Assessment of Current Health and Remaining Useful Life of Hard Disk Drives

A Thesis Presented

by

Yogesh G. Bagul

to

The Department of Mechanical and Industrial Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Systems Engineering

Northeastern University
Boston, Massachusetts USA

January 2009
Northeastern University
Graduate School of Engineering

Thesis Title: Assessment of Current Health and Remaining Useful Life of Hard Disk Drives

Author: Yogesh G. Bagul

Department: Mechanical and Industrial Engineering Department

Approved for Thesis Requirement of the Master of Science Degree

Thesis Advisor, Dr. Sagar V. Kamarthi

Date

Thesis Advisor, Dr. Ibrahim Zeid

Date

Department Chair, Dr. Hameed Metghalchi

Date

Director of Graduate School, Dean Yaman Yener

Date
I would like to express my sincere gratitude to my thesis advisor, Dr. Ibrahim Zeid, for his guidance and support. I would also like to thank Dr. Sagar Kamarthi for his inspiration and guidance. I learned many things related to research from both of them. Their truly scientist intuition and an oasis of ideas always inspired and motivated me to fulfill their expectations. I would also like to thank the faculty and staff of Mechanical and Industrial Engineering Department for their timely support. I would also like to thank Tom Papadopoulos and James Jones for all the technical help they did. Also thanks to the Northeastern University for awarding me a Stipended Graduate Assistantship and providing me with the financial means to complete this project.

I cannot end without thanking my family, especially Lt Cdr Rahul (my brother), on whose constant encouragement and love I have relied throughout my time at the Northeastern University. My successful completion of MS thesis would not have been possible without their loving support, caring thoughts, superior guidance, and endless patience.

Yogesh G. Bagul
ABSTRACT

Everything that has a beginning also has an ending and so does a hard disk drive, which is a crucial subunit of a computer. The failure of a hard disk drive may cause serious data loss and inconvenience. In spite of hard disk drive being such a crucial subunit of a computer, limited research has been done on hard disk drive failure mechanisms, diagnostics, and prognostics. By knowing in advance an impending hard disk drive failure, can we not only avoid data loss but also minimize computer down time and wastage of time and money. Through the proposed prognostics technology we can also extend the length of hard disk drive usage and delay their replacement.

We extensively investigated different degradation signatures for a hard disk drive that characterize its aging and failure. We identified reported uncorrect, hardware ECC recovered and read write rate as effective degradation signatures. Next we develop a neural network model to assess the current health and estimate the remaining useful life of a hard disk drive. We collected 320,800 data points by conducting experiments on 13 hard disk drives in an accelerated degradation mode. We used more than half of these data points for computing the neural network parameters and the rest for evaluating the accuracy of model predictions. The overall prediction accuracy of the model was found to be around 88.51%. This means, we can assess the health of a hard disk drive correctly 88 times out of 100 instances.
# Table of Contents

**ACKNOWLEDGEMENT** ......................................................................................................................................................... iv  

**ABSTRACT**........................................................................................................................................................................ v  

**TABLE OF CONTENTS** ......................................................................................................................................................... vi  

**LIST OF FIGURES** ............................................................................................................................................................... x  

**LIST OF TABLES** ................................................................................................................................................................. xii

**CHAPTER 1** ............................................................................................................................................................................ 1

**INTRODUCTION**

1.1 OVERVIEW
1.2 PROBLEM STATEMENT
1.3 RESEARCH OBJECTIVES
1.4 MOTIVATION
1.5 THESIS OUTLINE

**CHAPTER 2** .......................................................................................................................................................................... 5

**AN OVERVIEW ON THE RUL ESTIMATION MODELS**

2.1 CLASSIFICATION OF RUL METHODOLOGIES
   2.1.1 Fourier Transform Analysis
   2.1.2 Neural Network
   2.1.3 Physical Model-based Approach
   2.1.4 Statistical Model-based Approach
   2.1.5 Cumulative Damage Model
   2.1.6 Nonlinear Dynamics

2.2 HYBRID RUL METHODOLOGIES
   2.2.1 Fourier Transform with Neural Network
   2.2.2 Statistical Model with Neural Network
   2.2.3 Fuzzy Logic with Neural Network
   2.2.4 Wavelet Transform Analysis with Statistical Models

2.3 COMPARISON METRIC OF RUL ESTIMATION METHODOLOGIES
2.4 IMPORTANT OBSERVATIONS FROM THE LITERATURE REVIEW

**CHAPTER 3** ........................................................................................................................................................................... 22

**PROPOSED MODEL FOR ASSESSMENT OF CURRENT HEALTH AND RUL**

3.1 IDENTIFY THE DEGRADATION SIGNATURE
3.2 PERFORM CONDITION MONITORING
3.3 EXTRACT FEATURES OF THE DEGRADATION SIGNALS
3.4 ASSESS THE HEALTH AND RUL
CHAPTER 4.............................................................................................................................. 26
ACCELERATED DEGRADATION OF THE HDDS
4.1 OVERVIEW OF ACCELERATED DEGRADATION OF COMPONENT/PRODUCTS
   4.1.1 Usage Rate Acceleration
   4.1.2 Overstress Acceleration
4.2 LITERATURE REVIEW OF ACCELERATED DEGRADATION OF HDDS
4.3 APPROACHES CHOSEN FOR ACCELERATED DEGRADATION OF THE HDDS IN THE LABORATORY
   4.3.1 Introducing Artificial Scratches
   4.3.2 High Operating Temperature
   4.3.3 Increment of Voltage
   4.3.4 High Usage
4.4 CHALLENGES FACED
   4.4.1 Random Read/Write Program
   4.4.2 Record a Log of Processes Running in the HDD
   4.4.3 Operating System

CHAPTER 5.............................................................................................................................. 37
UNI-AXIAL AND TRI-AXIAL VIBRATION SENSORS
5.1 IDENTIFY THE DEGRADATION SIGNATURE
5.2 PERFORM CONDITION MONITORING
   5.2.1 Sensor Selection
   5.2.2 Data Acquisition System (DAS)
   5.2.3 Sampling Frequency
   5.2.4 Sensor Location
   5.2.5 Procedure and Experiment Setup
5.3 EXTRACT FEATURES OF THE DEGRADATION SIGNALS
5.4 ASSESS HEALTH AND RUL
   5.4.1 ANN Training
   5.4.2 RUL Calculation
   5.4.3 Current Health Calculation
5.5 RESULTS AND DISCUSSION

CHAPTER 6.............................................................................................................................. 49
ACOUSTIC EMISSION SENSORS
6.1 REPLACEMENT OF VIBRATION SENSOR
6.2 IDENTIFY THE DEGRADATION SIGNATURE
6.3 PERFORM CONDITION MONITORING
   6.3.1 Sensor Selection
   6.3.2 Data Acquisition System (DAS)
   6.3.3 Sampling Frequency
   6.3.4 Sensor Location
   6.3.5 Experiment Setup
   6.3.6 Experiment Procedure
6.4 EXTRACT FEATURES OF THE DEGRADATION SIGNALS
6.5 DISCUSSION
CHAPTER 7 .............................................................................................................................. 58
SMART ATTRIBUTES
7.1 SMART INTRODUCTION
   7.1.1 History
   7.1.2 Attribute Value
   7.2.3 Attribute Value Threshold
7.2 IDENTIFY THE DEGRADATION SIGNATURE
7.3 PERFORM CONDITION MONITORING
   7.3.1 SMART Attributes Monitoring Tools
   7.3.2 Selection of a SMART Attributes Monitoring Tool
   7.3.3 Automate SMART Monitoring
7.4 EXTRACT FEATURES OF THE DEGRADATION SIGNALS
   7.4.1 Initial Trend Cruves
   7.4.2 Outside Oven HDD
   7.4.3 Final Feature Selection
7.5 ASSESS HEALTH AND RUL
   7.5.1 Pre-processing of the Input Data
   7.5.2 Calculation of the Expected Health and RUL
   7.5.3 Results
7.6 DISCUSSION

CHAPTER 8 .............................................................................................................................. 78
READ WRITE RATE MONITORING
8.1 HYPOTHESIS
8.2 IDENTIFY THE DEGRADATION SIGNATURE
8.3 PERFORM CONDITION MONITORING
   8.3.1 Windows Task Scheduler
   8.3.2 Read Write Rate Monitoring
8.4 FILTERING (FEATURE EXTRACTION) IMPORTANT DATA FROM THE DEGRADATION
   SIGNATURE
8.5 CONCLUSION

CHAPTER 9 .............................................................................................................................. 85
ASSESSMENT OF CURRENT HEALTH AND RUL OF THE HDDS
9.1 IDENTIFY DEGRADATION SIGNATURE
9.2 PERFORM CONDITION MONITORING
9.3 FEATURE SELECTION
9.4 ASSESS HEALTH AND RUL
   9.4.1 Criterion for selecting the training HDD for a ANN
   9.4.2 Training of the ANNs
   9.4.3 Calculate the Current Health and the RUL
9.5 RESULTS
   9.5.1 Calculate the Error Percentage
   9.5.2 Final Results
LIST OF FIGURES

Fig. 2.1 - Classification of RUL estimation models

Fig. 2.2 - A simple neural network

Fig. 2.3 - Vibration signal from thrust bearing

Fig. 2.4 - Wear modeling architecture

Fig. 3.1 - Proposed model for assessment of health and RUL

Fig. 3.2 - Nature of expected degradation signature curves

Fig. 4.1 - Expected health degradation curve for scratch experiment

Fig. 4.2 - Microscopic observation of a head of a HDD

Fig. 5.1 - HDD condition monitoring setup

Fig. 5.2 - Experiment setup showing the oven with target HDDs, 6 systems, 1 parallel to serial hub and a single monitor

Fig. 5.3 - Zoomed in view of specially designed oven with HDDs in fixed slots

Fig. 5.4 - ANNs training and testing

Fig. 5.5 - Change in RMS value of the vibration signal

Fig. 5.6 - Band energies of a HDD

Fig. 6.1 - RMS value of AE signal for a PICO sensor

Fig. 6.2 - RMS value of AE signal for a UT sensor

Fig. 6.3 - 80-100 KHz band energy curve for the UT sensor

Fig. 7.1 - SMART attribute values from “smartmontools”

Fig. 7.2 - Hardware ECC recovered graph for first 50 hours

Fig. 7.3 - Seek error rate attribute graph for first 50 hours

Fig. 7.4 - Comparison of seek error rate attribute curve
Fig. 7.5 - Curves for selected attributes

Fig. 7.6 - ANN results for health and RUL of the HDDs

Fig. 8.1 - Read write rate monitoring setup

Fig. 8.2 - Read write rate graph

Fig. 9.1 - Final experiment setup

Fig. 9.2 - ANN to assess health and RUL of a HDD

Fig. 9.3 - RUL for different health states
LIST OF TABLES

Table 1 - RUL estimation methodologies comparison metric

Table 2 - AE sensor specifications

Table 3 - Comparison of SMART monitoring tools

Table 4 - Slots for the training HDDs

Table 5 - Bands for different health states

Table 6 - Accuracy of ANN prediction for the remainder HDDs
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Hard disk drives (HDD) replaced traditional floppy disks as a storage medium due to two reasons: (a) HDDs are rigid and last more than flexible floppy drives and (b) the tight tolerances allowed HDDs not only store more data but also higher read/write speeds than floppy drives. But the industry requirements (capacity, speed and price) kept rising and the scientist ended up in making HDDs more compact and fast. HDDs became one of the most important subunits of a computer system as HDDs played a significant role in following ways:

(a) Data Storage: A HDD can store changing digital information in a relatively permanent form. Again a bigger capacity HDD lets one to store more data and programs, thus reducing the need of an additional data storage media.

(b) Performance: HDD performance is an important aspect because the HDDs are one of the slowest internal components of the computer system, and therefore often limits the performance of the system as a whole. The speed at which a computer system boots up and programs load is directly related to HDD speed. The performance of the HDD is also critical for multitasking and processing large amounts of data such as graphics work, editing sound and video, or working with databases.

(c) Reliability: One of the methods to assess the importance of an item of hardware is to consider how much trouble is experienced if it fails. By this standard, the hard disk is
the most important component in a computer system and hence it must be highly reliable.

(d) Software support: The complexity of the softwares being used is increasing day by day. So it is imperative for the HDDs to support such softwares, though not all at a time but at least few required ones.

1.2 PROBLEM STATEMENT

Any damage to a HDD can lead to the loss of the data or poor HDD performance. In spite of HDD being such a critical subunit of a computer, there is a lack of literature on assessment of health and remaining useful life (RUL) of the HDDs. There exist several studies done on HDDs reliability [1-7] which gives a sound base for estimating the RUL of the HDDs. By predicting the moment when a HDD is going to die, not only the invaluable data can be saved but also adverse side effects can be avoided. It will also save resources being wasted and increase overall efficiency of work. This indicates a need of a model to assess the current health and RUL of the HDDs so that some precautionary measures can be taken at the appropriate time.

1.3 RESEARCH OBJECTIVES

Data loss is always a big problem in today’s industry and the prime reason behind that is the HDD failure. Though there are numerous data recovery softwares in the market, neither are they 100% reliable nor are they successful always. As prevention is better than cure, it is always better to protect the data than putting efforts in recovering it later on. Keeping this as a focal point, this thesis has the following research objectives.
(a) Investigate a degradation signature for the HDDs and subsequently assess the current health and the RUL

(b) Develop a model to assess current health and RUL of the HDDs

1.4 MOTIVATION

Businesses of all sizes are becoming more and more reliant on the electronic data, from client accounts, financial and product information to legal documents, designs and plans, and multimedia content. The volume of this data is exploding, and its value increases daily. Backing up this data ends up in duplicate data which can be used later on only if main data repository fails. In contrast, life may become easier if there is a system which will alarm before the main data repository fails. This thesis is primarily attempts to develop such a system. Through this thesis, we develop a prototype which assesses the current health and the time (RUL) when the main data repository is going to fail. After knowing the RUL, required maintenance actions can be planned and the invaluable data can be saved without any need of backup data repositories.

In addition to that, through the RUL estimation, the HDDs can be used over their entire life thus eliminating the need for preventive maintenance. This will increase the percentage life usage of the HDDs as well as the decrease or delay the expense of the new HDDs.

So investigating a scientific way to make life easy and to save money creates a motivation to develop the health and RUL assessment model for the HDDs.
1.5 Thesis Outline

This thesis is outlined as follows:

(a) Chapter 2 gives an overview on the RUL estimation models.

(b) Chapter 3 describes the proposed model for the assessment of the current health and the RUL of the HDDs.

(c) Chapter 4 describes the accelerated degradation process adopted for the HDDs.

(d) Chapter 5 and 6 show a failure of uni-axial and tri-axial vibration sensors and acoustic emission sensors to identify degradation signal for the HDDs.

(e) Chapter 7 and 8 describe how SMART attributes and read write rate monitoring satisfies the requirements of the degradation signature.

(f) Chapter 9 describes the fusion of smart attributes with read write rate to assess the current health and the RUL of the HDDs.

(g) Chapter 10 concludes the research work and gives the directions for further research.
CHAPTER 2

AN OVERVIEW OF THE RUL ESTIMATION MODELS

Following the basic steps prescribed for the research, a thorough literature review was performed. A considerable amount of work has been done in the prognostics and health management area. This chapter summarizes a review of this work which provided a solid foundation for the research work. Initially, the RUL estimation models were categorized. While doing so, a few hybrid RUL estimation models were also observed in the literature. On the basis of this literature review, a comparison metric was prepared for the commonly used RUL estimation models. A few important findings while performing this literature review are sighted in the last section of the chapter.

2.1 CLASSIFICATION OF RUL ESTIMATION MODELS

Various RUL estimation methodologies and their practical applications have been proposed in the literature [8-63]. Figure 2.1 gives a detailed classification of all these methodologies. Data driven methodologies are useful when a large quantity of noisy data needs to be transformed into a piece of logical information which further can be used to estimate the RUL [8, 9]. The accuracy of the RUL estimation highly depends on the quantity and quality of the data. In contrast, model-based methodologies depend on the fundamental understanding of the physics-of-failure of the system. As models contain functional mapping of the system parameters and behave accordingly, the accuracy of the model-based methodologies is substantially higher than that of the data driven methodologies [8].
However, it is feasible to use model-based methodologies only in selective cases, especially when the historical data of the system is not available.

2.1.1 Fourier Transform Analysis

When it comes to defect detection and RUL prediction through a signal analysis, frequency domain is the most sought option. Fourier analysis decomposes a function in terms of a sum of sinusoidal functions of different frequencies that can be recombined to obtain the original function. So the Fourier analysis decomposes a signal into different frequencies. Individual components can be ascertained as each type of fault has its own characteristic signature in the frequency spectrum. After extraction of such features, many irrelevant features are also present in the feature vector. So to obtain the relevant features indicative of the product performance, feature subset selection is employed. This selected feature vector is then used as an input to the classifier which ultimately estimates the RUL.

Scanlon, et al.[11] used sound pressure measurements for developing an automated degradation analysis system to estimate the RUL of the rolling element bearings. By using sufficiently high frequency resolution, additional relevant information required for effective automated monitoring was uncovered. A novel information theoretic approach was used to avoid increased computational burden demanded by the resultant large feature dimensionality. Acoustic data from the sensor signal was split and the Fast Fourier Transform was applied to deduce an estimate of the short-term, time-localized frequency content of the acoustic signal. Mutual Information (MI) was applied as a
Fig. 2.1 - Classification of RUL estimation models
feature subset selection to filter out irrelevant features. Class 1 through 10 were defined such that 1 refers to 100% RUL and 10 to 10% RUL. The RUL of the rolling element bearings was thus estimated based on the MI calculated through the feature subset selection.

Advantages: Full power of the resource reaches the detector with great resolution.

Disadvantages: Fourier Transform cannot produce good results without a suitable window [58].

2.1.2 Neural Network

Neural network is a strategy to investigate complex objects without any knowledge or assumptions about the internal make-up, structure or parts of an engineered system. This method targets either a formal description of the transformation rules linking inputs and outputs or the construction of a prototype exhibiting a behavior that approximates the engineered system [59] Figure 2.2 shows a configuration of a simplified 2-layer neural network.

![A simple neural network](image)

**Fig. 2.2 - A simple neural network**
As shown in Figure 2.2, a neural network processes information based on the neural structure. It receives a number of inputs either from original data, or from the output of other neurons in the neural network. Each input comes via a connection that has a strength (or weight); these weights correspond to synaptic efficacy in a biological neuron. Each neuron also has a single threshold value. The weighted sum of the inputs is formed, and the threshold is subtracted to compose the activation of the neurons. The activation signal is passed through an activation (or transfer) function to produce a single output.

Neural networks [8, 13-17, 60] have been used to develop the RUL methodologies. Shao and Nezu [13] proposed a new concept termed as progression-based prediction of remaining life (PPRL) to estimate the RUL of a bearing. This new concept manipulated the variables determined from online measurements via a compound model of a neural network. The steps of the PPRL concept included (i) applying different logistic rules to check the bearing state at the initial running stage; (ii) dividing the prediction work into general trend prediction and RUL prediction and building the prediction model in real time using the moving-window method; and (iii) obtaining the boundary of the RUL.

In the experiment, Shao and Nezu created 3-layer back-propagation neural network which contains 8 input neurons, 16 neurons in each hidden layers and one output neuron with approximation accuracy of 0.001. The patterns received while extracting features are used to train the neural network using a back-propagation learning algorithm. Vibration and temperature sensor values were given as inputs to the model and trend variables were determined to update prediction. In the end, if the predicted RUL is not as much as expected, a warning message is produced with the output.
Gebraeel, et al. [8, 60] developed a neural network to estimate the RUL of the rolling element bearings by monitoring its vibrations. Bearing life was divided into two phases, non-defective and defective phase. Defective phase was characterized by a distinctive frequency called as defective frequency which was a function of number of balls, the rotational speed, overall geometry and types of defects and their location. This frequency was getting excited with the birth of a defect and type of the defect. In the experiments, 92Hz was figured out as the defective frequency for the thrust ball bearings. This defective frequency along with its first 6 harmonics was used as input to a set of neural networks. 25 bearings were used to train 25 neural networks to give operating time (time since first defect) as output. By comparing each neural network output with observed operating time, weights for each neural network were calculated. These weights were subsequently used to calculate weighted average operating time of each validation bearing. The degradation signal was modeled as exponential function as shown in Figure 2.3 and the RUL was calculated.

![Prediction Using Exponential Parameters](image)

**Fig. 2.3 - Vibration signal from thrust bearing (modified from [61])**

Huang, et al. [14] exemplified a method to predict a ball bearing's RUL based on self-organizing map (SOM) and back propagation neural network methods.
Byington, et al.[15] developed a data-driven neural network methodology for remaining life predictions for aircraft actuator components. This methodology was verified by utilizing the command/response signal and hydraulic pressure data from F/A-18 stabilizer electro-hydraulic servo valves to estimate the RUL.

Advantages – Neural networks are very sophisticated modeling techniques capable of modeling extremely complex functions and hence powerful for prediction of the RUL. The other advantages are ease of use (as it learns by examples), robustness, generalization and model free RUL estimation[62].

Disadvantages – Neural networks are slow, especially in training phase.

2.1.3 Physical Model-based Approach

Physical models [18-24] are created to represent the engineered systems so that model parameters can be easily associated with the system parameters to ensure correct behavior of the model by defining how the failure modes and mechanisms occur in the real life. As these models allow simulation of the behavior of their corresponding systems, current outputs of the model can be used as the future input values to predict the future health status of the system.

To create a physical model, one has to start with the physical equations. Based on the intended goal of a model, critical parameters are selected. Some approximations can be made to trade off realism for simplification. Then the equations are solved using some numerical methods to carry out the integrations and other mathematical operations. If required at some
point, simulation of the system can be done and the outputs (of simulation) can be incorporated into the models for further calculations.

Watson, et al. [18] presented a model-based technique for the RUL estimation of highly dynamic, high-power dry clutch system by combining physics-based simulation model and wear-prediction model. A dynamic-clutch model was developed based on four distinct operating regimes: fully disengaged, engaging, fully engaged, and disengaging. The model also included a control system to maneuver the clutch. The model also included thermal considerations for heat transfer during operation and chemical wears. Then the double exponential smoothing (DXS) method was used for forecasting future values of a vector based on the past observations. DXS forecasts were made one time unit into the future and smoothing statistics were updated and used for the next prediction. Accordingly, the RUL was estimated till the wear approaches the wear limit of the clutch system.

Figure 2.4 depicts the RUL estimation approach developed to monitor and predict the clutch-plate wear during operation.

Line and Iyer [19] developed a material-based physics-of-failure model for the electronic systems, especially the ones used by the aerospace original equipement manufacturers. Through the finite element analysis (FEA) model and the probabilistic microstructural fatigue model, the RUL of the solder joints in the electronic systems was estimated.

Valentin et al. [20] assessed the RUL of the solder interconnects with the help following model:

\[ CTF = C \times (S)^n \]  (1)
Where,

\(CTF\) – Cycles-to-failure of the solder interconnect

\(C\) – Constant that depends on specific materials and product parameters

\(S\) – A stress metric

\(n\) – A material parameter

Fig. 2.4 - Wear modeling architecture (Courtesy [18])

Advantage: Good accuracy.

Disadvantages: One major problem with the physical modeling is that for each engineered system, an entirely new model and algorithm needs to be created to estimate the RUL.

2.1.4 Statistical Model-based Approach

Statistical models [25-31] are also used for estimation of the RUL of the products. The statistical model-based approach requires formulation of a statistical model using parameters of the engineered system under consideration. The data gathered over a period of time is used as input to this model and the RUL is determined.
Liao, et al. [25] estimated the RUL by using both a proportion hazards model and a logistic regression model. Their approach related multiple degradation features of the sensor signals to the specific reliability indices of the product. The progression hazards model used the entire history of the degradation features as input. In contrast, the logistic regression model used only the current features as input. Parameters for both models were estimated by a likelihood function created from observed data collected under given conditions. Finally, the RUL of the product was estimated by fitting the degradation features in the graph window. The application of this approach to the bearing RUL estimation shows that the results of the proportion hazards model were better than those of the logistic regression model as the later one ignores prior degradation features.

Akers and Rideout [26] used Photon Induced Positron Annihilation (PIPA) data of aircraft turbine engine components to develop reliable-performance-life remaining prognostic model based on statistics.

Kiddy [27] developed a Remaining Useful Life Estimation (RULE) methodology for the aircraft components based on known usage monitoring data. With Monte Carlo simulations, component loads and fatigue life values were predicted conservatively. Then by monitoring aircraft usage, components RULs were estimated with known reliability.

2.1.5 Cumulative Damage Model

A product may undergo various kinds of damages like wear, fatigue, crack, corrosion and erosion. Initially, these damages are always small in quantity. A damage grows over time and when it reaches a level Z, shock N, or time T after its instantiation, the product reaches its
end-of-life [64]. In such cases damage accumulation models [32, 63, 64] provide estimation of the RUL of a product.

Usynin et al. [63] proposed a methodology for formulating prognostic requirement for the designers of the electronic prognostics enabled systems using a cumulative damage model. They developed the cumulative damage model based on a finite state Markov Chain (MC). The damage accumulation process begined at State 1 and subsequently reached to States 2, 3, ..., b. Finally, the RUL of the product was estimated as follows.

\[ MTTF = \frac{b - y_{obs}}{q} \]  

(2)

Where,

\( MTTF \) – Mean Time to Failure of the product

\( b \) – Last damage state before failure

\( y_{obs} \) – Observed current damage state of the product with some degree of health condition uncertainty

\( q \) – Transition probability that damage state receives a unit-size increase during a duty cycle

Usynin et al. [63] used the cumulative damage model given by Eq. (2) to estimate the RUL of the aluminum strips subjected to cyclic loading at a certain amplitude. State \( b \) and transition probability \( q \) were taken from the published literature. The RUL was estimated with various values of health condition uncertainties (2.6%, 10.5% and 18.4%). The simulation results showed that the 10% health condition uncertainty should be a cut-off value to estimate the RUL.
2.1.6 Nonlinear Dynamics

A nonlinear system contains variable(s) which cannot be solved as a linear sum of the independent components. A dynamical system is given implicitly by a relation that gives the state of the system only a short time into the future. To determine the state for all future times requires iterating the relation many times - each time advancing a small step. A nonlinear dynamics model incorporates both of them. A nonlinear dynamic model for a system represents dynamic behavior(s) that is/(are) highly sensitive to initial conditions [65]. For example, the point of contact of the baseball with the bat decides in which direction it will go. The purpose of this section is to depict that a system can be not only described by statistical models but also by dynamic models.

Bukkapatnam [61] predicted the remaining life of the tool in the turning process. He deduced it as a nonlinear process as it cannot be described by linear theory due to reasons: (a) finite amplitude of tool vibrations, (b) uncertainty of chatter (the vibration of a cutting tool or work-piece, resulting in a poor finish, and sometimes tool or work-piece breakage) which is sensitive to the initial conditions, (c) absence of clear stability boundaries, (d) broadband response, (e) multiple degrees of severity of chatter, (f) sensitivity to process parameters and; (g) aperiodic normal response and periodic chatter response. He also deduced that the turning process is sensitive to the initial conditions and hence termed its behavior as nonlinear dynamics. Suboptimal wavelet packet representation (SWPR) was developed for the Acoustic Emission (AE) signals coming from the cutting tool. Two novel signal separation methods – (i) the neighborhood method (NM) and (ii) a modified wavelet method (MWM) – that seemed to be adequate for chaotic time-series data (TSD) with small, uniform Lyapunov exponents, were developed. Appropriate features were selected and extracted from the
available TSD. For forces and vibration signals, the fractal dimension values computed using a modified box-counting method were selected. For AE signals, the entropy values that were significantly high in the overall SWPR of the measured TSD were used. Two recurrent neural networks, one for fractal estimation (forces and vibration signals) and the other for AE signal based estimation, with 7 input nodes and single hidden layer with 6 nodes were trained to give flank wear estimate at the output node. Subsequently, for developing chatter control scheme, a nonlinear control model for turning dynamics was developed. Though Bukkapatnam’s focus was on the process control, it did the required ground work for the RUL estimation by developing nonlinear model for the turning process.

2.2 Hybrid RUL Methodologies

A single methodology does not always produce satisfactory RUL estimates. In some cases, one methodology when used in conjunction with another produces more accurate results. That methodology is termed as the hybrid methodology. Hybrid methodologies are used when there are limitations to use one methodology alone or more accuracy is required. Some of these are discussed in this section.

2.2.1 Fourier Transform with Neural Network

Here, Fourier Transform is used to extract useful data from the monitored signals. Neural network takes this extracted data and produces the RUL estimate for a product. As an example methodology, Hongmou and Zein-Sabatto [33] monitored a rotating bearing vibration signals. A spectral analysis was carried out using Fourier Transform to separate useful time-frequency features. Consequently, two 2-layer neural networks were designed,
one for diagnosis having three statuses (normal condition, unbalance failure and other failures) and another for RUL estimation of bearing. In the end, software was written which implements this approach and takes care of all the required calculations.

### 2.2.2 Statistical Model with Neural Network

In some cases, the neural network models are used in combination with the statistical models to estimate RUL of a product. As an example methodology, Mazhar, et al. [34] proposed a two-stage approach for RUL estimation of used components in the consumer products. In the first stage, Weibull analysis was employed to assess the mean life of the components using time-to-failure data. In the second phase, an artificial neural network model was developed for the condition monitoring, degradation analysis and estimation of the RUL of the consumer products.

### 2.2.3 Fuzzy Logic with Neural Network

Here, fuzzy logic sets are used to identify current health condition of the product. This is given as the input to a neural network to output the RUL. As an example methodology, Essawy [62] presented a general methodology to estimate machine RUL using history data. This was an indirect methodology in which parameters related to critical machine operating regions and transitions were defined first. The prediction models were built using these parameters. Fuzzy logic decision-making system was used to locate current machine operating region with a probability. Then neural networks were used to estimate the RUL of the machine with known tolerance limit.
2.2.4 Wavelet transform analysis with statistical models

In this methodology, wavelet transform analysis is used to extract features from the monitored signal of a product. Then these features are given as input to the statistical models to come up with the RUL of a product. As an example methodology, Yan and Lee [35] proposed a hybrid method for on-line assessment and performance prediction of a drilling tool by monitoring the vibration signals. Wavelet packet decomposition was used to extract useful features from the vibration signal. Then Fisher’s criterion was applied to select the feature components that contain discriminant information and filter components having little information. Based on this information, condition assessment was done using logistic regression model. Finally, RUL of the drilling tool was estimated using the auto-regressive moving average model.

2.3 COMPARISON METRIC OF THE RUL ESTIMATION METHODOLOGIES

Current literature lacks a quantitative analysis of RUL estimation methodologies. Table 1 gives a quantitative comparison metric of some RUL methodologies. Table 1 is subjective and based on the authors own anecdotal observations. Every comparison criterion is given equal weight (20%) to calculate the overall rating. It can be seen from the metric that the neural network proves to be the best methodology after considering all factors together. In spite of the existence of all these RUL estimation methodologies, a fool proof and widely applicable methodology is yet to be seen. Also, there is hardly any research work which gives emphasis on the cost aspect of using particular methodology while estimating RUL.
### Table 1 - RUL estimation methodologies comparison metric

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Complexity</th>
<th>Processing Time</th>
<th>Expertise Level Required</th>
<th>Scope</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>(less-more)</td>
<td>(1-10)</td>
<td>(10-1)</td>
<td>(10-1)</td>
<td>(10-1)</td>
<td>(1-10)</td>
<td>(1-10)</td>
</tr>
<tr>
<td>Fourier Transform</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5.80</td>
</tr>
<tr>
<td>Wavelet Transform</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6.80</td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>7.40</td>
</tr>
<tr>
<td>Physical Model-based</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4.20</td>
</tr>
<tr>
<td>Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proportion Hazards</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>4.80</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>5.60</td>
</tr>
<tr>
<td>Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cumulative Damage</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5.00</td>
</tr>
<tr>
<td>Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 2.4 Important Observations from the Literature Review

While performing the overview of the RUL methodologies, following observations were worth to note.
(a) The RUL estimation is very much dependent on the engineered system/product under consideration. Not every methodology is applicable to each and every engineered system/product.

(b) In some cases multiple methodologies are fused together so that the RUL estimate is more accurate than that of a single methodology [56, 66]. The algorithm takes data from all the sensors required by the various methodologies, performs stipulated steps and give optimum RUL estimate through the fusion of all estimates.

(c) It can be seen that nano-technology is making its in roads for estimation of the RUL from the research works by Casey et. al [43].

(d) The estimation of the RUL of a product is also useful for proper disposal of the product at its end-of-life. In some cases, recycling/disposal task is poorly performed which causes harm to environment. Lee and Thomas [67] discussed the physical potential and limitations of the GPS and radio (through tracking) in recycling/disposal of the products after their end-of-life.
CHAPTER 3

PROPOSED MODEL FOR ASSESSMENT OF CURRENT HEALTH AND RUL

Various definitions of the RUL of a product are proposed in the literature. This thesis defines the RUL of a product as the period of time, from the current time till the termination of the product (may be due to the performance under the required level) expressed as the portion of its expected useful life.

After reviewing the RUL estimation literature, a model to assess the current health and the RUL is proposed as shown in Figure 3.1. This model is kept generic so that it can be applied to a broad range of products. The steps of the proposed model for a product are explained below.

![Fig. 3.1 - Proposed model for assessment of health and RUL](image)

<table>
<thead>
<tr>
<th>Degradation</th>
<th>Condition</th>
<th>Feature Extraction or Data Processing</th>
<th>Assess Health &amp; RUL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Identification</td>
<td>Monitoring</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.1 IDENTIFY THE DEGRADATION SIGNATURE

In general, products exhibit one or more peculiar degradation signature(s) which reflects the changes in their health as they age. The degradation signature may show either an increase or a decrease in trend. For example, vibration signals have been proved to form a good degradation signature for indicating a bearing’s health. The same is true about temperature for electronic devices [12].

Oftentimes, a failure mode and effects analysis (FMEA) is carried out to detect the degradation signature for a product. The analysis identifies potential failure modes, determines the root cause(s) of the failure and the symptoms (or effects) which can characterize the failure(s). If there are number of failure modes, they are prioritized according to the seriousness of their consequences, the frequency and the ease of detection.

3.2 PERFORM CONDITION MONITORING

The signature(s) identified in the previous step is (are) monitored while the product is in use. This process may take long time depending upon the life of the product. Oftentimes, the condition monitoring is performed through sensors mounted on the product. For this the standard procedure described below is carried out.

(a) A sensor capable enough to detect the change in the degradation signature is selected.

(b) A data acquisition systems compatible with the sensor is selected.

(c) Appropriate sampling frequency is selected.
(d) The optimal locations where the magnitude of the degradation signature is high are identified.

(e) A standard procedure to collect the sensor readings is defined.

(f) The setup required for condition monitoring is degradation signature specific and can be described only after the degradation signature is identified. This setup is prepared and the experiment is started to collect the sensor data.

In case the sensors selected are already present at the optimal locations in/on the product, then they are readily used for the data collection (if it is not hampering the standard operation of the product).

3.3 EXTRACT FEATURES OR DATA PROCESSING OF THE DEGRADATION SIGNALS

Feature extraction is performed on the raw data collected from the sensors in previous step. This step is carried out to come up with the most compact and informative signal representation possible without the loss of structural features of interest from the sensor signals. Here the challenge is dealing with the high dimensionality of input space. The raw signal is shrunk through filters or wrappers to shave off unnecessary parameters.

After feature extraction or data processing, a consistent, more impressive and smoother degradation signature curve should be visible. In the absence of a consistent trend, it is not possible to assess the current health and the RUL of the product. Depending upon the
characteristic(s) possessed by the degradation signature, the trend curve is expected to take any of the shape as shown in Figure 3.2.

![Fig. 3.2 - Nature of expected degradation signature curves](image)

**3.4 Assess the Current Health and the RUL**

The important information that trickles down from the feature extraction is given as input to the artificial neural network (ANN) to assess the current health and the RUL of a product.

Neural network is a strategy to model complex functional relationships without any knowledge or assumptions about the internal make-up, structure or parts of an engineered system [59]. A feed forward artificial neural network (ANN), which is known to be an effective learning and prediction and statistical tool, will be used so that it can learn any kind of trend curve(s). There will be a batch for training the ANN and the second one for testing the same.
CHAPTER 4

ACCELERATED DEGRADATION OF THE HDDs

Traditional life data analysis involves analyzing time-to-failure data of a product, system or component obtained under normal operating conditions in order to quantify the life characteristics of the product, system or component. In many situations, and for many reasons, such life time data is very difficult, if not impossible, to obtain. The reasons for this difficulty include the long life times of products, the small time period between design and release, and the challenge of testing products that are used continuously under normal conditions. Given this difficulty, and the need to observe failures of products to better understand their failure modes and their life characteristics, reliability practitioners have attempted to devise methods to force these products to fail more quickly than they would under normal use conditions. In other words, they have attempted to accelerate their failures or the degradation of the product.

4.1 OVERVIEW

A variety of methods that serve different purposes come under the umbrella of accelerated degradation. As we use the term in this reference, accelerated degradation involves acceleration of failures with the single purpose of quantification of the life characteristics of the product at normal use conditions. Oftentimes, the product is subjected to conditions beyond normal specifications stipulated by the manufacturer. Due to this, a harsh
environment is created around the product which causes it to wear out faster and eventually reach its end-of-life earlier than that with normal use.

More specifically, accelerated degradation can be divided into following two areas: use rate acceleration, and overstress acceleration.

### 4.1.1 Usage Rate Acceleration

Usage rate acceleration is useful for the products which do not operate continuously under normal use. Such products are operated continuously to maximize its usage and to encounter failures earlier than if the units were operated at normal usage. For example, an iron operates 5 hours a week on average. Then under usage rate acceleration, the iron is operated more frequently or continuously so that it fails earlier.

Due to continuous operation, some stresses or secondary effects do not occur like ON-OFF action, over heating due to continuous usage. To generate the same, tests can be carried out with frequent intervals to induce such effects.

Usage rate acceleration is not useful for products which have high usage rate under normal conditions. This is a limitation of the usage rate acceleration.

### 4.1.2 Overstress Acceleration

Under overstress acceleration, a product is subjected to stress(es) that exceed the stress(es) a product encounters under normal use conditions. The time-to-failure data obtained under these conditions is then used to extrapolate to the use conditions. The stresses are normally induced by varying temperature, humidity, voltage, pressure or vibration. A combination of two or more can also be used.
4.2 Literature Review of Accelerated Degradation of

The HDDs

Traditional testing under the normal operating conditions takes large amount of time as the HDDs have long life (around 5 years). So to shorten this testing time, the life of the HDDs needs to be accelerated by subjecting it to an environment beyond the normal specifications stipulated by the manufacturer. While accelerating the life, it is of utmost importance that the degradation followed by this accelerated curve will be same as the one followed by the naturally degraded product. The degradation should be graceful and follow all the stages (observed in normal conditions). In the case of the HDDs, rise in temperature, voltage, relative humidity and duty cycle or particle induction shows accelerated degradation and hence short life [68-73].

Strom et. al. [72] showed that the number of head-disk collisions increases causing faster damage to both at elevated temperature. Spindle motor and bearings also fail early at high temperatures which ultimately end into head crash failure. Herbst [18] explained that for every 5°C rise in temperature of a HDD above normal operating temperature (usually 25°C) causes 10-15% increase in failure rates. Cole [2] corroborated the same by experimenting with Seagate HDD. In his reliability test, he operated the HDDs at different temperatures from 25°C to 70°C with increment of four and observes that the mean time between failure decreases with the rise in temperature while keeping other factors (humidity, duty cycle etc) constant. An acceleration rate named as “derating factor” has also been calculated.
4.3 Approaches Chosen for The Accelerated Degradation of the HDDs

A careful review of literature revealed the following points.

(a) The HDD have the temperature gradient limit around 20°C/hr. So having temperature gradient more than the limit stipulated by the manufacturer may become one of our accelerated degradation factor.

(b) There is a specification called as "Operating Shock" for the HDDs. The normal limit for "Operating Shock" is 63G (63 times g, the gravitational force). Using this as the limit the HDD life can be subjected to operating shocks by dropping it from a “safe” height for accelerated degradation of the HDD.

(c) There is one more specification called as "Operating Vibration." It is the maximum vibration levels that the drive can withstand while meeting the performance standards specified. The HDD can be put on a shaker table to generate the artificial vibrations. The lower limits for the experiment can be derived from the upper limits stipulated by the manufacturer.

(d) The HDD also have "Acoustics" limits for idle and operating conditions. By exceeding these limits, the HDD can be subjected to accelerated degradation.

(e) Humidity is one of the dangerous factors for the HDDs. By increasing the humidity, the electronics (PCB) gets affected and HDD degradation is accelerated.
(f) Corrosive environment degrades the HDD PCB faster. This environment may contain sulfurous gases, chlorine or nitric oxide. The silver, copper, nickel and gold films used in the HDD are especially sensitive to the presence of sulfide, chloride, and nitrate contaminants. Sulfur is found to be the most damaging corrosive chemical for the HDDs. So by using the HDDs in the corrosive environment may accelerate the degradation of the HDDs.

(g) A HDD also has limits for radio frequency environments. In this harsh environment, there are more read/write errors and the HDD degrades faster.

During the session, it was decided to check viability of the following three factors as described below.

4.3.1 Introducing Artificial Scratches

The scratch experiment had the following hypothesis. During operation, there are instances when the head touches the disk platters and creates a scratch on the disk platter surface. Over a period of time, head moves on these scratches and goes under wear and tear along with the disk platter. After certain level of degradation, head being the weaker part gets damaged and stops functioning. A simulation of the natural scratch formation on the HDD disk platter showed that the scratch created is always circular though the head moves in radial direction over the platter surface.

In the present experiment, the behavior of the HDD under different sets of scratches was tested. As an outcome of those tests, it was decided to introduce three scratches on the top surface of the disk platter at certain radii with equal gap. The HDD health degradation curve
that was expected is shown in Figure 4.1. In the Figure, the low gradient lines indicate that the HDD health degrades due to the normal aging and the high gradient lines indicate the health degradation caused due to the artificial scratches.

**Fig. 4.1 - Expected health degradation curve for scratch experiment**

### 4.3.2 High Operating Temperature

Several failures of the HDD occur earlier by rising temperature beyond the normal operating range. At elevated temperature, the number of head-disk collisions increases causing faster damage to head. Metal components inside the HDD start expanding due to the elevated temperature and cause damage to the HDD. Spindle motor and bearings also fail early at elevated temperatures which ultimately end into head crash failure.

After a careful dissection and observation of several HDDs, it was deduced that the head is the weakest component inside the HDD. It is the head which gets ultimately affected as an effect of 99% of the errors inside the HDD. Figure 4.2 indicates a marginal difference between the heads of a brand new and a dead HDD. The black dots on the dead HDD head
are the result of the friction between head and the disk platter at very high speed. These dots show the high amount stress induced at the point of contact.

(a) Brand new head                     (b) Dead HDD head

Fig. 4.2 - Microscopic observation of a head of a HDD
4.3.3 Increment of Voltage

As per the standards set in electronics, to accelerate the degradation oftentimes the electronic devices are subjected to higher voltage and current levels. The same concept was used to experiment with the HDDs. Under normal conditions, a HDD requires 5V supply across the actuator and 12V supply across the spindle motor. To accelerate the degradation, a voltage higher than 5V or/and 12V were supplied to the HDD. This resulted in a flat degradation curve till the voltage goes 1.5 times the normal voltage level. Then the curve takes a vertical drop to the lowest point even if one of the voltage levels goes beyond 1.5 times the stipulated value. This concluded that the change in input voltages to the HDDs do not give a graceful degradation. Hence this approach was crossed out.

4.3.4 High Usage

Usage rate acceleration is useful for the products which do not operate continuously under the normal use. Such products are operated continuously to encounter failures earlier than if they were operated at normal usage. Oftentimes, computer systems are not used 24X7. Even when the computers are being used 24X7, the HDD is not 100% busy. Therefore, the usage of the HDD remains far below 100% during the normal usage. By increasing this usage level, the normal life of the HDD can be shortened.

In the present experiment, a program is written in the high level language which continuously does read/write/seek operations on a HDD. It assures that the duty cycle is close to 100%. The program is designed to simulate the real conditions on the HDD. It reads, writes or deletes a random file at random location on the HDD thus increasing the usage of the HDD.
4.4 CHALLENGES FACED

The challenges faced during the experiments carried out to test accelerated degradation of the HDDs described below.

4.4.1 Random Read/Write Program

As explained before, the HDDs do not remain busy all the time. So the usage remains low in the normal operating conditions. To increase the usage, we needed a method not only to increase the usage but also to imitate the normal random read/write/delete process. To serve this purpose, we developed a Java program [Appendix A]. This program continuously performs read/write/seek operations on the HDD. It assures that the duty cycle is close to 100%. The program is designed to simulate the real life operations of randomly writing new data, reading it randomly and deleting randomly. All these operations are carried inside an infinite loop thus increasing the usage of the HDD. The program also displays the operations (read, write or delete) it is carrying out on the monitor screen in real time.

The program is so robust that it can also be started in the middle of the experiment (if the HDD halts and the computer needs a restart) without any prior knowledge of the number of written files and the operations performed earlier on the HDD. The same program also has an ability to pre-fill the HDD with the initial data. This pre-filling was used only during the initial experiments. Two sub-programs were also derived at a later time from this program for dedicated read and dedicated write operations due to the specific need of the experiment.
4.4.2 Record a Log of Processes Running in the HDD

There is always a chance that the PC may halt or die at any moment. Given that it is not possible to have human surveillance all the time, there is a need of a means by which operations being performed on the HDD can be tracked. To address this issue, Java program was modified to make an entry of every process that is being carried out on the HDD into a log file. There was one more limitation that the log file cannot be stored on the HDD as it may die at any point during the experiment. This will cause loss of the log file. So the Java program was modified to store this log file into a USB drive connected to the PC.

4.4.3 Operating System

In this experiment, there were two problems with having the operating system on the HDD.

(a) The HDD is expected to halt/die at any moment due to accelerated degradation. The damage location (bad sectors) has equal probability to be at any place on the platter surface. Even a small proportion of the damage present at the location where operating system resides (boot sectors) can prevent the HDD from booting up though everything else is fine and HDD can work in the other case (as a secondary drive). So it was critical to boot from alternative source than the HDD under consideration.

(b) A large number of HDDs were anticipated to be used in this experiment. Installing an operating system on every one of them not only would take time but also would create licensing/money issue (in case of Windows). To solve this problem, there was a need that the operating system resides at a different place than the HDD being used in the experiment. While exploring the literature, the options found were to boot an operating system through USB drive, a CD or a primary HDD (with experimental HDD
connected as secondary). For the CD option was an operating system “Knoppix” was found out. For the USB option, “Ubuntu” was experimented and verified. “Windows” was finalized as the operating system for the primary HDD when the experimental HDD is being used as secondary HDD. To get the “boot from” option, one needs to press “F12” at the start up.
CHAPTER 5

UNI-AXIAL AND TRI-AXIAL VIBRATION SENSORS – FIRST APPROACH

In this chapter, the model proposed in previous section is applied to assess the current health and the RUL of the HDDs.

5.1 IDENTIFY DEGRADATION SIGNATURE

To identify a degradation signature, a failure mode and effects analysis (FMEA) is carried out on the HDDs. Listed below are the failure modes for the HDDs:

(a) Electrical Failure: These failures include electrical problems like PCB damage due to power surges, open/short connections.

(b) Mechanical Failure: These failures are related to mechanics of the HDDs which mostly include spindle motor failure, bearing failure, actuator failure, head crash, improper flying height of head and head contamination.

(c) Logical failure: These failures include accidental deletion, accidental format, file corruption, software bugs, file system corruption, viruses.

(d) Firmware failure – These failures cripple the HDD even if everything else is working.
   This kind of failure is very rare.
(e) Bad sectors – These failures are caused by wear and tear of the platter surface, head crash or manufacturing defects.

Seagate [74] sites that out of all these failures in the HDDs, mechanical failures account for 60% of the failures. As change in vibration is a proven signature for mechanical systems, it is chosen as the degradation signature for the HDDs. Tandon and Agrawal [75] corroborated the same after monitoring the vibrations of the HDDs with different disk rotating speeds. Ki and Polycarpou [76] showed that the scratches on platter surface increase the vibration level. Though they generated those scratches artificially, these scratches are generated by head-disk platter collision at high rotating speed in the normal scenario. As the HDD gets old, the frequency of the head-disk platter contacts increases causing increase in the vibration level.

5.2 PERFORM CONDITION MONITORING

Ki and Polycarpou [76] confirmed that the vibrations generated due to head-disk contacts are reliably carried to the HDD casing and can be accurately measured by a miniature sensor mounted on the HDD casing. So to monitor the vibrations, steps described in the condition monitoring step of the model are followed as follows.

5.2.1 Sensor Selection

An extensive literature review was conducted to understand various characteristics of HDD vibrations [75, 77, 78]. Based on this understanding an uni-axial piezoelectric shear type accelerometer was selected which has these specifications: weight 0.8g; sensitivity 100Mv/g; and frequency range 2-10 kHz is. The details of the sensor are presented in Appendix B.
5.2.2 Data Acquisition System (DAS)

The accelerometer is connected to an Analog to Digital Converter (ADC), which has a built-in signal amplifier. The ADC is connected to the Data Acquisition System (DAS) chassis. The latter transfers the voltage signals from ADC to the National Instruments’s LABVIEW data acquisition application software to display the incoming signals. The LABVIEW software provides a slew of signal processing functions such as generating spectrograms from the amplitude signals, filtering the signals through desired frequency bands, and presenting output in the various forms.

5.2.3 Sampling Frequency

The natural frequency of vibration for a HDD is in the range of 2Hz to 10 KHz [9, 11, 12]. The time-series data collected from the vibration sensor was converted into the spectral data using a Fast Fourier Transform algorithm. As a tradeoff between the data size and the monitoring time window, the length of a vibration signal sample was set to 2048 points. To accommodate all the frequencies of vibration the sampling frequency is set at 20 KHz. So, the frequency of sampling was kept precisely at 20480Hz, which allows collecting one sample (2048 data points) signal in the time window of 0.1s.

5.2.4 Sensor Location

HDD casing was tested for the optimal energy points. The points near actuator shaft and spindle motor shaft showed maximum energy of the vibration. In the present research, vibration signals of a sample of HDDs are monitored with a setup as shown in Figure 5.1. DAS was used to collect the vibration signals by mounting the sensors on all the HDDs.
Then the DAS converted them to a format which could be read by a computer. By using LabVIEW software, these vibrations signals were viewed and related data was stored for the feature extraction. To shorten years of HDD life into days, HDD life was accelerated which was explained in the previous chapter.

![HDD condition monitoring setup](image)

**Fig. 5.1 - HDD condition monitoring setup**

### 5.2.5 Procedure and Experiment Setup

Three sets of data were collected in the duration of 24 hours, with the time difference of 8 hours in between the sets. Each set of data consisted of 10 signals. First set of data of each day was collected during 7:00 – 8:00Hrs and the last set during 23:00 – 00:00Hrs. For this set of experiments, HDDs used were of Hitachi make with specifications listed in Appendix C. The experiment is setup as shown in Figure 5.2 and the experimental procedure is described below.

(a) Pre-fill the HDDs with the initial data by using the UNIX script listed in Appendix D.

(b) Connect each HDD to a dedicated Pentium-IV computer with 1GB of RAM.
(c) Connect all the computers to a single monitor through a parallel-to-serial-port 24-channel monitoring hub.

(d) Boot each computer using Knoppix operating system on the CD.

(e) Keep the HDDs always busy by running an algorithm written in Java to keep the duty cycle of each HDD close to 100%.

(f) Place each HDD in an anti-damping fixture and collect the vibration signals of the healthy HDDs at an interval of 6 sec. The planar accelerometer sensor is placed adjacent to the HDD actuator shaft as identified in sensor location step.

(g) Place the HDDs in a specially designed oven as shown in Figure 5.3 and raise the temperature of the oven to 80°C.

(h) Switch off the oven after 8 Hrs.

(i) Cool down the HDDs to the normal operating temperature of 31°C using a dry air blower. Confirm the HDD surface temperatures using a surface thermometer to check the consistency of temperatures in the data collection.

(j) Dismount the HDDs from the oven and place in an anti-damping fixture. Collect the vibration signals at an interval of 6 sec by placing the planar accelerometer sensor adjacent to the HDD actuator shaft as identified before.

(k) Repeat the steps (e) through (i) until each one of the HDDs becomes non-operational.
Fig. 5.2 - Experiment setup showing the oven with target HDDs, 6 systems, 1 parallel to serial hub and a single monitor

Fig. 5.3 - Zoomed in view of specially designed oven with HDDs in the fixed slots
5.3 Extract Features of Degradation Signals

The vibration signals collected from the HDDs comprised of a wide band of frequencies ranging from 2 Hz to 10 KHz. When a system degrades, the certain vibration frequencies follow a peculiar trend. Some frequencies may show upward trend while others may diminish as the system deteriorates. These frequencies or frequency bands sensitive to the degradation form the basis for extracting signal features.

A careful spectral analysis carried over the collected time-series acceleration data of the HDDs revealed that the signals in the range of 2 Hz – 150 Hz are noise signals whose magnitude remains constant over the period of HDD deterioration. Therefore this noise bands was eliminated. After de-noising it was found that there were three bands in the frequency spectrum that showed a distinct trend correlating well with the HDD deterioration. These bands thus became the parameters for the feature extraction.

(a) Band B1: frequencies 5163-5395Hz
(b) Band B2: frequencies 6018- 6494Hz
(c) Band B3: frequencies 8093- 8447Hz

The bands $B_1, B_2, B_3$ were used for extracting the features from the sensor data. Thus these bands carry important information for representing the HDD deterioration.

An algorithm in C was developed to determine the features by calculating the RMS energy values for each of these 3 bands of each signal and then calculate the overall RMS energy contained in 30 signals that were collected per day. This can be summarized as:
\[ F_x = \sqrt{\sum_{i=1}^{10} \sum_{j=L_x}^{U_x} S_{ij}^2} \]  \hspace{1cm} (6)

Where,

\( F_x \) = Feature \( x \) extracted from frequency band \( x \), \( x = 1, 2 \) or 3

\( i \) = signal index between, \( i = 1, 2, \ldots, 10 \)

\( j \) = frequency ranging from \( L_x \) to \( U_x \)

\( L_x \) = Lower frequency of band \( x \)

\( U_x \) = Upper frequency of band \( x \)

\( S_{ij} \) = Signal \( i \) with values only between boundaries \( j \)

5.4 ASSESS HEALTH AND RUL

A feed forward artificial neural network (ANN), which is known to be an effective learning and prediction tool, statistical tool, is used to assess the current health and the RUL of the HDD. A three element vector \( I = [F_1, F_2, F_3] \) representing RMS value of the three features identified and calculated in the previous section was used as the input vector to the ANN. The ANN output was a two element vector \( O = [O_1, O_2] \). Here \( O_1 \) represented the current operating time in weeks and \( O_2 \) represented the current health (1 represents the best health and 0 represents the worst health). A C program was designed and developed for the ANN.

5.4.1 ANN Training

The sensor data collected for the training batch products is processed through the feature extraction or data processing step and the fed to the ANNs. As all the data over the entire life
of the product is available for the whole batch, the expected output of the ANN can be easily calculated. As shown in Figure 5.4, four ANNs were trained to assess the current health and RUL of the HDDs.

![Fig. 5.4 - ANNs training and testing](image)

**5.4.2 RUL Calculation**

For estimating the RUL of a new product, the sensor data is collected; features are extracted and fed to the ANN. The steps to estimate the RUL are given below.

Initially the error between ANN predicted current operating time and the actual operating time is calculated as,

$$e_i = \left| O^i - A \right|$$  \hspace{1cm} (3)

*Where,*

\(e_i\) is the error between predicted and actual operating time.
$e_i$ – Error of $i^{th}$ ANN

$O_i^1$ – First output (current operating time) of $i^{th}$ ANN

$A$ – Actual operating time.

An intermediate value, $Z$, for these selected ANNs is calculated as,

$$Z_i = \sum \frac{e_i}{e_i} \quad (4)$$

Then, the weights $W_i$ for these ANNs are calculated as,

$$W_i = \frac{Z_i}{\sum Z_i} \quad (5)$$

A higher weight value for a particular ANN indicates that the new product is more likely to behave as per the product which trained that ANN. Here the aim is to identify a product from the training batch which is close in terms of behavior with the new product. Finally, RUL of this new product is calculated as,

$$RUL = \sum (W_i \times RUL_i) \quad (6)$$

Where,

$RUL$ – RUL of the new product

$RUL_i$ – RUL of $i^{th}$ product which trained $i^{th}$ ANN (total life – current operating time).

### 5.4.3 Current Health Calculation

The current health level for the new HDD is calculated as follows,

$$Current\ Health = \sum (W_i \times O_i^1) \quad (7)$$
Where,

\( W_i \) – Weight of \( i^{th} \) ANN calculated in step 1

\( O_2^i \) – Second output (current health level) of \( i^{th} \) ANN

5.5 RESULTS AND DISCUSSION

In spite of the hard work, this approach was not able to give desirable and consistent results. Through the brainstorming session, two reasons were derived for the failure of this approach as listed below.

(a) The vibration sensors were not able to pick up the degradation of the HDDs. As shown in Figure 5.5, the overall change in the magnitude of the RMS value of the vibration signal was less than the fluctuations of the vibration signal.

![Fig. 5.5 - Change in RMS value of the vibration signal](image)
(b) The same is true for the band energies which are derived after processing the raw vibration signal. Also the band energies show some unexpected spikes as shown in Figure 5.6.

![Fig. 5.6 - Band energies of a HDD](image)
CHAPTER 6

ACOUSTIC EMISSION SENSORS

After studying the results of the vibration sensor experiment, it was concluded that the vibration sensors are not able to detect the head-disk collisions and other changes in the HDD properly happening during random read/write operations. To monitor the same, there was a need of a different approach.

6.1 REPLACEMENT OF THE VIBRATION SENSOR

After the dire failure of the vibration sensor, there was a need for a different approach or sensor. As stated before, any mechanical problem inside the HDD end up with head-disk collisions which ultimately result in damaging the head and creating scratches on the disk platter surface. Continuous rubbing against these scratches and continuos collisions result in the head crash. If not vibration sensor, there was a need of a different kind of sensor which can detect changes in vibration when a small surface (head) touches the platter causing a very minute amount of displacement of the material. As this head-disk contact happens when the disk is rotating at 7200rpm, very high level of stress is generated. There was one more problem with the conventional vibration sensors. The vibrations energy coming to the HDD casing was observed to be much lower than that at its source. The new sensor should also address this issue.

The sensor which addressed all these issues was found to be the acoustic emission (AE) sensor. AE waves satisfied all the following the issues found with vibrations sensors
(a) AE waves travel through only a solid medium.

(b) They are generated when there is a small surface displacement of a material and the stress level is high and a rapid release of energy in a material, or on its surface.

(c) It was cross checked that 99% of the energy of the AE waves is carried to the HDD casing.

Thus AE sensor overcomes all the drawbacks of the vibration sensors.

6.2 IDENTIFY DEGRADATION SIGNATURE

As discussed before, AE wave becomes the degradation signature to assess the current health and the RUL of the HDDs.

6.3 PERFORM CONDITION MONITORING

For condition monitoring, the same set of sequence was followed to carry out the experiments as described below.

6.3.1 Sensor Selection

An extensive literature followed by a market review was conducted to understand various characteristics of the AE sensors. Based on this understanding two AE sensors with the following specifications were selected:
### Table 2 - AE sensor specifications

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Model</th>
<th>Dimensions (Dia X Ht) mm/inches</th>
<th>Weight (grams)</th>
<th>Operating Temperature (ºC)</th>
<th>Peak Sensitivity (dB)</th>
<th>Operating Frequency (KHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UT1000</td>
<td>18 x 17 / .7 x .65</td>
<td>20</td>
<td>-65 to 177</td>
<td>-70.36</td>
<td>60 - 1000</td>
</tr>
<tr>
<td>2</td>
<td>PICO HF-1.2</td>
<td>5 x 4/ 0.2x0.15</td>
<td>0.1</td>
<td>-65 to 177</td>
<td>-76.67</td>
<td>500 – 1850</td>
</tr>
</tbody>
</table>

### 6.3.2 Data Acquisition System (DAS)

The AE sensors needed a different Analog to Digital Converter (ADC) than the one required by a vibration sensor. To maintain the AE signal strength, a signal amplifier was used in between the AE sensor and the ADC. The pre amplifier gain was set at 60dB. This completed the DAS for this set of experiments.

### 6.3.3 Sampling Frequency

The length of the AE signal sample was kept the same as before, 2048 points. To accommodate all the frequencies of vibration the sampling frequency was set at 2MHz.

### 6.3.4 Sensor Location

The same set of procedure was carried out to identify optimum energy point for the AE sensors again. Once again actuator shaft was a clear winner for the sensor location.

### 6.3.5 Experiment Setup

Two sets of data were collected every day, with the time difference of 12 hours in between each set. The number of signals in one set was increased from 10 to 100 as it was observed that more number of signals smoother trend curve. First set of data of each day was collected
at 8:00 – 9:00Hrs and the second set at 20:00 – 21:00Hrs. This time, we used Seagate SATA HDDs [Appendix E] instead of Hitachi PATA HDDs.

Meanwhile, a few refinements from the previous experiment were carried out as listed below.

(a) The Java program used for random read/write operation was rewritten in the C language. This showed a better performance over the existing Java program in terms of the read write rate. To give some statistics, the C program’s [Appendix F] read write rate was 4 times that of the Java program.

(b) Seven brand new computers were bought to carry out the experiment in a consistent environment. All these seven computers have the same configuration [Appendix G] and the operating system.

(c) The operating system was changed from Knoppix to Windows Vista as it offered additional features over the former one and came in-built with the new computers.

(d) The HDDs were not prefilled with any data at the time of the start of the experiment. The reason behind this is that the HDDs deteriorated (though very little) due to the high speed writing through the UNIX script.

6.3.6 Experiment Procedure

Following procedure was followed for taking the experiment readings.

(a) Power ON the 7 systems with 7 HDDs connected to them.

(b) Run the random “WRITE” C program on all systems.
(c) Power ON the preamplifier and connect the DAQ to the monitoring system using USB cable.

(d) Place the UT1000 sensor on the predetermined sensors position marked on the HDDs.

(e) Run Labview 8.6 on the monitoring system and open AE_UT.vi program file pre-stored.

(f) Keep the measurement ON switch in OFF position in AE_UT front panel.

(g) Run AE_UT and verify the time series signals are the proper AE signals and the instantaneous amplitude is in the range of approx. 0.15 to 0.25.

(h) If the range is not correct or the times series data is erroneous, check the following in the given order until you get correct signals:

   i. Sensor is sitting properly on the HDD surface, apply proper amount of wax to make the contact strong.

   ii. Preamplifier input is at single and power is ON.

   iii. Check if the “WRITE” program is running and files are getting written on the HDD.

   iv. Double click the DAQ monitor in AE_UT and verify the preset parameters.

   v. Turn OFF the machine and check the power cable and data cable of the HDD.
vi. Place the sensor on the other HDD and check the time series signals. If the signals are AE with correct type declare the former HDD as dead and remove it from the experiment.

vii. If the signals are erroneous even for later HDD, probably the sensor has gone bad. Change the sensor and repeat the process.

(i) Halt the AE_UT program. Change the HDD serial number on the front panel of AE_UT and turn the measurement switch ON. Run the program again using single RUN ( \( \rightarrow \) ) switch. Program will halt itself after recording 100 readings for time series, rms energy, spectral data and the moving average.

(j) Halt the “WRITE” program and run the random “READ/WRITE” program.

(k) Keep the HDD in the oven. Repeat the process for all 7 HDDs, except HDD7 which needs to be kept outside the oven.

(l) Turn ON the oven and keep the temperature at 95° C.

(m) Turn OFF the preamplifier and disconnect the DAQ from the monitoring station.

(n) Enter the reading number, oven OFF and ON time and any changes in the AE log.

(o) Repeat the process every 12 hours, at 8am and 8pm.
6.3 Extract Features of Degradation Signals

A careful spectral analysis carried over the collected time-series acceleration data. Frequency data of the HDDs were still giving the same fluctuating in the trend curve as given by that of the vibration sensors.

As shown in the Figure 6.1 and 6.2, the spectral analysis carried over the time-series data of the AE signal hardly shown any consistent trend. Each point on the degradation curve is an average of the 100 readings collected in a set for the respective week. The trend line is shown by the red curve. The $R^2$ value for the trend curve is 0.3532, which shows a poor correlation between the two.

![Fig. 6.1 - RMS value of AE signal for a PICO sensor](image)

---

*Fig. 6.1 - RMS value of AE signal for a PICO sensor*
Then a careful spectral analysis of the frequency data of the HDDs was carried out. This analysis showed that the pico sensor data does not show any consistent trend and hence is found almost useless. On the other hand, the same analysis carried over the UT sensor data yielded a band of frequency (80-100 KHz) with a trend as shown in Figure 6.3.
6.4 DISCUSSION

As the results shown by the AE sensor experiment were not favorable, this experiment was also terminated and new search for the degradation signature was started.

At the end of this experiment, the disk platter was also observed under the microscope for scratches. This revealed that the disk surface do not change over time.

So this approach also did not work. There was again a need of different approach to assess current health and the RUL of the HDDs.
CHAPTER 7

SMART ATTRIBUTES

After achieving the successive failures from the sensor experiments, a thorough literature search was carried out on different kinds of sensors and their applications. By this time, there was ample amount of learning about the behavior and failure mechanism of the HDDs. It was deduced that none of the sensor suits to the application of identifying and capturing the degradation signal for the HDDs. Then the scope was enlarged and out of the box thinking was started (without restricting to only sensors). While doing so, a literature [79-81] was found which described that the Self-Monitoring, Analysis, and Reporting Technology (SMART) attributes can be used for the prognostics of the HDDs.

7.1 SMART INTRODUCTION

7.1.1 History

IBM pioneered the hard disk monitoring technology first in 1992. IBM 0662 SCSI-2 was the first disk drive to implement this technology which later on was named as Predictive Failure Analysis (PFA). PFA aimed at monitoring a few key device health parameters (e.g., temperature, number of reallocated sectors, seek errors) through the drive firmware. At that time, the PFA output was designed to be binary – “OK” or “About to fail”.

Owing to the competition and to keep up the market share, Compaq in conjunction with disk drive manufacturers Seagate, Quantum, and Conner developed “IntelliSafe” technology which was pretty much similar to the PFA from IBM. [2]. IntelliSafe was also designed to
monitor disk health parameters and transfer the values of the parameters to the operating system and user-space monitoring software.

Finally in 1995, Compaq, IBM, Seagate, Quantum, and Conner along with the Small Form Factor Committee combined the two technologies – PFA and IntelliSafe and developed a new standard named as S.M.A.R.T. (Self-Monitoring Analysis and Reporting Technology). The disk health parameters to be monitored under SMART are also called as SMART attributes.

Nowadays, almost all the drive manufacturers like Samsung, Seagate, IBM (Hitachi), Fujitsu, Maxtor, Western Digital and ExcelStor support the SMART technology. Appendix H lists some of the SMART attributes with their meanings.

7.1.2 Attribute Value

Each attribute is normalized to a value ranging from 1 to 253. Higher attribute values could be better or worse depending on the attribute and the manufacturer. Also, each attribute may not start from 1 or 253. Instead, there will be default value set by the manufacturer. This value is normally 100 or 200. There is one more value associated with some of the attributes named as “raw” value. The raw value for the attributes, for example “Seek Error Rate”, indicates the actual number of errors (lets say 197,704,736) that happened till now. Whereas the current value shows the normalized value (always between 1 to 253) which is derived from this raw value.

SMART attributes change slowly over time (through the use of the HDD). The attribute value can only be updated by the HDD itself and is read only to the user of the HDD [82].
read the attribute value, the user needs to query the firmware of the HDD to get the current attribute value and threshold value in return (on Windows, the user needs to be an administrator).

### 7.1.3 Attribute Value Threshold

Each drive manufacturer kept researching about the SMART attributes and each of them not only set different priorities for the SMART attributes but also has different number of SMART attributes depending on the HDD’s design. There is also variation in the default values of the SMART attributes. The reason behind this is that the drive manufacturers sometimes want to keep it as their “trade secret”. In spite of these variations, the interface used to communicate the SMART attributes to the system and the way it is communicated is same across all the drive manufacturers [74].

Each SMART attribute has a preset threshold and when that threshold is exceeded by the current value, an alarm “threshold exceeded” is triggered by the SMART attribute. This event puts a red alarm announcing that the HDD can fail at any moment of time in the near future.

### 7.2 IDENTIFY THE DEGRADATION SIGNATURE

After learning that the SMART attributes indicate the health of the HDD, naturally it becomes the degradation signature in this experiment. If the performance of any component inside the HDD goes lower than usual, then it is immediately reflected into its respective SMART attribute value. For example, if the motor and/or the bearing performance degrade, then the drive spin-up time and the number of retries attribute value starts increasing. Or, if
the error correction is being used excessively, then it indicates that there is some contamination in the drive head or it is broken.

7.3 PERFORM CONDITION MONITORING

The condition monitoring was performed as described below.

7.3.1 SMART Attributes Monitoring Tools

There are a number of ready-made tools for all kinds of operating systems in the market which can display current value of the SMART attributes along with the threshold value. These tools can query the drive firmware to get the attribute values. To do that, one needs to “enable” SMART in the BIOS (Basic Input Output System) of the computer. For some HDDs like Hitachi, the SMART can be enabled/disabled from the HDD itself. In such cases, the manufacturer provides a small tool to enable/disable the SMART. When the computer boots, it does a memory check and HDD identification. At that time, along with a report it also shows an entry that says SMART capable and whether it is enabled or disabled currently. However, nowadays the computer manufacturers show their own page (oftentimes their logo) as a startup page and may not show such information.

With the evolution of the SMART attributes, the SMART attributes monitoring tools have also evolved and they are able to do much more than merely displaying the attribute values. In addition to displaying the SMART attribute values, these tools can document as well as analyze the SMART information and create warnings if necessary. The warnings are normally conveyed through highlighting the respective attributes with the red background or by popping up messages. Some SMART monitoring tools can inform the user by sending an
email notification as soon as the “threshold exceeded” event occurs. Some tools like ActiveSmart can keep a track of the changes happened over time through logs. Some SMART monitoring tools like Advanced SmartCheck can be used over a network to monitor HDDs of the computers connected amongst that network.

The SMART monitoring tools available in the market are: smartmontools, SMARTHDD, libatasmart, HDAT2, DriveSitter, HDD Health, Active Smart, SpeedFan, SMARTReporter, HDTune, Norton System Doctor, SMART Utility, DiskCheckup, Hard Disk DiskChecker, Sentinel and Advanced SmartCheck.

7.3.2 Selection of a SMART Attributes Monitoring Tool

Table 3 provides a comparison of the SMART attribute monitoring tools listed in the previous section [83].

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>SMART monitoring tool</th>
<th>Operating System</th>
<th>User Interface</th>
<th>Self Test</th>
<th>Notification at</th>
<th>Notification by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>smartmontools</td>
<td>Windows (native or Cygwin)</td>
<td>Command line, optional daemon or service</td>
<td>Yes (also scheduled)</td>
<td>choosable parameter changes, threshold, temperature</td>
<td>window (only Windows), e-mail, system log, run a certain command</td>
</tr>
<tr>
<td></td>
<td>Product</td>
<td>Operating System</td>
<td>Interface</td>
<td>Availability</td>
<td>Displayed Parameters</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----------</td>
<td>------------------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SMARTHDD</td>
<td>Windows</td>
<td>graphical</td>
<td>Yes</td>
<td>threshold, temperature, raw count changes window</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>libata smart</td>
<td>Linux</td>
<td>Command line, Programming Library</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>HDAT2</td>
<td>DOS</td>
<td>text menu</td>
<td>Yes</td>
<td>- window, choosable parameter changes, sound, e-mail, network message, run a certain command, shutdown/sleep /hibernate</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Drive Sitter</td>
<td>Windows</td>
<td>graphical</td>
<td>Yes</td>
<td>choosable parameter changes, threshold, temperature, raw count changes window, sound, e-mail, network message, system log, run a certain command, shutdown/sleep /hibernate</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HDD Health</td>
<td>Windows</td>
<td>graphical</td>
<td>Yes</td>
<td>every parameter change, temperature window, sound, e-mail, network message, system log</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Active Smart</td>
<td>Windows</td>
<td>graphical</td>
<td>No</td>
<td>threshold, temperature window, sound, e-mail, network message</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Speed Fan</td>
<td>Windows</td>
<td>graphical</td>
<td>No</td>
<td>choosable parameter changes, window, sound, e-mail, run a certain</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Tool</td>
<td>OS</td>
<td>Style</td>
<td>Active</td>
<td>Thresholds</td>
<td>Commands</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>----</td>
<td>-------</td>
<td>--------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>9</td>
<td>SMART Reporter</td>
<td>Mac OS X</td>
<td>graphical</td>
<td>No</td>
<td>threshold</td>
<td>window, e-mail, run a certain command</td>
</tr>
<tr>
<td>10</td>
<td>HD Tune</td>
<td>Windows</td>
<td>graphical</td>
<td>No</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Norton System Doctor</td>
<td>Windows</td>
<td>graphical</td>
<td>No</td>
<td>threshold, (for every single medium)</td>
<td>taskbar symbol, sound, administrative message</td>
</tr>
<tr>
<td>12</td>
<td>Disk Checkup</td>
<td>Windows</td>
<td>graphical</td>
<td>Yes</td>
<td>temperature</td>
<td>window, e-mail</td>
</tr>
<tr>
<td>13</td>
<td>Hard Disk Disk Checker</td>
<td>Windows, DOS, Linux</td>
<td>graphical</td>
<td>Yes</td>
<td>temperature, parameter changes, threshold, new problem found, low disk space</td>
<td>taskbar, Window, e-mail, network message, sound alert (optionally repeating), run a certain Command, run automatic backup projects, shutdown/hibernate</td>
</tr>
<tr>
<td>14</td>
<td>Sentinel</td>
<td>Windows, .NET Framework 3.5</td>
<td>graphical</td>
<td>Yes</td>
<td>at startup, low health</td>
<td>window, taskbar notification</td>
</tr>
<tr>
<td>15</td>
<td>Advanced</td>
<td>Windows</td>
<td>graphical</td>
<td>Yes</td>
<td>temperature</td>
<td>window, e-mail</td>
</tr>
</tbody>
</table>
After analysing each of the above tools, a decision was taken to use “smartmontools” as it offered the following features:

(a) Commandline operations offers a great amount of flexibility.

(b) It is independent of operating system which is useful if the operating system currently in use is replaced by the other tomorrow.

(c) Supports almost every kind of HDDs.

(d) Easy to use while comprehensive.

(e) Lightweight.

(f) Huge documentation and help available online [84].

Out of the two flavors of the smartmontools (smartctl, smartd), smartctl was the obvious choice for this set of experiments as it was the only option that works with Windows. Figure 7.1 shows how to use smartmontools with command line and its output. The “-A” option displays only SMART attribute values for the drive specified. The column names are self explanatory for the information it displays. In some cases, the “worst” column has created a
confusion. To clear that the worst column gives the worst value that attribute has ever attained. There is one more option “-a” to display all the HDD information on the screen in smartmontools.

```
C:\>smartctl -a c:
smartctl version 5.38 fi686-mingw32-xp-sp31 Copyright (C) 2002-8 Bruce Allen
Home page is http://smartmontools.sourceforge.net/

--- START OF READ SMART DATA SECTION ---
SMART Attribute Data Structure revision number: 10
Vendor Specific SMART Attributes with Thresholds:

<table>
<thead>
<tr>
<th>ID# ATTRIBUTE_NAME</th>
<th>FLAG</th>
<th>VALUE</th>
<th>WORST</th>
<th>THRESH</th>
<th>TYPE</th>
<th>UPDATED</th>
<th>WHEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Raw_Read_Error_Rate</td>
<td>0x00f</td>
<td>100</td>
<td>253</td>
<td>006</td>
<td>Pre-fail</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>3 Spin_Up_Time</td>
<td>0x003</td>
<td>099</td>
<td>098</td>
<td>085</td>
<td>Pre-fail</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>4 Start_Stop_Count</td>
<td>0x032</td>
<td>100</td>
<td>100</td>
<td>020</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>5 Reallocated_Sector_Count</td>
<td>445</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 Seek_Error_Rate</td>
<td>0x00f</td>
<td>073</td>
<td>060</td>
<td>030</td>
<td>Pre-fail</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>9 Power_On_Hours</td>
<td>1085</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 Spin_Retry_Count</td>
<td>0x013</td>
<td>100</td>
<td>100</td>
<td>034</td>
<td>Pre-fail</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>12 Power_Cycle_Count</td>
<td>449</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>187 Reported_Uncorrect</td>
<td>0x002</td>
<td>100</td>
<td>100</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>189 High_Fly_Writes</td>
<td>0x03a</td>
<td>001</td>
<td>001</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>190 Airflow_Temperature_Cal</td>
<td>0x022</td>
<td>063</td>
<td>053</td>
<td>045</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>191 C-Sense_Error_Rate</td>
<td>0x032</td>
<td>100</td>
<td>100</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>192 Power_Off_Retreat_Count</td>
<td>424</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>193 Load_Cycle_Count</td>
<td>19232</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>194 Temperature_Celsius_37</td>
<td>0x01a</td>
<td>037</td>
<td>047</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>195 Hardware_ECC_Recovered</td>
<td>0x012</td>
<td>078</td>
<td>063</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>197 Current_Pending_Sector</td>
<td>0x010</td>
<td>100</td>
<td>100</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
<tr>
<td>198 Offline_Uncorrectable</td>
<td>0x003</td>
<td>100</td>
<td>100</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>199 UDMA_CRC_Error_Count</td>
<td>0x000</td>
<td>280</td>
<td>280</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
<tr>
<td>200 Multi_Zone_Error_Rate</td>
<td>0x032</td>
<td>100</td>
<td>253</td>
<td>000</td>
<td>Old_age</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>202 Th_Increase_Count</td>
<td>0x000</td>
<td>100</td>
<td>253</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
<tr>
<td>240 Head_Flying_Hours</td>
<td>0x000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
<tr>
<td>NG_NOW 271946795117958</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>241 Unknown_Attribute</td>
<td>0x000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
<tr>
<td>NG_NOW 1546853326</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>242 Unknown_Attribute</td>
<td>0x000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
<tr>
<td>NG_NOW 721982</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>254 Unknown_Attribute</td>
<td>0x000</td>
<td>001</td>
<td>001</td>
<td>000</td>
<td>Old_age</td>
<td>Offline</td>
<td></td>
</tr>
</tbody>
</table>
```

Fig. 7.1 - SMART attribute values from “smartmontools”
7.3.3 Automate SMART Monitoring

After selecting the SMART attribute monitoring tool, the next task was to record the output from the command line into a file on the primary HDD. To do this, a C program [appendix I] was written which did following functions in sequence.

(a) Execute “time”, “date” and “smartctl –A f:” commands on the command line and store its output into a temporary text file on the primary drive.

(b) At the same time, write the same output to a permanent file (in append mode) to keep a backup of the SMART attributes data.

(c) Create new files to store the current values and worst values of the SMART attributes into a different format (only if the files does not exist on the specified location).

(d) Write the header of the newly created file (only if the file does not exist on the specified location).

(e) Read the current values and worst values of the SMART attributes from the temporary file.

(f) Write/append these values to the newly created files.

(g) Sleep for the time specified in the program freeing up the computer resources.

(h) Repeat the steps from (a) to (g) in an infinite loop.

The sleep time was specified as 1 minute to accumulate sufficient amount of data over the accelerated life time of the HDD. In case the amount data explodes, it was planned to filter...
out the required data. This care was taken so that the data should not be insufficient at the end of the experiment.

So for this set of experiments, this SMART attributes monitoring C program was ran in conjunction with the previous random read/write C program. The HDDs were not prefilled with the initial data as it was done in the previous set of experiment. All the HDDs were kept inside the oven at 95º C.

**7.4 Extract Features of the Degradation Signals**

With the help of the SMART attributes monitoring C program, the condition monitoring work took quite a less amount of time every day. Now with ample amount of data, the next task was to select the attributes showing not only good trends but also consistent trends. The expectation was that a few SMART attribute values will gradually decrease or increase as the HDDs approaches towards their end-of-life through the accelerated degradation.

Before carrying out the feature selection, the data gathered from the SMART attribute monitoring program was purified for the following cases.

(a) If the secondary HDD (HDD under consideration) goes out of detection for any reason, the SMART attribute recording program stored garbage values. Such samples were filtered out by cross checking such events with the log created by the random read/write program.

(b) When the HDD was near the death point, the SMART attribute monitoring program recorded values of the order $10^6$ for a few number of samples. Such records were less
than 5% of the samples and were filtered out as this behavior was not consistent across the HDDs.

7.4.1 Initial Trend Curves

During the first 50 hours of the experiment, following SMART attributes had consistency in trend curves across all the HDDs.

(a) **Hardware ECC Recovered**

The disk platter of the HDD is subject to occasional natural incidences of data loss due to the impact of alpha particles or cosmic rays. The data loss results in changing the value stored in the cell of the memory influenced by the collision. Normally only a single cell is influenced, but there are small amount of chances that more than one cell are influenced. This change in the cell value due to the collision is called as soft error. Generally, it just flips the current value in the cell. So to prevent such loss of the data, a technology was developed called as Error Correction Code (ECC). With this technology, every word of the data is stored along with a number of extra check bits in the memory. When the word is retrieved from the memory, a fresh set of check bits are recomputed and compared with the check that was stored in the memory. The result of this comparison is called the syndrome. If the syndrome gives “zero” as the output, it indicates a match between the two values. If it is not zero, then it indicates that there is an error in the data and the syndrome is used to find a single bit in the error and correct it. This correction is called as ECC correction and the time taken during such corrections is indicated by Hardware ECC Recovered. Increase in this time (raw value) is not a good indication with respect to the HDD health. As per the appendix H, decrease in Hardware ECC Recovered attribute value is not good. (As
the attribute value of the Hardware ECC recovered is normalized with respect to the raw value, it is always between 1 to 253 even if the raw value is in millions. And as it can start with any default value 100 or 200, the decrement from that value is not considered good.)

Figure 7.2 shows the attribute values over the first 50 hours of operation. This trend was consistent for all the other HDDs. So Hardware ECC Recovered attribute was selected as one of the attributes to keep a watch on.

![Graph showing Hardware ECC Recovered values over time.](image)

**Fig. 7.2 - Hardware ECC Recovered graph for first 50 hours**

(b) Seek Error Rate

Seek Error Rate is the rate of seek errors of the magnetic heads. This attribute is related to the actuator controlling the movement of the arm over the disk platters. If there is a failure in the mechanical positioning system, may be a servo damage or a thermal widening of the hard disk, seek errors arise. More seek errors (raw value) indicates a worsening condition of a disk surface and the mechanical subsystem (actuator). In this case, higher Seek Error Rate attribute values also indicate
worsening condition. The raw value indicated by the Seek Error Rate is a cumulative value or a count and is not an actual rate. It counts the number of seek errors that have happened since the HDD has been put into operation.

Figure 7.3 shows the attribute value over the first 50 hours. The sudden drop of 40 points (from 100 to 60 at before 1000 minutes) was observed for all the HDDs and the reason for this is unknown. This trend was consistent for all the other HDDs.

The other attributes which changed during first 50 hours were “temperature” and “airflow temperature” which was expected due to ON and OFF cycles of the oven. All the other attributes were unchanged for the first 50 hours.
7.4.2 Outside Oven HDD

While carrying out this set of experiment, we needed to compare the degradation of the HDDs inside the oven with that operating under normal conditions. It was necessary to know whether the HDDs inside oven have any effect of high temperature on their degradation or not. In addition to that, it would help to filter out those attribute which show the same trend irrespective of the environment it was in (inside or outside oven).

So a HDD of the same specifications was kept outside the oven with the random read/write C program running on it. All the environment variables were same for this HDD except the operating temperature.

7.4.3 Final Feature Selection

Many attributes changed over the time. Few of the attributes were specific to one or a subset of the total HDDs where a few were consistent for all the HDDs. The ones which could not make it to the list of consistent attributes due to inconsistency are Reallocated Sector Count, High Fly Writes, Current Pending Sector, Current Pending Sector Modified, Offline Uncorrectable and Offline Uncorrectable Modified.

The Seek Error Rate attribute was consistent across all the HDDs inside the oven, but the nature didn’t change much whether the HDD is operating inside a oven or operating at a normal room temperature. Figure 7.4 shows that the outside oven HDDs (blue curve) always lags behind the inside oven HDDs. The enlarged view in the inset shows that this lagging happens when the attribute values takes a sudden jump of ~35 points (from 100 to ~65) during first few hours of operation. Except for this lag, the nature of both the curves remained the same; increasing step by step after a specific interval of time. So though Seek
Error rate was consistent across inside oven HDDs, it was not selected due to the similarity with the outside oven HDD.

The attributes which remained consistent across all the HDDs are Reported Uncorrect and Hardware ECC Recovered. Figure 7.5 shows the trend curves for all these attributes. These two attributes were selected from the feature selection step to assess the current health and the RUL of the HDDs.

Fig. 7.4 - Comparison of seek error rate attribute curve
7.5 Assess health and RUL

A feed forward artificial neural network (ANN) was used to assess the current health and the RUL of the HDD. A two element vector $I = [F_1, F_2]$ representing SMART attribute values of the two features identified in the previous section was used as input vector to the ANN. The
ANN output was a two element vector $O = [O_1, O_2]$. Here $O_1$ represents the current operating time in weeks and $O_2$ represents the current health (1 represents the best health and 0 represents the worst health).

### 7.5.1 Pre-processing of the Input Data

The sample interval for tapping the SMART attributes was kept as one minute. It gave a large number of samples over the life time of the HDDs. So to reduce the sample size and to minimize the fluctuations in the ANN input, the attribute values were averaged for every 100 samples. It was observed that it increased accuracy of the ANN.

### 7.5.2 Calculation of the Expected Health and RUL

Calculating the expected RUL was easy and could be derived from the current time and the death of the HDD. Calculation of the expected health was a tricky part. For this purpose, the two attributes selected through feature selection were averaged out. Then a trend curve for the averaged graph was taken. With the help of this trend curve equation, the expected health was calculated by normalizing the Y axis values between “0” and “1”.

### 7.5.3 Results

The ANN results for one of the HDD is shown in Figure 7.3.
Fig. 7.6 - NN results for health and RUL of the HDDs
7.6 Discussion

As seen from the Figure 7.3, the results from the ANN were just satisfactory. There was a lot of scope for improvement. Also the predicted curve had steps which could be strongly correlated with one of the inputs, Reported Uncorrect. This created a need for further research.

While the experiment was going on and the ANN results were out, following observations were made to get the answer for the question “Why SMART is not reliable?”

(a) The SMART takes data from the sensors mounted inside the HDD and stores the data into the memory allocated by the firmware. This process is done in real time so that the user gets the current value. Now, if the HDD experienced problems sometime during past and sensors were not working properly at that time, they may not report the impending failure (or change in SMART attributes).

(b) There are SMART monitoring tools which can predict failure of the HDD. The creators of such tools themselves claim that SMART is not reliable and can predict only 60% of the failures.

(c) As an interesting observation, in Figure 7.1 smartmontools shows that the drive queried for SMART information is “FAILING_NOW”. To our surprise, this message is being shown since months for this drive and neither there was any visible problem with the drive since then nor it has failed.
CHAPTER 8

READ WRITE RATE MONITORING

As stated in the previous chapter, the SMART attributes monitoring cannot be used alone to assess the current health and the RUL of the HDD. Thereafter through various experiments, it was deduced that the read write rate for a HDD may reduce as it approaches to death.

8.1 HYPOTHESIS

After killing numerous HDDs, a calculated guess was made about the HDD behaviour. The hypothesis made for this set of experiments was that the read write rate of a HDD decreases over time provided that the operating conditions remain same.

Unlike the SMART attributes, the reduction in read write rate is universal. The universal word here means that it is independent of the make of the HDD. The above statement also means that if at the inception the HDD read write speed is x Bytes/sec, then it will gradually decrease over time to x/2, x/4, x/10, x/100, x/1000 and x/10^6 or 0. Following set of experiment was carried out to confirm the same by following the setps of the proposed model.

8.2 IDENTIFY THE DEGRADATION SIGNATURE

As stated earlier, read write rate becomes the degradation signature. There may be many factors responsible for this. Through the microscopic observation, it was clear from the black
dots on the head surface that the it deteriorates over time. This may be the prime reason for reduction in read write rate.

8.3 CONDITION MONITORING PROCEDURE AND SETUP

Monitoring the read write rate of the HDD was a challenging task. To solve that, various ways were tried out. Out of them, finally one was chosen on the basis of reliability and accuracy as described by following sub sections.

8.3.1 Windows Task Scheduler

The Microsoft Task Scheduler [85] is one of the tools that comes by default with the Windows installation (confirmed on 98, NT, 2000, XP, server 2003 and Vista). This tool is accessible through the “Control Panel” folder. This tool is used to schedule tasks. A task can be defined as a complete set comprising one or more execution operations, known as “actions” along with the launching conditions. For example, automate web testing and site monitoring or system tools like Disk Defragmenter can be scheduled to run at the specified time through the scheduler. One can also schedule back ups with this tool. It can also run the given task in a loop where the frequency can be set daily, monthly or a custom duration with precision till seconds. It can also be extended to schedule tasks at system start up. Command line operations and VBS scripts can also be run from the Windows task scheduler.

The task scheduler has a concept of “trigger” which implies a condition change in the system. A trigger is a set of rules that will cause the task to be executed. Windows had designed two kinds of triggers, time-based and event-based. If a certain task is time-based, it is fired up when the given time is reached. The event-based task is fired up as soon as that event occurs.
There are many such schedulers each offering different functionalities for different operating systems in the market, but Windows Task Scheduler was chosen as it comes embedded in the Windows requiring no additional information and on the top of that it served the purpose very well.

8.3.2 Read Write Rate Monitoring

The read write rate monitoring was made very easy by the Windows Vista task scheduler. Following set of procedure describes the set of instructions carried out to schedule read write rate monitoring task in the Windows scheduler.

(a) Open the “Performance Monitor” window through control panel.

(b) Expand the “Data Collector Sets” option in the left panel.

(c) Right click on the “User Defined” tab. Select “New–Data Collector Set”.

(d) When the window pops up, enter the name for the new data collector set (for example, RWR Monitor); select the “Create manually” radio button and click “Next”.

(e) Under the “Create data logs” radio button section, check mark on “Performance counter” and click “Next”.

(f) Click “Add” to add the performance counters.

(g) When the window pops up, expand the “Physical Disk” tab and add the parameters disk bytes/sec, disk read/sec, disk write/sec. Click “OK” to close the window. Click
finish to close the window and you will come back to “Performance Monitor” window.

(h) Now expand “User defined” drop down in the “Reliability and Performance Monitor” window; select the “RWR Monitor” data collector set in the left panel.

(i) Right click on the “DataCollector01” performance counter on right panel; select “Properties”.

(j) A new window pops up. In the “Performance Counters” tab, change the “Log Format” to tab separated so that the data will be understood by MS Excel. Make sure the sampling interval is 60 second. Select the “Maximum Samples” check box and change 0 to 1,000,000,000. Click “OK” to close the window.

(k) Navigate to “File” tab. Give the log file name (HDDx in this case) and click “OK”.

On the Windows Vista, the log file for the scheduler is stored in “C:\PerfLogs\RWR Monitor” folder with the given file name. To run the task scheduler, right click on the “RWR Monitor” in the “Reliability and Performance Monitor” window and select “Start”.

After monitoring these settings on all the computer systems, the setup looks as shown in Figure 8.1. The random read write C program was ran along with the read write monitoring through Windows task scheduler. This set of experiment contained a sample of only three HDDs as it was meant for the confirmation of the hypothesis.
8.4 Filtering (Feature Extraction) Important Data From the Degradation Signature

If the HDDs goes out of detection, the random read write C program halts and the read write rate goes down. Windows task scheduler records very small values for the read write rate during this period. Such events were confirmed through the log file created by the random read write C program and the relevant entries were removed.

After filtering the erroneous data from the log file, the read write rate graph was plotted as shown in Figure 8.2.
8.5 CONCLUSION

This verifies the hypothesis made at the beginning of the chapter. The read write rate can be used as one of the degradation signatures.
CHAPTER 9

ASSESSMENT OF CURRENT HEALTH AND RUL OF THE HDDs

During the exhaustive investigation for the past one and a half year, the experiment procedure went through numerous refinements. After that the final set of experiments was carried out to get all the necessary data to verify the proposed model for the HDDs. The previous data collected was discarded as it lacked at least one or two setup requirements. The steps described by the proposed model were followed one last time as described next.

9.1 IDENTIFY DEGRADATION SIGNATURE

The degradation signatures for the HDDs were indentified as

(a) Reported Uncorrect (SMART Attribute).

(b) Hardware ECC Recovered (SMART Attribute).

(c) Read Write Rate.

9.2 PERFORM CONDITION MONITORING

The refined condition monitoring setup is described next.

(a) Windows Vista is the operating system.
(b) The SMART attributes are monitored through the C program (in conjunction with SmartMonTools) that was developed in the previous chapter.

(c) The read write rate is monitored through the Windows Task Scheduler.

(d) The C program developed in chapter 6 is used for the random read, write and delete operations on the HDDs during the experiment.

(e) The computer systems as described in Appendix G will be used for this set of experiments.

(f) One HDD will be kept outside the oven to create the final reference curves.

(g) The HDDs will be inside the oven right after formatting and without the initial write (of 80% data of the total capacity).

(h) The HDD will be considered dead if all of the following conditions are met:

   (i) if it is not detected by the same computer after several attempts.

   (ii) if it is not detected by the same computer after changing the data cable with a new one.

   (iii) if it is not detected by the same computer after changing the power cable with a new one.

   (iv) if it is not detected by a different computer.

(i) The temperature of the oven is set at 95º C.
(j) The room temperature is kept at 23º C.

(k) The oven is not opened unless one of the HDDs dies or halts.

(l) The interval between two samples of the SMART attribute monitoring and read write rate monitoring is 60 seconds.

(m) Data for 12-15 HDDs is be collected to calculate the results. The oven can take only 6 HDD at a time, so the data is collected in two batches (6 HDDs in each batch) by putting one batch after the other. One batch could take around 45 days, so the total duration of the experiment was expected to be more than 3 months.

(a) Seven Computers, parallel to serial hub and oven setup
9.3 Feature Selection

All the data collected through the programs was readily available for feature extraction. Initially, the data was filtered for erroneous records. If a HDDs goes out of detection, the SMART attribute monitoring C program records erroneous values for those samples. At the same time, the Windows task scheduler records very small values for the read write rate. Such events were confirmed through the log file created by the random read write C program and the those entries were removed.
Then the parameters showing the trends were selected, which were quite obvious at this time — “Reported Uncorrect”, “Hardware ECC Recovered” and “Read Write Rate”.

To reduce the fluctuations in the data, especially the read write rate, the selected parameters were averaged out for every 100 samples.

**9.4 Assess Health and RUL**

During the experiment, few of the HDDs died prematurely. So data for such HDDs was discarded and those HDDs were not considered for the calculation. Out of the targeted HDDs (#13), 7 were used to train an ANN and 6 were used to test the ANN. The input to the ANN are the three features selected in the previous step as shown in Figure 9.2. The output vector has four nodes each one indicating the active state for the current health of the HDD. The flag “1” indicates the active state whereas the flag “0” indicates non-active state.

**9.4.1 Criterion for selecting the training HDD for a ANN**

It was observed that the ANN, if trained and tested with any arbitrary HDDs, calculates the current health and RUL with very low accuracy. So after a trial and error, a criterion was put for the HDD to train the ANN.

Different HDDs have different lives. So 9 slots were finalized which will cover entire range a HDD can fall in as shown in Table 4. Minimum number of the HDDs covering maximum number slots were selected for the training of the ANN so that the ANN will learn behaviours of almost all the HDDs with lives covering all slots. A second filter was also kept in case there are more than one HDD for the slot. The HDD close to the mean of the slot would be
chosen as the training HDD. As not all the slots were covered (due none of the HDD age falling into that slot), a HDD close to that slot was chosen.

![Diagram of ANN to assess health and RUL of a HDD](image)

Where,
- **RU** - Reported uncorrect
- **ECC** - Hardware ECC recovered
- **RWR** - Read write rate
- **LN** - Like new state
- **HL** - Healthy state
- **AG** - Aged / Deteriorated state
- **NR** - Needs replacement state

Fig. 9.2 - ANN to assess health and RUL of a HDD

Table 4 also shows the slots covered during the training of the ANN. As none of the HDD felt into slot 4 and slot 8 during the experiment, those slots remained open.

**Table 4 - Slots for the training HDDs**

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Slots</th>
<th>HDD</th>
<th>Sr. No.</th>
<th>Slots</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-6 months</td>
<td>HDD3</td>
<td>6</td>
<td>36-48 months</td>
<td>HDD2</td>
</tr>
<tr>
<td>2</td>
<td>6-12 months</td>
<td>HDD12</td>
<td>7</td>
<td>48-60 months</td>
<td>HDD6</td>
</tr>
<tr>
<td>3</td>
<td>12-18 months</td>
<td>HDD5</td>
<td>8</td>
<td>60-72 months</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>18-24 months</td>
<td>-</td>
<td>9</td>
<td>72+ months</td>
<td>HDD7</td>
</tr>
<tr>
<td>5</td>
<td>24-36 months</td>
<td>HDD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
9.4.2 Training of the ANNs

The monitored data after filtering was fed to an ANN as described in the above table. Before training the ANN, it was necessary to construct a expected output vector for the ANNs (which contained a single element, current health). The current health is calculated as follows.

The three attributes selected through feature selection were accordingly normalized between “0” and “1” (“0” indicating bad situation and “1” as good situation). Then an average value of those three was calculated. A trend curve was fitted into the plot of the averaged values. Each point on the trend curve represents the health of the HDD at that time. Then the Y axis of the curve was divided into 4 bands to represent 4 health states of the HDDs as described in Table 5.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Band</th>
<th>Health State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7 &lt; H</td>
<td>Like New (LN)</td>
</tr>
<tr>
<td>2</td>
<td>0.3 &lt; H &lt; 0.7</td>
<td>Healthy (HL)</td>
</tr>
<tr>
<td>3</td>
<td>0.3 &lt; H &lt; 0.2</td>
<td>Aged or Deteriorated (AG)</td>
</tr>
<tr>
<td>4</td>
<td>0.2 &lt; H</td>
<td>Needs replacement (NR)</td>
</tr>
</tbody>
</table>

Once again the same ANN C program was used for training the ANNs. The parameters, gain for modifying weights and threshold, number of iterations and number of hidden nodes, were varied and tested for the best possible error during testing the ANNs. At the same time, the input values were normalized to improve value of the error during training ANNs.
9.4.3 Calculate the Current Health and the RUL

The data enriched through feature extraction step was fed to the ANN trained in the previous step and the current health and RUL was assessed as follows.

The correlation between the current health and the RUL is described in Figure 9.3. The values given in the figure are based on the experience from the experiments carried out. All the values are subjective.

![Fig. 9.3 - RUL for different health states](image)

The difference between ANN predicted health state and the actual state was calculated on instance basis. For a total of $n$ instances, if the ANN health prediction is correct for $m$ instances then the accuracy is given by the following equation

$$\text{Accuracy} = \frac{m}{n} \times 100$$

9.5 RESULTS

Table 6 shows the final results calculated for the remainder of the HDDs in the experiment. So the overall accuracy of the model for the experiments carried out was calculated to 86.83%
Table 6 - Accuracy of ANN prediction for the remainder HDDs

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>HDD</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HDD4</td>
<td>89.13%</td>
</tr>
<tr>
<td>2</td>
<td>HDD8</td>
<td>87.37%</td>
</tr>
<tr>
<td>3</td>
<td>HDD9</td>
<td>82.97%</td>
</tr>
<tr>
<td>4</td>
<td>HDD10</td>
<td>89.91%</td>
</tr>
<tr>
<td>5</td>
<td>HDD14</td>
<td>84.8%</td>
</tr>
<tr>
<td>6</td>
<td>HDD13</td>
<td>96.9%</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td></td>
<td><strong>88.51%</strong></td>
</tr>
</tbody>
</table>
CHAPTER 10

CONCLUSION AND FUTURE RESEARCH

10.1 CONCLUSION

Though HDD are proving to be the most important subunit of the computer, very little work has been done on its failure prediction. In this information technology age, the data size is growing humongous while at the same time duplicate data is being created through the process of back up. In contrast, life may become easier if there is a system which can sound alarm before storage media, i.e. the HDD, fails.

Through this thesis, a prototype was developed which assesses the current health and the RUL of the HDDs. After knowing the RUL, required maintenance actions can be planned and the invaluable data can be saved without any need for backup data repositories. Initially, use of sensors was investigated for the same. It was concluded that sensors are not useful to identify the degradations signature in case of HDDs. Then SMART attributes and read write rate were identified as the degradation signatures. Subsequently through an experimental setup, the current health and the RUL were assessed thus proving the proposed model. The overall accuracy given by the model is 88.51%. The HDDs are subject to accelerated degradation to speed up the experiments.
10.2 **Future Research**

When the HDD is near to death, sometimes the SMART attribute values recorded by the C program are quite large (of the order of $10^6$). Though we have filtered out such values due to consistency reason, it has not been completely verified that this is not a signature of the death of the HDD. If an analogy is made with human body, it seems logical. Human body is also not able to carry out internal and external functions in the old age and not able to report the data stored in the memory as the memory cells get weak.
REFERENCES


74. Get S.M.A.R.T. for Reliability. [cited; Seagate White Paper]. Available from: 

75. Tandon, N., V.P. Agrawal, and V.V.P. Rao, Vibration monitoring of computer hard disk 

76. Ki Myung, L. and A.A. Polycarpou. Transient and steady state dynamic vibration 

77. Ki Myung, L. and A.A. Polycarpou, Noninvasive vibration diagnostics to detect head-

78. Tandon, N., V.V.P. Rao, and V.P. Agrawal, Vibration and noise analysis of computer 

Learning Research, 2005. 6: p. 34.

80. Hughes, G.F., et al., Improved disk-drive failure warnings. IEEE Transactions on 


83. Comparison of S.M.A.R.T. tools. [cited; Available from: 

84. SMARTCTL Documentation. [cited; Available from: 
APPENDIX A

RANDOM READ/WRITE JAVA PROGRAM

This program has two modes, 1. random read write and 2. prefilling the HDD with the initial data. For prefilling the HDD with intial data, pass “COPY” in the argument while running the program. To operate the program in random read write mode, pass “RUN” in the argument while running the program.

```java
import java.io.*;
import java.util.*;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class HardDriveV3 {
    static final int MILLION = 1*1000*1000;   //Constant
    static final int THOUSAND =1000;    //Constant
    static final int TOTALNUMBEROFFILES = 50*1000; //Constant
    static final int FILESIZEINKB = 4*1000;   //Constant

    Random generator = new Random();   //To generate random number
```
static Logger logger;  //Variable to log data

static
{
    try
    {
        boolean append = true;

        FileHandler fh = new FileHandler("/media/sdb1/
            HardDrive1RadialScratch.log", append);  //for Knoppix
        //FileHandler fh = new FileHandler("c:/Yogesh/JavaProgram
            /HardDriveLog.log", append);  //for Windows
        fh.setFormatter(new SimpleFormatter());

        logger = Logger.getLogger("TestLog");
        logger.addHandler(fh);
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}

static void copy(HardDriveV3 ahd, int aIndex)

    //creates TOTALNUMBEROFFILES on the HDD
{  
    for(int i=aIndex; i<=TOTALNUMBEROFFILES; i++)  
    {
        ahd.recreateFile("file"+i+".bin");
    }
}  

static void process(HardDriveV3 ahd) throws Exception
    //calls its subprocesses inside an infinite loop
{
    while(true)
    {
        TreeSet<Integer> ts = ahd.readAndDelete();
        ahd.recreateFile(ts);
    }
}  

TreeSet<Integer> readAndDelete() throws Exception
    //It reads and deletes random files
{
    int numberFilesToBeDeleted = generator.nextInt
(TOTALNUMBEROFFILES/20);
    //generates random number between 0 and TOTALFILES/2
//This number of files will be read and deleted
int[] array = new int[numberFilesToBeDeleted];

//Filenames containing these numbers will be read and deleted

//System.out.println("\n\nnumberFilesToBeDeleted = 
"+numberFilesToBeDeleted+"\n");

    //to debug

logger.info("numberFilesToBeDeleted = 
"+numberFilesToBeDeleted);

TreeSet<Integer> ts = new TreeSet<Integer>();

    /*This set stores unique numbers. Filenames containing these numbers
will be read and deleted. This will be subsequently used to create those
files again
*/

for(int i=0; i<numberFilesToBeDeleted; i++)

    //generates random numbers and adds to TreeSet
    //automatically ignores duplicate numbers
{

    int filenameToBeDeleted = generator.nextInt
(TOTALNUMBEROFFILES);

    array[i] = filenameToBeDeleted;

ts.add(filenameToBeDeleted);

    //adds "filenameToBeDeleted" element to TreeSet

}

//System.out.println(ts);  //to debug

for(int z=0; z<array.length; z++)
    //Calls read and delete methods on random files
    {
        int ii = array[z];
        //printTime();  //prints current time
        Logger.info("reading file"+ii+".bin");
        readFile("file"+ii+".bin");  //reads specified file
        deleteFile("file"+ii+".bin");  //deleted specified file
    }
    return ts;

}

void recreateFile(TreeSet<Integer> ats)

    //It creates files deleted by process method
    //It takes file name from the TreeSet
    {
        Iterator<Integer> iterator = ats.iterator();
        //Iterator to iterate over TreeSet
while (iterator.hasNext())
{
    int ii = iterator.next();
    //printTime();
    recreateFile("file"+ii+.bin);
}

void recreateFile(String fileName)
{
try
{
    File inputFile = new File("original.bin");
    File outputFile = new File(fileName);
    FileReader in = new FileReader(inputFile);
    FileWriter out = new FileWriter(outputFile);
    int c;

    while ((c = in.read()) != -1)
    {
        out.write(c);
    }
    in.close();
}
out.close();
logger.info(fileName+" recreated");
}

void createFile(String fileName)  //creates a binary file
{
    DataOutputStream dos = null;
    FileOutputStream fos = null;

    try
    {
        fos = new FileOutputStream(fileName);
        dos = new DataOutputStream(fos);

        for(int i=0; i<FILESIZEINKB*250; i++)
        {
            dos.writeInt(i);
        }
    } catch(Exception e)
    {
    
    }
e.printStackTrace();
}

Finally
{

try
{

dos.close();

fos.close();

//System.out.println(fileName+" created");

logger.info(fileName+" created");

} catch (IOException e)
{

e.printStackTrace();

}

}

void readFile(String fileName) throws Exception
{

FileInputStream fis = null;

DataInputStream dis = null;

try
{

fis = new FileInputStream(fileName);
dis = new DataInputStream(fis);

for(int i=0; ; i++)
{
    dis.readInt();
}
} catch (FileNotFoundException e)
{
    recreateFile(fileName);
    fis = new FileInputStream(fileName);
    dis = new DataInputStream(fis);
} catch(EOFException eof)
{
    //System.out.println(fileName+" read");
    logger.info(fileName+" read");
}

catch(Exception e)
{
    e.printStackTrace();
}finally
{
    try

void deleteFile(String fileName)
{
    File target = new File(fileName);
    target.delete();
    //System.out.println(fileName + " deleted");
    logger.info(fileName + " deleted");
}

void printTime()
{
    Calendar cal = new GregorianCalendar();
    // Get the components of the time
int hour24 = cal.get(Calendar.HOUR_OF_DAY); // 0..23
int min = cal.get(Calendar.MINUTE); // 0..59
int sec = cal.get(Calendar.SECOND); // 0..59
int ms = cal.get(Calendar.MILLISECOND); // 0..999
int ampm = cal.get(Calendar.AM_PM); // 0=AM, 1=PM
System.out.println(hour24+":"+min+":"+sec+":"+ms);
}

public static void main(String[] args) throws Exception
{
    logger.info("Program started ");
    if(args.length<1)
    {
        System.out.println("Please enter COPY or RUN");
        System.exit(1);
    }
    HardDriveV3 hd = new HardDriveV3();
    //constructs an object and writes TOTALNUMBEROFFILES to HDD
    if(args[0].equals("COPY"))
    {
        if(args.length<2)
        {
            //...
        }
    }
System.out.println("Please enter the index from where copying should start");
System.exit(1);

hd.createFile("original.bin");
//creates a base file from which other files will be copied

int index = Integer.parseInt(args[1]);
copy(hd, index);

hd.createFile("original.bin"); //creates the first object
process(hd); //calls top level process
}
APPENDIX B

SENSOR SPECIFICATIONS

Sensor Selection: Piezoelectric Accelerometers are the most preferred vibration monitoring sensors. The sensor specifications are:

1. **Vibration Level:** $\geq 1.04$G

   *(Reason – Maximum vibration acceleration specified for the target HDD is 1.04G, so sensor with maximum G above 1.04 is acceptable.)*

2. **Frequency Range:** 2Hz to 15KHZ

   *(Reason – As suggested by the reference literatures for monitoring the head actuator Vibrations.)*

3. **Temperature Range:** 2° to 65° C

   *(Reason – Determined by taking in consideration the ambient temperature range of target HDD)*

4. **Corrosive Chemicals:** Not Present.

5. **Combustible Atmosphere:** No.

6. **Intense Acoustic and Electromagnetic Fields:** Present.

7. **High Electrostatic Discharge:** No

8. **Vibrating Device Grounded:** Yes
9. **Sensor Size and Weight**: Recommended less than 3mm at each corner and 2.5gram in weight.

*(Reason – The Literature on vibration monitoring using increasing weights on actuator arm suggests the maximum tolerable weight before vibration becomes uncontrollable is 2-4grams. Further the Non Invasive Diagnostics suggests similar weight and size constraints.)*

10. **Sensitivity**: 100mV/G is recommended but 10mV/G is also acceptable, given the fact the Peak G level of vibration does not exceed the acceleration range of sensor.
APPENDIX C

HITACHI HDD SPECIFICATIONS

Model – Deskstar P7K500

<table>
<thead>
<tr>
<th>Interface</th>
<th>PATA-133</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (GB)</td>
<td>250</td>
</tr>
<tr>
<td>Data heads (physical)</td>
<td>2</td>
</tr>
<tr>
<td>Data disks</td>
<td>1</td>
</tr>
<tr>
<td>Max. Areal density (Gbits/sq. in.)</td>
<td>185</td>
</tr>
</tbody>
</table>

**Performance**

<table>
<thead>
<tr>
<th>Data buffer (MB)</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotational speed (RPM)</td>
<td>7200</td>
</tr>
<tr>
<td>Media transfer rate (Mbits/sec, max)</td>
<td>1138</td>
</tr>
<tr>
<td>Interface transfer rate (MB/sec, max)</td>
<td>133</td>
</tr>
</tbody>
</table>

**Power**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>+5 VDC (+/-5%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+12 VDC (+/-10%)</td>
</tr>
<tr>
<td>Startup current (A, max.)</td>
<td>2.0 (+12V), 1.1 (+5V)</td>
</tr>
<tr>
<td>Idle (W)</td>
<td>3.3 (1 disk)</td>
</tr>
</tbody>
</table>

**Physical size**

<table>
<thead>
<tr>
<th>z-height (mm)</th>
<th>26.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions (width x depth, mm)</td>
<td>101.6 (+/-0.25) x 147</td>
</tr>
<tr>
<td>Weight (typical, g, max.)</td>
<td>550</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----</td>
</tr>
<tr>
<td><strong>Environmental (operating)</strong></td>
<td></td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>0° to 60° C</td>
</tr>
<tr>
<td>Relative humidity (non-condensing)</td>
<td>8% to 90%</td>
</tr>
<tr>
<td>Shock (half-sine wave)</td>
<td>70</td>
</tr>
<tr>
<td>Vibration, random (G RMS 5 to 500 Hz)</td>
<td>0.67 (XY)</td>
</tr>
<tr>
<td><strong>Environmental (non-operating)</strong></td>
<td></td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>-40° to 70° C</td>
</tr>
<tr>
<td>Relative humidity (non-condensing)</td>
<td>5% to 95%</td>
</tr>
<tr>
<td>Shock (half-sine wave, G (2ms))</td>
<td>350</td>
</tr>
<tr>
<td>Vibration, random (G RMS 5 to 500 Hz)</td>
<td>1.04 (XYZ)</td>
</tr>
</tbody>
</table>
APPENDIX D

UNIX SCRIPT

This script is designed for pre-filling the HDD with initial data.

#!/bin/sh
i=0
while [ $i -le 50000 ]
do
  cp /media/sdb1/original.bin /media/hda1/
  mv /media/hda1/original.bin /media/hda1/file"$i".bin
  echo "file$i.bin copied"
  let i=i+1
done
### APPENDIX E

#### SEAGATE HDD SPECIFICATIONS

Model number - ST3250410AS

<table>
<thead>
<tr>
<th>Interface</th>
<th>Serial ATA-300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (GB)</td>
<td>250</td>
</tr>
<tr>
<td>Data heads (physical)</td>
<td>2</td>
</tr>
<tr>
<td>Data disks</td>
<td>1</td>
</tr>
<tr>
<td>Max. Areal density (Gbits/sq. in.)</td>
<td>163</td>
</tr>
</tbody>
</table>

**Performance**

| Data buffer (MB)              | 16                      |
| Rotational speed (RPM)        | 7200                    |
| Drive transfer rate (MB/sec, max) | 300                   |
| Seek Time (average)           | 11ms                    |

**Power**

| Requirement                   | +5 VDC (+/-5%)          |
|                               | +12 VDC (+/-10%)        |
| Startup current (A, max.)     | 2.7 (+12V), 0.9 (+5V)   |
| Idle (W)                      | 9.3 (1 disk)            |

**Physical size**

<p>| z-height (mm)                 | 20.32                   |
| Dimensions (width x depth, mm) | 101.6 (+/-0.25) x 147.3 |</p>
<table>
<thead>
<tr>
<th><strong>Weight (typical, g, max.)</strong></th>
<th>365.7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environmental (operating)</strong></td>
<td></td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>0° to 60° C</td>
</tr>
<tr>
<td>Relative humidity (non-condensing)</td>
<td>5% to 90%</td>
</tr>
<tr>
<td>Shock (half-sine wave)</td>
<td>63</td>
</tr>
<tr>
<td>Vibration, random (Gs 5 to 500 Hz)</td>
<td>0.25</td>
</tr>
<tr>
<td><strong>Environmental (non-operating)</strong></td>
<td></td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>-40° to 70° C</td>
</tr>
<tr>
<td>Relative humidity (non-condensing)</td>
<td>5% to 95%</td>
</tr>
<tr>
<td>Shock (half-sine wave, G (2ms))</td>
<td>350</td>
</tr>
<tr>
<td>Vibration, random (Gs 5 to 500 Hz)</td>
<td>0.25</td>
</tr>
</tbody>
</table>
APPENDIX F

RANDOM READ/WRITE C PROGRAM

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

#define MAX_FILES_TO_DEL 2500  /* Maximum number of files to delete*/
#define FILES_ON_DISK 80000  /* Total number of files on disk */
#define FILE_SIZE 4000 /* File size in KB */
#define LOG_FILE "D:/NewH13-3.log"

/* File name for log file */
#define DATA_FILE "F:/file%d.bin"

/* File name for log file */

#include <time.h>

int Create_Random_Number_Array();
void Write_Cpu_Time();
void Delete_Files();
void Write_Files();
void Write_Log_File();
char* Time_And_Date();
/*----------------------- Main program -------------------------------*/

main ()
{
    unsigned int Rand_Set[MAX_FILES_TO_DEL];
    int Nfiles;
    clock_t time_start, time_finish;
    int readymade_magic[FILE_SIZE];
    int j;

    Write_Log_File("Program started");
    printf("Program started");

    time_start = clock();
    ta=time(NULL);

    for(j=0; j< FILE_SIZE; j++){
        readymade_magic[j] = (int) (33 + (126-33)*((double) rand()/(double) RAND_MAX));
    }

    while(1){

    }
}
Nfiles = Create_Random_Number_Array(Rand_Set);
Delete_Files(Nfiles, Rand_Set, readymade_magic);
Write_Files(Nfiles, Rand_Set, readymade_magic);
}

time_finish = clock();
Write_Cpu_Time("rw.c","cpu.out", time_start, time_finish);
}
/*-------------------------- End of the main() -----------------------*/

/----------------- Create_Random_Number_Array() ---------------------*/
int Create_Random_Number_Array(rand_set)
unsigned int *rand_set;
{
    int nfiles, i, j;
    unsigned int iseed = (unsigned int)time(NULL);
    char Log_Entry[150];
    srand (iseed);
    nfiles = (int) (1 + (MAX_FILES_TO_DEL-1)*((double) rand()/(double) RAND_MAX));
    sprintf(Log_Entry,"\n\nNumber of files to be deleted = %d \n\n",nfiles);
    Write_Log_File(Log_Entry);
    printf("\n\nNumber of files to be deleted = %d \n\n",nfiles);
for(i=0; i<nfiles; i++){
    rand_set[i] = (unsigned int) (1 + (FILES_ON_DISK-1)*((double)
        rand()/(double) RAND_MAX));
    j=0;
    do{
    if(rand_set[i]==rand_set[j++]){  
        rand_set[i] = (unsigned int) (1 + (FILES_ON_DISK-1)*((double) rand()/(double) RAND_MAX));
        j=0;
    }
    }while(j<i);
}
return nfiles;

/*---------------------- Delete_Files() ------------------------------*/
void Delete_Files(nfiles, rand_set, readymade_magic)
int nfiles;
unsigned int *rand_set;
int *readymade_magic;
{
    FILE *inputfile_fp;
FILE *outputfile_fp;
char file_name[80];
char Log_Entry[150];
char magic;
int i, j;
int byte;

for(i=0; i<nfiles; i++) {
    sprintf(file_name, DATA_FILE, rand_set[i]);
    inputfile_fp = fopen(file_name, "r");
    if (!inputfile_fp) {
        outputfile_fp = fopen(file_name, "w");
        for(byte=0; byte< 1000; byte++) {
            for(j=0; j< FILE_SIZE; j++) {
                if(((j+1) % 80)==0) {
                   putc(\n', outputfile_fp);
                } else {
                    putc(readymade_magic[j], outputfile_fp);
                }
            }
        }
        fclose(outputfile_fp);
    }
}

fclose(outputfile_fp);
inputfile_fp = fopen(file_name,"r");
}

do{
    magic = getc(inputfile_fp);
    /*if (magic != EOF) {
        printf("%c",magic);
    }*/
} while (magic != EOF);
fclose(inputfile_fp);
if( remove(file_name) == -1 ) {
    sprintf(Log_Entry,"%s **** Error deleting %s ****
\n",Time_And_Date(), file_name);
    Write_Log_File(Log_Entry);
    if(difftime(time(NULL), ta) > 5.0) {
        ta = time(NULL);
        printf("%s **** Error deleting %s ****
\n",Time_And_Date(), file_name);
    }
    exit(1);
} else {
    sprintf(Log_Entry,"%s **** %s deleted ****
\n",Time_And_Date(), file_name);
Write_Log_File(Log_Entry);

if(difftime(time(NULL), ta) > 5.0) {
    ta = time(NULL);

    printf("%s **** %s deleted ****\n", Time_And_Date(), file_name);

    printf("File Count = %d\n", i);
}
}

return;
}

/*------------------------ Write_Cpu_Time() ------------------------*/
void Write_Cpu_Time(program_name, outputfile, start, finish)
char *program_name;
char *outputfile;
clock_t start;
clock_t finish;
{
    FILE *time_fp;
    float cpu_time_sec;
    float cpu_time_min;
cpu_time_sec = (float) (finish - start)/CLOCKS_PER_SEC;
cpu_time_min = (float) cpu_time_sec/60;
time_fp = fopen(outputfile,"w");

if(cpu_time_sec < 60.0) {
    fprintf(time_fp,"\n Execution time (sec) for %s = %f \n\n", program_name, cpu_time_sec);
} else {
    fprintf(time_fp,"\n Execution time (min) for %s = %f \n\n", program_name, cpu_time_min);
}
fclose(time_fp);
return;
}

/*--------------------------- Write_Files() ------------------------*/
void Write_Files(nfiles, rand_set, readymade_magic)
int nfiles;
unsigned int *rand_set;
int *readymade_magic;
{
    FILE *outputfile_fp;
    char file_name[80];
char Log_Entry[150];
int i, j;

int byte;

int nchar;

nchar = (int) ((float)FILE_SIZE*1000/(float) sizeof(char));

for(i=0; i<nfiles; i++) {
    //i=0;
    //rand_set[0] = 1;
    sprintf(file_name,DATA_FILE,rand_set[i]);
    outputfile_fp = fopen(file_name,"w");
    for(byte=0; byte< 1000; byte++) {
        for(j=0; j< FILE_SIZE; j++) {
            if(((j+1) % 80)==0) {
                putc("n", outputfile_fp);
            } else {
                putc(readymade_magic[j], outputfile_fp);
            }
        }
    }
    fclose(outputfile_fp);
if(difftime(time(NULL), ta) > 5.0) {
    ta = time(NULL);
    printf("%s %s has been written \n", Time_And_Date(), file_name);
    printf("File Count = %d\n", i);
}

sprintf(Log_Entry,"%s %s has been written \n", Time_And_Date(),
        file_name);
Write_Log_File(Log_Entry);
}
return;

/----------------------- Time_And_Date() --------------------------*/
char* Time_And_Date()
{
    struct tm *local;
    time_t t;
    t = time(NULL);
    local = localtime(&t);
    // printf("Local time and date: %s\n", asctime(local));
    // local = gmtime(&t);
    // printf("UTC time and date: %s\n", asctime(local));
return asctime(local);

}

/***************************************************************************/

void Write_Log_File(log_entry)
char *log_entry;
{
    FILE *logfile_fp;

    logfile_fp = fopen(LOG_FILE,"a");
    fprintf(logfile_fp,"%s \n",log_entry);
    fclose(logfile_fp);

    return;
}
APPENDIX G

VOSTRO 200 MINI TOWER CONFIGURATION

Intel® Core™ 2 Duo Processor E7200 (2.53GHz, 3M, 1066MHz FSB)

Intel® G31 (ICH7) Express Chipset

Genuine Windows Vista® Home Basic, SP1, with media, 32 Edition English

1.0GB DDR2 Non-ECC SDRAM, 800MHz, (1DIMM)

80GB 7200 RPM SATA 3.0Gb/s and 8MB DataBurst Cache™

16X DVD-ROM SATA, Data Only

Integrated Video, Intel® GMA3100

Standard I/O Ports

- (1) VGA
- (8) USB 2.0 2 front, 6 rear Optional via PCI add-in card
- PS2 (Optional via PCI add-in card)
- Ethernet (RJ45)
- Serial (9-pin) (16550 compatible)
- 1 parallel (25-hole, bi-directional)
- Line-in (stereo/microphone)
- Line-out (headphone/speaker)

Dimensions

Height: 16.10"/40.89 cm
Width: 7.36"/18.69 cm
Depth: 17.52"/44.50cm
Weight: 27.2lb/12.34kg

**Slots**

- 2 full height PCI - (H: 4.2" X L: 11")

- 1 PCIe x16 full height graphics, - (H: 4.2" X L: 9")
### APPENDIX H

**SMART ATTRIBUTES**

<table>
<thead>
<tr>
<th>ID</th>
<th>Hex</th>
<th>Attribute Name</th>
<th>Better</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01</td>
<td>Read Error Rate</td>
<td>🟠</td>
<td>Indicates the rate of hardware read errors that occurred when reading data from a disk surface. A non-zero value indicates a problem with either the disk surface or read/write heads.</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>Throughput Performance</td>
<td>🟠</td>
<td>Overall (general) throughput performance of a hard disk drive. If the value of this attribute is decreasing there is a high probability that there is a problem.</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>03</td>
<td>Spin-Up Time</td>
<td>Average time of spindle spin up (from zero RPM to fully operational [millisecond]).</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>----</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>04</td>
<td>Start/Stop Count</td>
<td>A tally of spindle start/stop cycles.</td>
</tr>
<tr>
<td></td>
<td>05</td>
<td>05</td>
<td>Reallocated Sectors Count</td>
<td>Count of reallocated sectors. When the hard drive finds a read/write/verification error, it marks this sector as &quot;reallocated&quot; and transfers data to a special reserved area (spare area). This process is also known as remapping and &quot;reallocated&quot; sectors are called remaps. This is why, on modern hard disks, &quot;bad blocks&quot; cannot be found while testing the surface — all bad blocks are hidden in reallocated sectors. However, the more sectors that are reallocated, the more read/write speed will decrease.</td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>06</td>
<td>Read Channel Margin</td>
<td>Margin of a channel while reading data. The function of this attribute is not specified.</td>
</tr>
<tr>
<td></td>
<td>07</td>
<td>07</td>
<td>Seek Error Rate</td>
<td>Rate of seek errors of the magnetic heads. If there is</td>
</tr>
</tbody>
</table>
A failure in the mechanical positioning system, a servo damage or a thermal widening of the hard disk, seek errors arise. More seek errors indicates a worsening condition of a disk surface and the mechanical subsystem.

<p>| 08 | 08 | Seek Time Performance | ➺ | Average performance of seek operations of the magnetic heads. If this attribute is decreasing, it is a sign of problems in the mechanical subsystem. |
| 09 | 09 | Power-On Hours (POH) | ➪ | Count of hours in power-on state. The raw value of this attribute shows total count of hours (or minutes, or seconds, depending on manufacturer) in power-on state. |
| 10 | 0A | Spin Retry Count | ➪ | Count of retry of spin start attempts. This attribute stores a total count of the spin start attempts to reach the fully operational speed (under the condition that the first attempt was unsuccessful). An increase of this attribute value is a sign of problems in the hard disk mechanical subsystem. |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0B</td>
<td>Recalibration Retries</td>
<td><img src="up" alt="Up" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This attribute indicates the number of times recalibration was requested (under the condition that the first attempt was unsuccessful). A decrease of this attribute value is a sign of problems in the hard disk mechanical subsystem.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Device Power Cycle Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This attribute indicates the count of full hard disk power on/off cycles.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Soft Read Error Rate</td>
<td><img src="down" alt="Down" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uncorrected read errors reported to the operating system. If the value is non-zero, you should back up your data.</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>BE</td>
<td>Airflow Temperature (WDC)</td>
<td><img src="down" alt="Down" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Airflow temperature on Western Digital HDs (Same as temp. (C2), but current value is 50 less for some models. Marked as obsolete.)</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>BF</td>
<td>G-sense error rate</td>
<td><img src="down" alt="Down" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frequency of mistakes as a result of impact loads</td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>C1</td>
<td>Load/Unload Cycle</td>
<td><img src="down" alt="Down" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Count of load/unload cycles into head landing zone position.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>194</td>
<td>C2</td>
<td>Temperature</td>
<td>Current internal temperature.</td>
</tr>
<tr>
<td>195</td>
<td>C3</td>
<td>Hardware ECC Recovered</td>
<td>Time between ECC-corrected errors.</td>
</tr>
<tr>
<td>196</td>
<td>C4</td>
<td>Reallocation Event Count</td>
<td>Count of remap operations. The raw value of this attribute shows the total number of attempts to transfer data from reallocated sectors to a spare area. Both successful &amp; unsuccessful attempts are counted.</td>
</tr>
<tr>
<td>197</td>
<td>C5</td>
<td>Current Pending Sector Count</td>
<td>Number of &quot;unstable&quot; sectors (waiting to be remapped). If the unstable sector is subsequently written or read successfully, this value is decreased and the sector is not remapped. Read errors on the sector will not remap the sector, it will only be remapped on a failed write attempt. This can be problematic to test because cached writes will not remap the sector, only direct I/O writes to the disk.</td>
</tr>
<tr>
<td>198</td>
<td>C6</td>
<td>Uncorrectable Sector Count</td>
<td>The total number of uncorrectable errors when reading/writing a sector. A rise in the value of this attribute indicates defects of the disk surface and/or problems in the mechanical subsystem.</td>
</tr>
<tr>
<td>Code</td>
<td>Hex</td>
<td>Description</td>
<td>Details</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>199</td>
<td>C7</td>
<td>UltraDMA CRC Error Count</td>
<td>The number of errors in data transfer via the interface cable as determined by ICRC (Interface Cyclic Redundancy Check).</td>
</tr>
<tr>
<td>200</td>
<td>C8</td>
<td>Write Error Rate / Multi-Zone Error Rate</td>
<td>The total number of errors when writing a sector.</td>
</tr>
<tr>
<td>201</td>
<td>C9</td>
<td>Soft Read Error Rate</td>
<td>Number of off-track errors. If non-zero, make a backup.</td>
</tr>
<tr>
<td>202</td>
<td>CA</td>
<td>Data Address Mark errors</td>
<td>Number of Data Address Mark errors (or vendor-specific)</td>
</tr>
<tr>
<td>203</td>
<td>CB</td>
<td>Run Out Cancel</td>
<td>Number of ECC errors</td>
</tr>
<tr>
<td>205</td>
<td>CD</td>
<td>Thermal Asperity Rate (TAR)</td>
<td>Number of thermal asperity errors</td>
</tr>
<tr>
<td>206</td>
<td>CE</td>
<td>Flying Height</td>
<td>Height of heads above the disk surface</td>
</tr>
<tr>
<td>207</td>
<td>CF</td>
<td>Spin High Current</td>
<td>Amount of high current used to spin up the drive</td>
</tr>
<tr>
<td>208</td>
<td>D0</td>
<td>Spin Buzz</td>
<td>Number of buzz routines to spin up the drive</td>
</tr>
<tr>
<td>Code</td>
<td>Column</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>209</td>
<td>D1</td>
<td>Offline Seek Performance</td>
<td>Drive’s seek performance during offline operations</td>
</tr>
<tr>
<td>220</td>
<td>DC</td>
<td>Disk Shift</td>
<td>Distance the disk has shifted relative to the spindle (usually due to shock). Unit of measure is unknown.</td>
</tr>
<tr>
<td>221</td>
<td>DD</td>
<td>G-Sense Error Rate</td>
<td>The number of errors resulting from externally-induced shock &amp; vibration.</td>
</tr>
<tr>
<td>222</td>
<td>DE</td>
<td>Loaded Hours</td>
<td>Time spent operating under data load (movement of magnetic head armature)</td>
</tr>
<tr>
<td>223</td>
<td>DF</td>
<td>Load/Unload Retry Count</td>
<td>Number of times head changes position.</td>
</tr>
<tr>
<td>224</td>
<td>E0</td>
<td>Load Friction</td>
<td>Resistance caused by friction in mechanical parts while operating.</td>
</tr>
<tr>
<td>225</td>
<td>E1</td>
<td>Load/Unload Cycle Count</td>
<td>Total number of load cycles</td>
</tr>
<tr>
<td>226</td>
<td>E2</td>
<td>Load 'In'-time</td>
<td>Total time of loading on the magnetic heads actuator (time not spent in parking area).</td>
</tr>
<tr>
<td>227</td>
<td>E3</td>
<td>Torque Amplification</td>
<td>Number of attempts to compensate for platter speed variations</td>
</tr>
<tr>
<td>Code</td>
<td>Count</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>228 E4</td>
<td>228</td>
<td>Power-Off Retract Cycle</td>
<td>The number of times the magnetic armature was retracted automatically as a result of cutting power.</td>
</tr>
<tr>
<td>230 E6</td>
<td>230</td>
<td>GMR Head Amplitude</td>
<td>Amplitude of &quot;thrashing&quot; (distance of repetitive forward/reverse head motion)</td>
</tr>
<tr>
<td>231 E7</td>
<td>231</td>
<td>Temperature Head Flying Hours</td>
<td>Drive Temperature</td>
</tr>
<tr>
<td>240 F0</td>
<td>240</td>
<td>-</td>
<td>Time while head is positioning</td>
</tr>
<tr>
<td>250 FA</td>
<td>250</td>
<td>Read Error Retry Rate</td>
<td>Number of errors while reading from a disk</td>
</tr>
</tbody>
</table>
APPENDIX I

SMART ATTRIBUTE MONITORING C PROGRAM

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include <windows.h>

#define SKIPLINES           7
  // line to be skipped while reading the temporary file
#define LINELENGTH          400
#define INTERVEL            60000           // Time delay in milliseconds
#define NUM_ATTRIBUTES     18                         // Number of attributes
#define DRIVE_MONITORED    "f:"                       // Drive bring monitored
#define SA_FILE             "d:\sm\ta.txt"           // Smart attribute file name
#define SA_BACKUP_FILE     "d:\sm\sabackup.txt"       // Smart attribute backup file name
#define SA_VALUE_FILE      "d:\sm\SmartValue.csv"
  // Smart current value file name
#define SA_WORST_FILE      "d:\sm\SmartWorst.csv"
void Read_Write_Smart_Attributes();
void Write_Header();
void Read_SA();

main()
{
    int z = 1;
    int f = 1;
    FILE *fp = fopen(SA_VALUE_FILE,"r");
    char command[80];
    if(fp) {
        fclose(fp);
    } else {
        Write_Header();
    }

    while(z==1) {
        system("time /T");
        sleep(INTERVEL);
        system("cd\");  
        if (f==1)
f=0;

else {
    strcpy(command,"del ");
    strcat(command,SA_FILE);
    system(command);
}

strcpy(command,"date /t >> ");
strcat(command,SA_FILE);
system(command);

strcpy(command,"time /t >> ");
strcat(command,SA_FILE);
system(command);

strcpy(command,"smartctl -A ");
strcat(command,DRIVE_MONITORED);
strcat(command," >> ");
strcat(command,SA_FILE);
system(command);

strcpy(command,"date /t >> ");
strcat(command,SA_BACKUP_FILE);
system(command);

strcpy(command,"time /t >> ");
strcat(command,SA_BACKUP_FILE);
system(command);
strcpy(command,"smartctl -A ");
strcat(command,DRIVE_MONITORED);
strcat(command," >> ");
strcat(command,SA_BACKUP_FILE);
system(command);
Read_Write_Smart_Attributes();

return (0);

void Write_Header()
{

FILE *valuefp;
FILE *worstfp;

valuefp = fopen(SA_VALUE_FILE, "a");
if (valuefp == NULL){
    printf("SmartValue file cannot be opened.\n");
    exit(1);
}

/*------------------------- Write_Header ---------------------------*/

}
worstfp = fopen(SA_WORST_FILE, "a");
if (worstfp == NULL){
    printf("SmartWorst file cannot be opened.\n");
    exit(1);
}

fprintf(valuefp,"Rec_Date,Rec_Time, Raw_Read_Error_Rate, Spin_Up_Time,
    Start_Stop_Count, Reallocated_Sector_Ct, Seek_Error_Rate,
    Power_On_Hours, Spin_Retry_Count, Power_Cycle_Count,
    Reported_Uncorrect, High_Fly_Writes, Airflow_Temperature_Cel,
    Temperature_Celsius, Hardware_ECC_Recovered, Current_Pending_Sector,
    Offline_Uncorrectable, UDMA_CRC_Error_Count, Multi_Zone_Error_Rate,
    TA_Increase_Count\n");

fprintf(worstfp,"Rec_Date, Rec_Time, Raw_Read_Error_Rate, Spin_Up_Time,
    Start_Stop_Count, Reallocated_Sector_Ct, Seek_Error_Rate,
    Power_On_Hours, Spin_Retry_Count, Power_Cycle_Count,
    Reported_Uncorrect, High_Fly_Writes, Airflow_Temperature_Cel,
    Temperature_Celsius, Hardware_ECC_Recovered, Current_Pending_Sector,
    Offline_Uncorrectable, UDMA_CRC_Error_Count, Multi_Zone_Error_Rate,
    TA_Increase_Count\n");

fclose(valuefp);
fclose(worstfp);

return;
}

/*------------------ Read_Write_Smart_Attributes -------------------*/

void Read_Write_Smart_Attributes()
{
  FILE *valuefp;
  FILE *worstfp;
  int SAValue, SAWorst, Num;
  char Rec_Date[40], Rec_Time[40];

  valuefp = fopen(SA_VALUE_FILE, "a");
  worstfp = fopen(SA_WORST_FILE, "a");

  if (valuefp == NULL){
    printf("SmartValue file cannot be reopened.
");
    exit(1);
  }

  if (worstfp == NULL){
    printf("SmartWorst file cannot be reopened.
");
  }
exit(1);

Read_SA(Rec_Date,Rec_Time,&SAValue,&SAWorst, Num);
fprintf(valuefp,"%s,%s,",Rec_Date, Rec_Time);
fprintf(worstfp,"%s,%s,",Rec_Date, Rec_Time);
for(Num=0; Num <NUM_ATTRIBUTES ; Num++){
    Read_SA(Rec_Date,Rec_Time,&SAValue,&SAWorst, Num);
    if(Num<NUM_ATTRIBUTES-1){
        fprintf(valuefp,"%d," ,SAValue);
        fprintf(worstfp,"%d," ,SAWorst);
    }
    else{
        fprintf(valuefp,"%d\n",SAValue);
        fprintf(worstfp,"%d\n",SAWorst);
    }
}
fclose(valuefp);
fclose(worstfp);
return;
/*-------------------------- Read_SA() -----------------------------*/

void Read_SA(recdate, rectime, savel, sawor, num)
char *recdate;
char *rectime;
int *savel;
int *sawor;
int num;
{

    FILE *ifp;
    int dummy;
    char recday[8];
    char ampm[4];
    char sa[40];
    char flag[10];
    char line[LINELENGTH];
    int i;

    ifp = fopen(SA_FILE,"r");
    if (ifp == NULL){
        printf("Input file cannot be opened.\n");
        exit(1);
    }

    fgets(line, LINELENGTH,ifp);

    scanf("%s %s %d %d %d", recday, ampm, savel, sawor, &num);

    if (feof(ifp)) {
        printf("File end of file\n");
        exit(1);
    }

    fclose(ifp);
}
sscanf(line, "%s %s\n", recday, reccdate);

fgets(line, LINELENGTH, ifp);
sscanf(line, "%s %s\n", rectime, ampm);
strcat(rectime, ampm);
for (i = 0; i < SKIPLINES + num + 1; i++) {
    fgets(line, LINELENGTH, ifp);
}
sscanf(line, "%d %s %s %d %d", &dummy, sa, flag, savalue, saworst);
fclose(ifp);
return;
}