ADAPTIVE FAULT DETECTION ON LIQUID PROPULSION SYSTEMS WITH VIRTUAL SENSORS: ALGORITHMS AND ARCHITECTURES

Bryan L. Matthews
Stinger Ghaffarian Technologies Inc. at NASA Ames Research Center
Moffett Field, CA
Ashok N. Srivastava, Ph.D.
NASA Ames Research Center
Moffett Field, CA

ABSTRACT

Prior to the launch of STS-119 NASA had completed a study of an issue in the flow control valve (FCV) in the Main Propulsion System of the Space Shuttle using an adaptive learning method known as Virtual Sensors. Virtual Sensors are a class of algorithms that estimate the value of a time series given other potentially nonlinearly correlated sensor readings. In the case presented here, the Virtual Sensors algorithm is based on an ensemble learning approach and takes sensor readings and control signals as input to estimate the pressure in a subsystem of the Main Propulsion System. Our results indicate that this method can detect faults in the FCV at the time when they occur. We use the standard deviation of the predictions of the ensemble as a measure of uncertainty in the estimate. This uncertainty estimate was crucial to understanding the nature and magnitude of transient characteristics during startup of the engine. This paper overviews the Virtual Sensors algorithm and discusses results on a comprehensive set of Shuttle missions and also discusses the architecture necessary for deploying such algorithms in a real-time, closed-loop system or a human-in-the-loop monitoring system. These results were presented at a Flight Readiness Review of the Space Shuttle in early 2009.

INTRODUCTION

During the launch of STS-126 shuttle *Endeavour* in November of 2008 the gaseous hydrogen FCV experienced a fault in one of the space shuttle main propulsion system (MPS). The fault occurred because a piece of the poppet (the component of the valve that regulates the flow of gaseous hydrogen) in the valve broke off and was carried down the return lines to the tank. Fortunately this did not puncture the line, which could have resulted in fuel leak and possibly catastrophic failure. Before the next shuttle launch of *Discovery* in March of 2009 a study on this issue for the STS-119 flight readiness review was conducted. The aim was to determine if the fault that occurred could be detected in the recorded pressure and temperature sensors and if a precursor to the faulty FCV could be detected in the flights leading up to the faulty flight. Multiple anomaly detection techniques were applied to the flight data including IMS[1], Orca[2], and Virtual Sensors[3,4] (VS), as well as manual examination of the data by the flight engineers. All three algorithms were able to detect an non-directed unintentional drop in the outlet pressure sensor at the suspected time of the fault in STS-126. This paper will discuss the system integration process of merging the VS algorithm with the current system as well how the algorithm performs on the flight data.

CURRENT METHODS

The current method for examining anomalies that occur in the liquid hydrogen system is to set single sensor thresholds or red lines. These thresholds define the normal operating conditions for each sensor in the system and can quickly and reliably detect events where a sensor reading exceeds the threshold. This approach cannot detect a situation if multiple sensors fall within the boundaries of normal operations but exhibit anomalous behavior when taken together. If a combination of sensors deviates from their normal operating region the anomaly may go undetected. In the case of the FCV fault discussed here, the pressure sensor fell out of the normal bounds. For anomaly detection methods in sequential data, these envelope detections do not apply[7].

VIRTUAL SENSORS ALGORITHM

The Virtual Sensors algorithm works by predicting the value of one or more sensors giving the output of another set of sensors. These sensors may be correlated with the target sensor either linearly or nonlinearly. The algorithm uses a set of so-called base models which could be any type of statistical regression method. For example, linear or polynomial regression techniques, multivariate adaptive regression splines, neural networks, and other techniques can be used to make a prediction.

In this paper, we use decision trees as the base model[6].We build the models as follows: We take the entire data set divided into two parts: a training part and the test part. The training part may be comprised of a large number of previous flights with the same or similar engine configurations. We randomize the order of this training and extract 80% of the data. Using this sample, we build a decision tree to predict the pressure sensors given other inputs such as the control signals to the engine, and related pressure and temperature signals. The resulting decision tree is stored and then another 80% sample is drawn with replacements from the entire training set. A second model is built as before. This process is replicated between 20 and 100 times. In this paper, we replicated the decision trees 100 times.

The addition of the virtual sensors is made by using the testing data as input into all 100 decision trees. The average of all 100 decision trees is used as the prediction of the target sensor value and the standard deviation of the predictions of the 100 decision trees is used as an estimate of the uncertainty. This approach is based on the concept of bagging predictors and leads to a mean prediction as well as an estimate of uncertainty for each point in the test set. In more complex situations, the regression functions can be modified to include prior information about the expected correlations between inputs and outputs.

RESULTS

For the STS-119 flight readiness review the VS algorithm had been applied to a set of 11 shuttle flights with the same valve configuration monitoring the FCVs on each SSME. The algorithm was able to detect two flights with persistent anomalies.

STS-126

The first flight detected was STS-126, which was the faulty flight that resulted in the damaged poppet. Using the outlet pressure sensor as the target, the algorithm was able to detect the anomalous event at the point in the flight where the engineers believe the fault occurred. Figure 1 shows the three anomaly scores for each of the FCVs corresponding to each SSME. The fault was detected on the FCV for SSME 2, which can be seen in the high prediction error in the figure, where as the FCVs for SSME 1 and 3 showed no signs of faults (the FCV for SSME 1 contained single exceedances, which were due to noisy data and can be corrected by applying the data quality filter as well as only signaling alarms only when consecutive exceedances are detected). Notice at approximately 50 seconds in all three FCV anomaly scores the adaptive threshold is able to adjust to transients induced by throttle changes and other system state changes. By incorporating the high variance in the model predictions to adjust the alarm threshold, false positives can be avoided that would otherwise be flagged when using a fixed threshold throughout the entire flight.

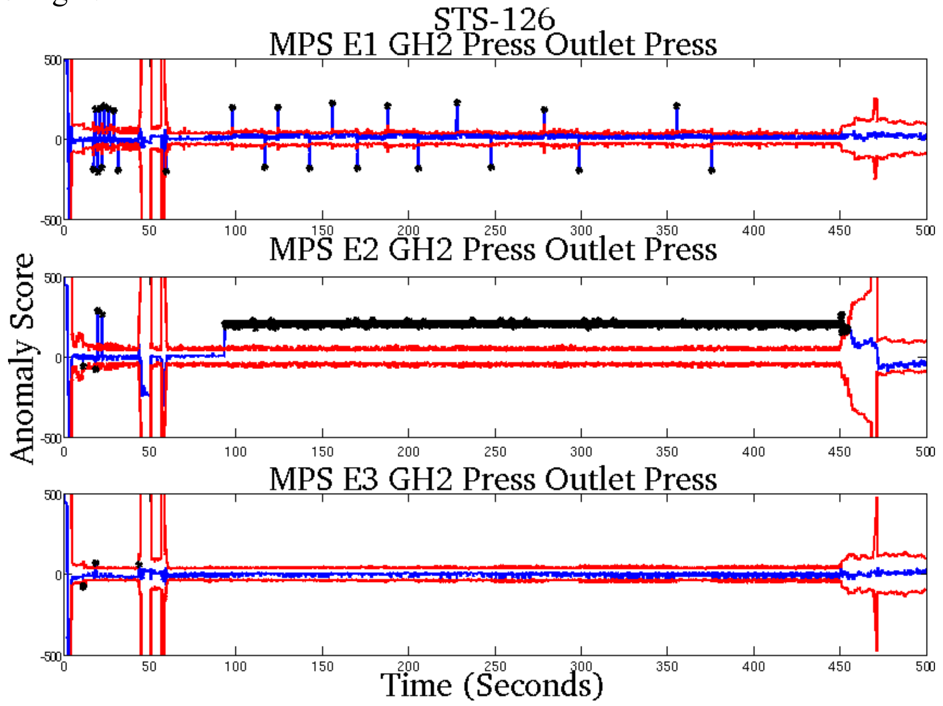

Figure 1: This figure shows the output of the Virtual Sensors algorithm on the y-axis as a function of time on the x-axis. The blue line corresponds to the prediction of the Virtual Sensor and the red lines indicate the point-wise estimate of uncertainty in the model. The points where the blue line falls outside the envelope defined by the redlines are tagged with a black dot. Notice that the MPS E2 GH2 Pressure Outlet Pressure is anomalous from about 93.6 seconds for the remainder of ascent.

STS-88

The other anomalous flight that the VS algorithm detected was STS-88. Notice in Figure 2 the FCV for SSME 3's prediction error is much lower, indicating that the outlet pressure is higher than predicted, while the FCV for SSME 1 & 2 appear to be functioning normally. This was presented to the domain experts and it was determined that SSME 3 was set up under a different engine configuration than the remaining 10 flights as well as the other two engines on STS-88. The alternate configuration results in a higher chamber pressure and therefore a higher input and output pressures for the FCV. The detection of this flight demonstrates that the algorithm is working correctly, however the flight engineers still consider this is a false positive, since the engine and FCV were operating normally. As with most data-driven algorithms, including additional training data that represents this particular SSME operating configuration would allow the models to learn this system mode, account for the adjusted pressure, and therefore report that the FCV is operating normally.

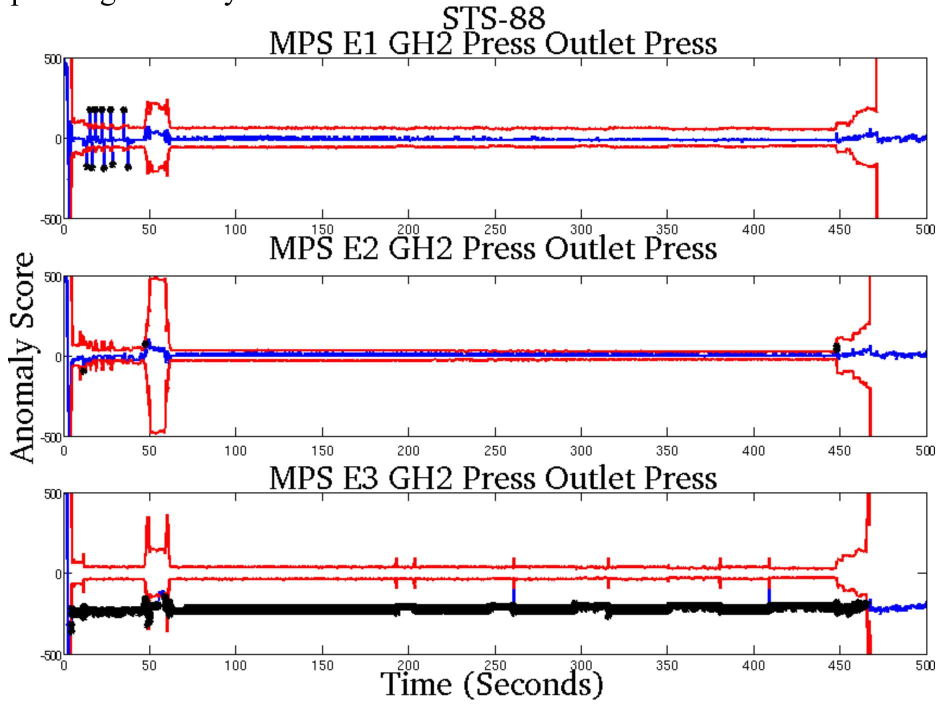

Figure 2. This figure shows the output of the Virtual Sensors algorithm on the y-axis as a function of time on the x-axis. The blue line corresponds to the prediction of the Virtual Sensor and the red lines indicate the point-wise estimate of uncertainty in the prediction. The points where the blue line falls outside the envelope defined by the redlines are tagged with a black dot. Notice that the MPS E3 GH2 Pressure Outlet Pressure shows an anomaly. However, further investigation shows that this anomaly was not due to a fault or failure in the system but was due to a different engine configuration.

SYSTEM INTEGRATION

When integrating any algorithm into an established system, two important factors must be considered. How the algorithm will interface with the system as well as requirements for the algorithm will need to be thoroughly reviewed before the algorithm can be implemented. In the proposed method of integration the algorithm will be running in a monitoring fashion and alert controllers to an anomaly that has been detected in any of the three FCVs. This method is similar to the implementation of the IMS algorithm, which has been used for monitoring International Space Station's Attitude Control and Thermal Operations[8]. This level of implementation allows the current system to operate normally and places a human in-the-loop for decision making purposes. The sensors that the algorithm will be monitoring are currently telemetered to the ground where the algorithm could cleanly connect to the streaming data for analysis. Figure 3 shows where the algorithm might be integrated into the current system, with the components in the dotted box representing the additions to the system. Since the VS algorithm operates on each time sample independently, the core functionality is already well-suited for a streaming real time environment without the need for modifications to the algorithm.

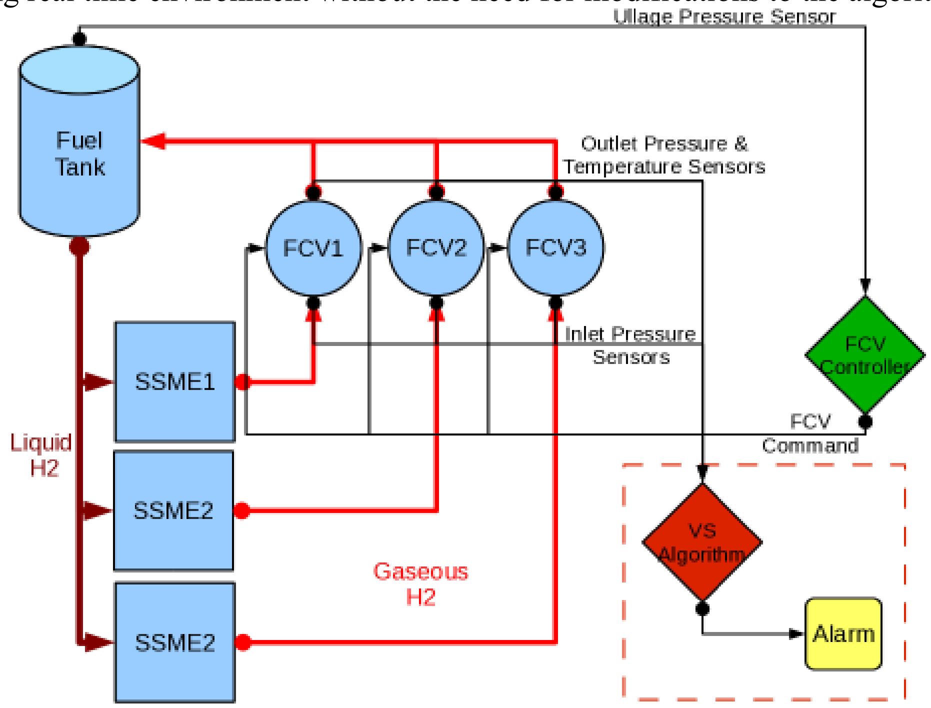

Figure 3: This figure is the system level diagram for the FCVs. The blocks in the dotted box are where the proposed algorithm may be inserted.

REQUIREMENTS

To properly integrate the algorithm into the current system, a set of requirements must first be specified to outline the algorithm's functionality and expected performance. The algorithm and resulting code must also go through both rigorous quality assurance tests as well as a thorough verification and validation process to assure the requirements have been fully met. A few set of high level requirements are as follows:

1. The algorithm must be able to process streaming data at the current sampling rate for the sensors being monitored (10Hz). The algorithm will be expected to provide alarm feedback (either nominal or anomalous) at the same input rate.
2. Any fixed delays between the measured signal and the alarm signal (such as processing time and/or windowed effect) should not exceed 1 second.

3. Algorithm's performance metric shall be determined by a false positive rate of less than 1% and correct detection rate of greater than 99% when tested on simulated or true nominal and faulty data.
4. The algorithm shall be able to detect and respond appropriately without reporting false positives to sensor failure events due to data drop-out and/or noisy sensors.

As the algorithm implementation process is explored additional software and hardware requirements may need to be described to help ensure the algorithm meets the performance requirements, such as programming languages, memory, and CPU speeds. Note the algorithm is currently coded in the Matlab environment and has not been tested for the speed requirements outlined above. A more efficient programming language such as C/C++ may be more appropriate for meeting the speed requirement. The false positive and correct detection rate requirements have been met when examining the two faulty flights discussed in the results section. For the requirement regarding the handling of the faulty and noisy sensors: more examples and feedback from domain experts will need to be obtained so a successful filter can be designed and tested.

SYSTEM ARCHITECTURE

The algorithm's system architecture is divided up into 2 main operations: an offline training phase and an online monitoring phase. For the offline phase there is no need for a strict training time requirement; only that the algorithm needs to compute the models within a reasonable time given the size of the training data (i.e. on the order of seconds for megabytes of data and minutes for 1 to 10 gigabytes of data). The algorithm in the training phase will take ***N*** sets of samples from historical and/or simulated flights to produce ***N*** models. These models will then be fixed and supplied to the online monitoring phase of the algorithm. Since the core functionality of the algorithm does not require the models to be of any one type, various regression techniques can be employed in this framework. At this point a set of requirements must be established to standardize and describe the model format that will be passed between the training and testing phases as well as any limits to the number of models that can be passed between the two sets of code due to any processing limitations of the hardware. This will ensure that the two portions of the algorithm are compatible and can be executed on the defined platform. A common technique is to allow the model building portion of the code to be modularized to allow various methods, including linear or nonlinear regression techniques, to be "plugged in". In this case the training and testing modules will have to define a common model format for each type that is passed between the two portions of code. The interfaces between module and the framework on both the training & monitoring sides must be standardized as well to allow the modules to be interchangeable.

In Figure 4 a flow chart of the online phase of the algorithm is shown. The algorithm will receive the streaming data from a well defined interface such as a TCP network connection. The proposed platform for the algorithm is not intended to be a true hardware real time operating system, but rather software that can be run on a desktop machine while being able to keep pace with the streaming data without dropping data or accumulating a delay. For this application a TCP network connection is sufficient. The data will be processed through a data quality filter, which will check each sensor for data loss or noise and correct the signal. Once the sensors have been corrected, each time sample vector will be passed through all models in parallel. The models will predict the target variables for the FCV for each SSME. The variance of the predictions across the multiple models will provide the adaptive threshold used to determine if the prediction error is large enough to signal an alarm. To reduce false positives, consecutive exceedance may be monitored before reporting an alarm. It is important to consider that this technique will result in a delay, which must not exceed the maximum delay specified in the requirements.

To ensure the algorithm is meeting the false positive and correct detection metrics specified in the requirements the algorithm must be tested on either simulated or real data with an associated ground truth. A technique known as cross validation, which selects multiple subsets for training and testing to evaluate the algorithm across the possible model parameters, can be used to select optimal parameters and appropriate thresholds that meet the specified metrics. The results can be compared against the ground truth to produce a Receiver Operating Characteristic (ROC) curve. The area under the curve (AUC) is an established metric[6] for determining the tradeoff between false positives and true positives. Once this metric is calculated for all possible model candidates then the optimal model parameters can be selected and fixed for operational use.

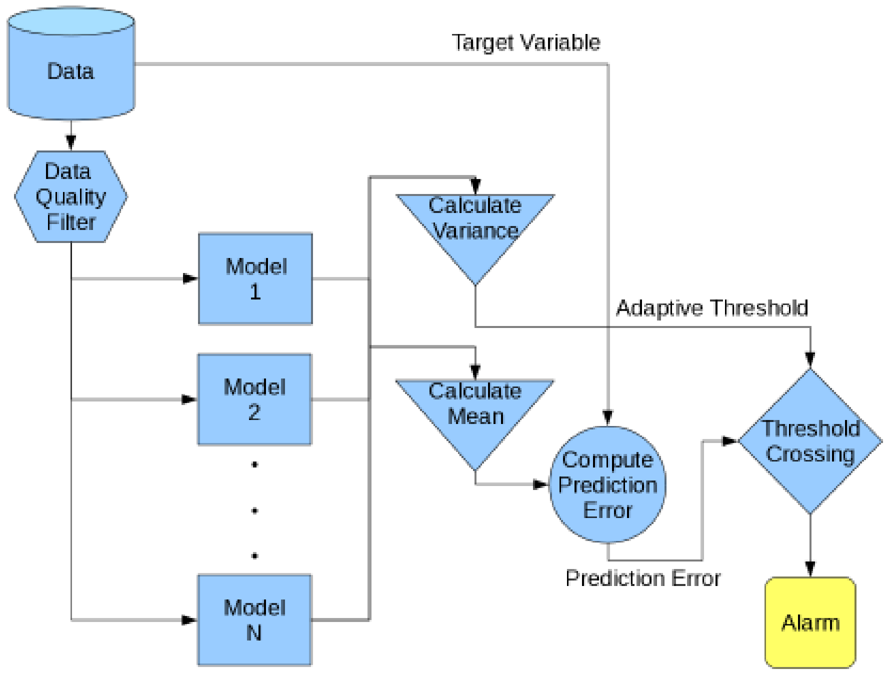

Figure 4: This figure is the flow chart of the system level monitoring phase of the algorithm. The models are built offline and are fixed throughout the testing.

SUMMARY AND CONCLUSIONS

This paper has discussed the benefit to using a multi-dimensional anomaly detection approach to detect anomalous behavior in the shuttle’s FCV subsystem over the current single variant exceedance method. The algorithm has also shown an ability to detect significant anomalies in the data while also minimizing false positives. Since the algorithm is fundamentally constructed to operate on single sample vectors of sensors, few modifications to the underlining algorithm are needed to adapt the anomaly detector to the current system. Additionally, system level integration issues have been discussed, considered, and proposed for the implementation of the algorithm with minimal modifications to the current system.

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 would like to thank Melissa Levy for providing us with the pre-processed data as well as Bill Lane and Bob Beil for their valuable insights into shuttle's engine configurations, FCV and fuel supply system.

REFERENCES

1. Iverson, D. L., **Inductive system health monitoring**, In *Proceedings of The 2004 International Conference on Artificial Intelligence* (IC-AI04), Las Vegas, Nevada, June 2004. CSREA Press.
2. 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.
3. 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 43 2005, no. 3.
4. 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.
5. A. P. Bradley. **The use of the area under the ROC curve in the evaluation of machine learning algorithms**. Pattern Recognition, 30(7):1145 – 1159, 1997.
6. L. Breiman, J. H. Friedman, R. A. Olshen, and J. C. Stone. (1983). **Classification and Regression Trees**. Wadsworth.
7. A. N. Srivastava, **Discovering Anomalies in Sequences with Applications to System Health**, Proceedings of the 2005 Joint Army Navy NASA Air Force Interagency Conference on Propulsion, Charleston SC, 2005.
8. Iverson, D. L., Martin, R., Schwabacher, M., et. al. **General Purpose Data-Driven System Monitoring for Space Operations**, Proceedings of the InfoTech Conference, Dayton Ohio, October 2009.