View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CiteSeerX

An Integrated Fault Diagnostics Model Using Genetic Algorithm and Neural Networks

Suresh Sampath

Riti Singh

Department of Power Propulsion and Aerospace,
School of Engineering,
Cranfield University,
Cranfield, Bedfordshire MK43 0AL, UK

This paper presents the development of an integrated fault diagnostics model for identifying shifts in component performance and sensor faults using the Genetic Algorithm and Artificial Neural Network. The diagnostics model operates in two distinct stages. The first stage uses response surfaces for computing objective functions to increase the exploration potential of the search space while easing the computational burden. The second stage uses the concept of a hybrid diagnostics model in which a nested neural network is used with genetic algorithm to form a hybrid diagnostics model. The nested neural network functions as a pre-processor or filter to reduce the number of fault classes to be explored by the genetic algorithm based diagnostics model. The hybrid model improves the accuracy, reliability, and consistency of the results obtained. In addition significant improvements in the total run time have also been observed. The advanced cycle Intercooled Recuperated WR21 engine has been used as the test engine for implementing the diagnostics model. [DOI: 10.1115/1.1995771]

Introduction

It is a commonly known fact that engine condition monitoring is an effective but complex way to improve safety and reduce operating and maintenance costs of gas turbines. Engine health monitoring systems have become increasingly important in recent years due to the development of engines with improved power to weight ratios. Additionally, the need to show enhanced reliability at reduced costs will require major advances in engine controls and engine fault diagnosis capability. In the fundamental sense, performance monitoring and fault diagnosis involves processing of engine measurements. In all the cases, a comparison of some parameter values of an engine under examination is performed with the corresponding values of an engine which is considered "healthy." The parameters used and the method of analyzing them characterizes each different diagnostic method. Broadly speaking, these techniques have not changed much from the 1970s and mainly rely on what is known as the Gas Path Analysis (GPA) [1,2]. The advantages and limitations of the GPA have been extensively debated and different ways to overcome the limitations were proposed [3]. In recent times the use of artificial intelligence techniques and optimization techniques like the genetic algorithm have been on the rise. Interesting research has been undertaken by several researchers [4,5] including work on Kalman filters [6]. Zedda and Singh [7] used the ANN technique for engine fault diagnosis and Sensor Fault Diagnosis and Isolation (SFDI) and reported high accuracy.

Engine Fault Diagnostics Using Ga

The use of a genetic algorithm for engine fault diagnostics has been investigated by several authors in the recent past and reported good accuracy. Zedda [8] used this technique for developing a diagnostics model for the EJ200 engine with test bed instrumentation. Gulati [9] extended the technique to a poorly instrumented engine using the concept of multiple objective point analysis. He validated the technique on the RB199 engine. Sampath et al. [10] have used the GA based technique for investigating faults in advanced cycle engines like the ICR WR21 and reported considerable success. All the methods used a similar objective function given by Eq. (1).

$$J(x) = \sum_{j=1}^{M} \frac{|z_j - h_j(x,w)|}{z_{odj}(w) \cdot \sigma_j} \tag{1}$$

Integrated Fault Diagnostics Model

The diagnostics system developed for the advanced cycle ICR WR21 was able to detect the component faults and instrumentation faults to reasonable accuracy. However, the model had its own limitations, particularly the long run times and, therefore, has necessitated further enhancement. It is clear that the center of gravity of the algorithm lies in the calculation of the objective function, which in turn is mapped to the fitness function. The fitness function is the parameter which dominates the search process.

The calculation of the objective function requires the performance model to be run twice (clean condition and faulty condition). Therefore, any reduction in the number of calls to the performance model can significantly reduce the overall run time of the algorithm. It has been estimated that the engine performance code run constitutes almost 75%–80% of the total runtime.

After carrying out a detailed study of the problem, several techniques have been worked out to overcome the limitations, while enhancing the accuracy, consistency, and reliability of the diagnostics system. The speed of convergence of the performance model is beyond the control of a diagnostics engineer, but one way to overcome this problem is to make fewer calls to the performance code. Perhaps, the use of a function or a response surface which is a suitable representation of the engine performance model for the initial generations to eliminate the weaker individual can reduce the total run time.

Response Surface Method

In many problems we have specific knowledge that allows us to construct approximate models of our problem. In turn, modeling capability allows us to create more or less accurate approximations to our objective function. With genetic algorithms, this knowledge can be put to good use by reducing the number of

Contributed by the International Gas Turbine Institute (IGTI) of ASME for publication in the JOURNAL OF ENGINEERING FOR GAS TURBINES AND POWER. Manuscript received October 1, 2003; final manuscript received March 1, 2004. IGTI Review Chair: A. J. Strazisar. Paper presented at the International Gas Turbine and Aeroengine Congress and Exhibition, Vienna, Austria, June 13–17, 2004, Paper No. GT2004-53914.

Copyright © 2006 by ASME

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

full-cost function evaluations. In many optimization and search problems, a single function evaluation is a fairly costly process, involving many layers of subroutines, numerical or symbolic computation, iterations and various coding and decoding functions. As a result, if savings in computation time are possible through approximate, perhaps a rough estimation, of the objective function, they are worth pursuing so that more evaluations can be performed in the same time. Interesting work in this regard have been carried out by Montgomery [11] and Myers [12]. This observation is particularly relevant to genetic algorithms, as we expect GAs to behave robustly under error and noise because of their population -sampling approach.

In this case a response surface is a complex nonlinear function representing the engine performance model. The aim of the response surface method is to obtain an objective function of a given string without having to run the performance code. This method, which completely avoids the performance model, is very useful in the earlier generations of the GA diagnostics model. A suitable representation of the performance code is essential in order to implement this technique, which can be beneficial in speeding up the overall process. The rationale behind this concept is to avoid spending time on evaluating strings which are to be discarded in the initial generations.

In order to implement the response surface method, the traditional objective function shown in Eq. (1) needs to be modified by splitting it into two objective functions. Since the response surface is created by implanting faults and comparing it with baseline values, the objective function obtained from the response surface will be with respect to the baseline and needs to be compared with the objective function obtained from the actual data and baseline. The modified objective function is given by:

$$\Delta J = \left| \sum_{j=1}^{NM} \frac{|z_j^b - h_j(x,w)|}{z_{odj}(w) \cdot \sigma_j} - \sum_{i=1}^{NM} \frac{|z_i^b - z_i^s|}{z_{odj}(w) \cdot \sigma_i} \right| \tag{2}$$

The parameters are the same except for the superscripts "*b*" and "*s*," which mean baseline and simulated, respectively. A set of measurements is obtained from the engine and is compared with the corresponding baseline (clean parameters) measurements. An objective function is calculated which is designated as J_1. This is required to be calculated only once. The other objective function, J_2, is directly obtained from the response surface. The optimization of ΔJ (difference between J_1 and J_2) tends to achieve the best solution or more appropriately, eliminates the bad solutions.

At this point it is pertinent to mention that the optimization of ΔJ shown in Eq. (2) may not produce the exact match, for the reason that faults with different signatures could also have similar values of objective function. The fault signatures producing $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ could be different and it is not the same as calculating $\boldsymbol{J}$ between the faulty data and the simulated parameter using Eq. (1). However, what is important at this stage is to identify the strong individuals and create a condition for the weak individuals to be eliminated early in the search process. The calculation of $\boldsymbol{J}_1$ gives the sum of deviations from the baseline. This value indicates that, deviations producing $\boldsymbol{J}_2$, which are close to $\boldsymbol{J}_1$ in magnitude are likely candidates for further examination. At this point, the signs of deviations are not considered as they will be eliminated when the objective function is calculated with measurements obtained from the performance model using Eq. (1).

Data for the response surface is generated using the engine performance model. It is very similar to generating the search space by varying the engine performance parameters (component flow capacities and efficiencies) in small steps from its baseline values and implanting it into the engine performance model. A set of measurements are obtained for these conditions and compared with corresponding engine baseline parameters. The sum of the deviations is the objective function. This data is used to generate a response surface. Some of the ways to develop a response surface which have been investigated are enumerated below.

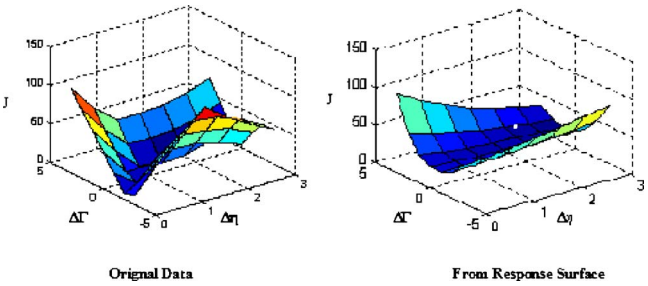

Fig. 1 Objective function using Gaussian function

The first method involved the development of a general regression function using the Gaussian functions and high-order polynomials. A sample is shown by Eq. (3) which represents a complex function like the search space. The aim is to evolve an equation which can readily return an objective function when given a set of deviation in the engine performance parameter. Once the regression coefficients have been obtained, the reanalysis and sensitivity analysis represented by Eqs. (3) and (4) require trivial computational effort. The constants and coefficients have been obtained using the MATLAB statistical toolbox.

$$f(x,y) = Ae^{(-b_1x-b_2y)} + Be^{(-b_3x-b_4y)} + \cdots. \tag{3}$$

in the case of engine fault analysis the equation can be re-written as

$$J_{RS} = Ae^{(-b_1\Delta\eta-b_2\Delta\Gamma)} + Be^{(-b_3\Delta\eta-b_4\Delta\Gamma)} + \cdots. \tag{4}$$

Figure 1 shows a comparison between the original data and the data generated from the response surface generated using the Gaussian function shown in Eq. (4). It is evident that the response surface is barely able to reproduce the original data and, therefore, would not be of much value considering the fact that it is expected to reproduce the surface details to reasonable accuracy.

The second option was to use the Radial Basis Functions (RBF). RBFs have attracted a great deal of interest due to their rapid training, generality, and simplicity. They have been widely used for the Generalized Regression Neural Networks (GRNN). They are several orders of magnitude faster in training when compared to the standard back propagation but have a major disadvantage: After training, they are generally slower to use, requiring more computation to perform a classification or function approximation. It can approximate any arbitrary function between input and output vectors, drawing the function estimate directly from the training data. Furthermore, it is consistent, that is, as the training set size becomes large, the estimation error approaches zero. The accuracy of the estimated function is very high as long as the input vector is close to the training vector. The RBFs are highly localized and, therefore, need large amounts of training data. The large amount of training data creates a large number of nodes. Though the training is very quick, the function approximation is a computationally intensive process.

Having investigated the possibility of representing the search space in the form of a response surface using the first two options and carefully considering the advantages and disadvantages of the process, it was decided to investigate the possibility of using a FFBPNN for the generation of a response surface. The FFBPNN is a standard neural network widely used for classification. The FFBPNN are known to represent complex functions very accurately when trained with appropriate and adequate samples. Figure 2 shows the response surface developed using the data from Fault class-2 (Fault in HPC) The network (8-20-20-1) was trained using data generated by implanting faults in the engine performance model and the objective function calculated with measurements obtained and baseline measurements.

From Fig. 2 it can be seen that the data represented using FFBPNN is accurate for the same data set. It is a trade-off between the

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

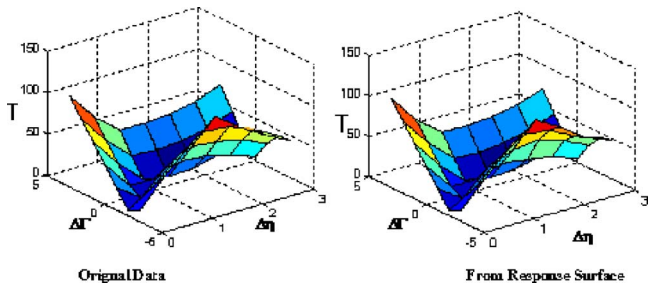

Fig. 2 Data representation using FFBPNN

other two methods and can be used effectively in the initial stages of the search process. While the calculation of an objective using the engine performance model takes a significant portion of a second, several hundred objective functions can be obtained during the same time using the response surface.

The number of generations for which the RSM is not fixed and is a matter of choice. Experiments show that using the RSM for the initial 25% of the total number of generations is beneficial.

Though a considerable amount of time can be saved by using the RSM, However, the algorithm has to search all the fault classes in order to arrive at a solution. Figure 3 shows a typical optimization process consisting of 100 generations out of which 25 generations use the RSM to compute the fitness values of strings. The diagnosis of multiple component faults lead to a large number of fault classes to be explored, which in turn leads to large run times. The number of fault classes that need to be searched could be further reduced by developing a hybrid system in which a pre-processor algorithm can be used to classify the fault classes into the appropriate groups and suggest a single/group of fault classes to be explored by the GA optimizer.

There are several fault diagnostics techniques and optimization techniques which can be combined with the GAs to form a hybrid system for efficient fault diagnostics. Researchers like Gulati [6] have suggested the use of the GA for a broader search and the use of calculus-based methods for the local search. The approach adopted here is to identify a method which could be used to reduce the number of fault classes, so that the GA module can be used to search the fault classes for faulty components and quantify the fault. The rationale behind this is to avoid searching fault classes which are not likely to have faulty components.

Hybrid Diagnostics Model Using Ga and ANN

When problem specific knowledge exists, it may be advantageous to consider a GA hybrid. Genetic algorithms can be crossed with various problem specific search techniques to form a hybrid

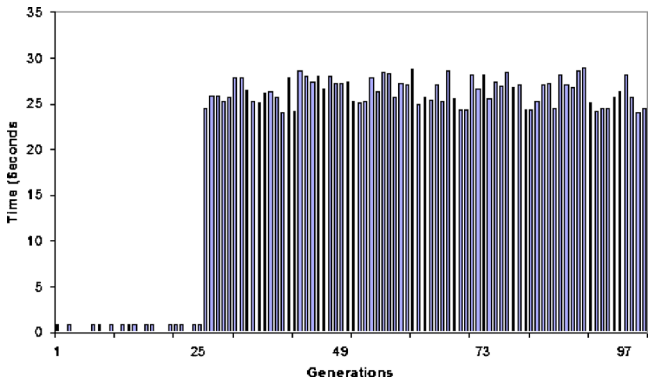

Fig. 3 Comparison of time taken for objective function

that exploits the global perspective of the GA and the convergence of the problem specific technique. A number of authors have suggested such hybridization (Bethke [13]; Bosworth et al. [14]; Goldberg [15]). However, there is not much published work describing the results of GA-hybrid studies. Nonetheless, the idea is simple, has merit and may be used to improve the ultimate genetic search performance.

For the problem in hand, an analogy can be drawn with the diagnostic engineers of yesteryear, when modern-day engine fault diagnosis techniques were in their infancy and when an analyst would make an intelligent assessment based on certain thumb rules. The method used was similar to the fault tree method in which the branches of the fault tree are traversed by eliminating certain criteria to arrive at the fault. The fault tree technique has the limitation that only single component faults can be identified. However, in the case of the proposed hybrid system, a Fault Class Classifier (FCC) is expected to identify the likely fault class(es) and the GA optimizer would subsequently explore the fault classes and quantify the faults associated with the components.

An extensive literature study showed that the feed forward back propagation network remains an effective paradigm and is by far the most commonly applied neural network for classification problems. Ogaji and Singh [16] have successfully used the concept of cascade neural networks for component and sensor fault identification. An informal count indicates that more than 85% of published applications have used the FFBPN. In difficult applications where the input-output relationships are nonlinear and/or involve high-order correlations among the input variables, back propagation has produced accurate results. The disadvantage of its slow training is partially offset by its rapid computation in the forward direction.

It was felt that the ability of the ANN to classify the given data with a relatively small network can be used to act as a preprocessor for the GA diagnostics. Even if the neural network is able to classify the given data as a single or multiple component fault, the search time is reduced to 25% of the original time in the case of single component faults (the GA module has to search only the first 7 fault classes) or 75% of the original time in case of multiple *components* fault (the GA module has to search 21 fault classes).

A schematic diagram of the concept of the hybrid model is shown in Fig. 4. The hybrid model consists of a Nested Neural Network (NNN) in which each network has a limited task of classifying the data into subgroups.

Perhaps, one way to classify is to train a network for all possible combinations of faults. But the training set required to fully represent all possible combinations of health parameters and sensor biases will be prohibitively large. It would take excessive time to train such a neural network and their performance might not reach satisfactory levels. In the light of the above, the problem domain has been partitioned into smaller and specific tasks. A node in the NNN is trained to classify the given input into any one of the subcategories, usually two or more subcategories or BRANCH nodes (described later).

This approach has been found to be more accurate when compared to training a single large network to identify the faulty components. The dark line shows the flow of the algorithm to arrive at the final stage, for an arbitrary set of measurements. The nodes are classified into "BRANCH" nodes and "TERMINAL" nodes. The bottom most level of each branch/path consists of the terminal nodes. The "TERMINAL" nodes have an important task of extracting and submitting the fault class(es) for optimization by the GA Module.

Input data from an engine is fed into the NNN pre-processor. The first step is to identify whether the input data was generated due to a faulty component or a faulty sensor. Once the classification has been made, the data is forwarded to the next level. If the input data is associated with a component fault, then it is forwarded to a node classifying the input data into Single or Multiple component faults. In the same way, the input data traverses

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

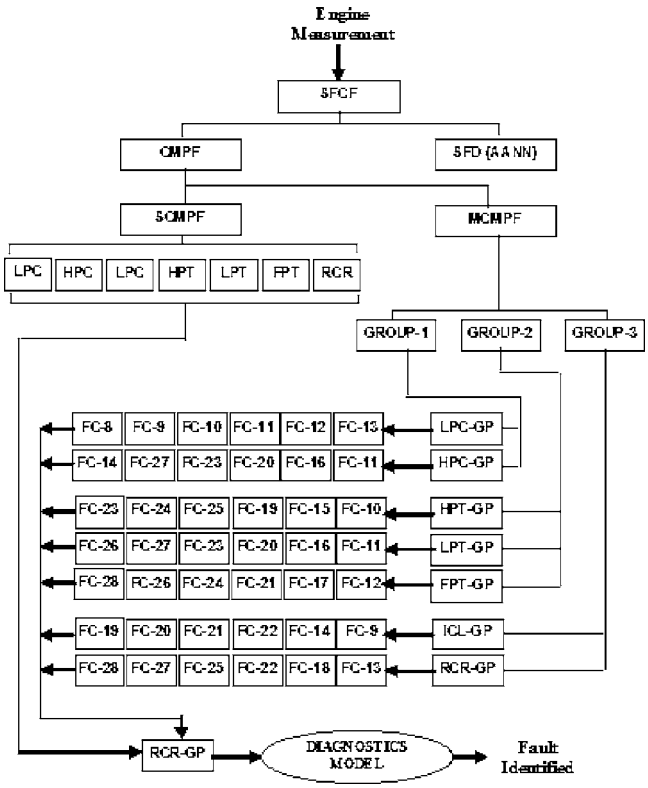

Fig. 4 Schematic of a hybrid diagnostics model

through the BRANCH nodes and arrives at the TERMINAL node. A terminal node will classify the input data into a single fault class or a group of fault classes to be explored by the GA optimizer.

Whether in classification or regression, it is necessary to employ appropriate training algorithms. For the feed forward back propagation network, various training algorithms such as resilient backpropagation, delta-bar-delta, conjugate gradient algorithms. Levenberg Marquardt, Bayesian regularization, etc. are available. The choice of algorithm is usually a trade-off between many factors such as minimum RMS error obtainable, length of training time or speed of convergence, memory requirements, nature of the problem. The choice of the algorithm is left to the user and the type of problem being solved.

In the present work, several algorithms were tested and the conjugate gradient method was found to be most suitable from the point of the speed of convergence and memory requirement. The algorithm gave good results by improving the generalization especially with respect to classification, which is the main requirement in the model. The radial basis functions were also experimented but they need a large amount of nodes and that gives a particularly good result if the input data is close to the training data.

The number of layers and number of processing elements per layer are important decisions to be made during the design of an ANN. The number of processing elements in the input and output layers are fixed by the number of input measurements and required output vectors, respectively, and, therefore, only the number of hidden layers and number of processing elements in the hidden layers are to be determined. There are no good solutions to this problem and is again dependent of the nature of the problem being addressed. In general, as the complexity in the relationship between the input data and the desired output increases, then the number of processing elements in the hidden layer should also increase. The number of processing elements may also depend on the amount and the quality of training data. It is noteworthy that a fewer number of processing elements would mean there are insufficient network parameters (weights and biases) to undertake the required tasks (which leads to under learning of the problem domain), while more than necessary processing elements in the hidden layers would lead to poor generalization as the features of the training patterns are memorized (making the network less capable to apply knowledge learned to patterns that were not included in the training process though within the problem domain). In other words, the network becomes useless on new data sets.

Table 1 shows the description of individual nodes. The letter "L" denotes the level and the number gives the level number. The letter "N" denotes the node and the number suffixed shows the position of the node in a particular level. Each node has a specific function to perform as part of the network. Table 2 shows the type of network and configuration of the individual nodes in the nested neural network.

Various configurations of the MLPs were tried and the configuration shown in Table 2 was found to give good classification. The fault classes used to train the nodes are shown in the last column of Table 2.

Confidence Rating of Networks

Once the individual networks have been trained to classify data into specific subgroups, they can be integrated to form a nested neural network for different levels of classification. Before entrusting the networks with the classification job, a Confidence Factor (CF) of each network has to be established to have confidence in the final output from the NNN. The CF of each network is obtained by simulating the network output with a large amount of randomly generated data set for that particular node. This is an important aspect of the HDM as the GA module depends on the classification ability of NNN. As it was described earlier, the nodes, particularly the TERMINAL nodes, are not constrained to give one fault class, but can suggest a set of fault classes to the

Table 1 Description of nodes

Node	Designation	Function
LIN1	SFCF	Classifies Sensor fault and Component Fault
L2N1	CMPF	Classifies between Single or Multiple Component Faults
L2N2	SFD	Identifies Sensor Fault
L3N1	SCMPF	Identifies fault classes in case of single component fault
L3N2	MCMF	Classifies multiple fault component faults into subgroups
L4N1	GROUP-1	Classifies the faults with compressors into LP and HP groups
L4N2	GROUP-2	Classifies the faults with turbines into subgroups
L4N3	GROUP-3	Classifies the faults with ICL and RCR into subgroups
L5N1	LPC-GP	Identifies fault classes with LPC as a common component
L5N2	HPC-GP	Identifies fault classes with HPC as a common component
L5N3	HPT-GP	Identifies fault classes with HPT as a common component
L5N4	LPT-GP	Identifies fault classes with LPT as a common component
L5N5	FPT-GP	Identifies fault classes with FPT as a common component
L5N6	ICL-GP	Identifies fault classes with ICL as a common component
L5N7	RCR-GP	Identifies fault classes with RCR as a common component

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

Table 2 Configuration of nodes

Network	Type	Configuration	FCs-Involved in training
LIN1	MLP	10-30-30-2	FC-1: FC-28
L2N1	MLP	10-30-30-2	FC-1: FC-28
L2N2	AANN	10-30-4-30-10	MEASUREMENTS
L3N1	MLP	10-25-25-3	FC-1: FC-7
L3N2	MLP	10-30-30-2	FC-8: FC-28
L4N1	MLP	10-25-25-2	FC-8:FC-18
L4N2	MLP	10-25-25-2	FC-23:FC28
L4N3	MLP	10-25-25-2	FC-19:FC22
L5N1	MLP	10-25-25-3	FC-8:FC-13
L5N2	MLP	10-25-25-3	FC-14:FC-18,FC-8
L5N3	MLP	10-25-25-3	FC-23:FC-25,FC-19,FC-15,FC-10
L5N4	MLP	10-25-25-3	FC-26,FC-27,FC-23,FC-20,FC-16,FC-11
L5N5	MLP	10-25-25-3	FC-28,FC-26,FC-24,FC-21,FC-17,FC-12
L5N6	MLP	10-25-25-3	FC-19,FC-20,FC-21,FC-22,FC-14,FC-9
LFN7	MLP	10-25-25-3	FC-28,FC-27,FC-25,FC-22,FC-18,FC-13

GA module. This feature has been provided to reduce the probability of a wrong fault class being given to the GA module. While classifying the fault classes the CF of that node plays an important role in the number of fault classes being suggested. The higher the CF the lesser will be the number of fault classes suggested, and also more confidence in the fault classes suggested.

High CF is particularly necessary in the BRANCH nodes as the decision made at the branch level is crucial for the progress of the solution in the correct direction. It is also possible that under some extreme conditions the network makes an incorrect classification in the beginning, which can lead to input data being passed through the wrong branches.

To avoid such a situation, different network topologies can be used for classification. Different topologies would mean longer training time. However, once the networks are trained for a particular operating point, the classification is instantaneous. The confidence factors shown in Table 3 have been obtained by subjecting the nodes to a large amount of test data. The various decisions made at the BRANCH nodes and TERMINAL nodes are dependent on these CF values. The CFs for classification of individual nodes and the layer are obtained and is indicative of the confidence one can have in the classification ability of the nodes

Discussion of Results From HDM

The process starts with a node classifying the input data as having a component fault or a sensor fault. If it is identified as a sensor fault, the data is forwarded to an Auto-Associative Neural Network (AANN) for isolation of faulty sensor(s) as well as assessment of the fault magnitudes. On the other hand, if a pattern is identified as a component fault by the node, it is then passed to the next node which classifies it as a single component or multiple component faults. If it is identified as a single component fault then the input data is passed onto to a node which further classifies it into the appropriate category (fault class). If it is classified as a multiple components fault, then it is forwarded to nodes for classification to appropriate subgroups and finally classified into fault classes. A brief description of the constituent nodes is presented:

Node L1N1. The test measurements are introduced to this node which is trained to accomplish an important task of classifying the data into: Component fault or sensor. The node was subjected to a series of rigorous tests with randomly simulated test patterns as follows:

Table 3 Confidence ratings of nodes in the NNN

NODE	CLASSIFICATION OF CATEGORIES (%)							CF(%)
L1N1	SF		CF					99.36
	99.45		99.27					
L2N1	SCMPF		MCMPF					94.54
	96.23		92.86					
L3N1	FC-1	FC-2	FC-3	FC-4	FC-5	FC-6	FC-7	99.74
	99.90	99.97	99.88	99.54	99.92	99.71	99.25	
L3N2	GROUP-1		GROUP-2		GROUP-3			98.87
	99.83		99.52		97.12			
L4N1	LPC-GP		HPC-GP					99.80
	100.00		99.60					
L4N2	HPT-GP		LPT-GP		FPT-GP			99.80
	98.44		98.87		99.26			
L4N3	ICL-GP		RCR-GP					98.95
	99.79		98.92					
L5N1	FC-8	FC-9	FC-10	FC-11	FC-12	FC-13		98.62
	98.50	99.17	97.88	98.54	98.92	98.71		
L5N2	FC-14	FC-15	FC-16	FC-17	FC-18	FC-8		98.47
	97.62	99.23	98.38	97.94	99.12	98.54		
L5N3	FC-23	FC-24	FC-25	FC-19	FC-15	FC-10		98.67
	98.49	99.56	98.67	98.89	97.84	98.59		
L5N4	FC-26	FC-27	FC-23	FC-20	FC-16	FC-11		98.34
	97.68	98.64	99.33	98.21	98.72	97.48		
L5N5	FC-28	FC-26	FC-24	FC-21	FC-17	FC-12		98.01
	97.29	99.09	97.78	98.67	97.37	97.88		
L5N6	FC-19	FC-20	FC-21	FC-22	FC-14	FC-9		98.43
	97.56	99.27	98.59	98.74	97.92	98.51		
L5N7	FC-28	FC-27	FC-25	FC-22	FC-18	FC-13		98.03
	97.34	99.21	96.73	98.67	97.45	98.78		

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

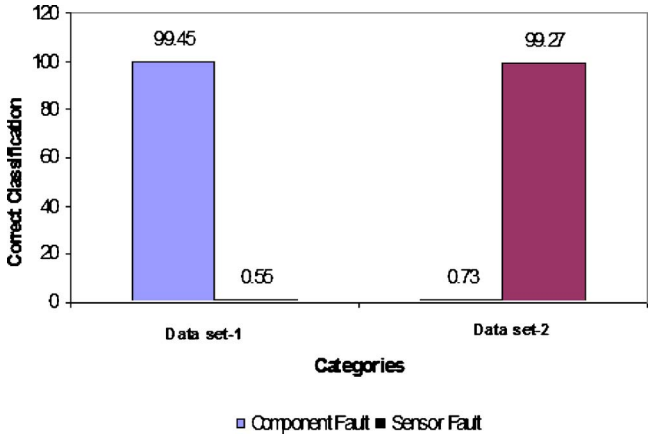

Fig. 5 Results from L1N1 classification

- data with only sensor faults
- data with only component faults
- data with component and sensor faults (not occurring simultaneously)

The break down of the classification by L1N1 is shown in Fig. 5. Data set-1 has patterns only from component faults and Data set-2 has patterns only from sensor faults.

The accuracy achieved by the network was over 99%. Investigation of the misclassified data revealed that some input data with low levels of sensor faults were classified as having component faults or input data with low levels of component faults were classified as having sensor faults. The levels of faults involved were very small and, therefore, the patterns were indistinguishable. For practical sense, such faults would not make a significant impact on the performance of the engine or endanger the safe operation of the engine. It should be noted that the data from 28 fault classes are involved and, therefore, the network is subjected to a large amount of data and needs good generalization capability.

Node-L2N1. This node is required to classify the input data as having a single component fault or multiple component faults (restricted to a maximum of two components simultaneously faulty). The node was subjected to a series of tests with randomly generated fault data. The classification accuracy or the overall confidence factor achieved was 94%. The individual break down of classification is shown in Fig. 6. Data set-1 consists of only single component faults and data set-2 consists of only multiple component faults.

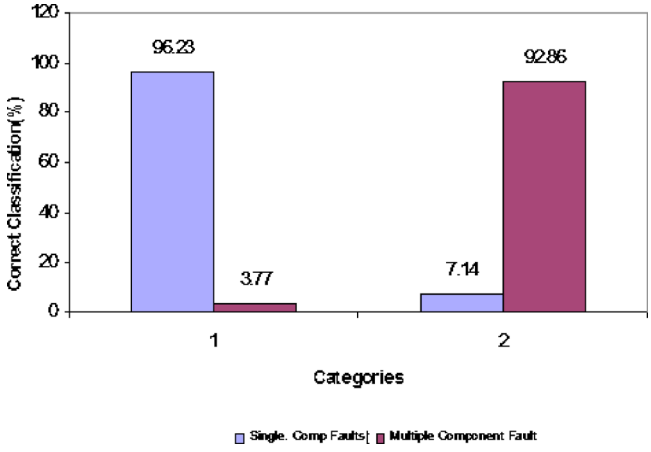

Fig. 6 Results from L2N1 classification

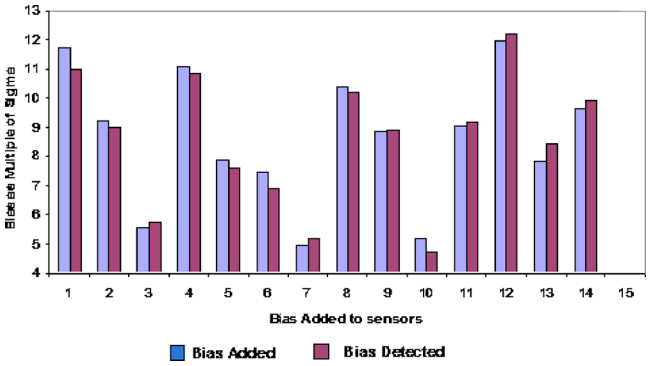

Fig. 7 Results from L2N2 node (Sensor bias detection)

The network achieved low accuracy due to the large variety of data involved in the training. The training data involved patterns from 28 fault classes, which implies various combinations of components and different fault levels. Therefore, the network generalization capability was poor. Many cases of overlaps in classification among the two categories were found. Further investigation revealed that single component fault with high fault levels were classified as multiple component faults and multiple component faults with very low fault levels were classified as single component faults. However, the misclassification does not seriously affect the diagnostics process, as closer examination showed that a single component fault misclassified as a multiple component fault normally detects the fault class which has the same component as the actual faulty component (single component fault) and "smears" as small amount of the fault to the other components. When multiple component faults are classified as single component faults, it is normally observed that one of the components has a low level fault and the other, which has a high level fault, can be identified by the nest level.

Node-L3N1. This is a TERMINAL node and is required to provide the fault classes for further examination by the GA module. This node exhibits a high level of classification accuracy as the patterns produced by the faults in the components are distinguishable by the network. The task of classification can be carried out by a network smaller than the ones used for the above two networks, i.e., L1N1 and L2N1. In general, if the confidence rating of the network is very high, it would identify only one fault class to be examined by the GA module. However, in order to make the NNN robust, a simple logic is incorporated, which considers the individual fault class classification levels before deciding the fault class(es) to be examined.

Node L2N2 (Sensor Fault Diagnosis). Once the L1N1 node identifies the data as having sensor fault, the input data is forwarded to the node identifying the sensor faults. The Auto-Associative Neural Network (AANN) is found to be most appropriate for the sensor fault diagnostics. The AANN basically consists of several layers of nodes which form a symmetrical structure with a bottleneck. The number and configuration of the nodes on either side of the bottleneck layer are identical. Several test were carried out to validate the performance of the node in isolating the faulty sensors. Bias ranging from 4σ to 15σ for respective instruments were used to generate the faulty sensor data. A sample of a bias implanted in the HPC (Exit) pressure sensor is shown in Fig. 7. The bias levels have been correctly identified by the network in all the cases. The small deviations in the identified values are mainly due to the measurement noise. It was observed that, small levels of bias which are close to noise levels are more difficult to isolate.

The hybrid model attempts to combine the strengths of two different techniques to create a more robust diagnostics model.

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

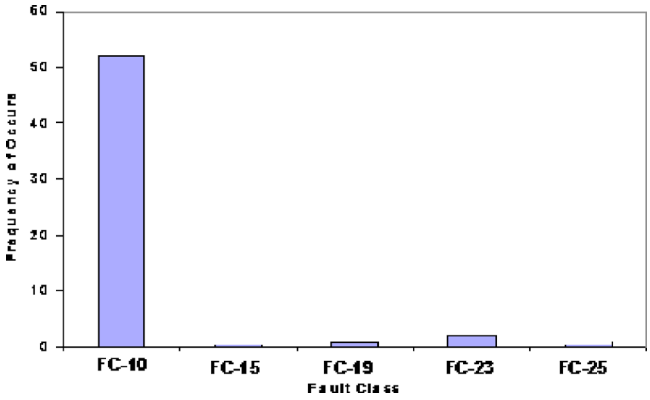

Fig. 8 Results from multiple runs of HDM

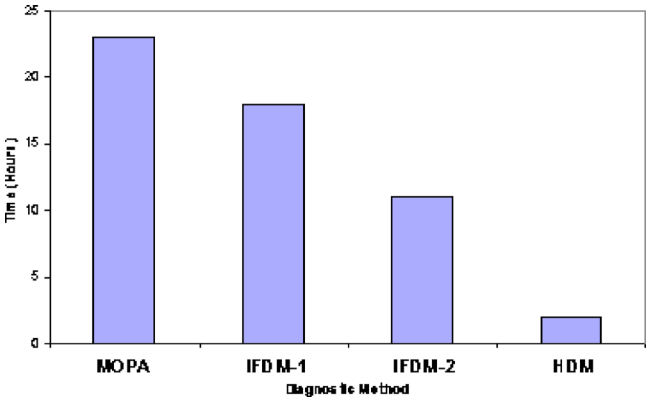

Fig. 10 Comparison of run times for different schemes

The ANNs are known to have good classification properties and also the ability to represent a complex function which is otherwise difficult to solve analytically. The hybrid system developed has reduced the total run time of the fault diagnostics model significantly. The reduction in the run time gives the opportunity for the diagnostics model to be run several times to ascertain the fault identified.

The long run times of the initial GA based diagnostics model was a discouraging factor in the implementation of the model. However, the HDM reduces the overall runtime which makes is possible to run the model several times to increase the confidence in the output. The FCC could suggest more than one fault class (depending on the CF of TERMINAL node) to be explored by the GA model. In a test case, the FCC suggested five fault classes to be explored (instead of 28). The GA diagnostics model uses probabilistic transition rules and, therefore, it is also possible that occasionally the model detects the faults in other fault classes erroneously, especially the competing fault classes, this is attributable mainly to the randomness in the application of the GA operators coupled with the measurement noise and model inaccuracies. The GA based diagnostics model was ran 60 times and it has identified the FC-10 to have the faulty components shown in Fig. 8.

A distribution of the faults quantified by the GA based diagnostics model is shown in Fig. 9. It can be seen that the average of the deviations are close to the implanted values and also the extreme values are within the acceptable limits. It is more important to identify the faulty component more accurately than the actual quantification of the fault as on most occasions the remedial action might be the same. E.g., if the actual fault had a 2% drop in compressor efficiency and the diagnostics model detected 2.5%, the corrective action remains unchanged.

Figure 10 shows a comparison of the overall runtimes for various schemes taken for the ICR WR21 engine. The MOPA based diagnostics model developed initially took ~22 h for convergence. The IFDM1 (with 25% RSM) took about 18 h and the IFDM2 (with 50% RSM) took close to 12 h. The HDM took about 2 h to converge to a solution. It is ten times faster than the initial model developed. In fact this time can further be reduced if the terminal node can suggest only one fault class to be searched (in the present case it has suggested five Fault classes).

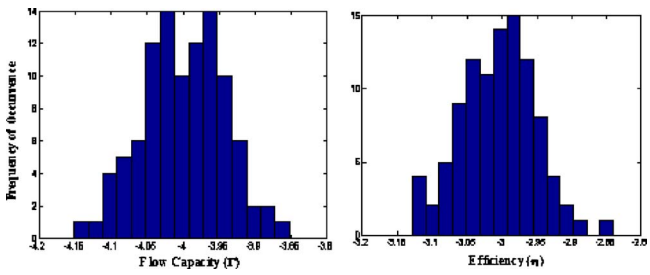

Fig. 9 Distribution of fault quantification

Comparison with Other Methods

In the present study and related studies in the past, the GA based engine diagnostics technique has been tested quite extensively for various combinations of component faults and instrumentation faults while varying the level of noise. It has been observed that the technique has certain advantages when compared with other techniques. One of the main advantages is that the technique is robust and consistent in the face of instrument noise and also the implementation of the Sensor Fault Detection and Accommodation is simple. The GA based method works with a group of solutions and once a reasonably close solution is identified, further refinement is possible. The GA based technique has the ability to detect very small deviations in component performance parameters due to its solution refinement capability. One major limitation in the implementation of the technique, is the long time taken for the algorithm to converge. While the initial GA based methods [9,10] took several hours to converge, the hybrid method developed by the authors has been successful in reducing the run times significantly (~10–12 times). However, the run time still remains large when compared with other techniques. A comparison of four different diagnostics method was conducted using the WR21 performance code and a sample result is presented in Table 4. It is evident that the GA based method scores over the other methods in identifying the fault levels which are very small.

A comparison of the overall fault identification capability was obtained by introducing the same set of one hundred faults to the four different diagnostics techniques. It can be observed from Fig. 11 that the GA based technique has been able to successfully identify the faults in almost 95% of the cases.

Table 4 Comparison of different diagnostics techniques

	Component-LPC		Component-HPT		
Method	η	Γ	η	Γ	RMS Error
Fuzzy Logic	−1.0	−2.6	−0.2	1.3	0.83
Gas Path Analysis	−0.82	−2.1	−1.1	1.2	1.29
Neural Network	−0.95	−2.87	−0.8	1.5	0.45
Genetic Algorithm	−1.12	−3.11	−0.56	1.88	**0.16**

Fault Implanted. Percentage Deviation in: LPC: η: −1.2% Γ: −3.0% Percentage Deviation in: HPT: η: −0.6% Γ: 1.8%

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use

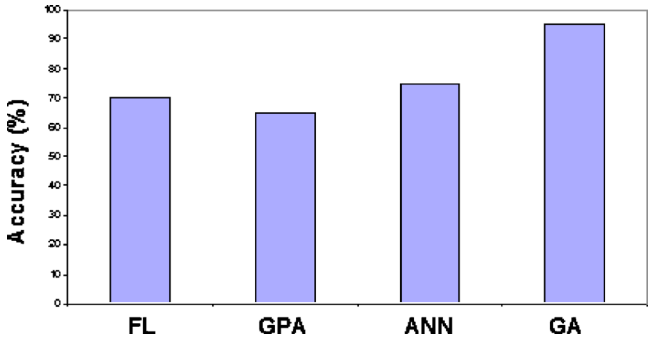

Fig. 11 Comparison of accuracy of fault detection

Large deviations in component performance parameters have been identified by all the techniques correctly with comparable RMS errors. However, the GA based technique has been able to identify more faults of small magnitudes and, therefore, has been shown to have identified a higher percentage of faults for the given sample data.

Conclusion

In this paper a novel method of combining the ANN and GAs have been examined. Though the diagnostic model based purely on the GA had merit but was reported to have long run times and, therefore, necessitated modification in order to be implemented for engine fault diagnostics. Despite research in the various methods for engine fault diagnostics, there is still no "magic formula" which can effectively address all issues. One way to approach the problem is to try and offset the limitations of one technique with the strength of the other. The hybrid model developed, to an extent, has attempted to bridge this gap. The results obtained were encouraging and further investigation is being carried out to make it more efficient.

Nomenclature

Γ = flow capacity
η = efficiency
Δ = small change
J = objective function
σ = standard deviation for noise

Abbreviations

RBF = radial basis function
AANN = auto associative neural network
ANN = artificial neural network
CF = confidence factor
FC = fault class
FFBPNN = feed forward back propagation neural network
GA = genetic algorithm
GPA = gas path analysis
HDM = hybrid diagnostics model
HPC = high pressure compressor
ICR = inter-cooled recuperated
NNN = nested neural network
FCC = fault class classifier
RSM = response surface method
MLP = multi-layer perceptron

References

[1] Doel, D. L., 1994, "TEMPER: Gas-Path Analysis Tool for Commercial Jet Engines," ASME J. Eng. Gas Turbines Power, **116**(1), pp. 82–89.
[2] Doel, D., 2002, "Interpretation of Weighted-Least-Squares Gas Path Analysis Results," ASME Turbo Expo 2002, June 3–5, Amsterdam.
[3] Deol, D. L., 1993, "Gas Path Analysis—Problems and Solutions," Symposium of Aircraft Integrated Monitoring Systems, Sept. 21–23, Bonn, Germany.
[4] Mathioudakis, K., Kamboukos, P., Stamatis, S., 2002, "Turbofan Performance Deterioration Tracking Using Non-Linear Models and Optimization Techniques," ASME TE-2002, June 3–5, Amsterdam, The Netherlands.
[5] Kamboukos, P., and Mathioudakis, K., 2002, "Comparison of Linear and Non-Linear Gas Turbine Performance Diagnostics," ASME TE-2003, June 16–19, Atlanta, GA.
[6] Kobayashi, T., and Simon, D. L., 2003, "Application of a Bank of Kalman Filters for Aircraft Engine Fault Diagnostics," ASME TE-2003, June 16–19, Atlanta, GA.
[7] Zedda, M., and Singh, R., 1998, "Fault Diagnosis of a Turbofan Engine Using Neural Networks: A Quantitative Approach," Paper No. AIAA 98-3062.
[8] Zedda, M., and Singh, R., 1999, "Gas Turbine Engine and Sensor Fault Diagnosis Using Optimisation Techniques," Paper No. AIAA 99–2530.
[9] Gulati, A., 2002, "An Optimization Tool for Gas Turbine Engine Diagnostics," Ph.D. thesis, School of Engineering, Cranfield University, UK.
[10] Sampath, S., Gulati, A., and Singh, R., 2002, "Fault Diagnostics Using Genetic Algorithm for Advanced Cycle Gas Turbine," ASME TE-2002, June 3–5, Amsterdam, The Netherlands.
[11] Montgomery, D. C., 1984, "Design and Analysis of Computer Experiments," 2nd ed., Wiley, New York.
[12] Myers, R. H., 1999, "Response Surface Methodology Current Status and Future Direction," J. Quality Technol., **31**(1), pp. 30–74.
[13] Bethke, A. D., 1981, "Genetic Algorithm as Function Optimizers," Doctoral dissertation, University of Michigan.
[14] Bosworth, J., Foo, N., and Zeigler, B. P., 1972, "Comparison of Genetic Algorithms With Conjugate Gradient Method," National Aeronautics and Space Administration, Washington, DC, Report No. CR-2093.
[15] Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.
[16] Ogaji, S. O. T., and Singh, R., 2002, "Gas Path Fault Diagnosis Framework for a 3-Shaft Gas Turbine," IMechE Journal of Power and Energy, **217**, A3.

Downloaded From: https://gasturbinespower.asmedigitalcollection.asme.org on 07/01/2019 Terms of Use: http://www.asme.org/about-asme/terms-of-use