it may provide an independent method of confirming the accuracy of valve-sensor readings.

3.5 Future Research Directions

In the process of conducting this research, a large number of questions remained unanswered. These questions are appropriate topics for future investigations. In some cases, enough data was collected on this project to begin to explore them; in others, more Smart BIT test data will be required.

3.5.1 Failure Determination

If the valve fails, can the fact that it failed, plus the kind of failure be distinguished by a neural network? In this regard, for example, can poor performance of a valve due to excessive wear, abrasions, or friction be distinguished? Can a network determine that the valve coil has shorted? Some data was collected that indicates the changes in the valve's signature in such events. The figures below illustrate these changes. Figure 3.15, for example illustrates a Marotta valve with a rather weak return spring. The signature is somewhat different from that of a Marotta valve with a medium-strength spring (Figure 3.16), or that of a Marotta valve with an unusually strong spring (Figure 3.17).

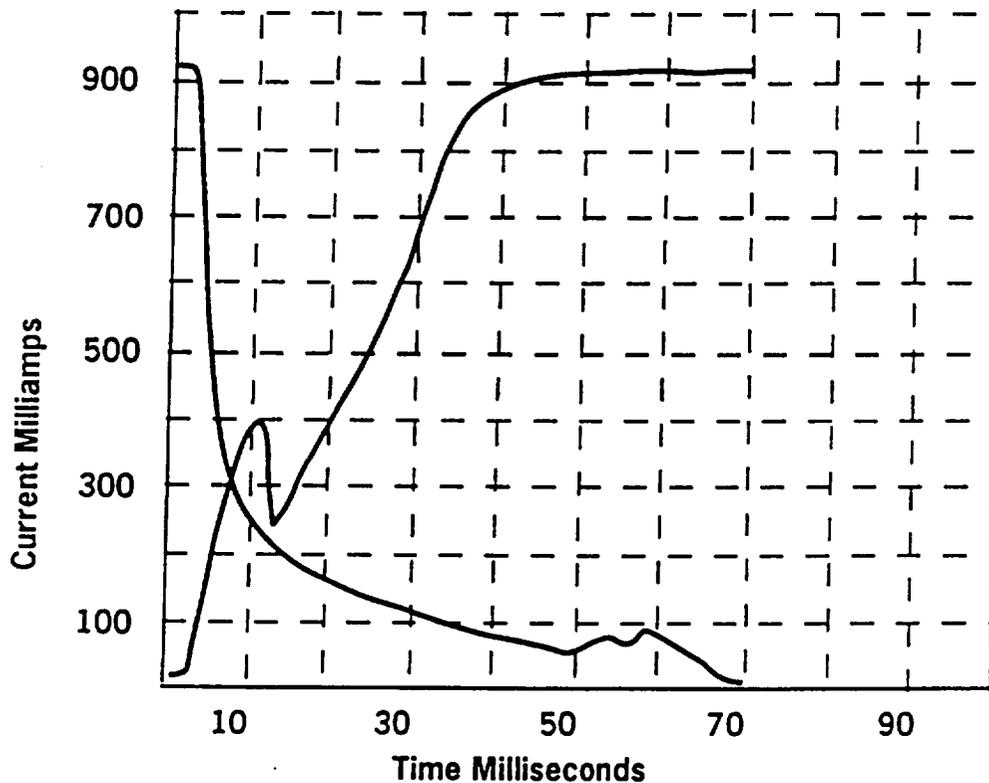


Figure 3.15. The signature of a Marotta valve with a weak spring (low spring constant). No other failures for this signature.

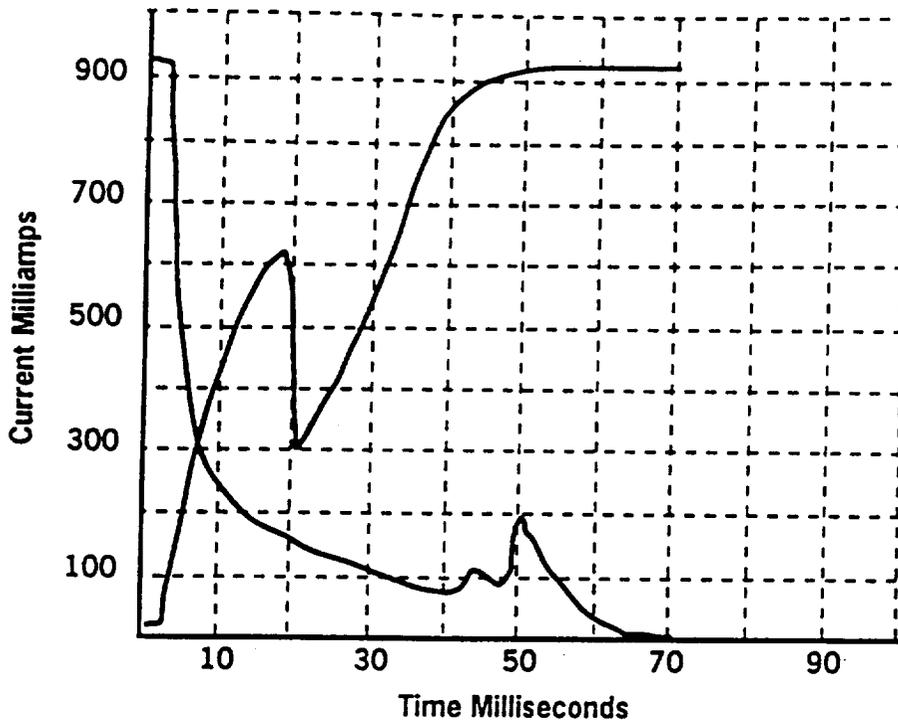


Figure 3.16. A Marotta valve signature with a medium-strength spring.

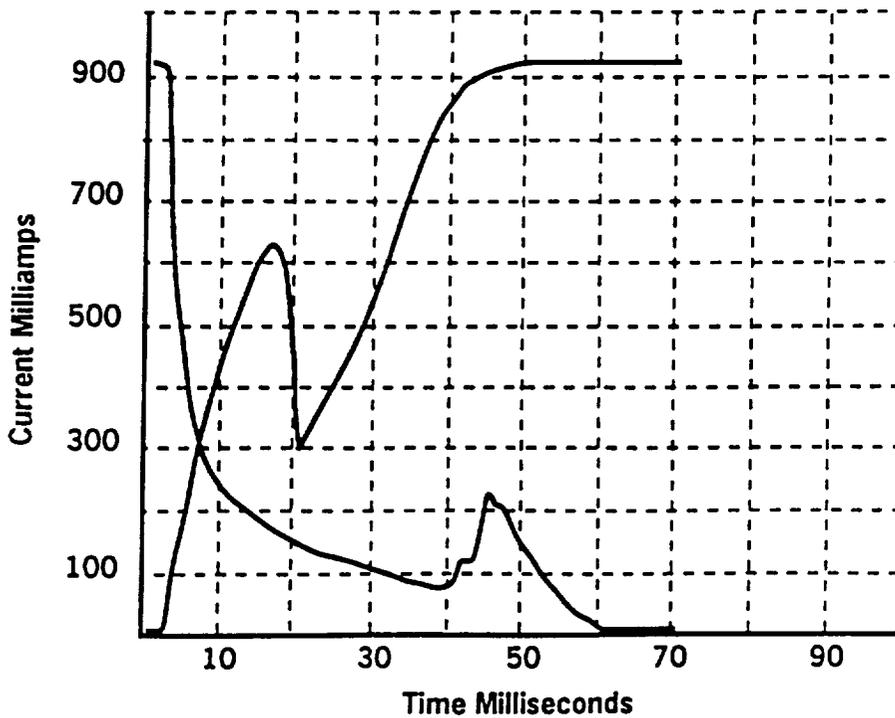


Figure 3.17. The signature of a Marotta valve with a strong spring.

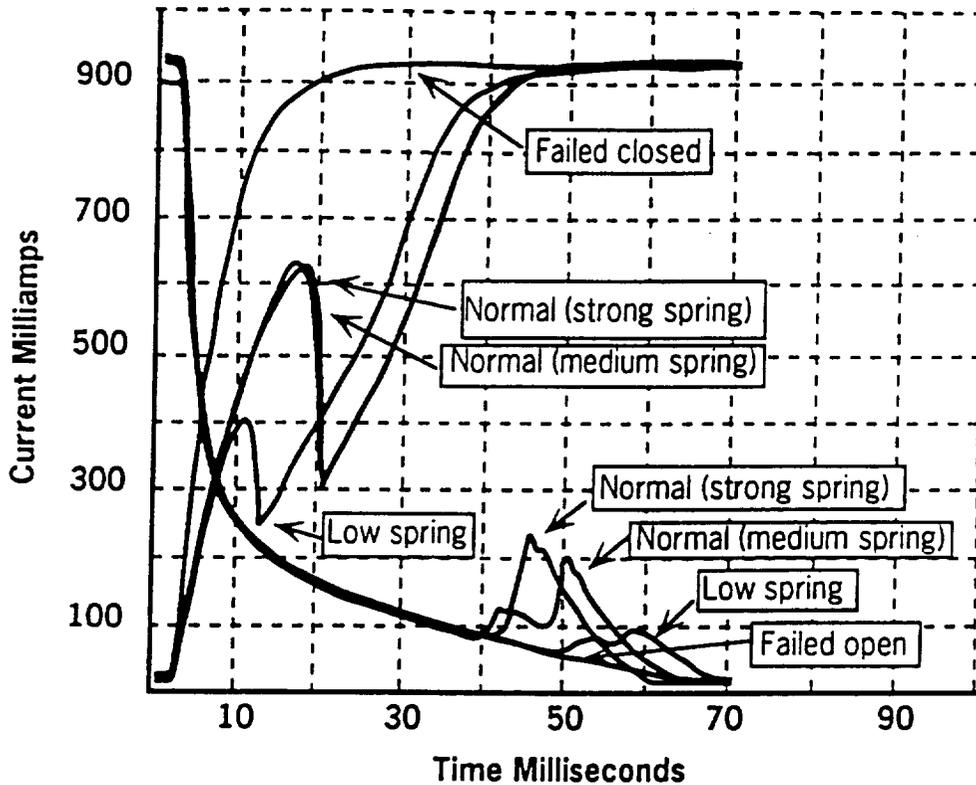


Figure 3.18. This chart summarizes the differences in signature for a Marotta valve with variations in the strength of the spring or with deliberate failures induced.

Figure 3.18 illustrates all three spring characteristics (strong, medium, and weak) for the valve-close case on a single graph to more clearly delineate the differences in the signatures. This plot also includes signatures for cases where the return spring was entirely missing (thus preventing the valve from closing properly), and when a deliberate blockage was introduced to keep the valve from opening. As can be seen in the figure, the differences in the curves are often small between individual cases (as, for example, the difference between the opening or closing of the valve with a strong vs. a medium spring). On the other hand, it may be possible to train a neural network to distinguish even these subtle differences.

Another kind of variation can be the simple differences between the adjustment of each valve. Data was collected on various settings for the adjustment screw of Circle Seal and Marotta valves. The figures below (Figures 3.19 through 3.21) illustrate how this adjustment can substantially alter the shape of the signature curves.

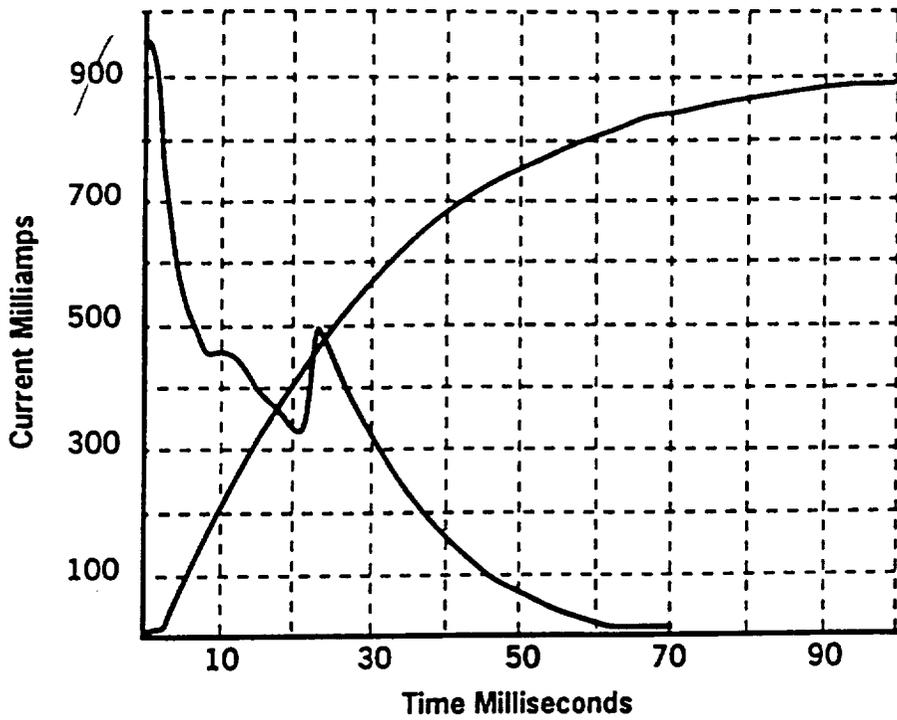


Figure 3.19. A Marotta valve with the screw adjustment tweaked has a remarkably altered signature curve.

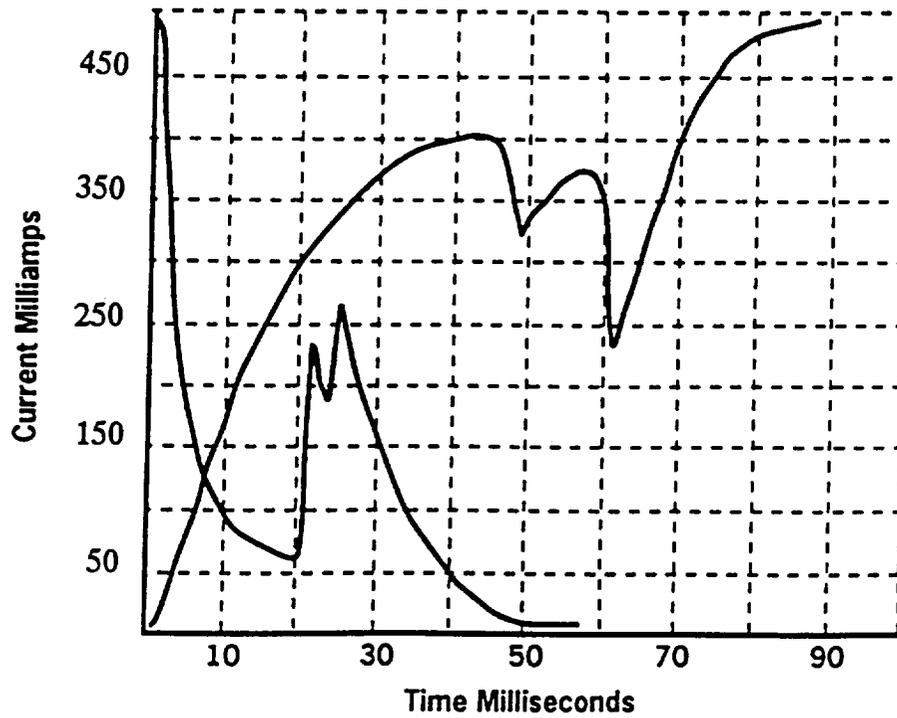


Figure 3.20. A Circle Seal valve's signature becomes quite complex when the screw adjustment is altered.

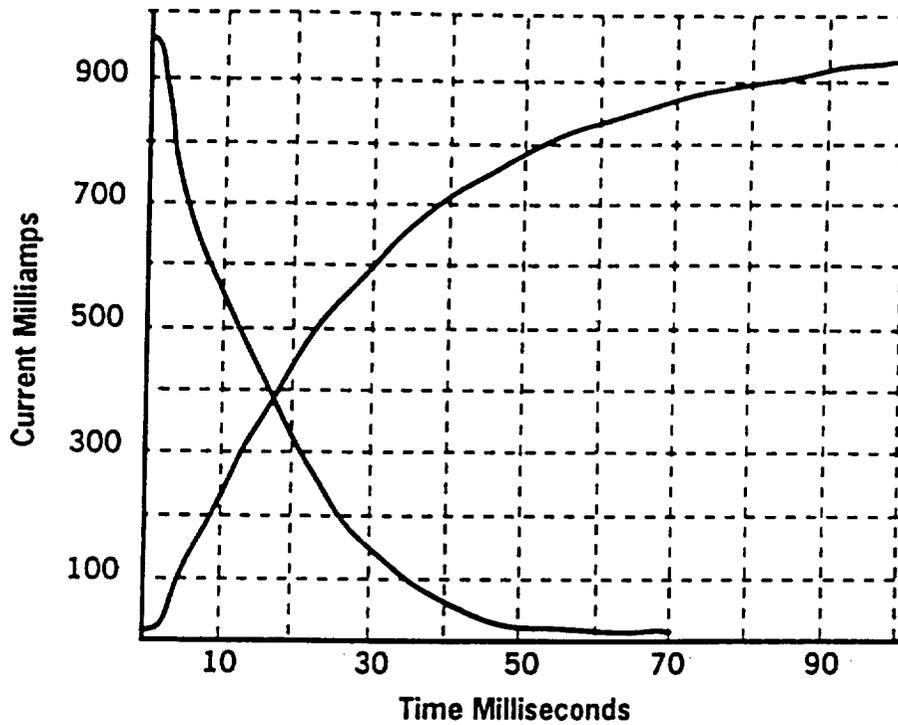


Figure 3.21. When the Circle Seal valve's screw adjustment is too extreme, the valve fails to operate. This is the resulting signature.

Figure 3.22 illustrates more generally how a Circle Seal valve's signature can change depending on the kind of adjustment or failure that it experiences. The challenge would be to train one or more neural networks to handle these cases and correctly interpret the signature data.

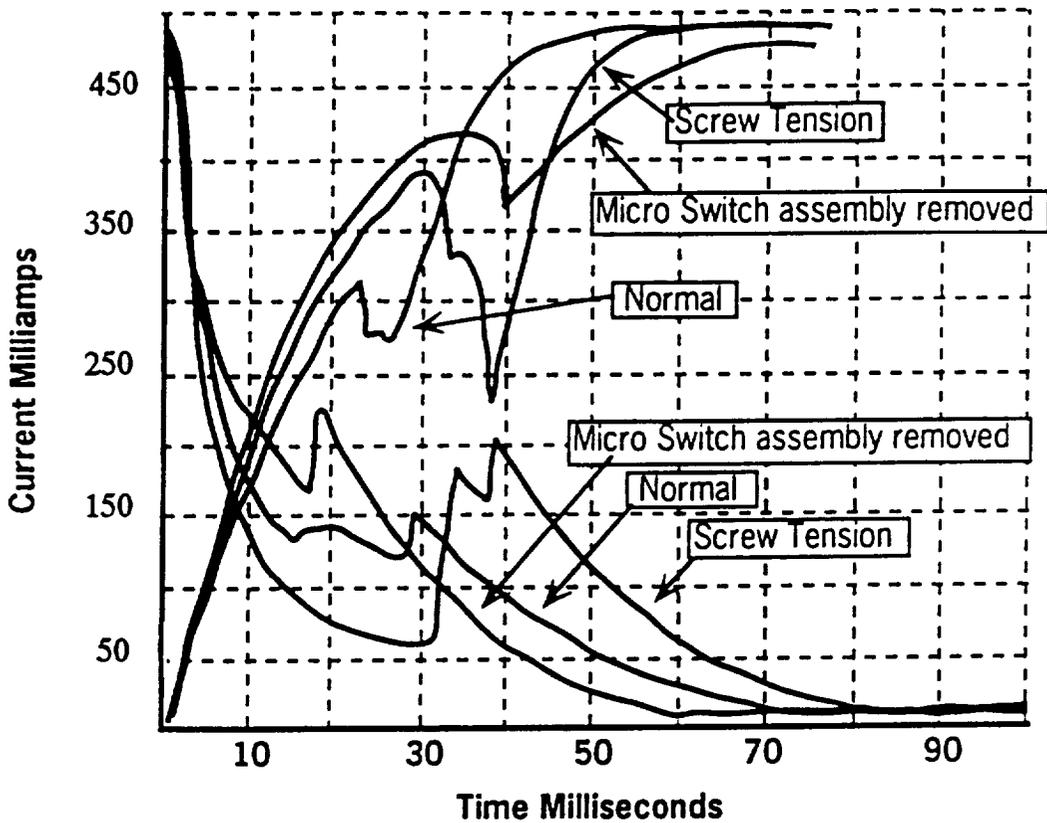


Figure 3.22. A Circle Seal valve's signature curve can change dramatically as failures or misadjustments occur.

3.5.2 Failure Prediction

Another key question that needs further investigation is that of whether a neural network could be used to actually predict valve failures by noting a trend in the signature data. No data was collected on this directly, but it was noticed that signature curves tend to slowly trend from their "cold start" position to a final "warm" position. It might be possible to train a network to notice when a valve's operation was beginning to move into unacceptable areas. This would permit reliable early warning of such failures before they actually occur and potentially provide a dramatic savings.

3.5.3 Temperature Effects

While taking data for this report it was noticed that there were some variations of a valve's signature as it warmed up. Figure 3.23 illustrates a typical case for a Marotta valve. The valve-open signature is slightly faster when the valve is cold than it is when it is warm. The valve-close signatures are all but indistinguishable. This leads to the question of whether such subtle differences can be trained for in a neural network. If so, could the network be used to help determine the temperature of the valve or the fluid flowing through it?

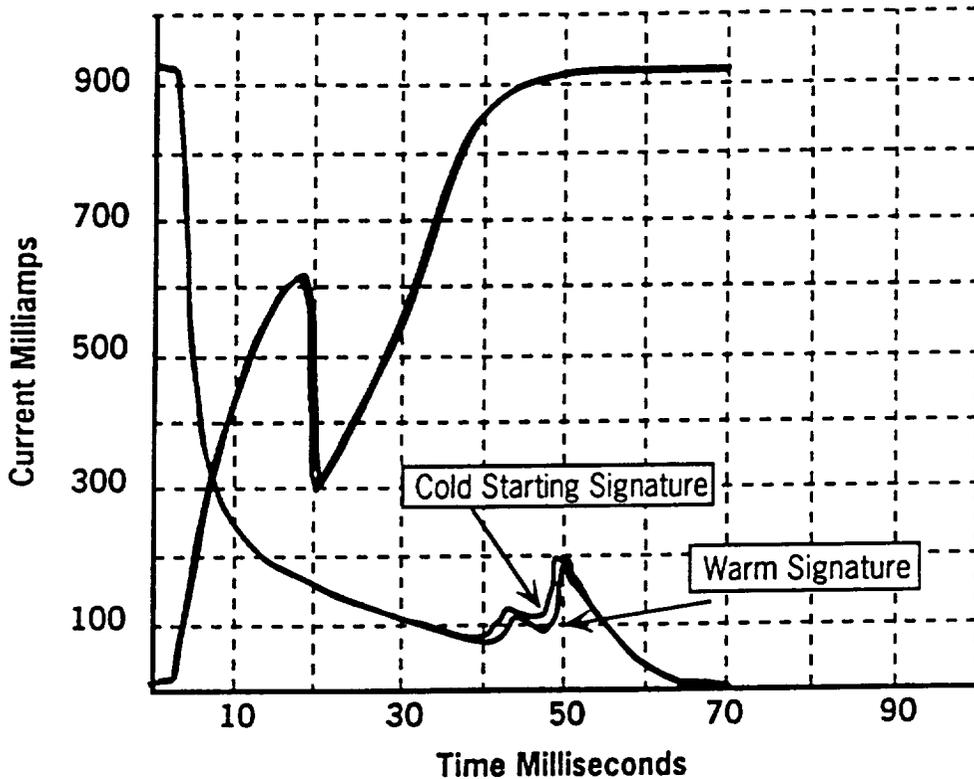


Figure 3.23. The signature of a Marotta valve varies slightly as it warms up under use. The curve slowly moves from the cold position to the warm position and then stabilizes there.

3.5.4 Pressure Effects

Yet another possible area for future research is how pressure affects the current signature of the valve. Some of the data recorded, shown in figure 3.24, shows such effects, and the possible implications are fascinating. For example, could a neural network be trained to estimate the actual pressure at the valve? If so, could this be used to confirm or refute other sensor readings? Or even replace them entirely? This offers the potential of, at a minimum, providing an independent means of confirming sensor data that does not require either additional (expensive) sensors, or additional wiring on the rocket. Since neural networks have been shown to be excellent at functional mapping applications, the likelihood is that research in this area would be very fruitful.

Other questions of great interest in this regard include: if a network could be used to determine pressure, does the fact that the line is being vented affect the current signature? What effect does having pressure on only one side of the valve (as opposed to on both sides) have on the signature? Can a network detect the difference?

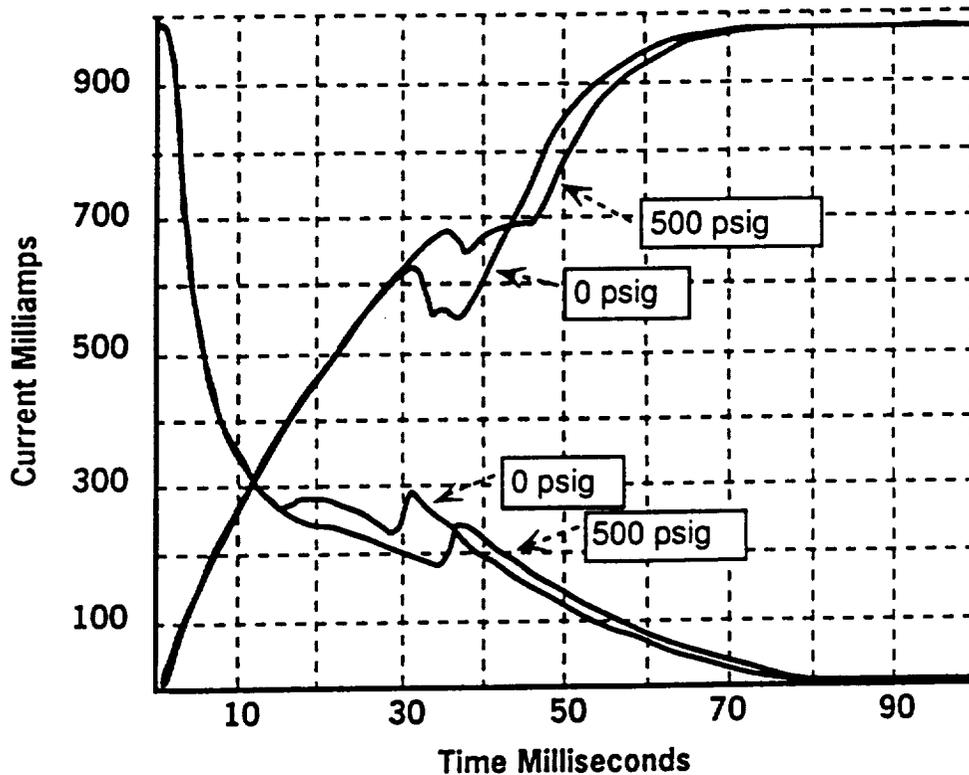


Figure 3.24. The effects of pressure at the valve are marked and highly distinguishable. It seems likely that a neural network could be trained to estimate pressure at the valve from the valve's current signature.

3.5.5 Multiple Valve Signatures

Yet another area that must be investigated is the processing of multiple valve actions that occur simultaneously. Data was recorded from a Circle Seal and a Marotta valve that were actuated (both to open and close) simultaneously. Figure 3.25 records the result of this brief experiment. The upper curve shows the mathematical sum of the individual curves of the two valves in question. the lower curve shows the actual measured result, clipped by the sensor's maximum range of 1000 ma. The two curves are virtually identical. The question, of course, is whether a neural network can be trained to understand such complex curves and correctly determine that two separate valves are acting.

Along with this question is whether a network could evaluate failures from such complex multi-valve signatures. If one of the two valves has a failure, can the network detect and identify both the fact that a failure occurred, and which valve failed?

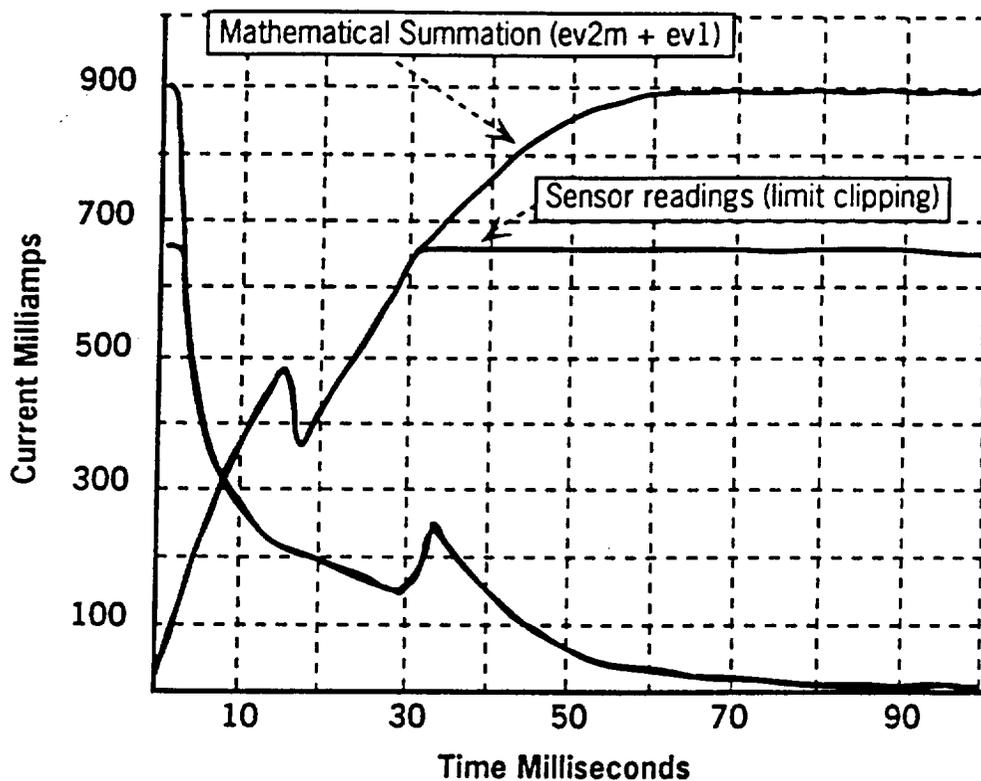


Figure 3.25. The signature of two valves acting simultaneously is the mathematical sum of the two individual signatures, clipped by the maximum range of the sensor.

3.5.6 Gravity Effects

Another question that occurred to us was what effect gravity plays in the current signature. If the valve poppet must pop up instead of down, is there a significant change in the current? Preliminary data is illustrated in figure 3.26. In this figure, a Marotta valve was operated upside down. Only slight effects were noticed (mostly in the form of a faster response) compared to rightside up operation. Still, this was only a single trial; it is not clear if this result extends to other valve types or orientations. Does a valve at a 45° angle operate differently, for example, than one vertically placed? What about a horizontal position?

Preliminary observations during testing of Circle Seal valves indicates that those valves may not be so insensitive to gravity. Of the five Circle Seal valve, one was extremely slow and had an anomalous signature reported earlier. The other four were grouped into two pairs, each pair having very similar signatures. The valves appeared to produce similar signatures based on their orientation (poppet up vs. poppet down)

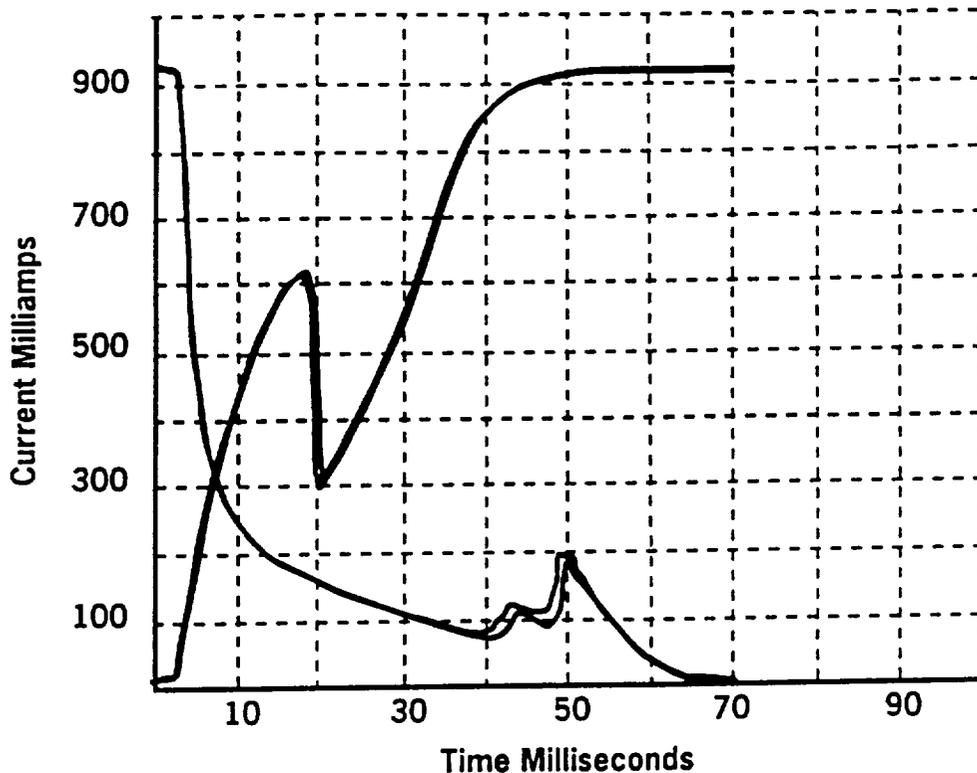


Figure 3.26. A Marotta valve operated upside down shows very little change in its signature.

3.5.7 Valve Consistency

It also appears that the valves may experience an effect similar to hysteresis in which the signature experiences a permanent change after an artificially induced failure is included and then removed from a valve. This implies that a network may have trouble over the long run in correctly identifying individual valves unless some form of on-line training can be maintained. More research needs to be done in this regard.

3.5.8 Conclusions

As can be seen from these brief notes, a great deal more work needs to be done with this project before solid conclusions about the usefulness of neural networks can be drawn. Nevertheless, it is felt that their potential as a valve-monitoring system for the ATLAS rocket has been demonstrated. While more research is needed, it seems clear that neural networks have a strong potential to both reduce the installation and equipment costs of the pneumatic system, and also to improve overall rocket reliability by reducing the reliance on fragile sensors.

SECTION 4

RF GAUGING OF CRYOGENIC FUEL TANKS USING NEURAL NETWORKS

4.1 Problem Statement

The problem addressed by the work described in this section of the report is that of gauging the quantity of subcritical cryogenic propellant in a metallic tank. The approach taken is based on measurement of the electromagnetic resonant modes of the fuel tank cavity, and has potential applications to microgravity propellant gauging.

The uniform addition of a dielectric fluid (i.e., a propellant) changes the resonant modes of the cavity of a tank in relation to the fluid density according to:

$$\frac{f_0}{f_n} = (\epsilon_n)^{1/2} \quad (1)$$

Where ϵ_n is the dielectric constant of the fluid, f_0 is the resonant frequency for the empty tank, and f_n is the resonant frequency for the non-empty tank.

As a result of this, measurement of changes in the resonant frequencies can be used to calculate the amount of mass in the tank. Specifically,

$$m = A - B \left(\frac{f_0^2 - f_n^2}{f_0^2 + 2f_n^2} \right) \quad (2)$$

where m is the mass, and A and B are constants.

Although Equation 2 provides an accurate means of measuring a uniformly distributed propellant mass, the unpredictable clumping of the liquid propellant in a gravity-free environment complicates the problem significantly under nonuniform conditions. In particular, spurious resonances arise and the measured resonances vary depending on the distribution of the cryogenic fluid and the locations of the signal sensing antennas.

Thus, the key to using the relationship expressed in Equation 2 to provide a reliable and accurate mass gauging system must be a technique which can correctly associate measured resonances under these conditions with the modes of an empty tank. The method discussed in this section of the report will that of using neural networks.

4.2 Experimental Data

Measurements were made of RF resonances in the 300 - 1000 MHz range using various antenna positions and distributions of simulated propellant masses. Wax was used to simulate the propellant because its dielectric constant is similar to that of cryogenic fluids at zero gravity. For each antenna, the resonant center frequencies, amplitudes, and qualitative measures of the line widths were recorded. The qualitative descriptions of the resonance widths as given in the left-hand column of Table 4.1 (here N stands for "narrow" and VVB for "very very broad") were quantized into the 7 numerical categories listed in the right-hand column of Table 4.1.

A total of 37 mass cases were measured, as summarized in Table 4.2. A number of the cases were missing data from one or more antennas.

Wave Code Notation	Numerical Weight
N	16.1
NB	9.3
B	5.0
BB	3.1
BBB	2.2
VB	1.8
VVB	1.0

Table 4.1. Classifications of Resonance Line Widths.

4.3. Approach 1 - Frequency Data Only

Several different approaches were taken to this problem, based on the nature of the data fed into a neural net to be trained to make estimations of mass. The first approach used only the frequencies that had been recorded as being the resonant frequencies of the filled tank. In this case, the 330 to 880 MHz band of these frequencies was divided into 1097 sub-bands of length .5 MHz each. A net was constructed with 1097 inputs, one hidden layer containing two units, and a single output unit. The inputs consisted of a string of 1097 values corresponding to the sub-bands of the 330-880 range, each value of which was either zero or one, the value being one if a frequency had been measured in the input's sub-band. The output unit was trained to be the mass corresponding to the inputs scaled so that the actual target outputs ranged from .1 to .9.

The full-scale error is defined to be the difference between the correct and predicted mass divided by the amount (mass) of the propellant in the tank when it is full, which in this case is 138. An error of 5% in this term is considered permissible. A net was trained on 31 mass cases to within an error tolerance of 1%. The trained network was then tested on five mass cases. The network had a worst case full-scale error of 7%, and an average full-scale error of 5%.

4.4 Approach 2 - Mode Averaging Network

An important aspect of the problem is the representation of the input spectral data. Two additional approaches were taken: first, the resonances, along with their amplitudes and band-widths, were presented to the network directly, as described in this section. Second, this information was converted into multi-channel spectra, as described in the next section.

	Antennas								Mass
	1	2	3	4	5	6	7	8	
1									0.0
2	spray bar								-
3				X					36.5
4				X			X	X	36.5
5				X					36.5
6									10.863
7									10.863
8	X								10.863
9									10.863
10									21.863
11									21.863
12									21.863
13									21.863
14									32.932
15	X		X	X					32.932
16									43.873
17									54.804
18									36.05
19									67.55
20									103.65
21									138.25
22									78.5
23									78.5
24									78.5
25									78.5
26				X					60.25
27									60.25
28									73.35
29									91.60
30									91.60
31									91.60
32									105.05
33	X			X	X		X	X	49.33
34	X			X	X		X	X	49.33
35	X			X	X		X	X	49.33
36	X			X	X		X	X	49.33
37	X			X	X		X	X	49.33

Table 4.2. Measured antennas and corresponding masses of 37 cases. Crosses indicate missing antenna data. Case 2 was not used.

In the direct method, a backpropagation network was presented with a sequence of resonance lines as follows:

FrequencyA1
AmplitudeA1
WidthA1

FrequencyA2
AmplitudeA2
WidthA2

FrequencyA3
AmplitudeA3
WidthA3

FrequencyB1
AmplitudeB1
WidthB1

•
•
•

FrequencyB5
AmplitudeB5
WidthB5

The data from the antennas was broken into two frequency bands according to the first two modes of the cavity (300 to 500 MHz and 600 to 900 MHz, corresponding to the A's and B's in the above table, respectively). Thus, the network presented with this data will learn to infer the shift in frequency, and hence the simulated propellant mass, from an ensemble of mode frequencies. Only the sharpest two resonances were used (*N* and *NB*), and the highest amplitude data was chosen for those resonances in the two bands.

The network was first trained by treating the antenna samples without regard to antenna number. Thus, the input layer consisted of the 24 inputs described above, corresponding to the readings from a given antenna at a particular point in time. The inputs were connected to a single hidden unit as well as to the output unit. A linear output element was used. The masses were linearly scaled into the range 0 to 1 by dividing by 150.

The data was broken into three sets: a test file consisting of cases of mass 91.60, 78.50, 36.50, 21.86, and 10.86; a test file of the 5 examples of mass 49.33; and a training file consisting of the remaining cases. The performance of the network after 13,000 training passes is shown in Tables 4.3 and 4.4.

The average full-scale error of the network results shown in Table 4.3 is 8%.

Test Mass	Predicted Mass	T - P	$\frac{T - P}{T}$	$\frac{T - P}{138}$
10.9	19.8	-8.9	82%	6.4%
21.9	29.7	-7.8	36%	5.6%
36.5	40.6	-4.1	11%	3.0%
78.5	57.1	21.4	27%	15.5%
91.6	79.0	12.6	14%	9.1%

Table 4.3. Performance of mode averaging network for individual antenna presentations. Predicted masses are averages for eight antennas. (This network has 24 inputs, 1 hidden, 1 linear output, inputs connected to outputs, 13,000 passes.)

Test Mass	Predicted Mass	T - P	$\frac{T - P}{T}$	$\frac{T - P}{138}$
49.3	66.4	-17.1	-35%	-12.4%
39.3	68.0	-18.7	-38%	-13.6%
49.3	30.0	19.3	39%	14.0%
49.3	32.6	16.7	34%	12.1%
49.3	65.8	-16.5	-33%	-12.0%
Average	52.5	-3.2	-6.5%	-2.3%

Table 4.4. Performance of mode averaging network for individual antenna presentations. Predicted masses are averages for three antennas (2, 3, and 6). Network is identical to the network used in Table 4.3. The average of the examples is also recorded.

To improve the accuracy of the network, the network was next presented with ensembles of 3 antennas (i.e., 72 inputs instead of 24). The results for a combination of antennas (2, 3, and 6) are presented in Tables 4.5 and 4.6. The average full-scale error for the examples in Table 4.5 is 15%.

Because the performance of the network shown in Tables 4.3 - 4.6 degraded from the single antenna to the multiple antenna cases, the network was retrained on a different ensemble of antennas (2, 3, and 8). The results for this combination of antennas is shown in Table 4.7. Note that the second test file could not be used since data for antenna 8 was not available. The average full-scale error in Table 4.7 is 4%.

Finally, a slightly different set of test examples was extracted from the data. The results for this case, which had an average full-scale error of 3%, are shown in Table 4.8.

Test Mass	Predicted Mass	T - P	$\frac{T - P}{T}$	$\frac{T - P}{138}$
10.9	21.2	-10.3	-94%	-8%
21.9	15.0	6.9	32%	5%
36.5	97.9	-61.4	-168%	-44%
78.5	57.6	20.9	27%	15%
91.6	88.3	3.3	4%	2%

Table 4.5. Performance of mode averaging network for multiple antenna presentations. Masses are predicted from antennas 2, 3, and 6. (This network has 72 inputs, 1 hidden, 1 linear output, inputs connected to outputs, 5000 passes.) Network performance is much worse than that shown in Table 4.3.

Test Mass	Predicted Mass	T - P	$\frac{T - P}{T}$	$\frac{T - P}{138}$
49.3	46.6	2.7	5%	2.0%
39.3	64.1	-14.8	-30%	-10.7%
49.3	7.8	41.5	84%	30.1%
49.3	58.9	-9.6	-19%	7.0%
49.3	50.3	-1.0	-2%	0.7%
Average 45.5		3.8	8%	2.7%

Table 4.6. Performance of mode averaging network for multiple antenna presentations. Masses are predicted from antennas 2, 3, and 6 using network in Table 4.5.

Test Mass	Predicted Mass	T - P	$\frac{T - P}{T}$	$\frac{T - P}{138}$
10.9	11.6	-0.7	6%	0.5%
21.9	24.9	-3.0	14%	2.2%
36.5	25.7	10.8	30%	7.8%
78.5	89.0	-10.5	13%	7.6%
91.6	90.0	1.6	2%	1.2%

Table 4.7. Performance of mode averaging network for multiple antenna presentations. Masses are predicted from antennas 2, 3, and 8 using network in Table 4.5.

Test Mass	Predicted Mass	T - P	$\frac{T - P}{T}$	$\frac{T - P}{138}$
0.0	12.5	-12.5	--	-9.1%
21.9	23.8	-1.9	-8.7%	-1.4%
36.5	38.7	-2.2	-6.0%	-1.6%
60.2	61.4	-1.2	-2.0%	-0.9%
78.5	79.9	-1.4	-1.8%	-1.0%

Table 4.8. Performance of network in Table 4.7 with different selection of training and test sets.

4.5 Discussion of Approach 2

The results for the various antenna configurations are summarized in Table 4.9.

	Test 1	Test 2	Test 2 Combined
Single Antenna	7.9%	12.8%	2.3%
Antennas 2, 3, 6	14.8%	10.1%	2.7%
Antennas 2, 3, 8	3.8%		

Table 4.9. Average full-scale errors for antenna configurations and test cases in Tables 4.3 - 4.8. For Test 2, both average errors and the error of the average are given.

If the Test 1, Antennas 2, 3, and 6, result (14.8%) is treated as anomalous, then two trends are evident in the data. First, the inclusion of multiple antennas provides only a slightly better predictor than a single antenna does. Second, averaging multiple samples of a given mass significantly improves the prediction.

These results are interpreted in terms of the limitations of the training set and the limitations of the RF gauging method itself. First, the lack of significant improvement with the use of additional antennas may be due to the reduction in training examples by a factor of 8. In the single antenna case, all antenna data is treated as if it were multiple examples from a single antenna; thus, for a 25 case training set, $25 \times 8 = 200$ training vectors can be generated. For the multiple antenna case, on the other hand, there are only 25 examples. At the same time, the number of degrees of freedom in the network (i.e., weights) triples in proportion to the increase in the input vector. Because there are more degrees of freedom (146) in the network than there are training examples (25), it is difficult for the network to generalize from the training set.

The trend in Table 4.9 toward improved results with averaged samples suggests a limitation in the RF gauging method itself. This is because the mass distribution in the tank is arbitrary, and differing combinations can produce large variations in the spectral signature of a given total mass at a given antenna location.

Table 4.9 suggests a solution to this problem, namely that the gauging technique not rely on a single measurement to estimate enclosed mass; rather, it should produce a read-out only after a number of samples. This solution requires, however, that the mixing time of fuel be short compared to the time scale on which the amount of fuel is to be measured, as determined by the burn rate of the fuel. It is expected that this condition would be met in an actual application.

4.6 Approach 3 - Mode Assignment Network

The resonances for a given antenna were combined to form a single spectrum. In particular, the input vectors were created by binning the resonance lines into a set of 100 spectral bins covering the range from 300 to 1000 MHz. In order to preserve the resolution of the frequency measurements, the lines were artificially broadened so that half of their amplitude would be binned into adjacent bins (i.e., rectangular line profiles of width 2 bins).

While this technique retains all of the information in the resonance measurements, the dimensionality of the input vectors must be increased substantially. This is not difficult from a theoretical standpoint; however, in practice the resultant network has many more weights than training examples, making it difficult to avoid overtraining.

In an attempt to further reduce the dimensionality of the input, a principle component analysis of the 25 training vectors was carried out. In particular, this involved the creation of a matrix S , consisting of 25 masses x 8 antennas = 200 rows and 200 columns; i.e., each row of S corresponded to the 100 channel spectrum of a particular antenna (masses are not included in this analysis). Next, the covariance matrix C was constructed from S . The eigenvalues of C were calculated and ranked, and the eigenvalues were used to create a 40 x 100 matrix P , which was then used to create a reduced set of 40 element input vectors R , according to $R = S \times P$.

Using the training set R , the network was trained to predict masses from antennas 2, 3, and 6, as before.

Table 4.10 shows the performance of the network on the training and testing data as a function of the number of passes through the training set. Clearly, the network has already begun to overtrain on the first pass through the data.

Number of Passes	MAE Training	MAE Test 1	MAE Test 2
1	33	35	17
2	30	38	21
5	24	47	24
10	22	54	27
20	18	62	26
30	14	66	30
100	4	85	-

Table 4.10. Performance of mode assignment network as a function of passes through the training set. Errors are mean absolute errors. The training set consisted of 25 mass examples. Test 1 is a set of 6 different mass examples. Test 2 is a set of 5 identical mass examples.

4.7 Discussion of Approach 3

The results in Table 4.10 indicate that the network overtrained in a single pass; therefore, this approach will not work for such a limited training set. The surprising ease with which the network overtrained is probably a consequence of the efficient representation of the data following the principle component analysis.

4.8 Conclusions

A backpropagation network is capable of learning to estimate masses from input spectral resonance measurements. Because of the high-dimensionality of the input space, effective solution of the problem requires finding a compact representation of the input space, as was done in the first approach, or requires a much larger training set. The networks that have been constructed have performed very close to the required error tolerance. It would be valuable to have a larger data set for training and testing to obtain a clearer understanding of their expected performance in a real setting.

Another possibility would be to implement the mapping with a nearest neighbor classifying network, such as counterpropagation. CPN can train on limited data, as well as perform more consistently in predicting intermediate values.

SECTION 5

CLASSIFICATION OF INDUSTRIAL INERTIAL WELDS USING A NEURAL NETWORK

5.1 Problem Statement

The goal of the project described in this section of the report was to develop and test a neural network to determine the quality of an industrial inertial weld by examining data accumulated during the actual occurrence of the welding operation.

5.2 Data Format

Every file that was used consisted of actual industrial welding data recorded during a single weld operation. Three types of data were obtained: press, rpm, and actupset [actupset = $f(\text{press}, \text{rpm})$]. Here press is a measure of the hydraulic pressure applied while performing the welding, rpm is the revolutions per minute of a flywheel used in the process, and actupset is a measure of the length of the metal shaft being welded, which may decrease slightly during the process. A reading for each type of data was taken every one-hundredth of a second. The intersection of the press and rpm curves and the intersection of the rpm and actupset curves were targeted for special attention in the analysis as two possible critical points in the data (see Figure 5.1).

In order to determine the critical points, the data was scaled to the dimensions shown in Figure 5.1. This was done by dividing press by 100 and rpm by 10 and by multiplying actupset by 100.

The following is an example.

original data				rescaled data			
press	rpm	actupset		press	rpm	actupset	
2509.9	67.2	0.064	->	25.099	6.72	6.400	
2510.3	67.2	0.065	->	25.103	6.72	6.500	
2509.6	67.1	0.068	->	25.096	6.71	6.800	<-- crit. pt #2
2511.2	67.0	0.069	->	25.112	6.70	6.900	

As shown, the second critical point occurs as soon as the scaled actupset becomes greater than the scaled rpm. Likewise, the first critical point occurs when the scaled press surpasses the scaled rpm.

There were a few files that had invalid entries, such as garbled data, invalid data, missing data, etc. These files were discarded.

5.3 Implementation of the Neural Network

A back propagation algorithm was used to implement this neural network. All networks were constructed to give results in terms of whether a weld was of type 'x' or type 'y', where the two types reflected the quality of the weld. The networks based their examinations on characteristics of previous welds that were already known to be either type 'x' or type 'y'. Therefore, each network constructed learned to recognize patterns for type 'x' and type 'y' welds and then compared each weld in a test set to these patterns. The results for a given weld were thus based on how well the weld fitted into the learned patterns.

A first attempt was made to train a neural network on one second's worth of data centered around each critical point. There was a limit to the number of input neurons available in the network, so every third or fourth line of data was used. The network showed no signs of converging for this training method. While training on welds of known quality, it never recognized anything in the data to categorize a type 'y' weld. The network always called a weld in its training data type 'x'.

The second attempt that was made converged. In this case, the first critical point was ignored, except for one line of data where the actual intersection occurred, so that only the second critical point (intersection of rpm and actupset) was examined. The network used full resolution (every line of data) for 1.65 seconds centered around the critical point. This resulted in 167 lines of data being used around the second critical point, with three numbers per line, for a total of 501 input neurons. This meant there had to be at least 83 lines of data before and after the line of data that depicted the point. There were a few files that had less than 83 lines of data after the point. These files were discarded because they were unsuitable for our neural network. The net structure called for two layers of 65 hidden neurons, and two output neurons, one for type 'x' and one for type 'y'.

There were 51 usable files of data that were taken from welds of known quality. For each attempt, ten of these files were excluded from the training set, so that they could be used for testing purposes after the network was trained on the remaining 41 files. A training tolerance of 0.1 was employed for each network. That is, the network, to be considered trained, was required to correctly predict the pattern as type 'x' or type 'y' with type 'x' = 1 and type 'y' = 0 within a tolerance of 0.1 (the network employed a scale from 0 to 1).

As mentioned above, the first attempt never converged due to the choice of the data sampling interval that was made, and was therefore abandoned. The second and subsequent attempts converged on the 41 training files, being able to correctly classify them all within a tolerance of 0.1. After these networks were trained in this way, they were tested utilizing the ten reserved data files and the trained network.

5.4 Testing the Neural Network

The trained network resulting from the convergent training of the second attempt described above was tested on the 10 files that were kept out of the training set. With a testing tolerance of 0.4, the network got nine out of ten correct. The file that was missed, ANNETDA02\P1\D3\F810340.F10, had data from a type 'x' weld, but the network showed results of 50% type 'x' and 75% type 'y', indicating that it had characteristics of both a type 'x' and type 'y' weld.

Using the same data format as before, investigators created two permutations of the original training and testing sets by interchanging the original testing files with some from the training set. Both sets of permuted training data were used to train a new network. All three versions were trained on 41 facts (input/desired output exemplars) and tested on 10 facts (two of which were type 'y'). All three test sets were different, with no single file appearing in more than one set. This allowed the generation of three totally different networks, tested on different sets.

The second network was trained with a tolerance of 0.05. When tested with a tolerance of 0.4, the network got eight out of ten of the test set correct. The first missed, file ANNETDA01\P1\D1\F810256.DAT, should have been type 'y', but the network showed 87.5% type 'x' and 25% type 'y'. The second missed file, ANNETDA02\P1\D3\F810722.F10, was a type 'x' file, but the network showed 12.5% type 'x' and 100% type 'y'.

The third network appeared to be much more successful. Like the second version, it was trained with a tolerance of 0.05. When tested with a tolerance of 0.4, the network achieved ten out of ten correct. The testing tolerance was repeatedly lowered to see exactly how good the network was in making its evaluations. The last testing tolerance at which the network was tested was 0.05, and it still evaluated all ten welds in its test set correctly.

A final neural network was trained on all 51 facts (i.e. leaving no testing facts available). This was done using a training tolerance of 0.05. Since a neural network gets better with more training facts, we expected this network to give a better performance than the previous three versions, which in fact seemed to be the case.

5.5 Results of the Neural Networks

Each of the four networks trained as described above were presented with 80 facts of unknown status. The first two networks yielded results in which most files were classified 100% type 'x' and 0% type 'y', with occasional files going down to 62.5% type 'x' and up to around 25% type 'y'. The third network showed results of 100% type 'x' and 0% type 'y' for all 80 facts. For the final network, the network trained on the complete set of 51 facts, the network fired 100% of the type 'x' output neuron and 0% of the type 'y' neuron on 79 facts. For one single fact, ANNETDA03\P2\D4\F860103.F10, both the type 'x' and the type 'y' neurons fired at 100% strength.

Later the 80 test facts were revealed to be all type 'x' files. Therefore, the final network was able to discriminate very reliably between type 'x' and type 'y' welds after training on all the 51 training facts available. The weakness of this testing set was lack of examples of 'y' data. More examples of the type 'y' welds were needed even during training to be able to create a better model of the type 'y' welds. The network, in spite of limited examples of data, was able to reliably classify the input samples from parameter sets which contained data that was not easily classifiable by a human observer. It seemed to have been keying on subtle cues in the data not immediately obvious to investigators. Further training with larger and more complete data sets would increase the reliability of the network to distinguish between type 'x' and type 'y' welds. This can be done in real time for quality control of the manufacturing process.

This work shows a real application to a practical manufacturing process. A trained network was able to reliably discriminate between type 'x' and type 'y' welds. This project demonstrates the usefulness of neural nets in difficult pattern recognition problems.

SECTION 6

RECOMMENDATIONS FOR FUTURE WORK

6.1 SSME Health Monitoring

We recommend completion of the development of the methods described above for making the SSME shutdown decision so as to enable them to be tested and then used in practice on the test-stand and in space.

The first step to be taken in this direction is the detailed analysis of and then incorporation into neural networks and other decision structures of large amounts of recent data. Data that is nominal is necessary. Data indicating minor faults in the engine, not requiring an engine shutdown on the test-stand or in space should be provided. Any recent data indicative of faults for which shuttle engineers now believe the engine should be shut down is important. The set of PIDs should also be expanded so that every major part of the engine is covered in terms of the detection of major faults. The kind of data analysis that has been developed during the course of these investigations, both at NASA and NETROLOGIC, should rapidly lead to a more complete feature set.

For reasons explained in Section 2, it is possible that the addition of more recent data to the training data sets, as well as, the values of more PIDs, and of more features calculated from raw PID values, may be adequate to handle the detection of all major SSME faults. Nevertheless, neural net fault detection when actual failure data is not available (SSMEs have not yet experienced any major failures in-flight) must be investigated. Three standard neural net approaches to this problem immediately suggest themselves for exploration, Kohonen's novelty detector, various methods for unsupervised instruction of neural nets in which the neural net itself divides its input data into categories that appear natural to it, and, the approach discussed in detail in Section 2 of this report that involves including synthetic fault data in training sets. These approaches could also be helpful when applying a network to nominal or anomalous data, whenever the amount of fault data is insufficient to guarantee that it spans the whole domain of possible fault data. Adding of features to about what is going on in the engine that was mentioned above is critical for this new investigation. The new approach to fault-typing that has been discussed in terms of the third (synthetic data) approach involved in having new output units representing the severity of deviance (weak, medium, strong, for instance) of individual PIDs, or of the whole cluster of PID values would be a natural part of these investigations.

The problem of handling failures during transients must also be addressed. In addition, methods for the first two seconds of the steady-state, while data is being gathered to go into the averages used in current features, should be developed. During this time

interval a network should work with raw PID values and determine whether a steady-state has actually been reached (case 307 appears to provide an example in which this was never the case), and also determine whether the values being taken on are within expected ranges (cf. the SAFD approach). The experience gained during the analysis of the steady-state situations will be applicable to this area.

6.2 Valve Signature Identification

To date, the valve Signature Identification Neural Network (SINN) project described in Section 3 of this report has determined the basic feasibility of our approach. Our prototype was able to identify valve types with "severe" simulated noise, and identify specific valves, within a given type, with "moderate" simulated noise. These investigations now need to be expanded to implement and integrate this neural network system into the actual Atlas vehicle environment.

The first thing required is to verify the data accuracy. Repeatable results are crucial to the performance of the system in the noisy environment of the vehicle. Part of this process should be studied in the laboratory using the Atlas Fluids System Integration Lab (SIL) and the Parallel Equipment Module (PEM) at General Dynamics to accurately provide the vehicle environment. Tests must include faults, pressure variations, multiple events interference, and electromagnetic interference effects. The next part of this process is to record this similar detailed data during an actual flight. This data would record engine noise interference and acceleration effects on the signature characteristics of the valves. Such an opportunity is available on the ALS Program Boost Recovery Module (BRM) Atlas mission.

The second task would develop a spatial-temporal network. This would allow continuous streaming of data to be sifted for the target signal characteristics. Once a signal was detected, it would be buffered to the evaluation neural networks for valve identification.

The fourth task would develop multiple neural networks to perform in parallel to identify the signature as to type and if possible the specific valve. These neural networks would operate collectively to enhance reliability and the degree of confidence in the analysis.

The fifth and final task would integrate the SINN system with an expert system designed to fuse the observed results with the on-board flight computer commands and thus close the feedback loop.

6.3 Cryogenic Fuel Tank Mass Gauging

There was an inadequate amount of data to determine whether neural networks could estimate fuel masses better than other methods. The neural network methods did perform comparably to existing methods. Since the results were promising and the problem is an important one, the understanding gained from these early investigations should be developed further with a larger data set.

6.4 Industrial Weld Quality Gauging

The work on the classification of industrial inertial welds described in Section 5 of this report shows great promise for the use of neural nets in the area of industrial quality control. We recommend that the methods of the investigations reported here be implemented in a situation that would test the process over a long period of time, and that the same techniques tried with other examples of industrial process and quality control. Some of the techniques discussed in Section 6.1 and in the body of Section 2 concerning training neural networks when the number of examples in one of the categories requiring classification is very limited could also be of value in many instances here (for instance, very few inertial welds are of inferior quality).

SECTION 7

BIBLIOGRAPHY

- [1] Cikanek, Harry A.: "Space Shuttle Main Engine Monitoring Experience And Advanced Monitoring Systems Development."
- [2] Cikanek, Harry A.: "SSME Failure Detection." Proceedings of the American Control Conference, June 19-21, 1985, NASA MSFC.
- [3] Dietz, Kiech, and Ali: "Neural Network Models Applied to Real-Time Fault Diagnosis." Journal of Neural Network Computing, Vol. 1 No. 1, 1989.
- [4] Kelley, Glover, Teeter, and Tischer: "Diagnostic Needs of the Space Shuttle Main Engine." SAE Technical Paper Series, Oct. 15-18, 1984.
- [5] Kerr, T.H.: "The Controversy Over Use of SPRT and GLR Techniques and Other Loose-Ends in Failure Detection." American Control Conference, Proceedings, Vol. 3, pp.966-977, 1983.
- [6] McClelland, J.L., and Rumelhart, D.E., "Learning Internal Representations by Error Propagation," pp. 318-364, in *Parallel Distributed Processing*, Vol. I., The MIT Press, Cambridge, MA, 1987.
- [7] Mehra and Peschon: "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems." Automatica, Vol.7, pp. 637-640, 1971.
- [8] Taniguchi, M.H.: "Literature Review Results." Failure Control Techniques For The SSME, Rocketdyne, Phase I report. April, 1987.
- [9] Van Leuven, K., "RF Modal Quantity Gaging", Proceedings 1989 JANNAF Propulsion Meeting, May 23-25, Vol. 1, Publication No. 515, CPIA, pp. 471-478.
- [10] Willsky, Alan S.: " A Survey of Design Methods For Failure Detection in Dynamic Systems." Automatica, Vol. 12, pp. 601-611, 1976.
- [11] Whitehead, Bruce A., Ferber II, Harry J, and Moonis, Ali: "Neural Network Approach to Space Shuttle Main Engine Health Monitoring." 26th AIAA Joint Propulsion Conference, Paper #AIAA-90-2259

- [12] Whitehead, Bruce A., Kiech, Earl L. and Moonis, Ali: "Rocket Engine Diagnostics using Neural Networks." 26th AIAA Joint Propulsion Conference, Paper #AIAA-90-1892

APPENDIX A

SSME GRAPHS

This section contains examples of different types of graphs that have been used during investigations of the SSME shutdown determination problem. The following kinds of graphs are given:

- A.1 Raw PID Values

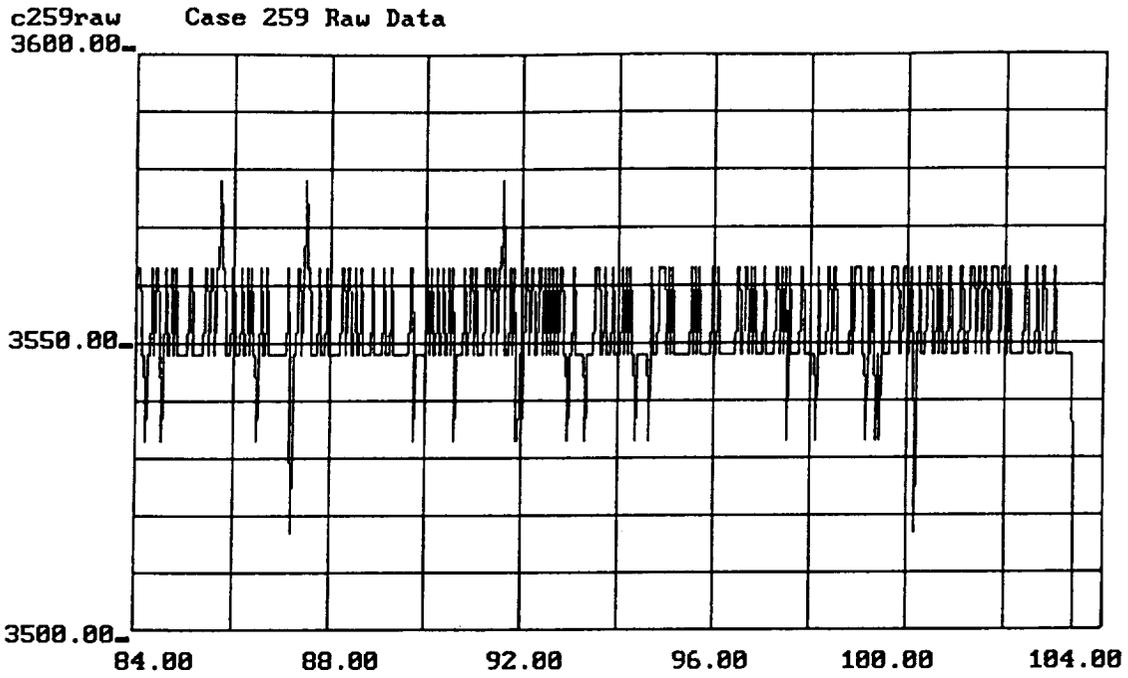
- A.2 Feature Values

- A.3 Neural Net Output Units for Eleven Holdout Cases

- A.4 Neural Network Output for the last time slices of Cases 457
 and 364 with PID 207 Features Replaced with Zeros

- A.5 Example of Euclidean Distance Fault Indication

On all the graphs in this section, the numbers along the bottom of the graph give the time in sections from the beginning of the firing.



-2-

Figure A.1.1 Graph of Raw Data Values, Case 259, PID 24

Main Combustion Chamber Hot Gas Injector Pressure A

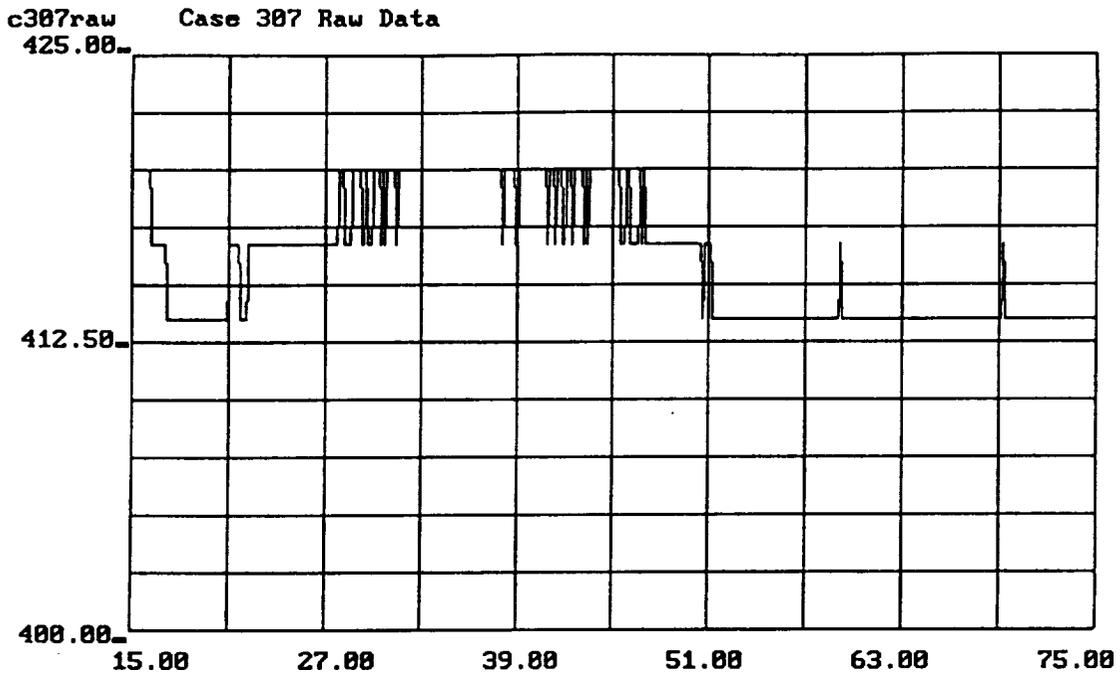


Figure A.1.2 Graph of Raw Data Values, Case 307, PID 18
Main Combustion Chamber Coolant Discharge Temperature B

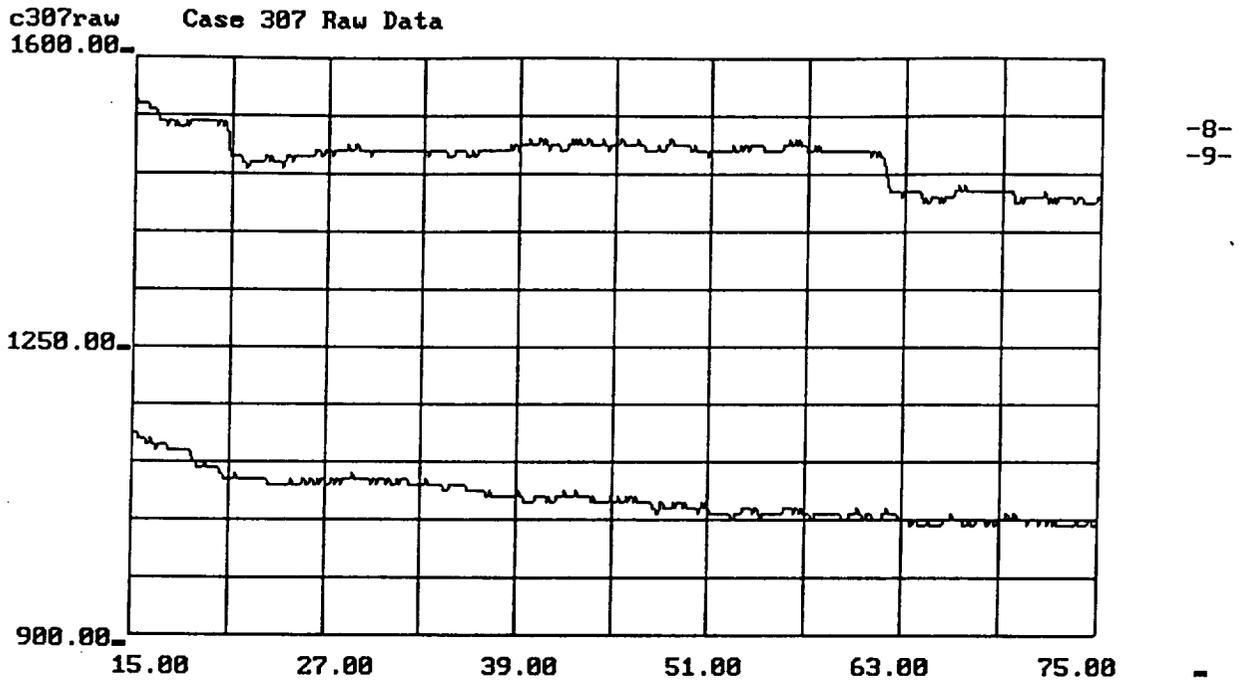
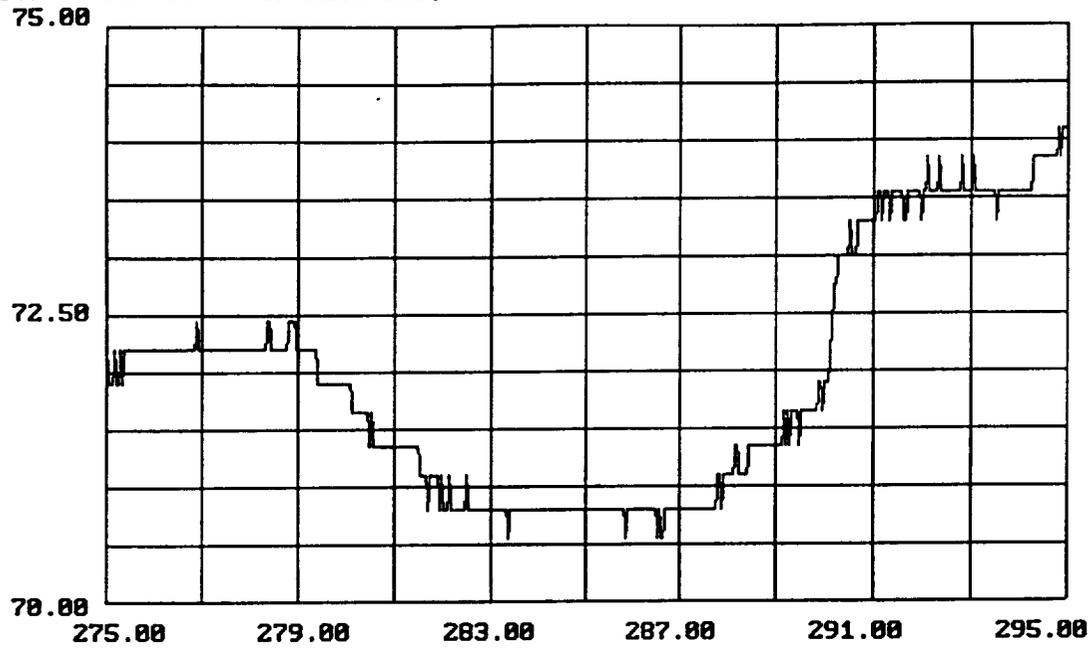


Figure A.1.3 Graph of Raw Data Values, Case 307, PID 231 and PID 232

High Pressure Fuel Turbine Discharge Temperatures A and B

(Correlated PID values)

c340 Raw Data for case 340, 12 PIDS



8

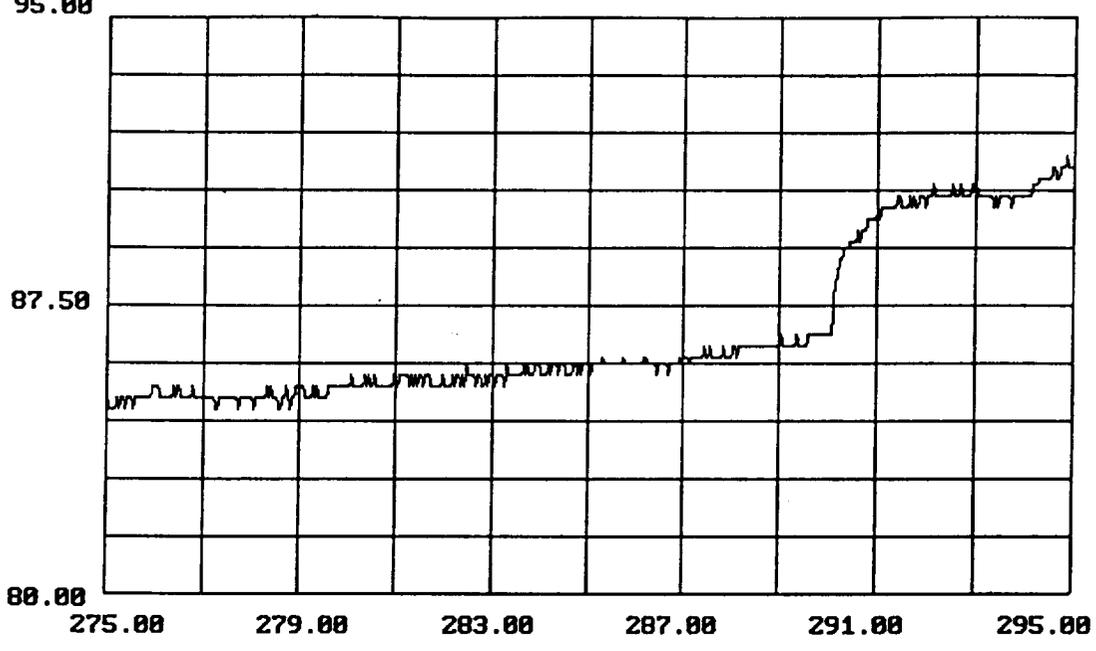
-3-

Figure A.1.4 Graph of Raw Data Values, Case 340, PID 40

Oxidizer-Preburner Oxidizer Valve Actuator Position A

c340 Raw Data for case 340, 12 PIDS

8

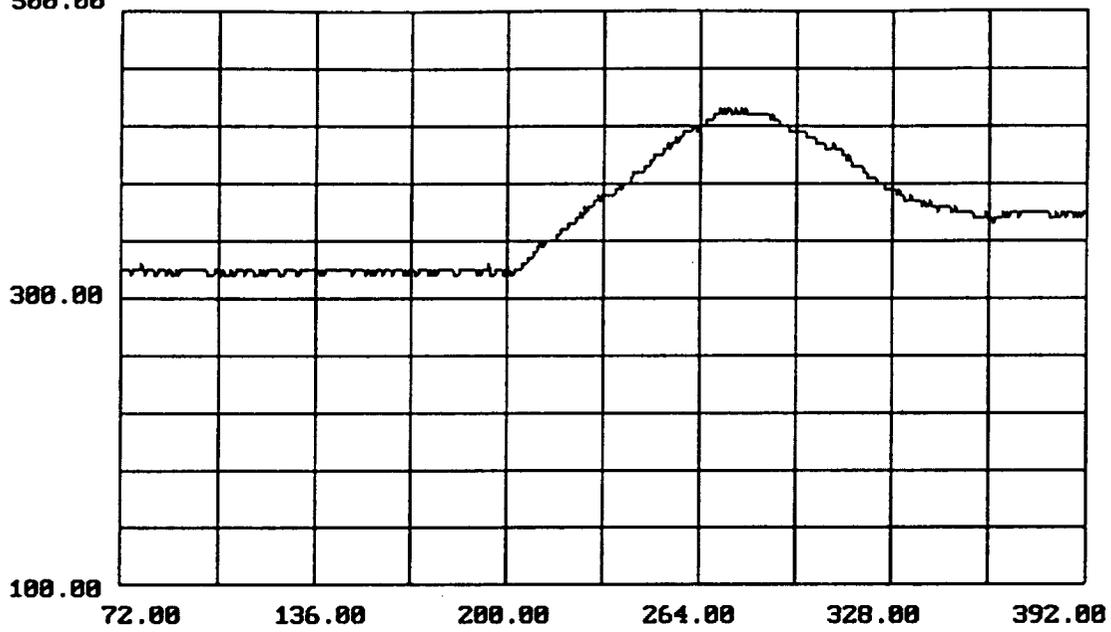


-4-

Figure A.1.5 Graph of Raw Data Values, Case 340, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A

c364 Raw Data for case 364, 12 PIDS

0

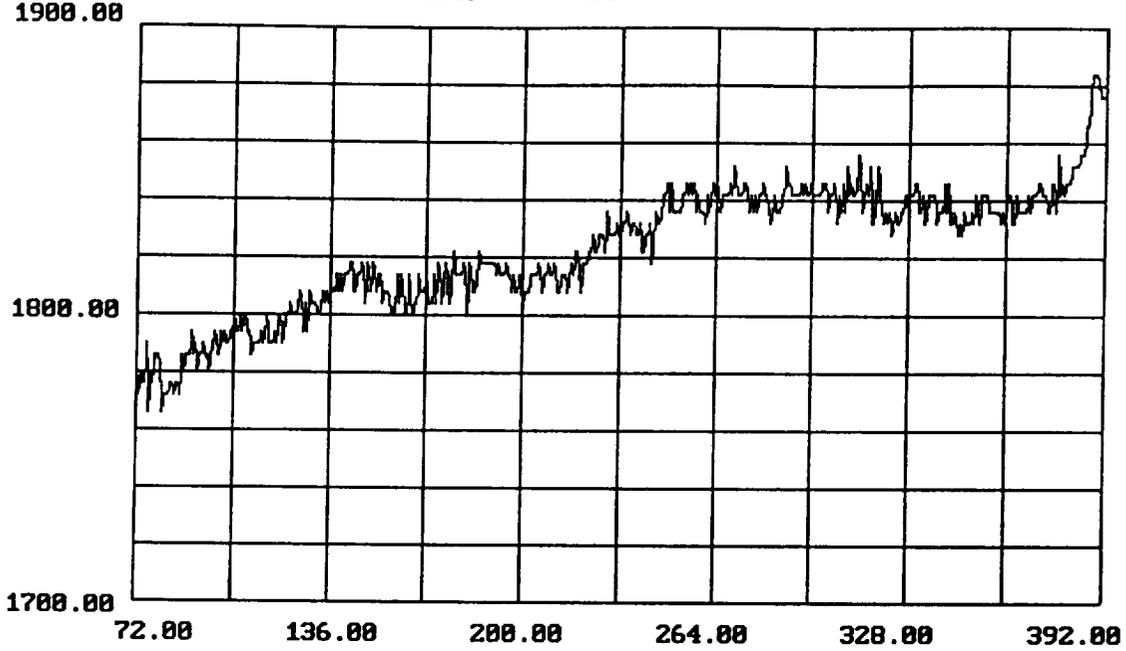


-7-

Figure A.1.6 Graph of Raw Data Values, Case 364, PID 209

Low Pressure Oxidizer Pump Discharge Pressure A

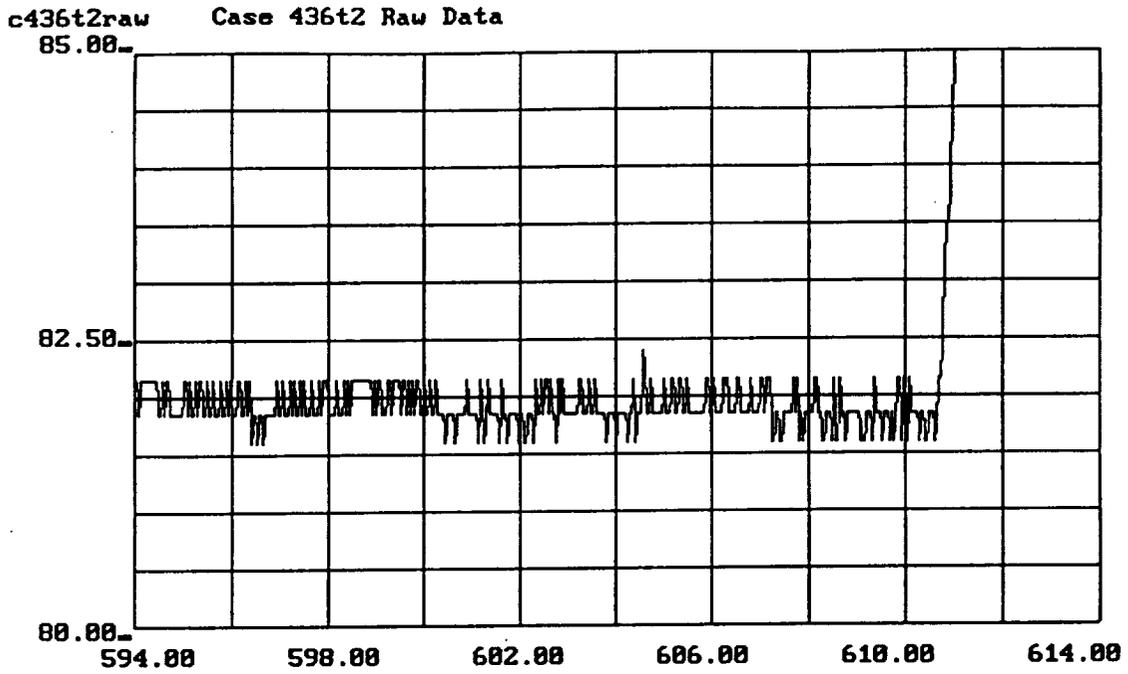
c364 Raw Data for case 364, 12 PIDS



0
-8-

Figure A.1.7 Graph of Raw Data Values, Case 364, PID 231

High Pressure Fuel Turbine Discharge Temperature A



-4-

Figure A.1.8 Graph of Raw Data Values, Case 436, PID 42

Fuel Preburner Oxidizer Valve Actuator Position A

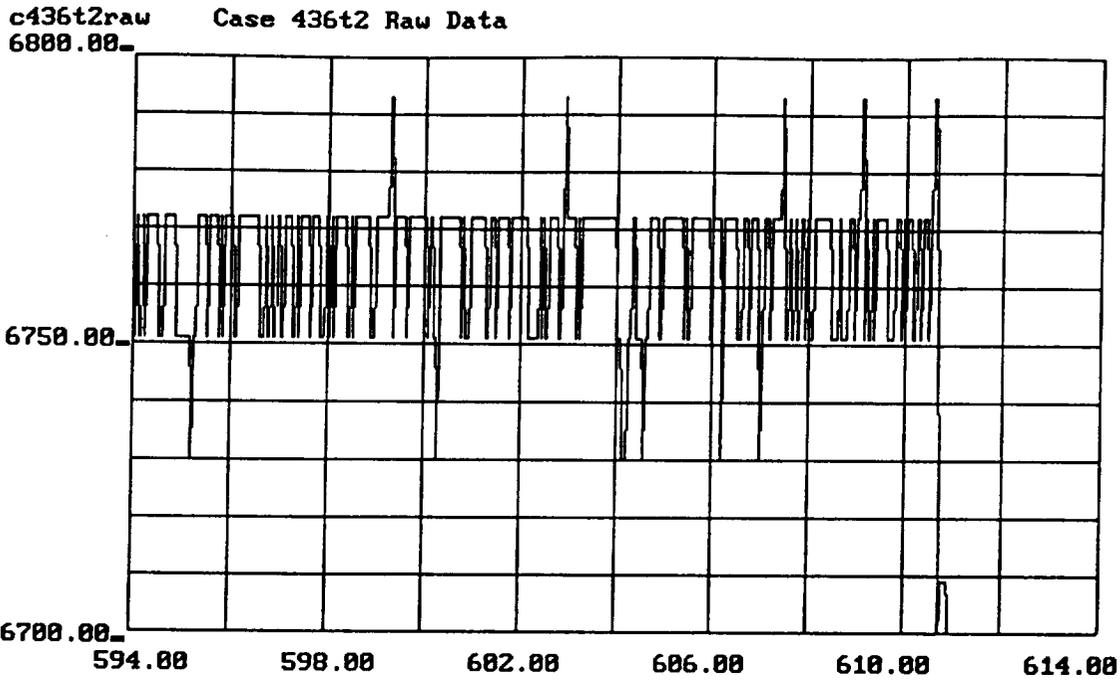
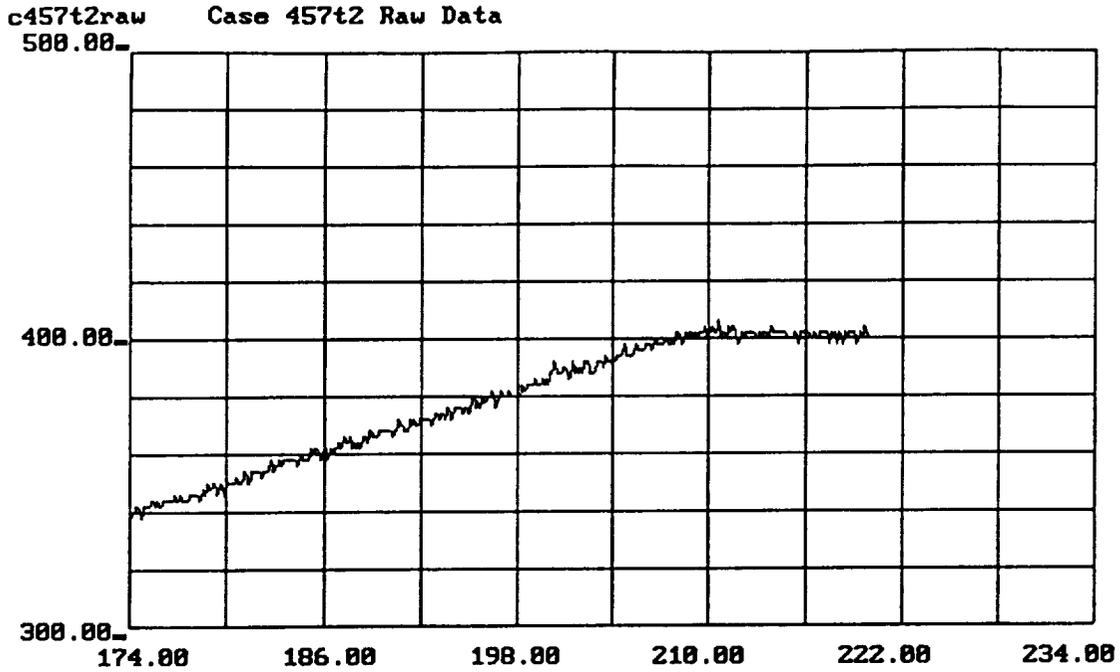


Figure A.1.9 Graph of Raw Data Values, Case 436, PID 52

High Pressure Fuel Pump Discharge Pressure A



-7-

Figure A.1.10 Graph of Raw Data Values, Case 457, PID 209

Low Pressure Oxidizer Pump Discharge Pressure A

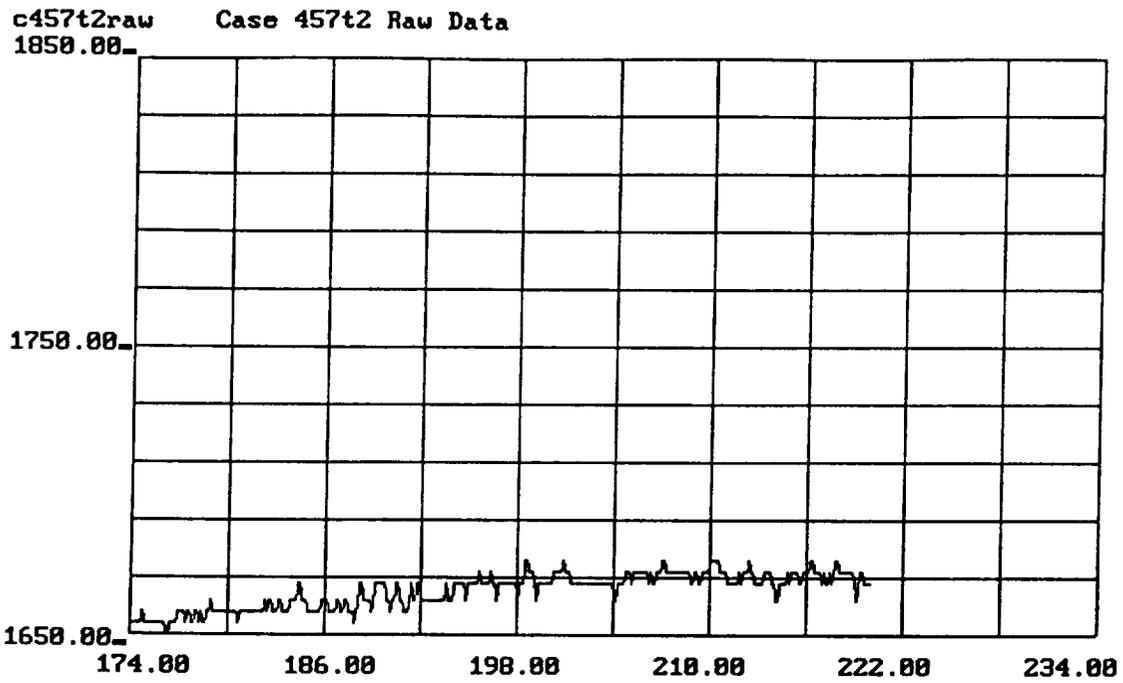
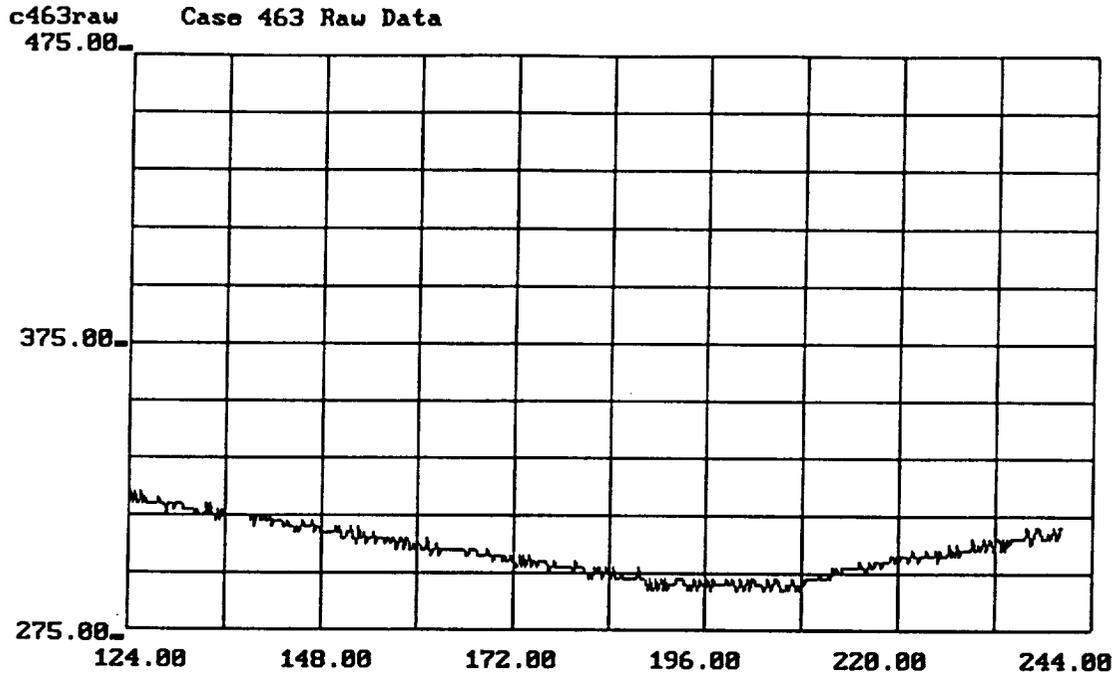


Figure A.1.11 Graph of Raw Data Values, Case 457, PID 231

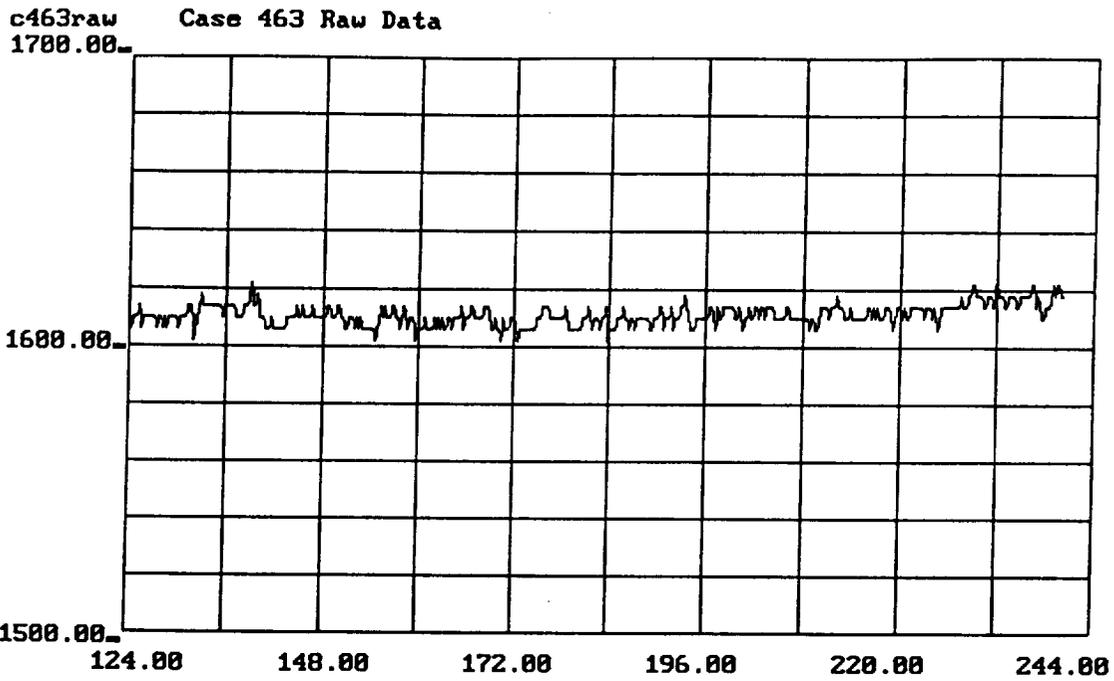
High Pressure Fuel Turbine Discharge Temperature A



-7-

Figure A.1.12 Graph of Raw Data Values, Case 463, PID 209

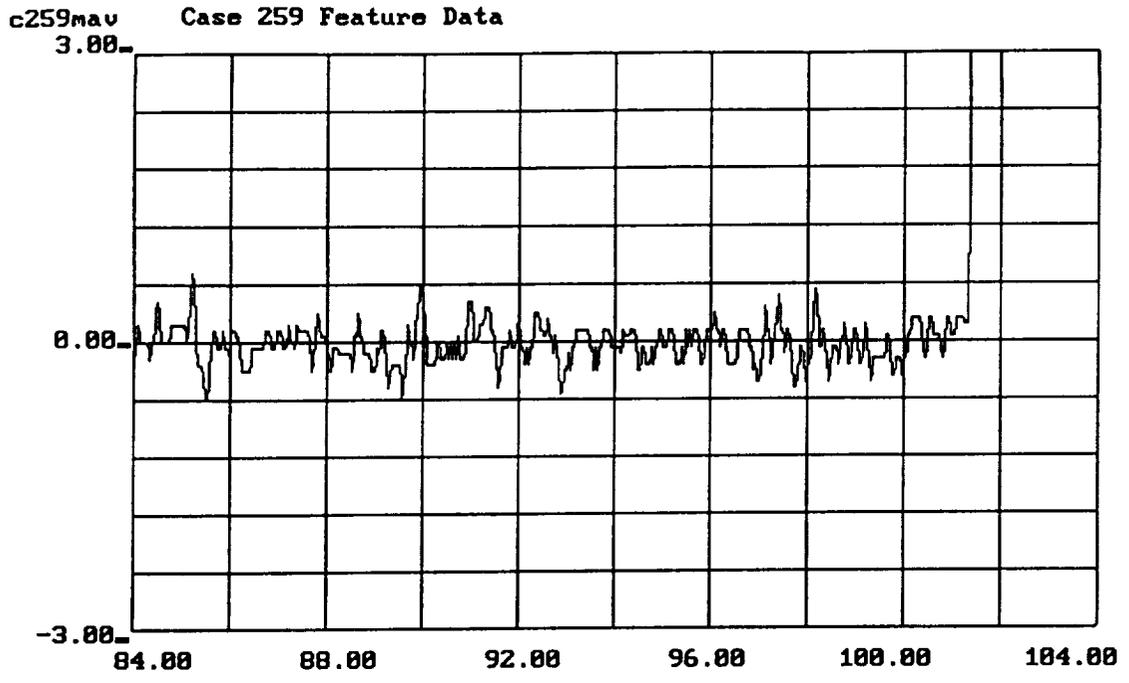
Low Pressure Oxidizer Pump Discharge Pressure A



-8-

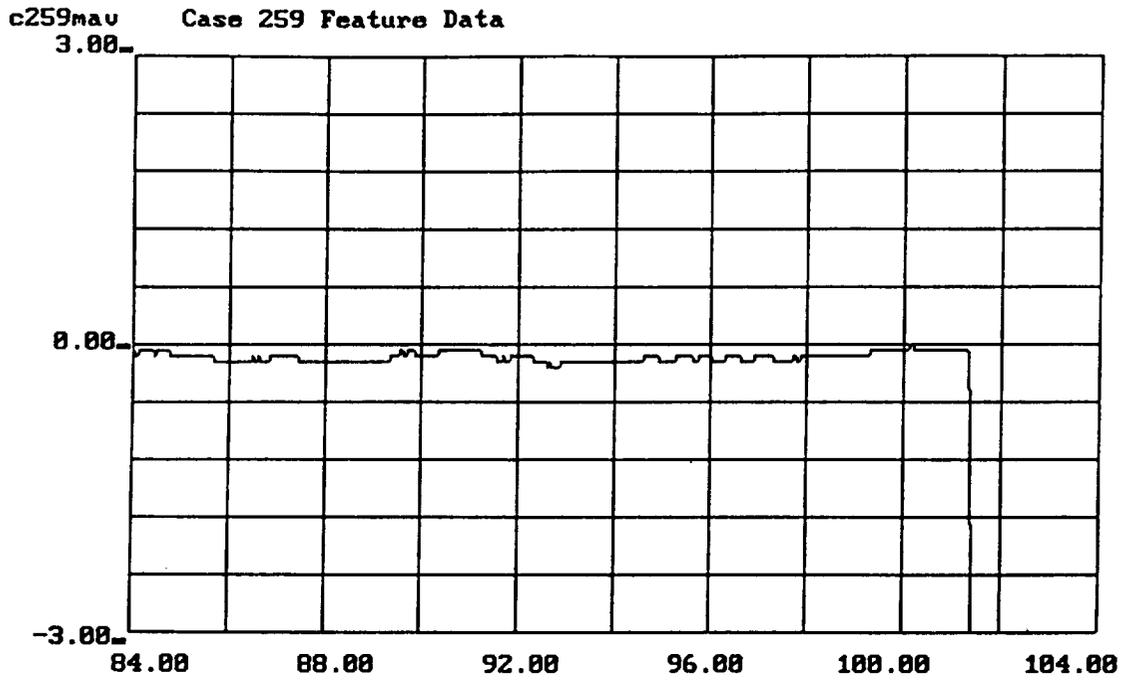
Figure A.1.13 Graph of Raw Data Values, Case 463, PID 231

High Pressure Fuel Turbine Discharge Temperature A



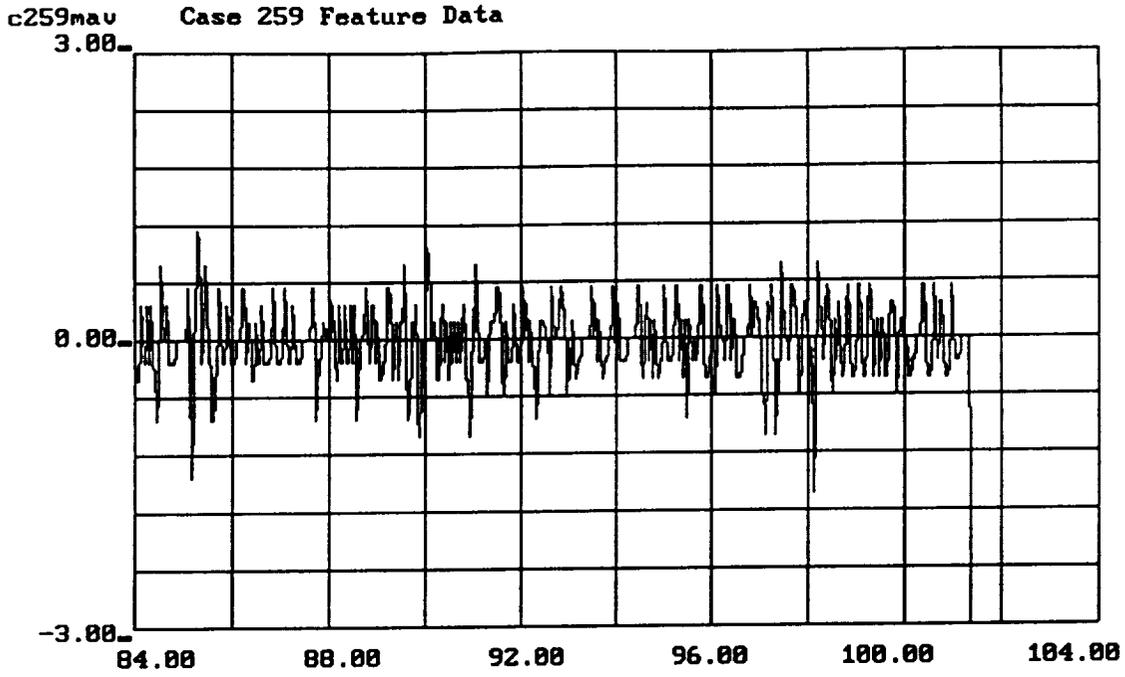
-6-

Figure A.2.1 Graph of Feature 1 Values, Case 259, PID 24
Main Combustion Chamber Hot Gas Injector Pressure A



-7-

Figure A.2.2 Graph of Feature 2 Values, Case 259, PID 24
Main Combustion Chamber Hot Gas Injector Pressure A



-8-

Figure A.2.3 Graph of Feature 3 Values, Case 259, PID 24

Main Combustion Chamber Hot Gas Injector Pressure A

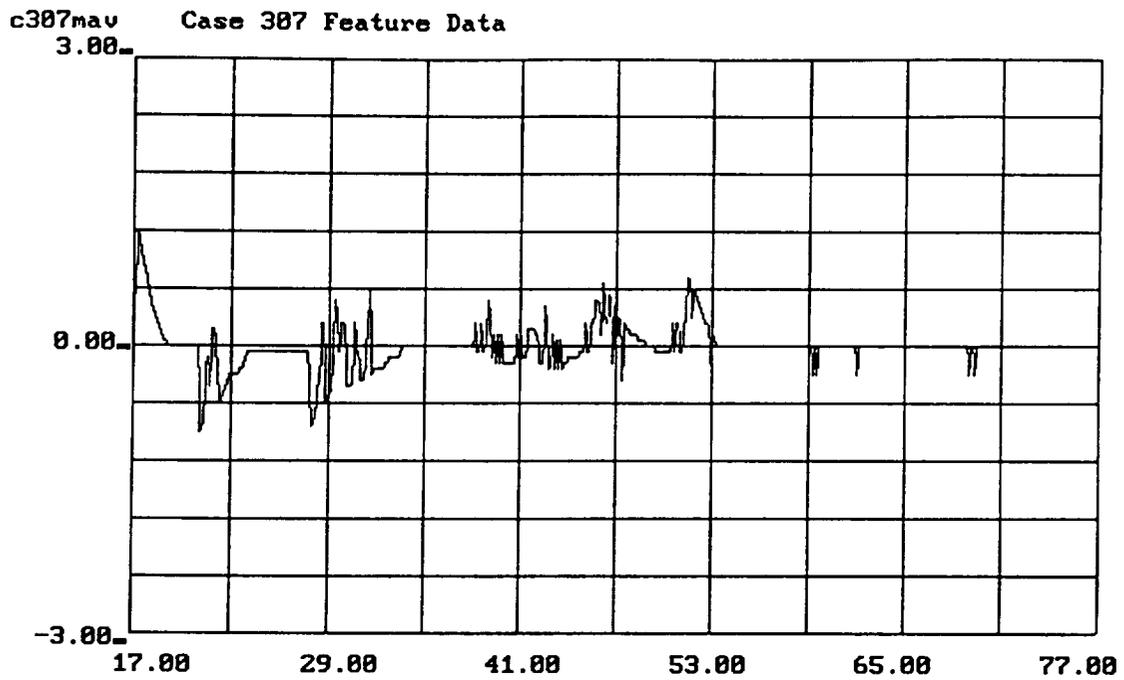
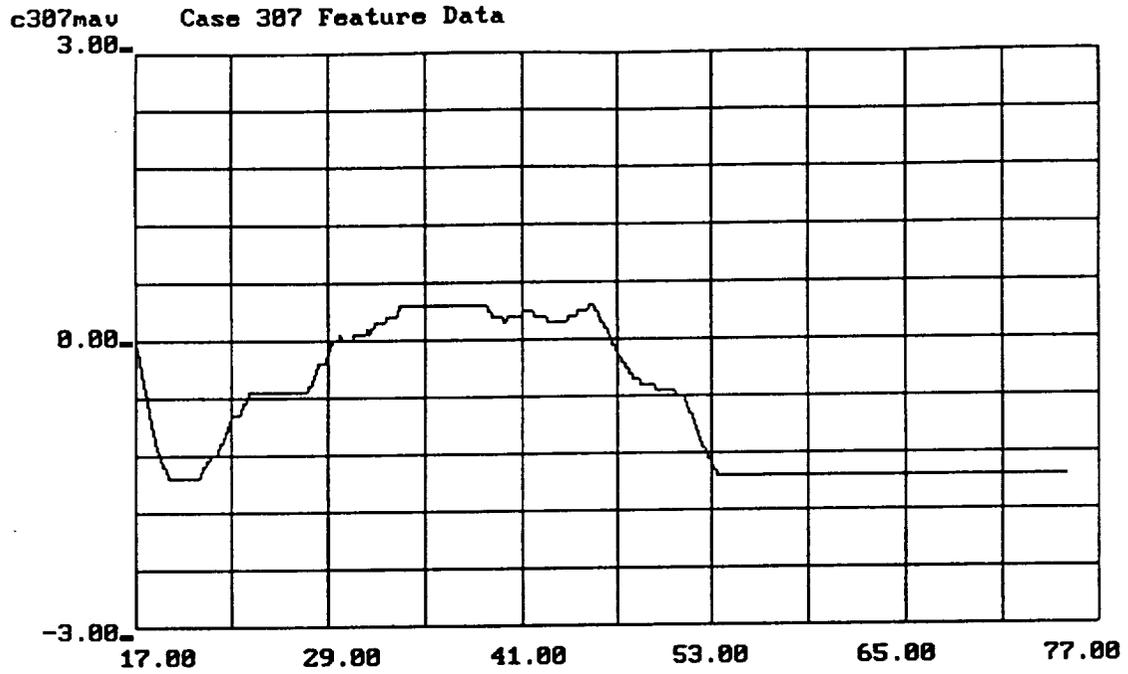
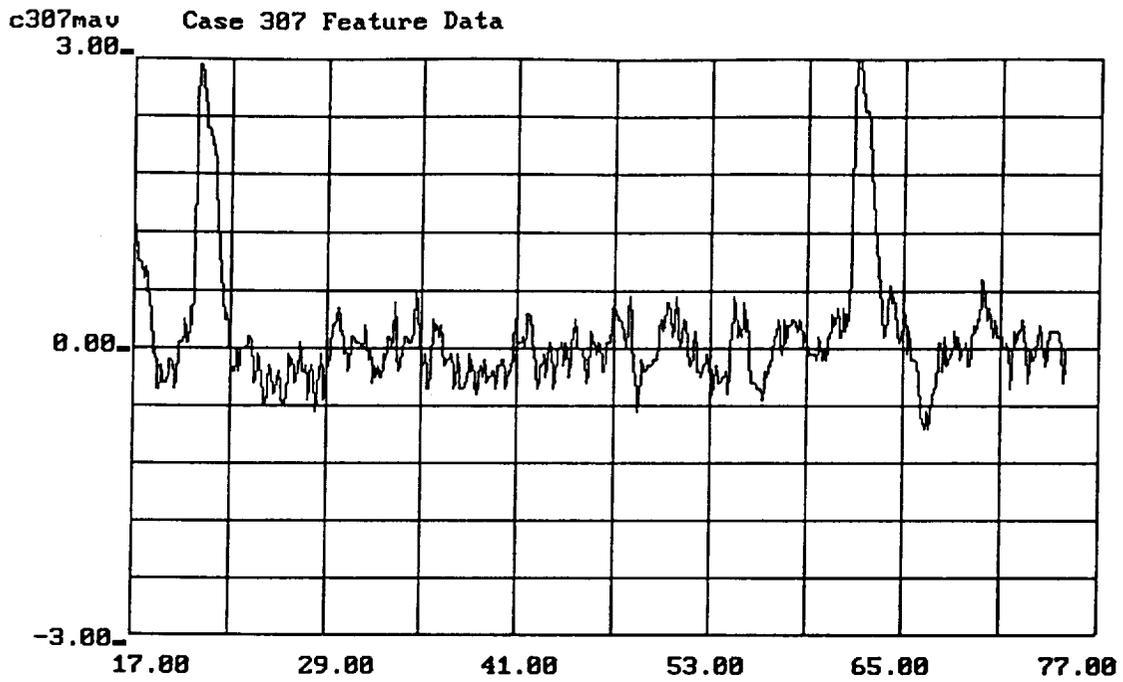


Figure A.2.4 Graph of Feature 1 Values, Case 307, PID 18
Main Combustion Chamber Coolant Discharge Temperature B



-3-

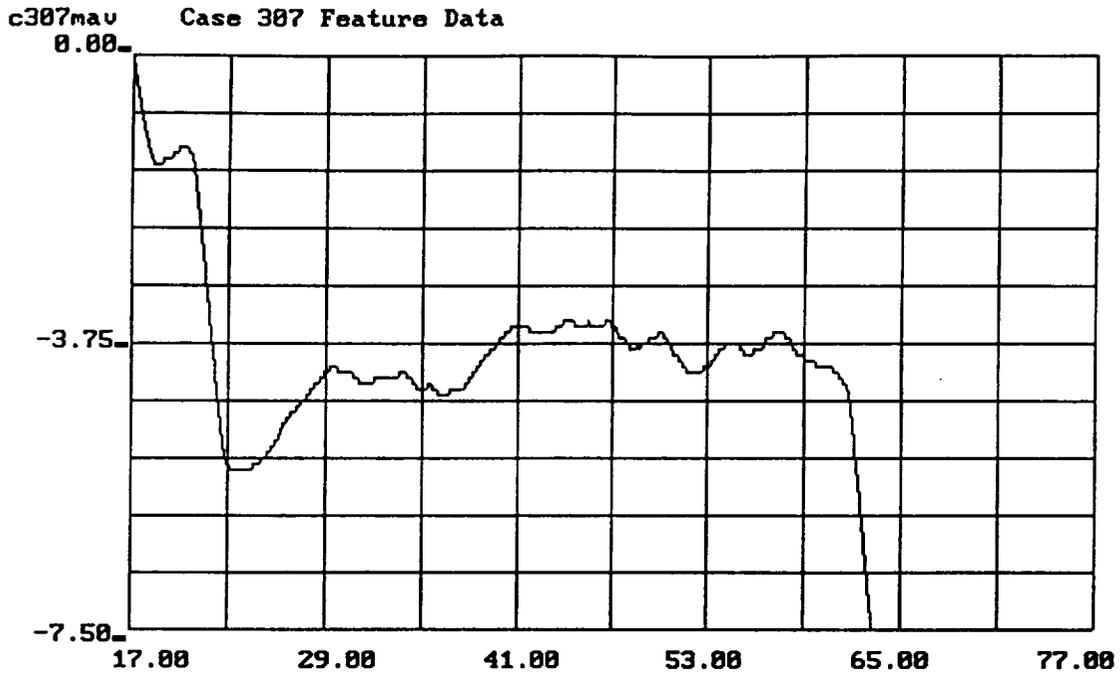
Figure A.2.5 Graph of Feature 2 Values, Case 307, PID 18
 Main Combustion Chamber Coolant Discharge Temperature B



-30-

Figure A.2.6 Graph of Feature 1 Values, Case 307, PID 231

High Pressure Fuel Turbine Discharge Temperature A



-31-

Figure A.2.7 Graph of Feature 2 Values, Case 307, PID 231

High Pressure Fuel Turbine Discharge Temperature A

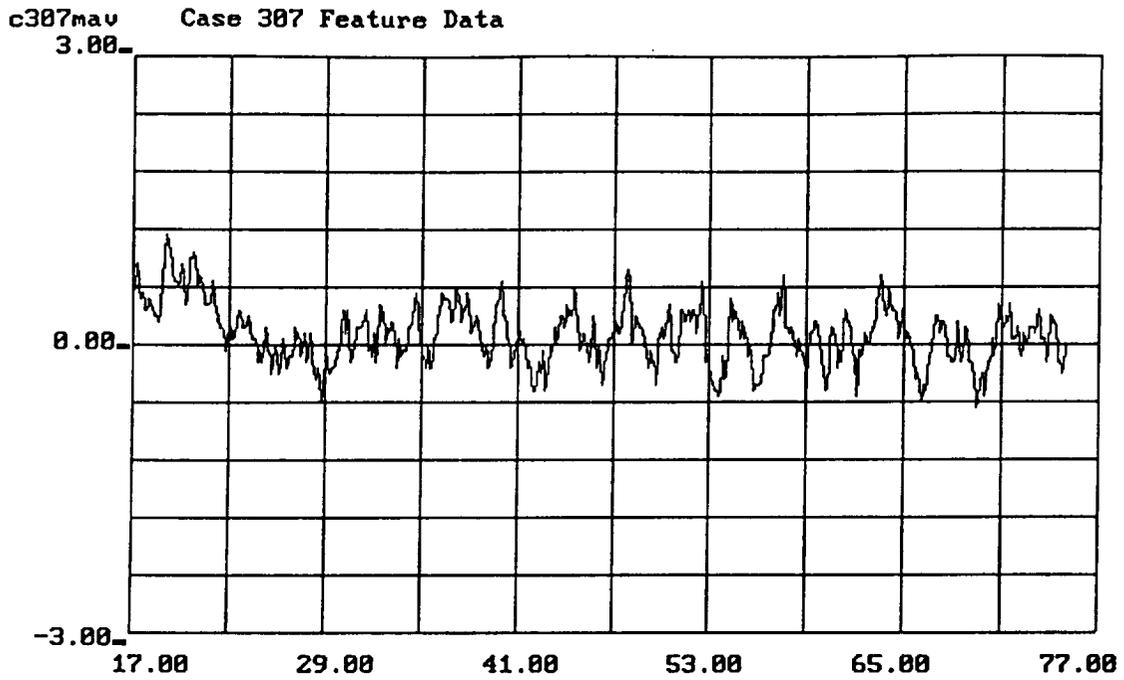
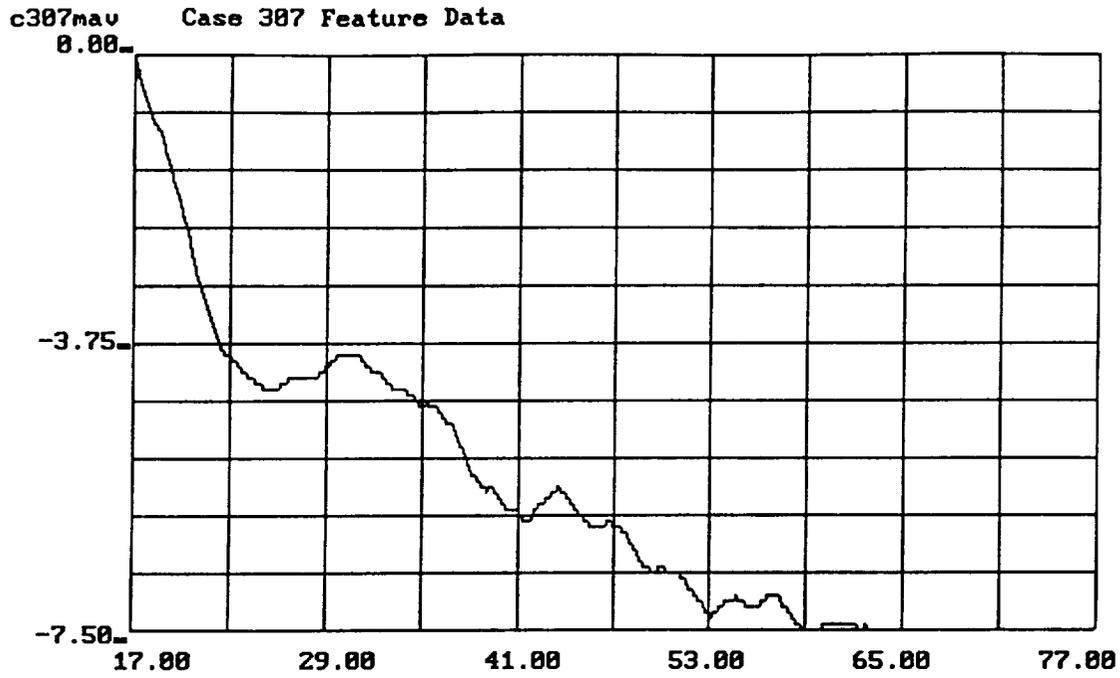


Figure A.2.8 Graph of Feature 1 Values, Case 307, PID 232

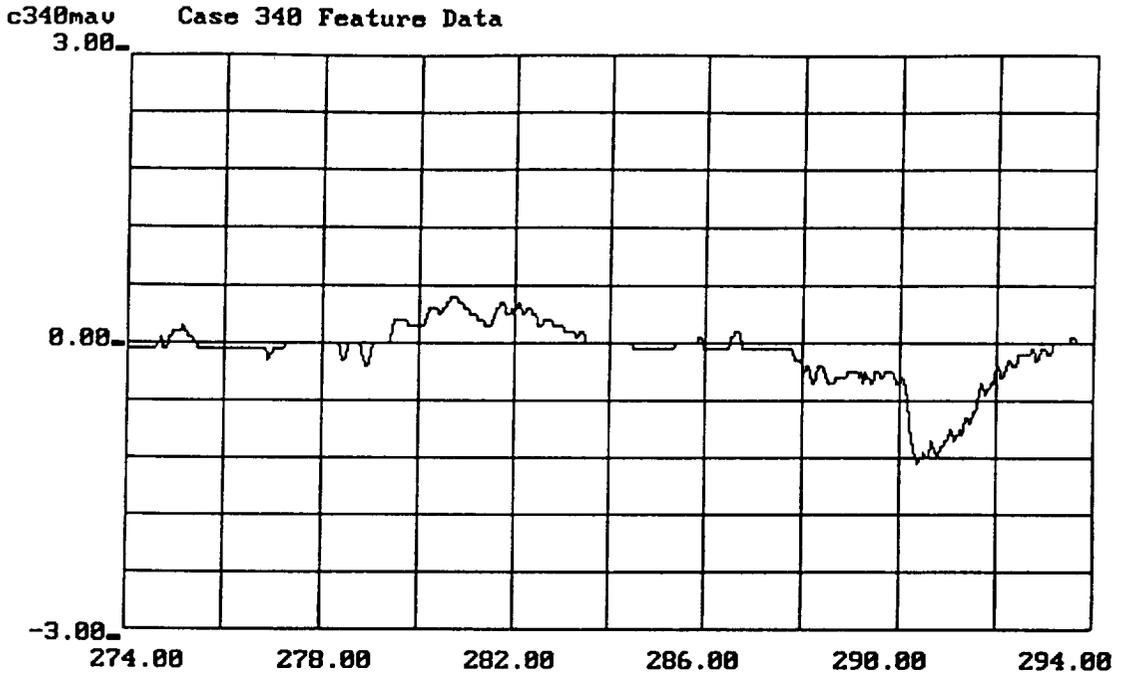
High Pressure Fuel Turbine Discharge Temperature B



-35-

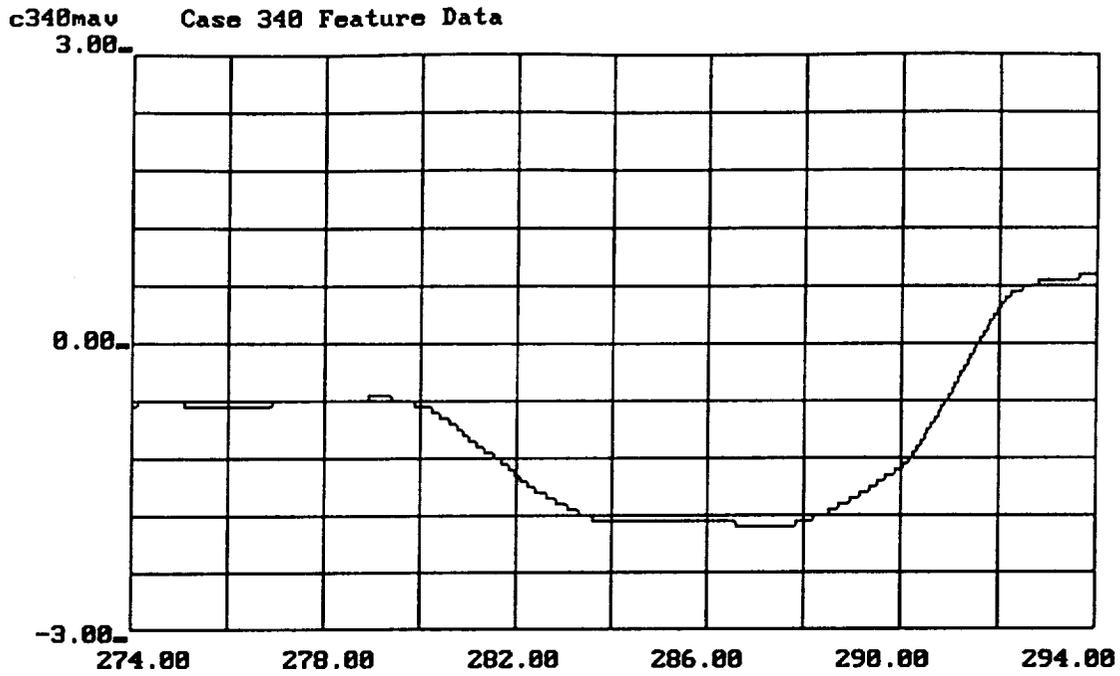
Figure A.2.9 Graph of Feature 2 Values, Case 307, PID 232

High Pressure Fuel Turbine Discharge Temperature B



-10-

Figure A.2.10 Graph of Feature 1 Values, Case 340, PID 40
 Oxidizer-Preburner Oxidizer Valve Actuator Position A



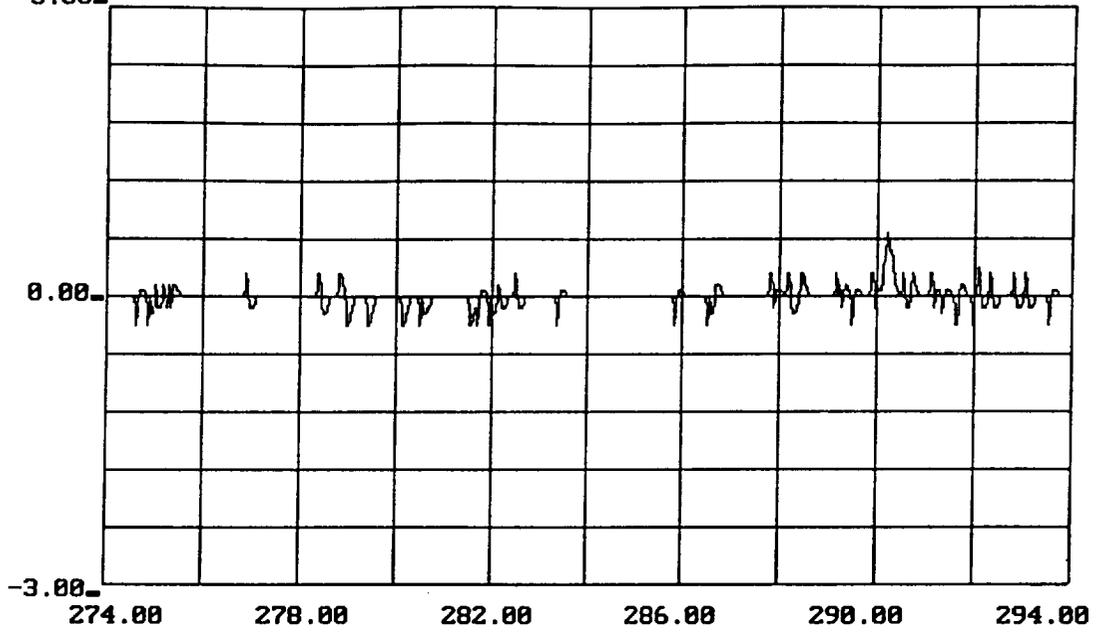
-11-

Figure A.2.11 Graph of Feature 2 Values, Case 340, PID 40

Oxidizer-Preburner Oxidizer Valve Actuator Position A

c340mav
3.00

Case 340 Feature Data



-12-

Figure A.A Graph of Feature 3 Values, Case 340, PID 40

Oxidizer-Preburner Oxidizer Valve Actuator Position A

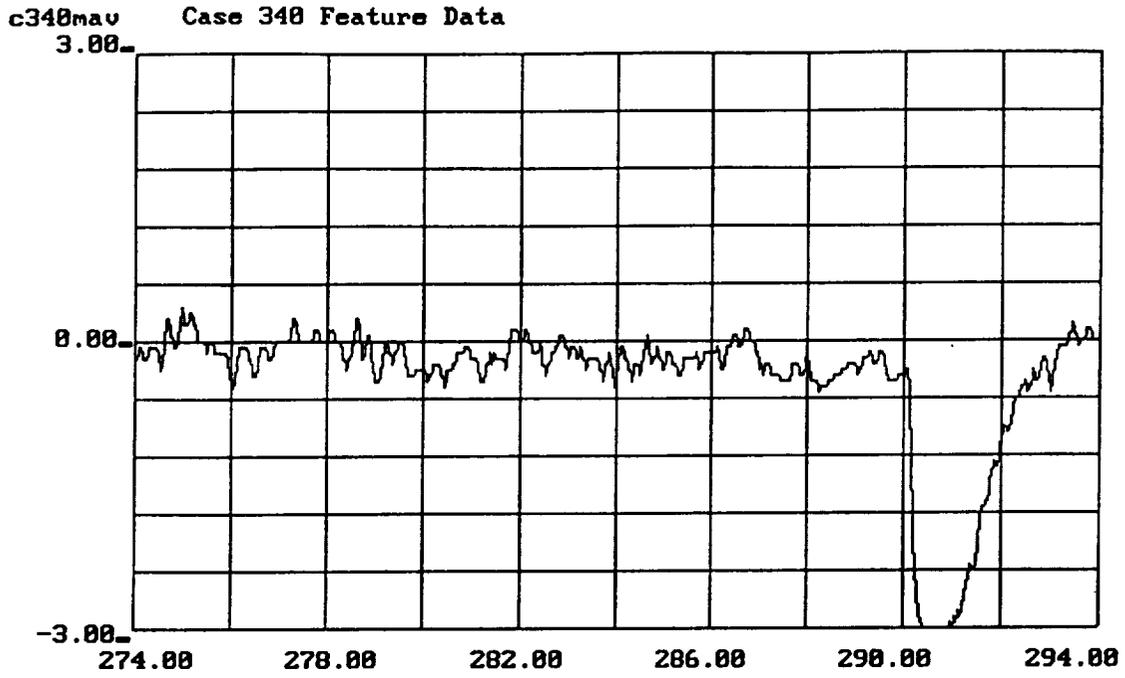
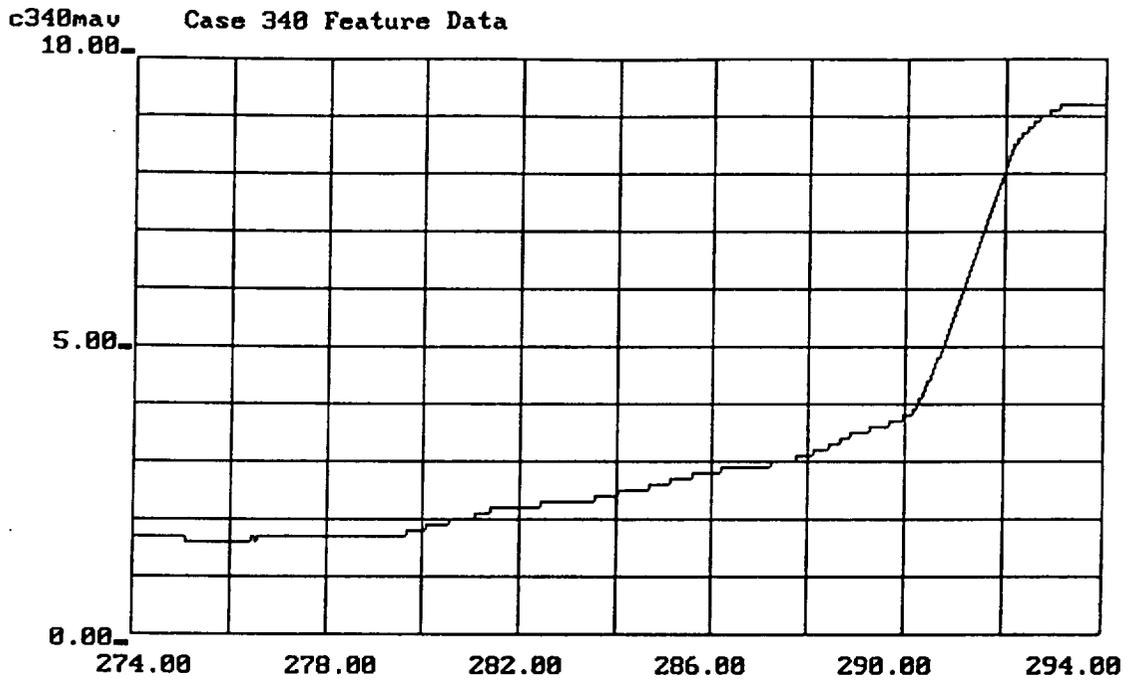


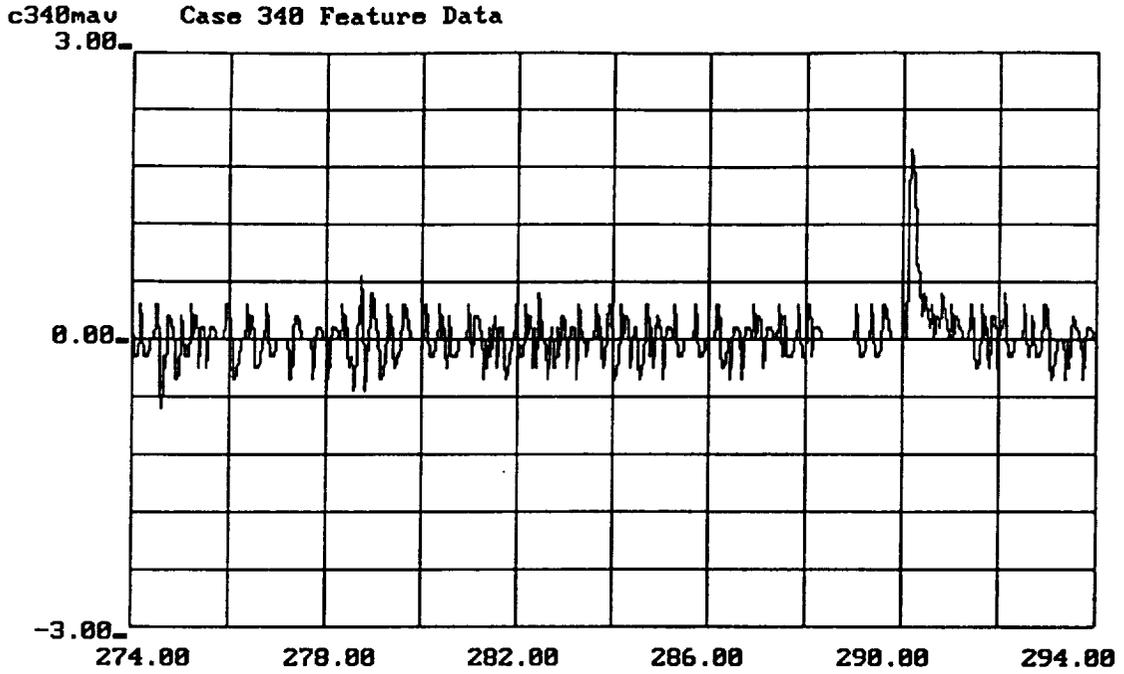
Figure A.2.13 Graph of Feature 1 Values, Case 340, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A



-15-

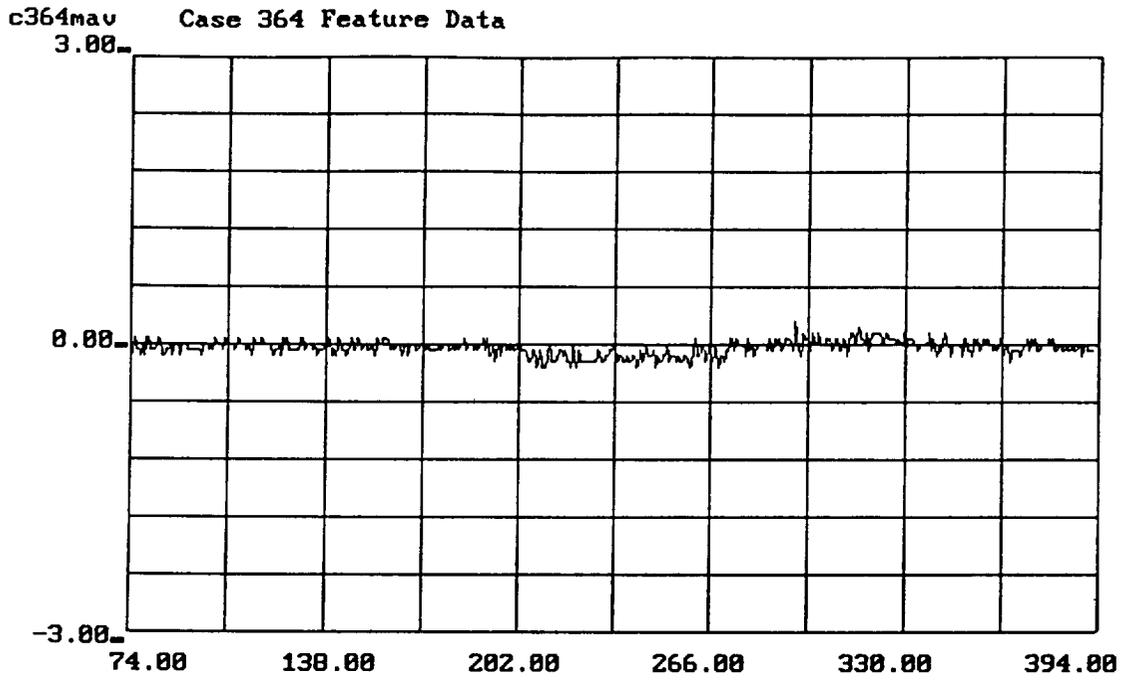
Figure A.2.14 Graph of Feature 2 Values, Case 340, PID 42

Fuel Preburner Oxidizer Valve Actuator Position A



-16-

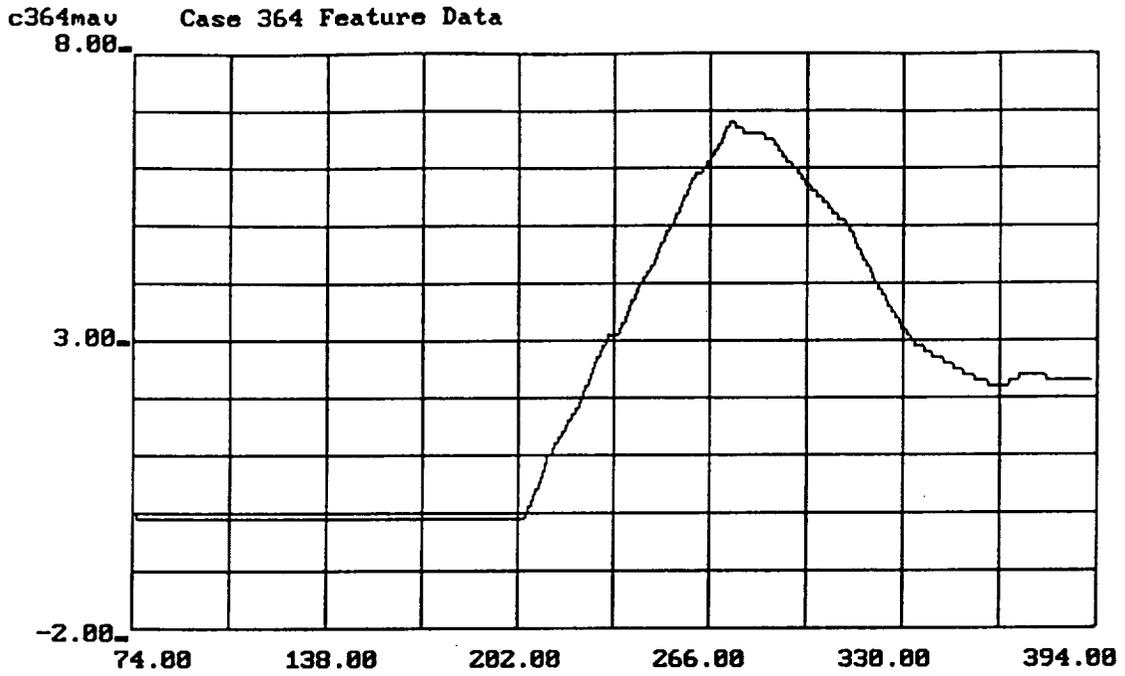
Figure A.2.15 Graph of Feature 3 Values, Case 340, PID 42
Fuel Preburner Oxidizer Valve Actuator Position A



-26-

Figure A.2.16 Graph of Feature 1 Values, Case 364, PID 209

Low Pressure Oxidizer Pump Discharge Pressure A



-27-

Figure A.2.17 Graph of Feature 2 Values, Case 364, PID 209

Low Pressure Oxidizer Pump Discharge Pressure A

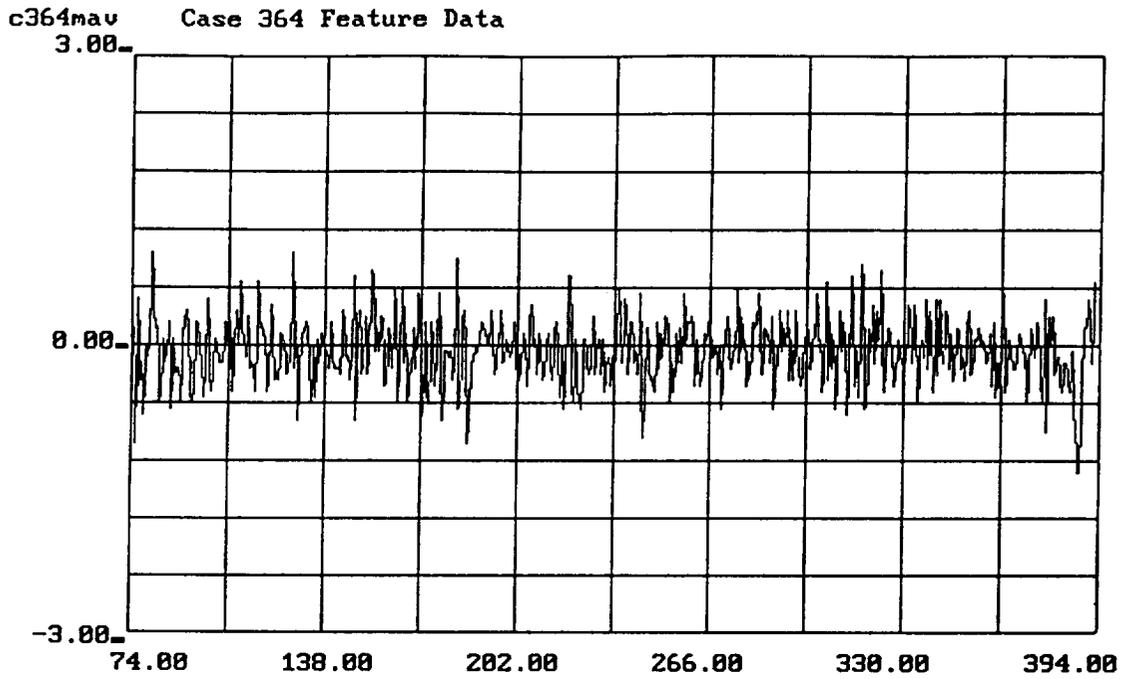
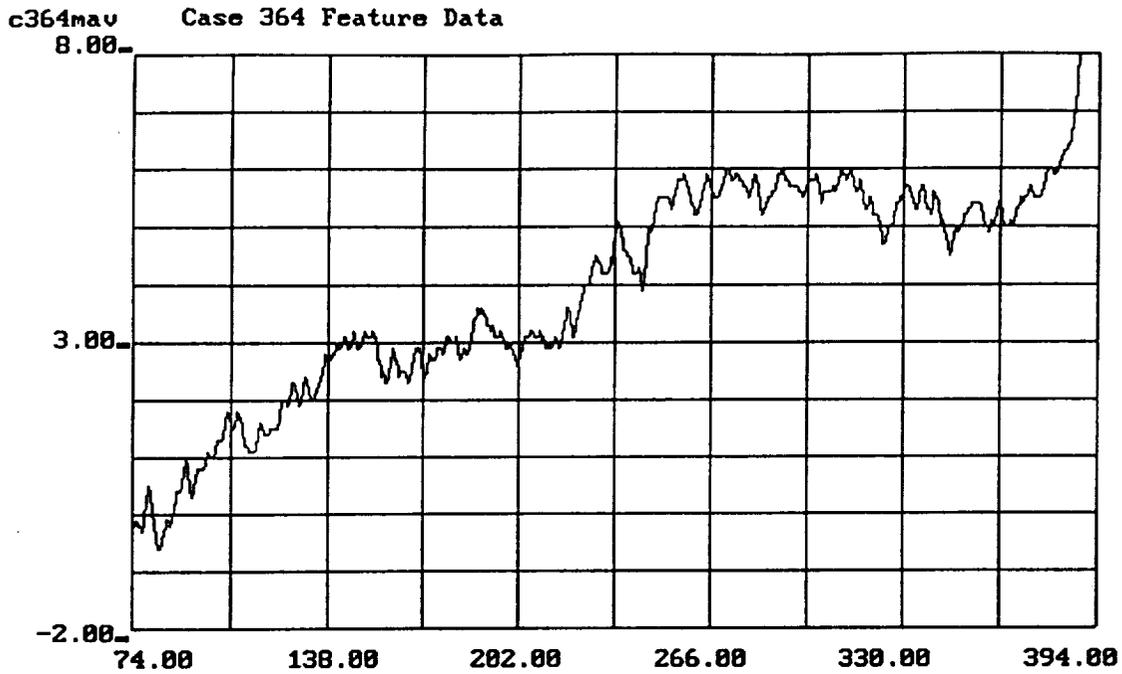


Figure A.2.18 Graph of Feature 1 Values, Case 364, PID 231

High Pressure Fuel Turbine Discharge Temperature A



-31-

Figure A.2.19 Graph of Feature 2 Values, Case 364, PID 231

High Pressure Fuel Turbine Discharge Temperature A

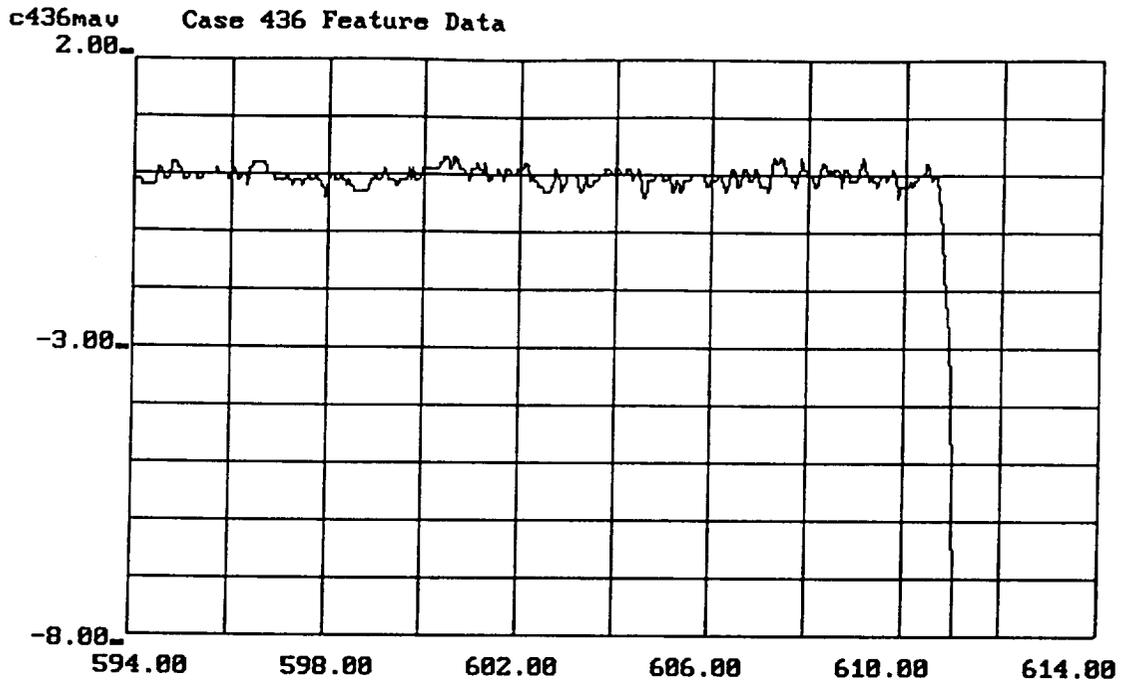
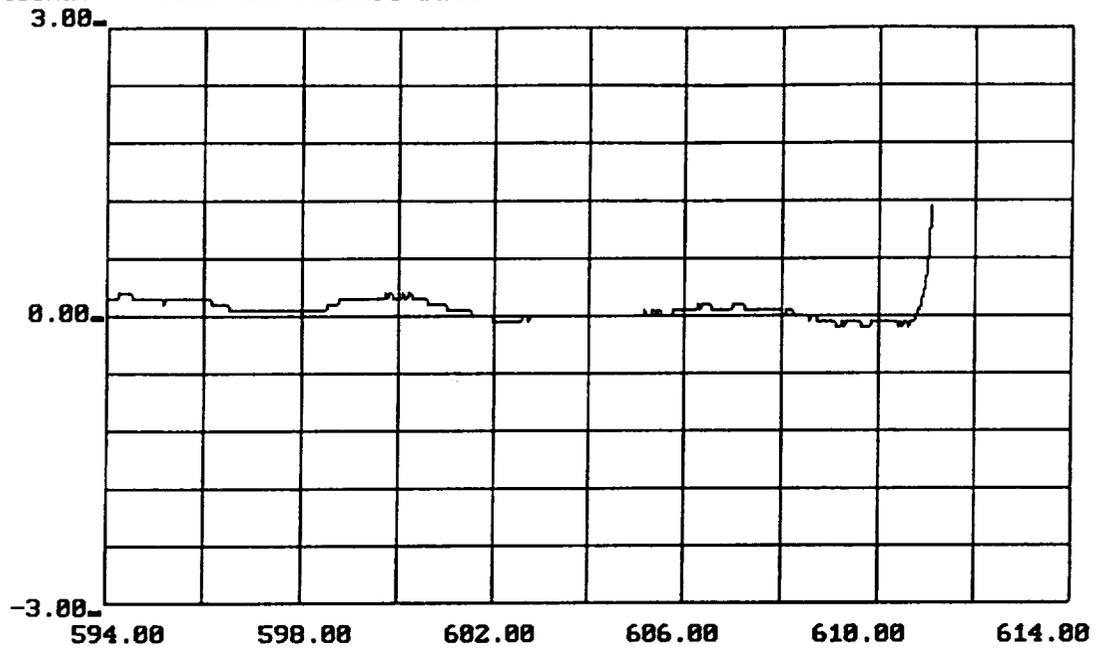


Figure A.2.20 Graph of Feature 1 Values, Case 436t2, PID 42

Fuel Preburner Oxidizer Valve Actuator Position A

c436nav Case 436 Feature Data



-15-

Figure A.2.21 Graph of Feature 2 Values, Case 436t2, PID 42

Fuel Preburner Oxidizer Valve Actuator Position A

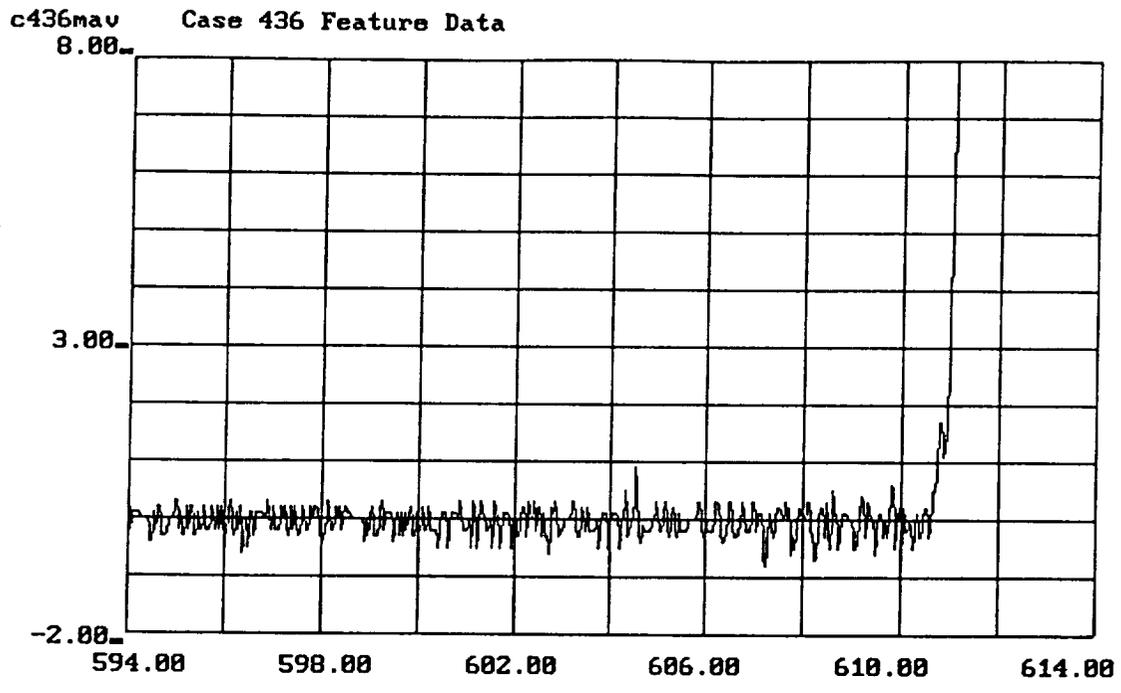


Figure A.2.22 Graph of Feature 3 Values, Case 436t2, PID 42
 Fuel Preburner Oxidizer Valve Actuator Position A

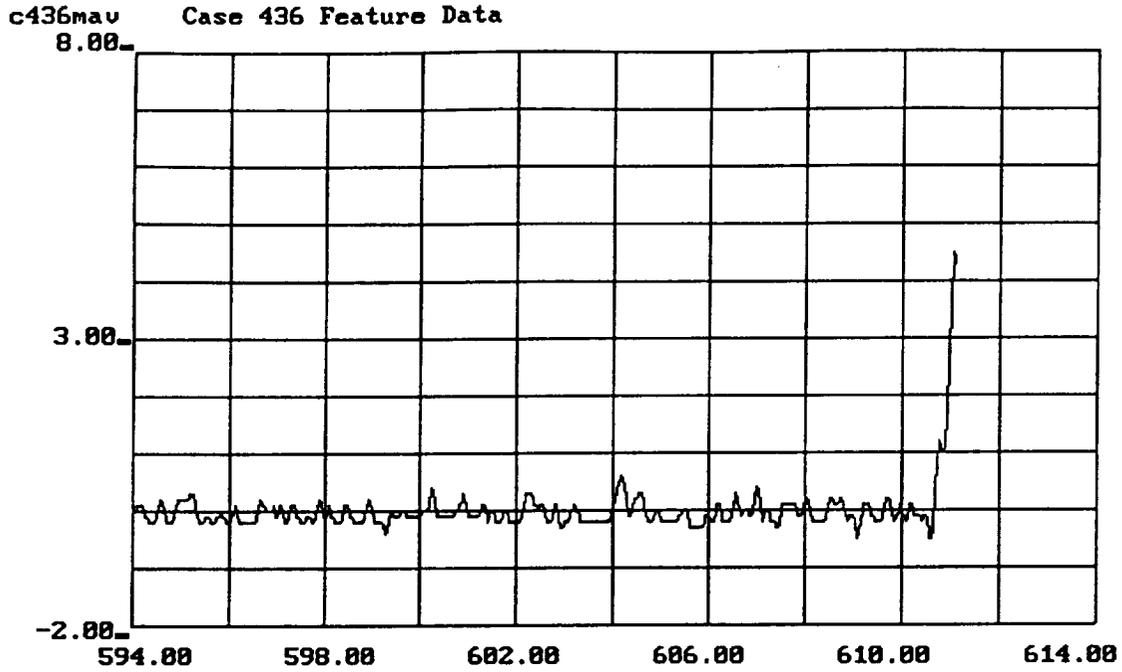
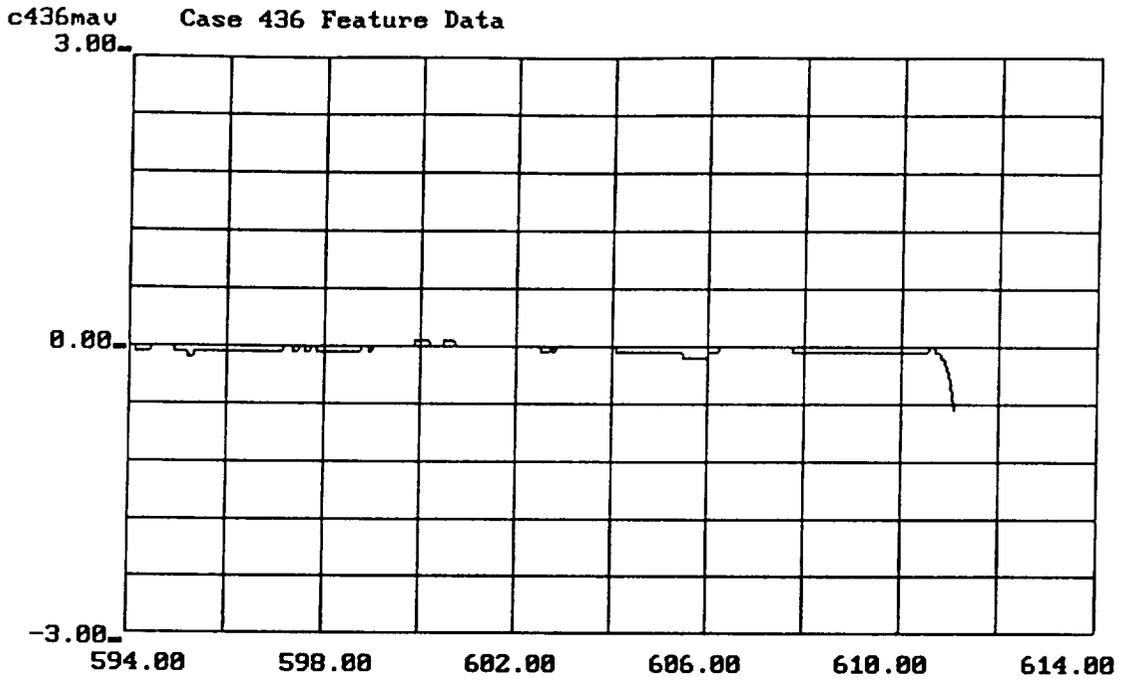


Figure A.2.23 Graph of Feature 1 Values, Case 436t2, PID 52

High Pressure Fuel Pump Discharge Pressure A



-19-

Figure A.2.24 Graph of Feature 2 Values, Case 436t2, PID 52

High Pressure Fuel Pump Discharge Pressure A

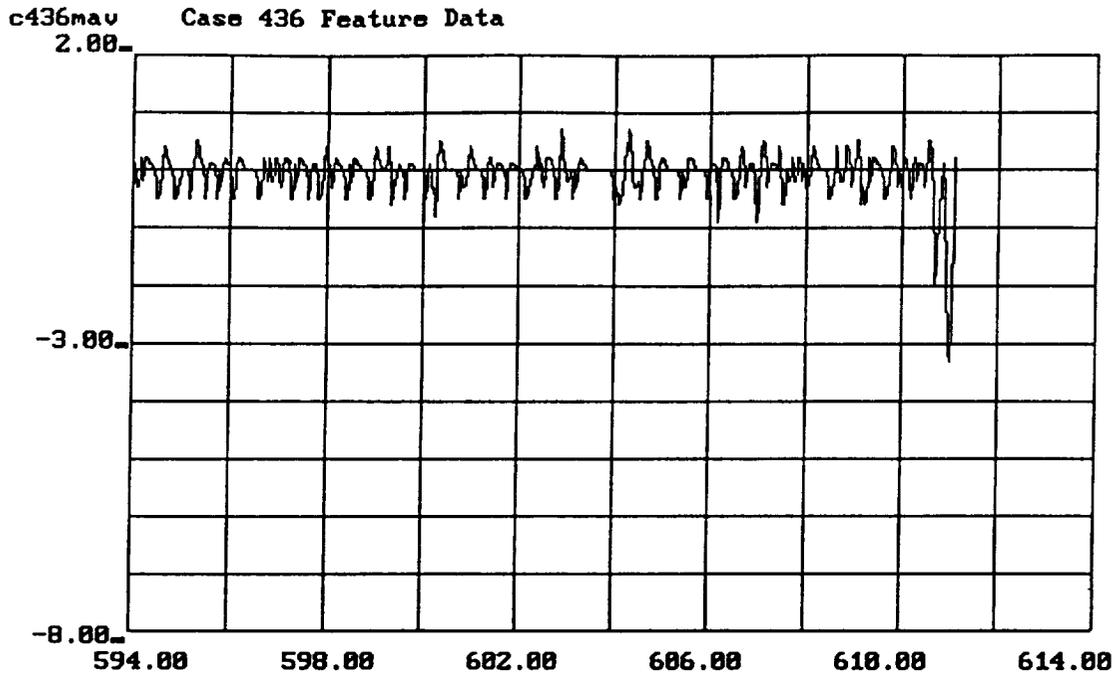
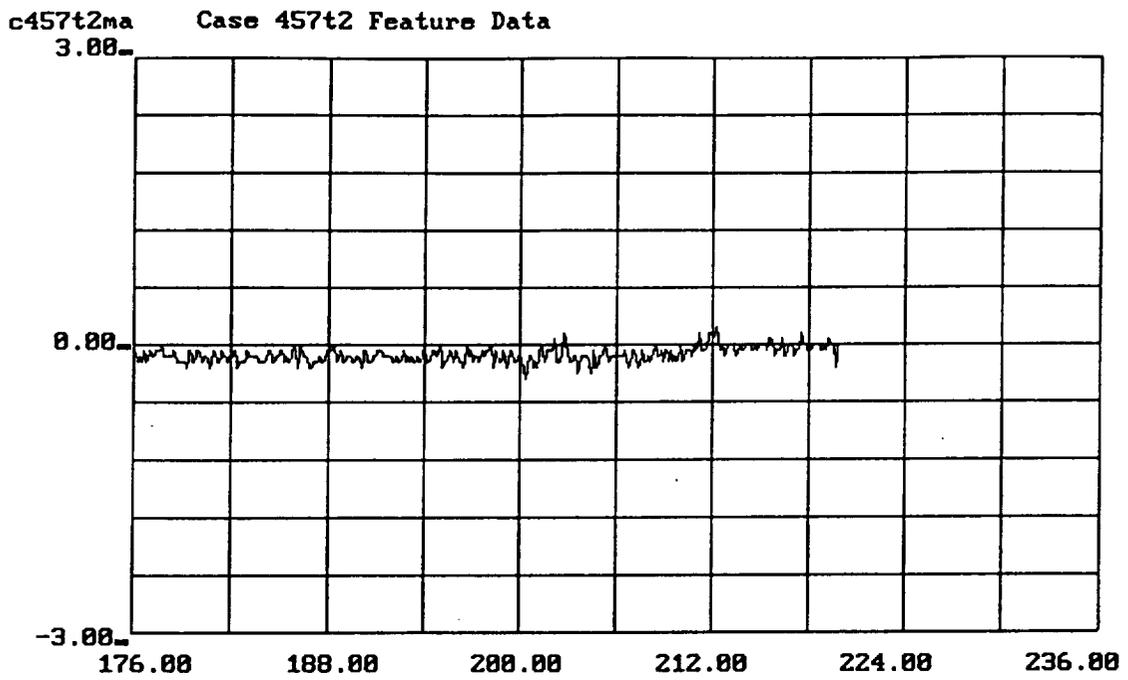


Figure A.2.25 Graph of Feature 3 Values, Case 436t2, PID 52

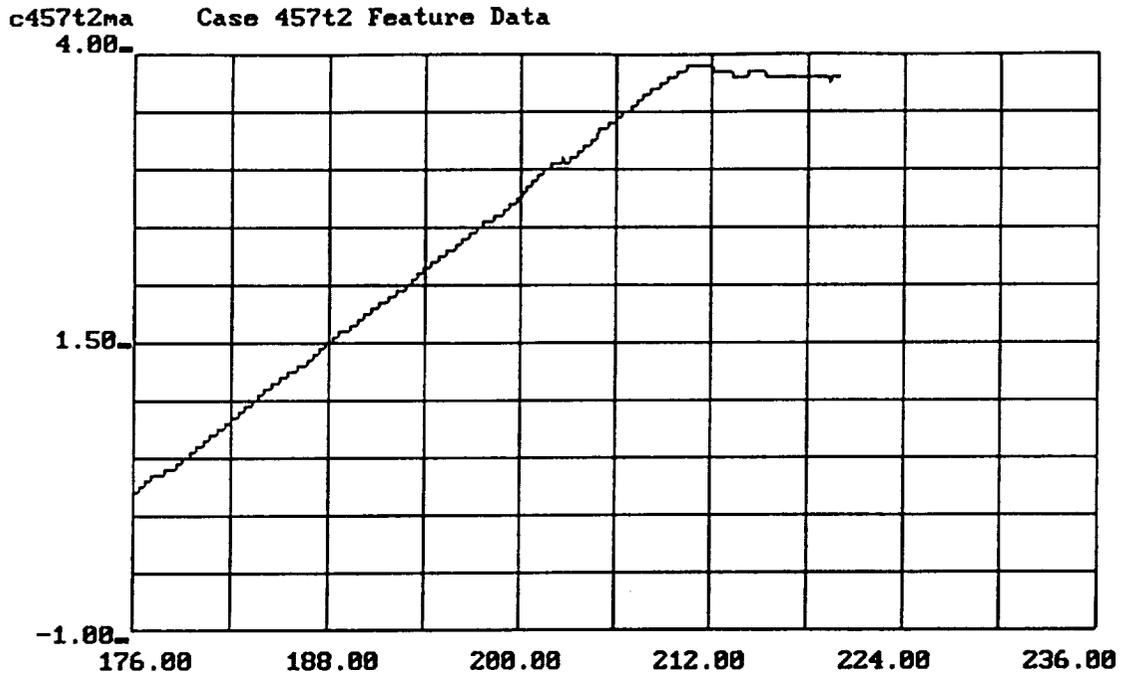
High Pressure Fuel Pump Discharge Pressure A



-26-

Figure A.2.26 Graph of Feature 1 Values, Case 457t2, PID 209

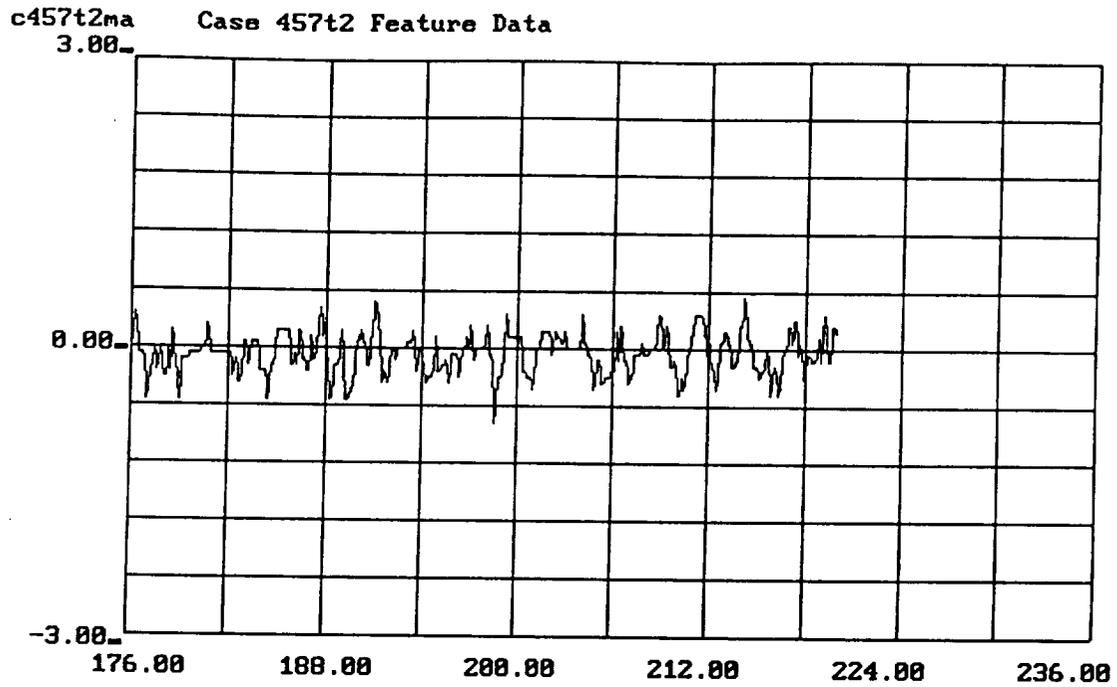
Low Pressure Oxidizer Pump Discharge Pressure A



-27-

Figure A.2.27 Graph of Feature 2 Values, Case 457t2, PID 209

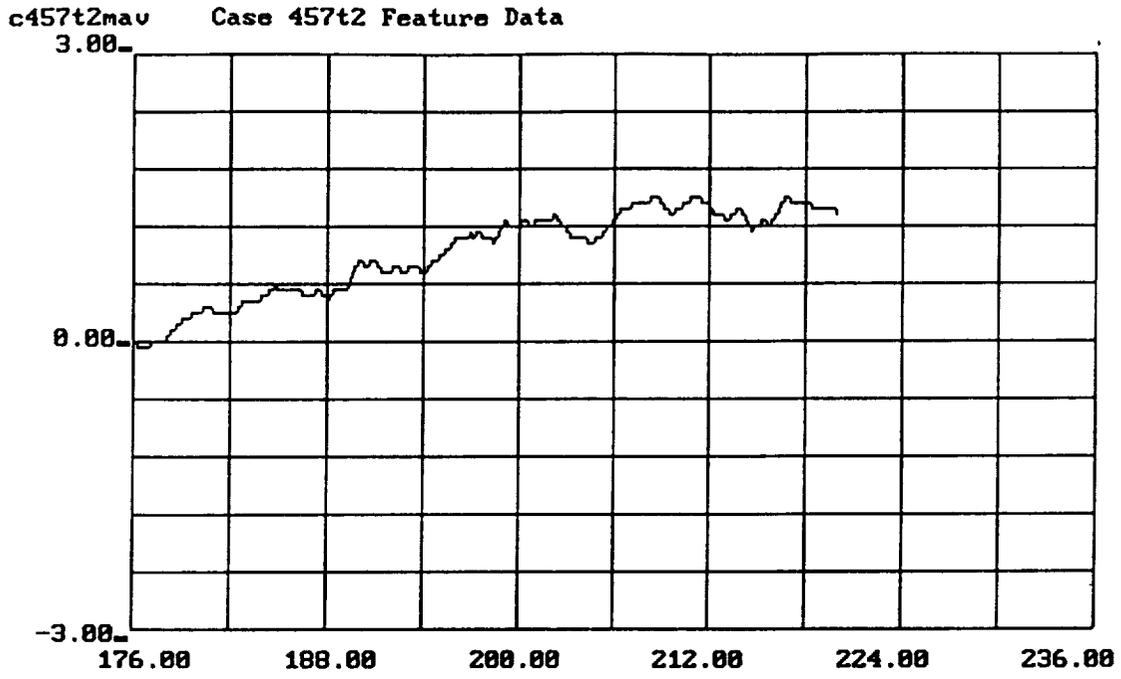
Low Pressure Oxidizer Pump Discharge Pressure A



-38-

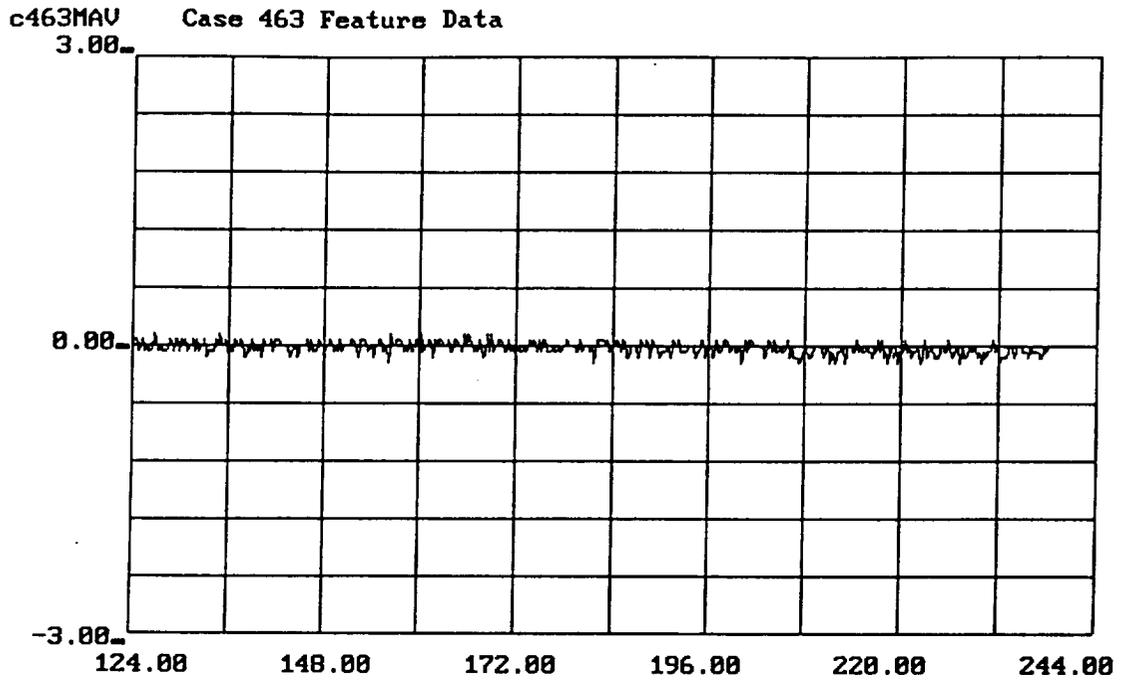
Figure A.2.28 Graph of Feature 1 Values, Case 457t2, PID 231

High Pressure Fuel Turbine Discharge Temperature A



-31-

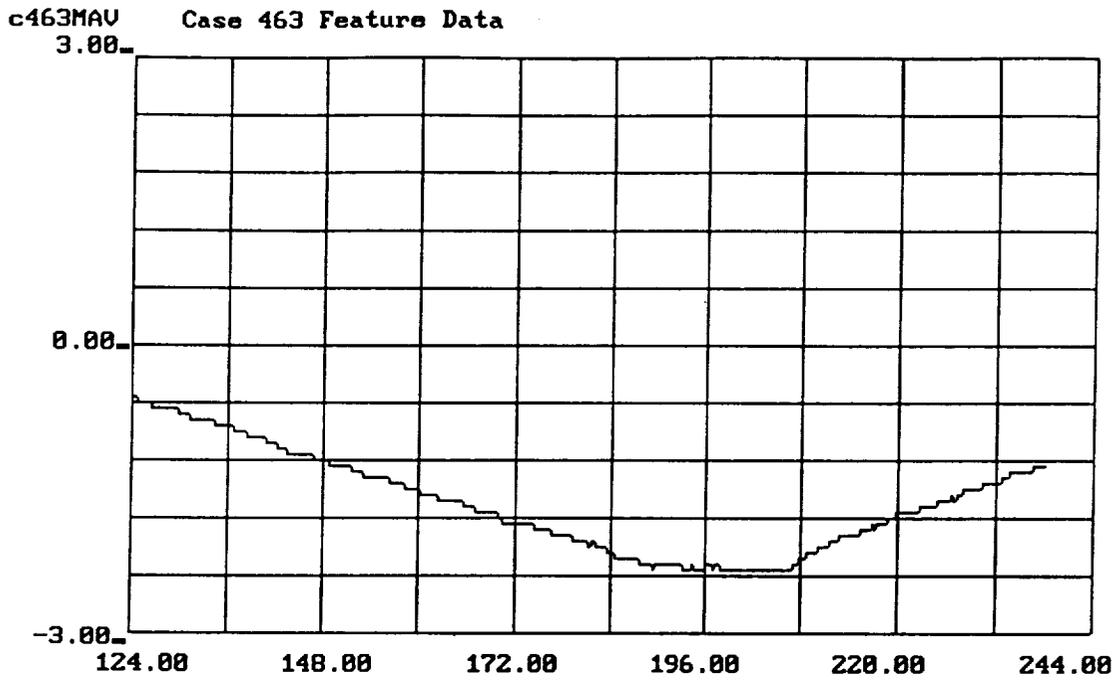
Figure A.2.29 Graph of Feature 2 Values, Case 457t2, PID 231
 High Pressure Fuel Turbine Discharge Temperature A



-26-

Figure A.2.30 Graph of Feature 1 Values, Case 463, PID 209

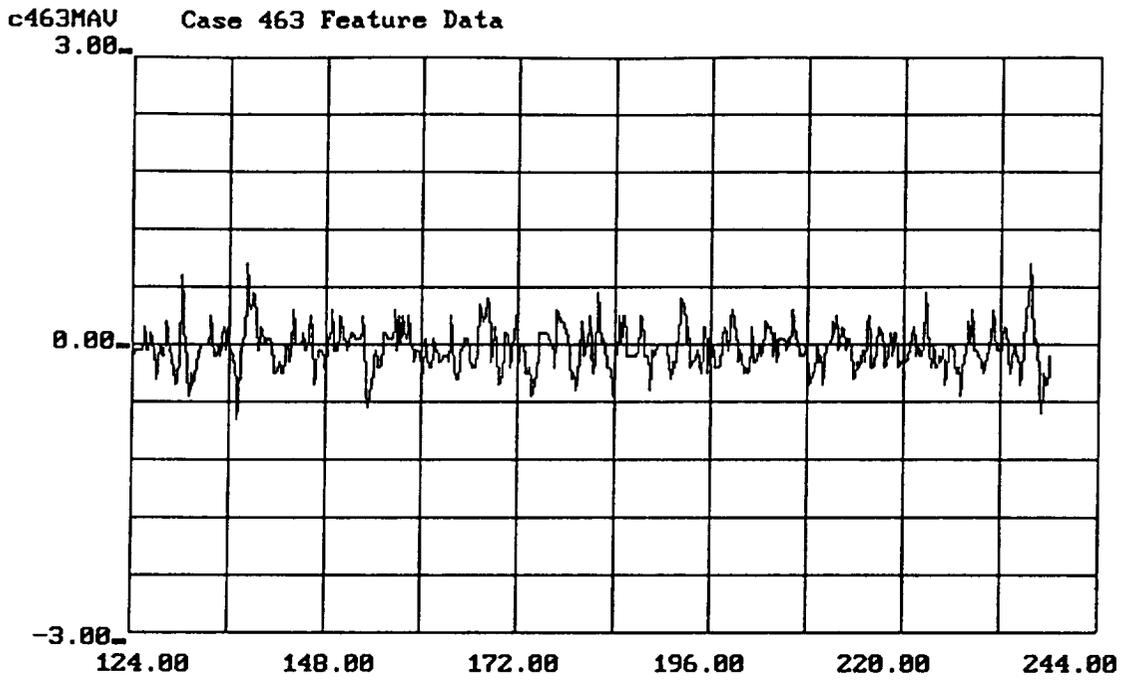
Low Pressure Oxidizer Pump Discharge Pressure A



-27-

Figure A.2.31 Graph of Feature 2 Values, Case 463, PID 209

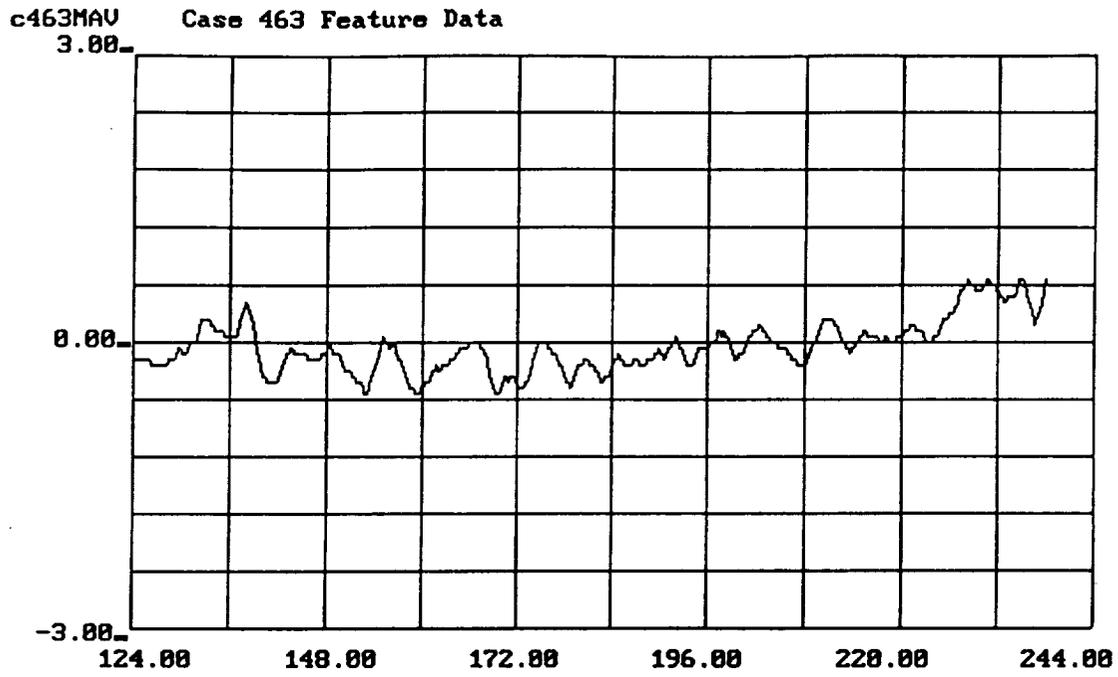
Low Pressure Oxidizer Pump Discharge Pressure A



-30-

Figure A.2.32 Graph of Feature 1 Values, Case 463, PID 231

High Pressure Fuel Turbine Discharge Temperature A



-31-

Figure A.2.33 Graph of Feature 2 Values, Case 463, PID 231

High Pressure Fuel Turbine Discharge Temperature A

no173 No173 Holdout Case 173

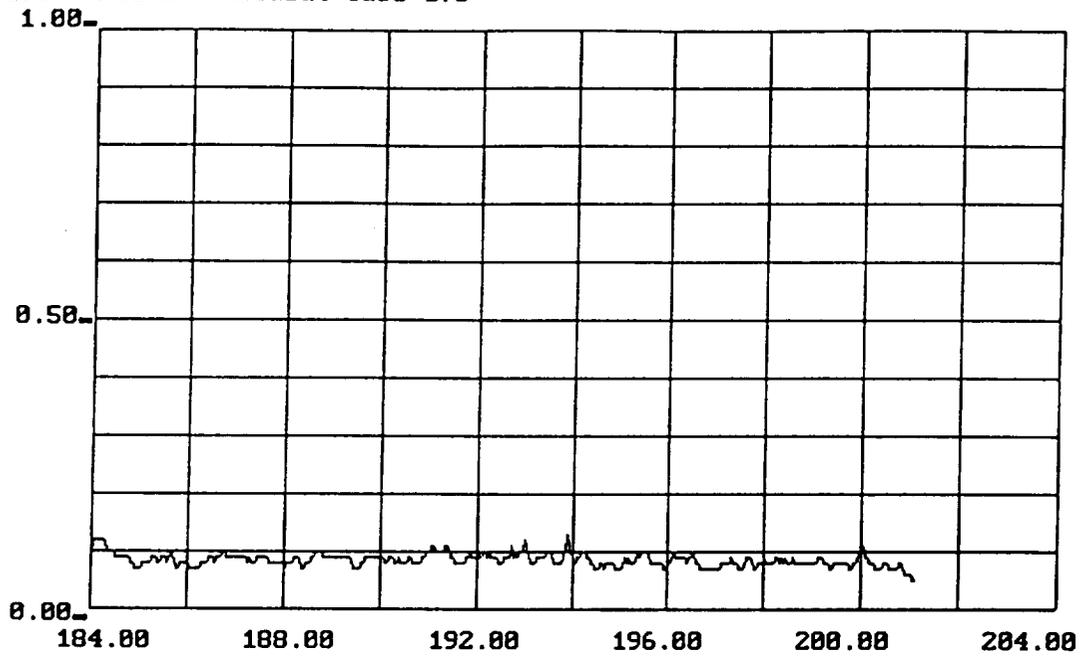
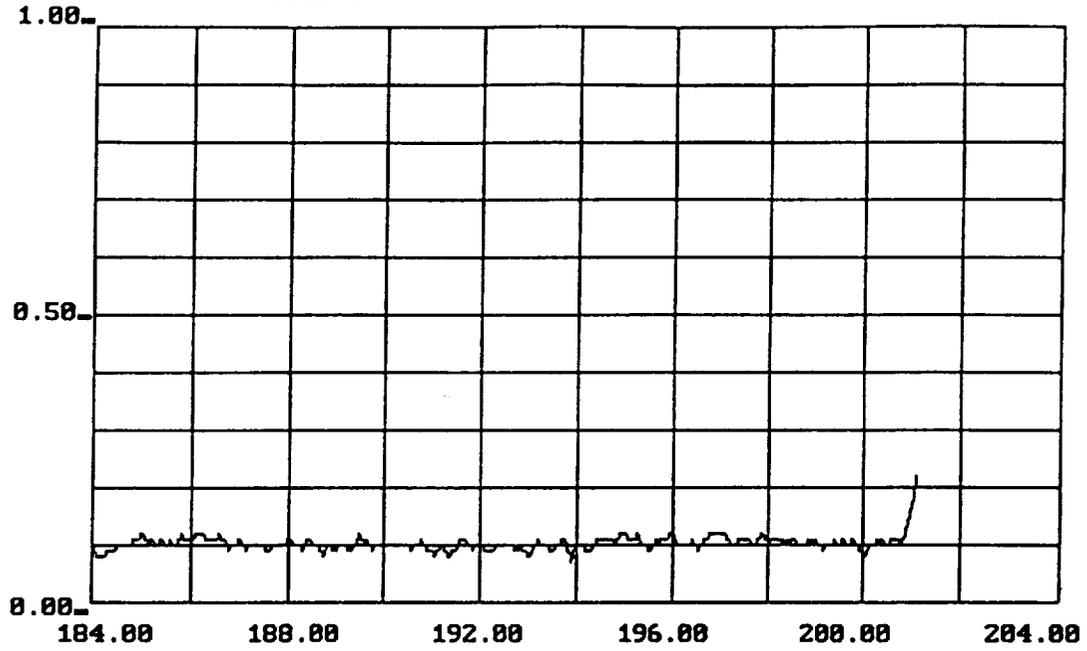


Figure A.3.1 Graph of Neural Net Output Unit 0, Holdout Case 173

Nominal/Fault Unit

no173 No173 Holdout Case 173

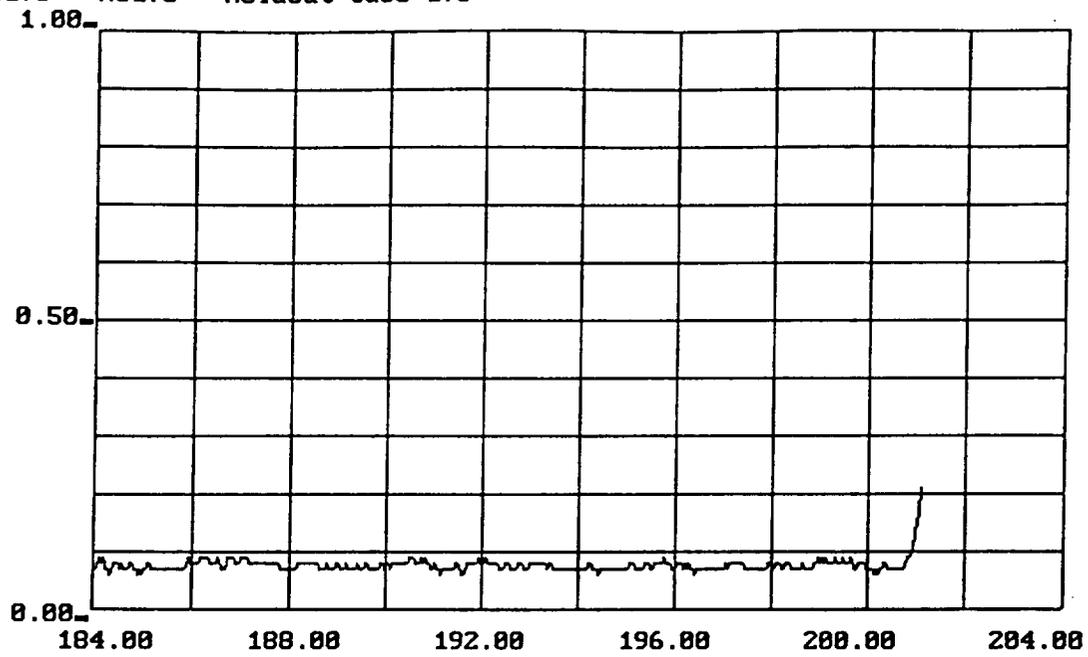


-4-

Figure A.3.2 Graph of Neural Net Output Unit 4, Holdout Case 173

Case 259 Unit

no173 No173 Holdout Case 173

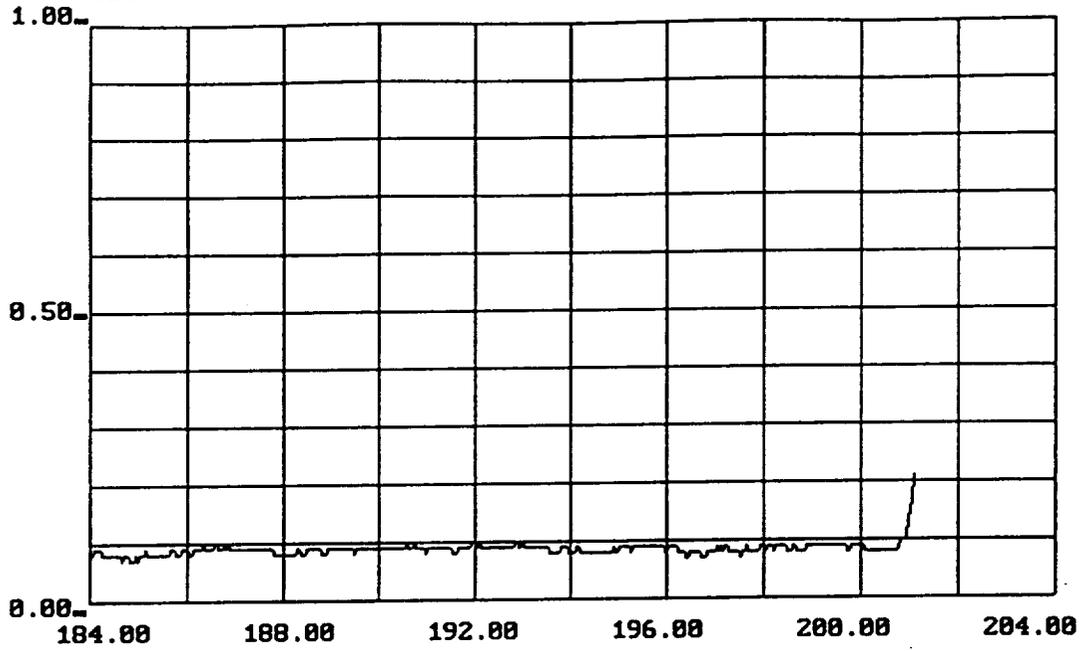


-6-

Figure A.3.3 Graph of Neural Net Output Unit 6, Holdout Case 173

Case 331 Unit

no173 No173 Holdout Case 173

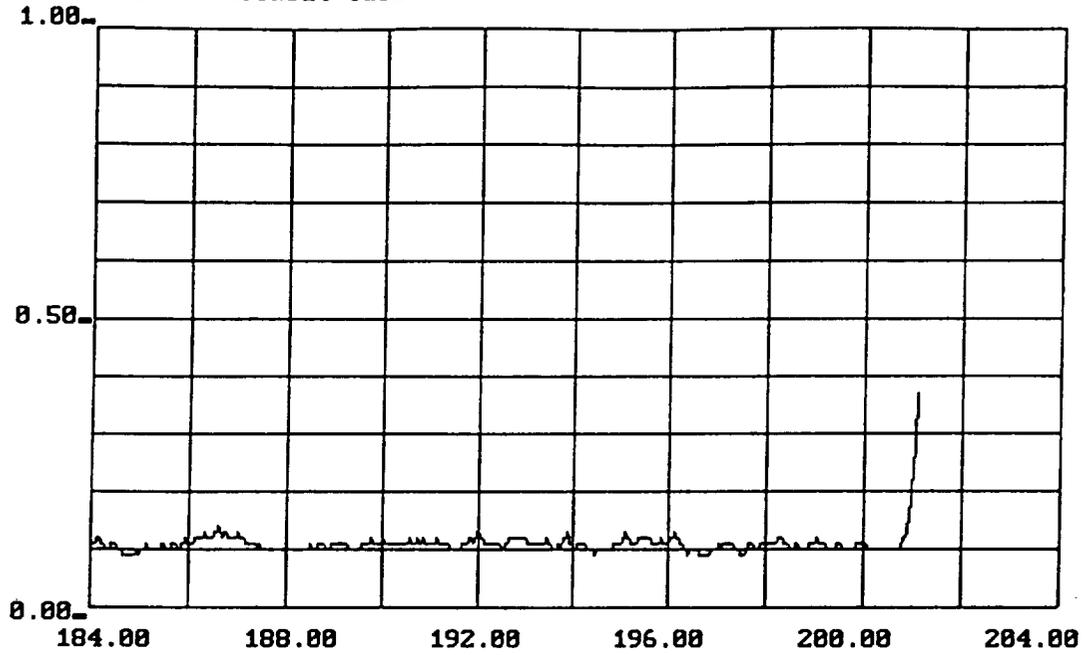


-7-

Figure A.3.4 Graph of Neural Net Output Unit 7, Holdout Case 173

Case 340 Unit

no173 No173 Holdout Case 173

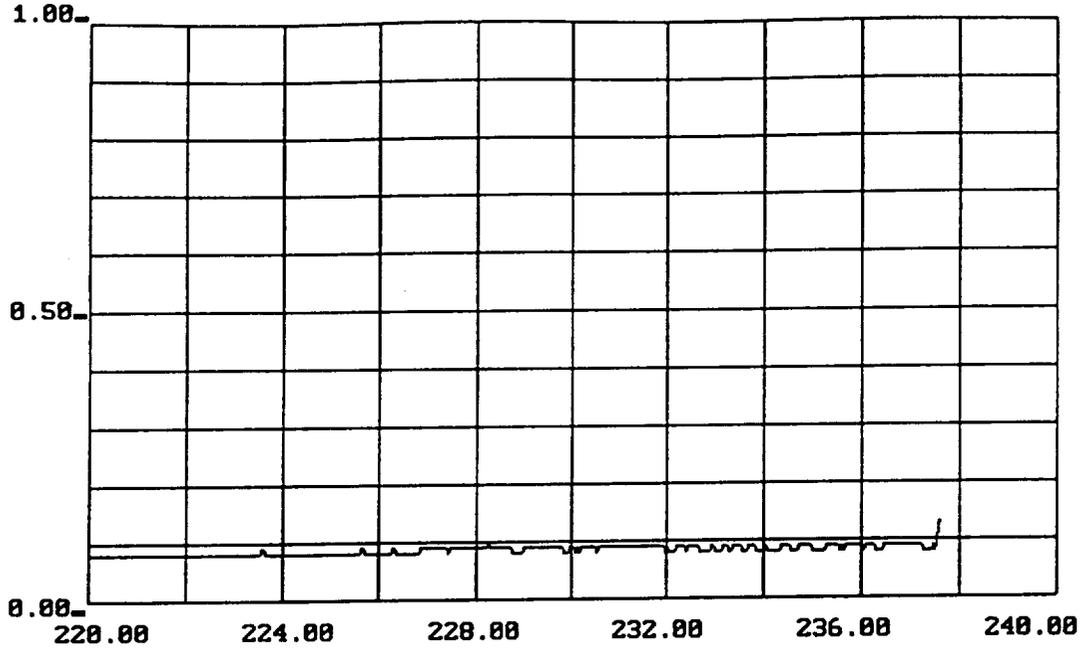


-8-

Figure A.3.5 Graph of Neural Net Output Unit 8, Holdout Case 173

Case 364 Unit

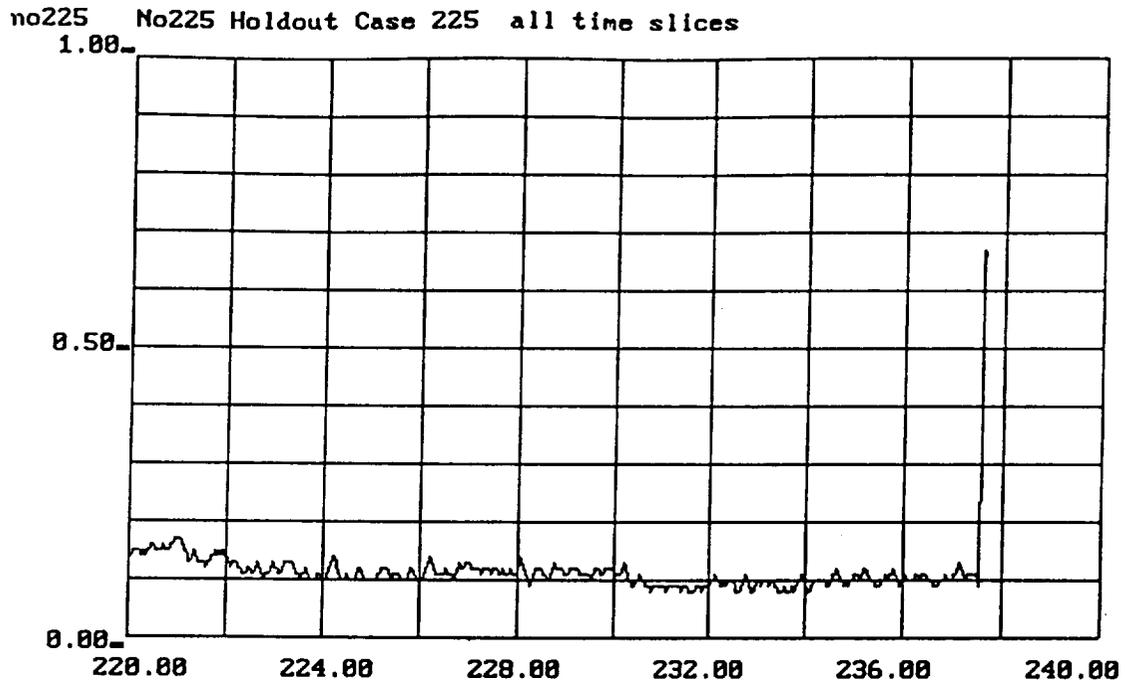
no225 No225 Holdout Case 225 all time slices



-8-

Figure A.3.6 Graph of Neural Net Output Unit 0, Holdout Case 225

Nominal/Fault Unit



-6-

Figure A.3.7 Graph of Neural Net Output Unit 6, Holdout Case 225

Case 331 Unit

no249 No249 Holdout Case 249

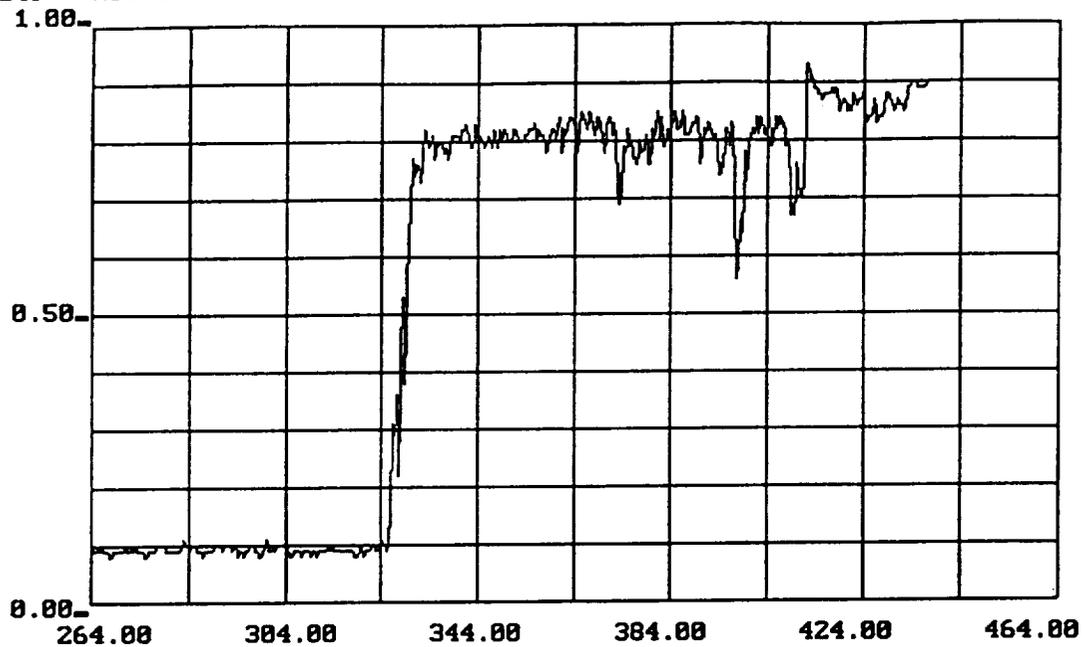
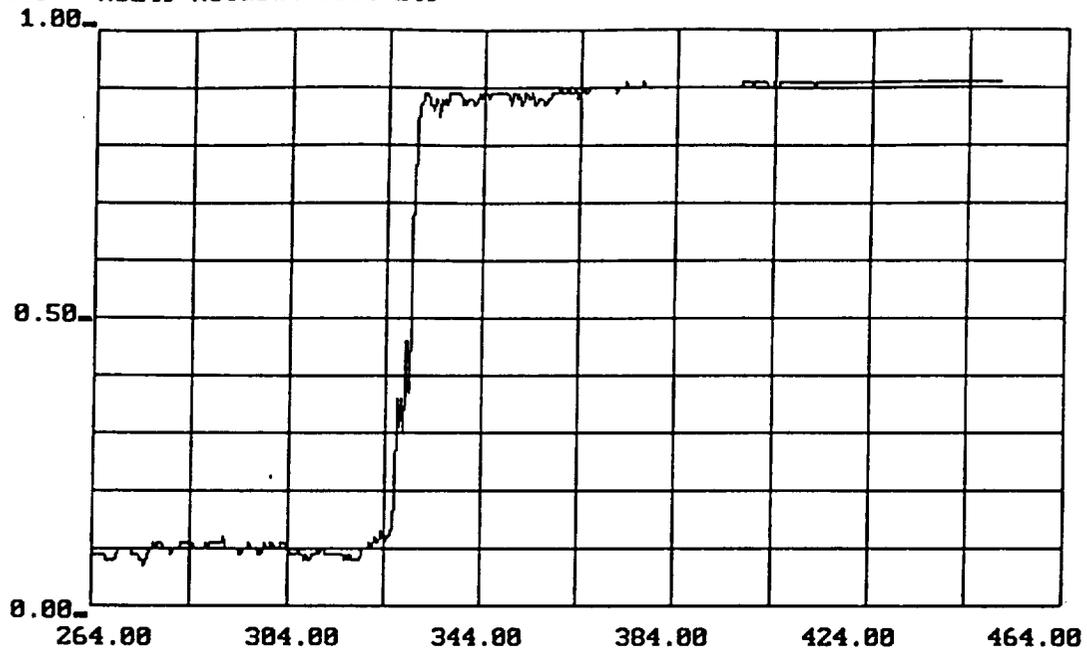


Figure A.3.8 Graph of Neural Net Output Unit 0, Holdout Case 249

Nominal/Fault Unit

no249 No249 Holdout Case 249

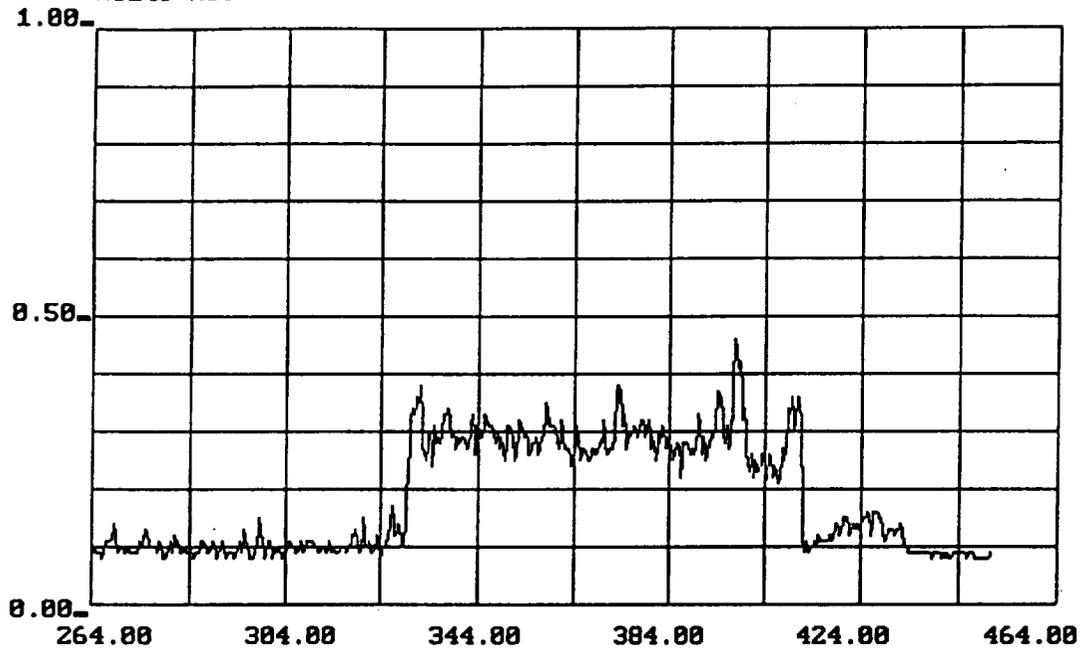


-8-

Figure A.3.9 Graph of Neural Net Output Unit 8, Holdout Case 249

Case 364 Unit

no249 No249 Holdout Case 249

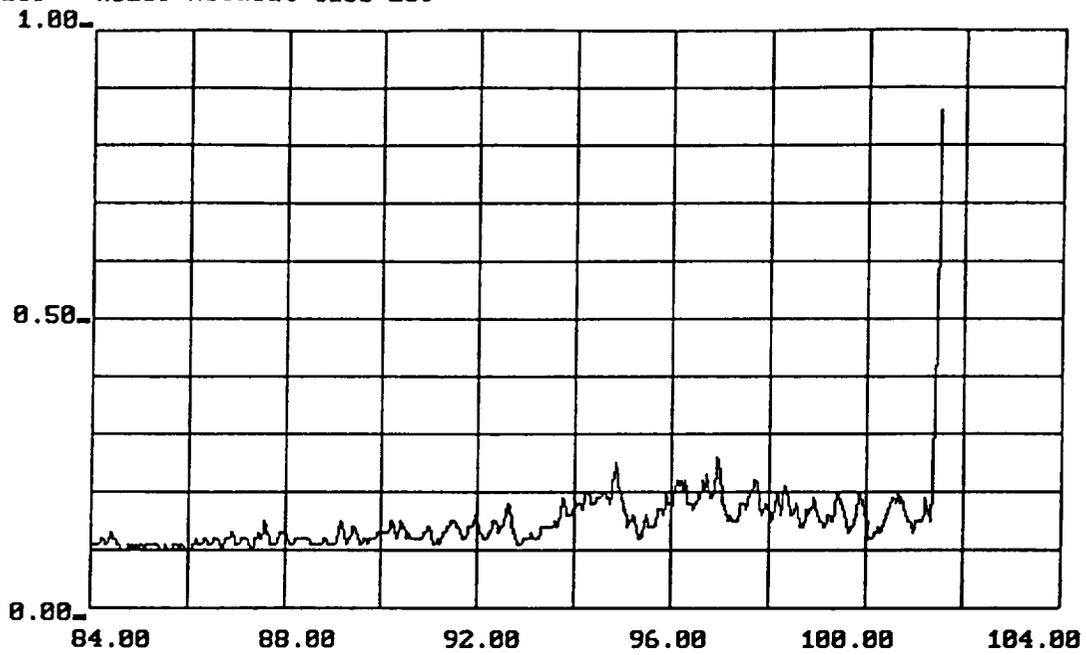


-9-

Figure A.3.10 Graph of Neural Net Output Unit 9, Holdout Case 249

Case 436 Unit

no259 no259 Holdout Case 259

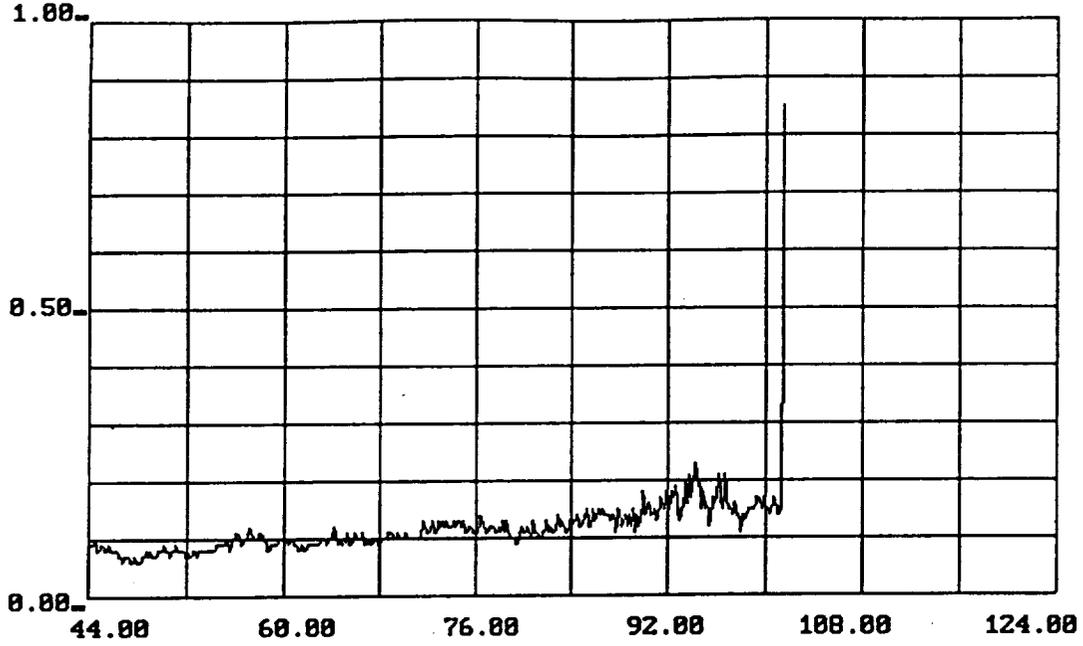


-8-

Figure A.3.11 Graph of Neural Net Output Unit 0, Holdout Case 259

Nominal/Fault Unit

no259 no259 Holdout Case 259



-7-

Figure A.3.12 Graph of Neural Net Output Unit 7, Holdout Case 259

Case 340 Unit

no387 Holdout Case 387

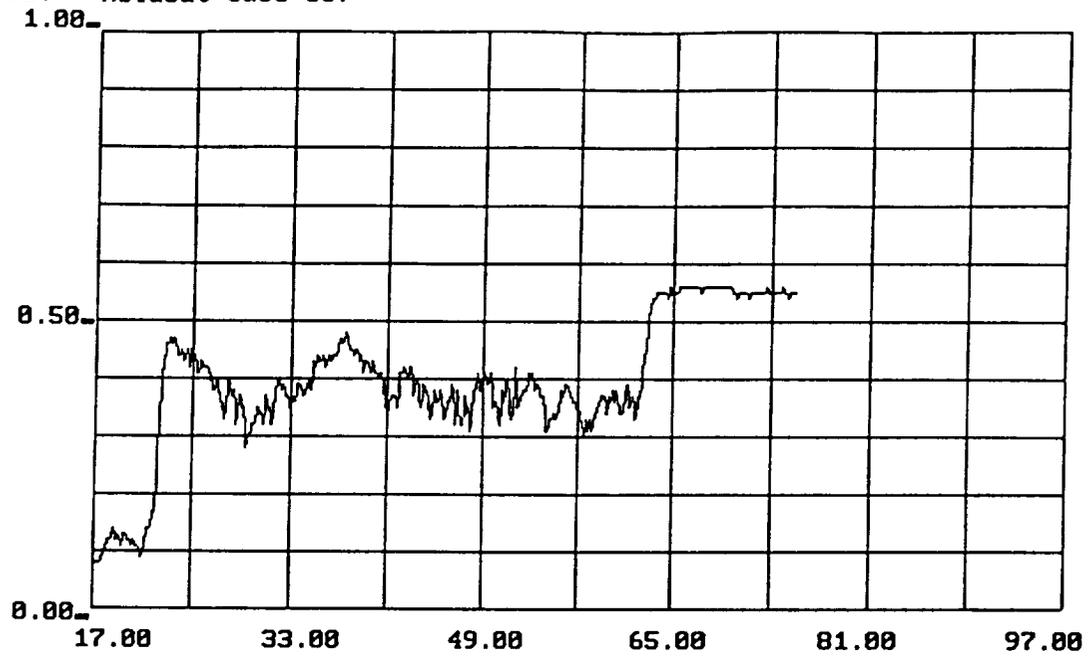
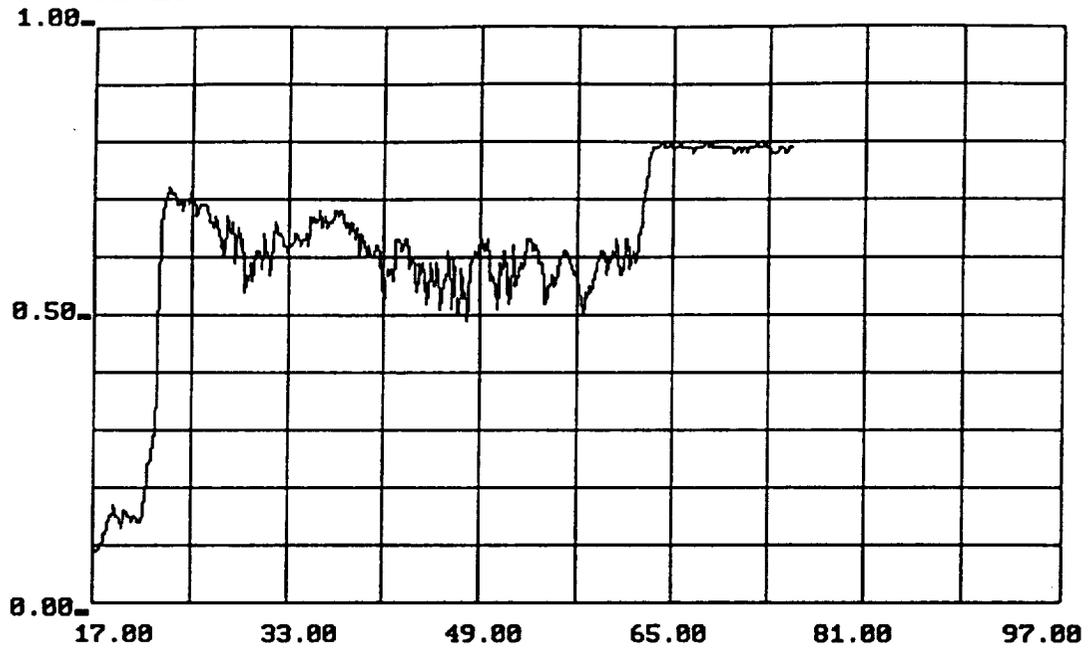


Figure A.3.13 Graph of Neural Net Output Unit 0, Holdout Case 307

Nominal/Fault Unit

no387 Holdout Case 387

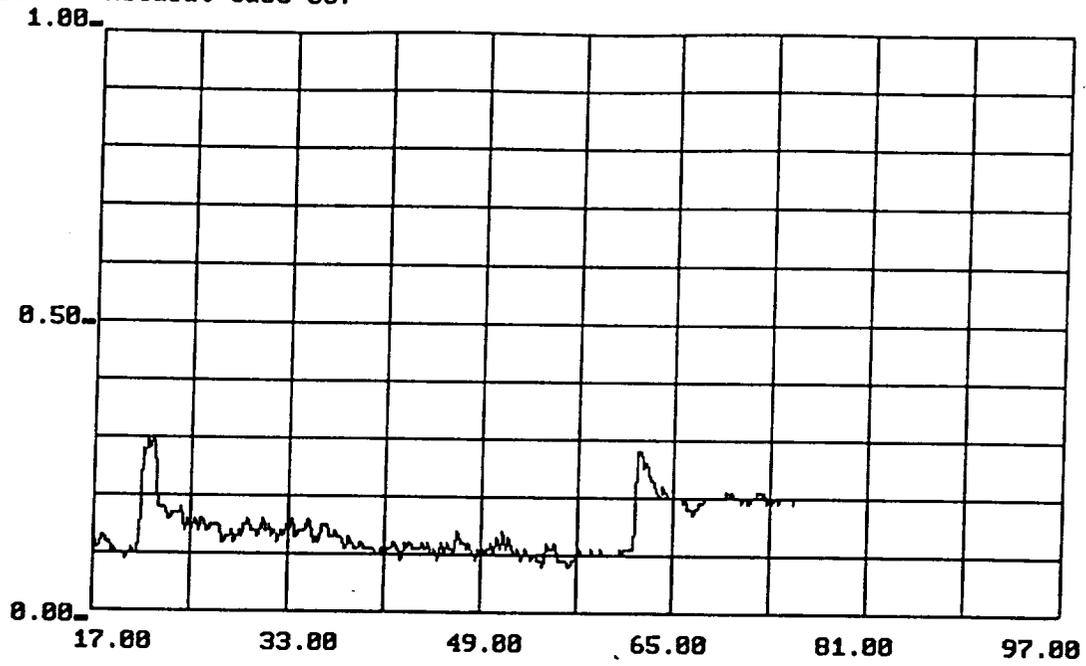


-2-

Figure A.3.14 Graph of Neural Net Output Unit 2, Holdout Case 307

Case 225 Unit

no387 Holdout Case 387

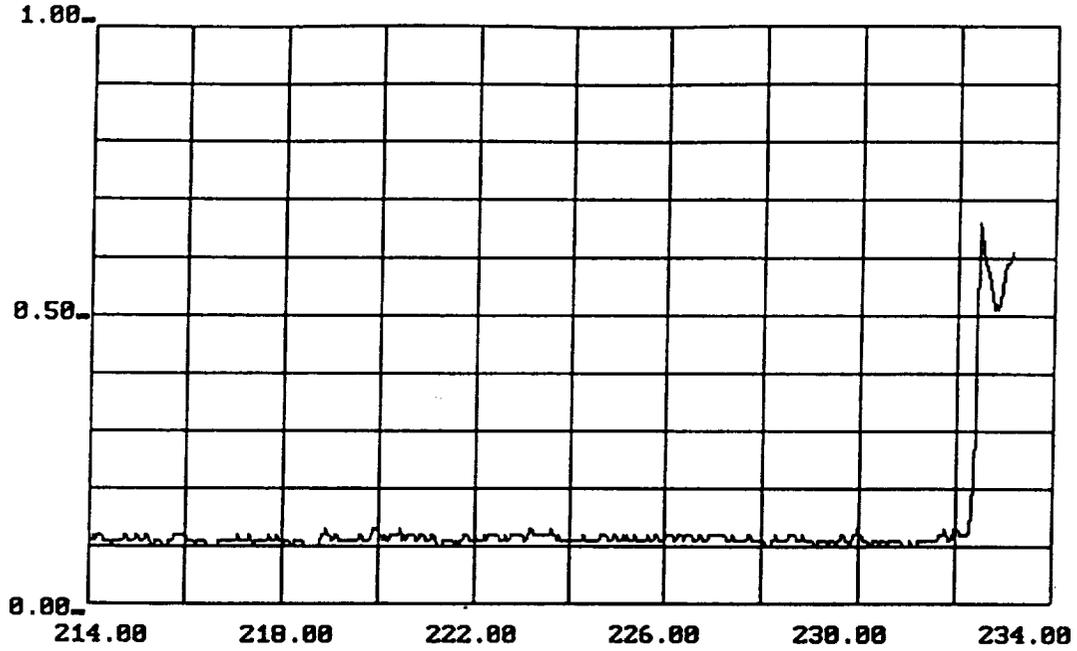


-4-

Figure A.3.15 Graph of Neural Net Output Unit 4, Holdout Case 307

Case 259 Unit

no331 No331 Holdout Case 331

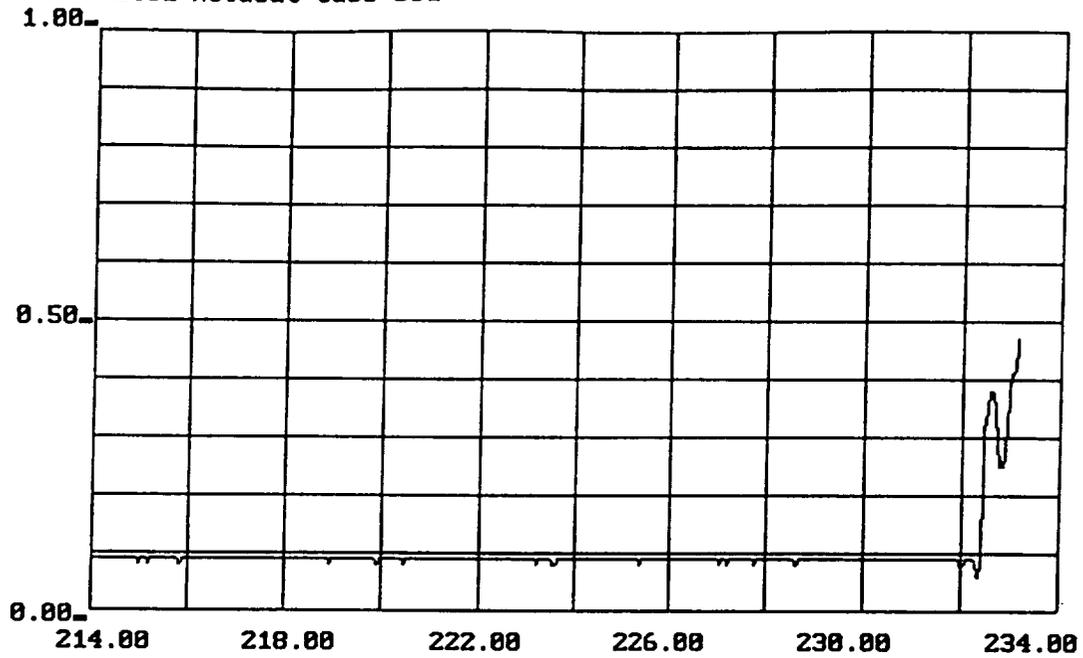


-8-

Figure A.3.16 Graph of Neural Net Output Unit 0, Holdout Case 331

Nominal/Fault Unit

no331 No331 Holdout Case 331

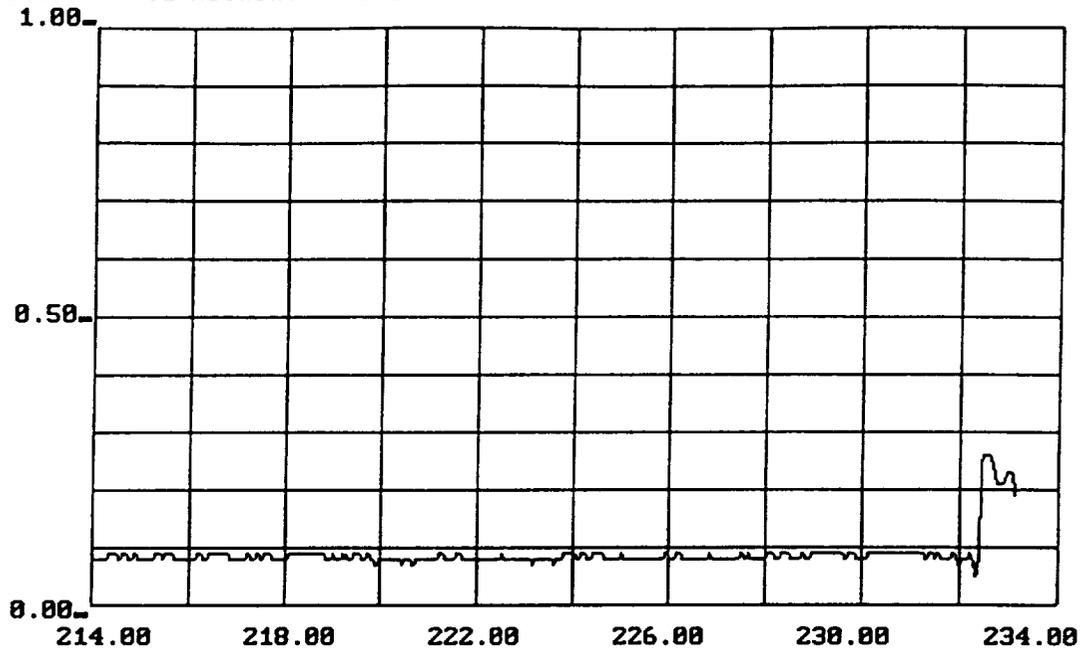


-3-

Figure A.3.17 Graph of Neural Net Output Unit 3, Holdout Case 331

Case 249 Unit

no331 No331 Holdout Case 331

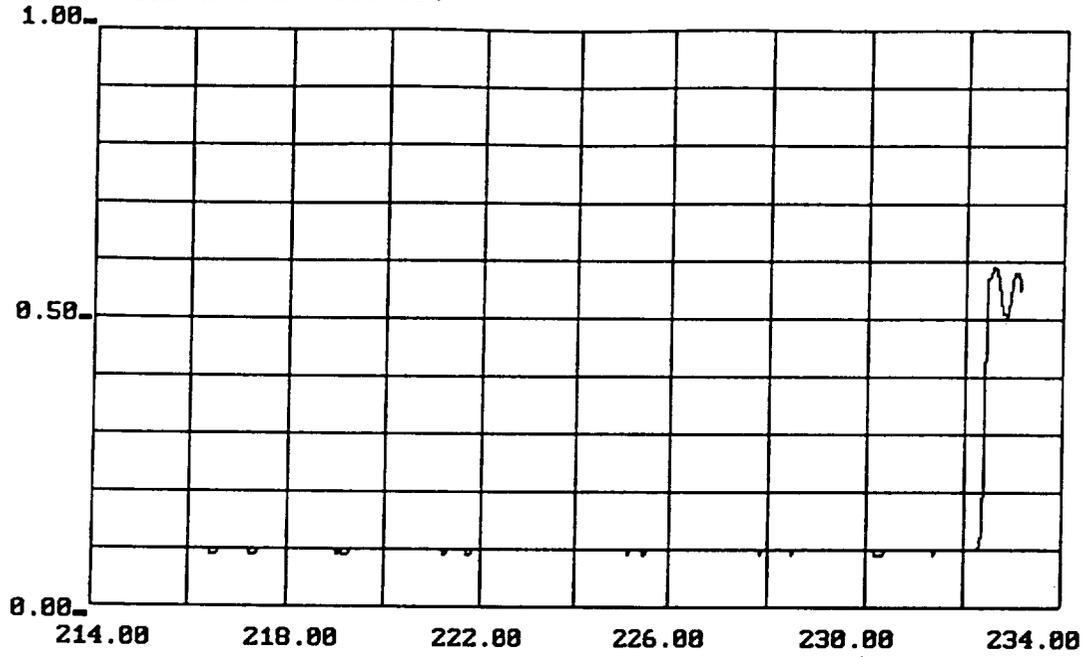


-4-

Figure A.3.18 Graph of Neural Net Output Unit 4, Holdout Case 331

Case 259 Unit

no331 No331 Holdout Case 331



-9-

Figure A.3.19 Graph of Neural Net Output Unit 9, Holdout Case 331

Case 436 Unit

no340 No340 Holdout Case 340

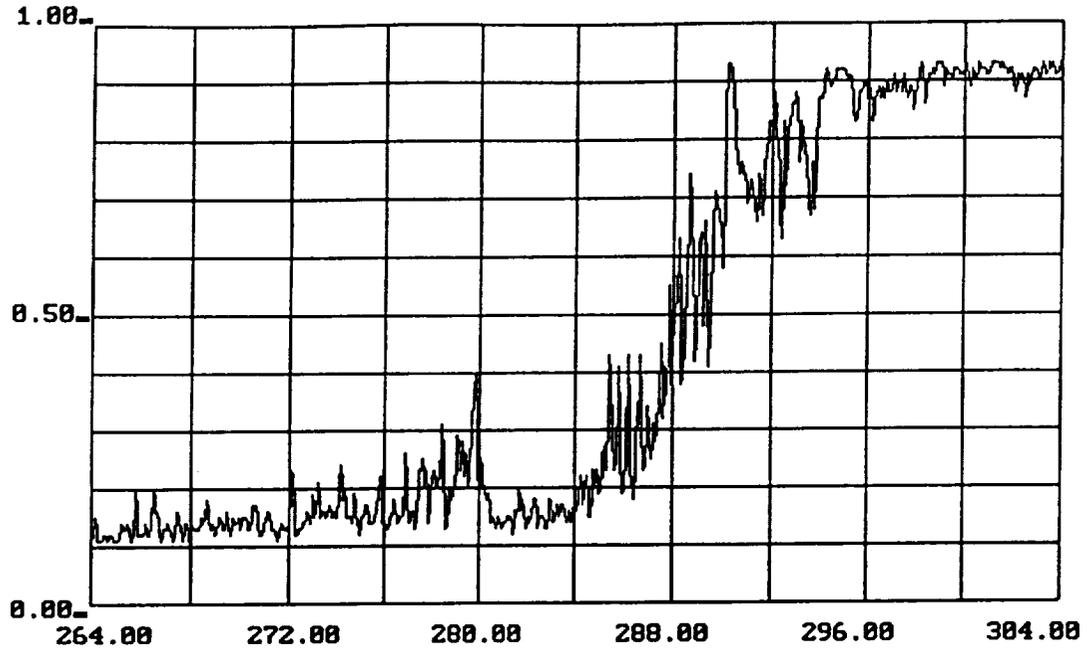
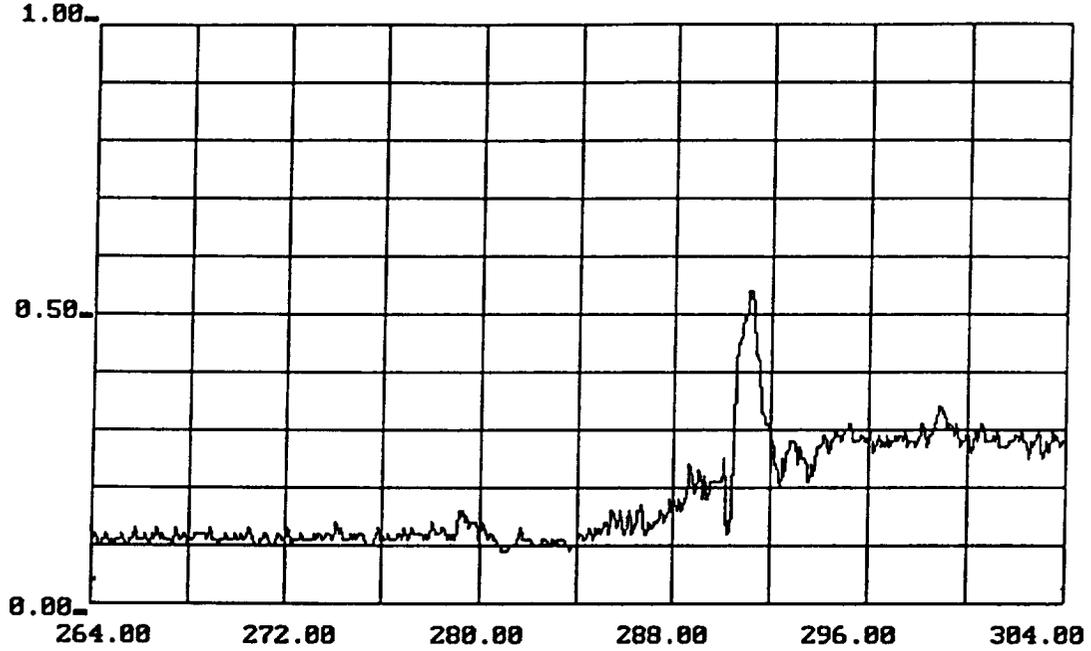


Figure A.3.20 Graph of Neural Net Output Unit 0, Holdout Case 340

Nominal/Fault Unit

no340 No340 Holdout Case 340

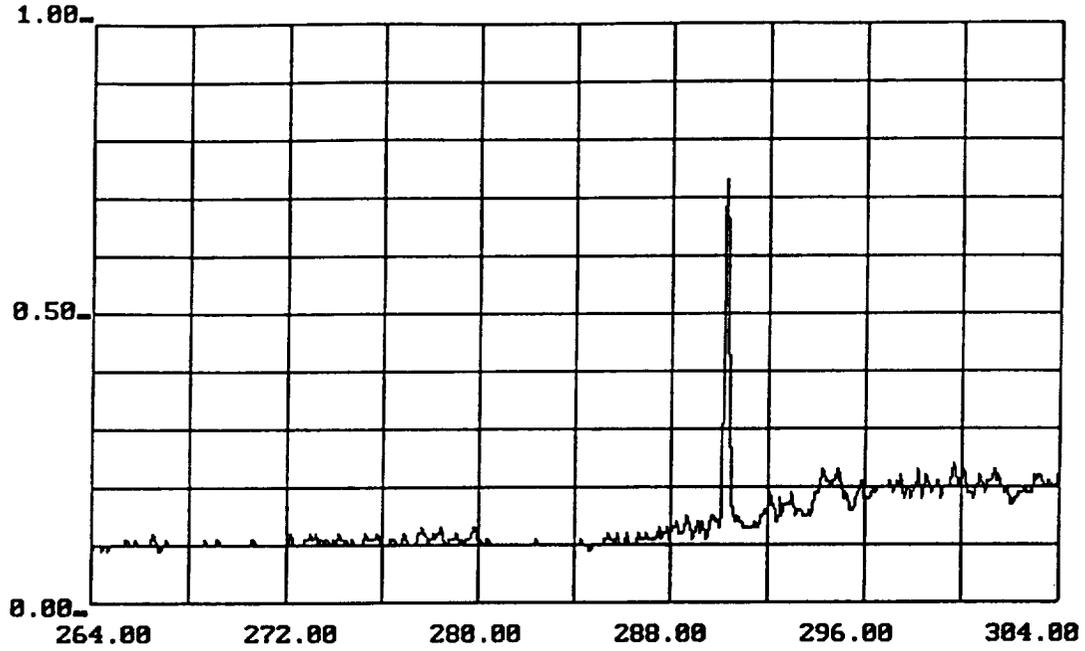


-2-

Figure A.3.21 Graph of Neural Net Output Unit 2, Holdout Case 340

Case 225 Unit

no340 No340 Holdout Case 340

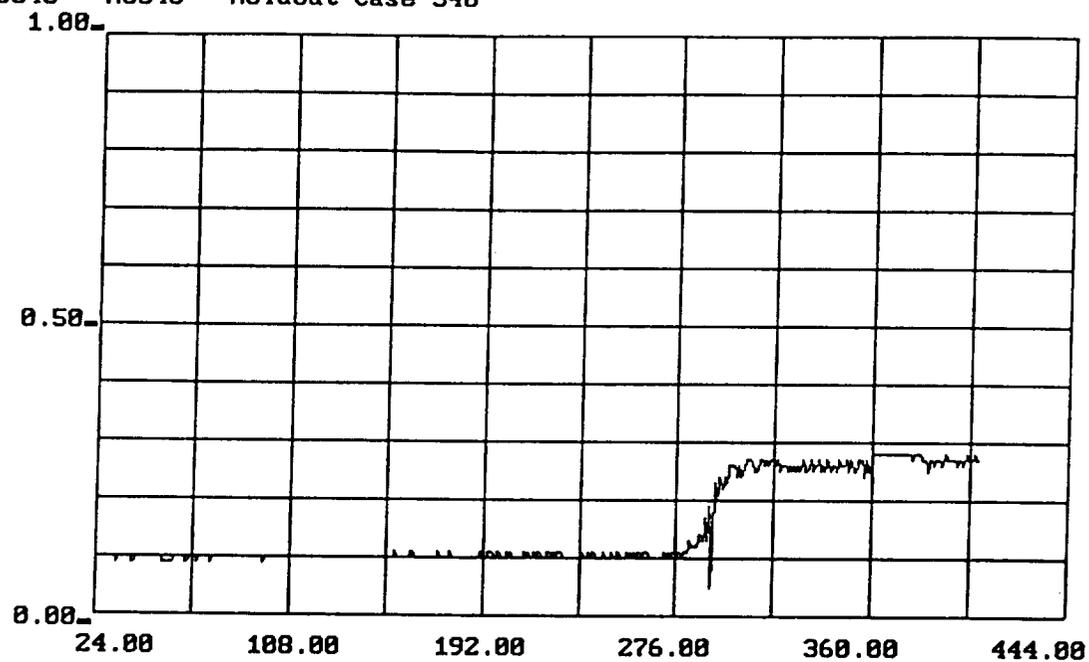


-3-

Figure A.3.22 Graph of Neural Net Output Unit 3, Holdout Case 340

Case 249 Unit

no340 No340 Holdout Case 340

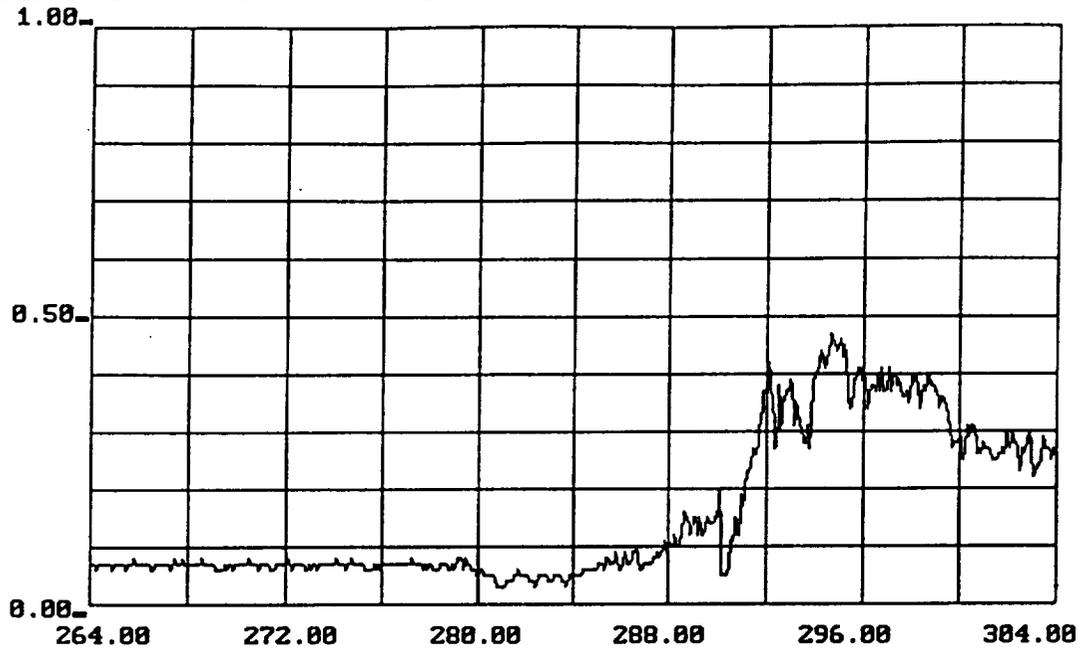


-5-

Figure A.3.23 Graph of Neural Net Output Unit 5, Holdout Case 340

Case 307 Unit

no340 No340 Holdout Case 340

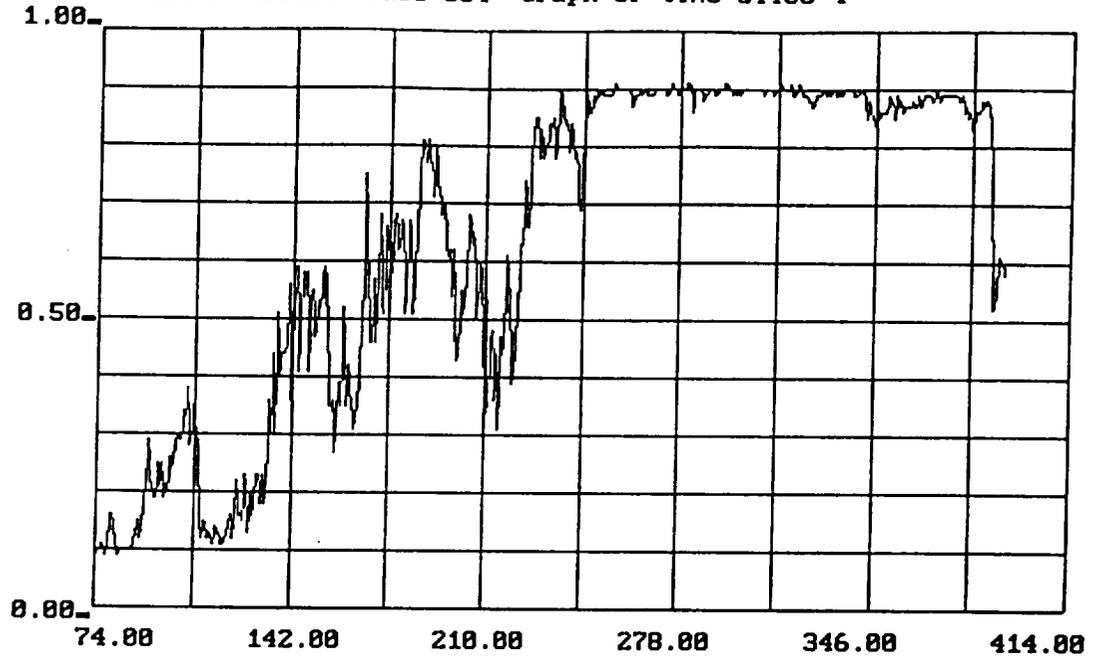


-8-

Figure A.3.24 Graph of Neural Net Output Unit 8, Holdout Case 340

Case 364 Unit

no364t4 No364 Holdout case 364 Graph of time slice 4

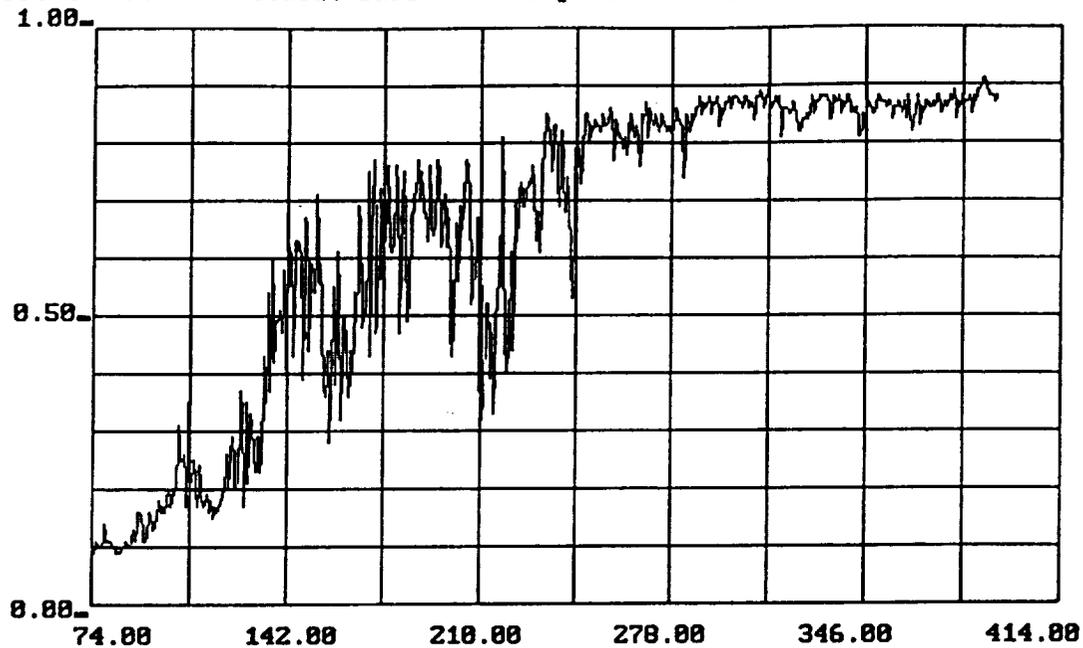


-8-

Figure A.3.25 Graph of Neural Net Output Unit 0, Holdout Case 364

Nominal/Fault Unit

no364t4 No364 Holdout case 364 Graph of time slice 4

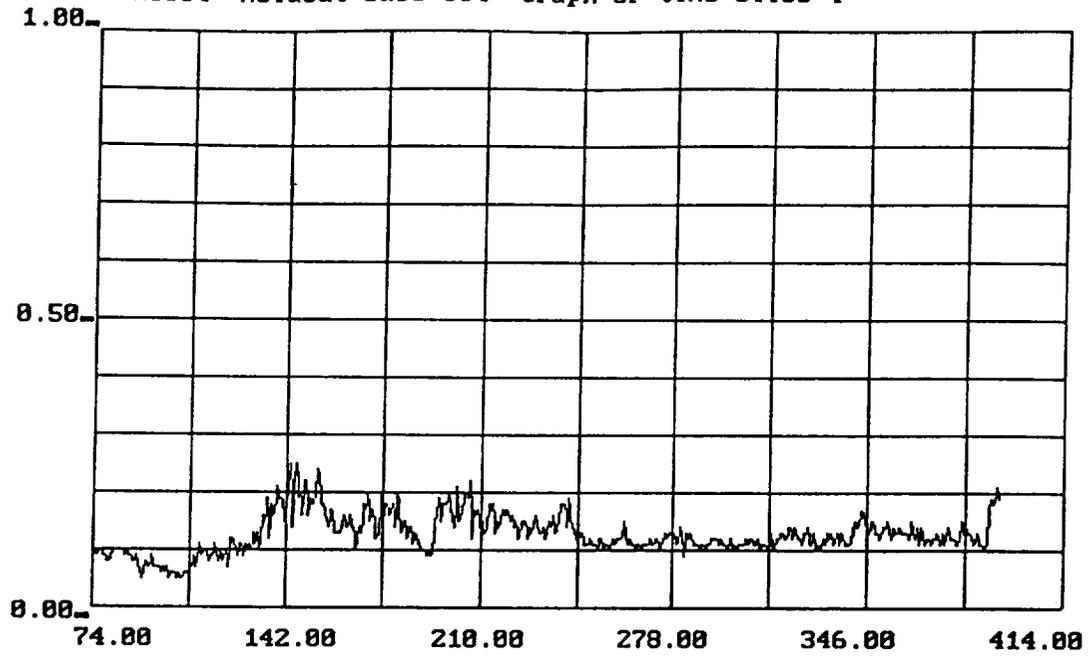


-3-

Figure A.3.26 Graph of Neural Net Output Unit 3, Holdout Case 364

Case 249 Unit

no364t4 No364 Holdout case 364 Graph of time slice 4

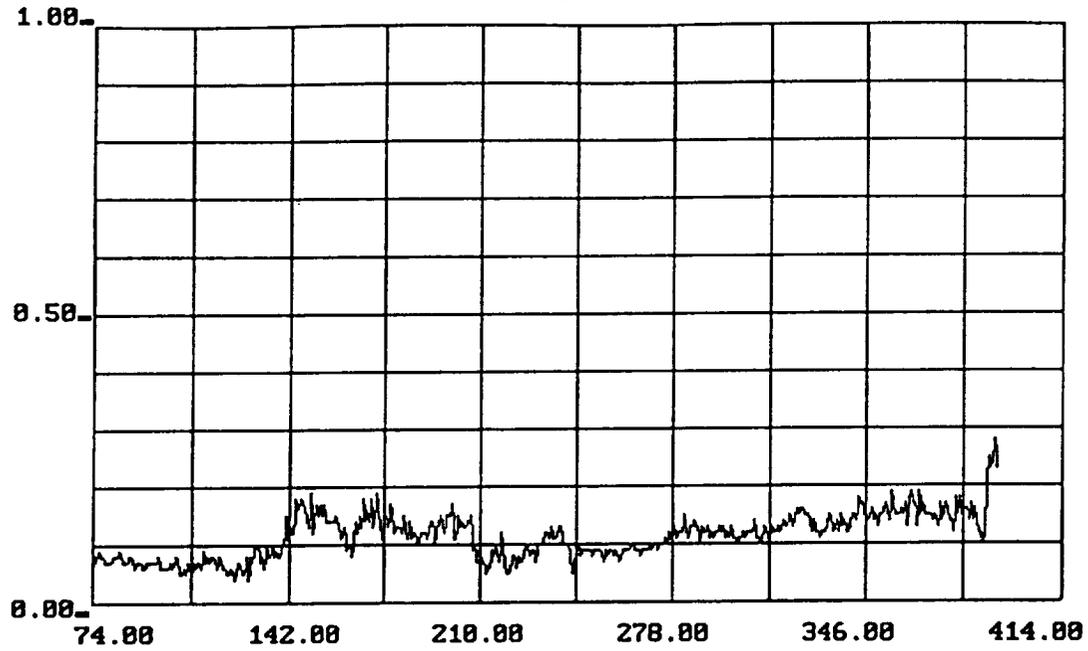


-4-

Figure A.3.27 Graph of Neural Net Output Unit 4, Holdout Case 364

Case 259 Unit

no364t4 No364 Holdout case 364 Graph of time slice 4

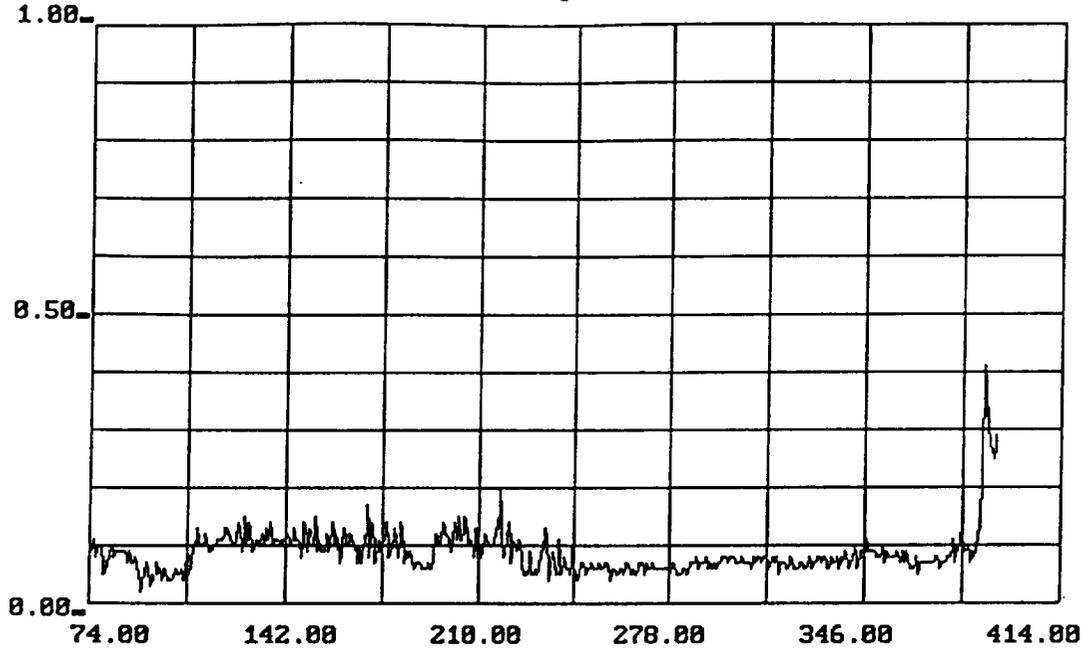


-7-

Figure A.3.28 Graph of Neural Net Output Unit 7, Holdout Case 364

Case 340 Unit

no364t4 No364 Holdout case 364 Graph of time slice 4

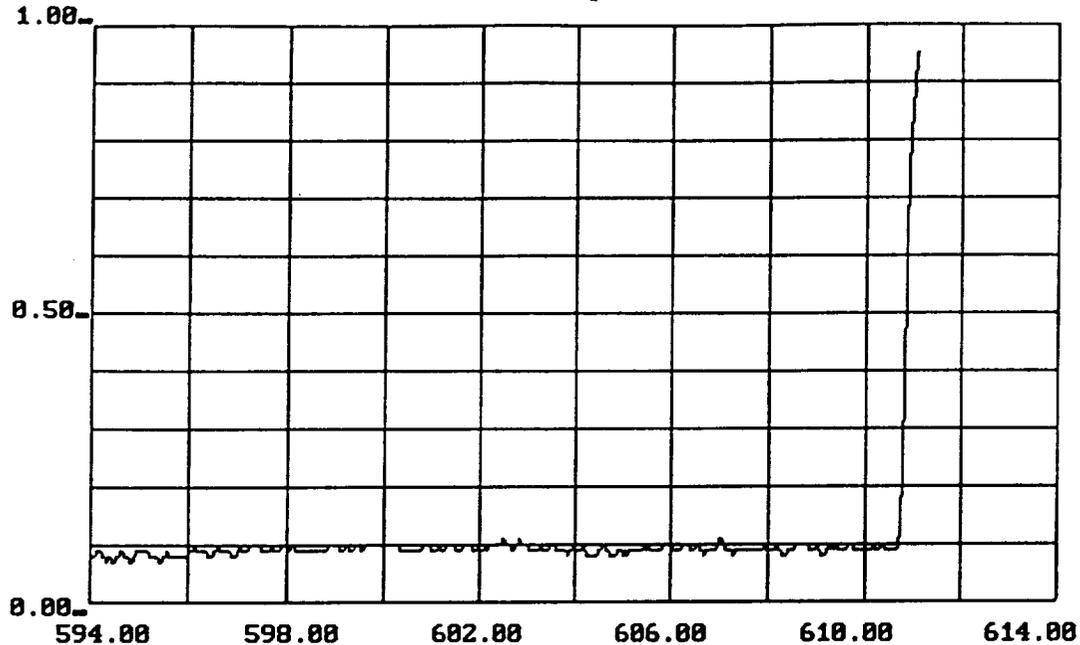


-9-

Figure A.3.29 Graph of Neural Net Output Unit 9, Holdout Case 364

Case 436 Unit

no436t2 No436 Holdout case 436 Graph of second time slice



-0-

Figure A.3.30 Graph of Neural Net Output Unit 0, Holdout Case 436

Nominal/Fault Unit

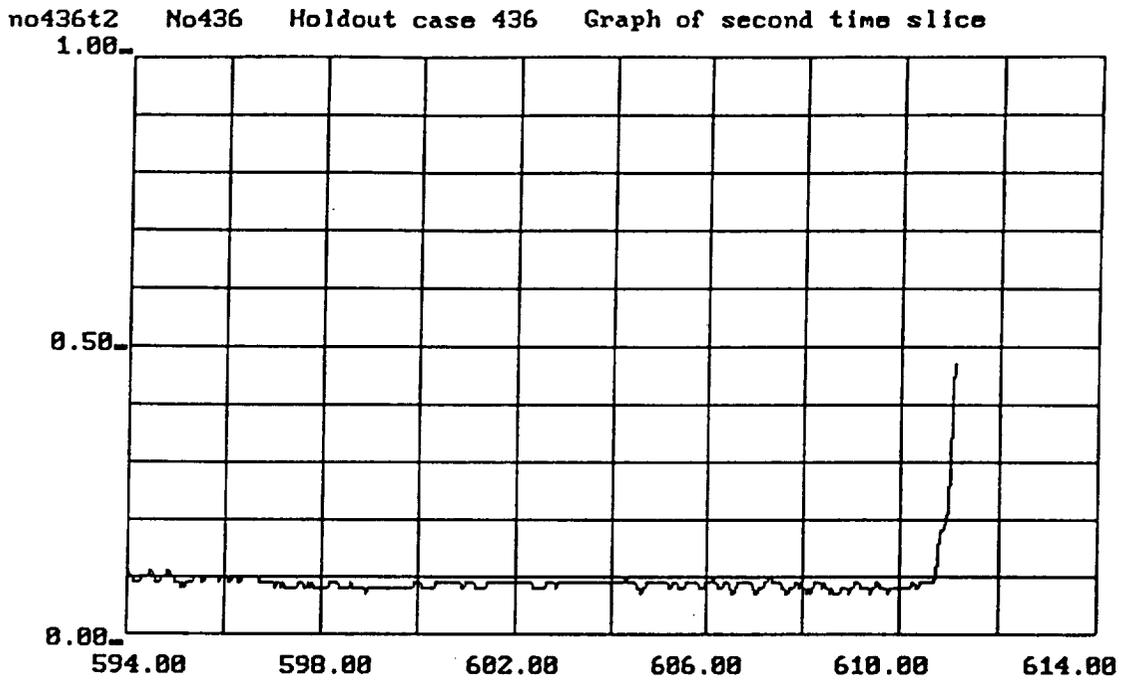
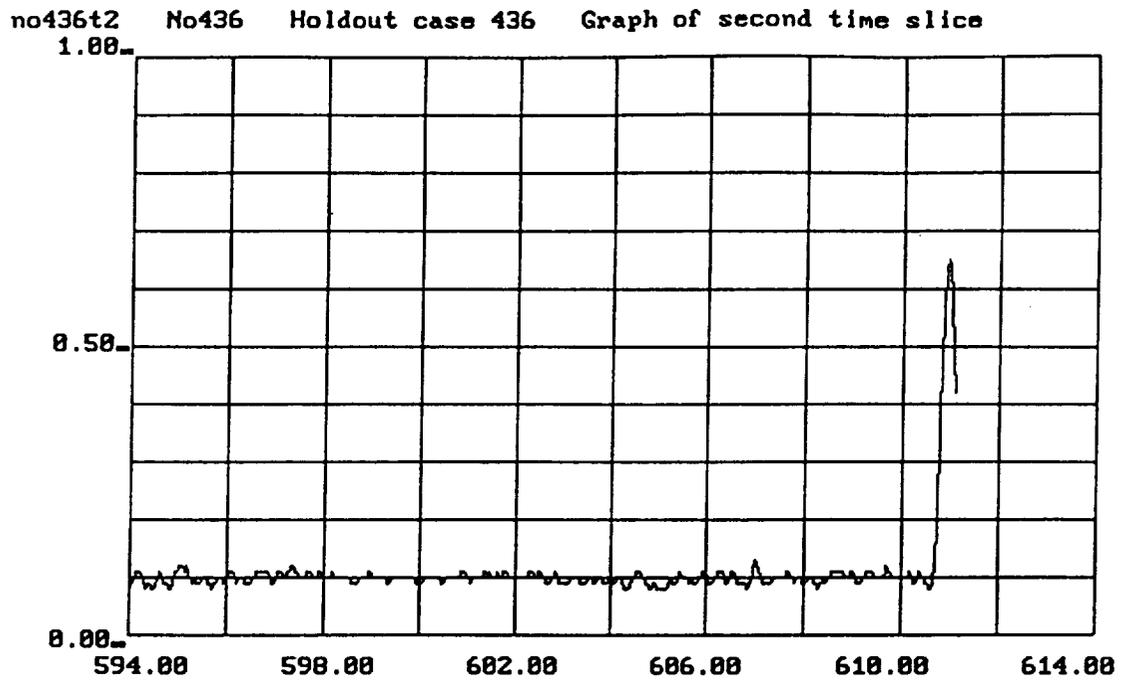


Figure A.3.31 Graph of Neural Net Output Unit 3, Holdout Case 436

Case 249 Unit

A-78

C-3



-6-

Figure A.3.32 Graph of Neural Net Output Unit 6, Holdout Case 436

Case 331 Unit

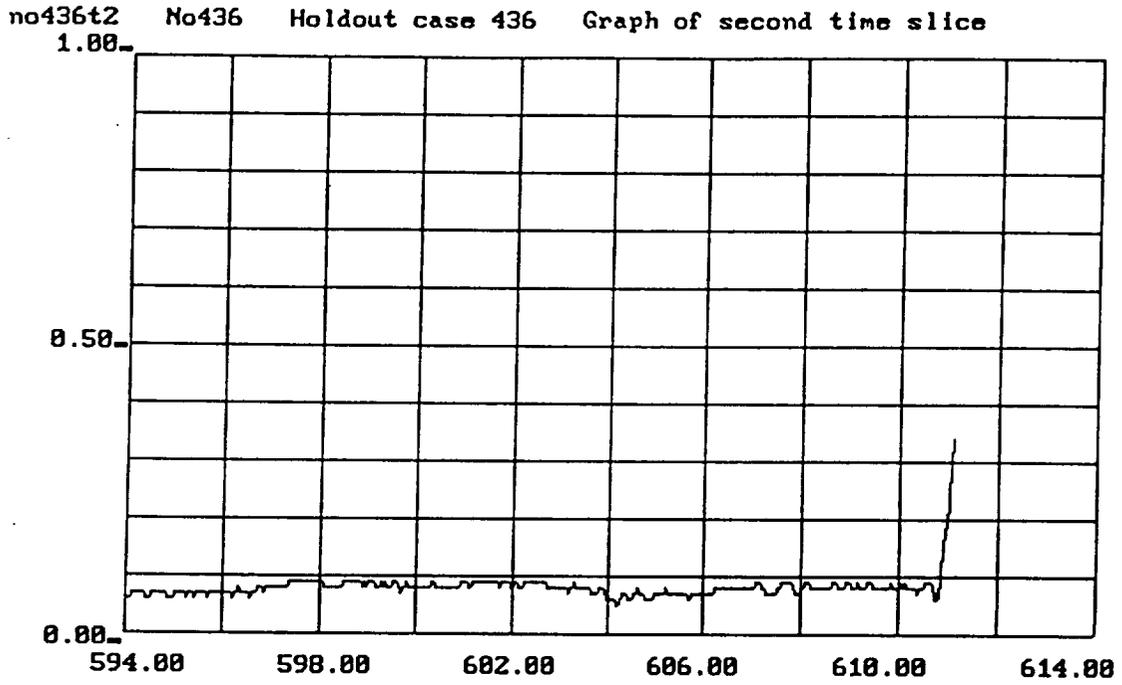


Figure A.3.33 Graph of Neural Net Output Unit 8, Holdout Case 436
Case 364 Unit

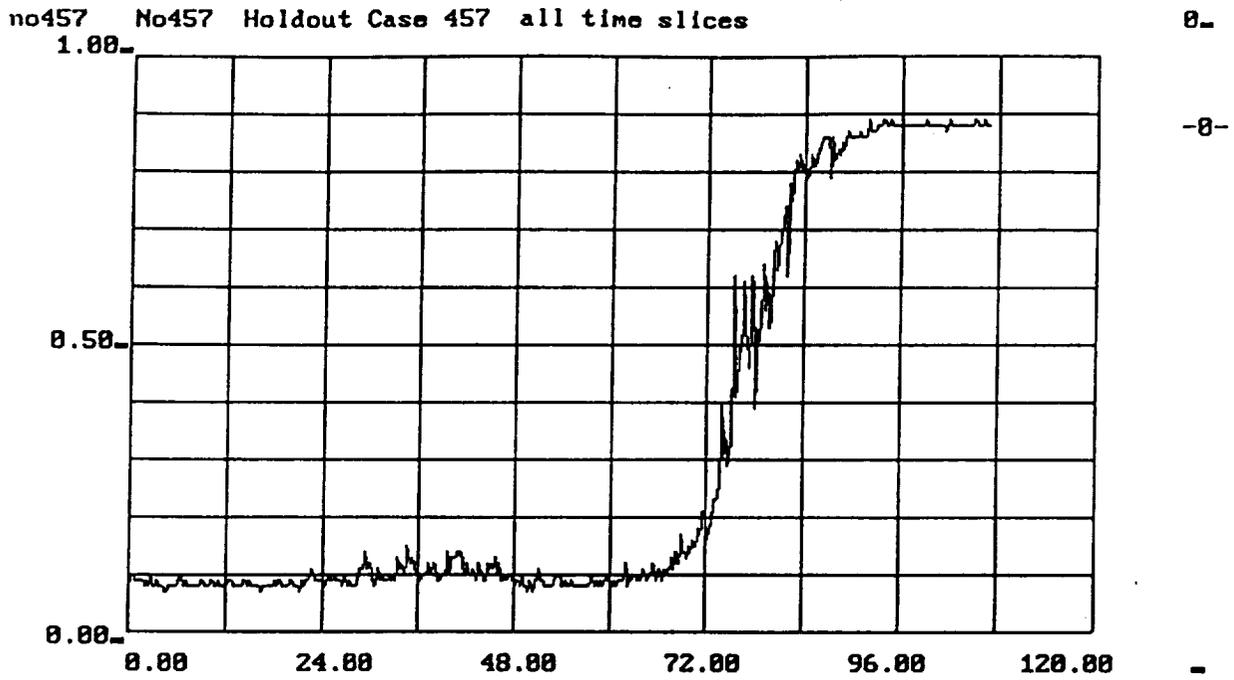
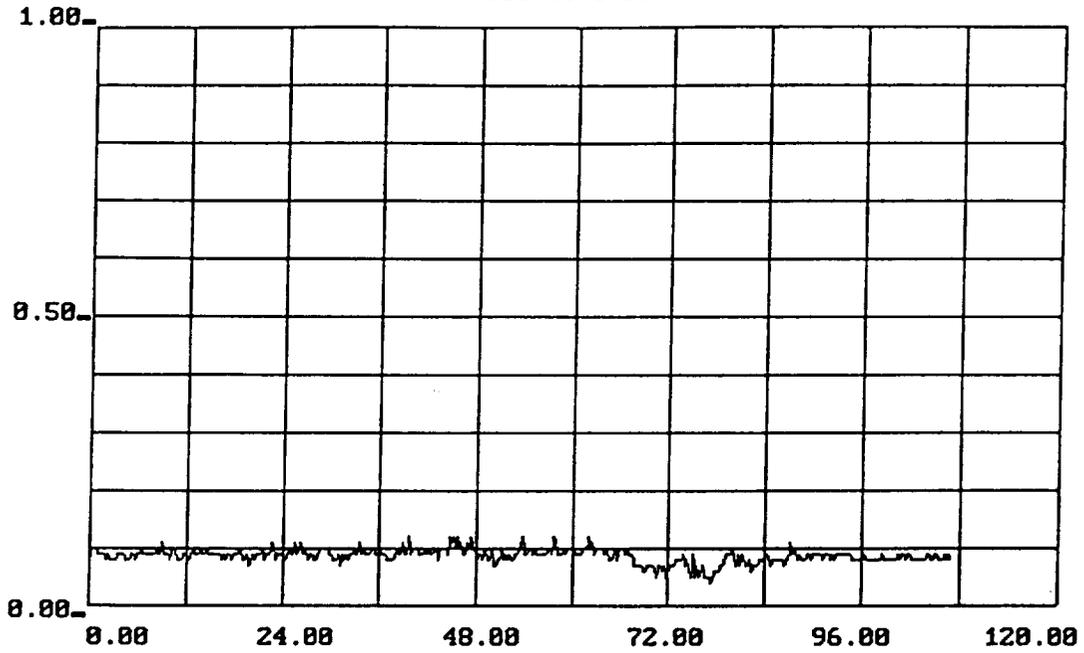


Figure A.3.34 Graph of Neural Net Output Unit 0, Holdout Case 457

Nominal/Fault Unit

no457 No457 Holdout Case 457 all time slices



0-

-2-

-

Figure A.3.35 Graph of Neural Net Output Unit 2, Holdout Case 457

Case 225 Unit

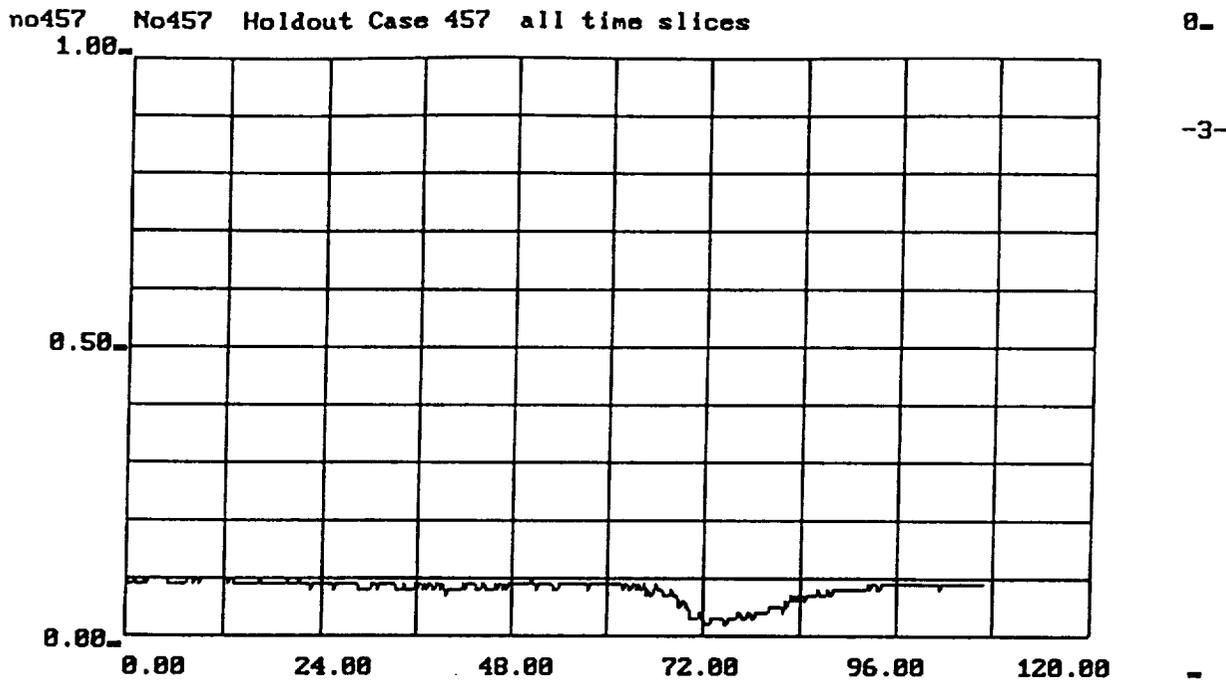


Figure A.3.36 Graph of Neural Net Output Unit 3, Holdout Case 457

Case 249 Unit

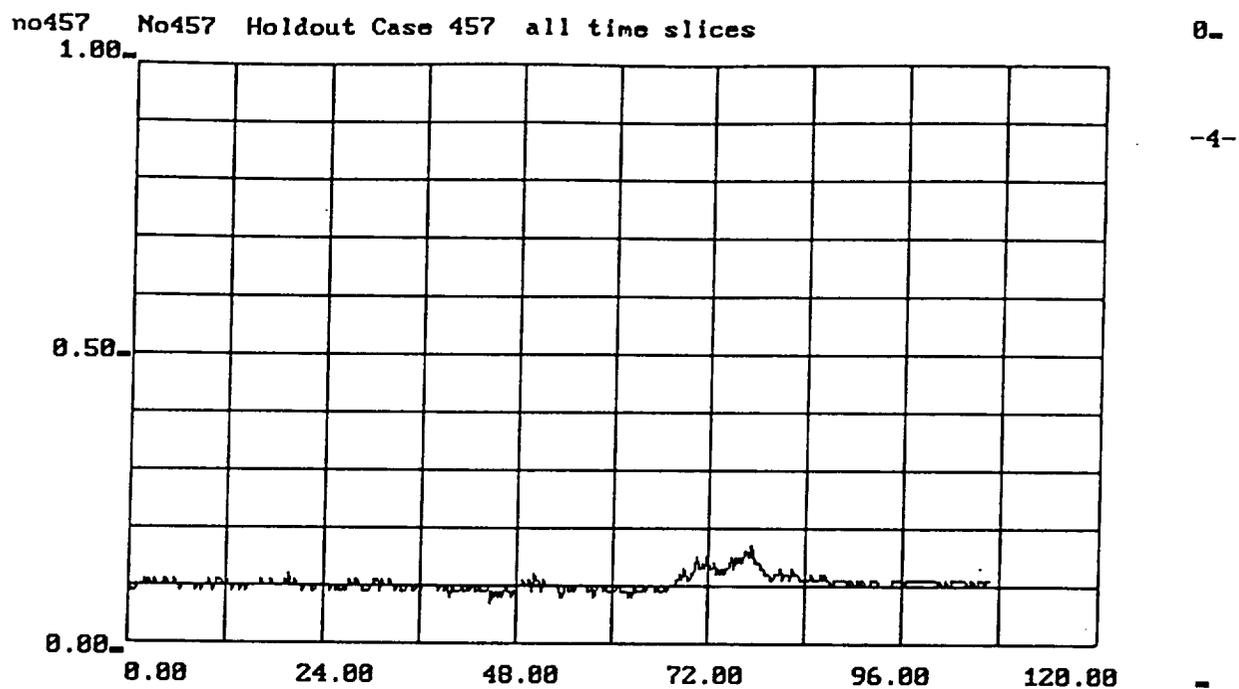


Figure A.3.37 Graph of Neural Net Output Unit 4, Holdout Case 457

Case 259 Unit

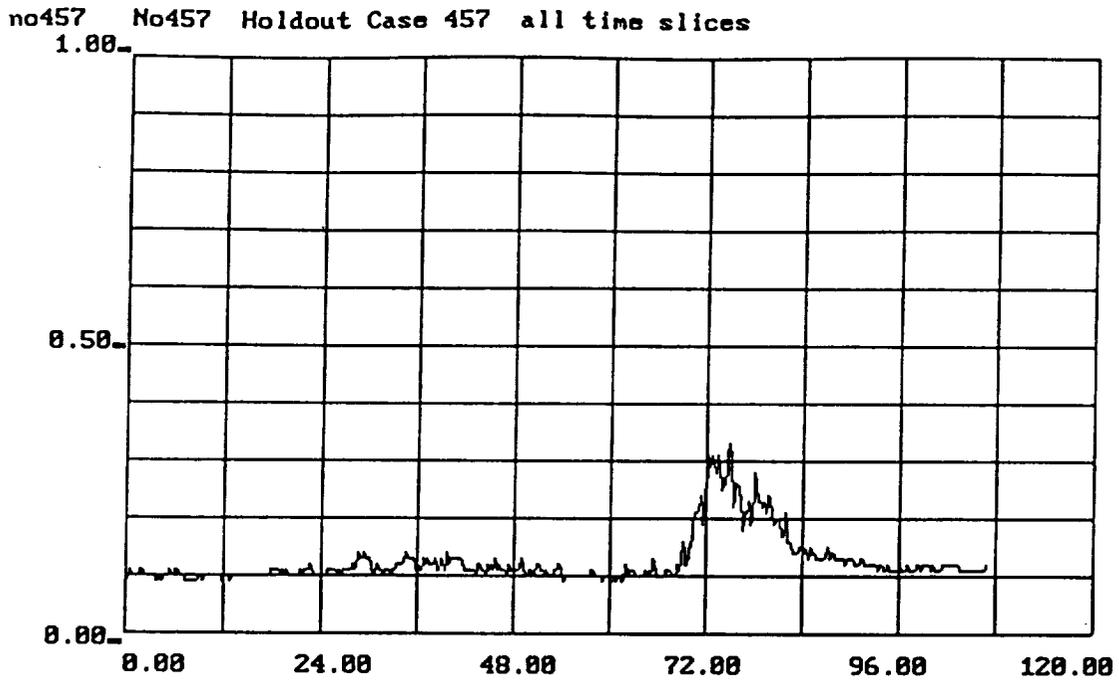


Figure A.3.38 Graph of Neural Net Output Unit 5, Holdout Case 457

Case 307 Unit

no457 No457 Holdout Case 457 all time slices

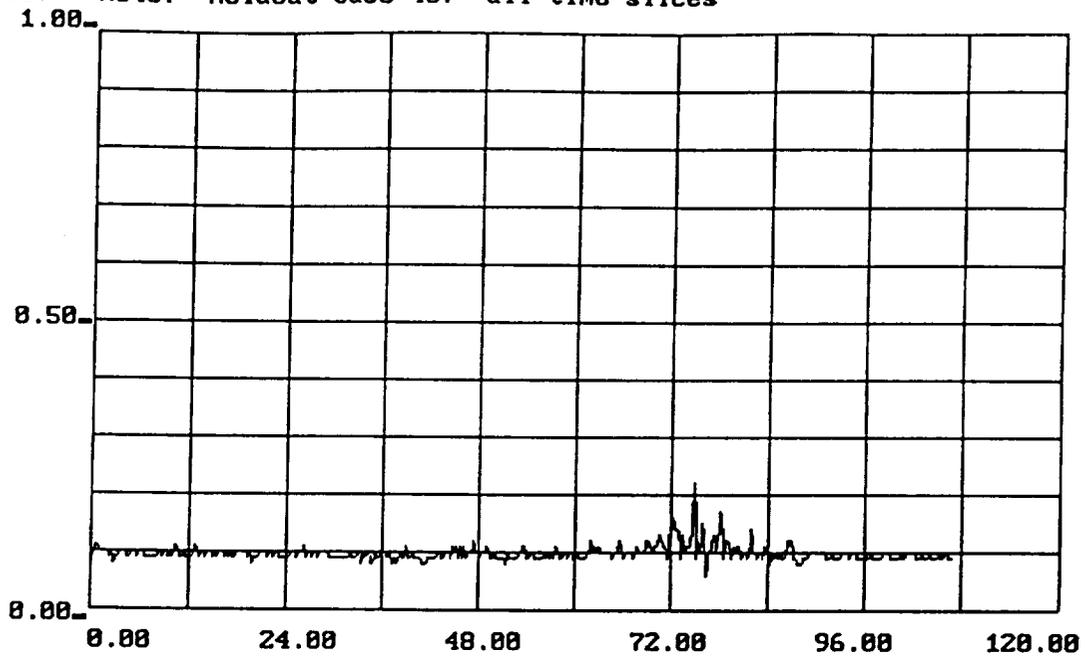


Figure A.3.39 Graph of Neural Net Output Unit 6, Holdout Case 457

Case 331 Unit

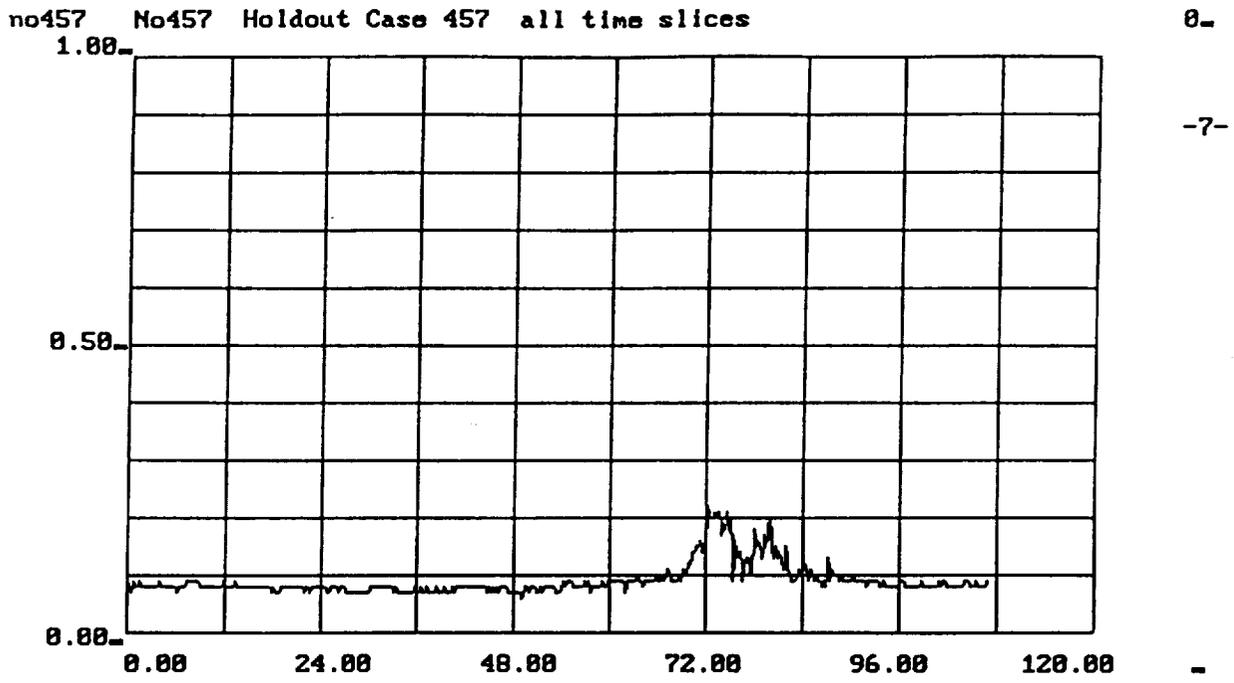


Figure A.3.40 Graph of Neural Net Output Unit 7, Holdout Case 457

Case 340 Unit

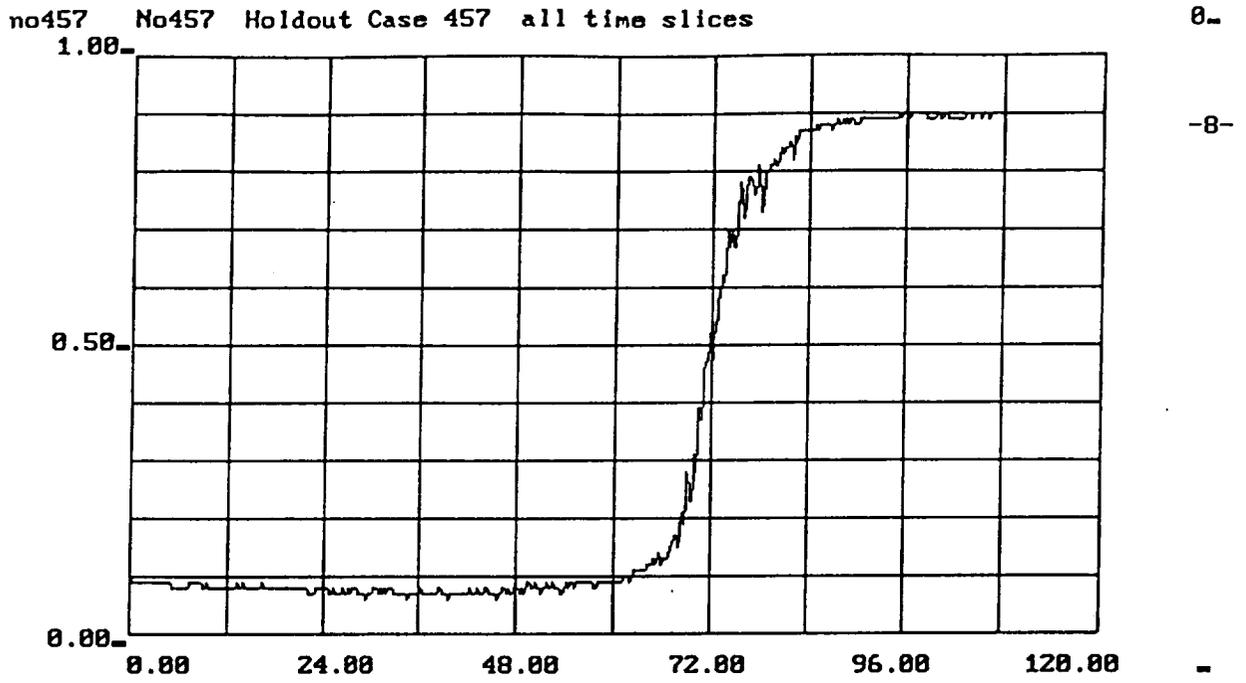


Figure A.3.41 Graph of Neural Net Output Unit 8, Holdout Case 457
Case 364 Unit

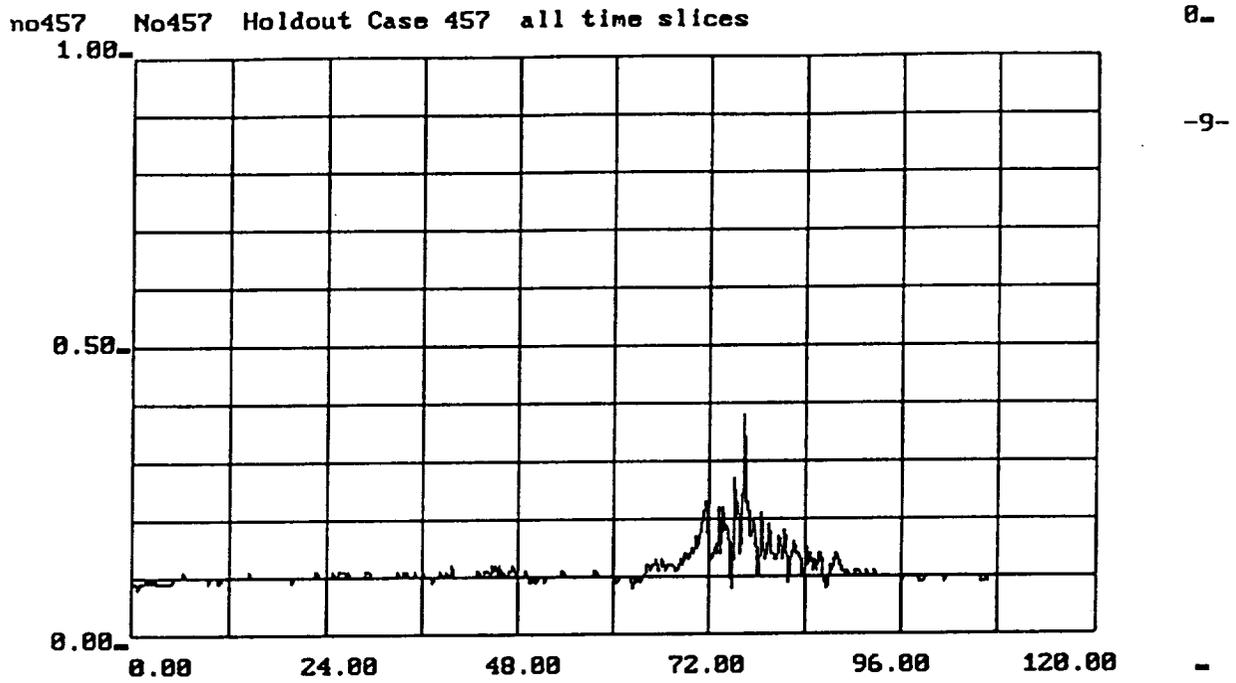
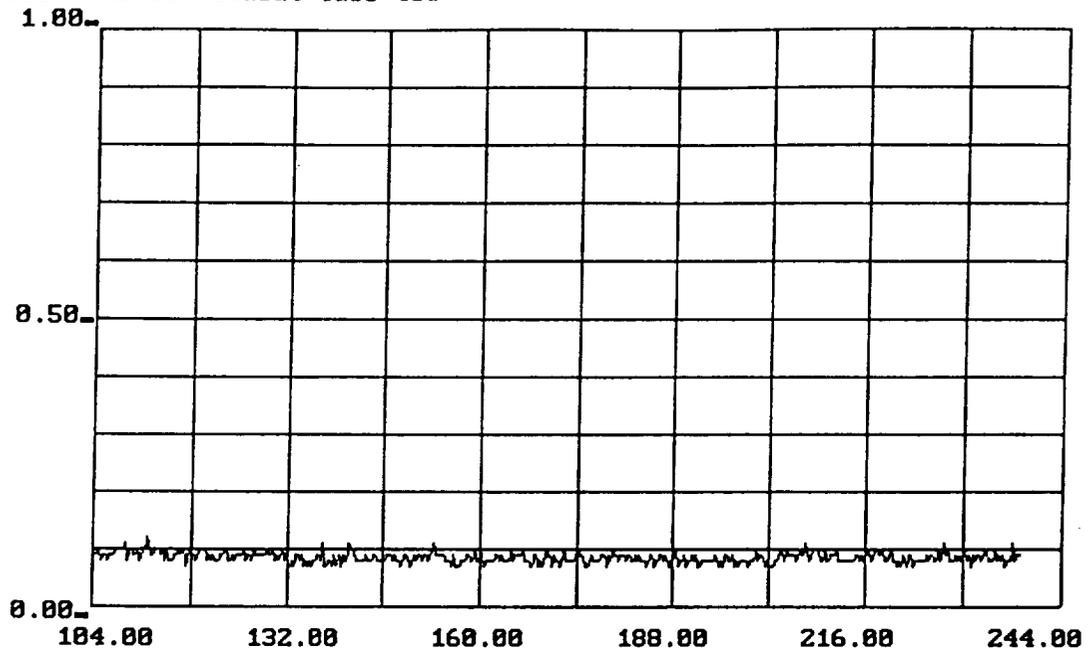


Figure A.3.42 Graph of Neural Net Output Unit 9, Holdout Case 457

Case 436 Unit

no463 No463 Holdout Case 463

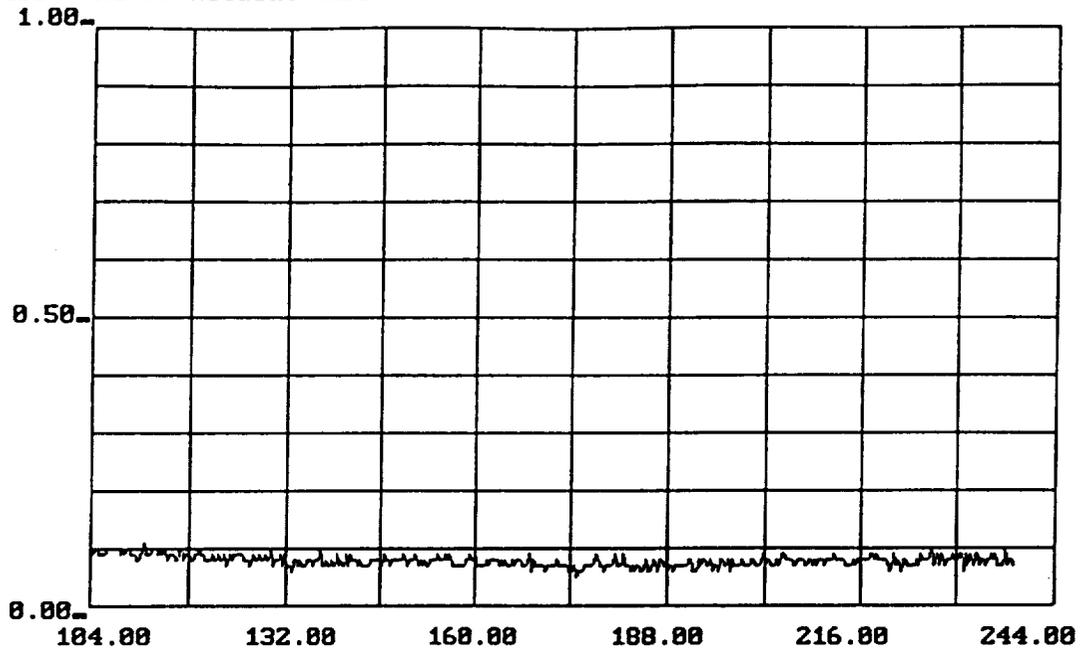


-8-

Figure A.3.43 Graph of Neural Net Output Unit 0, Holdout Case 463

Nominal/Fault Unit

no463 No463 Holdout Case 463

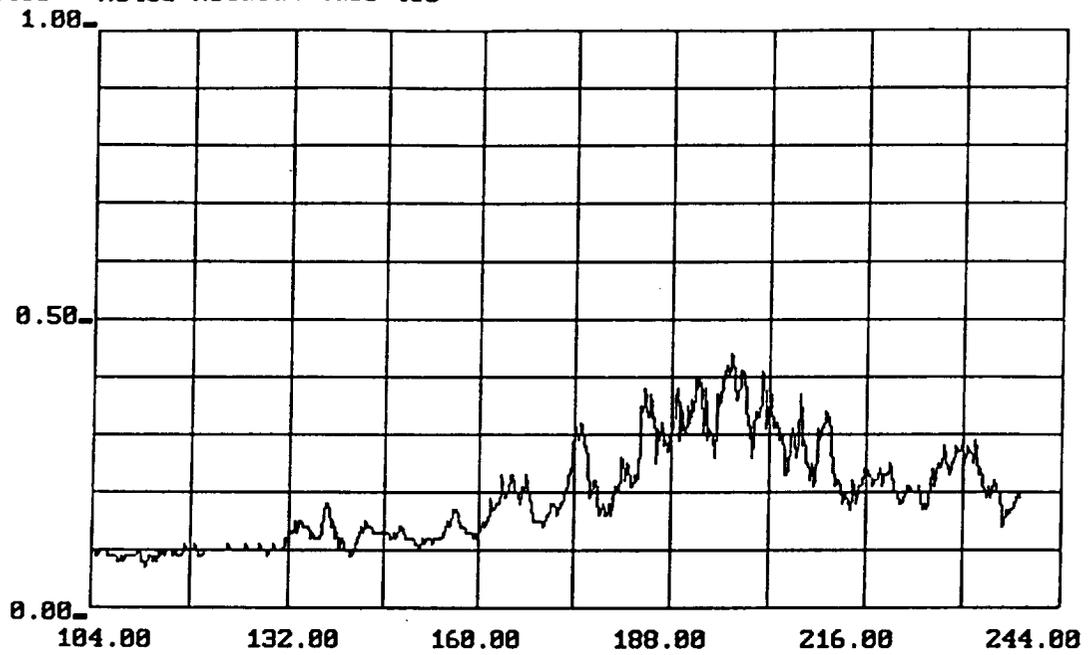


-2-

Figure A.3.44 Graph of Neural Net Output Unit 2, Holdout Case 463

Case 225 Unit

no463 No463 Holdout Case 463



-3-

Figure A.3.45 Graph of Neural Net Output Unit 3, Holdout Case 463

Case 249 Unit

no463 No463 Holdout Case 463

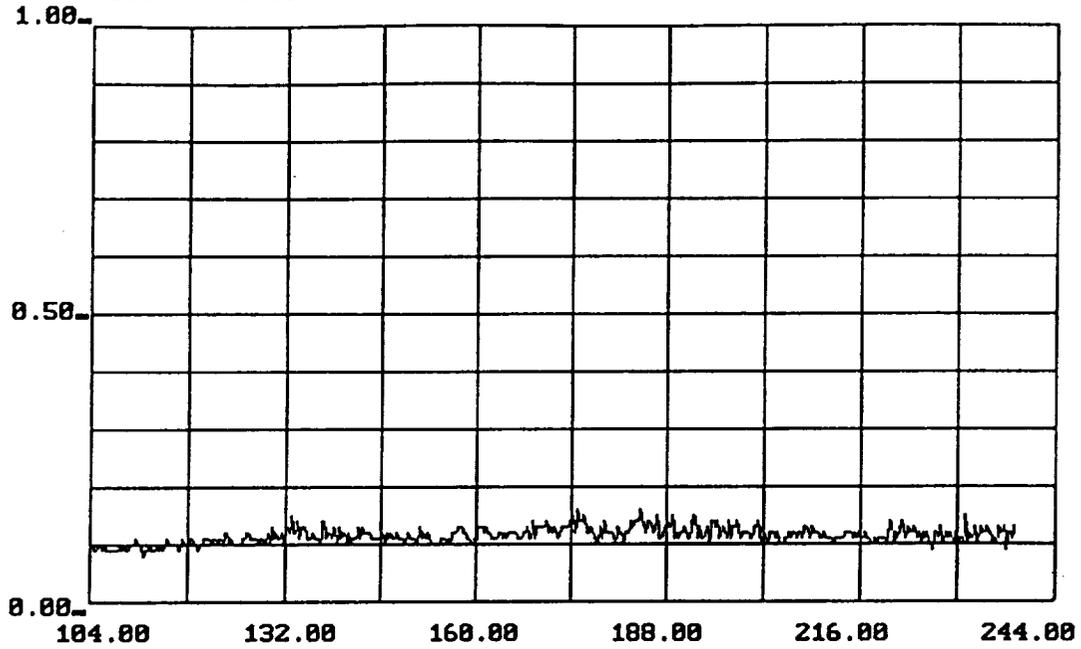
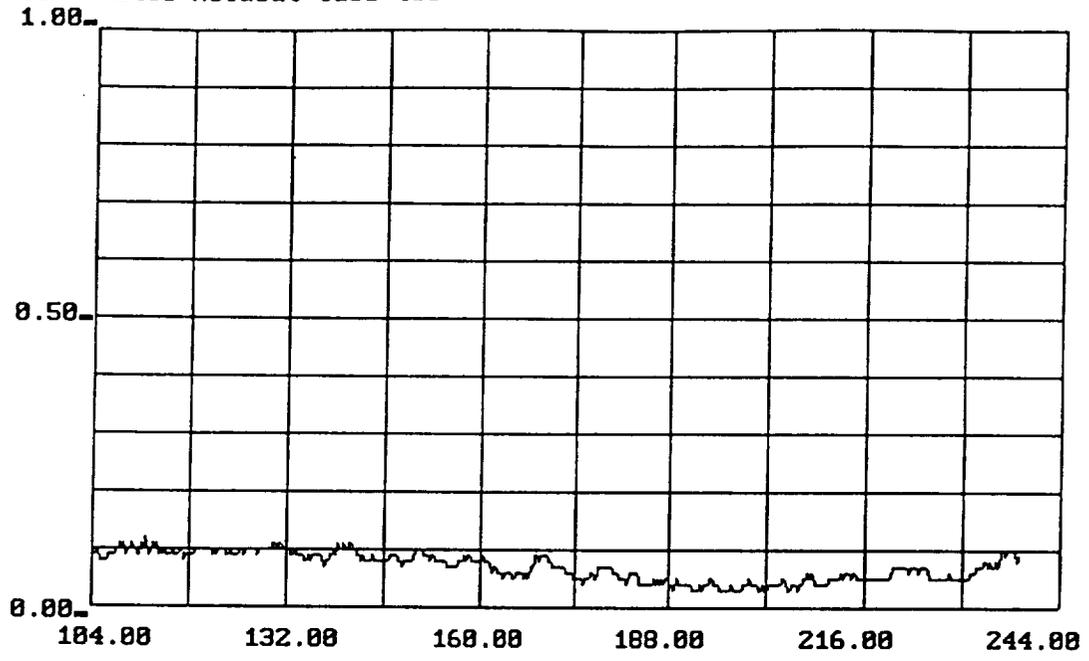


Figure A.3.46 Graph of Neural Net Output Unit 4, Holdout Case 463

Case 259 Unit

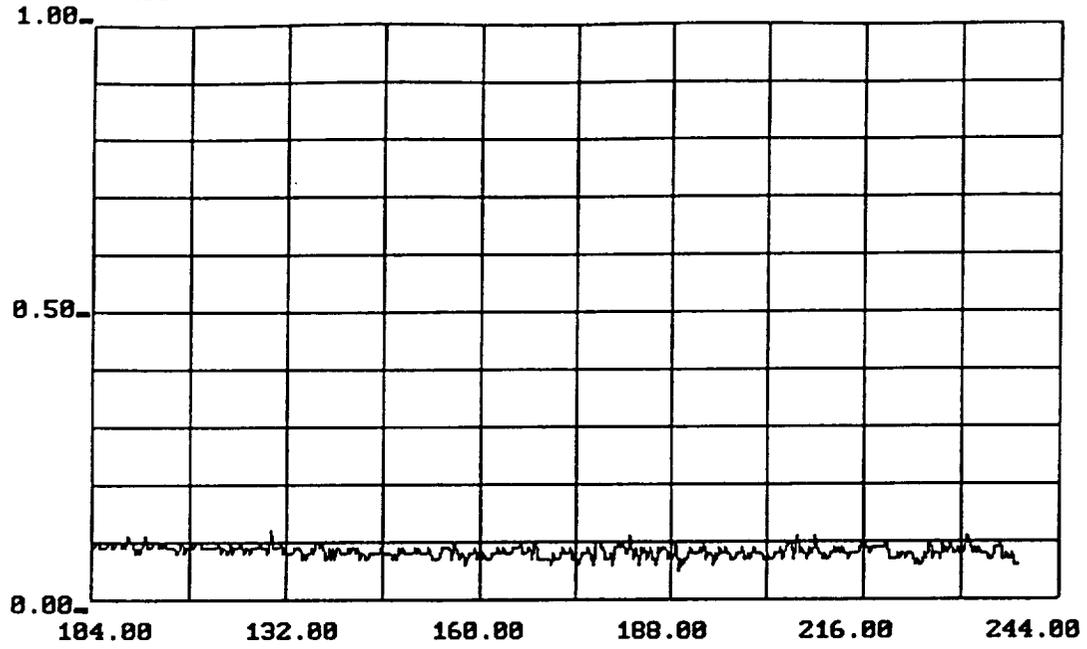
no463 No463 Holdout Case 463



-5-

Figure A.3.47 Graph of Neural Net Output Unit 5, Holdout Case 463
Case 307 Unit

no463 No463 Holdout Case 463

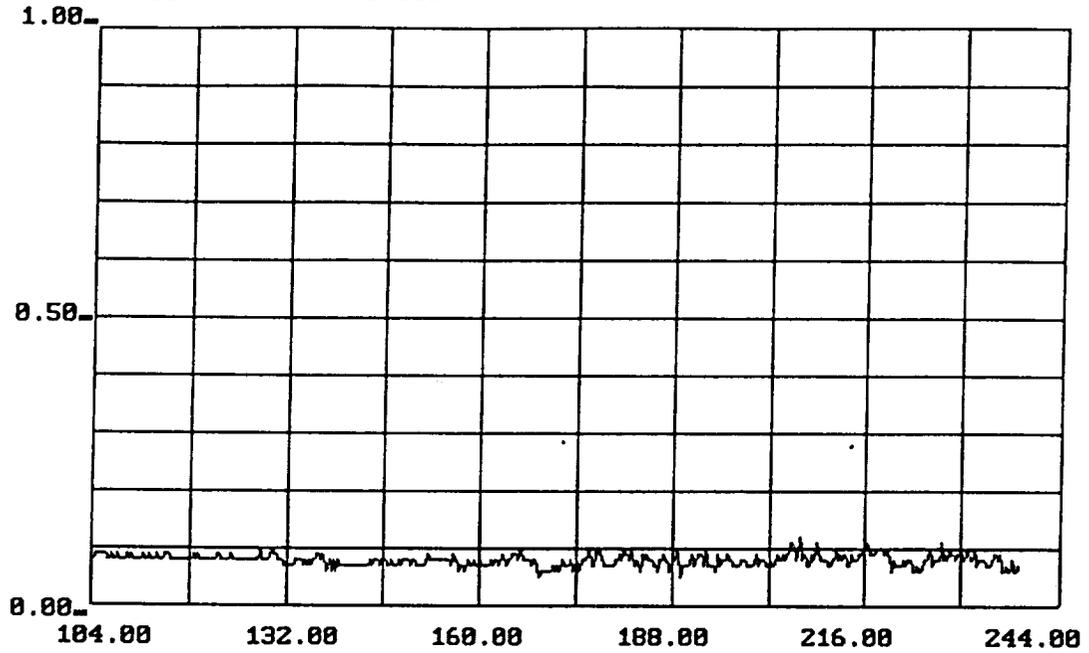


-6-

Figure A.3.48 Graph of Neural Net Output Unit 6, Holdout Case 463

Case 331 Unit

no463 No463 Holdout Case 463

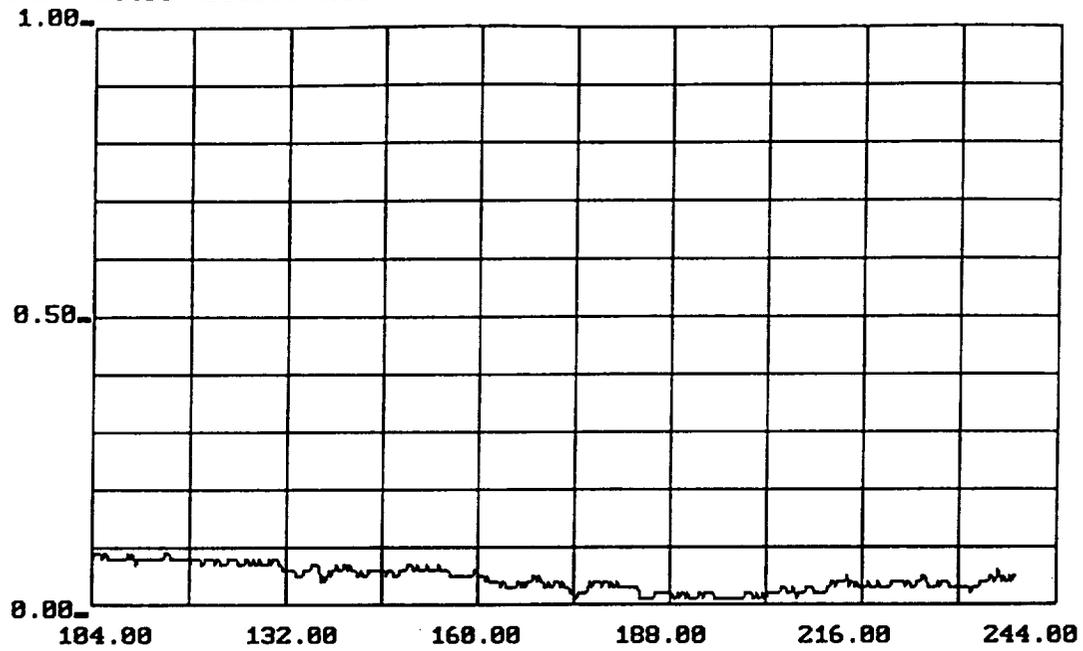


-7-

Figure A.3.49 Graph of Neural Net Output Unit 7, Holdout Case 463

Case 340 Unit

no463 No463 Holdout Case 463



-8-

Figure A.3.50 Graph of Neural Net Output Unit 8, Holdout Case 463

Case 364 Unit

no457 N0457t2 Pid 209 = 0

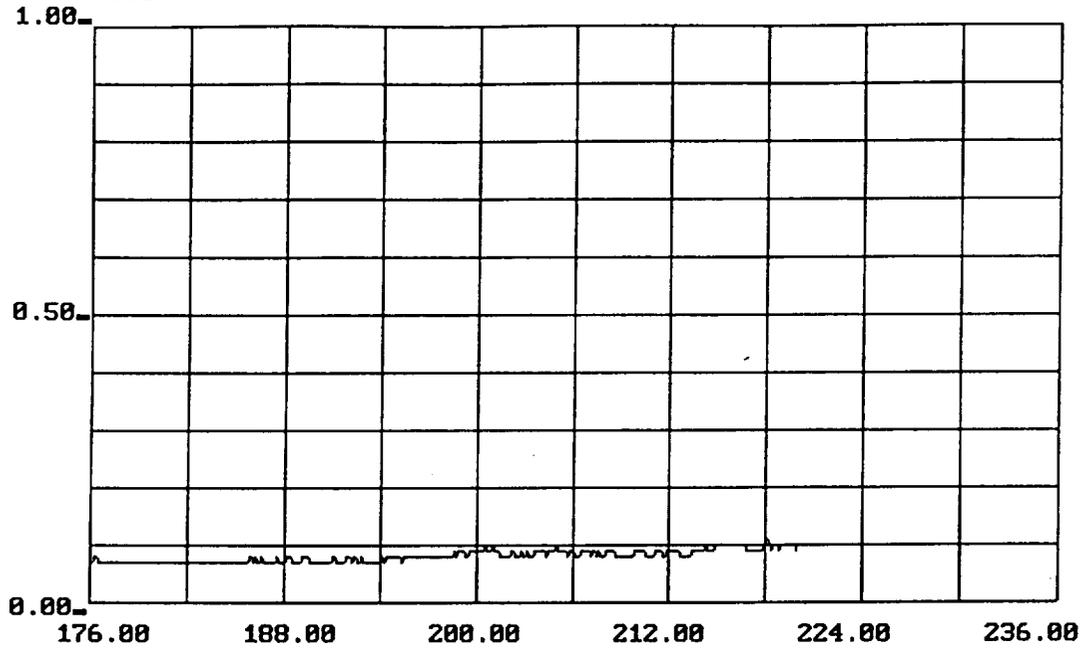


Figure A.4.1 Graph of Neural Net Output Unit 0, Holdout Case 457

PID 209 Features Replaced with Zeros

Nominal/Fault Unit

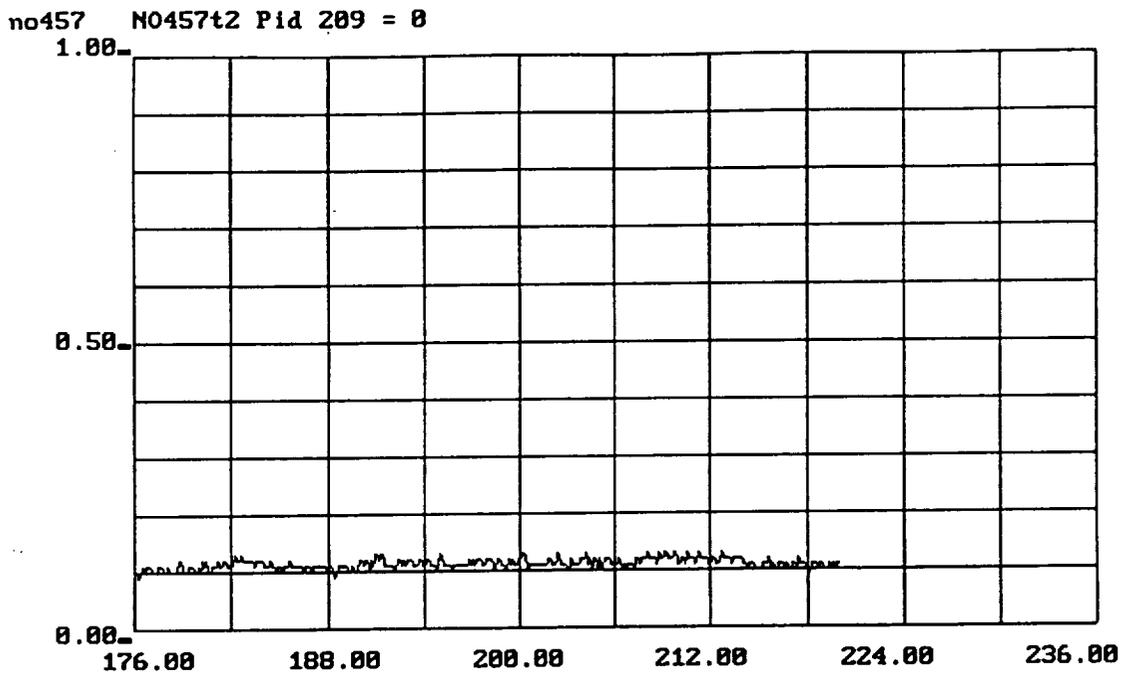
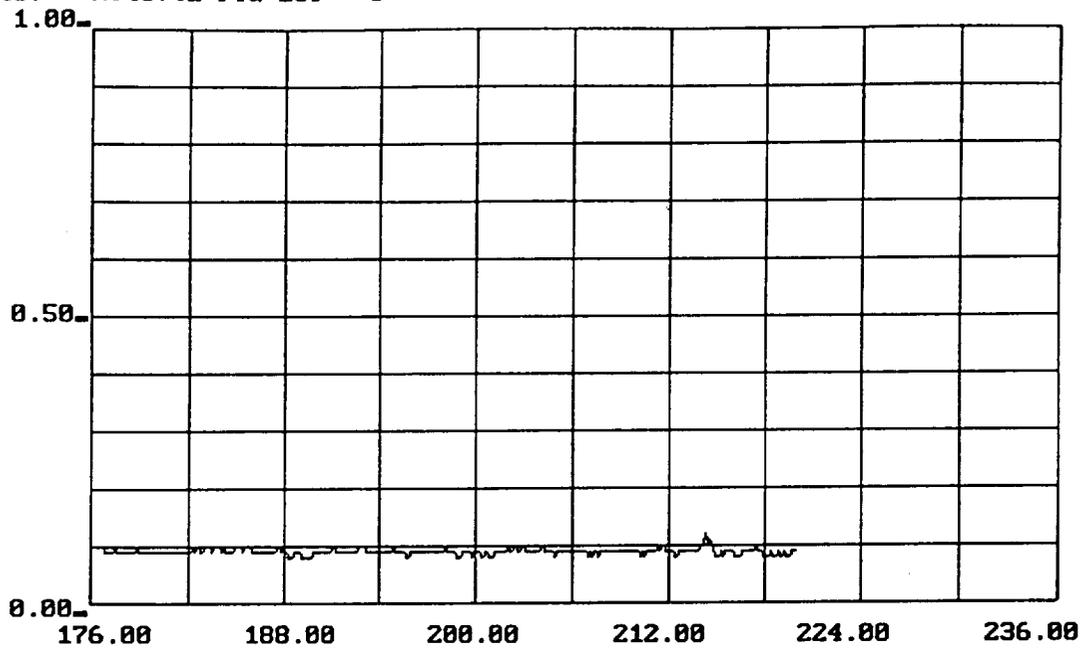


Figure A.4.2 Graph of Neural Net Output Unit 4, Holdout Case 457

PID 209 Features Replaced with Zeros

Case 259 Unit

no457 N0457t2 Pid 209 = 0



-5-

Figure A.4.3 Graph of Neural Net Output Unit 5, Holdout Case 457

PID 209 Features Replaced with Zeros

Case 307 Unit

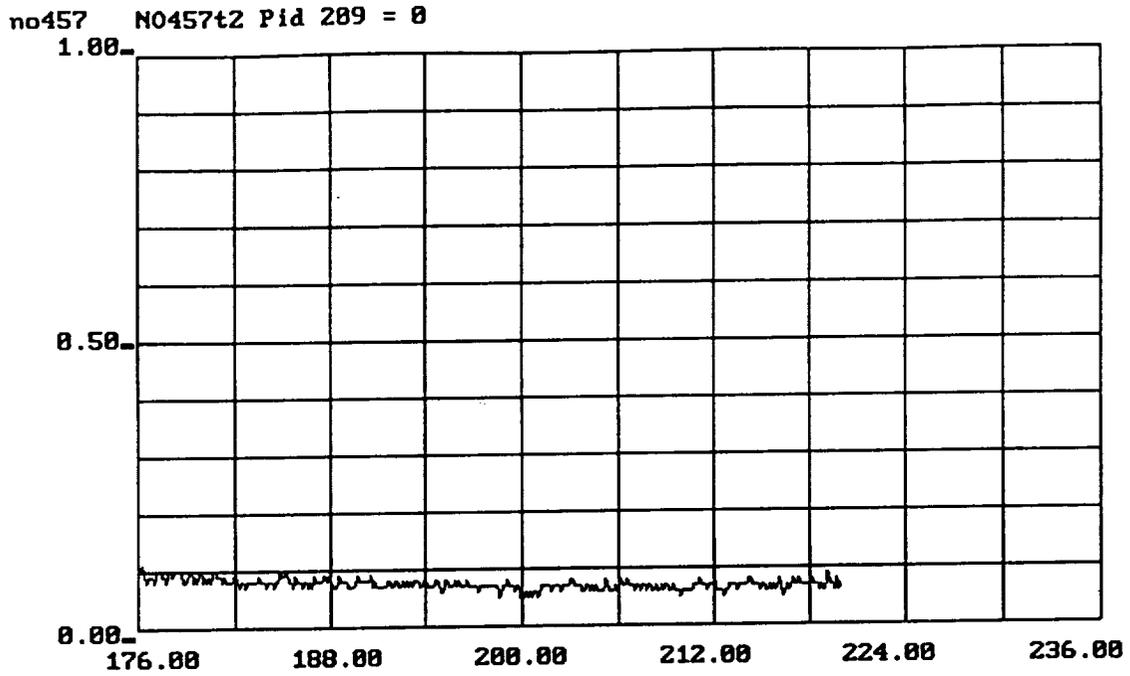
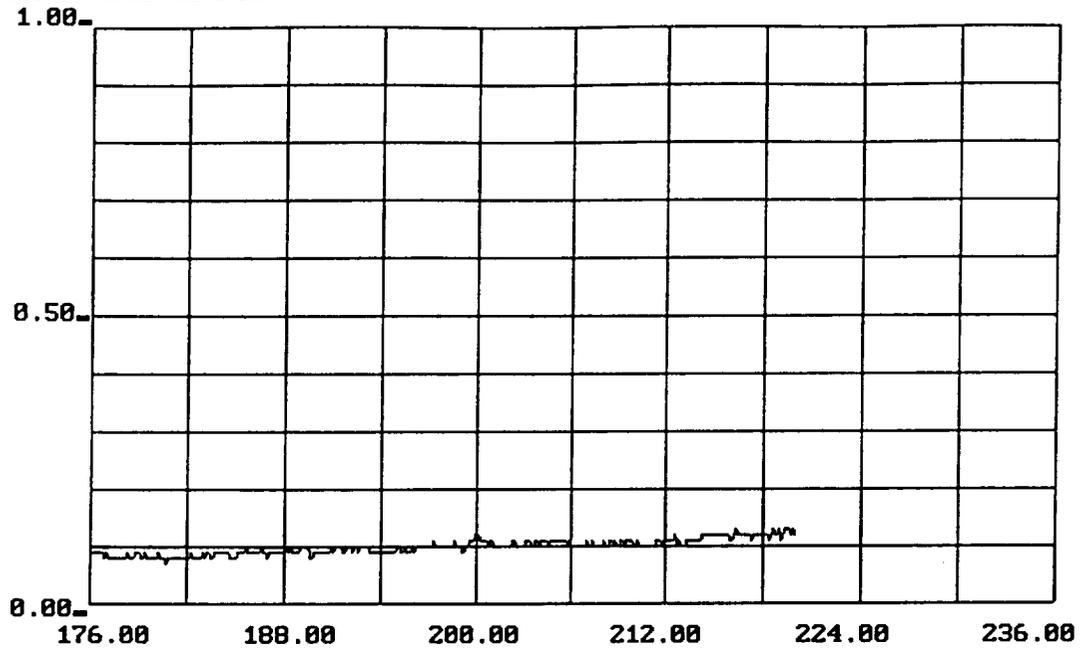


Figure A.4.4 Graph of Neural Net Output Unit 6, Holdout Case 457

PID 209 Features Replaced with Zeros

Case 331 Unit

no457 N0457t2 Pid 289 = 0



-7-

Figure A.4.5 Graph of Neural Net Output Unit 7, Holdout Case 457

PID 209 Features Replaced with Zeros

Case 340 Unit

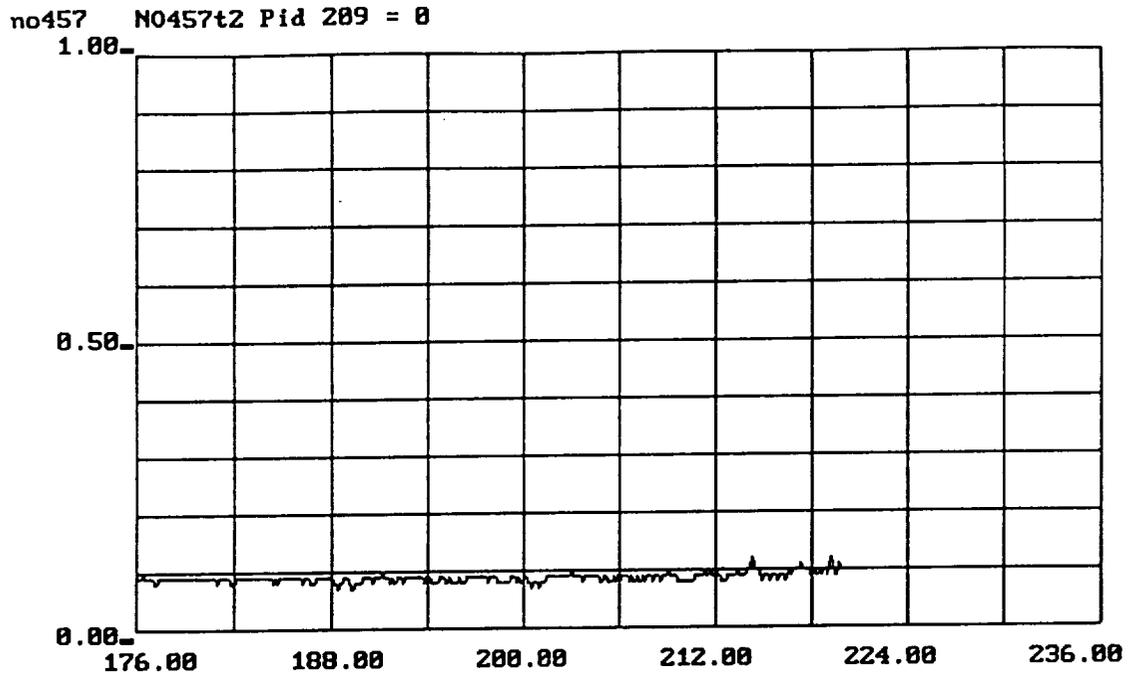
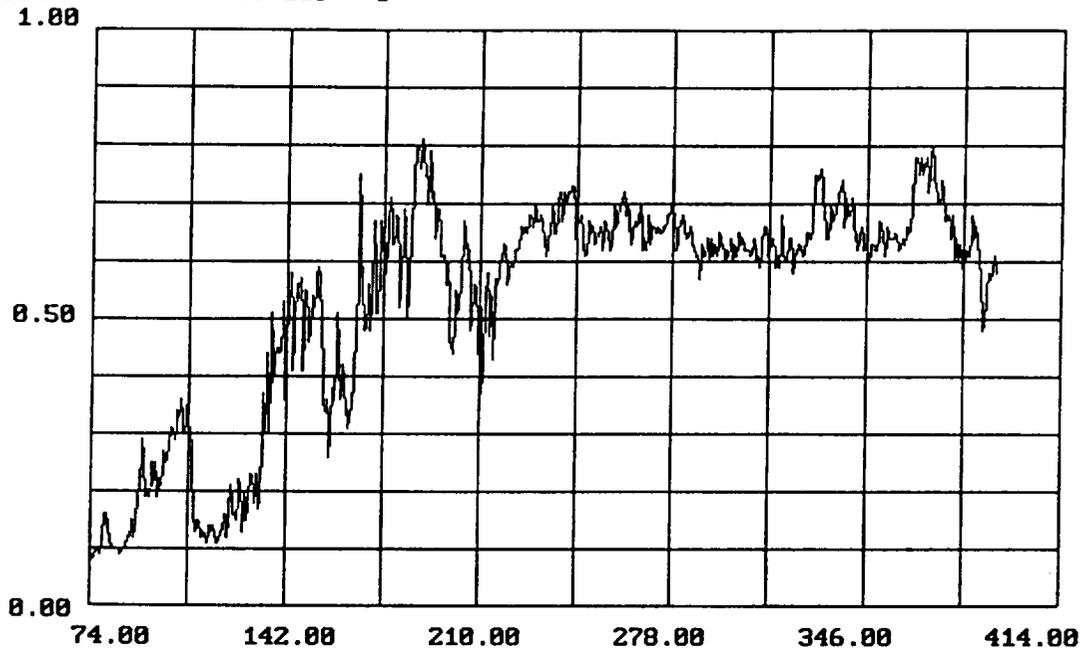


Figure A.4.6 Graph of Neural Net Output Unit 8, Holdout Case 457

PID 209 Features Replaced with Zeros

Case 364 Unit

no364 No364t4 Pid 209 = 0



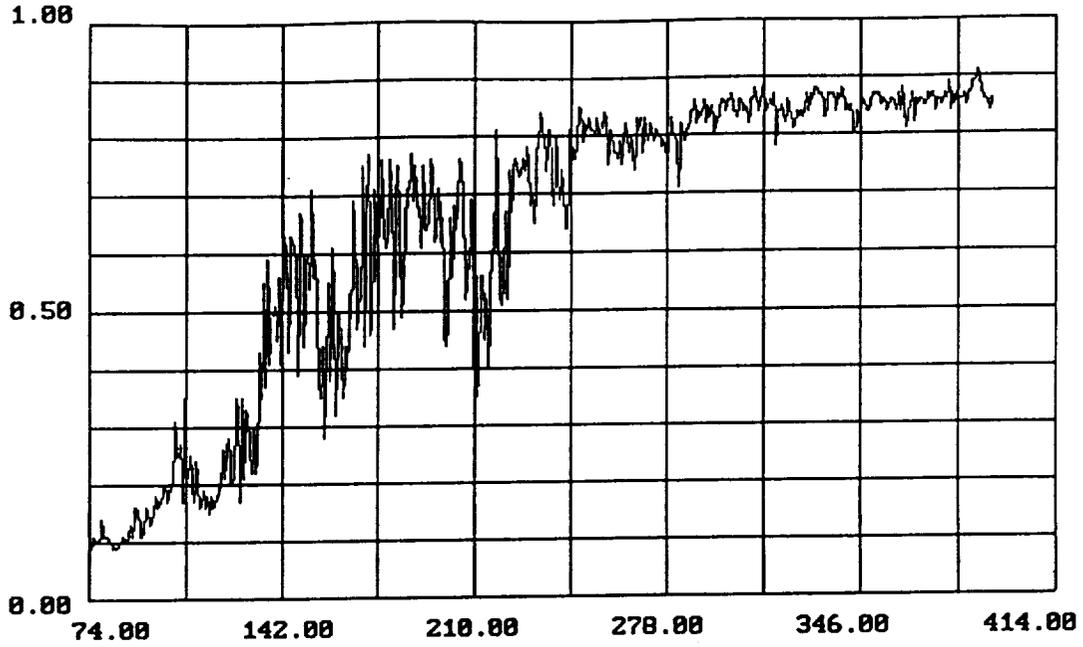
-8-

Figure A.4.7 Graph of Neural Net Output Unit 0, Holdout Case 364

PID 209 Features Replaced with Zeros

Nominal/Fault Unit

no364 No364t4 Pid 209 = 0



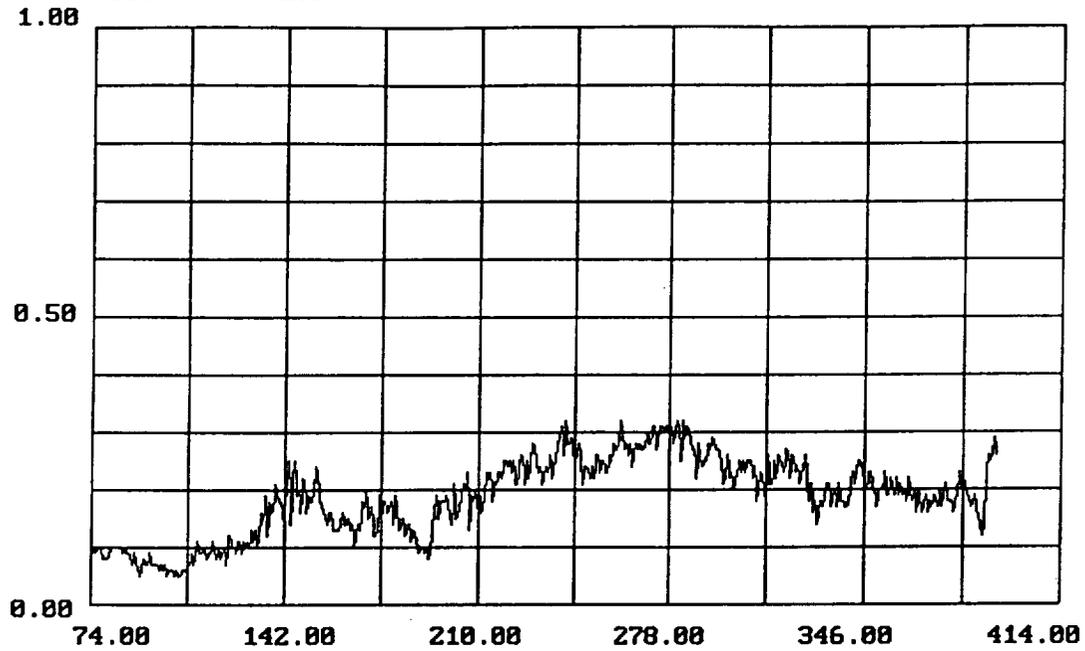
-3-

Figure A.4.8 Graph of Neural Net Output Unit 3, Holdout Case 364

PID 209 Features Replaced with Zeros

Case 249 Unit

no364 No364t4 Pid 209 = 0



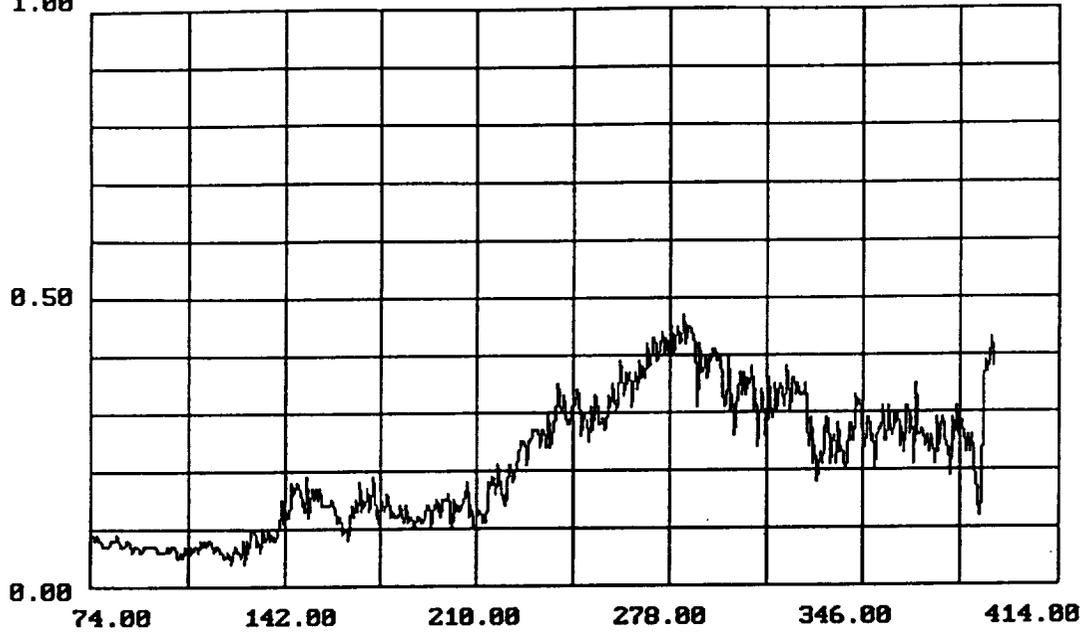
-4-

Figure A.4.9 Graph of Neural Net Output Unit 4, Holdout Case 364

PID 209 Features Replaced with Zeros

Case 259 Unit

no364 No364t4 Pid 209 = 0
1.00



-7-

Figure A.4.10 Graph of Neural Net Output Unit 7, Holdout Case 364
PID 209 Features Replaced with Zeros
Case 340 Unit

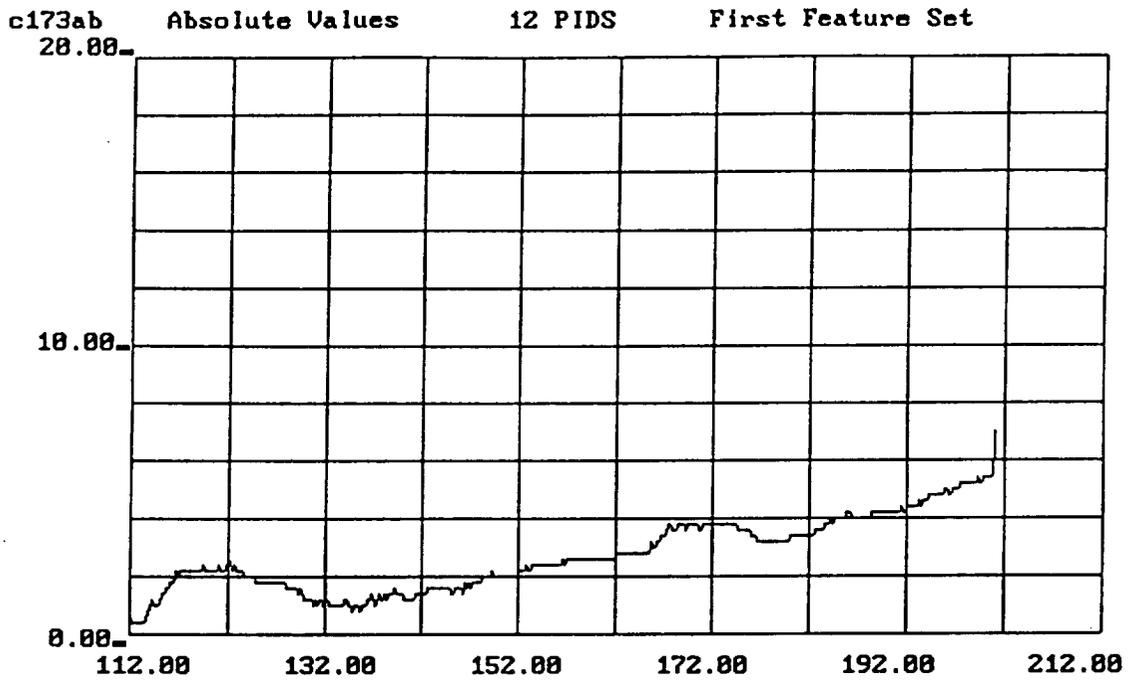


Figure A.5.1 Graph of Absolute Values, Case 173

Two Features

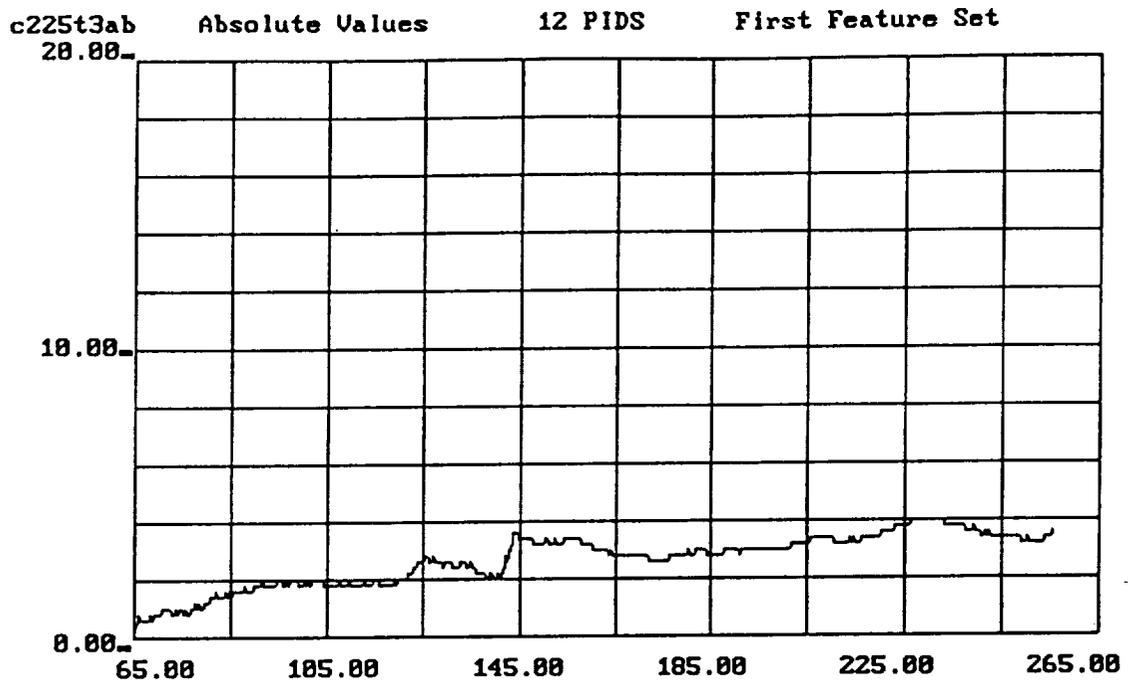


Figure A.5.2 Graph of Absolute Values, Case 225

Two Features

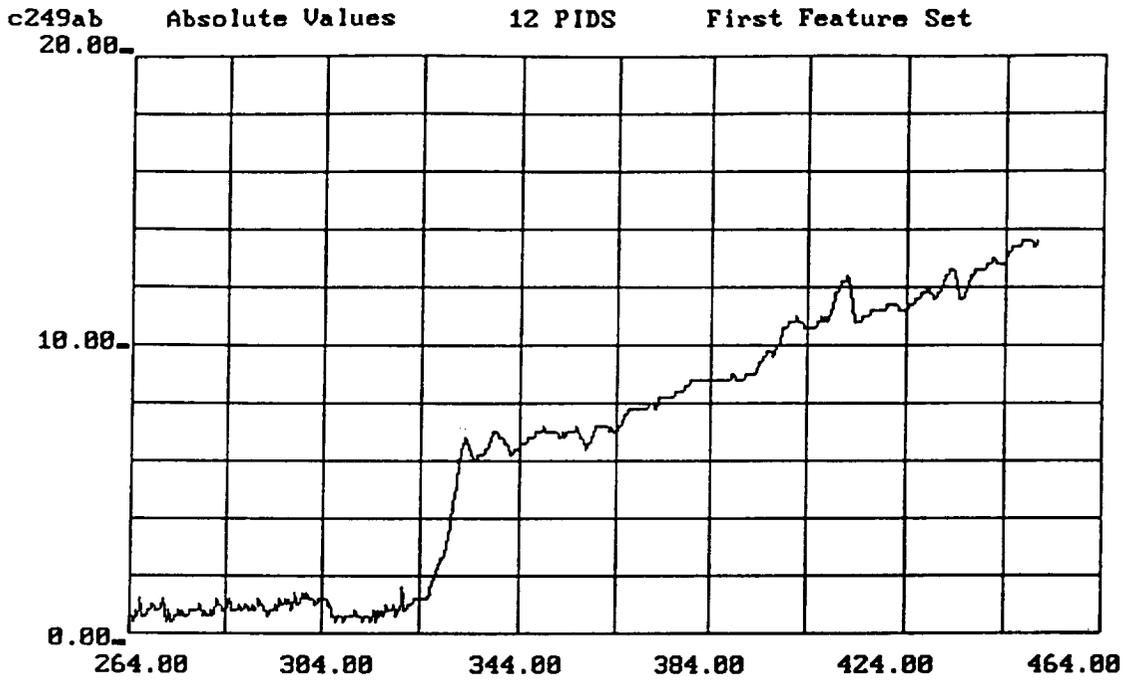


Figure A.5.3 Graph of Absolute Values, Case 249

Two Features

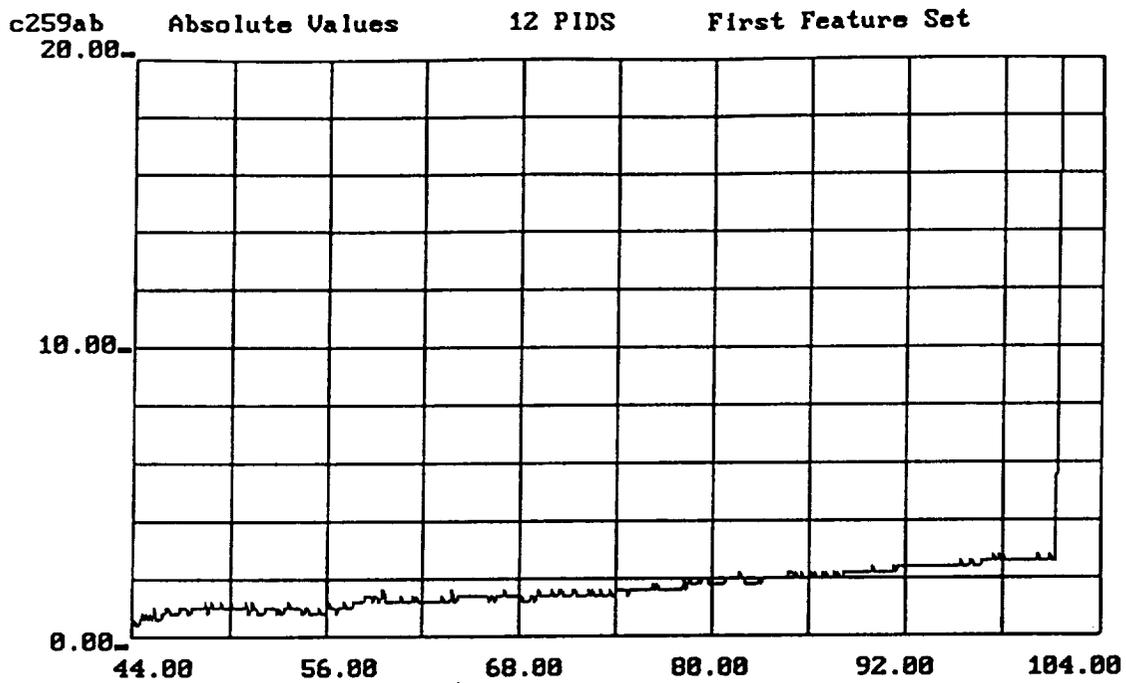


Figure A.5.4 Graph of Absolute Values, Case 259

Two Features

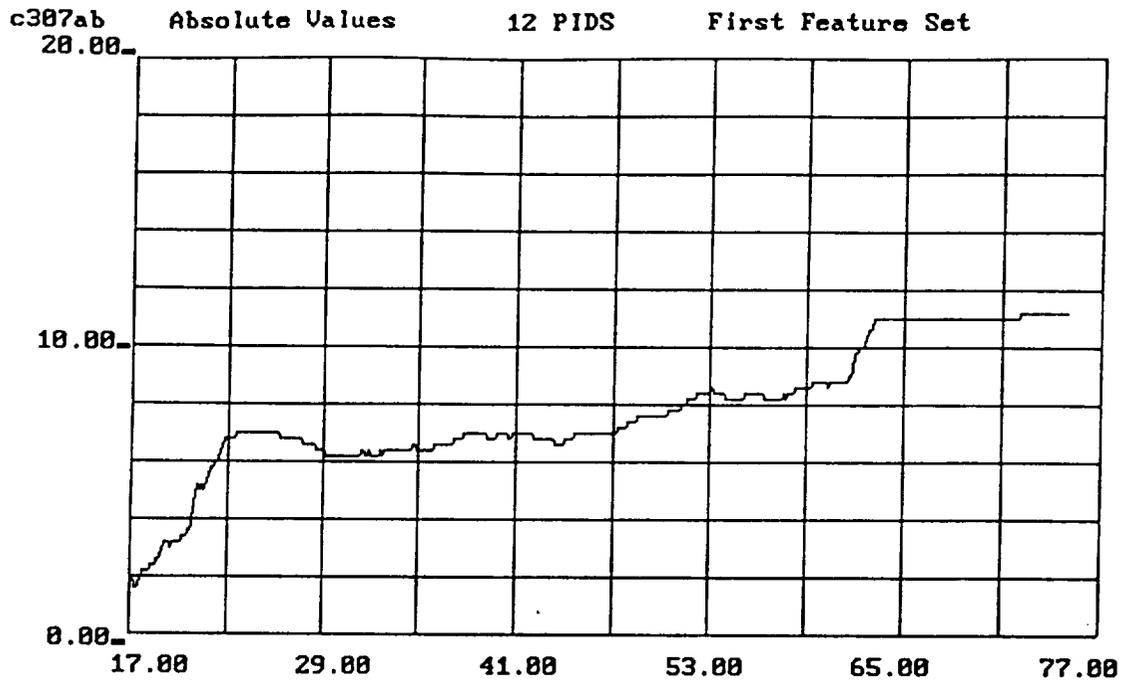


Figure A.5.5 Graph of Absolute Values, Case 307

Two Features

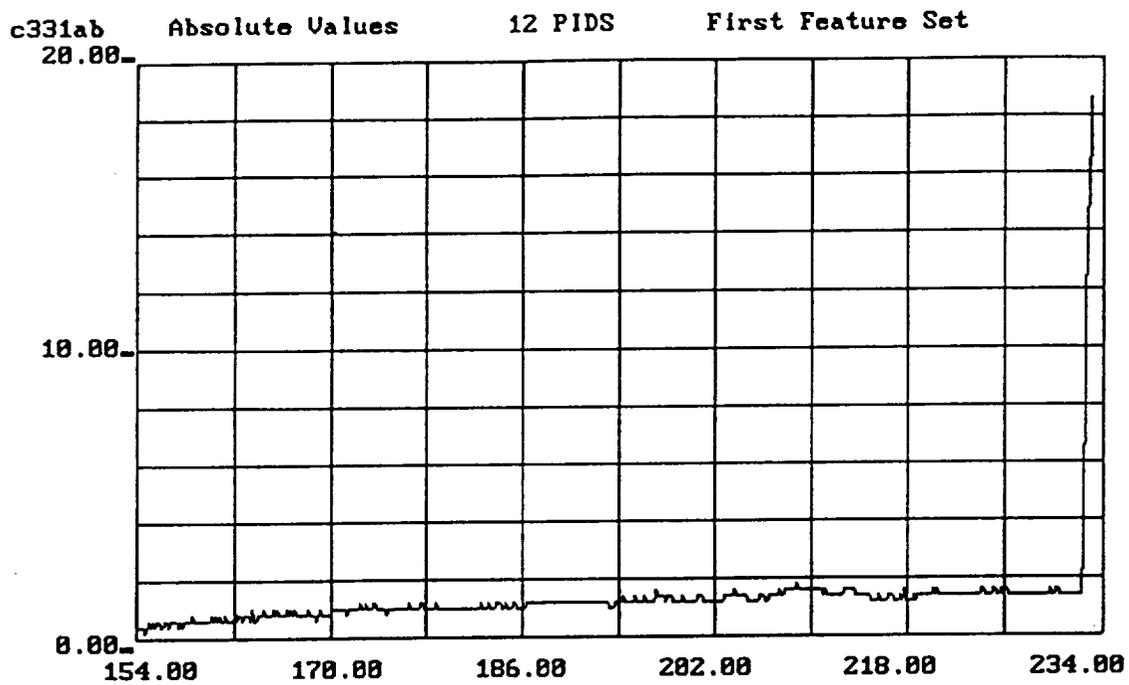


Figure A.5.6 Graph of Absolute Values, Case 331

Two Features

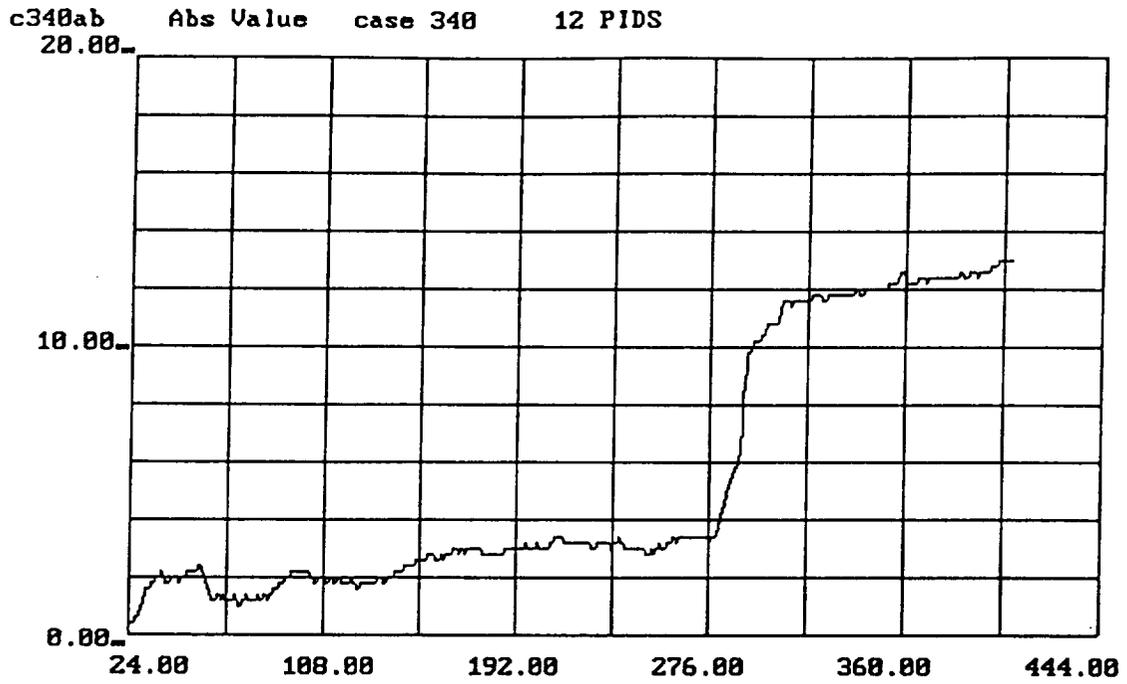


Figure A.5.7 Graph of Absolute Values, Case 340

Two Features

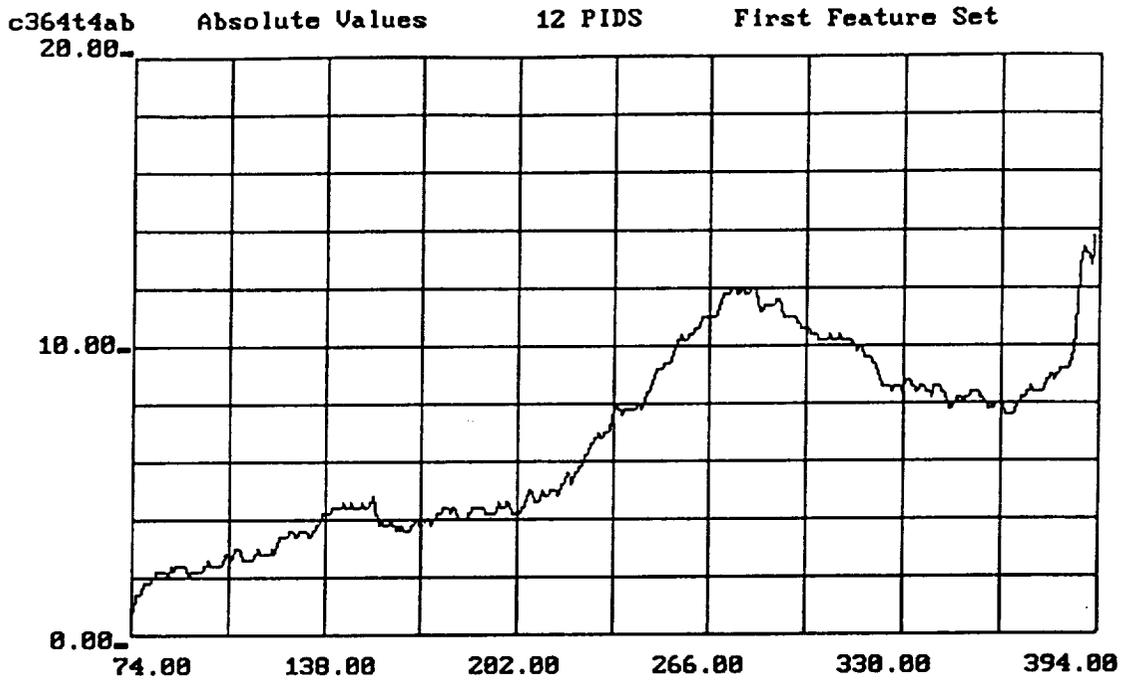


Figure A.5.8 Graph of Absolute Values, Case 364

Two Features

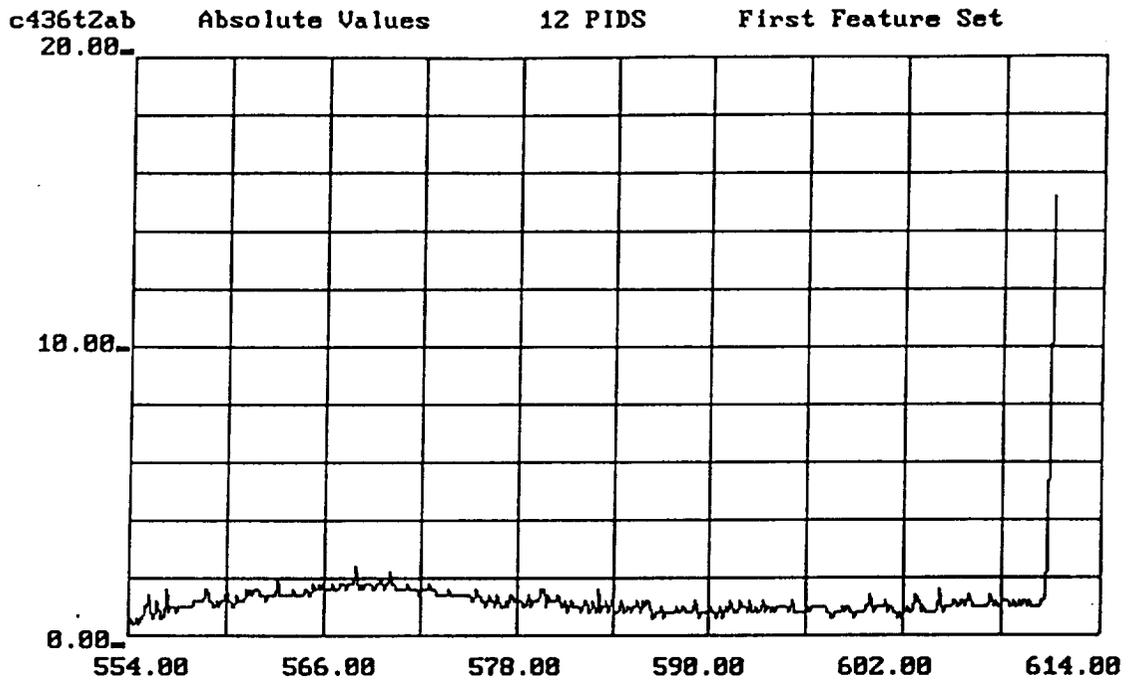


Figure A.5.9 Graph of Absolute Values, Case 436

Two Features

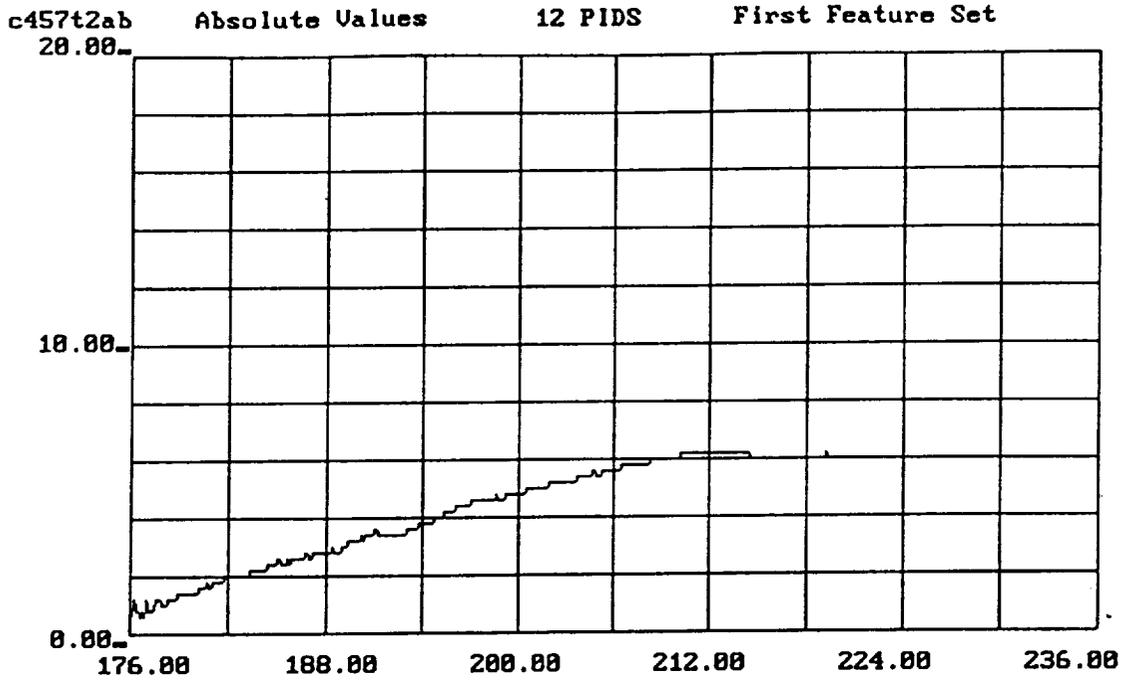


Figure A.5.10 Graph of Absolute Values, Case 457

Two Features

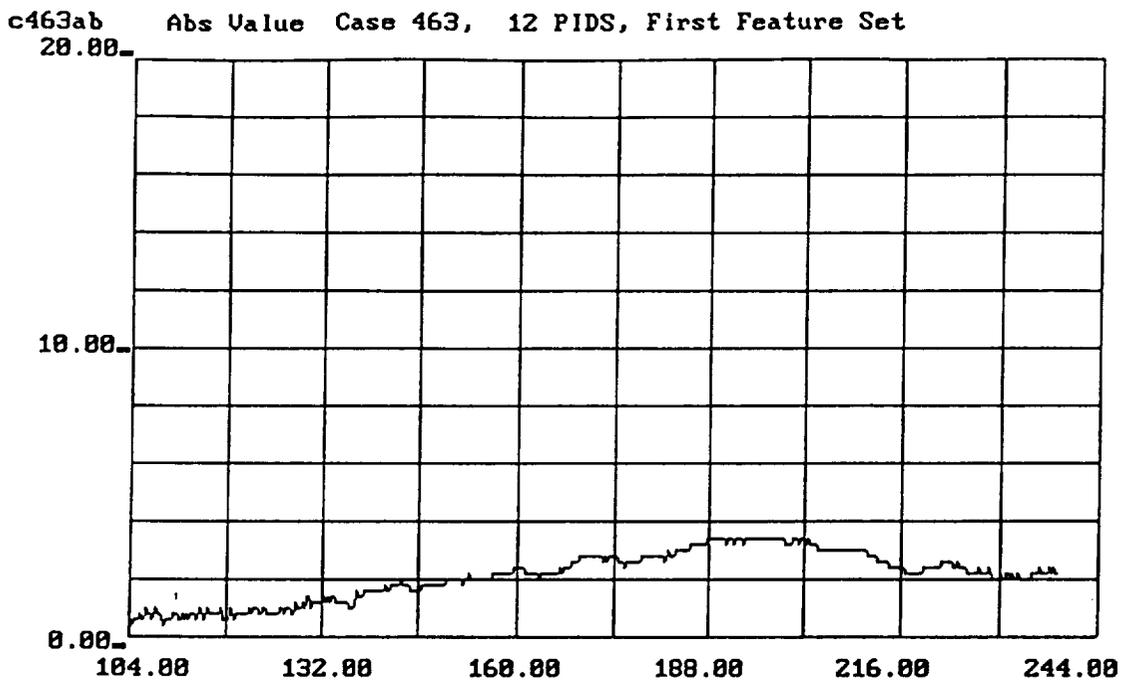


Figure A.5.11 Graph of Absolute Values, Case 463

Two Features

APPENDIX B

B.1 Data File Formats, Valve Signature Investigations

The ASYST data collection software has an auto file name generation portion, that puts each of the test runs into its own file, and continually keeps track of each of the tests that it has been requested to run. The data is written out in the binary format noted previously in Table 3.1. The data was collected for five Circle Seal valves, three Wright valves, and two Marotta valves. Each of the valves data runs were stored in separate directories on a diskette with the valve's name as the name of the directory. The tests that were run are stored on two diskettes and are listed in Table B.1.1 and Table 3.6.1.2. Table A.1 contains the files that are listed on the first disk and have the samples of each of the ten valves. This disk also contains the faulted data samples of the Marotta valve.

Table B.1.1 Files Included on Disk 1

Directory	Files	Comments
ev1	cs_21.12 - cs_40.12	Circle Seal valve ev1 (poppet up)
ev1	cs_41.13 - cs_60.13	Circle Seal valve ev1 (poppet up) [50 invisible runs]
ev2	cs_80.13 - cs_99.13	Circle Seal valve ev2 (poppet up)
ev4	cs_100.15 - cs_119.15	Circle Seal valve ev4 (poppet down)
ev6	cs_120.9 - cs_139.9	Circle Seal valve ev6 (poppet down)
ev5	cs_140.8 - cs_159.8	Circle Seal valve ev5 (poppet down)
rem1	cs_177.5 - cs_196.5	Wright
rem2	cs_197.4 - cs_216.4	Wright (noise close)
rem3	cs_217.11 - cs_236.11	Wright
rem3	cs_237.11 - cs_256.11	Wright [50 invisible runs]
ev2m	cs_259.13 - cs_278.13	Marotta (lose modifiable/faultable valve)
ev2m	cs_279.13 - cs_298.13	Marotta (lose faultable valve) [50 invisible runs]
ev3	cs_333.14 - cs_352.14	Marotta
ev2m	cs_353.13 - cs_372.13	Marotta (lose faultable valve) [weak spring=3]
ev2m	cs_373.13 - cs_392.13	Marotta (lose faultable valve) [failed open=1]
ev2m	cs_393.13 - cs_412.13	Marotta (lose faultable valve) [failed closed=2]

Total of 320 files

After the samples were collected for each of the valves, and a couple of failure data samples were taken and recorded, then the pressure was introduced into the system. The files that are listed on the second disk are from the pressure readings. Because these samples were taken on a different day, a repeated set of control data was taken that consisted of the valves normally cold state and the warm state (after 50 invisible runs). The control data were collected before the samples with the pressure was taken. Each of the readings with the different levels of pressure also had another set of invisible runs to ensure no temperature effects were observed. The files for the pressure reading are

listed in Table B.1.2.

Table B.1.2 Pressure Readings Files

Directory	Files	Comments
root	cs_413.12 - cs_432.12	Circle Seal valve ev1 (cold data)
root	cs_433.12 - cs_452.12	Circle Seal valve ev1 (warm data) [50 invisible runs]
root	cs_453.12 - cs_472.12	Circle Seal valve ev1 [300 psig=14] [20 invisible runs]
root	cs_473.12 - cs_492.12	Circle Seal valve ev1 [500 psig=16] [20 invisible runs] a difference in the error code of these file exists where the error code should be 18 for the pressure of 500 psig.
root	cs_493.12 - cs_512.12	Circle Seal valve ev1 [100 psig=10] [20 invisible runs]
root	cs_513.12 - cs_532.12	Circle Seal valve ev1 [raw data=9] [20 invisible runs] these files are 1600 bytes smaller than the rest of the files

Total of 120 files

The total number of files on the two diskettes is 440 files. The output of the raw data samples that were taken are 1600 bytes smaller than the rest of the files. The program was modified to output the raw data rather than a set of data that has already been smoothed. In this modification of the program the raw data may have been stored as integer data values rather than real and may have been the sensed data from the D-to-A converter, and may not be of any use here.

On the following pages, the complete source code listing for the ASYST data collection program is provided. The initial file merely loads the remaining files into the ASYST system. Two files, START.BIT and C_SIG.VAR, contain only variable definitions. One file, C_SIG.DIG, contains the hardware drivers. File C_SIG.A2D collects the data, and C_SIG.STA performs analytical functions on the data (these functions were not used in performing this CRAD). The final file, C_SIG.EDT, contains the screen plotting routines and saves the data to a file.

B.1.1 Initialization File

```
ECHO.OFF  
LOAD \NEWASYST\JIM\NETS\START.BIT  
LOAD \NEWASYST\JIM\NETS\C_SIG.VAR  
LOAD \NEWASYST\JIM\NETS\C_SIG.DIG  
LOAD \NEWASYST\JIM\NETS\C-SIG.A2D  
LOAD \NEWASYST\JIM\NETS\C-SIG.STA  
LOAD \NEWASYST\JIM\NETS\C-SIG.EDT  
ECHO.ON
```

B.1.2 C_SIG.VAR

```
ECHO.OFF
exp.mem> system.buffer
65000 system.buffer.size
: nd normal.display ;
: sd stack.display ;
: 2DUP OVER OVER ;
: LAST \ (ARRAY -- LAST_ELEMENT)
  [ 1 []DIM ] ;
: NUMBER>STRING \ ( NUMBER -- )
  "." 2 "LEN 2 - "SUB ;

SCALAR EV1 12 EV1 :-
SCALAR EV2 13 EV2 :-
SCALAR EV3 14 EV3 :-
SCALAR EV4 15 EV4 :-
SCALAR EV5 8 EV5 :-
SCALAR EV6 9 EV6 :-
SCALAR EV7 10 EV7 :-
SCALAR LREM2 6 LREM2 :-
SCALAR REM1 5 REM1 :-
SCALAR REM2 4 REM2 :-
SCALAR REM3 11 REM3 :-
SCALAR LREM3 7 LREM3 :-
SCALAR LREM4 0 LREM4 :-
SCALAR REGVENT 3 REGVENT :-

ECHO.OFF

DIM[ 400 ] DMA.ARRAY OPENDATA
DIM[ 400 ] DMA.ARRAY CLOSEDATA

TOKEN DIFFDATA
EXP.MEM> DIFFDATA

TOKEN OPEN TIMES
EXP.MEM> OPEN_TIMES

TOKEN CLOSE TIMES
EXP.MEM> CLOSE_TIMES

TOKEN OPENDATA.SMOOTH
EXP.MEM> OPENDATA.SMOOTH

TOKEN CLOSEDATA.SMOOTH
EXP.MEM> CLOSEDATA.SMOOTH

INTEGER SCALAR OPEN SAMPLES
OPENDATA 1 []DIM OPEN_SAMPLES :-
DROP

INTEGER SCALAR CLOSE SAMPLES
CLOSEDATA 1 []DIM CLOSE_SAMPLES :-
DROP

INTEGER SCALAR INVIS_RUNS

INTEGER DIM[ 7 ] ARRAY STATUS_ARRAY
```

INTEGER DIM[64] ARRAY RUNDATA_ARRAY

INTEGER SCALAR FAIL_CODE

REAL SCALAR OPEN DIFF THRESHOLD

REAL SCALAR CLOSE DIFF THRESHOLD

REAL SCALAR OPEN TIME

REAL SCALAR CLOSE TIME

REAL SCALAR DEFAULT OPEN TIME

REAL SCALAR DEFAULT_CLOSE TIME

INTEGER SCALAR VALUE BIT_NUM

INTEGER SCALAR XDUCER_CHANNEL_NUM

REAL SCALAR ATOD_GAIN

B.1.3 C_SIG.DIG

ECHO.OFF

4 VALVE_BIT_NUM :-

\ D/A CONVERTER SETUP

3 DIGITAL.TEMPLATE DIG.OUT \ TEMPLATE FOR D2820 WRITES TO BOTH BYTES
DIGITAL.INIT

: ON VALVE_BIT_NUM

2 SWAP **

2 16 ** 1 - swap -

DIGITAL.OUT ;

: OFF

2 16 **

1 -

DIGITAL.OUT ;

B.1.4 C_SIG.A2D

ECHO.OFF

\ SYSTEM PARAMETERS

5 XDUCER_CHANNEL_NUM :-
3 ATOD_GAIN :-
100 OPEN_TIME :-
100 CLOSE_TIME :-
200 DEFAULT_CLOSE_TIME :-
200 DEFAULT_OPEN_TIME :-

\ A/D SETUP

XDUCER_CHANNEL_NUM DUP A/D.TEMPLATE OPEN.TEMP
OPENDATA DMA.TEMPLATE.BUFFER
OPEN_TIME OPEN_SAMPLES / CONVERSION.DELAY
ATOD_GAIN A/D.GAIN
A/D.INIT

XDUCER_CHANNEL_NUM DUP A/D.TEMPLATE CLOSE.TEMP
CLOSEDATA DMA.TEMPLATE.BUFFER
CLOSE_TIME CLOSE_SAMPLES / CONVERSION.DELAY
ATOD_GAIN A/D.GAIN
A/D.INIT

XDUCER_CHANNEL_NUM DUP A/D.TEMPLATE DEFAULT_OPEN.TEMP
OPENDATA DMA.TEMPLATE.BUFFER
DEFAULT_OPEN_TIME OPEN_SAMPLES / CONVERSION.DELAY
ATOD_GAIN A/D.GAIN
A/D.INIT

XDUCER_CHANNEL_NUM DUP A/D.TEMPLATE DEFAULT_CLOSE.TEMP
OPENDATA DMA.TEMPLATE.BUFFER
DEFAULT_CLOSE_TIME CLOSE_SAMPLES / CONVERSION.DELAY
ATOD_GAIN A/D.GAIN
A/D.INIT

\ A/D WORDS

: BEGIN DMA ACQUIRE
ATOD_GAIN A/D.GAIN
A/D.INIT
A/D.IN>ARRAY(DMA) ;

: TIMES \ (CONVERSION.DELAY NUMBER_OF_SAMPLES -- ARRAY_OF_TIMES)
REAL RAMP 1 - * ;

: SET DELAYS
OPEN.TEMP
OPEN_TIME OPEN_SAMPLES / CONVERSION.DELAY
A/D.INIT
?CONVERSION.DELAY OPEN_SAMPLES TIMES BECOMES> OPEN_TIMES

CLOSE.TEMP
CLOSE_TIME CLOSE_SAMPLES / CONVERSION.DELAY
A/D.INIT
?CONVERSION.DELAY CLOSE_SAMPLES TIMES BECOMES> CLOSE_TIMES

: SCALEIT
OPENDATA -960 1040 A/D.SCALE OPENDATA :-
CLOSEDATA -960 1040 A/D.SCALE CLOSEDATA :-

B.1.5 C_SIG.STA

ECHO.OFF

```
: FIND PEAKS \ (DATA -- [MAXIMA] MINIMUM)
  2 SET.#.OPTIMA
  DUP LOCAL.MAXIMA NOT IF ." ERROR -- NO MAXIMA FOUND! " QUIT
  ELSE
  1 SET.#.OPTIMA
  LOCAL.MINIMA NOT IF ."ERROR -- NO MINIMUM FOUND! " QUIT
  THEN
  THEN ;

: AREA \ (DATA TIMES -- AREA) NOTE: MUST BE IN CORRECT TEMPLATE
  INTEGRATE.DATA LAST * ;

: OPEN EXP CURVE \ RETURNS EXPONENTIAL CURVE FIT BETWEEN 1ST AND 2ND PEAK
  OPENDATA.SMOOTH [ 20 ]
  2 SET.#.OPTIMA LOCAL.MAXIMA
  CATENATE
  ROT
  OPEN TIMES [ 20 ]
  CATENATE
  SWAP
  LEASTSQ.EXP.FIT
  OPEN TIMES OVER [ 1 ] *
  SWAP [ 2 ] + EXP ;

: CLOSE_EXP_CURVE ;

: OPENWORK \ ( -- OPEN WORK )
  OPENDATA.SMOOTH OPEN_TIMES AREA
  OPEN_EXP_CURVE OPEN_TIMES AREA
  - ;

: CLOSEWORK \ ( -- CLOSING WORK )
  CLOSE_EXP_CURVE CLOSE_TIMES AREA
  CLOSEDATA.SMOOTH CLOSE_TIMES AREA
  - ;
```

B.1.6 C_SIG.EDT

```
ECHO.OFF

3 OPEN_DIF_THRESHOLD :-
3 CLOSE_DIF_THRESHOLD :-

: SMOOTHIT \ SMOOTHS OPEN AND CLOSEDATA ARRAYS, STORES IN .SMOOTH ARRAYS
  .3 SET.CUTOFF.FREQ
  OPENDATA SMOOTH BECOMES> OPENDATA.SMOOTH
  CLOSEDATA SMOOTH BECOMES> CLOSEDATA.SMOOTH ;

: DRAW.GRAPHS
  CLOSE_TIMES CLOSEDATA.SMOOTH XY.AUTO.PLOT
  OPEN_TIMES OPENDATA.SMOOTH XY.DATA.PLOT ;

: STATS ; \ ARE TBD

: ROUNDTO \ ( DIVISOR NUMBER -- ROUNDED NUMBER )
  DUP ROT MODULO - ;

: C_SIG_ACQUIRE
  SET_DELAYS

  OPEN.TEMP
  BEGIN_DMA_ACQUIRE
  ON
  OPEN_TIME MSEC.DELAY

  200 MSEC.DELAY \ SETTLING TIME

  CLOSE.TEMP
  BEGIN_DMA_ACQUIRE
  OFF
  CLOSE_TIME MSEC.DELAY

  SCALEIT
  SMOOTHIT ;

: FIND_OPEN_END
  OPENDATA.SMOOTH [ ]MAX OPEN_DIFF_THRESHOLD -
  OPEN SAMPLES 20 DO
    OPENDATA.SMOOTH [ I ] OVER > IF DROP I LEAVE
    THEN LOOP
    ?CONVERSION.DELAY * 10 +
    5 SWAP ROUNDTO ; \ ROUNDING ASSURES REPEATABILITY OF SCALING

: FIND_CLOSE_END
  OPENDATA.SMOOTH [ ]MIN CLOSE_DIFF_THRESHOLD +
  CLOSE SAMPLES 20 DO
    CLOSEDATA.SMOOTH [ I ] OVER < IF DROP I LEAVE
    THEN LOOP
    ?CONVERSION.DELAY * 10 +
    5 SWAP ROUNDTO ;

: VALVE.INIT \ (VALVE_BIT_NUM -- )
  VALVE_BIT_NUM :- ;
```

```

: C_SIG
  C_SIG_ACQUIRE
  STATS
  DRAW.GRAPHS ;

: TEST_NUM.TEMP
  FILE.TEMPLATE
  0 COMMENTS
  INTEGER DIM[ 1 ] SUBFILE
  END ;

: NEW_TEST \ ( -- NEW_RUN_# ) INCREMENTS THE TEST COUNTER AND RETURNS A #.
  TEST_NUM.TEMP
  FILE_OPEN \NEWASYST\JIM\NETS\TEST.NUM
  1 SUBFILE
  FILE>UNNAMED.ARRAY 1 +
  DUP ARRAY>FILE
  [ 1 ]
  FILE.CLOSE ;

: DATA_WRITE
  new_test NUMBER>STRING
  "\NEWASYST\JIM\NETS\CS " "SWAP "CAT \ CREATES A STROMG CS_XX
  " ." VALVE BIT NUM NUMBER>STRING "CAT
  "CAT \ PUTS THEM ALL TOGETHER

  "DUP
  DEFER> RANDOM.FILE.CREATE
  DEFER> RANDOM.OPEN

"TIME " " "CAT
"DATE "CAT " : " "CAT RUNDATA ARRAY ">ARRAY \ TIME, DATE, COMMENTS
  FAIL_CODE STATUS_ARRAY [ 1 ] :-
  VALVE_BIT_NUM STATUS_ARRAY [ 2 ] :-
  OPEN_TIME STATUS_ARRAY [ 3 ] :-
  OPEN_SAMPLES STATUS_ARRAY [ 4 ] :-
  CLOSE_TIME STATUS_ARRAY [ 5 ] :-
  CLOSE_SAMPLES STATUS_ARRAY [ 6 ] :-
  INVIS_RUNS STATUS_ARRAY [ 7 ] :-
  RUNDATA_ARRAY RANDOM.PUT
  STATUS_ARRAY RANDOM.PUT
  OPENDATA.SMOOTH RANDOM.PUT
  CLOSEDATA.SMOOTH RANDOM.PUT
  RANDOM.CLOSE
;

: MULTI_C_SIG
  ND
  CR ." INPUT THE FAILURE CODE FOR THIS RUN (0=NO FAILURE): "
  "INPUT 0 "NUMBER FAIL CODE :-
  CR ." INPUT THE NUMEBR OF 'INVISIBLE' RUNS: "
  "INPUT 0 "NUMBER INVIS RUNS :-
  CR ." INPUT THE NUMBER OF RECORDED RUNS: "
  "INPUT 0 "NUMBER

```

```
GRAPHICS.DISPLAY
INVIS RUNS 1 DO
  C_SIG_ACQUIRE
LOOP

1 + 1 DO
  C SIG ACQUIRE
CR "." RUN #" I INVIS_RUNS + .
  1 I - IF open times opendata.smooth xy.auto.plot
    ELSE OPEN_TIMES OPENDATA.SMOOTH xy.data.plot
  THEN
  CLOSE_TIMES CLOSEDATA XY.DATA.PLOT
  DATA WRITE
```

B.2 Neural Network Data Files

The files included on the third disk (Disk #3) are specified here. These include: all the source code for the data preprocessing system; all the training and test set data, including the normalization files for the close-valve network; the network files that specify the final weights after training; and all the result files that demonstrate the network's responses during testing at all noise levels and with both the training and test set data files. The files are organized into directories for easier access. These directories are as shown below.

DATAFILE: contains all the training and test data, including normalization files

NETWORKS: contains the ".NET" files for the trained networks

PROGRAMS: contains the C source code for the preprocessing routines

RESULTS: contains the test results

OPEN: for the open-valve network

TRAIN: processing the training dataset at various noise levels

TEST: processing the test dataset at various noise levels

CLOSE: for the close-valve network

TRAIN: processing the training dataset at various noise levels

TEST: processing the test dataset at various noise levels

Results files are named according to their contents. The first letter of the file name is "O" for an open-valve network result, and "C" for a close-valve network result. The next two digits indicate the amount of noise present when the data was processed; this ranges from "00" for no noise to "50" for 50% noise. The final letters designate which data set was used, "TRN" for the training set and "TST" for the test set.

The internal format of the files can be read using the C program READFILE.C contained in the PROGRAMS directory. It is thoroughly described in the ANSim Users Manual as well.

B.3 List of Figures in Section 3

Figure

- 3.1 Schematic representation of Smart BIT pneumatic test bench
- 3.2 Photograph of Smart BIT pneumatic test bench
- 3.3 A Circle Seal valve
- 3.4 A Marotta valve
- 3.5 Applying a current through the coil windings causes the poppet to move, which generates a reverse current in the coil
- 3.6 The compression of the return spring when the poppet core pops up provides the tension necessary to force the poppet back down into the coil once the electromagnetic conditions of the coil permit
- 3.7 Sample valve signatures showing relative ease of identification
- 3.8 A typical current signature with features marked
- 3.9 Typical Circle Seal signatures for valve-open (increasing current) and valve-close (decreasing current) state changes
- 3.10 Typical Wright valve signature curves. Note the higher current and shorter response time compared to Circle Seal valves
- 3.11 A typical set of signatures for a Marotta valve. Notice that these valves have both the high current of the Wright and the long response time of the Circle Seal valves
- 3.12 Signatures from all three valve types are superimposed to delineate the

differences between them

- 3.13 One Circle Seal valve had an anomalous signature
- 3.14 A comparison between the signatures of the anomalous Circle Seal valve (#15) and a more typical one (#12)
- 3.15 The signature of a Marotta valve with a weak spring (low spring constant). No other failures for this signature
- 3.16 A Marotta valve signature with a medium-strength spring
- 3.17 The signature of a Marotta valve with a strong spring
- 3.18 This chart summarizes the differences in signature for a Marotta valve with variations in the strength of the spring or with deliberate failures induced
- 3.19 A Marotta valve with the screw adjustment tweaked has a remarkably altered signature curve
- 3.20 A Circle Seal valve's signature becomes quite complex when the screw adjustment is altered
- 3.21 When the Circle Seal valve's screw adjustment is too extreme, the valve fails to operate. This is the resulting signature
- 3.22 A Circle Seal valve's signature curve can change dramatically as failures or misadjustments occur
- 3.23 The signature of a Marotta valve varies slightly as it warms up under use. The curve slowly moves from the cold position to the warm position and then stabilizes there
- 3.24 The effects of pressure at the valve are marked and highly distinguishable.

It seems likely that a neural network could be trained to estimate pressure at the valve from the valve's current signature

- 3.25 The signature of two valves acting simultaneously is the mathematical sum of the two individual signatures, clipped by the maximum range of the sensor
- 3.26 A Marotta valve operated upside down shows very little change in its signature

B.4 List of Tables in Section 3

Table

- 3.1 Data run header format
- 3.2 Valve number and failure code clarification
- 3.3 Failure code meanings
- 3.4 The output pattern element definition
- 3.5 Open network's performance with training data
- 3.6 Open network's performance with test data
- 3.7 Closed network's performance with training data
- 3.8 Closed network's performance with test data

3.6.1.1 Files included on Disk #1

3.6.1.2 Pressure readings files

