

A Function Approximation Approach to Anomaly Detection in Propulsion System Test Data

Bruce A. Whitehead, W. Andes Hoyt

Abstract—Ground test data from propulsion systems such as the Space Shuttle Main Engine (SSME) can be automatically screened for anomalies by a neural network. The neural network screens data after being trained with nominal data only. Given the values of 14 measurements reflecting external influences on the SSME at a given time, the neural network predicts the expected nominal value of a desired engine parameter at that time. We compared the ability of three different function-approximation techniques to perform this nominal value prediction: a novel neural network architecture based on Gaussian bar basis functions, a conventional backpropagation neural network, and linear regression. These three techniques were tested with real data from six SSME ground tests containing two anomalies. The basis function network trained more rapidly than backpropagation. It yielded nominal predictions with a tight enough confidence interval to distinguish anomalous deviations from the nominal fluctuations in an engine parameter. Since the function-approximation approach requires nominal training data only, it is capable of detecting unknown classes of anomalies for which training data is not available.

I. INTRODUCTION

A fully-instrumented ground test of a propulsion system typically generates a very large quantity of data. In the case of the Space Shuttle Main Engine (SSME), data analysis is currently a labor-intensive process. Human experts spend a great deal of time examining the large volume of sensor data generated by each test. The experts look for any anomalies in the data which might indicate engine conditions warranting further investigation. In this paper, we do not propose that the time-consuming manual process can be entirely replaced by an automated system. Rather, we propose that human expertise could be used more efficiently if an automated system performed a “first-cut” screening of test data. The first cut would identify the relatively small volume of data which is unusual or anomalous in some way. Limited and expensive human resources could then focus on this small volume of unusual data requiring expert human analysis.

II. HYPOTHESIS

A liquid-fueled rocket engine, such as the SSME, can be viewed as a system with a set of physical and informational interfaces to other systems. A relatively clean interface is defined if the boundary is drawn around the

Presented to the 29th AIAA/SAE/ASME/ASEE Joint Propulsion Conference, Monterey, CA, June 1993.

Bruce A. Whitehead is with the University of Tennessee Space Institute, Tullahoma, TN 37388, USA. E-mail: bwhitehe@utsi.edu

W. Andes Hoyt is with ERC, Inc., P.O. Box 417, Tullahoma, TN 37388, USA. E-mail: ahoyt@edge.ercnet.com

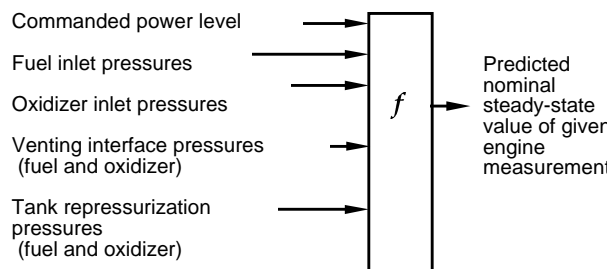


Fig. 1. Hypothesis that the nominal steady-state time-series values of a given internal engine parameter can be predicted from the external influences shown on the left of the figure.

engine controller and the system it controls (valves, turbopumps, etc.). The external influences which affect the state of the system include the informational interface of commands given to the controller, as well as various physical interfaces such as the fuel inlet, the oxidizer inlet, and the venting and repressurization interfaces to both the fuel and oxidizer tanks.

Under nominal steady-state operating conditions the behavior of the engine is, at least in principle, determined by what transpires at these interfaces. Given measurements of what is happening at the system interfaces, it would therefore be possible to predict the steady-state values of all parameters measured within the engine. In other words, to the extent that the SSME is a deterministic system, there would exist some function f (Figure 1) which predicts the nominal steady-state value of any desired engine parameter. f would predict this value from measurements at all interfaces crossing the system boundary. The function f might be approximated as either a “white-box” model based on the underlying physics of the SSME, or as a “black-box” model which attempts to approximate the function using empirically-derived relationships between inputs and outputs.

The utility of approximating this function f depends on three factors: (1) whether the behavior of the SSME is indeed sufficiently determined by these external influences, (2) whether adequate measurements of these influences are available, and (3) whether the function f could be approximated accurately enough. If these three conditions are met, then the approximation of f could serve as a tool for detecting anomalies in the behavior of the SSME. The argument for our approach is straightforward. If an anomaly within the SSME is detectable, it is presumably detectable because a measured parameter within the engine differs from the nominal value of that parameter *for the given*

operating conditions.

Anomaly detection would then be based on the idea of making and continuously updating predictions of the expected *nominal* values of all internal engine parameters which might conceivably indicate an anomaly. Each prediction would be the nominal value of that parameter based on the available measurements of interface conditions affecting the engine. Each predicted nominal value for an engine parameter would then be compared with the actual measured value of the parameter. If the measured value differs significantly from the predicted value, then a potential anomaly would be indicated. This process would be repeated for each data point in the time series of measurements to be screened. An automated system could conceivably screen large quantities of a test data in this manner. The system could flag each potential anomaly for further study by a human analyst. The analyst would then determine whether it is (1) an anomaly in the engine itself, (2) an anomaly in the sensor or data channel, or (3) an anomaly in our white-box or black-box model of f .

Unfortunately, a prediction which is possible in principle might not be accurate in practice. This might happen because of several sources of error:

- Our knowledge of the interfaces affecting the SSME may be incomplete.
- The available measurements of conditions at these interfaces may not be complete enough or accurate enough.
- The state of the SSME may be subject to internal fluctuations which cannot be predicted from the external interfaces.
- Our approximation of the function f may not be good enough.
- The function f itself may vary from one engine to another, or among different configurations of the same engine.

Due to any combination of these factors, actual parameter measurements might fluctuate a certain amount from the predicted nominal values even under nominal operating conditions. If the fluctuation can be treated statistically, all is not necessarily lost; in this case the predictions made by our model of f would need to be surrounded by some nominal confidence interval. The utility of predictions made by our model would then depend upon the extent to which fluctuations caused by genuine anomalies are statistically significant in comparison to nominal fluctuations in the measurement. (To the extent that the assumptions of signal detection theory are met, this would lead to a well-understood trade-off between “misses,” in which genuine anomalies are missed because their fluctuations fall within the confidence interval, and “false alarms,” in which nominal fluctuations fall outside the confidence interval and are erroneously flagged as anomalous.) For the purposes of data screening, any measurement deviating from the nominal prediction by more than a set confidence interval would be flagged for further study by a human analyst. The analyst would still have the task of distinguishing nominal from anomalous fluctuations among this reduced set of data

falling outside the nominal confidence interval.

The hypothesis of our study is that, despite the many sources of error enumerated above, the function f can be sufficiently approximated to yield predictions which have practical utility for automated data screening. Our criterion for practical utility is the ability to detect fairly subtle anomalies in SSME data without incurring an inordinate number of false alarms. The type of data screening system envisioned in Section 1 will not be useful unless it significantly reduces the volume of test data requiring human analysis. If under nominal engine conditions, the deviations of actual measurements from their predicted values could be modeled as Gaussian noise then a nominal confidence interval of three standard deviations above and below the predicted nominal value would perhaps yield an acceptable false alarm rate (on the order of 0.003). In fact, the nominal fluctuations observed in the present study do not fit the Gaussian assumption well enough to yield this low a false alarm rate with confidence intervals of three standard deviations. To compensate for this departure from Gaussian noise, we arbitrarily set a nominal confidence interval of five standard deviations around the predicted values. That is, only a measurement deviating more than five standard deviations from its predicted nominal value would be flagged as anomalous. This criterion was applied uniformly to all function approximation techniques to evaluate the validity of each method in detecting real anomalies.

Since neural networks are tools for approximating arbitrary nonlinear functions [1], [2], [3], [4], [5], [6], we compare the ability of candidate neural network architectures to approximate the function f described herein. A neural network does not assume any particular model or functional form for the system it is modeling. Instead, it converges to an approximation of a function based only on the actual data points observed for that function. A neural network is also particularly suitable for fine-tuning the approximation of f to each particular engine configuration. Since a neural network constructs its approximation by being trained on data points, it is conceivable that a neural network can be fine-tuned to adjust its predictions for each of several engine configurations, provided that test data is available for each configuration.

III. BACKGROUND

Many investigators [7], [8], [9], [10], [11], [12], [13], [14] have demonstrated that neural networks can learn to discriminate between nominal sensor data and various known classes of faults. Since neural networks learn from examples, they avoid the costs of explicitly building and maintaining large, complex knowledge bases. Furthermore, such an example-based training method is likely to be easier to adapt to new SSME designs (or to engines other than the SSME) than a knowledge base which has been built for a specific engine.

Since neural networks are trained by example, however, 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, conventional neural network training procedures require representative data on both sides of the discrimination, (i.e., representative nominal data and representative anomalous data).

In SSME testing (and in propulsion testing in general) it is possible to collect a representative sample of nominal data by systematically varying test conditions over the range of conditions for which engine behavior is of interest. Collecting a representative sample of anomalous data is, however, problematic. In testing any complex system, data is collected for only a very small fraction of all possible anomalies. Other possible anomalies might be simulated, but it is still not possible to anticipate and simulate every possible type of anomaly which might occur. Unfortunately, if a neural network has been trained on an incomplete or unrepresentative set of anomalies, there is no guarantee that it will reliably identify other types of anomalies which might occur.

The underlying problem is therefore to train a neural network to reliably discriminate between two categories (i.e., nominal versus anomalous) when representative training data is available for only one of these categories (i.e., nominal) [15].

Our approach to identifying anomalous behavior, as indicated in the previous section, is to train the neural network not to classify anomalies, but to predict nominal values of engine parameters. Representative anomalous data is not needed because the neural network is not trying to predict anything about anomalies. Rather, the neural network is trying to predict, as accurately as possible, the nominal steady-state value of each engine parameter *under the given interface conditions*. Anomalies are identified by comparing predicted nominal data to actual data.

In the terminology of the previous section, the function f is modeled under nominal conditions. Only nominal training data is needed to achieve this purpose. The neural network is never trained on anomalous data, and does not itself construct a decision surface between nominal and anomalous data.

Such a decision surface can be constructed outside the neural network by computing the confidence interval described in Section 2 above. The advantage of constructing a decision surface in this way is that it does not depend on our ability to characterize or gather data for all possible anomalous conditions. A potential anomaly is simply recognized as a greater-than-chance deviation from nominal. In practice, of course, this idea will work only if the neural network's predictions are sufficiently and consistently accurate enough to yield tight confidence intervals. Finding out whether this is the case is the purpose of the present investigation of neural network architectures for the propulsion system data screening application.

Neural network architectures for multivariate function approximation have been evaluated in the neural network literature [1], [2], [3], [4], [5], [6], [16], [17] and shown to be comparable to classical techniques in the quality of the approximation expected. In screening large volumes of propulsion sensor data, however, the type of example-based

training used in a neural network is likely to be more practical to implement than a classical function-approximation technique. Example-based training also has the advantage of avoiding any particular assumptions about the form of the function to be approximated.

IV. APPROXIMATION TECHNIQUES

Three function-approximation techniques were investigated in this study: (i) a novel neural network architecture based on Gaussian bar basis functions [18], [19]; (ii) a standard backpropagation neural network [20]; and (iii) linear regression. Backpropagation is the neural network technique most commonly used in real applications, and thus serves as a benchmark for evaluating other neural network architectures. The Gaussian bar basis function architecture was investigated because of two difficulties in the application of backpropagation to real problems: the slowness of the gradient descent optimization when multiple layers are involved, and the possibility of getting stuck on a local minimum. (Both of these difficulties were actually encountered in this application, as discussed in Sections 5.8 and 6.1 below.) Finally, a straightforward linear regression technique was included to assess whether, and to what extent, a nonlinear function-approximation procedure was necessary for our application.

Each of the three function-approximation techniques can be used to model a real-valued function of N real variables, which we term *input variables*. The cross product of these N input variables is termed the *input space*. This input space is the domain of the function f to be approximated. The number of input variables is the *dimension* of the input space. The value of the function f to be approximated is termed a *predicted variable*. (The term "output variable" is avoided, since in our application the variables to be predicted are typically measurements of the internal state of the SSME, not outputs per se.) If more than one variable is to be predicted, each will be considered as the range of a separate real-valued function f_i .

The three function-approximation techniques all assume the availability of a set of sample points (points in the input space) at which the value of the function f was observed. This set of sample points with corresponding observed function values is termed the *training set*, and fitting the function approximation to the training set is called *training*. In our anomaly detection application, the training set consists of nominal data only, as explained in Sections 2 and 3 above. The input variables (Figure 1) consist primarily of measurements of conditions at the interfaces we defined for the SSME. These interfaces are the lines supplying fuel and oxidizer to the engine, venting and repressurization interfaces which affect pressure in the fuel and oxidizer tanks, and the commanded main combustion chamber pressure which the engine controller works to maintain.

In our application, the predicted variable is a measurement of some engine parameter which is known to be nominal in the training data, but which might be either nominal or anomalous in new data to be screened. Even under

nominal conditions, measurements of the predicted variable are assumed to be subject to the sources of error and variability described in Section 2 above. All three function-approximation techniques studied are based on the idea of minimizing the root-mean-squared (RMS) error of the function approximation over the sample points in the nominal training set.

A. Gaussian Bar Basis Function Network

Basis function networks [2] are a family of neural network architectures useful for multivariate function approximation. Such networks attempt to approximate an arbitrary nonlinear function as a linear combination of a set of basis functions. Since the basis functions themselves are nonlinear, this family of architectures is capable of approximating nonlinear functions. Neural network architectures in this family normally have three layers. Each node in the input layer receives input from one variable x_k in the domain of the function to be approximated. The number of input nodes is thus the dimension of the input space. Each node in the middle layer computes the value of one of the chosen basis functions g_j . The number of middle nodes is thus the number of basis functions used for the function approximation. Finally, if more than one function is to be approximated from the same set of basis functions, then there can be a separate output node for each real-valued function f_i to be approximated.

The functional form of the basis functions, and the selection of the specific set used in a given network vary over this family of architectures as described below. Once the basis set is determined, however, the value computed by each output node can be expressed as a linear combination of the basis functions (i.e., a linear combination of the values computed by the middle nodes). Computation of the least-squares optimal choice of coefficients to form such a linear combination is straightforward—either algebraically or iteratively. If an iterative gradient descent method is used, the error surface whose gradient is being descended takes the form of a paraboloid with only one local minimum which is also a global minimum. Training is therefore likely to be orders of magnitude more rapid than backpropagation [17], and there is no danger of getting stuck on a non-optimal local minimum. Eliminating local minima is perhaps the chief appeal of basis function networks for practical applications.

Various architectures in this family of basis function networks are differentiated primarily by the set of basis functions to be computed by the middle layer of the network. If the dimension of the input space is small, radial Gaussian basis functions are attractive because of mathematical proofs of their ability to approximate a large class of nonlinear functions [1], [2], [3]. Each radial basis function g_j in the set takes the Gaussian form

$$g_j(\mathbf{x}) = \exp \left[- \sum_{k=1}^N \frac{(x_k - \mu_{jk})^2}{2\sigma_j^2} \right] \quad (1)$$

where the point $(\mu_{j1}, \dots, \mu_{jN})$ is the center of the basis

function g_j , and σ_j is its width. The symbol \mathbf{x} simply denotes the vector of inputs (x_1, x_2, \dots, x_N) .

Radial basis functions form a local function approximation in the sense that they are nonzero only in a local portion of the input space centered around one point. There must then be enough of these local basis functions to fill the volume of the input space to be approximated. Excellent results have been obtained with radial basis functions for input spaces of a few dimensions, but as the number of dimensions increases, the number of basis functions needed grows exponentially. For this reason, a radial basis function architecture was not considered for the present study in which the dimension of the input space is 14.

To adapt the basis-function technique to input spaces of high dimensionality, a set of semi-local Gaussian bar basis functions has been proposed [18], [19]. Each such basis function takes the one-dimensional Gaussian form

$$g_k(x_k) = \exp \left[- \frac{(x_k - \mu_k)^2}{2\sigma_k^2} \right] \quad (2)$$

where μ_k is the center of a one-dimensional Gaussian function. As originally proposed, [18], [19] there is only one Gaussian bar for each dimension of the input space. The center and width of each Gaussian bar is then adjusted by the same type of gradient descent procedure as used in backpropagation. Since there is only one Gaussian bar along each dimension of the input data, there is no particular reason to expect this architecture to be able to approximate an arbitrary nonlinear function. In fact, pilot studies with this architecture have been unsuccessful in approximating the training data of this study to the degree of accuracy required for differentiating nominal from anomalous data.

In light of these considerations, it appears to us that radial Gaussian basis functions form too rich of a basis set, theoretically capable of universal approximation, but requiring too many basis functions when a high-dimensional volume needs to be filled. Semi-local Gaussian bar functions can overcome this “curse of dimensionality,” but having only one basis function for each dimension appears to form too poor a basis set to give good performance in our application.

We therefore devised a set of basis functions falling between the two extremes of the radial basis and Gaussian bar functions. This new set of basis functions consists of M , rather than 1, Gaussian bars along each dimension of the input data. That is, the set of possible values of each input variable x_k is covered by a set of M basis functions

$$g_{jk}(x_k) = \exp \left[- \frac{(x_k - \mu_{jk})^2}{2\sigma_k^2} \right] \quad (3)$$

as shown in Figure 2. For computational efficiency, each basis function is truncated to zero for values of x_k outside the interval $[\mu_{jk} - 4\sigma_k, \mu_{jk} + 4\sigma_k]$. Altogether, there are accordingly MN basis functions g_{jk} , $j = 1, \dots, M$; and $k = 1, \dots, N$. The number of basis functions required therefore grows linearly, not exponentially, with the num-

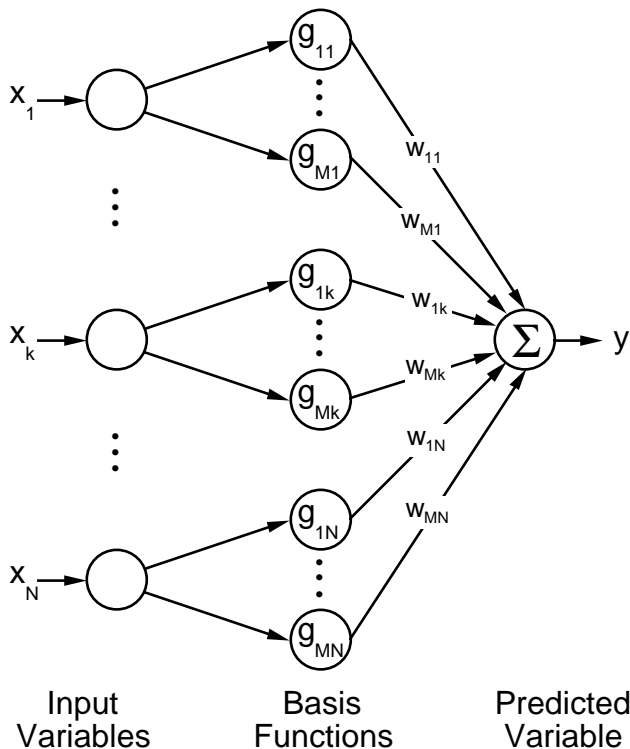


Fig. 2. Gaussian bar basis function neural network for approximating the function f of Figure 1. The predicted variable is listed as the measurement screened in Table 1. The input variables are listed in Table 2. The function f from the inputs to the predicted nominal value is approximated as a linear combination of the Gaussian bar basis functions g_{jk} .

ber of dimensions of the input space. Our new set of basis functions, like the original Gaussian bar architecture, is clearly not capable of approximating every possible nonlinear function. It provides for arbitrary nonlinear recoding of each input variable, but not for arbitrary nonlinear interactions among the inputs. We hypothesized that this limited ability to approximate nonlinearities might yield a sufficient approximation for our application, and if so, would yield a more robust and more rapid training procedure than backpropagation.

The Gaussian bar basis function network can be used to approximate a real-valued function $f(x)$ as a linear combination of basis functions. Thus, $f(x)$ is to be approximated by

$$y = w_{oo} + \sum_{j,k} w_{jk} g_{jk}(x_k) \quad (4)$$

where y is the output of the network, as illustrated in Figure 2, and the coefficient w_{jk} represents the “weight” of the connection from the basis function g_{jk} to the output node y . A constant bias term in the linear combination of the basis functions is w_{oo} . The network is trained by adjusting the weights to minimize the mean squared prediction error over the training set. The weights are adjusted according

to the widely-used gradient descent learning rule [21]

$$\Delta w_{jk} = \alpha \cdot y^{error} \cdot g_{jk}(x_k) \quad (5)$$

in which y^{error} is the difference between the approximation y produced by the network for a given training example \mathbf{x} , and the actual measured value of $f(\mathbf{x})$ for that training example. A constant α between 0 and 1, which controls the rate of gradient descent, is commonly called the “learning rate” in neural network literature. Training proceeds by making repeated passes through the training set, applying the learning rule (5) above for each successive training example in the training set. Section 5 below details the specific parameter settings and procedures used for our SSME application.

B. Backpropagation Neural Network

Backpropagation networks are the class of neural networks most commonly applied to real-world problems, and hence serve as a benchmark for comparison to a new architecture such as that described above. For our purpose, the most commonly used standard backpropagation algorithm with momentum was used. Since this algorithm and its justification have been extensively described in the neural network literature [20], such a description will not be repeated here.

A three-layer architecture was employed for our backpropagation neural network. The input layer contains one node for each input variable to be processed, as in the Gaussian bar architecture. Each node in the middle layer computes a linear combination of these inputs, and then passes the result through a sigmoid “squashing function.” Each node in the output layer, in turn, computes a linear combination of the middle-layer outputs and also passes the result through a sigmoid “squashing function.” The gradient-descent training procedure adjusts the coefficients (“weights”) of all these linear combinations to minimize the mean squared error in the function approximation computed by each output node. The form of gradient descent most commonly used in neural networks, and used in this study, employs a “momentum” term [20] which forms a linear combination of the currently-calculated steepest-descent gradient vector and the gradient vector used in the previous time step.

C. Linear Regression

To test whether a nonlinear function approximation technique is even necessary for this application, multivariate linear regression was employed as our third function-approximation technique. This linear regression was conducted using the method of singular value decomposition which yields appropriate results even if the matrix of sample data is singular or nearly so. (In fact, neither singularities nor near-singularities arose in the data sets we used.)

V. PROCEDURE

Unless noted otherwise, the steps below were followed identically for each of the three function approximation techniques described in Sections 4.1, 4.2, and 4.3.

SSME Test	Engine 2206 Component Nos.				Measurement Screened	NASA Analysis
	HPFP*	LPFP**	HPOP†	LPOP††		
901-671	#3	#1	#3	#2	Fuel Preburner	Nominal
901-672	#3	#1	#4	#2	Oxidizer Valve	Anomaly @ 96 s
901-637	#4	#2	#5	#2	Position, PID 42	Nominal
902-548	#1	#1	#1	#1	POGO Surge Suppression	Nominal
902-549	#1	#1	#2	#2	System, Pressure Change,	Anomaly @ 120 s
902-550	#2	#1	#2	#2	PID 221	Nominal

*High Pressure Fuel Turbo Pump

†High Pressure Turbo Pump

**Low Pressure Fuel Turbo Pump

††Low Pressure Oxidizer Turbo Pump

Table 1

A. Selection of SSME Tests and Predicted Variables

As explained in Section 3 above, our function approximation approach requires nominal training data which adequately represents the engine conditions for which data screening is desired. After training is complete, both nominal and anomalous data are required to test the performance of each technique. We therefore required one or more series of SSME ground tests meeting the following criteria:

1. The engine configurations within each series of ground tests were similar but not identical.
2. Each series contained known nominal data over a range of SSME power levels.
3. Each series also contained one known anomaly which occurred during steady-state operation.
4. Each series contained an adequate sample of nominal data at the same power level as the data containing the anomaly.
5. The indications of the anomaly in the data stream were subtle enough to be detectable by a trained NASA engineer, but not necessarily obvious to an untrained observer.

Any anomaly in which a given engine measurement goes outside its normal operating range would be easily detectable by many different techniques, and was not considered difficult enough to serve as an adequate test of our technique. We therefore considered only anomalies in which all engine parameters remained within their normal operating ranges, and which appeared anomalous to trained engineers only on the basis of a much more precise understanding of what is deemed nominal for a given set of operating conditions.

Based on our criteria, NASA engineers selected two series of three ground tests each. These tests, and the major engine components of each test, are shown in Table 1. The variables predicted were simply those engine measurements to be screened and classified by the system as nominal or anomalous. Since we wished to compare the ability of each function approximation technique to distinguish anomalous from nominal data, we chose the engine measurements which had been so-classified by NASA engineers. Table 1

shows the engine measurement whose nominal value is to be predicted for each test, as described in Section 4 above. Table 1 also indicates the onset time of each anomaly in these measurements, as identified by NASA engineers. To the best of our knowledge, the causes of these anomalies have not been explained.

The ability of each function approximation technique to detect anomalies without incurring false alarms is what we are testing. Thus, we performed two approximation tasks: (a) to predict nominal values of PID (Parameter Identification) 42 for Tests 901-671, 901-672, and 901-673; and (b) to predict nominal values of PID 221 for Tests 902-548, 902-549, and 902-550. Each function approximation technique would identify an anomaly whenever the measured value of the given PID differs significantly from the predicted nominal value.

B. Selection of Input Variables

Our choice of input variables was governed by the boundary defined around the SSME. As explained in Section 2 above, the external interfaces which cross this boundary include the informational interface of commands given by the controller, and the physical interfaces at the fuel inlet; the oxidizer inlet, and the venting and repressurization interfaces to both the fuel and oxidizer tanks. These types of inputs are depicted in Figure 1. Input variables were selected to capture information about physical conditions at these interfaces which might affect nominal operating conditions within the engine. Based on the advice of NASA engineers who analyze SSME test data on a routine basis, the input variables shown in Table 2 were selected as the most promising for our purpose. The same set of input variables were used in predicting PID 42 and PID 221.

The first of these variables, Commanded Main Combustion Chamber Pressure, is the desired chamber pressure sought by the engine controller. The remaining variables are pressure measurements at various physical interfaces to the fuel and oxidizer tanks as indicated in Table 2. We extracted data for these engine measurements from existing NASA databases in which measurements are sampled at 25 Hz in the case of the commanded chamber pressure, and

SSME PID*	Measurement Description	Units
287	Commanded Main Combustion Chamber Pressure	PSI
819	Engine Fuel Inlet Pressure #2	PSI
821	Engine Fuel Inlet Pressure #1	PSI
827	Engine Fuel Inlet Pressure #3	PSI
830	Fuel Bleed Interface Pressure	PSI
835	Fuel Pressurization Interface Pressure	PSI
836	Fuel Pressurization Venturi Inlet Pressure	PSI
837	Fuel Pressurization Venturi Delta Pressure	PSI
858	Engine Oxidizer Inlet Pressure #2	PSI
859	Engine Oxidizer Inlet Pressure #1	PSI
860	Engine Oxidizer Inlet Pressure #3	PSI
878	Heat Exchanger Interface Pressure	PSI
881	Heat Exchanger Venturi Inlet Pressure	PSI
883	Heat Exchanger Venturi Delta Pressure	PSI

*Parameter Identification Number

Table 2

at 50 Hz for all other parameters shown in Table 2.

The function f to be approximated might conceivably vary among different configurations of the same engine. To allow for this, the input variables also included four discrete variables to represent the different combinations of components shown in Table 1. These configuration variables were provided as inputs to allow each function approximation technique to bias its prediction on the basis of the specific components installed during each SSME test.

C. Time-averaging of Engine Measurements

All engine measurements were time-averaged over a sliding window of 0.2 seconds. We averaged data to improve the signal-to-noise ratio of the data. Admittedly, this time-averaging would reduce the ability of the system to detect subtle anomalies of very short duration.

D. Extraction of Steady-State Engine Data

Our current study focuses only on steady-state data. Thus we had to remove transient data from the data stream. Transients due to SSME power-level changes, fuel and oxidizer venting, and fuel and oxidizer repressurization were present in the raw data. Thrust-level profiles provided by NASA were used as a guide for locating transients, but the actual onset time and duration of the transients were determined from available engine measurements indicative of each type of transient. We preferred to overestimate rather than underestimate the duration of each transient, since we did not want to risk the accidental inclusion of any transient data in our steady-state training sets. For all transients, at least five seconds of data were removed from the data stream. In the case of major power level changes, at least 10 seconds of data were removed, and for engine startups, at least 20 seconds of data were removed.

In the case of SSME Tests 902-548, 902-549, and 902-550, a slow facility venting schedule was in effect through nearly the whole duration of these tests. Since removing data affected by this venting would have removed most of the data from these tests, only the onset and offset of these slow venting operations were removed from the data stream. Transients due to power-level changes and repressurizations were removed in all cases.

The remaining steps described below used only the steady-state engine data that remained after the removal of transient data. The portion of steady-state data sampled for training is described below. All steady-state data, however, was screened by the system after training was completed.

E. Random Sampling of Nominal Training Data

As explained in Sections 2 and 3 above, only nominal data was used for training (“fitting”) each function approximation. The same training sets were used for all three function approximation techniques. All steady-state data segments identified by NASA engineers that contained anomalies (Table 1) were excluded from the training sets. An additional 220 seconds of nominal steady-state data in SSME test 901-671 (from 400 to 620 seconds into the test) were excluded from the training set to serve as a check that nominal data excluded from the training set would still be classified as nominal by the system. The resultant data available for training from tests 901-671, 901-672, and 901-673 totalled approximately 50,000 data points. For the test series 902-548, 902-549, and 902-550; the total was approximately 14,000 data points.

For each series, the actual training data set was created by randomly sampling 10 percent of the available data points. This was done for two reasons. First, training with the complete data sets would have been computationally prohibitive for any of the three function approximation techniques. The second reason was that, after training was completed, it would be possible to test the response of each function approximation technique on a different sample of data than was used in training. (Altogether, three independent, 10% samples of the nominal data were created for our study: (1) to serve as the training set, (2) for use in estimating the approximation error to compute nominal confidence intervals in Section 5.9, and (3) for screening (i.e., to test the response of the trained function approximation in Section 5.10). These same three samples were used for each of the function approximation techniques.

The 10% sample of training data was allocated among different power levels to give each power level approximately equal representation in the sample. This was necessary because the criterion to be optimized by both neural network techniques was mean-squared error over the training sample. Any significant inequality in the representation at different power levels would translate into unequal weighting for these power levels in the mean-squared error criterion, allowing a heavily-represented power level to be approximated closely at the expense of a lightly-represented power level.

F. Scaling of Input Variables

In the case of the Gaussian bar basis function technique, basis function centers are equally spaced over the range of each input variable as described in Section 4.1 above. The units of measurement for these input variables are therefore irrelevant to the definition of the basis functions, and to the activation and training of the basis function network.

For the other two techniques, backpropagation and linear regression, however, it is desirable to rescale all input variables to the same dimensionless scale so that the weights resulting from training reflect the actual influence of each variable, rather than an artifact of the scale on which that variable is measured. Each input variable was therefore rescaled to have a mean of zero and a standard deviation of 0.25 over the training set. (The same scale was subsequently used for data to be screened.) The data was scaled to a standard deviation of 0.25 so that all training data within four standard deviations of the mean would fall within the interval $[-1, 1]$ which is desirable for good performance of the back propagation algorithm.

G. Scaling of Predicted Variables

For all techniques, the variables to be predicted were biased to have a mean of approximately zero over the training set. Since all three approximation techniques contain adjustable bias terms, this constant bias was not strictly necessary but it served to improve the speed of convergence of the iterative techniques. Units of measurement of the predicted variables are irrelevant to the linear regression and Gaussian basis function techniques. In these two cases, the original engineering units were used so that the prediction errors would be expressed in meaningful units.

In the case of backpropagation, the sigmoidal output units are capable of producing outputs only within the range $(-0.5, 0.5)$ and can only approach the endpoints of this range asymptotically. For backpropagation training, therefore, each variable to be predicted was rescaled to a range of $[-0.25, 0.25]$ so it would fall well within the sigmoidal output range. All prediction errors computed by the backpropagation algorithm were then scaled back to the original engineering units of the predicted variables for comparison with the prediction errors of the other two techniques.

H. Parameters and Training Procedures for the Approximation Techniques

All approximation techniques were configured for the same set of 18 input variables, consisting of the 14 engine measurements shown in Table 1 and the discrete engine configuration variables described in Section 5.2 above. Each approximation technique was given the same two tasks—predicting the two variables shown in Table 1 from these 18 inputs. For the linear regression technique, “training” simply consisted of computing the coefficients of the best linear approximation to the predicted variable. Singular value decomposition was employed to compute these coefficients. Since the singular value decomposition employed

32-bit floating-point arithmetic, singular values smaller than .000001 of the largest singular value were zeroed.

The Gaussian bar basis function network was implemented with a middle layer of 60 ($M = 60$) basis functions per input variable, with the width σ_k set to half the spacing between adjacent basis functions. The total storage required for the neural network weights was 844 floating-point numbers. Even with this large number of basis functions we expected the basis function technique to learn more rapidly than back propagation, and to have the advantage of reliably converging without getting stuck on local minima. Since each Gaussian basis function was truncated to the interval $[\mu_{jk} - 4\sigma_k, \mu_{jk} + 4\sigma_k]$, each basis function will be zero outside of its own neighborhood. For a given value of an input variable, at most four of the basis functions for that variable will be nonzero. Thus, for each training example, at most four basis functions per input variable can affect the neural network output and weight adjustment. Equal spacing of the basis functions has the computational advantage that it is possible for these four nonzero basis functions to be identified by a simple subscript calculation.

The backpropagation technique was evaluated for configurations of 1-8, 10, 12, and 16 nodes in the middle layer of the network. Each configuration was given a fixed amount of computation for training for each of the two tasks defined in Section 5.1. For comparing these backpropagation configurations with the basis function technique, training computation was quantified as the number of scalar weight adjustments computed. This is a reasonable measure since it represents the inner loop of both neural network training algorithms, yet is independent of implementation software details. (We wished to compare the techniques themselves, not the efficiency of their software implementation.) In all cases, training was limited to 25 million scalar weight adjustments. This amount of training was sufficient for some, but not all, of the neural configurations to perform each prediction task well. The training number was thus chosen to differentiate the performance of the different techniques and configurations.

Each neural network technique was trained at the highest learning rate (rate of gradient descent) for which convergence to a global minimum was reliably achieved in both prediction tasks by all network configurations tested. For the Gaussian bar basis function network, this rate was 0.04. For back propagation, convergence to a local minimum in mean squared error was observed regardless of the learning rate whenever the learning rate of the input-to-middle-layer connections was the same as that of the middle-to-output-layer connections. The output connections were observed to quickly converge to a non-optimal local minimum in error without significant changes in the input-to-middle layer weights. Such local minima yielded correct predictions for one SSME power level but incorrect predictions for substantially different power levels, and remained stuck in this situation for as much as one billion weight adjustments. The solution to this problem was to employ a 10:1 ratio between the input-to-middle-layer learning rate and the

middle-to-output-layer learning rate. With this ratio, backpropagation training was able to achieve a low error rate at all power levels. The maximum learning rate which reliably achieved convergence in both tasks for all backpropagation configurations was 0.041 for the input-to-middle layer and 0.0041 for the middle-to-output layer.

I. Statistical Estimation of Nominal Approximation Error

As explained in Sections 2 and 3 above, the detection of anomalies depends upon surrounding the predictions made by each function approximation technique with a nominal confidence interval. To achieve an acceptably low false alarm rate, the confidence interval was set to five standard deviations above and below the predicted value. Since the neural networks were to be tested on different samples of data than those on which they had been trained, it was desirable to estimate the standard deviation of the prediction error from data other than the training set. For this purpose, an independent random sample of nominal data was constructed for each task. We used the same procedures as those employed for the training sets, but with a different seed for the random number generator.

Using each of the trained neural networks, and the fitted linear regression, predictions were generated for the independent samples of nominal data. In each case, a nominal standard deviation was calculated as the RMS prediction error over this independent nominal data. This nominal standard deviation was then used in the screening procedure described in Section 5.11 below.

J. Sampling of Data to be Screened

To test the ability of each function approximation technique to distinguish nominal from anomalous data, we constructed data sets containing both nominal and anomalous data to be screened by each technique. All steady-state data from each test was screened. To construct data sets for screening, the steady-state data extracted from each SSME test (as described in Section 5.4 above) were periodically sampled at the rate of five samples per second. A separate screening data set was constructed for each of the six SSME tests shown in Table 1. Each screening data set contained the same 18 input variables described in Section 5.2 above.

K. Screening Procedure and Visualization of Results

Each trained neural network was applied to each screening data set to generate a time series of expected nominal values for the predicted variables shown in Table 1. Expected nominal values were also generated using the linear regression technique. (The nominal predictions are the middle dotted lines in Figures 4-6.) As explained in Sections 2 and 3, each prediction was surrounded by a confidence interval of five standard deviations above and below the nominal prediction, as calculated according to Section 5.9 descriptions.

The actual measured values of these predicted variables were then screened as follows: The measured value at each

time was compared to the expected nominal value predicted by the given neural network or by linear regression. Each measured value was classified as nominal if it fell within the confidence interval of five standard deviations above and below the predicted nominal value. This screening is portrayed graphically in Figures 4-6, in which the measured values are shown as a solid line and the confidence interval is bounded above and below by dotted lines. Measurements falling within the confidence interval are classified as nominal; those falling outside the confidence interval are classified as anomalous. The results shown in these figures are discussed in the next section.

This screening process was repeated on the same data sets using the trained Gaussian bar basis function network; trained backpropagation networks with 1-8, 10, 12, and 16 middle nodes; and the linear regression technique.

VI. RESULTS AND DISCUSSION

A. Training

Figure 3a shows the progress achieved by each neural network technique in learning to predict PID 42, fuel pre-burner oxidizer valve (FPOV) actuator position. The training set was composed of nominal steady-state data from SSME Tests 901-671, 901-672, and 901-673. The ordinate of Figure 3a shows RMS error over the training set, which is the criterion each technique is attempting to minimize. Because training reduces this error by more than two orders of magnitude a logarithmic scale is used. Each plotted line in Figure 3a shows the reduction in this error as a function of the amount of training. (The amount of training is measured as the number of scalar weight adjustments as explained in Section 5.8.) The dotted line in Figure 3a shows the reduction in error achieved by the Gaussian bar basis function technique. The solid lines show the same information for backpropagation networks with 1-8, 10, 12, and 16 nodes in the middle layer. Among these backpropagation configurations, best results were obtained with four middle nodes. Increasing the number of middle nodes slowed down training, as expected for the backpropagation algorithm.

Figure 3b shows same information for the prediction of PID 221, pressure change in the POGO surge suppression system, trained on nominal data from SSME Tests 902-548, 902-549, and 902-550. For this prediction, the best backpropagation results were obtained with seven middle nodes.

Since linear regression was computed directly rather than iteratively, it is not shown in Figure 3. Linear regression is inherently simpler than either neural network technique, and there is little question that it is the fastest of the three techniques by any reasonable measure of computation.

As expected, the Gaussian bar basis function technique shows more rapid training than backpropagation in both tasks. This result was expected because only one layer of weights is being updated, and also because only a small fraction of the weights are adjusted for any one training example, as discussed in Section 5.8 above. For the prediction shown in Figure 3a, the basis function network achieved

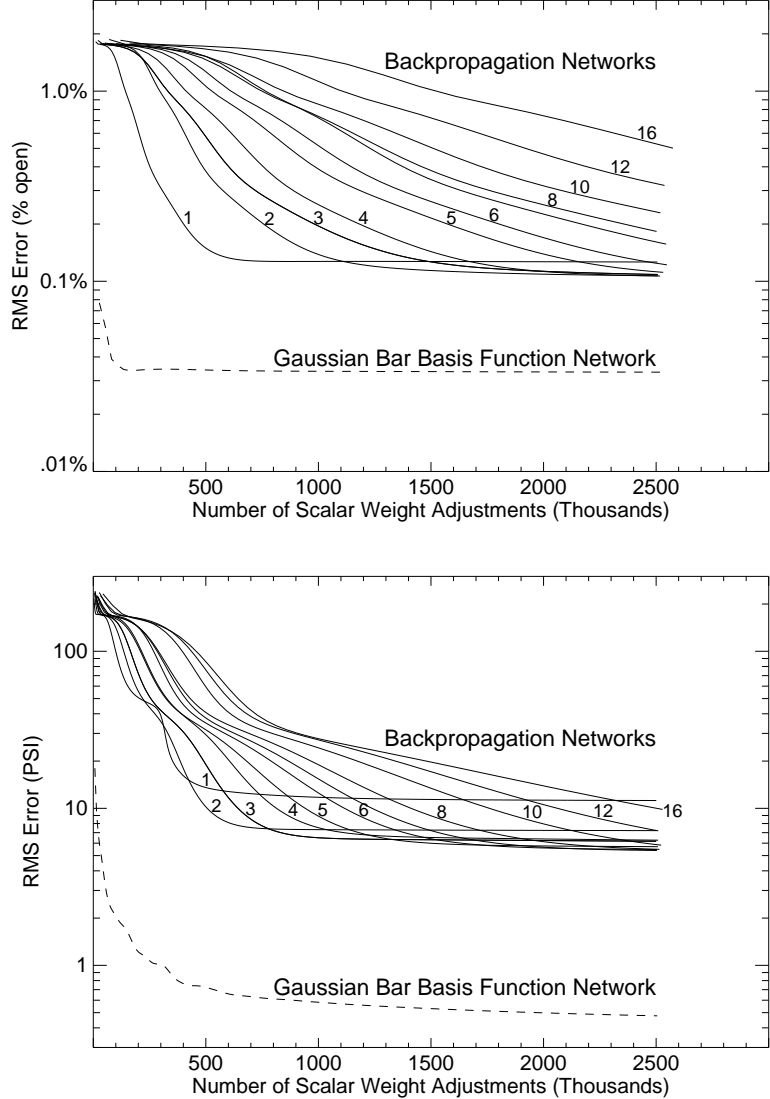


Fig. 3. Comparison of training speed of the neural network architectures. RMS error of each neural network prediction decreases as training progresses. Solid lines indicate backpropagation networks with 1-8, 10, 12, and 16 middle nodes, with the number of middle nodes indicated to the right of each line. Dotted lines indicate the Gaussian bar basis function network. This information is shown for the two tasks of Table 1: (a) Prediction of Fuel Preburner Oxidizer Valve Position; and (b) Prediction of Pressure Change in POGO Surge Suppression System.

one-third the RMS error of the best backpropagation network. For the task shown in Figure 3b, the final RMS error of the Gaussian bar basis function approximation was only one-tenth that of back propagation.

Figure 3 shows that a very large number of basis functions can be trained more rapidly than a back propagation network of comparable or even much smaller size. As discussed in Section 5.8, this advantage arises from the local nature of the basis functions. Since each basis function is zero outside its immediate neighborhood, only a few of the basis functions are affected by each training example, and hence only a few of the middle-to-output layer weights need to be adjusted. Since the basis functions themselves are fixed, there are no input-to-middle layer weights that need adjusting as in backpropagation.

Given the relative ease of training of the Gaussian bar basis function network, the question remaining is whether a function approximation as accurate as that of back propagation can be achieved. This question is addressed in the next section.

B. Screening

Each of Figures 4, 5, and 6 compares predictions made by the three techniques of Gaussian bar basis functions, backpropagation, and linear regression. Each of these figures has three parts corresponding to the three techniques tested. To avoid clutter, the backpropagation results shown are the best results obtained from the eleven different backpropagation configurations tested. The predictions made by each technique are indicated with dotted lines. In each

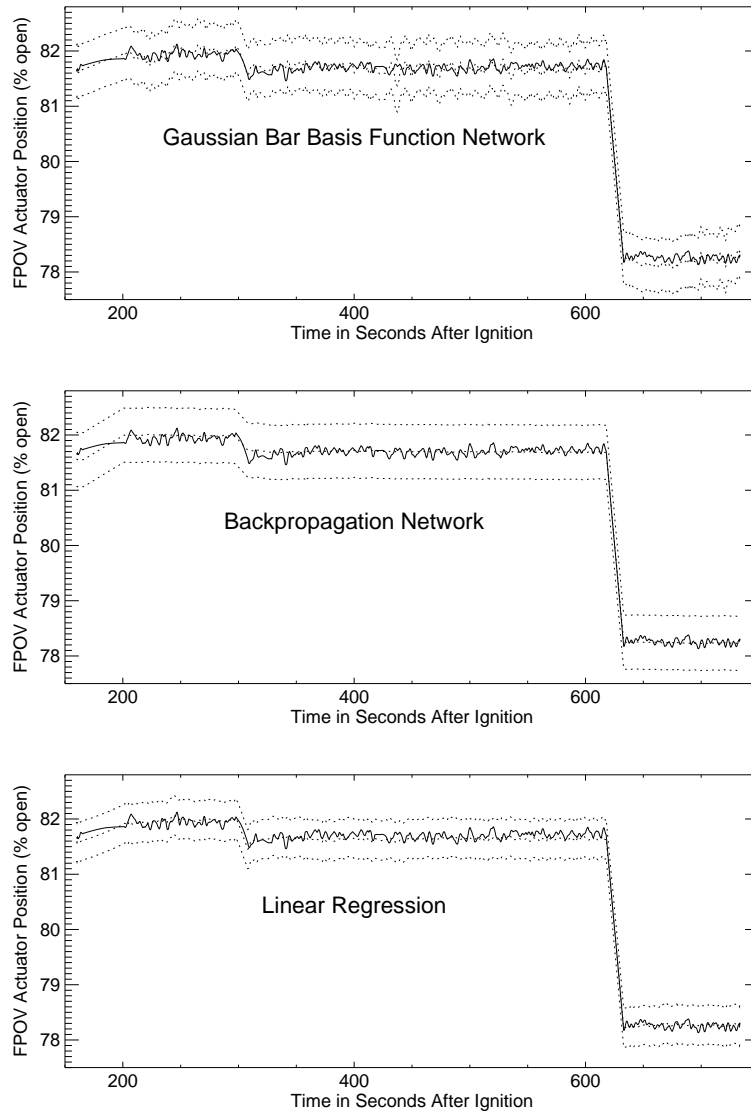


Fig. 4. Screening of data from a nominal SSME test (901-671) by three approximation techniques: (a) the Gaussian bar basis function network, (b) the best backpropagation network, and (c) linear regression. The engine parameter being predicted is Fuel Preburner Oxidizer Valve Actuator Position. In each plot, the nominal confidence interval calculated by the given technique is bounded by the upper and lower dotted lines. The measured value of the engine parameter is shown as a solid line. The measurements were correctly classified as nominal by all three techniques, with no false alarms.

case, the middle dotted line represents the expected nominal value predicted by the given technique from the set of inputs shown in Figure 1 and detailed in Table 2. The upper and lower dotted lines show the nominal confidence interval for this prediction, as explained in Sections 5.9 and 5.11. The solid line shows the actual measured value of the parameter being predicted. The solid line is identical in parts a, b, and c of each figure. The interpretation of these figures is based on whether the actual measurement falls inside or outside the nominal confidence interval at any given time. When the measurement falls inside the confidence interval, it is classified as nominal by the given technique; when it falls outside, it is identified as anomalous.

Figure 4 compares the screening performance of the three techniques on the known nominal data of SSME test 901-

671. The purpose is to verify that each technique will correctly classify a sample of nominal data different from the sample on which it was trained. Prior to 400 seconds in the test, this is merely a question of whether an independent 10% sample of the data (as defined in Sections 5.5 and 5.10) will be correctly classified. All data from 400 to 620 seconds, however, was deliberately excluded from the training sample to verify that this data would be correctly screened. All three techniques correctly classified this data as nominal with no false alarms, as shown in parts a, b, and c of Figure 4.

Figure 5 compares the screening performed by the three techniques on SSME test 901-672. As shown in Table 1, an anomalous shift in PID 42 (FPOV actuator position) was identified by NASA analysts. All three function-

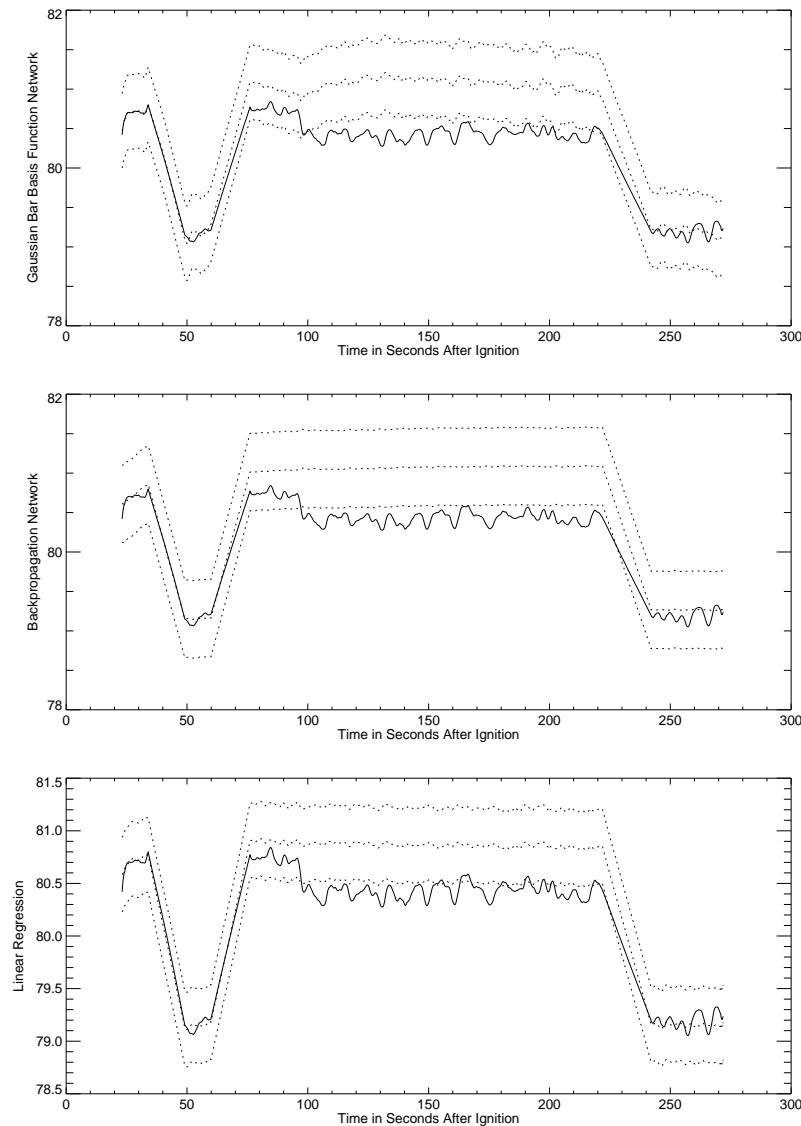


Fig. 5. Screening of data from SSME test 901-672, which was determined by NASA to have an anomaly at 96 seconds indicated in the Fuel Preburner Oxidizer Valve Actuator Position. All three techniques detected this anomaly. The anomaly is indicated when the measured value of this valve position (solid line) moves outside the nominal confidence interval (bounded by dotted lines).

approximation techniques correctly identify the anomalous shift in this measurement beginning approximately 96 seconds after ignition. Figures 4 and 5 show that nominal behavior of the SSME can be predicted from the external influences shown in Figure 1. Figure 5 shows the accuracy of the prediction that can be obtained. The accuracy is such that the fluctuation due to the anomaly can be clearly distinguished from the fluctuations that nominally arise inside the SSME and due to measurement errors. More specifically, the magnitude of the shift representing the anomaly is greater than five times the RMS error representing nominal fluctuations. This result is achieved by all three of the function-approximation techniques tested.

Figure 6 compares the three techniques in screening SSME test 902-549. In this case, the anomaly is a shift in PID 221, pressure change in the POGO surge suppression system. The anomaly appears to be more subtle than

that of test 901-672, and is not detected equally well by the three techniques. The Gaussian bar basis function network constructs a much tighter confidence interval than the other two techniques. In Figure 6a, the dotted lines representing the upper and lower bounds of this confidence interval are nearly indistinguishable from each other and from the actual measurements before the anomaly occurs. This is not too surprising given that the training of the Gaussian bar network achieved one-tenth the error rate of the backpropagation networks (Figure 3). The result is that the anomaly of Figure 6 is most clearly detected by the Gaussian bar network, falling well outside the five-standard-deviation nominal confidence interval. For the backpropagation network, on the other hand, the nominal confidence interval is considerably wider, and the anomaly does not fall outside this nominal confidence interval. The backpropagation technique thus fails to detect the anomaly.

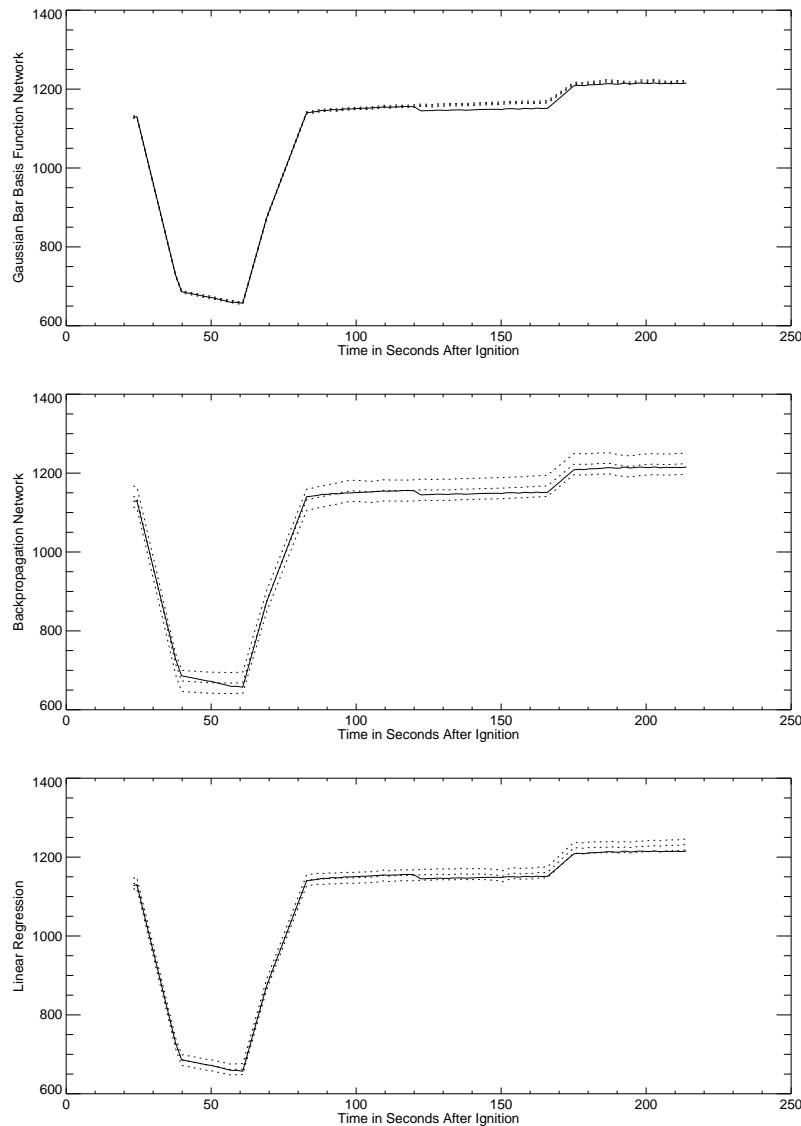


Fig. 6. Screening of data from SSME test 902-549, which was determined by NASA to have an anomaly at 120 seconds indicated in the Pressure Change in the POGO Surge Suppression System. In all plots, the dotted lines show a nominal confidence interval of plus or minus 5 times the nominal RMS prediction error. (a) The Gaussian bar basis function network constructs a very tight nominal confidence interval. The anomaly is detected when the measured value of this parameter (solid line) moves well outside this confidence interval, to a deviation greater than 20 times the nominal RMS prediction error. (b) Even the best of the backpropagation networks fails to detect the anomaly. The measured value stays well within the nominal confidence interval. The maximum deviation due to the anomaly ranges from 2 to 3 times the nominal RMS prediction error. (c) For linear regression, the measured value lies near the border of the confidence interval, ranging from 4 to 6 times the nominal RMS prediction error.

The results for linear regression fall between those of the other two techniques, with the anomaly falling near the boundary of the nominal confidence interval.

VII. CONCLUSION

The main hypothesis of this study was that the *nominal* value of a given engine parameter at a given time could be predicted from external influences on the SSME. Based on the boundary we drew around the SSME, these external influences included not only the commanded power level, but also the pressures in the fuel and oxidizer inlets, venting lines, and repressurization lines. These measurements of external influences were used to predict the nominal values

of two internal engine parameters. More specifically, our hypothesis was that this prediction could be made with a tight enough confidence interval to be useful for detecting anomalies, in spite of the many sources of variability within the SSME and its measurement channels. This hypothesis appears to be supported by the data we have presented. If the anomalies to be detected are considered signals, and nominal fluctuations in the measurements are considered noise, then both anomalies used as test cases were detected at over five times the RMS noise level using the Gaussian bar basis function network. With nominal predictions surrounded by a confidence interval of five times the RMS prediction error, no false alarms would be expected from

nominal fluctuations, and none were observed.

Nominal predictions and confidence intervals were also created using two other function approximation techniques, the backpropagation neural network and linear regression. The three techniques all gave comparable results in the task of screening engine parameter 42, FPOV actuator position. In the apparently more difficult task of screening engine parameter 221, pressure change in the POGO surge suppression system, the Gaussian bar basis function network detected the anomaly at 20 times the RMS noise level; linear regression at 4-6 times the RMS noise level; and backpropagation at only 2-3 times the RMS noise level. The basis function network also trained more rapidly than backpropagation and did not get stuck in local minima.

The chief appeal of the function-approximation approach to anomaly detection is that it is not limited to detecting specific, foreseen classes of anomalies, in contrast with most other types of neural network and expert system approaches. Instead, training requires nominal data only, and anomaly detection is based on nominal confidence intervals. Based on the results we have presented, it appears that the function-approximation approach, and its implementation using basis functions, merits more extensive investigation of its practicality for screening large amounts of propulsion system test data. The approach needs to be tested on a larger scale, and it needs to be extended to cover transient as well as steady-state data. Both types of further work are planned and will be reported at a later date.

VIII. ACKNOWLEDGEMENTS

A portion of this work was supported by NASA Contract NAS8-39184. The opinions expressed herein, however, are those of the authors only. We thank Timothy Choate for contributing to the software development and Jo Anne Malone for carrying out studies which preceded those reported here. We are grateful to Michael Whitley, Catherine McLeod, and Gary Lyles for valuable suggestions and for facilitating the work in many ways. We thank Marc Neely and Eric Sander for expert guidance in selecting and interpreting the SSME ground tests used in this study, and Jeff Cornelius for assistance in software installation. Subroutines written by the Boeing Corporation were used to access the NASA databases. Our software also made use of the National Institutes of Health C++ Class Library, and of a public-domain library of C++ routines developed by Choate and Whitehead.

REFERENCES

- [1] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321-355, 1988.
- [2] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481-1497, 1990.
- [3] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.
- [4] S. Geva and J. Sitte, "A constructive method for multivariate function approximation by multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 3, pp. 621-624, 1992.
- [5] Y. Ito, "Approximation of functions on a compact set by finite sums of a sigmoid function without scaling," *Neural Networks*, vol. 4, pp. 817-826, 1991.
- [6] Y. Ito, "Approximation of continuous functions on R^d by linear combinations of shifted rotations of a sigmoid function with and without scaling," *Neural Networks*, vol. 5, pp. 105-115, 1992.
- [7] V. Venkatasubramanian and K. Chan, "A neural network methodology for process fault diagnosis," *AIChE*, vol. 35, pp. 1993-2002, 1989.
- [8] T. H. Guo and J. Nurre, "Sensor failure detection and recovery by neural networks," in *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, pp. I-221-I-226, Piscataway, N.J.: IEEE, July 1991.
- [9] M. Kotani, H. Matsumoto, and T. Kanagawa, "Acoustic diagnosis for compressor with hybrid neural network," in *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, pp. I-251-I-256, Piscataway, N.J.: IEEE, July 1991.
- [10] P. A. Jokinen, "Comparison of neural network models for process fault detection and diagnosis problems," in *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, pp. I-239-I-244, Piscataway, N.J.: IEEE, July 1991.
- [11] K. J. Cios, R. E. Tjia, N. Liu, and R. A. Langenderfer, "Study of continuous ID3 and radial basis function algorithms for the recognition of glass defects," in *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, pp. I-49-I-54, Piscataway, N.J.: IEEE, July 1991.
- [12] M. Yuen Chow and S. O. Yee, "Robustness test of an incipient fault detector artificial neural network," in *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, pp. I-73-I-78, Piscataway, N.J.: IEEE, July 1991.
- [13] B. A. Whitehead, H. J. Ferber, and M. Ali, "Neural network approach to space shuttle main engine health monitoring," in *AIAA/ASME/SAE/ASEE 26th Joint Propulsion Conference*, July 1990. American Institute of Aeronautics and Astronautics, Paper 90-2259.
- [14] B. A. Whitehead, E. L. Kiech, and M. Ali, "Rocket engine diagnostics using neural networks," in *AIAA/ASME/SAE/ASEE 26th Joint Propulsion Conference*, July 1990. American Institute of Aeronautics and Astronautics, Paper 90-1892.
- [15] D. R. Hush and J. M. Salas, "A performance of neural network classifiers for the 1-class classifier problem," in *Proceedings of the International Joint Conference on Neural Networks*, pp. I-396-I-407, Piscataway, N.J.: IEEE, January 1990.
- [16] J. A. Leonard, M. A. Kramer, and L. H. Ungar, "Using radial basis functions to approximate a function and its error bounds," *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 624-627, 1992.
- [17] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281-294, 1989.
- [18] E. Hartman and J. D. Keeler, "Semi-local units for prediction," in *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, pp. II-561-II-566, Piscataway, N.J.: IEEE, July 1991.
- [19] E. Hartman and J. D. Keeler, "Predicting the future: Advantages of semilocal units," *Neural Computation*, vol. 3, no. 4, pp. 566-578, 1991.
- [20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Volume 1: Foundations* (D. E. Rumelhart, J. L. McClelland, and T. P. R. Group, eds.), pp. 318-364, Cambridge, MA: The MIT Press, 1986.
- [21] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 WESCON Convention*, pp. 96-104, New York: Institute of Radio Engineers, 1960.