



5-1999

Multi-Sensor Fusion for Induction Motor Aging Analysis and Fault Diagnosis

Ali Seyfettin Erbay
University of Tennessee - Knoxville

Follow this and additional works at: https://trace.tennessee.edu/utk_graddiss



Part of the [Nuclear Engineering Commons](#)

Recommended Citation

Erbay, Ali Seyfettin, "Multi-Sensor Fusion for Induction Motor Aging Analysis and Fault Diagnosis." PhD diss., University of Tennessee, 1999.
https://trace.tennessee.edu/utk_graddiss/2541

This Dissertation is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a dissertation written by Ali Seyfettin Erbay entitled "Multi-Sensor Fusion for Induction Motor Aging Analysis and Fault Diagnosis." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Belle R. Upadhyaya, Major Professor

We have read this dissertation and recommend its acceptance:

Robert E. Uhrig, Laurence F. Miller, Jack F. Wasserman

Accepted for the Council:

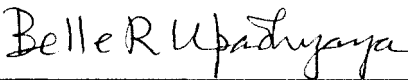
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:


I am submitting herewith a dissertation written by Ali Seyfettin Erbay entitled "Multi-Sensor Fusion for Induction Motor Aging Analysis and Fault Diagnosis." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.



Belle R. Upadhyaya, Major Professor

We have read this dissertation
And recommend its acceptance:







Accepted for the council:



Associate Vice Chancellor and
Dean of The Graduate School

**MULTI-SENSOR FUSION FOR INDUCTION MOTOR AGING
ANALYSIS AND FAULT DIAGNOSIS**

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Ali Seyfettin Erbay

May 1999

Copyright © 1999 by Ali Seyfettin Erbay

All rights reserved

DEDICATION

This dissertation is dedicated to my family.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Belle R. Upadhyaya, for his invaluable guidance and suggestions, and my other committee members, Dr. Robert E. Uhrig, Dr. Laurence F. Miller and Dr. Jack F. Wasserman, and the department head, Dr. Harold L. Dodds, for their comments and assistance. The cooperation of my teammate James P. McClanahan is gratefully acknowledged. The patience and understanding of my parents Rauf Aral and Gülay, my sister Gül and my wife Stacey Mildred, have provided great motivation during this study.

This research and development project was sponsored in part by The University of Tennessee Maintenance and Reliability Center (MRC). I wish to acknowledge the support provided by MRC in carrying out this important project. The continuing interest of the Center director, Dr. Tom Shannon, and his timely help during the project are gratefully acknowledged. I am indebted to MRC member companies for their support and suggestions: Computational Systems, Inc., Duke Energy Corporation, Eastman Chemical Company, Nissan Motor Manufacturing Corporation, PCB Piezotronics and U.S. Electrical Motors (USEM) Division of Emerson Electric Company.

The direct sponsorship of the project by Nissan Motor Manufacturing Corp. and USEM is greatly appreciated. Some of the measurement devices were provided by IMI, a division of PCB Piezotronics. I deeply appreciate the technical support provided by Dr.

Voyl Divljakovic (USEM), Marc Michaelson (Nissan) and Eric Saller (IMI). The test motors and the dynamometer were supplied by USEM. Additional support for equipment purchase was provided by Fisher-Rosemount. I appreciate the help provided by Gary Graves of the Nuclear Engineering Department, and his relentless effort during the facility development phase. I would also like to thank Dr. Serhat Şeker for his work in Multi-Resolution Analysis.

ABSTRACT

Induction motors are the most commonly used electrical drives, ranging in power from fractional horsepower (HP) to several thousand horsepowers. Several studies have been conducted to identify the cause of failure of induction motors in industrial applications. More than fifty percent of the failures are mechanical in nature, such as bearing, balance and alignment-related problems. Recent activities indicate a focus towards building intelligence into the motors, so that a continuous on-line fault diagnosis and prognosis may be performed.

The purpose of this research and development was to perform aging studies of three-phase, squirrel-cage induction motors; establish a database of mechanical, electrical and thermal measurements from load testing of the motors; develop a sensor-fusion method for on-line motor diagnosis; and use the accelerated aging models to extrapolate to the normal aging regimes. A new laboratory was established at The University of Tennessee to meet the goals of the project. The facility consists of three motor aging modules and a motor load-testing platform.

The accelerated aging and motor performance tests constitute a unique database, containing information about the trend characteristics of measured signatures as a function of motor faults. The various measurements facilitate enhanced fault diagnosis of

motors and may be effectively utilized to increase the reliability of decision making and for the development of life prediction techniques.

One of these signatures, which constitute the database, is the use of Multi-Resolution Analysis (MRA) using wavelets. In today's industry applications, vibration signatures are analyzed only up to several hundred Hertz. The use of MRA in trending different frequency bands has revealed that higher frequencies (2-4 kiloHertz) show a characteristic increase when the condition of a bearing is in question. This study effectively showed that the use of MRA in vibration signatures can identify a thermal degradation or degradation via electrical charge of the bearing, whereas other failure mechanisms, such as winding insulation failure, do not exhibit such characteristics.

A motor diagnostic system, called the Intelligent Motor Monitoring System (IMMS) was developed in this research. The IMMS integrated the various mechanical, electrical and thermal signatures, and artificial neural networks and fuzzy logic algorithms. The IMMS was then used for motor fault detection and isolation and for estimating its remaining operable lifetime. The performance of the IMMS was evaluated using the motor aging data, and showed that the stator thermal degradation, bearing thermal degradation and bearing fluting degradation could be effectively diagnosed and the prognosis of motor operation could be established.

Previous studies are primarily based on the 'cause' when calculating and estimating the remaining life of a motor and its components. This work has concentrated rather on the 'effects' in the detection and isolation of faults and the remaining lifetime of the motor and its components. Hence, when 'on-line' technology implementation is in question, measuring the effects is readily available, feasible, robust and definite, rather than trying to measure the cause (e.g. measuring the cause of motor winding insulation failure by means of ambient temperature only vs. measuring the effects of motor winding insulation failure by means of winding insulation temperature, leakage voltage and zero-component impedance).

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Statement of the Problem and the Proposed Solution.....	6
1.3 Review of Previous Work.....	11
1.3.1 Introduction.....	11
1.3.2 Magnetic Flux Analysis.....	13
1.3.3 Vibration Analysis.....	13
1.3.4 Motor Current Signature Analysis.....	14
1.3.5 Motor Power Signature Analysis.....	15
1.3.6 Temperature Analysis.....	16
1.3.7 Component Sequence Analysis.....	17
1.3.8 Data Fusion and System Integration.....	18
1.4 Contributions of the Dissertation.....	20
1.5 Organization of the Dissertation.....	23
2. A REVIEW OF INDUCTION MOTORS, FAILURES AND THEIR DETECTION .	24
2.1 Introduction.....	24
2.2 Basic Operation of an Alternating-Current Polly-Phase Squirrel-Cage Induction Motor.....	24
2.3 Project Motor Description.....	26

2.3.1	Electrical Description.....	26
2.3.2	Bearing Description.....	31
2.4	Failure Modes.....	35
2.4.1	Stator	35
2.4.1.1	Stator Frame	35
2.4.1.2	Stator Core.....	36
2.4.1.3	Stator Windings.....	36
2.4.2	Rotor.....	36
2.4.2.1	Problems Common to the Rotor.....	37
2.4.2.2	Rotor Bars	37
2.4.2.3	Rotational Problems (Vibration and Centrifugal Forces)	37
2.4.2.4	Thermal Expansion and Contraction of Windings.....	38
2.4.3	Bearings.....	38
2.4.3.1	Seals	38
2.4.3.2	Rolling Elements	39
2.4.3.3	Inner and Outer Races	39
2.4.3.4	Cage.....	39
2.4.3.5	Seating.....	40
2.4.4	Other Fault Areas	40
2.4.4.1	Electrical Faults.....	40
2.4.4.2	Loading.....	41
2.4.4.3	Bad Coupling.....	41

2.4.4.4 Soft Foot.....	41
3. ACCELERATED AGING OF INDUCTION MOTORS	42
3.1 Introduction	42
3.2 Thermal Aging of Electrical Motor Bearings	47
3.3 Thermal and Electrical Aging of Motor Stator Winding Insulation	55
3.4 Fluting of Electrical Motors	61
4. MOTOR LOAD PERFORMANCE TESTING	65
4.1 Introduction	65
4.2 Design and Installation	66
4.3 Computer Software For Aging Monitoring and Motor Performance Testing	83
4.3.1 Data Acquisition Program	83
4.3.2 Recorded Data Format	87
5. MOTOR FAILURE DETECTION	91
5.1 Introduction	91
5.2 Motor Current Analysis.....	91
5.3 Vibration Analysis.....	93
5.4 Torque Calculation.....	98
5.5 Power Calculation	99
5.5.1 Instantaneous Power.....	99
5.5.2 Torque Derived Power	100
5.6 Thermal Analysis	100
5.7 Magnetic Flux Analysis	101

5.8	Efficiency Calculation.....	102
5.9	Component Sequence Analysis.....	103
5.10	Remaining Life Estimation and Fault Classification	105
6. RESULTS OF INDUCTION MOTOR AGING ANALYSIS AND FAULT		
	DIAGNOSIS	115
6.1	Introduction	115
6.2	Failure of Electrical Motors	117
6.3	Analyses of Raw Data.....	119
6.4	Normalization of Measured Signatures.....	147
6.5	Trending of Selected Signatures	156
6.6	Fault Detection and Classification Using Fuzzy Logic.....	157
6.7	Remaining Life Estimation Using Artificial Neural Networks.....	250
6.8	Summary of Results	256
7. SUMMARY, CONCLUSIONS AND FUTURE WORK.....		
7.1	Summary	259
7.2	Conclusions	260
7.3	Recommendations for Future Work.....	264
REFERENCES.....		269
BIBLIOGRAPHY.....		275
APPENDICES.....		290
	Appendix A: Schedule of Motor Aging Testing Activities.....	291
	Appendix B: Program Listing of Data Acquisition Software.....	307

Appendix C: Program Listing of Fuzzy Logic Inference Engine.....	345
Appendix D: Program Listing of Backpropagation Artificial Neural Network.....	358
VITA.....	363

LIST OF TABLES

Table 1.1: Percentages of failure modes as given by major components	2
Table 1.2: Summary of selected failure modes for motors	3
Table 1.3: Correlation among condition monitoring technology	5
Table 2.1: Defect Frequencies for the motor ball bearings at 1,000 RPM.....	34
Table 3.1: Nameplate information of the test motors.....	44
Table 3.2: Horizontal motor dimensions for frame type 184T	45
Table 3.3: Specifications of laboratory ovens.....	50
Table 3.4: Experimental temperature distribution of the laboratory oven with the electrical motor inside	52
Table 3.5: Steady-state temperature variation inside the oven without a heat shield	53
Table 3.6: Contribution of stress factors for remaining life calculation	56
Table 3.7: Motor life versus stator winding temperature	57
Table 3.8: Timer settings and their corresponding times for one cycle to heat the electrical motor winding insulation to approximately 205 °C	60
Table 4.1: Load cell specification	67
Table 4.2: Selected torquemeter specifications.....	68
Table 4.3: Tachometer specifications.....	69
Table 4.4: Specifications of the voltage transducers used	74
Table 4.5: Specifications of current transducer.....	76
Table 4.6: Specifications of high bandwidth accelerometers.....	78
Table 5.1: Fuzzy Logic Rule Base	113

Table 6.1: Listing of the electrical motors for accelerated aging tests.....	116
Table 6.2: Sub-bands of Wavelet Multi-Resolution Analysis.....	143
Table 6.3: Matrix of Failure Detection.....	219
Table 6.4: Table of Trends	230
Table 6.5: Results of Fuzzy Logic Inferencing	248
Table 6.6: Short-End Bearing Inner Ring Residual Life Estimation Using ANN on Motor #12.....	254
Table 6.7: Results of SQL query.....	257

LIST OF FIGURES

Figure 1.1: The intelligent machinery monitoring system (IMMS).....	9
Figure 1.2: Data fusion in IMMS.....	22
Figure 2.1: Main components of an induction motor.....	25
Figure 2.2: Delta electric circuit configuration.....	27
Figure 2.3: Three phase currents for 60 Hz supply.....	29
Figure 2.4: Schematic of a bearing.....	33
Figure 3.1: Outer schematic of the 5-HP electrical motor.....	46
Figure 3.2: Schematic of the motor performance test setup.....	48
Figure 3.3: Photograph of the motor performance test setup.....	49
Figure 3.4: The motor bearing thermal aging setup.....	54
Figure 3.5: Graphical representation of Table 3.7.....	58
Figure 3.6: Photograph of the electrical motor stator winding insulation thermal aging test setup.....	62
Figure 3.7: Schematic of the electrical motor ball bearing fluting.....	63
Figure 4.1: Portable remote sensors, signal conditioning equipment and the signal patch panel.....	71
Figure 4.2: An outline of the connections of the sensors and the power supplies to the motor performance test setup.....	72
Figure 4.3: Photograph of universal patch-panel.....	73
Figure 4.4: Schematic of the high bandwidth accelerometer.....	79
Figure 4.5: Photographs of various parts of the data acquisition hardware.....	80

Figure 4.6: Data acquisition system showing the end-user interface.....	84
Figure 4.7: Graphical User Interface of the data acquisition system.	86
Figure 5.1: Calculation of remaining life based on established signature trends.....	109
Figure 5.2: Structure of the ANN to estimate the residual life of the short-end bearing inner ring.	111
Figure 6.1: Measurement of bearing temperature.	118
Figure 6.2: SKF6205 bearing disassembled.....	120
Figure 6.3: Bearing outer race of motor #11.....	121
Figure 6.4: Bearing inner race of motor #11.....	122
Figure 6.5: Bearing ball of motor #11.....	123
Figure 6.6: Bearing aging test process end 2:00 accelerometer RMS trends.....	124
Figure 6.7: Bearing aging test short end vertical accelerometer RMS trends.....	125
Figure 6.8: Bearing aging test process end bearing surface temperature trends.	126
Figure 6.9: Bearing aging test short end bearing surface temperature trends.	127
Figure 6.10: Bearing fluting test process end 2:00 accelerometer trends.	128
Figure 6.11: Bearing fluting test short end vertical accelerometer RMS trends.	129
Figure 6.12: Bearing fluting test process end bearing surface temperature trends.	130
Figure 6.13: Bearing fluting test short end bearing surface temperature trends.	131
Figure 6.14: Zero-sequence component impedance time, spectral and cepstral plots of a motor for which the insulation has been thermally aged.....	132
Figure 6.15: Motor #6 bearing aging test process end 2:00 accelerometer spectrum.....	133
Figure 6.16: Motor #7 bearing aging test process end 2:00 accelerometer spectrum.....	134

Figure 6.17: Motor #8 bearing aging test process end 2:00 accelerometer spectrum.	135
Figure 6.18: Motor #11 bearing fluting test process end 2:00 accelerometer spectrum.	136
Figure 6.19: Motor #12 bearing fluting test process end 2:00 accelerometer spectrum.	137
Figure 6.20: Motor #13 bearing fluting test process end 2:00 accelerometer spectrum.	138
Figure 6.21: Input – output power relation of the system.	139
Figure 6.22: Initial (light-color) and final (dark-color) input – output power coherence of motor #11 at 100% load level.	140
Figure 6.23: Initial (light-color) and final (dark-color) motor power - vibration coherence of motor #11 at 100% load level.	141
Figure 6.24: Initial (green) and final (blue) motor PE 2:00 – cover vibration coherence of motor #11 at 100% load level.	142
Figure 6.25: Comparison of band RMS values to total RMS values for vibration measurements.	144
Figure 6.26: Normalization of PE 2:00 RMS vibration.	148
Figure 6.27: Normalization of bearing surface PE temperature difference.	149
Figure 6.28: Normalization of bearing surface SE temperature difference.	150
Figure 6.29: Normalization of spectral amplitudes at line frequency of PE 2:00 vibration.	151
Figure 6.30: Normalization of spectral amplitudes at line frequency of motor power. ..	152
Figure 6.31: Normalization of spectral amplitudes at line frequency of motor current. .	153
Figure 6.32: Normalization of spectral amplitudes at line frequency of motor zero sequence component impedance.	154

Figure 6.33: Normalization of spectral amplitudes at line frequency of motor radial magnetic flux.....	155
Figure 6.34: Load power skewness at 100% load.....	158
Figure 6.35: Motor power skewness at 100% load.....	159
Figure 6.36: Motor current RPM frequency amplitude trend at 100% load.....	160
Figure 6.37: Motor zero component impedance RPM frequency amplitude at 100% load.....	161
Figure 6.38: Axial magnetic flux RPM frequency amplitude at 100% load.....	162
Figure 6.39: Radial magnetic flux RPM frequency amplitude at 100% load.....	163
Figure 6.40: Motor power derivative RPM frequency amplitude at 100% load.....	164
Figure 6.41: Motor power RPM frequency amplitude at 100% load.....	165
Figure 6.42: Motor work RPM frequency amplitude at 100% load.....	166
Figure 6.43: Motor PE vibration RPM frequency amplitude at 100% load.....	167
Figure 6.44: Motor zero component impedance slip frequency amplitude at 100% load.....	168
Figure 6.45: Axial magnetic flux slip frequency amplitude at 100% load.....	169
Figure 6.46: Radial magnetic flux slip frequency amplitude at 100% load.....	170
Figure 6.47: Motor zero component impedance slip * number of poles frequency amplitude at 100% load.....	171
Figure 6.48: Axial magnetic flux slip * number of poles frequency amplitude at 100% load.....	172
Figure 6.49: Axial magnetic flux slip * number of poles frequency amplitude at 75%	

load.....	173
Figure 6.50: Motor power slip * number of poles frequency amplitude at 100% load. .	174
Figure 6.51: Motor PE vibration slip * number of poles frequency amplitude at 100% load.....	175
Figure 6.52: Motor zero component impedance slip * number of poles / 2 frequency amplitude at 100% load.....	176
Figure 6.53: Axial magnetic flux slip * number of poles / 2 frequency amplitude at 100% load.....	177
Figure 6.54: Motor zero component impedance (1 + 2 * slip) * f _e frequency amplitude at 100% load.....	178
Figure 6.55: Radial magnetic flux (1 + 2 * slip) * f _e frequency amplitude at 100% load.....	179
Figure 6.56: Motor zero component impedance (1 - 2 * slip) * f _e frequency amplitude at 100% load.....	180
Figure 6.57: Radial magnetic flux (1 - 2 * slip) * f _e frequency amplitude at 100% load.....	181
Figure 6.58: Load power line frequency amplitude at 100% load.....	182
Figure 6.59: Motor zero component impedance line frequency amplitude at 100% load.....	183
Figure 6.60: Radial magnetic flux line frequency amplitude at 100% load.....	184
Figure 6.61: Motor PE vibration f _e + SE IR frequency amplitude at 100% load.....	185
Figure 6.62: Motor cover vibration f _e + SE IR frequency amplitude at 100% load.	186

Figure 6.63: Motor power f_e + SE IR frequency amplitude at 100% load.....	187
Figure 6.64: Motor current f_e + SE IR frequency amplitude at 100% load.	188
Figure 6.65: Motor power f_e + PE IR frequency amplitude at 100% load.....	189
Figure 6.66: Motor current f_e + PE IR frequency amplitude at 100% load.	190
Figure 6.67: Motor PE vibration f_e + PE IR frequency amplitude at 100% load.....	191
Figure 6.68: Motor cover vibration f_e + PE IR frequency amplitude at 100% load.	192
Figure 6.69: Motor power f_e + SE OR frequency amplitude at 100% load.	193
Figure 6.70: Motor current f_e + SE OR frequency amplitude at 100% load.....	194
Figure 6.71: Motor PE vibration f_e + SE OR frequency amplitude at 100% load.	195
Figure 6.72: Motor cover vibration f_e + SE OR frequency amplitude at 100% load.....	196
Figure 6.73: Motor power f_e + PE OR frequency amplitude at 100% load.	197
Figure 6.74: Motor current f_e + PE OR frequency amplitude at 100% load.....	198
Figure 6.75: Motor PE vibration f_e + PE OR frequency amplitude at 100% load.	199
Figure 6.76: Motor cover vibration f_e + PE OR frequency amplitude at 100% load.....	200
Figure 6.77: Motor power f_e + SE B frequency amplitude at 100% load.	201
Figure 6.78: Motor current f_e + SE B frequency amplitude at 100% load.....	202
Figure 6.79: Motor PE vibration f_e + SE B frequency amplitude at 100% load.	203
Figure 6.80: Motor cover vibration f_e + SE B frequency amplitude at 100% load.....	204
Figure 6.81: Motor current f_e + PE B frequency amplitude at 100% load.....	205
Figure 6.82: Motor PE vibration f_e + PE B frequency amplitude at 100% load.	206
Figure 6.83: Motor power f_e + RPM frequency amplitude plot at 100% load.....	207
Figure 6.84: Motor current f_e + RPM frequency amplitude plot at 100% load.	208

Figure 6.85: Axial magnetic flux $f_e + 2 * \text{RPM}$ frequency amplitude at 100% load.....	209
Figure 6.86: Radial magnetic flux $f_e - \text{RPM}$ frequency amplitude at 100% load.	210
Figure 6.87: Motor power – PE 2:00 vibration coherence slip frequency amplitude at 100% load.....	211
Figure 6.88: Motor power – PE 2:00 vibration coherence slip * number of poles frequency amplitude at 100% load.....	212
Figure 6.89: Motor power – PE 2:00 vibration coherence slip * number of poles / 2 frequency amplitude at 100% load.....	213
Figure 6.90: Motor power – PE 2:00 vibration coherence line frequency amplitude at 100% load.....	214
Figure 6.91: PE 2:00 RMS vibration trends for thermal aging and electrical discharge aging.....	215
Figure 6.92: Bearing surface PE temperature difference trends for thermal aging and electrical discharge aging.....	216
Figure 6.93: Bearing surface SE temperature difference trends for thermal aging and electrical discharge aging.....	217
Figure 6.94: MRA RMS ratios of PE 2:00 motor vibration for thermal aging and fluting.....	218
Figure 6.95: Fuzzy membership functions of temperature.	234
Figure 6.96: Fuzzy membership functions of vibration.	235
Figure 6.97: Fuzzy membership functions of coherence.	236
Figure 6.98: Fuzzy membership functions of magnetic flux.	237

Figure 6.99: Fuzzy membership function for impedance.	238
Figure 6.100: Fuzzy membership functions of motor power.	239
Figure 6.101: Fuzzy membership functions of motor power.	240
Figure 6.102: Fuzzy membership functions of motor current.....	241
Figure 6.103: Fuzzy membership functions of motor current.....	242
Figure 6.104: Fuzzy membership functions of motor current.....	243
Figure 6.105: Fuzzy membership functions of motor vibration.....	244
Figure 6.106: Fuzzy membership functions of motor vibration.....	245
Figure 6.107: Fuzzy membership functions of motor vibration.....	246
Figure 6.108: Fuzzy output membership functions of fault classification.	247
Figure 6.109: Training data set for the ANN using motor current.....	251
Figure 6.110: Training data set for the ANN using motor power.	252
Figure 6.111: Training data set for the ANN using motor vibration.....	253
Figure 6.112: Graphical representation of Table 6.6.	255
Figure 7.1: Time – frequency analysis of motor start-up data using motor current.....	265
Figure 7.2: Time – frequency analysis of motor start-up data using motor power.	266
Figure 7.3: Time – frequency analysis of motor start-up data using motor vibration.....	267

LIST OF ACRONYMS

AC	:	Alternating Current
ANN	:	Artificial Neural Network
B	:	Ball
BS	:	Bearing Surface
dB	:	deciBel
DC	:	Direct Current
EPRI	:	Electrical Power Research Institute
f_e	:	Line Frequency
FFT	:	Fast Fourier Transform
g	:	Acceleration Due to Gravity
HP	:	Horsepower
Hz	:	Hertz
I	:	Current
IMMS	:	Intelligent Machinery Monitoring System
IR	:	Inner Ring
MB	:	MegaBytes
MRA	:	Multi-Resolution Analysis
MRC	:	Maintenance and Reliability Center
OR	:	Outer Ring
P	:	Power

PC	:	Personal Computer
PE	:	Process End
RMS	:	Root Mean Square
s	:	Slip (%)
SCXI	:	Signal Conditioning eXtension for Instrumentation
SE	:	Short End
USEM:		U.S. Electrical Motors
V	:	Volts
W	:	Watts

Chapter 1

INTRODUCTION

1.1 Motivation

A nuclear power plant utilizes hundreds of electric motors of all sizes and types. Failure of these motors, especially those designed for safety operations, could impair plant safety. Although motors are known to be built rugged and have a record of reliable operation, they do fail and may disable safety systems in nuclear plants [1 - 3]. Induction motors are the most commonly used electrical motors. Depending on the applications, induction motors have a wide range of power outputs, from fractional horsepower (HP) to several thousand horsepowers. An induction motor is mechanically simple, rugged, can be adapted to widely differing operating conditions and is simple to control. Several studies were made to identify the cause of failure of induction motors in industrial applications. Table 1.1 provides a general breakdown of motor failures. Table 1.2 is a more detailed list of failure modes of motors.

Such studies have been conducted by EPRI [4], IEEE Industry Applications Society [5], and others [6]. The principal damage mechanisms that lead to the failure of these motors are overheating, insulation breakdown, mechanical failures and motor circuit faults. In some cases the root cause of motor anomalies were traced to harmonics

Table 1.1: Percentages of failure modes as given by major components *

<u>Failure Mode</u>	<u>Percentage</u>
<u>Bearing Related</u>	<u>Total - 41%</u>
Sleeve Bearings	16 %
Anti-Friction Bearings	8%
Seals	6%
Thrust Bearings	3%
Oil Leakage	3%
Other	3%
<u>Stator Related</u>	<u>Total - 37%</u>
Ground Insulation	23%
Turn Insulation	4%
Bracing	3%
Wedges	1%
Frame	1%
Core	1%
Other	1%
<u>Rotor Related</u>	<u>Total - 10%</u>
Cage	5%
Shaft	2%
Core	1%
Other	2%
All Others	Total - 12%

* This information was based on an 872 failed motor survey conducted by EPRI.

Table 1.2: Summary of selected failure modes for motors

Failure Mode	Percent of Total Failure
Stator Frame	0.8
Insulation - Ground	18.5
Insulation - Turn	3.7
Coil Connection	0.6
Loose Blocking	1.8
Line Cable	0.8
Stator Slot Wedges	1.3
Shaft	1.5
Bar Failure	3.5
Loose Iron	1.0
Balance Weights	0.6
Ball Bearing	4.9
Roller Bearing	2.3
Sleeve Bearing	9.7
Oil Leakage	4.1
Bearing Seal	2.3
Oil System	1.4
Thrust Bearing (V)	4.7
Thrust Bearing (H)	0.2
Accessories	0.5
All Other	35.9

of the line frequency in the power supply [7]. Table 1.1 indicates that more than fifty percent of the failures are stator winding insulation and bearing assembly failures. This is true for both nuclear and non-nuclear applications. It is, therefore, desirable to develop methods of monitoring the condition of these vulnerable motor components.

The interpretation of multiple technologies used for condition monitoring of motors is gaining popularity. Table 1.3 shows correlations among vibration analysis, lubricant wear particle analysis, thermography, motor current analysis and others [4]. An approach for an integrated on-line motor protection system, that uses motor current signature analysis for predicting mechanical and insulation failures, was proposed by Farag, Bartheld and Habetler [8]. Recent activities indicate a focus towards building intelligence into the motors, so that a continuous on-line failure prediction may be performed with the capability of predicting remaining useful life of machinery.

The purpose of the research and development effort at The University of Tennessee was to perform aging studies of three-phase squirrel-cage induction motors, establish a database from motor performance tests for each aging case, and to develop a multi-sensor fusion method for machinery diagnosis. A new laboratory was established to meet the goals of the project. The facility consists of motor aging modules and a motor testing platform. Electrical, mechanical and thermal measurements were performed in all cases. The product of the project will be applicable in enhancing the

Table 1.3: Correlation among condition monitoring technology [4]

	1	2	3	4	5	6	7	8	9
Vibration Analysis (1)		✓	✓	✓			✓	✓	
Lubricant Wear Particle Analysis (2)	✓								
Temperature/Thermography (3)	✓	✓							
Motor Current Signature Analysis (4)	✓	✓	✓						
Surge Comparison Testing (5)	✓			✓					
High-Potential testing (Hi-Pot) (6)					✓				
Motor Circuit Evaluation (7)	✓	✓	✓	✓	✓	✓			
Breakaway and Coastdown (8)	✓	✓	✓	✓			✓		
Insulation Resistance (9)			✓	✓	✓	✓	✓		

design of industrial motors and in developing on-line monitoring systems. The project was being conducted in close cooperation with industry collaborators.

In order to develop a multi-sensor fusion method for the machinery diagnosis, mechanical, thermal and electrical “features” are extracted. The features that indicate aging will be extracted and the functional form of the trend behavior will be determined. This analysis is used to develop an algorithm for extrapolating the accelerated aging feature shape function to normal operating conditions.

1.2 Statement of the Problem and the Proposed Solution

The overall purpose of the project is to establish a database of induction motor aging mechanisms and to develop a multi-sensor fusion method for analyzing the machinery data from accelerated aging, performing fault diagnosis and the estimation of remaining operability lifetime of induction motors.

In order to establish on-line signatures and features of motor failure, electrical motors have to be observed before failure. It is an impractical task to observe electrical motors for various measurements during their operational lifetime. Therefore it is imperative to accelerate the aging process of electrical motors under observation. IEEE 117 and UL 1446 describe the recommended accelerated aging methods, based on the

Arrhenius model [9, 10, 11]. Three forms of aging of motors have been considered. The following aging experiments were performed on 5-HP three-phase squirrel cage motors.

- Stator insulation thermal aging by continuous automated on-off switching by single-phase power supply. The insulation temperature was maintained at ~ 205 °C.
- Bearing thermal aging at ~ 140 °C, with induced corrosion by water soaking.
- Bearing fluting aging by shaft current technique combined with thermal aging.

The correspondence of these accelerated aging tests to normal operation conditions will be explained in detail in the following chapters. The importance of this study is to detect and identify the aging, in other words the failure mechanism, through various on-line measurements. If we consider that electrical motors consume a large amount of electrical energy produced, monitoring the health of electrical motors will decrease unexpected downtime and increase motor life by careful fault identification and timely repairs.

A total of twelve 5-HP motors were used for the aging studies, each undergoing up to 11 cycles of aging experiments mentioned above. At the end of each cycle, a complete load test was performed using an electrical dynamometer. The tests ranged from no-load to 115% full load. A PC-based data acquisition system was designed and developed to acquire measurements from thirty sensors: vibration, electrical, temperature,

speed and load variables. A database management program was developed to systematically record the data for future use. Simple calculations (such as mean value, RMS value, motor power, power factor and motor efficiency) can be performed during data acquisition. The detailed analysis, including signature trending, was performed off-line.

The database from these aging experiments are unique in nature and contain valuable information about the sensitivities of signatures to motor faults, their trend behavior and for developing on-line life prediction models.

This study aims to implement the intelligent machinery monitoring system (IMMS) as shown in Figure 1.1. In order to implement this system the following tasks were performed.

- Establishment of trends of statistical features. These include the RMS and mean values of measured variables as well as derived variables such as the phase impedance and spectral amplitudes at specific frequencies.
- A sensitivity matrix was established from which proven and new methods were used to extract features from raw and derived measurements. These techniques are explained in detail in the following chapters.

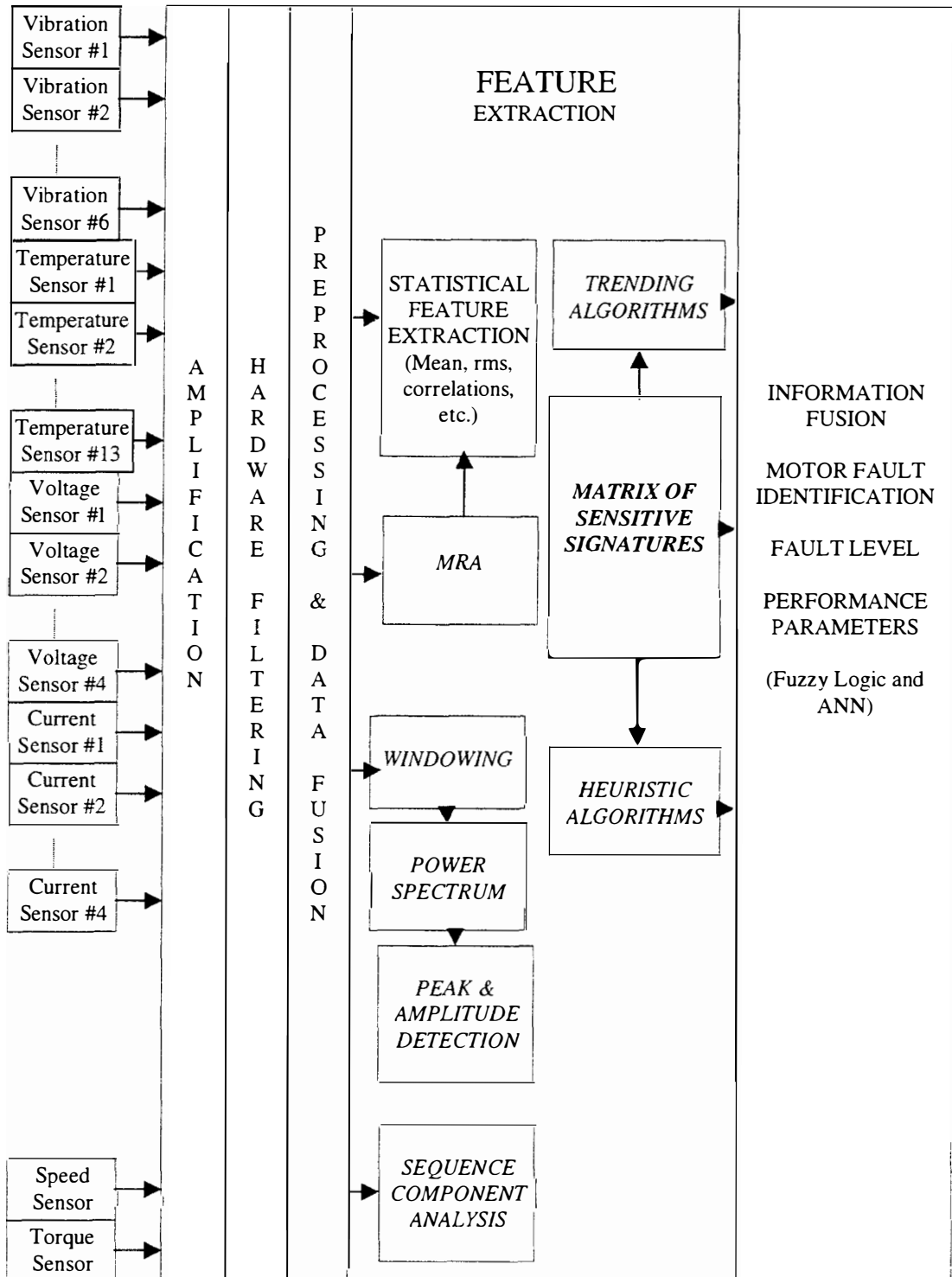


Figure 1.1: The intelligent machinery monitoring system (IMMS).

- Least – squares fits for the selected features. Some of the measured / calculated points are showing deviations from normal. Some of these deviations are caused by specific conditions, which are abnormal for accelerated aging tests. Motor #13, aging cycle #5 is such an example in which a loose terminal caused a phase imbalance resulting in higher temperatures and a current imbalance. This measurement has to be excluded from the least-squares fit calculations.
- An artificial neural network estimates the remaining life and percent failure of specific components. The network was trained using accelerated aging data. By this approach, a feature-level data fusion was accomplished.
- The sensitivity matrix was implemented by using fuzzy logic reasoning. If-and-then statements using fuzzy variables fuse various features and measurements. This allows the handling of imprecise data efficiently. Membership functions were established with the experience gained through accelerated aging tests: By determining the alarm levels of each selected measurement or feature, membership functions of the fuzzy measurements were established. Deviations from established design parameters were the basis for construction of these membership functions.

1.3 Review of Previous Work

1.3.1 Introduction

Present industry-wide maintenance tests on motors are primarily limited to the following off-line tests:

1. Dielectric Insulation Resistance Test (Megger),
2. Surge Test,
3. AC Dissipation Factor / Capacitance Test [12],
4. AC Partial Discharge Test [13],
5. AC / DC Leakage Current Test,
6. DC Winding Resistance Test (Wheatstone Bridge),
7. Winding End Turn Movement Test,
8. Bearing Grease Analysis.

The following are the primary on-line test methods in detecting electric motor faults:

1. Vibration Analysis,
2. Motor Current Signature Analysis,
3. Component Sequence Analysis,
4. Motor Power Signature Analysis,
5. Temperature Analysis,

6. Magnetic Flux Analysis.

The purpose of this study is to improve the on-line fault detection methods for detecting incipient faults. Most of the studies in the literature are concentrated on detecting a fault after it has developed; however, as it is in the case of motor winding insulation failure, some faults are catastrophic after they develop.

MotorStatus, developed by Status Technologies, collects and examines detailed information on operational temperature, flux, and vibration parameters to provide a comprehensive motor condition assessment [14]. It uses vibration measurements to assess the condition of the bearings and flux measurement to detect a stator short. However, provided that an electrical motor can be equipped with other built-in sensors, such as thermocouples, a fault can be detected by other measurements also, increasing the reliability and robustness of the fault detection. The unit is still under continuous improvement for application to incipient fault detection. The tested unit was not very accurate in determining the load-level of the electrical motor. In addition, an operator has to download the collected data and trend it manually in order to detect the incipient fault. The unit itself has the ability in its final version, to alarm an already developed fault.

1.3.2 Magnetic Flux Analysis

Presently, there is no commercial instrument or technique capable of detecting faults associated with the electrical insulation of electric motor stators while the motor is in operation (on-line). Technologies such as motor current analysis and negative sequence impedance have not, to date, been proven as effective methods of on-line stator monitoring. Although the MotorStatus unit uses a magnetic flux coil to detect a turn-to-turn short in the motor winding, it is not fine-tuned to detect a failure at the time the unit was used in this study. Detection of a winding insulation fault requires trending of the following frequencies in the magnetic flux spectrum [15, 16]:

- Line Frequency – 2 * Running Speed
- Line Frequency – 1 * Running Speed
- Line Frequency
- Line Frequency + 1 * Running Speed
- Line Frequency + 2 * Running Speed
- Line Frequency + 3 * Running Speed

1.3.3 Vibration Analysis

Many other studies exist, in which vibration analysis has been used successfully in detecting faults in electrical motors. This technique is explained in detail in later

chapters, in which monitoring of specific frequencies gives a condition assessment of the motor. Vibration analysis can detect the following failures [17 - 20]:

- Imbalance can be detected using spectral plots and polar plots.
- Resonance can be detected at 1x the running speed,
- Misalignment produces excitations at 1x, 2x the running speed,
- Mechanical faults like bearings, axial float, soft foot,
- Electrical faults due to non-symmetry in the air gap,
- Broken, cracked rotor bars will show up at 1x rotational speed, 2x line frequency, slip frequency x number of poles, rotational speed \pm slip frequency x number of poles.

1.3.4 Motor Current Signature Analysis

Motor Current Signature Analysis is effective in the range of 25% - 130% of full load conditions. It can detect the following failures [19, 21 - 26]:

- Broken rotor bar will show harmonics at $(1 \pm 2s)f_e$, where s is slip and f_e is line frequency of motor,
- Cracked end rings,
- High resistance squirrel cage joints,
- Casting porosity in die cast rotors,

- Static & dynamic air gap irregularities,
- Unbalanced magnetic pull.

This method should be used together with vibration analysis in order to reduce false alarms [27].

1.3.5 Motor Power Signature Analysis

Motor Power Signature Analysis can detect the following faults in a motor [28]:

- Broken rotor bars using frequency analysis,
- Shorted stator turns,
- Shorted laminations,
- High resistance connections,
- Motor bearing damage using frequency analysis,
- Rotor eccentricity using frequency analysis and high-pass filtering,
- Misalignment and overload using trending,
- Shorted circuits,
- Inter-turn short circuits,
- Ground faults,
- Phase failures.

This technique is more sensitive than processing the current signal alone. The characteristic spectral component of the power appears directly at the frequency of disturbance, independently of the synchronous speed of the motor. This is important in automated diagnostic systems, in which irrelevant frequency components, i.e. those at multiples of the supply frequency, are screened out [29]. This technique is not sensitive in detecting the above faults when load and supply voltage change.

1.3.6 Temperature Analysis

One of the famous studies in this area is on using the Arrhenius and Eyring model to estimate the residual life of an electric motor [30, 31]. The relative life-ratio for motor winding insulation is formulated as:

$$\text{Life-ratio} = \exp\{(\phi/k)(1/T_d - 1/T_m)\} \quad (1.1)$$

where

$T_d = 40^\circ\text{C} + T_{rise} + 273$ = motor design temperature, K

T_m = motor actual temperature, K

T_{rise} = motor rated temperature rise, °C

ϕ = insulation activation energy, eV

k = Boltzman constant, 0.00008617 eV

The average grease life of a motor bearing for “SRI Grease NLGI 2” lubricant and an “SKF 6203” bearing is calculated as [32]:

$$Life (hrs) = 6.12 - 1.14(n/N_{max}) - [0.018 - 0.006(n/N_{max})]T \quad (1.2)$$

where

n = speed of operation, rpm

N_{max} = rated speed capacity, rpm

T = Temperature, (70°C - 130°C)

1.3.7 Component Sequence Analysis

Component Sequence Analysis [33] is basically a transformation method in which instantaneous potentials, currents, impedance and power are resolved into their symmetrical components [34 - 36]. Although this method is well documented and used in theoretical computation and simulations, there are no commercial applications utilizing this approach. This method has the ability to detect

- Phase-to-phase faults,
- Turn-to-turn faults,
- Phase-to-ground faults

The method is independent of the slip frequency [8].

1.3.8 Data Fusion and System Integration

The area of fusing data in machinery diagnostics is new, although the technique has been successfully developed in various military and robotic applications [37]. The following are the main data-fusion levels:

- “Signal-Level Fusion” consists of single- or multi-dimensional signals. This methodology reduces the expected variance.
- “Feature-Level Fusion” consists of features extracted from signals. It reduces processing, increases feature measurement accuracy and has the value of additional features.
- “Symbol-Level Fusion” is a representation of the decision and is the highest information level. Data fusion in this level increases truth or probability values.

Signal Level Data Fusion is being used in normalization of electrical motor temperatures as

$$T = (T_m - T_{amb}) / \% Load \quad (1.3)$$

where

T_m = measured motor temperature

T_{amb} = ambient motor temperature

This level of data fusion makes the motor temperature measurement independent of ambient temperature, thus differentiating between the temperature rise as a result of motor fault or due to given running condition. Another signal level data fusion is when the motor temperatures are not available to the user and are derived from current measurements.

Feature-level fusion has been proposed using a Hilbert space formulation. Vibration and magnetic flux measurements are collected, after which their spectra are fused to create a Nyquist plot [32].

Symbol-Level Data Fusion has been investigated using artificial neural networks. Features, which are extracted from measurements are fused together to detect and classify failure in electrical motors. However, no work has been done so far in using fuzzy-logic reasoning in detection and classification of motor faults. This approach incorporates the technique of classical rule-based programming together with imprecise data. This approach is especially useful, as it will take into consideration, that even for a given electrical motor, the manufacturer specifications and behavior of the motor may change.

1.4 Contributions of the Dissertation

The original contributions of this dissertation are summarized in the following list.

- Development of a unique electrical motor accelerated aging experiment. This database incorporates most of the sensor measurements used today in the industry for on-line fault detection. The uniqueness of this database is that the diversity of synchronized measurements enables the user to apply many computational methods in detecting certain faults. The raw database is recorded in binary format, whereas another higher-level database is developed in Microsoft Access for further feature and database query.
- Most of the studies in the literature are concentrated on detecting a fault. However, it is very crucial in plant operations to detect an incipient fault. This unique database is an extensive collection of measurements, which can display trends of faults before they develop. Functional forms of degradation models will be developed, based on the signatures and derived features from the database.
- Use of manufacturer-installed thermocouples into an electrical motor is a cost-effective method in detecting bearing degradation failures, which may cause increased frictional forces, and hence increase local temperatures at faulty

locations. Trending of such temperatures involves simple algorithms and increases the effectiveness of a fault being detected. Also, several bearing degradation modes can be classified by using thermal analysis.

- Reliance upon data from a single source or sensor can result in potentially erroneous conclusions. Sensor fusion is a method for dealing with this problem. In order to implement a successful sensor-fusion, a “Sensitivity Matrix” is being formed. This matrix incorporates various features and methods to detect specified faults.

- Development of data fusion at three different levels:
 1. Signal-Level Fusion (Temperature normalization with respect to ambient temperature and power level)
 2. Feature-Level Fusion (Transfer functions, cross-power spectra, coherence, Ratio of RMS Temperature to RMS Vibration Level, etc.)
 3. Symbol-Level Fusion (Artificial Neural Networks, Fuzzy-Logic Decision Making). The probability of error could be reduced by using this method.

Figure 1.2 shows the overall strategy of data fusion.

- So far, no commercial product or technique has been able to detect an incipient winding insulation failure. Component-sequence analysis has yielded a unique signature from which winding insulation failures can be detected and trended

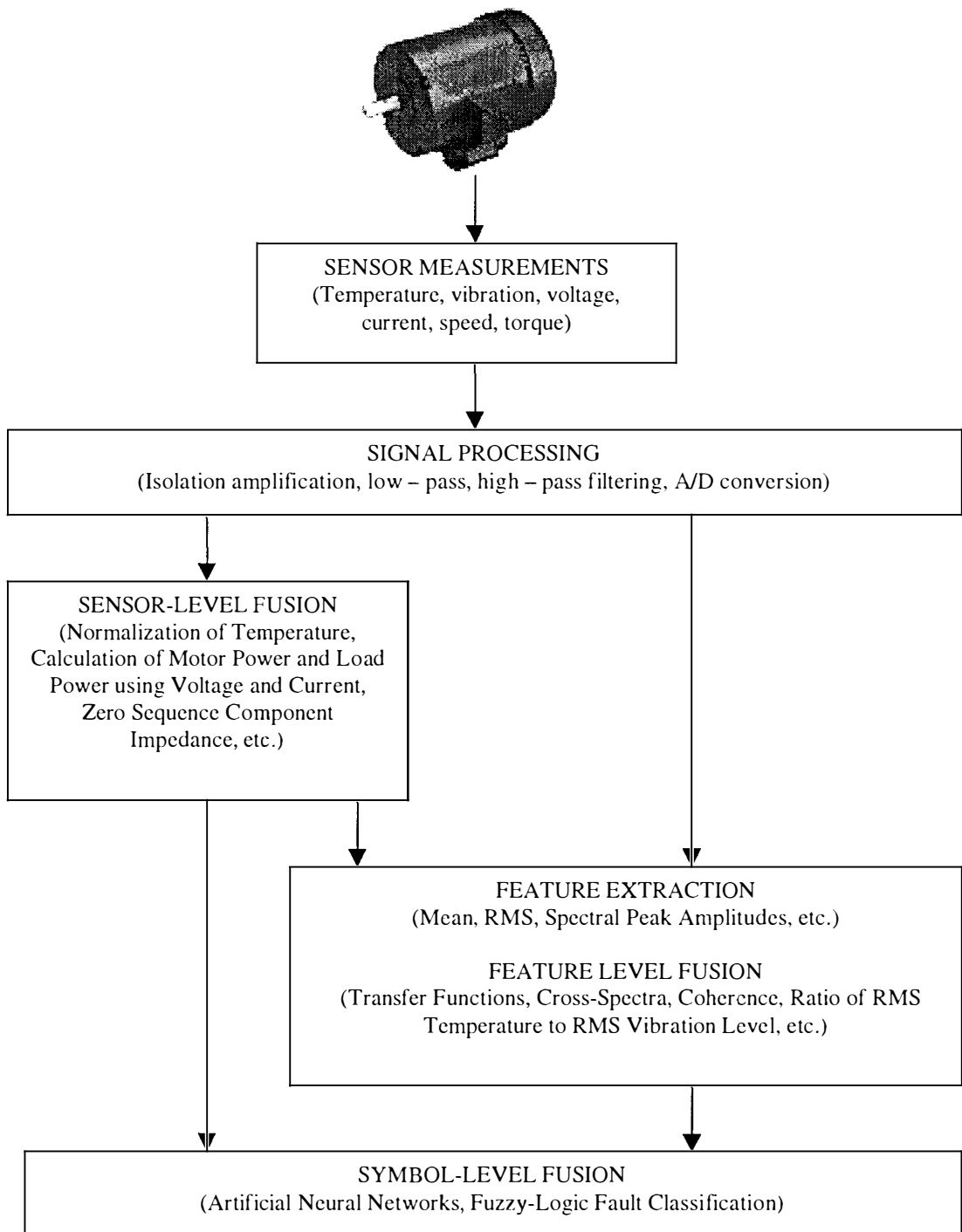


Figure 1.2: Data fusion in IMMS.

before they occur and shutdown the motor from operation.

- Multi-sensor fusion using degradation trend features from the “Sensitivity Matrix” and fuzzy-logic to identify the type and extent of motor fault. This method yielded a robust algorithm in detecting specific motor faults.

1.5 Organization of the Dissertation

Chapter 2 of this dissertation presents basic descriptions of induction motors, the motors used in these studies, their failure modes and failure detection methods as introduced in this chapter. Chapter 3 explains how the accelerated aging tests were performed. The study focuses on three main failure modes of the electrical motor. Each of them is explained in detail. Chapter 4 explains how the motors were tested on a performance platform, where various electrical, mechanical and thermal measurements were taken. A detailed ‘activity log’ of tests explained in Chapters 3 and 4 is also established in Appendix A. Signatures used for motor fault monitoring are described in Chapter 5. Finally, Chapter 6 presents results of the accelerated aging studies, showing signatures and the IMMS, which were used to detect and classify electrical motor failures, as well as the remaining life estimation of particular components. Finally, a summary and concluding remarks are given in Chapter 7.

Chapter 2

A REVIEW OF INDUCTION MOTORS, FAILURES AND THEIR DETECTION

2.1 Introduction

The purpose of this chapter is to describe briefly the induction motor, its modes of failure, and the methods to detect these failures [38].

2.2 Basic Operation of an Alternating-Current Poly-Phase Squirrel-Cage Induction Motor

As shown in Figure 2.1, there are two major components in an induction motor, the rotor and the stator. The stator is the outer most component. "The stator employs wound coils to generate the rotating magnetic field. Stacking a series of soft iron stampings forms the stator coil. This construction offers a low reluctance path to radial and tangential magnetic fluxes while presenting a high electrical resistance to axial currents: this reduces eddy current loss in the core. A series of axial slots run the length of the stator stack along its inside diameter. These slots serve as coil forms. Copper wire coils are pre-wound and fitted into the stator slots" [39]. The electrical current input into the motor system energizes the stator. The energized stator produces an electromagnetic field that is used by the rotor to induce rotation.

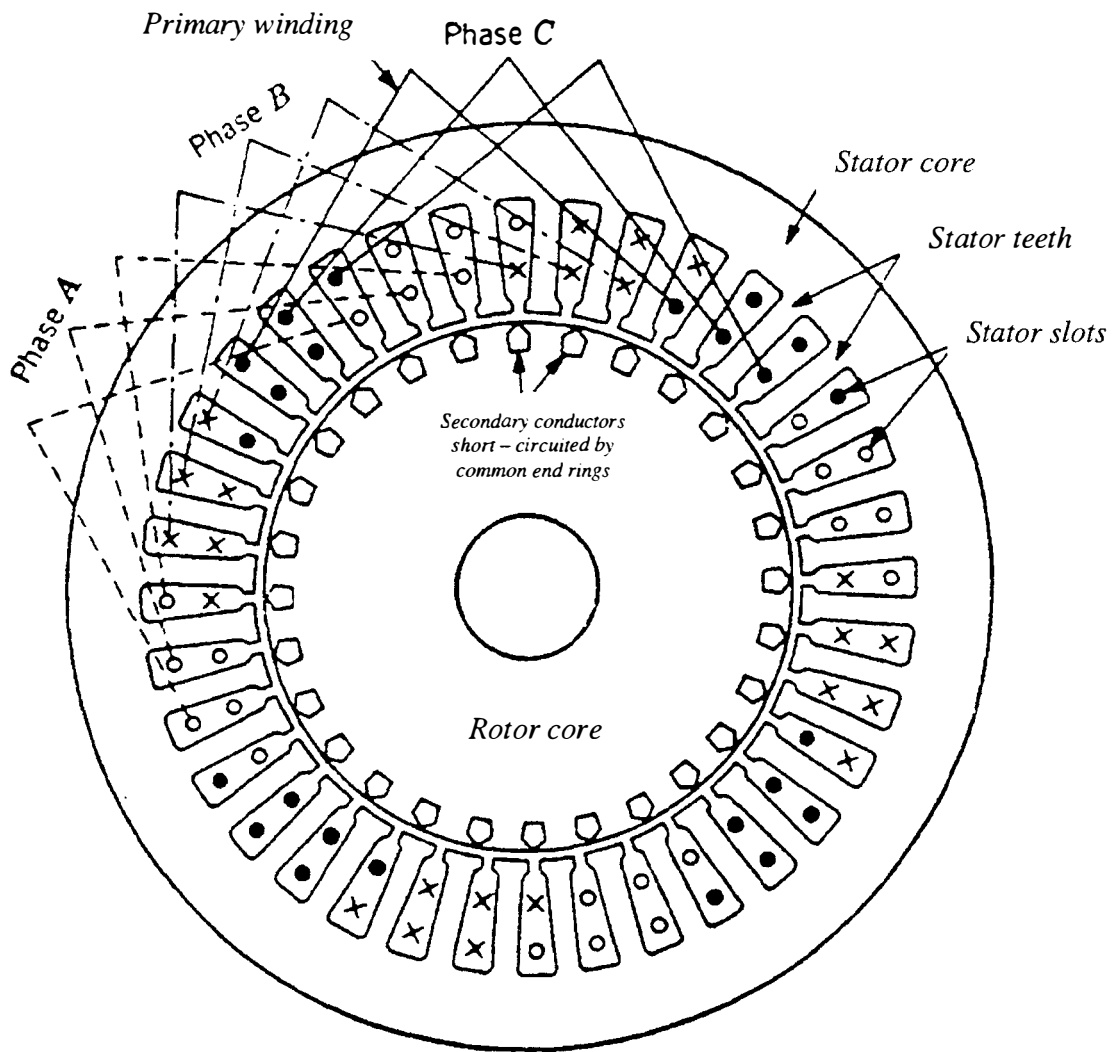


Figure 2.1: Main components of an induction motor.

"The squirrel-cage rotor consists of a series of stamped soft iron discs stacked along the shaft. This yields a structure, which is magnetically compliant in the radial and tangential directions and electrically resistant to axial eddy currents... Each disc is punched with a series of equally spaced holes near the perimeter. The discs are aligned when the rotor is stacked so that the holes form axial channels running the length of the rotor core (or slightly skewed) to the shaft. Electrically conductive rotor bars run through these channels. At each end, all of the rotor bars are joined to a single electrically conductive end ring" [39].

2.3 Project Motor Description

The induction motors used in this project are 5-HP, three-phase, 220 Volt, 15 Amperes (maximum start-up), 12 Amperes continuous, 1,800 RPM electrically operated squirrel-cage motor. The motors are supplied by U.S. Electrical Motors Division of Emerson Electric Company.

2.3.1 Electrical Description

The three-phase induction motor may be wired in two different forms, star and delta connections. The motors used in the project are in the delta configuration. The delta configuration is shown in Figure 2.2 with respect to the physical wire numbering.

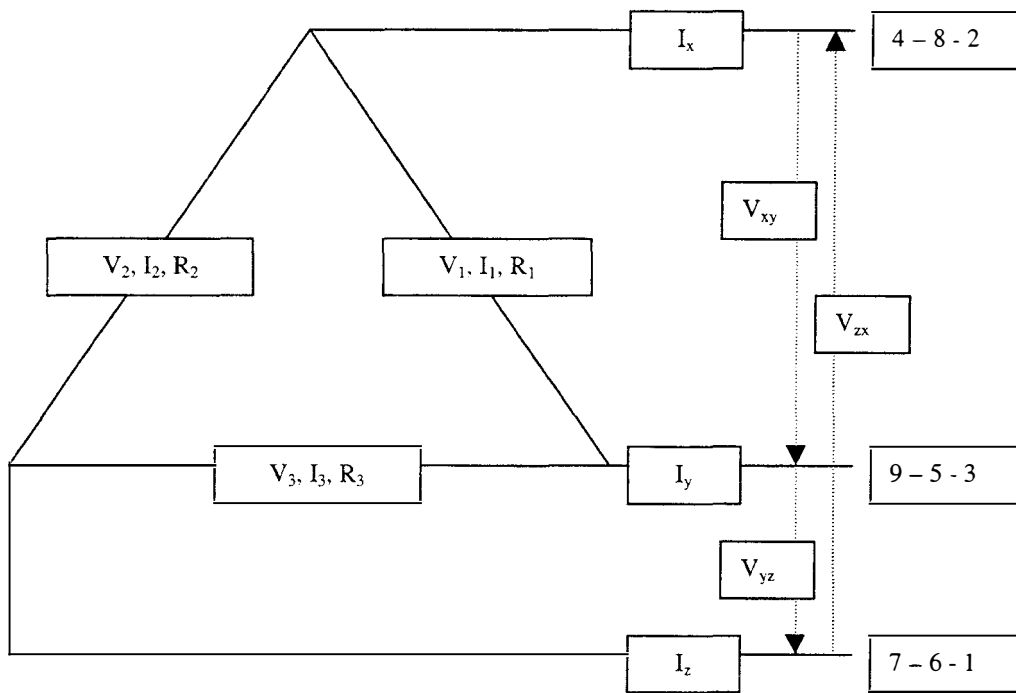


Figure 2.2: Delta electric circuit configuration.

There are three different voltages and currents measured within the motor. In this configuration the following are valid for voltage and current relations:

$$I_p = \frac{1}{\sqrt{3}} I_l \quad (2.1)$$

$$V_p = V_l \quad (2.2)$$

where

I_p = RMS Phase current

I_l = RMS Line current

V_p = RMS Phase voltage

V_l = RMS Line voltage

Since there are three different phases, three separate electrical waveforms are used by the motor. Each electrical waveform is separated from the other by a phase angle of 60 degrees. If the currents or voltages are plotted against time, each waveform is identical except for the phase difference. We also know that electricity supplied by power companies has a frequency of 60 Hz. Therefore we can graph the three voltages versus time for 0.10 seconds as shown in Figure 2.3. The apparent power of this balanced system is given by

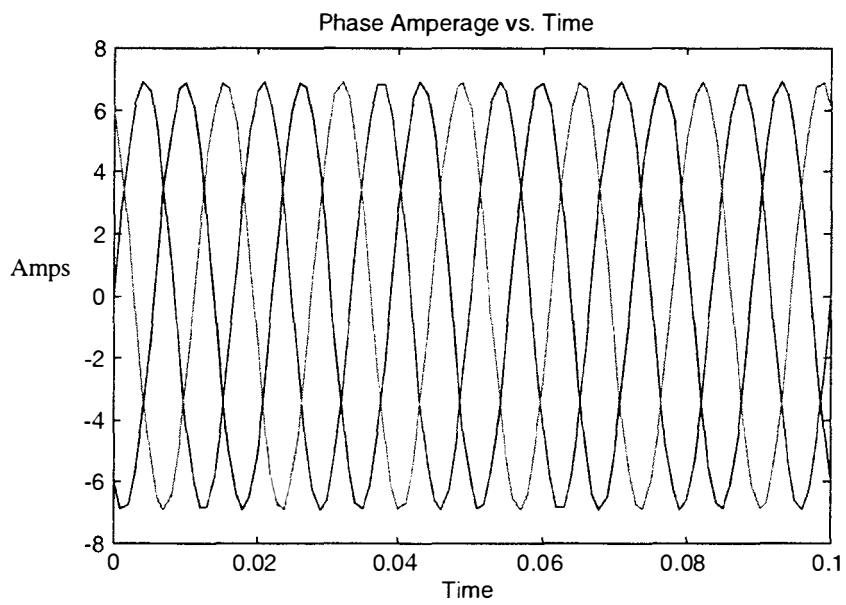


Figure 2.3: Three phase currents for 60 Hz supply.

$$P = \sqrt{3}I_l V_l \cos(\theta) \quad (2.3)$$

where θ is the angular displacement of the sinusoidal voltage waveform to the sinusoidal current waveform, expressed in degrees, and also referred to as the phase angle. Apparent power is expressed in volt-amperes (VA).

However if the system is not balanced or if the waveform is not exactly sinusoidal, instantaneous power can be calculated as

$$P_{instantaneous} = \frac{1}{\sqrt{3}} (I_x V_{xy} + I_y V_{yz} + I_z V_{zx}) \quad (2.4)$$

where the line current and voltages are shown in Figure 2.2. The time average of the instantaneous power over one period of the wave is referred to as active power and has the units of Watts.

Similarly, apparent power is defined using the RMS values of line current and voltages as:

$$P_{apparent} = \frac{1}{\sqrt{3}} (rms(I_x)rms(V_{xy}) + rms(I_y)rms(V_{yz}) + rms(I_z)rms(V_{zx})) \quad (2.5)$$

The ratio of the active power to the apparent power now defines the power factor and is equal to $\cos(\theta)$ for a balanced system:

$$\cos(\theta) = \frac{P_{active}}{P_{apparent}} \quad (2.6)$$

Measurement of the input currents to the motor yields a slightly different graph for the three currents compared to that shown above. The time-domain graph of the currents shows small variations in the input current. The variations are rather small in nature and occur at a higher or lower frequency with respect to the line frequency. The variations in the measured input current to the motor are caused by the electrical design and abnormalities associated with the motor.

2.3.2 Bearing Description

We can obtain some of the frequencies that are caused by the bearing vibration and are defined in the following:

$$\text{For Outer Race Defect:} \quad f_o = \frac{n}{2} f_r \left[1 - \frac{BD}{PD} \cos(\beta) \right] \quad (2.7)$$

$$\text{For Inner Race Defect:} \quad f_i = \frac{n}{2} f_r \left[1 + \frac{BD}{PD} \cos(\beta) \right] \quad (2.8)$$

For Ball Defect:
$$f_b = \frac{PD}{BD} f_r \left[1 - \left(\frac{BD}{PD} \right)^2 \cos^2(\beta) \right] \quad (2.9)$$

Important Bearing Frequencies: $f_o = f_e \pm m.f_o, f_l = f_e \pm m.f_i, f_B = f_e \pm m.f_b \quad (2.10)$

where

$m = 1, 2, 3, \dots$

PD = Pitch Diameter (Refer to Figure 2.4)

BD = Ball Diameter

n = number of balls

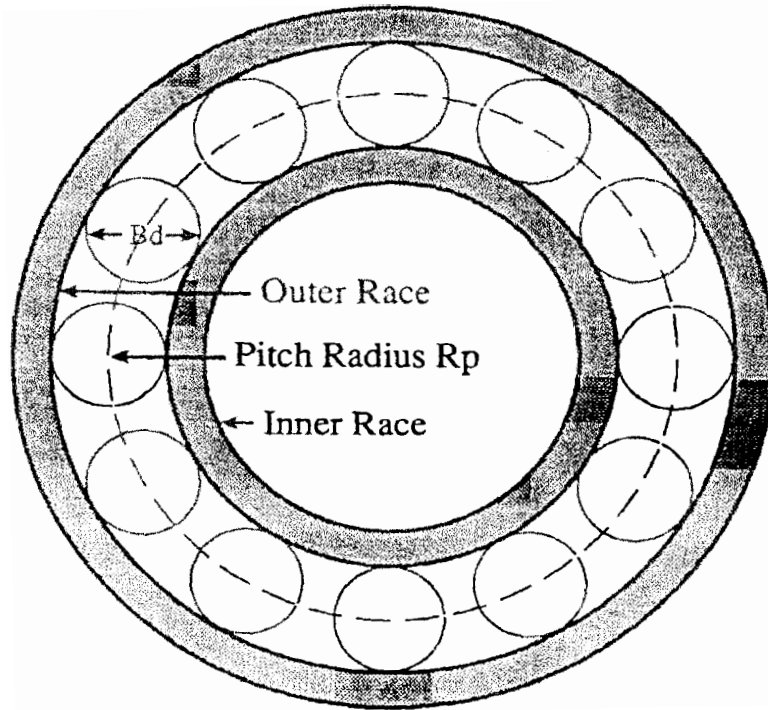
f_r = relative revolutions per second between inner and outer races

f_e = electrical line frequency

These are some of the vibrational frequencies that should be seen in an example measured vibration spectrum. These characteristic bearing vibration frequencies also have integer harmonics associated with each component.

Table 2.1 shows the defect frequencies of the SKF USA Inc.'s motor bearings at a speed of 1000 RPM. To calculate the actual frequency the following formula is used:

$$\text{Frequency Actual (Hz)} = 0.001 * (\text{RPM}) * (\text{Precalculated Frequency}) \quad (2.11)$$



Contact Angle β

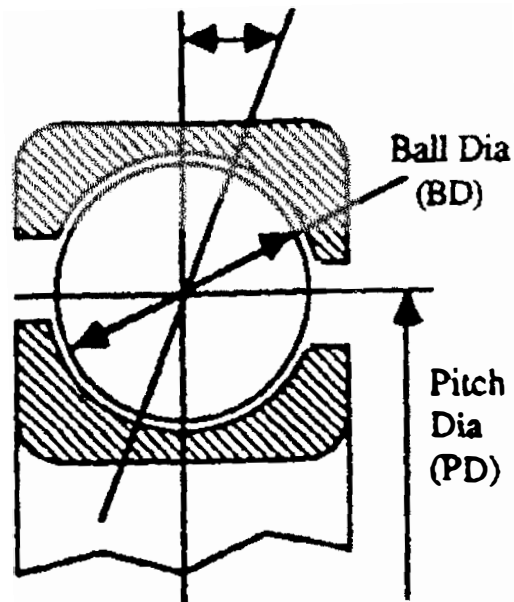


Figure 2.4: Schematic of a bearing.

Table 2.1: Defect Frequencies for the motor ball bearings at 1,000 RPM

Bearing Designation	Number of Balls	Inner Ring Defect (Hz)	Outer Ring Defect (Hz)	Cage Train Freq. (Hz)	Ball Defect Freq. (Hz)	Ball Diameter (mm)	Pitch Diameter (mm)	Contact Angle (degree)
6205	9	90.250	59.750	6.639	78.580	7.938	39.040	0
6206	9	90.530	59.470	6.608	77.039	9.525	46.000	0

2.4 Failure Modes

Failure modes of an induction motor may be divided into four groups, stator, rotor, bearings, and other. The three groups given by stator, rotor, and bearing are the major components of an induction motor and account for 63.6 % of the total failures for motors [40]. Given the large percentage for these three modes, their failure modes will be discussed first. These three main failure modes may be further classified as shown in Table 1.2. For the group given by other fault areas, only the major problems in this group are discussed.

2.4.1 Stator

The stator, as seen in Figure 2.1, is the external group of coils that apply the electromagnetic force that causes the rotor to rotate. The stator is composed of three main parts: the stator frame, the stator core, and the stator windings.

2.4.1.1 *Stator Frame*

'The stator frame encloses the stator and protects it from the environment. It should be robust enough to withstand the mechanical vibrations in the machine and the high torsional forces transmitted to it from the air gap during start-up of the machine or due to excessive load' [41].

2.4.1.2 Stator Core

“Insulation shrinkage of stator core laminations is a problem in large motors, but for smaller motors, this does not pose any significant problem [41].”

2.4.1.3 Stator Windings

“Stator windings consist of an insulated conductor. The conductor is usually copper in larger motors and aluminum in small motors. The insulation is the most common site of failure in electrical machines. Every strand, turn and coil is insulated from its neighbor. Insulation failure leads either to turn-to-turn or phase-to-phase or a three-phase short circuit [41].”

2.4.2 Rotor

The rotor is the rotational part of the motor. This is the inner most part of the motor as shown in Figure 2.1. The electromagnetic field induced in the rotor coils by the stator field causes the rotation. The rotor applies the rotational force to the external equipment. There are three main areas of the rotor: the rotor bars, the shaft, and the frame that allows the rotor bar to be mounted to the shaft.

2.4.2.1 Problems Common to the Rotor

“The rotor is subjected to the same forces as the stator, such as forces due to load, machine vibration, misalignment and environmental factors [41].” Since the windings are not composed of wound copper wire, but instead metal bars, there are different problems associated with the rotor assembly. These problems are discussed below.

2.4.2.2 Rotor Bars

“The hypothesis on which broken-rotor-bar detection is based is that the apparent rotor resistance of an induction motor will increase when a rotor bar breaks... Typical detectors examine axial flux, stator current, rotor velocity, and stator vibration in search of variation resulting from the unbalanced currents caused by the broken rotor bars [42].”

2.4.2.3 Rotational Problems (Vibration and Centrifugal Forces)

If the rotor is not balanced properly, or if the shaft is not straight, the rotation of the rotor will cause vibration. With this rotation, the components of the rotor are also subjected to centrifugal forces that pull the rotor components away from the shaft and away from each other in a direction perpendicular to the axis of rotation.

2.4.2.4 Thermal Expansion and Contraction of Windings

“Rotor cages undergo thermal excursion due to load. The energy losses produced during acceleration are directly related to the energy stored in the rotating inertia of the drive system at rated speed and may produce more pronounced thermal excursions [41].”

2.4.3 Bearings

The bearings allow the rotor to rotate within the motor independent of other motor components. The bearings are composed of four main elements: the seals, the balls (or rolling elements), the inner and outer races, and the cage. The seating is not a physical component, but rather the geometrical relationship between the two motor bearings.

2.4.3.1 Seals

Bearing seals perform two important tasks, first, the seal does not allow outside contamination to infiltrate the interior of the bearing, and second, the seal keeps internal lubrication from leaking into the motor or the environment.

2.4.3.2 Rolling Elements

This element of the bearing transmits the forces between the inner and outer races. If for any reason this element has a defect, the result will be seen in a specific frequency of vibration of the bearing itself. The vibrational frequency is sometimes referred to as the Ball Spin Frequency.

2.4.3.3 Inner and Outer Races

These bearing elements are the surfaces that the rolling elements (ball) are in contact with. The inner race is in contact with the rotor shaft and the outer race is in contact with the motor frame that houses the internal components of the motor (rotor, stator...). Both the inner and outer races are usually securely mounted. If for any reason these elements have a defect, the result will be seen in specific frequencies of vibration of the bearing itself. These vibrational frequencies are sometime referred to as the Ball Pass Frequency of the Inner or Outer Race.

2.4.3.4 Cage

The function of the cage or the retaining element is to keep proper spacing between the rolling elements (Ball) within the bearing. Any problem with this component will also cause a vibration that is related to the frequency that the retainer

turns with the rolling elements. These vibration frequencies are sometimes referred to as Fundamental Train Frequencies.

2.4.3.5 Seating

Seating refers to the alignment of the bearings with each other, with the rotor, and with the stator. Any misalignment will cause vibration of the bearings and also the motor system.

2.4.4 Other Fault Areas

The other main fault areas include electrical faults, loading, bad couplings, and soft foot.

2.4.4.1 Electrical Faults

"The most prominent electrical fault is current imbalance. Current imbalance is mainly caused by imbalance in phase impedances. Resistive imbalance may be caused by a number of defects mainly found in power circuit connections, contacts and external terminals... circuit reactance changes when windings short with one another (turn-to-turn) or with the ground (phase-to-phase) [43]." But, circuit reactance is included with stator or winding faults above. These faults cause the insulation of the windings to overheat and degrade at an accelerated rate.

2.4.4.2 Loading

"Another important reason for the failure of motors is overloading them mechanically. By reducing the shaft stress by 10%, the fatigue life of the shaft increases by 1.8 times and the bearing life increases 37%; a 20% load reduction gives 3.4 times longer shaft life and 95% more bearing life; and a 50% reduction yields 47 times the original shaft life and 800% increase in bearing life [44]."

2.4.4.3 Bad Coupling

A bad coupling refers to the actual linkage between the induction motor and an external system and/or the fact that the external system is not a perfect system. The linkage may be of a number of different types and also the problems that the external system introduces to the induction motor are complex as well.

2.4.4.4 Soft Foot

The term "soft foot" implies that the induction motor is not properly anchored to a base. There are two main reasons why this happens: (1) the motor is not properly secured, and (2) the motor base and/or the anchoring base do not seat together properly.

Chapter 3

ACCELERATED AGING OF INDUCTION MOTORS

3.1 Introduction

The purpose of the aging experiments is to expose electric motors to heat, mechanical stress, moisture and electrical stress in repeated cycles, which will represent the cumulative deteriorating effects of service on an accelerated basis.

“IEEE Standard Test Procedure for Evaluation of Systems of Insulation Materials for Random-wound AC Electric Machinery” and “Standard for Systems of Insulating Materials” have been accepted as basis for designing the test setup [45].

This current study is focused on three different failure modes of electric motors:

1. Bearing failure due to high temperature and moisture
2. Stator winding insulation failure due to high temperature and electrical stress
3. Bearing failure due to fluting.

In order to have a statistical distribution of each motor aging test, three 5-HP motors were tested for each failure mode. All the 5-HP 4-pole induction motors were supplied by US Electrical Motors Division of Emerson Electric Company. The

nameplate information is given in Table 3.1. In addition to these specifications each motor was equipped with 12 J-Type thermocouples, each being mounted at the following locations in the motor:

1. Short-End Bearing
2. Process End Bearing
3. Short-End Winding Insulation 10:00
4. Short-End Winding Insulation 6:00
5. Short-End Winding Insulation 2:00
6. Process-End Winding Insulation 12:00
7. Process-End Winding Insulation 8:00
8. Process-End Winding Insulation 4:00
9. Laminated Steel
10. Air Gap Between Winding and Casing
11. Cover Box Temperature
12. Ambient Temperature Inside Cover Box

The three-phase, cast iron TEFC, explosion-proof horizontal motor dimensions are given in Table 3.2. Refer to Figure 3.1 for the dimensions.

Table 3.1: Nameplate information of the test motors

Type of Motor	Premium Efficiency Motor
Manufacturer	US Electrical Motors Division of Emerson electric Co. St. Louis, MO
Model Number	7965B
Horsepower	5
RPM	1750
Phase	3
Frame Number	184T
Frame Type	TCE
Volts	230 / 460
Amperes	13.0 / 6.5
Service Factor	1.15
Hertz	60
NEMA Design	B
Insulation Class	F
Enclosure	TE
Maximum Ambient Temperature	40 °C
LR KVA Code	J
Duty	Continuous
Power Factor	82.5
Efficiency	90.2
Maximum KVAR	1.4
Shaft End Bearing Type	6206-2Z-J/C3
Drive End Bearing Type	6205-2Z-J/C3
ID #	Z08Z177R190F /

Table 3.2: Horizontal motor dimensions for frame type 184T

Dimension	Value in Inches
A	8-7/8
B	6 ¾
C	16-3/16
D	4 ½
2F	5 ½
H	13/32
Maximum N	2-15/16
Maximum O	9-5/8
Maximum P	9 ½
U	1-1/8
Maximum AA	¾
SQ Key	¼

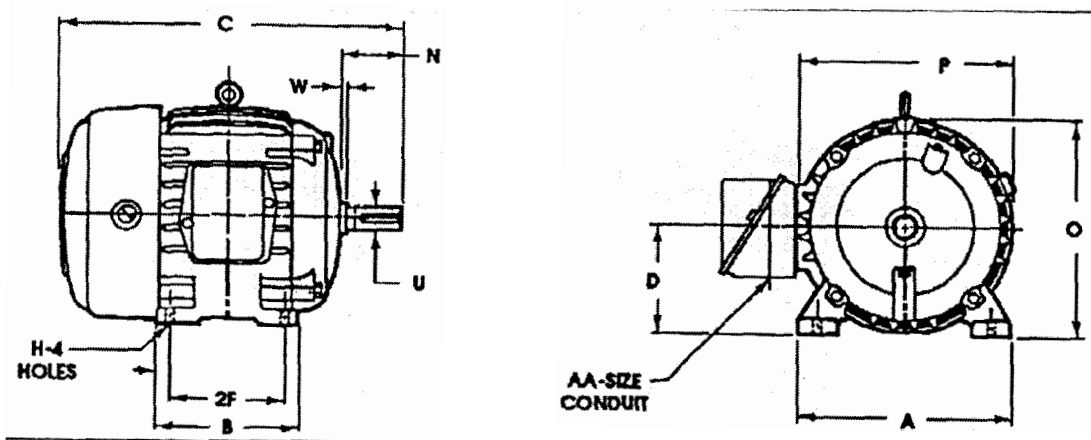


Figure 3.1: Outer schematic of the 5-HP electrical motor.

These motors are tested on a performance test setup, which consists of a 3 kW, 1800 RPM dynamometer (LSC 132 MC 7 Type) and adjustable motor mounting platform coupled with universal joints. The dynamometer is excited with 0 – 2 Amperes and 0 – 200 Volts DC, and loaded with a 21.63 Ohms (at room temperature) open-air resistor.

The test setup is shown in Figure 3.2 and a photograph of the facility is given in Figure 3.3. The motor is loaded with a manual switch, which changes the excitation current and voltage to the dynamometer. A torque sensor and a tachometer are also used for motor load measurements.

3.2 Thermal Aging of Electrical Motor Bearings

The purpose of this setup is to age the electrical motors thermally and chemically on accelerated test cycles. This aging test basically consists of heating the motor at a constant temperature of 140 °C. The motors are totally inserted into a laboratory grade oven with the specifications given in Table 3.3.

The heat is transferred from the bottom up through the oven chamber by natural convection and exits through vents at the oven top. Adjustable fresh-air inlet damper at the oven bottom provides control of the amount of fresh air entering the oven. In order to have less temperature fluctuations the inlet and outlet air vents are closed. A bimetal control thermostat maintains the desired oven temperature. Thermal cutoff limits

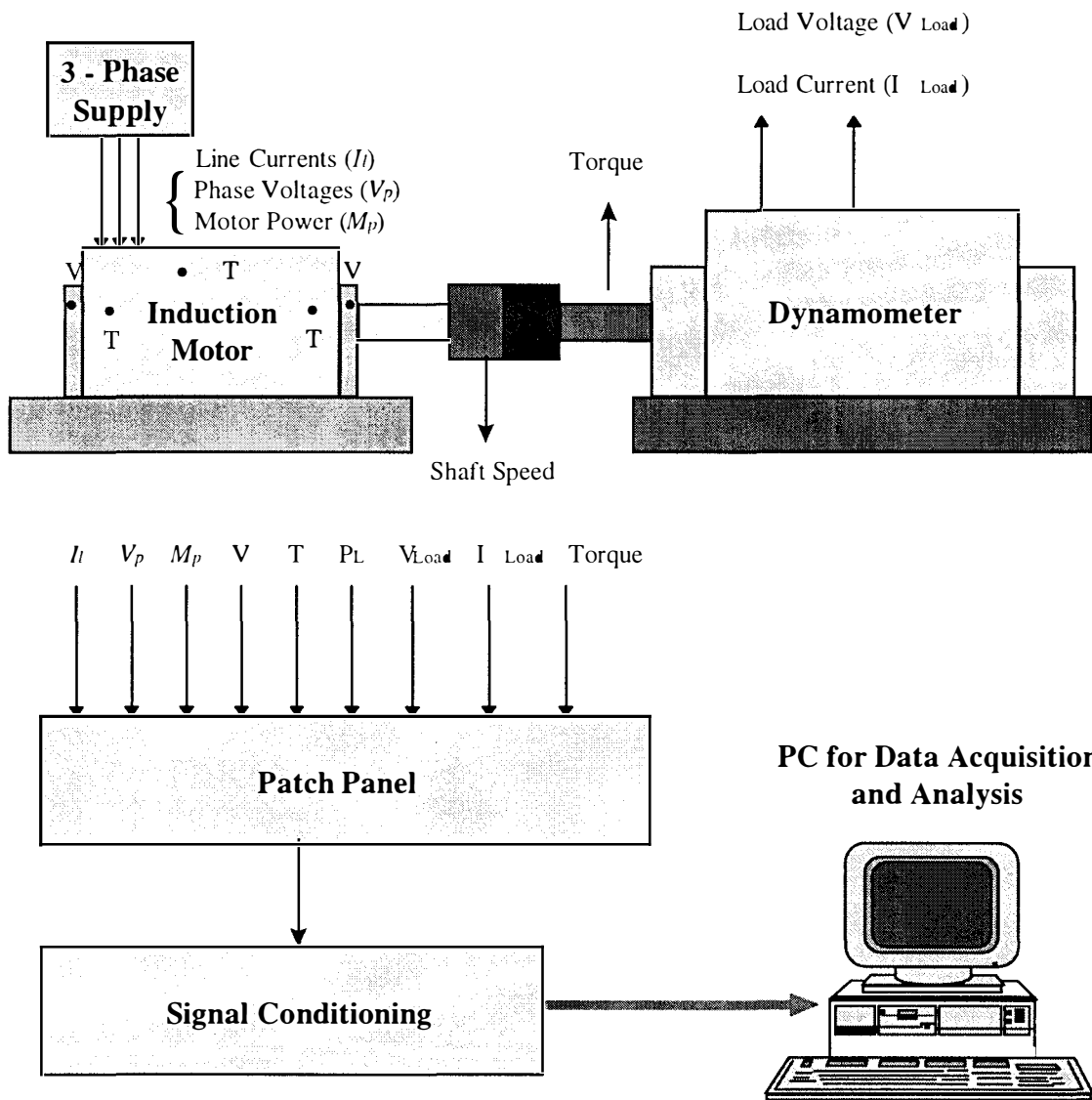


Figure 3.2: Schematic of the motor performance test setup.

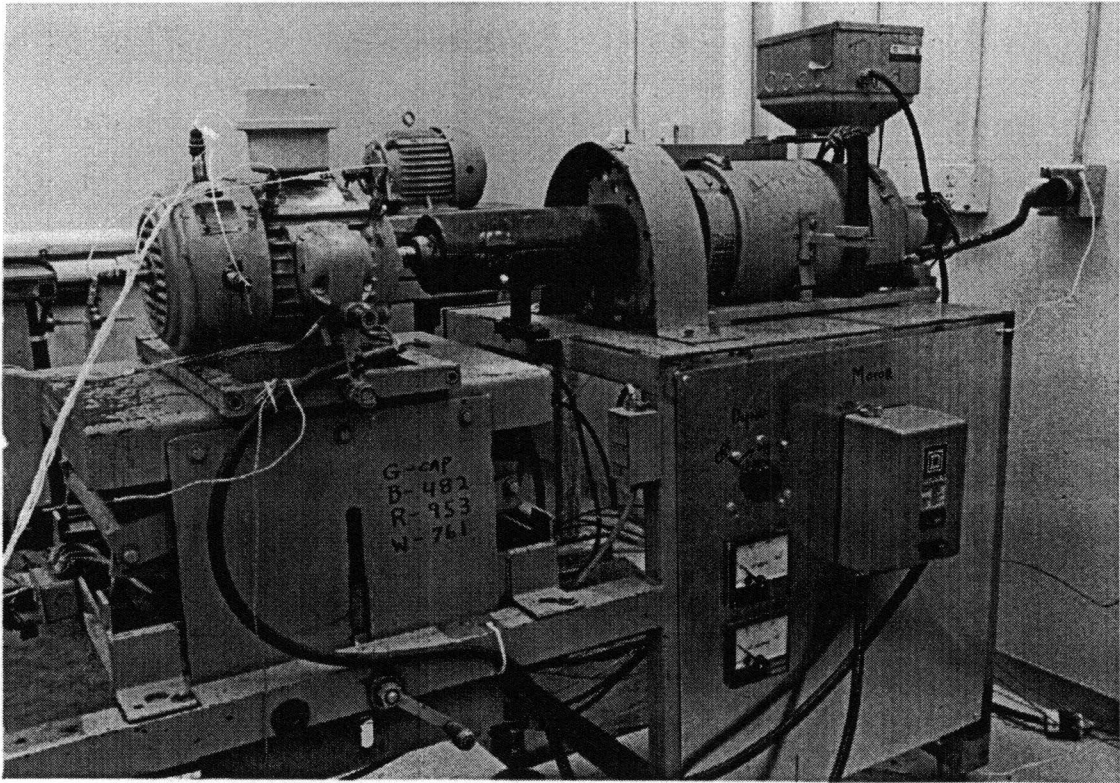


Figure 3.3: Photograph of the motor performance test setup.

Table 3.3: Specifications of laboratory ovens

Manufacturer	The Grieve Corporation 500 Hart Road Round Lake, Illinois 60073-9989
Model #	LW-201C
Work Space Dimensions	18" Width, 12" Depth, 16" Height
Work Space Volume	2 cubic feet
Outside Dimensions	20" Width, 14" Depth, 25" Height
Maximum Temperature	200 °C
Number of Shelves	2
Insulation	1"
Watts	1600
Approximate Shipping Weight	66 lbs.

maximum oven temperature.

Several tests have been performed on the temperature distribution of these ovens as shown in Table 3.4. Table 3.5 shows the steady state temperature variation. As we can see the temperature differences at the bottom and top of the motor is 5 °F on the average and 17 °F maximum. These measurements are without using a thermal shield between the heating elements and the motor. Steady-state temperatures are achieved in about 6 hours, and the temperature distribution on the motor is much better than the ambient temperature distribution. An additional thermal shield, which is basically an aluminum foil on the bottom shelf, reduced the temperature variation inside the oven to a maximum of 10 °F.

The motor bearing aging setup is pictured in Figure 3.4.

The accelerated aging procedure for one cycle of this type of test is as follows:

1. The 5-Hp motor is put into the oven, which is at 140 °C.
2. The motor is heated at this temperature for 72 hours. It is then removed from the oven and cooled for 6 hours.
3. The motor is then fully immersed in water for 15 minutes and then taken out and placed in the oven immediately for another 72 hours of heating. This induced corrosion in the motor components.

Table 3.4: Experimental temperature distribution of the laboratory oven with the electrical motor inside

Elapsed Time	T ₁	T ₂	T ₃	T ₄
0	72 °F	75 °F	76 °F	79 °F
1 hr 10 min.	320 °F	200 °F	211 °F	182 °F
1 hr 45 min.	300 °F	215 °F	226 °F	198 °F
8 hr 15 min.	304 °F	288 °F	273 °F	266 °F

Where

T₁: Ambient temperature on the heating elements of the oven (Thermocouple).

T₂: Ambient temperature at the bottom of the motor (Thermocouple).

T₃: Ambient temperature at the top of the motor (Thermocouple).

T₄: Ambient temperature at the top of the motor (Thermometer).

Table 3.5: Steady-state temperature variation inside the oven without a heat shield

	T ₁	T ₂	T ₃	T ₄
Minimum	281 °F	284 °F	271 °F	
Average	304 °F	288 °F	273 °F	266 °F
Maximum	327 °F	292 °F	275 °F	

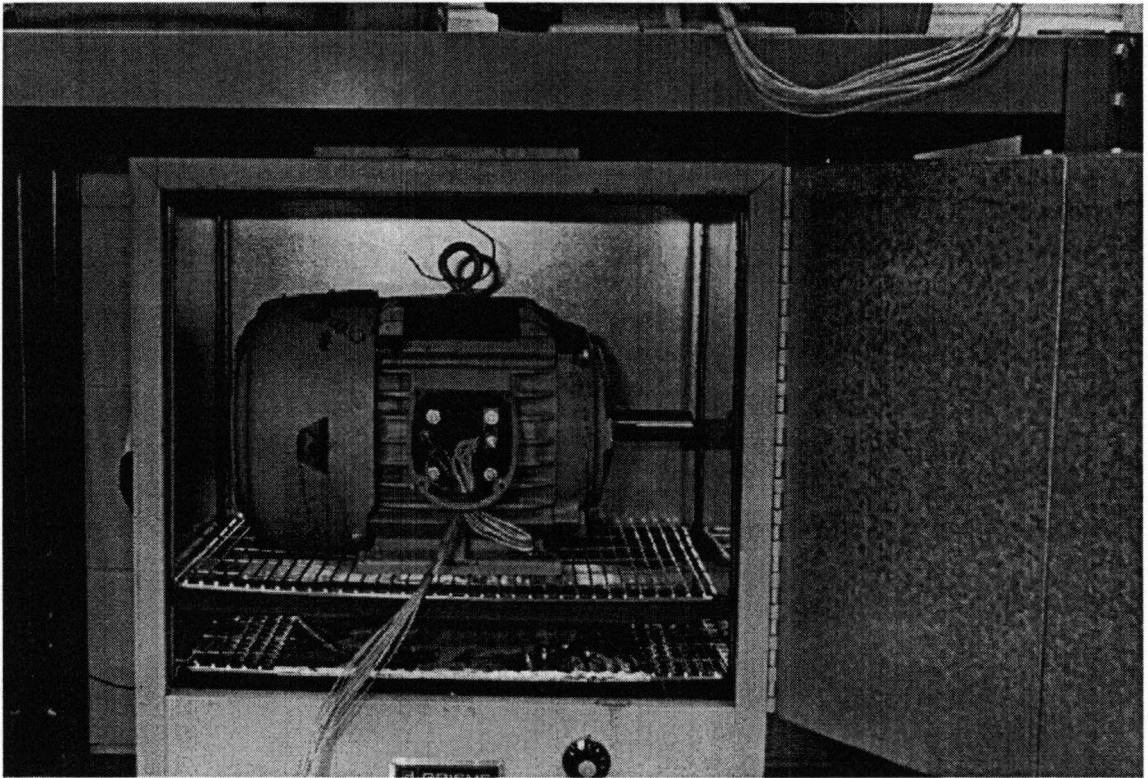


Figure 3.4: The motor bearing thermal aging setup.

4. The motor is taken out of the oven after 72 hours and cooled for 6 hours.
5. The motor is tested on the load performance test setup.

The defect frequencies of the SKF USA Inc.'s motor bearings are given in Table 2.1 and by Equation (2.11).

3.3 Thermal and Electrical Aging of Motor Stator Winding Insulation

Stator winding insulation failure is the second most common failure mode in electrical motors [46]. Table 3.6 shows the main factors and their importance affecting the life of the winding insulation.

For Class F insulation the life of the electrical motor is predicted as shown in Table 3.7 and Figure 3.5.

The electrical motor draws 5 to 10 times greater current at start-up compared to its steady-state operation after a couple of seconds [21]. Since heating during acceleration is proportional to I^2t where I is current and t is

$$t = (Load \times RPM) / (Torque) * (unit \ conversion \ factor) \quad (3.2)$$

Table 3.6: Contribution of stress factors for remaining life calculation [31]

Index	Contribution (%)
Winding Temperature	25
Number of Starts	12.5
Cleanliness	12.5
Humidity	6.25
Operation	12.5
Insulation Resistance	6.25
Polarization Index	12.5
Vibration Level	12.5

Table 3.7: Motor life versus stator winding temperature *

Temperature in °C	Motor Life in Hours
155	20,000
165	10,000
175	5,000
185	2,500
195	1,250
205	625

* Provided for Class F motor winding insulation by USEM.

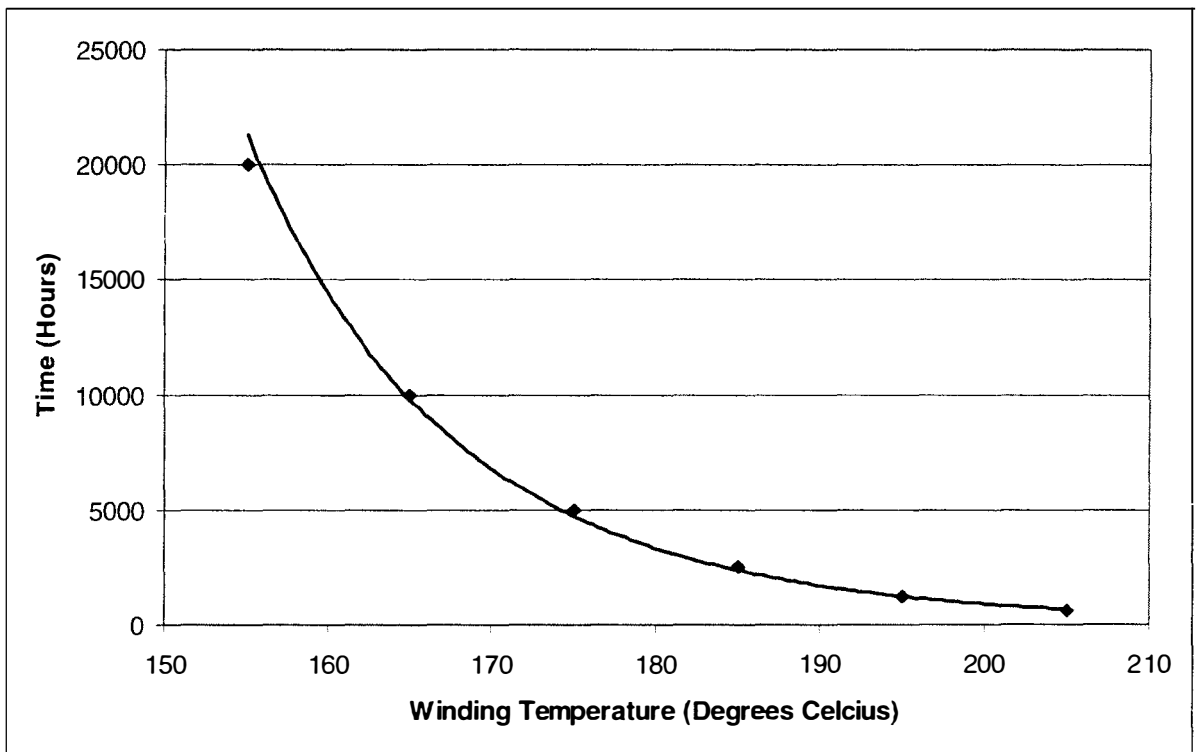


Figure 3.5: Graphical representation of Table 3.7.

The motor itself can be utilized to heat the motor to a desired temperature. In order to accomplish this the motor is subjected to repeated on-off cycling, by controlling the motor relay using a Macromatic Time Ranger TM Multi-Range Plug-In Time Delay Relay model # SS-63122. This unit is powered with a standard 110 Volts AC line source. Upon the application of the power (the on and off button of the whole test setup), a preset delay begins. This is the on delay time and can be programmed from 0.6 seconds up to 24 hours. At the end of the preset delay, the relay contacts transfer and remain in that condition for a specified time, independently adjustable preset time. This time is called the off time and is also adjustable from 0.6 seconds up to 24 hours. At the end of this preset time, the relay contacts drop out and the sequence repeats until the power is cut off.

The Winding Insulation Aging Test has the following time settings, shown in Table 3.8, to heat the winding insulation to 205 °C. In order to achieve this temperature a 100% imbalance in one of the phases is introduced by connecting only 2 out of the 3 wires of the delta connection of the electrical motor. This motor connection is kept always the same in the following order with respect to Figure 2.2:

Phase Z = Connection 7-6-1

Phase X = Connection 4-8-2

Table 3.8: Timer settings and their corresponding times for one cycle to heat the electrical motor winding insulation to approximately 205 °C

Description	On	Off	Period
Heat-Up	3 (~ 1.17 seconds)	0 (= 2.5 seconds)	30 minutes
Transient	1	1 - 7.2	1 hour
Steady-State	1 (~ 0.79 seconds)	7.2 (~ 5.84 seconds)	47 hours

Figure 3.6 is a photograph of the electrical motor winding insulation thermal aging test setup.

In order to reduce the bearing failure due to high temperatures, the bearings are lubricated with SRI Grease NLGI 2 after each aging cycle. Also, the plastic cooling fan has also been replaced whenever it melted due to high temperature. The setup was also thermally isolated in order to reduce heat losses from the motor.

3.4 Fluting of Electrical Motors

Fluting is a fault in which the ball bearing surface, and / or the inner and outer races of the bearing get damaged caused by electrical sparking. As a result, rolling elements and the races get damaged. This surface degradation causes extreme vibration levels of the bearing and its eventual failure.

In order to simulate the electrical discharge from the shaft to the bearing, a special test setup was designed. The test setup schematic is shown in Figure 3.7.

The fluting run has duration of 30 minutes with the motor rotating at no load, and an externally applied shaft current of 27 Amperes at 30 Volts AC.

The motor is delta connected with the following wire configuration with respect

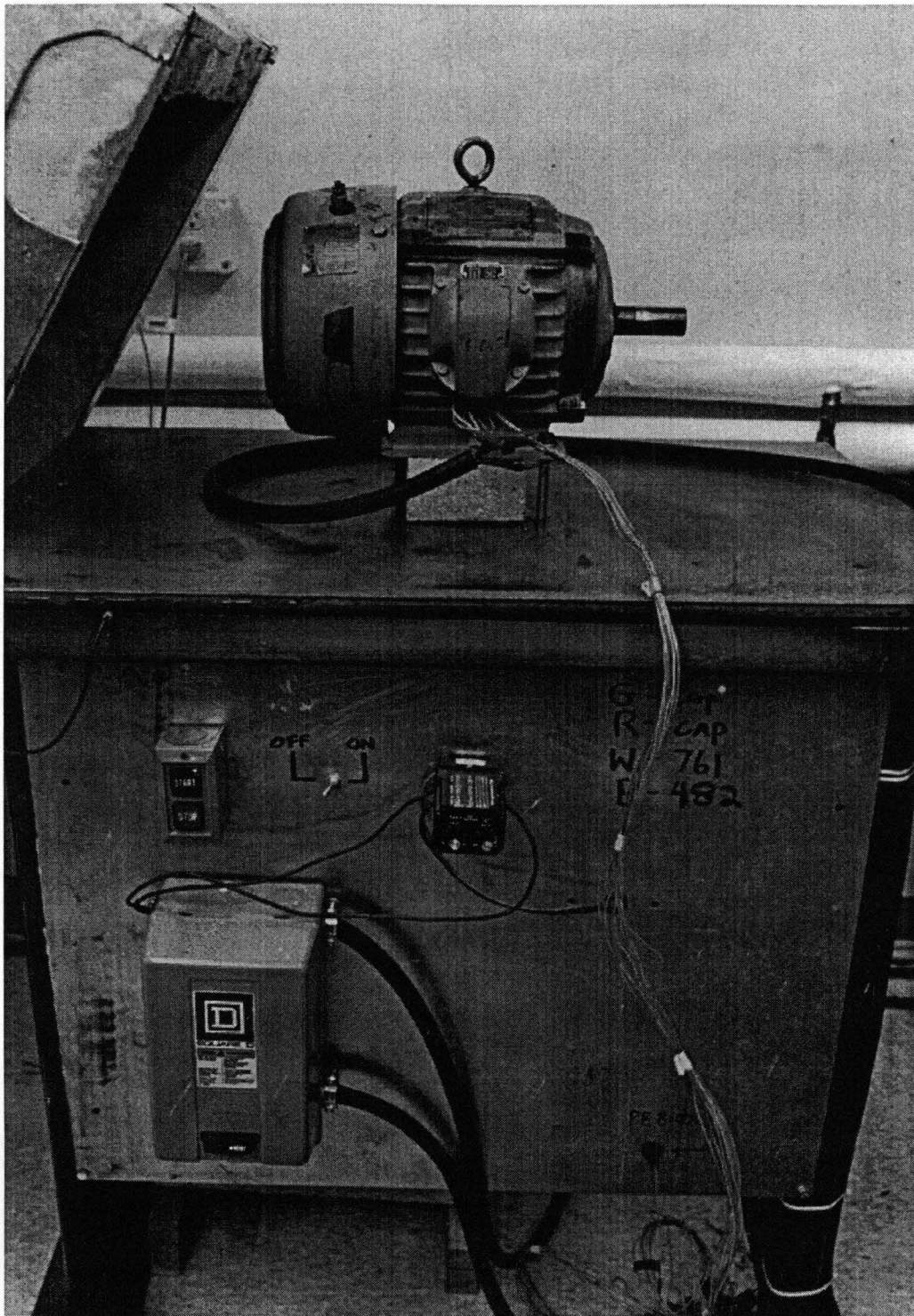


Figure 3.6: Photograph of the electrical motor stator winding insulation thermal aging test setup.

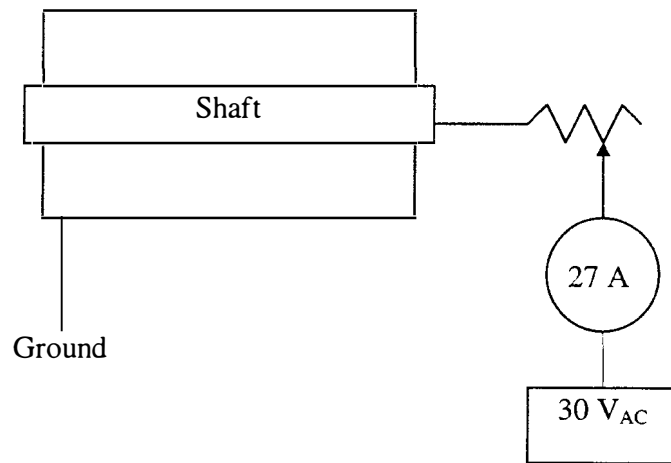


Figure 3.7: Schematic of the electrical motor ball bearing fluting.

to Figure 2.2:

Phase X (Black): Connection 4-8-2

Phase Y (Red): Connection 9-5-3

Phase Z (White): Connection 7-6-1

As was the case with other tests, this wire connection was preserved for all test motors and for the connection for the performance test setup.

The fluting aging was followed by thermal and chemical aging in order to increase and accelerate the aging process. One test cycle is defined as follows:

1. The 5-HP motor is fluted for 30 minutes on the fluting test setup.
2. Afterwards, the 5-HP motor is placed in the oven, which is at 140 °C.
3. The motor is heated at this temperature for 72 hours. It is then removed out of the oven and cooled for 6 hours.
4. The motor is then fully immersed in water for 15 minutes and then taken out and placed in the oven immediately for another 72 hours of heating. This induced corrosion in the motor components.
5. The motor is taken out of the oven after 72 hours and cooled for 6 hours.
6. The motor is tested on the load performance test setup.

Chapter 4

MOTOR LOAD PERFORMANCE TESTING

4.1 Introduction

At the end of each aging cycle, the motor performance was tested under various loads. These varied from zero to 115% of full load. These tests also provide information about the condition of the motors by monitoring mechanical, electrical and thermal parameters. The following measurements were recorded by the data acquisition system.

- 12 temperatures,
- 3 motor voltages,
- 3 motor currents,
- Dynamometer load current,
- Dynamometer load voltage,
- Motor speed,
- Load torque,
- Acceleration at 6 different locations on the motor using piezoelectric accelerometers.

The heart of the experiment is the data acquisition in proper manner. For this purpose the highest sensitive measurement equipment were utilized as much as possible.

Making the data acquisition and the sensor integration as flexible as possible was another task which was accomplished during this setup. The hardware interface of the data acquisition allows the user to further extend the number of sensors and incorporate this into the software. Signal amplification, filtering and acquisition were performed with National Instruments' hardware and software.

4.2 Design and Installation

The motor performance test setup consists of a 3 kW dynamometer, an adjustable motor platform, a universal coupling, a tachometer and a torque sensor.

The load cell of the torquemeter, which is located 10 inches away from the center of the dynamometer with an arm, has the specifications shown in Table 4.1. The details of the torquemeter unit, which also produces an analog output for the data acquisition board, are specified in Table 4.2. It provides power to the load cell and outputs and displays the torque value.

The tachometer consists of a retro-reflective sensor and a panel tachometer with an analog output module. The retro-reflective sensor measures the shaft speed of the motor by producing a square pulse for each beam of light reflected from the motor shaft. The tachometer panel specification is given in detail in Table 4.3.

Table 4.1: Load cell specification

Type	Super-Mini Load Cell
Manufacturer	Interface, Inc. 7401 E. Butherus Dr. Scottsdale, Arizona 85260
Model	SM-100
Serial #	D09912
Capacity	100 lb. _f
Output, Tension	3.005 mV/V
Input Resistance	350 + 40 / -3.5 Ohms
Output Resistance	350 ± 3.5 Ohms
Non-Linearity	< ± 0.03% Rated Output
Hysteresis	< ± 0.02% Rated Output
Temp. Range Compensated	0 to 150 °F
Temperature Effect on Zero	± 0.15% Rated Output / 100 °F
Zero Balance	< ± 1% Rated Output

Table 4.2: Selected torquemeter specifications

Type	INFCS INFINITY C Strain Gage Meter
Manufacturer	Newport Electronics, Inc. 2229 South Yale St. Santa Ana, CA 92704-4426
Model #	9210-001
Serial #	6141673
Analog to Digital Internal Resolution	15 bits
Accuracy at 25 °C	± 0.03% of Reading, 1 – 2 sec. Step Response
Analog Output Linearity	0.2%
Analog Output Step Response Time	2 – 3 seconds to 99% of Final Value

The torquemeter was calibrated for 0 – 30 lb._f to produce an analog output of 0 – 10 Volts.

Table 4.3: Tachometer specifications

Type	Panel Tachometer
Model	DT-5TG with DOP-FV add-on and RS220H sensor
Accuracy	$\pm 0.5\%$ F.S.
Response Time	20 mS > 50 Hz
Sampling Time	10 ms
Max. Ripple	50 mV p-p
Operating Temperature	0 °C to 45 °C

The tachometer was calibrated to produce a 0 – 10 Volts output for 0 – 2000 RPM.

Additional transducers and signal conditioning equipment were installed on a portable platform, which is pictured in Figure 4.1. This platform holds the following equipment:

- Tachometer panel
- Strain gage meter (torquemeter)
- 2 Power supplies for accelerometers
- 4 voltage transducers
- 4 Hall-effect current sensors
- 2 power supplies for current sensors
- National Instruments' SCXI signal conditioning units
- Universal Patch-Panel for easy sensor-output integration
- 110 Volts AC surge protector.

A schematic of the connections of these units on the platform is shown in Figure 4.2.

The patch-panel that serves as an interface to the data acquisition and instrumentation is shown in Figure 4.3.

The voltage transducers, used to measure the voltage in each phase and to measure the load voltage on the dynamometer, are described in detail in Table 4.4.

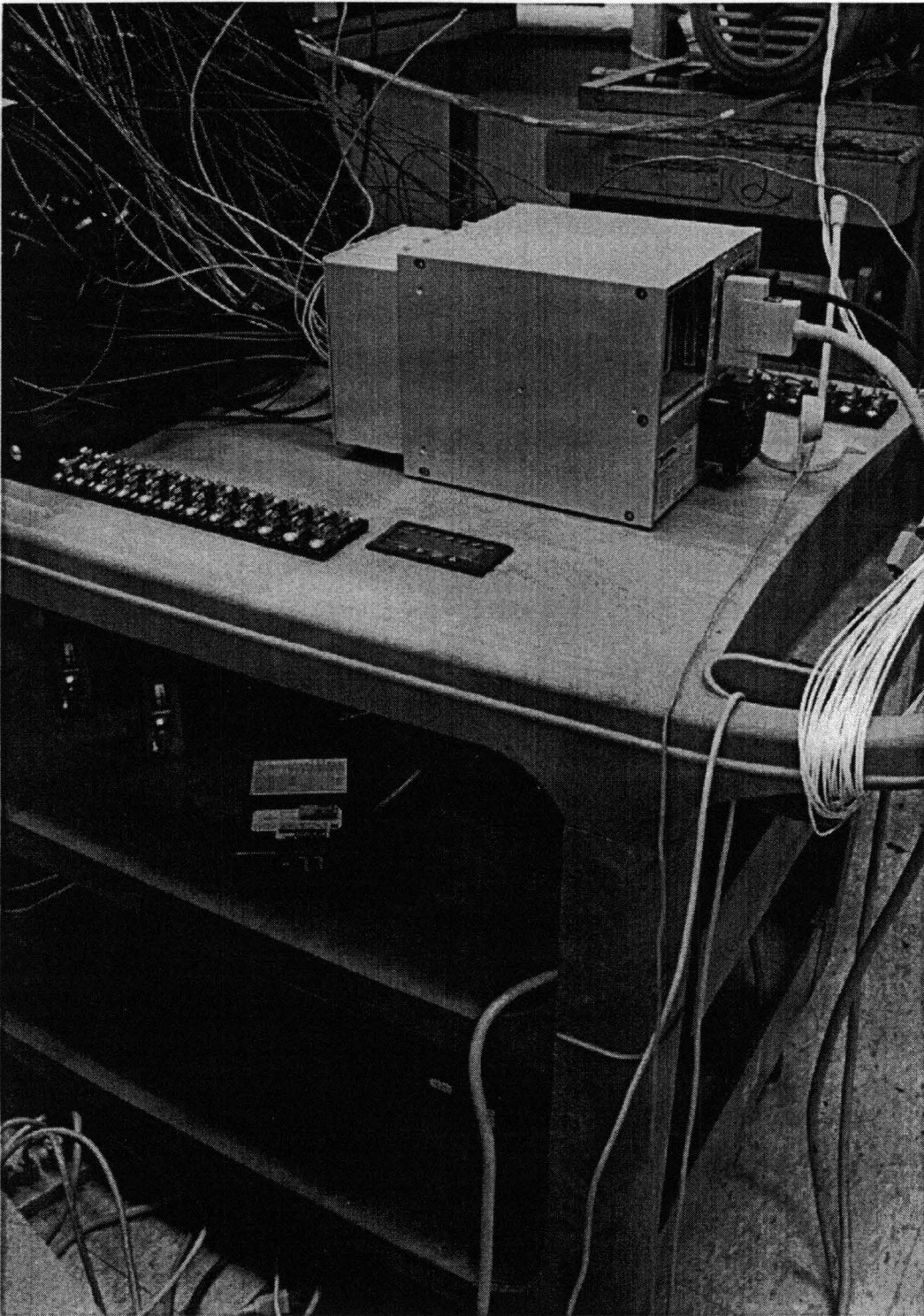


Figure 4.1: Portable remote sensors, signal conditioning equipment and the signal patch panel.

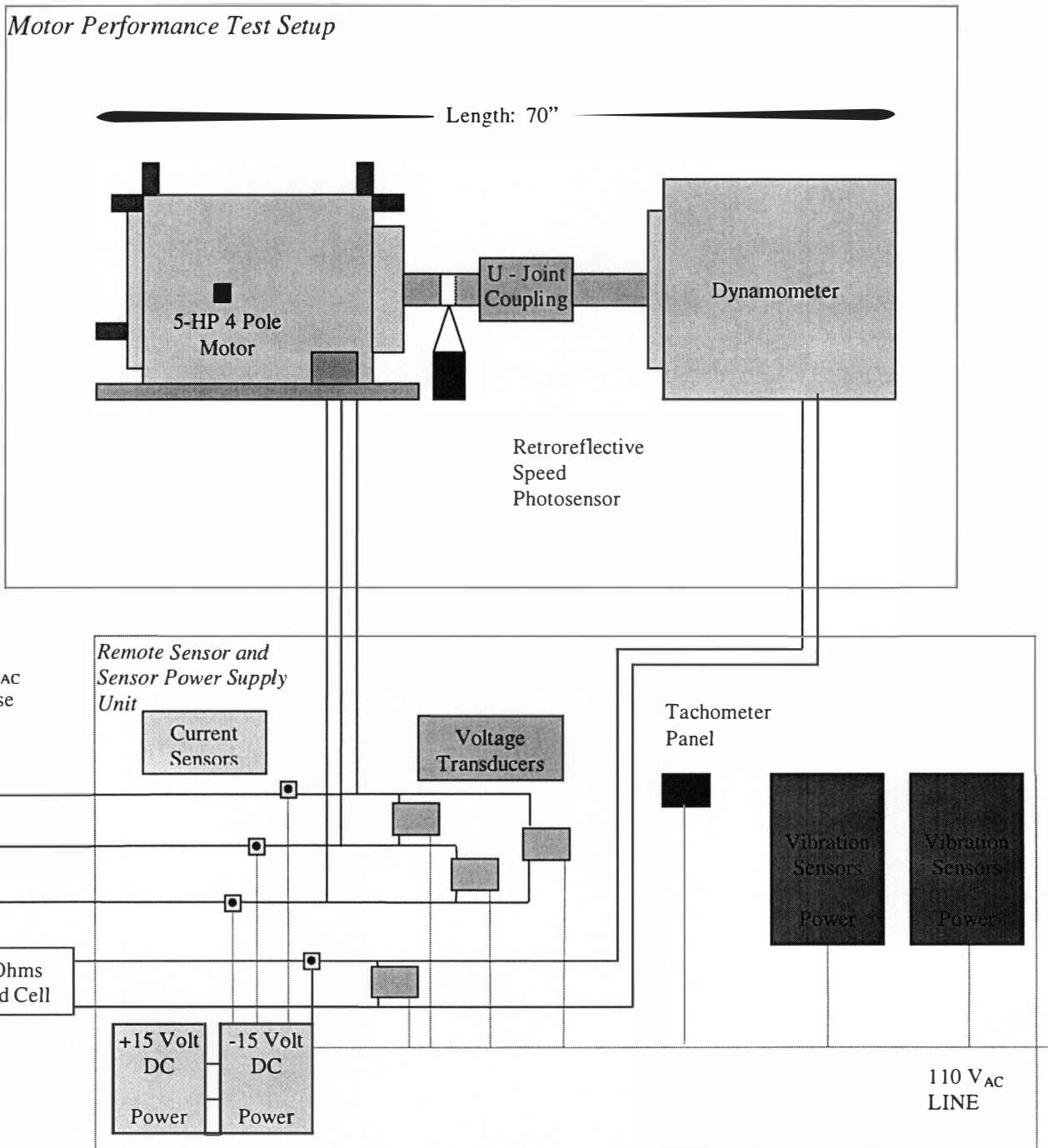


Figure 4.2: An outline of the connections of the sensors and the power supplies to the motor performance test setup.

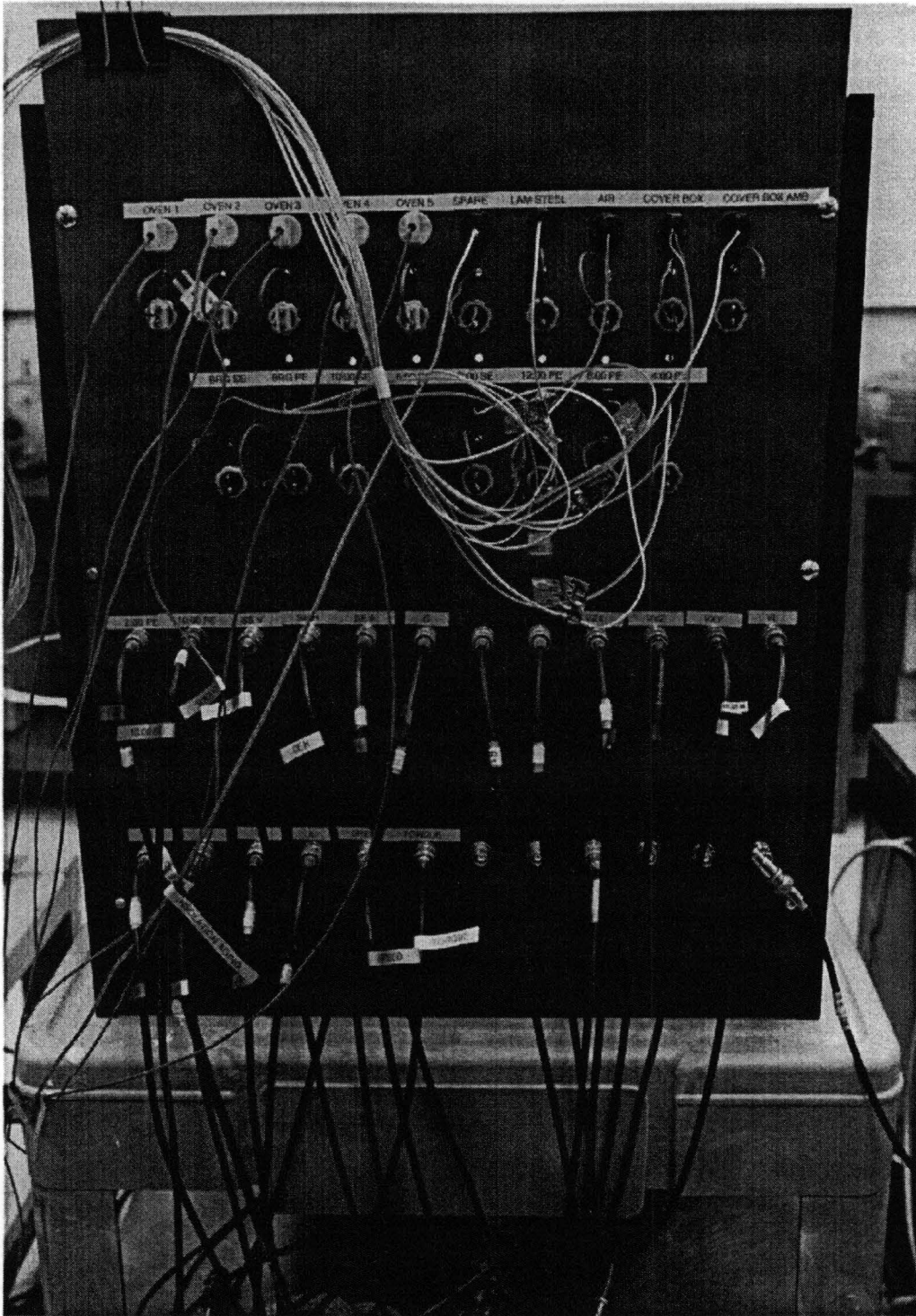


Figure 4.3: Photograph of universal patch-panel.

Table 4.4: Specifications of the voltage transducers used

Sensor Type	DC voltage Transducer
Manufacturer	Ohio Semitronics, Inc. 4242 Reynolds Dr. Hilliard, Ohio 43026-1264
Model #	VT7-009D
Frequency Range	DC to 10000 Hertz
DC Voltage	0 to 500 Volts
Output	$\pm 10 V_{DC}$
Overload	2 times full scale rating
Dielectric Test (Input / Output / Case)	1500 V_{AC}
Burden	Greater than 100000 Ohms
Response	50 microseconds
Field Adjustable Cal.	$\pm 10\%$
Accuracy	$\pm 0.25\%$ F.S. @ dc. Including effects of linearity and repeatability
Temperature Effect	-10 °C to 60 °C: $\pm 1.0\%$ RDG.
Output Loading	2000 Ohms minimum
Instrument Power	110 V_{AC} , 60 Hz, 5 VA

If a conducting material is placed in a magnetic field perpendicular to the direction of current flow, a voltage is developed across the material in a direction perpendicular to both the initial current direction and the magnetic field. This voltage is called the Hall voltage, after E. H. Hall who first observed the effect in 1879. The Hall voltage arises from the deflection of the moving charge carriers from their normal path by applied magnetic field and its resulting transverse electric field. The Hall Effect current sensors, which were used to measure the phase currents of the motor and the current of the dynamometer load, are specified in detail in Table 4.5.

The voltage output of the voltage transducer and the current sensor are directly proportional to the input waveform. However, the current sensor had an offset, which is measured at the beginning of the data acquisition and subsequently biased.

Another type of sensors used in this experiment was high bandwidth accelerometers. The accelerometers are placed at the following locations on the motor:

- Process end 2:00
- Process end 10:00
- Short end vertical
- Short end horizontal
- Short end axial

Table 4.5: Specifications of current transducer

Sensor Type	Hall Effect Current Sensor
Manufacturer	Ohio Semitronics, Inc. 4242 Reynolds Dr. Hilliard, Ohio 43026-1264
Model #	CTF-501T
Input Current	500 A _{AC} , 700 A _{DC}
Output @ 500 A	100 mA
Temperature Effects	- 40 °C to 80 °C: ± 0.5% FS
Accuracy	± 0.3% FS Including effects of linearity, setpoint, and offset @ 25 °C.
Linearity	± 0.1% FS
Overcurrent	800 A 3 Min/H
DC Insertion Loss	None
Response Time	1 microseconds
di/dt	50 A/μsec
Bandwidth	150 kHz
Isolation (Input / Output)	2500 V _{AC}
Load	250 Ohms
Supply Voltage	± 15 V ±1 V _{DC}
Excitation Current	< ± 135 mA
Quiescent Current	< 35 mA + I _N

- Cover box

A description of these accelerometers is given in Table 4.6 and shown in Figure 4.4.

As mentioned earlier, the motors were also equipped with 12 J-type thermocouples. Each of these thermocouples had a temperature range of 0 to 760 °C and could be used in oxidizing, reducing, inert or vacuum atmospheres. It was composed of iron and constantan.

On the other hand, the ovens in which the electrical motor bearings were thermally aged did not require such precise measurements. Therefore, to monitor the oven temperatures, K-type thermocouples were used. These thermocouples operate from -200 to 1260 °C and were composed of Chromega™ and Alomega™.

All the sensor outputs were terminated at the patch panel. The signals were preprocessed through a signal conditioning device. This was manufactured by National Instruments and consists of the following components:

- Model SCXI-1000 chassis with 110 V_{AC} power supply pictured in Figure 4.5.a.

Table 4.6: Specifications of high bandwidth accelerometers

Sensor Type	High Frequency Industrial Accelerometer
Manufacturer	Industrial Monitoring Instrumentation 3425 Walden Av. Depew, New York
Model #	323B01
Sensitivity	100 mV/g \pm 5%
Measurement Range	\pm 50 g
Resolution	0.0002 g
Frequency Range	144 – 480000 cpm \pm 5% 102 – 600000 cpm \pm 10% 48 – 9000000 cpm \pm 3 dB
Mounted Resonant Frequency	40 kHz
Amplitude Linearity	\pm 1%
Transverse Sensitivity	\leq 5%
Shock Limit	5000 g pk
Temperature Range	-54 to 121 °C
Settling Time within \pm 1% of Output Bias	5 seconds
Discharge Time Constant	\geq 0.2 seconds
Spectral Noise	4.0 μ g/ \sqrt Hz (10 Hz) 1.2 μ g/ \sqrt Hz (100 Hz) 0.4 μ g/ \sqrt Hz (1 kHz)
Case Isolation Protection	RFI / ESD
Broadband Electrical Noise	200 μ g (1 – 10 kHz)
Mounting Thread	10-32 female
Sensing Element Structure	Ceramic / shear
Power Supply	Model 482A05 I.C.P. Power Supply

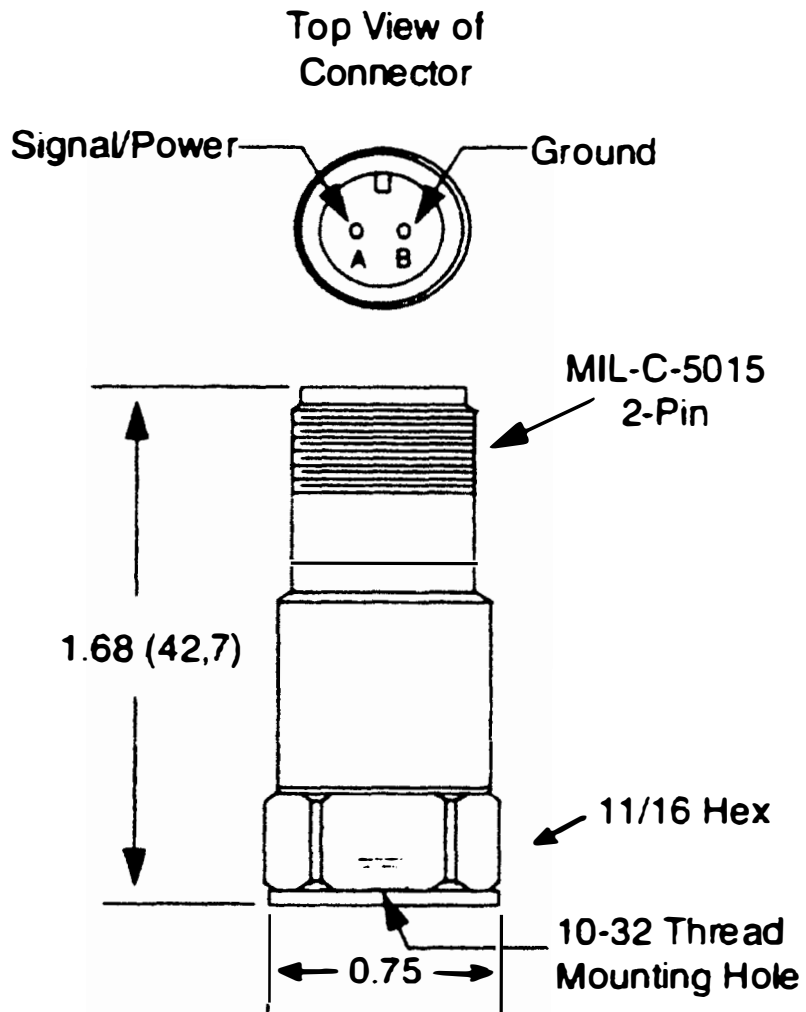
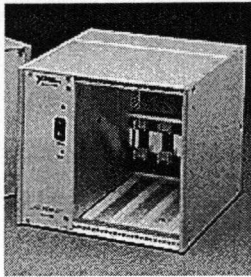
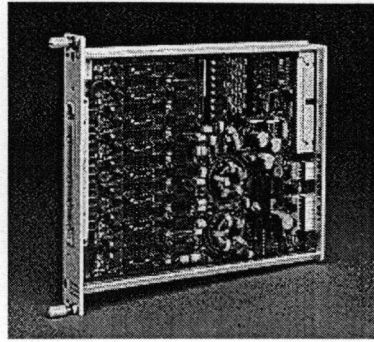


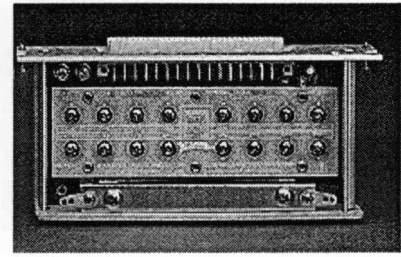
Figure 4.4: Schematic of the high bandwidth accelerometer.



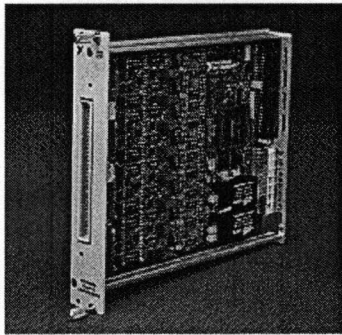
a.



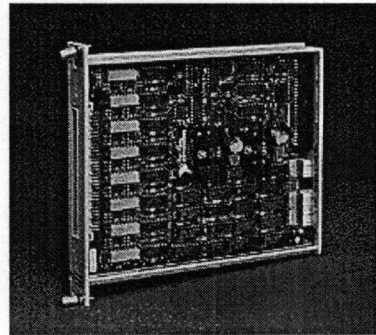
b.



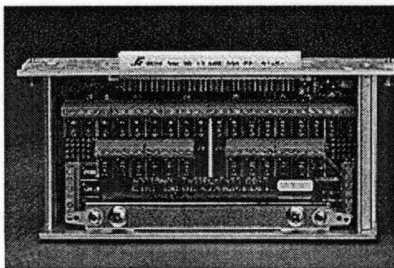
c.



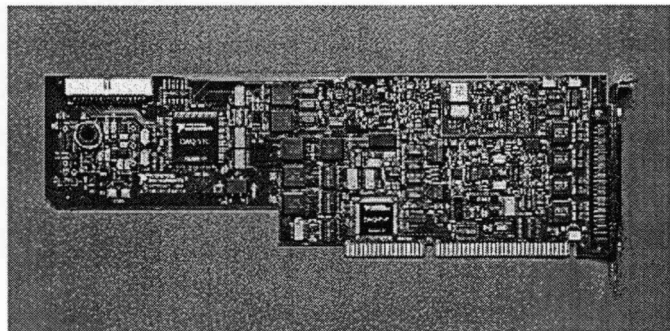
d.



e.



f.



g.

Figure 4.5: Photographs of various parts of the data acquisition hardware.

- Two model SCXI-1120 8-channel isolation amplifiers with SCXI-1328 terminal blocks for measuring 12 motor temperatures and one ambient room temperature using J-type thermocouples. The units have 250 V_{rms} working isolation. The units are configured for a gain of 200 and a 3-pole RC filter that has a cutoff frequency of 4 Hz. The units can scan up to 333 kS/s. One of the units is pictured in Figure 4.5.b with the terminal block in Figure 4.5.c.
- Model SCXI-1141 8-channel elliptic low-pass filter module. This unit was used to filter the measurements from the accelerometer and amplify them. Each channel has an eighth-order elliptic low-pass filter with an –80 dB stop-band and 135 dB/octave rolloff. These filters were of hybrid switched capacitor filters and continuous time filters which are software programmable in order to prevent aliasing in the digitized data. Figure 4.5.d shows a picture of this unit. It had a scan rate of 120 kS/s. A model SCXI-1328 terminal block was also used for AC coupling and signal ground referencing. The terminal block included isothermal construction that minimized errors caused by thermal gradients between terminals and the high-precision cold-junction sensor. Low-pass filters were set 1/3rd of the sampling frequency.
- Model SCXI-1100 32-channel multiplexer amplifier took the remaining type of signals and conditions them for digitizing. It had a scan rate up to 250 kS/s. The module gain was set to 1 and the low-pass filter was set to 10kHz. The unit is

pictured in Figure 4.5.e and was used with model SCXI-1300 general-purpose terminal block that had an onboard temperature sensor for measuring the reference junction temperature when using the thermocouples. The terminal block is pictured in Figure 4.5.f.

- After signal conditioning with National Instruments' SCXI modules, the signals were digitized using a data acquisition board. This is National Instruments' AT-MIO-16E3 Series Multifunction I/O board for ISA bus systems. It was located in an ISA bus of the IBM compatible Personal Computer. The data acquisition board could sample 32 differential channels up to 500 kS/s with 12-bit resolution. Its software selectable gain is set to 0.5, giving an input range of ± 10 Volts. The CMRR, DC to 60 Hz is 95 dB and relative accuracy is ± 1.5 LSB. System noise is $0.15 \text{ LSB}_{\text{rms}}$. This hardware is pictured in Figure 4.5.g.
- Finally, the digitized data was processed and stored in an IBM compatible PC. It had a 200 MHz Pentium processor with 64 MB EDO RAM, 2 GByte hard disk space, one CD-ROM, one Write-Once-Read-Many CD-ROM, one 100 MB external Zip drive, one 3.5 inch floppy drive, 128-bit graphics card with 4 MB VRAM, 17" color monitor working at 1600x1200 resolution, one Epson Stylus Color 600 Printer, 16-bit stereo sound card and a PCI Ethernet card. The PC has well enough computing power for on-line data acquisition and data processing.

database management, data storage and retrieval, program development, internet communications and report generation. This system is shown in Figure 4.6.

4.3 Computer Software For Aging Monitoring and Motor Performance Testing

4.3.1 Data Acquisition Program

The data acquisition software was developed using National Instruments' LabWindows/CVI under Microsoft Windows 98. It is a full-featured C programming environment in which the data acquisition system was developed as a stand-alone executable.

The following are the main functions of the data acquisition software:

- Calibrate the SCXI signal conditioning modules and the data acquisition board before motor performance test-run.
- Acquire high frequency data sampling at 12,000 Hz for 10 seconds and low frequency data at 666.67 Hz for 60 seconds per channel. Before 05/22/1997, the low frequency data were sampled at 200 Hz for 60 seconds in which aliasing had occurred.



Figure 4.6: Data acquisition system showing the end-user interface.

- Display RMS and mean values of the signals instantaneously to the user as shown in Figure 4.7. If desired, time and frequency plots could also be displayed on-line. Spectral plots were produced using an FFT window of 16384 points and 50% overlap. The window function (e.g. Hanning, Blackman, etc.) is user selectable and has a default Blackman window defined by:

$$W_i = 0.42 - 0.5\cos(2\pi i/n) + 0.08\cos(4\pi i/n) \quad \text{for } i = 0, 1, \dots, n-1 \quad (4.1)$$

- Provide a visual alarm when the motor operation reaches steady state. The criterion of steady-state operation was defined by USEM as follows. If the motor temperature does not change more than 2°C within 30 minutes then the motor is said to be at steady-state operation. As this kind of validation imposed a longer time period, the steady-state validation was modified as: if the motor temperature does not change more than 0.2°C within 3 minutes then the motor is said to be at steady-state operation.
- Record the acquired data into binary data files. This is explained in detail in the following section.

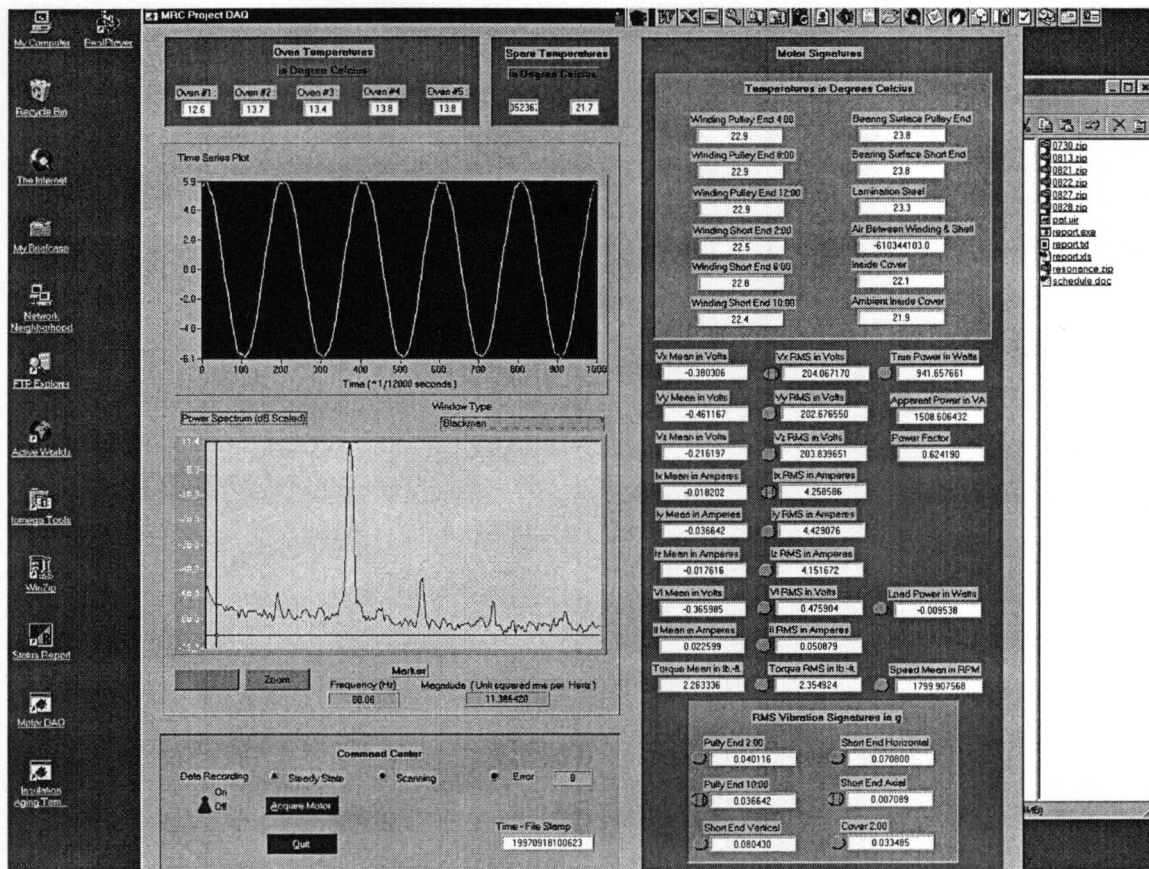


Figure 4.7: Graphical User Interface of the data acquisition system.

4.3.2 Recorded Data Format

There are mainly 2 data file formats: the “.tc” file recorded the temperatures every minute in 8-byte double precision format in the following order:

1. Oven #1
2. Oven #2
3. Oven #3
4. Oven #4
5. Oven #5
6. Spare (Usually temperature on the Winding Insulation Aging Test)
7. Lamination
8. Air Between Winding and Shell
9. Cover
10. Ambient inside cover
11. Winding Pulley End 4:00
12. Winding Pulley End 8:00
13. Winding Pulley End 12:00
14. Winding Short End 2:00
15. Winding Short End 6:00
16. Winding Short End 10:00
17. Bearing Surface Pulley End

18. Bearing Surface Short End.

After 05/22/1997, this file format was expanded in the following order:

1. Oven #1
2. Oven #2
3. Oven #3
4. Oven #4
5. Oven #5
6. Winding Pulley End 4:00
7. Winding Pulley End 8:00
8. Winding Pulley End 12:00
9. Winding Short End 2:00
10. Winding Short End 6:00
11. Winding Short End 10:00
12. Bearing Surface Pulley End
13. Bearing Surface Short End
14. Spare (Usually temperature on the Winding Insulation Aging Test)
15. Lamination
16. Air Between Winding and Shell
17. Cover
18. Ambient inside cover

19. Ambient outside the motor.

The “.mt” file holds all other motor signatures. First the high frequency data, then the low frequency data were recorded in the following order:

1. V_{xy} (Volts)
2. I_x (Amperes)
3. V_{yz} (Volts)
4. I_y (Amperes)
5. V_{zx} (Volts)
6. I_z (Amperes)
7. V_{load} (Volts)
8. I_{load} (Amperes)
9. Speed (Revolutions per minute)
10. Torque (Pound - feet)
11. Cover Vibration (g)
12. Short End Axial Vibration (g)
13. Short End Horizontal Vibration (g)
14. Short End Vertical Vibration (g)
15. Pulley End 10:00 Vibration (g)
16. Pulley End 2:00 Vibration (g)

After 05/22/1997, this file format was changed so that the low frequency contains 60 seconds of 666.67 Hz data instead of 200 Hz.

The naming of these files was such that, the time stamp included year, month, date, hour, minute and second. In this way, the time stamp information can always be calculated from the file name itself, saving valuable disk space.

The order of these data files is as follows:

1. 0% start-up (dynamometer uncoupled)
2. 0% steady-state (dynamometer uncoupled)
3. 0% shutdown (dynamometer uncoupled)
4. 0% start-up
5. 25% load
6. 50% load
7. 75% load
8. 100% load
9. 115% load
10. 115% shutdown

It should be noted that there were no shutdown data available before 05/22/1997.

Chapter 5

MOTOR FAILURE DETECTION

5.1 Introduction

In this section, the various on-line (non-intrusive) detection methods to determine the failure mode or modes is listed and examined. These methods are also the basis for tabulating the “Sensitivity Matrix” for various fault modes.

The “Sensitivity Matrix” is a matrix representing various signatures, which are sensitive to specific motor faults. Using this matrix, trends of motor aging are established. When the accelerated aging effect is taken into account as a scale factor in these trends, the remaining life of the motor may be estimated easily.

5.2 Motor Current Analysis

“The motor current analysis can be used to determine the condition of various parameters within the motor system. These parameters include: rotor cage defects, broken rotor bars, shorted stator turns, shorted laminations, high resistance connections, and motor bearing damage [19].” The motor acts a transducer, with the motor current signature changing with the various drive train anomalies.

The power spectrum of the currents should reveal a large spike at 60 Hz, and also other peaks associated with the harmonics of line current and harmonics of the actual operating frequency of the motor and movement of the rotor shaft. The predicted stator current frequencies (f_{bng}) caused by bearing vibrations that induce rotor movements are:

$$f_{bng} = |fe \pm m * fv_{bng}| \quad (5.1)$$

where:

$$m = 1, 2, 3...$$

fe = electrical supply frequency

fv_{bng} = characteristic vibration frequencies calculated by Equation (2.11).

The operating frequency of the motor is less than 60 Hz when the synchronous frequency is 30 Hz.

There are also motor current spectral frequencies generated by the difference in the frequency of rotation of the shaft and the motor's synchronous frequency. The equation below gives most of the harmonic frequencies seen in a current spectrum plot caused by this difference:

$$f_{slip} = fe * [1 + or - (k * slip * P)] \quad (5.2)$$

where:

f_e = electrical supply frequency

P = number of poles wound in stator = 4

$f_{synch} = 2 * f_e / P$

f_{shaft} = shaft turn frequency = 30

$slip = (f_{synch} - f_{shaft}) / f_{synch}$

$k = 1, 2, 3, \dots$ [47]

Thus, by using the equation above, we obtain more frequency peaks that should be seen in the current spectrum.

5.3 Vibration Analysis

The movement within the motor causes vibration itself. The purpose of a motor is to supply torque to an outside system. The observable moving part of the motor is the motor shaft. The motor shaft has the rotor mounted on it. The rotor and shaft rotate together within the motor. The rotor/shaft assembly is allowed to rotate within the motor by using bearings.

The bearings are solid mounted to the motor casing and to the rotor/shaft assembly, thereby allowing the rotor shaft to rotate within the motor but also to be a "solid" fixture of the motor system.

There are two main areas of the motor system that cause vibration. One is within the motor and related to the motion of the shaft, the other is the manner in which the motor is coupled to the outside system. But before we analyze these problems, we must first define our fundamental frequency of rotation.

We use the speed of rotation to determine the fundamental frequency of the motor system. The unit of frequency is a Hertz (cycles / second), with the fundamental frequency (f_{fund}) of the motor defined by:

$$\text{Fundamental frequency} = f_{fund} = RPM / 60 \quad (5.3)$$

where

RPM = revolutions per minute.

This frequency is used throughout vibrational analysis to determine the frequency components in the vibration spectra caused by various problems. We will now focus on the various causes of vibration associated with a rotating system.

Imbalance, bearings, and looseness could cause the motion within the motor. Imbalance is caused when the center of mass of the rotor/shaft system is not centered along the axis of rotation. The imbalance is characterized by vibrations with a frequency ($f_{imbalance}$) equal to that of the rotational speed of the shaft.

$$f_{imbalance} = I * f_{fund} \quad (5.4)$$

where

f_{fund} = fundamental frequency of rotation

The highest magnitude vibrations are at 90 degrees perpendicular to the shaft.

Bearing problems can be divided into six categories: fundamental train, ball spin, ball pass of the outer race, ball pass of the inner race, misalignment, and oil whirl (journal bearings).

Misalignment vibrations occur at 1 and 2 times the rotational speed. "Oil whirl happens at integer multiples of 38-49% of the fundamental rotational frequency. All other associated vibrations happen at characteristic sub-harmonic and integer multiples of the rotational frequencies and associated harmonics. These characteristic numbers are given by physical characteristics of the bearings themselves. Thus, the bearing vibrational frequencies ($f_{misalign}$, $f_{bearings}$, and $f_{oilwhirl}$) are defined by the fundamental frequency of rotation by:

$$f_{misalign} = N * f_{fund} \quad (5.5)$$

and

$$f_{bearings} = M * (\text{characteristic sub-harmonic or integer frequency}) \quad (5.6)$$

and also

$$f_{oilwhirl} = 0.38 \text{ to } 0.49 * M * f_{fund} \quad (5.7)$$

with

$$N = 1 \text{ and } 2$$

$$M = 1, 2, 3...$$

We must note that, in practical applications, the bearing frequencies usually show up in the higher frequencies of the vibrational spectrum. Higher frequencies meaning between 5-50 times the fundamental frequency [48].”

Looseness is caused in the motor or in the attached system because of parts that are not securely mounted. Vibration due to looseness occurs at 1, 2, 3... times rotational speed and the associated harmonics. Thus looseness frequencies (f_{loose}) is defined in terms of the fundamental frequency as

$$f_{loose} = N * f_{fund} \quad (5.8)$$

where

$$N = 1, 2, 3...$$

The two major problems with rotating systems that are linked (coupled) together are usually caused by imbalance or misalignment. Imbalance of this nature occurs when the shaft is coupled to a system whose center of mass does not rotate along the axis of rotation. The vibration is also found in the same frequency range as the internal imbalance defined above.

Misalignment occurs when the motor shaft axis is not exactly positioned along the axis of rotation for the coupled system. The two types of misalignment are called offset and angular. Angular misalignment has a strong vibration at the same frequency as that of the rotation of the shaft. Offset misalignment causes vibration at 1 and 2 times the rotational frequency. This is also the same as the internal misalignment found above.

Bearing damages resulting from shaft voltages and currents (i.e. fluting) can be seen in high frequency high-resolution vibration spectrum. This type of fault consists of high frequency modulated energy in the high frequency range between 2000 – 4000 Hz. The location of the mound of energy due to electrical discharge machining does not appear to be related to running speed or any other speed related variable [49]. Therefore, using a band-pass filter for the above-mentioned frequencies or using multi-resolution analysis using wavelets, general RMS trends can be calculated for the measured vibration

signals.

We must note, however, that there are problems that are seen in the motor current analyses that do not appear in the vibration data. “These problems are broken rotor bars, cracked end rings, high resistance squirrel cage joints, casting porosity in the die cast joints, static and dynamic air gap irregularities, and unbalanced magnetic pull [21].” Therefore, since there is extra information contained within the current signature, there should not be an exact match between current and vibration.

5.4 Torque Calculation

Torque can be calculated using a dynamometer. The dynamometer applies a specific load to the shaft of the motor by coupling the shaft of the dynamometer to that of the motor. The dynamometer's stator produces a magnetic field that reacts with the permanent magnet mounted on the dynamometer's rotor. The resistance produced is in opposite direction to that of the induction motor's rotation. A load cell mounted at a specific distance on the dynamometer from the axis of rotation measures the torque. The torque is given by:

$$\text{Torque} = \text{force} * \text{distance} \quad (5.9)$$

The torque may also be measured in a variety of other ways. For example J. S. Hsu has proposed a methodology in which air-gap torque can be observed by measuring instantaneous voltage and currents [50]. However, this method requires the knowledge of line-to-line resistance values as an input to the calculations. This method has not been tested in industrial applications yet to detect cracked rotor bars or shorted stator coils as claimed by the author. Moreover presumption of resistance value, which is directly related with the fault itself, can be erroneous in calculations of derived measurements. Torque may be used to calculate the power input to the dynamometer for the setup mentioned above. The power input into the dynamometer can be assumed to be the power output of the motor at the shaft. The power output at the shaft can be used by a variety of methods, one such method involves the calculation of motor efficiency.

5.5 Power Calculation

Power analysis, as will be discussed in this section, refers to either instantaneous power input into the stator windings (power input to the motor) or the power input into the dynamometer (power output of the motor).

5.5.1 Instantaneous Power

Instantaneous power is defined as the instantaneous voltage between any two phases multiplied by the instantaneous current drawn by either phase. The formulation

has been given in Equation (2.4). “If mechanical abnormality develops in a drive system, harmonic torques are generated in the motor, accompanied by speed oscillation, and modulation of the stator current. Frequency components characteristic for the type of abnormality appear in the power spectrum of the stator current. The location of these components allows identifying the abnormality” [29]. The motor power analysis is very similar to the motor current analysis.

5.5.2 Torque Derived Power

The power input into the dynamometer can be determined by using the RPM, and the torque. The equation relating the power to RPM and torque is given by:

$$Power = conversion\ constant * RPM * torque \quad (5.10)$$

The torque and RPM are time average readings that are multiplied together, which has been used to derive the power output from the motor.

5.6 Thermal Analysis

Thermal analysis is usually performed by two methods; measurement by direct contact or by thermal imaging. Direct contact gives a more precise reading as far as positioning is concerned. The detection device may be positioned almost anywhere

inside or outside of the motor. Thermal imaging gives an overview of the temperatures on the outside of the motor. But both can give valuable results.

“Excessive and prolonged heat is the main factor responsible for shortening the life expectancy of motors. The two components most affected by excessive heat are the insulation system and bearing. Trending temperature measurements is a simple and fast method for estimating motor overheating. Common methods of overheating are overloading, bearing seizure, misalignment, ventilation, single phasing, high ambient temperatures, excessive duty cycles, and power supply variations. Motor temperature rise is a function of bearing friction, windage, core loss, copper losses (referred to as I^2R losses) and stray losses [16].”

5.7 Magnetic Flux Analysis

“Any small unbalance in the magnetic or electric circuit of motors is reflected in the axially and radially transmitted fluxes. A simple sensor for detecting these signals is a flux coil. Many of the frequencies are related to the speed of the motor. It also appears that flux measurement can supply a preliminary indication of rotor bar defects, electrical faults, unbalanced voltage, stator faults ... shaft currents, and others [51].” The coils are placed on the axial outboard end of the motor and on top of the motor. The magnetic flux is thought to detect many of the same problems that motor current analysis yields.

Detection of a winding insulation fault requires trending of the following frequencies in the magnetic flux spectrum:

- Line Frequency – 2 * Running Speed
- Line Frequency – 1 * Running Speed
- Line Frequency
- Line Frequency + 1 * Running Speed
- Line Frequency + 2 * Running Speed
- Line Frequency + 3 * Running Speed

5.8 Efficiency Calculation

The efficiency of a motor is given by the power output divided by the power input. Thus, the equation is:

$$\text{Efficiency} = \text{Power out} / \text{Power in} \quad (5.11)$$

This is a dimensionless quantity ranging from 0 to 1 or 0% to 100%. This measurement tells directly how well a motor is functioning.

5.9 Component Sequence Analysis

The following transformations are used in order to calculate the zero-, positive- and negative-sequence components of each instantaneous current and voltage measurement as indicated in Figure 2.2:

$$\begin{bmatrix} I^0 \\ I_x^+ \\ I_x^- \\ I_y^+ \\ I_y^- \\ I_z^+ \\ I_z^- \end{bmatrix} = \frac{1}{3} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & \Psi & \Psi^2 \\ 1 & \Psi^2 & \Psi \\ \Psi & 1 & \Psi^2 \\ \Psi^2 & 1 & \Psi \\ \Psi & \Psi^2 & 1 \\ \Psi^2 & \Psi & 1 \end{bmatrix} \begin{bmatrix} I_x \\ I_y \\ I_z \end{bmatrix} \quad (5.12)$$

where

$$\Psi = e^{j\left(\frac{2}{3}\pi\right)} = -0.5 + j0.866 \quad \text{and} \quad \Psi^2 = e^{j\left(\frac{4}{3}\pi\right)} = -0.5 - j0.866 \quad (5.13)$$

The superscript 0 denotes the zero, + denotes positive and – denotes negative-sequence-component of the corresponding x, y, z phase currents. Component analysis for voltage has the same format as Equation (5.13):

$$\begin{bmatrix} V^0 \\ V_x^+ \\ V_x^- \\ V_y^+ \\ V_y^- \\ V_z^+ \\ V_z^- \end{bmatrix} = \frac{1}{3} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & \Psi & \Psi^2 \\ 1 & \Psi^2 & \Psi \\ \Psi & 1 & \Psi^2 \\ \Psi^2 & 1 & \Psi \\ \Psi & \Psi^2 & 1 \\ \Psi^2 & \Psi & 1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \quad (5.14)$$

where the equivalent phase-to-neutral voltages are expressed in terms of phase-to-phase voltages as:

$$V_x = \frac{V_{xy}}{\sqrt{3}}, V_y = \frac{V_{yz}}{\sqrt{3}} \text{ and } V_z = \frac{V_{zx}}{\sqrt{3}} \quad (5.15)$$

Component analysis for impedance is given by

$$\begin{bmatrix} Z^0 \\ Z_x^+ \\ Z_x^- \\ Z_y^+ \\ Z_y^- \\ Z_z^+ \\ Z_z^- \end{bmatrix} = \begin{bmatrix} V^0 / I^0 \\ V_x^+ / I_x^+ \\ V_x^- / I_x^- \\ V_y^+ / I_y^+ \\ V_y^- / I_y^- \\ V_z^+ / I_z^+ \\ V_z^- / I_z^- \end{bmatrix} \quad (5.16)$$

It is seen that, except for the zero-sequence component all the other components are complex. So, the RMS magnitude and angle of the component sequences are trended

for current, voltage and impedance. Using this method, the amount of phase imbalance can be accurately determined independent of motor load and source imbalance by using negative-sequence component impedance. This sequence impedance index or impedance due to phase imbalance is a good indicator of the turn-to-turn insulation condition since it is independent of slip. However, in order to obtain a signature for incipient fault detection, zero-sequence component impedance gives a better indication than any of the other transformation given in Equations 5.12 – 5.16. The reason for this is, if all of the phases are balanced, hence the winding is in good condition, the zero-sequence component of impedance should be close to zero, whereas positive and negative sequence components yield transformations of much larger amplitude. Therefore, if a very small change in the condition is to be observed, it is better to use the zero-sequence component.

5.10 Remaining Life Estimation and Fault Classification

The techniques explained in the previous sections of this Chapter are used to constitute the “Sensitivity Matrix.” This matrix is a cross-reference between the signatures and the failure modes. The signatures indicated in this matrix are trended using accelerated aging data. Upper and lower limits of the measured values are trended together with an established mean trend. This trend will be the basis for estimating the remaining life of the motor. The IMMS measures the specified signature and uses the appropriate trend graph to find the corresponding time index. Based on the ambient

operating temperature, the remaining life of the motor undergoing a thermal, corrosional and / or electrical discharge failure, may be estimated using the following procedure:

1. Normalization of the temperature measurement with respect to the ambient temperature (and power). This is accomplished by using

$$T_{Normalized} = (T_{raw} - T_{Ambient})/f(Load) + \Delta T_{Steady-State} \quad (5.17)$$

where

$T_{Normalized}$ = Normalized temperature in °C.

T_{raw} = Raw temperature in °C.

$T_{Ambient}$ = Ambient room temperature in °C.

$\Delta T_{Steady-State}$ = Correction factor for the semi-steady-state operation.

$f(Load)$ = Load function which normalizes temperature to 100% load.

This load function can be shown in the following example: Suppose the raw temperature measurement of the winding insulation temperature at process end is 45 °C at 50% load and the ambient room temperature measurement is 20 °C. Now, the first normalization is performed with respect to the ambient room temperature by subtracting the ambient temperature from the winding insulation temperature and calling it $\delta T_{50\%}$. Now, suppose that there exists a function, which correlates this differential temperature (δT) to percent load as:

$$\delta T(\text{load}) = 0.0036 * (\text{load})^2 - 0.1266 * (\text{load}) + \text{constant}$$

By putting the values above, we find:

$$(45-25) = 0.0036 * (50)^2 - 0.1266 * (50) + \text{constant}$$

Hence, $\text{constant} = 22.33 \text{ }^\circ\text{C}$.

Thereafter, the same equation is used to calculate the differential temperature at 100% load, with the constant known this time. This variable is called $\delta T_{100\%}$.

Now, the function $f(\text{Load})$ can be defined as

$$f(\text{Load}) = \delta T_{50\%} / \delta T_{100\%}$$

If the relationship between the temperature and power level would have been linear, Equation (5.17) would be equal to Equation (1.3) and $f(\text{Load})$ would be 0.5. However, since the relationship between power level and temperature is not linear, as is shown in the next chapter, $f(\text{Load})$ is computed to be 0.52.

Using Equation (5.17), with a $\Delta T_{\text{Steady-State}} = 2.33 \text{ }^\circ\text{C}$, we find $T_{\text{Normalized}} = 48 \text{ }^\circ\text{C}$.

2. Based on a sampling time Δt a trend is established. In the accelerated aging studies, each aging cycle corresponded to 48 hours of motor operation (for motor winding insulation experiment), while in normal operating conditions this would correspond to 64 days of motor operation. This factor is derived from the Arrhenius Equation as depicted in Figure 3.5. Hence, the relationship of converting the accelerated aging data into normal operating data, is to multiply them with a scaling factor in time domain: 64 days for winding insulation aging experiments and 192 days for bearing thermal and electrical aging experiments for each aging cycle. In a field application, any Δt can be chosen any value, as trends are formed in calculating the remaining lifetime of the motor or its components.
3. Accelerated aging studies provided “alarm levels” of failure. These alarm levels are tabulated in the next chapter for each feature in the ‘Sensitivity Matrix.’
4. Based on the “alarm level” and the trend the remaining life of the motor can be estimated by extrapolation as shown in Figure 5.1.

When more than one degradation trends are used, the worst possible is assumed to estimate the overall remaining life of the motor or its components.

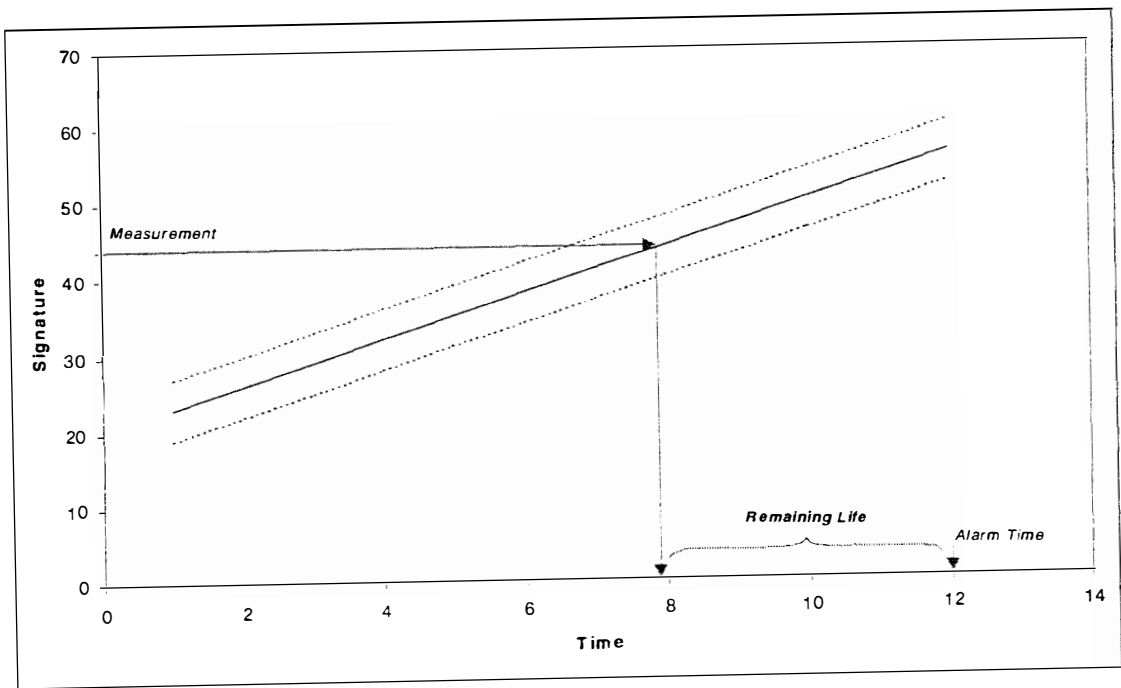


Figure 5.1: Calculation of remaining life based on established signature trends.

Another approach is to use back-propagation artificial neural networks as shown in Figure 5.2. For each signature, that has the ability to detect a specific failure mode, an artificial neural network was constructed. A fast back-propagation neural network is used to develop these networks. The input of these networks, are mainly two parameters:

1. Signature amplitude,
2. Trend of the signature amplitude.

Each of the inputs was normalized in the range of [0,1] with respect to the maximum possible values, derived from the accelerated aging. Each processing elements of the back-propagation neural network has the sigmoid transfer function as:

$$Output_i = \frac{1}{1 + e^{-\sum_i weight_i * input_i}} \quad (5.18)$$

The output of each artificial neural network (ANN) gives an estimation of the remaining lifetime of the motor or its components. The training of these artificial neural networks were accomplished by an artificial training data set, which covers the possible input space homogeneously.

For each component failure or motor failure mode, artificial neural networks are developed to estimate the remaining lifetime of the motor or its components. Figure 5.2

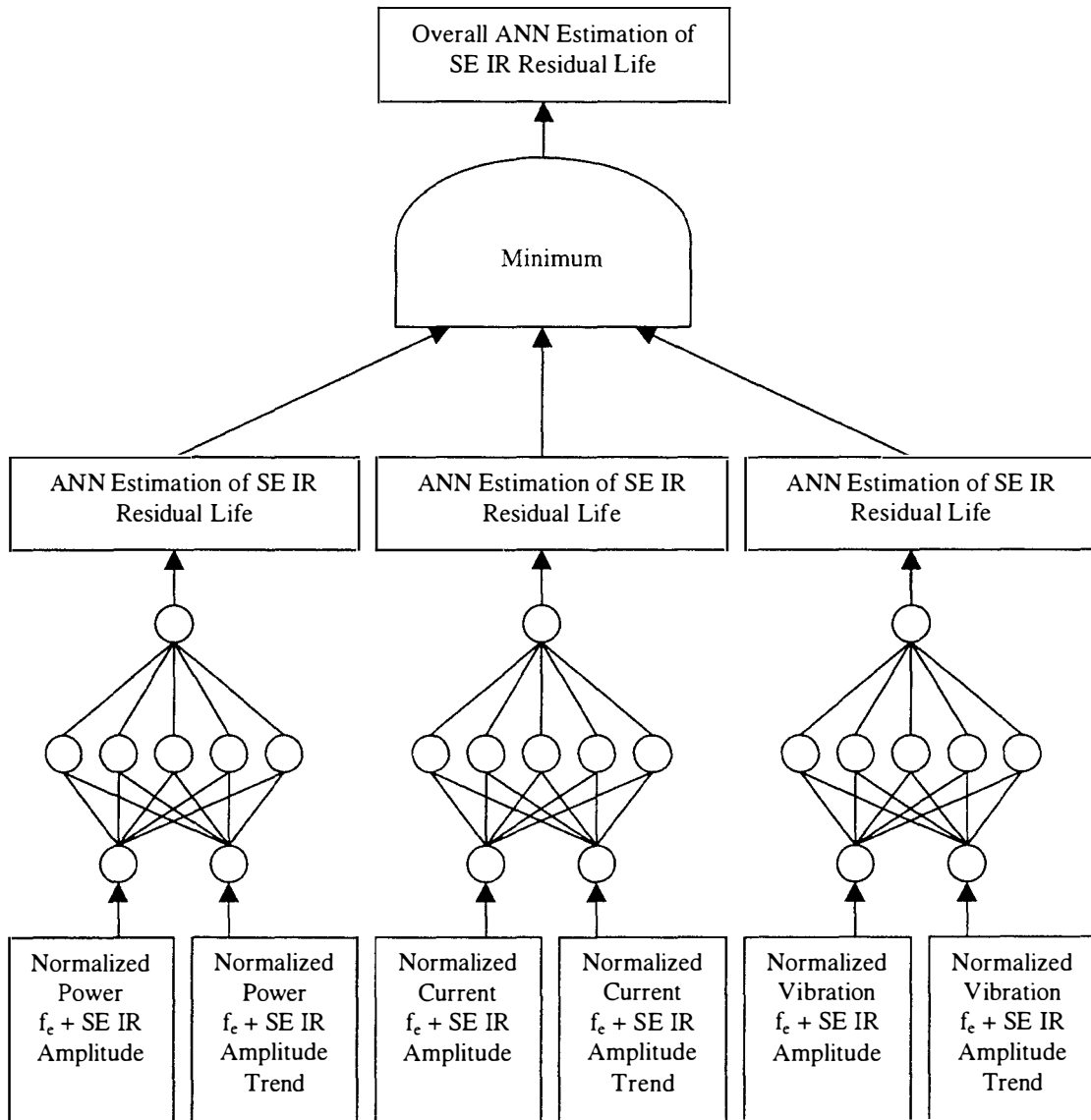


Figure 5.2: Structure of the ANN to estimate the residual life of the short-end bearing inner ring.

shows such an artificial neural network, which predicts the remaining lifetime of the motor short end bearing inner ring remaining lifetime based on three sensitive signatures:

1. Motor power spectral amplitude at $f_e + \text{SE IR}$ frequency
2. Motor current spectral amplitude at $f_e + \text{SE IR}$ frequency
3. Motor vibration spectral amplitude at $f_e + \text{SE IR}$ frequency

For each of these signatures, ANN's were developed, each of them estimating the remaining lifetime of the motor short end bearing inner ring. The overall remaining lifetime was calculated to be the minimum of these estimations.

A fuzzy – logic inference engine is implemented in order to classify motor faults [52 - 54]. By this approach, a symbol-level data fusion is established, combining several measurements and features into linguistic variables. Table 5.1 gives the fuzzy logic rule base, which is used for fault detection and classification. This rule base is derived from the 'Sensitivity Matrix,' as a result of the accelerated aging studies. As it can be seen from the rule base the failure of the motor or its components are based on the effect, not on the cause. This does not mean however, fault detection based on driving forces cannot be incorporated into this database: For example the measurement of ambient temperature may be used to track abnormal operating conditions and alarm the user in the form:

Table 5.1: Fuzzy Logic Rule Base

```
if Power_fe+IRSE is Decreasing
  and Motor_current_fe+IRSE is Decreasing
  and Motor_Vibration_fe+IRSE is Increasing
  then SE_B_IR_Damage is True
if Power_fe+ORSE is Decreasing
  and Motor_current_fe+ORSE is Decreasing
  and Motor_Vibration_fe+ORSE is Increasing
  then SE_B_OR_Damage is True
if Power_fe+BSE is Decreasing
  and Motor_current_fe+BSE is Decreasing
  and Motor_Vibration_fe+BSE is Increasing
  then SE_B_BSE_Damage is True
if Power_fe+IRPE is Decreasing
  and Motor_current_fe+IRPE is Decreasing
  and Motor_Vibration_fe+IRPE is Increasing
  then PE_B_IR_Damage is True
if Power_fe+ORPE is Decreasing
  and Motor_current_fe+ORPE is Decreasing
  and Motor_Vibration_fe+ORPE is Increasing
  then PE_B_OR_Damage is True
if Power_fe+BPE is Decreasing
  and Motor_current_fe+BPE is Decreasing
  and Motor_Vibration_fe+BPE is Increasing
  then PE_B_BPE_Damage is True

if Motor_vibration is Highly_Increasing
  and Motor_SE_BS_Temperature is Slowly_Increasing
  and Power_vibration_coherence_line_freq is Decreasing
  and Power_vibration_coherence_slipxnumberofpoles/2 is Decreasing
  and Motor_vibration_MRA_ratio is Highly_Increasing
  then Motor_Electrical_Discharge is True
if Motor_vibration is Highly_Increasing
  and Motor_PE_BS_Temperature is Slowly_Increasing
  and Power_vibration_coherence_line_freq is Decreasing
  and Power_vibration_coherence_slipxnumberofpoles/2 is Decreasing
  and Motor_vibration_MRA_ratio is Highly_Increasing
  then Motor_Electrical_Discharge is True

if Motor_vibration is Slowly_Increasing
  and Motor_SE_BS_Temperature is Highly_Increasing
  and Power_vibration_coherence_line_freq is Decreasing
  and Power_vibration_coherence_slipxnumberofpoles/2 is Decreasing
  and Motor_vibration_MRA_ratio is Highly_Increasing
  then Motor_Thermal_Damage is True
if Motor_vibration is Slowly_Increasing
  and Motor_PE_BS_Temperature is Highly_Increasing
  and Power_vibration_coherence_line_freq is Decreasing
  and Power_vibration_coherence_slipxnumberofpoles/2 is Decreasing
  and Motor_vibration_MRA_ratio is Highly_Increasing
  then Motor_Thermal_Damage is True

if Axial_Magnetic_flux_RPM is Increasing
  and Radial_Magnetic_Flux_sideband_freq is Increasing
  and Radial_Magnetic_Flux_line_freq is Increasing
  then Winding_Insulation_Damage is True
if Motor_Zero_Component_Impedance_line_freq is Increasing
  then Winding_Insulation_Damage is True
```

If *Ambient_Temperature* is *High* then *Abnormal_Operation_Temperature* is *True*

or if there is a current imbalance, wheter it is a result of improper connection or a result of failure, the following rule to alarm the user can be also added:

If *Negative_Sequence_Component_of_Impedance* is *High*
then *Unbalanced_Currents* is *True*

Membership functions for the fuzzy variables used in the fuzzy rule base were derived as following:

1. The corresponding signatures were first normalized, with respect to load. If it is a temperature measurement, as indicated in Equation (5.17), ambient temperature was also used in the normalization.
2. Based on accelerated aging data, trends of these normalized data were established using a linear or exponential fit by least-squares fitting technique.
3. The slopes of these trends are thereafter used in the fuzzy domain of the membership functions to establish the corresponding membership functions.

Chapter 6

RESULTS OF INDUCTION MOTOR AGING ANALYSIS AND FAULT DIAGNOSIS

6.1 Introduction

The analysis of the data includes establishment of a database in which statistical as well as derived values of measurements were recorded. Microsoft Access was used as a standard relational database management system. Use of this standard tool enables the user to extract and add additional features in the future. This database contains the RMS as well as the mean values of the measurements, active power and apparent power, power factor and many other features. Table 6.1 lists the motors used in the three aging experiments which are included in this database.

Spectral analysis of the motor current and vibration signals was performed in order to verify the data collection technique and its quality. As indicated before, this kind of study revealed aliasing in the low frequency measurements (up to 200 Hz without filtering), so that the low frequency data collection was increased to 666.67 Hz with a fifth order Butterworth filter.

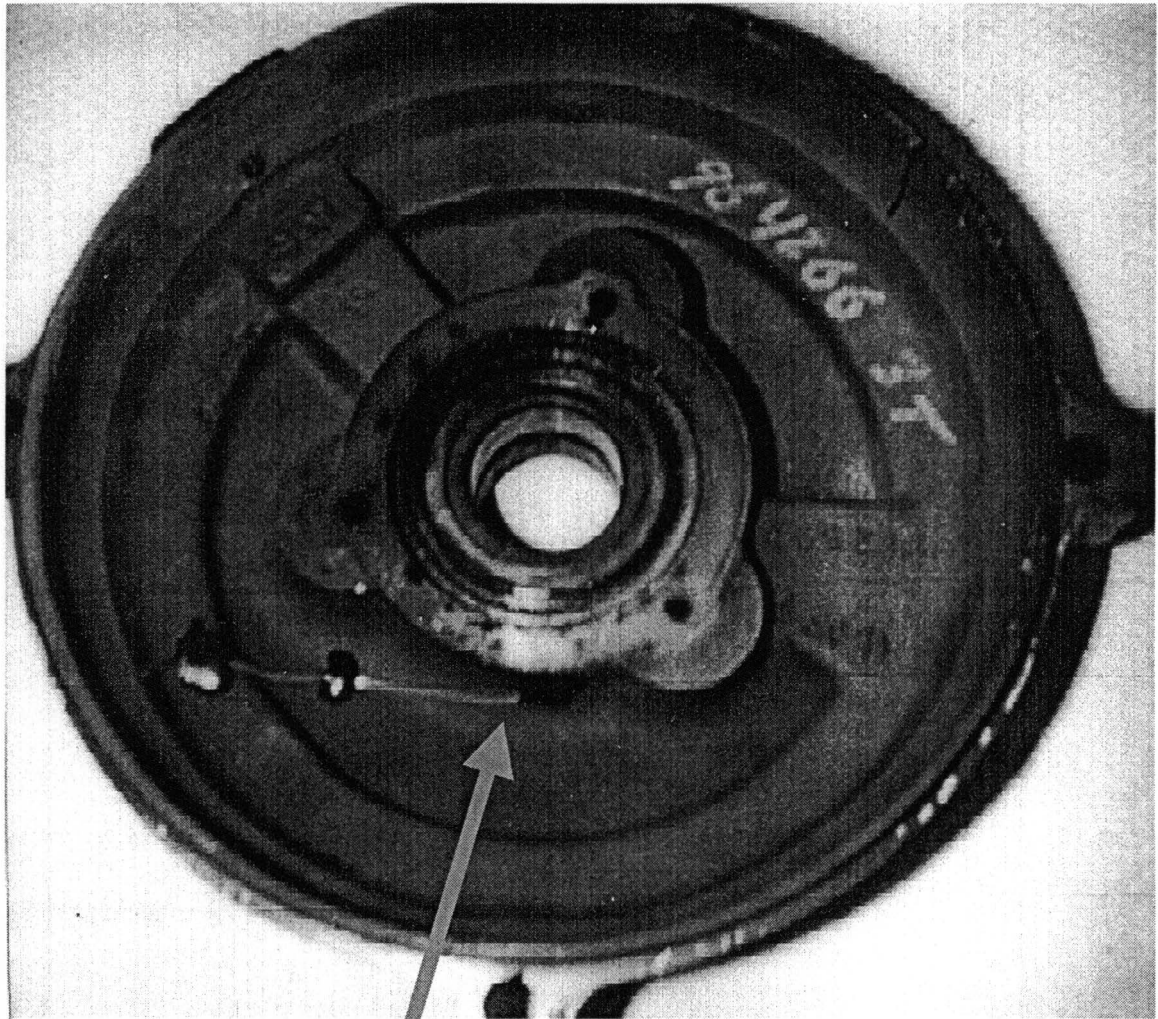
Table 6.1: Listing of the electrical motors for accelerated aging tests

Type of Accelerated Aging Test	Motor #
Thermal and electrical aging of stator winding insulation	1, 2, 3
Thermal aging of bearings	6, 7, 8
Fluting	11, 12, 13

6.2 Failure of Electrical Motors

The electrical motors were aged until it was determined that they were no longer operable. The assessment of the failed condition was made as follows:

- If the motor winding insulation was subjected to accelerated aging, a short would occur, therefore causing an electrical short. This left the electrical motor inoperable.
- If the electrical motor was subjected to fluting, then the vibration levels of the motor would increase. When the vibration levels were no more tolerable, which was merely determined by the acoustic noise generated, then the motor was marked as failed.
- If the motor was subject to thermal aging, then the motor was marked as failed, whenever the rotor locked after the aging cycle.
- The damage to the winding insulation is very obvious and a close examination revealed no damage to the thermocouples, which could have caused a false measurement (Figure 6.1). A visual inspection of a failed motor, namely motor #11 which was subjected to electrical discharges, was also performed. Visual



J Type Thermocouple

Figure 6.1: Measurement of bearing temperature.

inspection of the bearing elements revealed pitting and frosting damage, caused by electrical discharge as shown in Figure 6.2 - Figure 6.5.

6.3 Analyses of Raw Data

The analyses of the temperature and vibration data, which are shown in Figure 6.6 - Figure 6.25, indicate the following features in the measured signatures.

- Bearing surface temperature increases as a function of aging cycle for both bearing thermal aging and bearing fluting cases. This observation is shown in Figure 6.8, Figure 6.9, Figure 6.12 and Figure 6.13
- RMS vibration signatures show trends for both bearing aging and bearing fluting cases. However, the RMS vibration signatures for bearing fluting as shown in Figure 6.10 and Figure 6.11 are greater than those for bearing thermal aging as shown in Figure 6.6 and Figure 6.7. The difference is the effect of fluting.
- Zero-sequence component of impedance shows signs of degradation. Figure 6.14 shows the time series, the spectral and the cepstral plots of a motor under winding insulation accelerated aging experiment. As seen, as the winding insulation degrades, the electrical resistance of the winding also changes slightly, which is reflected on the electrical line frequency in spectral plots and multipels of the line

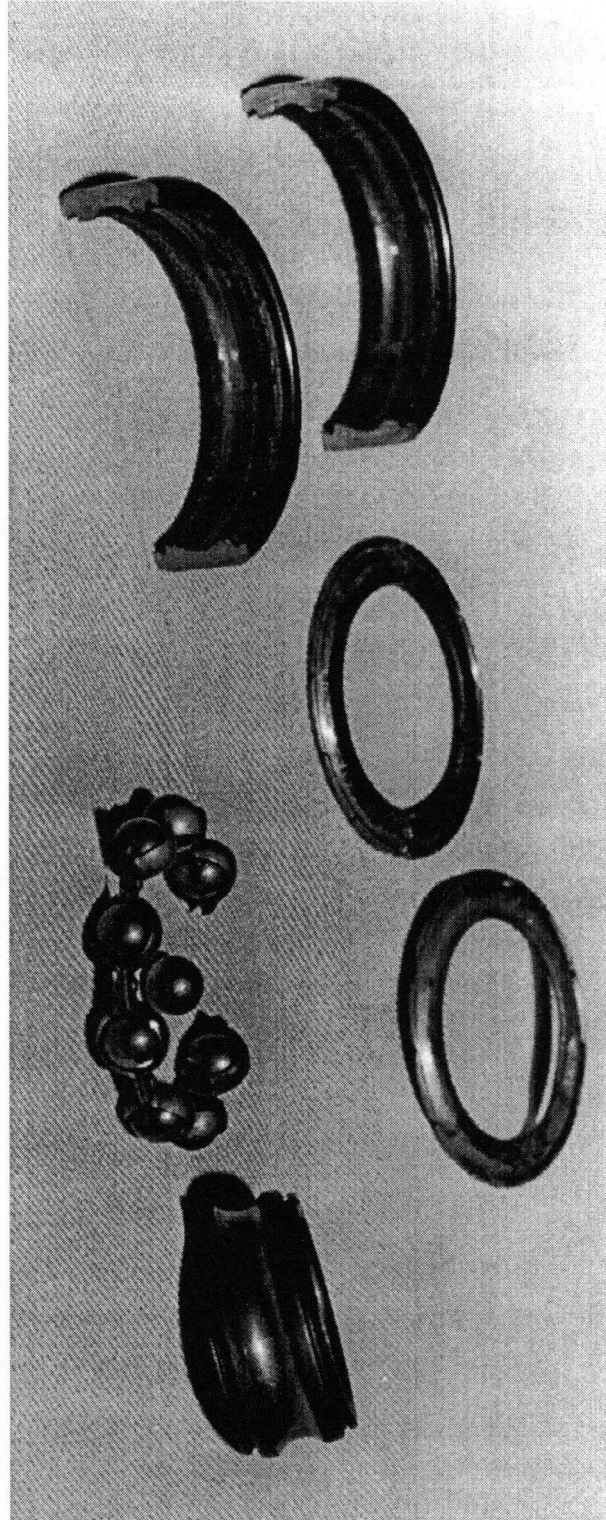


Figure 6.2: SKF6205 bearing disassembled.

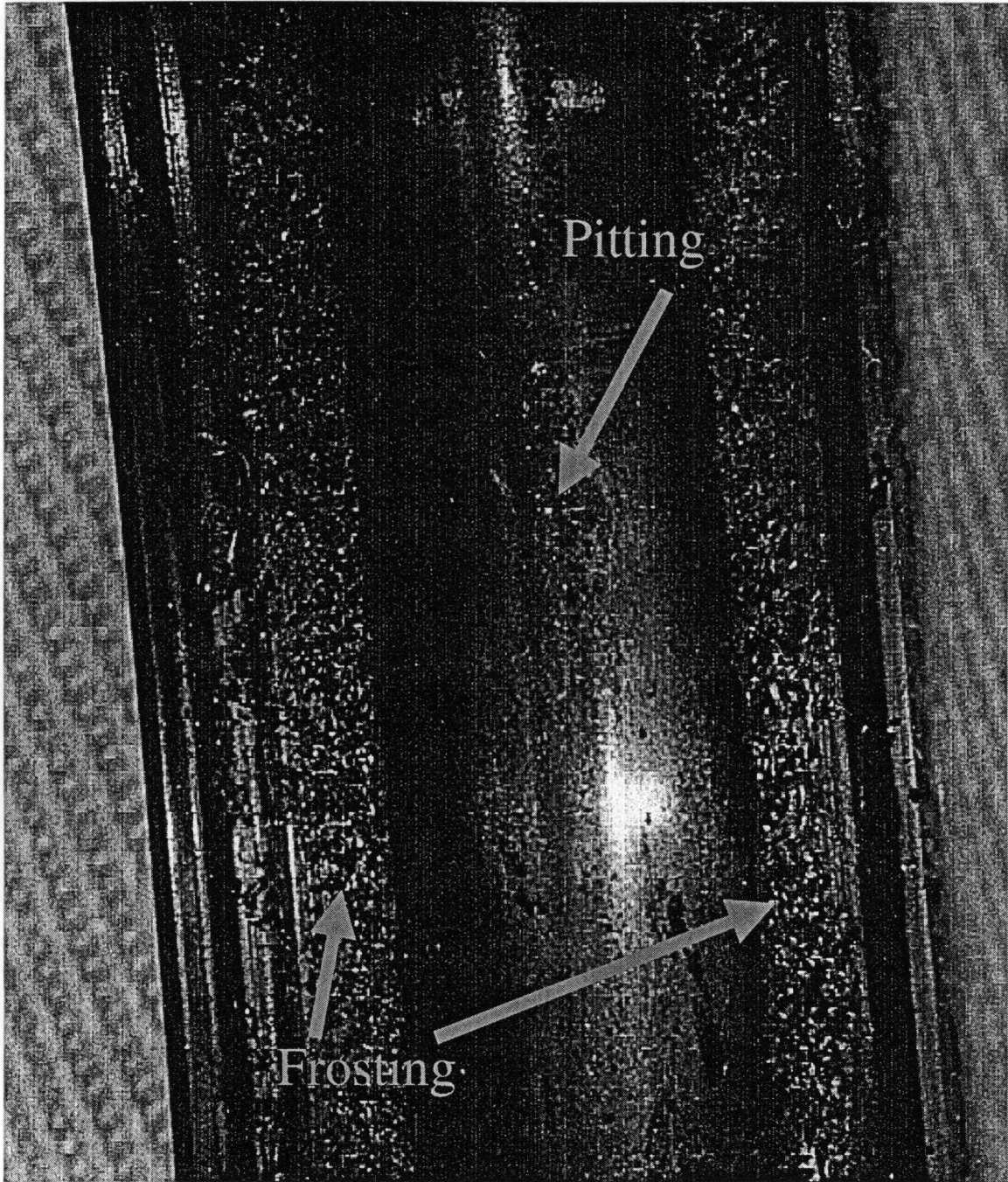


Figure 6.3: Bearing outer race of motor #11.

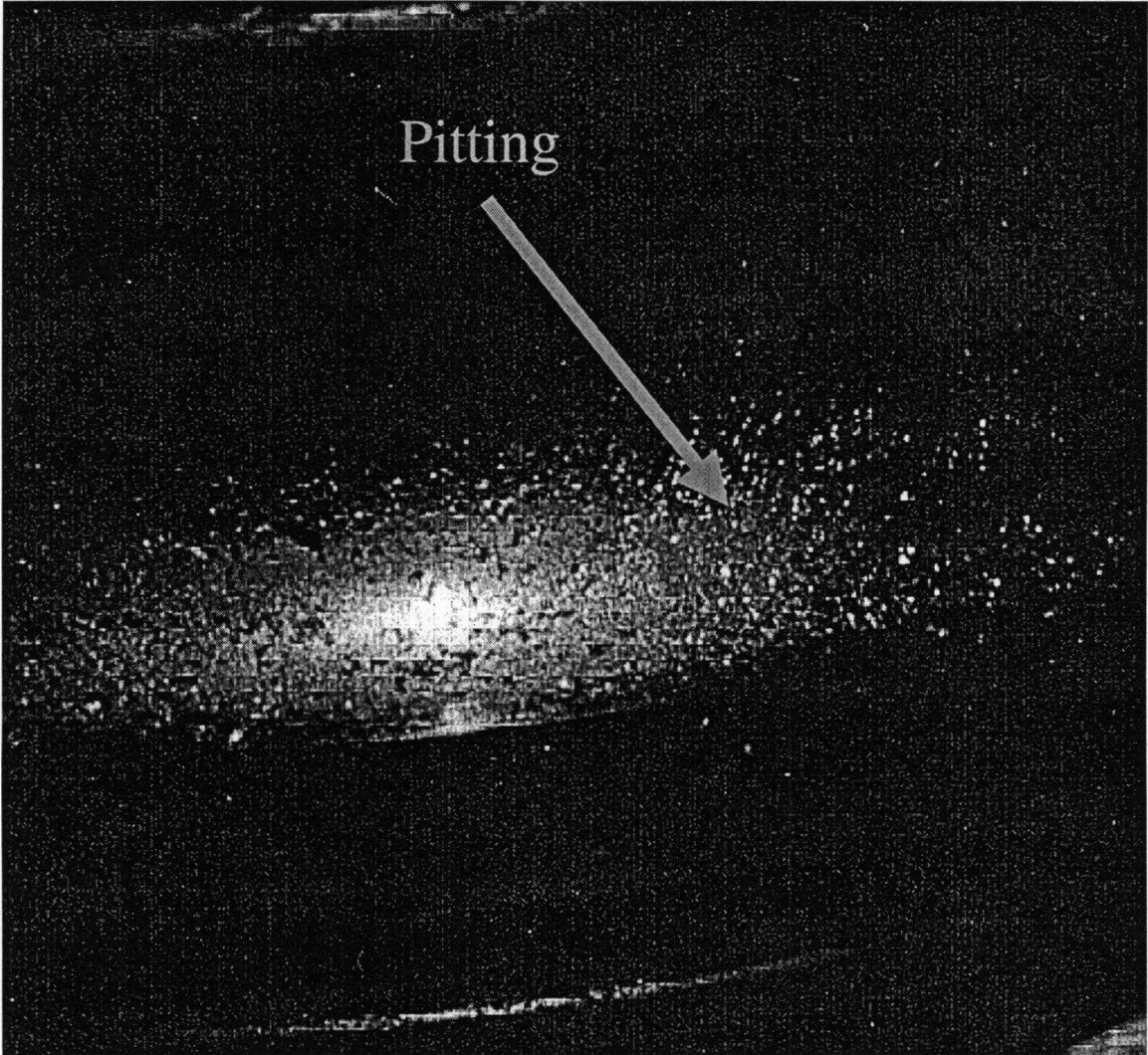


Figure 6.4: Bearing inner race of motor #11.

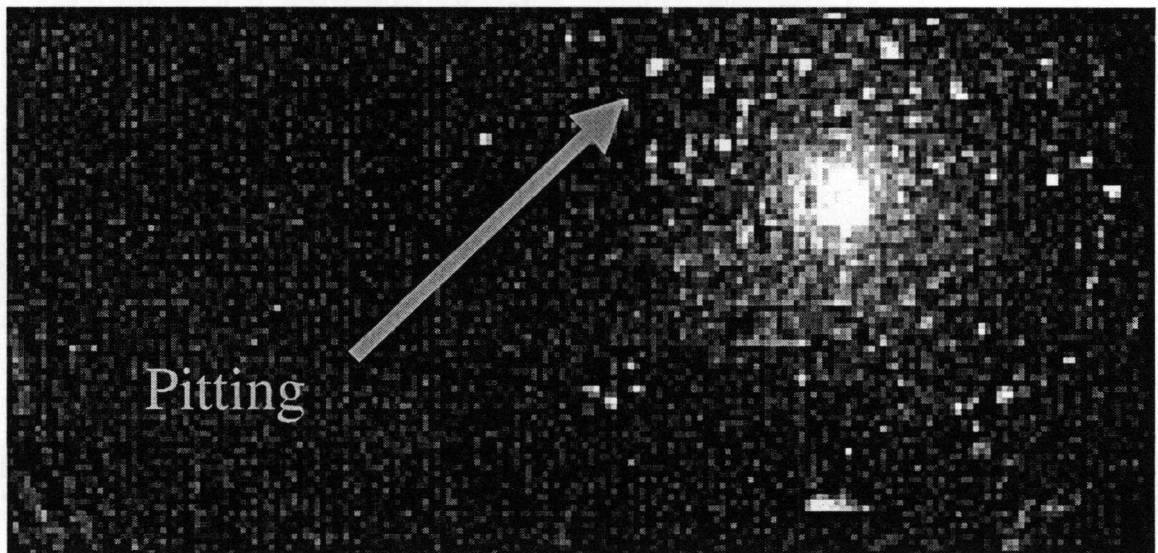
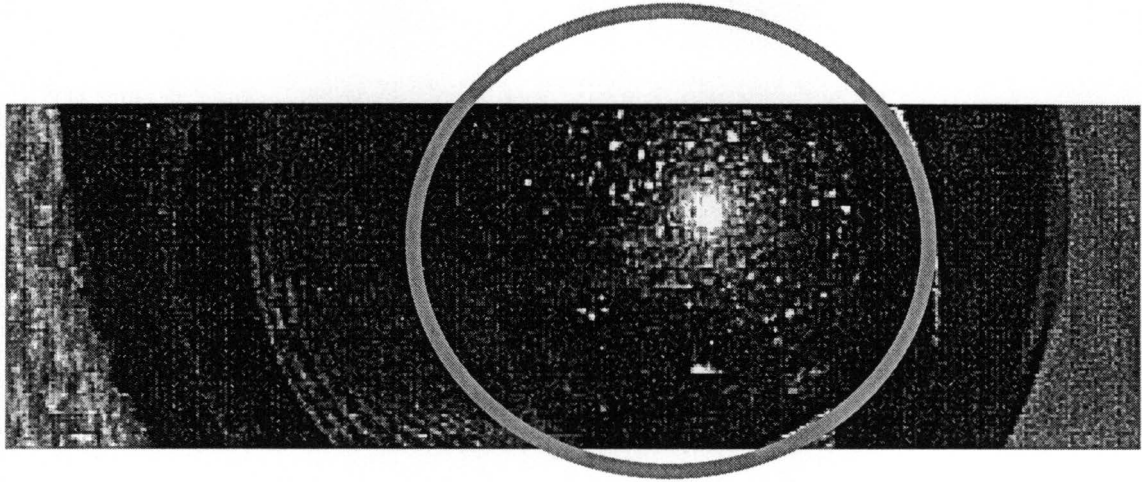


Figure 6.5: Bearing ball of motor #11.

BEARING AGING PE 2:00 ACCELEROMETER RMS TRENDS

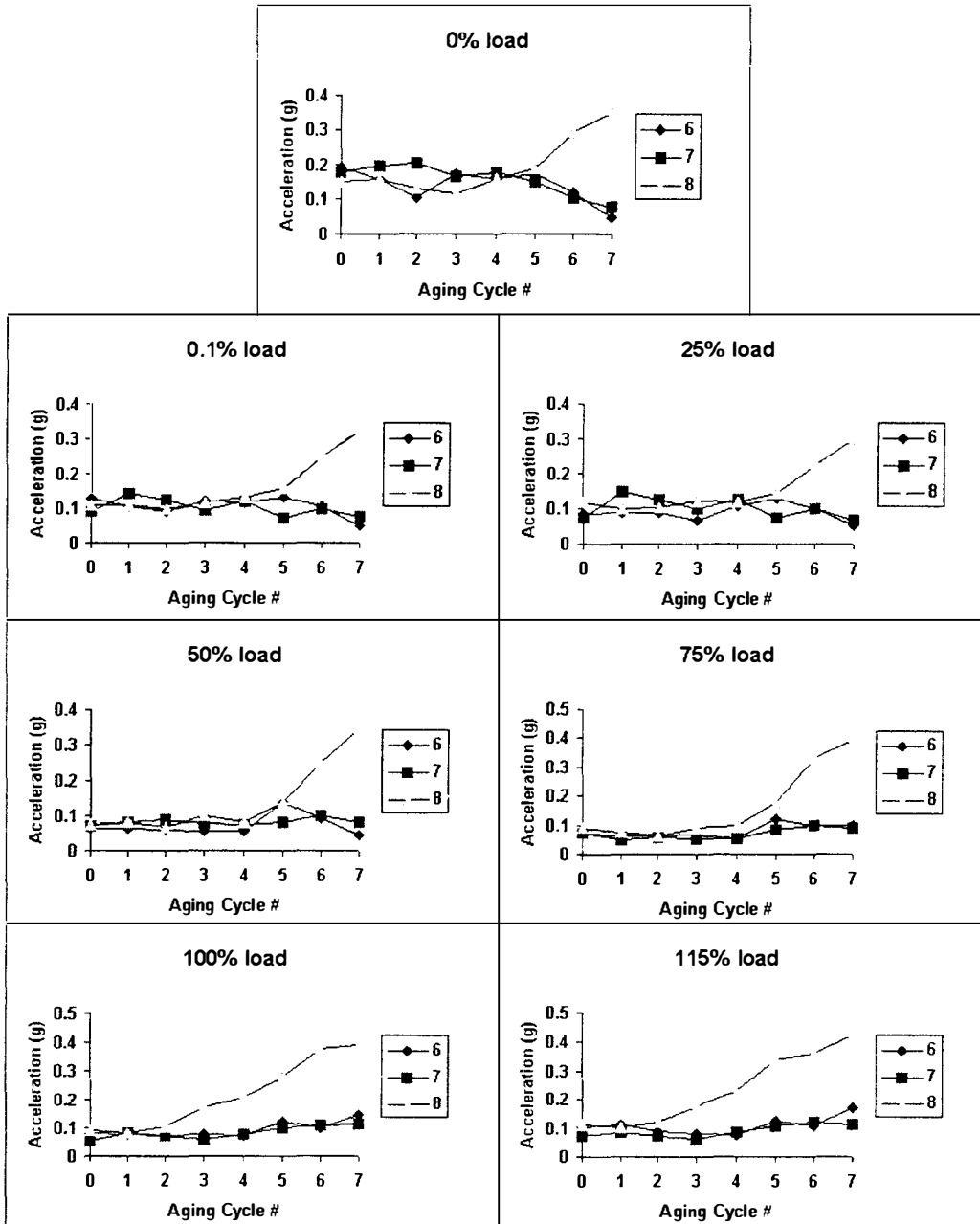


Figure 6.6: Bearing aging test process end 2:00 accelerometer RMS trends.

BEARING AGING SE V ACCELEROMETER RMS TRENDS

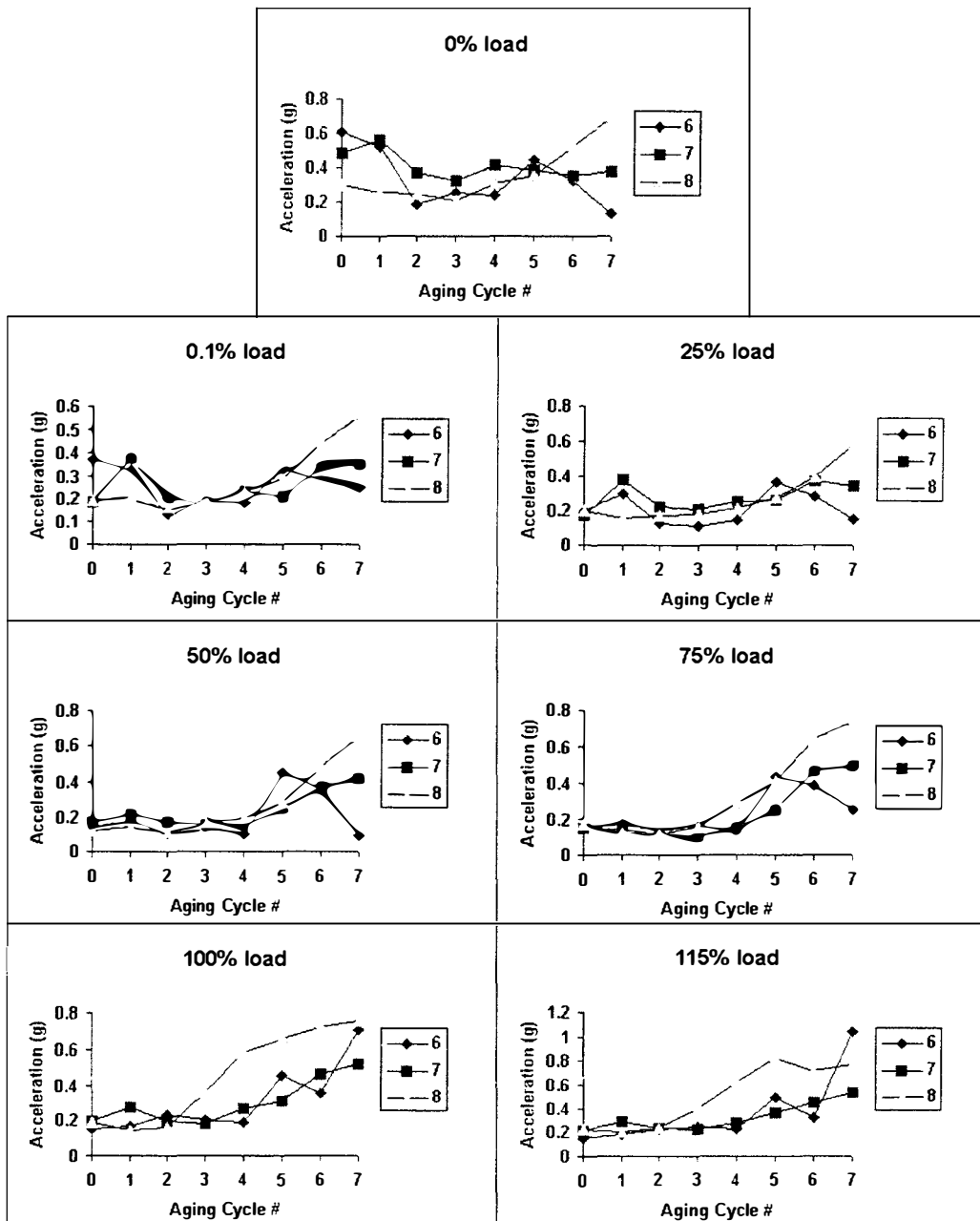


Figure 6.7: Bearing aging test short end vertical accelerometer RMS trends.

BEARING AGING BEARING SURFACE PE TEMPERATURE TRENDS

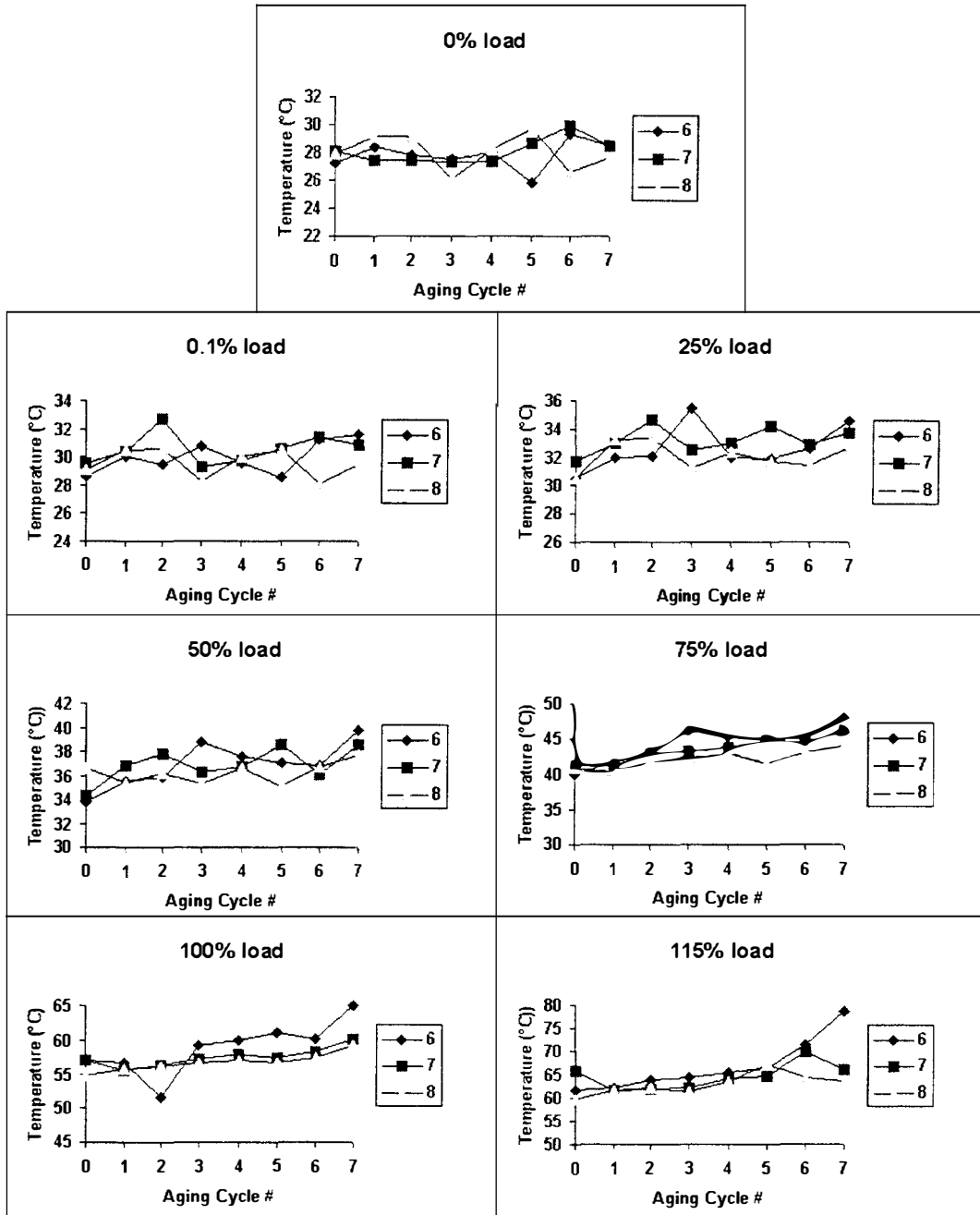


Figure 6.8: Bearing aging test process end bearing surface temperature trends.

BEARING AGING BEARING SURFACE SE TEMPERATURE TRENDS

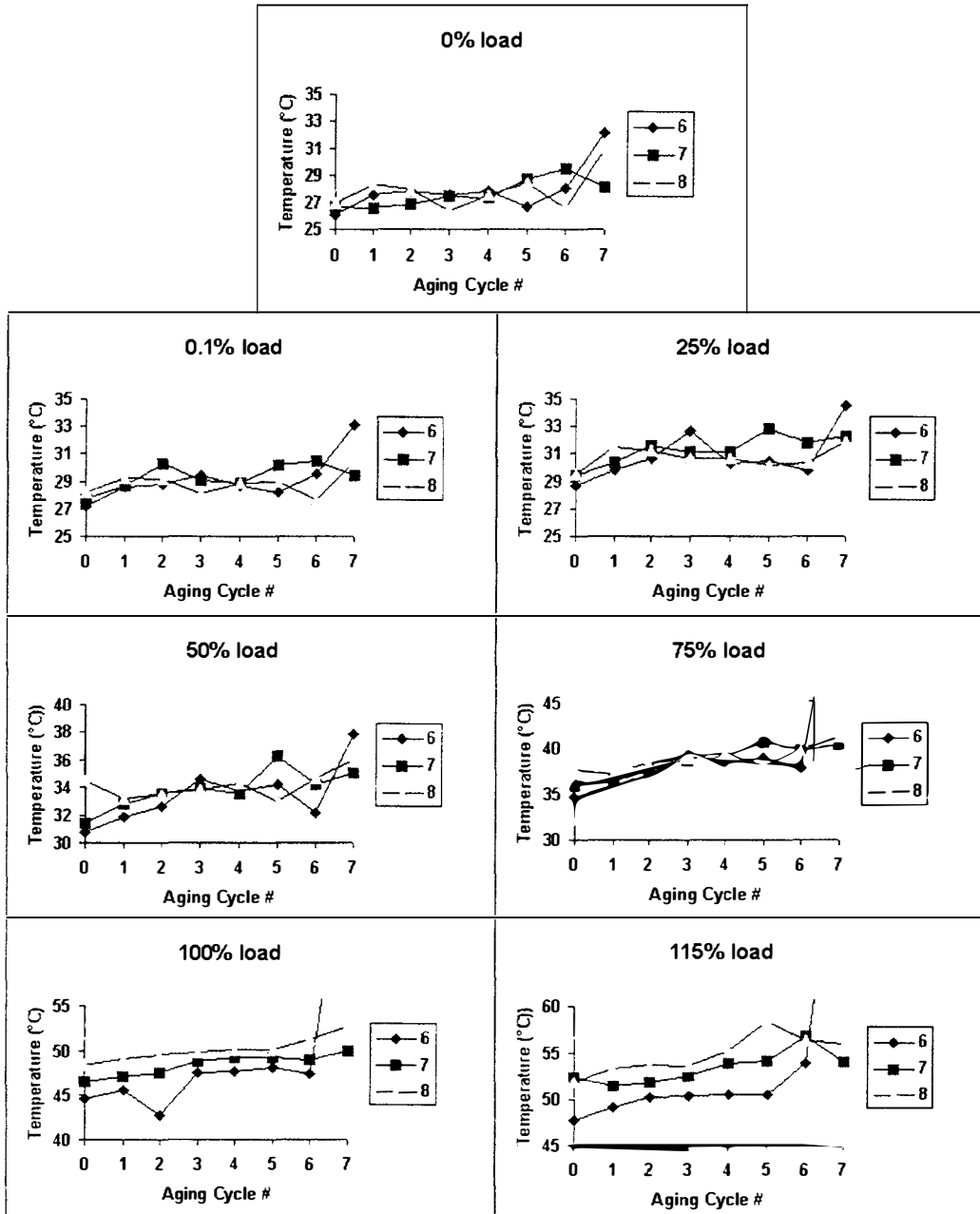


Figure 6.9: Bearing aging test short end bearing surface temperature trends.

BEARING FLUTING AND AGING PE 2:00 ACCELEROMETER RMS TRENDS

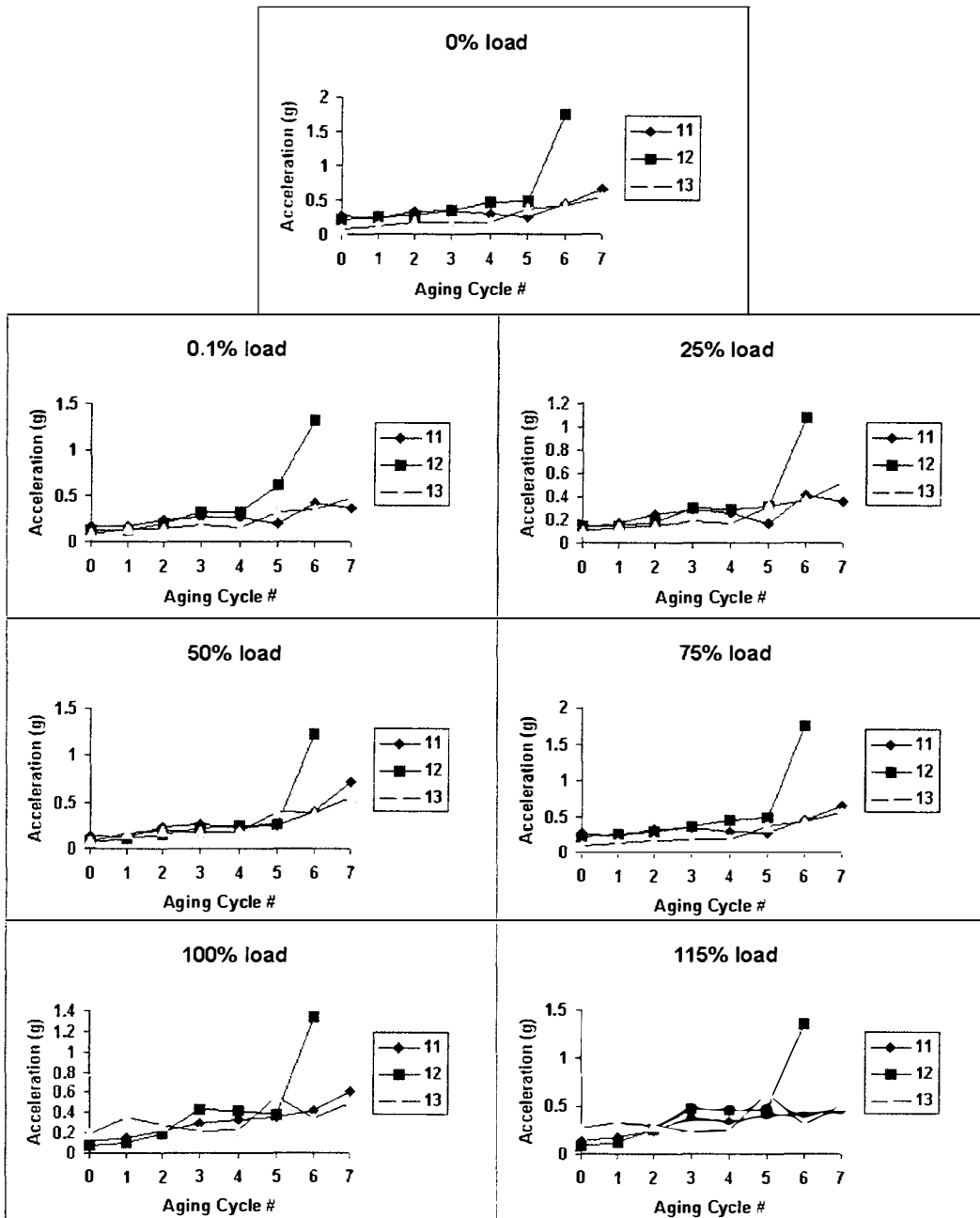


Figure 6.10: Bearing fluting test process end 2:00 accelerometer trends.

BEARING FLUTING AND AGING SE V ACCELEROMETER RMS TRENDS

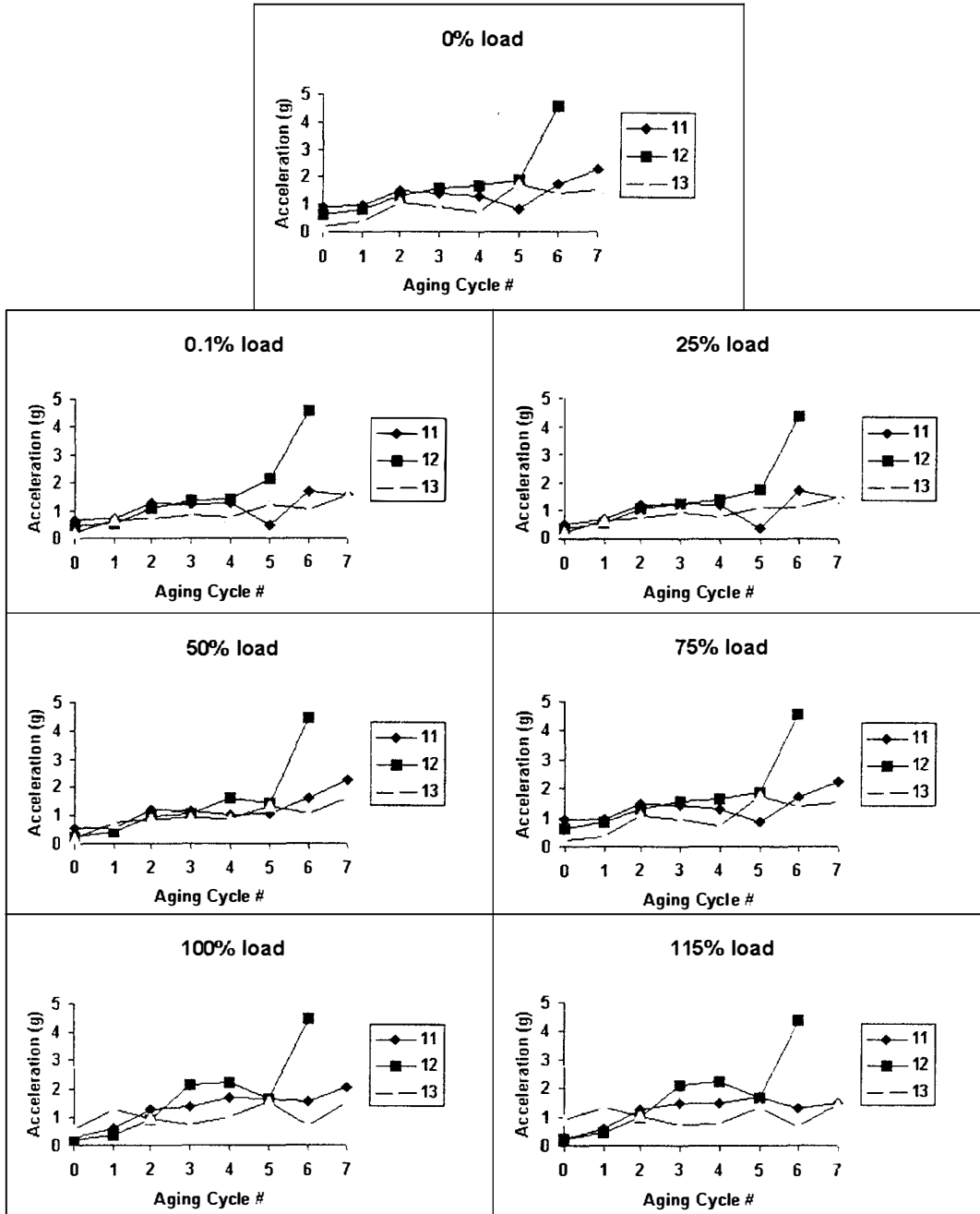


Figure 6.11: Bearing fluting test short end vertical accelerometer RMS trends.

BEARING FLUTING AND AGING BEARING SURFACE PE TEMPERATURE TRENDS

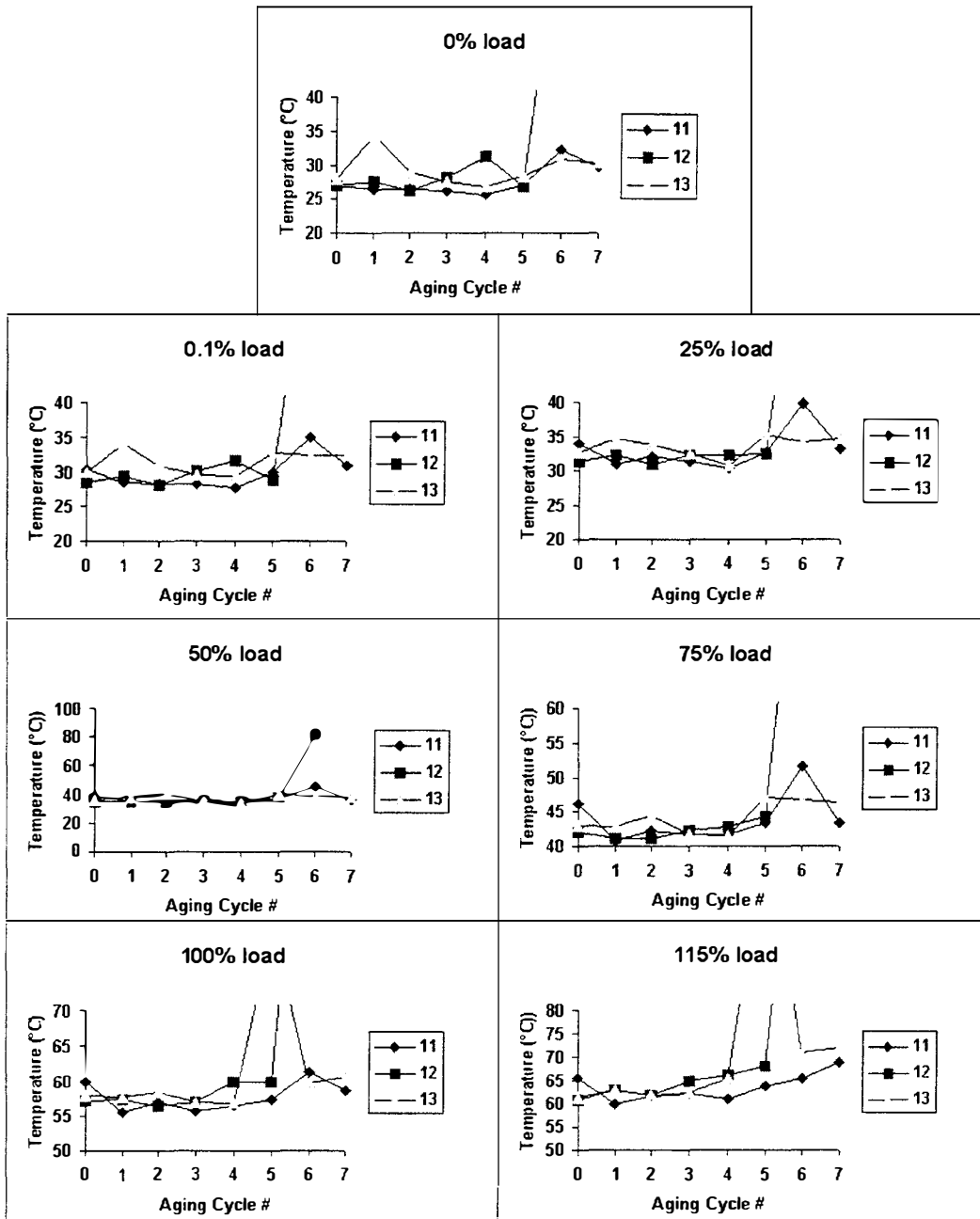


Figure 6.12: Bearing fluting test process end bearing surface temperature trends.

BEARING FLUTING AND AGING BEARING SURFACE SE TEMPERATURE TRENDS

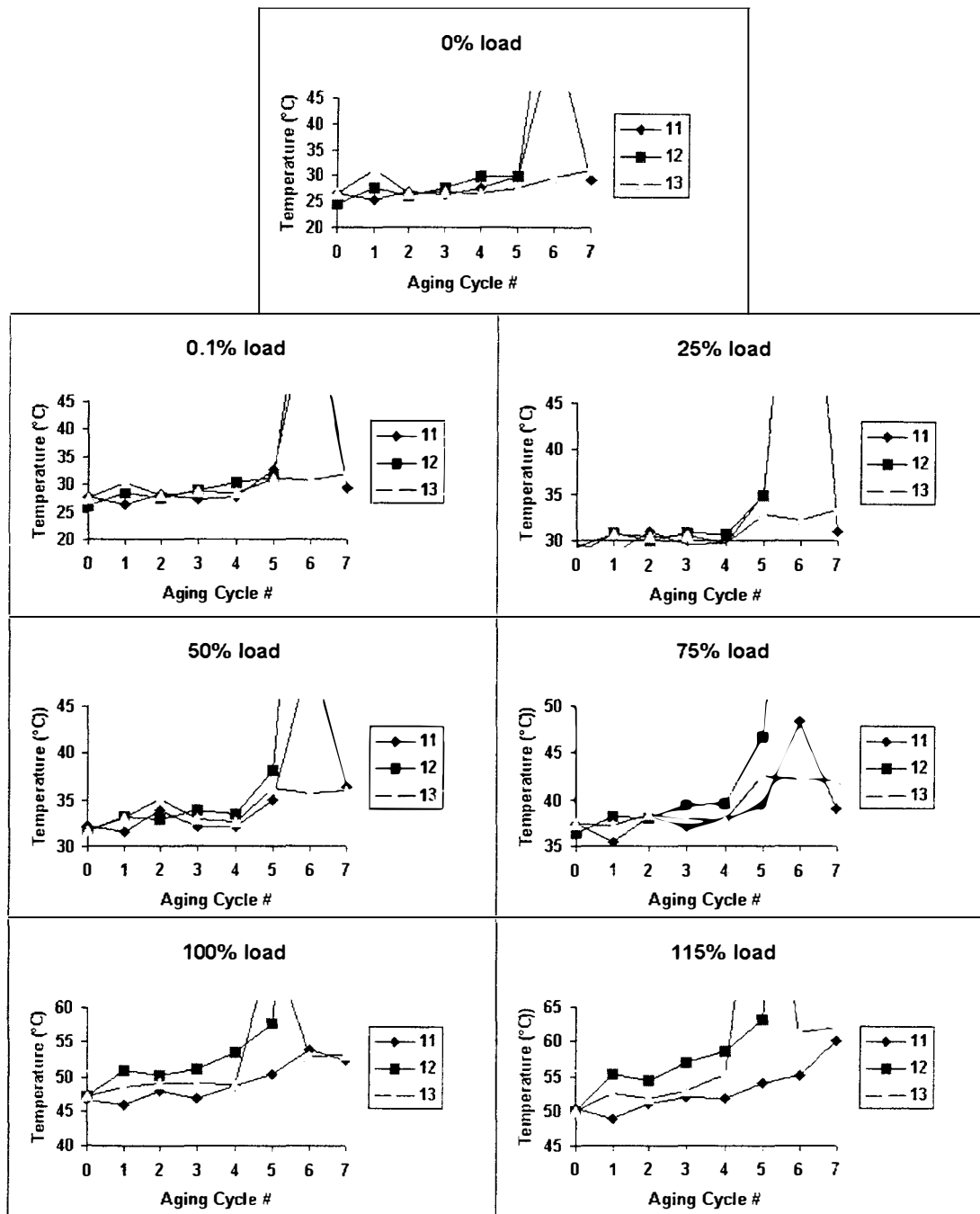
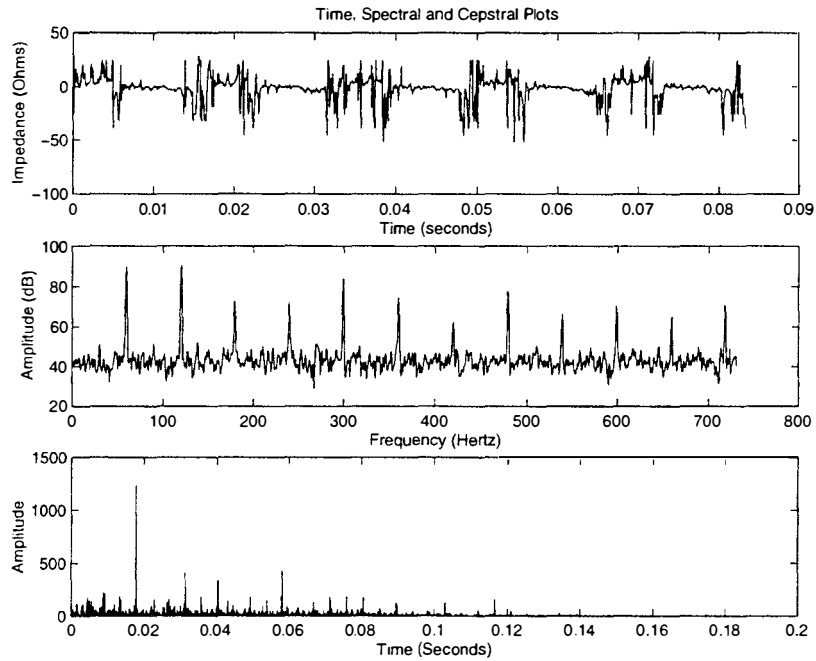
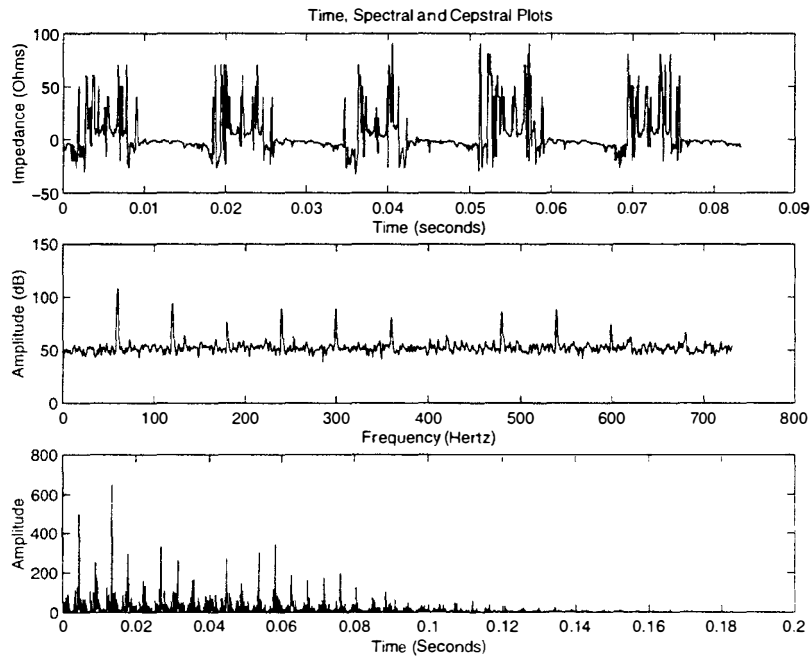


Figure 6.13: Bearing fluting test short end bearing surface temperature trends.



a) Initial



b) After 8 aging cycles

Figure 6.14: Zero-sequence component impedance time, spectral and cepstral plots of a motor for which the insulation has been thermally aged.

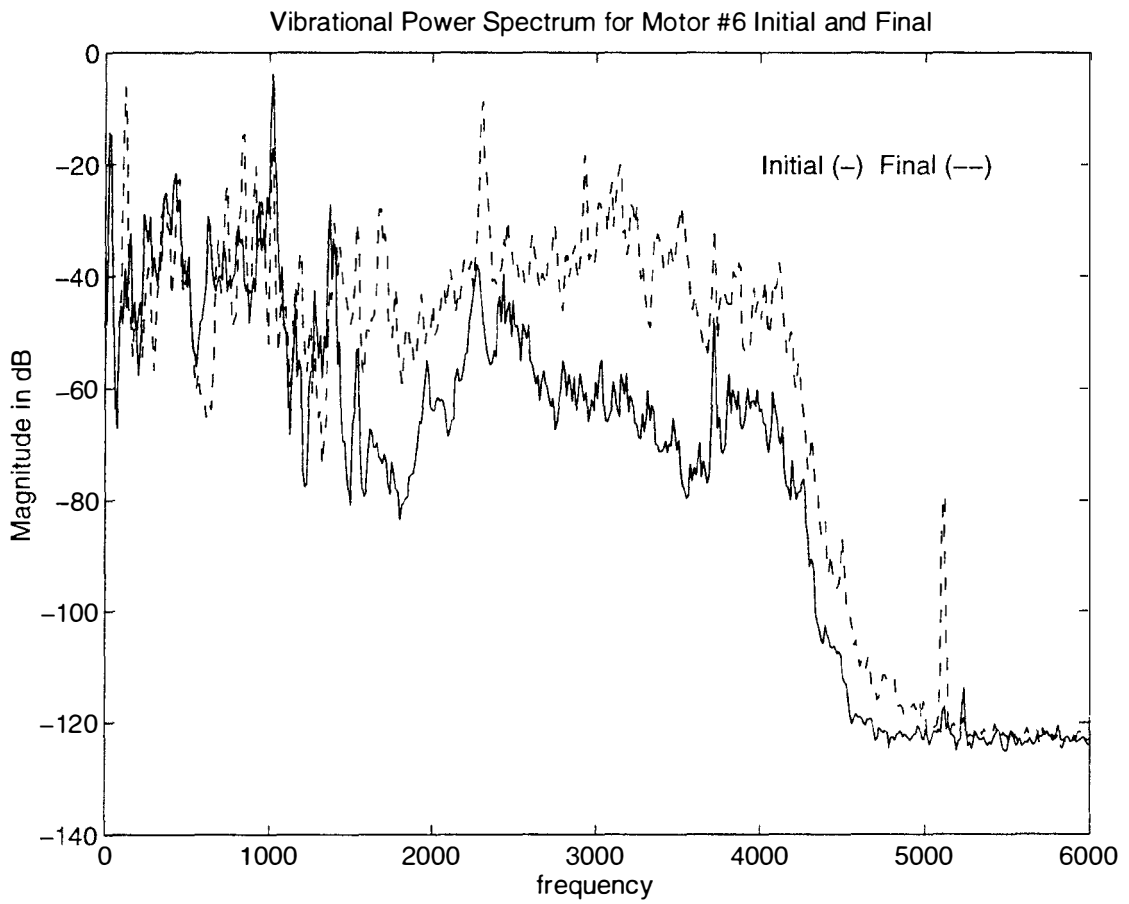


Figure 6.15: Motor #6 bearing aging test process end 2:00 accelerometer spectrum.

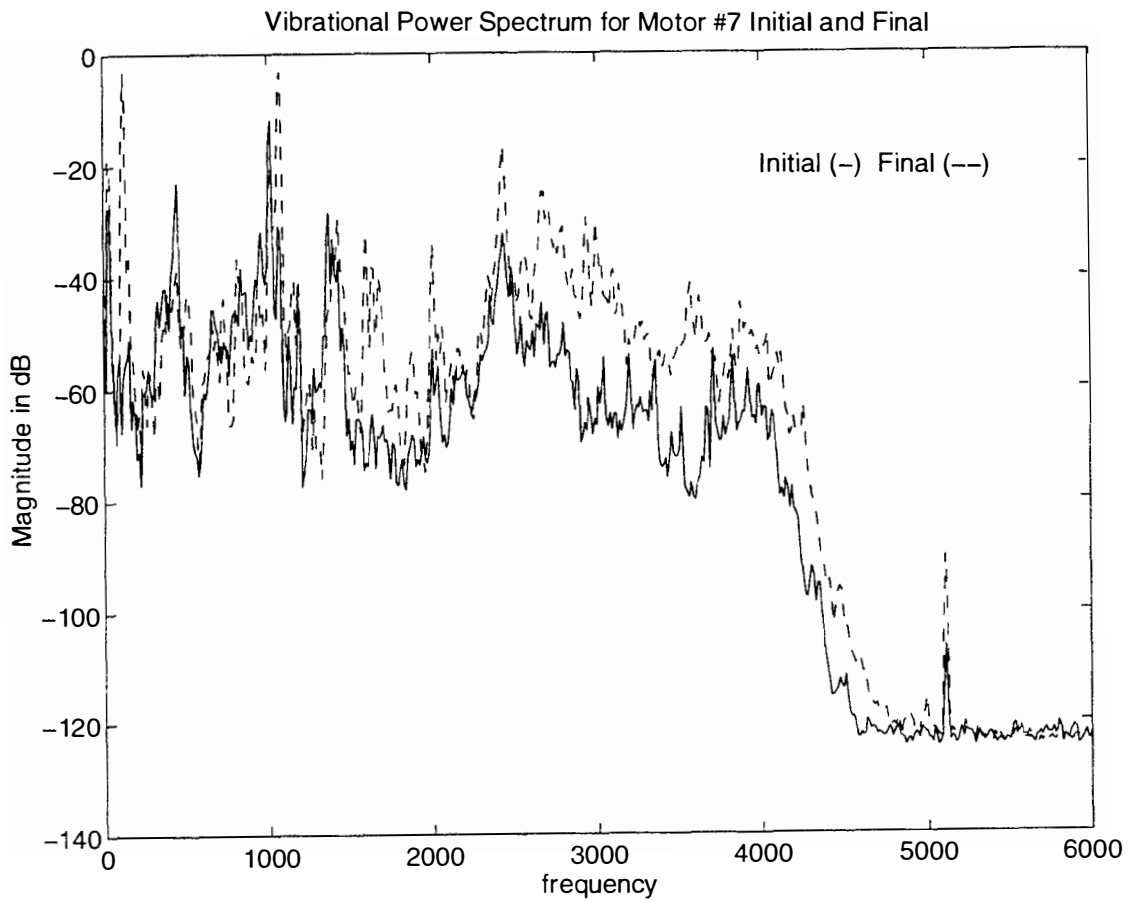


Figure 6.16: Motor #7 bearing aging test process end 2:00 accelerometer spectrum.

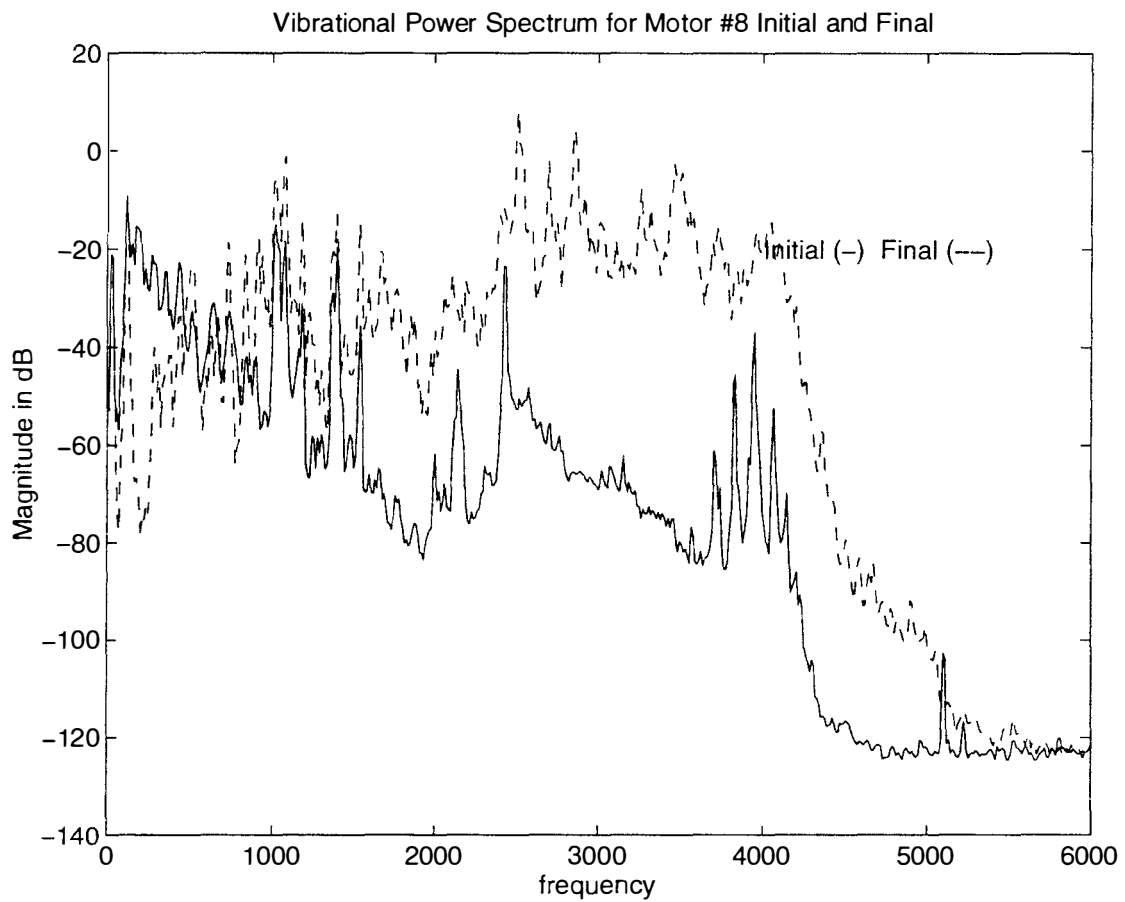


Figure 6.17: Motor #8 bearing aging test process end 2:00 accelerometer spectrum.

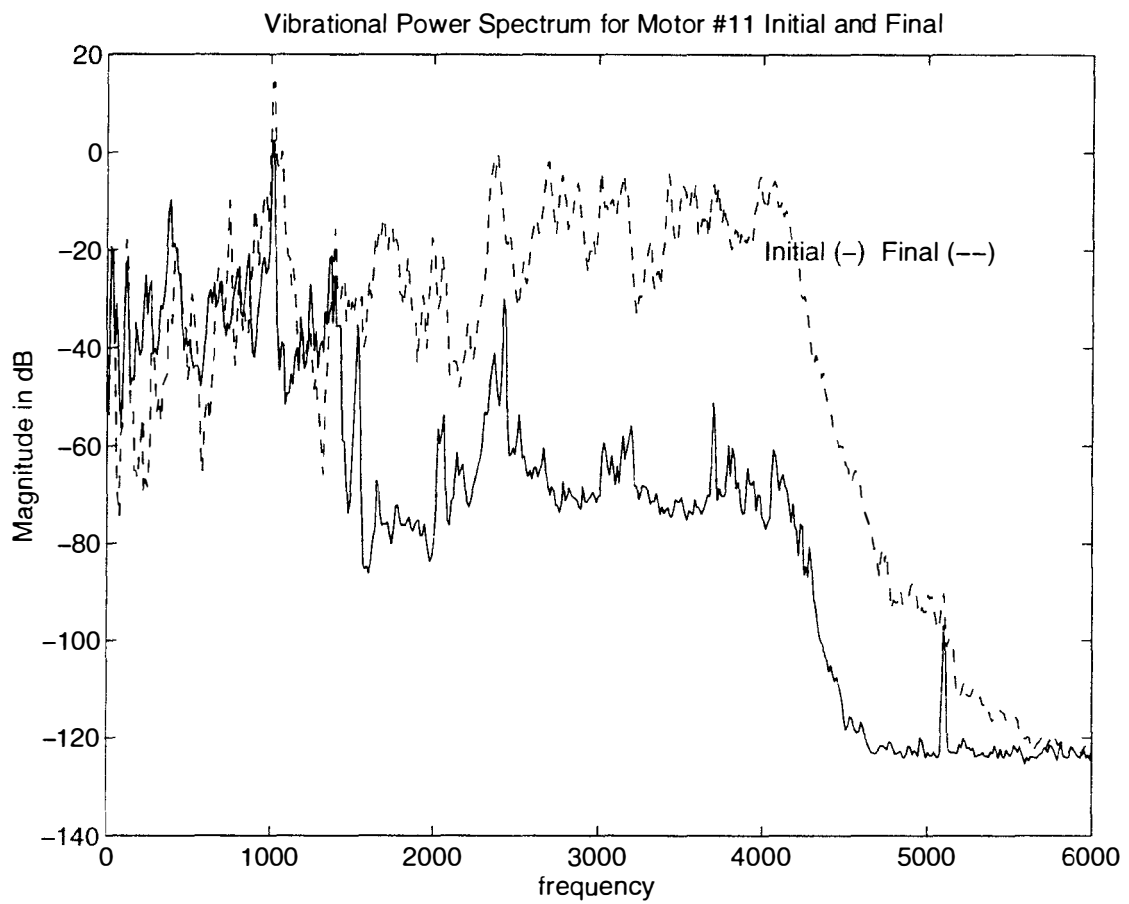


Figure 6.18: Motor #11 bearing fluting test process end 2:00 accelerometer spectrum.

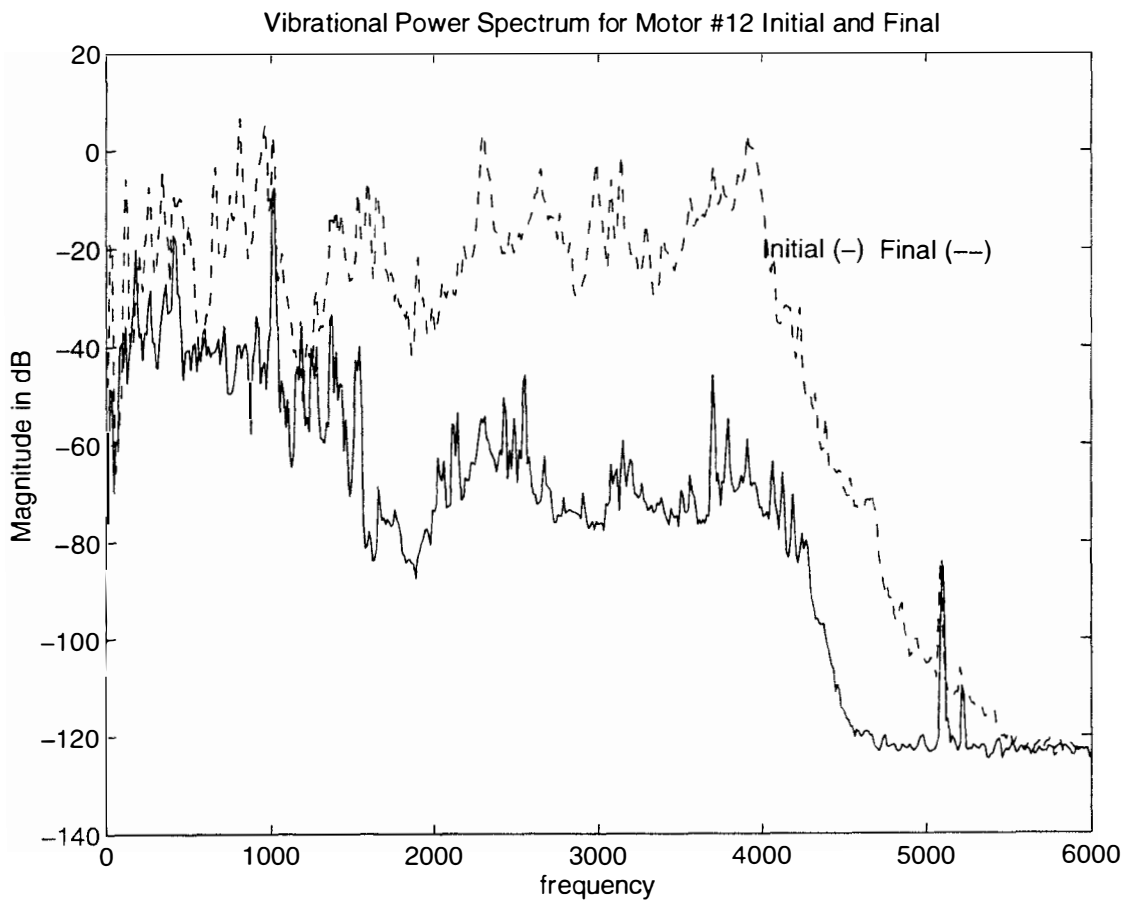


Figure 6.19: Motor #12 bearing fluting test process end 2:00 accelerometer spectrum.

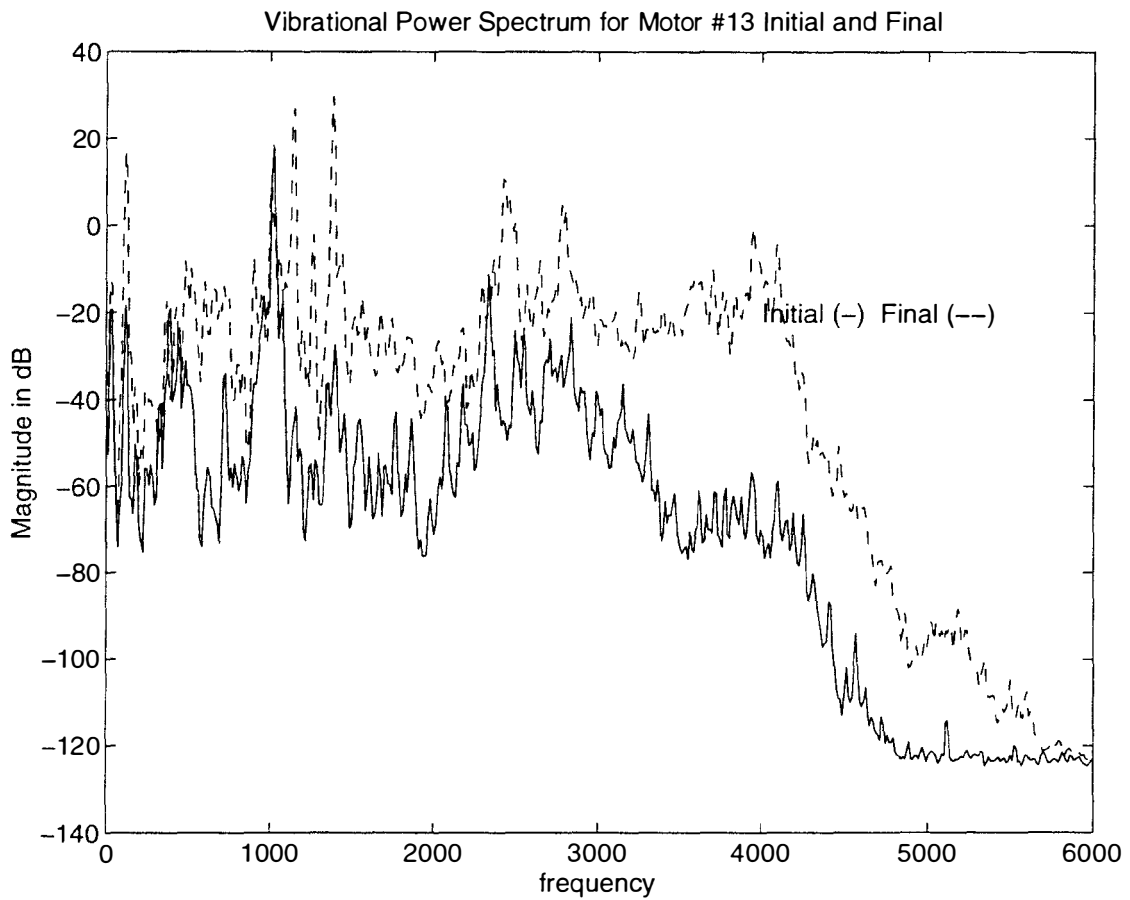
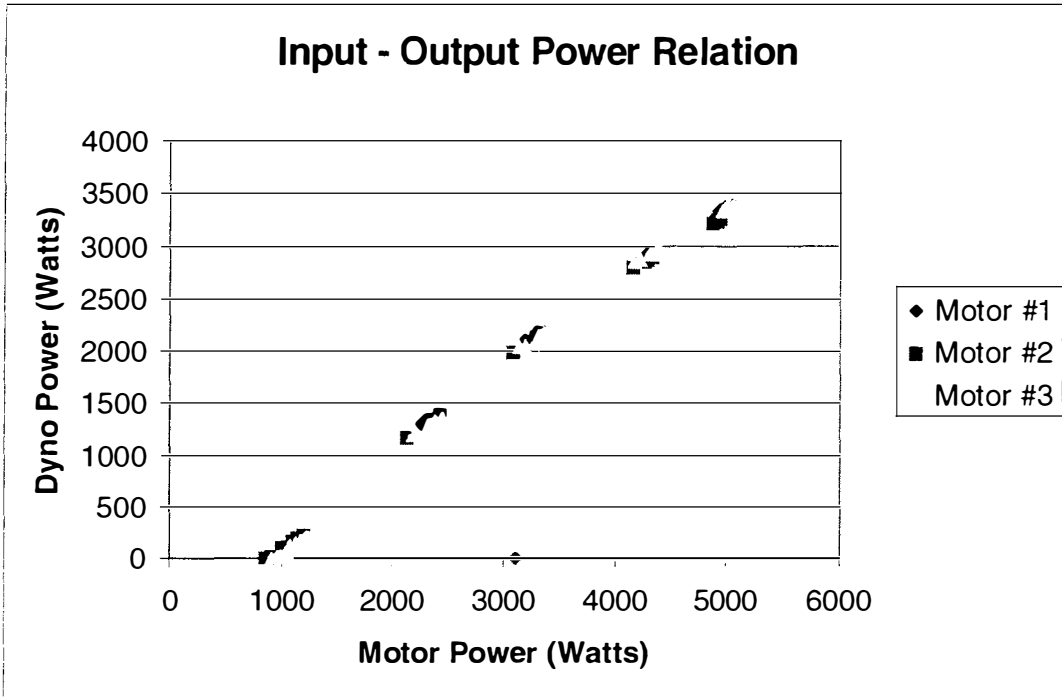
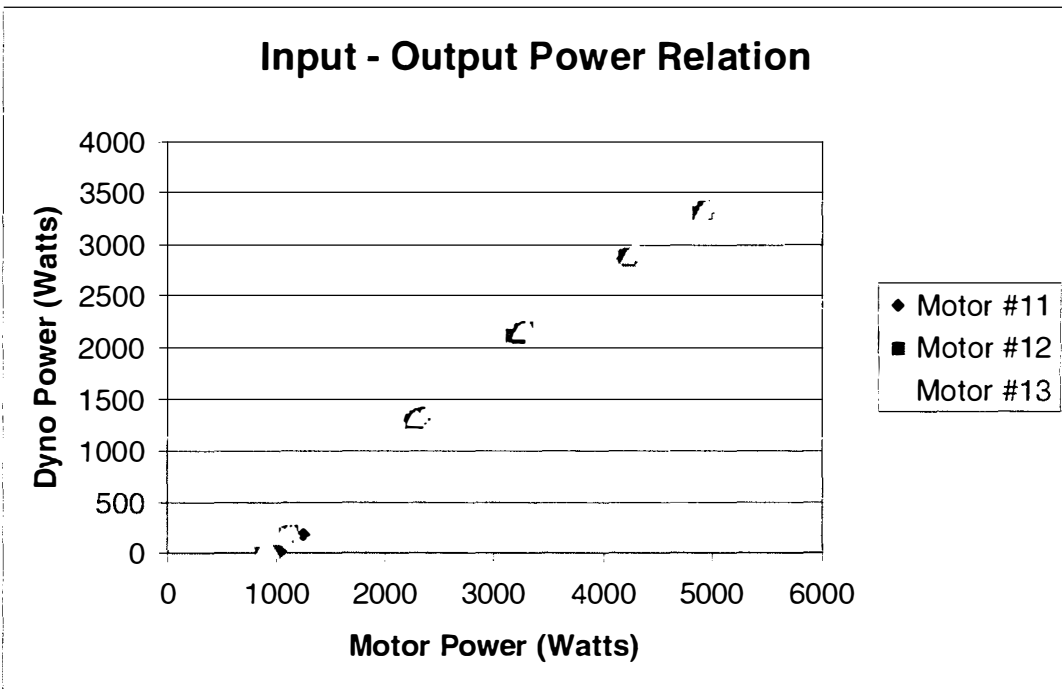


Figure 6.20: Motor #13 bearing fluting test process end 2:00 accelerometer spectrum.



a) Motor #1 had winding insulation failure.



b) Motor #13 had phase current / voltage imbalance due to bad terminal connection.

Figure 6.21: Input – output power relation of the system.

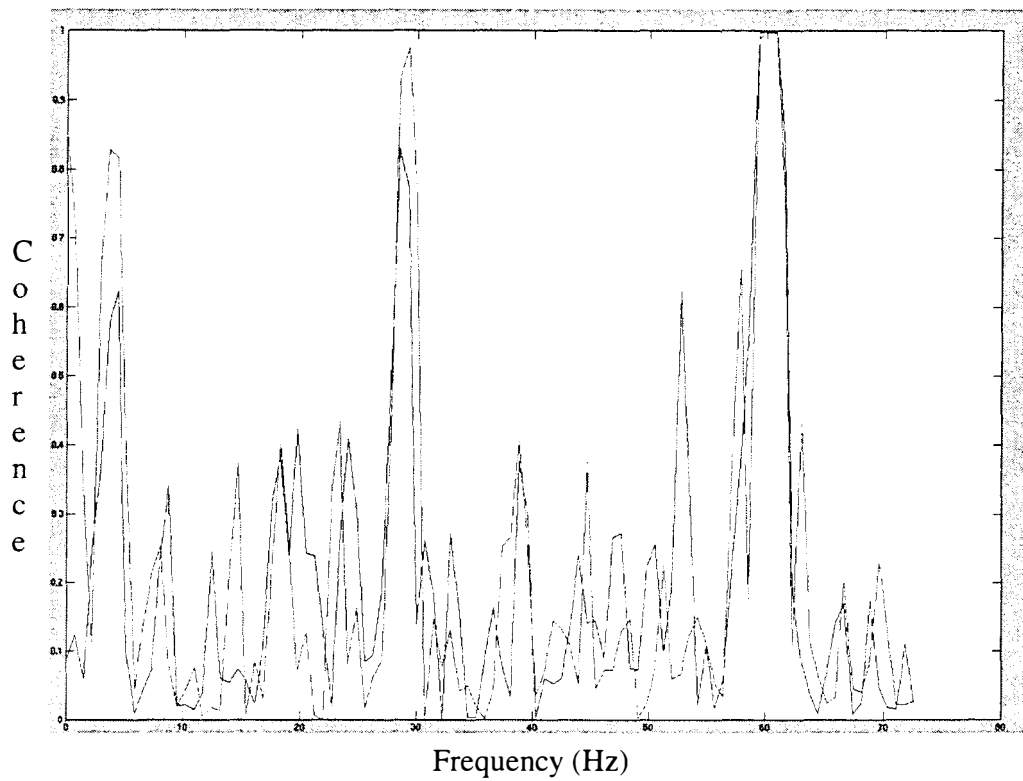


Figure 6.22: Initial (light-color) and final (dark-color) input – output power coherence of motor #11 at 100% load level.

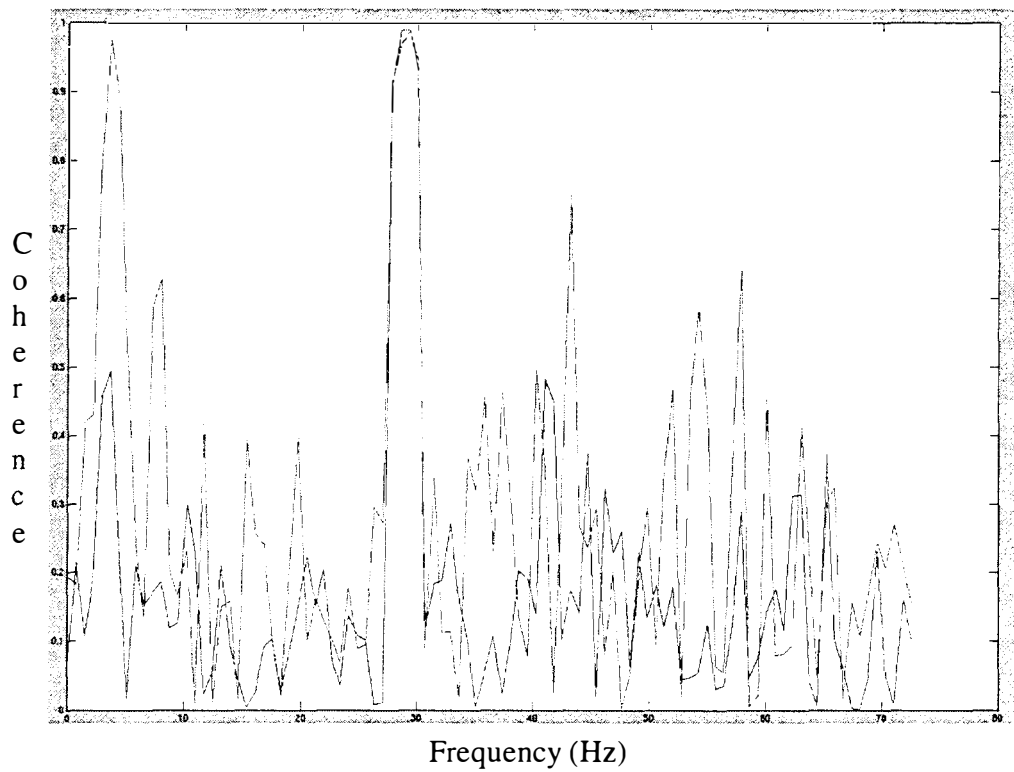


Figure 6.23: Initial (light-color) and final (dark-color) motor power - vibration coherence of motor #11 at 100% load level.

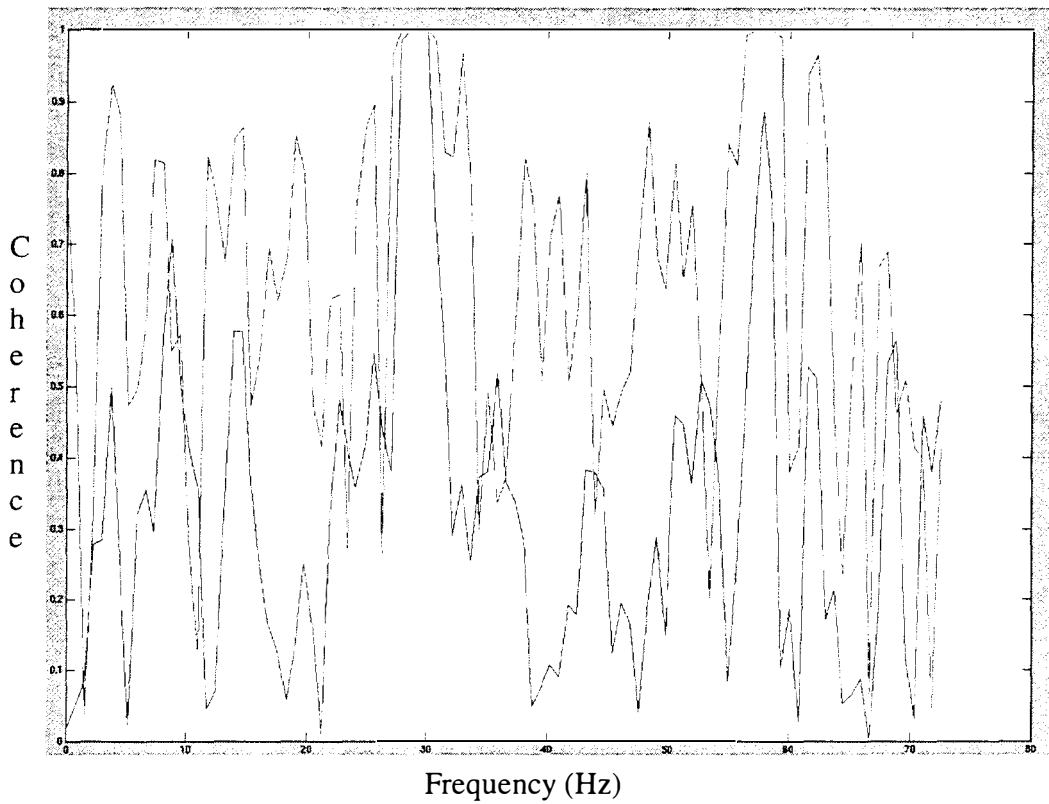
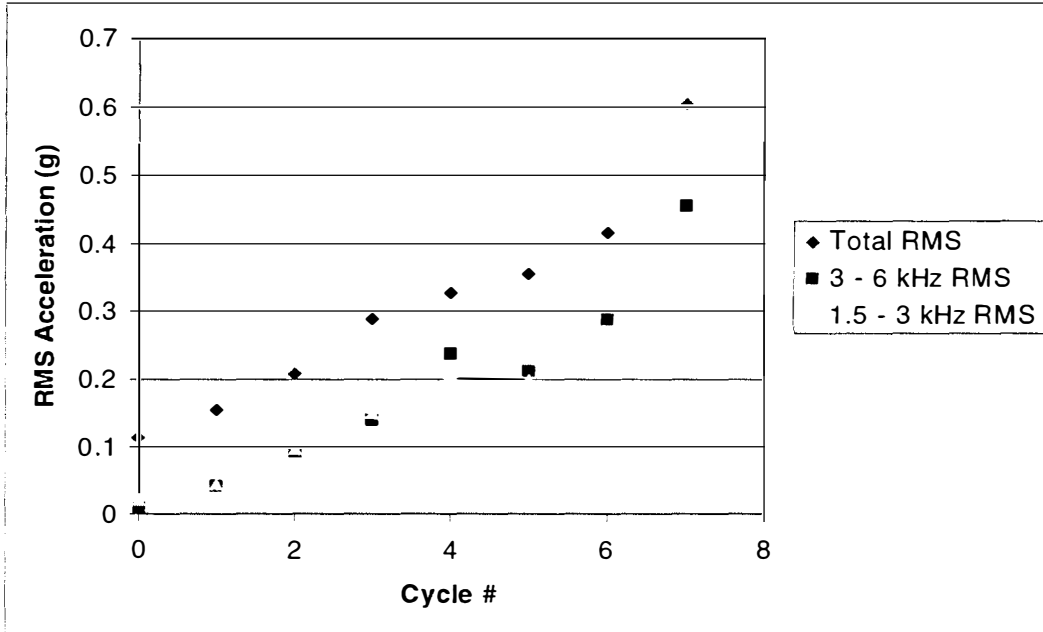


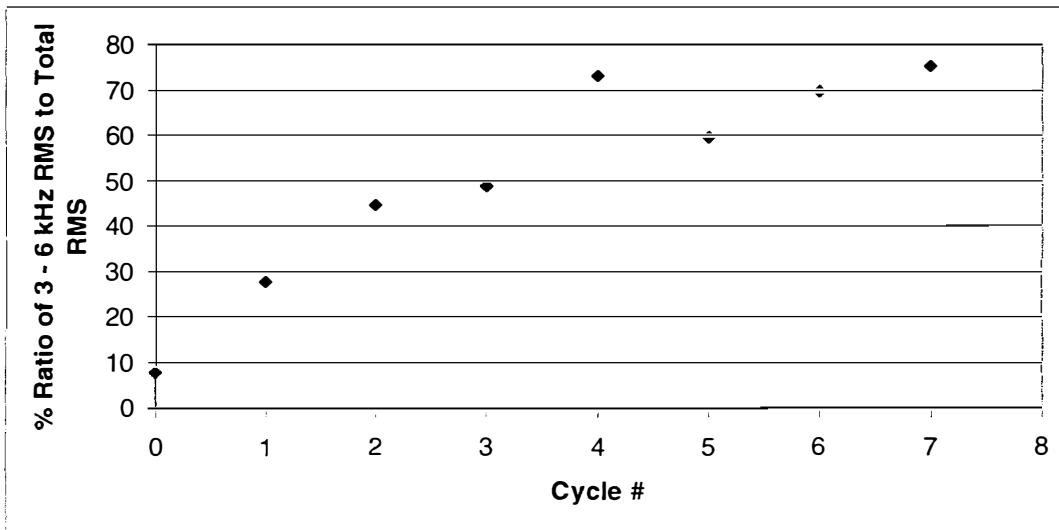
Figure 6.24: Initial (green) and final (blue) motor PE 2:00 – cover vibration coherence of motor #11 at 100% load level.

Table 6.2: Sub-bands of Wavelet Multi-Resolution Analysis [55].

Details	Sub-bands (Hz)	Approximations	Sub-bands (Hz)
D1	6000 – 3000	A1	0 – 3000
D2	3000 – 1500	A2	0 – 1500
D3	1500 – 750	A3	0 – 750
D4	750 – 375	A4	0 – 375
D5	375 – 187.5	A5	0 – 187.5
D6	187.5 – 93.75	A6	0 – 93.75
D7	93.75 – 46.875	A7	0 – 46.875
D8	46.875 – 23.4375	A8	0 – 23.4375



a) RMS values of process end 2:00 acceleration signal for motor in accelerated bearing fluting aging at 100% load.



b) Visual representation that the higher frequencies dominate the overall signal as motor tends for bearing failure.

Figure 6.25: Comparison of band RMS values to total RMS values for vibration measurements.

frequency in the cepstral plots. As the winding insulation fails, electrical discharges between the windings occur more often, however, since these charges are small in magnitude, they are difficult to detect with other classical methods such as motor signature analysis.

- Use of multi-resolution analysis (MRA) on vibration data revealed that the higher frequency components (3 – 6 kHz) dominate the overall signal as the motor ages and causes failure of the motor bearings [55]. The MRA is capable of effectively trending information related to motor bearings without signal distortion. When vibration was measured at process-end 2:00 at full load, it was observed that there exists a similarity between the overall RMS trend in the signal and the overall RMS trends in some sub-band signals. The first *detail (D1)* in the 3-6 kHz frequency band, shows the most dominant contribution to the signal energy. This is followed by second *detail (D2)* in the 1.5-3 kHz range. These observations can be seen in Figure 6.25, where the frequency bands of MRA are given in Table 6.2.
- It can be concluded from the results that electrical fluting damage of motor bearings give rise to high-frequency vibration signatures that are not normally seen in electrical motors. Vibration spectrums shown in Figure 6.18- Figure 6.20 exhibit a much higher amplitude increase in higher frequency bands than those seen corresponding to thermal bearing aging alone as shown in Figure 6.15 -

Figure 6.17.

- Efficiency is not a good indicator of incipient failure detection, but rather is a validation tool that a motor has failed. This observation can be easily seen in Figure 6.21, where the input and output power is plotted as a different representation of efficiency. Tracking back the points in the graph, where they do not fit linear relationship, we see that in one of the cases (motor #13) had a bad terminal connection, causing a current imbalance, while in another case (motor #1) had already developed a short in the winding.
- If the motor is considered to be an input-output system, in which the possible input and outputs are
 1. Motor power – dynamometer output power,
 2. Motor power – motor vibration,
 3. Motor PE 2:00 – cover vibration,

Then, we can look into the corresponding coherence spectrums and see how the spectrum changes with respect to aging. These plots are shown in Figure 6.22 - Figure 6.24, and show that as the motor ages, it loses the ability to convert electrical energy into mechanical energy effectively (Decrease in amplitude at several frequencies).

6.4 Normalization of Measured Signatures

In order to make a decision at any power level, we have to normalize the signatures with respect to a certain load (namely 100% load level) and if it is a temperature, with respect to the ambient temperature. This is because, it is nearly impossible to formulate an algorithm at specific load levels and expect the algorithm to work in reality only when the motor is operating at that certain load level. The proper way to process the measurement signature is to normalize it to a specific load level. Also, when dealing with temperature measurements, the variation in ambient (surrounding) temperature of the motor will also affect thermally any temperature measurements inside the motor. If a trending algorithm is to be established based on temperature, then the ambient temperature has to be compensated also, so that the temperature, which is a result of energy loss, can be measured accurately. Equation (5.17) is the solution for this problem. Figure 6.26 - Figure 6.33 show the function, which is used to normalize the measured variable to 100% load level as given in Equation (5.17). The study of these normalization techniques based on experimental data is unique.

An example of how to use such a temperature normalization function was given in Chapter 5. As an another example, we can consider normalizing PE 2:00 RMS vibration as following:

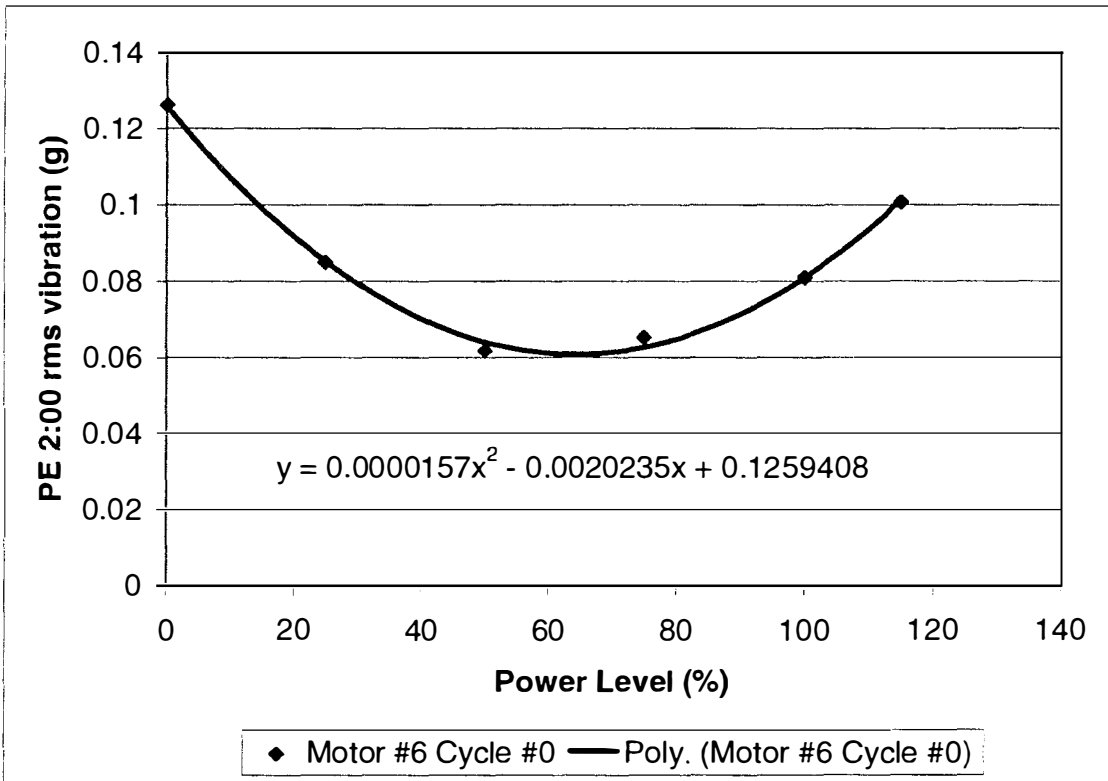


Figure 6.26: Normalization of PE 2:00 RMS vibration.

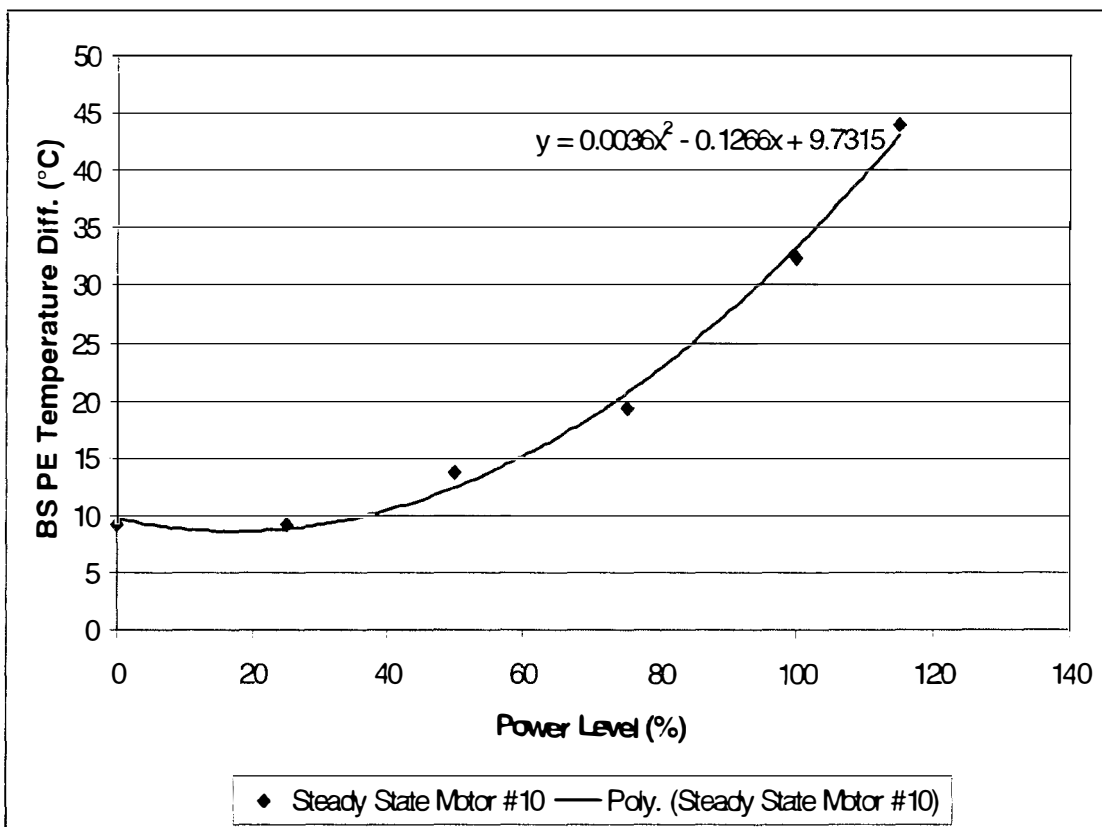


Figure 6.27: Normalization of bearing surface PE temperature difference.

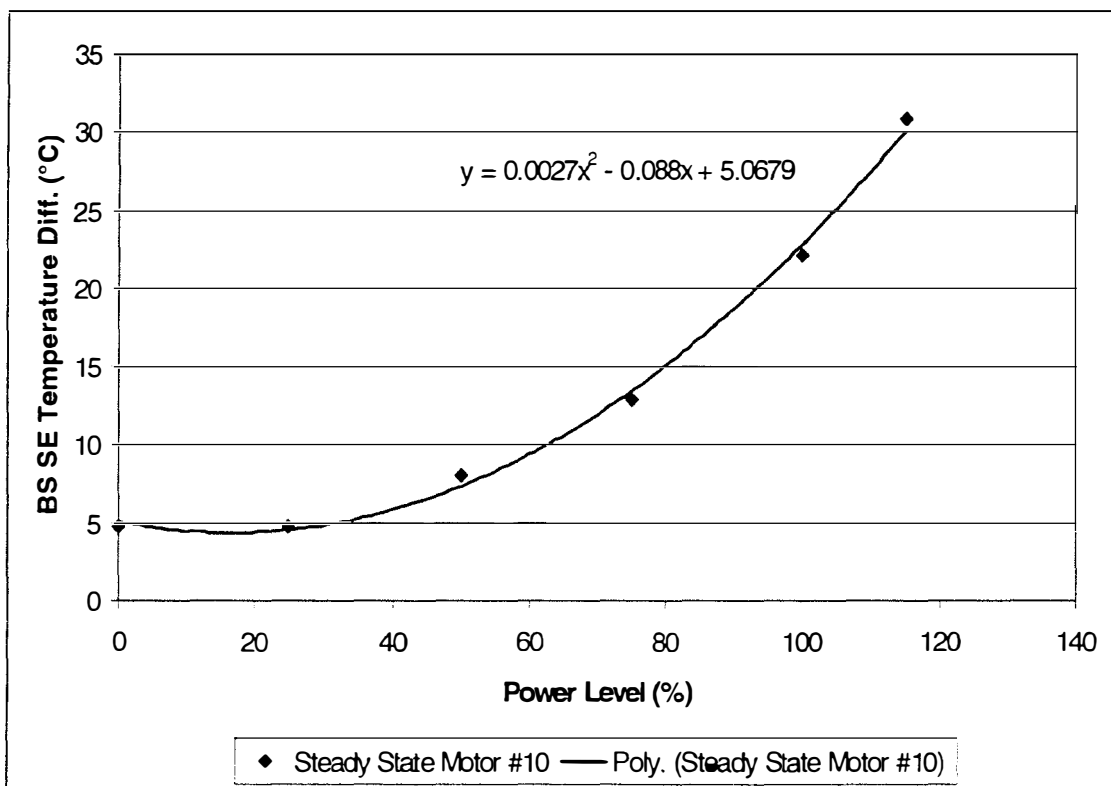


Figure 6.28: Normalization of bearing surface SE temperature difference.

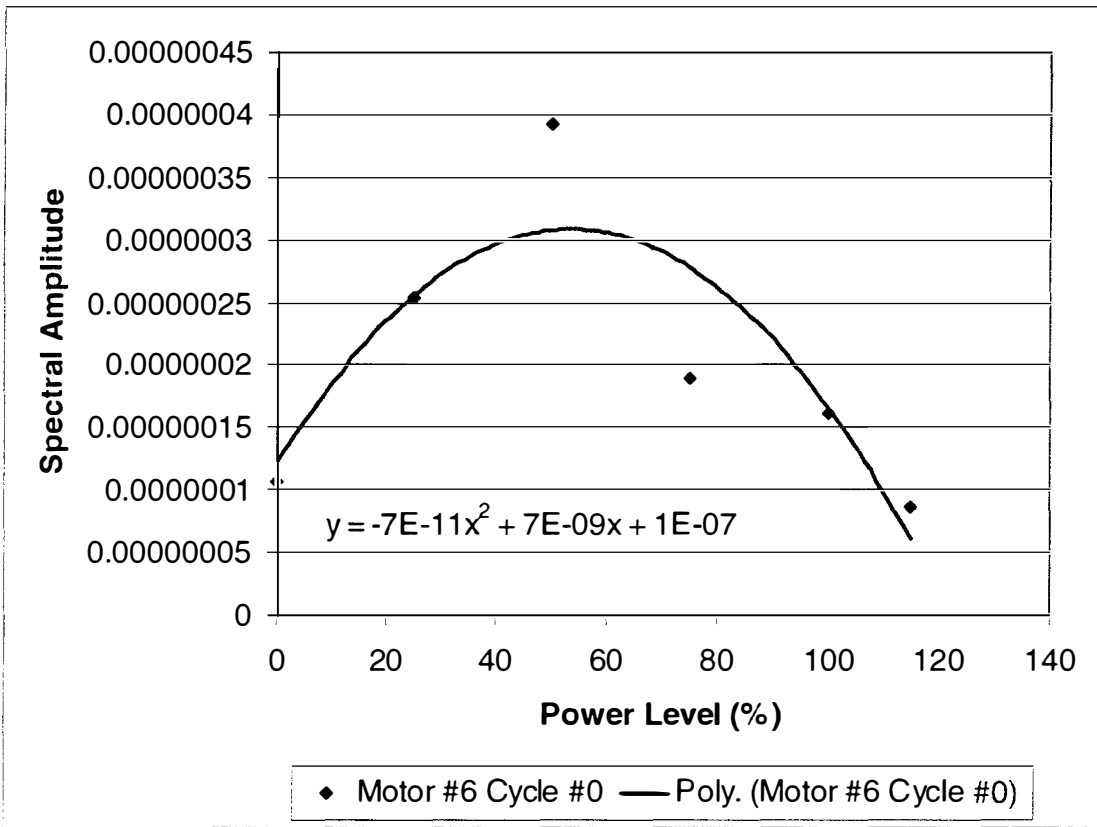


Figure 6.29: Normalization of spectral amplitudes at line frequency of PE 2:00 vibration.

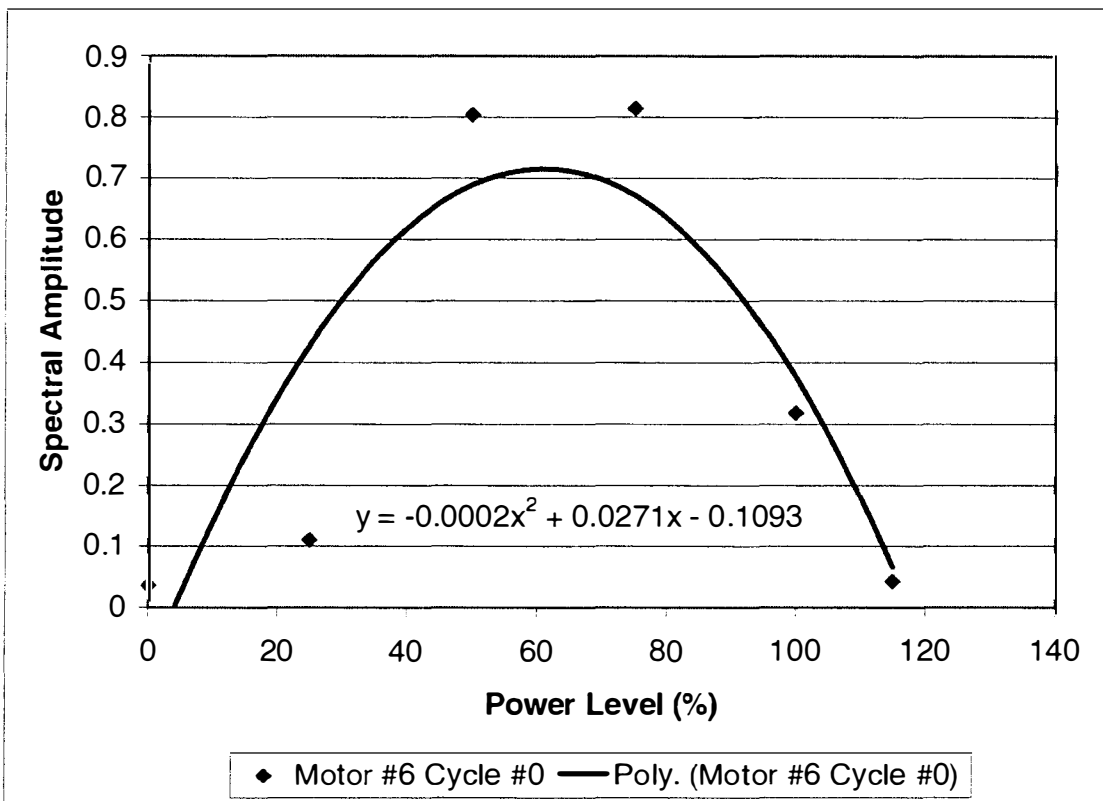


Figure 6.30: Normalization of spectral amplitudes at line frequency of motor power.

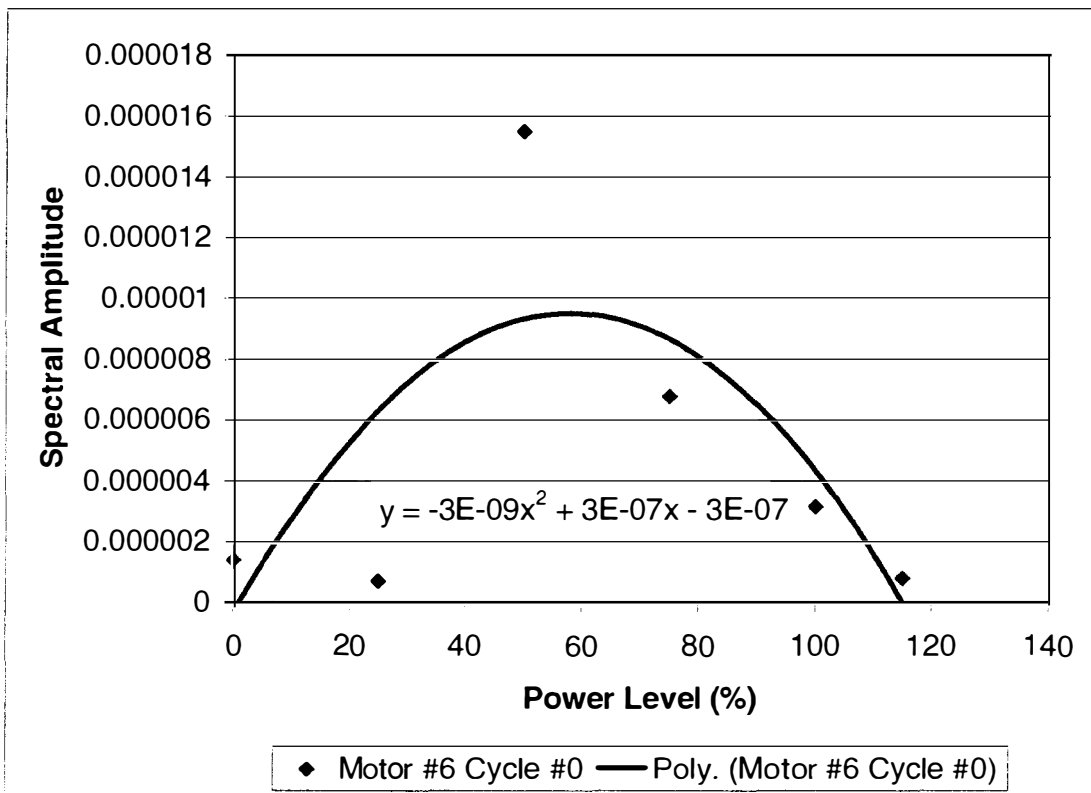


Figure 6.31: Normalization of spectral amplitudes at line frequency of motor current.

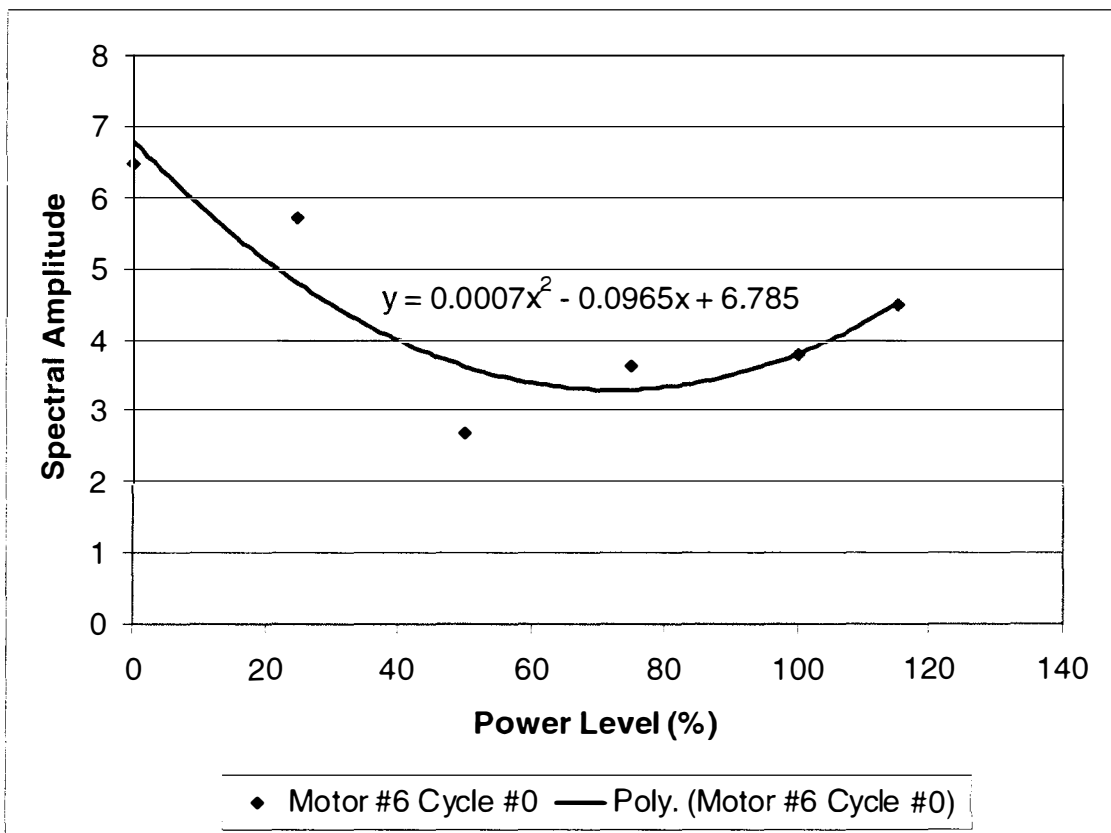


Figure 6.32: Normalization of spectral amplitudes at line frequency of motor zero sequence component impedance.

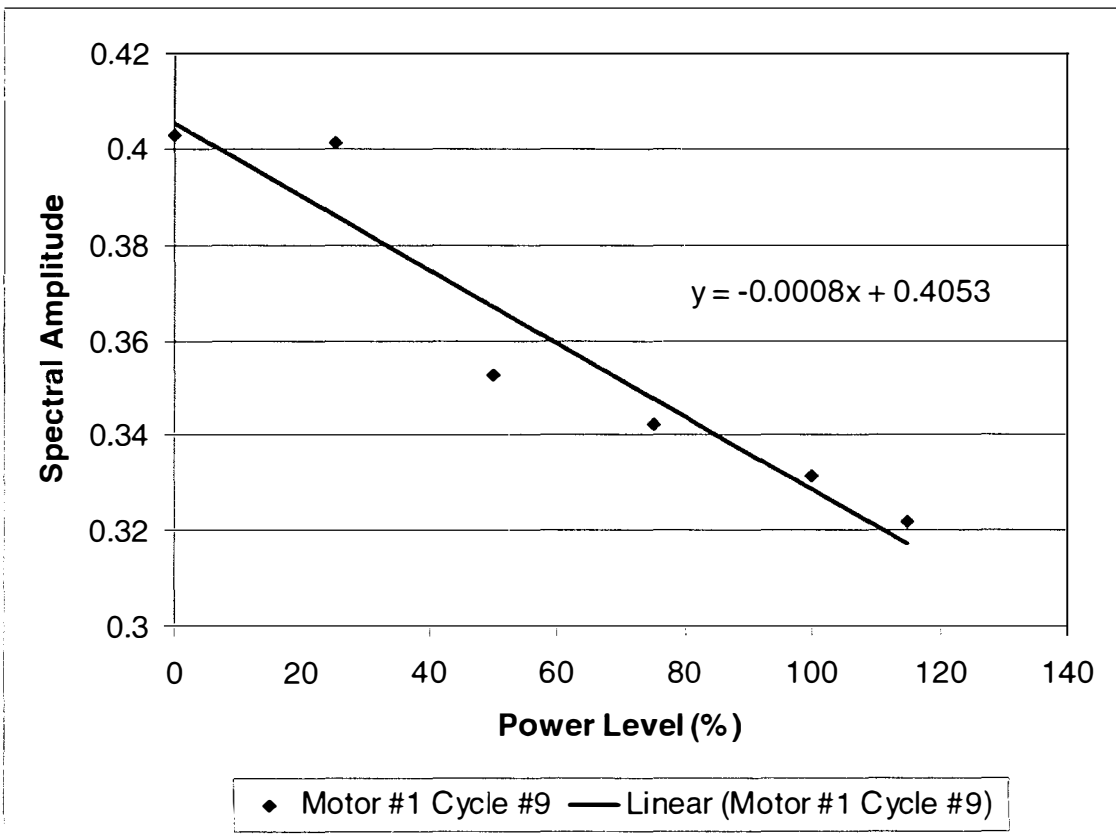


Figure 6.33: Normalization of spectral amplitudes at line frequency of motor radial magnetic flux.

First let's assume that we have measured the PE 2:00 RMS vibration to be 0.075 g's at 50% load level. In order to normalize this measurement to 100% load level, we first calculate the bias constant, which changes as a result of fault:

Given the equation from Figure 6.26:

RMS Vibration =

$$0.0000157 \times (\text{Power Level})^2 - 0.0020235 \times (\text{Power Level}) + \text{Constant}$$

we calculate the constant to be 0.186925 g's. Thereafter, the normalized RMS vibration is calculated at 100% load level as

Normalized RMS Vibration =

$$0.0000157 \times (100\%)^2 - 0.0020235 \times (100\%) + 0.186925$$

which we find to be 0.141575 g's.

One important aspect of this normalization is that the relationship between power level and temperature is not linear, as some studies assume in their calculations. Furthermore, the relationship depends on the sensor location: Different locations in the motor have different heat transfer and mechanical characteristics.

6.5 Trending of Selected Signatures

Once the normalization is performed, the trending of selected signatures is accomplished. These signatures include RMS values of vibration, current and zero-sequence components of impedance as well as mean values of temperature and spectral

amplitudes at specified frequencies. Skewness was also calculated as third moment of the data. Figure 6.34 - Figure 6.94 show the results of the least-square-based trends for each signature that indicates an incipient failure. It should be noted that the fits are performed on data that has inevitable random variations in reality. Table 6.3 is a simple review of the 'Sensitivity Matrix' which tells us which signatures are usable in electrical motors fault detection. This table was created by using the trends and signatures given in Figure 6.34 - Figure 6.94 and visually analyzing them. Later, a summary of the trends is given in Table 6.4.

6.6 Fault Detection and Classification Using Fuzzy Logic

Using the average trend slopes given in Table 6.4, fuzzy membership functions of the signatures, which are sensitive to the specified incipient faults, are derived. These membership functions are given in Figure 6.95 - Figure 6.108. Using the fuzzy logic rule base, given in, classification is achieved successfully as shown in Table 6.5. A very few measurements showed false alarms of winding insulation failure, caused by an insufficient amount of data sets of magnetic flux measurements. Other than this, incipient fault detection of electrical motors was successfully implemented.

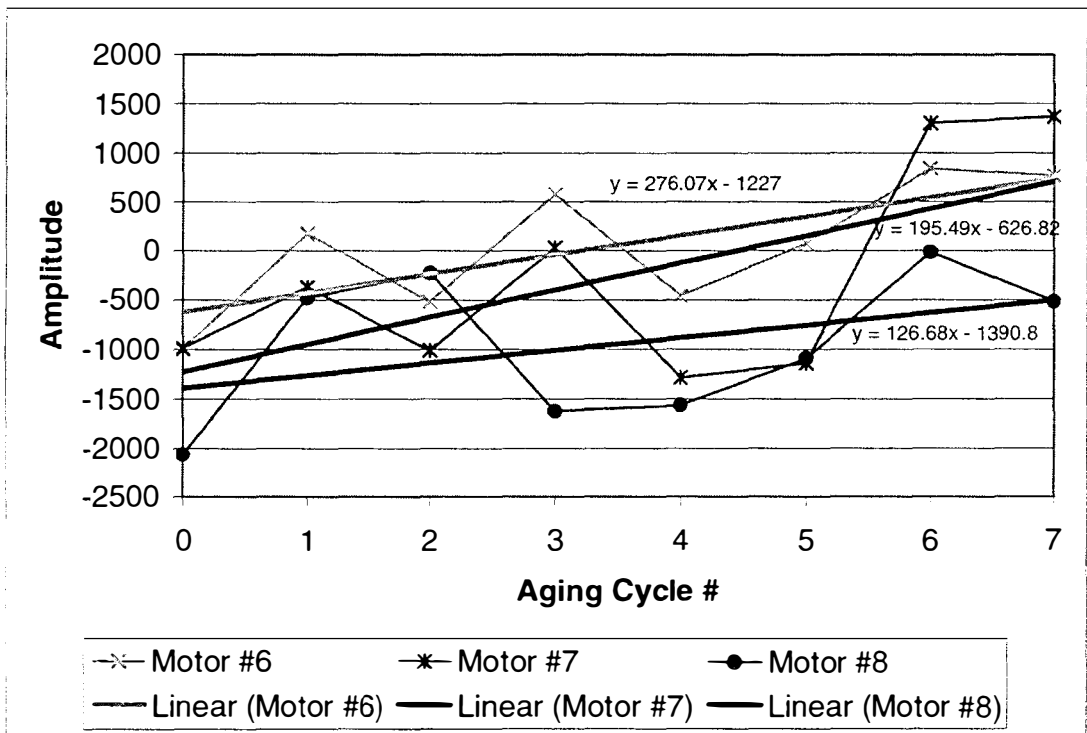
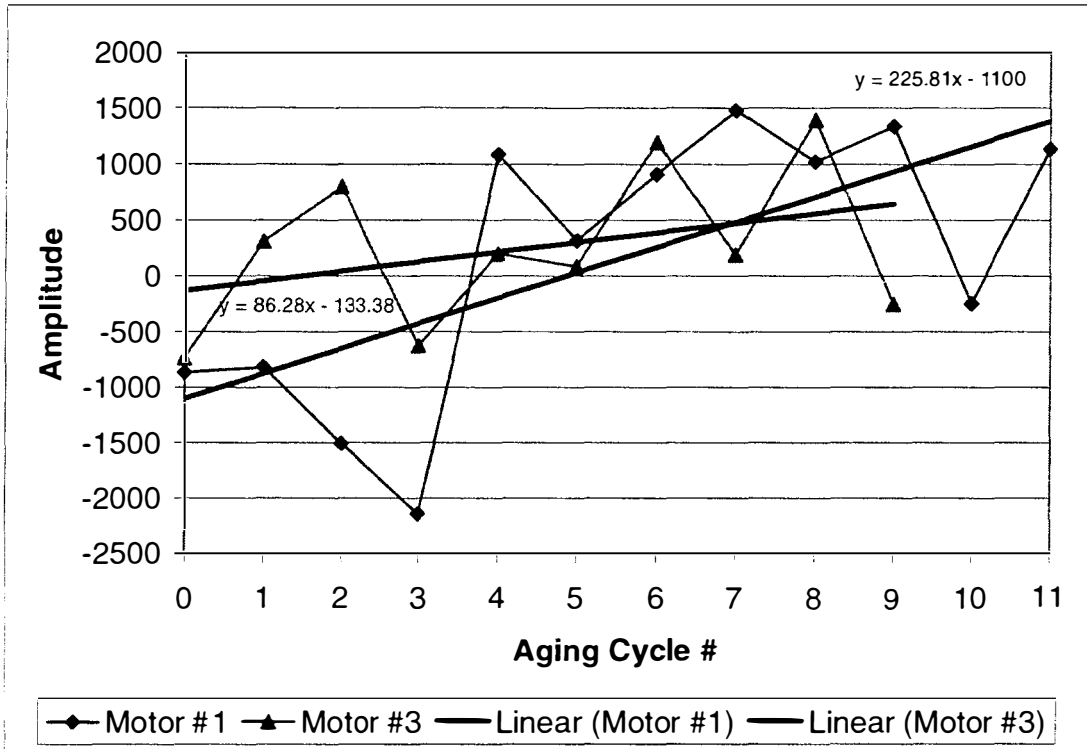


Figure 6.34: Load power skewness at 100% load.

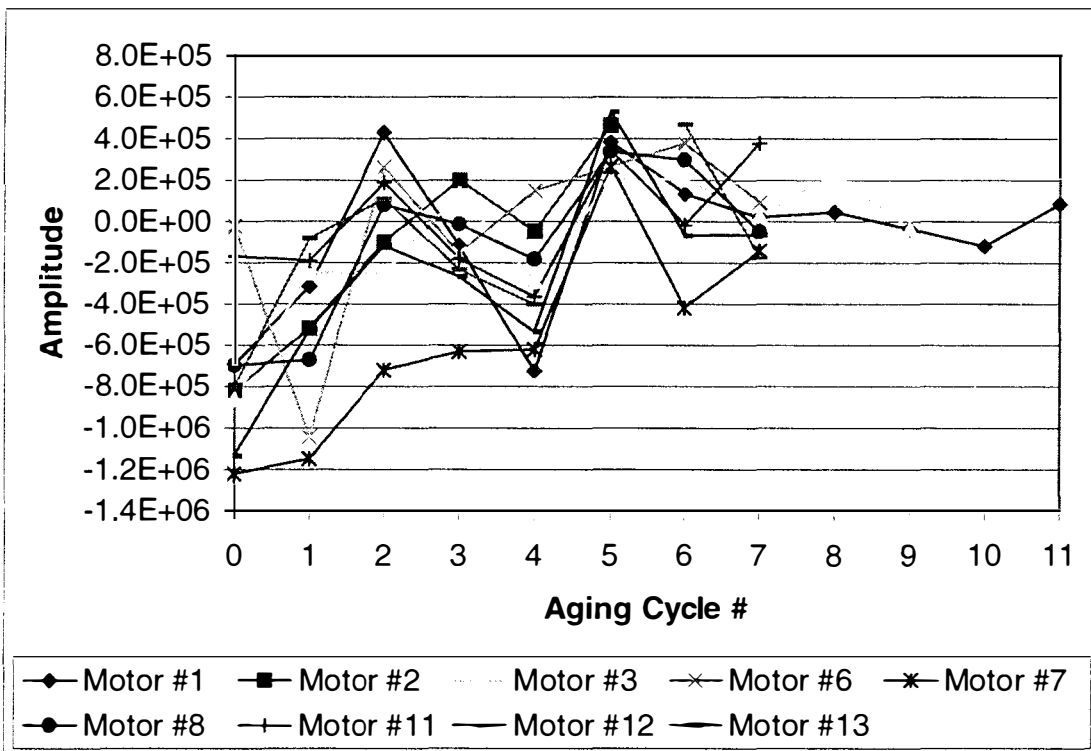


Figure 6.35: Motor power skewness at 100% load.

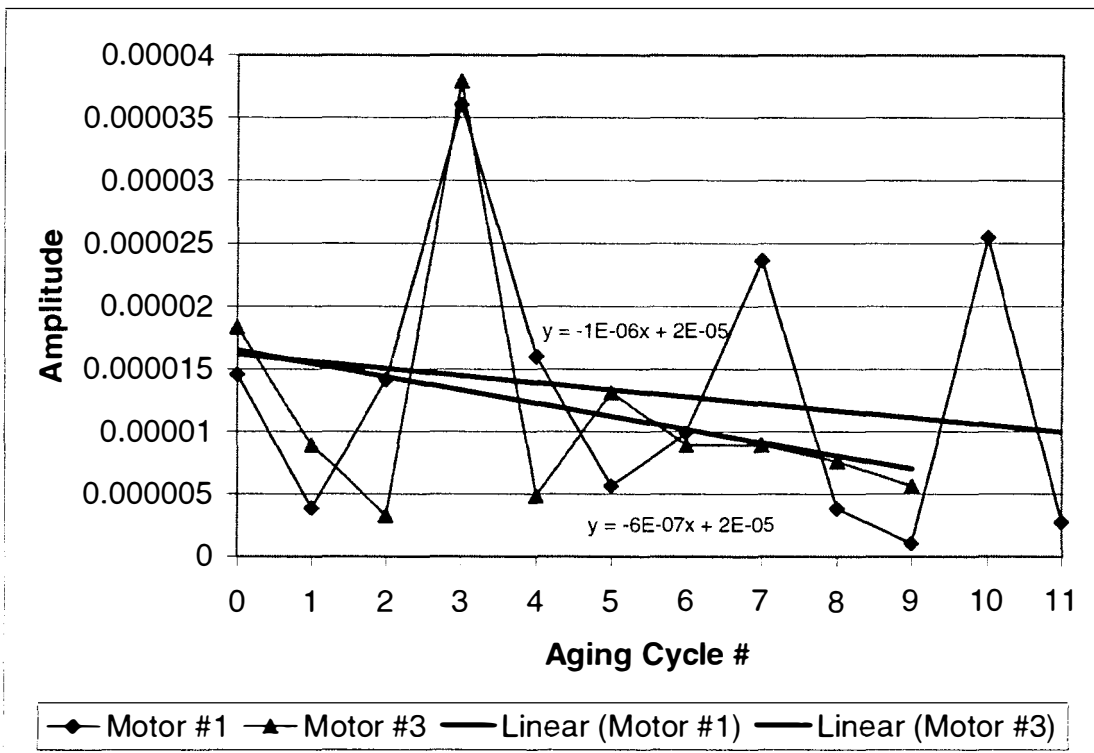


Figure 6.36: Motor current RPM frequency amplitude trend at 100% load.

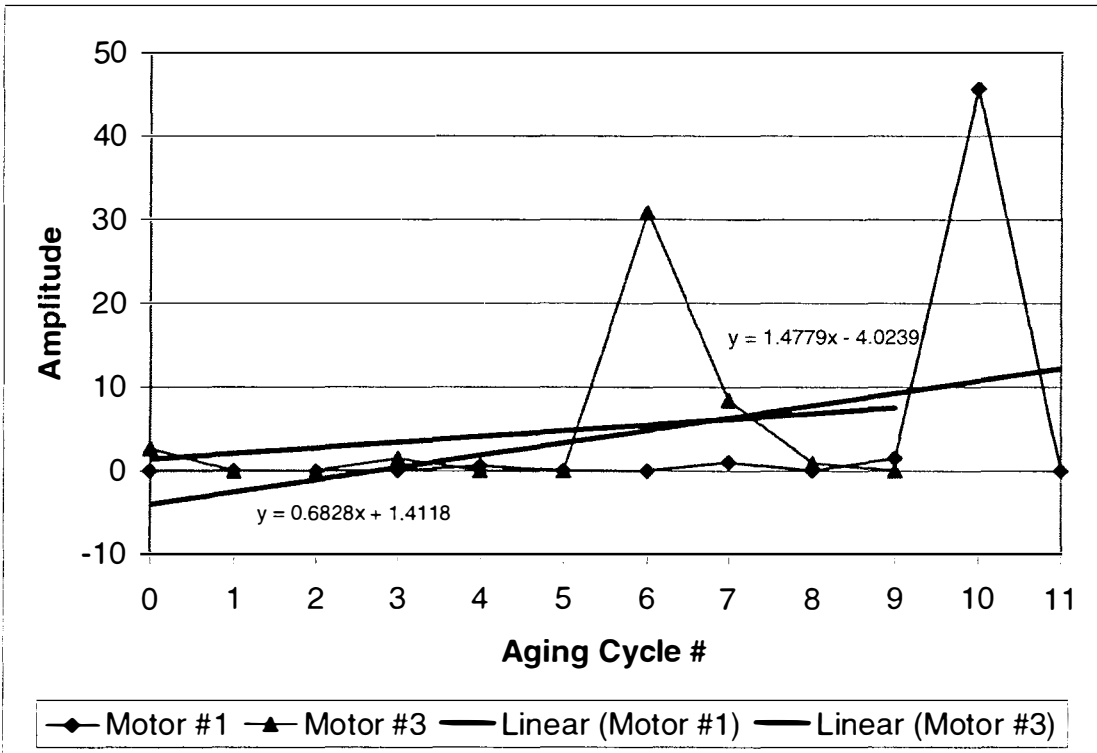


Figure 6.37: Motor zero component impedance RPM frequency amplitude at 100% load.

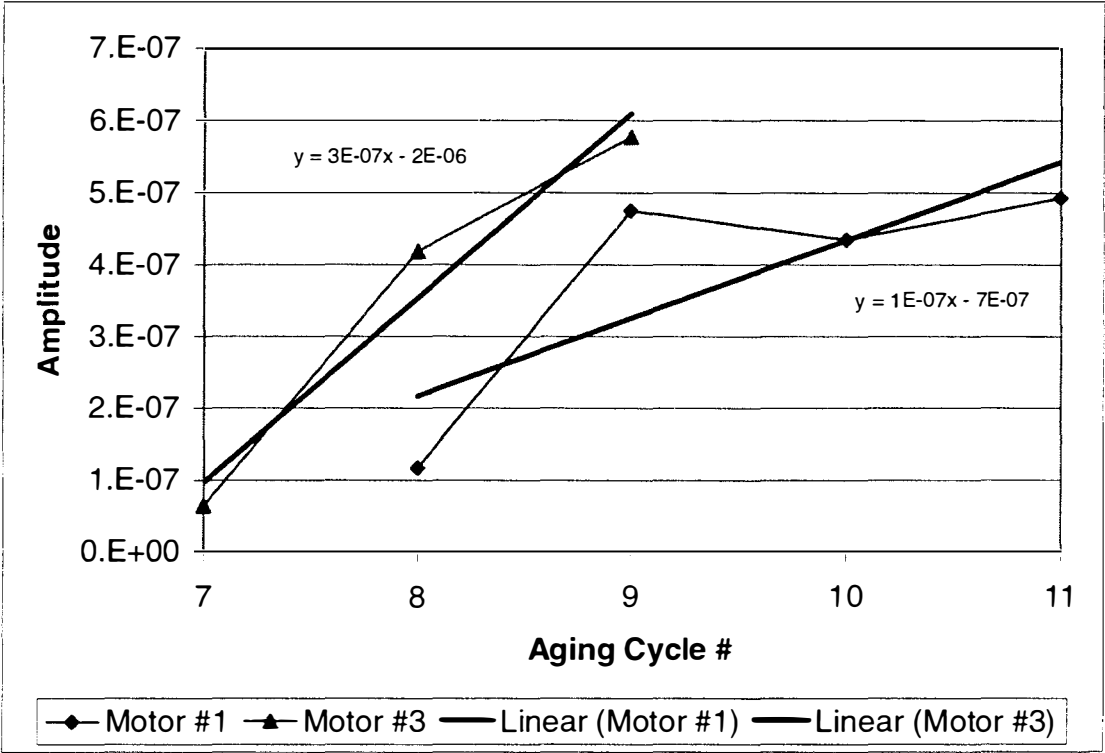


Figure 6.38: Axial magnetic flux RPM frequency amplitude at 100% load.

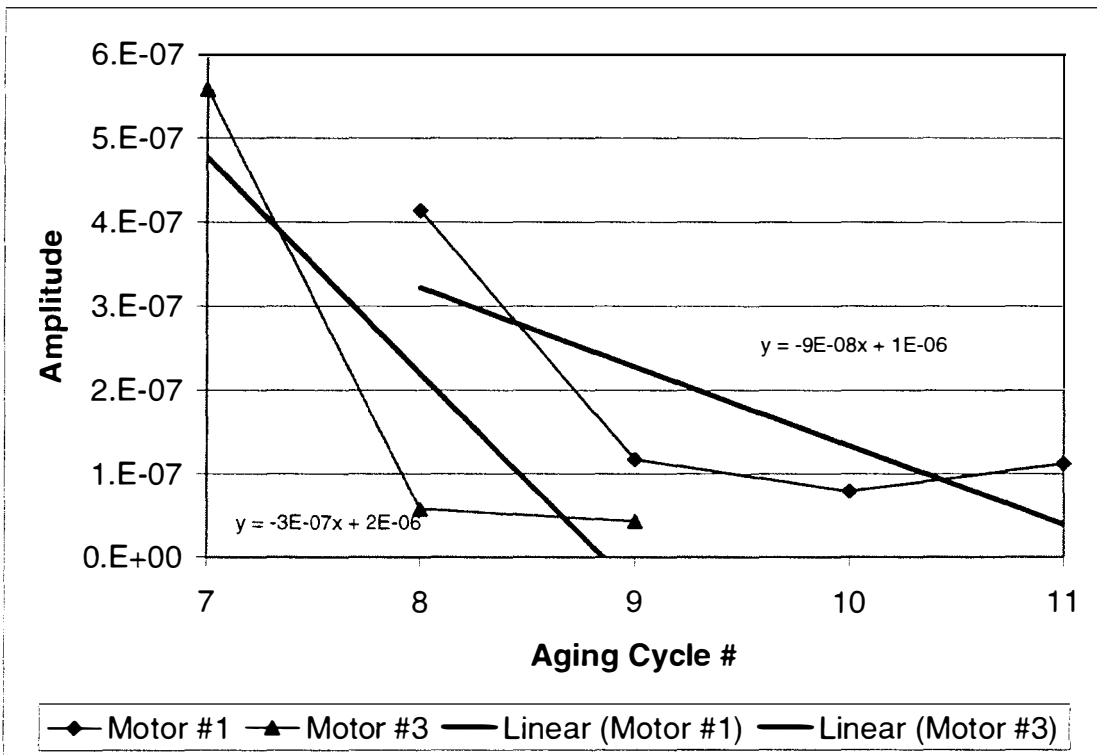


Figure 6.39: Radial magnetic flux RPM frequency amplitude at 100% load.

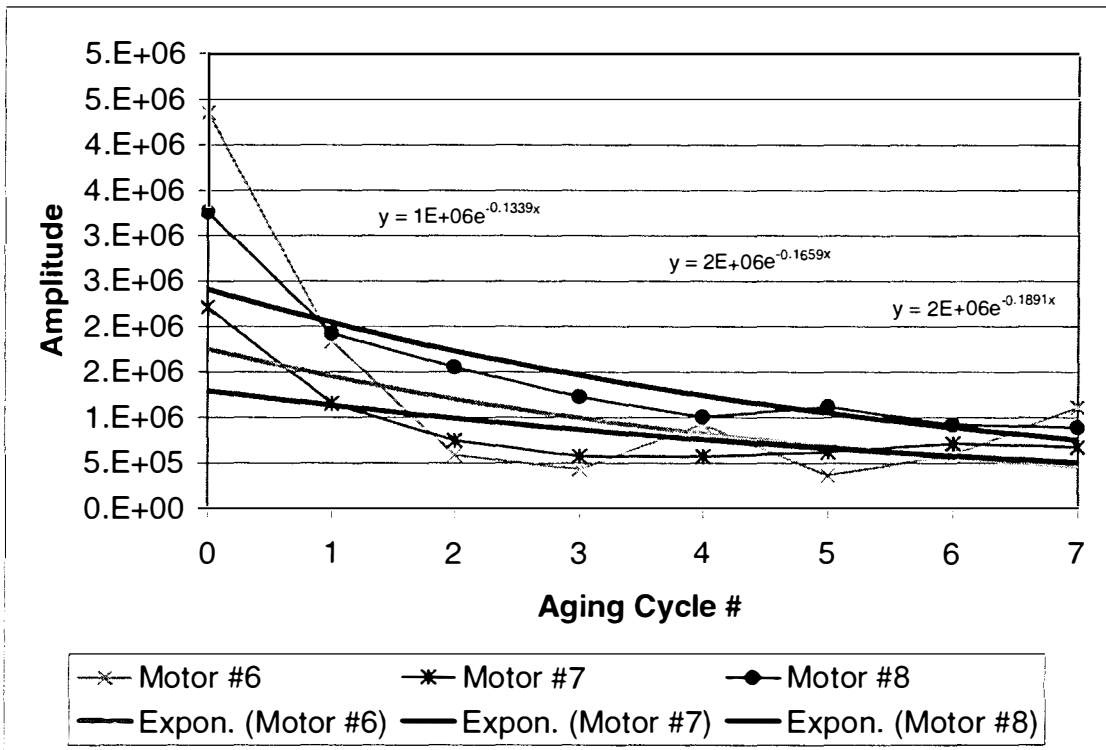


Figure 6.40: Motor power derivative RPM frequency amplitude at 100% load.

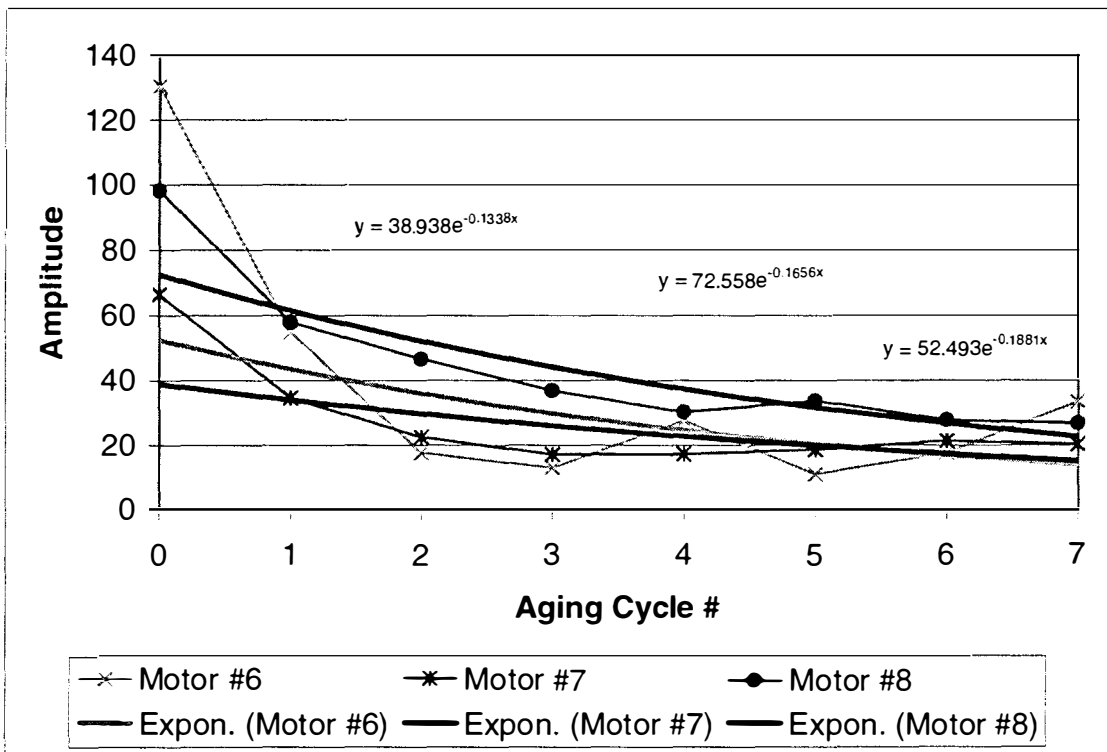


Figure 6.41: Motor power RPM frequency amplitude at 100% load.

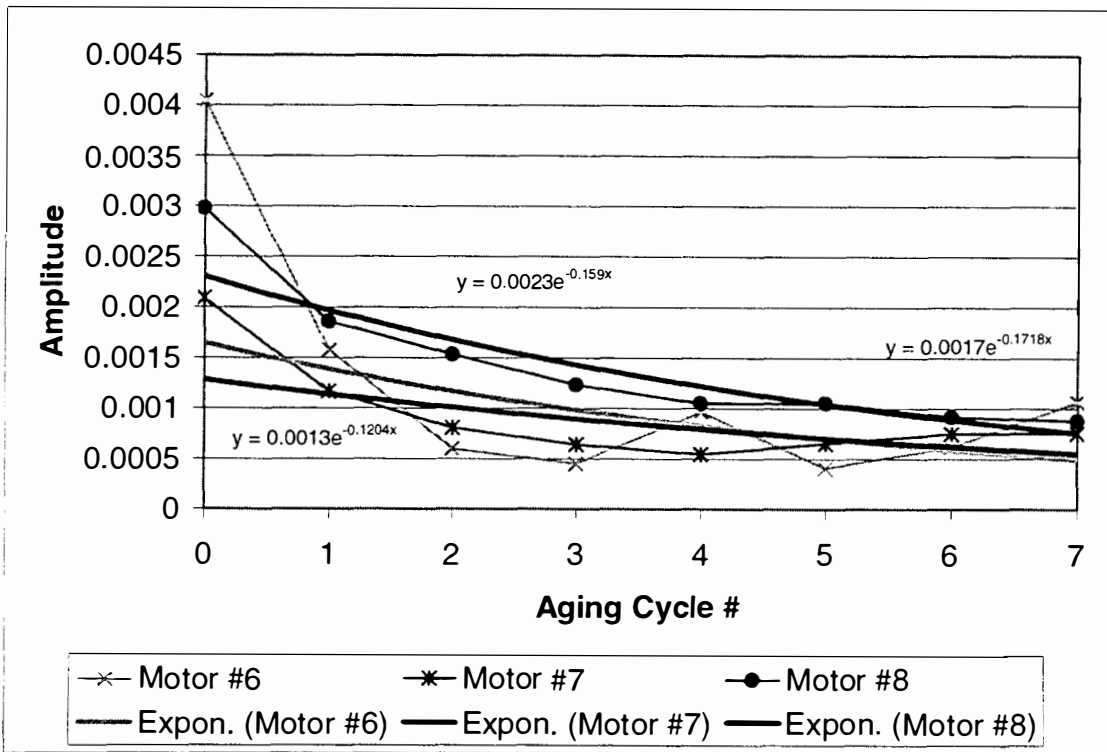


Figure 6.42: Motor work RPM frequency amplitude at 100% load.

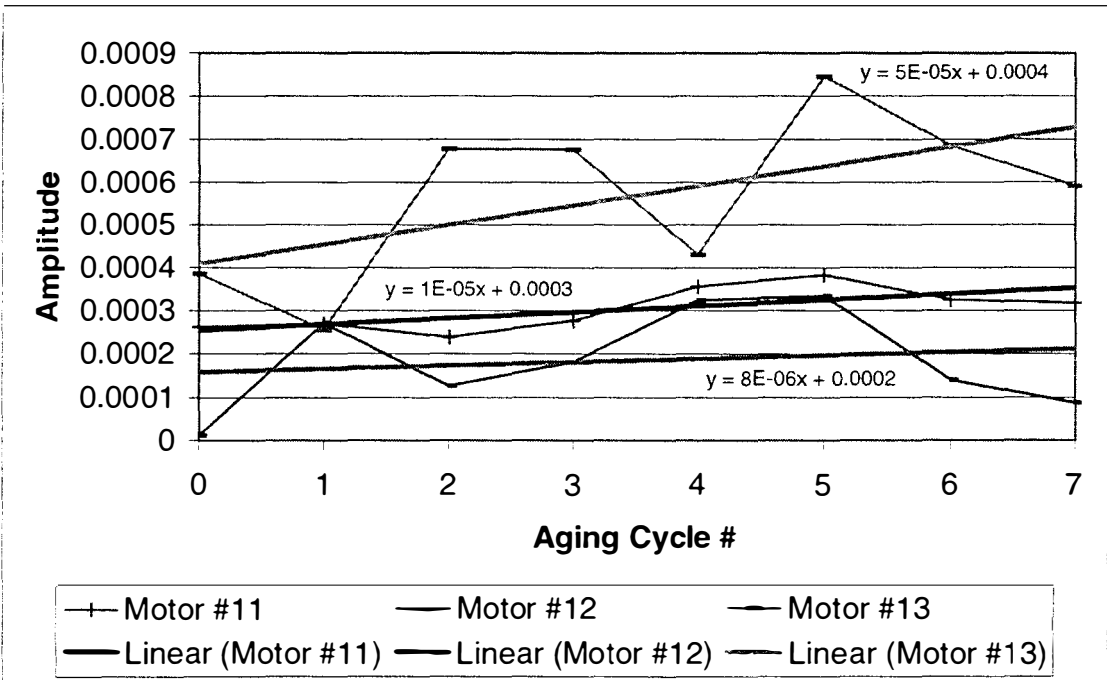
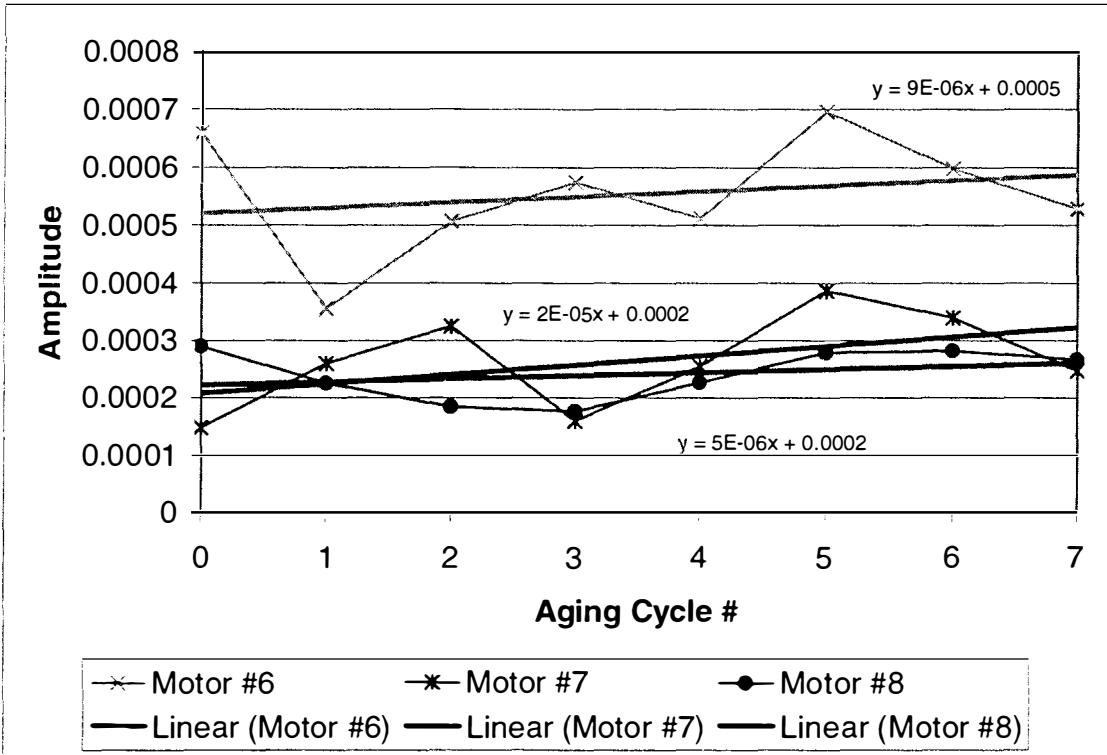


Figure 6.43: Motor PE vibration RPM frequency amplitude at 100% load.

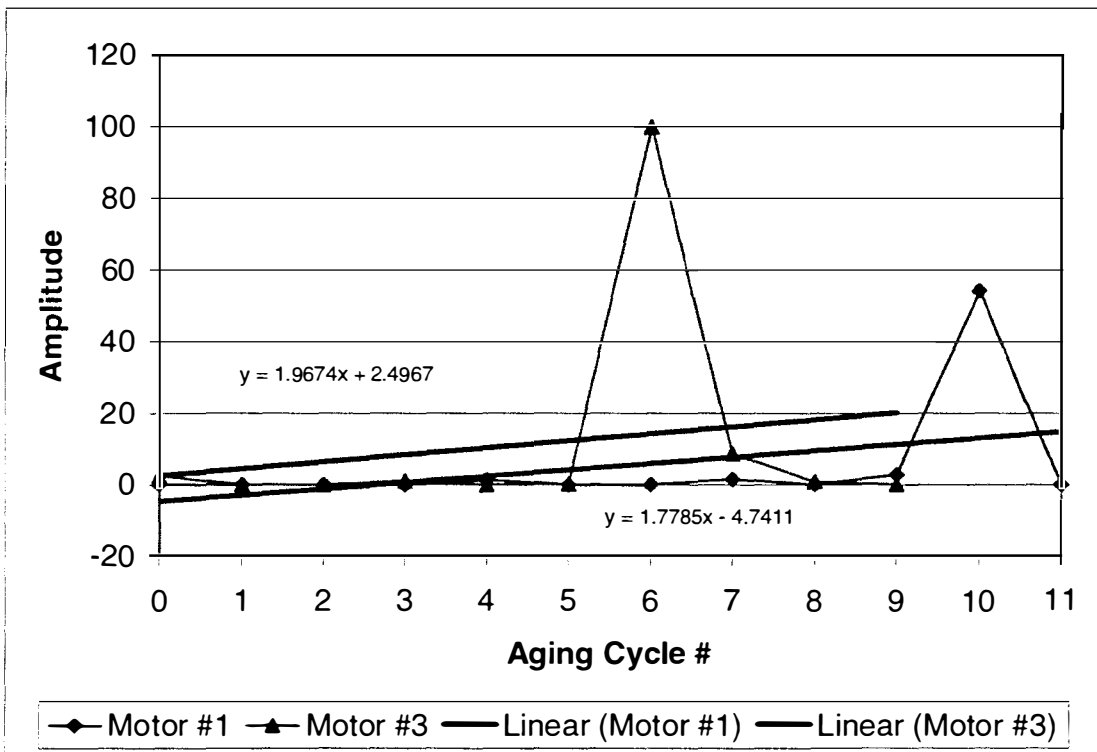


Figure 6.44: Motor zero component impedance slip frequency amplitude at 100% load.

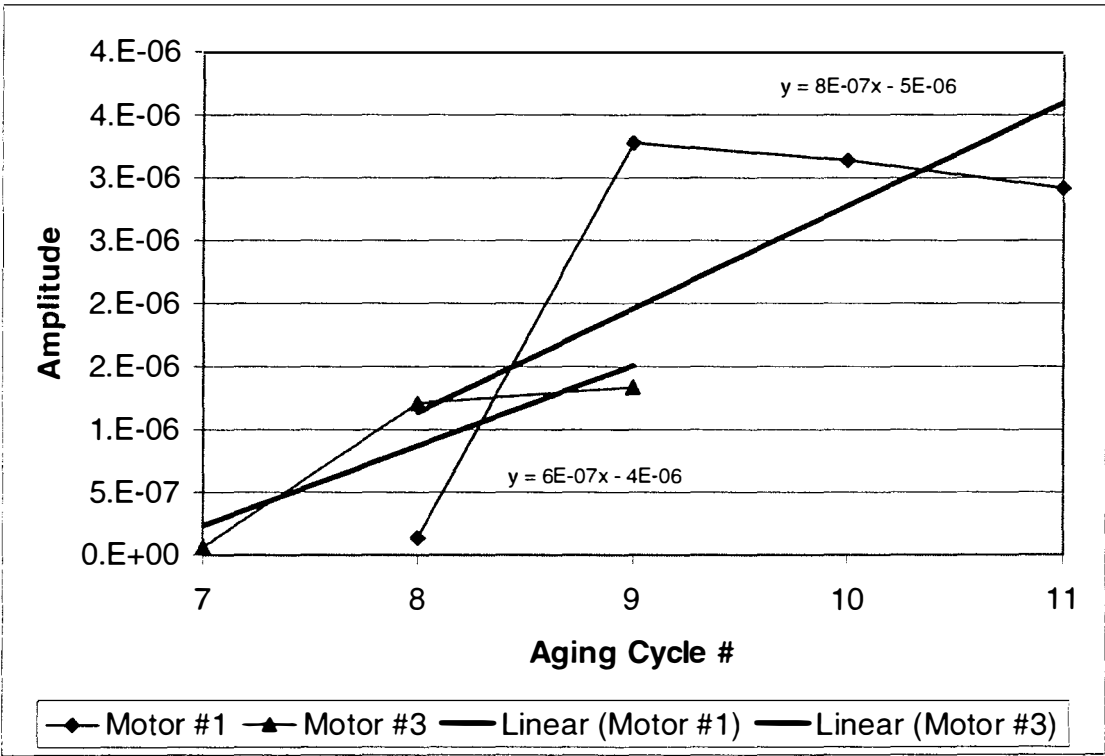


Figure 6.45: Axial magnetic flux slip frequency amplitude at 100% load.

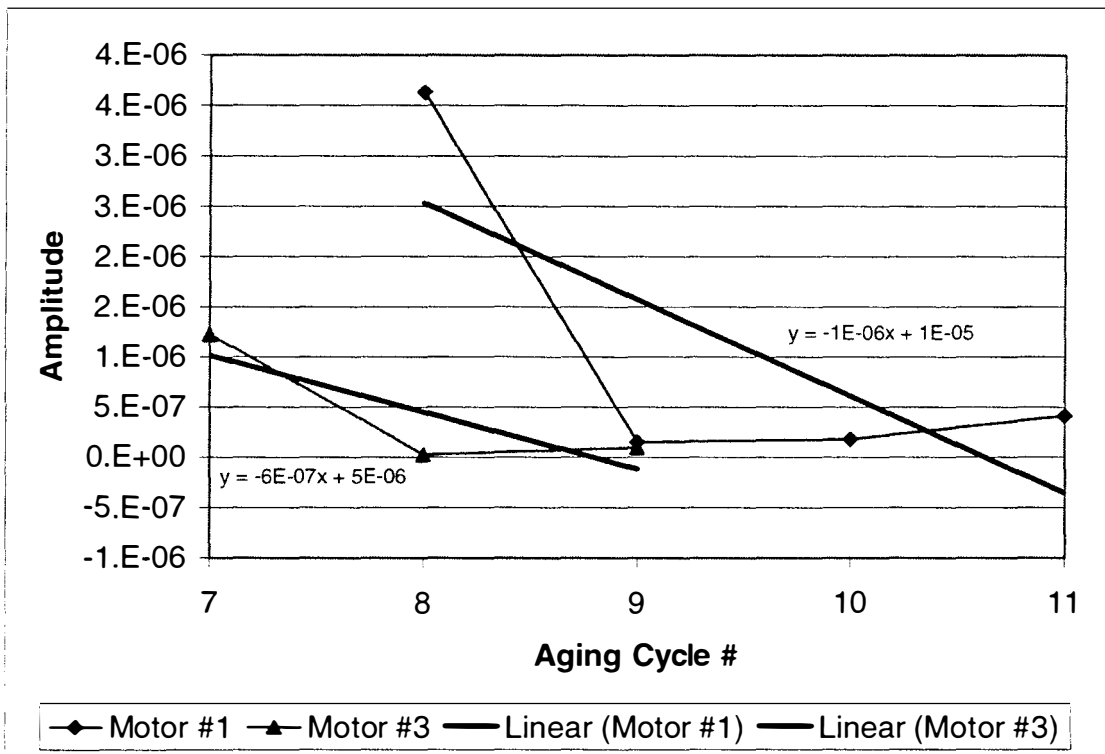


Figure 6.46: Radial magnetic flux slip frequency amplitude at 100% load.

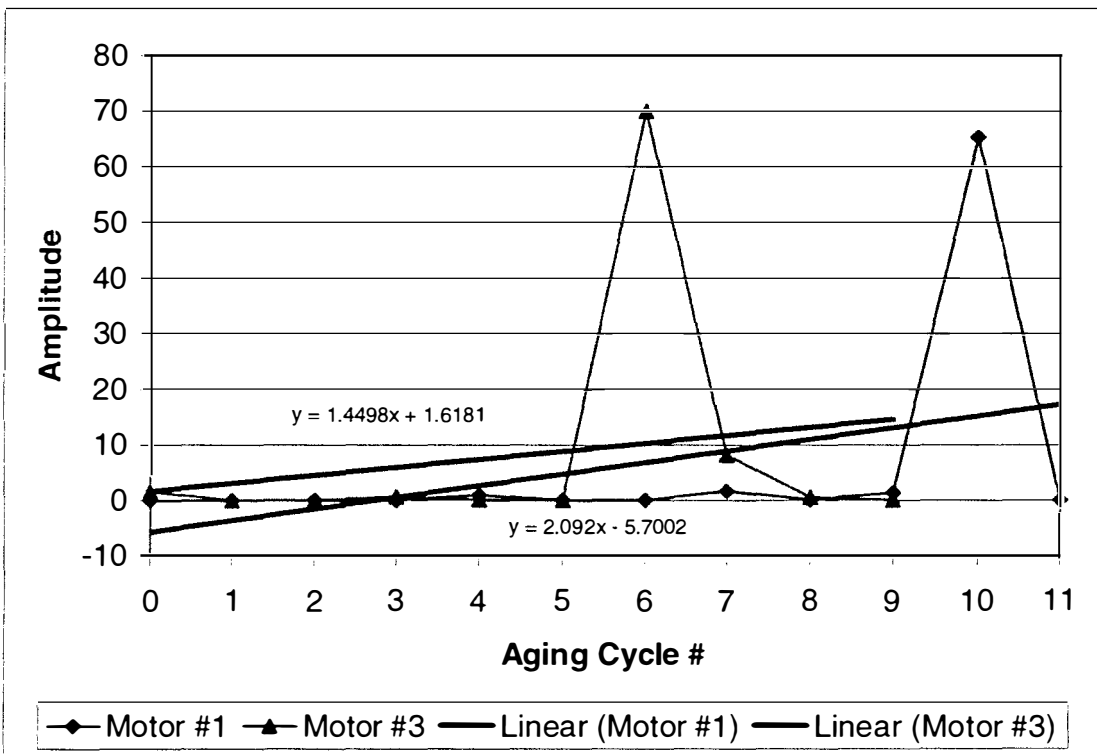


Figure 6.47: Motor zero component impedance slip * number of poles frequency amplitude at 100% load.

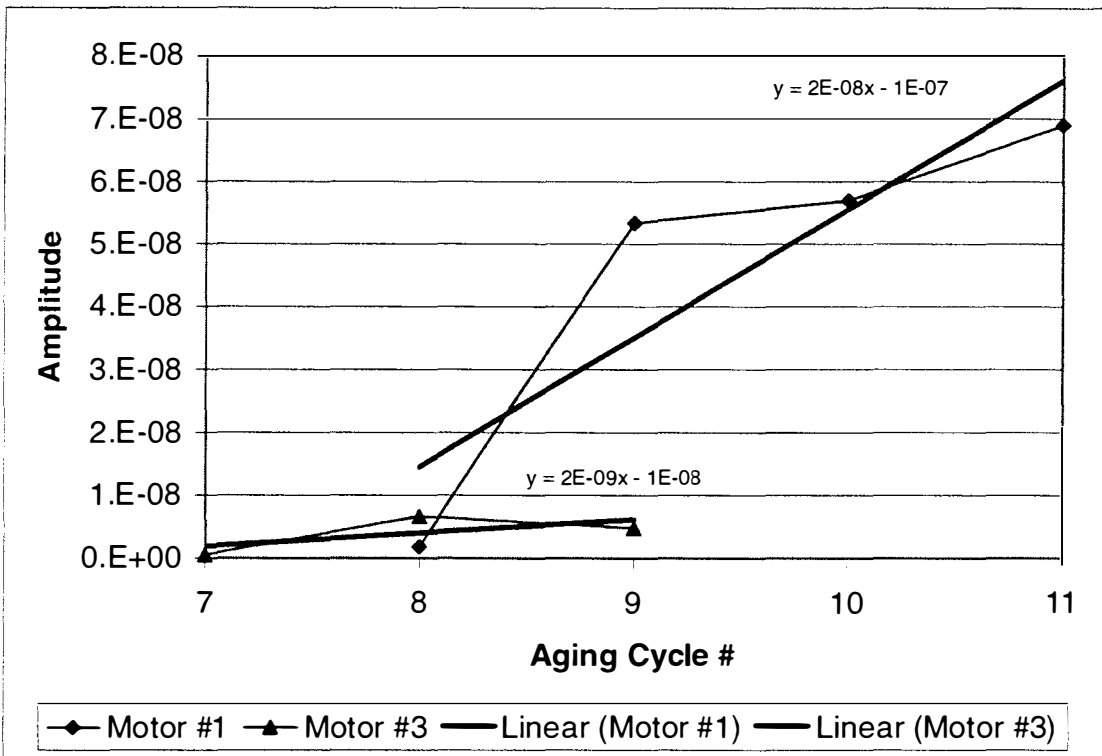


Figure 6.48: Axial magnetic flux slip * number of poles frequency amplitude at 100% load.

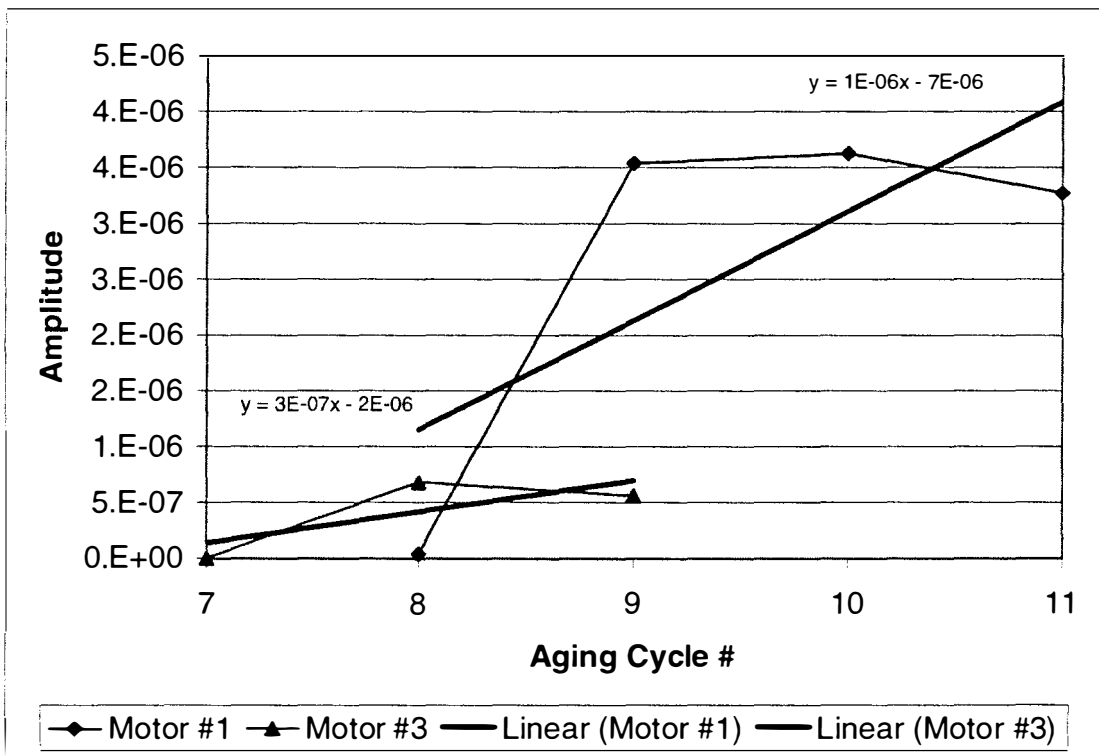


Figure 6.49: Axial magnetic flux slip * number of poles frequency amplitude at 75% load.

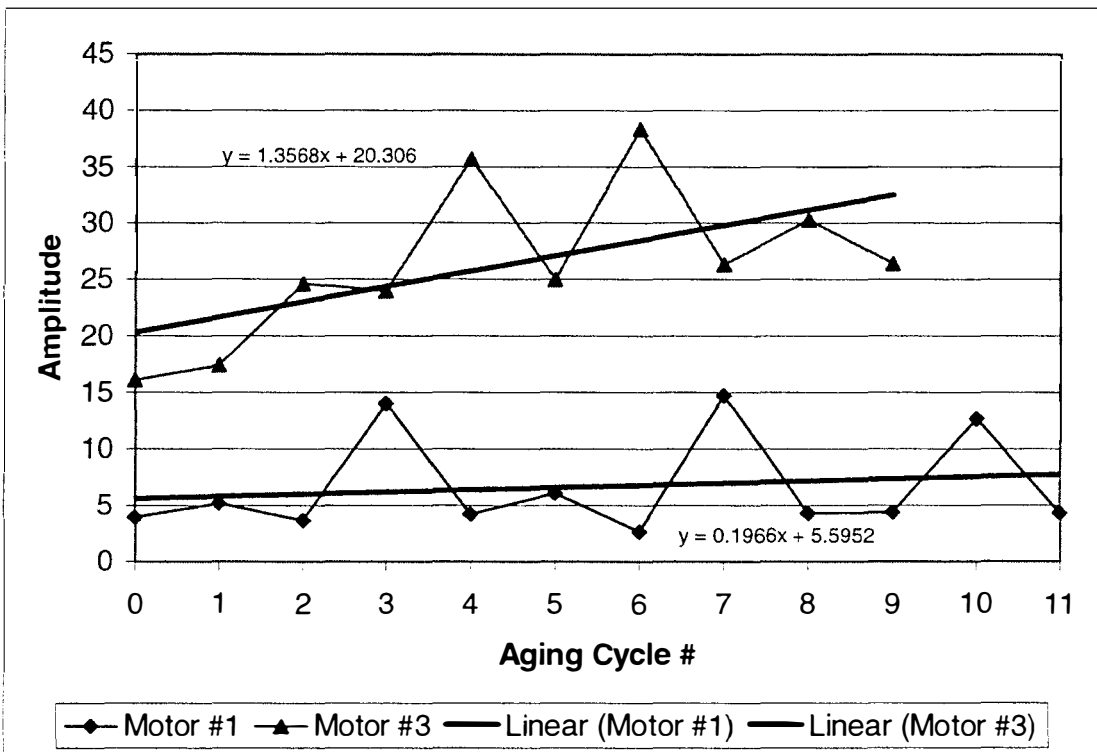


Figure 6.50: Motor power slip * number of poles frequency amplitude at 100% load.

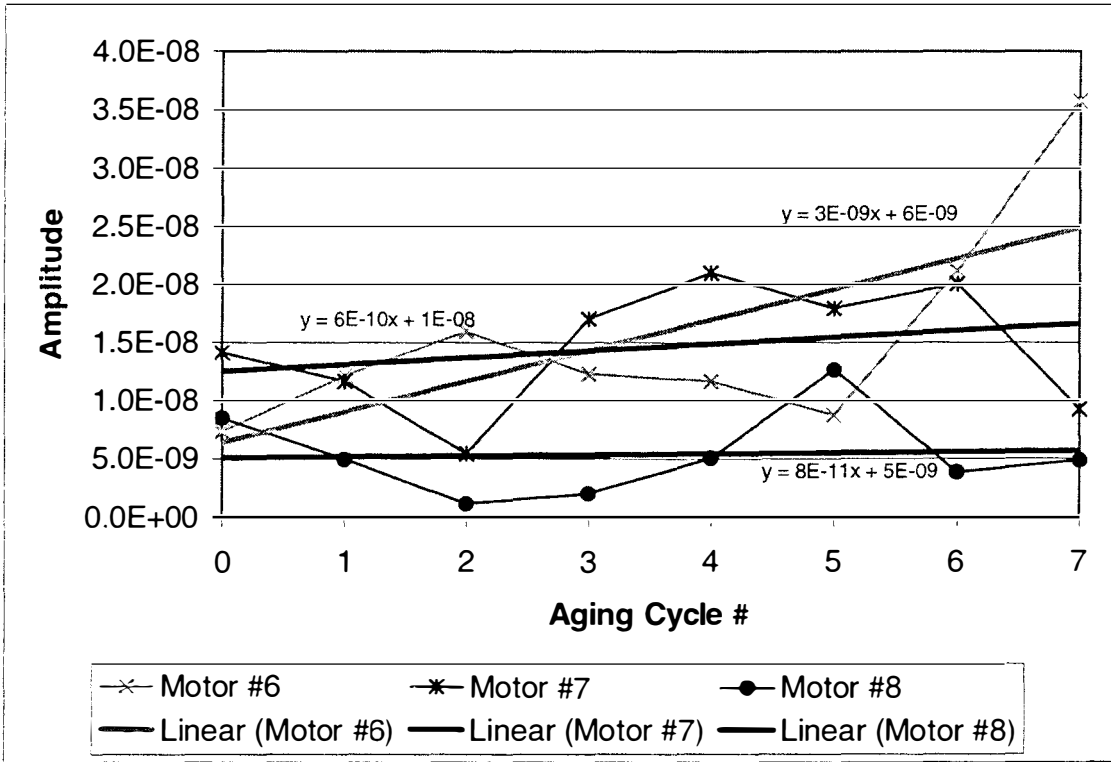


Figure 6.51: Motor PE vibration slip * number of poles frequency amplitude at 100% load.

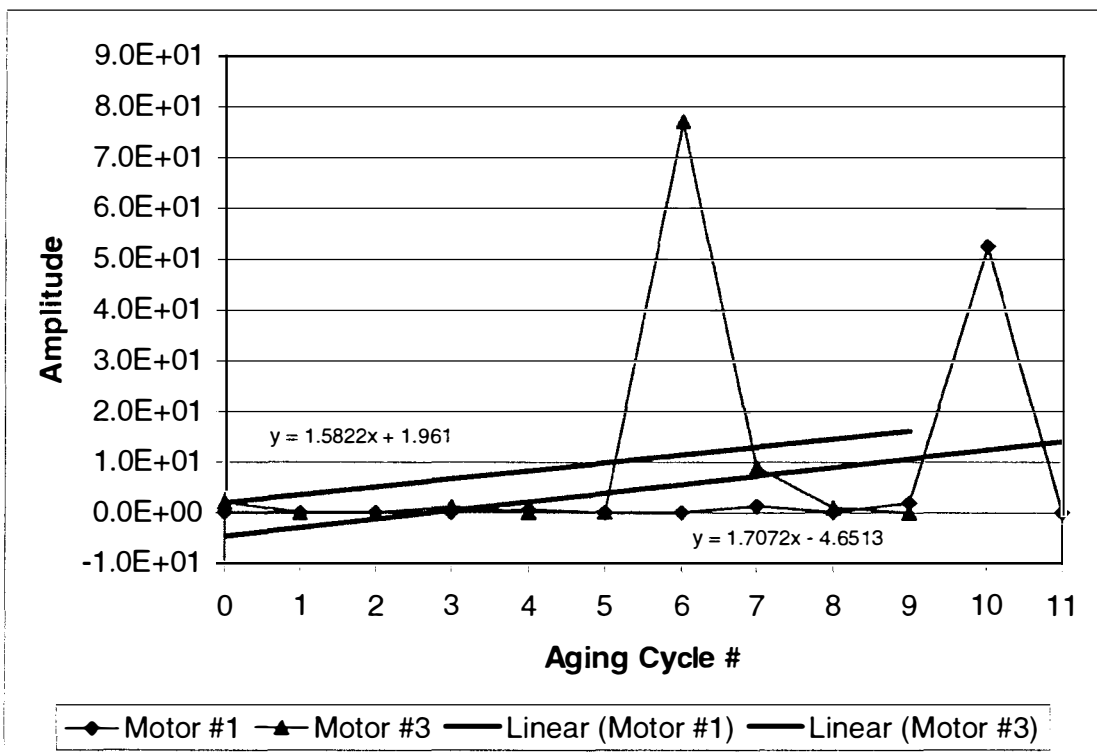


Figure 6.52: Motor zero component impedance slip * number of poles / 2 frequency amplitude at 100% load.

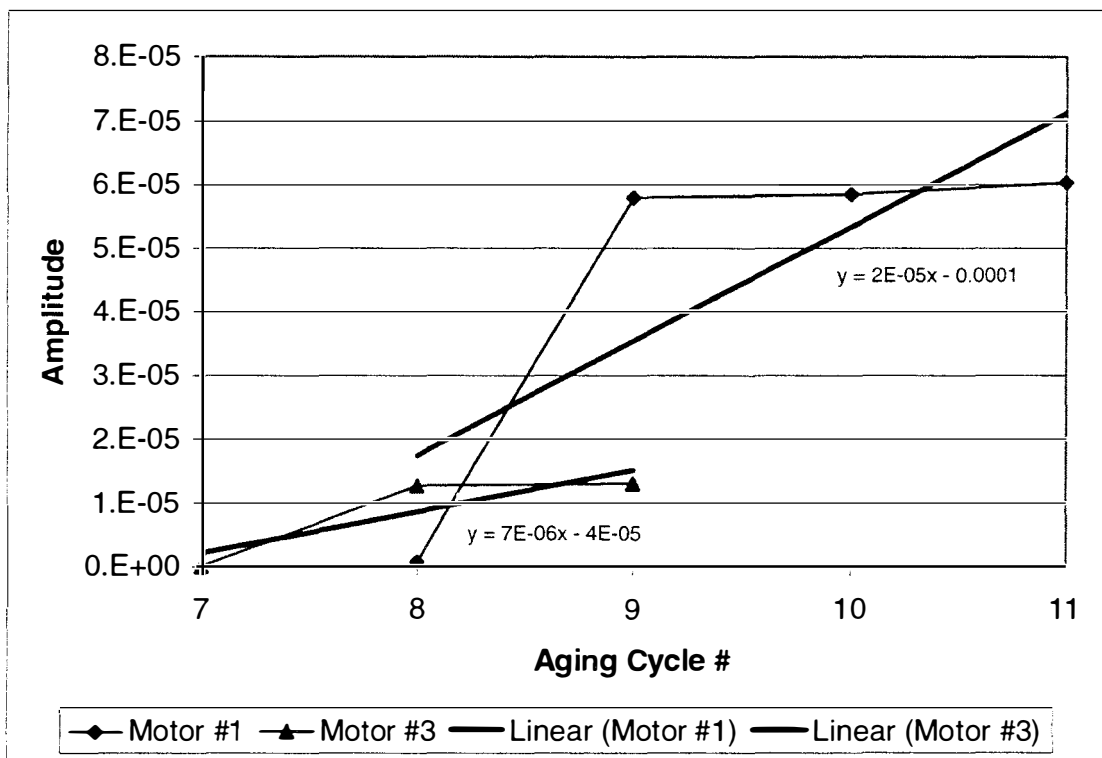


Figure 6.53: Axial magnetic flux slip * number of poles / 2 frequency amplitude at 100% load.

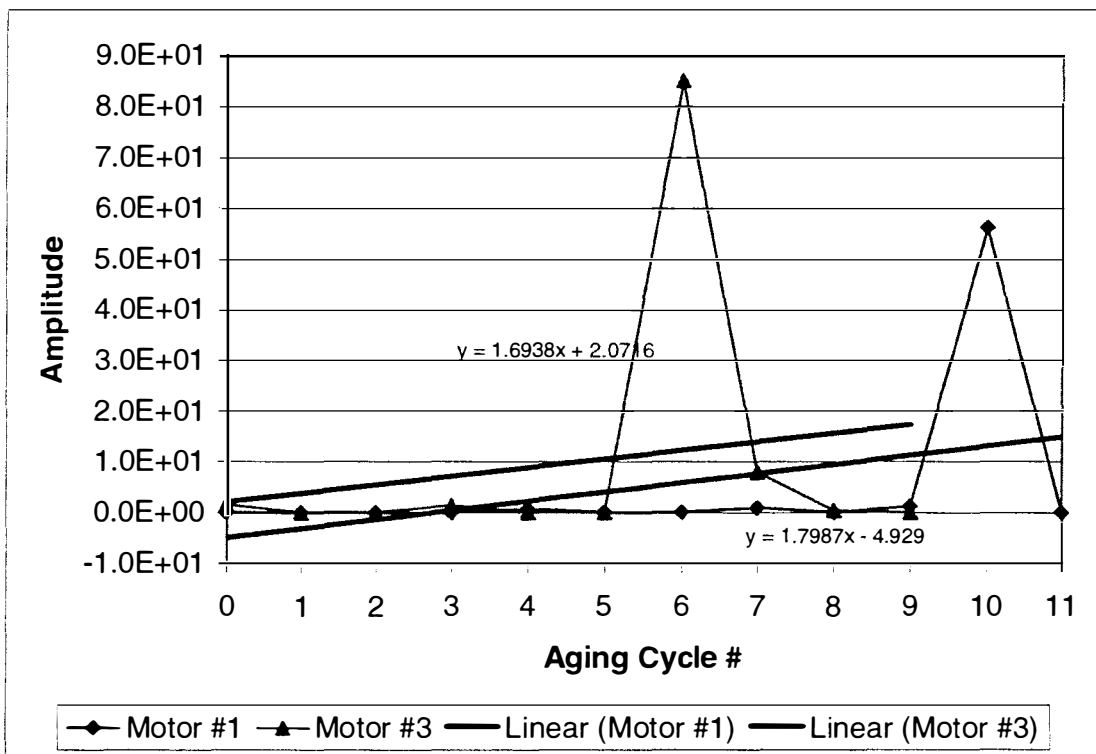


Figure 6.54: Motor zero component impedance $(1 + 2 * \text{slip}) * f_e$ frequency amplitude at 100% load.

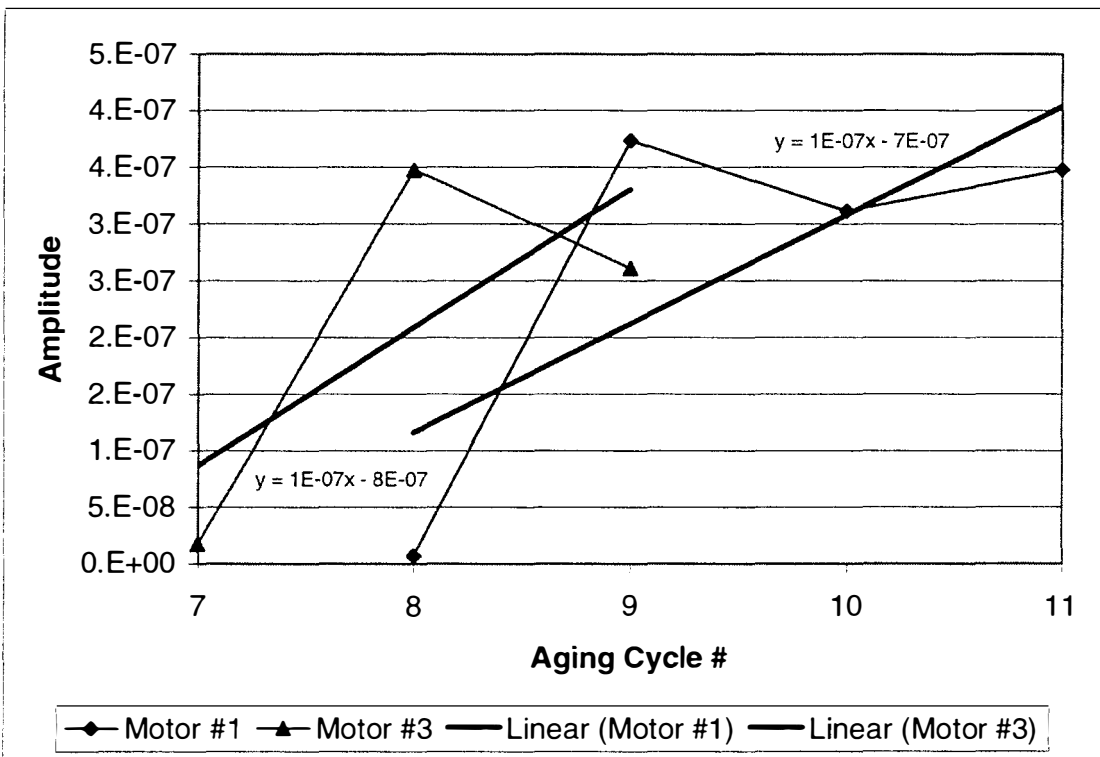


Figure 6.55: Radial magnetic flux $(1 + 2 * \text{slip}) * f_e$ frequency amplitude at 100% load.

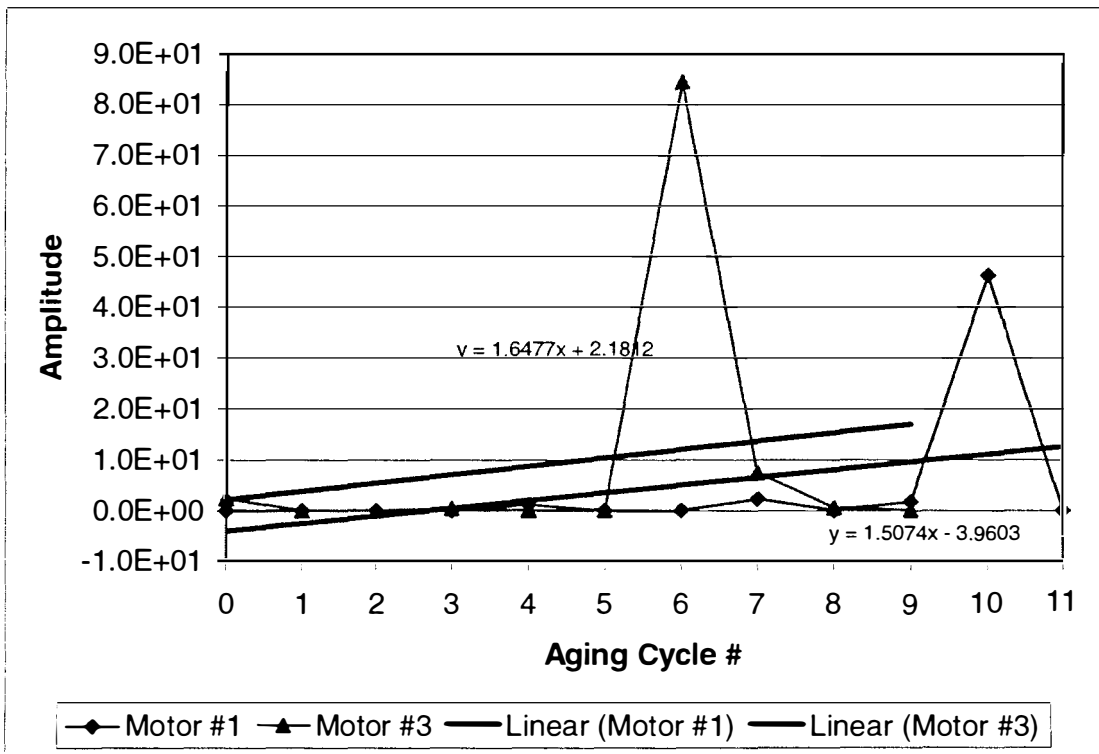


Figure 6.56: Motor zero component impedance $(1 - 2 * \text{slip}) * f_e$ frequency amplitude at 100% load.

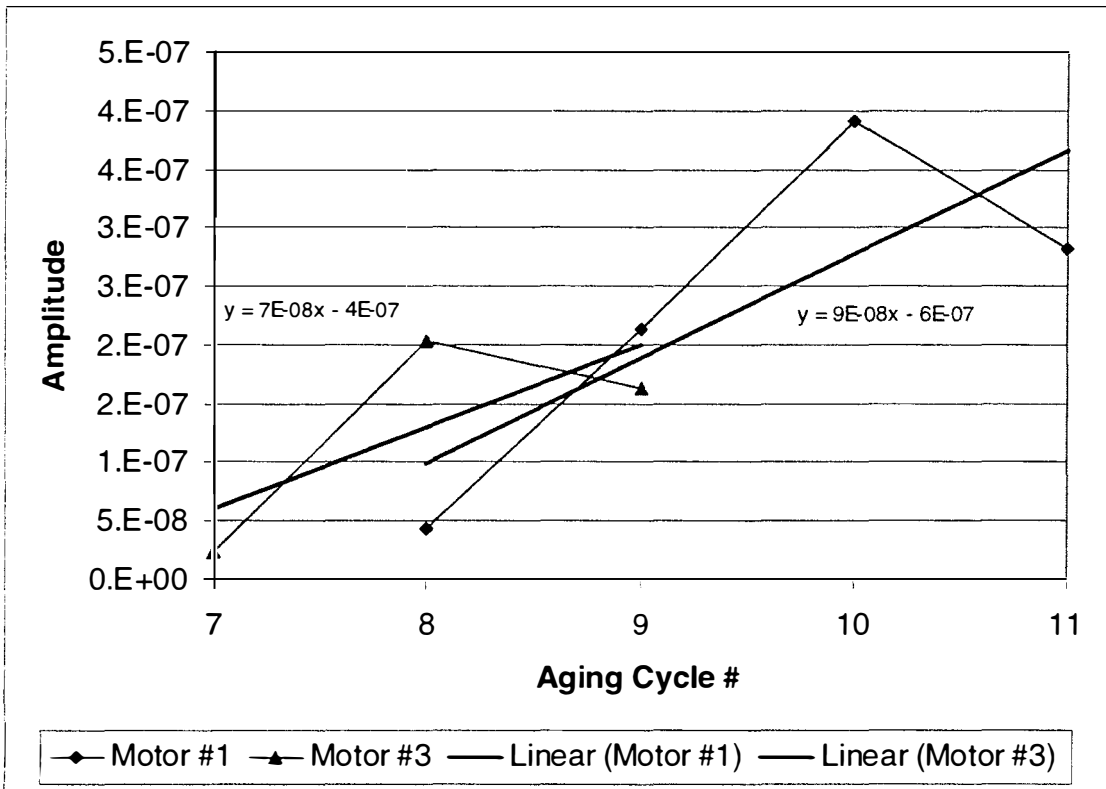


Figure 6.57: Radial magnetic flux $(1 - 2 * \text{slip}) * f_e$ frequency amplitude at 100% load.

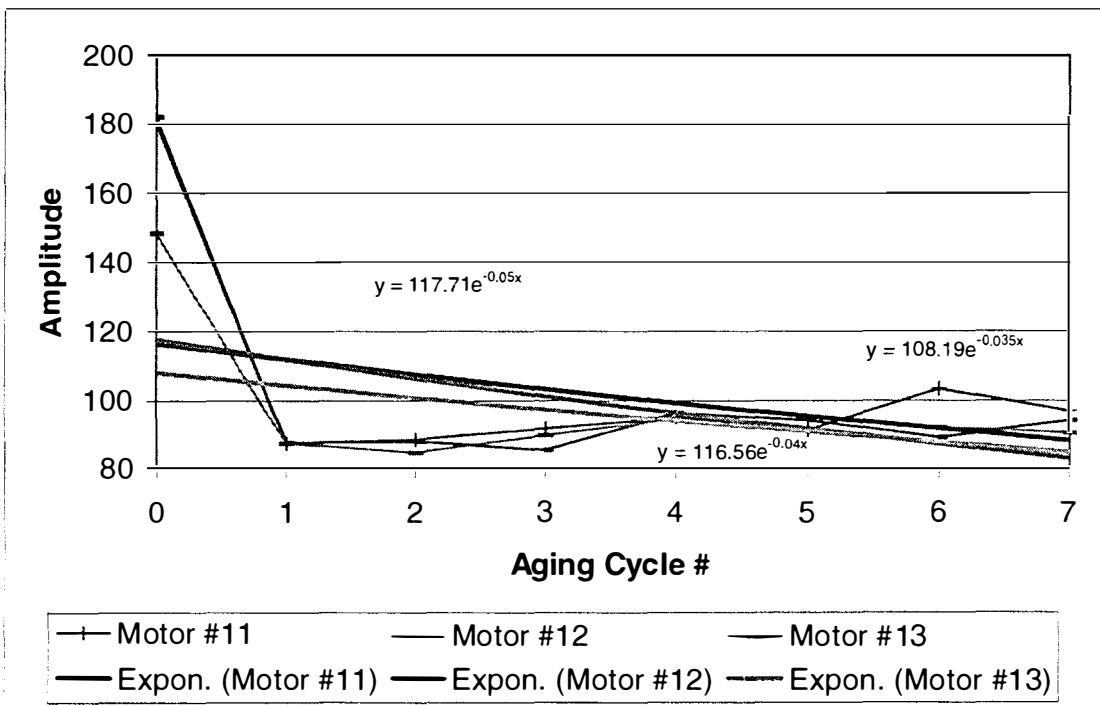
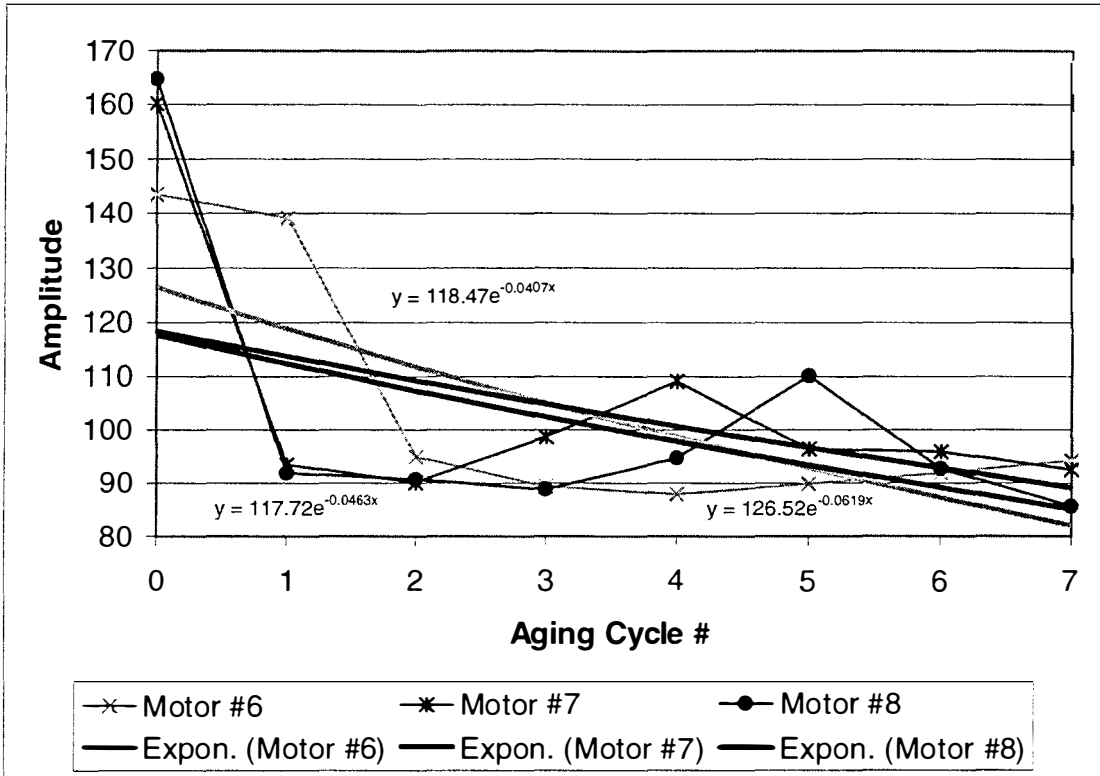


Figure 6.58: Load power line frequency amplitude at 100% load.

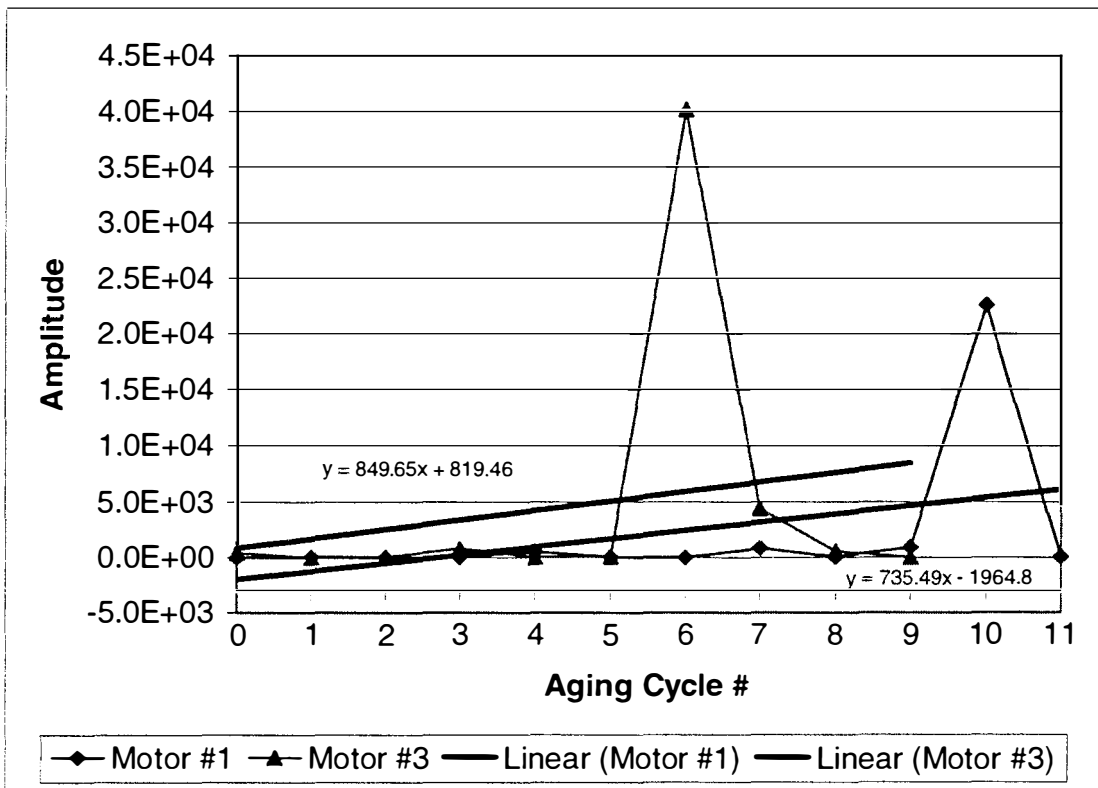


Figure 6.59: Motor zero component impedance line frequency amplitude at 100% load.

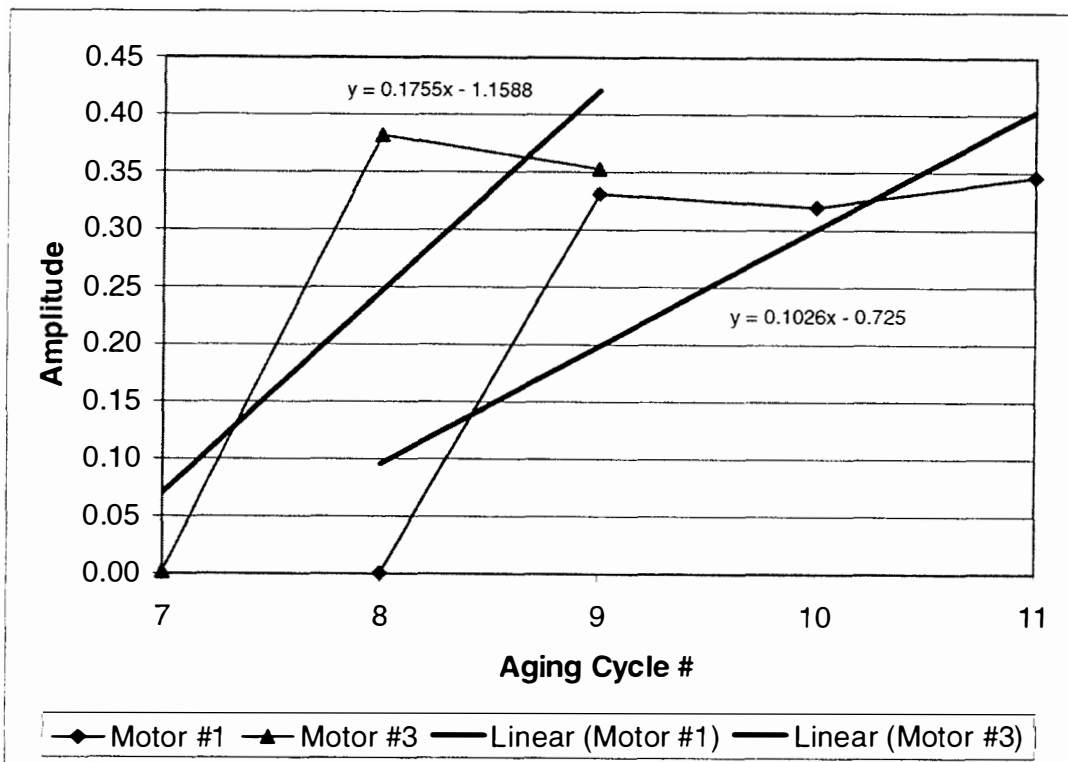


Figure 6.60: Radial magnetic flux line frequency amplitude at 100% load.

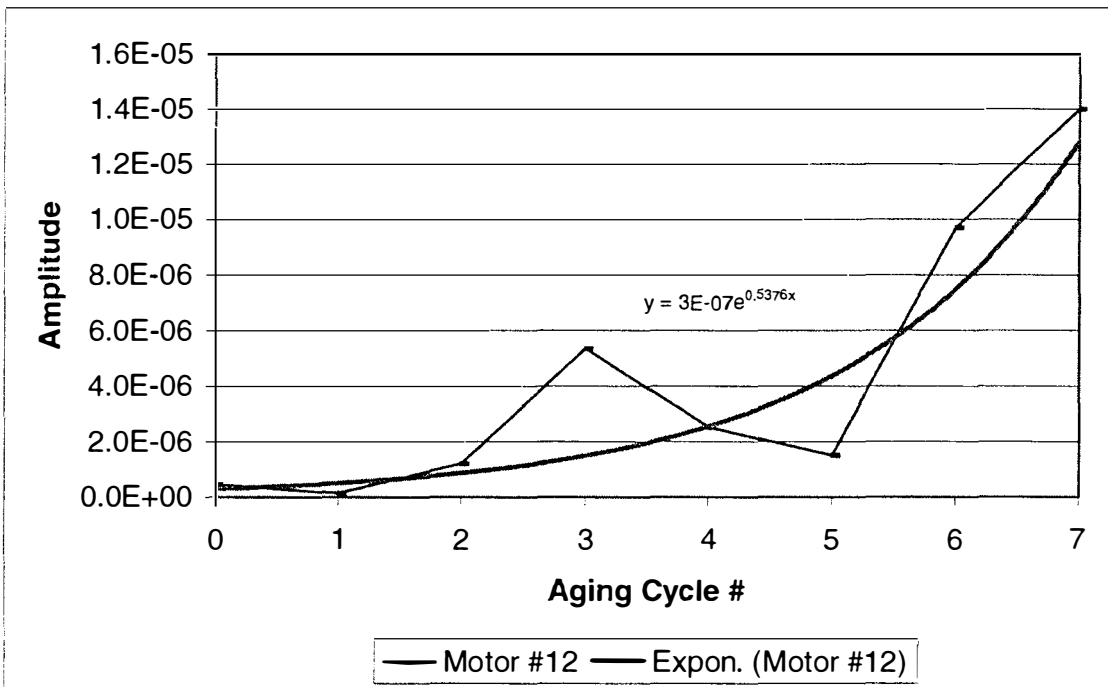


Figure 6.61: Motor PE vibration f_e + SE IR frequency amplitude at 100% load.

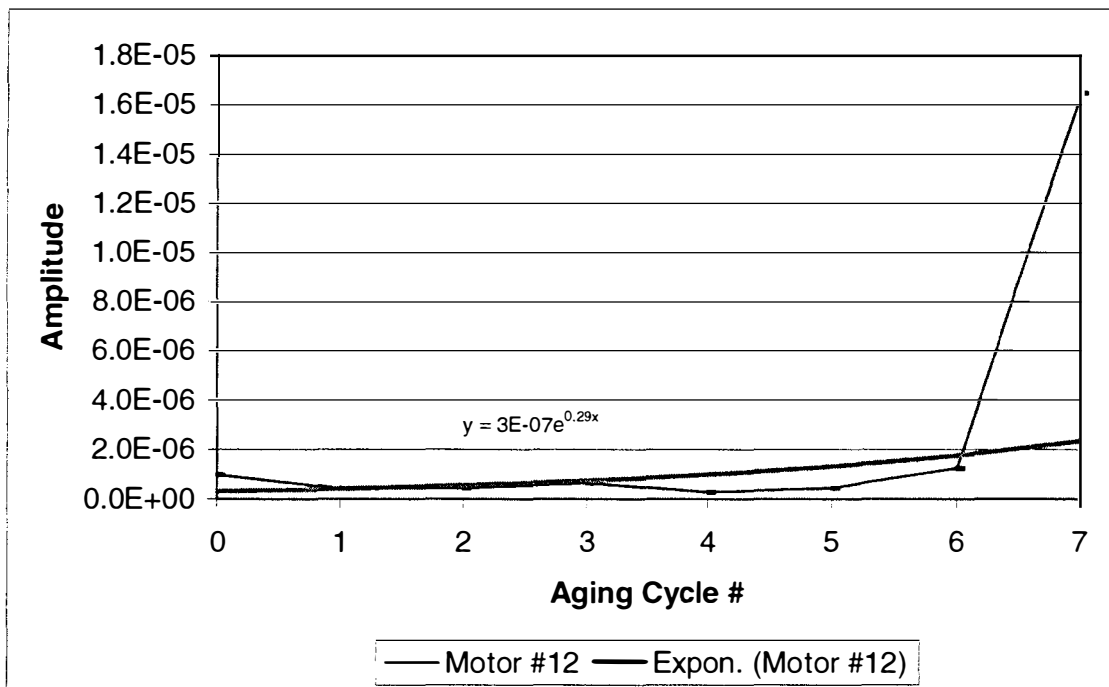


Figure 6.62: Motor cover vibration f_e + SE IR frequency amplitude at 100% load.

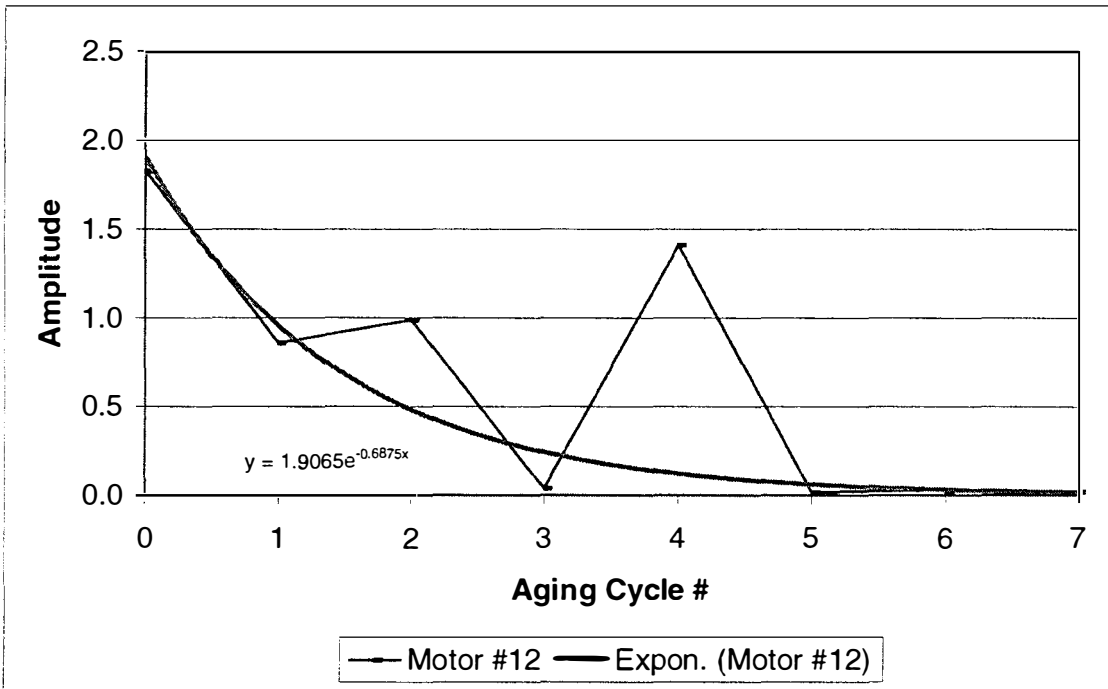


Figure 6.63: Motor power f_e + SE IR frequency amplitude at 100% load.

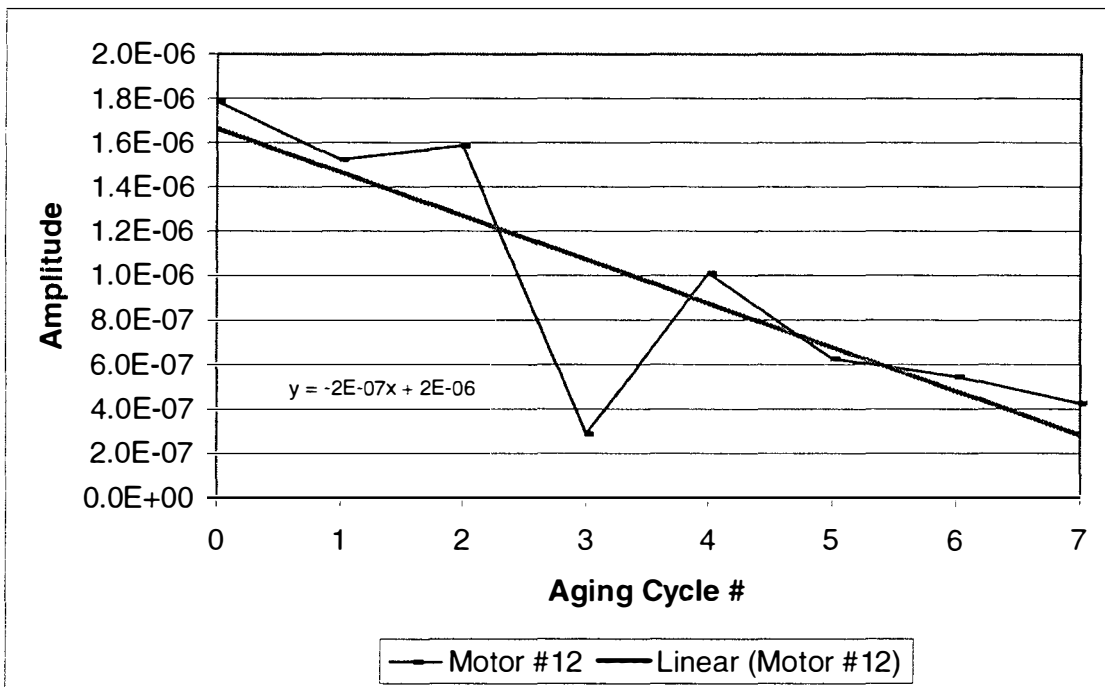


Figure 6.64: Motor current $f_e + SE$ IR frequency amplitude at 100% load.

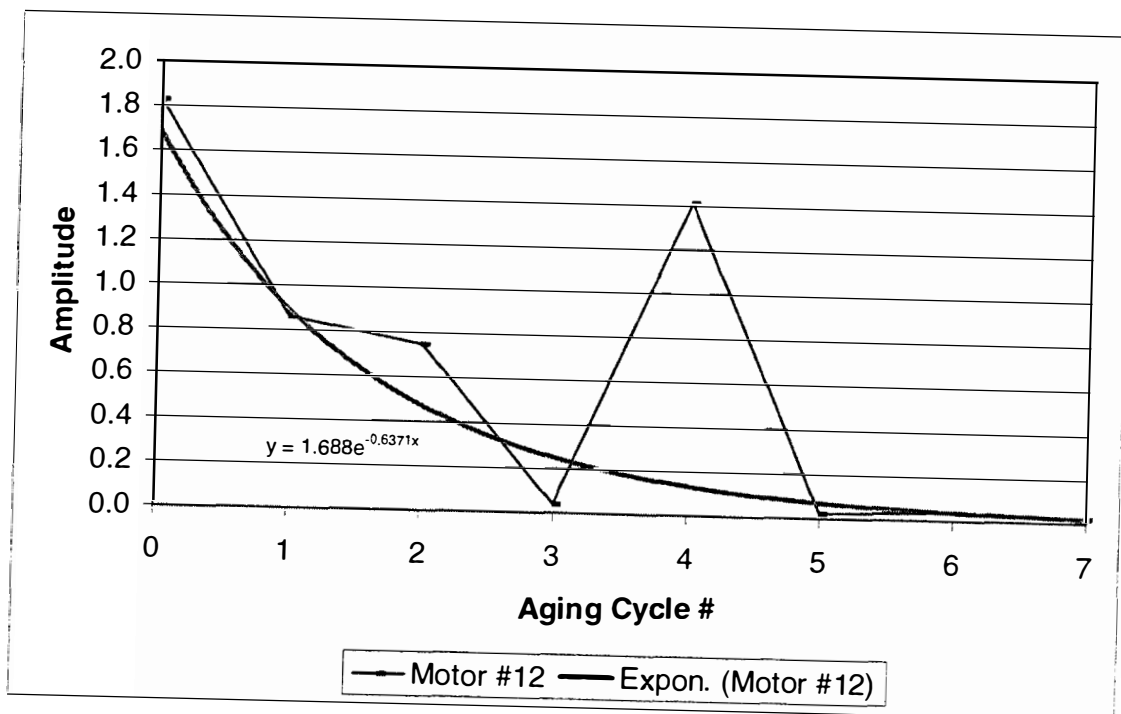


Figure 6.65: Motor power f_e + PE IR frequency amplitude at 100% load.

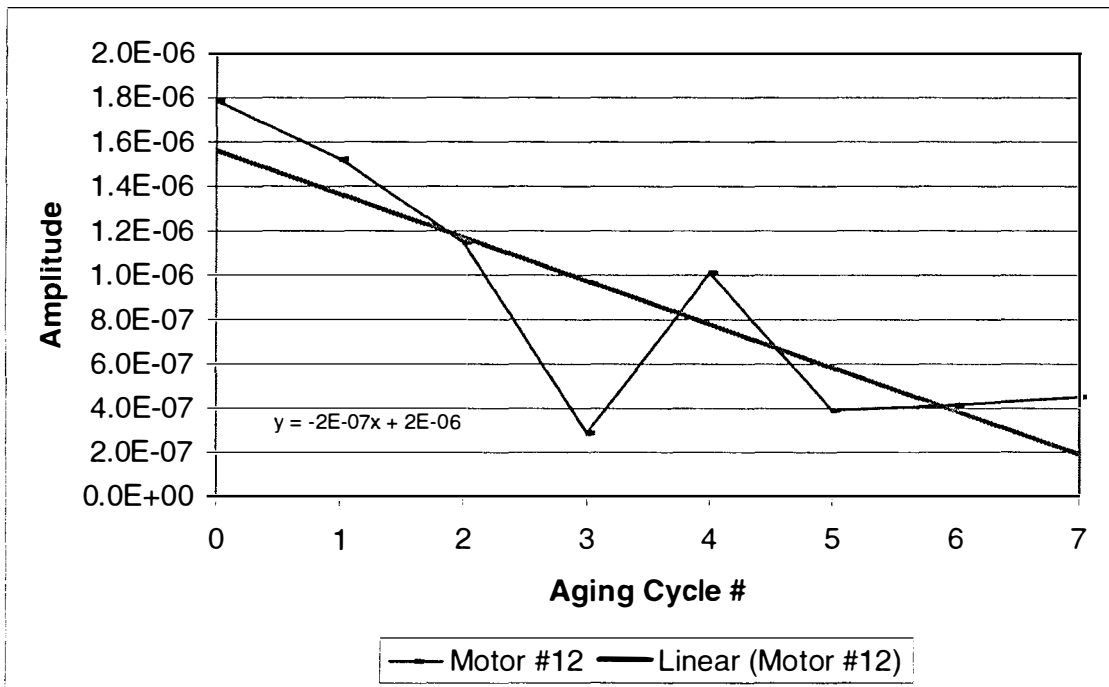


Figure 6.66: Motor current f_e + PE IR frequency amplitude at 100% load.

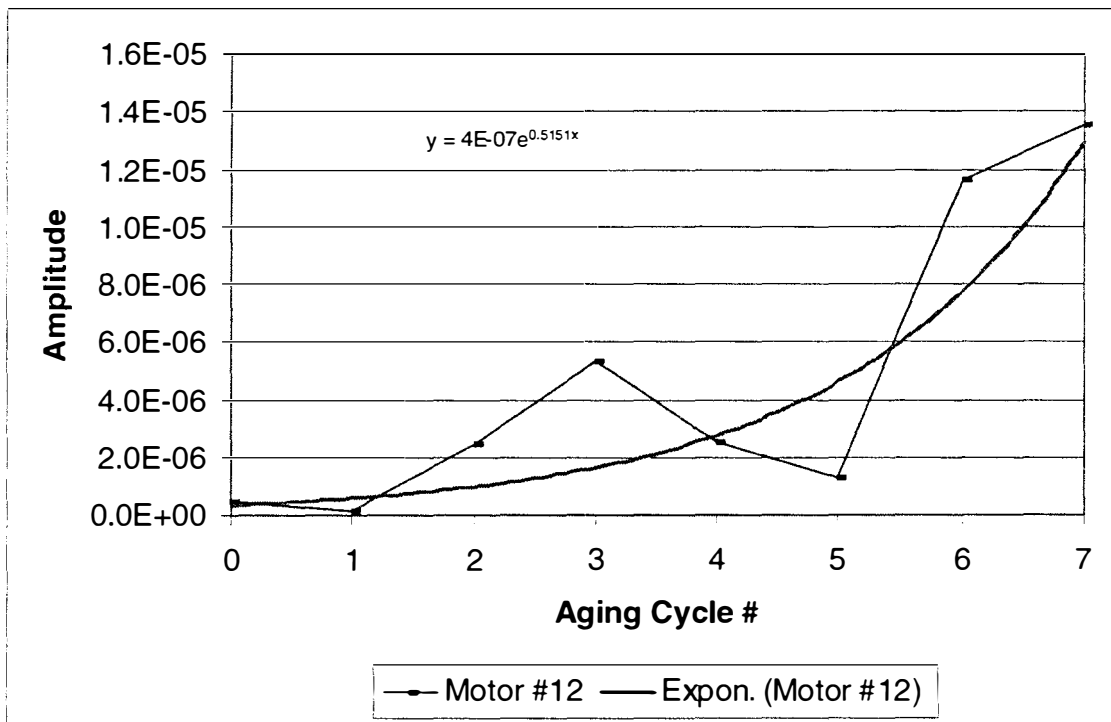


Figure 6.67: Motor PE vibration $f_e + PE\ IR$ frequency amplitude at 100% load.

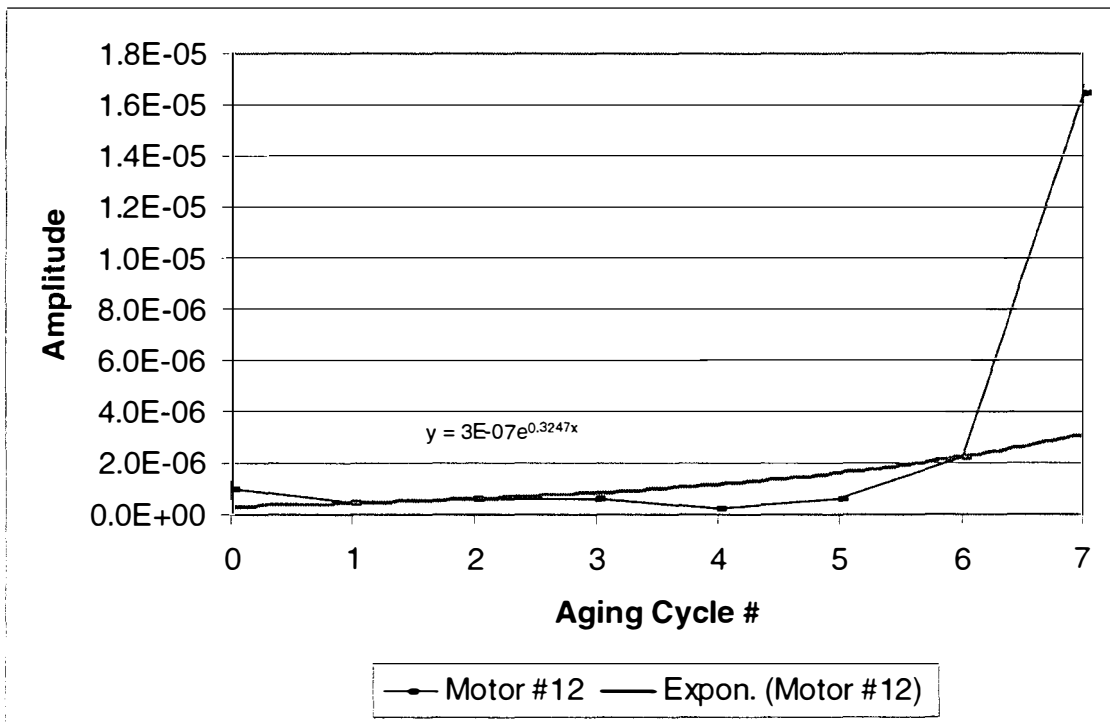


Figure 6.68: Motor cover vibration $f_e + PE$ IR frequency amplitude at 100% load.

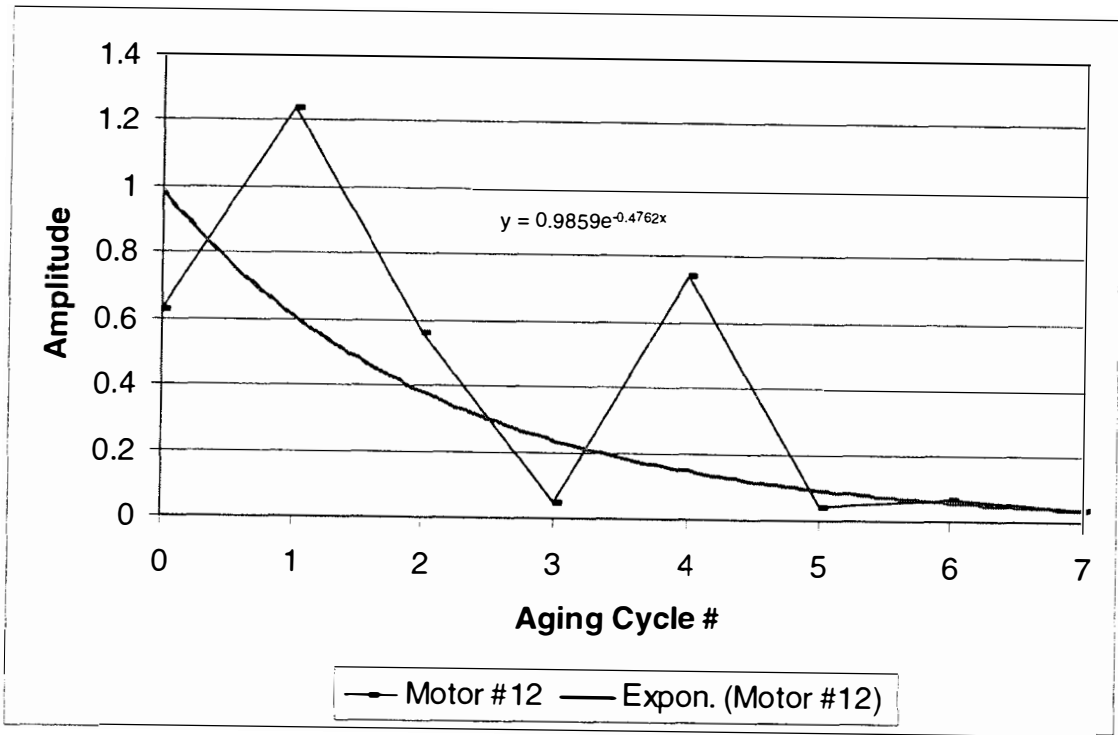


Figure 6.69: Motor power $f_e + SE$ OR frequency amplitude at 100% load.

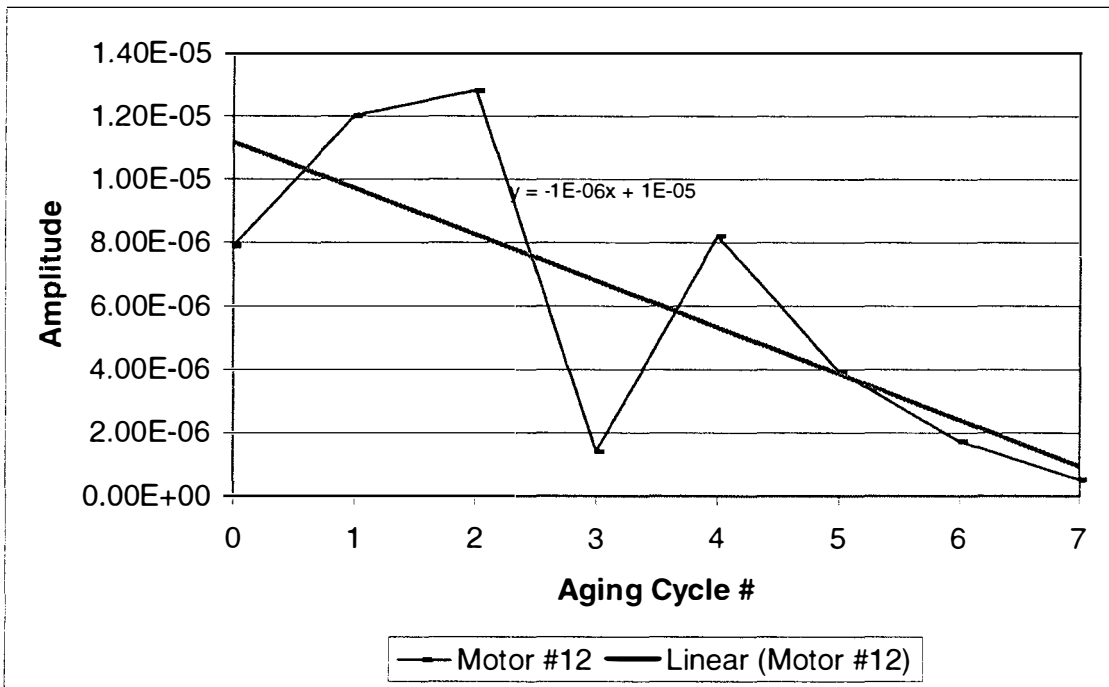


Figure 6.70: Motor current f_e + SE OR frequency amplitude at 100% load.

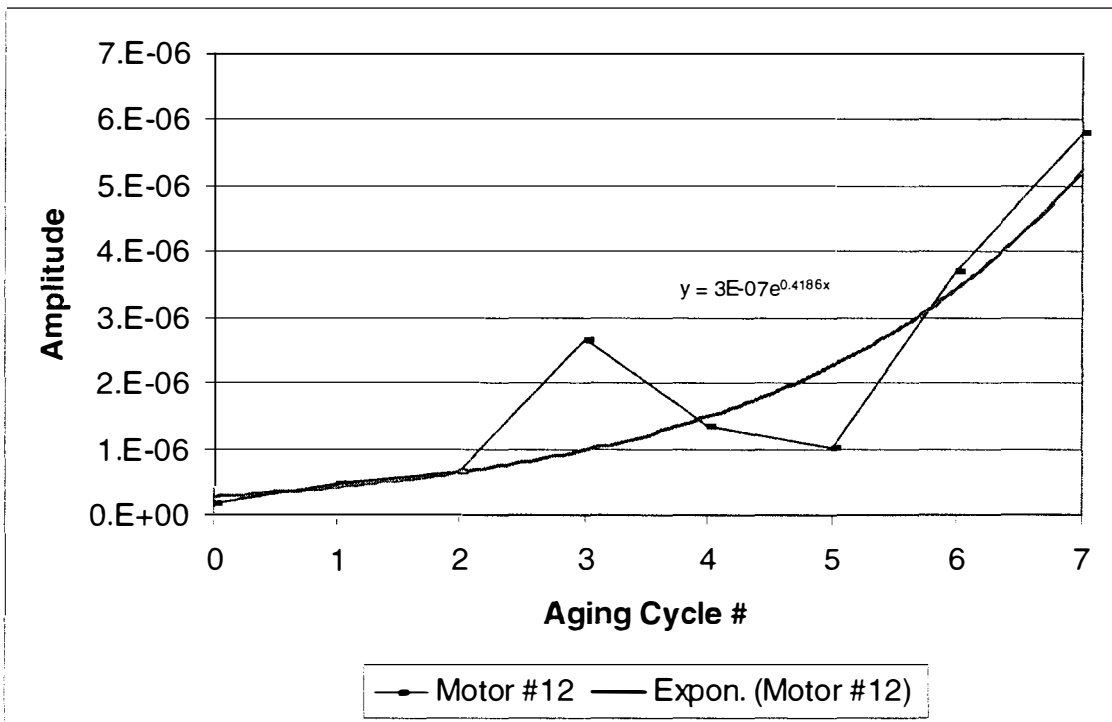


Figure 6.71: Motor PE vibration f_e + SE OR frequency amplitude at 100% load.

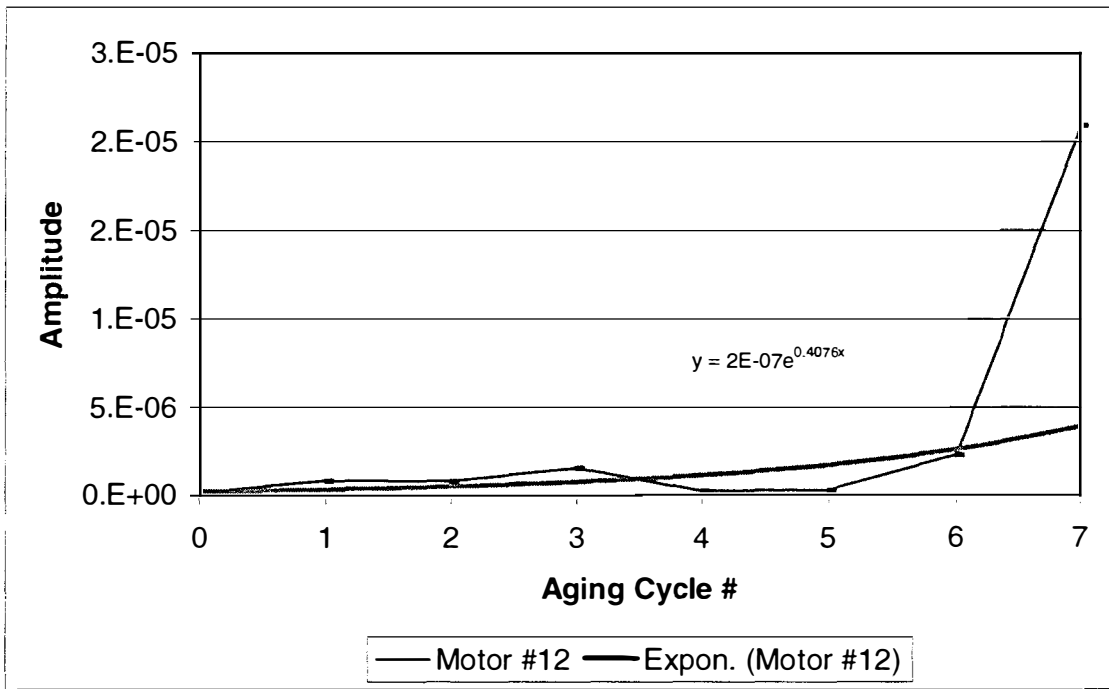


Figure 6.72: Motor cover vibration $f_e + SE$ OR frequency amplitude at 100% load.

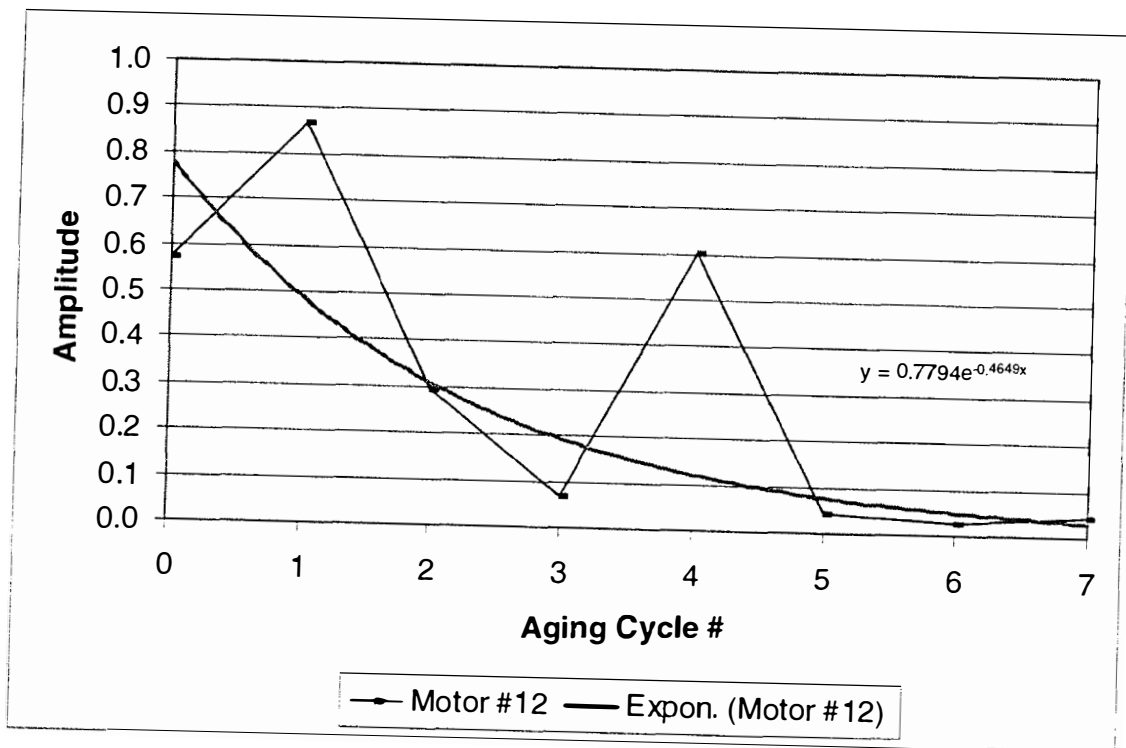


Figure 6.73: Motor power $f_e + PE$ OR frequency amplitude at 100% load.

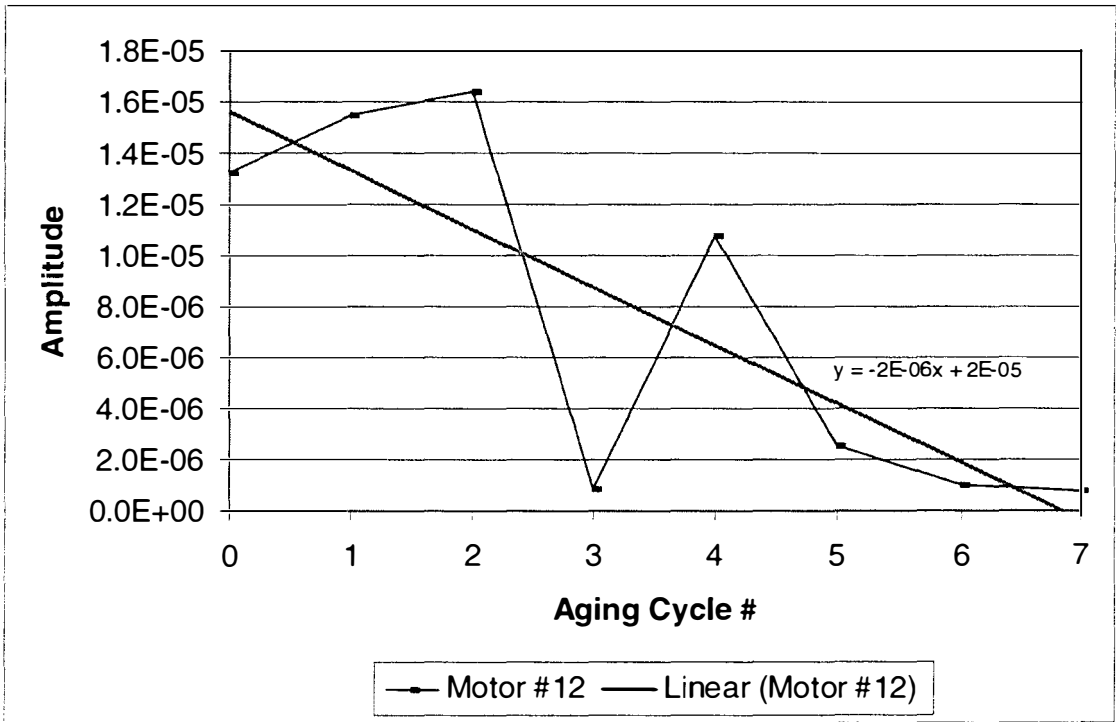


Figure 6.74: Motor current $f_e + PE$ OR frequency amplitude at 100% load.

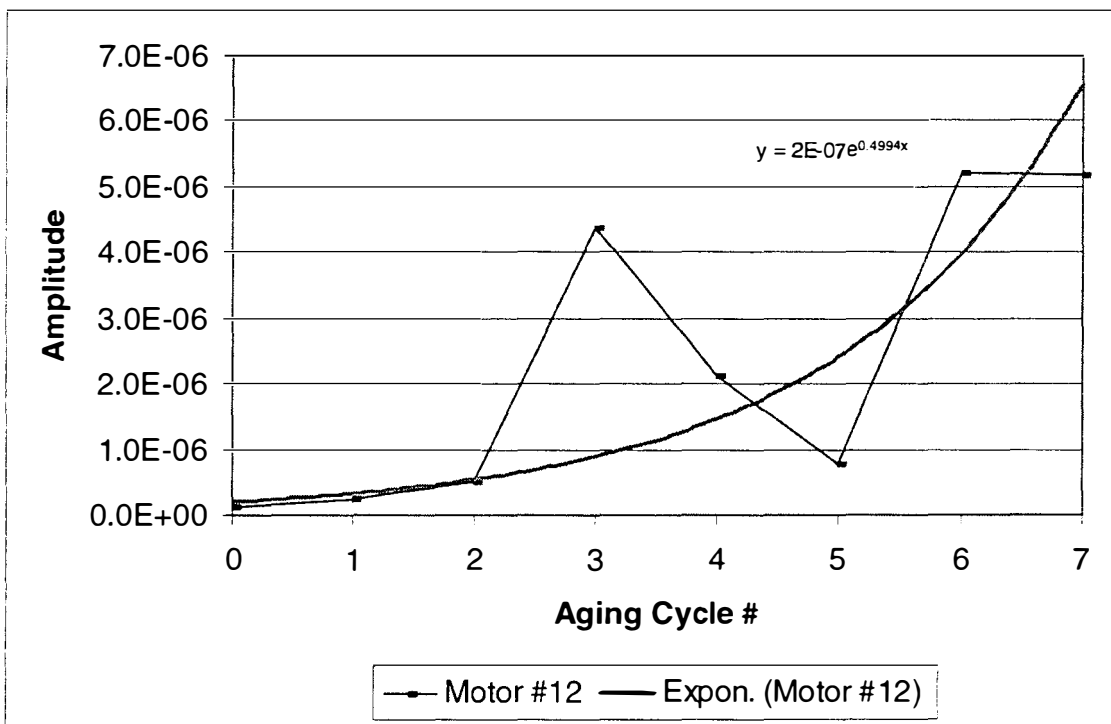


Figure 6.75: Motor PE vibration f_e + PE OR frequency amplitude at 100% load.

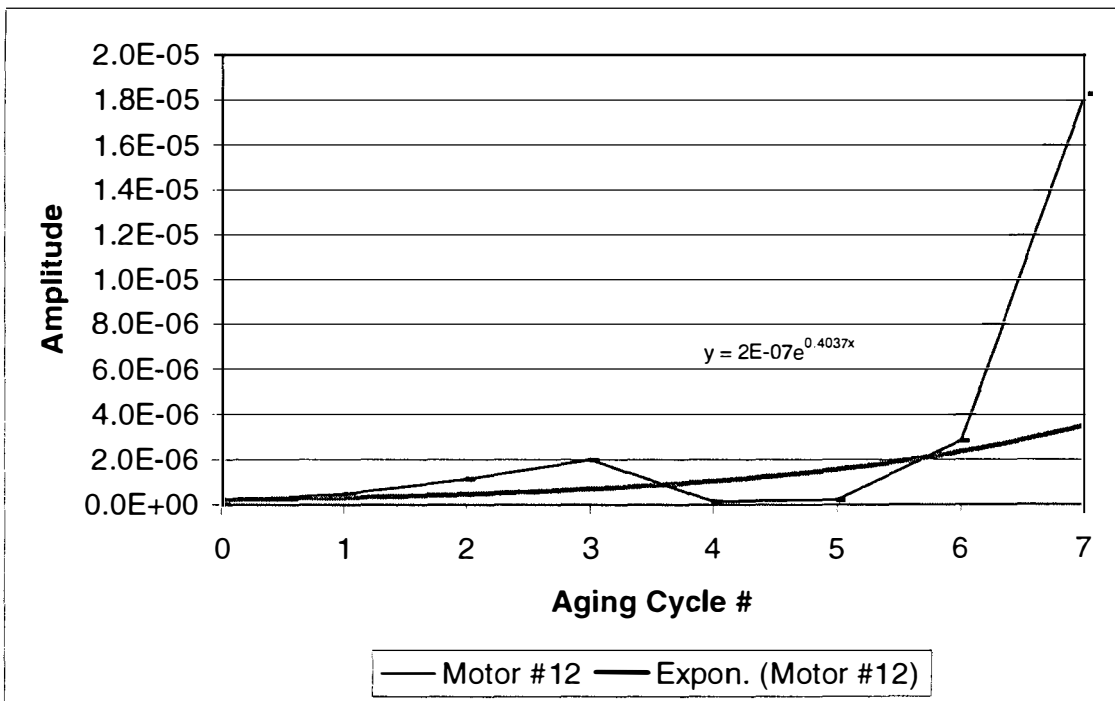


Figure 6.76: Motor cover vibration $f_e + PE$ OR frequency amplitude at 100% load.

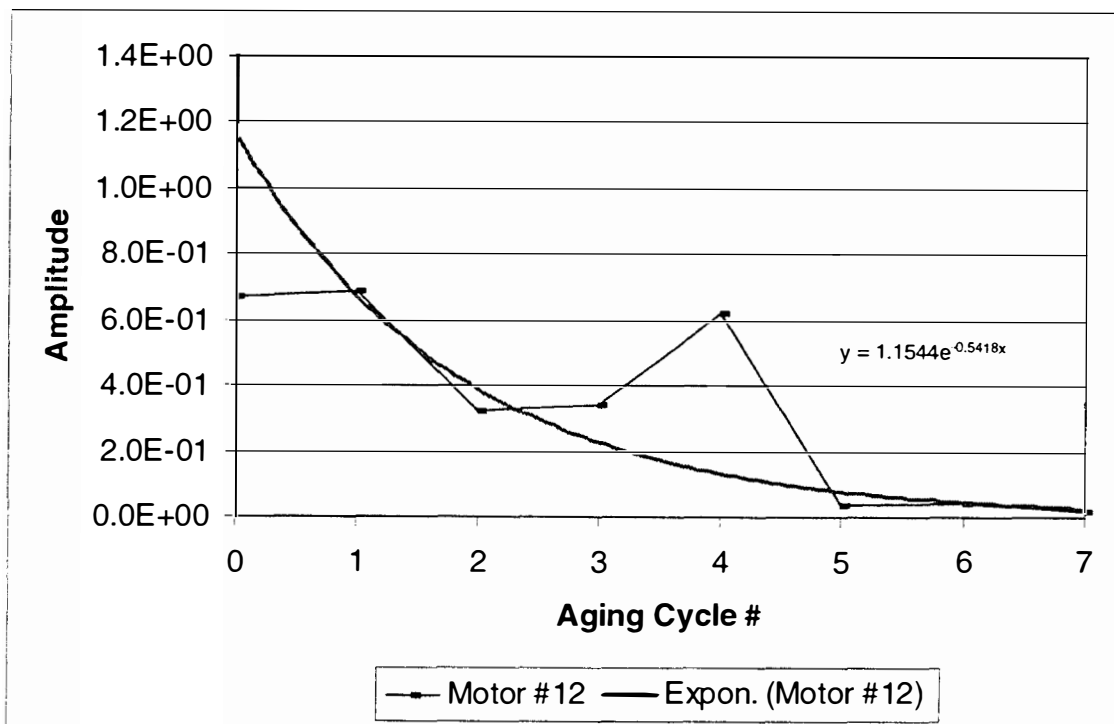


Figure 6.77: Motor power f_c + SE B frequency amplitude at 100% load.

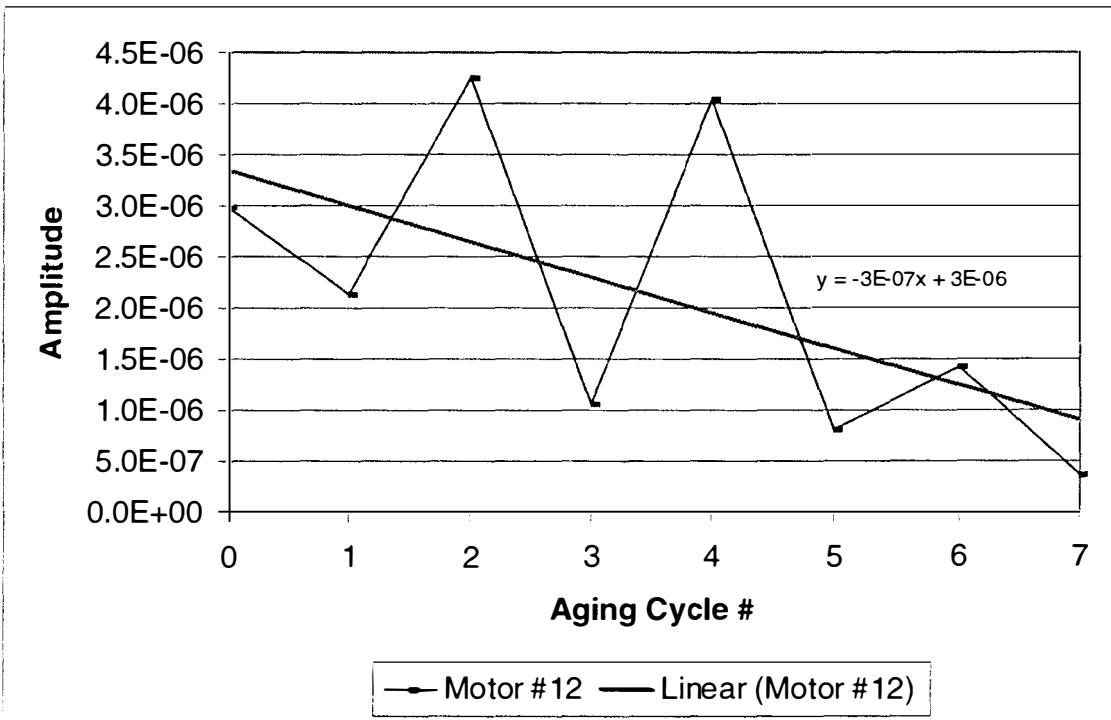


Figure 6.78: Motor current $f_c + SE B$ frequency amplitude at 100% load.

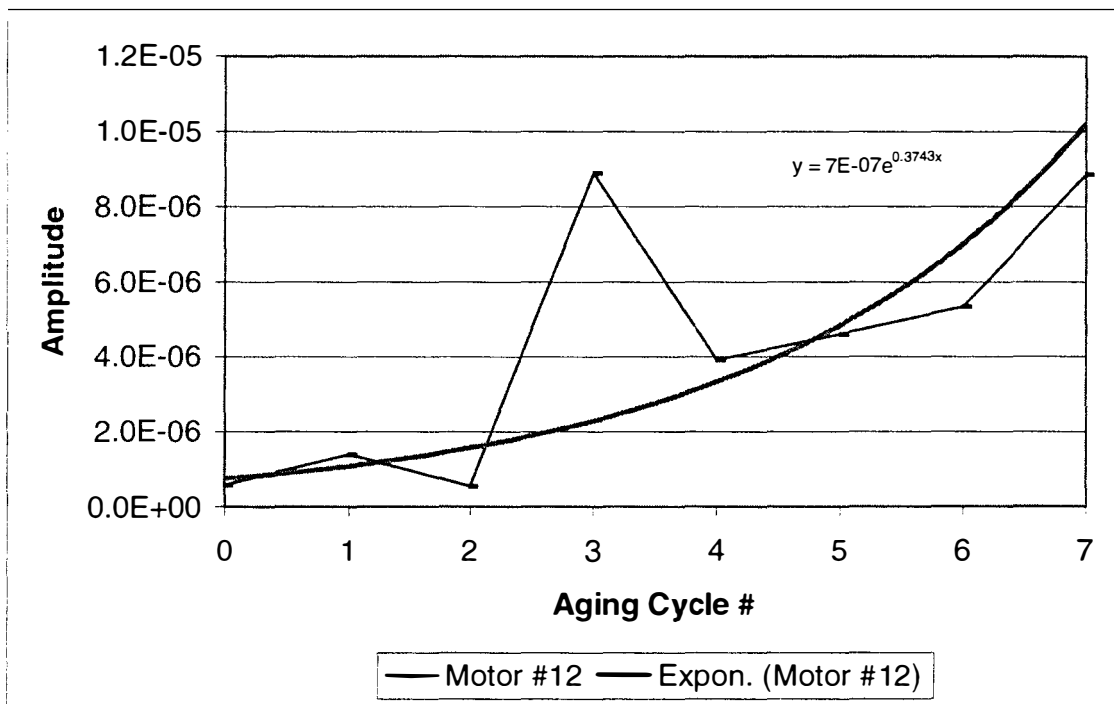


Figure 6.79: Motor PE vibration $f_c + SE B$ frequency amplitude at 100% load.

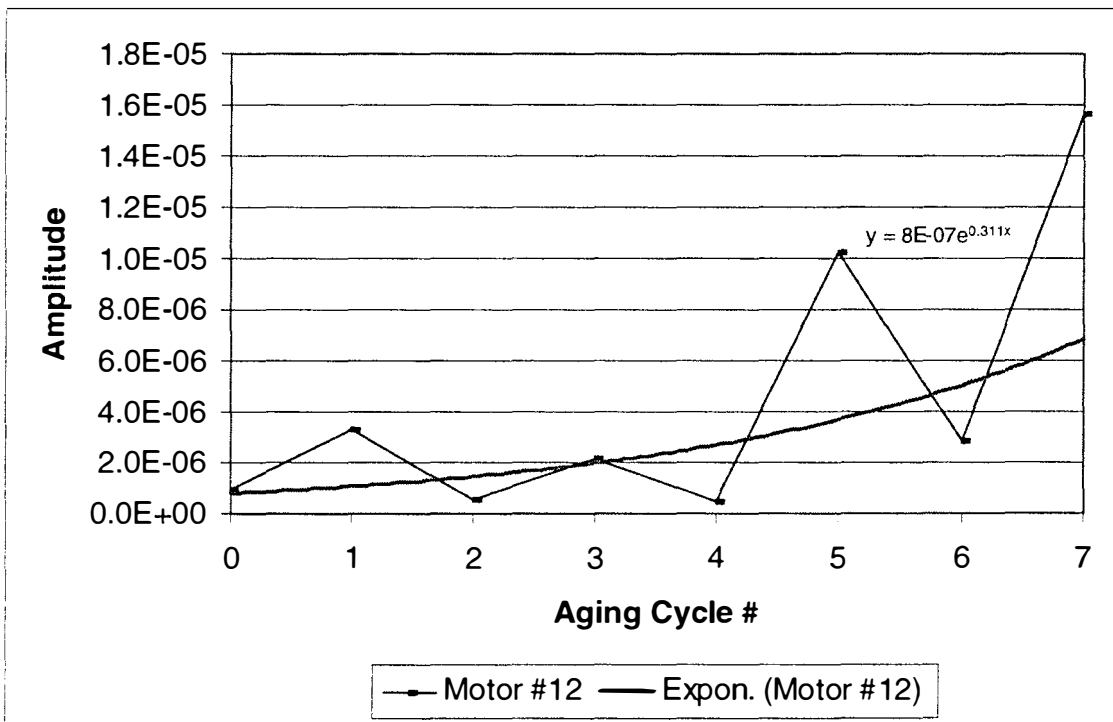


Figure 6.80: Motor cover vibration $f_e + SE\ B$ frequency amplitude at 100% load.

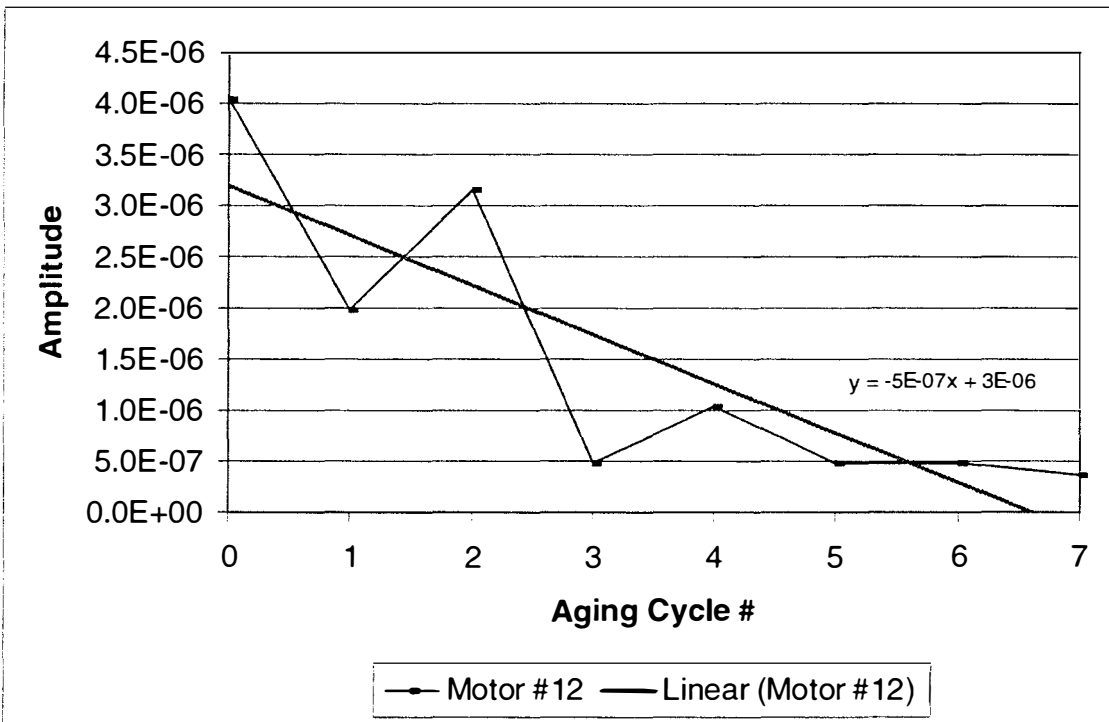


Figure 6.81: Motor current $f_c + PE$ B frequency amplitude at 100% load.

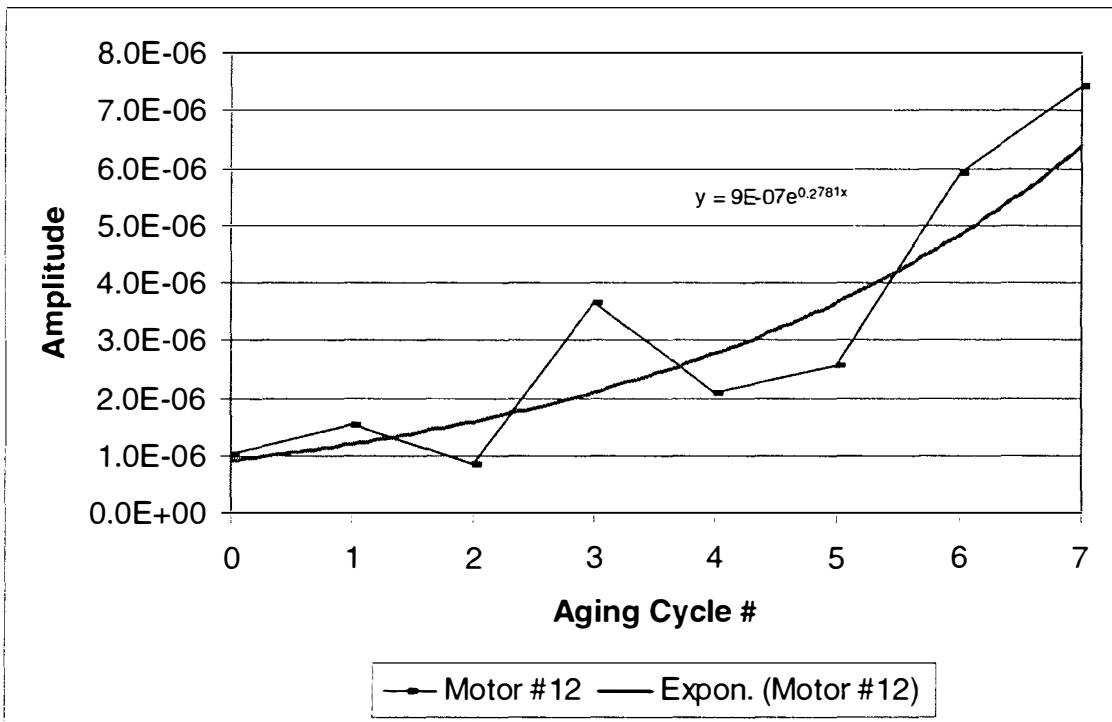


Figure 6.82: Motor PE vibration f_c + PE B frequency amplitude at 100% load.

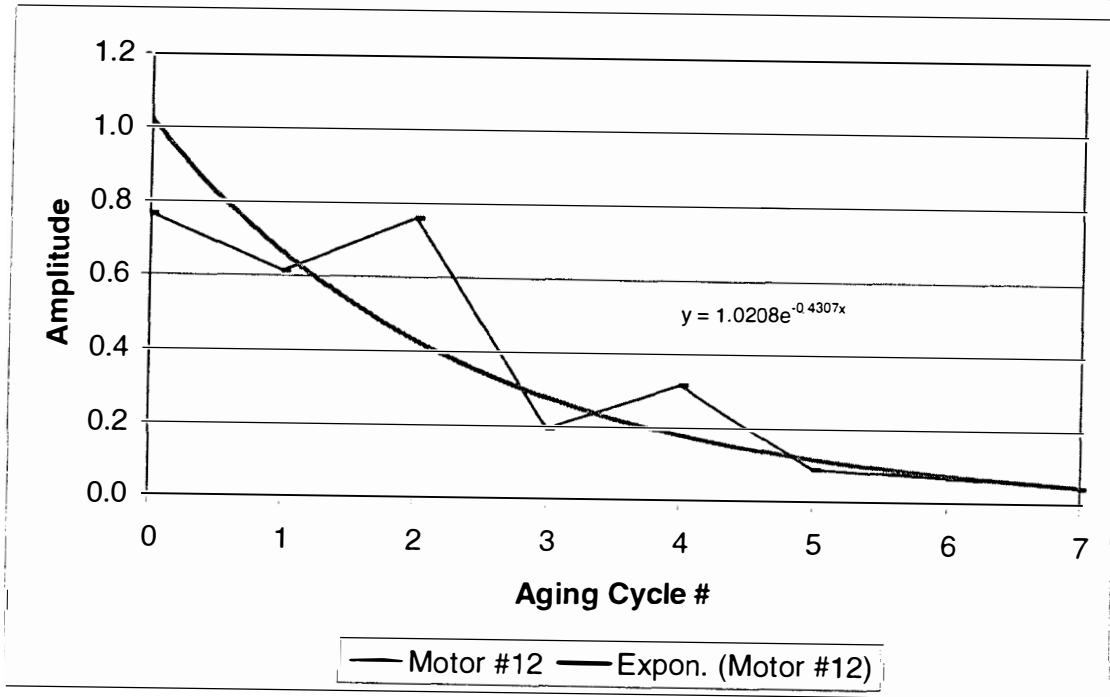
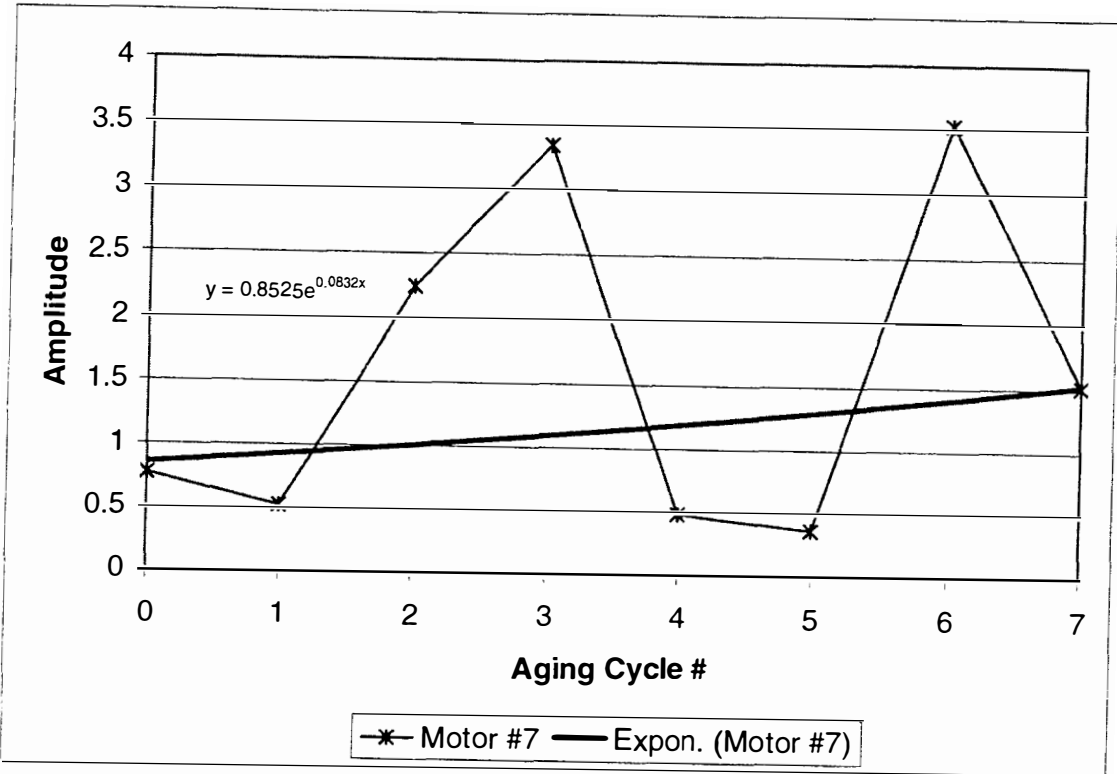


Figure 6.83: Motor power f_e + RPM frequency amplitude plot at 100% load.

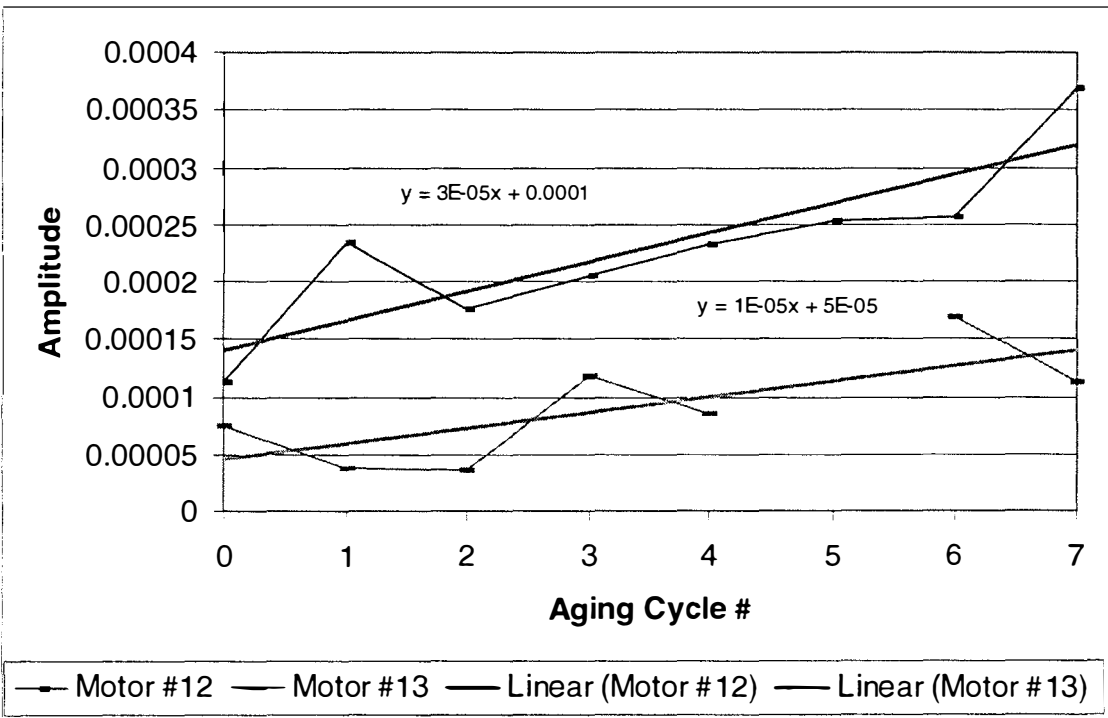
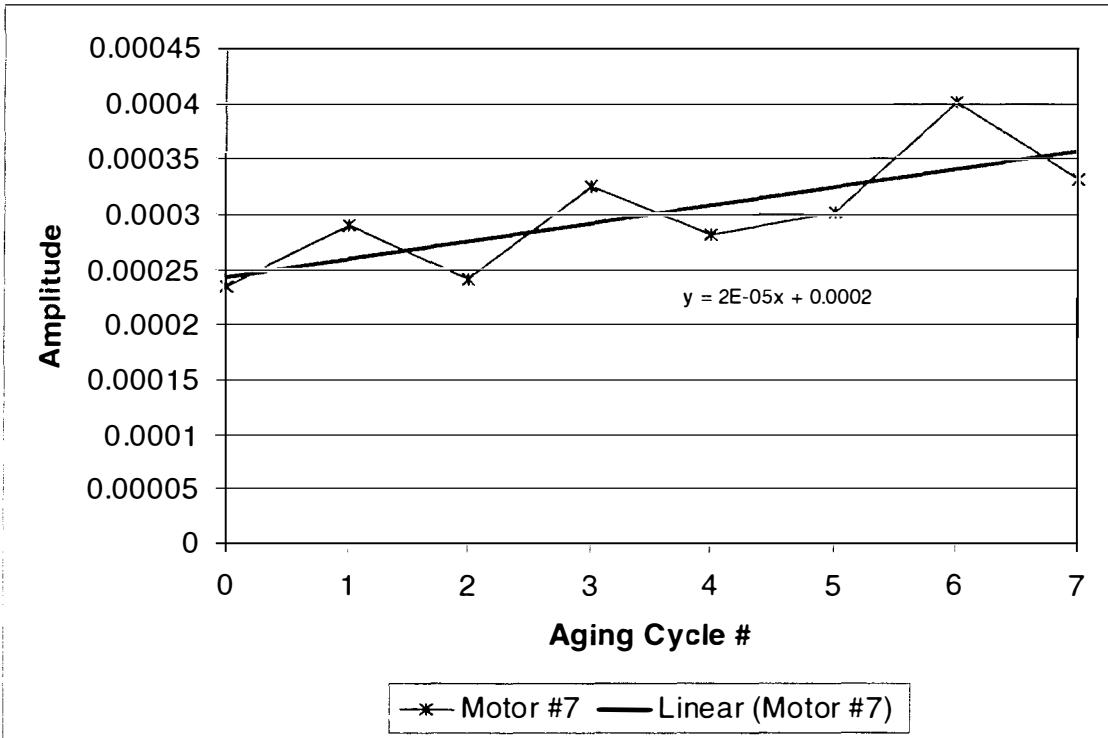


Figure 6.84: Motor current f_e + RPM frequency amplitude plot at 100% load.

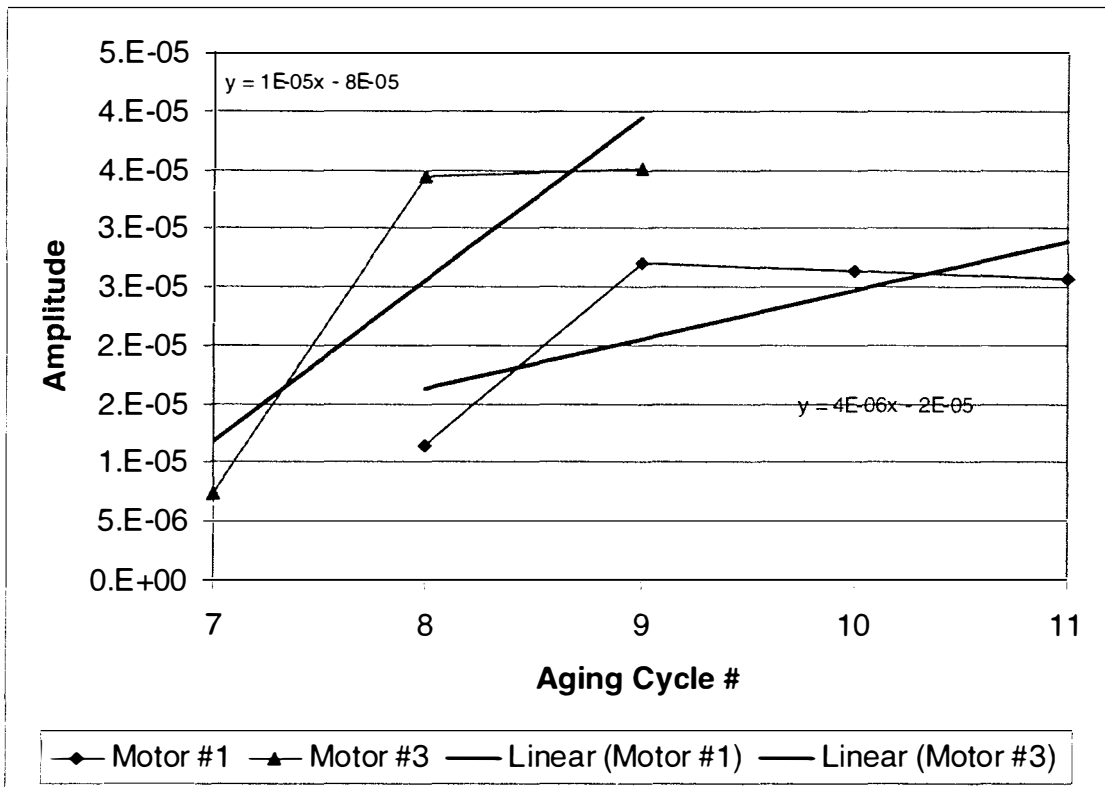


Figure 6.85: Axial magnetic flux $f_e + 2 * \text{RPM}$ frequency amplitude at 100% load.

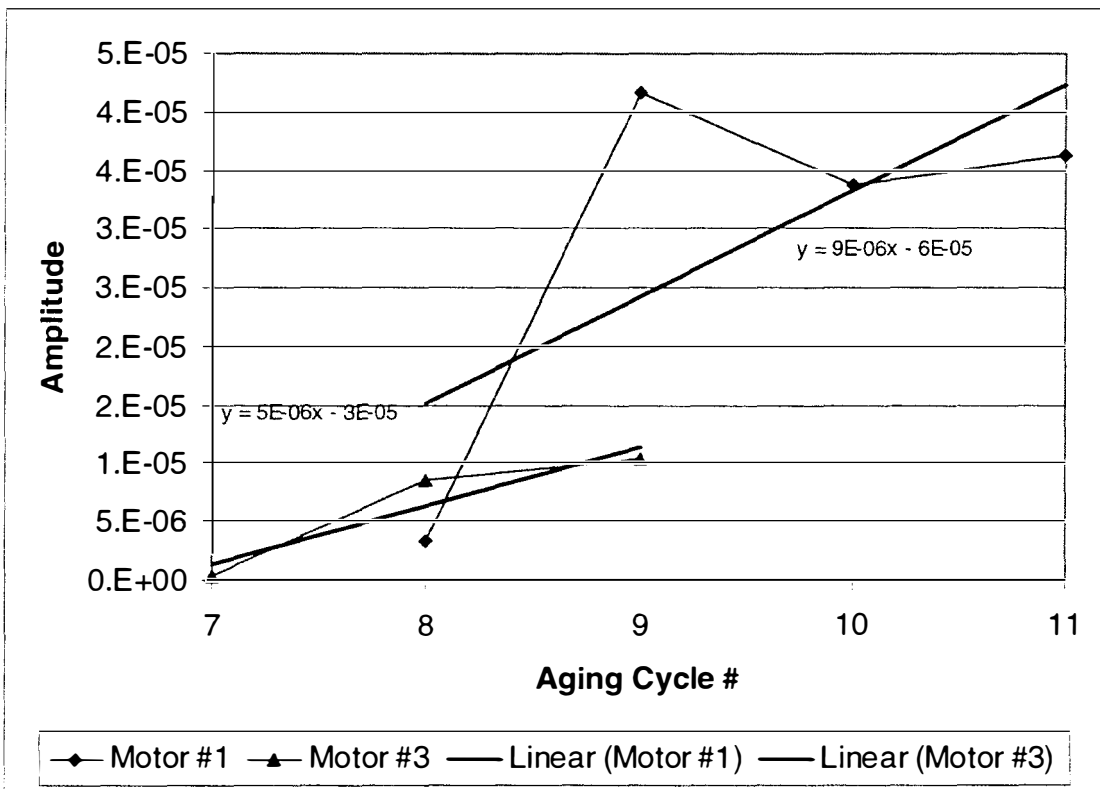


Figure 6.86: Radial magnetic flux f_e - RPM frequency amplitude at 100% load.

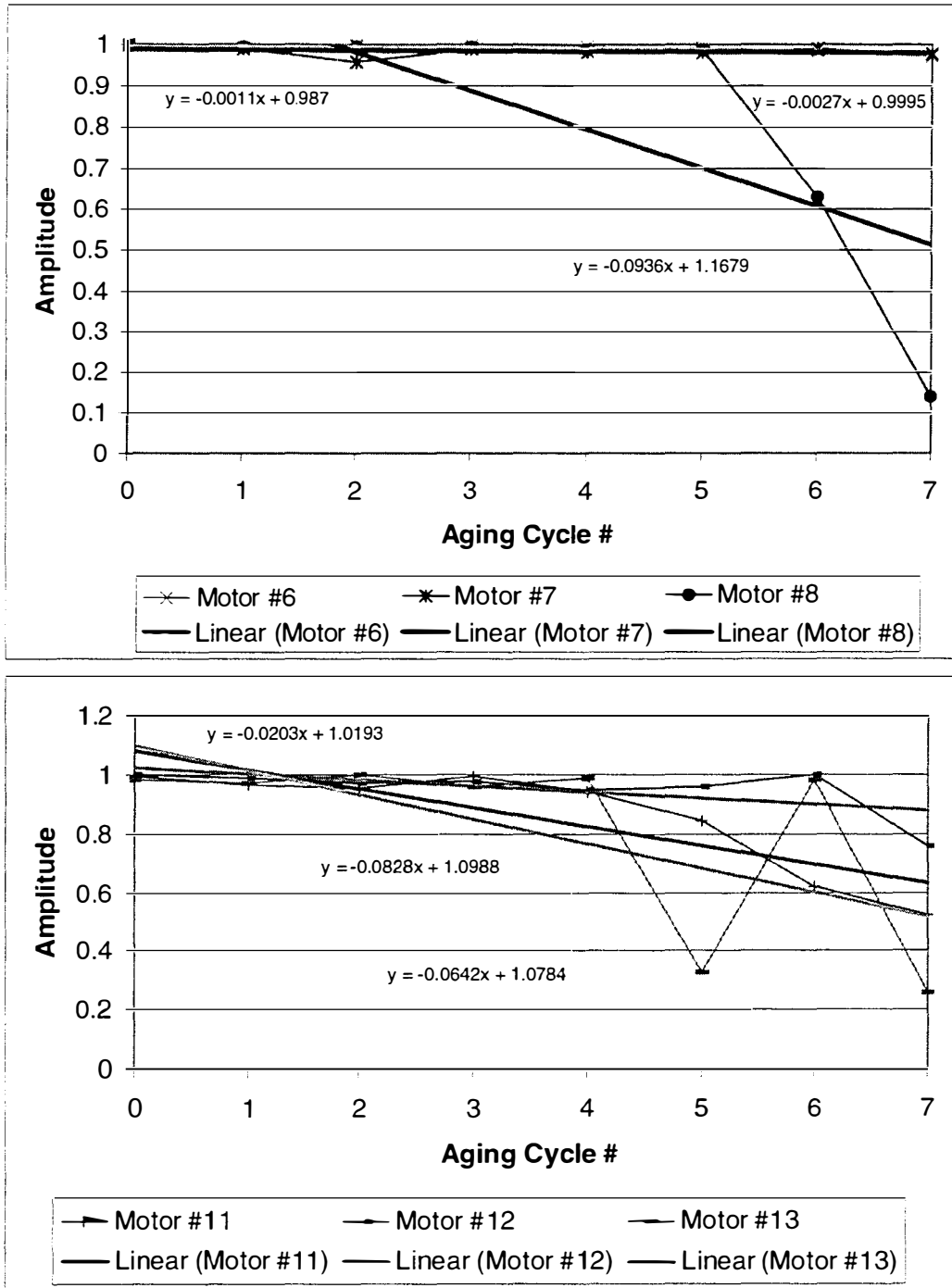


Figure 6.87: Motor power – PE 2:00 vibration coherence slip frequency amplitude at 100% load.

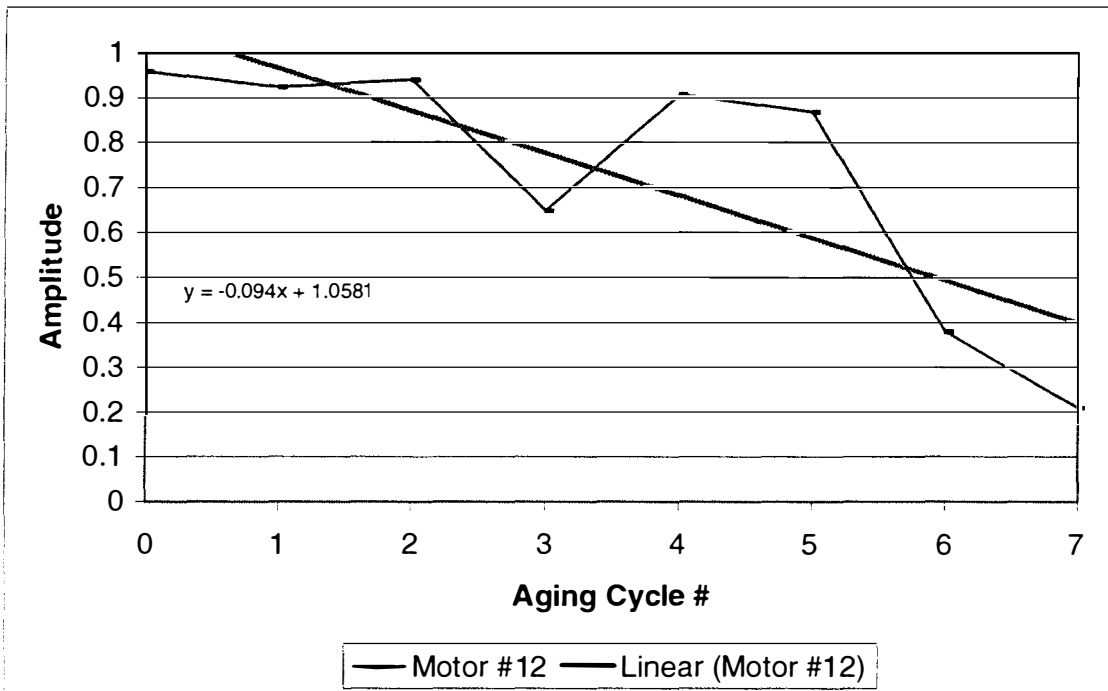


Figure 6.88: Motor power – PE 2:00 vibration coherence slip * number of poles
frequency amplitude at 100% load.

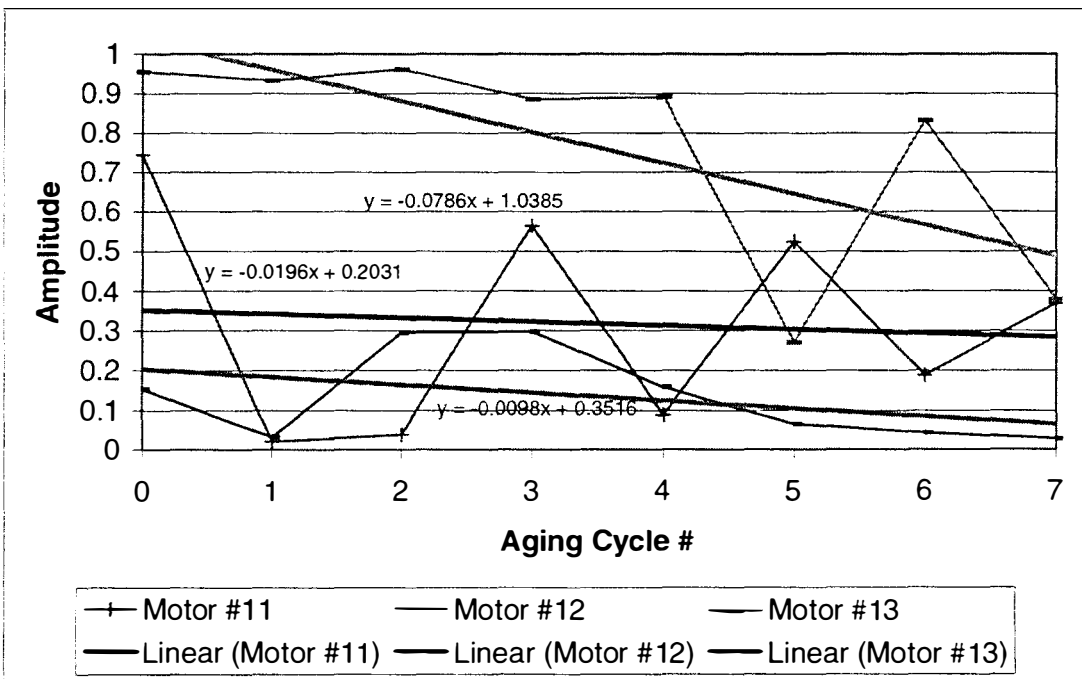
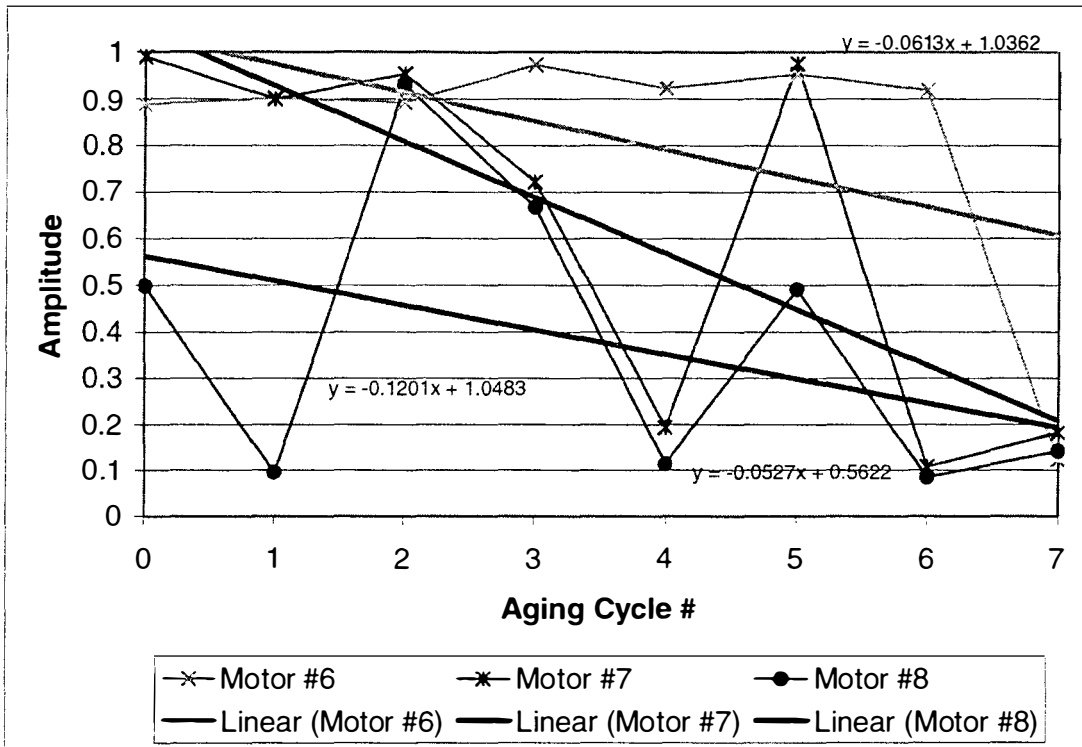


Figure 6.89: Motor power – PE 2:00 vibration coherence slip * number of poles / 2

frequency amplitude at 100% load.

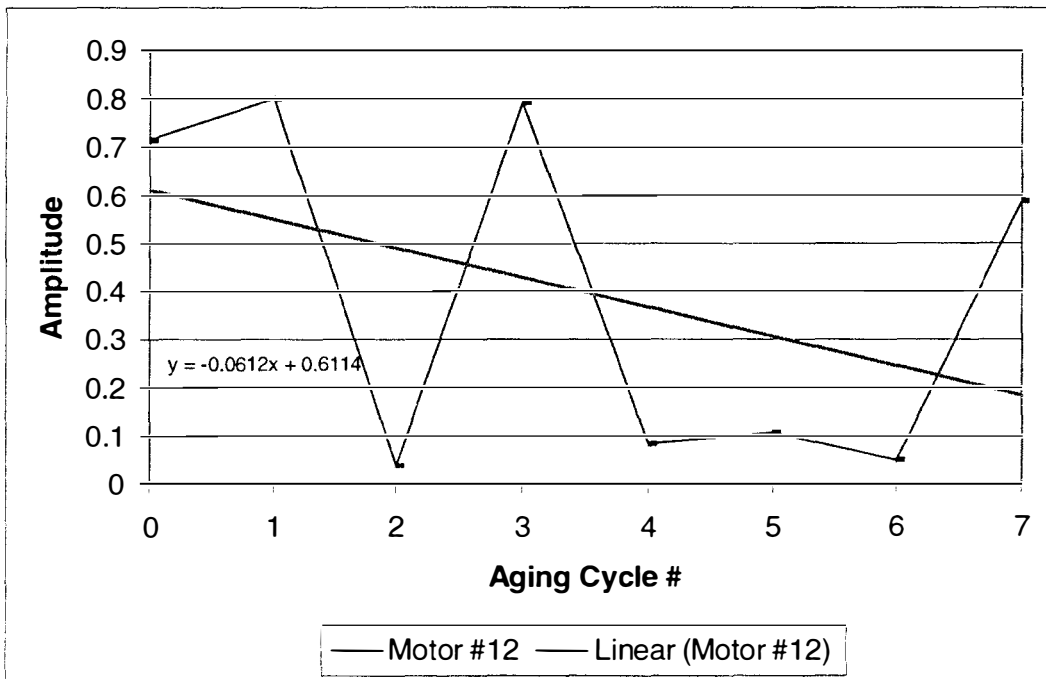
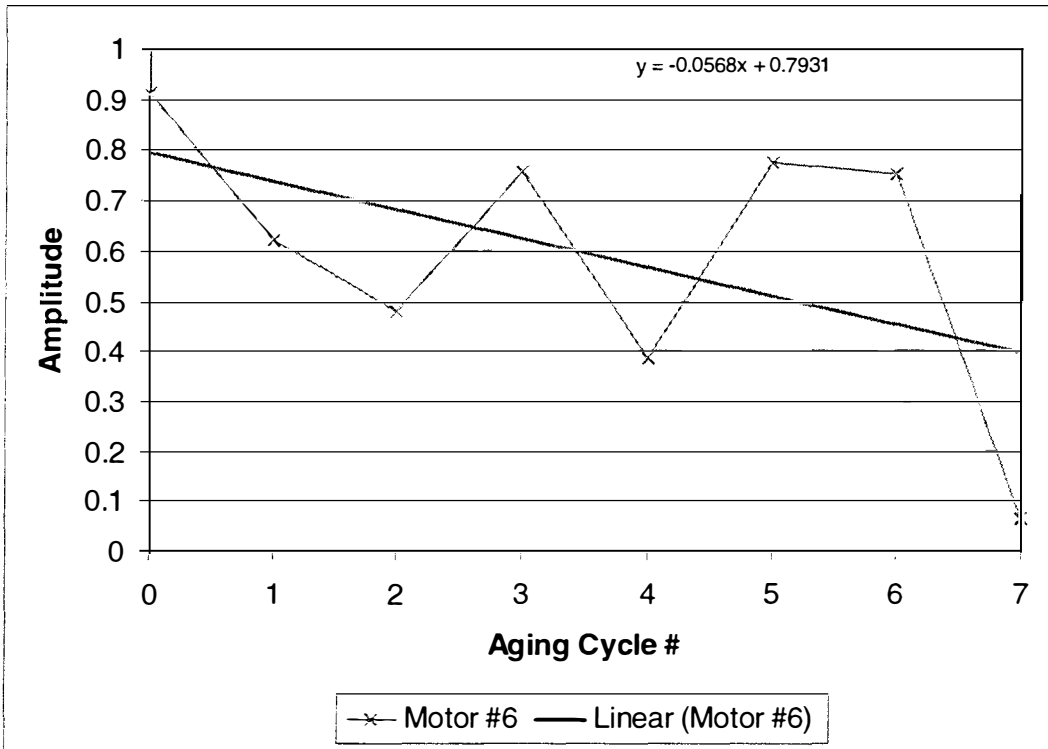


Figure 6.90: Motor power – PE 2:00 vibration coherence line frequency amplitude at 100% load.

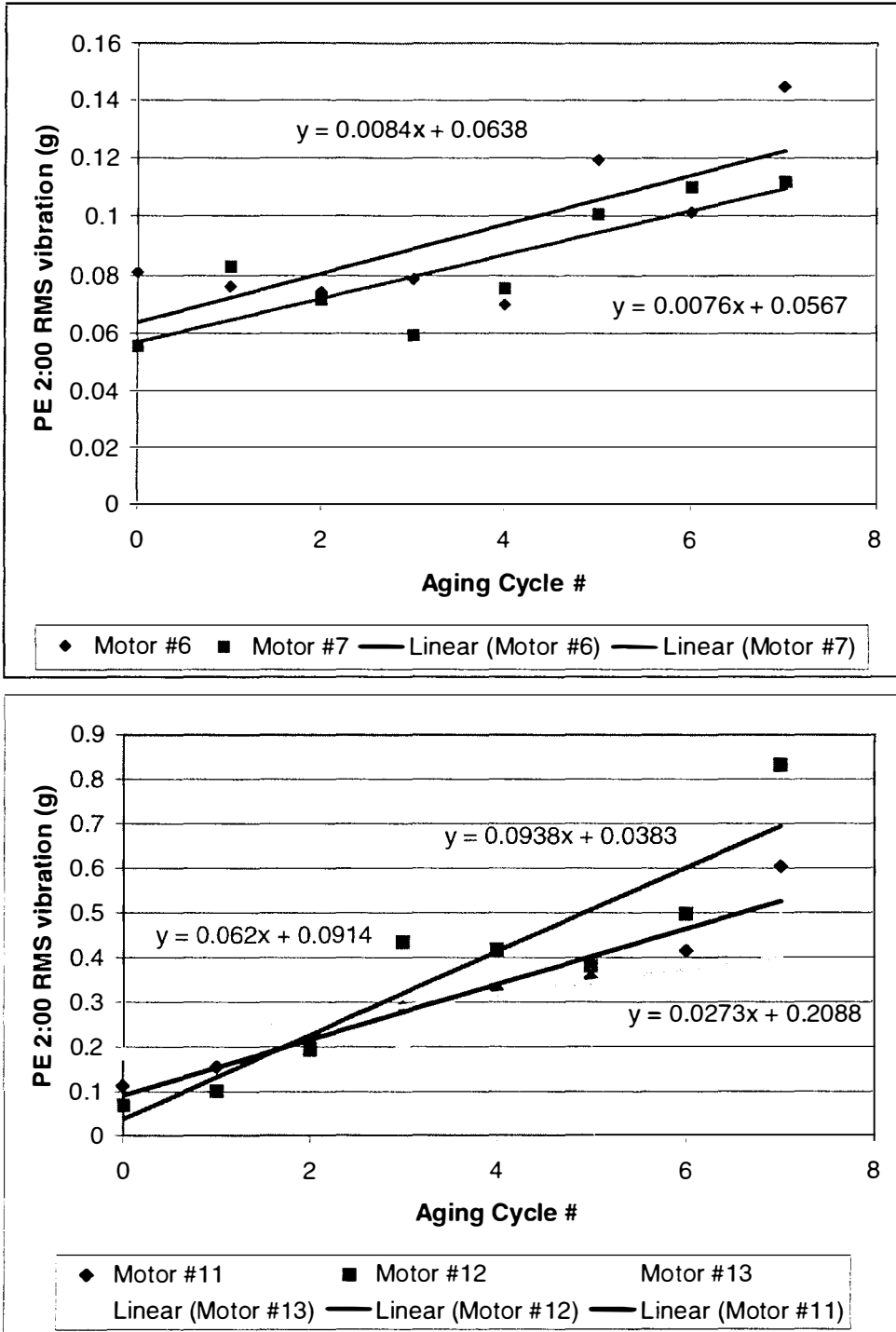


Figure 6.91: PE 2:00 RMS vibration trends for thermal aging and electrical discharge aging.

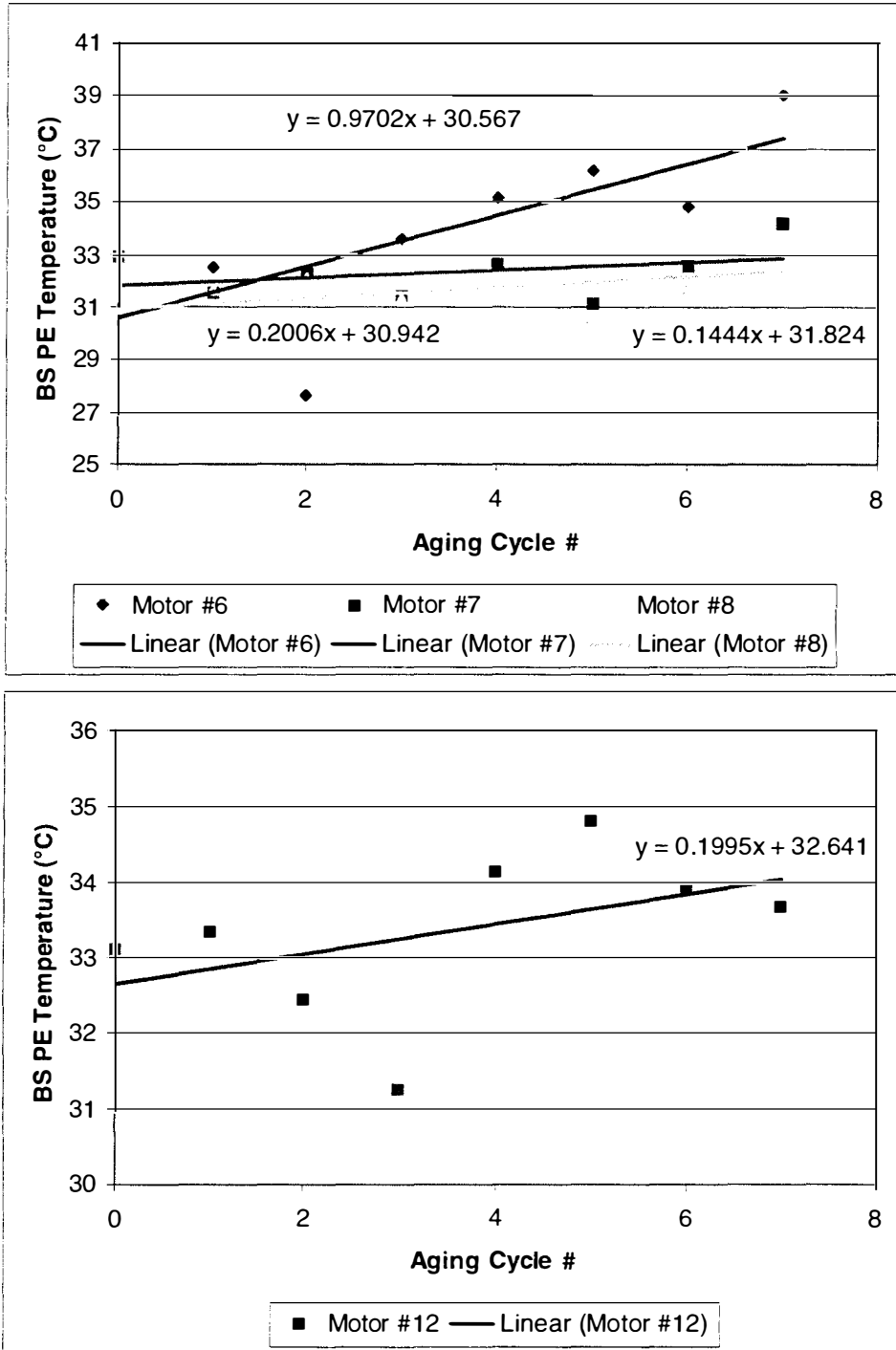


Figure 6.92: Bearing surface PE temperature difference trends for thermal aging and electrical discharge aging.

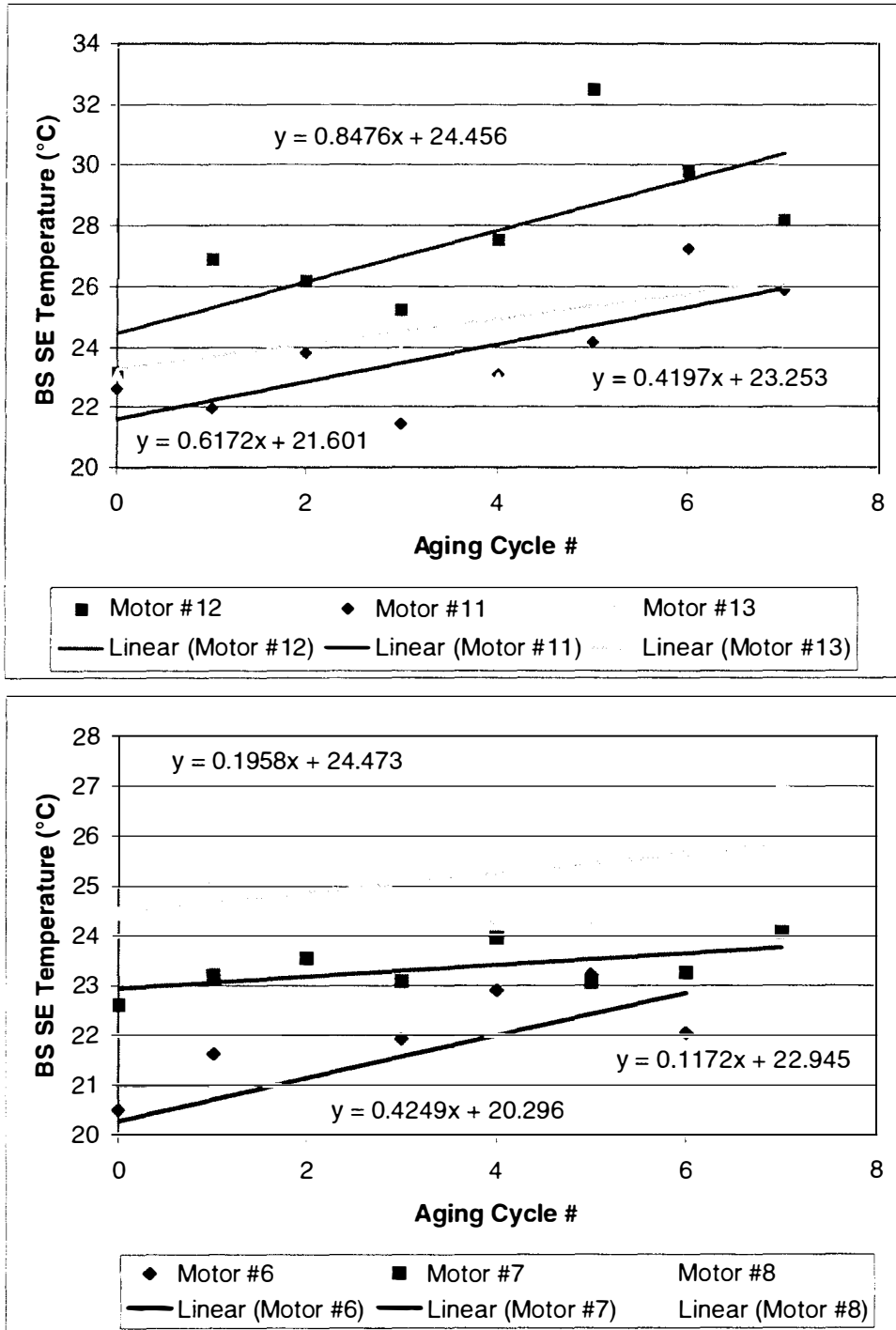


Figure 6.93: Bearing surface SE temperature difference trends for thermal aging and electrical discharge aging.

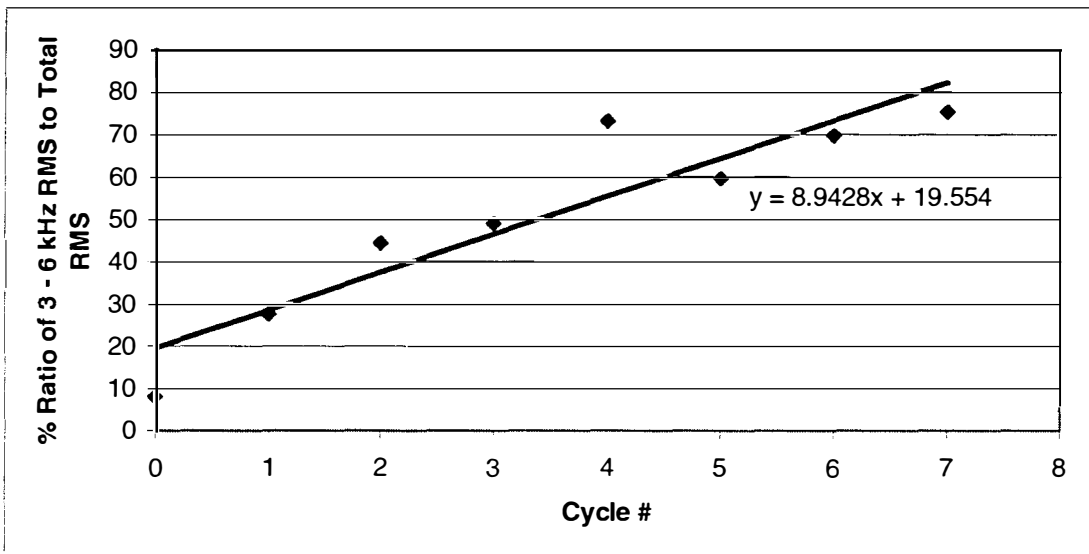


Figure 6.94: MRA RMS ratios of PE 2:00 motor vibration for thermal aging and fluting

Table 6.3: Matrix of Failure Detection

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Lamination Temperature	Mean				
Air Gap Temperature	Mean				
Cover Box Temperature	Mean				
Ambient Temperature in Cover Box	Mean				
PE 4:00 Winding Temperature	Mean				D
PE 8:00 Winding Temperature	Mean				D
PE 12:00 Winding Temperature	Mean				D
SE 2:00 Winding Temperature	Mean				D
SE 6:00 Winding Temperature	Mean				D
SE 10:00 Winding Temperature	Mean				D
PE Bearing Surface Temperature	Mean	D	D		
SE Bearing Surface Temperature	Mean	D	D		
Motor Voltage Phase X Phase Y Phase Z	RMS				

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Motor Voltage Phase X Phase Y Phase Z	RMS				
Motor Current Phase X Phase Y Phase Z	RMS				D
	Skewness				
	RPM	D			
	Slip				
	Slip X number of poles				
	Slip X number of poles / 2				
	Slip X number of poles X 2				
	(1 + 2s) fe				
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE	D	D		
	fe + IR PE	D	D		
	fe + OR SE	D	D		
	fe + OR PE	D	D		
	fe + B SE	D	D		
	fe + B PE	D	D		
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe				
	fe * 2				
	fe + RPM	D	D		
	fe + 2 * RPM				
fe + 3 * RPM					

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Motor Power	Mean				D
	Skewness	D	D	D	
	RPM	D	D		
	Slip				
	Slip X number of poles				
	Slip X number of poles / 2				
	Slip X number of poles X 2				
	(1 + 2s) fe				
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE	D	D		
	fe + IR PE	D	D		
	fe + OR SE	D	D		
	fe + OR PE	D	D		
	fe + B SE	D	D		
	fe + B PE	D	D		
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe				
	fe * 2				
	fe + RPM				
	fe + 2 * RPM				
fe + 3 * RPM					

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Load Power	Mean				D
	Skewness	D	D	D	
	RPM				
	Slip				
	Slip X number of poles				
	Slip X number of poles / 2				
	Slip X number of poles X 2				
	(1 + 2s) fe				
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE				
	fe + IR PE				
	fe + OR SE				
	fe + OR PE				
	fe + B SE				
	fe + B PE				
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe		D	D	
	fe * 2				
	fe + RPM				
	fe + 2 * RPM				
fe + 3 * RPM					

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Vibration PE 2:00 PE 10:00 SE Vertical SE Horizon. SE Axial Cover	RMS	D	D		
	MRA	D	D		
	RPM	D	D		
	Slip				
	Slip X number of poles				
	Slip X number of poles / 2				
	Slip X number of poles X 2				
	(1 + 2s) fe				
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE	D	D		
	fe + IR PE	D	D		
	fe + OR SE	D	D		
	fe + OR PE	D	D		
	fe + B SE	D	D		
	fe + B PE	D	D		
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe				
	fe * 2				
	fe + RPM				
	fe + 2 * RPM				
	fe + 3 * RPM				

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Motor Power – Vibration Coherence	RPM				
	Slip	D	D		
	Slip X number of poles	D	D		
	Slip X number of poles / 2	D	D		
	Slip X number of poles X 2				
	(1 + 2s) fe				
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE				
	fe + IR PE				
	fe + OR SE				
	fe + OR PE				
	fe + B SE				
	fe + B PE				
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe		D	D	
	fe * 2				
	fe + RPM				
	fe + 2 * RPM				
	fe + 3 * RPM				
	RMS				
	Skewness				

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Motor Power – Load Power Coherence	RPM				
	Slip	D	D		
	Slip X number of poles	D	D		
	Slip X number of poles / 2	D	D		
	Slip X number of poles X 2				
	(1 + 2s) fe				
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE				
	fe + IR PE				
	fe + OR SE				
	fe + OR PE				
	fe + B SE				
	fe + B PE				
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe		D	D	
	fe * 2				
	fe + RPM				
	fe + 2 * RPM				
	fe + 3 * RPM				
	RMS				
	Skewness				

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Motor Magnetic Flux	RPM			D	
	Slip			D	
Radial Axial	Slip X number of poles				
	Slip X number of poles / 2				
	Slip X number of poles X 2				
	(1 + 2s) fe			D	
	(1 - 2s) fe				
	IR SE				
	IR PE				
	OR SE				
	OR PE				
	B SE				
	B PE				
	CT SE				
	CT PE				
	fe + IR SE				
	fe + IR PE				
	fe + OR SE				
	fe + OR PE				
	fe + B SE				
	fe + B PE				
	fe + CT SE				
	fe + CT PE				
	fe - CT SE				
	fe - CT PE				
	fe - RPM				
	fe				D
	fe * 2				
	fe + RPM				
	fe + 2 * RPM				
	fe + 3 * RPM				
	RMS				
Skewness					

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault	
Zero-Sequence Component Impedance	RMS			D		
	RPM					
	Slip			D		
	Slip X number of poles					
	Slip X number of poles / 2					
	Slip X number of poles X 2					
	(1 + 2s) fe					
	(1 - 2s) fe					
	IR SE					
	IR PE					
	OR SE					
	OR PE					
	B SE					
	B PE					
	CT SE					
	CT PE					
	fe + IR SE					
	fe + IR PE					
	fe + OR SE					
	fe + OR PE					
	fe + B SE					
	fe + B PE					
	fe + CT SE					
	fe + CT PE					
	fe - CT SE					
	fe - CT PE					
	fe - RPM					
	fe				D	
	fe * 2					
	fe + RPM					
	fe + 2 * RPM					
	fe + 3 * RPM					
RMS						

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Zero – Sequence Component Voltage	RMS				
Zero – Sequence Component Current	RMS				
Positive Sequence Component Impedance Phase X Phase Y Phase Z	RMS				D
Negative Sequence Component Impedance Phase X Phase Y Phase Z	RMS				D
Positive Sequence Component Voltage Phase X Phase Y Phase Z	RMS				
Negative Sequence Component Voltage Phase X Phase Y Phase Z	RMS				

Table 6.3 Continued

Variable	Frequency	Electrical Discharge Damage on Bearings	Thermal Damage on Bearings	Winding Insulation Damage	Stator Line to Line Fault
Positive Sequence Component Current Phase X Phase Y Phase Z	RMS				
Negative Sequence Component Current Phase X Phase Y Phase Z	Mean				
Motor Efficiency	Mean				D
System Efficiency	Mean				D
Power Factor	Mean				
Speed	Mean				D

D = Detectable signature for fault / degradation

Table 6.4: Table of Trends

SIGNATURE	FAILURE TYPE	TREND EQUATIONS	TREND SLOPE	FAILURE LIMIT
Axial Magnetic Flux RPM Frequency Amplitude	Winding Insulation	$Y = 3 \times 10^{-7} X - 2 \times 10^{-6}$ $Y = 1 \times 10^{-7} X - 7 \times 10^{-7}$	2×10^{-7}	5×10^{-7}
Radial Magnetic Flux RPM Frequency Amplitude	Winding Insulation	$Y = -3 \times 10^{-7} X + 2 \times 10^{-6}$ $Y = -9 \times 10^{-8} X + 1 \times 10^{-6}$	-2×10^{-7}	0
Motor Power RPM Frequency Amplitude	Thermal Aging	$Y = 0.0023e^{-0.1590 X}$ $Y = 0.0017e^{-0.1718 X}$ $Y = 0.0013e^{-0.1204 X}$	-0.1504	0.0005
Motor PE Vibration RPM Frequency Amplitude	Thermal Aging	$Y = 9 \times 10^{-6} X + 0.0005$ $Y = 2 \times 10^{-5} X + 0.0002$ $Y = 5 \times 10^{-6} X + 0.0002$	1.14×10^{-5}	$Y_1 + 0.0001$
	Thermal Aging & Fluting	$Y = 5 \times 10^{-5} X + 0.0004$ $Y = 1 \times 10^{-5} X + 0.0003$ $Y = 8 \times 10^{-6} X + 0.0002$	2.27×10^{-5}	
Zero Component Impedance Slip Frequency Amplitude	Winding Insulation	$Y = 1.9674 X + 2.4967$ $Y = 1.7785 X - 4.7411$	1.873	17
Radial Magnetic Flux Slip Frequency Amplitude	Winding Insulation	$Y = -1 \times 10^{-6} X + 1 \times 10^{-5}$ $Y = -6 \times 10^{-7} X + 5 \times 10^{-6}$	-8×10^{-7}	0
Radial Magnetic Flux (1 + 2 x Slip) f_c Frequency Amplitude	Winding Insulation	$Y = 1 \times 10^{-7} X - 7 \times 10^{-7}$ $Y = 1 \times 10^{-7} X - 8 \times 10^{-7}$	1×10^{-7}	4×10^{-7}
Zero Component Impedance Line Frequency Amplitude	Winding Insulation	$Y = 849.65 X + 819.46$ $Y = 735.49 X - 1964.8$	792.57	7000
Radial Magnetic Flux Line Frequency Amplitude	Winding Insulation	$Y = 0.1755 X - 1.1588$ $Y = 0.1026 X - 0.725$	0.1391	0.4

Table 6.4 Continued

SIGNATURE	FAILURE TYPE	TREND EQUATIONS	TREND SLOPE	FAILURE LIMIT
Motor PE Vibration $f_c + IR_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 3 \times 10^{-7} e^{0.5376 X}$	0.5376	1.2×10^{-5}
Motor Power $f_c + IR_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 1.9065 e^{-0.6875 X}$	-0.6875	0.01
Motor Current $f_c + IR_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = -2 \times 10^{-7} X + 2 \times 10^{-6}$	-2×10^{-7}	3×10^{-7}
Motor Power $f_c + IR_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 1.688 e^{-0.6371 X}$	-0.6371	0.01
Motor Current $f_c + IR_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = -2 \times 10^{-7} X + 2 \times 10^{-6}$	-2×10^{-7}	3×10^{-7}
Motor PE Vibration $f_c + IR_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 4 \times 10^{-7} e^{0.5151 X}$	0.5151	1.2×10^{-5}
Motor Power $f_c + OR_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 0.9859 e^{-0.4762 X}$	-0.4762	0.036
Motor Current $f_c + OR_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = -1 \times 10^{-6} X + 1 \times 10^{-5}$	-1×10^{-6}	3×10^{-6}
Motor PE Vibration $f_c + OR_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 3 \times 10^{-7} e^{0.4186 X}$	0.4186	5×10^{-6}
Motor Power $f_c + OR_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 0.7794 e^{-0.4649 X}$	-0.4649	0.030
Motor Current $f_c + OR_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = -2 \times 10^{-6} X + 2 \times 10^{-5}$	-2×10^{-6}	6×10^{-6}

Table 6.4 Continued

SIGNATURE	FAILURE TYPE	TREND EQUATIONS	TREND SLOPE	FAILURE LIMIT
Motor PE Vibration $f_c + OR_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 2 \times 10^{-7} e^{0.4994 X}$	0.4994	5×10^{-6}
Motor Power $f_c + B_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 1.1544 e^{-0.5418 X}$	-0.5418	0.026
Motor Current $f_c + B_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = -3 \times 10^{-7} X + 3 \times 10^{-6}$	-3×10^{-7}	9×10^{-7}
Motor PE Vibration $f_c + B_{SE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 7 \times 10^{-7} e^{0.3743 X}$	0.3743	1×10^{-5}
Motor PE Vibration $f_c + B_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = 9 \times 10^{-7} e^{0.2781 X}$	0.2781	6.5×10^{-6}
Motor Current $f_c + B_{PE}$ Frequency Amplitude	Thermal Aging & Fluting	$Y = -5 \times 10^{-7} X + 3 \times 10^{-6}$	-5×10^{-7}	0
Motor Current $f_c + RPM$ Frequency Amplitude	Thermal Aging And / Or Fluting	$Y = 1 \times 10^{-3} X + 0.00005$ $Y = 2 \times 10^{-5} X + 0.0002$ $Y = 3 \times 10^{-5} X + 0.0001$	2×10^{-3}	$Y_1 + 1.5 \times 10^{-4}$
Motor Power – PE Vibration Coherence Amplitude at Slip Frequency	Thermal Aging And / Or Fluting	$Y = -0.0027 X + 0.9995$ $Y = -0.0011 X + 0.987$ $Y = -0.0936 X + 1.1679$ $Y = -0.0203 X + 1.0193$ $Y = -0.0828 X + 1.0988$ $Y = -0.0642 X + 1.0784$	-0.08	0.5
Motor Power – PE Vibration Coherence Amplitude at Slip x Number of Poles Frequency	Thermal Aging And / Or Fluting	$Y = -0.094 X + 1.0581$	-0.094	0.5

Table 6.4 Continued

SIGNATURE	FAILURE TYPE	TREND EQUATIONS	TREND SLOPE	FAILURE LIMIT
Motor Power – PE Vibration Coherence Amplitude at (Slip x Number of Poles) /2 Frequency	Thermal Aging And / Or Fluting	Y = -0.0613 X + 1.0363 Y = -0.1201 X + 1.0483 Y = -0.0527 X + 0.5622 Y = -0.0622 X + 1.0385 Y = -0.0196 X + 0.2031 Y = -0.0098 X + 0.3516	-0.055	0.5
Motor Power – PE Vibration Coherence Amplitude at Line Frequency	Thermal Aging And / Or Fluting	Y = -0.0568 X + 0.7931 Y = -0.0612 X + 0.6114	-0.058	0.5
Motor PE RMS Vibration	Thermal Aging	Y= 0.0084 X + 0.0638 Y = 0.0076 X + 0.0567	0.008	0.12
	Thermal Aging & Fluting	Y = 0.0938 X + 0.0383 Y = 0.062 X + 0.0914 Y = 0.0273 X + 0.2088	0.061	0.5
Motor PE Bearing Surface Normalized Temperature	Thermal Aging	Y = 0.9702 X + 30.567 Y = 0.2006 X + 30.942 Y = 0.1444 X + 31.824	0.4384	34
	Thermal Aging & Fluting	Y = 0.1995 X + 32.641	0.1995	
Motor SE Bearing Surface Normalized Temperature	Thermal Aging	Y = 0.8476 X + 24.456 Y = 0.6172 X + 21.601 Y = 0.4197 X + 23.253	0.6282	25
	Thermal Aging & Fluting	Y = 0.1958 X + 24.473 Y = 0.1172 X + 22.945 Y = 0.4249 X + 20.296	0.246	
MRA Analysis of Motor PE Vibration	Thermal Aging & Fluting	Y = 8.9428 X + 19.554	8.9428	70

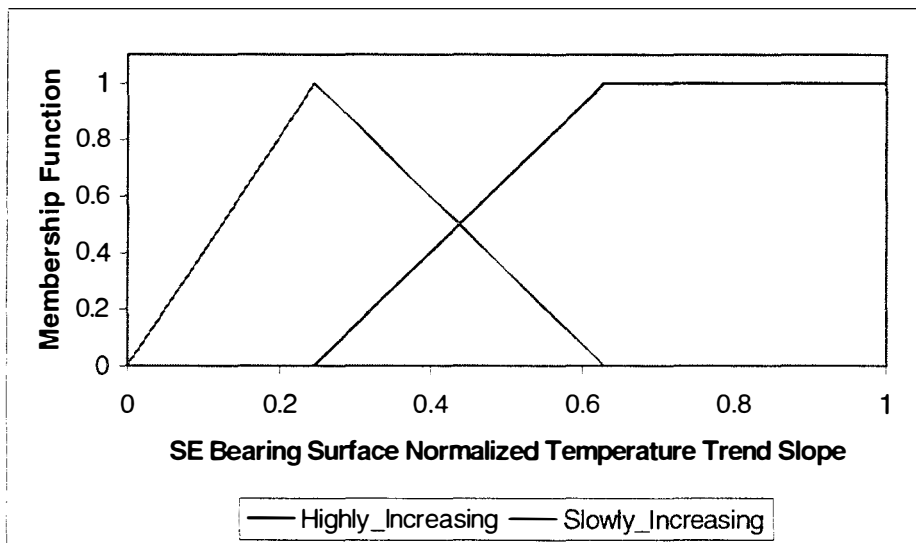
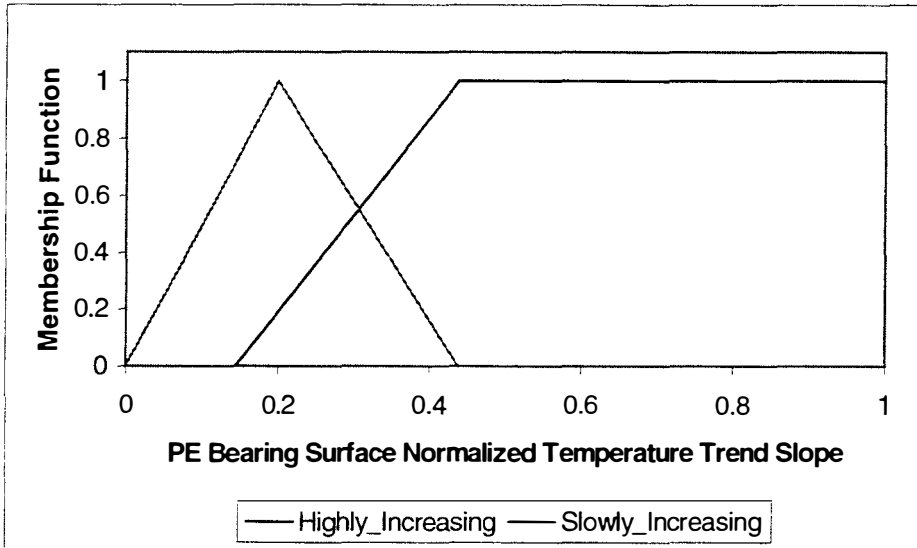


Figure 6.95: Fuzzy membership functions of temperature.

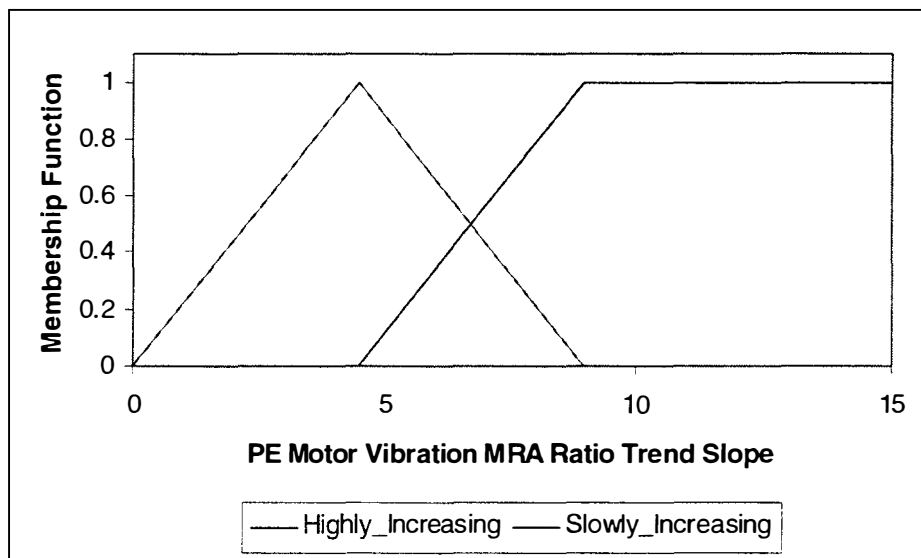
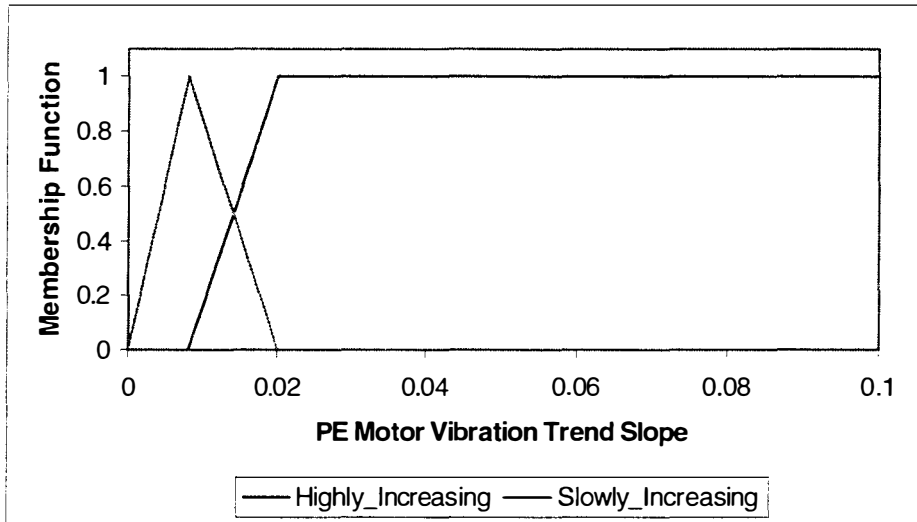


Figure 6.96: Fuzzy membership functions of vibration.

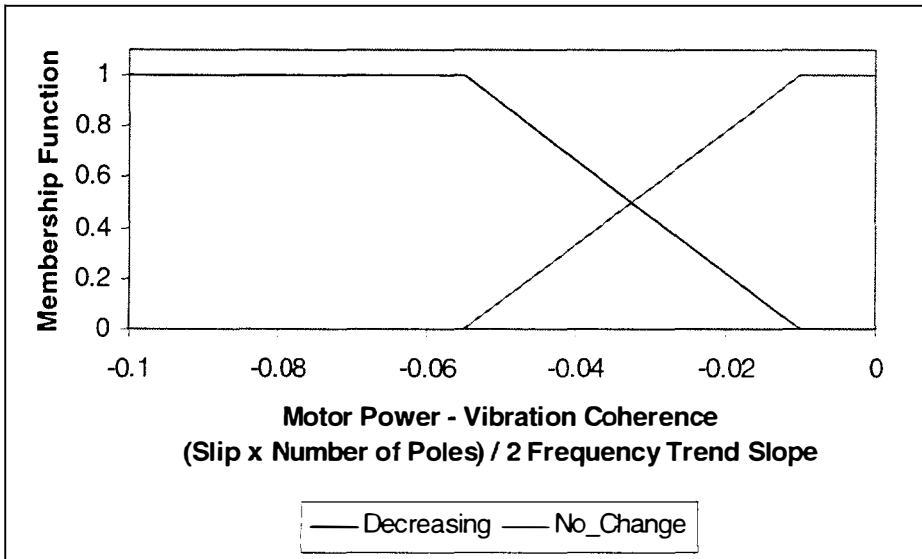
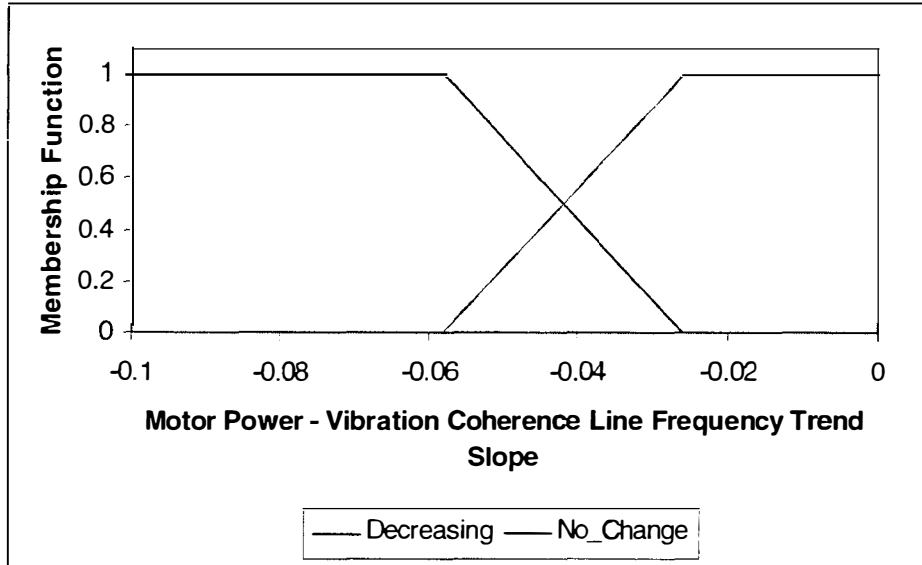


Figure 6.97: Fuzzy membership functions of coherence.

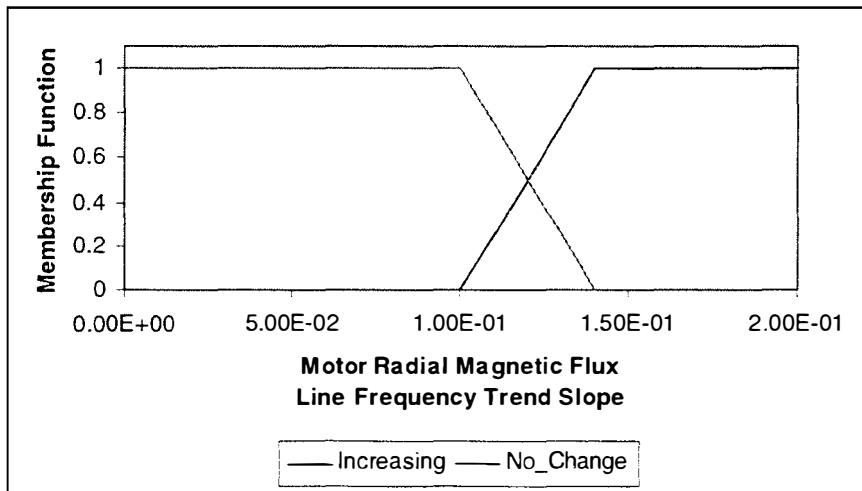
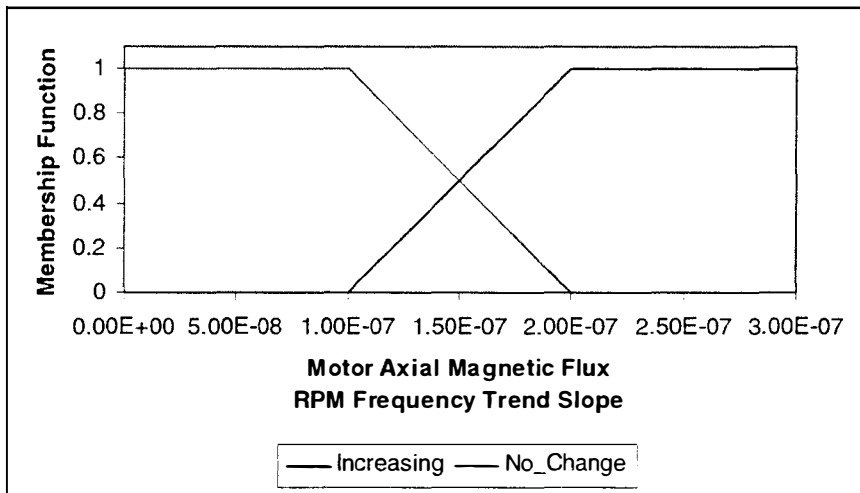
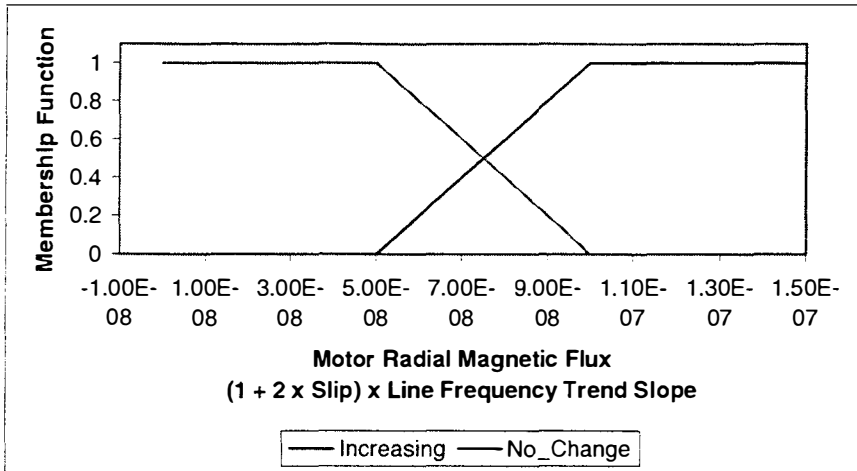


Figure 6.98: Fuzzy membership functions of magnetic flux.

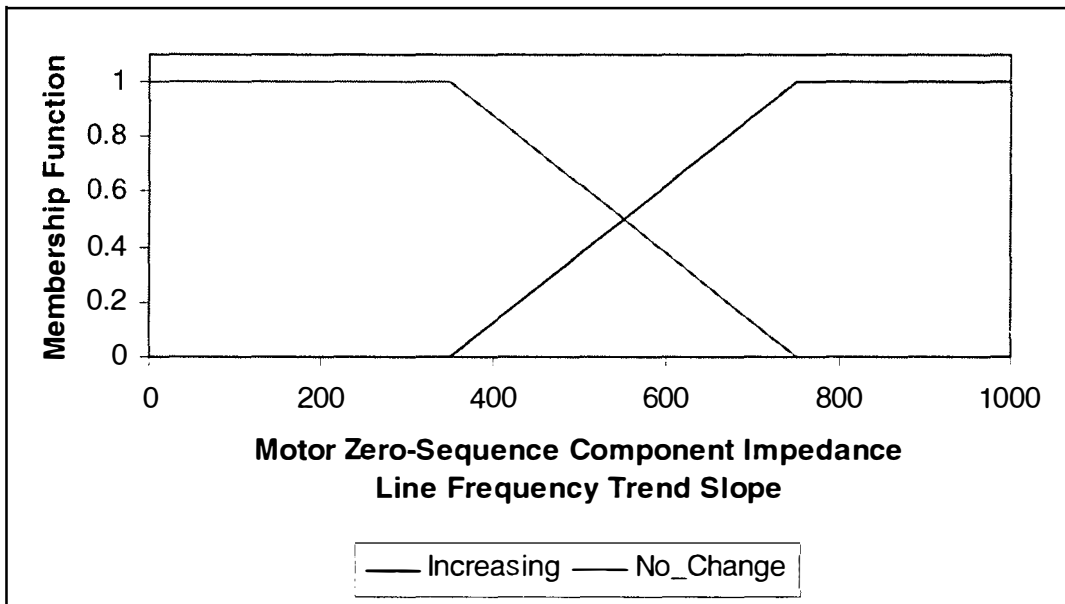


Figure 6.99: Fuzzy membership function for impedance.

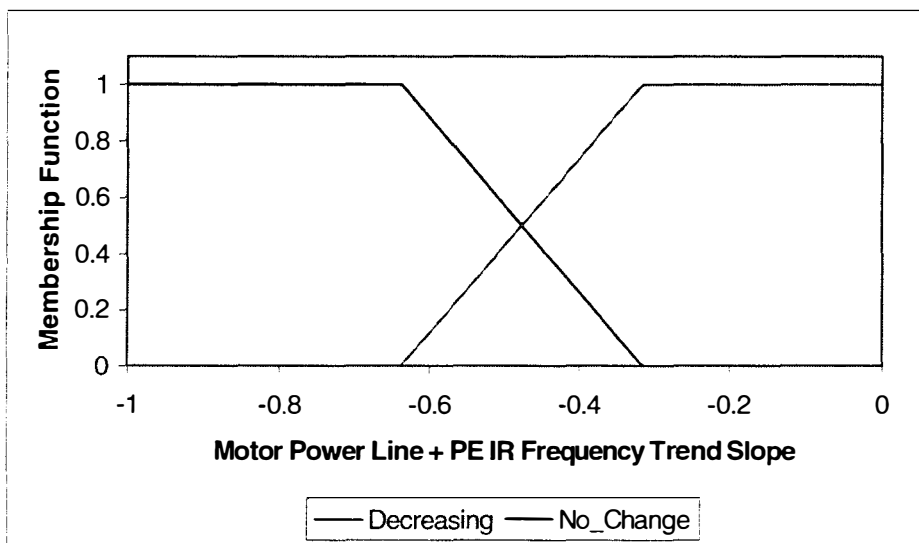
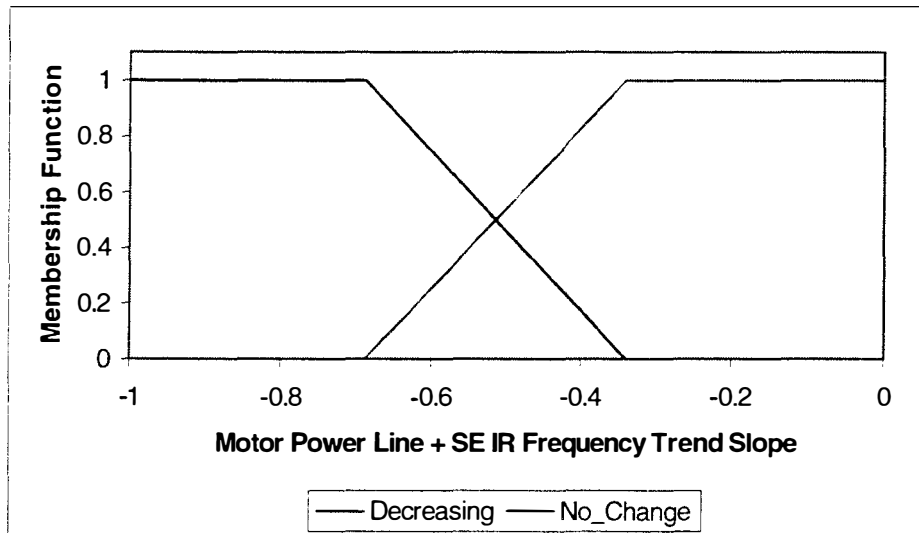


Figure 6.100: Fuzzy membership functions of motor power.

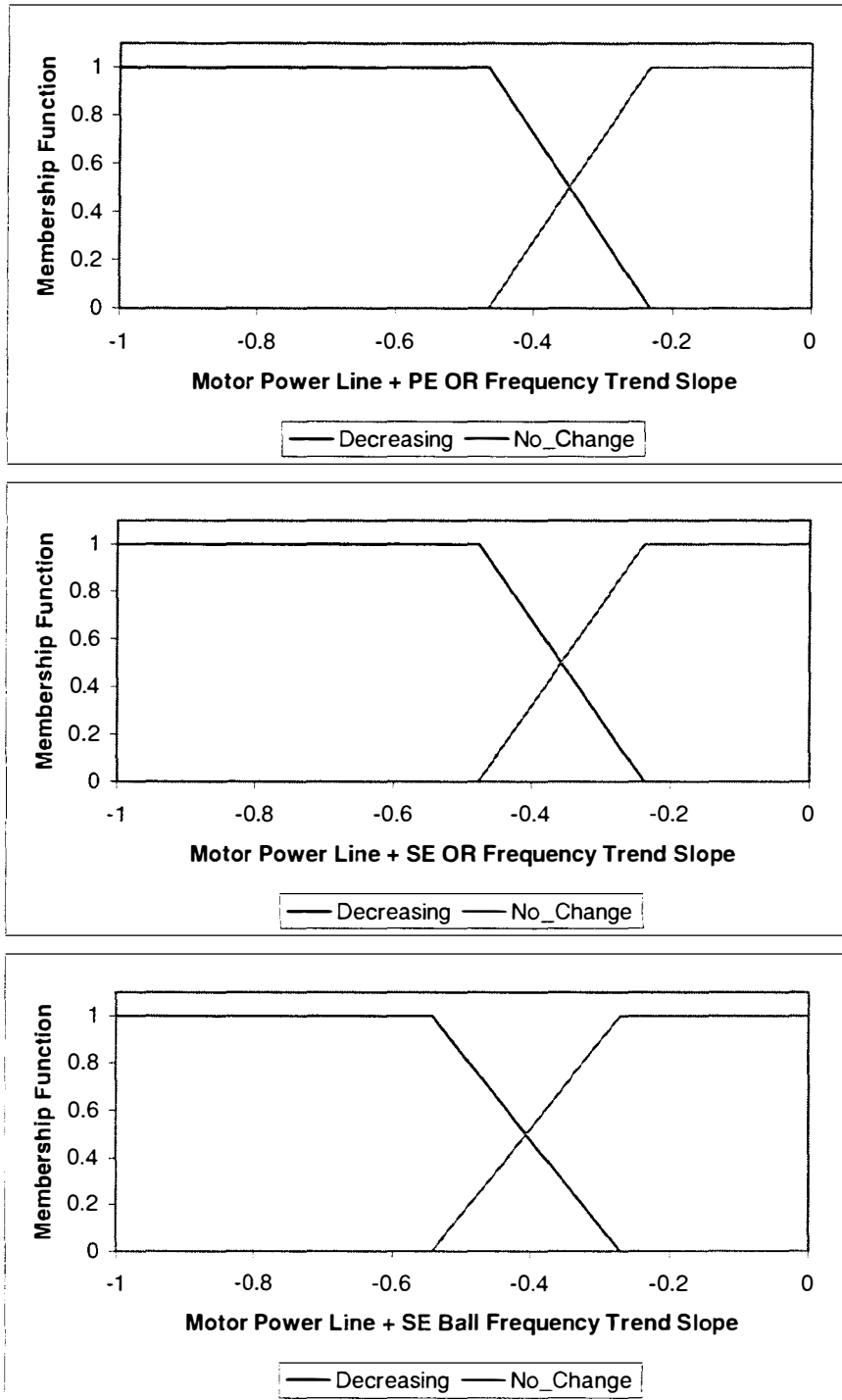


Figure 6.101: Fuzzy membership functions of motor power.

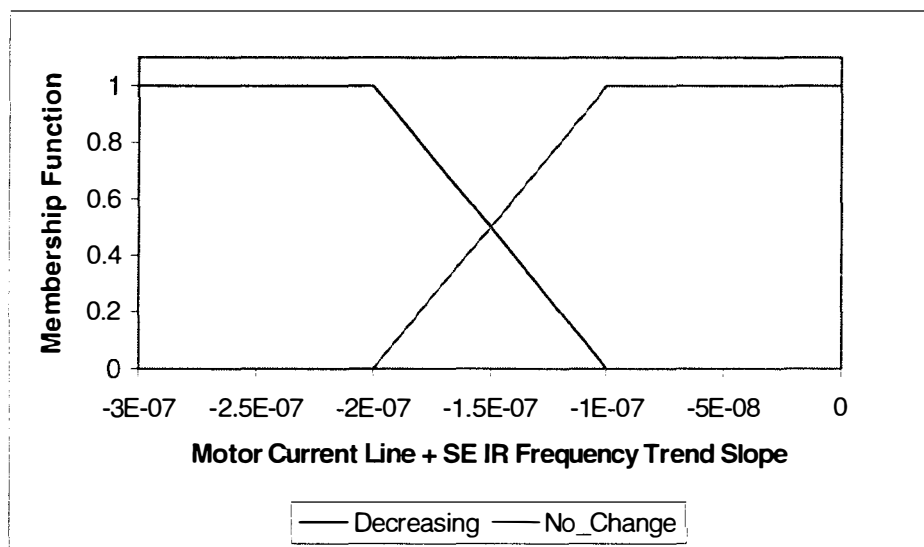
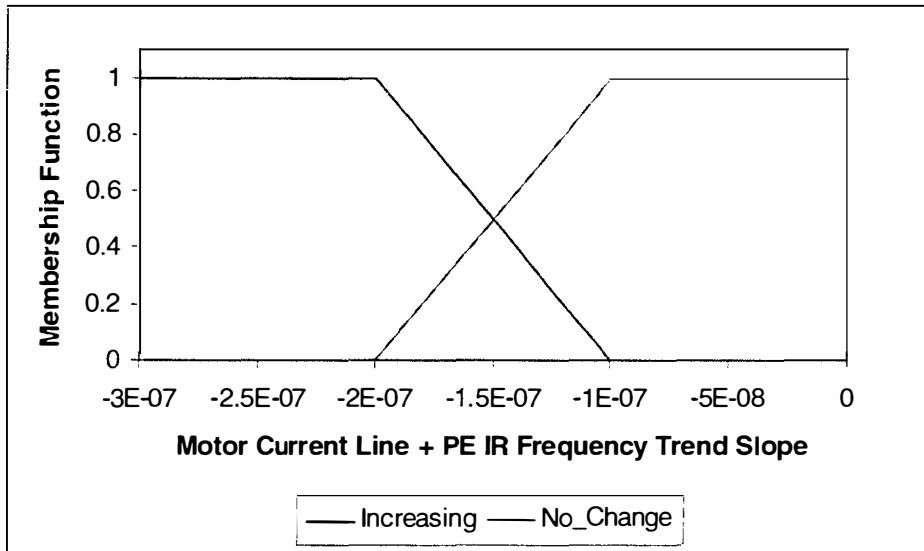


Figure 6.102: Fuzzy membership functions of motor current.

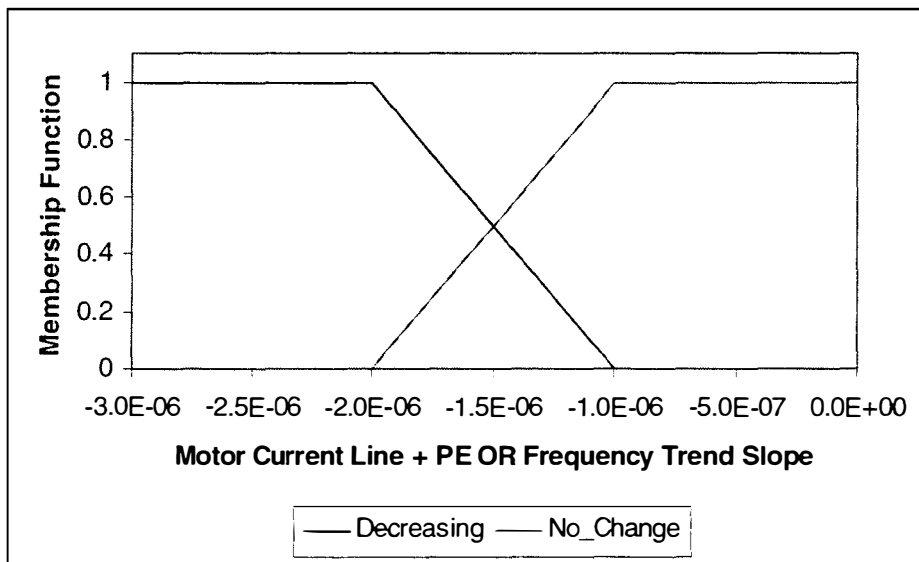
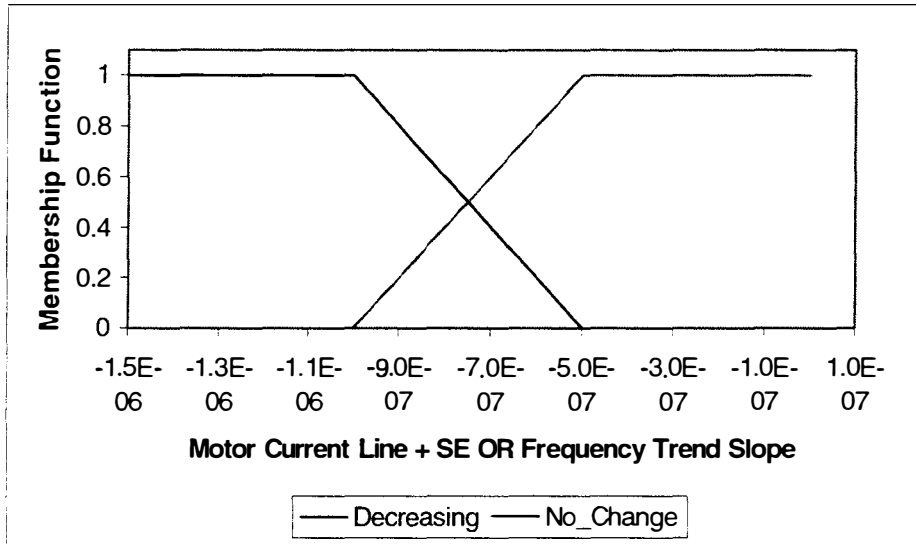


Figure 6.103: Fuzzy membership functions of motor current.

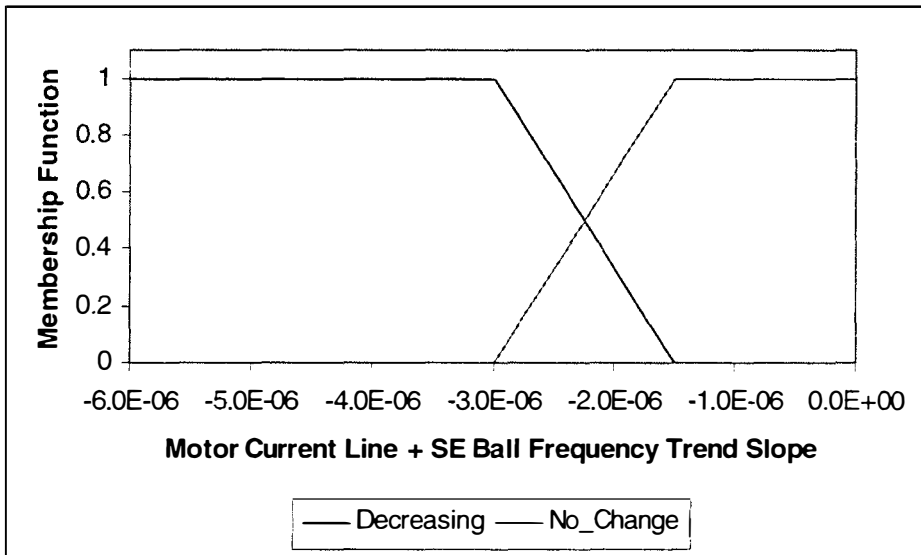
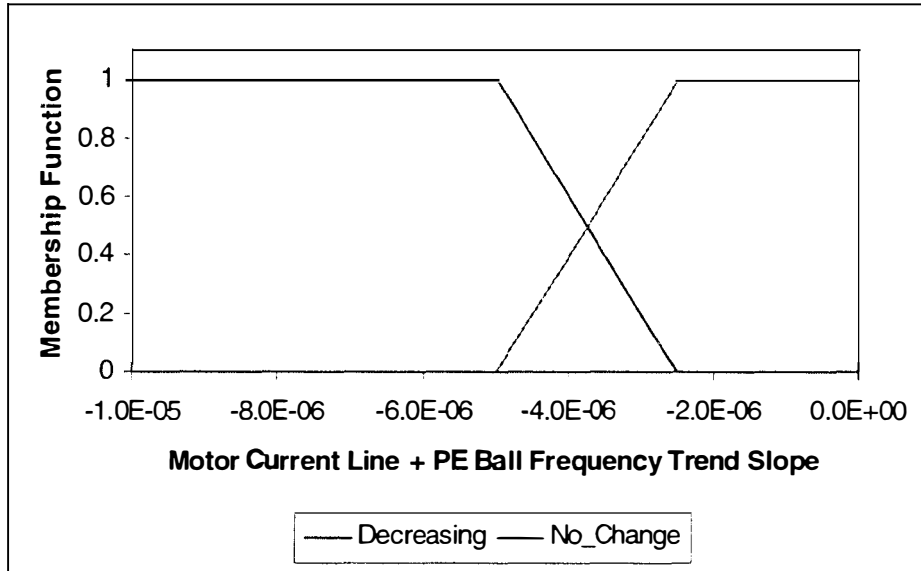


Figure 6.104: Fuzzy membership functions of motor current.

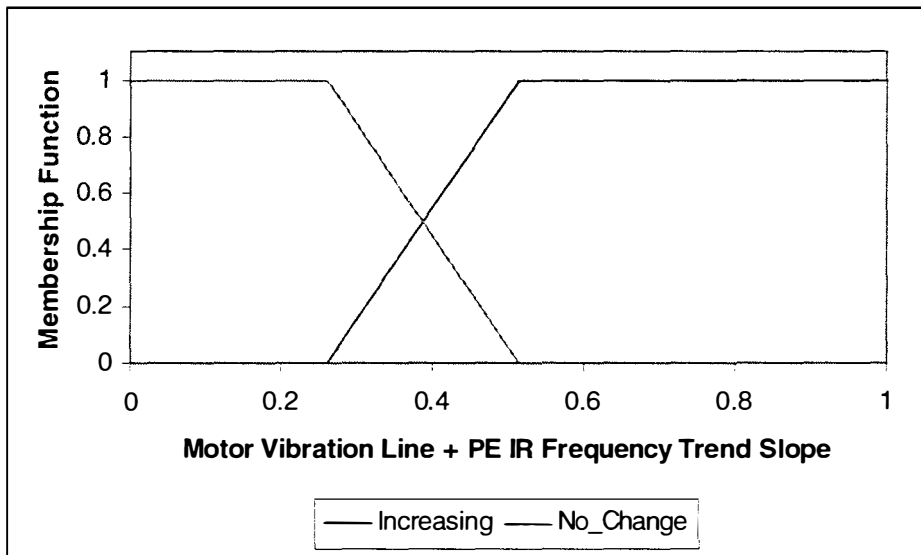
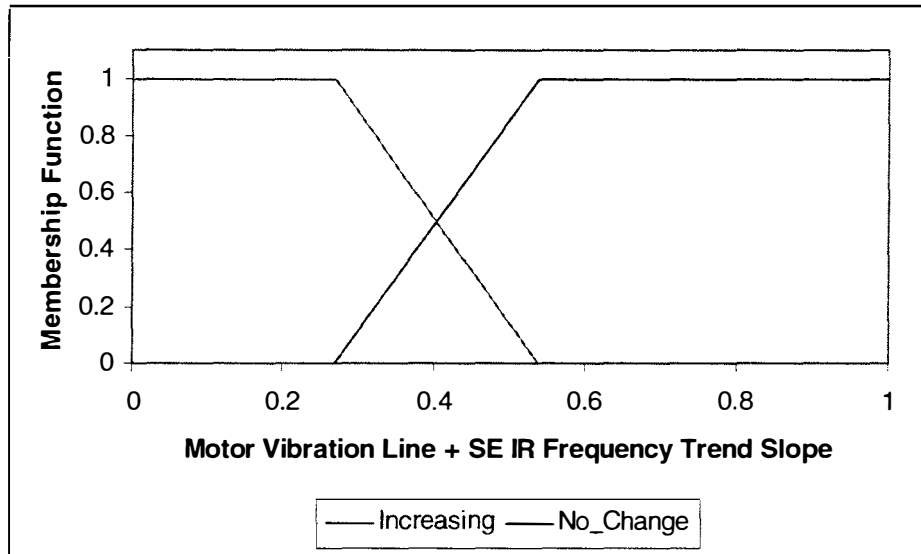


Figure 6.105: Fuzzy membership functions of motor vibration.

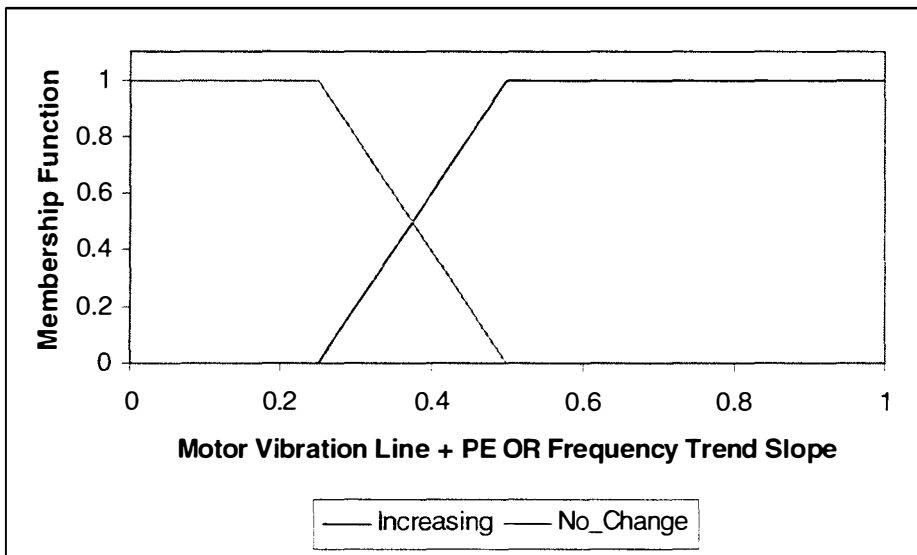
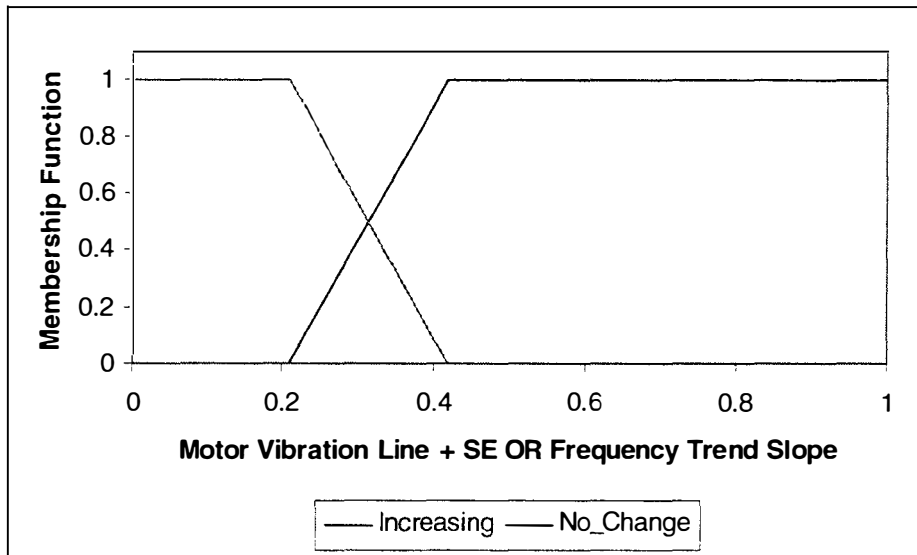


Figure 6.106: Fuzzy membership functions of motor vibration.

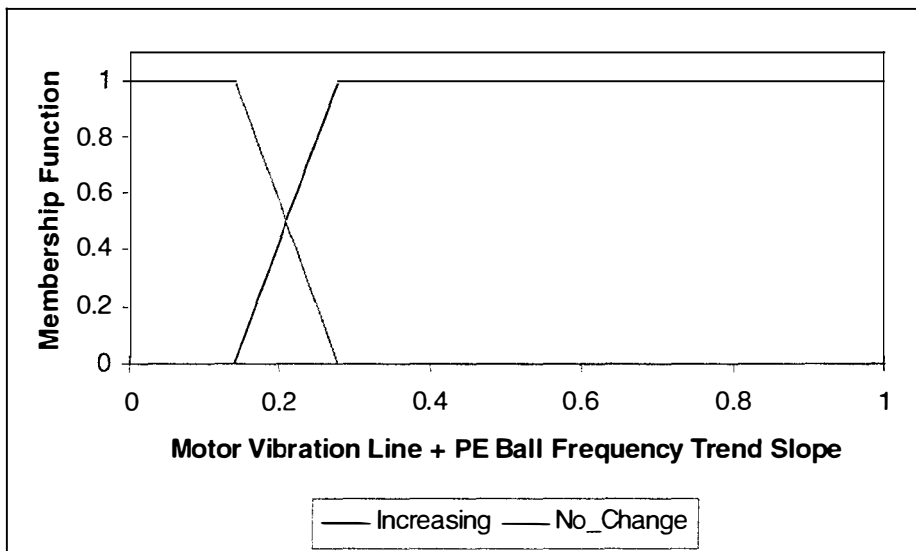
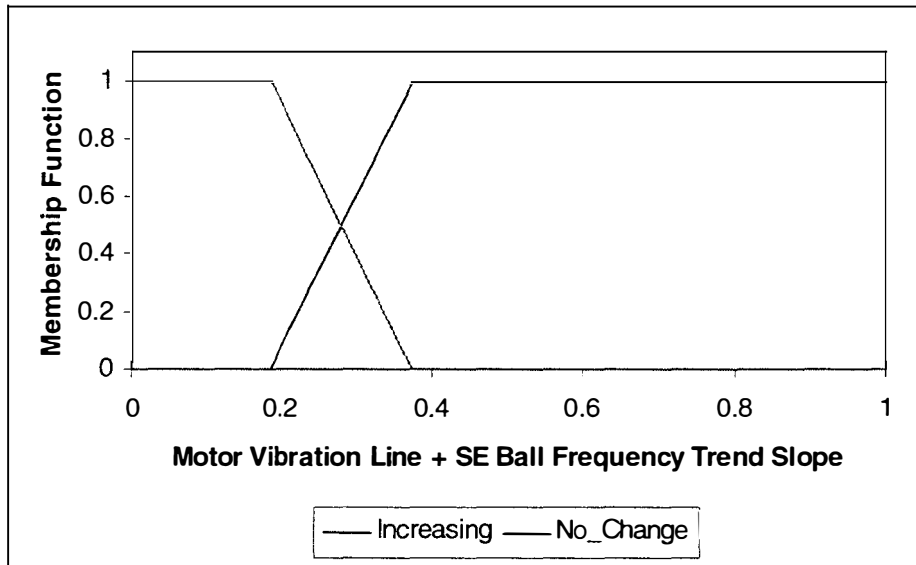


Figure 6.107: Fuzzy membership functions of motor vibration.

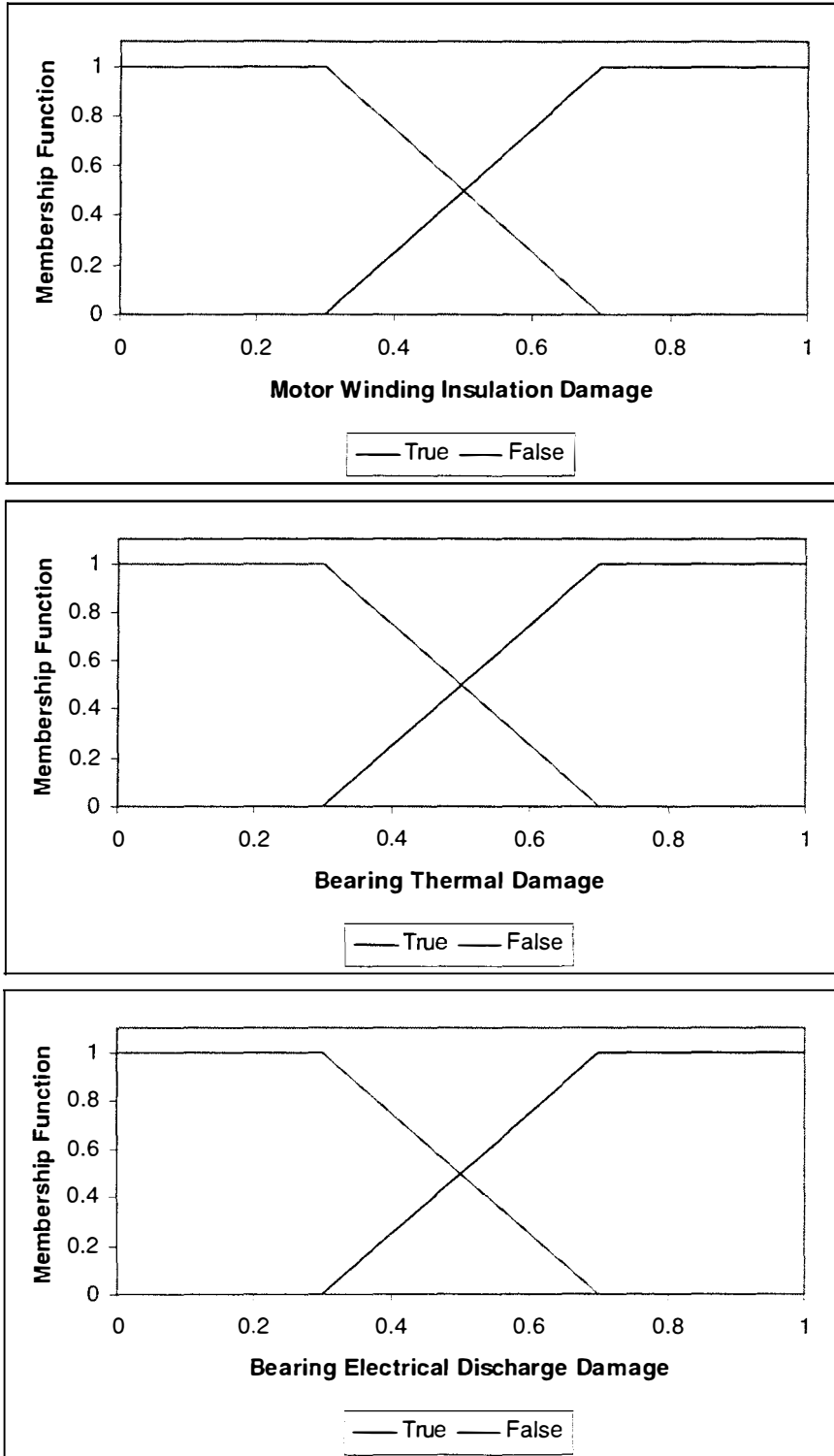


Figure 6.108: Fuzzy output membership functions of fault classification.

Table 6.5: Results of Fuzzy Logic Inferencing

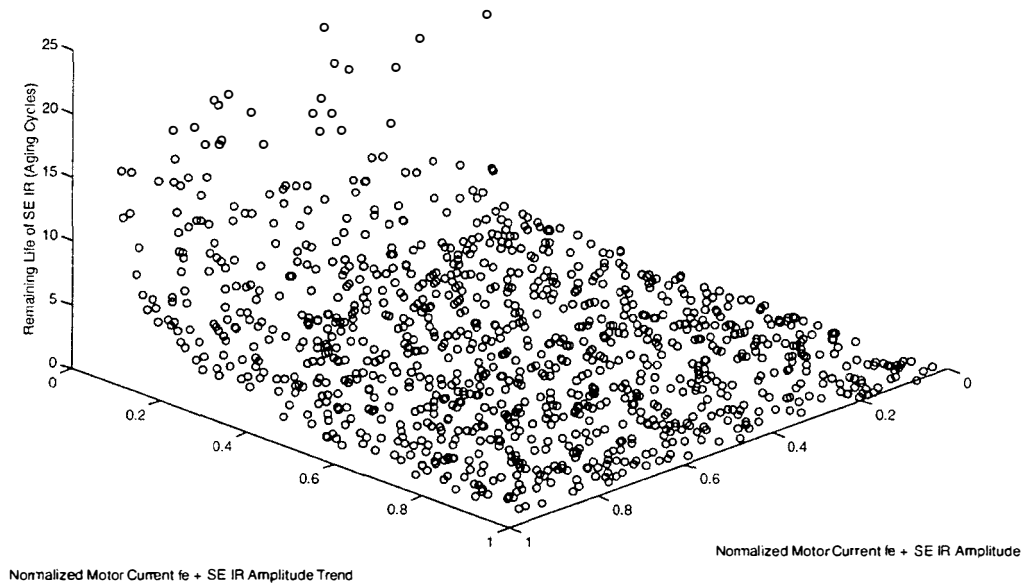
Motor #	Cycle #	Winding Insulation Damage	Electrical and Thermal Damage	Thermal Damage	PE IR Damage	PE OR Damage	PE Ball Damage	SE IR Damage	SE OR Damage	SE Ball Damage
1	0									
1	1			TRUE						
1	2			TRUE						
1	3			TRUE						
1	4			TRUE						
1	5			TRUE						
1	6			TRUE						
1	7			TRUE						
1	8			TRUE						
1	9	TRUE		TRUE						
1	10	TRUE		TRUE						
1	11			TRUE						
2	0									
2	1			TRUE						
2	2			TRUE						
2	3			TRUE						
2	4			TRUE						
2	5			TRUE						
3	0									
3	1			TRUE						
3	2			TRUE						
3	3			TRUE						
3	4			TRUE						
3	5			TRUE						
3	6	TRUE		TRUE						
3	7	TRUE		TRUE						
3	8	TRUE		TRUE						
3	9	TRUE		TRUE						
6	0									
6	1			TRUE						
6	2			TRUE						
6	3			TRUE						
6	4	TRUE		TRUE						
6	5			TRUE						
6	6			TRUE						
6	7			TRUE						
7	0									
7	1			TRUE						
7	2	TRUE		TRUE						
7	3	TRUE		TRUE						
7	4			TRUE						

Table 6.5 Continued

Motor #	Cycle #	Winding Insulation Damage	Electrical and Thermal Damage	Thermal Damage	PE IR Damage	PE OR Damage	PE Ball Damage	SE IR Damage	SE OR Damage	SE Ball Damage
7	5			TRUE						
7	6			TRUE						
7	7			TRUE						
8	0									
8	1			TRUE						
8	2			TRUE						
8	3			TRUE						
8	4			TRUE						
8	5			TRUE						
8	6			TRUE						
8	7			TRUE						
11	0									
11	1		TRUE							
11	2		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
11	3		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
11	4		TRUE							
11	5		TRUE							
11	6		TRUE							
11	7		TRUE							
12	0									
12	1	TRUE	TRUE							
12	2		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
12	3		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
12	4		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
12	5		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
12	6		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
12	7		TRUE		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
13	0									
13	1		TRUE							
13	2		TRUE							
13	3		TRUE							
13	4		TRUE							
13	5		TRUE							
13	6		TRUE							
13	7		TRUE							

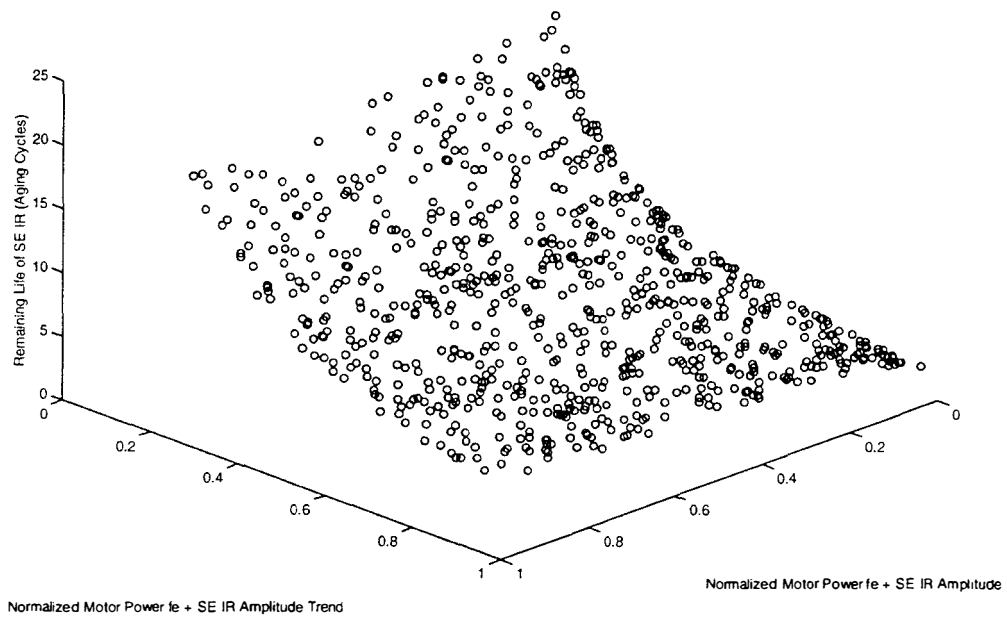
6.7 Remaining Life Estimation Using Artificial Neural Networks

As outlined in Figure 5.2, ANN's are constructed in order to estimate the remaining life of the motor or its particular components. As a demonstration of this technique, the bearing inner race remaining life at the short end of the electrical motor was estimated. The sensitive signatures, which determine the residual life of the component, are the same as those used in fuzzy logic based fault detection and classification. A comprehensive view of these signatures established the total range used as input to the ANN's. The artificially generated training patterns were normalized with respect to their possible maximum values, and the training parameters and patterns for these ANN's are given in Figure 6.109 - Figure 6.111. The results of each individual ANN were introduced to a 'minimum function,' (taking the minimum of different residual lifetime estimations) which would be a conservative estimation of the residual life. The result of this ANN structure in estimating the residual life of the short end bearing inner race is given in Table 6.6 and Figure 6.112. A bad measurement point at cycle #3, as they are also seen in the motor current, power and vibration plots, effected the remaining life calculation of the ANN of Motor #12. Other ANN's for estimating other components of the electrical motor may be constructed in a similar manner. The input signatures for these ANN's would be the same as those used in fuzzy logic based fault detection and classification, given in Table 5.1.



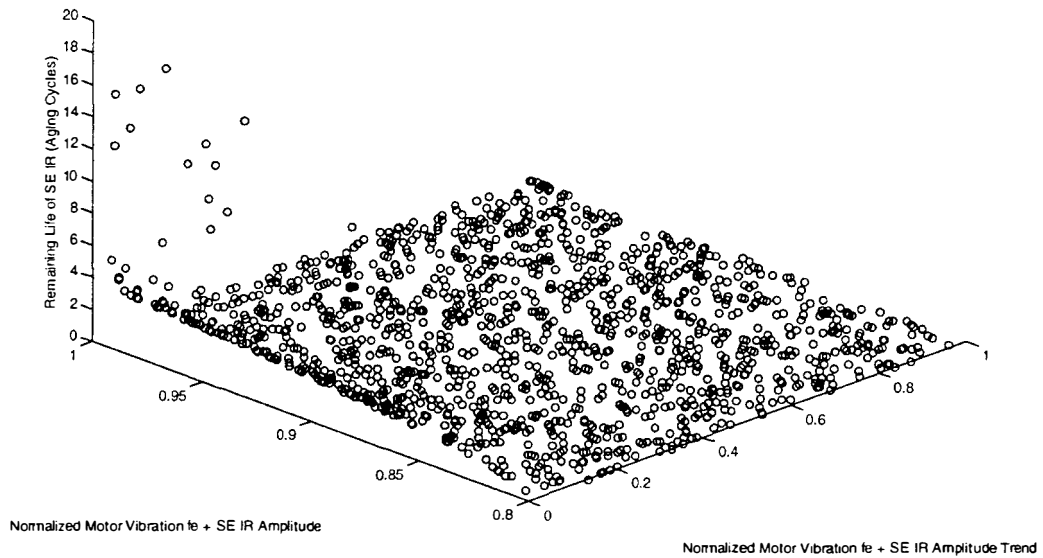
Number of processing elements in Input Layer: 2 (Normalized Signature Amplitude and Signature Trend)
 Number of processing elements in Hidden Layer: 5
 Number of processing elements in Output Layer: 1 (Remaining Life of Component)
 Number of Training Patterns: 1044
 Number of Iterations: 100,000,000
 Average Error in the Trained ANN: 0.06 Time Units (Aging Cycle)

Figure 6.109: Training data set for the ANN using motor current.



Number of processing elements in Input Layer: 2 (Normalized Signature Amplitude and Signature Trend)
 Number of processing elements in Hidden Layer: 5
 Number of processing elements in Output Layer: 1 (Remaining Life of Component)
 Number of Training Patterns: 782
 Number of Iterations: 100,000,000
 Average Error in the Trained ANN: 0.09 Time Units (Aging Cycle)

Figure 6.110: Training data set for the ANN using motor power.



Number of processing element in Input Layer: 2 (Normalized Signature Amplitude and Signature Trend)
 Number of processing element in Hidden Layer: 5
 Number of processing element in Output Layer: 1 (Remaining Life of Component)
 Number of Training Patterns: 1379
 Number of Iterations: 100,000,000
 Average Error in the Trained ANN: 0.09 Time Units (Aging Cycle)

Figure 6.111: Training data set for the ANN using motor vibration.

Table 6.6: Short-End Bearing Inner Ring Residual Life Estimation Using ANN on Motor

#12.

Cycle #	Vibration ANN Estimation	Power ANN Estimation	Current ANN Estimation	Overall Estimation
0	N/A	N/A	N/A	N/A
1		5.91	4.70	4.70
2	4.71	15.01	12.94	4.71
3	0.43	1.54	0.36	0.36
4	0.63	10.34	1.33	0.63
5	2.08	2.32	1.02	1.02
6	0.24	1.28	0.83	0.25
7		0.57	0.52	0.52

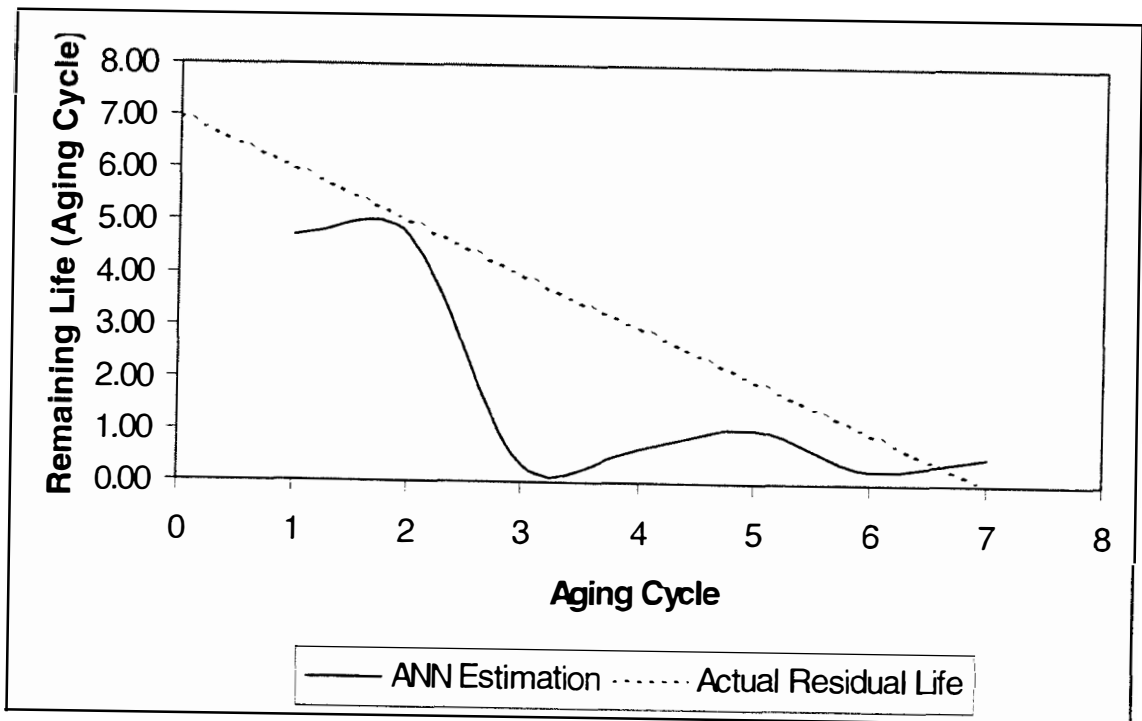


Figure 6.112: Graphical representation of Table 6.6.

6.8 Summary of Results

In this chapter, a systematic approach of constructing an automated system of electrical motor fault detection and remaining life estimation is presented. The accelerated aging tests, as explained in previous chapters constituted the base data in developing these algorithms. The translation of the time domain of these accelerated aging tests are given by Arrhenius Equations as depicted in Figure 3.5 and explained in detail in Section 5.10.

By first establishing a relational database in Microsoft Access, all of the raw data were processed into derived signatures. These derived signatures were examined if an obvious and simple approach of detecting an incipient fault was possible. This was accomplished by SQL queries. For example to select the bearing surface process end temperature of motors subjected to bearing fluting and thermal aging at 100% load, the following query had to be performed, which yielded the result given in Table 6.7.

```
SELECT DISTINCTROW report1.[Date & Time], report1.[Motor #],  
report1.[Cycle #], report1.[BS PE]  
FROM report1  
WHERE (((report1.[Motor #])=11 Or (report1.[Motor #])=12 Or  
(report1.[Motor #])=13) AND ((report1.[Power Level (%)])=100))  
ORDER BY report1.[Motor #], report1.[Cycle #];
```

Table 6.7: Results of SQL query.

Date & Time	Motor #	Cycle #	BS PE
199703131910	11	0	59.971597
199703221537	11	1	55.434036
199704082243	11	2	57.038488
199705151001	11	3	55.601956
199705292339	11	4	56.479363
199706231227	11	5	57.37963
199707171720	11	6	61.416061
199708271506	11	7	58.671179
199703141700	12	0	57.119731
199703232216	12	1	57.354915
199704121441	12	2	56.435981
199705151534	12	3	56.995712
199705301745	12	4	59.932702
199706191452	12	5	59.931469
199707111538	12	6	60.422135
199707241316	12	7	59.576358
199703151742	13	0	57.909454
199703251830	13	1	57.625075
199704131926	13	2	58.367657
199705161150	13	3	56.979938
199705311514	13	4	56.716514
199706241635	13	5	81.653686
199707181322	13	6	59.66445
199708281726	13	7	60.673679

However, the detection of an incipient fault turned out to be not that easy, as making just SQL queries. Also, since the accelerated aging tests were performed at specific load levels, for the implementation in real life, the algorithms had to be developed for all load levels. This is accomplished by normalizing the data with respect to the current load level. Previous experimental studies assumed that this normalization be linear, however, the use of a relational database and making the correct queries, we found out that the relationship between these signatures are not linear for most situations. The established normalization functions given in this chapter are unique with respect to the signatures, to the measuring devices and to the electrical motors used. In this way, sensor level and feature level data fusion has been established.

After the normalization of the selected signatures, they were trended with respect to time. Most of the trends were least-square fits using linear and exponential functions. The slope of these functions determines the incipient fault detection in electrical motors. Hence this was the input into the fuzzy logic based fault detection and classification algorithm. The alarm limits, which were established during the trending of these signatures constituted another input to the ANN's for remaining life calculations. At this final level, symbol level data fusion was accomplished.

Chapter 7

SUMMARY, CONCLUSIONS AND FUTURE WORK

7.1 Summary

On-line signatures and features of motor failure are established by observing electrical motors during their degradation. Accelerated aging methods, based on the Arrhenius model, were used to simulate real-life conditions. Three forms of aging of motors were considered. The following aging experiments were performed on 5-HP three-phase squirrel cage motors.

- Stator insulation thermal aging by continuous automated on-off switching by single-phase power supply. The insulation temperature was maintained at ~ 205 °C.
- Bearing thermal aging at ~140 °C, with induced corrosion by water soaking.
- Bearing fluting aging by shaft current technique combined with thermal aging.

These failure mechanisms were carefully recorded by a performance test system explained in this dissertation.

Thereafter, a relational database was established in which signal features were recorded. These features were normalized with respect to load and if possible with

respect to ambient temperature. These signatures were then trended using linear and exponential function least-square fits.

Thereafter, the 'sensitivity matrix,' showing all of the techniques used in this study and the relevance to a particular fault mode, was established. Using this tabulation and the trend slopes, a fuzzy logic based fault detection and classification algorithm was developed. The fuzzy logic rule base was able to successfully detect and classify the aforementioned induction motor faults.

ANN's were also used to estimate the remaining life of a particular component in an electric motor. The input signatures of the ANN's are the same as those used in the fuzzy logic based fault detection. Also, the alarm limits, which were established in the signature trends, were used as input.

7.2 Conclusions

In general, the results obtained from the studies in this dissertation have shown the feasibility of implementing data fusion for electric motor fault detection. A laboratory was established at The University of Tennessee in order to simulate the failure mechanisms of motors. This laboratory contained accelerated aging setups as well as a state-of-the-art performance test system which is used for recording electrical,

mechanical and thermal properties on-line. The database of measurements, which consists of raw and higher-level derived signatures, is unique in the field of this study.

The analysis of the measured variables indicated the following:

- Multi-resolution analysis of vibration indicated clearly the shift in spectral energy with degradation: As the electric motor progresses towards failure via electrical discharge mechanism, the spectral energy between 2 – 4 kHz also increases and dominates most of the spectrum.
- Zero-sequence component analysis of impedance and magnetic flux measurements showed indications of winding insulation failure. Until recent years, no commercial methodology existed for detecting an ‘incipient failure’ of this nature when most of the failure modes occur in the winding insulation.
- Efficiency was found to be an indicator of failure and not an indicator of ‘incipient failure.’
- The use of motor current and power yields complimentary results. While the trends for motor current were linear, the trends for motor power were exponential.

This is a result of Ohm’s law:

$$Power = Resistance \times (Current)^2 \quad (7.1)$$

- Monitoring of bearings using thermocouples provided an efficient way of detecting an incipient fault. However, thermal analysis should be accompanied by vibration analysis in order to increase the confidence of fault detection and classification.
- Normalization of signatures with respect to load was found to be non-linear in most of the cases. This is an important observation, as most of the previous studies in literature used a linear normalization with respect to load.
- Normalization also changes with respect to the position of the accelerometer or the temperature sensor because of the thermal and mechanical changes in the motor.
- The use of permanent mount sensors reduces the variation of measurements. Variations in vibrations were caused as a result of loose sensor mountings and not using the same specific sensor at the same location for each performance test. Variations in temperature were as a result of semi-steady state operation of the motor.
- The use of fuzzy logic based fault detection and classification method developed in this research was implemented successfully with a few false alarms (in winding insulation). This was a result of insufficient data set from magnetic flux

measurements, which should compliment zero-sequence component analysis.

- The coherence function between motor power and load power and the coherence function between motor power and motor vibration showed physically meaningful degradation trends of failure, however, it alone cannot classify fault degradation.

- If we had to sort some of the sensor measurements with respect to the amount of information they carry, from least to most important, the sequence would be:
 1. Temperature
 2. Current
 3. Power
 4. Vibration.

This grading varies as a function of the fault type.

In this study, most of the on-line motor performance techniques were integrated with some new techniques. Data fusion through sensory, feature and symbolic level provided a robust decision and detection methodology for electrical motors. Fuzzy logic based fault detection and classification as well as ANN based remaining life estimation were successfully implemented in this study.

7.3 Recommendations for Future Work

Although this study has been an important milestone in the field of electrical motor failure detection, opportunities for future work are extensive. Some of these are given below:

- This study has mainly concentrated on the steady-state operation of the motor. However, start-up and shut-down data were also collected during the tests. Further study of this transient data, by means of time-frequency analysis, may yield additional features, which can be used in fault detection. A preliminary study by James P. McClanahan is shown in Figure 7.1 - Figure 7.3. Trending of frequency time features of specific signals may provide additional insight into motor degradation.
- More experimental studies need to be conducted with magnetic flux sensors in order to obtain better statistical features.
- Other ANN's can be constructed to predict the residual lifetime of particular components of the electric motors. The necessary method has been established in this dissertation.

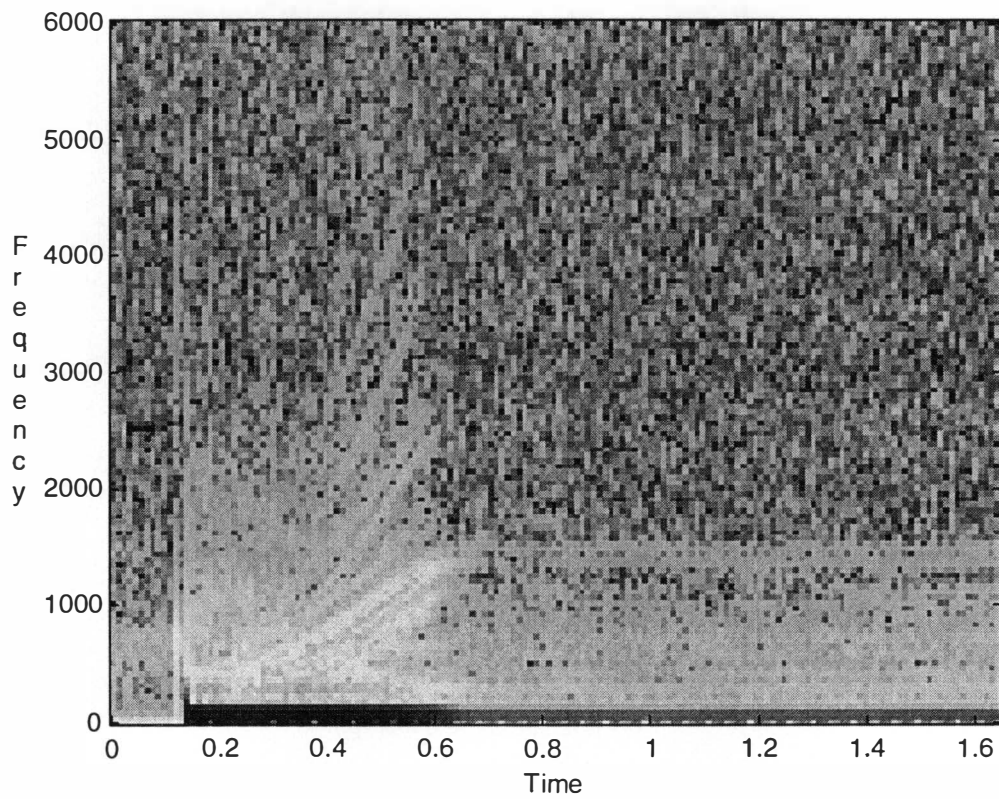


Figure 7.1: Time – frequency analysis of motor start-up data using motor current.

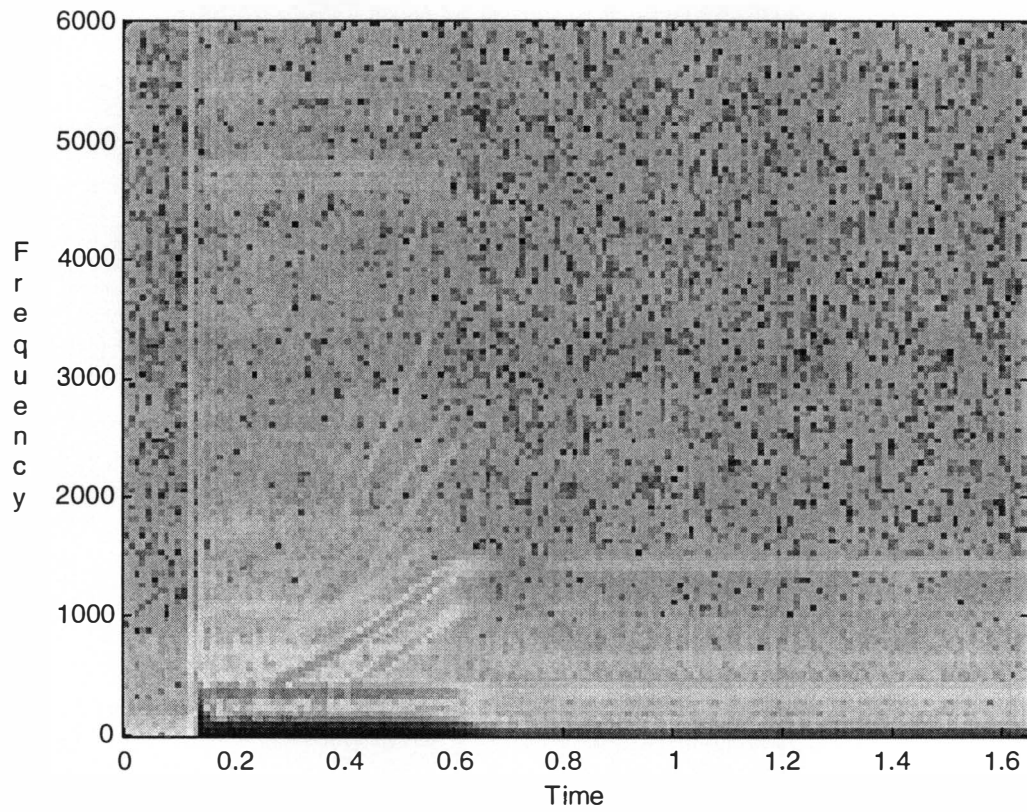


Figure 7.2: Time – frequency analysis of motor start-up data using motor power.

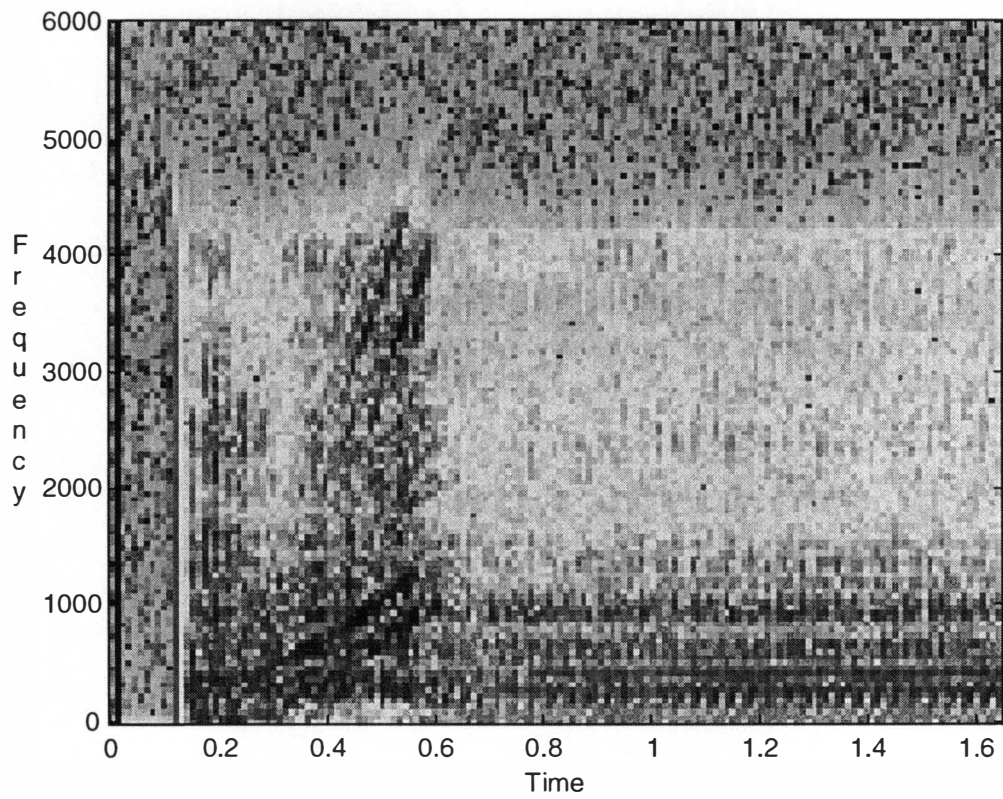


Figure 7.3: Time – frequency analysis of motor start-up data using motor vibration.

- The study of this dissertation can be extended to other failure modes of electric motors. However, the failure due to the causes is small compared to the failure mode investigated in this study (see Table 1.2).
- Further features and measurements can be incorporated into the IMMS easily. For example, leakage voltage can be measured as an additional feature in detecting winding insulation degradation and incorporated into the fuzzy logic based fault detection algorithms as well as into remaining lifetime estimations.

REFERENCES

REFERENCES

1. M. Subuthi, J. H. Taylor, R. Lofaro, A. C. Sugarman, M. W. Sheets and K. M. Skreiner, "Improving Motor Reliability in Nuclear Power Plants, Volume 1: Performance Evaluation and Maintenance Practices," U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Contract No. DE-AC02-76CH00016 FIN A-3270, November 1987.
2. M. Subuthi, J. H. Taylor, R. Lofaro, A. C. Sugarman, M. W. Sheets and K. M. Skreiner, "Improving Motor Reliability in Nuclear Power Plants, Volume 2: Functional Indicator Tests on a Small Electric Motor Subjected to Accelerated Aging," U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Contract No. DE-AC02-76CH00016 FIN A-3270, November 1987.
3. M. Subuthi, J. H. Taylor, R. Lofaro, A. C. Sugarman, M. W. Sheets and K. M. Skreiner, "Improving Motor Reliability in Nuclear Power Plants, Volume 1: Failure Analysis and Diagnostics Tests on a Naturally Aged Large Electric Motor," U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Contract No. DE-AC02-76CH00016 FIN A-3270, November 1987.
4. J. R. Nicholas, Jr., "Predictive Condition Monitoring of Electric Motors," *P/PM Technology*, pp. 28 - 32, August 1993.
5. IAS Motor Reliability Working Group, "Report of Large Motor Reliability Survey of Industrial and Commercial Installations, Part I," *IEEE Transactions on Industry Applications*, Vol. IA-21, No. 4, pp. 853 - 864, July 1985.
6. A. H. Bonnett and G.C. Soukup, "Cause and Analysis of Stator and Rotor Failures in Three-Phase Squirrel-Cage Induction Motors," *IEEE Transactions on Industry Applications*, Vol. 28, No. 4, pp. 921 - 937, August 1992.
7. J. D. Kueck, D. A. Casada and P. J. Otaduy, "A Comparison of Two Energy Efficient Motors," ORNL, Nov. 1995.
8. S. F. Farag, R.G. Bartheld and T.G. Habetler, "An Integrated On-Line Motor Protection System," *IEEE Industry Applications Magazine*, pp. 21 - 26, March/April 1996.
9. P. F. Albrecht, J. C. Appiarius, E. P. Cornell, D. W. Houghtaling, R. M. McCoy, E. L. Owen, "Assessment of The Reliability of Motors in Utility Applications," *IEEE Transactions on Energy Conversion*, Vol. EC-2, No. 3, pp. 396 - 402, September 1987.

10. "IEEE Std 117-1974/ANSI C50.32-1976. IEEE Standard Test Procedure for Evaluation of Systems of Insulation Materials for Random-Wound AC Electric Machinery," 1976.
11. "UL 1446. Fifth Edition of the Proposed Standard for Systems of Insulation Materials – General," Underwriters Laboratory, November 1996.
12. G. C. Stove, H. G. Sedding, B. A. Lloyd and K. Gupta, "The Ability of Diagnostic Tests to Estimate The Remaining Life of Stator Insulation," IEEE Transactions on energy Conversion, Vol. 3, No. 4, pp. 833 – 841, December 1988.
13. B. F. Kane, B. R. Lease and A. Golubev, "Practical Applications of Periodic Monitoring of Electrical Equipment for Partial Discharges," P/PM Technology, pp. 46 – 52, June 1997
14. Status Technologies, "MotorStatus Information Sheet," 1997.
15. S. V. Bowers, "Integrated Strategy for Predictive Maintenance of AC Induction Motors," P/PM Technology, pp. 34 - 40, October 1996.
16. S. V. Bowers and K. R. Piety, "Proactive Motor Monitoring Through Temperature, Shaft Current and Magnetic Flux Measurements," Computational Systems, Inc. 1993 Users Conference, pp. 20 – 24, September 20 - 24, 1993.
17. P. Cooper and D. Roberts, "Vibration Monitoring Justification Based on Saves," P/PM Technology, pp. 62 - 63, August 1996.
18. S. J. Richards, "Motor Vibration Analysis: Key to Effective Troubleshooting," Power, pp. 46 - 48, January 1988.
19. R. Schoen, T. G. Habetler, F. Kamran and R. G. Bartheld, "Motor Bearing Damage Detection Using Stator Current Monitoring," IEEE Transactions on Industrial Applications, Vol. 1, No. 1, pp. 110 - 116, 1994.
20. S. Danehover and J. Mitchell, "Real Time Acquisition and Storage of Vibration Information – Justification," P/PM Technology, pp. 42 – 45, June 1997.
21. A. E. Preusser, G. L. Hadley, "Motor Current Signature Analysis as a Predictive Maintenance Tool," Proceedings of the American Power Conference, pp. 286 - 291.
22. R. C. Kryter and H. D. Haynes, "How to Monitor Motor-Driven Machinery by Analyzing Motor Current," Power Engineering, pp. 35 - 39, October 1989.

23. Iris Power Engineering, "Motor stator Monitoring Complements Bearing and Rotor Monitoring," *Maintenance Technology*, pp. 33 - 34, June 1997.
24. N. M. Burstein, "ESA Spots Power Sag Transmitted By Electric Circuit," *Maintenance Technology*, pp. 35 – 37, June 1997.
25. N. M. Burstein, "ESA Reveals Low-Frequency Vibration," *Maintenance Technology*, pp. 41 – 42, March 1997.
26. R. R. Schoen, B. K. Lin, T. G. Habetler, J. H. Schlag and S. Farag, "An Unsupervised, On-Line System For Induction Motor Fault Detection Using Stator Current Monitoring," *IEEE Transactions on Industrial Applications*, Vol. 1, No. 1, pp. 103 - 109, 1994.
27. J. Berry, "Detection of Multiple Cracked Rotor Bars on Induction Motors Using Both Vibration and Motor Current Analysis," *P/PM Technology*, pp. 48 - 59, June 1996.
28. M. DelZingaro and C. Matthews, "Using Power Signature Analysis to Detect the Behavior of Electric Motors and Motor-driven Machines," *P/PM Technology*, pp. 32 - 34, August 1995.
29. S. F. Legowski, A. H. M. S. Ula and A. M. Trzynadlowski, "Instantaneous Power as a Medium for the Signature Analysis of Induction Motors," *IEEE Transactions on Industry Applications*, Vol. 32, No. 4, pp. 904 - 909, July / August 1996.
30. R. E. Thomas, "When is a Life Test Truly Accelerated?" *Electronic Design* pp. 64 - 71, January 6, 1964.
31. Motor Task Group, "Work-In-Progress Report on Maintenance Good Practices For Motors in Nuclear Power Generating Stations," *IEEE Transactions on Energy Conversion*, Vol. 3, No. 3, pp. 589 - 600, September 1988.
32. V. Divljakovic, M. Henderson and R. Hannula, "Prediction of Machine Failure from Accelerated Aging Test Data," *Machinery Conference, Society of Machinery Failure Prevention Technology*, Virginia Beach, VA, pp. 431 – 440, March 30 – April 3, 1998.
33. G. T. Homce, "Early Detection in Insulation Failure in Electric Motors," *United States Department of The Interior Bureau of Mines Information Circular*, IC9211, 1993.
34. V. Lyon, "Transient Analysis of Alternating-Current Machinery, An Application of The Method of Symmetrical Components," MIT 1954.

35. R. Natarajan, "Failure Identification of Induction Motors By Sensing Unbalanced Stator Currents," IEEE Transactions on energy Conversion, Vol. 4, No. 4, pp. 585 – 590, December 1989.
36. S. Williamson and K. Mirzoian, "Analysis of Cage Induction Motors with Stator Winding Faults," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 7, July 1985.
37. M. A. Abidi and R. C. Gonzalez, "Data Fusion in Robotics and Machine Intelligence," Academic Press, 1992.
38. B. R. Upadhyaya, A. S. Erbay and J. P. McClanahan, "Accelerated Aging Studies of Induction Motors and Fault Diagnosis," The University of Tennessee Maintenance and Reliability Center Annual Report, MRC/BRU/96-03, September 1997.
39. G. F. Lang, "3-Phase AC Motor Primer," P/PM Technology, pp. 54 – 57, August 1997.
40. P. F. Albrecht, J. C. Appiarius, E. P. Cornell, D. W. Houghtaling, R. M. McCoy, E. L. Owen, "Assessment of The Reliability of Motors in Utility Applications," IEEE Transactions on Energy Conversion, Vol. EC-2, No. 3, pp. 396 - 402, September 1987.
41. "Improved Motors for Utility Applications," Vol. 1, EL-2678, EPRI Research Project 1763-1, Final Report, October 1982.
42. K. R. Cho, J. H. Lang and S. D. Umans, "Detection of Broken Rotor Bars in Induction Motors Using State and Parameter Estimation," IEEE Transactions on Industry Applications, Vol. 28, No. 3, pp. 702 - 709, May / June 1992.
43. B. Raychaudhuri, "Fault Monitoring and Residual Life Estimation of Electric Motors," MS Thesis, The University of Tennessee, Knoxville, 1995.
44. G. A. Bisbee, "Why Do Motor Shafts and Bearings Fail?" Tappi Journal, Vol. 77, No. 9, pp. 251 - 252, September 1994.
45. "Proposal of the Fifth Edition of The Standard for Systems of Insulation Materials – General, UL 1446," Underwriters Laboratories Inc., January 1997.
46. R. P. Derrick, W. J. Martiny and W. J. McDonald, "Motors covered with Pulp: The Effect on Motor Life," Tappi Journal, pp. 57 - 64, May 1987.

47. S. W. Bowers, K. R. Piety, R. W. Piety and R. J. Colsher, "Evaluation of the Field Application of Motor Current Analysis," Proceedings of the Meeting of the Vibration Institute, 1993.
48. A. Bradley, "Reliability Based Maintenance," Computation Systems Incorporated, The University of Tennessee, Knoxville, NE494 class notes, 1992.
49. A. Kowal, "Bearing Damage Resulting from Shaft Voltages and Currents," Computational Systems, Inc., 1997.
50. J. S. Hsu, "Monitoring of Defects in Induction Motors Through Air – Gap Torque Observation," IEEE Transactions on Industry Applications, Vol. 31, No. 5, pp. 1016 – 1021, September / October 1995.
51. J. Kochensparger, "Applying Predictive Maintenance Testing to Minimize Motor Failure Downtime," Plant Engineering, pp. 186 - 190, March 12, 1987.
52. A. Russo and G. Ramponi, "A Totally Fuzzy Approach to Multi-sensor Instrumentation: Pre-Processing Techniques," IEEE Instrumentation and Measurement Technology Conference, Japan, Vol. 3, pp. 1143 – 1146, May 10 – 12, 1994.
53. A. Russo, "A Fuzzy Approach to Digital Signal Processing: Concepts and Applications," IEEE Instrumentation and Measurement Technology Conference, USA, Vol. 2, pp. 640 – 645, May 12 – 14, 1992.
54. A. Viot, "Fuzzy Logic: Concepts to Constructs," AI Expert, pp. 26 - 33, November 1993.
55. S. Seker, B. R. Upadhyaya, A. S. Erbay, J. P. McClanahan and A. A. DaSilva, "Rotating Machinery Monitoring and Degradation Trending Using Wavelet Transforms," Maintenance and Reliability CONference, Knoxville, Tennessee, pp. 23.01 – 23.11, May 12 – 14, 1998.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. **“Electrical Formulas,”** Brownel Electro Inc. Product Catalog, 1996.
2. **“Motor Power Signature Analysis,”** Duke Power Company, 1997.
3. F. Russo and G. Ramponi, **“A Totally Fuzzy Approach to Multisensor Instrumentation: Pre-Processing Techniques”** IEEE Instrumentation and Measurement Technology Conference, Vol. 3, Homamatsu, Japan, May 10 - 12, 1994.
4. F. Russo, **“A Fuzzy Approach to Digital Signal Processing: Concepts and Applications,”** IEEE Instrumentation and Measurement Technology Conference, New York, USA, May 12 – 14, 1992.
5. D. Walch and W. Bastl, **“On-Line Condition Monitoring of Large Rotating Machinery in NPPs,”** Proceedings of the 1996 American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, pp. 1313-1320, The Pennsylvania State University, May 6-9, 1996.
6. R. Maier, **“Protection of Squirrel-Cage Induction Motor Utilizing Instantaneous Power and Phase Information,”** IEEE Transactions on Industrial Applications, Vol. 28, No. 2, pp. 376-380, March / April 1992.
7. K. R. Cho, J. H. Lang and S. D. Umans, **“Detection of Broken Rotor Bars in Induction Motors Using State and Parameter Estimation,”** IEEE Transactions on Industry Applications, Vol. 28, No. 3, pp. 702-709, May / June 1992.
8. **“Motors,”** Power Transmission Design, pp. A303-A311, 1996.
9. B. E. Preusser, G. L. Hadley, **“Motor Current Signature Analysis as a Predictive Maintenance Tool,”** Proceedings of the American Power Conference, pp. 286-291.
10. D. Strugar and R. Weiss, **“Why Electric Motors Fail,”** Plant Engineering, pp. 65-66, July 1994.
11. P. Walker, **“Preventing Motor Shaft-Current Bearing Failures,”** Plant Engineering, pp. 100-103, December 1991.
12. S. J. Richards, **“Motor Vibration Analysis: Key to Effective Troubleshooting,”** Power, pp. 46-48, January 1988.
13. M. J. Dowling, **“Application of Non-stationary Analysis to Machinery Monitoring,”** pp. I59-I62.

14. C. V. Doreswamy and R. Doreswamy, **“A Method of Calculating Transient Short Circuit Currents in Electrical Penetration Assemblies of Nuclear Power Generating Stations,”** IEEE Proceedings of 1990 Southeastcon, pp. 332-336, 1990.
15. H. W. Lorenzen, R. Nuscheler and A. Meyer, **“Experimental Investigation of The Operating Behavior of Linear Synchronous Motors,”** IEEE Transactions on Energy Conversion, Vol. EC-1, No. 4, pp. 73-82, December 1986.
16. W. R. Finley, R. R. Burke, **“Troubleshooting Motor Problems,”** IEEE Transactions on Industry Applications, Vol. 30, No. 5, pp. 1383-1396, September / October 1994.
17. G. A. Bisbee, **“Why Do Motor Shafts and Bearings Fail?”** Tappi Journal, Vol. 77, No. 9, pp. 251-252, September 1994.
18. R. P. Derrick, W. J. Martiny and W. J. McDonald, **“Motors covered with Pulp: The Effect on Motor Life,”** Tappi Journal, pp. 57-64, May 1987.
19. M. R. Cohen, **“Predictive Maintenance of Operational Motors Using Digital Data Collection and Trend Analysis,”** Tappi Journal, pp. 151-155, January 1991.
20. J. Kochensparger, **“Applying Predictive Maintenance Testing to Minimize Motor Failure Downtime,”** Plant Engineering, pp. 186-190, March 12, 1987.
21. P. Lapsey and G. Blalock, **“How to Estimate DSP Processor Performance,”** IEEE Spectrum, pp. 74-78, July 1996.
22. K. Karnofsky, **“Speeding DSP Algorithm Design,”** IEEE Spectrum, pp. 79-82, July 1996.
23. R. Schoen, T. G. Habetler, F. Kamran and R. G. Bartheld, **“Motor Bearing Damage Detection Using Stator Current Monitoring,”** 1994 IEEE Industrial Applications Meeting, Vol. 1, pp. 110-116, 1994.
24. J. Berry, **“Detection of Multiple Cracked Rotor Bars on Induction Motors Using Both Vibration and Motor Current Analysis,”** P/PM Technology, pp. 48-59, June 1996.
25. P. V. Goode M. Y. Chow, **“Using a Neural/Fuzzy System to Extract Heuristic Knowledge of Incipient Faults in Induction Motors,”** IEEE Transactions on Industrial Electronics, Vol. 42, No. 2, pp. 130-146, April 1995.
26. S. V. Bowers and K. R. Piety, **“Proactive Motor Monitoring Through Temperature, Shaft Current and Magnetic Flux Measurements”** CSI, 1996.

27. G. J. Anders, J. Endrenyi, G. L. Ford and G. C. Stone, "**A Probabilistic Model for Evaluating The Remaining Life of Electrical Insulation in Rotating Machines,**" IEEE Transactions on Energy Conversion, Vol. 5, No. 5, pp. 761-766, December 1990.
28. L. Langnau, "**Predicting Machine Failure,**" Power Transmission Design, pp. 43-45, July 1995.
29. J. R. Thalimer, J. J. McClelland and G. T. Homce, "**Motor Monitoring System for a Continuous Miner,**" United States Department of The Interior Bureau of Mines Information Circular, IC9306, 1992.
30. D. Kastha, "**Fault Mode Investigation and Fault Tolerant Control of Voltage-Fed Inverter Induction Motor Drive,**" Ph.D. Dissertation, The University of Tennessee, Knoxville, December 1993.
31. P. F. Albrecht, J. C. Appiarius, E. P. Cornell, D. W. Houghtaling, R. M. McCoy, E. L. Owen, "**Assessment of The Reliability of Motors in Utility Applications,**" IEEE Transactions on Energy Conversion, Vol. EC-2, No. 3, pp. 396-402, September 1987.
32. Motor Task Group, "**Work-In-Progress Report on Maintenance Good Practices For Motors in Nuclear Power Generating Stations,**" IEEE Transactions on Energy Conversion, Vol. 3, No. 3, pp. 589-600, September 1988.
33. "**Motor Testing Saves \$215,000, Detects Incipient Failures,**" Power Engineering, p. 46, June 1993.
34. S. T. Lakhavani, "**Reliability of Large Motors: Synchronous Have Best Record,**" Power Engineering, p. 42, October 1984.
35. C. A. Protopapas, S. D. Kaminaris, A. V. Machias and B. C. Papadias, "**An Expert System for Fault Repairing and Maintenance of Electric Motors,**" IEEE Transactions on Energy Conversion, Vol. 5, No. 1, pp. 79-83, March 1990.
36. R. C. Kryter and H. D. Haynes, "**How to Monitor Motor-Driven Machinery by Analyzing Motor Current,**" Power Engineering, pp. 35-39, October 1989.
37. S. F. Farag, R. G. Bartheld and T. G. Habetler, "**An Integrated On-Line Motor Protection System,**" IEEE Industry Applications Magazine, pp. 21-26, March / April 1996.
38. D. Berry, "**Creating a Comprehensive Motor Management Program,**" Maintenance Technology, pp. 11-15, May 1996.

39. S. V. Bowers, **"Integrated Strategy for Predictive Maintenance of AC Induction Motors"** CSI, 1996.
40. M. DelZingaro and C. Matthews, **"Using Power Signature Analysis to Detect the Behavior of Electric Motors and Motor-driven Machines,"** P/PM Technology, pp. 32-34, August 1995.
41. D. A. Casada and S. L. Bunch, **"The Use of The Motor as a Transducer to Monitor Pump Conditions,"** P/PM Technology, pp. 36-48, February 1996.
42. L. P. Gradin, W. B. Cartwright, N. M. Burstein, **"Test Method Improves Motor Bearing Wear Assessment at Calvert Cliffs,"** Power Engineering, pp. 32-33, June 1994.
43. G. T. Homce, **"Predicting the Failure of Electric Motors,"** United States Department of The Interior Bureau of Mines Information Circular, IC 9211, 1989.
44. G. T. Homce, **"Early Detection in Insulation Failure in Electric Motors,"** United States Department of The Interior Bureau of Mines Information Circular, IC9450, 1993.
45. D. V. Richardson, **"Handbook of Rotating Electric Machinery,"** Reston Publishing, Reston, Virginia, 1979.
46. J. E. Traister, **"Handbook of Electric Motors: Use and Repair,"** The Fairmont Press, Lilburn, Georgia, 1992.
47. M Clifford, **"Modern Electric Electronic Motors,"** Prentice Hall, Englewood Cliffs, New Jersey, 1990.
48. T. Litman, **"Efficient Electric Motor Systems Handbook,"** The Fairmont Press, Lilburn, Georgia, 1995.
49. F. T. Bartho, **"Industrial Electric Motors and Control Gear,"** MacDonald, London, 1965.
50. R. W. Smeaton, **"Motor Application and Maintenance Handbook,"** McGraw-Hill, 1969.
51. B. Raychaudhuri, **"Fault Monitoring and Residual Life Estimation of Electric Motors,"** MS Thesis, The University of Tennessee, Knoxville, 1995.
52. M. Subuthi, J. H. Taylor, R. Lofaro, A. C. Sugarman, M. W. Sheets and K. M. Skreiner, **"Improving Motor Reliability in Nuclear Power Plants, Volume 2: Functional Indicator Tests on a Small Electric Motor Subjected to Accelerated**

Aging,” U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Contract No. DE-AC02-76CH00016 FIN A-3270, November 1987.

53. M. Subuthi, J. H. Taylor, R. Lofaro, A. C. Sugarman, M. W. Sheets and K. M. Skreiner, “**Improving Motor Reliability in Nuclear Power Plants, Volume 1: Performance Evaluation and Maintenance Practices,**” U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Contract No. DE-AC02-76CH00016 FIN A-3270, November 1987.
54. M. Subuthi, J. H. Taylor, R. Lofaro, A. C. Sugarman, M. W. Sheets and K. M. Skreiner, “**Improving Motor Reliability in Nuclear Power Plants, Volume 1: Failure Analysis and Diagnostics Tests on a Naturally Aged Large Electric Motor,**” U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Contract No. DE-AC02-76CH00016 FIN A-3270, November 1987.
55. L. E. Atlas, G. D. Bernard and S. B. Narayanan, “**Applications of Time-Frequency Analysis to Signals from Manufacturing and Machine Monitoring Sensors,**” Proceedings of the IEEE, Vol. 84, No. 9, pp. 1319 - 1329, September 1996.
56. B. Samimy and G. Rizzoni, “**Mechanical Signature Analysis Using Time - Frequency Signal Processing: Application to Internal Combustion Engine Knock Detection,**” Proceedings of the IEEE, Vol. 84, No. 9, pp. 1330 - 1343, September 1996.
57. S. F. Legowski, A. H. M. S. Ula and A. M. Trzynadlowski, “**Instantaneous Power as a Medium for the Signature Analysis of Induction Motors,**” IEEE Transactions on Industry Applications, Vol. 32, No. 4, pp. 904 - 909, July / August 1996.
58. A. H. Bonnett and G. C. Soukup, “**Rotor Failures in Squirrel Cage Induction Motors,**” IEEE Transactions on Industry Applications, Vol. IA-22, No. 6, pp. 1165 - 1173, November / December 1986.
59. A. H. Bonnett and G. C. Soukup, “**Cause and Analysis of Stator and Rotor Failures in Three - Phase Squirrel - Cage Induction Motors,**” IEEE Transactions on Industry Applications, Vol. 28, No. 4, pp. 921 - 937, August 1992.
60. J. R. Nicholas, “**Motor Electrical Monitoring and Condition Analysis,**” Maintenance Technology, pp. 22 - 28, September 1996.
61. D. M. Brzezowski, “**Selecting Machinery for Vibration Monitoring,**” Maintenance Technology, pp. 29 -30, September 1996.
62. B. K. Bose, “**Adjustable Speed AC Drives,**” IEEE Press, 1996.

63. M. G. Na, "**Temperature Distribution of the Motor,**" Special Report Prepared for The University of Tennessee, Knoxville, August 1996.
64. M. G. Na, "**The Power Loss and Temperature Rise of the Motor,**" Special Report Prepared for The University of Tennessee, Knoxville.
65. PowerCAD Users's Manual, 1996.
66. P. Ferreira, A. Silber and B. R. Upadhyaya, "**Introduction to Preventive Maintenance Technology, Laboratory Demonstration of Data Acquisition and Analysis,**" The University of Tennessee, Knoxville, 1996.
67. G. Viot, "**Fuzzy Logic: Concepts to Constructs,**" AI Expert, pp. 26 - 33, November 1993.
68. C. C. Hung, "**Building a Neuro-Fuzzy Learning Control System,**" AI Expert, pp. 40 - 49, November 1993.
69. R. E. Thomas, "**When is a Life Test Truly Accelerated?**" Electronic Design pp. 64 - 71, January 6, 1964.
70. J. S. Lai, "**Simnon Simulation for Induction Motor Drives,**" IEEE 22nd SSST Conference, March 11-13, 1990.
71. P. Cooper and D. Roberts, "**Vibration Monitoring Justification Based on Saves,**" P/PM Technology, pp. 62 - 63, August 1996.
72. J. H. Steele, M. Archuleta, G. Brown and T. Seifert, "**SHARP - System Health Assessment and Real - Time Prediction,**" P/PM Technology, pp. 64 - 71, August 1996.
73. S. V. Bowers, "**Integrated Strategy for Predictive Maintenance of AC Induction Motors,**" P/PM Technology, pp. 34 - 40, October 1996.
74. M. S. Hussain, "**A Study of Amplitude Modulation in Mechanical Systems,**" P/PM Technology, pp. 42 - 46, October 1996.
75. J. R. Nicholas, "**Arriving a New Way to Analyze Resistance to Ground,**" P/PM Technology, pp. 56 - 58, October 1996.
76. D. Huntington, "**Successfully Implementing Expert System Technology,**" P/PM Technology, pp. 60 - 64, October 1996.
77. D. E. Lynn and K. Grabert, "**Soft Foot: A Fairy Tale?**" P/PM Technology, pp. 72 - 73, October 1996.

78. M. G. Na, "**MATLAB Simulation for Induction Motor Drives,**" Short internal report for The University of Tennessee, November 1996.
79. B. R. Upadhyaya, "**Progress Report: An Integrated On-Line Machinery Diagnosis and Prognosis System,**" MRC/BRU/96-01, The University of Tennessee, November 15, 1996.
80. Iris Power Engineering, "**Motor stator Monitoring Complements Bearing and Rotor Monitoring,**" Maintenance Technology, pp. 33 - 34, June 1997.
81. N. M. Burstein, "**ESA Spots Power Sag Transmitted By Electric Circuit,**" Maintenance Technology, pp. 35 – 37, June 1997.
82. J. C. Robinson, "**Monitoring Slow-Speed Power Transmission Components,**" Maintenance Technology, pp. 37 – 39, March 1997.
83. N. M. Burstein, "**ESA Reveals Low-Frequency Vibration,**" Maintenance Technology, pp. 41 – 42, March 1997.
84. E. R. Saller, "**Proper Selection and Installation of Industrial Accelerometers,**" P/PM Technology, pp. 22 – 24, December 1996.
85. N. M. Burstein, "**Case Histories of Rotating Equipment Condition Assessment by Electrical Signal Analysis,**" P/PM Technology, pp. 36 – 39, December 1996.
86. M. S. Hussain, "**Vibration Analysis of Slow Speed Machines,**" P/PM Technology, pp. 60 – 63, February 1997
87. Workshop Panel Discussion, "**AC and DC Motors,**" P/PM Technology, pp. 12 – 20, April 1997.
88. Workshop Panel Discussion, "**Resonance,**" P/PM Technology, pp. 22 – 27, April 1997.
89. S. Danehover and J. Mitchell, "**Real Time Acquisition and Storage of Vibration Information – Justification,**" P/PM Technology, pp. 42 – 45, June 1997.
90. C. F. Kane, B. R. Lease and A. Golubev. "**Practical Applications of Periodic Monitoring of Electrical Equipment for Partial Discharges,**" P/PM Technology, pp. 46 – 52, June 1997
91. N. S. V. Rao and E. M. Oblow, "**Majority and Location Based Fusers for Systems of PAC Concept Learners,**" IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 5, pp. 713 – 727, May 1994.

92. N. S. V. Rao and S. S. Iyengar, "**Distributed Decision Fusion Under Unknown Distributions,**" Optical Engineering, Vol. 35, No. 3, pp. 617 – 624, March 1996.
93. N. S. V. Rao, "**Computational Complexity Issues in Operative Diagnosis of Graph-Based Systems,**" IEEE Transactions on Computers, Vol. 42, No. 4, pp. 447 – 457, April 1993.
94. N. S. V. Rao, "**Fusion Methods for Multiple Sensor Systems with Unknown Error Densities,**" Journal of Franklin Institute, Vol. 331B, No. 5, pp. 509 – 530, 1995.
95. N. S. V. Rao, "**Multiple Sensor Fusion Under Unknown Distributions,**" Proceedings of Workshop on Foundations of Information / Decision Fusion with Applications to Engineering Problems, Washington, D.C., pp. 174 – 183, August 7 – 9, 1996.
96. R. N. Madan and N. S. V. Rao, "**Preface to Foundations of Information / Decision Fusion with Applications to engineering Problems,**" Proceedings of Workshop on Foundations of Information / Decision Fusion with Applications to Engineering Problems, Washington, D.C., pp. vi - x, August 7 – 9, 1996.
97. N. S. V. Rao, "**Distributed Decision Fusion Using empirical Estimation,**" IEEE Transactions on Aerospace and Electronic systems, Vol. 33, No. 4, October 1997.
98. N. S. V. Rao, "**Nadaraya-Watson Estimator for Sensor Fusion,**" Optical Engineering, Vol. 36, No. 3, pp. 642 – 647, March 1997.
99. IEEE Standards Board, "**An American National Standard IEEE Standard Test Procedure of Insulation Materials for Random-Wound AC Electric Motors,**" ANSI C50.32-1976, IEEE Std 117-1974, November 1996.
100. "**Guides to Lubricate Motors,**" U.S. Electrical Motors, March 1997.
101. "**Proposal of the Fifth Edition of The Standard for Systems of Insulation Materials – General, UL 1446,**" Underwriters Laboratories Inc., January 1997.
102. G. F. Lang, "**3-Phase AC Motor Primer,**" P/PM Technology, pp. 54 – 57, August 1997.
103. J. Traner and G. White, "**Combining Spectra, Cepstra and Demodulated Spectra in an Expert System,**" P/PM Technology, pp. 68 – 74, August 1997.
104. M. G. Na, "**Temperature Distribution of a Motor,**" Short internal report for The University of Tennessee, June 23, 1997.

105. Y. Zhou, H. Leung and P. C. Yip, “**An Exact Maximum Likelihood Registration Algorithm for Data Fusion,**” IEEE Transactions on Signal Processing, Vol. 45, No. 6, pp. 1560 – 1573, June 1997.
106. Entek IRD International Corp., “**Diagnose Electrical Faults With Spike Energy,**” Maintenance Technology, pp. 39 – 41, January 1997.
107. R. Natarajan, “**Failure Identification of Induction Motors By Sensing Unbalanced Stator Currents,**” IEEE Transactions on energy Conversion, Vol. 4, No. 4, pp. 585 – 590, December 1989.
108. G. B. Kliman and R. A. Koegl, “**Noninvasive detection of Broken Rotor Bars in Operating Induction Motors,**” IEEE Transactions on Energy Conversion, Vol. 3, No. 4, pp. 873 – 879, December 1988.
109. G. G. Richards, “**Reduced Order Model for single and Double Cage Induction Motors During Startup,**” IEEE Transactions on Energy Conversion, Vol. 3, No. 2, pp. 335 – 341, June 1988.
110. A. M. A. Mahmoud and R. W. Menzies, “**A Complete Time Domain Model of The Induction Motor For Efficiency Evaluation,**” IEEE Transactions on Energy Conversion, Vol. EC-1, No. 1, pp. 68 – 76, March 1986.
111. O. Wasynczuk, Y. M. Diao and P. C. Krause, “**Theory and Comparison of Reduced Order Models of Induction Machines,**” IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 3, pp. 598 – 605, March 1985.
112. J. D. Lavers and R. W. Y. Cheung, “**A Software Package For The Steady State and Dynamic Simulation of The Induction Motor,**” IEEE Transactions on Power Systems, Vol. PWRS-1, No. 2, pp. 167 – 173, May 1986.
113. S. Williamson and K. Mirzoian, “**Analysis of Cage Induction Motors with Stator Winding Faults,**” IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 7, July 1985.
114. S. N. Ghani, “**Digital Computer Simulation of Three-Phase Induction Machine Dynamics – A Generalized Approach,**” IEEE Transactions on Industry Applications, Vol. 24, No. 1, pp. 106 – 114, January / February 1988.
115. B. R. Upadhyaya, “**Analysis of Nonstationary Signals**”, class notes, 1997.
116. S. F. Farag, R. G. Bartheld and T. G. Habetler, “**An Integrated On-Line Motor Prediction System,**” IEEE Industry Applications Magazine, pp. 21 – 26, March / April 1996.

117. R. R. Schoen, B. K. Lin, T. G. Habetler, J. H. Schlag and S. Farag, “**An Unsupervised, On-Line System For Induction Motor Fault Detection Using Stator Current Monitoring,**” 1994 IEEE Industrial Applications Meeting, Vol. 1, pp. 103-109, 1994.
118. R. Schoen, T. G. Habetler, F. Kamran and R. G. Bartheld, “**Motor Bearing Damage Detection Using Stator Current Monitoring,**” 1994 IEEE Industrial Applications Meeting, Vol. 1, pp. 110-116, 1994.
119. R. R. Schoen and T. G. Habetler, “**Effects of Time Varying Loads on Rotor Fault Detection in Induction Machines,**” IEEE Transactions on Industry Applications, Vol 31, No. 4, pp. 900 –906, July / August 1995.
120. J. D. Irwin, “**Multisensor Fusion and Integration for Intelligent Systems,**” The Industrial Electronics Handbook, pp. 1592 – 1659, CRC Press & IEEE Press, 1997.
121. B. Samimy and G. Rizzoni, “**Mechanical Signature Analysis Using Time-Frequency Signal Processing: Application to Internal Combustion Engine Knock Detection,**” Proceedings of The IEEE, Vol. 84, No. 9, pp. 1330 – 1343, September 1996.
122. D. L. Hall and J. Llinas, “**An Introduction to Multisensor Data Fusion,**” Proceedings of The IEEE, Vol. 85, No. 1, pp. 6 – 23, January 1997.
123. B. V. Dasarathy, “**Sensor Fusion Potential Exploitation – Innovative Architectures and Illustrative Applications,**” Proceedings of The IEEE, Vol. 85, No. 1, pp. 24 – 38, January 1997.
124. R. T. Antony, “**Database Support to Data Fusion Automation,**” Proceedings of The IEEE, Vol. 85, No. 1, pp. 39 - 53, January 1997.
125. R. Viswanathan and P. K. Varshney, “**Distributed Detection With Multiple Sensors: Part I – Fundamentals,**” Proceedings of The IEEE, Vol. 85, No. 1, pp. 54 - 63, January 1997.
126. R. S. Blum, S. A. Kassam and H. V. Poor, “**Distributed Detection With Multiple Sensors: Part II – Advanced Topics,**” Proceedings of The IEEE, Vol. 85, No. 1, pp. 64 - 79, January 1997.
127. J. S. Hsu, “**Monitoring of Defects in Induction Motors Through Air – Gap Torque Observation,**” IEEE Transactions on Industry Applications, Vol. 31, No. 5, pp. 1016 – 1021, September / October 1995.

128. T. W. S. Chow and G. Fei, **“Three Phase Induction Machines Asymmetrical Faults Identification Using Bispectrum,”** IEEE Transactions on Energy Conversion, Vol. 10, No. 4, pp. 688 – 693, December 1995.
129. F. Bllabjerg, J. K. Pedersen, E. Ritchie and P. Nielsen, **“Determination of Mechanical Resonances in Induction Motors by Random Modulation and Acoustic Measurements,”** IEEE Transactions on Industry Applications, Vol. 31, No. 4, pp. 823 – 829, July / August 1995.
130. G. T. Homce and J. R. Thalimer, **“Reducing Unscheduled Plant Maintenance Delays – Field Test of a New Method to Predict Electric Motor Failure,”** IEEE Transactions on Industry Applications, Vol. 32, No. 3, pp. 689 – 694, May / June 1996.
131. B. C. Lesiuetre, P. W. Sauer and M. A. Pai, **“Development and Comperative Study of Induction Machine Based Dynamic P, Q Load Models,”** IEEE Transactions on Power Systems, Vol. 10, No. 1, pp. 182 – 191, February 1995
132. P. Ju, E. Handschin, Z. N. Wei and U. Schlücking, **“Sequential Parameter Estimation of A Simplified Induction Motor Load Model,”** IEEE Transactions on Power Systems, Vol. 11, No. 1, pp. 319 – 324, February 1996.
133. R. Yacamini and S. C. Chang, **“Noise and Vibration from Induction Machines Fed from Harmonic Sources,”** IEEE Transactions on Energy Conversion, Vol. 10, No. 2, June 1995.
134. H. Chai and P. P. Acarnley, **“Induction Motor Parameter Estimation Algorithm Using Spectral Analysis,”** IEE Proceedings-B, Vol. 139, No. 3, pp. 165 – 174, May 1992.
135. J. W. Griffith, R. M. McCoy and D. K. Sharma, **“Induction Motor Squirrel Cage Rotor Winding Thermal Analysis,”** IEEE Transactions on Energy Conversion, Vol. EC-1, No. 3, pp. 22 – 25, September 1986.
136. J. Reason, **“Pinpoint Induction – Motor Faults by Analyzing Load Current,”** Power, pp. 87 – 88, October 1987.
137. J. R. Cameron, W. T. Thomson and A. B. Dow, **“Vibration and Current Monitoring for Detecting Airgap Eccentricity in Large Induction Motors,”** IEE Proceedings, Vol. 133, Pt. B, No. 3, pp. 155 – 163. May 1986.
138. S. Ansuji, F. Shokooh and R. Schinzingler, **“Parameter Estimation for Induction Machines Based on Sensitivity Analysis,”** IEEE Transactions on Industry Applications, Vol. 25, No. 6, pp. 1035 – 1040. November / December 1989.

139. R. B. Smith, "**The Inverse Gamma Model in Field Oriented Control,**" MS Thesis, The University of Tennessee, Knoxville, May 1992.
140. Y. H. Wang, "**A Dynamic Model for an Induction Motor and its Verification,**" MS Thesis, The University of Tennessee, Knoxville, June 1982.
141. R. E. Steven, "**Electrical Machines and Power Electronics,**" Van Nostrand Reinhold, 1984.
142. S. A. Nasar, "**Handbook of Electric Machines,**" McGraw-Hill, 1987.
143. M. A. Abidi and R. C. Gonzalez, "**Data Fusion in Robotics and Machine Intelligence,**" Academic Press, 1992.
144. D. Stegermann, W. Reimche, U. Stüdmersen and O. Pietsch, "**Failure Root Cause Analysis in Turbomachinery by advanced Vibration Analysis,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 7.01 – 7.14, May 20 – 22, 1997.
145. A. D. Silva and B. R. Upadhyaya, "**Rotating Machinery Monitoring and Diagnosis Using Short – Time Fourier Transform and Wavelet Techniques,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 14.01 – 14.15, May 20 – 22, 1997.
146. W. Wang, M. Harrap and J. Mathew, "**Detecting Bearing Defects Under Severe Signal to Noise Ratio Conditions,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 18.01 – 18.12, May 20 – 22, 1997.
147. A. H. Bonnett and G. C. Soukup, "**The Cause and Analysis of Stator and Rotor Failures in AC Induction Machines,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 29.01 – 29.20, May 20 – 22, 1997.
148. G. F. Lang and P. J. Sock, "**Measuring The Efficiency and Output of AC Induction Motors,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 32.01 – 32.07, May 20 – 22, 1997,
149. M. E. H. Benbouzid and R. Beguenane, "**Induction Motor diagnostics Via Stator Current Monitoring,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 36.01 – 36.10, May 20 – 22, 1997.
150. J. C. Jung and P. H. Seong, "**Development of An Induction Motor Abnormality Monitoring System (IMAMS) Using Power Line Signal Analysis,**" Maintenance And Reliability CONFERENCE Proceedings, Knoxville, Tennessee, pp. 39.01 – 39.15, May 20 – 22, 1997.

151. A. Lucifredi and M. Rossi, “**Advanced Signal Processing Techniques for Plant Monitoring, Diagnostics and Maintenance,**” Maintenance And Reliability CONference Proceedings, Knoxville, Tennessee, pp. 66.01 – 66.07, May 20 – 22, 1997.
152. J. R. Nicholas, “**Correlation of Motor Circuit Analysis Data With Motor Temperature and Infrared Thermography,**” Maintenance And Reliability CONference Proceedings, Knoxville, Tennessee, pp. 76.01 – 76.11, May 20 – 22, 1997.
153. E. D. Blakeman and R. C. Kryter, “**Noninvasive Testing of Solenoid-Operated Valves Using Transient Current Signature Analysis,**” Maintenance And Reliability CONference Proceedings, Knoxville, Tennessee, pp. 84.01 – 84.14, May 20 – 22, 1997.
154. Y. Maki and K. A. Lopara, “**A Neural Network Approach to Fault Detection and Detection and Diagnosis in Industrial Processes,**” IEEE Transactions on Control Systems Technology, Vol. 5, No. 6, pp. 529 – 541, November 1997.
155. D. L. Center, “**Some Instrumentation Considerations in Rolling Element Bearing Condition Analysis,**” 17th Annual Meeting of The Vibration Institute, June 1993.
156. A. V. Barkov, N. A. Barkova and J. S. Mitchell, “**Assessing the Condition and Lifetime of Rolling Element Bearings from a Single Measurement,**” 19th Annual Meeting of The Vibration Institute, 1995.
157. J. Feeley, “**Customer Focus Leads to Advanced Motor Condition Monitoring,**” Control, pp. 58 – 63, January 1998.
158. G. C. Stone, H. G. Sedding, B. A. Lloyd and B. K. Gupta, “**The Ability of Diagnostic Tests to Estimate The Remaining Life of Stator Insulation,**” IEEE Transactions on Energy Conversion, Vol. 3, No. 4, pp. 833 – 841, December 1988.
159. A. M. Bruning, D. G. Kasture and H. E. Ascher, “**Absolute vs. Comparative End-of-Life Age,**” IEEE Transactions on Dielectrics and Electrical Insulation, Vol. 3, No. 4, pp. 567 – 576, August 1996.
160. W. R. Finley and R. R. Burke, “**Proper Specification and Installation of Induction Motors,**” IEEE Industry Applications Magazine, pp. 56 – 69, January / February 1997.
161. P. Langhorst and C. Hancock, “**Dealing with Motor Shaft Voltages and Bearing Currents,**” Maintenance Technology, pp. 24 – 27, November 1997.

162. D. Busse, J. Erdman, R. Kerkman, D. Schlegel and G. Skibinski, “**Characteristics of Shaft Voltage and Bearing Currents,**” IEEE Industry Applications Magazine, pp. 21 – 32, November / December 1997.
163. D. Kowal, “**Bearing Damage Resulting from Shaft Voltages and Currents,**” CSI, 1997.

APPENDICES

APPENDIX A

SCHEDULE OF MOTOR AGING AND TESTING ACTIVITIES

Date (1997)	Events (Comments)
02 / 28	<p>File 0228.zip contains motor # 1 test results as:0% start-up</p> <ul style="list-style-type: none"> • 0% start-up • 115% load • 100% load • 75% load • 50% load • 25% load • 0% load • 0% start-up without dynamometer • 0% without dynamometer
03 / 01	<p>Started insulation aging motor #1 @ 13:25</p> <p>File 0301.zip contains motor #2 test results.</p>
03 / 02	File 0302.zip contains motor #3 test results.
03 / 03	<p>Stopped insulation aging motor #1 @ 14:00</p> <p>Motor #2 has short on TC PE 4:00, caused damage to SCXI-1100</p>
03 / 04	Ordered a new SCXI - 1100 module
03 / 05	Ordered a new SCXI - 1120 module, so all of the TC connections from the motor will pass through an isolation amplifier (Protected up to 250Vrms).
03 / 06	Installed ¼ amperes fuses to TC connections for extra protection.
03 / 07	<p>Started insulation aging motor #2 @ 12:00</p> <p>File 0307.zip contains motor #6 test results with the following exceptions: TC PE 12:00 @ 0% start-up is incorrect (wrong TC polarity) TC SE 10:00 @ 25% is incorrect for same reason.</p> <p>Also following load pattern is used:</p> <ul style="list-style-type: none"> • 0% start-up without dynamometer • 0% without dynamometer • 0% start up • 25% load • 50% load • 75% load • 100% load • 115% load <p>Steady state condition is changed from $\Delta T \leq 2^{\circ}\text{C}$ in 30 minutes to $\Delta T \leq 0.2^{\circ}\text{C}$ in 3 minutes. Also each load pattern is pre-heated by applying 115% load in between. This gives steady state values for winding insulation and bearing temperatures, the most important heat sources.</p>
03 / 08	<p>Started bearing aging motor #6 @ 12:12 in oven #1</p> <p>File 0308.zip contains motor #7 test results.</p>

Date (1997)	Events (Comments)
03 / 09	<p>Started bearing aging motor #7 @ 12:55 in oven #2</p> <p>Stopped insulation aging motor #2 @ 12:30</p> <p>Started insulation aging motor #3 @ 13:25</p> <p>File 0308.zip contains motor #8 test results.</p>
03 / 10	<p>Started bearing aging motor #8 @ 11:20 in oven #3</p> <p>Changed SCXI-1304 coupling from DC to AC, so that DC component of vibration signals are removed with cutoff frequency of 0.16 Hz.</p> <p>File 0310.zip contains motor #1 test results.</p>
03 / 11	<p>Stopped bearing aging motor #6 @ 12:12, cooled it for 6 hours and quenched it for 15 minutes and started aging back @ 18:21</p> <p>Stopped insulation aging motor #3 @ 13:55</p> <p>Started insulation aging motor #1 @ 14:34</p> <p>File 0311.zip contains motor #2 test results with following exceptions: TC PE 8:00 = TC Ambient TC PE 4:00 = TC PE 8:00</p>
03 / 12	<p>Stopped bearing aging motor #7 @ 12:55, cooled it, quenched it for 15 minutes and started aging back @ 18:45</p> <p>File 0312.zip contains motor #3 test results.</p>
03 / 13	<p>Stopped bearing aging motor #8 @ 11:20, cooled it, quenched it for 15 minutes and started aging back @ 17:30</p> <p>Stopped insulation aging motor #1 @ 15:20</p> <p>Started insulation aging motor #2 @ 16:16</p> <p>File 0313.zip contains motor #11 test results.</p>
03 / 14	<p>Fluted motor #11 for 30 min and put it in oven #4 @ 15:00</p> <p>Stopped bearing aging motor #6 @ 12:21</p> <p>File 0314.zip contains motor #12 test results with the following exception: TC Ambient not working, replaced it with another J-type TC, but not same size.</p>
03 / 15	<p>Stopped insulation aging motor #2 @ 16:50</p> <p>Started insulation aging motor #3 @ 18:35</p> <p>Fluted motor #12 for 30 minutes and put it into oven #5 @ 19:03</p> <p>Stopped bearing aging motor #7 @ 18:45</p> <p>Installed CSI Motor Status device and software.</p> <p>File 0315.zip contains motor #13 test results.</p>

Date (1997)	Events (Comments)
03 / 16	<p>Corrected DAQ program Short End Axial vibration Gain from 20 to 5.</p> <p>Stopped bearing aging motor #8 @ 17:30</p> <p>Fluted motor #13 for 30 minutes and put it in oven #1 @ 19:45</p> <p>File 0316.zip contains motor #6 test results.</p>
03 / 17	<p>Started bearing aging motor #6 in oven #6 @ 14:07</p> <p>Stopped aging motor #11 @ 15:00, cooled it, quenched it for 15 minutes and put it back @ 21:10</p> <p>Stopped insulation aging motor #3 @ 19:55</p> <p>File 0317 contains motor #7 test results with the following exceptions: Cover & SE H vibrations were misplaced for 0% dynamometer uncoupled tests. PE 10:00 accelerometer is magnetic mount, instead of stud mount.</p>
03 / 18	<p>Started bearing aging motor #7 in oven #2 @ 16:15</p> <p>Stopped aging motor #12 @ 19:03, cooled it and quenched it for 15 minutes.</p> <p>File 0318.zip contains motor #8 test results.</p>
03 / 19	<p>Had breakfast @ 9:00 with UT MRC members & UTNE department head.</p> <p>Changed CSI Motor Status parameters as: Short: 240 Long: 1440</p> <p>Put motor #12 into oven #3 @ 11:40 for second half of aging cycle.</p> <p>Started bearing aging motor #8 in oven #5 @ 11:55.</p> <p>Stopped aging motor #13 @ 19:45, cooled it and quenched it for 15 minutes.</p>
03 / 20	<p>Put motor # 13 into oven # 1 @ 11:25 for second half of aging cycle.</p> <p>Stopped bearing aging motor #6 @ 14:07, cooled it and quenched it</p>
03 / 21	<p>Stopped aging motor #11 @ 12:10 (15 hours late!)</p> <p>Put motor #6 in oven #4 @ 12:40 for second half of aging cycle.</p> <p>Stopped bearing aging motor #7 @ 16:15, cooled it and quenched it for 15 minutes.</p>
03 / 22	<p>Put motor #7 in oven #2 @ 14:11 for second half of aging cycle.</p> <p>Stopped bearing aging motor #8 @ 11:55, cooled it and quenched it for 15 minutes.</p> <p>File 0322.zip contains motor #11 test results.</p>

Date (1997)	Events (Comments)
03 / 23	Put motor #8 in oven # 5 @ 18:10 for second half of aging cycle. Stopped aging motor #13 @ 11:25 Fluted motor #11 for 30 minutes and put it into oven #3 @ 21:35 File 0323 contains motor #12 test results with the following exception: PE 10:00 vibration sensor is magnetic mount.
03 / 24	Stopped bearing aging motor #6 @ 12:40 Fluted motor #12 for 30 minutes and put it into oven #6 @ 17:15
03 / 25	Installed & calibrated torque sensor. Stopped bearing aging motor #7 @ 14:11 File 0325.zip contains motor #13 test results. Fluted motor #13 for 30 minutes an put it into oven #2 @ 20:10
03 / 26	File 0326.zip contains motor #1 test results. Started insulation aging motor #1 @ 17:15 Stopped bearing aging motor #8 @ 18:10 Stopped aging motor #11 @ 21:35, cooled it and quenched it for 15 minutes.
03 / 27	Put motor #11 in oven #3 @ 17:25 for second half of aging cycle. File 0327.zip contains motor #6 test results. Started bearing aging motor #6 in oven #5 @ 16:30 Stopped aging motor #12 @ 17:15, cooled it and quenched it.
03 / 28	Stopped insulation aging motor #1 @ 17:45: Terminal on wires 7-6-1 was melted down. Stopped aging motor #13 @ 20:10, cooled it and quenched it for 15 minutes. Put motor #12 into oven #1 for second half aging cycle.
03 / 29	Put motor # 13 in oven #2 @ 16:55 for second-half aging cycle. File 0329.zip contains motor #7 test results. Started bearing aging motor #7 in oven #4 @ 18:50

Date (1997)	Events (Comments)
03 / 30	Stopped aging motor #11 @ 17:25, cooled it and quenched it for 15 minutes. Stopped bearing aging motor #6 @ 16:30, cooled it and quenched it for 15 minutes.
03 / 31	Put motor #11 into oven #3 @ 10:50 for second half of aging cycle. Put motor #6 into oven #5 @ 11:30 for second half of aging cycle. Stopped aging motor #12 @ 20:40
04 / 01	Stopped aging motor # 13 @ 16:55 Stopped bearing aging motor #7 @ 18:50. cooled it and quenched it for 15 minutes.
04 / 02	Wrote program to convert MRC data from binary to ASCII with desired measurements. Wrote program to generate mean and rms values of measured data and to create a table.
04 / 03	File 0403.zip contains motor #2 test results with the following exception: TC PE 4:00 = TC PE 8:00 Stopped aging motor #11 @ 10:50 Stopped bearing aging motor #6 @ 11:30
04 / 05	File 0405.zip contains motor #3 test results with the following exception: PE 10:00 accelerometer is magnetic mount. Started insulation aging motor #2 @ 17:30
04 / 06	File 0406.zip contains motor #1 test results with the following exception: Ignore 0% Torque, since it was not calibrated correctly). Started bearing aging 2 nd half for motor #7 in oven #5 @ 15:40 (Had forgotten the 2 nd part before)
04 / 07	File 0407.zip contains motor #8 test results. Stopped insulation aging motor #2 @ 19:05. Started insulation aging motor #3 @ 23:30 Started bearing aging motor #8 in oven #4 @ 23:45
04 / 08	File 0408.zip contains motor #11 test results.
04 / 09	Stopped bearing aging motor #7 @ 15:40 Stopped insulation aging motor #3 @ 24:03
04 / 10	Took out motor #8 of oven #4 @ 23:45 & quenched it. Fluted motor #11 and put it in oven #1 @ 23:00
04 / 12	File 0412.zip contains motor #12 test data with the following excetion: PE 10:00 is magnetic.

Date (1997)	Events (Comments)
04 / 13	<p>File 0413.zip contains motor #13 test data.</p> <p>Fluted motor # 12 and put it in oven #2 @ 12:45</p> <p>Put motor #8 in oven #3 @ 13:30</p> <p>Put motor #13 in oven #4 @ 21:00</p>
04 / 14	Took out motor #11 from oven #1 @ 22:00 and quenched it.
04 / 15	Put motor #11 in oven #1 @ 23:45
04 / 16	<p>Took out motor #12 @ 12:45 and quenched it.</p> <p>Stopped bearing aging of motor #8. @ 13:30</p> <p>Took out motor #13 from oven #4 @ 21:00 and quenched it.</p>
04 / 17	<p>Put motor #12 in oven #12 @ 10:10</p> <p>Put motor #13 in oven #3 @ 13:35</p>
04 / 18	Stopped aging of motor # 11 @ 23:45
04 / 19	<p>Took out motor #12 out of oven #2 @ 10:10</p> <p>Took out motor #13 from oven #3 @ 13:35</p>
05 / 10	<p>File 0510.zip contains motor #2 test data with the following exceptions: Accelerometer PE 2:00 is magnetic, TC PE 8:00 = PE 4:00, Dynamometer speed measurement is missing. New DAQ settings are being used as:</p> <ol style="list-style-type: none"> 1. A new TC measurement is added in the .tc files to measure ambient temperature (1" away from motor air intake) 2. Sampling rate of low frequency has been changed to 4000 samples per seconds for each channel <ol style="list-style-type: none"> a. Accelerometer filters (hardware) are set to 200 Hz b. All other data is filtered by a 5th order low-pass Butterworth filter c. Low pass data is recorded at 666.67 Hz <p>Started insulation aging motor #1 @ 17:25.</p>
05 / 11	<p>File 0511.zip contains motor #3 test data.</p> <p>CSI measurement is missing because forgot to download the data.</p>
05 / 12	<p>File 0512.zip contains motor #6 test data.</p> <p>Stopped insulation aging motor #1 @ 17:55</p> <p>Put motor #6 in oven #1 @ 18:00</p> <p>Started insulation aging motor #3 @18:18</p>

Date (1997)	Events (Comments)
05 / 13	<p>File 0513.zip contains motor #7 test data.</p> <p>Put motor #7 in oven #2 @ 18:40</p>
05 / 14	<p>File 0514.zip contains motor #8 test data.</p> <p>Put motor #8 in oven #3 @ 15:40</p> <p>Stopped insulation aging motor #3 @ 18:48</p> <p>Started insulation aging motor #2 @ 19:45</p>
05 / 15	<p>File 0515.zip contains motor #11 test data.</p> <p>Fluted motor #11 and put it into oven #4 @ 13:45</p> <p>File 0515b.zip contains motor #12 test data.</p> <p>Fluted motor #12 and put into oven #5 @ 17:20</p> <p>File 0515c.zip contains motor #1 test data with the following exception: TC AIR not working.</p>
05 / 16	<p>File 0516.zip contains motor #13 test data.</p> <p>Fluted motor #13 and put into oven #6 @ 13:30</p> <p>Took out motor #6 from oven #1 @ 18:10 and quenched it (24 hours late)</p> <p>Took out motor #7 from oven #2 @ 18:40 and quenched it</p> <p>Stopped insulation aging motor #2 @ 20:15</p> <p>File 0516b.zip contains motor #3 test data with the following exceptions: 0% coupled start-up may be bad. RPM tape came off at some time. Check files before the ones listed below: 19970516140651.mt. Delete 3rd and 4th .tc files. no rpm data.</p>
05 / 17	<p>File 0517.zip contains motor #2 test data with the following exceptions: Wax melted on PE 10:00 and PE 2:00 accelerometers. so that they were loosely attached.</p> <p>Took out motor #8 @ 18:40 and quenched it.</p> <p>Started insulation aging motor #1 @ 15:30</p> <p>Put motor #6 in oven #1 @ 16:00</p> <p>Put motor #7 in oven #2 @ 16:20</p>

Date (1997)	Events (Comments)
05 / 18	Took out motor #11 from oven #4 @ 13:45 and quenched it Took out motor #12 from oven #5 @ 17:20 and quenched it
05 / 19	Insulation aging motor #1 was stopped because of melt-down of terminals @ 13:30 Put motor #8 in oven #3 @ 14:05 Put motor #12 in oven #4 @ 15:00 Put motor #11 in oven #5 @ 15:35 Took out motor #13 from oven #13 from oven #6 @ 15:45 and quenched it.
05 / 20	Took out motor #6 from oven #1 @ 16:00 Took out motor #7 from oven #2 @ 16:20 Put motor #13 in oven #6 @ 13:30
05 / 22	File 0522.zip contains motor #6 test data with the following exception: It also has stop data. Started insulation aging motor #2 @ 12:35 Took out motor #8 from oven #3 @ 14:05 Took out motor #12 from oven #4 @ 15:00 Took out motor #11 from oven #5 @ 15:35
05 / 23	Took out motor #13 from oven #6 @ 13:30 Put motor #6 in oven #1 @ 13:10
05 / 24	Stopped insulation aging motor #2 @ 17:05
05 / 25	File 0525.zip contains motor #7 test data Put motor #7 in oven #2 @ 19:20
05 / 26	Took out motor #6 from oven #1 @ 13:10 and quenched it Started insulation aging motor #3 @ 23:45
05 / 28	File 0528.zip contains motor #8 test data. Put motor #6 in oven #1 @ 13:00 Put motor #8 in oven #3 @ 17:15 Took out motor #7 @ 19:20 and quenched it
05 / 29	File 0529.zip contains motor #11 test data. Stopped insulation aging motor #3 @ 00:16 Put motor #7 in oven #2 @ 16:50

Date (1997)	Events (Comments)
05 / 30	File 0530.zip contains motor #12 test data with the following exceptions: Stop data for 0% uncoupled is missing. PE 10:00 is magnetic Fluted motor #11 and put it in oven #4 @ 01:15 Fluted motor #12 and put it in oven #5 @ 19:25
05 / 31	File 0531.zip contains motor #13 test data with the following exceptions: 50% load may have been recorded twice, use the first one Fluted motor #13 and put it in oven #6 @ 16:45 Took out motor #8 from oven #3 @ 17:50 and quenched it. Took out motor #6 from oven #1 @ 13:00
06 / 02	Took out motor #11 from oven #4 @ 01:15 and quenched it Took out motor #7 from oven #2 @ 22:30 (18 hours late) Took out motor #12 from oven #5 @ 19:25 and quenched it
06 / 03	Put motor #8 in oven #1 @ 14:15 Put motor #11 in oven #2 @ 15:03 Put motor #12 in oven #3 @ 15:30 Took out motor #13 from oven #6 and quenched it.
06 / 06	Took out motor #8 from oven #1 @ 14:15 Took out motor #11 from oven #2 @ 15:03 Took out motor #12 from oven #3 @ 15:50
06 / 09	Put motor #13 in oven #6 @ 14:35
06 / 11	File 0611.zip contains motor #6 test data.
06 / 12	File 0612.zip contains motor #1 test data with the following exceptions: No TC AIR data. CSI device not working. Started insulation aging testing of motor #1 @ 14:40 File 0612b.zip contains motor #7 test data with the following exception: Accelerometer PE 10:00 is magnetic Put motor #7 in oven #2 @ 18:05 Put motor #6 in oven #1 @ 01:30 and took it out @ 14:35

Date (1997)	Events (Comments)
06 / 13	Put motor #8 in oven #3 @ 12:35 File 0613.zip contains motor #8 test data. Put motor #8 in oven #3 @ 12:35 Quenched motor #6 & put it in oven #1 @ 09:27
06 / 14	Stopped insulation aging motor #1 @ 13:15
06 / 16	Took out motor #6 from oven #1 @ 16:21 Took out motor #7 from oven #2 @ 16:23 (1 day late) Took out motor #8 from oven #8 @ 16:25 Took out motor #13 from oven #6 @ 16:23 (3 days late)
06 / 17	Motor #11 test failed, possibly because of motor failure.
06 / 19	Tested Motor #12 into file 0619.zip with the following exceptions : PE 10:00 is magnetic 0% shutdown is actually not shutdown. Started using CSI device just replaced battery.
06 / 20	Fluted motor #12 Put Motor #12 in oven #4 @ 00:15 Took out motor #6, 7, 8.
06 / 21	Tested motor #3 in file 0621.zip with the following exceptions 0% dynamometer startup coupled temperature unreliable! Started insulation aging motor #3 @ 16:24
06 / 23	Took out motor #12 @ 10:00 ,quenched it and put it back in oven #5 @ 17:09 Tested motor #11 in file 0623.zip Fluted Motor #11 and put it in oven #4 @ 17:05 Stopped motor #3 insulation aging.
06 / 24	Tested motor #13 in file 0624.zip with the following exception: X phase had an imbalance, possibly from loose terminal connection Fluted motor #13 and put it in oven #6 @ 18:50
06 / 25	Tested motor #2 into file 0625.zip with the following exception: PE 10:00 is magnetic ! TC PE 4:00 = PE 8:00 Started insulation aging motor #2 @ 13:00 Forgot the motor & it went to 250 °C. Within 15 minutes it burned out!
06 / 26	Took out motor # 11 from oven # 4 and quenched it Took out motor # 12 from oven #5 and quenched it

Date (1997)	Events (Comments)
06 / 27	Took out motor # 13 from oven #6 @ 18:50 Stopped insulation aging motor # 2 @ 13:30
06 / 28	Tested motor #6 in file 0628.zip Started bearing aging motor #6 in oven #1 @ 12:35
06 / 29	Tested motor #7 in file 0629.zip Started bearing aging motor #7 in oven #2 @ 13:15
07 / 02	Took out motor #6 out of oven @ 9:35 Took out motor #7 out of oven @ 21:30
07 / 03	Quenched and put motor #6 in oven #1 @ 08:05 Quenched and put motor #7 in oven #2 @ 08:45
07 / 06	Took out motor #6 from oven #1 @ 13:36 Took out motor #7 from oven #2 @ 13:39
07 / 11	Tested motor #8 in file 0711.zip Put motor #8 in oven #1 @ 12:30 Tested motor # 12 in file 0711b.zip with the following exceptions: PE 10:00 is magnetic!
07 / 13	Fluted & put motor #12 in oven #2 @ 21:00 Tested motor #1 in file 0713.zip with the following exceptions: TC Air not working! No CSI device used.
07 / 14	Took out motor #8 and quenched it and put it back in oven @ 23:50 Started insulation aging motor #1 @ 13:27
07 / 16	Stopped insulation aging motor #1 @ 13:57
07 / 17	Took out motor # 12 from oven #12 @ 13:00, quenched it and put it in oven #2 @ 22:30 Tested motor # 11 in file 0717.zip Fluted motor #11 and put it in oven #3 @ 23:00 Took out motor #8 from oven #1 @ 23:50
07 / 18	Tested motor #13 in file 0718.zip Fluted motor #13 and put it in oven #1 @ 15:25 Put new fans on motor # 1, 2, 3.
07 / 20	Took out motor #12 from oven # 2 @ 9:30
07 / 22	Took out motor #11 from oven #7 @ 20:10 (2 days late)
07 / 23	Took out motor #13 from oven #1 @ 17:15 (2 days late), quenched it and put it back @ 24:45

Date (1997)	Events (Comments)
07 / 24	<p>Tested motor #12 in file 0724.zip with the following exceptions: PE 10:00 is magnetic</p> <p>Fluted motor #12 and put it into oven #2 @ 22:40</p> <p>Tried to test motor #2, currents went up to $I_x = 22$ Amps $I_y = 54$ Amps $I_z = 30$ Amps And then tripped the circuit. When tried to run the second time, flames came out of the cover box.</p>
07 / 25	Took out motor #13 from oven #13 @ 9:00
07 / 26	<p>Tested motor #11 into file 0726.zip with the following exceptions: Motor fan melted down at possibly 100% load, even though it was newly replaced. 0% shutdown without the dynamometer and 115% shutdown data are missing</p>
07 / 27	<p>Took out motor #12 from oven #2 @ 22:40 and quenched it.</p> <p>Started insulation aging motor #1 @ 22:45</p>
07 / 28	Put motor #12 in oven #1 @ 11:50
07 / 29	Stopped insulation aging motor #1 @ 23:30
07 / 30	<p>Took out motor #12 from oven #1 @ 11:50</p> <p>Tested motor #6 into file 0730.zip (Se-V vibration had spikes...)</p> <p>Put motor #6 in oven #1 @ 17:50</p>
08 / 05	Took out motor #6 from oven #1 @ 19:20
08 / 13	<p>Tested motor #3 into file 0813.zip (One of the accelerometers may not be working)</p> <p>Started insulation aging motor#3 @ 15:00</p>
08 / 15	Stopped insulation aging motor #3 @ 15:30
08 / 21	<p>Tested motor #7 in file 0821.zip with the following exception: No CSI device used.</p>
08 / 22	<p>Tested motor #8 into file 0822.zip with the following exception: No CSI device used.</p> <p>Put motor #8 into oven #2 @ 17:15</p>
08 / 27	<p>Took out motor #7 from oven #1 @ 18:40</p> <p>Tested motor #11 in file 0827.zip with the following exceptions: Test data is unreliable, as error conditions have not been observed. esp. after 50% power load.</p> <p>Fluted motor #11 and put it in oven # 3 @ 17:15</p>

Date (1997)	Events (Comments)
08 / 28	<p>Took out motor #8 from oven #2 @ 17:15</p> <p>Tested motor #13 in file 0828.zip</p> <p>CSI unit failed to download data on this test.</p> <p>Fluted motor #13 and put it in oven #4 @ 19:12</p>
08 / 29	Took out motor # 11 and 13 from the ovens as they are classified as failed.
09 / 18	<p>Tested motor #1 into file 0918.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial TC Air is not working 115% shutdown data is missing...</p> <p>Started insulation aging motor #1 @ 19:00</p>
09 / 19	<p>Tested motor #3 into file 0919.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial</p>
09 / 20	Stopped insulation aging motor #1 @ 19:00
09 / 21	Started insulation aging motor #3 @ 15:50
09 / 23	Stopped insulation aging motor #3 @ 14:00
09 / 25	<p>Tested motor #1 into file 0925.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial TC Air not working Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 0.93 Ohms 9-5-3 – 7-6-1 : 0.83 Ohms 4-8-2 – 7-6-1 : 0.96 Ohms</p> <p>Started insulation aging motor # 1 @ 16:36</p>
09 / 26	<p>Tested motor #3 into file 0926.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 1.65 Ohms 9-5-3 – 7-6-1 : 1.29 Ohms 4-8-2 – 7-6-1 : 1.26 Ohms</p> <p>Up to 25% there is a current imbalance due to bad terminal connection, was corrected later on...</p>
09 / 27	Stopped insulation aging motor #1 @ 17:06
09 / 30	Started insulation aging motor #3 @ 17:00

Date (1997)	Events (Comments)
10 / 01	Tested motor #1 into file 1001.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial TC Air not working Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 0.96 Ohms 9-5-3 – 7-6-1 : 1.02 Ohms 4-8-2 – 7-6-1 : 1.21 Ohms
10 / 02	Stopped insulation aging motor #3 @ 24:40
10 / 03	Tested motor #3 into file 1003.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 1.41 Ohms 9-5-3 – 7-6-1 : 1.30 Ohms 4-8-2 – 7-6-1 : 1.19 Ohms Started insulation aging motor #1 @ 10:07
10 / 05	Stopped insulation aging motor #1 @ 10:40
10 / 08	Started insulation aging motor #3 @ 16:21
10 / 09	Motor #3 failed during insulation aging between 13:00 – 15:30 The following resistance values are recorded thereafter: 9-5-3 – 4-8-2 : 3.26 Ohms 9-5-3 – 7-6-1 : 1.93 Ohms 4-8-2 – 7-6-1 : 1.92 Ohms
10 / 10	Tested motor #1 into file 1010.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial TC Air not working Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 1.18 Ohms 9-5-3 – 7-6-1 : 1.29 Ohms 4-8-2 – 7-6-1 : 1.18 Ohms
11 / 16	Started insulation aging motor # 1 @ 14:17
11 / 18	Stopped insulation aging motor # 1 @ 16:14
11 / 19	Tested motor # 4 in 1119.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 1.29 Ohms 9-5-3 – 7-6-1 : 1.15 Ohms 4-8-2 – 7-6-1 : 1.32 Ohms 0% coupled shut-down data missing! This motor is suspected to have rotor bar problem.

Date (1997)	Events (Comments)
11 / 21	Tested motor #1 in 1121.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial TC Air not working! Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 8.0 Ohms 9-5-3 – 7-6-1 : 4.70 Ohms 4-8-2 – 7-6-1 : 1.55 Ohms Motor failed on-line at 0% coupled start-up.
11 / 24	Tested motor #5 in 1124.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 0.86 Ohms 9-5-3 – 7-6-1 : 0.83 Ohms 4-8-2 – 7-6-1 : 0.97 Ohms Started insulation aging motor # 5 @ 16:20 at 215°
11 / 26	Stopped insulation aging motor #5 @ 18:20
12 / 1	Tested motor #5 in 1124.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 0.85 Ohms 9-5-3 – 7-6-1 : 0.84 Ohms 4-8-2 – 7-6-1 : 0.84 Ohms Started insulation aging motor # 5 @ 16:40 Failed around 18:30
12 / 3	Stopped insulation aging motor #5 @ 15:20
12 / 5	Tested motor #5 in 1205.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 1.02 Ohms 9-5-3 – 7-6-1 : 1.85 Ohms 4-8-2 – 7-6-1 : 1.85 Ohms Failed after 25% load.
12 / 8	Tested motor #9 in 1208.zip with the following exceptions: Accelerometer SE – V = Magnetic Flux Axial Accelerometer SE – H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 0.88 Ohms 9-5-3 – 7-6-1 : 0.86 Ohms 4-8-2 – 7-6-1 : 0.83 Ohms Started insulation aging motor #9 @ 17:05 at 215°
12 / 10	Stopped insulation aging motor #9 @ 18:40

Date (1997)	Events (Comments)
12 / 11	Tested motor #9 in 1211.zip with the following exceptions: Accelerometer SE - V = Magnetic Flux Axial Accelerometer SE - H = Magnetic Flux Radial Resistance values for the motor before test was: 9-5-3 – 4-8-2 : 0.85 Ohms 9-5-3 – 7-6-1 : 0.83 Ohms 4-8-2 – 7-6-1 : 0.84 Ohms Started insulation aging motor #9 @ 17:25 at 215°
12 / 12	Motor #9 was at 192°C instead of 215°C, possible failure may have occurred.
12 / 13	Stopped insulation aging motor #9 @ 18:10
12 / 15	Tested motor #9, however it failed immediately with the following resistance values: 9-5-3 – 4-8-2 : 0.98 Ohms 9-5-3 – 7-6-1 : 0.95 Ohms 4-8-2 – 7-6-1 : 1.62 Ohms

APPENDIX B

LISTING OF DATA ACQUISITION SOFTWARE

```
#include <formatio.h>
#include <ansi_c.h>
#include <analysis.h>
#include <utility.h>
#include <dataacq.h>
#include <cvirte.h>          /* Needed if linking in external compiler; harmless
otherwise */
#include <userint.h>
#include "ai_acq.h"
#include "convert.h"

/* module type constants */
#define SCXI_1121  2L
#define SCXI_1120  4L
#define SCXI_1100  6L
#define SCXI_1140  8L
#define SCXI_1122 10L
#define SCXI_1102 30L
#define SCXI_1141 32L
/* these modules are NOT supported in this example */
#define SCXI_1124 20L
#define SCXI_1160 12L
#define SCXI_1161 14L
#define SCXI_1162 16L
#define SCXI_1162HV 24L
#define SCXI_1163 18L
#define SCXI_1163R 28L

/* data acquisition device type constants */
#define LAB_PC      9
#define PC_LPM_16   13
#define LABPC_PLUS  28
#define SCXI_1200   30
#define DAQCARD_700 31
#define DAQPAD_1200 33
#define DAQCARD_1200 48
#define PC_LPM_16PNP 56
#define LAB_PC_1200 57
#define LAB_PC_1200AI 58
#define PCI_1200    201

#define MAX_BUFFER_SIZE 3840000
/* max number of SCXI channels in one chassis */
#define MAX_SCXI_CHANS 384

/* binary data buffer */
short dataBuf[MAX_BUFFER_SIZE];

short chanBuf[240000];
double voltBuf[240000];

double motorpower[240000];
double loadpower[240000];

double offset[36];

double currentoff[4];
```

```

double CJTemp[3];

double checksteady[4];

/* user interface panel handles */
int muxHandle, entryHandle, parallelHandle;

/* SCXI multiplexed scan paramters */
int numModules;
short  numChansList[12], moduleList[12], startChansList[12];

static int panelHandle;

char timestampstring[15];

int tc_file_handle;
int mt_file_handle;

int logging;

void StoreArray ( double x[], int count, int file_handle)
{
    FmtFile (file_handle, "%*f<%*f", count, count, x);
}

void StorePoint ( double x, int file_handle)
{
    FmtFile (file_handle, "%f<%f",x);
}

void OpenTCFile()
{
    char filename[30];
    int i;

    CopyString(filename,0,"c:\\mrc\\data\\",0,12);
    CopyString(filename,12,timestampstring,0,14);
    CopyString(filename,26, ".tc",0,3);

    tc_file_handle=OpenFile(filename,2,0,0);

    if (tc_file_handle<0) {
        SetCtrlVal (panelHandle, PANEL_error, tc_file_handle);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
}

void OpenMTFile()
{
    char filename[30];
    int i;

    CopyString(filename,0,"c:\\mrc\\data\\",0,12);
    CopyString(filename,12,timestampstring,0,14);
    CopyString(filename,26, ".mt",0,3);

    mt_file_handle=OpenFile(filename,2,0,0);

    if (tc_file_handle<0) {
        SetCtrlVal (panelHandle, PANEL_error, mt_file_handle);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
}

```



```

}

/* Check for Steady State */

int CheckSteadyState()
{
    int i,j;
    double temperature;
    GetCtrlVal (panelHandle, PANEL_MOTOR_1, &temperature);
    for(i=0;i<3;i++) checksteady[i]=checksteady[i+1];
    checksteady[3]=temperature;
    temperature=abs(checksteady[3]-checksteady[0]);
    if(temperature<=0.2)
        SetCtrlVal (panelHandle, PANEL_steadystate, 1);
    else
        SetCtrlVal (panelHandle, PANEL_steadystate, 0);
    return 0;
}

/* Stuff with time stamp */

int PutTimeStamp ()
{
    int i,j,err,i1,i2,i3;
    double timestamp,datestamp;
    char timestring[7];
    char datestring[15];
    char tempstring[7];

    err=GetSystemTime(&i1,&i2,&i3);
    timestamp= i1*10000.0 + i2 * 100.0 + i3;
    err=GetSystemDate(&i1,&i2,&i3);
    datestamp = i3*10000.0 + i1 * 100.0 + i2;
    if(timestamp<100000.0)
    {
        if(timestamp<10000.0)
        {
            Fmt( timestring, "%s<%f[p0]",0.0);
            Fmt( tempstring, "%s<%f[p0]",0.0);
            CopyString(timestring,1,tempstring,0,1);
            Fmt( tempstring, "%s<%f[p0]",timestamp);
            CopyString(timestring,2,tempstring,0,5);
        }
        else
        {
            Fmt( timestring, "%s<%f[p0]",0.0);
            Fmt( tempstring, "%s<%f[p0]",timestamp);
            CopyString(timestring,1,tempstring,0,5);
        }
    }
    else
    {
        Fmt( timestring, "%s<%f[p0]",timestamp);
    }
    Fmt( datestring, "%s<%f[p0]",datestamp);
    for(i=0;i<8;i++) timestampstring[i]=datestring[i];
    for(i=0;i<7;i++) timestampstring[8+i]=timestring[i];
    SetCtrlVal (panelHandle, PANEL_FILESTAMP, timestampstring);
    return 0;
}

int CVICALLBACK UpdateMarker (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{

```

```

double x,y;
int plohandle, index;
double freq;
    switch (event) {
        case EVENT_COMMIT:
            GetGraphCursor (panelHandle, PANEL_GRAPHFREQ, 2, &x, &y);
            GetGraphCursorIndex (panelHandle, PANEL_GRAPHFREQ, 2, &plohandle,
&index);
            SetCtrlVal (panelHandle, PANEL_YREADOUT, y);
            freq = (double) index * 12000.0 / 16384.0 ;
            SetCtrlVal (panelHandle, PANEL_INDEX, freq);
            break;
    }
    return 0;
}

```

```

int CollectData( int collectmode)
{
    int err, deviceID, chassisID, opMode, numDAQchans,
        numSCXIchans, sampleCount,
        samplesPerChan, DAQchanIndex, bufIndex, n, eventHandle;
    double sampleRate, scanRate;
    short brdType, finalScanOrder[16], DAQchans[16], DAQgains[16];
    double actualFreq;
    int i,j;
    double const1, const2;
    short offsetbin;
    double meanval, rmsval;

    /*
    double dt,df;
    WindowConst windowConstant;
    char unitString[240];

    double spectra[16384];
    double spectratime[16384];
    */

    switch (collectmode) {
        case 0:

            /* initialize the device and get the device type */
            err = Init_DA_Brds (1, &brdType);
            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }

            /* load the SCXI configuration that was entered in
            WDAQCONF for this chassis */
            err = SCXI_Load_Config (1);
            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }

            /* reset the chassis (-1 means reset entire chassis) */
            err = SCXI_Reset (1, -1);
            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }

            /* Set gains for each module */

```

```

err = SCXI_Set_Gain(1, 1, -1, 1);
    err = SCXI_Set_Gain(1, 3, -1, 20);
    err = SCXI_Set_Gain(1, 3, 3, 5);

/* Read the offsets and calibrate the SCXI 1100 chassis */
for(i=10;i<20;i++)
{
    err = SCXI_Single_Chan_Setup (1, 1, i, 1);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    err = SCXI_Calibrate_Setup (1, 1, 1);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    err = DAQ_Op (1, 0, -1, dataBuf, 4000, 12000.0);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    for(j=0;j<4000;j++) voltBuf[j]= (double) dataBuf[j];
    err = Mean (voltBuf, 4000, &offset[i]);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    offsetbin = (short) offset[i];
    err = SCXI_Cal_Constants (1, 1, -1, 1, 0, 5, 1, 1, 0, -1,
1.0, 0.0,
    offsetbin, 10.0, 10, &const1, &const2);

    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    err = SCXI_Calibrate_Setup (1, 1, 0);
}

/* Read the offsets and calibrate the SCXI 1141 chassis */
for(i=0;i<7;i++)
{
    err = SCXI_Single_Chan_Setup (1, 3, i, 1);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    err = SCXI_Calibrate_Setup (1, 3, 1);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    err = DAQ_Op (1, 0, -1, dataBuf, 4000, 12000.0);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
    for(j=0;j<4000;j++) voltBuf[j]= (double) dataBuf[j];
    err = Mean (voltBuf, 4000, &offset[i]);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }
}

```

```

        offsetbin = (short) offset[i];
        if(i==3)
        {
            err = SCXI_Cal_Constants (1, 3, i, 1, 0, 5, 5, 1, 0,
-1, 1.0, 0.0,
            offsetbin, 10.0, 10, &const1, &const2);
        }
        else
        {
            err = SCXI_Cal_Constants (1, 3, i, 1, 0, 5, 20, 1,
0, -1, 1.0, 0.0,
            offsetbin, 10.0, 10, &const1, &const2);
        }
        if (err<0) {
            SetCtrlVal (panelHandle, PANEL_error, err);
            SetCtrlVal (panelHandle, PANEL_alarm, 1);
        }
        err = SCXI_Calibrate_Setup (1, 3, 0);
    }

    /* Read the cold Junction Temperatures on SCXI 1100 */

    err = SCXI_Single_Chan_Setup (1, 1, -1, 1);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    err = DAQ_Op (1, 0, -1, dataBuf, 4000, 12000.0);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    err = SCXI_Scale (1, 1, -1, 1, 1, 1, 0, -1, 4000,
        dataBuf, voltBuf);

    err = Mean (voltBuf, 4000, &CJTemp[0]);
    CJTemp[0]*=100.0;

    /* Read the cold Junction Temperatures on SCXI 1120 */

    err = SCXI_Single_Chan_Setup (1, 2, -1, 1);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    err = DAQ_Op (1, 0, -1, dataBuf, 4000, 12000.0);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    err = SCXI_Scale (1, 2, -1, 1, 1, 1, 0, -1, 4000,
        dataBuf, voltBuf);

    err = Mean (voltBuf, 4000, &CJTemp[1]);
    const1=CJTemp[1];
    Thermistor_Convert (5000.0, 2.5, const1, CELSIUS, &CJTemp[1]);

    /* Read the cold Junction Temperatures on SCXI 1120 */

```

```

err = SCXI_Single_Chan_Setup (1, 4, -1, 1);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

err = DAQ_Op (1, 0, -1, dataBuf, 4000, 12000.0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}
err = SCXI_Scale (1, 4, -1, 1, 1, 1, 0, -1, 4000,
                 dataBuf, voltBuf);
err = Mean (voltBuf, 4000, &CJTemp[2]);
const1=CJTemp[2];
Thermistor_Convert (5000.0, 2.5, const1, CELSIUS, &CJTemp[2]);

/* Measure Current Sensor Offsets */

/* Initialize variables */
for (i=0; i<16; i++) {
    DAQchans[i] = 0;
    DAQgains[i] = -1;
}

/* initialize multiplexed SCXI scan parameters */
for (i=0; i<12; i++) {
    moduleList[i] = 1;
    numChansList[i] = 1;
    startChansList[i] = 0;
}
numModules = 1;
numDAQchans = 1;

    moduleList[0]=1;
    startChansList[0]=14;
    numChansList[0]=4;

deviceID=1;
chassisID=1;
opMode=0;

samplesPerChan = 12000;
sampleRate = 48000.0;
scanRate = 0.0;

SetCtrlVal (panelHandle, PANEL_error, 0);
SetCtrlVal (panelHandle, PANEL_alarm, 0);

err = SCXI_Set_Gain(chassisID, 1, -1, 1);

/* total up the number of channels */
numSCXIchans = 0;
for (i=0; i<numModules; i++)
    numSCXIchans += numChansList[i];

err = SCXI_SCAN_Setup (chassisID, numModules, moduleList, numChansList,
                      startChansList, deviceID, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

```

```

/* set up the mux counter value to equal the total number of channels
to be scanned in one pass through the module scan list, because
we are only going to scan one channel on the DAQ device */
err = SCXI_MuxCtr_Setup (deviceID, 1, 1, numSCXIchans);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/*
DeleteGraphPlot (panelHandle, PANEL_GRAPHTIME, -1, 1);
DeleteGraphPlot (panelHandle, PANEL_GRAPHFREQ, -1, 1);
*/

SetCtrlVal (panelHandle, PANEL_scanning, 1);
sampleCount = numSCXIchans * samplesPerChan;
err = SCAN_Op (deviceID, numDAQchans, DAQchans, DAQgains,
              dataBuf, sampleCount, sampleRate, scanRate);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* demux the data; numSCXIchans was determined in the subroutine */
err = SCAN_Demux (dataBuf, sampleCount, numSCXIchans, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

for(i=0;i<4;i++) currentoff[i]=0.0;

for(j=0;j<4;j++)
{
    for(i=0;i<12000;i++) chanBuf[i]=dataBuf[i+j*12000];
    err = SCXI_Scale (1, 1, 14+j, 1, 1, 1, 0, -1, 12000,
                    chanBuf, voltBuf);
    err = Mean (voltBuf, 12000, &currentoff[j]);
}

SetCtrlVal (panelHandle, PANEL_scanning, 0);

break;

case 1:

/* Real DAQ */
/* Initialize variables */

for (i=0; i<16; i++) {
DAQchans[i] = 0;
DAQgains[i] = -1;
}

/* initialize multiplexed SCXI scan parameters */
for (i=0; i<12; i++) {
    moduleList[i] = 1;
    numChansList[i] = 1;
    startChansList[i] = 0;
}
numModules = 2;

numDAQchans = 1;

moduleList[0]=1;

```

```

        startChansList[0]=10;
        numChansList[0]=10;
        moduleList[1]=3;
        startChansList[1]=2;
        numChansList[1]=6;

deviceID=1;
chassisID=1;
opMode=0;

samplesPerChan = 120000;
sampleRate = 192000.0;
scanRate = 0.0;

SetCtrlVal (panelHandle, PANEL_error, 0);
SetCtrlVal (panelHandle, PANEL_alarm, 0);

        err = SCXI_Set_Gain(chassisID, 1, -1, 1);
        err = SCXI_Set_Gain(chassisID, 3, -1, 20);
        err = SCXI_Set_Gain(chassisID, 3, 3, 5);
        err = SCXI_Configure_Filter (chassisID, 3, -1, 1, 4000.0, 0, 0,
&actualFreq);
        err = SCXI_Configure_Filter (chassisID, 3, -1, 3, 4000.0, 0, 0,
&actualFreq);

        /* total up the number of channels */
numSCXIchans = 0;
for (i=0; i<numModules; i++)
    numSCXIchans += numChansList[i];

err = SCXI_SCAN_Setup (chassisID, numModules, moduleList, numChansList,
startChansList, deviceID, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* set up the mux counter value to equal the total number of channels
to be scanned in one pass through the module scan list, because
we are only going to scan one channel on the DAQ device */
err = SCXI_MuxCtr_Setup (deviceID, 1, 1, numSCXIchans);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

        /*
DeleteGraphPlot (panelHandle, PANEL_GRAPHTIME, -1, 1);
DeleteGraphPlot (panelHandle, PANEL_GRAPHFREQ, -1, 1);
*/

    SetCtrlVal (panelHandle, PANEL_scanning, 1);
sampleCount = numSCXIchans * samplesPerChan;
    err = SCAN_Op (deviceID, numDAQchans, DAQchans, DAQgains,
dataBuf, sampleCount, sampleRate, scanRate);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* demux the data; numSCXIchans was determined in the subroutine */
err = SCAN_Demux (dataBuf, sampleCount, numSCXIchans, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);

```

```

    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* Now compute the values for each of them */
for(i=0;i<120000;i++) motorpower[i]=0.0;

/* Vx */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+2*120000];

    err = SCXI_Scale (1, 1, 12, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_13, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_14, rmsval);

/* Ix */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+4*120000];

    err = SCXI_Scale (1, 1, 14, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[0])*20.0;
        motorpower[i]+=loadpower[i]*voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_24, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_19, rmsval);

/* Vy */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+120000];

    err = SCXI_Scale (1, 1, 11, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_15, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_16, rmsval);

```



```

/* Iy */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+5*120000];

    err = SCXI_Scale (1, 1, 15, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[1])*20.0;
        motorpower[i]+=loadpower[i]*voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_23, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_22, rmsval);
/* Vz */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i];

    err = SCXI_Scale (1, 1, 10, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_17, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_18, rmsval);

/* Iz */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+6*120000];

    err = SCXI_Scale (1, 1, 16, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[2])*20.0;
        motorpower[i]+=loadpower[i]*voltBuf[i];
        motorpower[i]/=sqrt(3.0);
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_21, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_20, rmsval);
/* V1 */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+3*120000];

```

```

err = SCXI_Scale (1, 1, 13, 1, 1, 1, 0, -1, 120000,
                 chanBuf, voltBuf);

for(i=0;i<120000;i++) {
    voltBuf[i]*=50.0;
    loadpower[i]=voltBuf[i];
}

err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_28, meanval);

err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_27, rmsval);

/* I1 */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+7*120000];

err = SCXI_Scale (1, 1, 17, 1, 1, 1, 0, -1, 120000,
                 chanBuf, voltBuf);

for(i=0;i<120000;i++) {
    voltBuf[i]=(voltBuf[i]-currentoff[3])*20.0;
    loadpower[i]*=voltBuf[i];
}

err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_26, meanval);

err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_25, rmsval);

/* Speed */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+8*120000];

err = SCXI_Scale (1, 1, 18, 1, 1, 1, 0, -1, 120000,
                 chanBuf, voltBuf);

for(i=0;i<120000;i++) voltBuf[i]*=200.0;

err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_32, meanval);

/* Torque */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+9*120000];

err = SCXI_Scale (1, 1, 19, 1, 1, 1, 0, -1, 120000,
                 chanBuf, voltBuf);

for(i=0;i<120000;i++) voltBuf[i]*=2.5;

err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_30, meanval);

err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_29, rmsval);

/* Cover */

```

```

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+10*120000];

    err = SCXI_Scale (1, 3, 2, 20, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) voltBuf[i]*=10.0;

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_41, rmsval);

/* Short End Axial */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+11*120000];

    err = SCXI_Scale (1, 3, 3, 5, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) voltBuf[i]*=10.0;

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_40, rmsval);

/* Short End Horizontal */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+12*120000];

    err = SCXI_Scale (1, 3, 4, 20, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) voltBuf[i]*=10.0;

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_39, rmsval);

/* Short End Vertical */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+13*120000];

    err = SCXI_Scale (1, 3, 5, 20, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) voltBuf[i]*=10.0;

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_38, rmsval);

/* Pulley End 10:00 */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+14*120000];

    err = SCXI_Scale (1, 3, 6, 20, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) voltBuf[i]*=10.0;

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_37, rmsval);

/* Pulley End 2:00 */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+15*120000];

```

```

err = SCXI_Scale (1, 3, 7, 20, 1, 1, 0, -1, 120000,
                 chanBuf, voltBuf);

for(i=0;i<120000;i++) voltBuf[i]*=10.0;

err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_36, rmsval);

/* Load Power */

err = Mean ( loadpower, 120000, &meanval);

SetCtrlVal(panelHandle, PANEL_MOTOR_34, meanval);

/* True Power */

err = Mean ( motorpower, 120000, &meanval);

SetCtrlVal(panelHandle, PANEL_MOTOR_33, meanval);

/* Apparent Power */

rmsval=0.0;

GetCtrlVal(panelHandle, PANEL_MOTOR_14, &const1);
GetCtrlVal(panelHandle, PANEL_MOTOR_19, &const2);
rmsval+=const1*const2;
GetCtrlVal(panelHandle, PANEL_MOTOR_16, &const1);
GetCtrlVal(panelHandle, PANEL_MOTOR_22, &const2);
rmsval+=const1*const2;
GetCtrlVal(panelHandle, PANEL_MOTOR_18, &const1);
GetCtrlVal(panelHandle, PANEL_MOTOR_20, &const2);
rmsval+=const1*const2;
rmsval/=sqrt(3.0);

SetCtrlVal(panelHandle, PANEL_MOTOR_31, rmsval);

/* Overall Power Factor */

GetCtrlVal(panelHandle, PANEL_MOTOR_33, &meanval);

meanval/=rmsval;

SetCtrlVal(panelHandle, PANEL_MOTOR_35, meanval);

/* Set gains for each module */

SetCtrlVal (panelHandle, PANEL_scanning, 0);

break;
case 2:

/* Real DAQ */
/* Initialize variables */

for (i=0; i<16; i++) {
DAQchans[i] = 0;
DAQgains[i] = -1;
}

```

```

/* initialize multiplexed SCXI scan parameters */
for (i=0; i<12; i++) {
    moduleList[i] = 1;
    numChansList[i] = 1;
    startChansList[i] = 0;
}
numModules = 3;

numDAQchans = 1;

    moduleList[0]=1;
    startChansList[0]=0;
    numChansList[0]=5;
    moduleList[1]=2;
    startChansList[1]=0;
    numChansList[1]=8;
    moduleList[2]=4;
    startChansList[2]=0;
    numChansList[2]=6;

deviceID=1;
chassisID=1;
opMode=0;

samplesPerChan = 600;
sampleRate = 22800.0;
scanRate = 0.0;

SetCtrlVal (panelHandle, PANEL_error, 0);
SetCtrlVal (panelHandle, PANEL_alarm, 0);

    err = SCXI_Set_Gain(chassisID, 1, -1, 200);

    /* total up the number of channels */
numSCXIchans = 0;
for (i=0; i<numModules; i++)
    numSCXIchans += numChansList[i];

err = SCXI_SCAN_Setup (chassisID, numModules, moduleList, numChansList,
    startChansList, deviceID, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* set up the mux counter value to equal the total number of channels
to be scanned in one pass through the module scan list, because
we are only going to scan one channel on the DAQ device */
err = SCXI_MuxCtr_Setup (deviceID, 1, 1, numSCXIchans);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

SetCtrlVal (panelHandle, PANEL_scanning, 1);
sampleCount = numSCXIchans * samplesPerChan;
err = SCAN_Op (deviceID, numDAQchans, DAQchans, DAQgains,
    dataBuf, sampleCount, sampleRate, scanRate);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* demux the data; numSCXIchans was determined in the subroutine */
err = SCAN_Demux (dataBuf, sampleCount, numSCXIchans, 0);

```

```

if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* Now Compute the Temperatures */

/* Oven Temperature 1 */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i];

err = SCXI_Scale (1, 1, 0, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (3, 25.6, 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_OVEN_1, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Oven Temperature 2 */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+1*600];
err = SCXI_Scale (1, 1, 1, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (3, CJTemp[0], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_OVEN_2, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Oven Temperature 3 */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+2*600];
err = SCXI_Scale (1, 1, 2, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (3, CJTemp[0], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_OVEN_3, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Oven Temperature 4 */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+3*600];
err = SCXI_Scale (1, 1, 3, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (3, CJTemp[0], 1, meanval, &rmsval);

```

```

SetCtrlVal( panelHandle, PANEL_OVEN_4, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Oven Temperature 5 */

for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+4*600];
err = SCXI_Scale (1, 1, 4, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (3, CJTemp[0], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_OVEN_5, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Winding Pulley End 4:00 Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+5*600];
err = SCXI_Scale (1, 2, 0, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_1, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Winding Pulley End 8:00 Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+6*600];
err = SCXI_Scale (1, 2, 1, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_2, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Winding Pulley End 12:00 Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+7*600];
err = SCXI_Scale (1, 2, 2, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);
err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_3, rmsval);

```

```

if(logging) StorePoint(rmsval,tc_file_handle);

/* Winding Short End 2:00 Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+8*600];
err = SCXI_Scale (1, 2, 3, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_4, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Winding Short End 6:00 Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+9*600];
err = SCXI_Scale (1, 2, 4, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_5, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Winding Pulley End 10:00 Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+10*600];
err = SCXI_Scale (1, 2, 5, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_6, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Bearing Surface Pulley End Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+11*600];
err = SCXI_Scale (1, 2, 6, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_7, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

```



```

/* Bearing Surface Short End Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+12*600];
err = SCXI_Scale (1, 2, 7, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[1], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_8, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

SetCtrlVal (panelHandle, PANEL_scanning, 0);

/* Spare Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+13*600];
err = SCXI_Scale (1, 4, 0, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[2], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_SPARE, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Lamination Steel Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+14*600];
err = SCXI_Scale (1, 4, 1, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[2], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_9, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

/* Air Between Windowing & Shell Temperature */
for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+15*600];
err = SCXI_Scale (1, 4, 2, 200, 1, 1, 0, -1, 600,
                 chanBuf, voltBuf);

err = Mean (voltBuf, 600, &meanval);

err = Thermocouple_Convert (2, CJTemp[2], 1, meanval, &rmsval);
SetCtrlVal( panelHandle, PANEL_MOTOR_10, rmsval);
if(logging) StorePoint(rmsval,tc_file_handle);

```

```

        /* Inside Cover Temperature */
        for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+16*600];

        err = SCXI_Scale (1, 4, 3, 200, 1, 1, 0, -1, 600,
                        chanBuf, voltBuf);

        err = Mean (voltBuf, 600, &meanval);

        err = Thermocouple_Convert (2, CJTemp[2], 1, meanval, &rmsval);
        SetCtrlVal( panelHandle, PANEL_MOTOR_11, rmsval);
        if(logging) StorePoint(rmsval,tc_file_handle);

        /* Ambient Inside Cover Temperature */
        for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+17*600];

        err = SCXI_Scale (1, 4, 4, 200, 1, 1, 0, -1, 600,
                        chanBuf, voltBuf);

        err = Mean (voltBuf, 600, &meanval);

        err = Thermocouple_Convert (2, CJTemp[2], 1, meanval, &rmsval);
        SetCtrlVal( panelHandle, PANEL_MOTOR_12, rmsval);
        if(logging) StorePoint(rmsval,tc_file_handle);

/* Ambient Temperature */

        for(i=0;i<600;i++) chanBuf[i]=dataBuf[i+18*600];

        err = SCXI_Scale (1, 4, 5, 200, 1, 1, 0, -1, 600,
                        chanBuf, voltBuf);

        err = Mean (voltBuf, 600, &meanval);

        err = Thermocouple_Convert (2, CJTemp[2], 1, meanval, &rmsval);
        SetCtrlVal( panelHandle, PANEL_Ambient, rmsval);
        if(logging) StorePoint(rmsval,tc_file_handle);

break;
    case 3:

        /* Real DAQ */
        /* Initialize variables */

        for (i=0; i<16; i++) {
            DAQchans[i] = 0;
            DAQgains[i] = -1;
        }

        /* initialize multiplexed SCXI scan parameters */
        for (i=0; i<12; i++) {
            moduleList[i] = 1;
            numChansList[i] = 1;
            startChansList[i] = 0;
        }
        numModules = 2;
        numDAQchans = 1;

```

```

        moduleList[0]=1;
        startChansList[0]=10;
        numChansList[0]=10;
        moduleList[1]=3;
        startChansList[1]=2;
        numChansList[1]=6;

deviceID=1;
chassisID=1;
opMode=0;

samplesPerChan = 120000;
sampleRate = 192000.0;
scanRate = 0.0;

SetCtrlVal (panelHandle, PANEL_error, 0);
SetCtrlVal (panelHandle, PANEL_alarm, 0);

        err = SCXI_Set_Gain(chassisID, 1, -1, 1);
        err = SCXI_Set_Gain(chassisID, 3, -1, 20);
        err = SCXI_Set_Gain(chassisID, 3, 3, 5);
        err = SCXI_Configure_Filter (chassisID, 3, -1, 1, 4000.0, 0, 0,
&actualFreq);
        err = SCXI_Configure_Filter (chassisID, 3, -1, 3, 4000.0, 0, 0,
&actualFreq);

        /* total up the number of channels */
numSCXIchans = 0;
for (i=0; i<numModules; i++)
    numSCXIchans += numChansList[i];

err = SCXI_SCAN_Setup (chassisID, numModules, moduleList, numChansList,
startChansList, deviceID, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* set up the mux counter value to equal the total number of channels
to be scanned in one pass through the module scan list, because
we are only going to scan one channel on the DAQ device */
err = SCXI_MuxCtr_Setup (deviceID, 1, 1, numSCXIchans);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

        /*
DeleteGraphPlot (panelHandle, PANEL_GRAPHTIME, -1, 1);
DeleteGraphPlot (panelHandle, PANEL_GRAPHFREQ, -1, 1);
*/

SetCtrlVal (panelHandle, PANEL_scanning, 1);
sampleCount = numSCXIchans * samplesPerChan;
err = SCAN_Op (deviceID, numDAQchans, DAQchans, DAQgains,
dataBuf, sampleCount, sampleRate, scanRate);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* demux the data; numSCXIchans was determined in the subroutine */
err = SCAN_Demux (dataBuf, sampleCount, numSCXIchans, 0);

```

```

if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* Now compute the values for each of them */

for(i=0;i<120000;i++) motorpower[i]=0.0;

/* Vx */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+2*120000];

    err = SCXI_Scale (1, 1, 12, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_13, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_14, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Ix */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+4*120000];

    err = SCXI_Scale (1, 1, 14, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[0])*20.0;
        motorpower[i]+=loadpower[i]*voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_24, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_19, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Vy */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+120000];

    err = SCXI_Scale (1, 1, 11, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);

```

```

SetCtrlVal(panelHandle, PANEL_MOTOR_15, meanval);

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_16, rmsval);

if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Iy */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+5*120000];

    err = SCXI_Scale (1, 1, 15, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[1])*20.0;
        motorpower[i]+=loadpower[i]*voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);

SetCtrlVal(panelHandle, PANEL_MOTOR_23, meanval);

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_22, rmsval);

if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Vz */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i];

    err = SCXI_Scale (1, 1, 10, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);

SetCtrlVal(panelHandle, PANEL_MOTOR_17, meanval);

    err = RMS (voltBuf, 120000, &rmsval);

SetCtrlVal(panelHandle, PANEL_MOTOR_18, rmsval);

    if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Iz */

for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+6*120000];

    err = SCXI_Scale (1, 1, 16, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[2])*20.0;
        motorpower[i]+=loadpower[i]*voltBuf[i];
        motorpower[i]/=sqrt(3.0);
    }

```

```

    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_21, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_20, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* V1 */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+3*120000];

    err = SCXI_Scale (1, 1, 13, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]*=50.0;
        loadpower[i]=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_28, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_27, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* I1 */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+7*120000];

    err = SCXI_Scale (1, 1, 17, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) {
        voltBuf[i]=(voltBuf[i]-currentoff[3])*20.0;
        loadpower[i]*=voltBuf[i];
    }

    err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_26, meanval);

    err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_25, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Speed */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+8*120000];

    err = SCXI_Scale (1, 1, 18, 1, 1, 1, 0, -1, 120000,
                    chanBuf, voltBuf);

    for(i=0;i<120000;i++) voltBuf[i]*=200.0;

```

```

        err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_32, meanval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Torque */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+9*120000];
        err = SCXI_Scale (1, 1, 19, 1, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);
        for(i=0;i<120000;i++) voltBuf[i]*=2.5;
        err = Mean (voltBuf, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_30, meanval);
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_29, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Cover */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+10*120000];
        err = SCXI_Scale (1, 3, 2, 20, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);
        for(i=0;i<120000;i++) voltBuf[i]*=10.0;
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_41, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Short End Axial */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+11*120000];
        err = SCXI_Scale (1, 3, 3, 5, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);
        for(i=0;i<120000;i++) voltBuf[i]*=10.0;
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_40, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Short End Horizontal */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+12*120000];
        err = SCXI_Scale (1, 3, 4, 20, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);

```

```

        for(i=0;i<120000;i++) voltBuf[i]*=10.0;
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_39, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Short End Vertical */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+13*120000];
        err = SCXI_Scale (1, 3, 5, 20, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);

        for(i=0;i<120000;i++) voltBuf[i]*=10.0;
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_38, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Pulley End 10:00 */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+14*120000];
        err = SCXI_Scale (1, 3, 6, 20, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);

        for(i=0;i<120000;i++) voltBuf[i]*=10.0;
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_37, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Pulley End 2:00 */
for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+15*120000];
        err = SCXI_Scale (1, 3, 7, 20, 1, 1, 0, -1, 120000,
                           chanBuf, voltBuf);

        for(i=0;i<120000;i++) voltBuf[i]*=10.0;
        err = RMS (voltBuf, 120000, &rmsval);
SetCtrlVal(panelHandle, PANEL_MOTOR_36, rmsval);
if(logging) StoreArray(voltBuf,120000,mt_file_handle);

/* Load Power */
        err = Mean ( loadpower, 120000, &meanval);
SetCtrlVal(panelHandle, PANEL_MOTOR_34, meanval);
/* True Power */
        err = Mean ( motorpower, 120000, &meanval);

```



```

        SetCtrlVal(panelHandle, PANEL_MOTOR_33, meanval);

        /* Apparent Power */

        rmsval=0.0;

        GetCtrlVal(panelHandle, PANEL_MOTOR_14, &const1);
        GetCtrlVal(panelHandle, PANEL_MOTOR_19, &const2);
        rmsval+=const1*const2;
        GetCtrlVal(panelHandle, PANEL_MOTOR_16, &const1);
        GetCtrlVal(panelHandle, PANEL_MOTOR_22, &const2);
        rmsval+=const1*const2;
        GetCtrlVal(panelHandle, PANEL_MOTOR_18, &const1);
        GetCtrlVal(panelHandle, PANEL_MOTOR_20, &const2);
        rmsval+=const1*const2;
        rmsval/=sqrt(3.0);

        SetCtrlVal(panelHandle, PANEL_MOTOR_31, rmsval);

        /* Overall Power Factor */

        GetCtrlVal(panelHandle, PANEL_MOTOR_33, &meanval);

        meanval/=rmsval;

        SetCtrlVal(panelHandle, PANEL_MOTOR_35, meanval);

/* Low Frequency Scanning */

/* Real DAQ */
    /* Initialize variables */

        for (i=0; i<16; i++) {
        DAQchans[i] = 0;
            DAQgains[i] = -1;
        }

/* initialize multiplexed SCXI scan parameters */
for (i=0; i<12; i++) {
    moduleList[i] = 1;
    numChansList[i] = 1;
    startChansList[i] = 0;
}
numModules = 2;

numDAQchans = 1;

    moduleList[0]=1;
    startChansList[0]=10;
    numChansList[0]=10;
    moduleList[1]=3;
    startChansList[1]=2;
    numChansList[1]=6;

deviceID=1;
chassisID=1;
opMode=0;

samplesPerChan = 240000;
sampleRate = 64000.0;
scanRate = 0.0;

```

```

SetCtrlVal (panelHandle, PANEL_error, 0);
SetCtrlVal (panelHandle, PANEL_alarm, 0);

err = SCXI_Set_Gain(chassisID, 1, -1, 1);
err = SCXI_Set_Gain(chassisID, 3, -1, 20);
err = SCXI_Set_Gain(chassisID, 3, 3, 5);
err = SCXI_Configure_Filter (chassisID, 3, -1, 1, 200.0, 0, 0,
&actualFreq);
err = SCXI_Configure_Filter (chassisID, 3, -1, 3, 200.0, 0, 0,
&actualFreq);

/* total up the number of channels */
numSCXIchans = 0;
for (i=0; i<numModules; i++)
    numSCXIchans += numChansList[i];

err = SCXI_SCAN_Setup (chassisID, numModules, moduleList, numChansList,
startChansList, deviceID, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* set up the mux counter value to equal the total number of channels
to be scanned in one pass through the module scan list, because
we are only going to scan one channel on the DAQ device */
err = SCXI_MuxCtr_Setup (deviceID, 1, 1, numSCXIchans);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

SetCtrlVal (panelHandle, PANEL_scanning, 1);
sampleCount = numSCXIchans * samplesPerChan;
err = SCAN_Op (deviceID, numDAQchans, DAQchans, DAQgains,
dataBuf, sampleCount, sampleRate, scanRate);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* demux the data; numSCXIchans was determined in the subroutine */
err = SCAN_Demux (dataBuf, sampleCount, numSCXIchans, 0);
if (err<0) {
    SetCtrlVal (panelHandle, PANEL_error, err);
    SetCtrlVal (panelHandle, PANEL_alarm, 1);
}

/* Now compute the values for each of them */

/* Vx */
for(i=0;i< 240000;i++) chanBuf[i]=dataBuf[i-2* 240000];

err = SCXI_Scale (1, 1, 12, 1, 1, 1, 0, -1, 240000,
chanBuf, voltBuf);

for(i=0;i< 240000;i++) {
    voltBuf[i]*=50.0;
}

err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);

for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];

```

```

if(logging) StoreArray(voltBuf,40000,mt_file_handle);

/* Temporary

DeleteGraphPlot (panelHandle, PANEL_GRAPHTIME, -1, 1);
err = PlotY (panelHandle, PANEL_GRAPHTIME, voltBuf, 1001,
            VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID,
            i, VAL_WHITE);
err = Mean (voltBuf, 120000, &meanval);
for(i=0;i<120000;i++) voltBuf[i]-=meanval;
dt = 1.0 / 4000.0;

for(i=0;i<16384;i++) spectra[i]=0.0;

for(j=0;j<13;j++)
{
    for(i=0;i<16384;i++) spectratime[i]=voltBuf[i+j*8192];

    err = ScaledWindow(spectratime, 16384, 6, &windowConstant);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    err = AutoPowerSpectrum (spectratime, 16384, dt,
spectratime, &df);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    err = SpectrumUnitConversion (spectratime, 8192, 0, 1, 6,
df,
windowConstant, spectratime, unitString);
    if (err<0) {
        SetCtrlVal (panelHandle, PANEL_error, err);
        SetCtrlVal (panelHandle, PANEL_alarm, 1);
    }

    for(i=0;i<16384;i++) spectra[i]=(spectra[i]* (double) j +
spectratime[i]) / (double) (j+1);
}

DeleteGraphPlot (panelHandle, PANEL_GRAPHFREQ, -1, 1);
err = PlotY (panelHandle, PANEL_GRAPHFREQ, spectra, 8192,
            VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID,
            i, VAL_RED);

*/

/* Ix */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+4*240000];

err = SCXI_Scale (1, 1, 14, 1, 1, 1, 0, -1, 240000,
                chanBuf, voltBuf);

for(i=0;i<240000;i++) {
    voltBuf[i]=(voltBuf[i]-currentoff[0])*20.0;
}

err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);

```

```

        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Vy */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+240000];
        err = SCXI_Scale (1, 1, 11, 1, 1, 1, 0, -1, 240000,
                           chanBuf, voltBuf);
        for(i=0;i<240000;i++) {
            voltBuf[i]*=50.0;
        }
        err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Iy */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+5*240000];
        err = SCXI_Scale (1, 1, 15, 1, 1, 1, 0, -1, 240000,
                           chanBuf, voltBuf);
        for(i=0;i<240000;i++) {
            voltBuf[i]=(voltBuf[i]-currentoff[1])*20.0;
        }
        err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Vz */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i];
        err = SCXI_Scale (1, 1, 10, 1, 1, 1, 0, -1, 240000,
                           chanBuf, voltBuf);
        for(i=0;i<240000;i++) {
            voltBuf[i]*=50.0;
        }
        err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Iz */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+5*240000];
        err = SCXI_Scale (1, 1, 16, 1, 1, 1, 0, -1, 240000,
                           chanBuf, voltBuf);
        for(i=0;i<240000;i++) {
            voltBuf[i]=(voltBuf[i]-currentoff[2])*20.0;
        }

```

```

err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
if(logging) StoreArray(voltBuf,40000,mt_file_handle);

/* V1 */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+3*240000];
err = SCXI_Scale (1, 1, 13, 1, 1, 1, 0, -1, 240000,
                 chanBuf, voltBuf);
for(i=0;i<240000;i++) {
    voltBuf[i]*=50.0;
}
err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
if(logging) StoreArray(voltBuf,40000,mt_file_handle);

/* I1 */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+7*240000];
err = SCXI_Scale (1, 1, 17, 1, 1, 1, 0, -1, 240000,
                 chanBuf, voltBuf);
for(i=0;i<240000;i++) {
    voltBuf[i]=(voltBuf[i]-currentoff[3])*20.0;
}
err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
if(logging) StoreArray(voltBuf,40000,mt_file_handle);

/* Speed */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+8*240000];
err = SCXI_Scale (1, 1, 18, 1, 1, 1, 0, -1, 240000,
                 chanBuf, voltBuf);
for(i=0;i<240000;i++) voltBuf[i]*=200.0;
err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
if(logging) StoreArray(voltBuf,40000,mt_file_handle);

/* Torque */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+9*240000];
for(i=0;i<240000;i++) voltBuf[i]*=2.5;
err = SCXI_Scale (1, 1, 19, 1, 1, 1, 0, -1, 240000,
                 chanBuf, voltBuf);
err = Bw_LPF (voltBuf, 240000, 4000.0, 200.0, 5, voltBuf);
for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];

```

```

        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Cover */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+10*240000];
        err = SCXI_Scale (1, 3, 2, 20, 1, 1, 0, -1, 240000,
                        chanBuf, voltBuf);
        for(i=0;i<240000;i++) voltBuf[i]*=10.0;
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Short End Axial */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+11*240000];
        err = SCXI_Scale (1, 3, 3, 5, 1, 1, 0, -1, 240000,
                        chanBuf, voltBuf);
        for(i=0;i<240000;i++) voltBuf[i]*=10.0;
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Short End Horizontal */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+12*240000];
        err = SCXI_Scale (1, 3, 4, 20, 1, 1, 0, -1, 240000,
                        chanBuf, voltBuf);
        for(i=0;i<240000;i++) voltBuf[i]*=10.0;
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Short End Vertical */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+13*240000];
        err = SCXI_Scale (1, 3, 5, 20, 1, 1, 0, -1, 240000,
                        chanBuf, voltBuf);
        for(i=0;i<240000;i++) voltBuf[i]*=10.0;
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);
/* Pulley End 10:00 */
for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+14*240000];
        err = SCXI_Scale (1, 3, 6, 20, 1, 1, 0, -1, 240000,
                        chanBuf, voltBuf);
        for(i=0;i<240000;i++) voltBuf[i]*=10.0;
        for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];
        if(logging) StoreArray(voltBuf,40000,mt_file_handle);

```

```

/* Pulley End 2:00 */

for(i=0;i<240000;i++) chanBuf[i]=dataBuf[i+15*240000];

    err = SCXI_Scale (1, 3, 7, 20, 1, 1, 0, -1, 240000,
                    chanBuf, voltBuf);

    for(i=0;i<240000;i++) voltBuf[i]*=10.0;

    for(i=0;i< 40000;i++) voltBuf[i]=voltBuf[i*6];

    if(logging) StoreArray(voltBuf,40000,mt_file_handle);

    SetCtrlVal (panelHandle, PANEL_scanning, 0);

    break;
}
return 0;
}

int CVICALLBACK QuitCallback (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
    }
    return 0;
}

int CVICALLBACK AcquireCallback (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            if(logging)
            {
                Beep();

                SuspendTimerCallbacks();

                OpenMTFile();

                CollectData(3);
                CollectData(2);
                CollectData(1);

                CloseFile(mt_file_handle);

                PutTimeStamp();

                CloseFile(tc_file_handle);
                OpenTCFile();

                ResumeTimerCallbacks();
                Beep();
            }
            break;
    }
    return 0;
}

int CVICALLBACK RestoreGraph (int panel, int control, int event,

```

```

        void *callbackData, int eventData1, int eventData2)
    {
        switch (event) {
            case EVENT_COMMIT:
                SetAxisRange (panelHandle, PANEL_GRAPHFREQ, VAL_AUTOSCALE, 0, 100,
VAL_AUTOSCALE , 0, 100);
                break;
        }
        return 0;
    }

int CVICALLBACK ZoomIn (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
    {
        double x1, x2, y1, y2;
        double temp;
        switch (event) {
            case EVENT_COMMIT:
                GetGraphCursor (panelHandle, PANEL_GRAPHFREQ, 1, &x1, &y1);
                GetGraphCursor (panelHandle, PANEL_GRAPHFREQ, 3, &x2, &y2);
                if (x1 > x2)
                {
                    temp = x1;
                    x1 = x2;
                    x2 = temp;
                }
                if (y1 > y2)
                {
                    temp = y1;
                    y1 = y2;
                    y2 = temp;
                }
                SetAxisRange (panelHandle, PANEL_GRAPHFREQ, VAL_MANUAL, x1, x2, VAL_MANUAL,
y1,y2);
                break;
        }
        return 0;
    }

int CVICALLBACK DataRecord (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
    {
        switch (event) {
            case EVENT_COMMIT:
                GetCtrlVal (panelHandle, PANEL_BINARYSWITCH, &logging);
                if (logging)
                {
                    OpenTCFile();
                }
                else
                {
                    CloseFile(tc_file_handle);
                }
                break;
        }
        return 0;
    }

int CVICALLBACK PlotMotorData (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
    {
        int i,j;
        int err;
        int imodule , ichannel, igain;
        /* chanBuf and voltBuf will hold any one channel's data; we assume we will

```


be scanning at least 2 channels, so the array size is half the data buffer */

```
double localgain;
double voltrms;
double meanval;
double dt,df;
WindowConst windowConstant;
char unitString[240];

double spectra[16384];
double spectratime[16384];

switch (event) {
    case EVENT_COMMIT:
        switch (control){
            case PANEL_COMMANDBUTTON_1:
                j=2;
                localgain=50.0;
                imodule=1;
                ichannel=12;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_2:
                j=1;
                localgain=50.0;
                imodule=1;
                ichannel=11;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_3:
                j=0;
                localgain=50.0;
                imodule=1;
                ichannel=10;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_4:
                j=4;
                localgain=20.0;
                imodule=1;
                ichannel=14;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_5:
                j=5;
                localgain=20.0;
                imodule=1;
                ichannel=15;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_6:
                j=6;
                localgain=20.0;
                imodule=1;
                ichannel=16;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_7:
                j=3;
                localgain=50.0;
                imodule=1;
                ichannel=13;
                igain=1;
                break;
            case PANEL_COMMANDBUTTON_8:
                j=7;
                localgain=20.0;
```

```

        imodule=1;
        ichannel=17;
        igain=1;
        break;
case PANEL_COMMANDBUTTON_9:
    j=9;
    localgain=2.5;
    imodule=1;
    ichannel=19;
    igain=1;
    break;
case PANEL_COMMANDBUTTON_10:
    j=8;
    localgain=200.0;
    imodule=1;
    ichannel=18;
    igain=1;
    break;
case PANEL_COMMANDBUTTON_11:
    j=16;
    break;
case PANEL_COMMANDBUTTON_12:
    j=17;
    break;
case PANEL_COMMANDBUTTON_13:
    j=15;
    localgain=10.0;
    imodule=3;
    ichannel=7;
    igain=20;
    break;
case PANEL_COMMANDBUTTON_14:
    j=14;
    localgain=10.0;
    imodule=3;
    ichannel=6;
    igain=20;
    break;
case PANEL_COMMANDBUTTON_15:
    j=13;
    localgain=10.0;
    imodule=3;
    ichannel=5;
    igain=20;
    break;
case PANEL_COMMANDBUTTON_16:
    j=12;
    localgain=10.0;
    imodule=3;
    ichannel=4;
    igain=20;
    break;
case PANEL_COMMANDBUTTON_17:
    j=11;
    localgain=10.0;
    imodule=3;
    ichannel=3;
    igain=5;
    break;
case PANEL_COMMANDBUTTON_18:
    j=10;
    localgain=10.0;
    imodule=3;
    ichannel=2;
    igain=20;
    break;
}

```

```

        if(j<16)
        {
            for(i=0;i<120000;i++) chanBuf[i]=dataBuf[i+j*120000];

            err = SCXI_Scale (1, imodule, ichannel, igain, 1, 1, 0, -1,
120000,
                                chanBuf, voltBuf);

            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }
            if(ichannel>13 && ichannel<18) for(i=0;i<120000;i++)
voltBuf[i]=(voltBuf[i] - currentoff[ichannel-14])*localgain;
            else for(i=0;i<120000;i++) voltBuf[i]=voltBuf[i]*localgain;
        }
        else
        {
            if(j==16) for(i=0;i<120000;i++) voltBuf[i]=loadpower[i];
            else for(i=0;i<120000;i++) voltBuf[i]=motorpower[i];
        }

        DeleteGraphPlot (panelHandle, PANEL_GRAPHTIME, -1, 1);
        err = PlotY (panelHandle, PANEL_GRAPHTIME, voltBuf, 1001,
            VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID,
            i, VAL_WHITE);
        err = Mean (voltBuf, 120000, &meanval);
        for(i=0;i<120000;i++) voltBuf[i]-=meanval;
        dt = 1.0 / 120000.0;

        for(i=0;i<16384;i++) spectra[i]=0.0;

        for(j=0;j<13;j++)
        {
            for(i=0;i<16384;i++) spectratime[i]=voltBuf[i+j*8192];
            GetCtrlVal (panelHandle, PANEL_LISTBOX, &i);
            err = ScaledWindow(spectratime, 16384, i, &windowConstant);
            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }
            err = AutoPowerSpectrum (spectratime, 16384, dt,
spectratime, &df);
            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }
            err = SpectrumUnitConversion (spectratime, 8192, 0, 1, 6,
df,
windowConstant, spectratime, unitString);
            if (err<0) {
                SetCtrlVal (panelHandle, PANEL_error, err);
                SetCtrlVal (panelHandle, PANEL_alarm, 1);
            }
            for(i=0;i<16384;i++) spectra[i]=(spectra[i]* (double) j +
spectratime[i]) / (double) (j+1);
        }

        DeleteGraphPlot (panelHandle, PANEL_GRAPHFREQ, -1, 1);
        err = PlotY (panelHandle, PANEL_GRAPHFREQ, spectra, 8192,
            VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID,
            i, VAL_RED);

```

```

        break;
    }
    return 0;
}

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0) /* Needed if linking in external compiler;
harmless otherwise */
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "ai_acq.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    PutTimeStamp();
    Beep();
    CollectData(0);
    CollectData(2);
    CollectData(1);
    Beep();
    RunUserInterface ();
    return 0;
}

int CVICALLBACK AcquireData (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_TIMER_TICK:

            Beep();

            PutTimeStamp();

            CollectData(2);

            CollectData(1);

            CheckSteadyState();

            Beep();

            break;

    }
    return 0;
}

```

APPENDIX C

LISTING OF FUZZY LOGIC INFERENCE ENGINE SOFTWARE

```
class MF {
public:
    MF(); // default constructor
    ~MF(); // default destructor
    CString label; // label of fuzzy variable
    double degree; // store degree of membership or truncation
    MF *next; /* pointer to next MF */
    CFuzzySet FuzzySet;
};

class FuzzyRule {
public:
    FuzzyRule(); // default constructor
    ~FuzzyRule(); // default destructor
    MF *ifside; // if side of the rule
    MF *thenside; // then side of the rule
    FuzzyRule *next; // pointer to next rule
};

class Data {
public:
    Data(); // default constructor
    ~Data(); // default destructor
    CString label; // label of data variable
    double dat; // the real data
    Data *next; // pointer to next data
    Data *Header; // header of data list
};

// CFuzzySt.h
// Header file for CFuzzySt classes
// Version .90b last updated Oct 29, 1996
// (C) copyright Timothy Lee 1996
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define TRIANGLE 1
#define S_CURVE_UP 2
#define PI_CURVE 3
#define BETA_CURVE 4
#define LEFT_SHOULDER 5
#define RIGHT_SHOULDER 6
#define S_CURVE_DOWN 7
#define COMBO_SET 8
#define NO_SHAPE 0
#define CENTROID 1
#define COMPOSITE_MAX 2
#include<afxtempl.h>
#include<math.h>

class AFX_EXT_CLASS CFuzzySet: public CObject
{
friend class MF;
friend class FuzzyRule;
protected:
    void equals(const CFuzzySet& set2);
//functions
    int m_shape;
    BOOL setScale();
    CFuzzySet();//empty constructor for serialization
    DECLARE_SERIAL(CFuzzySet);
    void initMemVector(double initValue=-1, int num_elements=2001);
```

```

        int fToInt(double X);
        BOOL updateRegion(CArray<CPoint,CPoint>& ptArray);

//Attributes
        CString m_name;
        CString m_description;
        CArray<double,double> domain;
        double m_alphaCut;
        CArray<float,float> memVector;
        double m_scale;//x axis scale
        BOOL drawn;
        CRect currentRect;
        CRgn* region; //needs to be made embedded member
        int nearestElement(double num);
        void interpolateVector();
        void setDomain(double low, double high);

public:
        void setAlphaCut(double alphaValue, CString type="STRONG");
        BOOL ptInSet(CPoint pt);
        void draw(CRect rect,CDC* pDC);
        ~CFuzzySet();
#ifdef _DEBUG
        void Dump(CDumpContext& dc) const;
#endif
        CString getName() {return m_name;}
        void clear();
        void setName(CString string) {m_name=string; return;}
        float maxTruth();
        double defuzzify(CString how="CENTROID");
        CFuzzySet (const CFuzzySet& set2);
        CArray<float,float>& getMemVector();
        int getShape();
        void AssertValid() const;
        double findTruth(double x);
        double getAlphaCut() {return m_alphaCut;}
        CArray<double,double>& getDomain(void);
        virtual void Serialize(CArchive& ar);
        const CFuzzySet& operator=(CFuzzySet& set2);
        friend __declspec(dllimport) CFuzzySet operator&&(CFuzzySet& set1,CFuzzySet&
set2);
        friend __declspec(dllimport) CFuzzySet operator||(CFuzzySet& set1,CFuzzySet&
set2);
        friend __declspec(dllimport) CFuzzySet operator*(const CFuzzySet& set1,const
double num);
        friend __declspec(dllimport) CFuzzySet operator*(const double num, const
CFuzzySet& set1);
        friend __declspec(dllimport) CFuzzySet pow(const CFuzzySet& set,const double num);
        friend __declspec(dllimport) CFuzzySet pow(const double num, const CFuzzySet&
set);

        CFuzzySet CFuzzySet::truncate(const double maxValue);
        CFuzzySet not();//cannot use ! operator because that must return boolean
        CFuzzySet(CString name);
};//end CFuzzySet class declaration

////////////////////////////////////
//CFuzzyTriangle class declaration

class AFX_EXT_CLASS CFuzzyTriangle: public CFuzzySet
{
public:
        const CFuzzyTriangle& operator=(const CFuzzySet& set2);
        CFuzzyTriangle(double base1, double tip, double base2,  CString name="", double
height=1) ;

```

```

}; // end CFuzzyTriangle class declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CFuzzyBeta class declaration

class AFX_EXT_CLASS CFuzzyBeta : public CFuzzySet
{
public:
    const CFuzzyBeta& operator=(const CFuzzySet& set2);
    CFuzzyBeta(double center, double flexDist, CString name="");

}; //end CFuzzyBeta class declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CFuzzySCurve class declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class AFX_EXT_CLASS CFuzzySCurve : public CFuzzySet
{
public:
    const CFuzzySCurve& operator=(const CFuzzySet& set2);
    CFuzzySCurve(double bottom, double top, CString name="");

}; //end CFuzzySCurve class declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CFuzzyPiCurve class declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

class AFX_EXT_CLASS CFuzzyPiCurve : public CFuzzySet
{
public:
    const CFuzzyPiCurve& operator=(const CFuzzySet& set2);
    CFuzzyPiCurve(double left, double right, CString name="");

}; //end CFuzzyPiCurve class declaration

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CFuzzyShoulder class declaration
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

class AFX_EXT_CLASS CFuzzyShoulder : public CFuzzySet
{
public:
    const CFuzzyShoulder& operator=(const CFuzzySet& set2);
    CFuzzyShoulder(double bottom, double top, double topLength, CString name="",
double height=1);
    CFuzzyShoulder(double bottom, double top, CString name="", double height=1);
    // operator CFuzzySet();
    //conversion done implicitly through CFuzzySet copy constructor
}; //end CFuzzyShoulder class declaration

// Fuzzy Test lView.cpp : implementation of the CFuzzyTestlView class
//

#include "stdafx.h"
#include "Fuzzy Test l.h"

#include "Fuzzy Test lDoc.h"
#include "Fuzzy Test lView.h"

#include <iostream.h>
#include <fstream.h>
#include <ctype.h>
#include "utkne.h"

// default constructor
Data::Data()
{
    label="NULL";

```

```

        dat = 0.0;
        next = NULL;
        Header = NULL;
};

// default destructor
Data::~Data()
{
    /*
    Data *p;
    while(Header)
    {
        p = Header->next;
        delete Header;
        Header = next;
    }
    */
};

// default constructor
MF::MF()
{
    degree = 0;
    label = "NULL";
    next = NULL;
    CFuzzySet FuzzySet("NULL");
};

// default destructor
MF::~MF()
{
    MF *p,*q;
    p = next;
    do
    {
        q = next;
        delete p->next;
        delete p;
        p = q;
    } while (p!=NULL);
};

// default constructor
FuzzyRule::FuzzyRule()
{
    ifside = new MF;
    thenside = new MF;
    next = NULL;
};

// default destructor
FuzzyRule::~FuzzyRule()
{
    FuzzyRule *p, *q;
    delete ifside;
    delete thenside;
    p = next;
    do
    {
        q = next;
        delete p->next;
        delete p;
    }
};

```



```

        p = q;
    } while (p!=NULL);
};

FuzzyRule *UTFuzzyRule, *HeaderRule;
MF *UTMF, *HeaderMF;
MF *Header_if, *Header_then;

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CFuzzyTest1View

IMPLEMENT_DYNCREATE(CFuzzyTest1View, CView)

BEGIN_MESSAGE_MAP(CFuzzyTest1View, CView)
    //{{AFX_MSG_MAP(CFuzzyTest1View)
    // NOTE - the ClassWizard will add and remove mapping macros here.
    //      DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

////////////////////////////////////
// CFuzzyTest1View construction/destruction

CFuzzyTest1View::CFuzzyTest1View()
{
    // TODO: add construction code here
}

CFuzzyTest1View::~CFuzzyTest1View()
{
}

BOOL CFuzzyTest1View::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}

////////////////////////////////////
// Function to read word by word from specified file

CString ReadWord(ifstream& InputStream)
{
    char InputChar;
    CString myString="";
    do
    {
        InputChar=InputStream.get();
        InputChar=toupper(InputChar);
        if(InputChar<33) break;
        myString += InputChar;
    }while(!InputStream.eof());
    return myString;
}

```

```

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Fuzzy Rules Construction by reading the system from files

void ConstructFuzzySystem(CString MFFileName, CString RBFileName, CString OutFileName)
{
    CString myString;
    CString nameString1, nameString2;
    double point1, point2, point3;

    ifstream InFile(MFFileName, ios::in);

    ofstream OutFile(OutFileName, ios::out);

    UTMF = new MF;
    HeaderMF = new MF;
    HeaderMF = UTMF;

    do {

        myString=ReadWord(InFile);
        if(myString!="")
        {
            nameString1=myString;

            if(UTMF->label!="NULL")
            {
                MF *templ;
                templ= new MF;
                UTMF->next = templ;
                UTMF = UTMF->next;
            }

            myString=ReadWord(InFile);
            nameString2=myString;

            myString=ReadWord(InFile);
            if(myString=="SHOULDER")
            {
                InFile >> point1;
                InFile >> point2;
                UTMF->label=nameString1;
                CFuzzyShoulder tempmf(point1,point2,nameString2);
                UTMF->degree = 0.0;
                UTMF->FuzzySet = tempmf;
            }
            else
            {
                if(myString=="TRIANGLE")
                {
                    InFile >> point1;
                    InFile >> point2;
                    InFile >> point3;
                    UTMF->label=nameString1;
                    CFuzzyTriangle
tempmf(point1,point2,point3,nameString2);
                    UTMF->FuzzySet = tempmf;
                }
                else
                {
                    OutFile << "Error in membership functions!" << endl;
                }
            }
        }
    }
}

```

```

} while(!InFile.eof());

InFile.close();

InFile.open(RBFileName, ios::in);

UTFuzzyRule = new FuzzyRule;
HeaderRule = new FuzzyRule;
HeaderRule = UTFuzzyRule;

Header_if = new MF;
Header_then = new MF;

Header_if = UTFuzzyRule->ifside;
Header_then = UTFuzzyRule->thenside;

int we_are_in_if;

do {

    myString=ReadWord(InFile);
    if(myString!="")
    {
        if(myString=="IF")
        {

            UTFuzzyRule->ifside=Header_if;
            UTFuzzyRule->thenside=Header_then;

            if(UTFuzzyRule->ifside->label!="NULL")
            {
                FuzzyRule *templ;
                templ= new FuzzyRule;
                UTFuzzyRule->next = templ;
                UTFuzzyRule = UTFuzzyRule->next;
                Header_if = UTFuzzyRule->ifside;
                Header_then = UTFuzzyRule->thenside;
            }

            myString=ReadWord(InFile);
            nameString1=myString;

            myString=ReadWord(InFile);

            if(myString!="IS")
                OutFile << "Error in rule base!" << endl;

            myString=ReadWord(InFile);
            nameString2=myString;

            UTMF=HeaderMF;
            do
            {
                if(UTMF->label==nameString1 && UTMF-
>FuzzySet.getName()==nameString2)
                {

                    if(UTFuzzyRule->ifside->label!="NULL")
                    {
                        MF *templ;
                        templ= new MF;
                        UTFuzzyRule->ifside->next = templ;
                        UTFuzzyRule->ifside = UTFuzzyRule-
>ifside->next;
                    }
                }
            }

```

```

UTFuzzyRule->ifside->label=UTMF->label;
UTFuzzyRule->ifside->FuzzySet=UTMF-
>FuzzySet;

UTFuzzyRule->ifside->next=NULL;
break;
}
UTMF=UTMF->next;
} while(UTMF!=NULL);
we_are_in_if = 1;
}
else
{
if(myString=="AND" && we_are_in_if==1)
{
myString=ReadWord(InFile);
nameString1=myString;

myString=ReadWord(InFile);

if(myString!="IS")
    OutFile << "Error in rule base!" << endl;

myString=ReadWord(InFile);
nameString2=myString;

UTMF=HeaderMF;
do
{
    if(UTMF->label==nameString1 && UTMF-
        {
            if(UTFuzzyRule->ifside-
                {
                    MF *templ;
                    templ= new MF;
                    UTFuzzyRule->ifside->next =
                        UTFuzzyRule->ifside =
                }
                UTFuzzyRule->ifside->label=UTMF-
                    UTFuzzyRule->ifside->FuzzySet=UTMF-
                        UTFuzzyRule->ifside->next=NULL;
                        break;
                    }
                    UTMF=UTMF->next;
                } while(UTMF!=NULL);
            }
            else
            {
                if(myString=="THEN")
                {
                    myString=ReadWord(InFile);
                    nameString1=myString;

                    myString=ReadWord(InFile);

                    if(myString!="IS")
                        OutFile << "Error in rule base!" <<

endl;

                    myString=ReadWord(InFile);

```

```

nameString2=myString;

UTMF=HeaderMF;
do
{
    if(UTMF->label==nameString1 && UTMF-
        {
            if(UTFuzzyRule->thenside-
                {
                    MF *templ;
                    templ= new MF;
                    UTFuzzyRule->thenside-
                        UTFuzzyRule->thenside
                }
            UTFuzzyRule->thenside-
                UTFuzzyRule->thenside-
                    UTFuzzyRule->thenside-
                        break;
                    }
                UTMF=UTMF->next;
            } while(UTMF!=NULL);
            we_are_in_if=0;
        }
    else
    {
        if(myString=="AND" && we_are_in_if==0)
        {
            myString=ReadWord(InFile);
            nameString1=myString;

            myString=ReadWord(InFile);

            if(myString!="IS")
                OutFile << "Error in rule

            myString=ReadWord(InFile);
            nameString2=myString;

            UTMF=HeaderMF;
            do
            {
                if(UTMF->label==nameString1
                    {
                        if(UTFuzzyRule-
                            {
                                MF *templ;
                                templ= new MF;
                                UTFuzzyRule-
                                    UTFuzzyRule-
                            }
                        UTFuzzyRule->thenside-
                            UTFuzzyRule->thenside-
                                >FuzzySet.getName()==nameString2)
                                >thenside->label!="NULL")
                                >thenside->next = templ;
                                >thenside = UTFuzzyRule->thenside->next;
                                >label=UTMF->label;
                                >FuzzySet=UTMF->FuzzySet;
                                base!" << endl;
                                && UTMF->FuzzySet.getName()==nameString2)
                                >thenside->label!="NULL")
                                >thenside->next = templ;
                                >thenside = UTFuzzyRule->thenside->next;
                                >label=UTMF->label;
                                >FuzzySet=UTMF->FuzzySet;

```

```

        UTFuzzyRule->thenside-
>next=NULL;
        break;
    }
    UTMF=UTMF->next;
} while(UTMF!=NULL);
}
}
}
} while(!InFile.eof());

UTFuzzyRule->ifside=Header_if;
UTFuzzyRule->thenside=Header_then;

UTFuzzyRule = HeaderRule;

OutFile.close();
}

////////////////////////////////////
// Fuzzy Rules Evaluation

void EvaluateFuzzyRules(CString OutFileName, Data *inputdata)
{
    double truncation;

    // Create sample data to test the inference engine

    MF *Output, *HeaderOutput;

    Output = new MF;
    HeaderOutput = new MF;
    HeaderOutput = Output;

    UTFuzzyRule = HeaderRule;

    ofstream OutFile(OutFileName, ios::app);

    do
    {
        truncation = 1.0;

        Header_if = UTFuzzyRule->ifside;
        Header_then=UTFuzzyRule->thenside;

        do
        {
            double templ;

            do
            {
                if(UTFuzzyRule->ifside->label == (inputdata->label) )
                {
                    templ=UTFuzzyRule->ifside-
>FuzzySet.findTruth(inputdata->dat);
                    break;
                }
                inputdata=inputdata->next;
            }while(inputdata->next!=NULL);
        }
    }
}

```

```

        inputdata=inputdata->Header;

        if(templ<truncation) truncation = templ;
        UTFuzzyRule->ifside=UTFuzzyRule->ifside->next;
    } while(UTFuzzyRule->ifside!=NULL);

do
{
    CFuzzySet Solution("Solve");
    int foundOutput = 0;
    Output = HeaderOutput;

    do
    {
        if(Output->label==UTFuzzyRule->thenside->label)
        {
            Solution = UTFuzzyRule->thenside->FuzzySet;
            Solution = Solution.truncate(truncation);
            foundOutput=1;
            break;
        }
        Output = Output->next;
    } while(Output!=NULL);

    if(foundOutput==0)
    {
        Output = HeaderOutput;
        if(Output->next!=NULL)
        {
            do
            {
                Output = Output->next;
            } while(Output->next!=NULL);
        }
        if(Output->label!="NULL")
        {
            MF *templ;
            templ = new MF;
            Output->next=templ;
            Output=Output->next;
        }
        Output->label=UTFuzzyRule->thenside->label;
        Output->FuzzySet= UTFuzzyRule->thenside-
>FuzzySet.truncate(truncation);
    }
    else
    {
        Output->FuzzySet = Output->FuzzySet || Solution;
    }

    UTFuzzyRule->thenside = UTFuzzyRule->thenside->next;

} while(UTFuzzyRule->thenside!=NULL);

UTFuzzyRule->ifside=Header_if;
UTFuzzyRule->thenside=Header_then;

UTFuzzyRule = UTFuzzyRule -> next;

} while(UTFuzzyRule!=NULL);

Output = HeaderOutput;

do

```

```

    {
        OutFile << Output->label << "=";
        OutFile << Output->FuzzySet.defuzzify( "centroid" ) << " ";
        Output = Output->next;
    } while(Output!=NULL);

    OutFile << endl;

    delete Output;

    OutFile.close();

    /*
    WinExec("command.com",SW_SHOW); Use this for the wavelet filtering stuff...
*/
}

//////////////////////////////////////
// CFuzzyTest1View drawing

void CFuzzyTest1View::OnDraw(CDC* pDC)
{
    CFuzzyTest1Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // TODO: add draw code for native data here
/*
    CRect rect;
    GetClientRect(&rect);

    CFuzzyTriangle tri(0,10,20);
    tri.draw(rect,pDC);
*/

    ConstructFuzzySystem("c:\\ase\\utk.mf", "c:\\ase\\utk.rb", "c:\\ase\\fuzzy.out");

    Data *data, *temp;

    data = new Data;
    temp = new Data;

    data->Header = data;
    data->label="CH1";
    data->dat=40.0;
    temp->Header=data->Header;
    temp->label="CH2";
    temp->dat=160.0;
    data->next=temp;
    data=data->next;
    temp = new Data ;
    temp->Header=data->Header;
    temp->label="CH3";
    temp->dat=160.0;
    data->next=temp;
    data=data->Header;

    EvaluateFuzzyRules("c:\\ase\\fuzzy.out",data);
    EvaluateFuzzyRules("c:\\ase\\fuzzy.out",data);

}

//////////////////////////////////////
// CFuzzyTest1View printing

BOOL CFuzzyTest1View::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation

```



```

        return DoPreparePrinting(pInfo);
    }

void CFuzzyTest1View::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CFuzzyTest1View::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

////////////////////////////////////
// CFuzzyTest1View diagnostics

#ifdef _DEBUG
void CFuzzyTest1View::AssertValid() const
{
    CView::AssertValid();
}

void CFuzzyTest1View::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CFuzzyTest1Doc* CFuzzyTest1View::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFuzzyTest1Doc)));
    return (CFuzzyTest1Doc*)m_pDocument;
}
#endif // _DEBUG

////////////////////////////////////
// CFuzzyTest1View message handlers

```

APPENDIX D

LISTING OF BACKPROPAGATION ARTIFICIAL NEURAL NETWORK SOFTWARE

```
/*
Name      : Ali S. Erbay
Course    : ECE571 Pattern Recognition
Project   # 1.2
Subject   : Implementation of Fast Back Propagation Algorithm
OS        : SunOS 4.1.2
Language  : C
*/

#include <stdio.h>
#include <math.h>

/*
weight = weight vector[layer][i][j]
weightchange = change in weight [layer][i][j]
neuron = Output of PE [layer][i]
errors = Error of Pe [layer][i]
input_data = input vector [number_of_input]
output_data = desired output vector [number_of_output]
scale_max = Maximum of I/O data
scale_min = Minimum of I/O data
learning_coeff = learning coefficient
momentum = momentum term
*/

float weight[2][200][200],weightchange[2][200][200],neuron[3][200],errors[2][200];
float input_data[200],output_data[200];
float learning_coeff,momentum;
float scale_max,scale_min;

int number_of_input,number_of_hidden,number_of_output;
int total_number_of_iteations;

/*
Initialize weights
*/

void initialize_weights()
{
int i,j,k;
for(j=1;j<=number_of_hidden;j++) for(i=0;i<=number_of_input;i++)
{
weight[0][i][j]= (((float) random()) - ((float) pow(2.0,30.0)))/((float)
pow(2.0,31.0))*0.2;
weightchange[0][i][j]=0.0;
}
for(j=1;j<=number_of_output;j++) for(i=0;i<=number_of_hidden;i++)
{
weight[1][i][j]= (((float) random()) - ((float) pow(2.0,30.0)))/((float)
pow(2.0,31.0))*0.2;
weightchange[1][i][j]=0.0;
}
}

/*
Load weights from file for recall
*/

void load_weights(weightfile)
FILE *weightfile;
{
int i,j;
for(j=1;j<=number_of_hidden;j++) for(i=0;i<=number_of_input;i++) fscanf(weightfile,"%e
",&weight[0][i][j]);
}
```

```

    for(j=1;j<=number_of_output;j++) for(i=0;i<=number_of_hidden;i++) fscanf(weightfile,"%e
",&weight[1][i][j]);
}

/*
  Save weights after training
*/

void save_weights(weightfile)
FILE *weightfile;
{
    int i,j;
    for(j=1;j<=number_of_hidden;j++) for(i=0;i<=number_of_input;i++) fprintf(weightfile,"%e
",weight[0][i][j]);
    for(j=1;j<=number_of_output;j++) for(i=0;i<=number_of_hidden;i++) fprintf(weightfile,"%e
",weight[1][i][j]);
}

/*
  Sigmoid function
*/

float sigmoid(z)
float z;
{
    return(1.0/(1.0+((float)exp(-z))));
}

/*
  Read input & desired output vector and present it to the first [=input] layer

  All output are scaled in this subroutine
*/

present_input(datafile,irecall)
FILE *datafile;
int irecall;
{
    int i;
    if(!feof(datafile))
    {
        if(irecall) exit(0); else rewind(datafile);
    }
    for(i=1;i<=number_of_input;i++)
    {
        fscanf(datafile,"%f",&input_data[i]);
        if(!feof(datafile))
        {
            if(irecall) exit(0);
            rewind(datafile);
            i=1;
            fscanf(datafile,"%f",&input_data[i]);
        }
        neuron[0][i]=input_data[i];
    }
    if(!irecall)
    {
        for(i=1;i<=number_of_output;i++)
        {
            fscanf(datafile,"%f",&output_data[i]);
            output_data[i]=0.6/(scale_max-scale_min)*(output_data[i]-scale_min)+0.2;
        }
    }
}

/*
  Propagate the input until the last [=output] layer
*/

void propagate_input()
{
    int i,j;
    float sum;
    for(j=1;j<=number_of_hidden;j++)
    {

```

```

    sum=0.0;
    for(i=0;i<=number_of_input;i++) sum+=weight[0][i][j]*neuron[0][i];
    neuron[1][j]=sigmoid(sum);
}
for(j=1;j<=number_of_output;j++)
{
    sum=0.0;
    for(i=0;i<=number_of_hidden;i++) sum+=weight[1][i][j]*neuron[1][i];
    neuron[2][j]=sigmoid(sum);
}
}

/*
  Compute errors in each layer
*/

void compute_errors()
{
    int i,j;
    float sum;
    for(j=1;j<=number_of_output;j++) errors[1][j] = neuron[2][j] * (1.0 - neuron[2][j]) * (
output_data[j] - neuron[2][j] );
    for(j=1;j<=number_of_hidden;j++)
    {
        sum=0.0;
        for(i=1;i<=number_of_output;i++) sum+=errors[1][i]*weight[1][j][i];
        errors[0][j]=neuron[1][j] * (1.0 - neuron[1][j]) * sum;
    }
}

/*
  Adjust the weights
*/

void adjust_weights()
{
    int i,j;
    for(i=0;i<=number_of_hidden;i++)
    {
        for(j=1;j<=number_of_output;j++)
        {
            weightchange[1][i][j]=learning_coef*errors[1][j]*neuron[1][i] +
momentum*weightchange[1][i][j];
            weight[1][i][j]+=weightchange[1][i][j];
        }
    }
    for(i=0;i<=number_of_input;i++)
    {
        for(j=1;j<=number_of_hidden;j++)
        {
            weightchange[0][i][j]=learning_coef*errors[0][j]*neuron[0][i] +
momentum*weightchange[0][i][j];
            weight[0][i][j]+=weightchange[0][i][j];
        }
    }
}

/*
  Main program
*/

main(argc,argv)
int argc;
char *argv[];
{
    FILE *datafile,*structfile,*weightfile;
    int i,j,k;

/*
  Input filename of Back Propagation ANN structure and I/O
*/

    if((structfile=fopen(argv[1],"r"))==0)
    {
        printf("File access error for %s\n",argv[1]);
        exit(0);
    }
}

```

```

}

/*
Input filename of data
*/

if((datafile=fopen(argv[2],"r")==0)
{
printf("File access error for %s\n",argv[2]);
exit(0);
}

/*
Input filename of weight vectors if this is a recall
*/

if(argc>3)
{
if((weightfile=fopen(argv[3],"r")==0)
{
printf("File access error for %s\n",argv[3]);
exit(0);
}
}

/*
Otherwise create a file to save the weights of the ANN
*/

else
{
if((weightfile=fopen("weight.bp","w+")==0)
{
printf("File access error for weight.bp\n");
exit(0);
}
}

/*
Read ANN parameters
*/

fscanf(structfile,"%d",&number_of_input);
fscanf(structfile,"%d",&number_of_hidden);
fscanf(structfile,"%d",&number_of_output);
fscanf(structfile,"%f",&learning_coef);
fscanf(structfile,"%f",&momentum);
fscanf(structfile,"%f %f",&scale_max,&scale_min);
fscanf(structfile,"%d",&total_number_of_iterations);

/*
Set bias
*/

neuron[0][0]=neuron[1][0]=1.0;

/*
Initialize or load the weights
*/

if(argc>3) load_weights(weightfile); else initialize_weights();

/*
Training or recall begins,
If this is a recall, then an EOF will terminate the program
*/

for(k=0;k<total_number_of_iterations;k++)
{

/*
Present input to input layer
*/

present_input(datafile,(argc-3));

```

```

/*
Propagate to the output layer
*/

propagate_input();

/*
If this is a recall print out the output layer in de-scaled form
*/

if((argc-3)==1)
{
for(i=1;i<=number_of_output;i++) printf("%f ",(scale_max-scale_min)/0.6*(neuron[2][i]-
0.2)+scale_min);
printf("\n");
}

/*
Otherwise back-propagate and train
*/

else
{
compute_errors();
adjust_weights();
}

/*
At the end of training, save the weights
*/

save_weights(weightfile);
}

```

VITA

Ali Seyfettin Erbay was born in Bruchsal, Germany on July 19, 1967. He finished Schönbornschule and Stirumschule primary schools in Germany and came to Türkiye in 1978. He finished Beylerbeyi Primary School and attended Beylerbeyi Middle School for one year. Thereafter, he moved to Izmir with his family, where he finished Eşrefpaşa Middle School and Atatürk High School. In 1985, he entered the Nuclear Engineering Department of Hacettepe University in Ankara, Türkiye. He received his Bachelor of Science degree in Nuclear Engineering in June 1990. In September 1990, he entered the Master of Science program of the Computer Science and Engineering Department at Hacettepe University, where he also worked as a research assistant until December 1991. In January 1992, he entered the Nuclear Engineering Department of The University of Tennessee, and in December 1994, he received his Master of Science degree in Nuclear Engineering. He worked with Professor Belle R. Upadhyaya on many projects related to signal validation, digital signal processing, artificial intelligence, expert systems, non-destructive testing, electrical motors and predictive and preventive maintenance issues. His research work has been documented in various publications in professional journals and national and international conferences. He graduated in May 1999 with a doctoral degree in Nuclear Engineering from The University of Tennessee, Knoxville. He has been also selected in “Who’s Who in Science and Engineering.”