Data Driven Framework for Prognostics

by

Gayathri Varadarajan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

December 2010

Data Driven Framework for

Prognostics

by

Gayathri Varadarajan

has been approved

October 2010

Graduate Supervisory Committee:

Huan Liu, Chair
Jieping Ye
Hasan Davalcu

ACCEPTED BY THE GRADUATE COLLEGE

ABSTRACT

Prognostics and health management (PHM) is a method that permits the reliability of a system to be evaluated in its actual application conditions. This work involved developing a robust system to determine the advent of failure.

Using the data from the PHM experiment, a model was developed to estimate the prognostic features and build a condition based system based on measured prognostics. To enable prognostics, a framework was developed to extract load parameters required for damage assessment from irregular time-load data. As a part of the methodology, a database engine was built to maintain and monitor the experimental data. This framework helps in significant reduction of the time-load data without compromising features that are essential for damage estimation. A failure precursor based approach was used for remaining life prognostics.

The developed system has a throughput of 4MB/sec with 90% latency within 100msec. This work hence provides an overview on Prognostic framework survey, Prognostics Framework architecture and design approach with a robust system implementation.

To my dearest family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Huan Liu for giving me the opportunity to work with him as part of building the prognostics framework that helps in determining the stability of the system. It was a great learning experience for the past two years working with Dr. Liu. I would like to thank him for accepting my interest and encouraging me to work on Prognostics project for my Master's thesis. I would also like to also thank Dr. Hasan Davulcu and Dr. Jieping Ye for the useful ideas and feedback they gave as part of my thesis committee.

I am grateful in general to the Data Mining and Machine Learning Laboratory (DMML Lab) and NASA for providing a research environment with right tools and data sets without which this thesis wouldn't have been possible. I would specifically like to thank my NASA mentor Edward Balaban for his guidance and all the lab mates in the DMML Lab who helped me in my thesis.

I would also like to thank all my family, friends and well wishers for all their encouragement and support.

TABLE OF CONTENTS

Page

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

Prognostics 2

Existing Prognostic Models 3

Key Terminologies 4

Thesis Organization 5

2 DATASET FOR PROGNOSTICS 6

Dataset from Battery Aging 6

Dataset from Composite Fatigue Tests 7

3 ARCHITECTURE OF A PROGNOSTIC SYSTEM 10

Prototype System Architecture 11

Challenges 12

System Middleware 12

Asynchronous Messaging 18

Database Design 20

Programmatic Schema Creation 22

Data Loading 23

Transaction Support 23

Data Distribution 24

CHAPTER Page

4 SYSTEM EXTENSION 26

Database Frontend UI 26

Algorithm Plugins 27

Data Visualization Support 27

5 EXPERIMENTAL RESULTS 28

Fatigue Tests and Numerical Validation 28

Data Reduction Methods on Battery Experimental Data 31

6 FUTURE WORK AND CONCLUSION 35

REFERENCES 36

APPENDIX

A PERL CODE FOR DEVELOPING THE DATABASE ENGINE . 38

LIST OF FIGURES

Figure		Page
1.	Aging Platform for Batteries	7
2.	Four-ply composite plate loaded in a MTS servo hydraulic test frame	8
3.	Echo-therm images at health and damaged states	8
4.	Unidirectional composite specimen	9
5.	Prognostic Framework	11
6.	Runtime Path Flow Through Application Middleware	16
7.	Data Flow Across the Network	17
8.	Message Ordering	19
9.	Publish - Subscribe Topology	20
10.	Structure of the Database Engine	21
11.	Data Distribution Scheme	25
12.	File Upload in Different Format for Feature Extraction	27
13.	Composite cruciform specimen and experimental setup	30
14.	Configuration of composite cruciform specimen	30
15.	Damage states and RUL prediction for composite specimen	30
16.	Comparison of predicted RULE and actual RULE	31

Chapter 1

INTRODUCTION

Condition health Management is the process of enabling a means of early warning of any predictable failure in the future to prevent from any serious loss or damage. By forcing a structured review of the consequences of each failure mode in terms of the above categories, it integrates the operational, environmental and safety objectives of the maintenance function. This helps to bring safety and the environment into the mainstream of maintenance management.

The consequence evaluation process also shifts emphasis away from the idea that all failures are bad and must be prevented. In so doing, it focuses attention on the maintenance activities which have most effect on the performance of the organization, and diverts energy away from those which have little or no effect. It also encourages us to think more broadly about different ways of managing failure, rather than to concentrate only on failure prevention. Failure management techniques are divided into two categories:

- Proactive tasks: these are tasks undertaken before a failure occurs, in order to prevent the item from getting into a failed state. They embrace what is traditionally known as 'predictive' and 'preventive' maintenance, although we will see later that RCM uses the terms scheduled restoration scheduled discard and on-condition maintenance
- Default actions: these deal with the failed state, and are chosen when it is not possible to identify an effective proactive task. Default actions include failure-finding, redesign and run-to-failure.

Prognostics

Prognostics is the process of predicting the future reliability of a product by assessing the extent of deviation or degradation of a product from its expected normal operating conditions. It is an emerging concept in condition based maintenance (CBM) of critical systems. Health monitoring is a process of measuring and recording the extent of deviation and degradation from a normal operating condition. Along with developing the fundamentals of being able to confidently predict Remaining Useful Life (RUL), the technology calls for fielded applications as it inches towards maturation. Currently, prognostics concepts lack standard definitions and suffer from ambiguous and inconsistent interpretations. This lack of standards is in part due to the varied end-user requirements for different applications, time scales, available information, domain dynamics and such others to name just a few issues. Instead, the research community has used a variety of definitions based largely on convenience with respect to their respective requirements.

The goal is to develop novel ways to identify anomalies and patterns within very large data sets containing multiple parameters both qualitative and quantitative and develop real-time reduced order modeling for failure prediction. Work in the areas of reliability modeling and prediction, pattern recognition, time series forecasting, machine learning, and fusion technologies is ongoing. The aim is to evaluate the use of intelligent reasoning technologies to model and manage the life cycle of electronic products. In this work some important concepts have

been defined using a notational framework that enables a coherent interpretation of prognostics in different applications.

Various forecasting applications have been categorized in several categories based on an extensive survey of domains like weather, finance and economics, nuclear power, medicine, automotive, electronics, and aerospace, where forecasting of events into the future holds significant payoffs both in safety and economic terms. With enormous focus on prognostic health management (PHM), prognostics is being redefined for safety critical systems from a CBM point of view and hence slight modifications are required to adapt from existing forecasting frameworks employed in other domains.

Existing Prognostic Models

Identifying the optimal fusion architecture and approach at each level is a vital factor in assuring that the realized system truly enhances health monitoring capabilities. A brief explanation of fusion architectures will be provided here. The centralized fusion architecture fuses multi-sensor data while it is still in its raw form. In the fusion center of this architecture, the data is aligned and correlated during the first stage. This means that the competitive or collaborative nature of the data is evaluated and acted upon immediately.

Theoretically, this is the most accurate way to fuse data, however, it has the disadvantage of forcing the fusion processor to manipulate a large amount of data. This is often impractical for real-time systems with a relatively large sensor network.

The autonomous fusion architecture quells most of the data management problems by placing feature extraction before the fusion process. The creation of features prior to the actual fusion process provides the significant advantage of reducing the dimensionality of the information to be processed. The main undesirable effect of pure autonomous fusion architecture is that the feature fusion may not be as accurate as in the case of raw data fusion because a significant portion of the raw signal has been eliminated.

Hybrid fusion architecture takes the best of both and is often considered the most practical because raw data and extracted features can be fused in addition to the ability to "tap" into the raw data if required by the fusion center.

Key Terminologies

DMS: Data management system is a administrative system that is used to manage the prognostics data in different format and file types. Using DMS, we can manage the work flow needed to collaboratively create, edit, review, index, search, publish and archive various and kinds of digital media and electronic text.

A DMS typically comprises of two elements: Data management application (DMA) and the Data monitoring application (DMA). Features of a DMS can vary, but most include window-based CRUD operation, format management, feature extraction, indexing, search, and retrieval and scheduled maintenance.

MySQL: MySQL is a open source Relational Database Management System provided under GNU public license.

Qt Jambi is a new and noteworthy alternative to Swing and Java 2D for developing rich, cross-platform desktop-application interfaces. RabbitMQ is being used for asynchronous message generation.

Thesis Organization

In this document, Chapter 1 provides the introduction into the focused work and explains Prognostics system, in a electronic System. Chapter 2 provides insight into the state of art existing methodologies in the prognostics domain. Chapter 3 describes about the architecture of the Prognostics System explaining the database and software design of this system. Chapter 4 mainly discusses about different experiments and validation approaches to prove the existence of a Prognostics system. Chapter 5 throws insight into the different ways by which the system can be extended in the future to increase performance and reliability of a prognostics system. Finally Chapter 6 concludes the work by summarizing and explaining the potential benefits of the prognostics health management.

Chapter 2

DATASET FOR PROGNOSTICS

Data for Prognostics is primarily contributed from two experiments. One experiment is done on a lithium-ion battery and another is done on composites. The data collection process focused on the parameters that are measurable and has tendency to degrade gradually.

Dataset From Battery Aging

The data used had been collected from second generation 18650-size Li-ion cells (i.e., Gen 2 cells) that were cycle-life tested at the Idaho National Laboratory. The cells were aged at 60% state-of-charge (SOC) and various temperatures (25^oC and 45^oC). We use the 25^oC data for training purposes and the 45^oC data for testing in order to determine the robustness of our approach to unknown operational conditions. Figure 1 shows a zoomed view of the shift in Electrochemical Impedance Spectroscopy (EIS) data of one of the test cells aged at 25^oC. Since the expected frequency plot of a resistance and a capacitance in parallel is a semicircle, we fit semicircular curves to the central sections of the data in a least-square sense, shown by black dashed lines in Figure 1. The left intercept of the semicircles give the R_E values while the diameters of the semicircles give the R_{CT} values. Time series of these values are the inputs to the prognostic routines while battery capacity is the predicted output. The end-of-life (EOL) of the batteries is defined to be 70% of rated capacity.

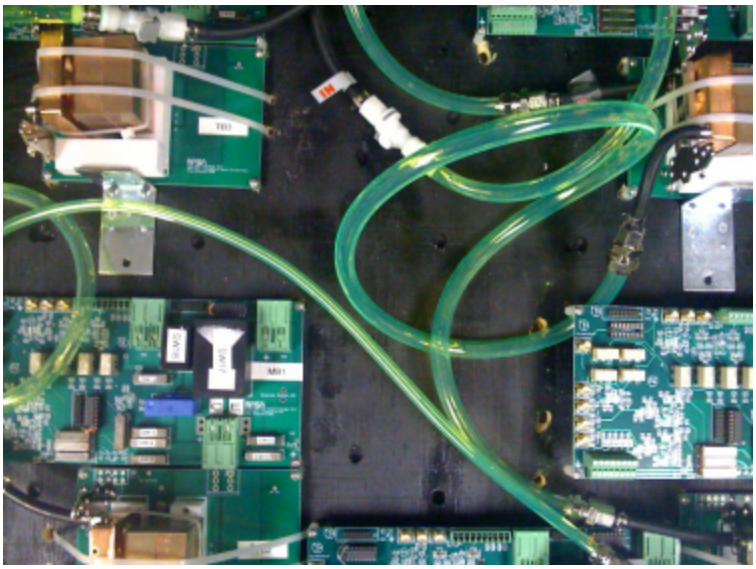


Figure 1. Aging Platform for Batteries

Dataset from Composite Fatigue Tests

The composite specimen loaded on the test frame can be seen in Figure 2. A four-ply composite beam was selected as the test specimen and was prepared with unidirectional carbon fiber/epoxy composite material. The matrix used was HEXION EPON 863 and EPI-CURE 3290. The specimen had a stacking sequence of $[0/90]_s$ and its dimensions are shown in Figure 3. Three strain gages were mounted on the surface of the specimen, two on the top side and one on the back side of the specimen, as shown in Figure 3. The specimen was subjected to constant amplitude fatigue loading with maximum amplitude (σ_max) of 11 k_N and load ratio R=0.1 on a MTS uni-axial fatigue frame operating at a frequency of 10 Hz. It is noted that a 0.75 inch wide notch in the center of the specimen was made to create an initial damage. An Omega OM2-163 strain gage amplifier was used to increase signal-to-noise ratio (SNR) of strain signals and a 48 channel NI PXI system was used to collect the strain gage

signals. In addition, the initial healthy state and final damaged state of the composite specimen was validated by using flash thermograph (Echo-Thermal) system. The flash thermo graphic images of the healthy and damaged states are shown in Figure 4. The developed real-time MATLAB based prognosis algorithm was synchronized with the NI data acquisition system (DAQ) to estimate the current damage states and to predict both the future damage states and the residual useful life.

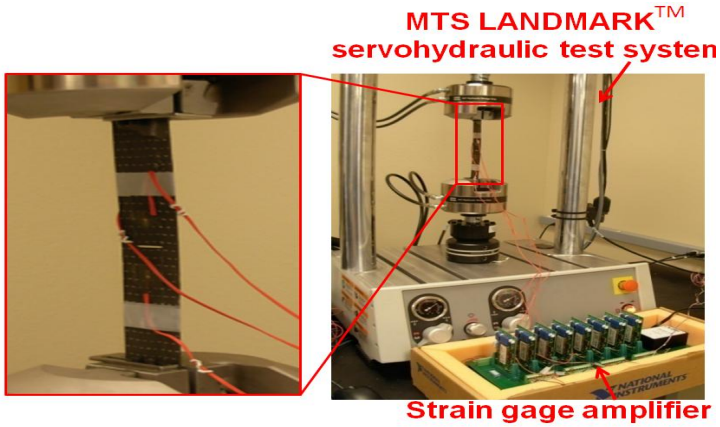


Figure 2. Four-ply composite plate loaded in a MTS servo hydraulic test frame.

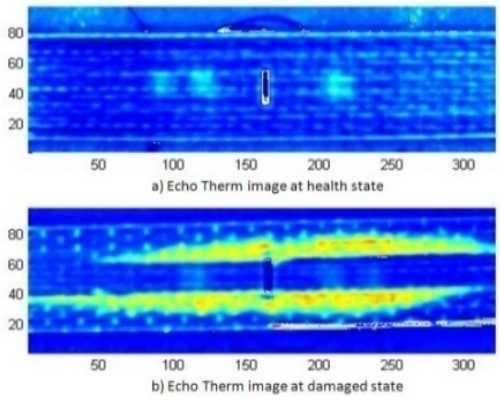
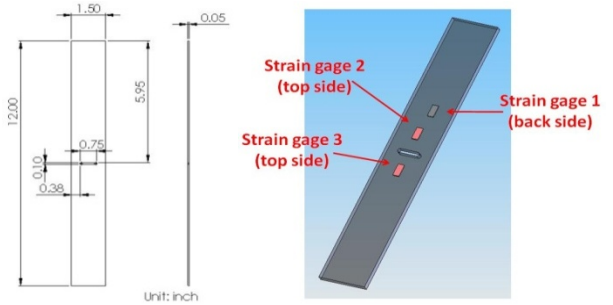


Figure 3. Echo-therm images at health and damaged states.



(a) Configuration of specimen (b) Test specimen with strain gages

Figure 4. Unidirectional composite specimen

Chapter 3

ARCHITECTURE OF A PROGNOSTIC SYSTEM

The existing metrics fail to capture two important aspects for prognostic applications. The first aspect is the timeliness of the predictions. A model that predicts a failure too early leads to non-optimal component use. On the other hand, if the failure prediction is too close to the actual failure then it becomes difficult to optimize the maintenance operation. To take timeliness of the predictions into account, the evaluation method needs to consider the delta time between the prediction of a failure and its actual failure (time-to failure). The second aspect relates to coverage of potential failures. Because the learned model classifies each report into one of two categories (replace component; do not replace component), a model might generate several alerts before the component is actually replaced. More alerts suggest a higher confidence in the prediction. However, we clearly prefer a model that generates at least one alert for most component failures over one that generates many alerts for just a few failures.

That is, the model's coverage is very important to minimizing unexpected failures. Given this, we need an overall scoring metric that considers alert distribution over the various failure cases.

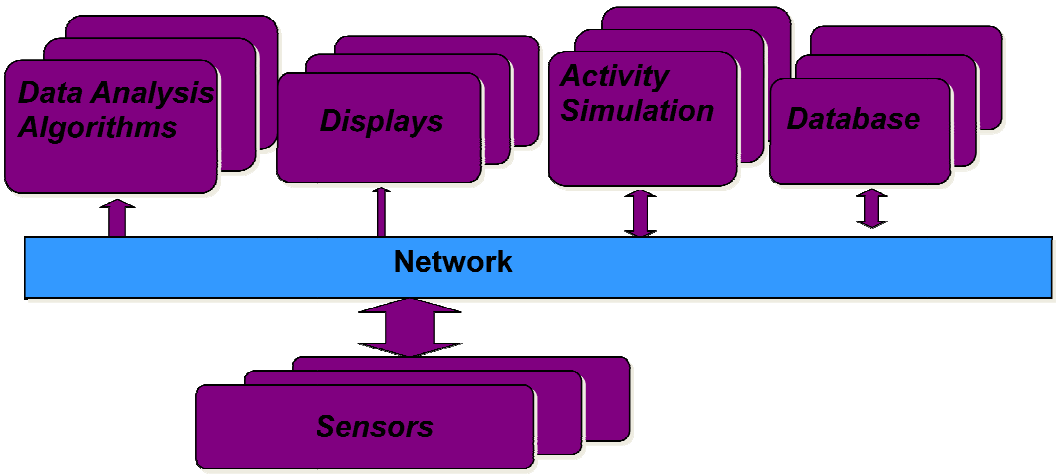


Figure 5. Prognostic Framework

Prototype System Architecture

The Prognostic framework collects information from the sensor or the Data acquisition system and relays it over the network. This data is then run through data reduction algorithms to extract relevant features suitable for prognostics and then formalize it into a robust stable system. This system not only stores data but also has the advantage of pluging in the right algorithms for different tasks. It also aids in providing a visual interface for viewing the system result.

Detection Threshold Metric measures an algorithms ability to identify anomalous operation associated with incipient faults with a specified confidence level. Confidence levels of 67% and 95%, corresponding to one and two standard deviations, are used to calculate the detection threshold metric. An algorithm that detects an incipient fault with high confidence will receive a high Overall

Confidence score, while an algorithm that does not report a fault until it becomes very severe would receive a low score.

Challenges

The design goal was to build a robust system that can store up to 10 MB/s with various record lengths. It should also be able to simulate the asynchronous message generation in the acquisition system. The system also must able to support multiple clients writing into different tables (up to 100 tables) with different record lengths(from 10 bytes to 10 MB, 1,000,000 times). All the data that was generated should be stored into the relational table using primary key with AUTO_INCREMENT option. The system should have the capability to retrieve the latest data using primary key, ORDER and LIMIT query parameters and store the latest retrieved data in the temporary derived table.

Apart from this the system must also have the search capability to find the record using all available fields only in the derived table. This was quite challenging considering the system should have asynchronous messaging system and the overall snappy system response.

System Middleware

Middleware systems are comprised of abstractions and services to facilitate the design, development, integration and deployment of distributed applications in heterogeneous networking environments.

The Communication Manager is a set of functions and data structures responsible for multiplexing the various methods of data requests and retrievals. All data is routed through a set of data structures known as data stores. These

stores are responsible for storing and accessing data. Specifically, the Communication Manager maintains functions to control and access the Local Data store, the Relative Data store, the Publish Data store, and finally the Group Data store. Each of these stores will be examined in more details. With respect to the functions that operate on such stores, the Communication Manager presents a uniform interface based on the communication API described earlier in this report.

One of the primary responsibilities of the Communication Manager is to multiplex the various requests from the Computation Manager and ensure that the Computation Manager receives the proper data regardless of where that data originated. For instance, with respect to data requests (collect operations), the Communication Manager must decide whether that request should come from the Local Data store or whether that request should be forwarded to the network. This decision is made by using information supplied by the Communication Manager. If the requesting function is a collective computation, then the request is forwarded to the Group Data manager. Otherwise, the Communication Manager contacts the Local Data store. The Communication Manager also provides functions to request relative data. In the case that the data is not available from the relevant store, the Communication Manager stores the pending request and informs the Computation Manager. This ensures that the Computation Manager is correctly notified when the data becomes available.

The Communication Manager also handles the messages coming from the network and the sensing hardware and properly informs the relevant stores.

Messages are generally characterized by a message session ID, the type of message, and the group involved. The session ID simply identifies messages that are replies to previous request messages. For instance, a request message to determine relative link quality will be associated with reply messages with the same session ID. The different types of network messages interacting with the Communication Manager may include: data sent from the group leader to be inserted into the Local Data store, requests sent by the group leader in the form of collect requests, data aggregated from a collect request, answering relative data requests, and receiving the data from a relative data request. Depending on the type of request, the appropriate data store is contacted along with the appropriate actions.

The Group Data store is responsible for collecting data for collective computation. Unlike node level computation, collect requests from collective computation requires the system runtime to create an appropriately formatted message that is sent to all group members. Once a sufficient number of group members have replied, the Group Data manager notifies the Communication Manager. The Group Data manager must ensure that networked messages are correctly routed to the appropriate application by dividing all the messages into groups similar to the Local Data store.

The Relative Data store manager is responsible for collecting and presenting relative data such as link quality to the Communication Manager. When the Communication Manager files a request for some relative data, the Relative Data manager creates the appropriate messages and relays those

messages between the appropriate neighbor nodes using Relative Groups manager. This is desirable since the Relative Groups manager knows how many hops it must take to retrieve the data and can verify that the collective computation belongs to a Relative Group. Upon receiving a relative data request, the Relative Data manager is handed the request and constructs an appropriate reply. This reply is then sent back to the network.

The Local Data store is the simplest type of data store. First, all data is partitioned into groups ensuring that different applications that belong in a specific group only access the relevant data. The data is also marked as whether belonging to the group or to the individual node. This is because the Local Data store must ensure that node-level computation should not access data meant to be manipulated by the collective computation possibly running on the same node. In the case that the Local Data store does not contain the requested data, the store manager is capable of interacting with the sensing interface to collect new sensor data. Otherwise if the data is not available from the sensing interface, the store manager simply waits for data to appear from the network.

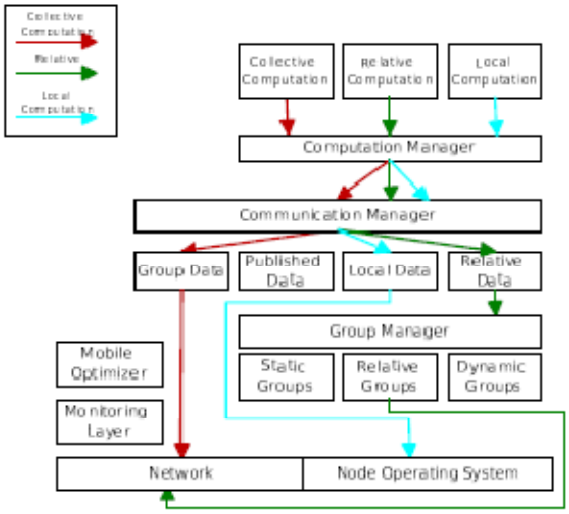


Figure 6. Runtime Path Flow through Application Middleware

The Published Data store is distinguished from the Local Data store by using the network to gather data from other nodes. While the Local Data is available to the application that requested the data, Published Data is only available to the collective computation. As such, although any node can publish data, only the leader of the group can actually access that data. The collective computation must issue a subscribe message for any published data it intends on collecting. This allows the Published Data manager to estimate how long a data item should be cached and allow it to make certain optimizations. The manager also keeps track of the number of collect calls issued from specific nodes over some range of time, allowing it to make further optimizations.

Unlike traditional network message structures, messages in the framework are never directly addressed to specific nodes. Instead messages are sent to specific groups. Since many of these groups only contain unique IDs with respect

to a larger parent group (dynamic groups) and groups can be hierarchical, the message structure must contain enough group IDs to identify the proper group. Also the message must specify whether the message is targeted for members of the group or the leader of the group. Finally, the message must contain a unique session ID to identify groups of related messages used for state-full communication.

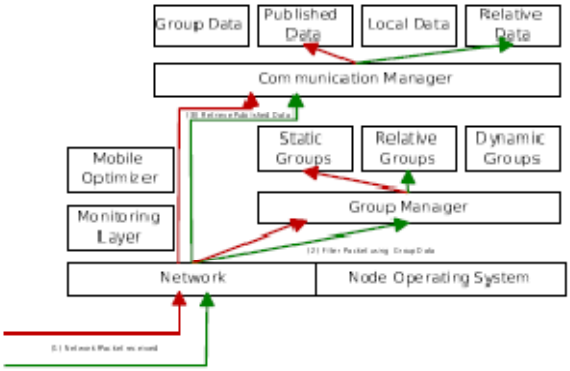


Figure 7. Data Flow across the Network

System allows users to specify collective operations on arbitrary groups without burdening the user of specifying the node in which the computation should run. This allows the system runtime to quickly adapt to the presence of new nodes and the failure of existing nodes without unduly burdening the user. An integral part of the architecture that allows this to occur is the Mobile Optimizer. Like the Offline Optimizer discussed earlier, the Mobile Optimizer attempts to conserve a particular groups energy by adjusting the placement of the collective operations within the group. However, unlike the Offline Optimizer, the Mobile Optimizer only has limited information and computation capabilities and such only attempts to move computation conservatively.

Currently, function migration may occur during the following scenarios:

- Load balancing - Because group leader exhibit different computation and communication patterns, it may be desirable to rotate the group leader amongst the nodes to conserve overall group energy.
- Introduction of new nodes - New nodes may feature different capabilities such as a powerful CPU or radio which can change the energy characteristics of the group.
- Deletion of nodes - The removal of nodes may introduce topology changes that force functionality to be moved around.
- Dynamic groups - Dynamic groups can be viewed as a group that occasionally introduces and removes nodes and as such demands adaptive behavior.

The Mobile Optimizer uses a lightweight monitoring tool to garner information about the state of the network group and uses that information along with data from the Communication Manager to make leader decisions. It is important to note that neither the Mobile Optimizer nor its associated monitoring tool needs to run on the group leader. The Mobile Optimizer must also communicate with the Group Manager and Computation Manager to adjust the location of the leader and its associated computation.

Asynchronous Messaging

In distributed computing, one of the harder problems is scalability. For instance, a web site may have a page where users enter an order to purchase an

item. The code behind the web page needs to validate and then deliver the order to a back-end system for further processing.

One of the main problems is that while waiting for a response, the thread of execution is suspended; as the number of orders increases, the number of threads that wait for a response from the Order-Server increases as well. Therefore, it would be more advantageous to send the order to the order service without needing to wait. At most, a confirmation that the order has been stored for processing may be required.

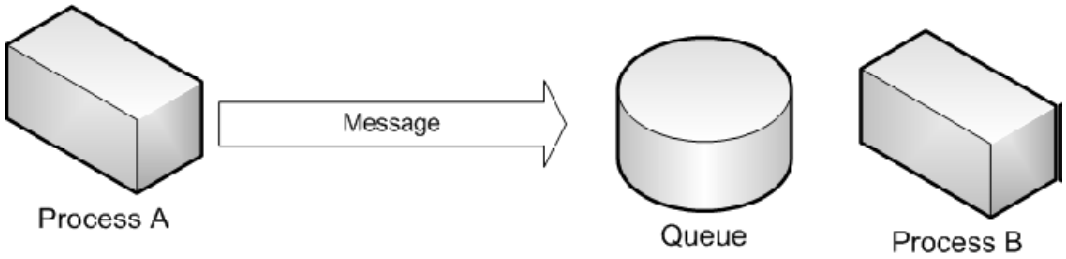


Figure 8. Message Ordering

In the order service example outlined above, there are two actors: a client sending orders and the order service that processed them. Also of interest is another asynchronous messaging scenario called publish-subscribe (or pub-sub, or even-ting.) With pub-sub, an actor sends a message to any number of potential subscribers asynchronously. In this example, the subscribers have subscribed to a topic the message is a part of. In some scenarios, the first subscriber, who has resources (such as processing time), will then process the message. In other scenarios, each subscriber will process the message entirely and separately (usually each subscriber has a different task to perform in such cases). Going back

to the order processing example, it is easy to imagine that a logging service will want to log each order before any kind of processing is done to it. This is an orthogonal operation to the actual order processing and can be performed in parallel. So, we can modify the messaging pattern used to pub-sub.

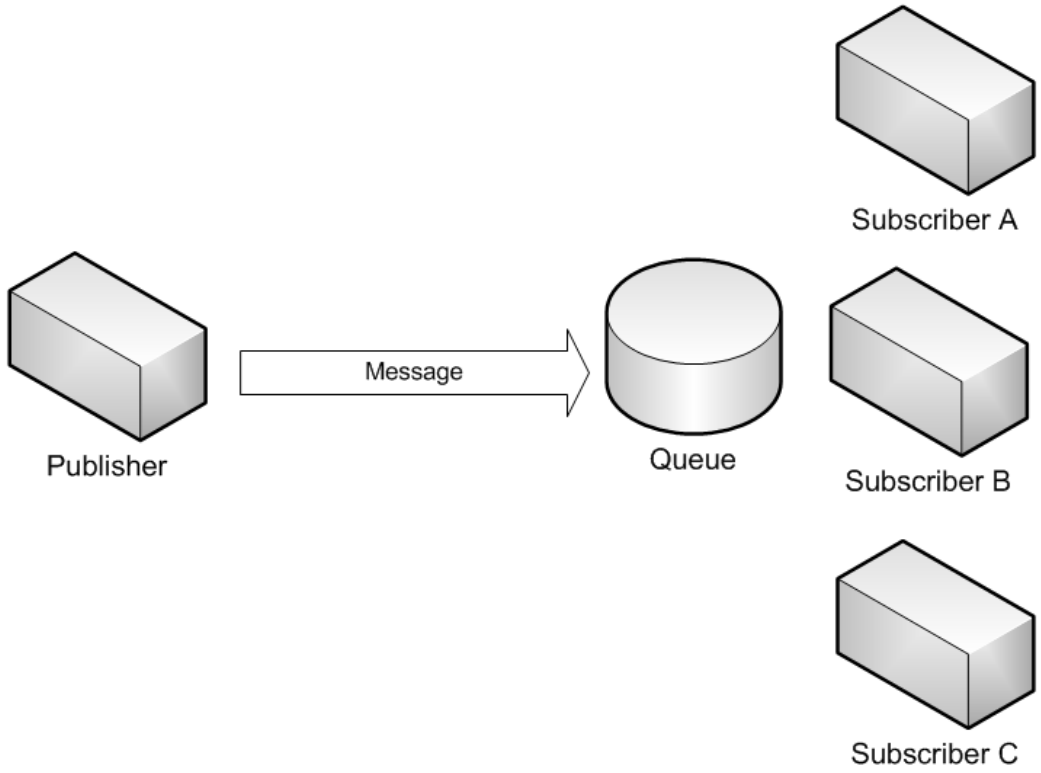


Figure 9. Publish – Subscribe Topology

Database Design

The default MySQL database for Diagnostics framework has been restructured in order to make the Prognostic system more efficient. A table named "TBI-Algorithm" was created to hold the algorithm that determines the data feature selection process of the given data. A table named "TBI-Features" was created to hold the features, their respective predefined component and related

metadata which play a key role in the Prognostics framework, which acts as the center of the condition maintenance system interface. The structure of the table is given in the figure 10.

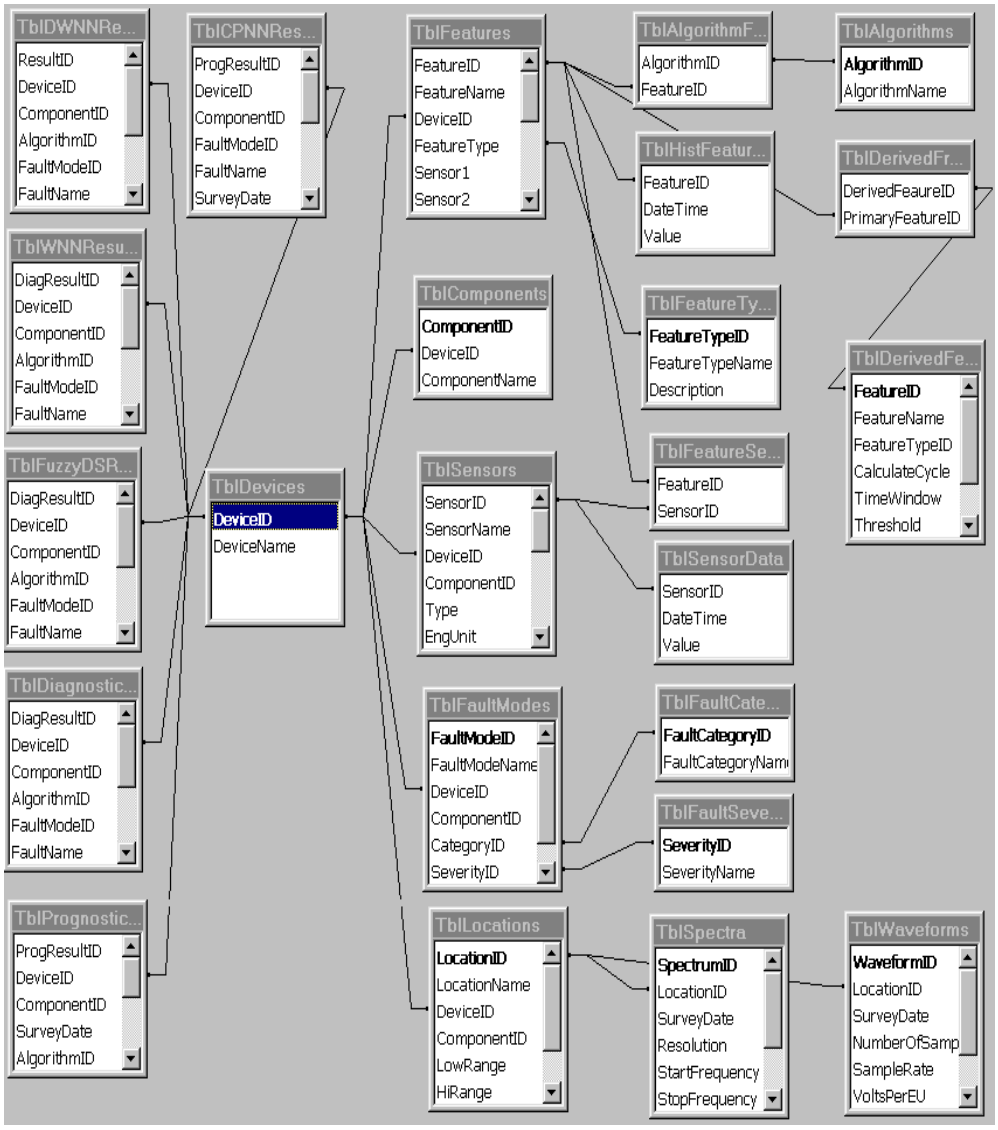


Figure 10. Structure of the Database Engine

Programmatic Schema Design

A capability of the SDE which was found to be crucial to the prototype is the SDE's ability to create all database entities through its API. The prototype system exploits this capability by performing the following through the SDE's API.

- Programmatic Layer Creation: Layers need not be created before a data import operation. All layer creation details are handled by the FME.
- Programmatic Attribute Creation: Layer attribute tables are created automatically, eliminating the error-prone task of manually defining attribute tables. Both optional and required attributes may be created.
- Programmatic Attribute Index Creation. Attribute indices can be specified within FME control files. These indices are used to enhance the performance of the non-spatial component of searches.
- Programmatic Layer and Attribute Verification: When loading data into an existing spatial database, the prototype verifies that the feature definitions specified in the control file match the existing SDE layer and attribute definitions.
- Feature Logging Support: An SDE log-file can be specified. This log-file, contains SDE feature identifiers of the features which are loaded into the SDE.

Making the data loader module responsible for all aspects of layer creation and verification radically reduced the amount of time required to prepare a new SDE database for use.

Data Loading

Once a preliminary data model had been decided upon, the focus of the prototype activity changed to developing a method of storing the data within the SDE. The SDE performs a great deal of geometric integrity checking when data is being loaded. This integrity checking ensures that data which is loaded into the SDE is valid. During the prototype development it was discovered early that occasional input features have invalid or unsuspected geometry. This results in the SDE rejecting the feature and the FME aborting the data load operation. The data load operation then had to be restarted after all features which were successfully loaded during the data load operation were first removed from the database. This was not an ideal situation and was resolved as described below.

Transaction Support

Initially, the data load process did not take advantage of the SDE's transaction model and thus when a data load operation was aborted it was difficult to recover in such a way that no features were skipped and no features were loaded into the SDE twice. The prototype was then upgraded to take advantage of the SDE transaction model. This enables the prototype to recover from erroneous data in a graceful and controlled manner. The impact to the prototype system was surprisingly small and required only 2 days of effort. The prototype system now performs a transaction commit after every 100 features and prints the transaction

number to an FME log file. If the data load operation is aborted because of bad data or other erroneous events the user simply corrects the problem(s)and reruns the data load operation with the last successful transaction specified. The FME ensures that the loading of features into the SDE begins at the correct point of the data load operation. No features are lost and none are duplicated.

Data Distribution

The prototype system enables data to be exported either through the use of pre-scanned queries or custom control files for those who have direct access.

The prototype incorporates a Data Distribution System (DDS) that users can directly interface with. The goal of this component was to allow users who are connected to the same network as the prototype (intranet or internet) to be able to perform adhoc queries on the database. The DDS module accepts the query and returns the requested data in the desired format. The prototype accomplishes this by using the FME's SDE Query-Factory module.

When interoperability layer of the prototype receives a query, it:

- Starts an SDE Session on the destination database of the query. It is thus possible to have a single interoperability layer sit on top of multiple SDE databases.

- Uses the SDE C API to construct a query and sends the query to the SDE.
- Each feature which is retrieved by the query is then passed on to the rest of the FME for output in the format requested by the query.

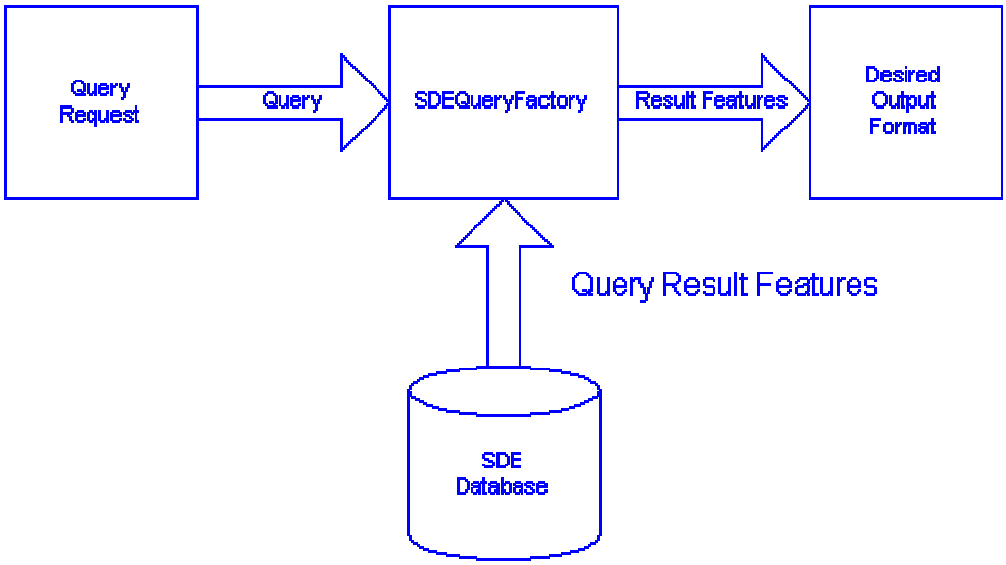


Figure 11. Data Distribution Scheme

The Data Distribution module of the prototype is driven by simple ASCII files, making it simple for other software modules to communicate with the prototype system. In the prototype system there are two such software modules. A text-based front-end is used by the public and replaces an older batch ordering system. The data distribution component of the prototype is one of the most interesting components of the prototype. It enables the prototype to answer a great number of queries very quickly.

Chapter 4

SYSTEM EXTENSION

The prototype system consists of two key pieces of technology. The first component, the SDE, forms the foundation upon which the rest of the system is built. The second component, the FME, provides an interoperability layer on top of the SDE. Using the SDE, it is possible to enhance the entire existing system by adding plug-ins for algorithmic execution and display of results.

Database Front End UI

The front end of the system, which is based on QT, is reengineered in each of its aforementioned different sections enabling the users to load all types of content with their own keywords. In some sections where they can upload their data, an Automatic data storing system has been implemented which involved in extracting data in different format in the titles of the uploaded files and identifying the words which can be associated to the data as features.

Major challenges were involved in finding the exact locations to edit the default Thousands of lines of QT code for the front end provided by NASA Diagnostic lab, in order to implement the auto storing system. Perl was used to enable scheduled maintenance of the data storage architecture.

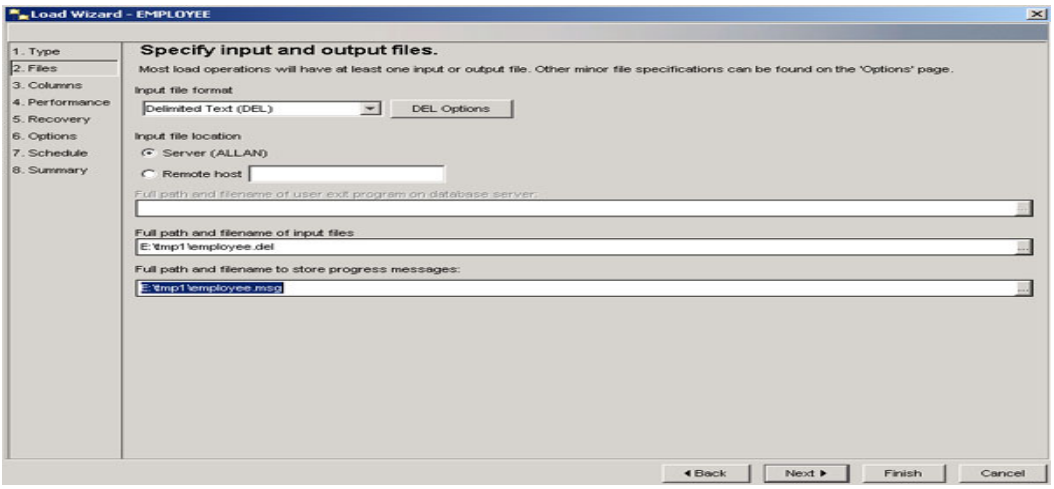


Figure 12. File Upload in Different Format for Feature Extraction

Algorithmic Plug-in

The system has the support to plug-in different algorithm for different task mentioned during the programmatic schema creation. This was enabled by the open application programming interfaces exposed to COM components by mat lab libraries to have a joint synchronization with the database front end GUI.

Data Visualization Support

Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. Automatic graph drawing has many important applications in software engineering, database and web design, networking, and in visual interfaces for many other domains.

Graph-viz is open source graph visualization software that has several main graph layout programs. This was used to provide visualization support to Prognostics framework.

Chapter 5

EXPERIMENTAL RESULTS

Goal of this task is to validate the developed methodologies using sophisticated simulations and real test-bend data from a set of representative focus problems and to construct various types of structural hot spots such as sensor integrated bolted joints, layered structures and scales composite wings using existing facilities. This would be used to validate the proposed methodology and its feasibility.

Fatigue Tests and Numerical Validation

The prognosis model developed is capable of predicting the future states and remaining useful life when composite beams were subjected to uni-axial fatigue loading. To validate the prognosis model for complex loading, another set of fatigue tests were performed by applying bi-axial loading on composite cruciform specimen. The composite cruciform and the experimental setup can be seen in Figure 13. To uniformly distribute the load in the web area of the cruciform specimen, two interface areas were machined by CNC machine at both front and back sides of the cruciform. The dimensions of the specimen are shown in Figure 14. The final failure of the cruciform specimen is the interface delaminating in the web area. The specimen was subjected to a constant amplitude fatigue loading with maximum amplitude (σ_{max}) of 5 kips and load ratio $R = 0.1$ at a frequency of 10 Hz. Based on uni-axial tensile tests with dog bone specimens, the yield stress is approximated as $\sigma_Y = 8$ kips. It should be noted that both the x-axis and y-axis actuator of the biaxial frame were subjected

to in-phase fatigue loading. For on-line state estimation, strain gage signals were used. One strain gage rosette is mounted in the web area of the cruciform specimen (gage area), and another strain gage rosette is mounted in the flange area. Similar to the uni-axial tensile fatigue test, the strain gage signals were collected by an Omega OM2-163 strain gage amplifier and a 48 channel NI PXI DAQ system.

The damage state prediction for the composite biaxial loading test is recursively completed with multi-step ahead prediction method. This is because feature information from the first several damage levels cannot capture the complex damage trend over the entire fatigue life. Three prediction series are shown in Figure 15. They each start from 164k cycles, 191k cycles and 218k cycles, respectively. The predicted damage states correlate well with the actual damage states. The residual useful life at a given damage level is estimated. Figure 16 shows the comparison of predicted RUL and actual RUL calculated from the 160k cycles to the failure of cruciform specimen. From the figure, it can be seen that there is a good correlation between predicted and actual RUL. Comparing predicted damage state and predicted RUL in Figure 15 and Figure 16, it is clear that both of the prediction accuracies are improved by using more on-line feature data because the two types of prediction are interrelated to each other.

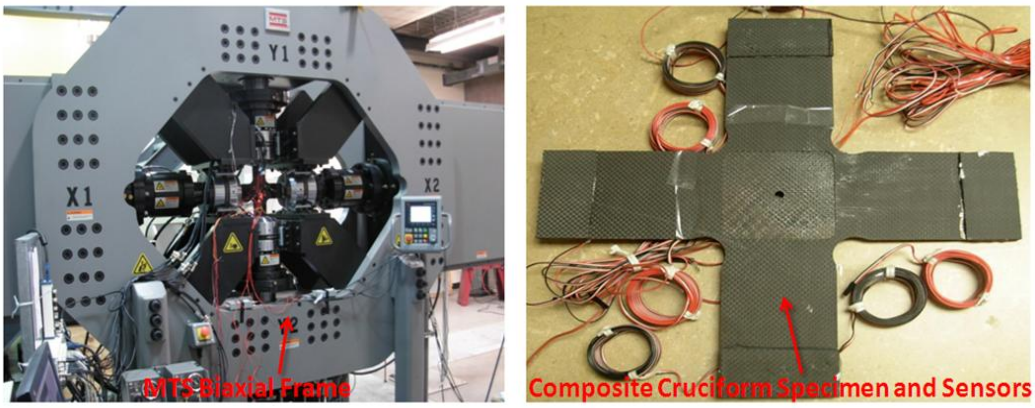


Figure 13. Composite cruciform specimen and experimental setup.

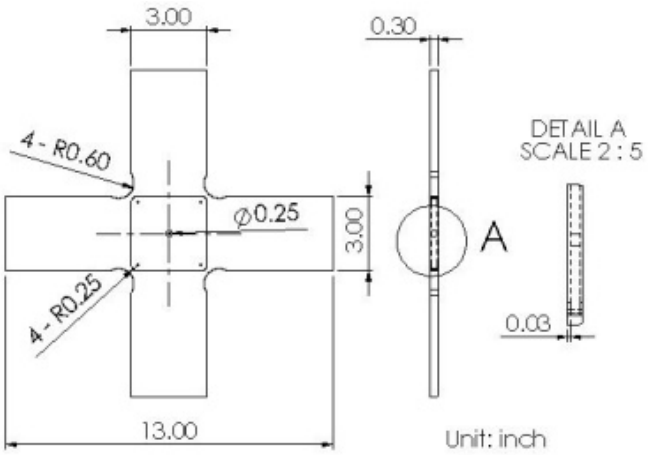


Figure 14. Configuration of composite cruciform specimen.

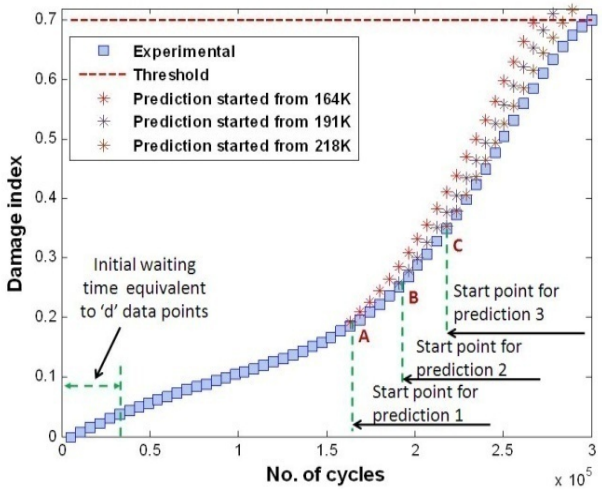


Figure 15. Damage states and RUL prediction for composite cruciform specimen.

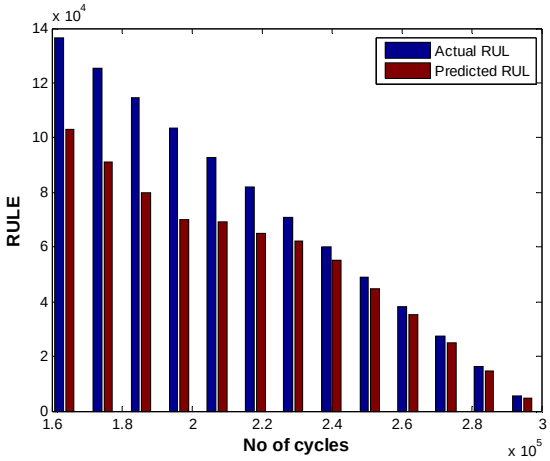


Figure 16. Comparison of predicted RULE and actual RULE

Data Reduction Method on Battery Experimental Data

Unsupervised learning is useful to create new features (dimensions) defined as functions over all features. It makes use of the data points instead of the class label to get new features. The goal is to project the data points from higher dimensions into lower dimensional space while preserving as much data as possible. This is usually done by choosing the projection that minimizes the squared error in reconstructing the original data.

Data reduction methods help achieve a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results. It helps to select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features. Our goal in this work is to explore current methods that are used to reduce the data containing useful features in battery experimental data for prognostics.

Principal Component Analysis

Principal Components Analysis (PCA) is the predominant linear dimensionality reduction technique, and it has been widely applied on datasets in all scientific domains, from the social sciences and economics, to biology and chemistry. Given N data vectors in m-dimensional space, the goal of principal component analysis is to find a f dimensional space such that $f \leq m$ that can be best used to represent data .The original data set is reduced to one consisting of N data vectors on f principal components (reduced dimensions) where each data vector is a linear combination of the f principal component vectors. PCA works best for numeric data when the number of dimensions is large. Assuming a data set of Y with n observations and m variables and is a $n \times m$ matrix; PCA divides Y into two matrices the scores matrix X with dimension $n \times f$ and the loading matrix P with dimension $m \times f$ plus a matrix of residuals$E$ with dimension$n \times m$.

We seek efficient i.e., polynomial in $m\ and\ n$, feature selection algorithms that identify in an unsupervised manner, a subset of exactly $f$out of m features such that if PCA is applied only on these k features, then the resulting embedding is close to the embedding that emerges when PCA is applied on all m features.

Non Negative Matrix Factorization Algorithm

NNMF is commonly used on numeric data whose number of dimensions is large. Given a data set $n \times m$ in matrix form with m samples in n-dimensional space, where each of entry is nonnegative, NNMF is used to find an

approximation as $V \approx WH$ where W is an $n \times d$ matrix and H a $d \times m$ matrix, m is the number of samples in the dataset and both W and H are also nonnegative. Usually d is chosen to be smaller than n or m so that W and H matrices have smaller order than$V$. Each data vector v is approximated by a linear combination of the columns of W, weighted by the components of h, where v and h correspond to the columns of V and H when the original matrix is re-written column by column as $v \approx Wh$. Each column of W can be considered as a basis vector that is optimized for the linear approximation of the data in V, and each column of H can be considered as a new feature vector corresponding to the original data.

PCA done on the battery dataset reveals that majority of the variance of the dataset can be represented just by using a fewer principal components. In the B0005, it was found that the majority of the variance can be represented using the first two Principal Components.

	PC1	**PC2**	**PC3**	**PC4**
Variance	86.5974%	11.2882%	2.0747%	0.0397%

In the composite dataset, there is not much help through PCA. The distribution of variance doesn't help to reduce the dataset column-wise using PCA as seen in the below table.

	PC1	**PC2**	**PC3**
Variance	65.9323%	17.4546%	16.4532%

A multiplicative update method based on a mean squared error objective function and alternating least square based algorithm was employed on the dataset on B0005 after 50 iterations. The table specifying the variance and root mean residual can be seen below.

Algorithm Used	**Final root mean square residual**
Multiplicative Update	0.4687814
Alternate Least Square	0.2697836

On the composite dataset, the following final root mean square residual value is obtained as follows,

Algorithm Used	**Final root mean square residual**
Multiplicative Update	0.51688776
Alternate Least Square	0.36798836

A multiplicative update method based on a mean squared error objective function and alternating least square based algorithm was employed on the composite dataset obtained from the fatigue test after initial data processing. Consistently in both the dataset the Alternate least square based algorithms performed well.

For nonnegative data, the result of Non Negative Matrix Factorization provides easier interpretation than that of PCA. From computation point, we can see that PCA essentially depends on empirical covariance matrix in which a large number of samples may be necessary. Thus for PCA, larger the samples, better will be the result.

Chapter 6

FUTURE WORK AND CONCLUSION

The system already supports inclusion of classic state of the art algorithm evaluation into the Prognostic framework. This can be extended to expose the application programming interface to make it more flexible for the researchers to plugin their custom built algorithm and tweak the existing support for the file format support. The Asynchronous messaging system can also be extended to let the researchers choose between getting timed alerts. Thus a very flexible yet powerful system can be developed if included these capabilities.

According to researches [7, 8] conducted, in a diagnostic system Research group, if a complete Prognostic system is properly implemented, it would help researchers get to the failure point looking for in a fast, easier manner to avoid the loss of data and valid system information. The work mentioned in this document is an effort to engineer and implement an efficient Prognostic system. One of the major aspects of this work involves different methods of storing the data collected by the researchers in an organized manner, where feature extraction play a very important role. Other major aspect involves the display of a data GUI which consists of all the features related to the results of the current extraction algorithm and the display of results in an organized manner based on the relevance and importance of the results. A Prognostic Data Base system implemented in this manner could help the researchers identify the feature they are looking for in an efficient way thereby satisfying the aim of the Condition Based Maintenance System.

REFERENCES

[1] Zhang, J., A. Papandreou-Suppappola, and R. Murray. "Estimation of Sparse LTV System Representation Using Compressive Sensing." Sensor, Signal and Information Processing Workshop. Sedona, AZ, 2008.

[2] Tichavsky, P., C. H. Muravchik, and A. Nehorai. "Posterior Cramer-Rao bounds for discrete-time nonlinear filtering." IEEE Transactions on Signal Processing 46 (1998): 1386-1396.

[3] Young, R. K. Wavelets Theory and Its Applications. Boston: Kluwer Academic Publishers, 1993.

[4] Candes, E. J., and T. Tao. "The Dantzig selector: Statistical estimation estimation." Annals of Statistics 35 (2007): 2313–2351.

[5] Candès, E., J. K. Romberg, and T. Tao. "Stable signal recovery from incomplete and inaccurate measurements." Communications on Pure and Applied Mathematics 59, no. 8 (2006): 1207-1223.

[6] Hurtado, M., T. Zhao, and A. Nehorai. "Adaptive polarized waveform design for target tracking based on sequential Bayesian inference." IEEE Transactions on Signal Processing 56 (March 2008): 1120-1133.

[7] Jolliffe, I.T. Principal Component Analysis. Springer, 2002.

[8] Jones, D., and T. Parks. "A high resolution data-adaptive time-frequency representation." IEEE Transactions on Acoustics, Speech, and Signal Processing 38, no. 12 (1990): 2127-2135.

[9] Lee, D and Seung, H. "Algorithms for non-negative matrix factorization." Advances in Neural Information processing Systems. MIT Press, 2001. 556-562.

[10] Liu, Y., S. Mohanty, A. Chattopadhyay, and J. Wei. "Damage Prognosis of Composite Specimens under Biaxial Loading." International Workshop on Structural Health Monitoring. Stanford, CA, 2009.

[11] MacKay, D. Introduction to Gaussian processes. Vol. 168, in Neural Networks and Machine Learning, edited by C. M. Bishop, 133-165. Berlin: Springer, 1998.

[12] Mallat, S. G., and Z. Zhang. "Matching pursuits with time-frequency dictionaries." IEEE Transactions on Signal Processing 41, no. 12 (Dec. 1993): 3397-3415.

[13] Mohanty, S., R. Teale, A. Chattopadhyay, P. Peralta, and C. Willhauck. "Mixed gaussian process and statespace." International Workshop on Structural Heath Monitoring. 2007. 1108–1115.

[14] Papandreou-Suppappola, A., ed. Applications in Time-Frequency Signal Processing. Boca Raton, Florida: CRC Press, 2002.

[15] Rasmussen, C., and C. Williams. Gaussian Processes for Machine Learning. Cambridge, MA: The MIT Press, 2006.

[16] Ristic, B., S. Arulampalam, and N. Gordon. Beyond the Kalman filter: particle filters for tracking applications. Boston: Artech House, 2004.

[17] Zhang, J., and A. Papandreou-Suppappola. "Compressive sensing and waveform design for the identification of Linear time-varying systems." Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. Las Vegas, NV, 2008. 3865-3868.

[18] Zhang, J., K. Narayan, and A. Papandreou-Suppappola. "On the use of compressive sensing in integrated vehicle health management." Annual conference of the prognostics and health management society. San Diego, CA, 2009.

[19] Zhang, J., W. Zhou, N. Kovvali, A. Papandreou-Suppappola, and A. Chattopadhyay. "On the use of the posterior Cramer-Rao lower bound for damage estimation in structural health management." 1st ASME Conference on Smart Materials, Adaptive Structures and Intelligent Systems. Oxnard, CA, 2009.

[20] Zhou, W., N. Kovvali, A. Papandreou-Suppappola, , P. Peralta, and A. Chattopadhyay. "Progressive Damage Estimation using Sequential Monte Carlo Techniques." 7th International Workshop on Structural Health Monitoring. Stanford, CA, 2009.

APPENDIX A

PERL CODE FOR PROGNOSTIC DATABASE ENGINE

```
// DATABASE ENGINE FOR PERL

Readonly my $benchdb => 'bench';
Readonly my $nrecs => 50000;
Readonly my $nquery => { # No of queries to run => No of iterations
   1 => 50000,
   3 => 50000,
   10 => 50000,
   30 => 20000,
   100 => 1000,
   300 => 1000,
   1000 => 100,
   3000 => 100,
   10000 => 20,
   30000 => 10,
   100000 => 5,
};

run_bench();

sub dbconnect {
   my $dbname = shift;
   my $dbh = DBI->connect
                  ("dbi:mysql:", "root", "", { PrintError => 1, RaiseError =>
                   AutoCommit })
                 or die "Failed to connect: " . $DBI::errstr;
  $dbh->do("USE $dbname") if $dbname;
   $dbh->do("SET NAMES utf8");
   return $dbh;
}

sub setup {
   my $dbh = dbconnect();

   $dbh->do("DROP DATABASE IF EXISTS $benchdb");
   $dbh->do("CREATE DATABASE $benchdb");
   $dbh->do("USE $benchdb");
   $dbh->do(q{
           CREATE TABLE test1 (
                                   id INT(11) auto_increment,
                                   data text character set utf8 collate utf8_bin NOT NULL,
                                   PRIMARY KEY  (`id`)
                                   ) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

```
COLLATE=utf8_unicode_ci
            });

    $dbh->begin_work;
    my $sth = $dbh->prepare_cached(q{
            INSERT INTO test1 (data) VALUES (?)
            });

    for (1 .. $nrecs) {
            $sth->execute("Record #" . $_);
            }
    $sth->finish;
    $dbh->commit;
}

sub use_single_selects {
    my ($dbh, $recs) = @_;

    my $sth = $dbh->prepare_cached(q{
            SELECT data FROM test1 WHERE id=?
            });
    for my $id (@$recs) {
            $sth->execute($id);
            my @row = $sth->fetchrow_array;
            $sth->finish;
    }
}
```