Space Shuttle Main Propulsion System Anomaly Detection: *A Case Study*

Bryan L. Matthews, Ashok N. Srivastava, David Iverson, Bob Beil & Bill Lane
National Air & Space Administration

ABSTRACT

The space shuttle main engine (SSME) is part of the Main Propulsion System (MPS) which is an extremely complex system containing several sub-systems and components, each of which must work precisely in order to achieve a successful mission. A critical component under study is the flow control valve (FCV) which controls the pressure of the gaseous hydrogen between the SSME and the external fuel tank. The FCV has received added attention since a Space Shuttle Mission in November 2008, where it was discovered during the mission that an anomaly had occurred in one of the three FCV's. Subsequent inspection revealed that one FCV cracked during ascent. This type of fault is of high criticality because it can lead to potentially catastrophic gaseous hydrogen leakage. A supervised learning method known as Virtual Sensors (VS), and an unsupervised learning method known as the Inductive Monitoring System (IMS) were used to detect anomalies related to the FCV in the MPS. Both algorithms identify the time of the anomaly in a multi-dimensional time series of temperatures, pressures, and control signals related to the FCV. This discovery corroborates the results of the inspection and also reveals the time at which the anomaly likely occurred. The methods were applied to data obtained from the March 2009 launch of Space Shuttle Discovery to determine whether an anomaly occurred in the same sub-system. According to our models, the FCV sub-system showed nominal behavior during ascent.

INTRODUCTION

Anomaly detection in complex engineered systems is particularly challenging because, in general, these systems can operate as feedback-control systems and can behave in a highly non-linear fashion. The feedback-control systems pose a particular challenge because the control system is designed to compensate for a disturbance in the system. This compensation can mask the underlying disturbance unless an appropriate set of sensors is in place. For our purposes herein, we assume that the data generating process under study has a set of sensors that are appropriately placed to discover useful anomalies.

In the case of the Main Propulsion System of the Space Shuttle, we consider the data generating process to be those parts of the engine that are related to the flow control valve. After discussion with several experts, we were provided with thirteen parameters related to pressure, temperature, throttle, and the control signals to each FCV for each of the three SSMEs for over 80 flights of the Space Shuttle. The specific parameters provided include the SSME Main Combustion Chamber Pressure, the SSME Gaseous Hydrogen Outlet Pressure, and the SSME Gaseous Hydrogen Outlet Temperature. We were also given the Pressurization Disc Pressure and included in our analysis. Thus. at each time step we have a 10-dimensional multivariate time series for analysis [1]. The input parameters that were provided are the binary commands to the flow control valves on each engine and the outputs are continuous measurements.

Author's Current Address:
B.L. Matthews, A.N. Srivastavay, D. Iversonz, B. Beilx and B. Lane, SGT Inc., NASA Ames Research Center Moffett Field, CA 94035, USA.

Manuscript received by January 12, 2011. Review was handled by M. DeSanctis.

[1] For each of the three engines, we had two pressure and one temperature sensors upstream of the FCVs plus one pressure sensor downstream of the manifold FCVs external tank interface.

There has been considerable research in the area of systems health management on the Space Shuttle Main Engine in the area of optical plume anomaly detection [1–3], anomaly detection [4], and prognostics [5]. There is also a vast literature on monitoring and condition-based maintenance of aircraft and rotorcraft engines which is reviewed in [6]. This demonstrates the ability to detect anomalies in the MPS using a purely data-driven approach.

This is organized as follows. We begin with a discussion of the propulsion sub-system and the anomaly which we are studying followed by a discussion of the rapid and resilient analysis methodology we followed. The subsequent section describes the anomaly detection algorithms we used along with an extensive compilation of results on real data sets. We conclude with a discussion of next steps in our final section.

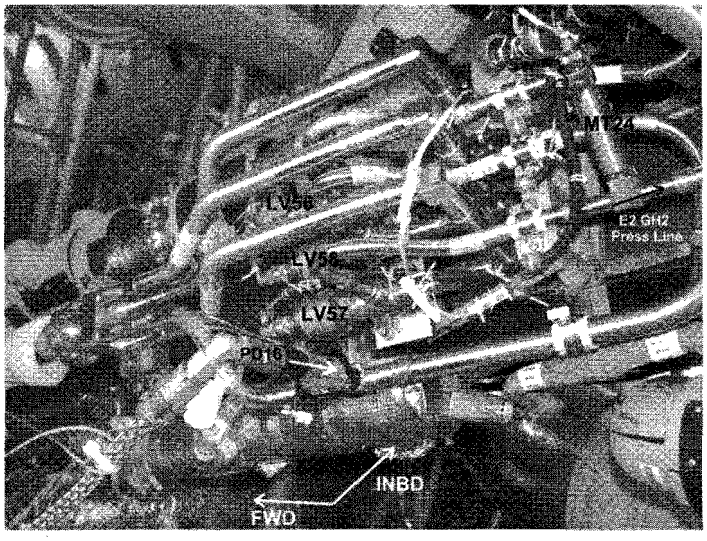

Fig. 1. This shows the GH_2 pressurization lines on the SSME and the direction of flow of the hydrogen. The flow control valves are located at positions LV-56, LV-57, and LV-58 in the system

PROPULSION SUB-SYSTEM AND THE ANOMALY

The flow control valve (FCV) is a component in the main propulsion system of the space shuttle. The main propulsion system consists of three Space Shuttle Main Engines (SSME), the orbiter MPS propellant management system (including the three Flow Control Valves) followed by the external tank, in order of repressurant flow. The MPS also contains the helium sub-system, four ascent thrust vector control units, and six SSME hydraulic servo-actuators [7]. The FCVs under study herein control the flow of gaseous hydrogen (GH_2) from the main engines to the external fuel tank [8]. The FCVs are like binary switches, having an on and an off (high flow/low flow) positions to regulate the gaseous hydrogen. Typically, one FCV cycles about 15 times during normal operations. The position of the FCV in the larger MPS system is shown in Figure 1. These valves regulate the flow of the GH_2 to the External Fuel Tank maintaining the liquid H_2 tank ullage pressure in a prescribed pressure band.

As discussed in a NASA document on the matter [8], *"During space shuttle Endeavour's STS-126 mission in November 2008, flight controllers identified that GH_2 was flowing from one of the shuttles engines at a higher than normal rate. To compensate, the other two gaseous hydrogen flow control valves reduced the amount of their flow and there were no issues during launch. After landing, the main propulsion system was inspected and engineers discovered the GH_2 flow control valve poppet on the suspect line was cracked and a small piece was missing. The poppet on the valve acts like a pop-up on a sprinkler to let the GH_2 flow."* Figure 2 shows the cracked Flow Control Valve with a section of the poppet shown liberated. This crack allowed additional GH_2 through the valve and could have led to a potential safety issue because the broken material can penetrate the fuel lines or cause other issues downstream from the FCV. The purpose of our research was to determine if automated algorithms could detect the failure and potentially identify a precursor to the failure on STS-126. In addition to that flight, we tested our methods across a large number of previous missions to see if we could detect similar anomalies in the FCV or related sub-systems.

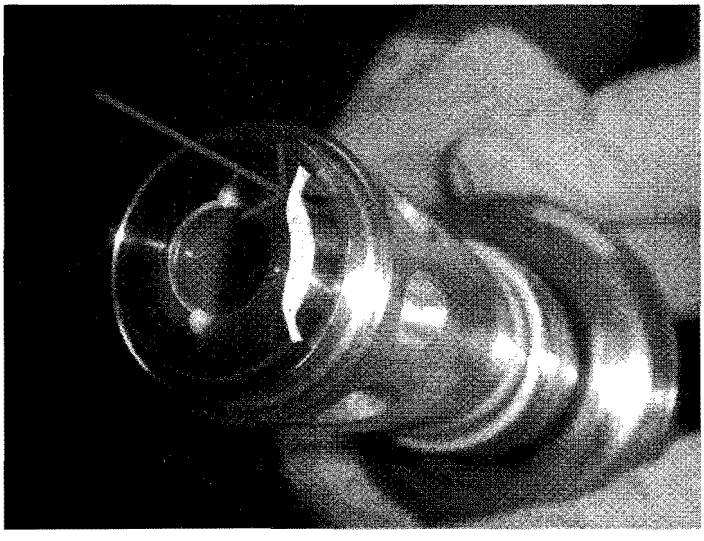

Fig. 2. The arrow indicates where the break occurred in the poppet of the Flow Control Valve on one of the SSMEs during the mission STS-126 of Space Shuttle Endeavor. The break resulted in increased flow of gaseous hydrogen but did not result in an issue during launch

EXTREME PROGRAMMING MEETS DATA MINING

The data mining analysis needed to be performed quickly and efficiently since the results would be used in support of a Flight Readiness Review of the Space Shuttle Mission STS-119. We thus needed to adopt a methodology that would

minimize the errors that can arise during a rapid analysis environment while maximizing our ability to cross-check and differentiate the discovery of true anomalies from errors in data preparation, other artifacts, and sensor issues.

Various software engineering methodologies have been developed to write robust code under a variety of circumstances, depending on the domain and the criticality of the system. One key methodology, known as Agile Programming, emphasizes the development of software with rapid prototyping and an emphasis on simplicity of design. A particular methodology in the area of Agile Programming, is known as Extreme Programming [9]. We adapted some key ideas from Extreme Programming to help us rapidly generate meaningful results for the Shuttle Program Office which we outline below.

Rather than giving an in-depth overview of the Extreme Programming methodology, we will give a summary of key points in the methodology and discuss how they were adapted to fit our needs. It is our hope that other teams interested in rapid design and deployment of data mining techniques will find this discussion beneficial.

Extreme Programming can be divided in to the following four rules and practices [10]:

- **Planning**
  During the planning stage of the activity, we discussed and developed several strategies for manipulating the data sets so that they could be represented as multivariate time series with evenly spaced sample points. Because the data sets from the SSME that we were given are sampled at non-uniform time intervals, they needed to be standardized into a single time frame for analysis. Two individuals in the team took responsibility to develop code independently to enable rapid verification and validation of results in the testing phase of the methodology. In this phase, we also discussed the algorithms that would be used in the analysis and their suitability for this type of data. A key point of interest was the ability of the algorithms to perform anomaly detection for streams of both continuous and discrete signals. We also discussed a plan for iterating our work so that we would each be able to build upon another team's success once appropriate testing was completed.

- **Designing**
  As in Extreme Programming, the design phase of our project focused significantly on simplicity, both in terms of algorithm and code complexity, but also in terms of the interpretability of results. Some methods, such as One-Class Support Vector Machines [11] have been used in a variety of settings, but it can be difficult to interpret the reasons why an anomaly is tagged as such. For flight-critical systems, this turns out to be as important as the discovery of the anomaly itself, since it can indicate a potential root-cause of the problem, thereby enabling a fault diagnosis and prognostic capability. We also developed spike solutions as described in [10] in order to study the data processing techniques that would be appropriate for the non-uniform time sampled data. After several designs and subsequent unit tests, we concluded that the best method to address the sampling issue would be to use a "sample-and-hold" interpolation method in which a given value in a time series is held until a new value is observed. This decision was based on the engineering design of the SSME, in which all signals are assumed to be constant unless a change is recorded. We also refrained from adding functionality to our code too early to reduce complexity and improve the success of our unit tests.

- **Coding**
  Our anomaly detection algorithms have been tested by us and others over many years [4]. However, the appropriate parameter settings and the methods used for data pre-processing were critical for the success of this project. Thus, we communicated with the Shuttle Program Office and other representatives regularly in order to ensure that our work was consistent with their needs. As in Extreme Programming, we coded in pairs, although the partners did not sit together. Instead, code was developed simultaneously along with unit testing code in order to ensure that our results were consistent and reproducible.

- **Testing**
  A key component of the Extreme Programming methodology is the testing phase. We adhered to these testing standards rigorously. All code was unit tested and passed unit tests before release to other team members. When we uncovered a bug in the software, we developed new tests and checked our work against those tests before releasing to other team members. This procedure helped us isolate errors in the code and also helped us understand and interpret the data we were given. The data sets covered nearly 100 flights of the 5 orbiters over a 20 year period with about 20 parameters in all per flight. This massive volume of highly complex engineering data necessitated a strong unit testing approach. In fact, it was this need that

Table 1. Sample IMS Data Vector

ME-1 MCC Pres (Avg)	ME-2 MCC Pres (Avg)	ME-3 MCC Pres (Avg)	MPS E1 GH_2 Pres Outlet Pres	MPS E2 GH_2 Pres Outlet Pres	MPS E3 GH_2 Pres Outlet Pres	MPS GH_2 Pres Disc Pres
2660	2760	2750	2260	2440	2400	210

drove us to adopt the Extreme Programming methodology which proved to suit our needs well.

The methodology described above stands in sharp contrast to the waterfall development process given by Royce [12]. This development process moves sequentially from requirements to design, to implementation, to verification, and finally, maintenance. It is clear that this process can be valuable for much larger activities, however, it is not suited for the iterative nature of the application of data mining techniques to real-world problems, particularly in situations where rapid results are required with high confidence. Other data mining methodologies, such as CRISP-DM [13] allow for the iterative process of data mining but do not focus as much on unit-testing and the design and coding process. The methodology that we discuss herein does not depend on the nature of the algorithms used assuming that there is an iterative process for deployment and testing. As such, the multiple anomaly detection algorithms given below were chosen to verify and validate the results in parallel and are a set of algorithms that we deemed appropriate for this task. Other algorithms could be easily deployed in this methodology such as one class support vector machines [11].

The system architecture that we developed for this application consists of a data layer followed by anomaly detection algorithms. These algorithms were run in parallel on many of the tests we performed. The extreme programming framework discussed above was invaluable in ensuring that the algorithms and results could be easily verified and validated.

ANOMALY DETECTION ALGORITHMS

The main idea discussed herein is to use automated algorithms to identify anomalies in time series that contain both continuous and binary data streams. We approach the problem using two methods that have been established in the literature: the Inductive Monitoring System [14] and Virtual Sensors [15, 16]. Pre-processing steps such as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) were not used due to the fact that the individual parameters become linearly combined into basis vectors, hindering the algorithms' ability to diagnose the parameter(s) that detected the fault. We also tested a third algorithm known as Orca [17] which gave similar results to IMS and do not present those results herein for the sake of brevity. There are numerous other anomaly detection schemes which could be tested on this data. However, the purpose of the study was to validate established methods on this system.

Inductive Monitoring System

The Inductive Monitoring System (IMS) is a distance-based anomaly detection tool that uses an unsupervised data-driven technique called clustering to extract models of nominal system operation from archived data. The basic data structure used for distance-based analysis is a vector of concurrent parameter values seen in Table 1. For the FCV analysis, we formed vectors from a sub-set of the continuous observed variables as specified in Table 1. IMS treats vectors containing N normalized values as points in an *N-dimensional vector space*, using an appropriate metric to calculate distance between these points. The familiar Euclidean distance metric has proven effective in many applications, and was used in the FCV analysis. *Z-Score normalization*, scaling each parameter to zero mean and unit standard deviation, was applied to each FCV data value used in the IMS vectors.

The underlying premise of distance-based anomaly detection is that anomalous data points will fall a significant distance away from typical, nominal data points. The IMS learning process uses a dynamic clustering algorithm to delimit areas of the vector space containing data points collected during normal system operation. These areas, called nominal operating regions, are defined by *N-dimensional* hyper-boxes specifying the upper and lower boundary value for each point contained in the region (see Table 2). IMS considers all data vectors that fall within these hyper-boxes as representative of normal system behavior, thus generalizing from examples of nominal behavior presented in the training data sets. The collection of nominal operating regions representing the behavior of a given system as derived from a training data set is called an IMS knowledge base.

To construct an IMS knowledge base, archived training data collected during normal system operation is formatted into input vectors. Typically, vector parameter values represent a single time slice of time series operations data. This was the case with the FCV analysis. Mean and standard

Table 2. Sample IMS Nominal Operating Region

	ME-1 MCC Pres (Avg)	ME-2 MCC Pres (Avg)	ME-3 MCC Pres (Avg)	MPS E1 GH_2 Pres Outlet Pres	MPS E2 GH_2 Pres Outlet Pres	MPS E3 GH_2 Pres Outlet Pres	MPS GH_2 Pres Disc Pres
High	**2670**	**2762**	**2760**	**2287**	**2448**	**2412**	**217**
Low	**2655**	**2748**	**2747**	**2255**	**2432**	**2397**	**209**

deviation values are calculated for each vector parameter in the training data set to use for *Z-Score normalization*. The input vectors are normalized and fed, in temporal sequence, to the IMS learning algorithm that constructs the nominal operating regions. The IMS learning algorithm is provided in detail on our collaborative website known as dashlink <http://dashlink.arc.nasa.gov>. After processing all of the training data, the learning algorithm outputs a knowledge base containing multiple nominal operating regions which together contain all of the points presented in the training data. The number of nominal operating regions produced will depend on the nature of the training data and the learning parameters used with the IMS routine.

After an IMS knowledge base is constructed, it is used to analyze new data to determine if the new data exhibits behavior similar to system operations during collection of the nominal training data. For each new data vector processed, IMS outputs a scalar distance from nominal score, indicating how far that data vector falls from the closest nominal operating region cluster. A larger distance score indicates a more unusual, or anomalous, input vector as compared to the nominal data used to train IMS. Large distance scores often indicate a system failure or degradation. IMS also produces a measure of how far each individual parameter in the new vector falls from the closest cluster along its particular dimension. This can indicate which parameters have been affected by the system anomaly, and thus provide insight into which system components may be operating incorrectly.

Virtual Sensors

In contrast to Inductive Monitoring System's one-class modeling method described above, Virtual Sensors is a supervised algorithm in which the value of a specific parameter (or set of parameters) is predicted given all data available up to that point in time. In the current application, we do not consider a forecasting model where the Virtual Sensor predicts events at times beyond the current time. Such forecasting methods have been discussed extensively in [18].

Thus, in the Virtual Sensors paradigm, we create a statistical estimator of a specific output variable as a function of other observed variables. We assume that we do not have a physical model that ties the output variable to the observed variables. Thus, we choose a non-parametric model to approximate the function. In the simplest form, the Virtual Sensor delivers a point estimate which can be compared with the observed signal to form an error signal:

> *In the simplest case, this signal is monitored and an alarm is sounded if the error exceeds a pre-defined threshold which can be calculated using an assumed nominal data set. In the examples shown herein for a constant threshold, we computed the standard deviation of the error signal across an entire nominal flight and set the threshold equal to ± 3 standard deviations about zero to fall within a 99% confidence interval [19] (see <http://dashlink.arc.nasa.gov> for more information regarding this method).*

In many data generating processes, such as the SSME, there are different regimes of behavior which have transient characteristics that make adaptive modeling difficult. For example, in the SSME, when the engine throttle changes, there can be a brief period (lasting from 1-10 seconds) of transients, where the system has not yet settled down to a quiescent point. Similarly, at start-up, there are transients that are difficult to model using adaptive methods. Such transients often generate error signals that exceed any fixed threshold specified relative to and entire nominal flight, thereby increasing the false-alarm rate.

In order to address these issues, we approached the problem by calculating multiple models from various samples and use the distribution of the estimates to produce an average estimate as well as an uncertainty level that can be used as a dynamic threshold. This is done by choosing to model the first and second moments of the probability distribution function so that the mean of the distribution gives the point estimate and the variance of the distribution gives the uncertainty in that estimate. We used a two-step approach to building the non-parametric model to obtain an estimate of the uncertainty in the prediction. The first step is to build a set of bagged predictors from a sample of the training data as defined by Leo Breiman [20]. Using the base model of a decision tree, we built an ensemble of N point predictive trees (where N is 100) and estimated the mean and standard deviation for the error using standard formulae.

The second step that is taken is to perform bootstrap sampling again across another sample of training data for the

Table 3. Summary IMS Scores for Endeavor

STS-89	STS-88	STS-99	STS-97	STS-100	STS-108	STS-111	STS-113	STS-118	STS-123	STS-126
0.1514	2.8891	0.0663	0.1504	0.0530	0.0895	0.0293	0.0203	0.0341	0.0210	2.0173

bagged predictors. This approach treats each bagged predictor as a single estimator (rather than an ensemble) and averages over 100 additional bootstrap samples. The final estimate of uncertainty is given by computing the average variance of each model and taking the square root to obtain a standard deviation. This yields an uncertainty measure across bagged predictors, and due to model averaging, is robust to small changes in the data generating process, noise, and other perturbations in the system. Using this approach, we are able to obtain an estimate of model uncertainty for each time. The adaptive threshold is set at 3 times the estimated model uncertainty. When the error signal exceeds this value, an anomaly event is recorded. The three sigma threshold is a standard measure of deviation from the mean assuming that the residuals are independent and identically distributed normal random variables. Using standard probability tables, this corresponds to representing over 99% of the expected

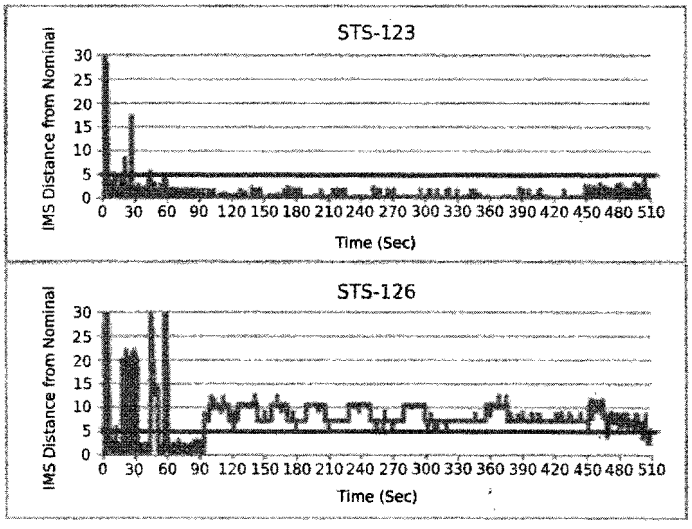

Fig. 3. This shows the output of IMS for STS-123 (top), which is known to be a nominal flight and STS-126 (bottom), which had the failure. The horizontal line at 5 indicates a 3-sigma threshold calculated from STS-123's IMS scores. The scores for both flights show transient effects early in the launch sequence that subsequently drop to a low value once the flight has become stable. However, after approximately 93.6 seconds in STS-126, IMS shows that an anomaly occurred and persisted through the remainder of the flight

residuals. Higher multiples of the empirical standard deviation would increase the width of the detection threshold, thereby reducing the false positive rate but potentially hurting the true positive rate. These thresholds can be chosen based on the application. It is critical that the VS algorithm be trained on a wide variety of nominal data sets in order to obtain a representative estimate of expected variation from the mean. Indeed, this requirement is nearly universal for most data-driven methods.

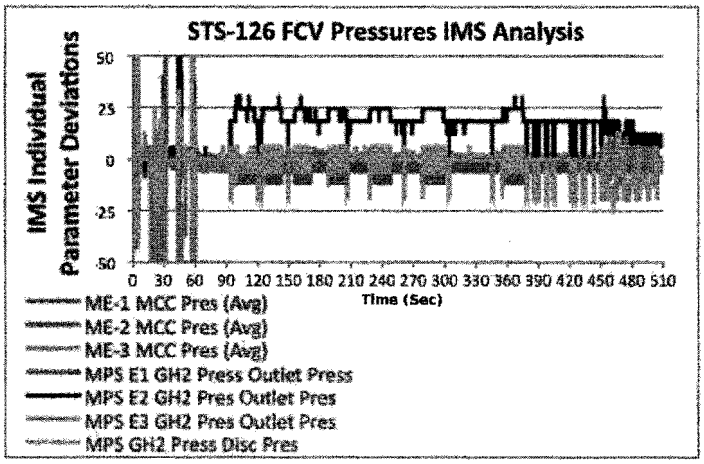

Fig. 4. This shows the contribution of each input variable to the IMS anomaly score. Notice that the MPS E2 GH_2 Outlet Pressure sensor is the main driver of the anomaly score

RESULTS

We present tests of these methods on real data from the Space Shuttle Main Engine for the Inductive Monitoring System and Virtual Sensors with and without adaptive thresholds. These results corroborate and have been corroborated by inspections of the Flow Control Valves in the Space Shuttle Main Engine [8].

Results for the Inductive Monitoring System

Table 4 shows the sensors used along with the training and testing flights that were analyzed using IMS. STS-123 was confirmed by domain experts to exhibit nominal behavior during flight and valve inspection, while STS-126was found to experience the faulty valve. In Figure 3, the IMS scores indicate a few short bursts of anomalous behavior on

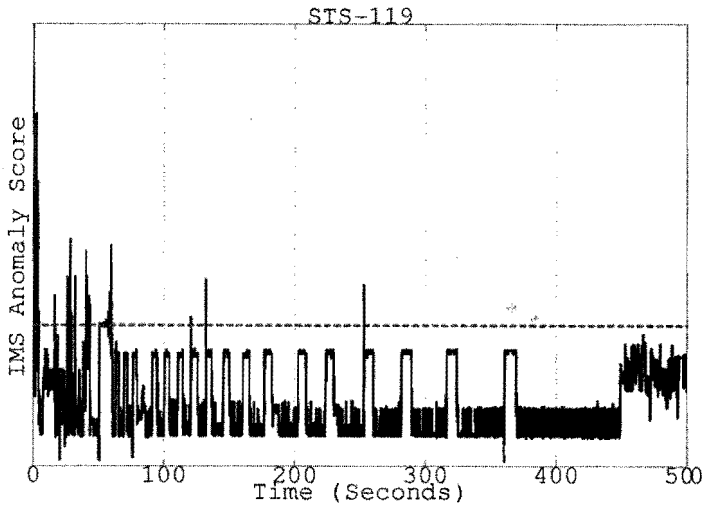

Fig. 5. This shows the IMS scores for STS-119. The dotted line is a threshold calculated from the IMS scores from STS-120. There does not appear to be any anomalous trends outside the early transient periods in the launch

principle contributor to the overall anomaly score. When examining the sensor traces identified by IMS a drop in pressure was observed at 93.6 seconds with no corresponding change in throttle or valve control command. Domain experts believe this behavior is indicative of the valve failure. IMS was also applied to telemetered data from the STS-119. For this analysis previous flights of Discovery were used for algorithm training (see Table 5). Figure 5 shows the IMS scores for STS-119 with a threshold calculated from a holdout flight. As before there are some anomalous trends during the transient phase of flight, however the remainder of the flight behaves nominally with minimal false alarms.

To compare how anomalous the identified faulty flight was compared to other flights with the same valve configuration, IMS was applied to all 11 flights in a hold-one-out train and test approach. In this approach each flight was allowed to cycle as the testing flight while using the IMS model learned from the remaining 10 flights. The mean IMS anomaly score was calculated for each flight to produce a summary IMS score that can be found in Table 3. As expected, STS-126 exhibited a significantly high IMS

Table 4.

Table 4A. This shows the flights used to build and test the models described herein

Training Flights	Testing Flights
STS-89	STS-118
STS-88	STS-123
STS-99	STS-126
STS-97	
STS-100	
STS-108	
STS-111	
STS-113	

Table 4B. The left column shows identified sensors used for each algorithm. The right column denotes which algorithm was used with the corresponding sensor. Virtual Sensors used the sensor noted in bold* as the target. The Inductive Monitoring System does not have an output target, so all items identified as IMS were included as inputs

Sensors	Algorithm
ME-1 MCC Pressure (Avg)	IMS, VS
ME-2 MCC Pressure (Avg)	IMS, VS
ME-3 MCC Pressure (Avg)	IMS, VS
MPS GH_2 Pressurization Disc Press	IMS, VS
MPS E1 GH_2 Press Outlet Press	IMS
MPS E2 GH_2 Press Outlet Press*	IMS, VS
MPS E3 GH_2 Press Outlet Press	IMS
MPS E1 GH_2 Press Outlet Temp	VS
MPS E2 GH_2 Press Outlet Temp	VS
MPS E3 GH_2 Press Outlet Temp	VS
MPS GH_2 Press FCV 1 (LV56) Cl Pwr	VS
MPS GH_2 Press FCV 2 (LV57) Cl Pwr	VS
MPS GH_2 Press FCV 3 (LV58) Cl Pwr	VS
SSME Throttle	VS

STS-123 during engine transient states due to throttle changes while the vehicle is traveling through max Q (approximately 25 to 56 seconds). After the transients subside the scores stabilize for the rest of the flight indicating normal behavior. Figure 3 also shows similar anomalous behavior on STS-126 during the transient period, although after stabilizing the scores rise again around 93.6 seconds and continue to hold high for the remainder of the flight. Upon further examination of the contributing sensors scores in Figure 4 the MPS E2 GH_2 Outlet Pressure sensor appears to exhibit the same anomalous trend as well as being the

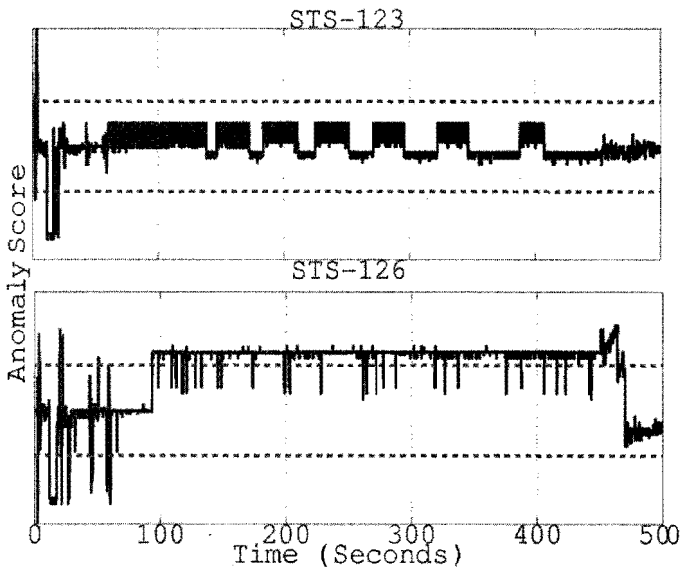

Fig. 6. The output of the Virtual Sensors prediction error for STS-123 in the top panel and STS-126 is shown the lower panel. STS-123 shows nominal behavior for the entire mission except for a transient early in the flight. STS-126 shows transient behavior in the early part of the flight but then at 93.6 seconds, it shows a persistent anomaly for the remainder of the flight. The results shown are for a fixed anomaly threshold

summary score, however, STS-88 was found to have a higher anomaly score. Upon further examination of STS-88 the MPS E3 GH_2 Outlet Pressure sensor was operating approximately 250 PSIA above the remaining flights normal pressure ranges. These findings were presented to the domain experts at KSC who identified this anomaly as one where SSME 3 was running under a different engine configuration and was expected to be running at a higher chamber pressure. Even though this particular anomaly did not turn out to be significant, IMS demonstrated the ability to flag unusual behavior while also verifying nominal behavior with lower summary scores.

Virtual Sensors

Table 4 shows the sensors used for both input and target as well as the training and testing flights that were analyzed with VS. The same set of training and testing flights were used as in the IMS analysis, however additional temperature, valve commands, and throttle sensors were used as input. Figure 6 shows the VS scores with the fixed 3-sigma threshold calculated from STS-123 and applied to STS-123 and STS-126. Both flights exhibit threshold exceedances during transient periods, however they reach a stable state after 60 seconds. While STS-123 remains stable for the remainder of the flight, STS-126 reveals a persistent anomaly beginning at 93.6 seconds and continuing on to the end-of-flight. Figure 7 demonstrates the adaptive threshold computed from the bagged models on STS-123. The transient

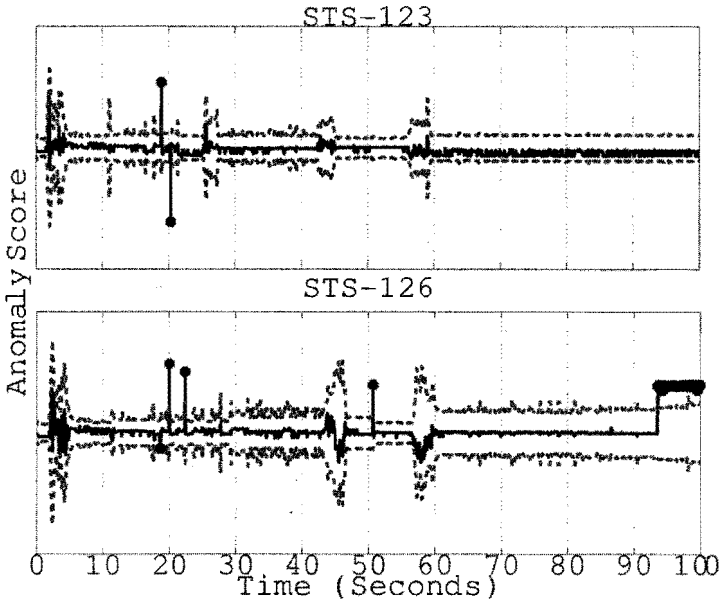

Fig. 7. The Virtual Sensors algorithm with the adaptive threshold for the initial 100 seconds of the flights for STS-123 (top) and STS-126 (bottom). Threshold crossings are indicated by dark circles. The periods of high uncertainty correspond to the initial start-up phase and command throttle changes. The uncertainty estimates are discovered automatically in the algorithm. Note that the model uncertainty is high for three time segments in the flights resulting in the errors for both flights staying well within the bounds for these segments. However in STS-126, even with the adaptive threshold, the estimated error crosses the bounds at 93.6 seconds and persists for the remainder of the flight

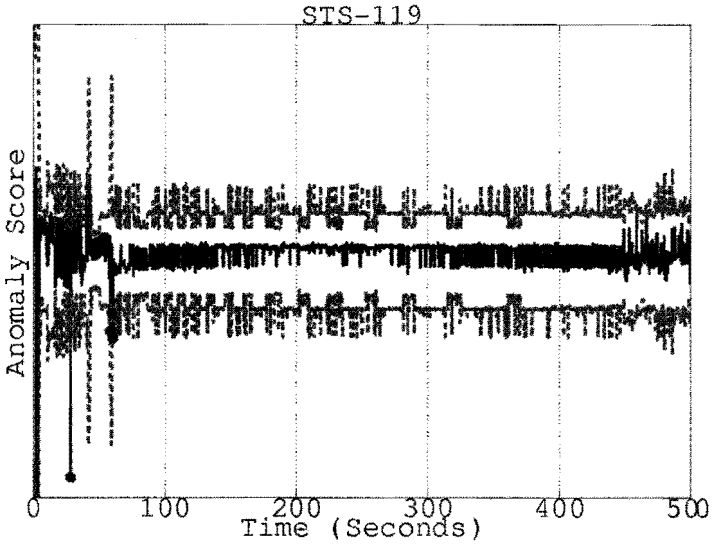

Fig. 8. This figure shows the full flight VS scores for STS-119. The estimate error stays well within the adaptive threshold bounds throughout the entire flight. Only a few false alarms are present during the transient periods

Table 5. This table shows the flights used to build and test the models for analysis of STS-119

Training Flights	Testing Flights
STS-82	STS-120
STS-85	STS-124
STS-91	STS-119
STS-95	
STS-96	
STS-103	
STS-92	
STS-102	
STS-105	
STS-114	
STS-121	
STS-116	

periods are given higher thresholds due to higher variance across the assorted models. The advantages are that fewer false alarms are observed with this thresholding method and that a smoother signal is produced due to multi-model estimate averaging. Figure 7 also shows the same adaptive threshold method for STS-126. Again the transient periods are given higher thresholds and therefore only a few false alarms are reported. It is also important to note that the threshold continues to preserve the fault at 93.6 seconds on until the end. This technique was used to analyze STS-119 as well. Figure 8 shows the VS scores using the adaptive threshold method. Only a small set of threshold crossing can be observed, however the majority of the flight is well-behaved and within the threshold regime.

CONCLUSIONS

Herein we have discussed the analysis methodology and algorithms used for detecting anomalies in the Flow Control Valve in the Main Propulsion System of the Space Shuttle. The methodology is an adaptation of Extreme Programming, which offers a framework for rapid, robust, and repeatable development which was essential due to the rapid response required by our team. We presented the results of two key algorithms, the Inductive Monitoring System, and Virtual Sensors, and showed that these algorithms detected anomalies in data from the November 2008 launch of Space Shuttle Endeavor (STS-126). The Inductive Monitoring System provides a one-class modeling approach for anomaly detection by learning the past nominal behavior of the data generating process and then comparing the currently observed values against a library of clusters of nominal behavior. This algorithm identifies the anomaly and also sheds light on the source of the anomaly which was previously corroborated by independent physical inspection. The Virtual Sensors algorithm builds an internal predictive model of nominal behavior and compares the predictions with the actual values. Significantly large residuals are flagged as anomalies. We showed methods for calculating fixed and adaptive thresholds for the Virtual Sensors algorithm. The next steps in this research program include building models to enable prognostics on these systems.

ACKNOWLEDGMENTS

This research was supported by the Integrated Vehicle Health Management Project in the NASA Aviation Safety Program and the NASA Engineering and Safety Center (NESC). The authors thank Melissa Levy for providing the pre-processed data.

REFERENCES

[1] A.N. Srivastava and W. Buntine,
Predicting engine parameters using the optical spectrum of the space shuttle main engine exhaust plume,
AIAA Conference Proceedings, 1995.

[2] D.A. Benzing and K.W. Whitaker,
Approach to space shuttle main engine health monitoring using plume spectra,
Journal of Spacecraft and Rockets,
Vol. 35, No. 6, pp. 830–836, 1998.

[3] G.D. Tejwani, D.B. van Dyke, F.E. Bircher and D.J. Chenevert,
Emission spectra of selected ssme elements and materials,
1992, Technical Report:NASA-RP-1286,
Document ID: 0133990.

[4] M. Schwabacher,
Machine learning for rocket propulsion health monitoring,
SAE Transactions, Vol. 114, No. 1, 2005.

[5] ——,
A survey of data-driven prognostics,
Proceedings of the AIAA Infotech, 2005.

[6] I. Tumer and A. Bajwa,
A survey of aircraft engine health monitoring systems,
Proceedings of the AIAA, 1999.

[7] Shuttle reference manual,
NASA,
<http://spaceflight.nasa.gov/shuttle/reference/>, 1988.

[8] Gaseous hydrogen flow control valves,
NASA,
<www.nasa.gov/pdf/313985main Flow_Valve_Fact.pdf>,
2009.

[9] A.A.R. Jeffries and C. Hendrickson,
Extreme Programming Installed,
Addison-Wesley, 2001.

[10] D. Wells,
Extreme programming: A gentle introduction,
<www.extremeprogramming.org>, 2006.

[11] B. Schölkopf, J.C. Platt, J.C. Shawe-Taylor, A.J. Smola and R.C.Williamson,
Estimating the support of a high-dimensional distribution,
Neural Comput., Vol. 13, No. 7, pp. 1443–1471, 2001.

[12] W. Royce,
Managing the development of large software systems,
Proceedings of the IEEE WESCON, 1970.

[13] Cross industry standard process for data mining,
<www.crisp.dm.org>, 2009.

[14] D.L. Iverson,
Inductive system health monitoring,
Proceedings of the 2004 International Conference on Artificial Intelligence (IC-AI'04), CSREA Press, Las Vegas, NV, 2004.

[15] A.N. Srivastava, N.C. Oza and J. Stroeve,
Virtual sensors: Using data mining techniques to efficiently estimate remote sensing spectra,
IEEE Transactions on Geoscience and Remote Sensing, Vol. 43, No. 3, 2005.

[16] M.J. Way and A.N. Srivastava,
Novel methods for predicting photometric redshifts from broad band photometry using virtual sensors,
Submitted to Astrophysical Journal, 2006.

[17] S.D. Bay and M. Schwabacher,
Mining distance-based outliers in near linear time with randomization and a simple pruning rule,
Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[18] A.N. Srivastava and S. Das,
Detection and prognostics on low dimensional systems,
IEEE Transactions on Systems Man and Cybernetics, Part C, Vol. 39, No. 1, 2009.

[19] M. Abramowitz and I.A. Stegun,
Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables,
Dover Publications, 1972.

[20] L. Breiman,
Bagging predictors,
Machine Learning, Vol. 24, No. 2, pp. 123–140, 1996, [Online]. Available: <citeseer.ist.psu.edu/breiman 96bagging.html>.