SSC17-WK-06

A Data-Driven Approach to CubeSat Health Monitoring

Serbinder Singh. John Bellardo, Jordi Puig-Suari
California Polytechnic University, San Luis Obispo
1 Grand Avenue, San Luis Obispo, CA 93407; (209)-298-4031
serbinder.singh01@gmail.com

ABSTRACT

Spacecraft health monitoring is essential to ensure that a spacecraft is operating properly and has no anomalies that could jeopardize its mission. Many current methods of monitoring system health are difficult to use as the complexity of spacecraft increase, and are in many cases impractical on CubeSat's which have strict size and resource limitations. To overcome these problems, new data-driven techniques such as Inductive Monitoring System (IMS), use data mining and machine learning on archived system telemetry to create models that characterize nominal system behavior. These characterizations can then be autonomously compared against real-time telemetry on-board the spacecraft to determine if the spacecraft is operating nominally.

This paper presents an adaption of IMS to create a spacecraft health monitoring system for CubeSat missions developed by the PolySat lab. This system is integrated into PolySat's flight software and provides real time health monitoring of the spacecraft during its mission. Any anomalies detected are reported and further analysis can be done to determine the cause. The system was successful in the detection and identification of known anomalies in archived flight telemetry from the IPEX mission. In addition, real-time monitoring performed on the satellite yielded great results that give us confidence in the use of this system in all future missions.

INTRODUCTION

There have been many advancements and improvements made throughout the years on the capabilities and functions of various satellites. The design of such systems has consequently become extremely sophisticated and complex. Unfortunately, the monitoring of such systems also becomes very complex as there are many sensor and component interactions that become hard to predict and classify as nominal through traditional techniques [2]. There are currently many traditional methods of monitoring spacecraft that include parameter limit checking, model-based, and rule-based techniques that become difficult and cumbersome as the complexity of the spacecraft increases. These challenges are exacerbated in CubeSat satellites where using extra downlink capacity for high resolution engineering telemetry is not feasible.

New data-driven techniques based on data-mining and machine learning have been developed to make this task of monitoring much more manageable and autonomous. One such technique, the Inductive Monitoring System (IMS), uses nominal archived data to create clusters that represent nominal system behavior. This model represented as a knowledge base of clusters can then be compared with new input to perform monitoring. IMS has been successfully implemented in various applications.

This paper uses the research and work done on IMS and applies it to create a validated and flight ready monitoring system for use on CubeSat satellites produced by the PolySat lab. This system is integrated into the flight software to provide real time monitoring and analysis of archived events.

BACKGROUND

Inductive Monitoring System

Monitoring the health of a spacecraft usually involves a large team of people including mission controllers and system engineers who analyze down-linked data to find any anomalies. A big team for this task is usually not possible for institutions that run CubeSat missions as the number of people involved is usually much smaller. Therefore a more autonomous approach is needed that can provide health monitoring with the least amount of input from people.

Traditional techniques of health monitoring include parameter limit checking where a reference table of nominal sensor values is created for all sensors on a system. This table is then compared against real-time telemetry to determine if the values fall within the ranges. If not, then that sensor may have an anomaly. This method of health monitoring is very inefficient and time consuming because as the number of components increase, the generation of this reference table becomes

extremely hard. It is difficult to correctly determine what would constitute a healthy sensor value. Also, multiple reference tables would have to be made for each of the satellites different operational modes due to different component interactions. Another drawback of such an approach is that it only considers individual parameter ranges when making its decision, and can't model complex interactions that may involve several concurrent parameters in the operating context [3].

Data driven techniques such as Inductive Monitoring System (IMS) were created to address the challenges of monitoring increasingly complex component interactions of spacecraft, and provide a more autonomous way of finding anomalies within a system. These techniques have been made possible by the abundance of archived system telemetry that exists for several different spacecraft and applications. IMS uses this archived telemetry to characterize nominal system behavior models that can be compared against real time telemetry to provide monitoring of system health. If the system is performing nominally, the telemetry will fall under one of the models. If not, than this may indicate a potential anomaly or fault in the system.

IMS is a distance-based anomaly detection tool that models the relationship between a set of sensors in time-series data as clusters. It uses vectors as a data structure that holds values of several related system parameters for a specific time. In its learning phase, it goes through the archived data, forms these vectors, and groups vectors with similar or consistent values in the same cluster. Therefore, each cluster defines a different characterization or nominal state that the system can be in and the sensor ranges that represent it. The cluster defines a nominal operating region that is represented as an N-dimensional hyper-cube in the vector space where N is the number of parameters chosen. Each dimension of this hyper-cube specifies a minimum and maximum value for that parameter in the given cluster. This is beneficial because it allows us to model interactions between related parameters instead of looking at each one individually. The end result of the learning phase is a knowledge base of many clusters that define a model of the nominal states of the system. This knowledge base can then be queried with new input to see if it falls within a nominal operating region. It is important that the training data is free of any anomalies to ensure bad behavior isn't incorporated into the system.

Once a knowledge base has been created from the learning phase, it can be used for real time monitoring or the analysis of archived events. This monitoring produces a deviation value that signifies how well the system is conforming to the model. Large deviation values may highlight a precursor to a malfunction or a malfunction itself. This monitoring phase does not explicitly pinpoint the exact problem with the system, rather it gives details as to which features are causing the issue and where it is occurring so that a mission controller can later do a closer inspection.

IMS monitoring starts by formatting real time data coming from the system to be monitored into the predefined data vectors from the learning phase. This data vector is normalized to ensure the parameters have the same scale, and can be further scaled with weights given to each parameter to given more significant features higher sensitivity. The data vector is then compared against each cluster in the knowledge base to find the one it falls within or has the minimum distance. If the vector does not fall within a cluster, this distance can be considered a deviation score signifying how far the input vector falls from a nominal cluster. The higher the deviation value, the more significant the anomaly. Deviation scores that are small may indicate nominal behavior that was not captured in the training data. A threshold value is usually given as a parameter to the monitoring algorithm that accepts input vectors that have deviation scores that fall under the threshold. Along with the deviation score, the individual parameter contributions to the score can be saved to give the operator more details as to which sensors are causing the issues. An overview of the two phases can be seen in Figure 1.

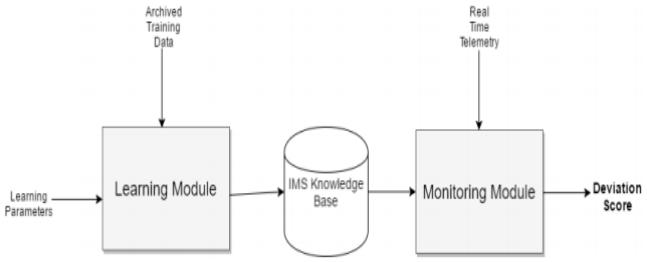

Figure 1: Overview of IMS

Overall, IMS's monitoring capabilities are robust and powerful. It gives the ability to model complex system behavior easily by just using nominal archived data. It is also very adaptable for system monitoring applications. The knowledge base can be updated at any time to provide a more accurate model of the system as more nominal telemetry is gathered. The features that are monitored can also be updated to remove or add new features that may provide better results. The strengths of IMS have led it to become very successful in a number of system applications.

PolySat Software Architecture

PolySat is a multidisciplinary lab run by students on Cal Poly's campus devoted to the design, fabrication, testing, and integration of CubeSats. While there is a certain level of fault tolerance built into the CubeSat's designed by the lab, there is no health monitoring system that exists that alerts members of anomalies occurring during a mission. To solve this problem, IMS is integrated into the flight software to provide monitoring capabilities.

PolySat's flight software is designed to be highly modular, extensible, and robust so that it can be used reliably for many missions. It is built on top of the Linux kernel and takes advantage of the large amounts of pre-existing code and libraries that exist to handle low-level tasks of managing hardware, drivers, and communication. The software also operates in an event-driven fashion where timed or command initiated event cause something to occur. The overall software architecture consists of three layers: processes, abstraction libraries, and drivers [5].

The flight software uses Linux's process model to provide address space and code isolation of major spacecraft functionality into separate processes. Processes are the highest level of software in the architecture and each serve a very specific role. Examples of some processes include System manager which is responsible for maintaining the state of the avionics system, Beacon which periodically broadcasts spacecraft health information, and SatComm which controls the radio. Processes can communicate with one another using Inter-process Communication which leverages the UDP/IP Networking protocol in Linux. Each process also contains an event-handler that allows it to respond to some event, or block otherwise.

There is a large set of functionality that is common across all processes. Examples of this include event and command handling, inter-process communication, and configuration management. These common services are provided by a standard set of abstraction libraries that expose an API that processes can use. By using these libraries, the development of the process is significantly easier and faster.

SYSTEM OVERVIEW

PolySat's system health monitoring abilities are only limited to detection of problems that are glaringly obvious through its beacon or manual examination of flight telemetry. These methods are not only inefficient and time-consuming for mission controllers, but also require increased bandwidth to send large amounts of telemetry which could be utilized more effectively for mission tasks.

Fortunately, PolySat has collected a large archive of system flight data throughout many missions that make it possible to use data-driven monitoring techniques such as IMS to monitor the health of the satellite. By using this technique, we gain all of the advantages IMS offers such as simplicity, adaptability, and efficiency. To be successfully integrated, the new monitoring system needs to meet a set of requirements to ensure that it doesn't negatively impact spacecraft operations, and can successfully run on a resource restricted CubeSat. The system must be fast and efficient and output anomaly reports in real time. It must have low resource consumption in terms of computing power and memory usage. It must also be generic so that it can be used in multiple missions with little change required. And most importantly it must be adaptable so that it can be updated as the mission progresses to provide the most accurate model of system behavior.

High Level Design

The new system meets these requirements and uses IMS's learning and monitoring algorithms to provide anomaly reporting capabilities. The high level design of this system consists of a learning phase which occurs on the ground, and monitoring which occurs on the spacecraft during its mission. The idea behind this separation is to perform the more resource intensive learning phase on a computer on the ground, and then load the knowledge base onto the spacecraft so that it can perform the less intensive monitoring. The monitoring returns a report on the health status of the system. The steps can be summarized as such:

1. Gather all nominal telemetry from archives
2. Select features to monitor
3. Run learning algorithm on selected features and archived data
4. Upload resulting knowledge base of clusters to the satellite
5. Run monitoring
6. Report health status

The model that is generated during the learning phase can be updated as the mission progresses and the spacecraft down-links relevant nominal telemetry. This new telemetry is added to the archive, and a new model is generated and the cycle can continue. This design in shown in Figure 2.

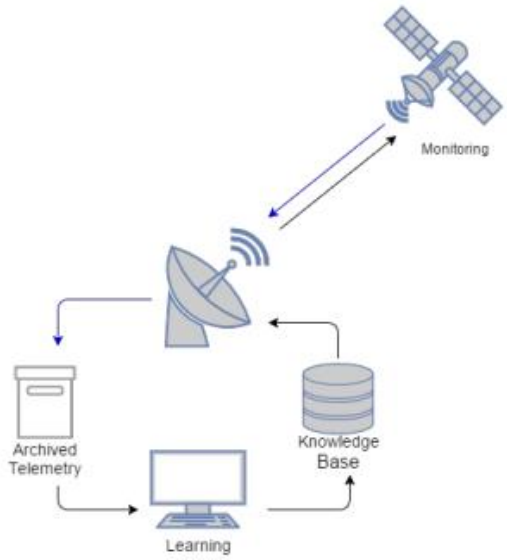

Figure 2: High Level system design

Reporting of the health status is done through the satellites beacon packet. The beacon packet is received by the ground station and the health status of the spacecraft can be determined. Additionally, the monitoring algorithm's deviation score and individual feature contributions can be saved on a file on board that can be down-linked for further analysis.

The final system consists of multiple IMS modules that define different feature sets to be monitored. Each module will be trained separately on the ground and run independently on the satellite. Each module belongs to a process on the satellite and the monitoring capabilities are provided as a library that the process uses.

IMPLEMENTATION

Learning Module

One of the first steps in this health monitoring application is to create the models that will represent nominal system behavior for the satellite. This is done using IMS learning and is performed on a local computer on the ground. The output of this phase is a set of cluster and configuration files that represent the knowledge base to be used by monitoring.

This module uses archived nominal flight data from the Intelligent Payload Experiment (IPEX) to create the models that represent nominal system behavior [4]. This training data set has been filtered of any outliers and bad readings so incorrect system behavior isn't captured. The data set consists of about 50,000 time series data points that were taken every 10 minutes and contain data for over 150 features including temperature, current, and voltage sensors.

Since IMS works best with small feature sets of highly correlated parameters, this data was broken up into separate modules that run independently on the satellite. To create the feature sets, a correlation matrix is used to find the pairwise correlation coefficient between each parameter in the data. This coefficient determines the strength of correlation between any two parameters. The result of the correlation matrix created for the IPEX data led to the creation of two main modules: one for temperature sensors, and another for power sensors.

Once these modules were created, the IMS learning algorithm was implemented to train the system against the data using the feature sets defined by the module. The IPEX data would be parsed to include only the selected features in the module, and the clustering algorithm would run to generate the clusters that define the model for each module. The resulting cluster file includes the clusters, along with some scaling information needed for the data normalization such as the means, standard deviations, and weights of each parameter. Along with the cluster file, a configuration file is provided that gives details about the module including the features that it consists of. These cluster and configuration files are then uploaded to the satellite as the knowledge base that monitoring uses.

Monitoring Library

The monitoring module that was created to run on the spacecraft was designed to fulfill all the requirements listed in the overview section. The code to perform the monitoring is compatible with the flight software and integrated in way that makes full use of the libraries and abstractions provided. The first instinct was to implement the health monitoring system as a new process on the spacecraft. This idea made sense because monitoring can be seen as its own entity in the modular design of the architecture, and would therefore fit well. It would interact with other processes to obtain information to do the monitoring and would exist in its own code-space. However, this design doesn't fit well with the requirement of making this system generic. The hardware components and complexity of each spacecraft differ from mission to mission and this would require frequent updates to the code of the process to make the system compatible to the new hardware of a different mission.

To better fit this requirement, the monitoring phase is provided as a new library that processes can import for monitoring. Through this design, each process can have a unique monitoring module running with its own parameters and feature sets. Also, the library can be easily and separately updated to coincide with any changes to hardware components without any major changes needed in the system process itself. The library

will initialize the data structure objects containing the information for monitoring, and have methods for checking anomalies and cleanup.

At a high level, the monitoring library adds scheduled events to each importing process so that the monitoring can run continuously. The configuration and cluster files are read to set up the data structures needed to perform monitoring. An anomaly checking event sends UDP commands to obtain the values for each feature in the module. Once all the data has arrived, the monitoring algorithm return a deviation score that is reported. Along with the deviation score, the individual feature contributions to the score are recorded so faulty sensors can be easily identified. This cycle continues throughout the mission life cycle.

The monitoring library also allows the models to be easily updated as the mission progresses. This update process consists of adding or changing a cluster/configuration file that exists in the file system on the satellite. It does this by creating a SSH session with the satellite and transferring the new files while the mission is in progress. When an update is detected, the library automatically adapts itself to use the new files that define a more comprehensive model of system behavior.

VALIDATION

The integration of the new Inductive Monitoring System into the PolySat flight software code base was thoroughly tested and validated to ensure that the system provides good results and doesn't waste any precious resources on the CubeSat.

Algorithm Validation

The first step in validating the system was to ensure that the IMS algorithms used generated the correct results. It was very important to test this because the success of the system is highly dependent on these to properly identify anomalies. The validation of the learning algorithm involved examining the clusters that were being generated from various data. Correct cluster formation and hyper-cube generation is vital for the correct modeling of the nominal operating regions of the satellite, and for the monitoring algorithm to provide accurate results.

To do this, multiple two dimensional data sets of clusters were generated and our clustering algorithm was run against this data set to see if it could properly identify the correct clusters. After creating several such data sets and running our algorithm, we saw good results that properly identified each cluster in the data.

The IMS monitoring algorithm was also tested to verify that it would correctly identify any anomalous results and produce a good deviation value. This algorithm takes input data vectors and uses the cluster knowledge base to predict whether or not the system is performing nominally. To validate the monitoring algorithm, a reference training and test data set with multiple features was used. The training set was used by the learning algorithm to create the cluster knowledge base. The test set was used to form input vectors that would be fed into the monitoring algorithm to produce a deviation score. A graph of the resulting deviation scores was compared against a validated graph that used the original IMS algorithm. The results of our run matched the ones in the validated set which gave us confidence that our algorithms work properly.

Experimental Tests on archived data

The next phase of testing was to utilize real CubeSat flight data with known anomalies to see if the system could identify the problems. Archived flight data from the IPEX mission was used for this purpose. The original IPEX flight dataset contained anomalies that occurred to the satellite during the mission. There were two major problems that the data reflected. To create a valid nominal training data set, the sensors that were contributing to these anomalies were manually corrected to reflect nominal behavior. This training set would be used for training the system and creating the knowledge base, while the original data set would be used as the test set for monitoring. The goal was to have the system identify the anomalous sensors. The two temperature and power models created by the correlation matrix were used to define the feature sets that were tested. To identify which sensors/features were causing the issue, the individual sum that each feature contributed to the overall deviation score was also stored. These modules were run separately and the deviation score along with the individual sums were written to a file to analyze.

The first run of the monitoring system against the IPEX flight data used the temperature model and produced the individual feature deviation sums seen on the left graph of Figure 3.

This graph reflects the overall sum of the error produced by each feature throughout the run and contains a partial set of the features. Looking at this graph, the ***boardpx*** sensor contributed to the most

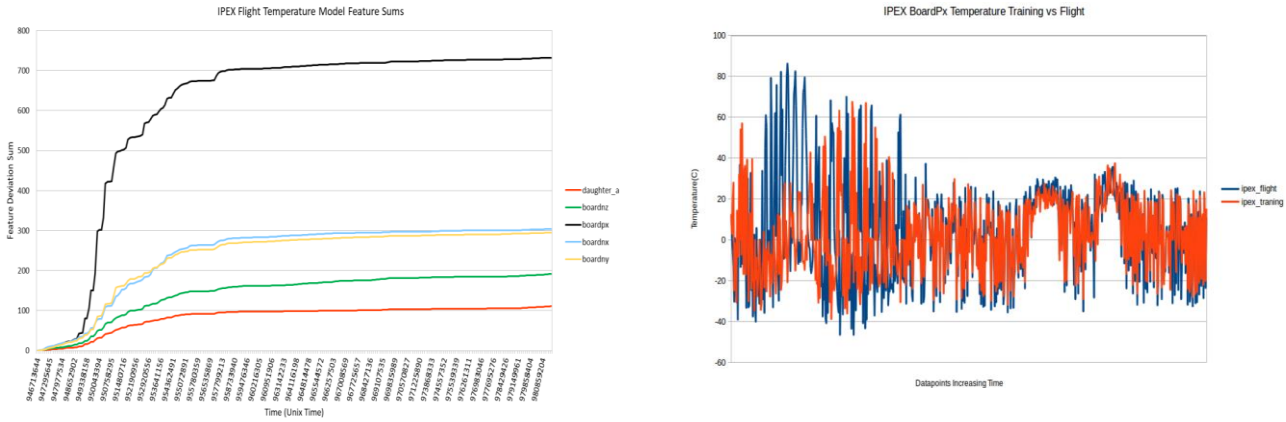

Figure 3: Results of running temperature model against archived data.

error. This sensor measures the temperature on one of the side panels of the satellite, in this case the positive X panel that is oriented in the x-axis of the satellite's reference frame. Error sums were also seen in the other side panel temperature sensors, but to a less extent.

Most of the contribution to the error sum was seen in the first quarter of the test set, with very little seen in the latter part. The graph on the right shows a line chart of the temperature data for the ***boardpx*** sensor that was contained in both the training and test set. The red line represents the temperature data for the training set while the blue line is for the test set. Looking at this chart, it can be seen that the temperature of the sensor in the flight set was higher than the seen anywhere in the training set, especially in the beginning. This matches with the large increase in the error sum we saw in the beginning of graph on the left.

After looking at these results, it was obvious that there was an anomaly on the ***boardpx*** side panel. There was in fact a problem with the side panels that occurred during the IPEX mission. The panels were found to not have good thermal conductivity and would get too hot when facing the sun. The positive X panel in particular would get very hot as reflected by the sensor readings. The other panels had a brass mass placed behind that absorbed most of this heat, and this is why those sensors contributed much less error. Over time, the thermal conductivity increased and more heat was absorbed as reflected by the flattening of the error sum. These results on the temperature model show that our system was able to successfully identify one of the anomalies that occurred during the IPEX mission.

The second run of the monitoring system against the IPEX flight data used the power model that contained most of the voltage and current sensors in the data. This model contained a sensor, ***threeV_volt,*** that was the source of an anomaly on the main system board's 3.3 voltage line. There was a hardware defect that caused the voltage on this line to be badly regulated which resulted in higher than expected voltages. The initial run of the model did in fact capture this anomaly indicating that the system correctly identified this problem. However, it also captured a false positive result that gave a nominally operating feature an anomalous result.

At first, the explanation for this behavior was a poorly defined model that did not capture the specific relationship between this sensor and the others which caused the system to produce bad deviation scores. However, after taking a look again at the correlation matrix, there was very weak correlation found between some of the features in the power model. There were a total of 14 features in this module including the power sensors for the solar panels. We decided to remove and separate some of the solar panel sensors that had weak correlation to another model. After running the monitoring again, we obtained the results in Figure 4.

The results in this chart look far better and only contain high error for the problematic **threeV_volt** sensor. This result highlights the need for well-formed and highly correlated features in a given model. Large models with a lot of features are good if you want to monitor a lot of sensors. However, they require much more training data to capture every possible behavior between each sensor in the set. If the models contain fewer, highly correlated features, then not as much training data is needed and the results are usually more accurate.

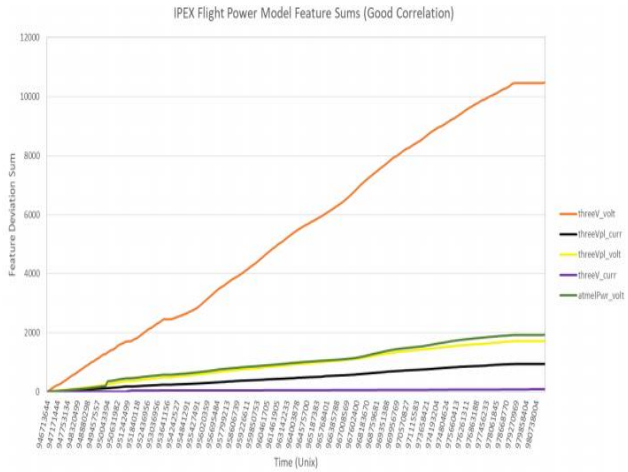

Figure 4: Individual feature contributions of running monitoring on Power model

Real-time CubeSat testing

Through previous testing we were able to conclude that given well-formed feature sets, our system worked successfully to identify any anomalies that occurred in the data. The final step involves testing the full IMS system that was integrated into the PolySat flight software to see if we could reliably use this system in future missions.

These tests were run on an experimental test unit for the Ionospheric Scintillation explorer (ISX) mission that is planned to launch in late 2017 [1]. During the time of testing, this test unit had most of its components assembled except the side panels which contain the solar panels. This meant that readings from all possible sensors were not possible, because some of the hardware wasn't connected.

For monitoring, two modules were created that used the feature sets in the temperature and power models. To train the model, the archived IPEX data was used. Since the side panels were not attached, the corresponding sensors were removed from the two modules. These modules were trained and the resulting configuration and cluster files were uploaded to the appropriate directories in the satellites file system. A temporary process was created in the flight software which initialized the monitoring library objects and set the event for the anomaly detection to run every five seconds.

The result of running the temperature module on the ISX test unit was very good. The system was run for several minutes and produced a deviation score of zero for every input data vector. Since the IPEX data contains temperatures seen in space, the ambient temperatures in the PolySat lab may have resulted in an anomalous result. However, somewhere in the training flight data, similar temperature conditions to our lab were captured and modeled in the knowledge base which led to nominal results.

The deviation scores produced by the power module, however, were too high indicating that it was picking up anomalies. There was one feature specifically, ***atmel_curr,*** that was causing the deviation score to be high. The individual feature sums for this run are shown in Figure 5. Looking at the sensor values for ***atmel_curr*** in the training data, we found that the average current draw from IPEX was about 35 mA. The standard deviation of this feature in the training data was also very low indicating that this value did not fluctuate much. Looking at the current readings from ISX, we found that the average current draw for this sensor was about 25 mA. This isn't much of a difference and isn't indicative of a major malfunction. This difference can be due to the fact that this is a test unit for a completely different mission, and may have different current draws from the processor. However since the training data does not capture this difference, it resulted in the system reporting a large anomaly for this sensor.

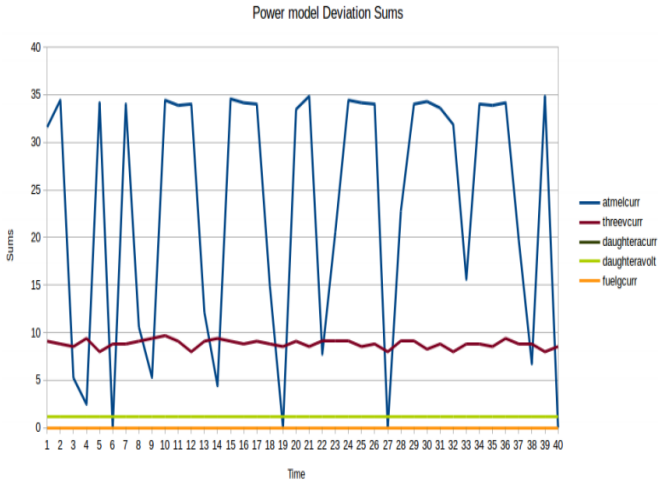

Figure 5: Deviation Sums of Real time run against Power Model.

This result once again shows the importance of a good, comprehensive training data set that contains as much component interaction behavior as possible. The more data in the training set, the better the algorithm can model the behavior of the system. This result also demonstrates one of the drawbacks of using this system. Not all missions are the same, and telemetry that corresponds to one mission may not be what is experienced by another. Therefore training data must be carefully selected and used only for missions and features that should experience similar behavior. After updating the model to include the new reading's

experienced by ISX, the system performed much better and we saw very low deviation scores.

CONCLUSION

The application of the Inductive Monitoring System for system health monitoring of CubeSat satellites has great potential. Such a system can be accurately and efficiently used to monitor for anomalies in a size and resource restricted CubeSat satellite. Archived telemetry can be used by data driven health monitoring techniques such as IMS to characterize models of nominal system behavior from the data itself instead of having to rely on more traditional parameter checking methods or more complicated model based techniques. As the amount of archived telemetry and data increases the more missions that are flown, these models can be updated to provide better and more accurate results that generate a more comprehensive model of the system's behavior.

The results of our tests indicate that the system performed very well in finding errors in archived flight telemetry and real time monitoring given good training models. The training data used must be comprehensive to include all possible system behavior. Since this requirement is somewhat impractical, the ability for the system to update itself given new flight telemetry allows for a more accurate representation of the system as the mission progresses. The importance of well correlated feature sets for monitoring was also examined so the system can perform at its best and have the least amount of false positives. The resource usage and efficiency of the integrated system was well within the limits of the resource constrained CubeSats, making such a system completely practical for use. The success of this system makes it perfect for use as an autonomous tool for system health monitoring in all future PolySat missions.

REFERENCES

1. Collaborative Research: Cubesat--Ionospheric Scintillation Explorer (ISX). (2016, October). Retrieved from National Science Foundation: https://www.nsf.gov/awardsearch/showAward?AWD_ID=1445468

2. Iverson, D. (2005). *Inductive System Health Monitoring With Statistical Metrics*. Moffet Field: NASA Ames Reseach Center.

3. Iverson, D., Martin, R., Schwabacher, M., Spirkovska, L., Taylor, W., Mackey, R., & Castle, J. P. (2009). General Purpose Data-Driven System Monitoring for Space Operations. *AIAA Infotech*. Seattle.

4. Kramer, H. J. (n.d.). *IPEX (Intelligent Payload Experiment) on CubeSat Mission*. Retrieved from eoportal.org: https://directory.eoportal.org/web/eoportal/satellite-missions/i/ipex

5. Manyak, G., & Bellardo, J. (2011). *PolySat's Next Generation Avionics Design*. San Luis Obispo: Digital Commons Cal Poly.

6. Singh, S. (2017). *A Data-Driven Approach to CubeSat Health Monitoring*. San Luis Obispo: Digital Commons Cal Poly.