R&D H038

ERC-R-92-022

Final Report

Automated Screening of Propulsion System Test Data by Neural Networks

Phase I

by
W. Andes Hoyt, ERC
Dr. Bruce A. Whitehead, UTSI

Contract No. NAS8-39184
SBA 4-91-2-0357

for the period
October 4, 1991—April 3, 1992

Submitted to
National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, AL 35812

April 17, 1992

ERC, Inc.
P. O. Box 417, Tullahoma, TN 37388
(615) 455-9915 • Fax. (615) 454-2042

Preface

This report summarizes the work completed during the Phase I effort entitled, "Automated Screening of Propulsion System Test Data by Neural Networks." A contractor team composed of personnel from Engineering Research and Consulting, Inc. (ERC), and The University of Tennessee Space Institute (UTSI) completed the work. The work was performed at ERC's Tullahoma, Tennessee, office and at the UTSI campus located near Tullahoma. Harry McDaris of ERC, Huntsville, played a key role in bringing together the appropriate NASA and contractor personnel who saw the Phase I project through from conception to completion.

Michael Whitley, EP52/NASA/MSFC, served as the Contracting Officer's Technical Representative. Other NASA/MSFC personnel who provided valuable direction, assistance and input for the project included Gary Lyles, Catherine McLeod and Marc Neely. Martin Marietta personnel under the direction of Eric Sander supported the project by loading test data to the tapes. In particular, Jeff Cornelius assisted with installation of the data screening system onto NASA's SUN workstation. Subroutines written by BCSS (Boeing) to access NASA CADS and facility databases are used by the data screening system.

The ERC team consisted of Jo Anne Malone, Tim Choate, Terry Bartholomew and W. Andes Hoyt. Dr. Bruce Whitehead, the originator of the neural network approach, works for the University of Tennessee.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Concept

The evaluation of propulsion system test and flight performance data involves reviewing an extremely large volume of sensor data generated by each test. An automated system that screens large volumes of data and identifies propulsion system parameters which appear unusual or anomalous will increase the productivity of data analysts. Data analysts may then focus on a smaller subset of anomalous data for further evaluation of propulsion system tests. Such an automated data screening system would give NASA the benefit of a reduction in the manpower and time required to complete a propulsion system data evaluation. This report details a six-month Phase I effort to develop a prototype data screening system.

1.2. Objectives

1.2.1. Overall

After preliminary work, project personnel and NASA employees agreed upon three overall project objectives:

1. To verify correctness and evaluate performance of the system over a significant number of propulsion system ground tests.
2. To ultimately field a fully operational automated system which will actually be used on a routine basis by analysts.
3. To design the system in such a way that it can later be tailored to different propulsion systems.

For this Phase I effort and throughout any continuation, neural networks will detect anomalies based on nominal propulsion system data only. We designed the system so it does not utilize any anomalous data during neural network training. Utilizing only nominal data for developing the data screening system represents the key element of our effort and objectives.

1.2.2. Phase I

Specifically, NASA and contractor personnel adopted a Phase I objective—to develop a prototype propulsion data screening system with the capability of detecting significant anomalies in steady-state data, using neural network technology. In Phase I we successfully built the prototype system to validate the overall concept of screening propulsion system data by using neural networks trained on nominal data only. The prototype used data from six different Space Shuttle Main Engine (SSME) tests.

1.3. Approach

1.3.1. Background

Hush and Salas [3][1]; Venkatusubramanian and Chan [6]; Whitehead, Ferber, and Ali [8]; and Whitehead, Kiech, and Ali [9] demonstrated that neural networks can learn to discriminate between nominal sensor data and various known classes of faults. Because neural networks are trained by example, their reliability depends upon the availability of representative training data for the discriminations to be learned by the network. If the discrimination to be learned is that of nominal versus anomalous data, then typical neural network training procedures require representative data on *both* sides of the discrimination, i.e. representative nominal data and representative anomalous data.

In rocket system propulsion testing it is possible to collect a representative sample of nominal data by systematically varying test conditions over the range of conditions to be used during testing. Collecting a representative sample of *anomalous* data is, however, problematic. In testing any complex system, only a very small fraction of all possible anomalies and/or malfunctions ever occur during data collection. Unfortunately, if a neural network has been trained on an incomplete or unrepresentative set of anomalies[2], there is no guarantee that it will reliably identify new types of anomalies which might occur.

The underlying problem, therefore, is to train a neural network to reliably discriminate between two categories (e.g. nominal versus anomalous) when representative training data are available for only one of these categories (e.g. nominal).

The commonly-used back propagation training algorithm for neural networks (Werbos [7] and Rumelhart, Hinton, and Williams [5]) learns to discriminate one category from another on the basis of representative training examples of both categories. The problem is that an extremely large number of different types of anomalies are typically possible. Since some of these possible anomalies have never been anticipated, it is not feasible to produce a set of training examples which is truly representative of the set of all possible anomalies.

1.3.2. ERC Approach

ERC took a different approach from those discussed in Section 1.3.1 above. In brief, our approach classified data into nominal and other-than-nominal sets by comparing the current test

---

[1]Numbers in brackets refer to similarly numbered references in the bibliography.

[2]Henceforth, we will use the term anomaly to represent any data that is outside of a hypothetical nominal range.

data with expected nominal values predicted by the neural network for the specific test configuration and test conditions.

Our approach pursued a more fundamental solution to defining neural network architectures than the previously discussed back-propagation methods. We developed an architecture which required training examples from one category only (Hush and Salas [3]; Whitehead, Ferber, and Ali, [8]). In this approach, the neural network training algorithm tried to match the expected range of nominal sensor data for a particular engine, given the time-series values of all independent test variables such as control parameters, control events, facility parameters, and facility events. The expected value of a data parameter was approximated as a function of all these independent test variables. Thus, the method resulted in a technique of function approximation.

Figure 1 depicts the multivariate function f to be approximated for each data parameter. Neural network architectures for multivariate function approximation were evaluated in literature (Broomhead and Lowe [1]; Cotter [2]; Moody and Darken [4]) and shown comparable to classical techniques in the quality of the approximation expected.

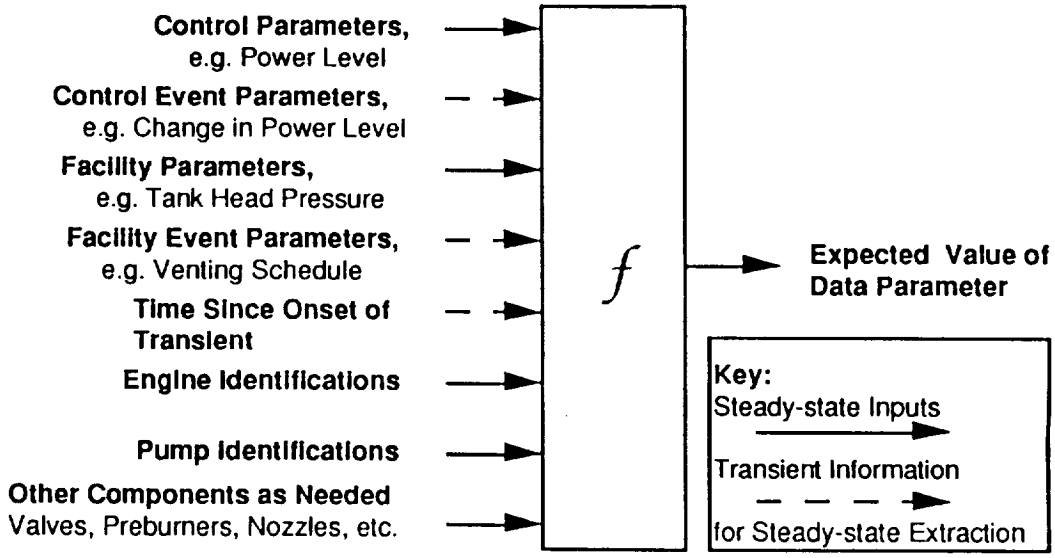


Figure 1. Neural Network Function Approximation

For the anomaly detection task, we trained the neural network function approximation using a representative sample of nominal steady-state sensor data from several different engines, for a range of power levels. The network then approximated the function f from time-series values of independent variables (inputs) to expected parameter values. In application these variables come from engine test data.

Figure 2 shows how such a trained neural network screens new test data for potential anomalies in a specific data parameter. Given time-series values of the independent variables (inputs), the

trained network approximates the function from these variables to the expected value of the data parameter under nominal conditions. This expected value is then compared to the actual value of the data parameter observed in the new test. Statistically significant deviations from the expected value are flagged as potential anomalies requiring further study by a human expert.

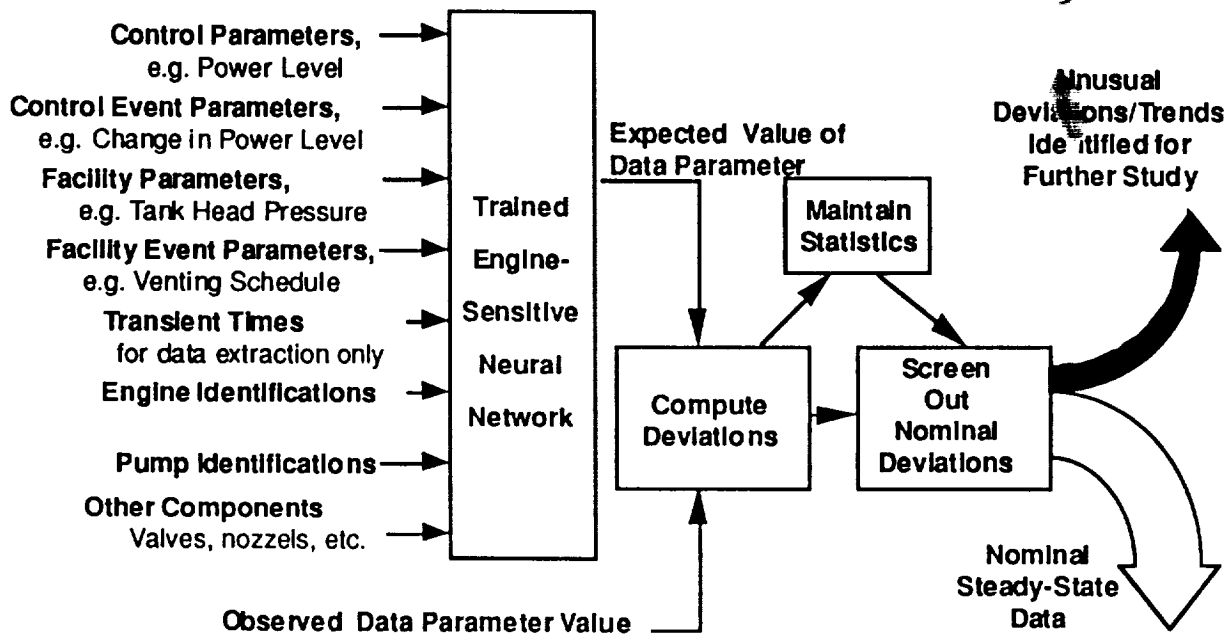


Figure 2. Use of the Trained Neural Network for Data Screening

In summary, the neural-network-based function approximation technique for data screening provides the following advantages:

1. The neural network learns to screen new data on the basis of prior training data.
2. There is no need to obtain representative training samples for all types of faults to be detected.
3. A new type of anomaly (one for which prior training data is unavailable) would be detected, provided only that it causes some parameter to differ significantly from the nominal value predicted by the network for the given test conditions.
4. While certainly not eliminating the need for expert human analysis, our neural network approach screens out a large proportion of nominal data, so that human resources can focus on the much smaller volume of potentially anomalous data.

Thus, the function approximation technique offers a great advantage to engineers—it significantly reduces the expenditure of analysts' time required to examine data from tests of current and future rocket propulsion systems.

2. Software Design and Development

2.1. Top-Level Design

A top-level design approach satisfied two of the requirements needed for successful Phase I completion of the project. Of utmost importance, the objective to achieve good screening performance drove our Phase I work. The objectives also included the ability for future adaptation to other propulsion systems. We considered this future requirement during the design process.

Basically, the criteria applied to the design resulted from the expectation that good screening performance would require substantial experimentation with the training and function approximation algorithms. We adopted a modular design that used object-oriented programming for the software development. This modular design allowed for easy modifications of the data screening system parameters and associated algorithms. The result produced a system capable of both efficient experimentation with the algorithms and possible adaptation to other propulsion systems.

Therefore, our design focused on the software attributes which facilitated experimentation with the screening process. Allowing experimentation gave us the ability to determine which inputs and algorithm definitions affected data screening performance the most. The software attributes which met these criteria included the following:

1. Compile-time modifiability (modularity):
   A "building-block" approach facilitates changes in neural network training and function approximation algorithms without major recoding each time the developer makes a change.

2. Run-time modifiability:
   At run time, the user can change specification parameters that define as much of the variable selection, preprocessing, training, and function approximation algorithms as possible. The user assigns values to these parameters in user-built specification files and command-line arguments. Therefore, the user can change and experiment with all properties of the algorithms which are parameterized in this way without any recompiling of code.

3. Information hiding:
   To the greatest extent possible, modules with well-defined access functions encapsulate all data structures and associated algorithms. The C++ object-oriented inheritance hierarchy supports this modular encapsulation. Each C++ building block hides the implementation details of a particular data structure or algorithm from the

rest of the software. This process localizes changes to one or a few modules, and minimizes the side-effects of changing any one module.

We performed code development using Sun Microsystems C++, Version 2.1, the PV-WAVE CL 3.10 script language, and Sun OS 4.1.1 UNIX scripts.

2.1.1. Object-Oriented Design

We used object-oriented programming to develop modular software using a building-block approach to achieve compile-time modifiability and information hiding as stated in (1) and (2) in the preceding section. This programming technique contributed to our ability to implement rapid prototyping of the data screening system. Also, as a result of our approach to top-level design using a common object-oriented software base, we worked several parts of the software development tasks concurrently. Figures 3a and 3b show the modular organization of our system as represented by its class hierarchy.

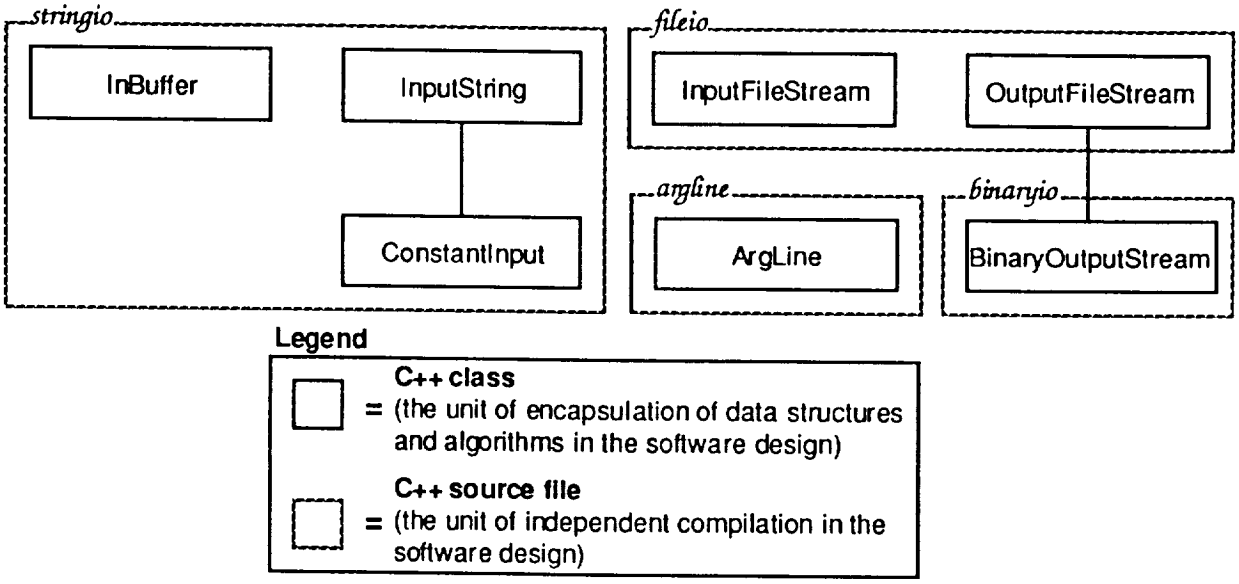


Figure 3a. Object-Oriented Software Design Inheritance Hierarchy Input/Output Files and Classes

2.1.2. User-built Specification Files

To complete the screening process, we constructed the system to access user-built specification files. These files allow extensive run-time modifiability as stated in (2) of Section 2.1 above. The following paragraphs explain, in brief, how these files provide for the experimentation process required of the software development effort. The explanations highlight some, but not all, of the user-built specification files accessed by the data screening system.

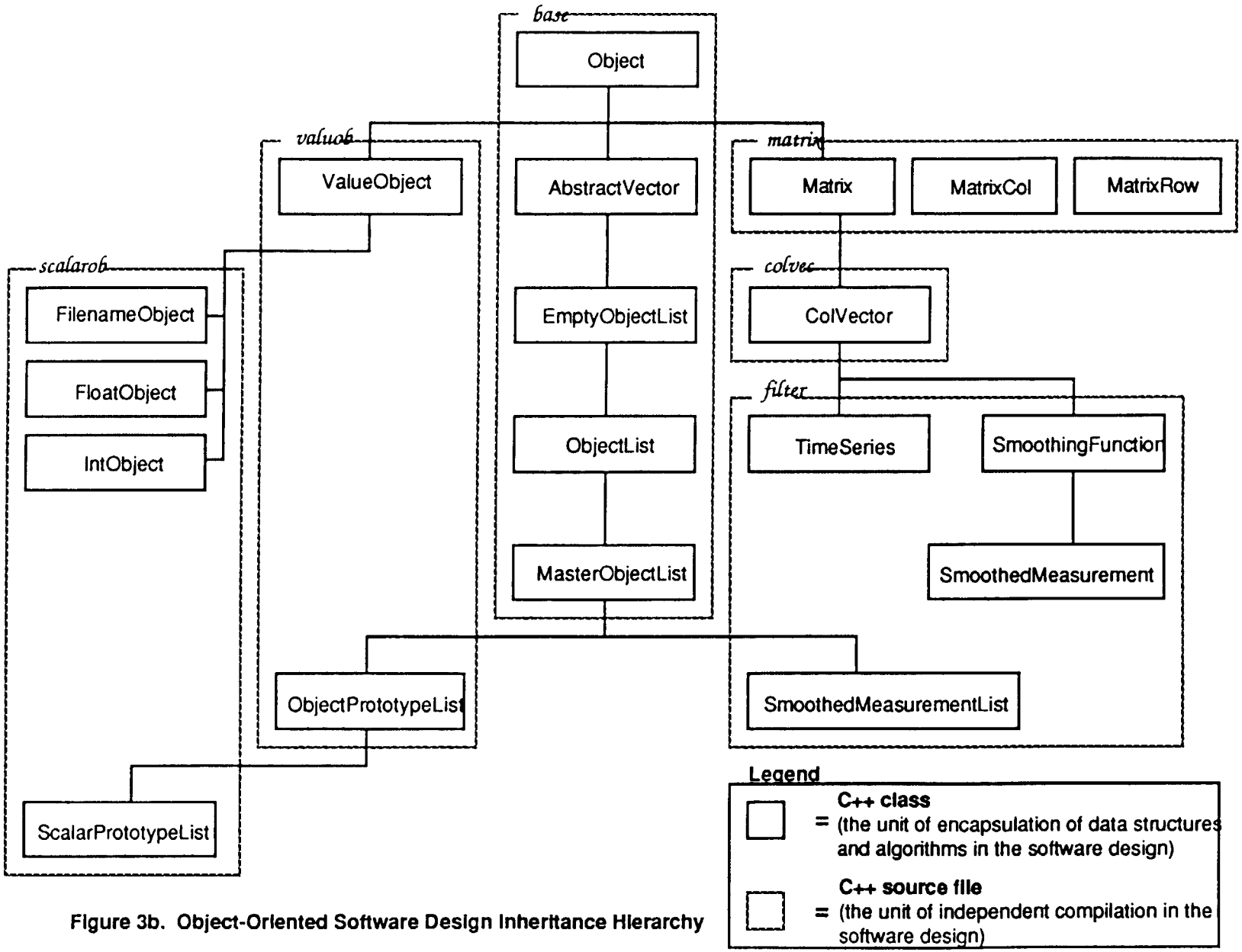


Figure 3b. Object-Oriented Software Design Inheritance Hierarchy

The Variable Definition File and Test Configuration Files enable the user to define variables of the following types to be used by the neural network in performing its function approximation:

1. Per-test variables to be used by the engine-sensitive neural network to discriminate performance characteristics of different engine configurations.
2. Per-transient variables to encode the control parameters and facility parameters to be used as independent variables for function approximation by the neural network.
3. User-supplied values of the per-transient variables for the steady-state segment of data following each transient.
4. User-supplied start and stop times for each controller-induced or facility-inducted transient.
5. Per-sample variables representing the dependent variables whose nominal values are to be predicted by the neural network. These are divided into the Controller Automated Data Acquisition System (CADS) and facility parameters to mirror the organization of the NASA databases. The screening system also uses these databases as an additional source of independent variables, reducing the number of values which must be provided by the user in (3) above.

The user builds a separate Variable Definition File, and hence a separate neural network to be trained, for each dependent variable to be screened. Transient information is included in each Test Configuration File and in the Phase I effort to enable the neural network training algorithms to ignore transient information for the steady-state-only data screening task.

2.2. Processing Modules

We built the data screening system as a sequence of separate processing modules. The steps needed for overall processing—from the raw NASA databases, through several preprocessing steps, into neural network training and screening, yielding statistical analysis and output—appear in Figure 4. This figure represents the sequence of steps needed to perform data processing, neural network training and/or data screening. It shows that the modular approach to the software development results in modular operation of the screening system as well. This means that, to experiment with different parameter settings, the user only needs to rerun the steps from the affected module forward, not the entire process.

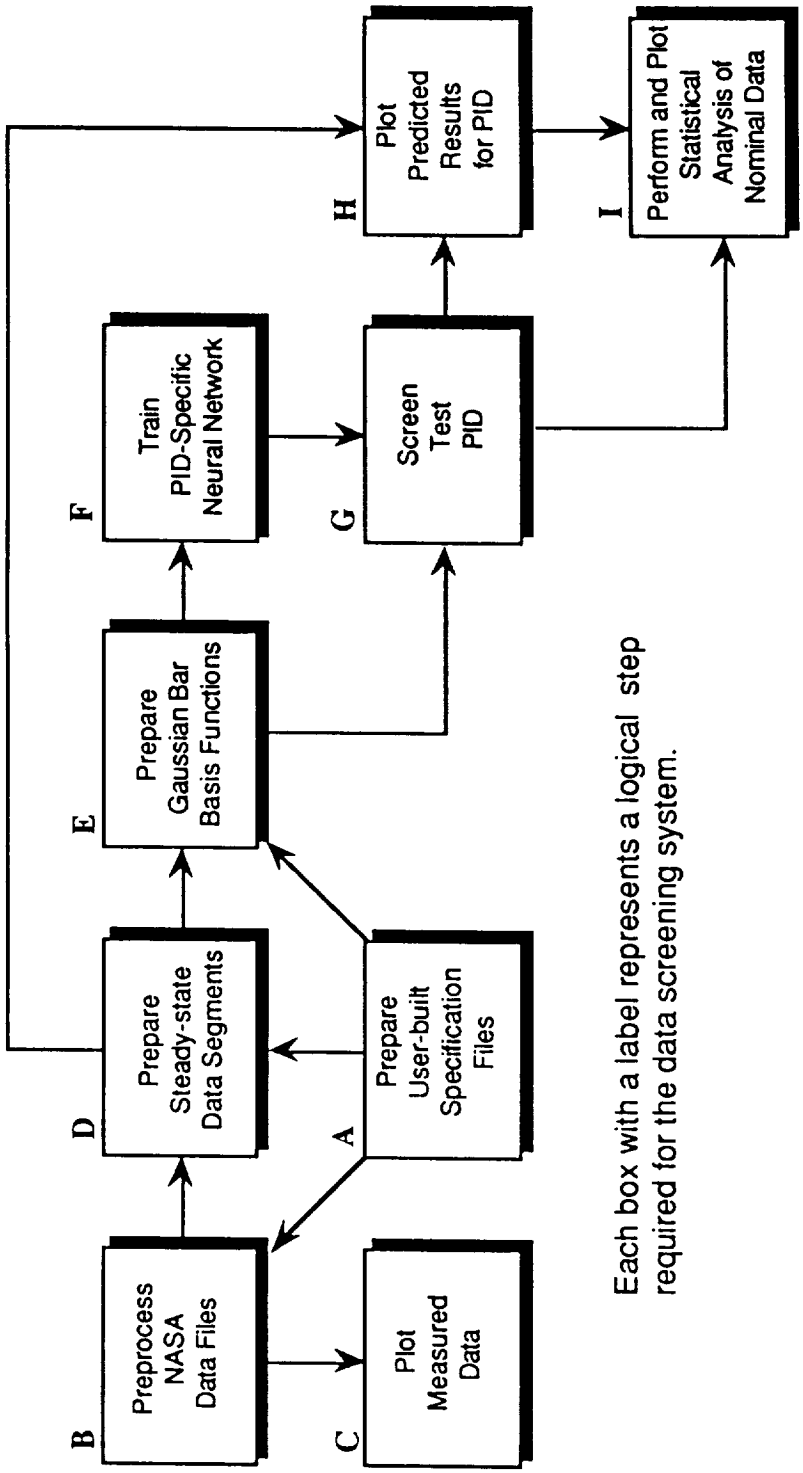


Figure 4. Processing Modules for the Data Screening System

Figures 6 through 12 (following) contain details of each step presented in Figure 4 below. Figure 5 (at right) presents a legend, for the symbols contained in the figures, that shows the details of each step of the screening system.

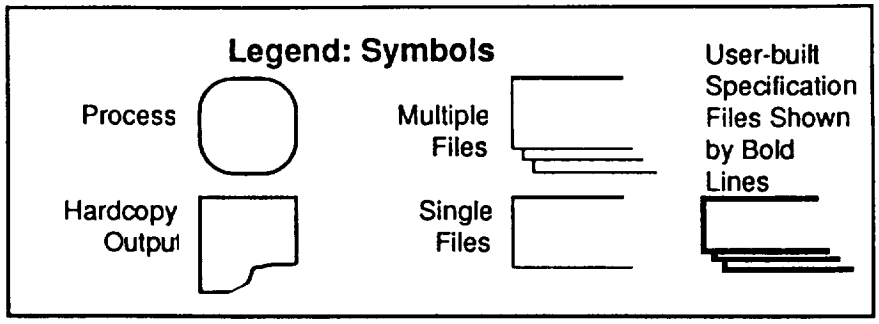


Figure 5. Legend for Symbols Used in Figures Showing the Steps of Each Screening System Module

2.2.1. Data Interface

In the NASA databases, the entire time series for each sensor is stored contiguously. Our software buffers and reorganizes this data by time step, so that measurements of all sensors at the same time step are stored contiguously. To minimize the need to reprocess the large NASA databases during experimentation, the software to interface NASA-supplied data to the neural network contains two modules, (1) Preprocessing and (2) Steady-state Data Extraction, as discussed below.

2.2.1.1. Preprocessing NASA Data Files

One software module processes existing NASA SSME test data into the organization and format needed for neural network training. Each parameter identification (PID) number needed for input to the neural network as an independent variable must be extracted from the NASA database. Also, the neural network training algorithm requires the time-series values of the PID to be approximated. After processing, the system stores the extracted and reorganized files in intermediate files so the user can experiment with different options for training and screening algorithms without reprocessing the large NASA database files. Figure 4 shows this preprocessing as the box labeled "B." Figure 6 expands this box into the specific steps taken by the software for data preprocessing.

The preprocessing module:

- Interfaces with the NASA-supplied database access code and the NASA-supplied CADS and facility data.
- Extracts data for those PIDs specified by the user in the Variable Definition File.

- Buffers and reorganizes this data from the NASA-supplied organization (all times for one PID are stored together) into the organization required by the neural network (all PID values for one time are stored together).

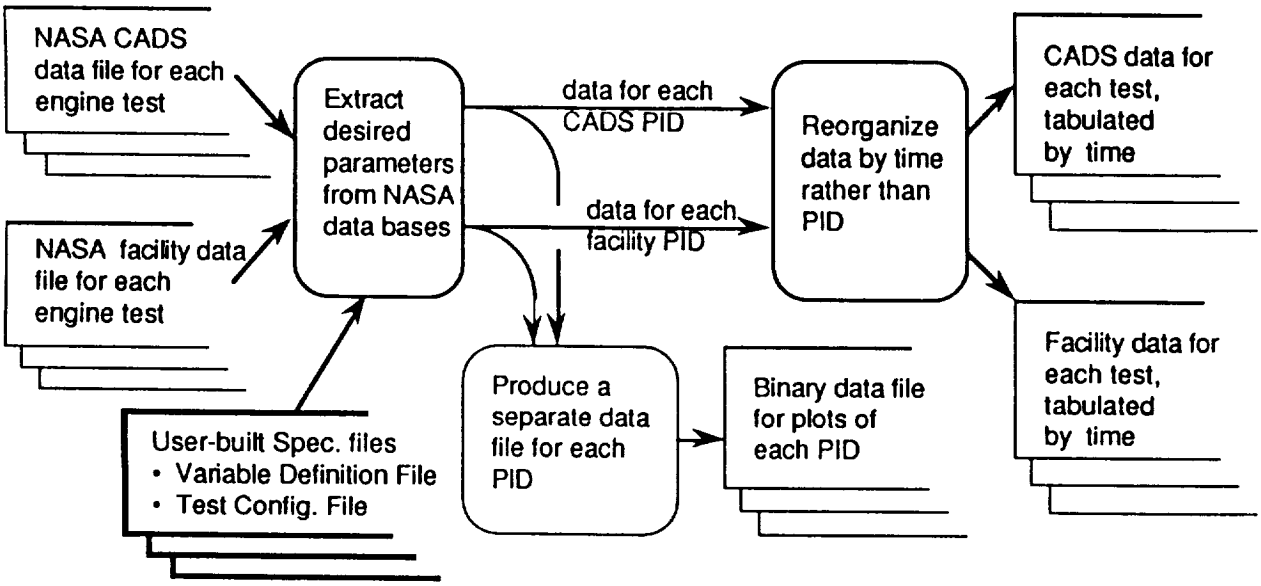


Figure 6. Steps in Module "B" to Preprocess NASA Data Files

NASA supplied the CADS and facility data on 1/4" tapes. Interaction with tapes is restricted to this module, "B." When the software is used in the NASA MSFC facility, the CADS and facility databases need to be on-line while module "B" is running. Results are stored in disk files containing the desired PID values which all modules that perform subsequent processing can access without further use of tapes. This module also builds a separate binary (time-series) file for each extracted PID in a form convenient for plotting by the data visualization software, PV-WAVE.

2.2.1.2. Steady-state Data Extraction

Module "D" extracts steady-state segments of data from a schedule of transients supplied by the user in each Test Configuration File. This module also synchronizes the per-test variables, per-transient variables, CADS variables, and facility variables to be input to the neural network. Figure 7 shows the steps taken by the software for steady-state data extraction, expanding box "D" of Figure 4. User-built specification files provide the following information for steady-state data extraction:

- Transient information to enable the neural network training algorithms to ignore transient information for the steady-state-only data screening task

- Extracted engine measurements needed by the neural network for training and screening

For each steady-state segment:

- Timestamps on the facility data are synchronized with timestamps on the CADS data.
- These synchronized CADS and facility data are extracted for each sample time within the steady-state segment.
- Per-test variables (in the case of engine-sensitive training or screening) for the current test, and per-transient variables (if desired) for the preceding transient are merged into each record of CADS and facility data.
- If a data set for training the neural network is being prepared, then the data records produced are randomly sampled to achieve the desired statistical representation of each steady-state segment in the total training set. In particular, only steady-state data known to be nominal is marked for inclusion in the training set.
- If a data set is being prepared for screening by a trained neural network, then all steady-state data segments are put into the data set for screening.

The results are saved in an intermediate file so that different subsets of the engine measurements extracted from the database can be selected for neural network training, without repeating the steps needed for extracting steady-state segments from the databases.

2.2.2. Neural Network

2.2.2.1. Architecture

The final neural network architecture consists of three layers:

1. Input layer: CADS and facility PIDs, and hardware identifications
2. Middle Layer: Gaussian bar basis function activations
3. Output Layer: Predicted value for a specific PID of interest.

Figure 7. Steps In Module "D" to Prepare Steady-state Data Segments

Figure 8 contains a graphic representation of the data screening system neural network architecture. This architecture resulted from the training algorithms employed to reduce the amount of computational time required for training. This type of neural network architecture based on Gaussian bar basis functions has been shown, in several training tasks, to converge at least an order of magnitude faster than traditional back-propagation techniques [10].

2.2.2.2. Training Algorithms

The function approximation behavior of the neural network resulted from a computation of Gaussian bar basis function activations, instead of weighted sums, for the connections from the first layer to the middle layer. We used a traditional weighted-sum calculation for the connections from the middle layer to the output layer of the neural network. Hardware inputs connected directly to the output layer through a weighting factor. We trained a separate neural network for each data parameter to be screened. However, each such network was trained to approximate the expected nominal value of that parameter over the range of data from different engines and different test conditions present in the training data.

Due to the use of the Gaussian bar basis functions, two processing modules, as represented by labels "E" and "F" of Figure 4, completed neural network training. The following sections explain the training process for each of the two modules.

2.2.2.2.1. Gaussian Bar Basis Function

We employed a Gaussian bar basis function neural network for our training algorithm. This method utilized a series of overlapping Gaussian distributions of trainable height to introduce non-linearity into the middle layer of the neural network. The Gaussian bar algorithm reduced training time as opposed to using a typical back-propagation technique with a non-linear squashing function. Figure 9 contains the steps needed to calculate the Gaussian bar basis function activations, labeled "E" of Figure 4. The following paragraphs explain the system's use of the Gaussian bar basis function.

In the Specification File for the number of Gaussian bar basis functions, the user specifies how many Gaussian bar functions N_j will be used to cover the range of possible values of each input PID x_j. For each input PID x_j, the software then calculates the minimum $\{x_j\}_{min}$ and maximum $\{x_j\}_{max}$ measured values of that PID in the steady-state data. The software then divides the range between the minimum and maximum into the specified number of equally-spaced centers, yielding an equally-spaced set of numbers

$$\mu_{j0}=\{x_j\}_{min}, \mu_{j1}, \mu_{j2}, \ldots, \mu_{jk}, \ldots, \mu_{jN_j} = \{x_j\}_{max} \quad (1)$$

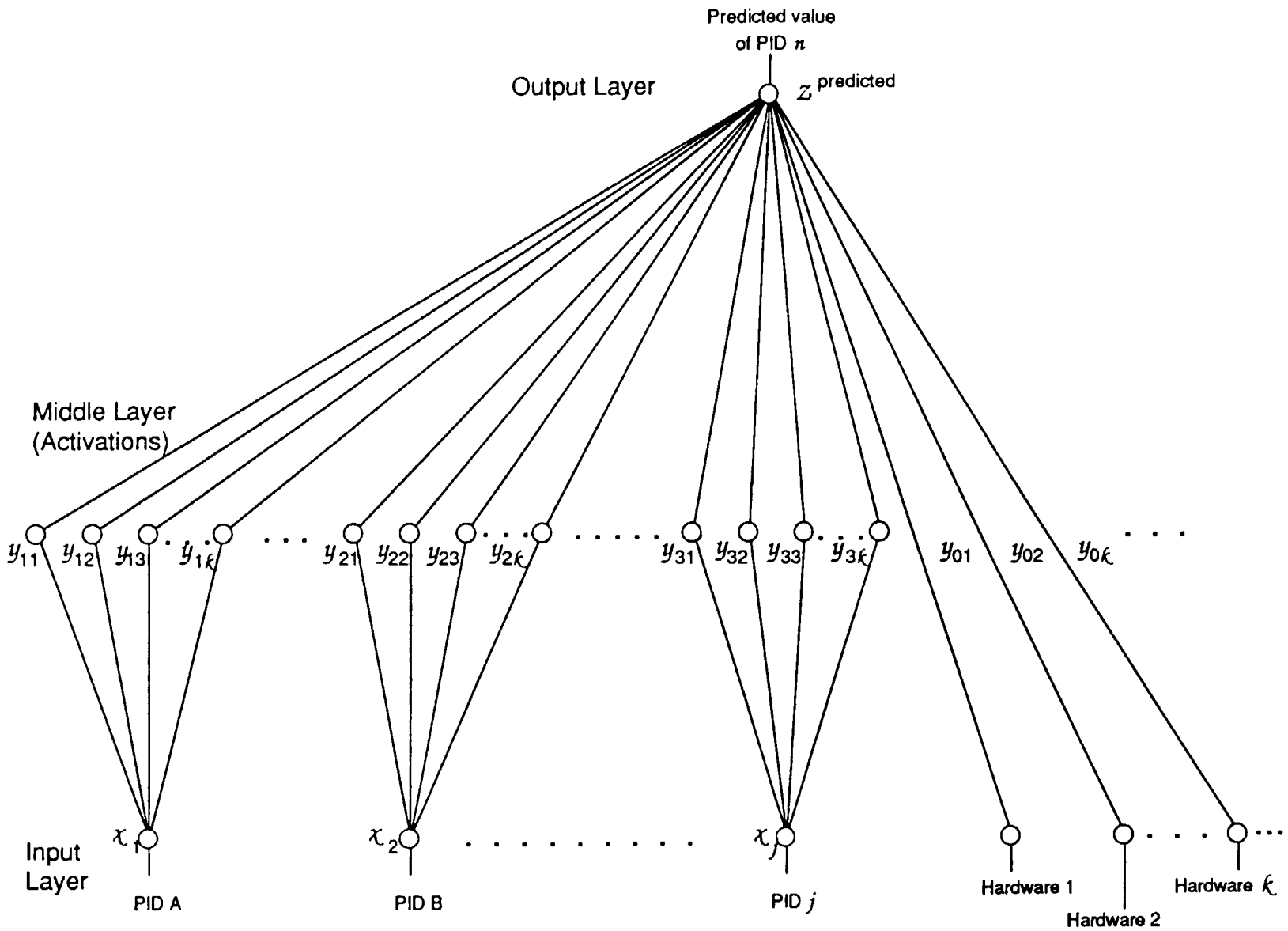


Figure 8. Neural Network Architecture for the Data Screening System

Each of these numbers μ_{jk} will serve as the center of a Gaussian bar function which responds to a subinterval of PID x_j centered around the value μ_{jk}. Each Gaussian bar function centered around μ_{jk} is represented by one node y_{jk} in the middle layer which has an activation function

$$y_{jk} = \exp\left(-\frac{(x_j - \mu_{jk})^2}{2\sigma_j^2}\right) \tag{2}$$

where each radius σ_j is chosen to be one-half the spacing between successive μ_{jk}'s in order to yield a series of overlapping Gaussian bell-shaped curves which span the range of the PID x_j. Since each weight from the middle layer to the output layer in effect multiplies its particular Gaussian bar function by a constant (the weight), the result is a set of overlapping bell-shaped curves of trainable height which has been shown mathematically [1,2] to be an excellent function approximation technique.

The centers and radii calculated by the software for each input PID are stored in a human-readable file, which in Figure 9 is termed "Definition of Gaussian bar basis functions and variables used for neural network training or testing."

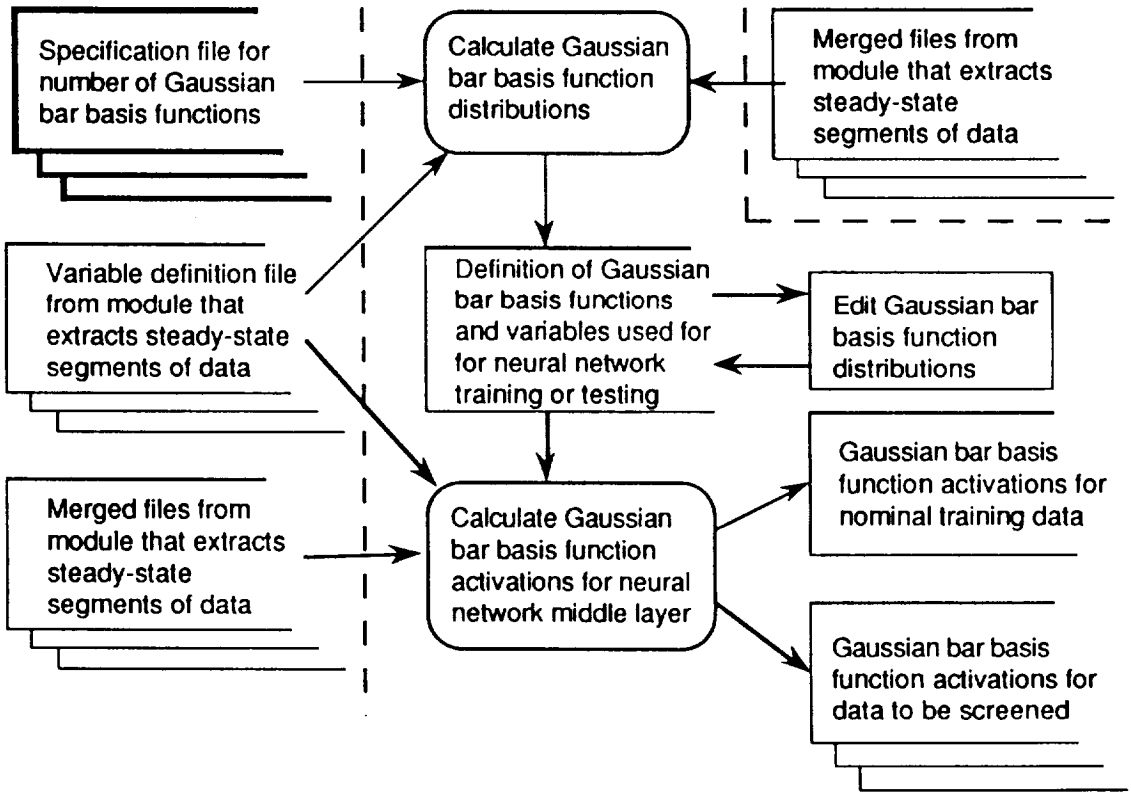


Figure 9. Steps in Module "E" to Prepare Gaussian Bar Basis Functions

Since in Phase I we dealt with steady-state data at a discrete set of power levels, many of the PIDs did not actually take on values over their whole range. Instead, there was commonly a gap between the highest value measured at one power level and the lowest value measured at the next power level present in the data. The middle-layer nodes y_{jk} whose centers μ_{jk} fell in these gaps were never activated, that is, the values y_{jk} given by equation (2) above were always zero for these nodes. While these nodes did not harm the training algorithm, they represent useless computation which slowed down the training and screening. Therefore, we gave the user the option of editing the file containing Gaussian bar definitions to eliminate those bars which fall in gaps in the range of data for each PID. This editing was optional in that it did not change the final results of training or screening, it merely accelerated the process.

In general, since each hardware component input has a value of either 0 (absent) or 1 (present), no Gaussian bar basis functions are used for the hardware inputs. The hardware inputs arbitrarily numbered.

$$y_{01}, y_{02}, \ldots, y_{0k}, \ldots \tag{3}$$

to distinguish them from the Gaussian bar basis functions and feed directly into equation (4) below.

2.2.2.2.2. Weights

As shown in Figure 10 (label "F" of Figure 4), after the Gaussian bar basis function activations are calculated, the neural network is trained, and the resulting weights are saved in a file. The algorithm calculates the weights by minimizing the mean squared error of the function approximation performed by the neural network. The activation of the output node z represents the value of the screened PID predicted by the neural network at each time step. This output is calculated as a weighted sum of the activations in the middle layer:

$$z^{\text{predicted}} = \sum_{j,k} w_{jk} y_{jk} \tag{4}$$

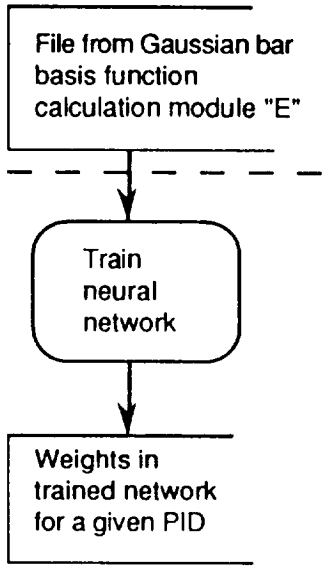


Figure 10. Steps In Module "F" to Train the Neural Network

Since each middle-layer activation is really a Gaussian bar function of the input, equation (4) actually produces the sum of a set of overlapping bars of trainable height, where the bar centered at μ_{jk} has height w_{jk}. As mentioned in the section above, this set of overlapping bell-shaped curves has the ability to approximate an arbitrary non-linear function for

each input PID. This approximation is formed by calculating the error between the predicted and measured values for each training sample. The mean-squared error is then minimized by a standard gradient descent rule:

$$w_{jk}(t+1) = w_{jk}(t) - \alpha \cdot \left(z^{measured} - z^{predicted}\right) \cdot y_{jk} \qquad (5)$$

where α is the rate of descent of the gradient (commonly called the "learning rate" in neural network applications of gradient descent). During training, equation (4) is used to calculate the prediction $z^{predicted}$ made by the neural network, and then equation (5) is used to change the weights. This process is iterated for each training sample for each pass through the training data.

The result of training is a set of weights w_{jk} which perform the best function approximation (in the least-mean-squares sense) that can be obtained from the given set of Gaussian bar functions. This set of weights is saved in a file for use in screening new data.

To use the trained network for screening, the weights that resulted from training are read from a file. The function from input PIDs and hardware to output PIDs is then given by equations (3) and then (4). Taken together, these equations transform each input PID by an arbitrary non-linear function of the overlapping bell-shaped curves given in (3), where each bell-shaped curve centered at μ_{jk} receives a height w_{jk} by the training algorithm. For computational efficiency, the calculation of the Gaussian bar basis function activations of the middle layer is performed for all the data in module "E" and the results are then fed into module "F" which calculates the prediction of the neural network (i.e. its output layer activation) as a weighted sum of the Gaussian bar activations given in the middle layer. The neural network output therefore represents the expected nominal value of the output PID predicted by the neural network from the input PIDs at each time step. The deviation of the actual measured value from the expected nominal value is then the basis for screening the data, as explained in the next section.

As stated before, in this Phase I project, we held a separate neural network training session for each PID to be screened.

2.2.2.3. Screening

We discussed the method we used for screening in Section 1.3.2. The design and software development for the screening module results from our training algorithm method presented in the preceding section. Figure 11 shows the steps taken by the software to perform actual data screening, label "G" of Figure 4.

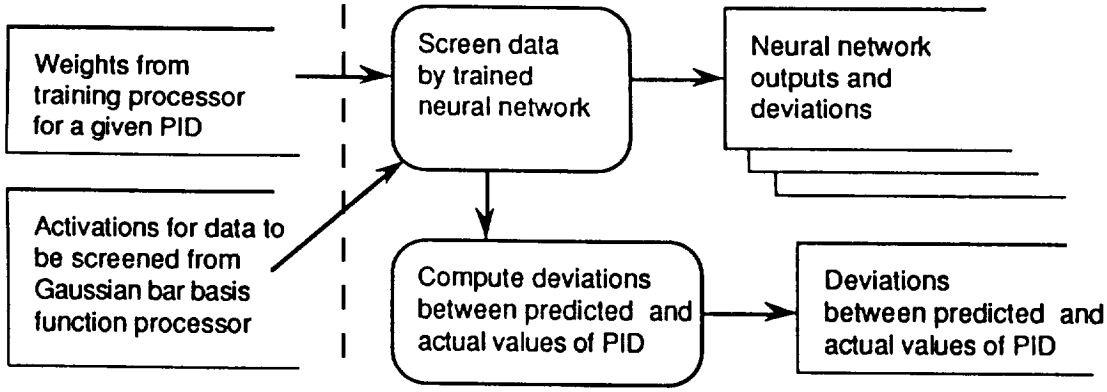


Figure 11. Steps In Module "G" to Screen Test Data for a Specific PID

2.2.3. Statistical Analysis

We developed software routines to perform the statistical analysis of the deviations of actual engine parameters from the expected engine parameters output by the neural network. Figure 12 shows the steps taken by the software to perform the statistical analysis, label "I" of Figure 4. The phase I system outputs a plot showing statistical information for analysis by engineers. The following paragraphs describe how the statistical analysis software produces the plots for analysis.

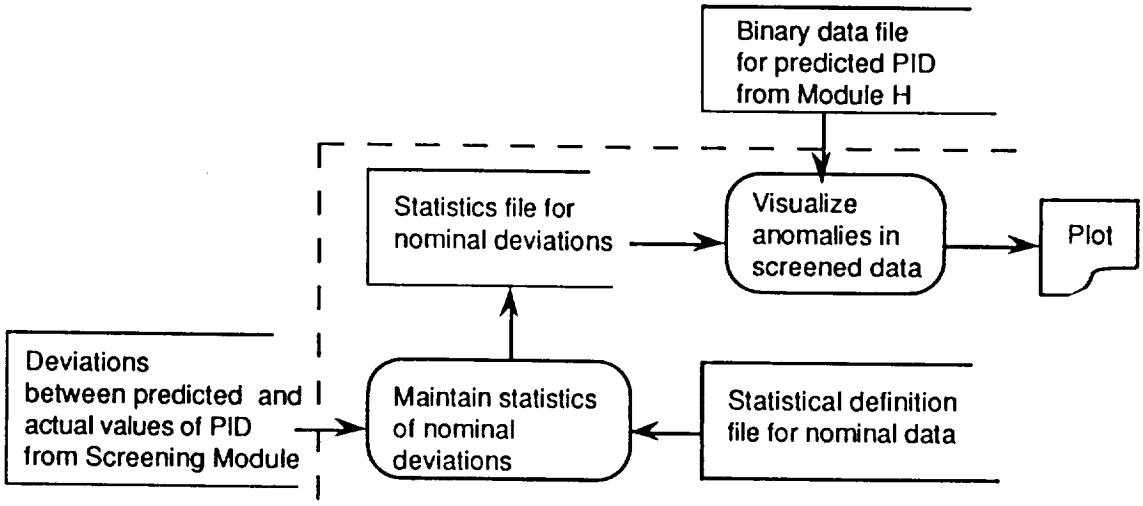


Figure 12. Steps In Module "I" to Perform Statistical Analysis of Test Data

After neural network training, first this software runs (screens) the training data back though the trained neural network. It then calculates the expected magnitude of deviations between predicted and measured values over the "screened" nominal training data deviation for nominal data.

Statistical calculations use a user-defined sliding window of deviations of the measured values from the expected values produced by the neural network. We tested two- and six-second window sizes. The size of this sliding window can be altered at run time without recompiling the code.

The input to the statistical analysis consists of the successive deviation results produced by sliding the window forward in 0.2-second increments and calculating the average deviation over each new window. The 0.2-second increment can also be changed without recompiling the code. The algorithm then calculates the standard deviation of the resulting time series of window deviations over the nominal training data. We termed this the "nominal sigma" for the parameter.

After completing this calculation for the nominal training data, engine tests are screened by calculating the output of the neural network at successive 0.2-second intervals and comparing the expected nominal values output by the neural network with the actual measurements. The deviations between actual and expected are then calculated over the same 2-second sliding window mentioned above, advancing in increments of 0.2-seconds (these constants again are alterable at run time). Finally, a plot (Figure 13) is produced containing the following time series of data:

a. Expected nominal value of the parameter output by the neural network, shown for each 0.2-second increment of the 2-second sliding window

b. Expected nominal value plus five nominal standard deviations (i.e. the "plus-5-sigma line")

c. Expected nominal value minus five nominal standard deviations (i.e. the "minus-5-sigma line")

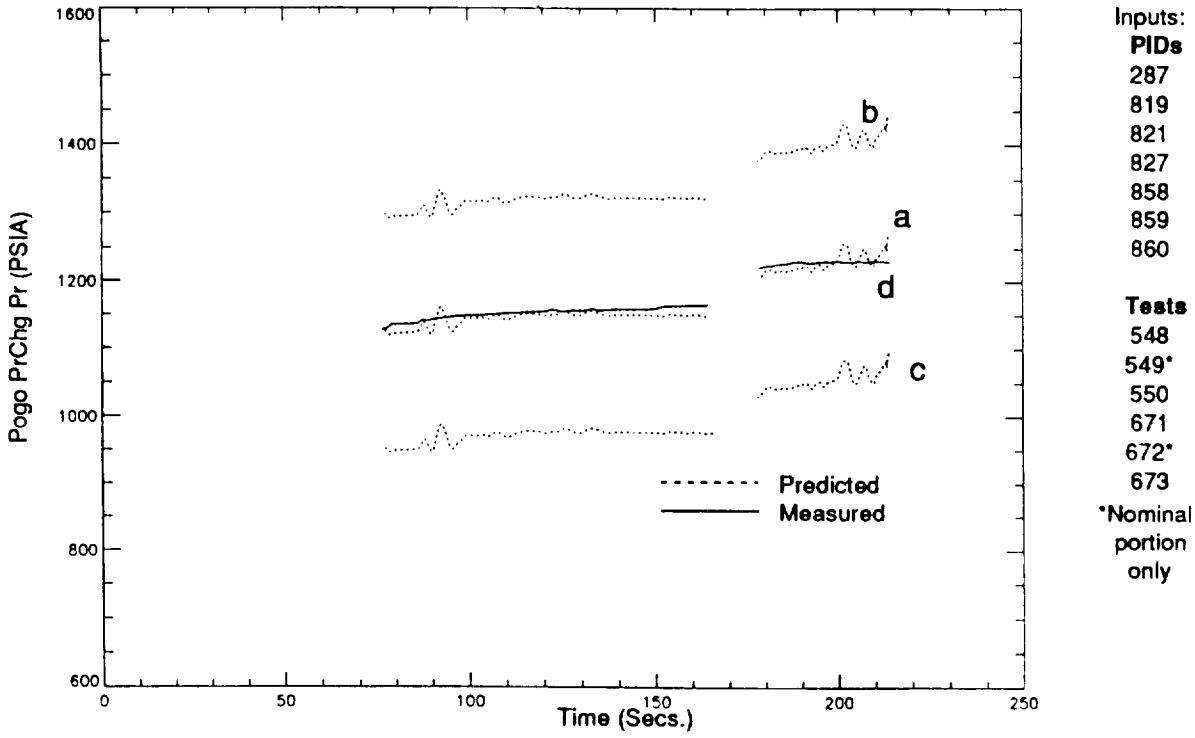


Figure 13. Example of Statistical Analysis Error Bands

d. The actual measured value of the parameter, shown for each 0.2-second increment of the 2-second sliding window for comparability with (a), (b), and (c)

The space between (b) and (c) on the plot is considered to be the band of nominal deviations from the expected values produced by the neural network. The definition of this nominal band as plus or minus five nominal standard deviations is, again, alterable without recompiling of the code. The lines on the plot which define this nominal band [(a), (b), and (c) above] are shown as dotted lines on the plot. The actual measured values over comparable windows [(d) above] are shown as a solid line on the plot. The user can then identify a potential anomaly as any time interval on the plot during which the solid line (representing actual measurements) crosses above the plus-five-sigma line or below the minus-five-sigma line.

2.2.4. Data Visualization

Through the use of PV-WAVE and associated scripts the system performs plotting functions at several places within the data screening system. To facilitate the neural network experimentation process we needed data visualization for the following tasks:

- Creation of user-defined files by plotting specified engine measurements
- Analysis of measured input engine parameters
- Analysis of tests with known anomalies since the system must not be trained with anomalous data
- Visualization of the function approximation surfaces for subsequent refinement of the baseline neural network and training algorithms
- Output of results from the neural network for the PID to be predicted
- Output of the statistical analysis data.

Hard copy plots are produced during the operation of processing modules "C," "H," and "I" as labeled in Figure 4.

3. Screening System Implementation

3.1. Data Selection

3.1.1. Engine Tests

NASA engineers and support contractor personnel together with ERC and subcontractor personnel chose which SSME tests and which sensors to use for neural network training and testing for

Phase I. Initially, three tests[3] were chosen. Two tests, 901-671 and 901-673, contained nominal data and the third test, 901-672, contained anomalous data. Sensors appropriate for detecting this anomaly were identified. We later acquired three more tests from the same engine: 902-548, 902-549 and 902-550; for further neural network training. The combination of these additional tests and the tests used for initial training constituted a large sample of the nominal training data for the Phase I work. The thrust-level profiles for all tests appear in Figure 14.

3.1.2. Initial Input Parameter Selection

NASA and contractor personnel selected the parameters to be used as independent variables for inputs to the neural network. The initial parameters chosen for neural network training and testing included the power level, facility venting schedules, and fuel and oxidizer inlet conditions. We selected the PIDs listed in Table 1 for initial training and testing.

Table 1. Initial Independent Parameters for Neural Network Training and Testing

PID No.	Description	PID No.	Description
287	PC CNTL REF	858	ENG OX IN PR 1
819	ENG FL IN PR 2	859	ENG OX IN PR 2
821	ENG FL IN PR 1	860	ENG OX IN PR 3
827	ENG FL IN PR 3		

We adopted the criteria that any controller input and physical engine interface qualified as a candidate for an independent variable (input). We regarded the engine as two-sided: oxidizer side and fuel side. In later meetings with NASA we discussed other input parameters for refining and testing the neural network. Section 3.4, Refinement and Testing, discusses the additional inputs used in an effort to refine the neural network outputs (predictions).

3.2. Initial Training and Testing

We completed initial neural network training for predicting parameter identification (PID) 42, fuel preburner oxidizer valve position (FPOV), as specified in Table 2 below.

Table 2. Initial Training Tests and Inputs for Predicting PID 42

Input Tests	671	672	673				
Input PIDs	287	819	821	827	858	859	860

[3]Test numbers preceded with 901 indicate the location of the test in California, while the prefix, 902, indicates a location at NASA Stennis . Henceforth, we will refer to all tests used in Phase I without the prefix 901 or 902.

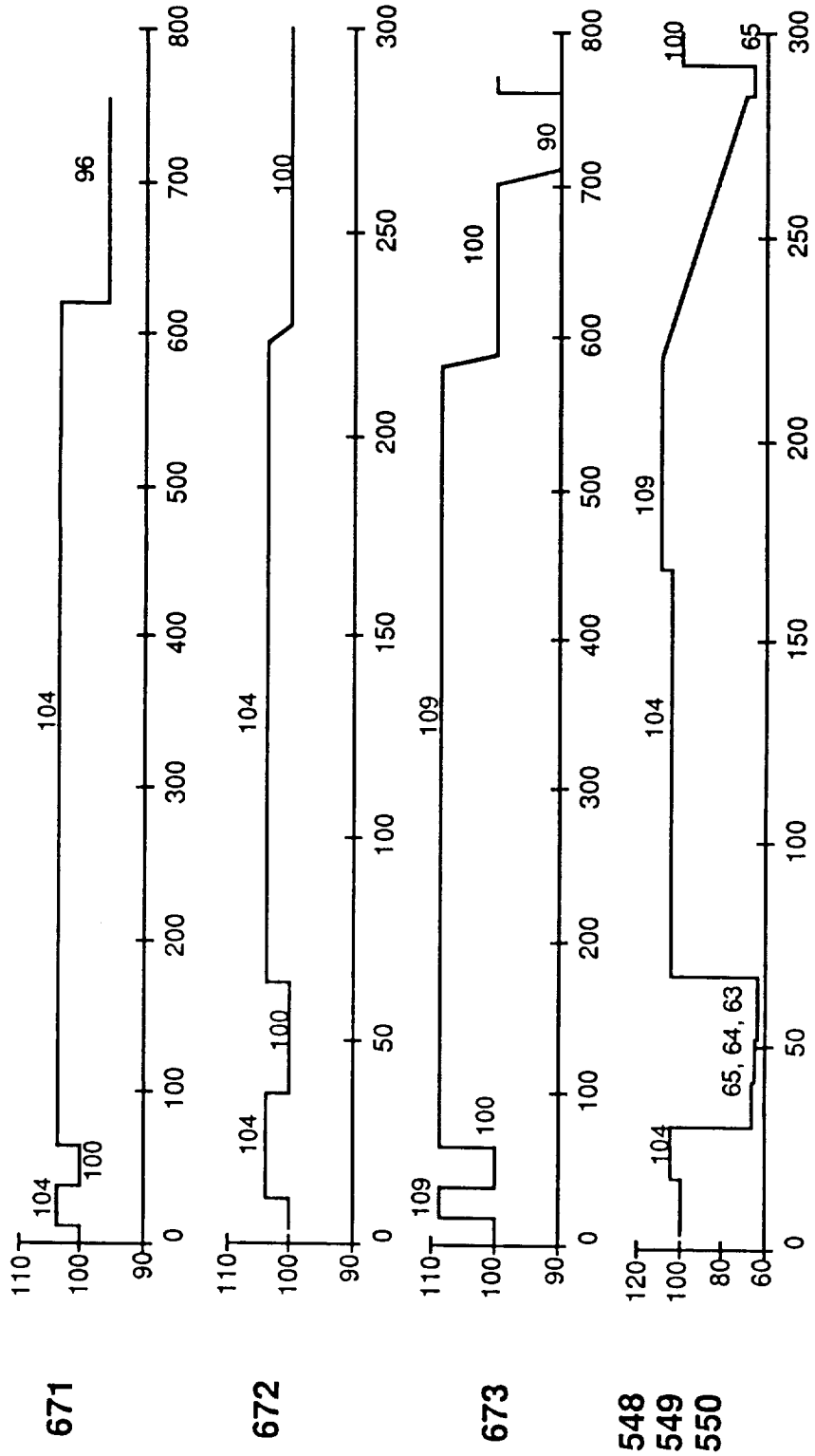


Figure 14. Thrust Level Profiles of Tests Used in Phase I

Since Tests 671 and 673 were nominal tests, we used nearly all of the steady-state data from these two tests for the training input. For Test 672 we trained up to the last transient that occurred prior to the anomaly. The anomaly in PID 42 (FPOV ACT POS A) occurred approximately 95 seconds into the test. By not using data past the last transient prior to the anomaly in Test 672, we ensured that neural network training used only nominal data. This initial training used 1000 iterations.

After training, we used the neural network to screen all of the data from Test 672[4]. The results from the screening run appear in Figure 15. This plot clearly shows that the predicted values for PID 42, based on the input PIDs, are different from the actual (measured) values once the anomaly occurred. These initial results supported our theory of training with nominal data only, rather than training with known anomalies. Figure 15 also shows how our system predicted that the steady-state values of PID 42 would remain within a given range (plateau) during operation from approximately 70 to 220 seconds.

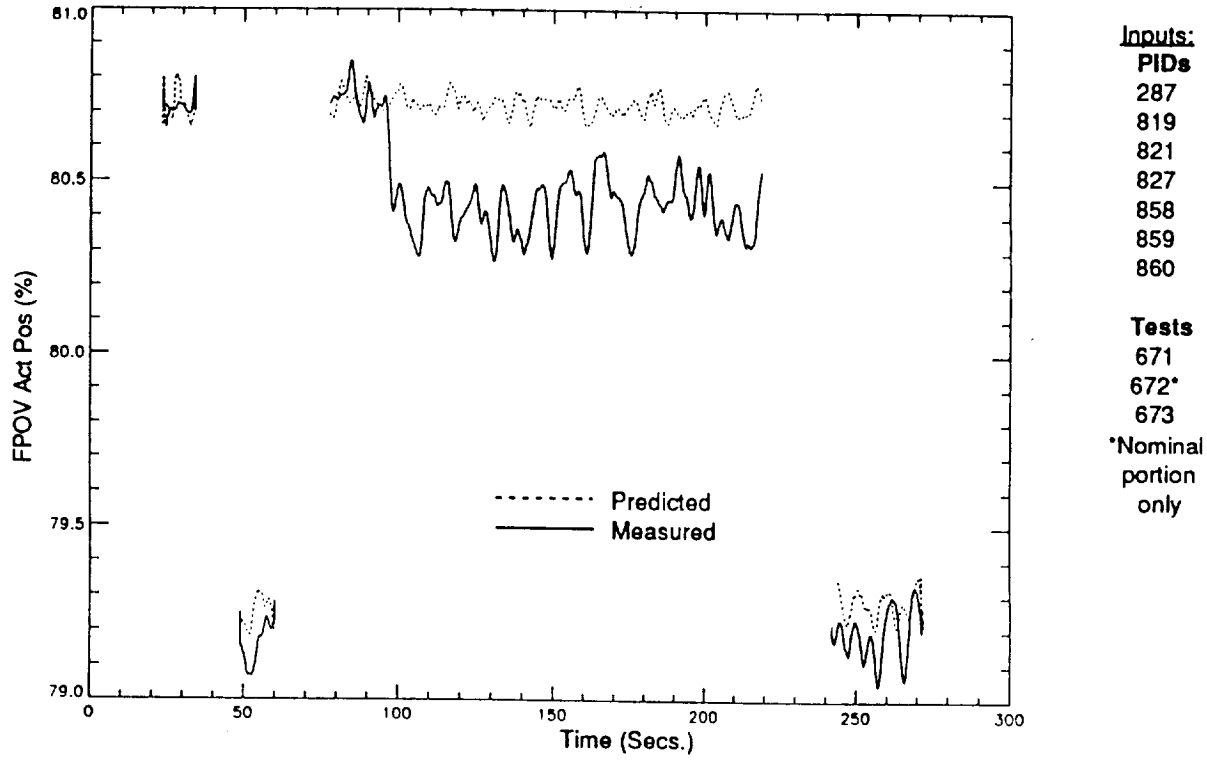


Figure 15. Initial Screening Results, PID 42

[4]Note that the plot does not show continuous lines of data for the predicted or measured value of the PID. We purposely omitted transient data since Phase I did not use any transient data for training or screening. All subsequent plots also omit transient data.

3.3. Confidence Building

To verify that our system trained according to the data from the input tests, we screened both nominal tests 671 and 673 using the initial training output files. The results presented in Figure 16 show that our neural network training produced a prediction that matched the input data sufficiently.

After the initial research, we tested our methodology by training two other PIDs: 40, Oxidizer Preburner Oxidizer Valve Position (OPOV); and 8, mixture ratio (MXRT). Figure 17 shows the prediction made for PID 8. Although the tests used for training did not contain an anomaly in PID 8, we trained to show our system's ability to learn and predict the value of additional PIDs. The prediction of PID 40 (Figure 18) experienced less success. However, as a prototype, the system showed that our methodology works.

We looked into why the PID 40 prediction did not work as well as those for PID 8 and 42. Initially, we hypothesized that the neural network needed more independent parameters to predict this PID properly. NASA personnel suggested that prediction of PID 40 may require additional independent parameters to the neural network such as the pressurization flow control valve position and data from repressurization interfaces.

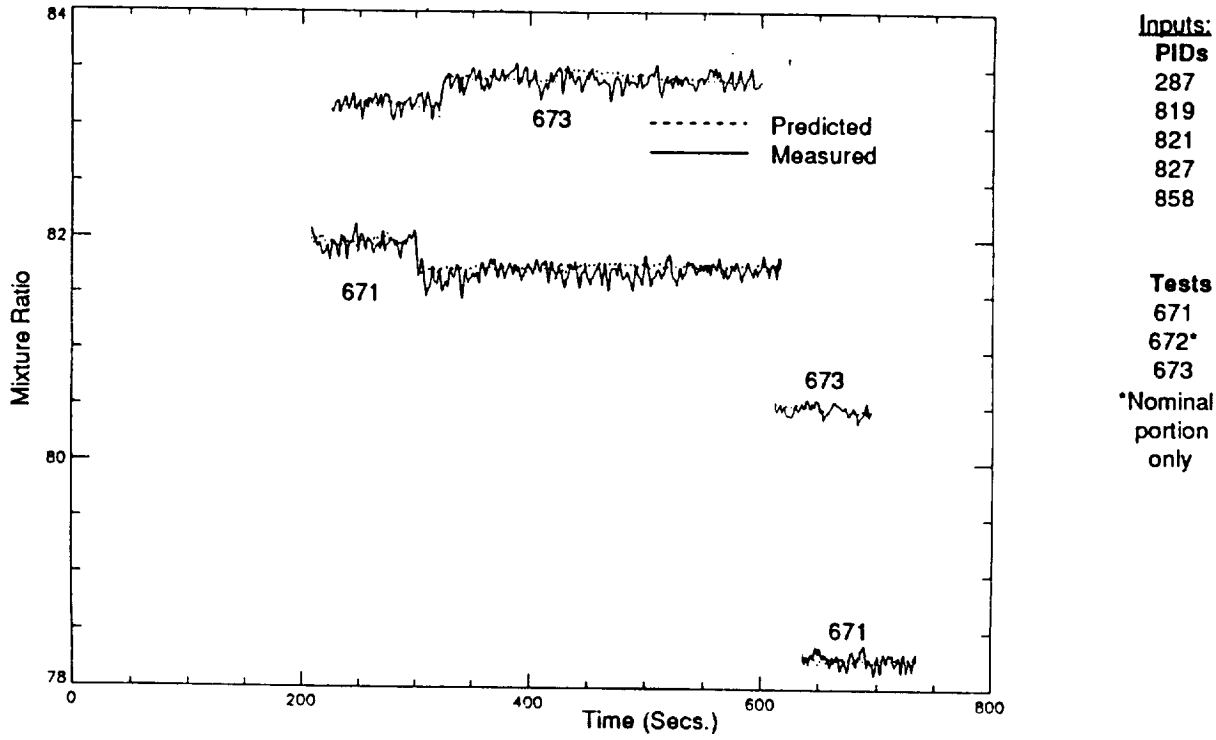
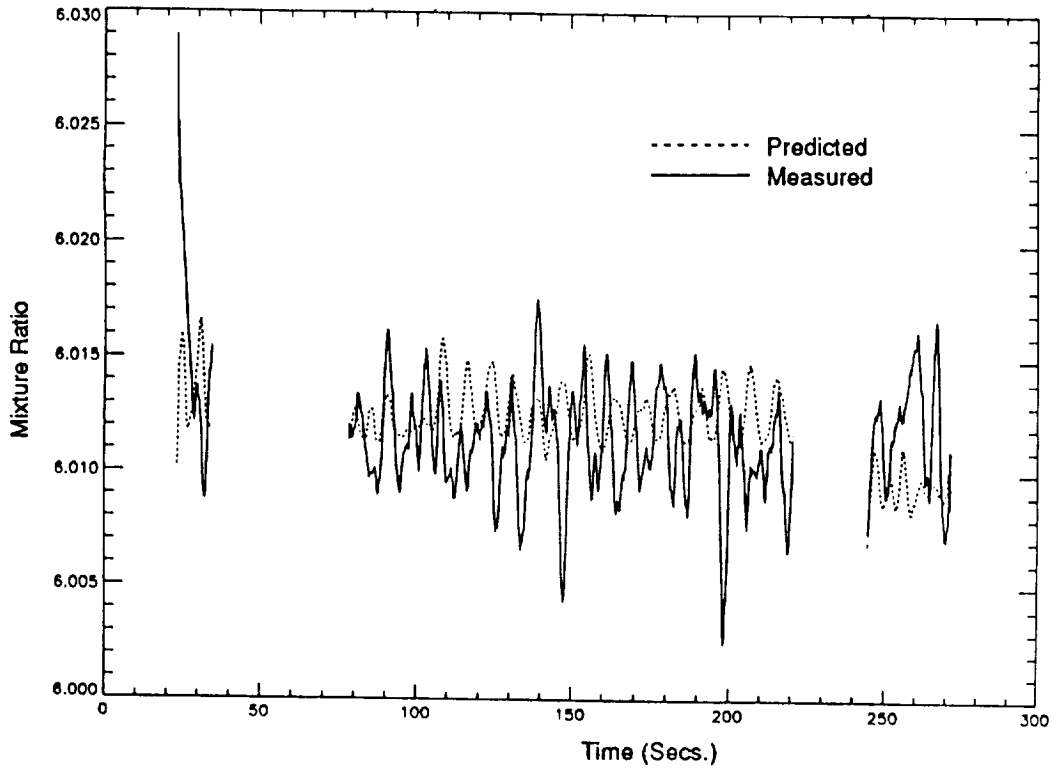


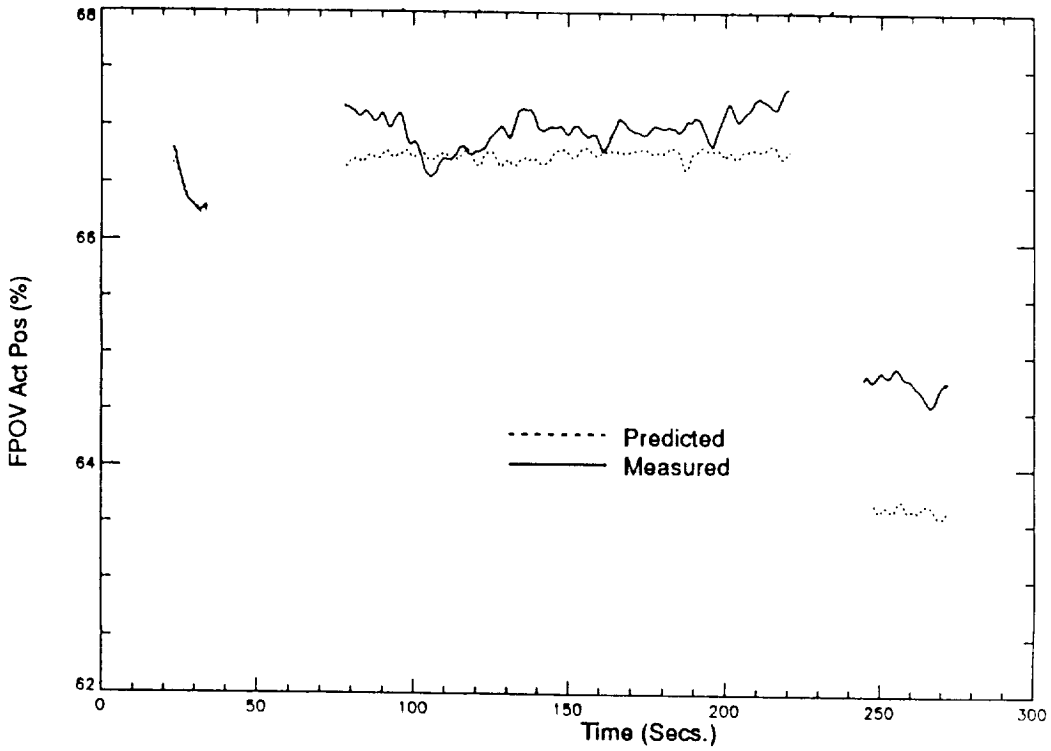
Figure 16. Prediction Results for Nominal Tests 671 and 673, PID 42



Inputs:
PIDs
287
819
821
827
858

Tests
671
672*
673
*Nominal portion only

Figure 17. Initial Screening Results, PID 8



Inputs:
PIDs
287
819
821
827
835
858

Tests
671
672*
673
*Nominal portion only

Figure 18. Initial Screening Results, PID 40

3.4. Refinement and Testing

Based on the initial results and confidence building runs, we began refining and testing the neural network by experimentation using different sets of nominal and anomalous data. We conducted studies by individually changing the following inputs or algorithm definition parameters by:

1. Varying the Gaussian bar basis function spacing over the data range for each input PID
2. Varying the number of neural network training iterations
3. Varying the number of input PIDs (independent variables) at the fuel and oxidizer inlets
4. Adding PIDs for the fuel and oxidizer repressurization interfaces to the input PID list
5. Changing the definition of a facility transient

Table 3 presents a summary of the PID to be predicted and the number of times each test was used as a data set for neural network training runs that we conducted during Phase I. We made a total of 30 training runs during Phase I.

Table 3. Test Usage for Neural Network Training Runs

Predicted PID No.	Input Test # 548	549	550	671	672	673
8				1	1	1
40				3	3	3
42*	3	3	3	20	**20***	19
221*	3	**3***	3	3	3	3

*Indicates PID and test where anomaly occurred.

3.4.1. Variation of Gaussian Bar Basis Functions

We varied the spacing of the Gaussian bar basis functions to test the neural network's sensitivity to the distribution of these basis functions. The user can specify the spacing of the basis functions for the independent input variables. Figure 19 shows the difference of increasing the total number of basis functions from 70 to 100. We made the change by increasing the number of basis functions for PID 819 from 30 to 60. The predictions improved as a result of this change. Since the middle layer of the neural network, and thus the amount of training time, depends on the number of basis functions specified we limited most training sessions to 370 of these basis functions. We trained with as little as 30 to as many as 430 basis functions.

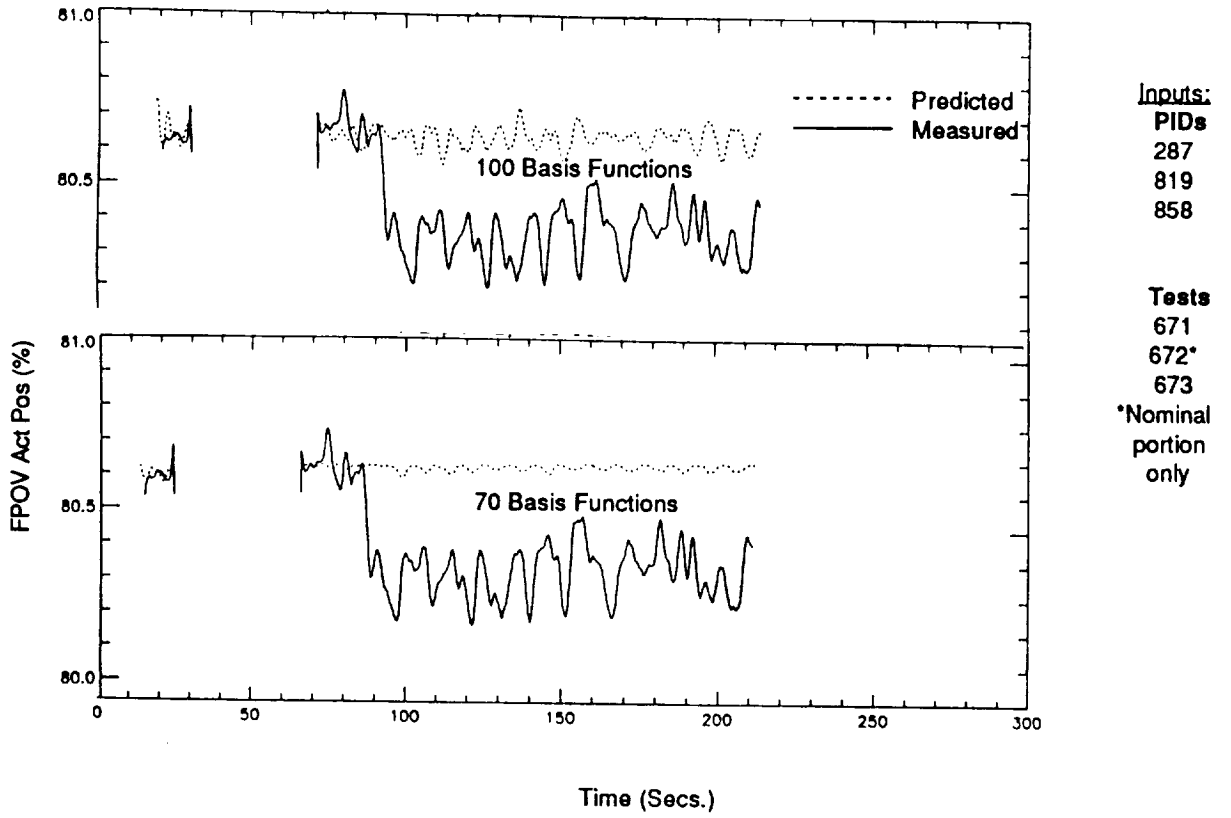


Figure 19. Change in Gaussian Bar Basis Function Specifications, PID 42

3.4.2. Change in the Number of Training Iterations

We conducted training runs to test how the number of training iterations affects the prediction results. Figure 20 shows that an increase of 100 iterations, from 300 to 400, improved prediction results for PID 40, Test 673, as shown by the labels A and B. The amount of training time directly depended on the number of iterations specified. We trained with as little as 100 to as many as 2000 iterations. We chose a certain number of iterations for a particular session based on the training time required--typically overnight or over a weekend.

3.4.3. Change in Input Parameters

In an effort to refine the neural network function approximation, we experimented with the independent variables used as the training data set. We held a meeting with Michael Whitley and Marc Neely of NASA MSFC and discussed other possible PIDs for use as input to the neural network. ERC engineers suggested that the bleed and repressurization line sensor measurements be included as independent variable inputs. After Mr. Neely explained which lines were completely shut during engine operation, we concluded that PIDs 835 (FUEL PRESS INT PR), 878 (HX INT PR) and 879 (HX INT T) presented good candidates for inclusion in the input layer of the neural network.

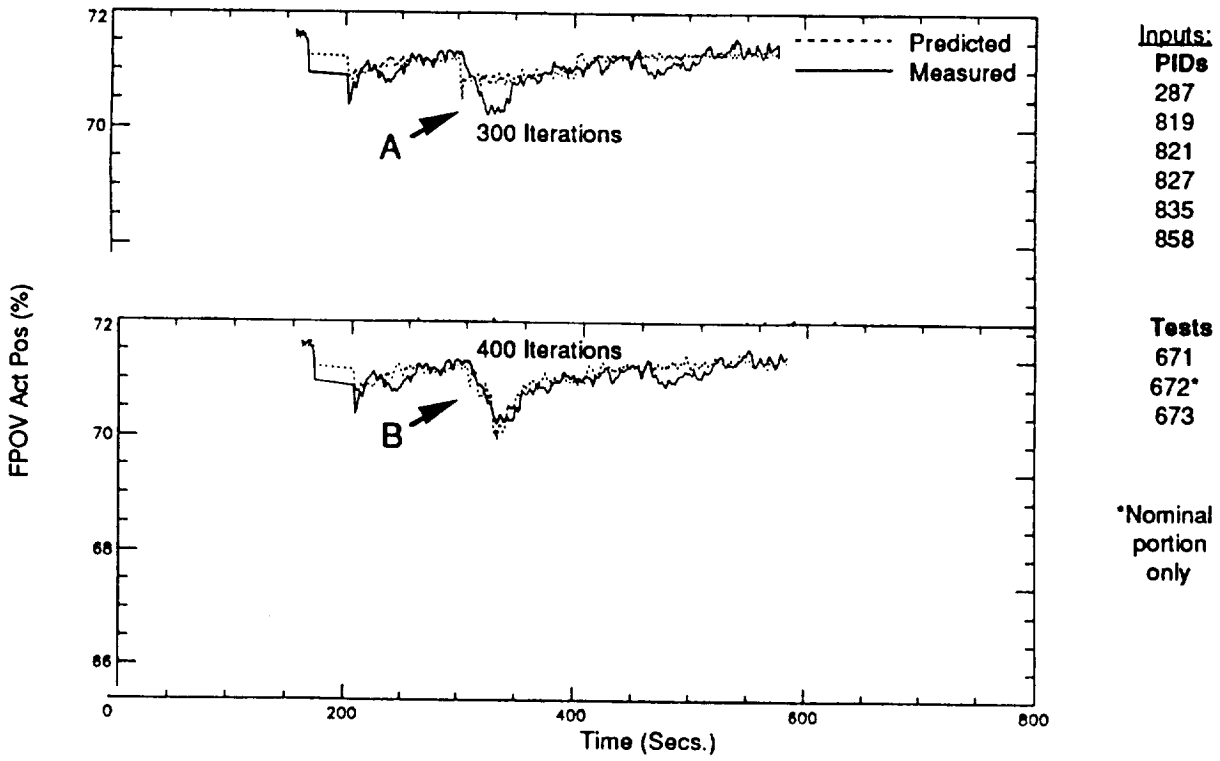


Figure 20. Change in Iterations, PID 40

3.4.3.1. Input Parameter Reduction

Based on our successful initial results, we reduced the number of input PIDs from seven to three. We made this change to test the sensitivity of neural network training to reduced inputs. We tested as shown in Table 4.

Table 4. Parameters with Reduced Input Training Data Set

	Initial Runs	Reduced Inputs
Controller	287	287 (no change)
Fuel Side	819, 821, 827	819
Oxidizer Side	858, 859, 860	858

Figures 21a and 21b show that reducing the inputs from seven to three PIDs still produced reasonable predictions for PID 42. We surmised that since the PIDs removed represented data from the same engine interfaces (fuel and oxidizer inlets) that the PIDs supplied redundant information. However, when included in the training data set the additional PIDs provided better fidelity between the measured and predicted values of PID 42.

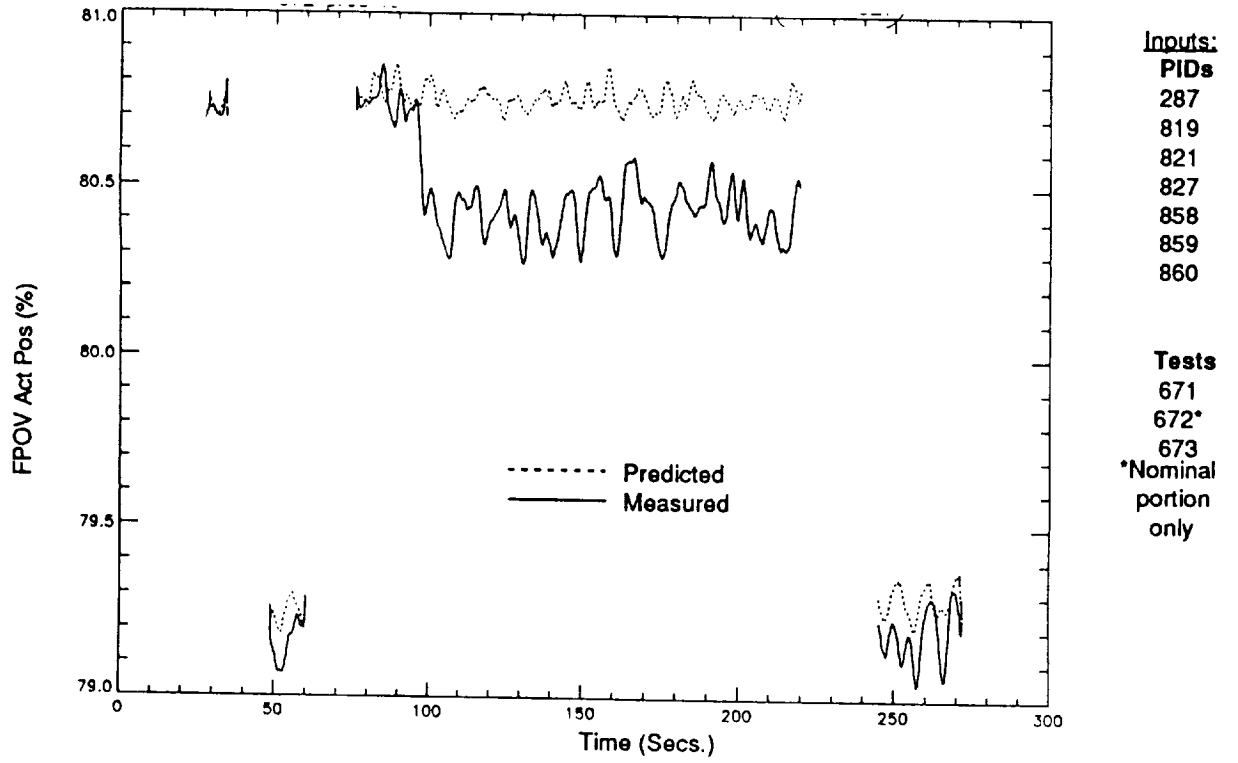


Figure 21a. Reduction in Input PIDs: Trained with Seven, PID 42

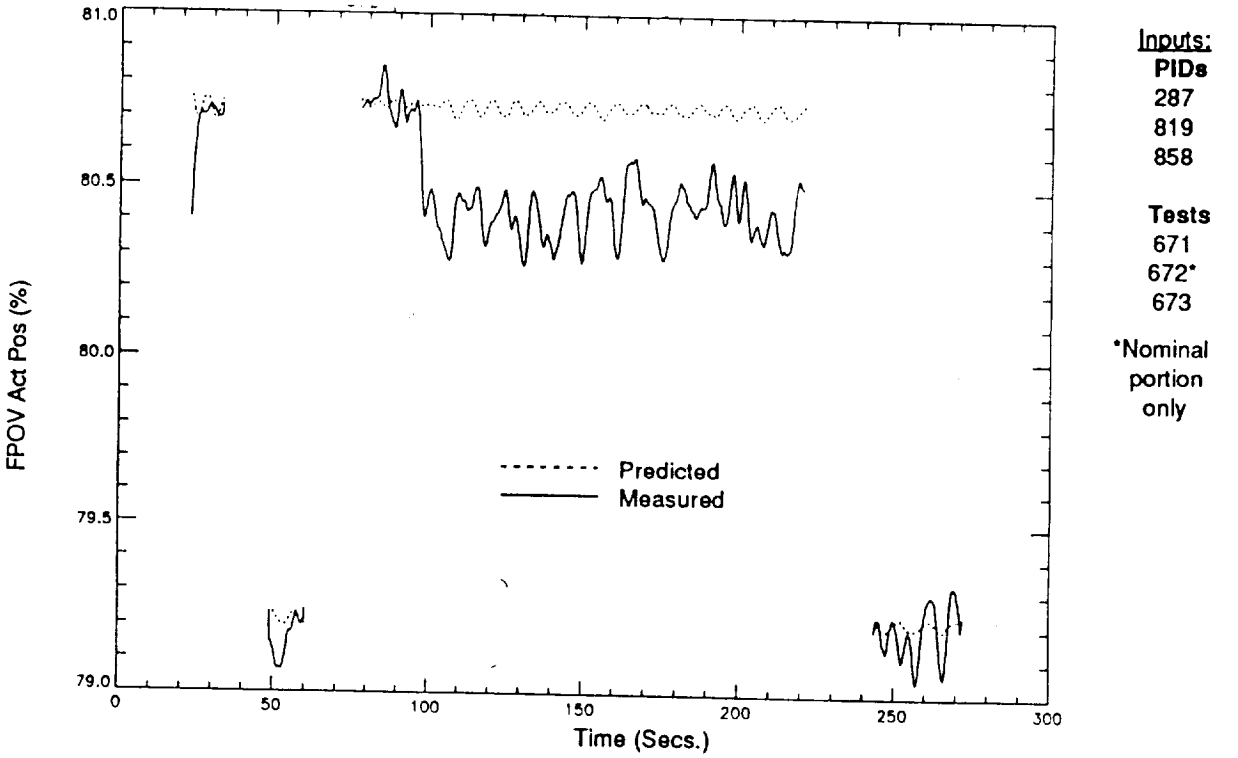


Figure 21b. Reduction in Input PIDs: Trained with Three, PID 42

3.4.3.2. Fuel Side

We added PID 835 (FUEL PRESS INT PR) to the training data set. As a result we made a better prediction for PID 42 for Test 671 when comparing Figures 22 and 23, especially in the area of the fuel repressurization event that occurred 300 seconds into the test.

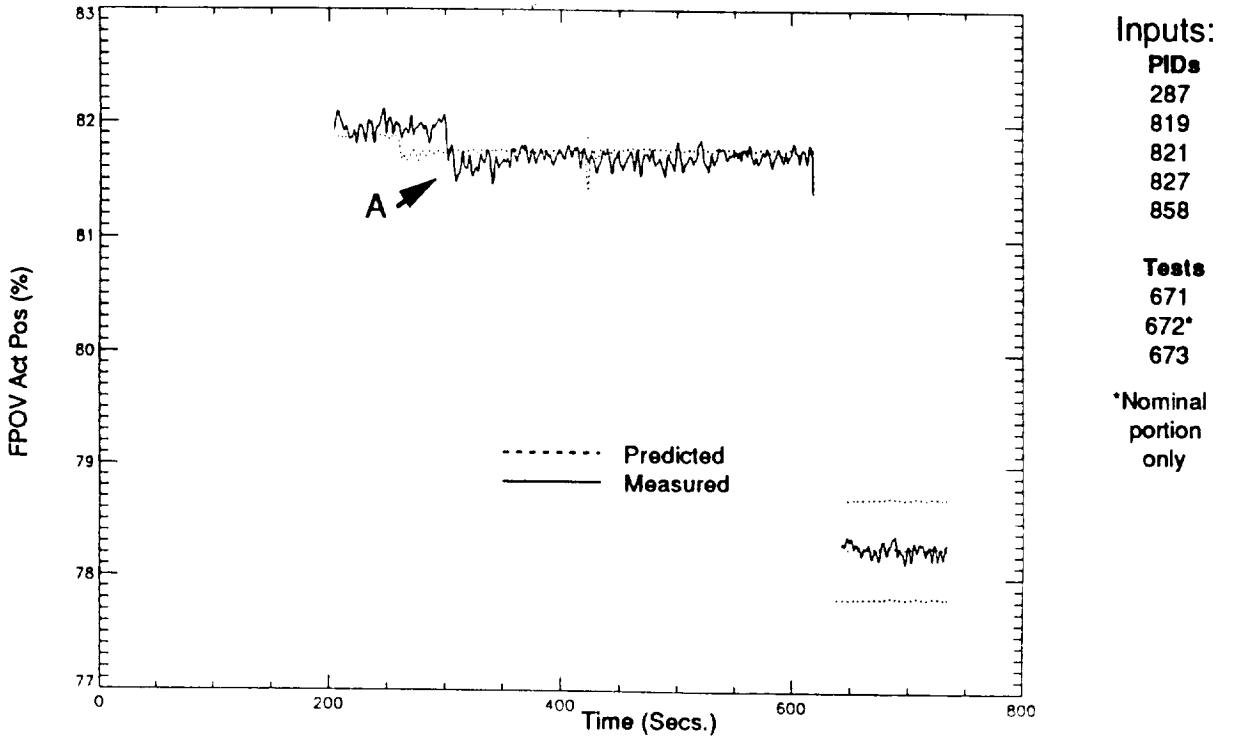


Figure 22. Screening without PID 835 in the Training Data Set, PID 42

3.4.3.3 Oxidizer Side

Figure 24 shows the results of screening after we included PIDs 878 and 879 (oxidizer heat exchanger interface temperature and pressure) in the training data set for PID 42. Even though PIDs 878 and 879 are on the oxidizer side, they improved the prediction in the area of the fuel repressurization event similar to that noted above when including PID 835 as an input.

3.4.4. Change in Transient Definition

For Tests 548, 549 and 550, we modified our definition of transients from that used for the initial training runs. Initially, any time the data contained a power-level change or change in the facility venting pressure we treated the data as transient. The facility venting schedules of these three tests forced the change in our definition of a transient. The venting schedules for these tests effectively precluded the use of any test data when using the transient definition employed during initial training.

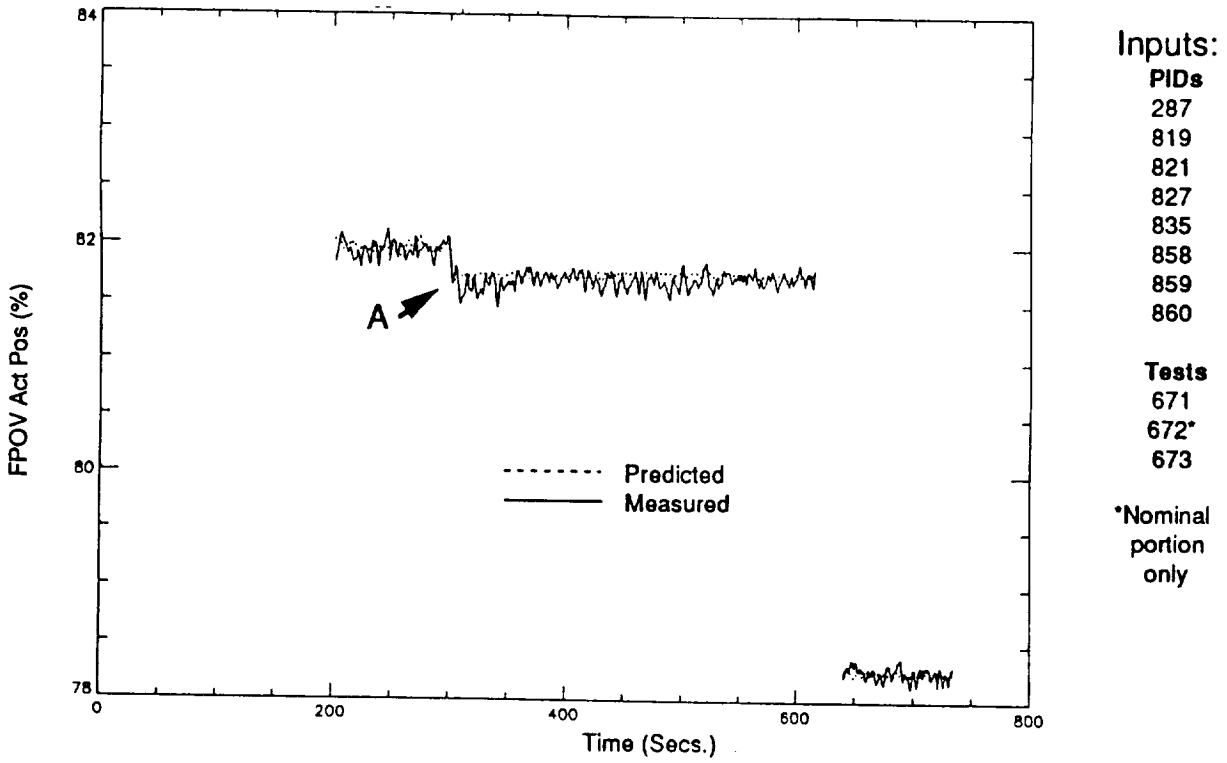


Figure 23. Screening with PID 835 in the Training Data Set, PID 42

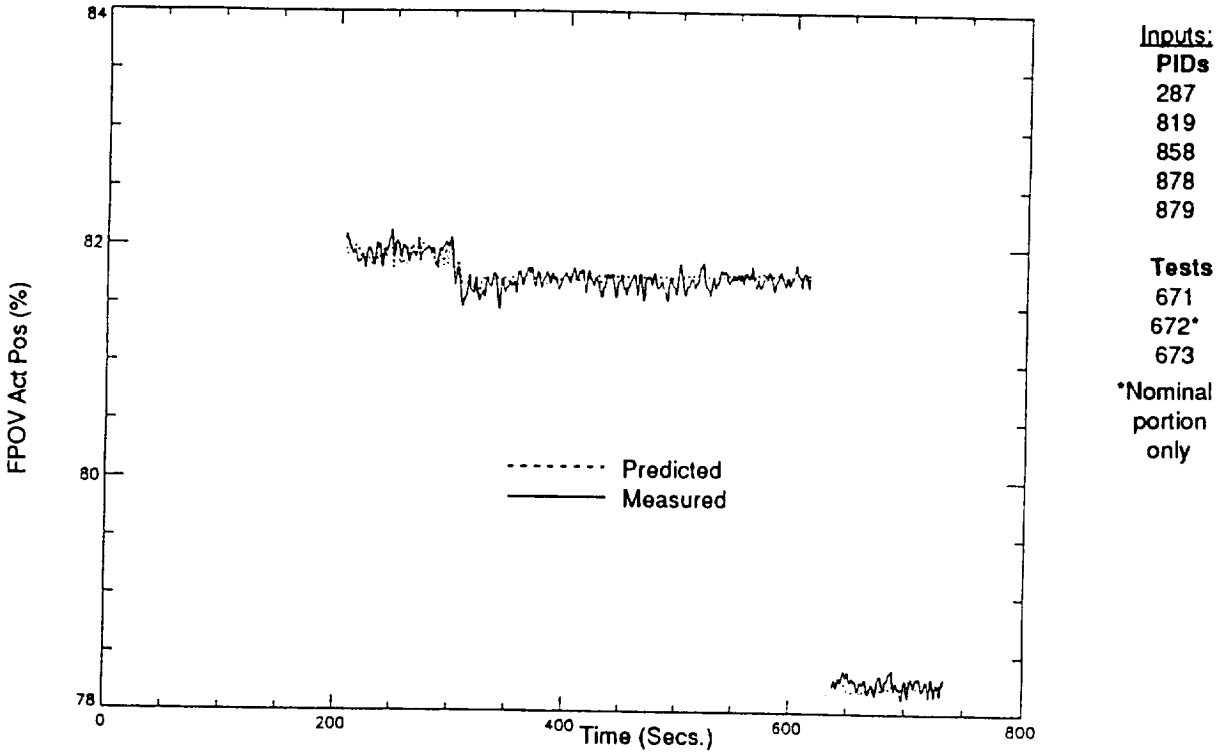


Figure 24. Screening Results after adding Input PIDs 878 and 879, PID 42

For these tests (548, 549 and 550) we defined a transient as one of the two following events:

1. Any change in engine power level including start-up and shut-down
2. Any time the second derivative of a facility venting schedule became non-zero

To test the validity of our new transient definition, we trained the neural network using Tests 548, 549 and 550. We then screened PID 42 for all these tests and Tests 671, 672 and 673. The results showed that the neural network properly predicted engine operation performance in spite of simple-ramp transients during facility venting schedules. Figure 25 shows the results of predicting PID 42 for one of these tests, 549. Figure 26 shows examples of the simple-ramp transients from Tests 548, 549 and 550 (same vent schedule for each test).

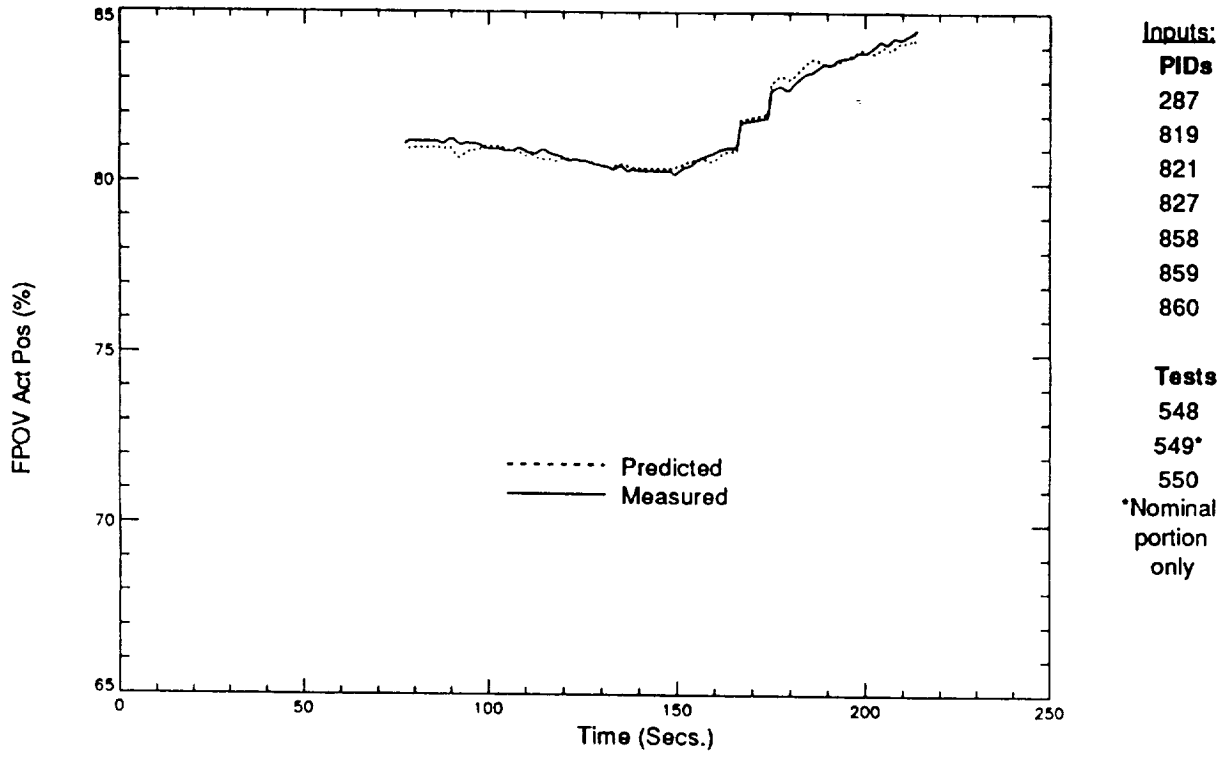


Figure 25. PID 42 Predictions for Test 548 with Venting

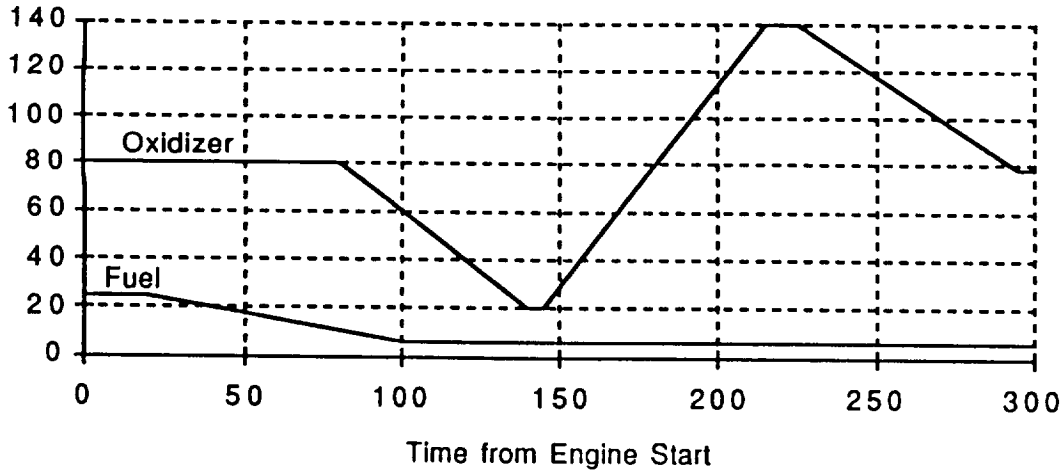


Figure 26. Fuel and Oxidizer Vent Schedules for Tests 548, 549 and 550

4. Results

4.1. Engine Sensitive vs. Baseline

We conducted training sessions for two types of neural networks, baseline and engine-sensitive, to assess the results of hardware on prediction performance. The baseline training algorithm did not attempt to discriminate among engines, but resulted in a generic "baseline" neural network for the "average" engine. It is important to note that this baseline network was *not* expected to yield accurate predictions by itself. Its purpose was to characterize overall behaviors of the SSME, and to help us understand the effects caused by hardware changes and using hardware as inputs. Since a multi-layer neural network can approximate arbitrary nonlinear mappings, it can, if necessary, learn to approximate engine biases whether their effects on the prediction function are linear or highly nonlinear.

We found that hardware biases were significant but not nearly as significant as choosing the correct input PIDs for neural network training.

The initial training discussed in Section 3.2 did use a single engine component change as input since hardware differed between Tests 671 and 672. For a typical training session the neural network received a weight of -0.15 for the hardware change input from Test 671 to 672. We found that hardware biases were significant but not nearly as significant as choosing the correct input PIDs for neural network training.

As expected for the anomalies screened in Phase I, the screening system performance improves when using engine-sensitive neural networks to distinguish an anomaly from nominal fluctuations in the data.

We included a greater amount of engine-sensitive inputs to the neural network when training with all six tests. Table 5 lists the hardware components for the six tests. NASA tested the same engine, 2206, for the entire series of tests that we utilized for Phase I. As the table below shows, all six tests utilized for training contained different hardware combinations. As expected for the anomalies screened in Phase 1, the screening system performance improves when using engine-sensitive neural networks to distinguish an anomaly from nominal fluctuations in the data.

Table 5. Hardware Tested by NASA as Used for Training Data

Test No.	HPOP	LPOP	HPFP	LPFP	MCC
901-671	0810	2118	4406 R1	2218 R2	2021
901-672	2315 R1	2106 R2	4406 R1	2218 R2	2021
901-673	2315 R1	2106 R2	4406 R2	2218 R2	2021
902-548	9409	2106 R2	4604	2218 R2	2021
902-549	2030	2106 R2	4604	2218 R2	2021
902-550	4108	2106 R2	6108	2109 R6	2021
No. of Tested Components	5	2	4	2	1

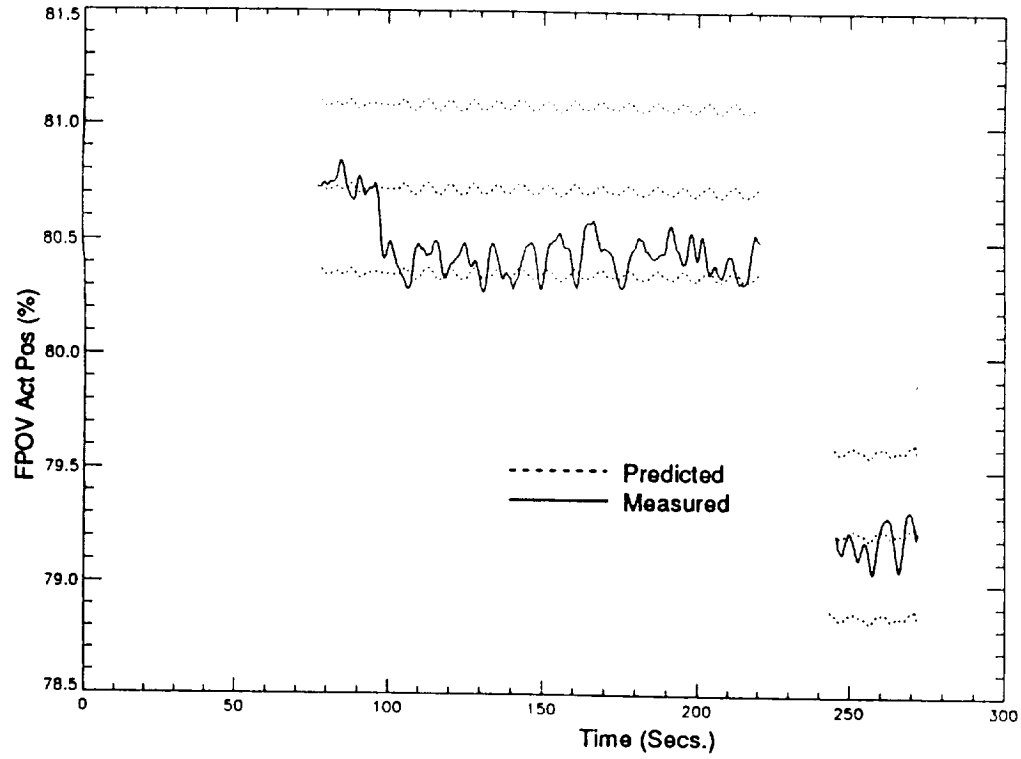
4.2. Anomalous vs. Nominal Data

We found that the threshold necessary to detect anomalous statistical deviations varies with the PID that the data screening system processes—from five sigma with PID 42 to unsuccessful screening at any level with PID 221.

Based on the Phase I results, it appears that a reasonable goal for an operational system would be to screen out at 95% of the nominal data, leaving less than 5% of the test data needing further analysis by human experts.

We varied the screening threshold (width of the nominal band as a multiple of the nominal standard deviation) to determine the amount of data that can be screened out as nominal without also screening out the anomalous data. For PID 42 we found that up to a five-sigma threshold would detect the anomaly that occurred in Test 672 (Figure 27).

Although an obvious shift in the data occurred for PID 221 in Test 549, even a one-sigma threshold in the data did not screen the anomaly correctly. Figure 28 shows the plot for PID 221 from Test 549.

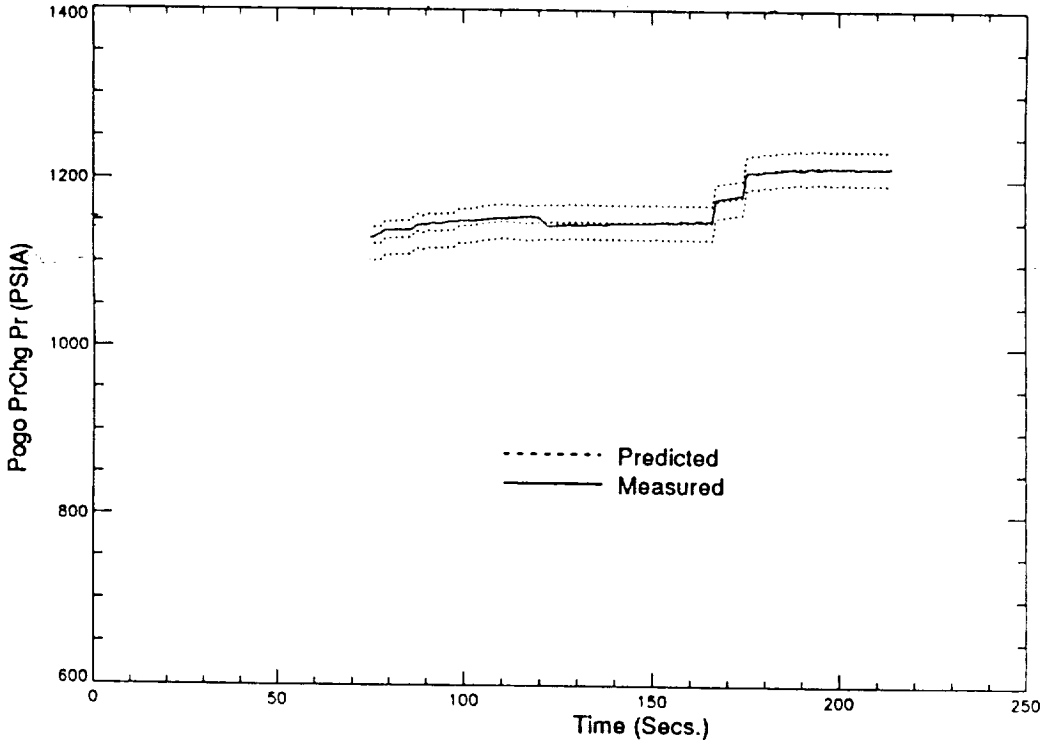


Inputs:
PIDs
287
819
821
827
858
859
860

Tests
671
672*

*Nominal portion only

Figure 27. Test 672 PID 42, Anomaly at 95 Seconds



Inputs:
PIDs
287
819
821
827
858
859
860

Tests
548
549*
550

*Nominal portion only

Figure 28. Test 549 PID 221, Anomaly at 120 Seconds

We found that the threshold for statistical deviations varies with the PID that the data screening system processes—from five sigma with PID 42 to unsuccessful screening at any level with PID 221.

Based on the Phase I results, it appears that a reasonable goal for an operational system would be to screen out at 95% of the nominal data, leaving less than 5% of the test data needing further analysis by human experts.

4.3. Predictions by Neural Networks Trained with Limited Data

Screening tests with neural networks trained from an independent set of tests proved feasible and showed that the system is capable of screening untested engine configurations.

We trained the neural network on a large sample of nominal data from Tests 548, 549 and 550 for PID 42. With this set of trained weights, we then screened Tests 671, 672 and 673. We showed that the network could predict the trends of the nominal data in the screened tests that were not used for training. The neural network predictions for the nominal Tests 671 and 673 showed a bias. Figure 29 shows the result for Test 671. The training session did not use any hardware inputs, thus the plot represents a baseline prediction. The paragraph below explains what may have caused the shift in predicted data for this training session.

The shift (bias) occurred in Tests 671, 672 and 673. The shift in Test 672 predictions based on training from Tests 548, 549 and 550 exactly cancelled the anomaly of Test 672. Either one or a combination of the following may be the reason for the result obtained (Figure 30):

- The bias due to hardware differences between Test 672 and the tests used for training exactly cancelled the anomaly of PID 42 in Test 672
- The anomaly that first occurred in the engine during Test 672 continued in Tests 548, 549 and 550 that were used for training.

The shift in Test 672 predictions based on training from Tests 548, 549 and 550 exactly cancelled the anomaly of Test 672.

Screening tests with neural networks trained from an independent set of tests proved feasible and showed that the system is capable of screening untested engine configurations.

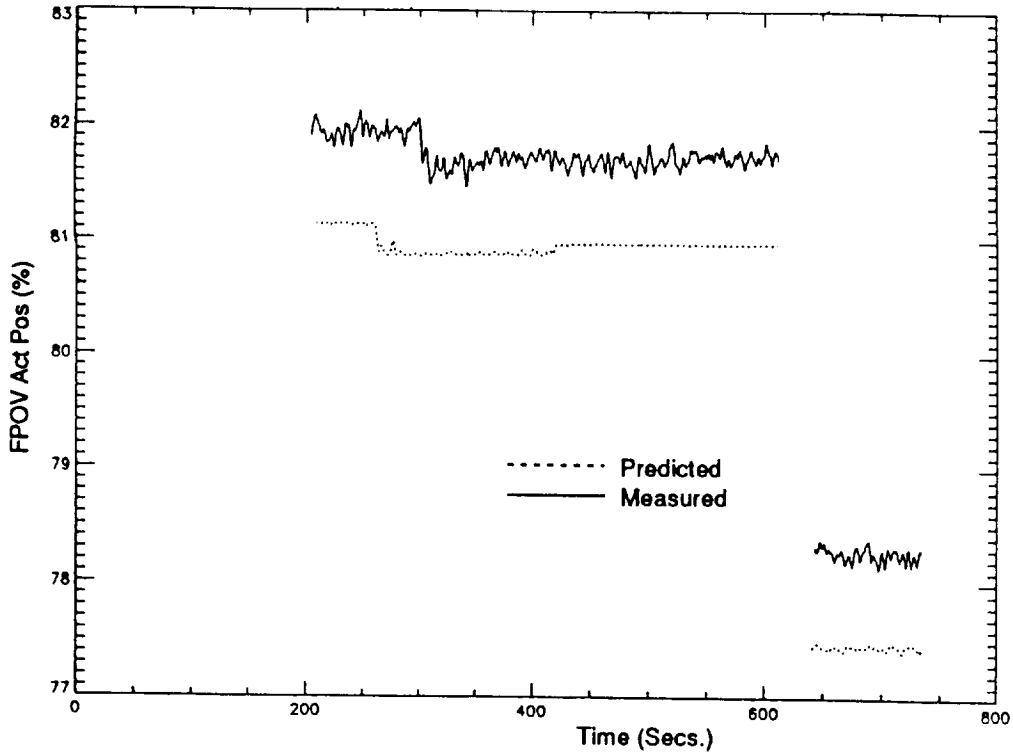


Figure 29. Prediction for Test 671, Training Based on Tests 548, 549 and 550, PID 42

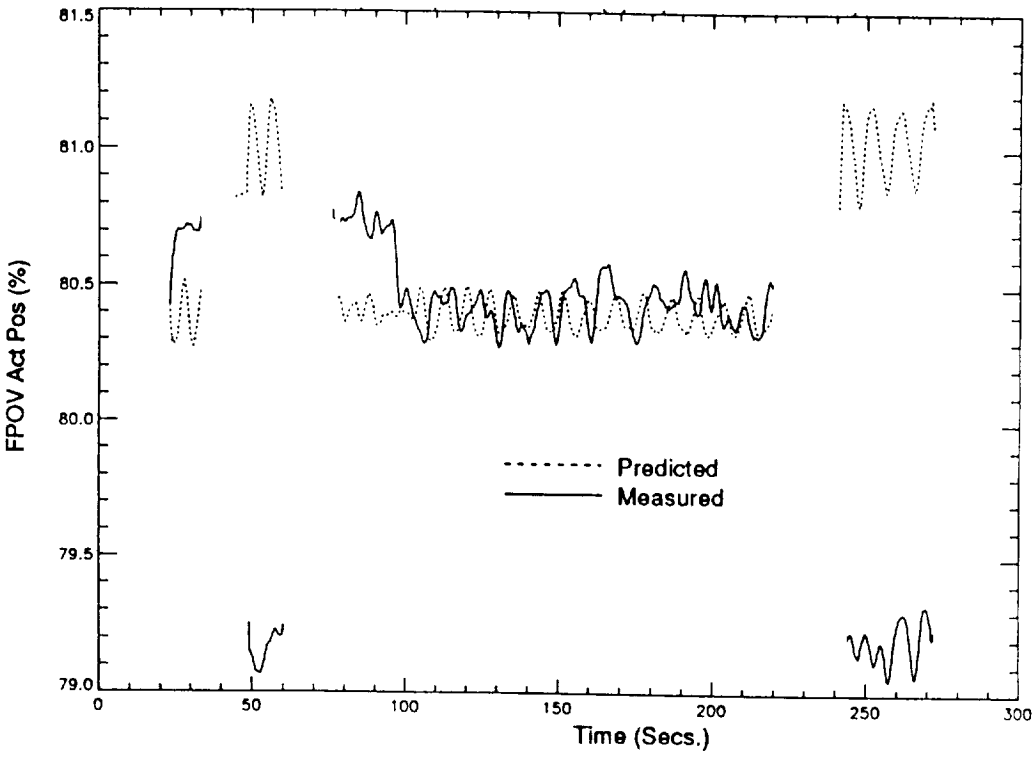


Figure 30. Prediction for Test 672, Training Based on Tests 548, 549 and 550, PID 42

5. Conclusions

5.1. Assessment of Results

The Phase I prototype showed that a properly trained neural network can screen propulsion system test data and differentiate between nominal and potentially anomalous engine conditions. As stated in Section 4.2, based on the Phase I results, it appears that a reasonable goal for an operational system would be to screen out 95% of the nominal data, leaving less than 5% of the test data needing further analysis by human experts.

Based on the results gained from Phase I we concluded the following:

- Where redundant sensors provide data, the lack of data for one sensor will not cause the screening system to fail. However, when all of the redundant sensors, for a specific engine interface, are lost then the screening system will not work properly (first order effect).
- A minimum of only three independent parameters appropriately modeled PIDs 42 and 8 when no facility venting or repressurization occurred.
- A minimum of only four independent parameters appropriately modeled the fuel side (PID 42) when facility venting occurred.
- The modeling of some PIDs showed that different predictive-PID-specific neural networks may require different inputs to account for differences between the behaviors of the oxidizer side and fuel side of the engine.
- As expected, training and testing the neural network confirmed that hardware differences affect overall engine performance and therefore the neural network's ability to correctly model engine operating conditions.
- More closely spaced Gaussian bar basis function distributions slightly improved predictions, but lengthened training times.
- The transients in facility venting schedules did not affect steady-state screening of test data when we input oxidizer and fuel inlet pressures as independent variables to the training algorithms.
- The repressurization events did affect screening performance. Including the fuel repressurization interface pressures helped screening performance for the fuel side (PID 42) in tests with venting and the repressurization event. However, the same input hurt screening performance during Test 672 that did include use facility venting or repressurization.

- The number of iterations used to train the neural network greatly increases the fidelity of the neural network model.

5.2. Recommendations for Further Work

As stated in the Introduction, the overall project objective is to develop a propulsion data screening system with the capability of detecting significant trends and anomalies in transient and steady-state data. The ultimate goal is to field a fully operational system which will actually be used on a routine basis by NASA analysts.

We feel that continued work can meet the overall project objective. As a next step to fielding a fully operational system, NASA can obtain a system for use on an experimental basis by both NASA analysts and contractor personnel to screen data during controller-induced transients as well as steady-state data. The data screening system should access historical as well as current data for anomaly detection.

In light of the ultimate goal to develop an operational data screening system, the next step should address screening for both ground test data and flight data. The system capabilities would include screening data from (1) single engine ground tests and (2) the Main Propulsion System during flight. The specific objectives of future work should encompass the following:

1. Further testing and refining of the prototype steady-state data ground test data screening system developed in Phase I.

2. Development of a neural network architecture capable of distinguishing between nominal controller-induced transients and potentially anomalous operation during controller-induced transients for ground test data in which facility measurements are available.

3. Development of training algorithms for this neural network which require training on nominal transient data only. That is, the neural network is to be trained on an appropriately weighted sample of nominal power level changes of different magnitudes and starting at different power levels. Potential anomalies are to be detected on the basis of deviation from nominal, rather than by requiring training data for each possible type of anomaly.

4. Development of a neural network architecture, based on a set of data available from the Main Propulsion System flight data, designed to screen both steady-state and controller-induced transient flight data.

5. Developing of training algorithms specifically for the neural network which screens Main Propulsion System flight data. These algorithms will require training on nominal steady-state and controller-induced transient data only.

6. Incorporation of historical data into the data screening algorithms for both ground test data and flight data.

7. Validation of the ground test screening neural networks and their training algorithms by measuring the accuracy of the function approximations performed by the neural networks. The accuracy of function approximation will be measured over a varied sample of power level changes drawn from 15 separate ground tests. The neural networks will also be validated by assessing their effectiveness in detecting anomalies in those tests which have been found by NASA engineers to contain anomalies.

8. Validation of the Main Propulsion System flight data screening neural networks and their training algorithms as in Objective 7, drawing from a set of flight data from 15 separate flights.

9. Providing step-by-step written instructions and hands-on training to NASA employees to enable them to use the system on an experimental basis.

10. Developing a specific plan for subsequent phases of the project for adding screening of startup transients, shutdown transients, flight-induced transients and facility-induced transients; for adding automatic recognition of each type of transient in the data stream; and for fielding a fully operational system for routine use by NASA analysts.

6. References

1. Broomhead, D. S. and Lowe, D. 1988. "Multivariable Functional Interpolation and Adaptive Networks." *Complex Systems 2*, Complex Systems Publications, Inc. pp. 321-355.

2. Cotter, N. E. 1990. "The Stone-Weierstrass Theorem and Its Application to Neural Networks." IEEE Transactions on Neural Networks, Vol. I, No. 4, December 1990. p. 290.

3. Hush, D. R. and Salas, J. M. 1990. "Performance of Neural Network Classifiers for the 1-Class Problem." Proceedings of the International Joint Conference on Neural Networks, Washington, D.C., January. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 396-399.

4. Moody, J. and Darken, C. 1989. "Fast Learning in Networks of Locally-Tuned Processing Units." *Neural Computation*, Massachusetts Institute of Technology. pp. 281-294.

5. Rumelhart, D. E., Hinton, G. E. and Williams, R. J. 1986. "Learning Internal Representations by Error Propagation." In Rumelhart, D.E. and McClelland, J.L. (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Cambridge, MIT Press.

6. Venkatusubramanian, V. and Chan, K. 1989. "A Neural Network Methodology for Process Fault Diagnosis" *AIChE Journal.*, Vol. 35, No. 12. p. 1993

7. Werbos, P. 1974. *Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Unpublished Ph.D. dissertation, Harvard University, Department of Applied Mathematics.

8. Whitehead, B. A., Ferber, H. J., and Ali, M. 1990. "Neural Network Approach to Space Shuttle Main Engine Health Monitoring." Paper 90-2259, AIAA/ ASME/SAE/ASEE 26th Joint Propulsion Conference, Orlando, Florida, July 1990.

9. Whitehead, B. A., Kiech, E. L., and Ali, M. 1990. "Rocket Engine Diagnostics Using Neural Networks." Paper 90-1892, AIAA/ASME/SAE/ASEE 26th Joint Propulsion Conference, Orlando, Florida, July 1990.

10. Hartman, E. and Keeler, J. D. 1991 "Predicting the Future: Advantage of Semilocal Units." *Neural Computation*, Vol. 3, pp. 566-578.

National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle

Automated Screening of Propulsion System Test Data by Neural Networks

Final R&D Status Report

5. Report Date

4/17/92

6. Performing Organization Code

7. Author(s)

W. Andes Hoyt
Dr. Bruce Whitehead

8. Performing Organization Report No.

ERC-R-92-022

10. Work Unit No.

9. Performing Organization Name and Address

ERC, Inc.
P.O. Box 417
Tullahoma, TN 37388

11. Contract or Grant No.

NAS8-39184
SBA 4-91-2-0357

12. Sponsoring Agency Name and Address

NASA
Marshall Space Flight Center
Huntsville, AL 35815

13. Type of Report and Period Covered

Final
10/4/91 – 4/3/92

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The evaluation of propulsion system test and flight performance data involves reviewing an extremely large volume of sensor data generated by each test. An automated system that screens large volumes of data and identifies propulsion system parameters which appear unusual or anomalous will increase the productivity of data analysts. Data analysts may then focus on a smaller subset of potentially anomalous data for further evaluation of propulsion system tests. Such an automated data screening system would give NASA the benefit of a reduction in the manpower and time required to complete a propulsion system data evaluation. This report details a six-month Phase I effort to develop a prototype data screening system. The report discusses the software development, data selection, system refinement and testing, and results. Conclusions drawn from the results and recommendations for further work are included. In summary, the prototype data screening system shows that neural networks can be successfully applied to screening data from the operation and testing of a rocket propulsion system.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Data Screening Propulsion Data Neural Networks	Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	43	

NASA FORM 1626 OCT 86