Published in the Proceedings of The 2004 International Conference on Artificial Intelligence (IC-AI'04), CSREA Press, Las Vegas, NV, June 2004.

Inductive System Health Monitoring

David L. Iverson
NASA Ames Research Center
Mail Stop 269-4
Moffett Field, CA 94035

David.L.Iverson@nasa.gov
Phone: (650) 604-3115

Abstract - The Inductive Monitoring System (IMS) software was developed to provide a technique to automatically produce health monitoring knowledge bases for systems that are either difficult to model (simulate) with a computer or which require computer models that are too complex to use for real time monitoring. IMS uses nominal data sets collected either directly from the system or from simulations to build a knowledge base that can be used to detect anomalous behavior in the system. Machine learning and data mining techniques are used to characterize typical system behavior by extracting general classes of nominal data from archived data sets. IMS is able to monitor the system by comparing real time operational data with these classes. We present a description of learning and monitoring methods used by IMS and summarize some recent IMS results.

Keywords: System Health Monitoring, Inductive Learning, Clustering, IVHM

1.0 Introduction

Model based reasoning is a powerful method for performing system monitoring and diagnosis. Typical model based reasoning techniques compare a system model or simulation with system sensor data to detect deviations between values predicted by the model and those produced by the actual system. [1] In effect, a model based reasoner uses the collected system parameter values as input to a simulation and determines if a particular set of input values is consistent with the simulation model. When the values are not consistent with the model a "conflict" occurs, indicating that the system operation is off nominal (when compared to the presumably correct model). [2] Building models for model based reasoning is often a difficult and time consuming process. The Inductive Monitoring System (IMS) provides a method that can monitor the health of a system with nearly the same fidelity as a model based reasoner, but without the need to manually build a model. IMS automatically defines groups of consistent system parameter data by examining and generalizing from examples of nominal system data. If a system model were available, a set of data values selected from one of these groups and presented to the model would typically compute without conflicts. With a sufficiently broad training data set, the knowledge base produced by IMS should contain most or all of the consistent parameter value combinations necessary to effectively characterize and monitor nominal system operation. After learning how

the system behaves when operating correctly, IMS can identify off nominal behavior and send appropriate alert messages to system operators.

2.0 Learning Nominal System Behavior

IMS automatically builds monitoring knowledge bases from nominal data sets collected either directly from the system or from simulations. Machine learning and data mining techniques are used to characterize typical system behavior by extracting general classes of nominal data from archived data sets. In particular, IMS uses *clustering* to group sets of consistent parameter values found in the training data. Clustering is the unsupervised assignment of elements of a given set to groups or clusters of similar points. [3] The implementation of IMS described here draws from two clustering techniques: K-means clustering [4] and density-based clustering [5].

2.1 Data Vectors

The basic data structure of the IMS algorithm is a vector of system parameter values. (Fig. 1) Each vector is an ordered list of parameters collected from the monitored system by a data acquisition process. The vectors can also contain derived parameters computed from collected data. These vectors define the points in N-dimensional space that are grouped by the IMS clustering algorithm. The values used in a given vector may be collected simultaneously by the data acquisition system, or collected over a period of time. The user specifies the size and contents of the vector structure to match the monitoring application.

Pressure A	*Valve 1 Position*	*Pressure B*	*Valve 2 Position*	*Pressure C*	*Temperature 1*	*Temperature 2*
2857.2	86.4%	1218.4	96.2%	1104.1	49.8	37.6

Figure 1 - Sample IMS vector

	Pressure A	*Valve 1 Position*	*Pressure B*	*Valve 2 Position*	*Pressure C*	*Temperature 1*	*Temperature 2*
High	2857.6	86.8%	1219.2	96.3%	1105.0	50.1	38.2
Low	2855.8	86.2%	1215.7	95.5%	1103.2	49.6	37.5

Figure 2 - Sample IMS cluster structure

2.2 Building Clusters from Nominal Data

IMS processes the training data by formatting input data into the predefined vector format and building a knowledge base containing clusters of related value ranges for the vector parameters. (Fig. 2) Each cluster defines a range of allowable values for each parameter in a given input vector. The vector of high values and the vector of low values in a cluster can be thought of as corners defining a minimum bounding rectangle in N-space. Points that fall inside or very near

this rectangle are considered to be within the system's nominal operating range, since the rectangles are defined by nominal data. The high and low ranges for each element in the cluster can also be considered as allowable ranges for the corresponding parameter, provided the other parameters are within the other ranges specified in that cluster. This view is similar to model-based reasoning with interval arithmetic where simulations are performed using a range of possible values for each parameter, rather than a single value. [6]

IMS starts the training process with an empty cluster database. It reads the nominal training data and formats it into vectors. The first vector is inserted in the database as the initial cluster. Each subsequent input vector is compared to the contents of the cluster database to find the cluster that is closest to the vector. The distance between a vector and a cluster can be measured using a variety of metrics. The standard Euclidean distance metric has proven effective for many applications. To calculate the distance between a given cluster and a vector, a point contained in the cluster is selected and the distance between that point and the vector of interest is computed with the distance formula. Different methods for selecting the point in the cluster can be used depending on desired results. One option, based on the K-means clustering method [4], measures the distance from the centroid of the cluster found by forming a vector from the average of the high and low values for each cluster parameter. For each new training input vector, IMS finds the cluster in the database that is closest to that input vector. IMS then determines if the input vector is contained in the bounding rectangle defined by that cluster, or if it is close enough to be incorporated into the cluster. As in density-based clustering [5], a threshold value, ε, specified by the user defines the maximum allowable distance between a cluster and vector to determine if the vector should be incorporated into the cluster. If the vector is close enough (distance less than or equal to ε), the cluster parameter intervals are expanded as necessary to include the new vector. If the distance between the training vector and the closest cluster in the database is greater than ε, a new cluster containing the vector is formed and inserted in the database. This learning process repeats until all of the training data has been processed and incorporated into clusters in the knowledge base. A smaller ε value will result in smaller clusters that provide tighter monitoring tolerance, but will sometimes produce a larger than desired monitoring knowledge base. A smaller knowledge base allows real-time monitoring at a higher data rate. The value of ε can be adjusted to balance knowledge base size and speed versus monitoring tolerance.

It can be useful to scale or normalize the data values before they are inserted in the vectors. For instance, each parameter can be scaled to represent a percentage of the maximum range for that parameter and the user can select a metric such that the distances between vectors and clusters are a percentage of the maximum distance found in the training data. This can give the user a more intuitive understanding of the monitoring knowledge base and the significance of any off-nominal system behavior encountered during system monitoring. Data values can also be relatively scaled to provide weighting of given parameters. For instance, scaling a parameter to have a larger possible range relative to other parameters in the vector will amplify any deviations in that parameter. Other combinations of data normalization and distance metrics may be useful for various situations.

After IMS processes all training data, the result is a database of clusters that characterize system performance in the operating regimes covered by the training data. In essence, each cluster defines constraints on the values allowed for each parameter in any particular monitoring input vector. If there is no cluster in the database that contains a given input vector or is "near" that input vector, then the system is behaving in an unexpected manner, indicating a possible system anomaly.

3.0 System Health Monitoring

To use the cluster knowledge base for system health monitoring, IMS simply formats the real time data into vectors and queries the knowledge base to locate the cluster that is closest to each input vector. The fastest IMS monitoring schemes require that the input data vectors be contained inside at least one of the knowledge base clusters (all parameter values are within the ranges specified by the cluster limits). This eliminates the need to perform distance calculations. A more informative monitoring technique will locate the cluster in the monitoring knowledge base that is closest to the input vector, and report the distance of that vector from the cluster. This will give the operator an idea of how far the system behavior is deviating from nominal operation as represented by the training data. Typically, if a vector is not contained within a cluster, the distance between the vector and the closest point in the bounding rectangle defined by the nearest cluster is reported. The monitoring program can account for incomplete training data or measurement inaccuracies by setting a tolerance on the maximum allowable distance of the input vector from the nearest cluster for the vector to be considered nominal data; i.e., the input data must be "close enough" to previous nominal data to be considered nominal. The monitoring program could also use multiple tolerance values to determine the level of alert to send to the operators. An input vector further than a given distance from the nearest cluster may indicate a significant problem requiring immediate attention, while a vector somewhat closer than that limit may be suspect enough to warrant extra vigilance, but not require immediate action. To gauge general system health or to track general system behavior over time, the monitoring interface could graph or summarize the distance the input vectors collected during the time of interest have fallen from the nominal clusters.

In order to use the IMS generated cluster database for real time or near real time system monitoring, an efficient cluster indexing and retrieval scheme may be required. Several applicable schemes have been developed in the area of nearest neighbor searching. [7] In order to allow searching the database for the closest cluster, the scheme must include a distance metric and the ability to return the record that is nearest to the query point, not just those that contain the query point. The search and retrieval speed must also be sufficiently fast to keep up with the expected data acquisition rate. An efficient indexing and retrieval scheme can also help to speed up the initial IMS training process since it makes similar closest cluster queries. IMS applications to date have successfully monitored 1 KHz data rates without distance calculations and 50 Hz data rates with distance calculations on computers running with clock speeds under 1 GHz.

4.0 IMS Application Example

The IMS methodology is domain independent and can be used in a variety of system monitoring situations including aerospace, transportation, manufacturing, power generation and transmission, medical, or process monitoring applications. Of particular interest to NASA is the application to Integrated Vehicle Health Management (IVHM), either on board a vehicle or in a mission control room. To demonstrate the utility of IMS in a mission control setting, we recently built an IMS knowledge base to monitor temperature sensors in the wings of a Space Shuttle Orbiter, and used that knowledge base to analyze archived telemetry data collected from the ill-fated STS-107 Columbia Space Shuttle mission. This flight came to a disastrous end when the Columbia orbiter was destroyed during reentry, claiming the lives of all seven crew members. The ultimate cause of the accident was determined to be a breach in the Thermal Protection System on the leading edge of the left wing, caused by a piece of insulating foam that struck the wing approximately 82 seconds after launch. The first indication of the damage that was noticed by mission controllers monitoring telemetry data was not seen until Orbiter re-entry, 17 days after launch. That anomaly was a slight

increase in a brake line temperature of the left main landing gear that occurred about seven minutes before the destruction of the vehicle. [8]

4.1 IMS STS-107 Analysis

The post mission IMS analysis of the STS-107 Columbia flight concentrated on telemetered data from temperature sensors in the wings of the orbiter. Analyses of telemetry data from two flight phases, launch/ascent and on-orbit, were conducted. In both analyses IMS detected anomalies much earlier in the data than monitoring systems available in mission control. This example will focus on the STS-107 launch and ascent analysis.

IMS knowledge bases for the launch and ascent analysis were generated from training data collected during five previous Columbia flights. Separate knowledge bases were generated for each wing. Training vectors were formed from four corresponding temperature sensors in each wing of the Orbiter. (Fig. 3) Since ambient temperatures differed on each flight, the data vectors were normalized to the most centrally located sensor (Main Gear Brake Line Temperature B) by expressing the other sensor values relative to the value of that sensor in each telemetry time slice. The resulting vectors contained three parameters each. In the results shown here, monitoring was further focused by comparing sets of data in a moving four second window (typically 12 to 16 data points). These data windows were normalized by shifting the points to move the centroid of the window to the origin. The points retained their relative positions within each window. The data sets used for training and analysis covered the time period from launch through ascent to just before Main Engine cut off. The resulting left wing knowledge base contained 502 clusters and the right wing knowledge base contained 240 clusters.

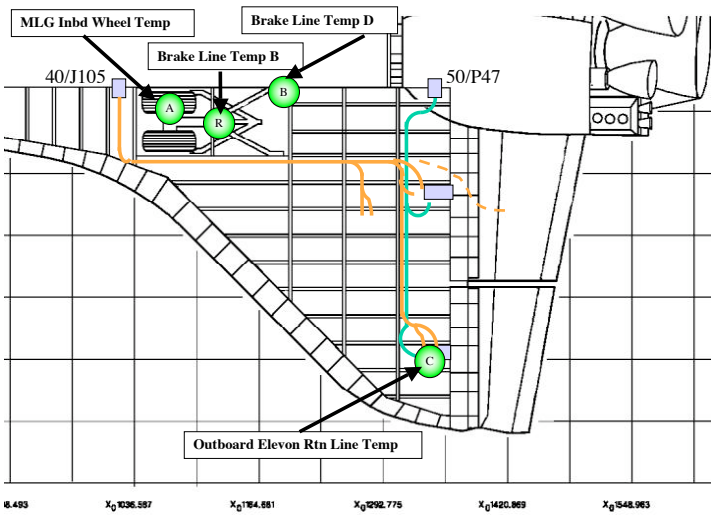

Figure 3 – Left Wing Sensors used in Columbia Ascent Analysis

4.2 IMS STS-107 Analysis Results

The results of the IMS analysis of the STS-107 Columbia launch are graphed in figure 4. The horizontal axis represents time, beginning at the moment of lift off. The vertical axis represents

the IMS measure of deviation from nominal behavior, that is, the distance of the input vector from the closest nominal cluster. The distance measure has been scaled to represent a percentage of the maximum distance in the vector space covered by the training data. Results from the left wing are represented by the lighter line, while right wing results are shown as a darker line for comparison. The vertical line near time 15:40:22 shows the moment of the foam impact event that breached the Thermal Protection System on the left wing. Notice that the IMS results for the left and right wings track each other fairly well until shortly after the foam impact, at which point they begin to diverge sharply. The IMS deviation values for the right wing continue to show results within a reasonable range of nominal, while the left wing deviation values increase to nearly 5 times those of the right wing. Although this analysis was performed off-line using archived data, the techniques used could be implemented for real time monitoring. Significant deviations in a group of sensors or asymmetrical results for identical sensor sets, especially of large magnitude such as shown in this analysis, may provide indication of anomalies earlier in a mission than current telemetry monitoring tools.

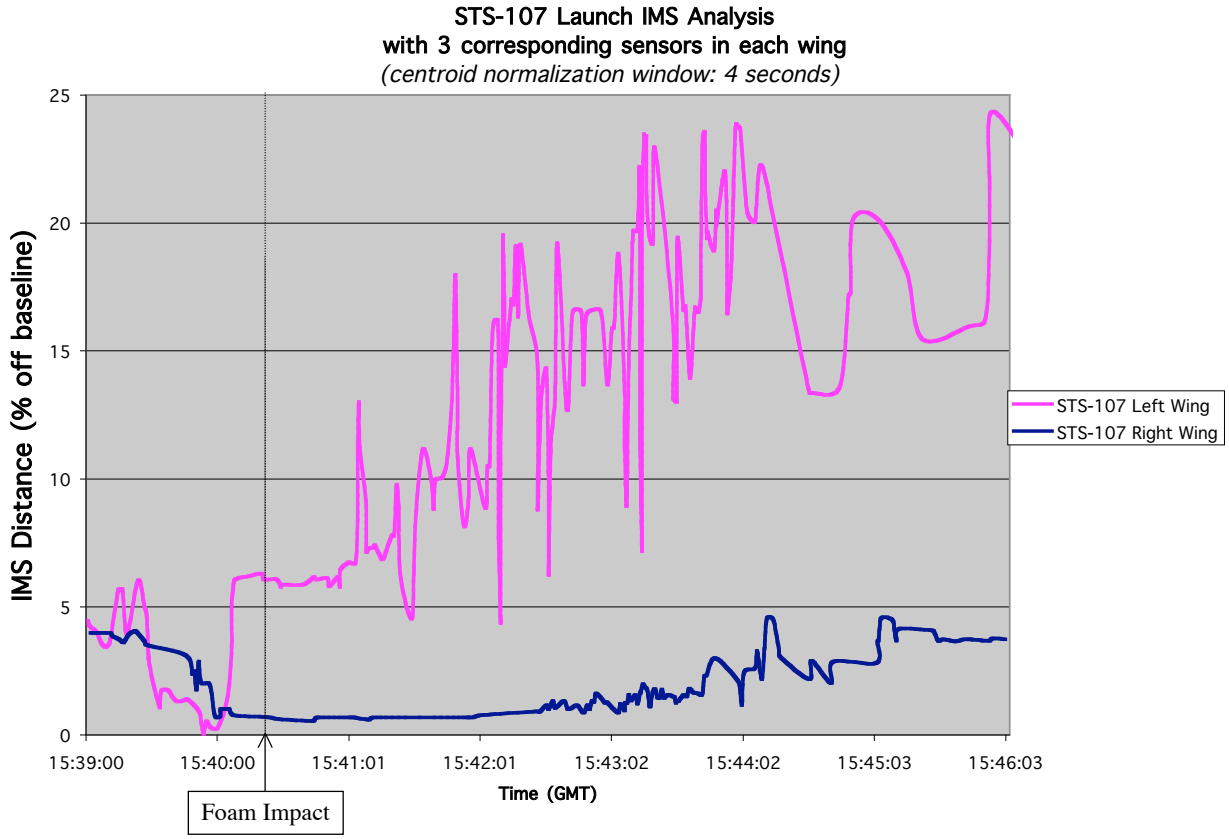

Figure 4 – Results of IMS analysis of STS-107 Columbia launch

5.0 Conclusions and Future Work

These early results from the Inductive Monitoring System show that it is feasible to automatically construct a useful system monitoring knowledge base from archived system data using clustering techniques. These knowledge bases could provide system monitoring capability comparable to

that obtained by model based reasoning techniques, without requiring the cost and effort of building system models. In addition, the IMS monitoring routine may be used for real time or near real time system monitoring. As a mission control tool, IMS could help augment controller awareness of vehicle health and provide early detection of possible anomalies. As shown by the STS-107 analysis, IMS revealed evidence of the Thermal Protection System anomaly within minutes of the foam strike while current mission control tools did not detect the problem until Orbiter re-entry, 17 days later.

IMS applications currently under development include a turbine aircraft engine monitoring system, and monitors for various subsystems of the Stratospheric Observatory for Infrared Astronomy (SOFIA), an airborne observatory co-developed by NASA and the German space agency DLR. Techniques to assist in parameter selection for IMS vectors, analyze IMS result trends, and integrate IMS monitoring with diagnostic routines are also being studied.

6.0 References

[1] D. Dvorak and B. Kuipers. "Model-Based Monitoring of Dynamic Systems*", Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Morgan Kaufman, LosAltos, CA., 1989.

[2] R. Reiter. "A Theory of Diagnosis from First Principles*", Artificial Intelligence*, 32(1):57-96, Elsevier Science, 1987.

[3] P.S. Bradley, O.L. Mangasarian, and W.N. Street. "Clustering via Concave Minimization*", Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordon, and T. Petsche(Eds.), pp 368-374, MIT Press, 1997.

[4] P.S. Bradley and U. M. Fayyad. "Refining initial points for K-means clustering", in *Proceedings of the International Conference on Machine Learning (ICML-98),* pp 91--99, July 1998.

[5] M. Ester, H-P Kreigel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*", Proceedings of the 2nd ACM SIGKDD*, pp 226-231, Portland, OR, 1996.

[6] W.C. Hamscher. "ACP: Reason maintenance and inference control for constraint propagation over intervals*", Proceedings of the 9th National Conference on Artificial Intelligence*, pp 506-511, Anaheim, CA, July, 1991.

[7] J.M Kleinberg. "Two Algorithms for Nearest-Neighbor Search in High Dimensions*", Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pp 599-608, El Paso, TX, May, 1997.

[8] H.W. Gehman, et al., "Columbia Accident Investigation Board Report", U.S. Government Printing Office, Washington, D.C., August 2003.