Early Fault Detection of Spacecraft Actuator System using Machine Learning: Case Study – Reaction Wheels

BITS ZG628T: Dissertation

by

GOPIKA DINESH

2015HT12063

Dissertation work carried out at

ISRO Satellite Centre, Bengaluru

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

November 2017

Early Fault Detection of Spacecraft Actuator System using Machine Learning: Case Study – Reaction Wheels

BITS ZG628T: Dissertation

by

GOPIKA DINESH

2015HT12063

Dissertation work carried out at

ISRO Satellite Centre, Bengaluru

Submitted in partial fulfillment of M.Tech. Software Systems degree programme

Under the Supervision of
PARTHA BANDYOPADHYAY
ISRO Satellite Centre, Bengaluru

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

November, 2017

CERTIFICATE

This is to certify that the Dissertation entitled **Early Fault Detection of Spacecraft Actuation System using Machine Learning: Case Study – Reaction Wheels** and submitted by **Gopika Dinesh** having ID-No. **2015HT12063** for the partial fulfillment of the requirements of M.Tech. **Software Systems** degree of BITS, embodies the bonafide work done by her under my supervision.

[illegible]

Signature of the Supervisor

Place : BANGALORE

Date : 20/09/2017

PARTHA BANDYOPADHYAY

Name, Designation & Organization &Location

Birla Institute of Technology & Science, Pilani

Work-Integrated Learning Programmes Division

First Semester 2017-2018

BITS ZG628T: Dissertation

ABSTRACT

BITS ID No.	**:** 2015HT12063
NAME OF THE STUDENT	**:** GOPIKA DINESH
EMAIL ADDRESS	**:** 2015ht12063@wilp.bits-pilani.ac.in
STUDENT'S EMPLOYING ORGANIZATION & LOCATION	**:** ISRO Satellite Centre, Bengaluru
SUPERVISOR'S NAME	**:** PARTHA BANDYOPADHYAY
SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION	**:** ISRO Satellite Centre, Bengaluru
SUPERVISOR'S EMAIL ADDRESS:	parthab@isac.gov.in

DISSERTATION TITLE **:** Early Fault Detection of Spacecraft Actuation System using Machine Learning: Case Study – Reaction Wheels

ABSTRACT : (Should be neatly word processed; should not exceed one page)

Early detection of faults in satellite systems improves its safety and reliability by preventing serious damage to attitude and orbit control system by initiating the recovery mechanisms in advance. The goal of this work is to develop an early fault detection model for the spacecraft actuator system using Machine Learning techniques. Artificial Neural Network is used for learning the nonlinear dynamics of actuator system. Reaction Wheels (actuators) are considered in this case study. The four reaction wheels are modeled individually and as a combined system and evaluated. The neural networks are developed to model the nominal behavior of reaction wheel in normal mode and switchover conditions. They are further trained for early detection of failures in the above cases. The training and testing data for the neural networks are obtained from the historic telemetry data of on-orbit satellites stored in ground station.

Broad Academic Area of Work: Artificial Intelligence

Key words : Artificial Neural Network, Machine Learning, Multilayer Perceptron, Reaction Wheels

Gopika	Partha Bandyopadhyay
Signature of the Student	**Signature of the Supervisor**
Name: GOPIKA DINESH	**Name:** PARTHA BANDYOPADHYAY
Date: 20/9/2017	**Date:** 20/09/2017
Place: Bangalore	**Place:** Bangalore

ACKNOWLEDGEMENT

I express my sincere gratitude to my Mentor Mr. Jagannath Das, Scientist/Engineer-G and my Supervisor Mr. Partha Bandyopadhyay, Scientist/Engineer-SG for giving me constant support and guidance throughout the project.

I thank Mr. A Baskar, BITS Faculty, for providing feedback and advice while realizing this project.

I also thank my colleague Mr. Sourajith Debnath and my family for the help rendered and support which was instrumental in carrying out this project.

Table of Contents

1. INTRODUCTION ... 1
2. ATTITUDE AND ORBIT CONTROL SYSTEM (AOCS) ... 3
   2.1 Onboard Computer: ... 3
3. REACTION WHEELS ... 5
   3.1 Fundamental Block Diagram: ... 5
   3.2 Friction Compensation in WDE: ... 6
   3.3 Dynamic Friction Compensation: ... 6
   3.4 DFC Gain Switch in Hysterisis Speed Logic: ... 7
   3.5 Wheel Modes of Operation: ... 8
4. FEATURE SET ... 9
   4.1 Telemetry Data: ... 9
   4.2 Logic for generating feature set: ... 11
5. MODELING THE WHEEL USING ARTIFICIAL NEURAL NETWORKS ... 16
   5.1 Implementation details: ... 17
6. MLPREGRESSOR ... 18
   6.1 Scikit Learn: ... 18
   6.2 MLPRegressor Parameters: ... 18
   6.2.1 Loss Function: ... 18
   6.2.2 Weight Optimization Algorithm: ... 19
   6.2.2.1 ADAM ... 19
   6.2.2.2 LBFGS ... 20
   6.2.3 Mini batches: ... 20
   6.2.4 Activation function for the hidden layer: ... 21
   6.2.5 Early Stopping: ... 21
   6.2.6 Score: ... 21
7. IMPLEMENTATION ... 22
   7.1 Approach-1 ... 22
   7.1.1 Wheel 1 ... 24
   7.1.1.1 MLPRegressor Parameters: ... 24
   7.1.1.2 Training Results: ... 25
   7.1.1.3 Loss curve: ... 25
   7.1.1.4 Validation scores: ... 25
   7.1.1.5 Network weights: ... 26
   7.1.1.6 Biases: ... 26
   7.1.1.7 Testing Results: ... 26
   7.1.2 Wheel 2 ... 27
   7.1.2.1 MLPRegressor Parameters: ... 27
   7.1.2.2 Training Results: ... 28
   7.1.2.3 Loss curve: ... 28
   7.1.2.4 Validation scores: ... 28
   7.1.2.5 Network weights: ... 29
   7.1.2.6 Biases: ... 29
   7.1.2.7 Testing Results: ... 29
   7.1.3 Wheel 3 ... 30
   7.1.3.1 MLPRegressor Parameters: ... 30

7.1.3.2 Training Results: 30
7.1.3.3 Loss curve: 31
7.1.3.4 Validation scores: 31
7.1.3.5 Network weights: 31
7.1.3.6 Biases: 32
7.1.3.7 Testing Results: 32
7.1.4 Wheel 4 33
7.1.4.1 MLPRegressor Parameters: 33
7.1.4.2 Training Results: 33
7.1.4.3 Loss curve: 34
7.1.4.4 Validation scores: 34
7.1.4.5 Network weights: 34
7.1.4.6 Biases: 35
7.1.4.7 Testing Results: 35
7.2 Approach-2 36
7.2.1 Wheel model during Normal mode without maneuvering 39
7.2.1.1 MLPRegressor Parameters: 40
7.2.1.2 Training Results: 40
7.2.1.3 Loss curve: 40
7.2.1.4 Validation scores: 41
7.2.1.5 Network weights: 41
7.2.1.6 Biases: 41
7.2.1.7 Testing Results: 42
7.2.2 Wheel model during Normal mode with maneuvering 43
7.2.2.1 MLPRegressor Parameters: 45
7.2.2.2 Training Results: 45
7.2.2.3 Network weights: 46
7.2.2.4 Biases: 46
7.2.2.5 Testing Results: 46
8. FEASIBILITY STUDY FOR ONBOARD IMPLEMENTATION 47
8.1 Ground Development of Reaction Wheel Model: 47
8.2 Onboard implementation of wheel model 47
9. SUMMARY 50
10. BIBLIOGRAPHY 52
11. REFERENCES 53
12. APPENDIX 54
13. CHECKLIST 55

List Of Figures

Figure 1: Block Diagram of Active Control 3
Figure 2: Onboard Computer (OBC) 4
Figure 3: Ideal Reaction Wheel System 5
Figure 4: DFC Gain Switch in Hysterisis Speed Logic 7
Figure 5: Previous sample computation 13
Figure 6: A node in the hidden layer of individual wheel NN 23
Figure 7: Loss Curve of Wheel 1 NN 25
Figure 8: Validation score curve of Wheel 1 NN 25
Figure 9: Loss Curve of Wheel 2 NN 28
Figure 10: Validation score curve of Wheel 2 NN 28
Figure 11: Loss Curve of Wheel 3 NN 31
Figure 12: Validation score curve of Wheel 3 NN 31
Figure 13: Loss curve of Wheel 4 NN 34
Figure 14: Validation score curve of Wheel 4 NN 34
Figure 15: A node in the hidden layer of all wheel NN 38
Figure 16: Loss Curve of all wheels NN 40
Figure 17: Validation score curve of all wheel NN 41
Figure 18: Model inputs at time t 48
Figure 19: Model inputs at time t+1 49

List Of Tables

Table 1: Telemetry Details (Satellite_XYZ)....................10
Table 2: Telemetry Details (Satellite_PQR)....................43
Table 3: List Of Abbreviations....................54

1. INTRODUCTION

Fault Diagnosis plays a vital role in improving the safety, reliability and availability of satellite systems. The complexity of satellites is increasing rapidly day by day as space missions are becomingly more sophisticated. This emphasizes the need for more dependable systems with minimal anomalies. Due to the ground testing limitations and the complexity of space environment, onboard failures occur quite often in a satellite. The main aim of fault detection and isolation is to effectively detect a fault and accurately isolate the failing component in the shortest time possible. Thus the Fault Detection, Isolation and Recovery logics is the major component of a satellite's Onboard Software.

Early fault detection is the key to improve reliability and safety of the satellite. Early detection of faults helps to initiate recovery mechanism in advance and hence prevent serious damage to the Attitude and Orbit Control System (AOCS). The sooner the failures are detected and failed components are repaired or replaced, higher will be the success rate of the mission. Early fault detection requires the satellite to be more autonomous and intelligent.

Machine Learning can significantly enhance the satellite capabilities, robustness and efficiency.[1] Machine learning algorithms for anomaly detection and fault diagnosis helps in early detection of failures. These algorithms can make predictions that can guide better decisions and smart actions in real time. This helps to enhance the reliability, autonomy and duration of space missions.

The attitude control subsystem stabilizes the spacecraft and orients it in the desired direction using actuators which could be a combination of momentum wheels, reaction wheels, control moment gyros, thrusters or magnetic torquers. The FDIR logics implemented in the onboard software overcomes the catastrophic effects of actuator failure. However these are executed after the failure has occurred, isolating the failed component and initiating the subsequent recovery mechanisms. At times this transition can cause huge disturbance in the satellite system affecting the AOCS. This is where machine learning of actuators for early fault detection comes into the picture. The Neural Network models of the actuators are trained to predict the behavior of actuator system few cycles in advance. This, in case of failure, provides early indication and thereby helps to initiate the isolation and recovery steps in advance. This ensures a smooth change over for the spacecraft and helps to prevent serious damage to the AOCS and the complex scientific payloads.

Machine Learning methods utilize large amount of past telemetry data stored in ground stations as training data to acquire the system behavior models. The training data helps to capture the rules, models and patterns of the spacecraft sub system. It can also be used to update the parameter values in the expert's diagnostic model. These models are further validated with unseen test data to access the model accuracy before they are implemented in spacecraft's onboard computer. The models here learn from data which represent 'past experiences'. This does not require complete and accurate expert knowledge in advance. The learned system helps the system to perform better.

2. ATTITUDE AND ORBIT CONTROL SYSTEM (AOCS)

The orientation of spacecraft is very important whether it is for telecommunication, remote sensing or astronomical missions. The spacecraft experiences various disturbances like Atmospheric Drag, Solar Radiation Torque, Gravity Gradient Torque, Magnetic Torque, Internal Torque etc. The attitude control subsystem stabilizes the spacecraft and orients it in the desired pointing direction (attitude) as it proceeds along the orbital path using actuators which could be a combination of momentum wheels, reaction wheels, control moment gyros, thrusters or magnetic torquers. A set of sensors like gyroscopes, star sensors, magnetometers, sun sensors, earth sensors etc measure the satellite's pointing direction and feeds the data continuously to the Onboard Computer. The sensor outputs are compared with the reference data to generate the control inputs for the actuator. The control inputs to the actuators are continuously adjusted in order to minimize the control error.

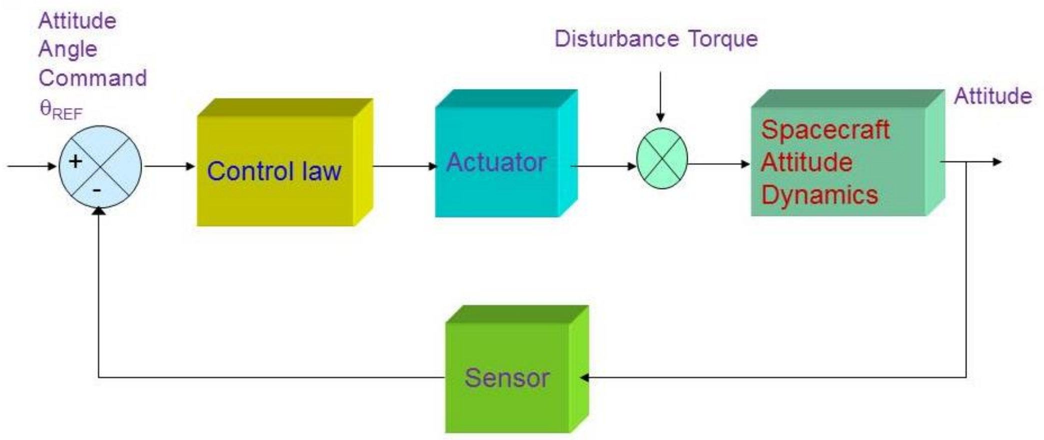

Attitude Angle Command θ_{REF}
+
-
Control law
Actuator
Disturbance Torque
Spacecraft Attitude Dynamics
Attitude
Sensor

Figure 1: Block Diagram of Active Control

2.1 Onboard Computer:

An onboard computer (OBC) is the brain of any satellite which monitors and controls each and every action of the satellite. OBC is a processor based system built around MAR31750 microprocessor. An OBC is designed to monitor and control the operations of a satellite. The software in the OBC consists of modules related to attitude control, command decoding and distribution, telemetry monitoring, data storage and playback, auto temperature control and Safety Logics. The software is developed using Ada language. The software is configured as time synchronous round-robin type of scheduler.

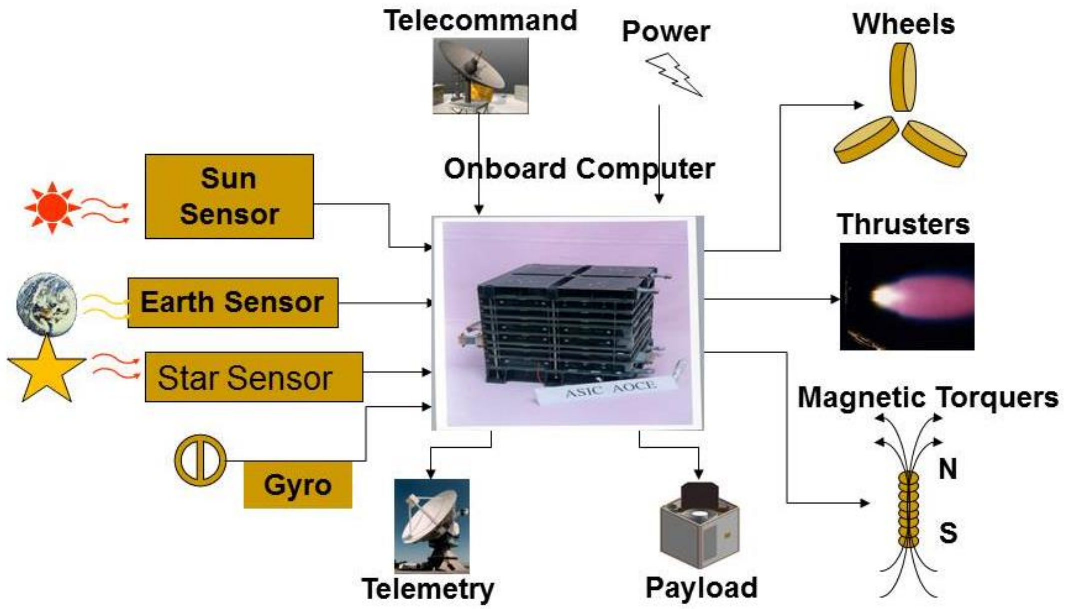


Figure 2: Onboard Computer (OBC)

3. REACTION WHEELS

As a case study, early fault detection of Reaction Wheels is studied in this project.

Reaction wheels are momentum exchange devices which provide reaction torque to spacecraft and store angular momentum. Reaction wheel consists of rotating flywheel, typically suspended by ball bearings driven by brushless DC motor using control signals from the Wheel Drive Electronics (WDE). The driver circuit in WDE delivers the commuted current proportional to the input Torque Command Signal (TCS).

Typically all remote sensing space crafts are configured as zero momentum system with four RWs in tetrahedral configuration arrangement in order to cater the required high agile body rotations. The three-axis stabilization system using reaction wheels provide high pointing accuracy, low jitter and long-term attitude stability.

3.1 Fundamental Block Diagram:

The fundamental block diagram of an ideal reaction wheel system is shown in Figure1. The input TCS is converted to current by the WDE which is fed to the BLDC motor which generates a torque (Tm). The friction torque (Tf) is the only loss which is subtracted from the motor torque and the net torque will accelerate or decelerate the flywheel.

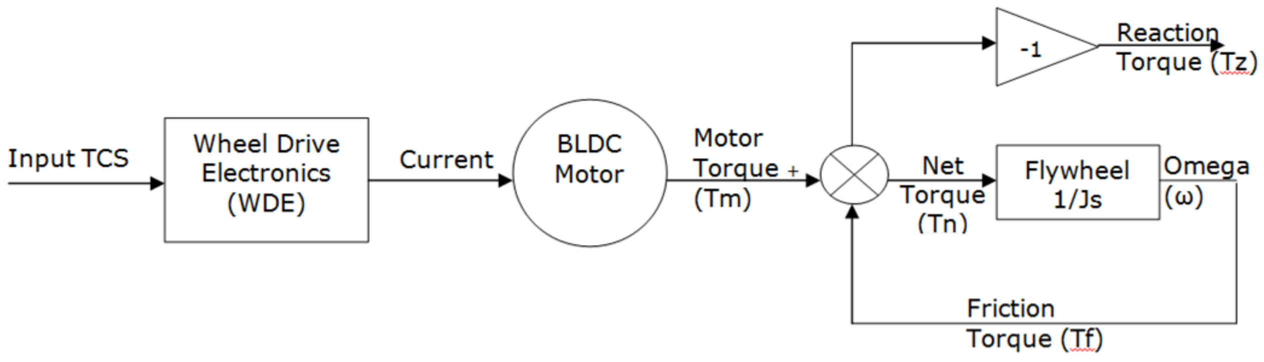


Figure 3: Ideal Reaction Wheel System

3.2 Friction Compensation in WDE:

Friction compensation in WDE is implemented considering that the friction torque (current) from the bearing can be modeled using the formulae

$$Y = mx + c \text{ -------------- } (1)$$

Here x is speed and the constants m and c is arrived from the Wheel level bearing torque characterization.

3.3 Dynamic Friction Compensation:

DFC estimates the friction torque real time and compensates it by pumping additional motor current. The friction torque or loss torque estimated by DFC is the uncompensated friction torque of the friction compensation in WDE. The loss torque estimation is done from the residue from the expected speed change and actual speed change over a pre-defined sampling time. This loss torque is added to Torque Control signal (TCS). DFC is having TCS and speed as inputs.

The TCS is integrated over a fixed time (DFC update period) to get the expected change in momentum.

$$\boldsymbol{\nabla Lexp} = \boldsymbol{L2} - \boldsymbol{L1} = \int_{t0}^{t1} \tau \, dt \quad \text{-------------- } (2)$$

Where $\nabla Lexp$ is expected change in momentum, $t0$ & $t1$ is the start and end of DFC update period, $\tau\ is\ TCS$.

The wheel speed is sampled at the beginning and end of DFC update period and the change in speed is used to compute the actual change in momentum (over DFC update period).

$$\boldsymbol{\nabla Lact} = (\boldsymbol{\omega 2} - \boldsymbol{\omega 1}) \times \boldsymbol{J} \text{ -------------- } (3)$$

Where $\boldsymbol{\nabla Lact}$ is the actual change in momentum, $\omega 2$ and $\omega 1$ are speeds at the end and beginning of DFC update period respectively and J is the inertia of flywheel kgm^2.

The difference between expected change in momentum and actual change in momentum is the loss in momentum as a result of friction torque.

Loss in momentum = Expected change in momentum – Actual change in momentum

= $\nabla \boldsymbol{Lexp}$ - $\nabla \boldsymbol{Lact}$ -------------- (4)

This loss in momentum multiplied by the time corresponding to DFC update period gives the loss torque.

DFC Wheel Torque = Loss in momentum * DFC Time Interval Selected ------- (5)

DFC Wheel Torque is passed through a Low Pass Filter (LPF). The output of LPF is saturated to 0.01Nm and then added to TCS.

DFC Update Interval and DFC Gain (used for conversion of Loss Momentum to Compensation Torque) are selected automatically from the High Speed / Low Speed whenever the wheel speed is above or below the threshold. The Speed threshold is chosen same as the speed used for changing the measurement clock frequency.

During this threshold crossing, the Low Pass Filter Coefficients are also switched between High Speed and Low Speed Set. In additions, there are 2 selectable sets for the LPF in both High Speed and Low Speed set.

3.4 DFC Gain Switch in Hysterisis Speed Logic:

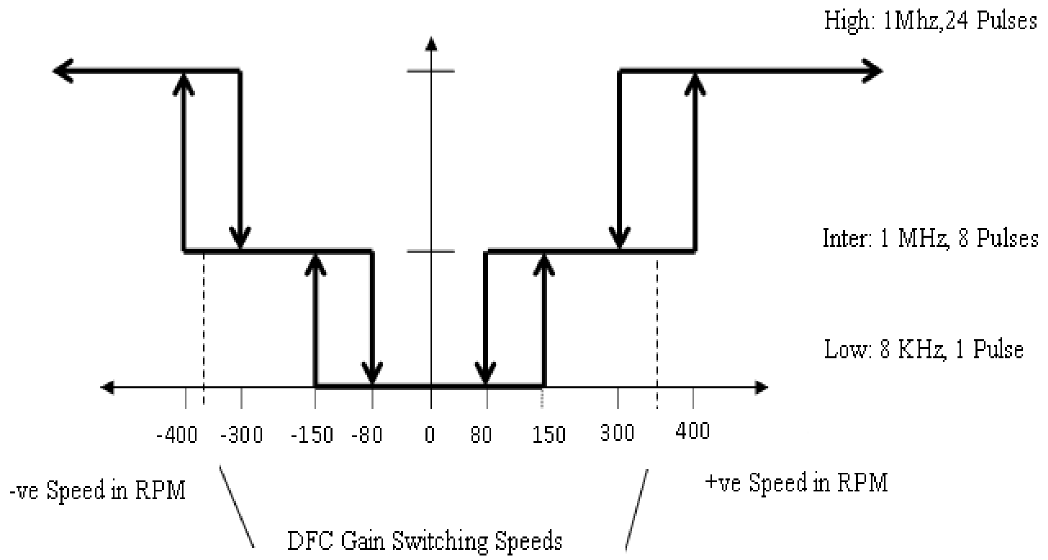


Figure 4: DFC Gain Switch in Hysterisis Speed Logic

```
IF (ABS(Wheel Speed(I))>=Speed DFC Change) THEN

        DFC Timer Limit Select = DFC Timer High
        DFC Time Inverse Select = DFC Time Inverse High
        DFC LPF Gain Select = DFC LPF Gain High

ELSE

        DFC Timer Limit Select = DFC Timer Low
        DFC Time Inverse Select = DFC Time Inverse Low
        DFC LPF Gain Select = DFC LPF Gain Low

END IF
```

3.5 Wheel Modes of Operation:

There are two modes of operation for wheels – Torque Output Control (TOC) mode and Nominal Speed Control (NSC) mode.

In NSC mode, full torque is applied to drive all wheels towards nominal speed. Thrusters are the control actuators in NSC mode. The wheels are having a fixed wheel speed which is the commanded wheel speed. For zero momentum biased condition, same speed is applied to all four wheels taking care of polarity.

Wheels are the control actuators in TOC mode. In this mode, due to uncompensated friction, extra torque has to be applied to bring it to nominal speed. Here variable speed is applied based on body error.

4. FEATURE SET

The most important step in any machine learning problem is defining the feature set. The features in the feature set should be measurable properties of the phenomena being observed.

The feature set for developing the reaction wheel model consists of Wheel Input Torque (TCS), Wheel Speed, DFC Torque. Here the input set consists of previous samples of wheel speed, input torque, DFC torque and the output set contains the current sample of wheel speed, input torque, DFC torque. Let N be the number of previous samples required for early prediction of wheel behavior. Therefore for each entry in the output training data set, there are corresponding N entries corresponding to N previous samples. The model predicts the current wheel speed based on previous samples of wheel speed, current wheel torque based on previous samples of input torque and current DFC torque from previous samples of wheel speed, input torque and DFC torque. The DFC torque is computed based on wheel speed and input torque as explained in previous chapter.

The variation in DFC torque gives a clear indication of the health of the wheel. This is useful for detecting failures.

In this project, the data for training the wheel is obtained from historic telemetry data stored in ground stations. The telemetry data of *Satellite_XYZ* which had an on-orbit wheel failure is chosen for training the model.

4.1 Telemetry Data:

Telemetry system must reliably and transparently convey measurement information from remotely located data generating sources to users located in space or earth. Typically, data generators are scientific sensors, science housekeeping sensors, engineering sensors and other subsystem on-board a spacecraft. Telemetry function consists of data acquisition system and data formatting system. The OBC processor stores the acquired telemetry data in telemetry RAM. The telemetry formatting is software based and hence more flexible. It creates the telemetry frame to be transmitted by adding the frame sync, sub-frame id etc. and dumps the data to telemetry interface hardware periodically. Function of the telemetry software is to acquire the data, format the data and dump the data to the telemetry hardware at regular intervals.

The Onboard Computer of the Spacecraft organizes the parameters to be telemetered to ground station in different predefined slots. The data is arranged in channels and frames. Every master frame has 32 sub frames (Frame No: 0 to 31). Each sub frame in a master frame contains 256 8-bit channels. These 8192 channel master frame comprise the entirety of the telemetry for all its purposes.

The details of the parameters chosen as the feature set is as follows:

Table 1: Telemetry Details (Satellite_XYZ)

Sl No	Parameter	Sampling Rate	Channel No	Frame ID	Range	Resolution	Type
1	Wheel 1 Input Torque	4s	94, 95	5, 13, 21, 29	±0.05 Nm	1.525E-06 Nm	16 bit 2 words
2	Wheel 2 Input Torque	4s	96, 97	5, 13, 21, 29	±0.05 Nm	1.525E-06 Nm	16 bit 2 words
3	Wheel 3 Input Torque	4s	98, 99	5, 13, 21, 29	±0.05 Nm	1.525E-06 Nm	16 bit 2 words
4	Wheel 4 Input Torque	4s	100, 101	5, 13, 21, 29	±0.05 Nm	1.525E-06 Nm	16 bit 2 words
5	Wheel 1 speed	1s	32, 33	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30	±5000 rpm	1.8311E-01 rpm	16 bit 2 words
6	Wheel 2 speed	1s	34, 35	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30	±5000 rpm	1.8311E-01 rpm	16 bit 2 words
7	Wheel 3 speed	1s	36, 37	0, 2, 4, 6, 8,	±5000 rpm	1.8311E-01 rpm	16 bit 2 words

				10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30			
8	Wheel 4 speed	1s	38, 39	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30	±5000 rpm	1.8311E-01 rpm	16 bit 2 words
9	DFC Torque Wheel 1	16s	150, 151	17	±0.01Nm	3.05185e-7 Nm	16 bit 2 words
10	DFC Torque Wheel 2	16s	152, 153	17	±0.01Nm	3.05185e-7 Nm	16 bit 2 words
11	DFC Torque Wheel 3	16s	154, 155	17	±0.01Nm	3.05185e-7 Nm	16 bit 2 words
12	DFC Torque Wheel 4	16s	156, 157	17	±0.01Nm	3.05185e-7 Nm	16 bit 2 words

4.2 Logic for generating feature set:

For training the model, the data has to be divided into two– input set (X) and output set (Y). If 'N' is the number of previous samples required to make prediction, then for each entry in Y, there will be N entries in X. **Therefore the size of the feature set with N previous samples will be N times the size of the feature set with 1 previous sample for prediction.** Here the input set consists of previous samples of wheel speed, input torque, dfc torque which are required for predicting each entry in the output set containing wheel speed, input torque, dfc torque.

As seen above, it is not possible to get wheel speed, input torque and DFC torque at the same instant of time. This is because the parameters are available in different frames due to their different sampling rates. Therefore the training sets X and Y are formed as follows:

Assumptions made:

1. Since the frame to frame time gap is 0.5 seconds, the variation of parameters during this time frame is considered to be negligible.
2. Instead of interpolating data to obtain at the same instant of time, samples of the different parameters from nearby frames are considered assuming their variation to be minimal.
3. The logic derived can be applied for all wheels as the data is available in the same frame for all four wheels (only channels are different).

DFC Torque data is available only in Frame 17 (16 sec sampling), Wheel Input Torque is available in Frames 5, 13, 21, 29 (4 sec sampling) and Wheel Speed data is available in frames 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 (1 sec sampling).

The output training set Y (ie current sample) be formed by DFC Torque data (@Frame 17), Wheel Input Torque data (@Frame 21) and Wheel Speed (@Frame 18).

	DFC Torque	**Wheel Input Torque**	**Wheel Speed**
Current data	**F17**	**F21**	**F18**

Wheel speed and torque are available in nearby sub frames like (12,13) , (4,5), (28,29). Since DFC Torque is available only once in a master frame, the current DFC sample is itself considered as the previous sample when the wheel speed and torque frames are closer to frame 17 (i.e. frames 13, 12, 5, 4). The previous DFC sample is considered when the speed and torque data at frames 28 and 29 respectively are chosen.

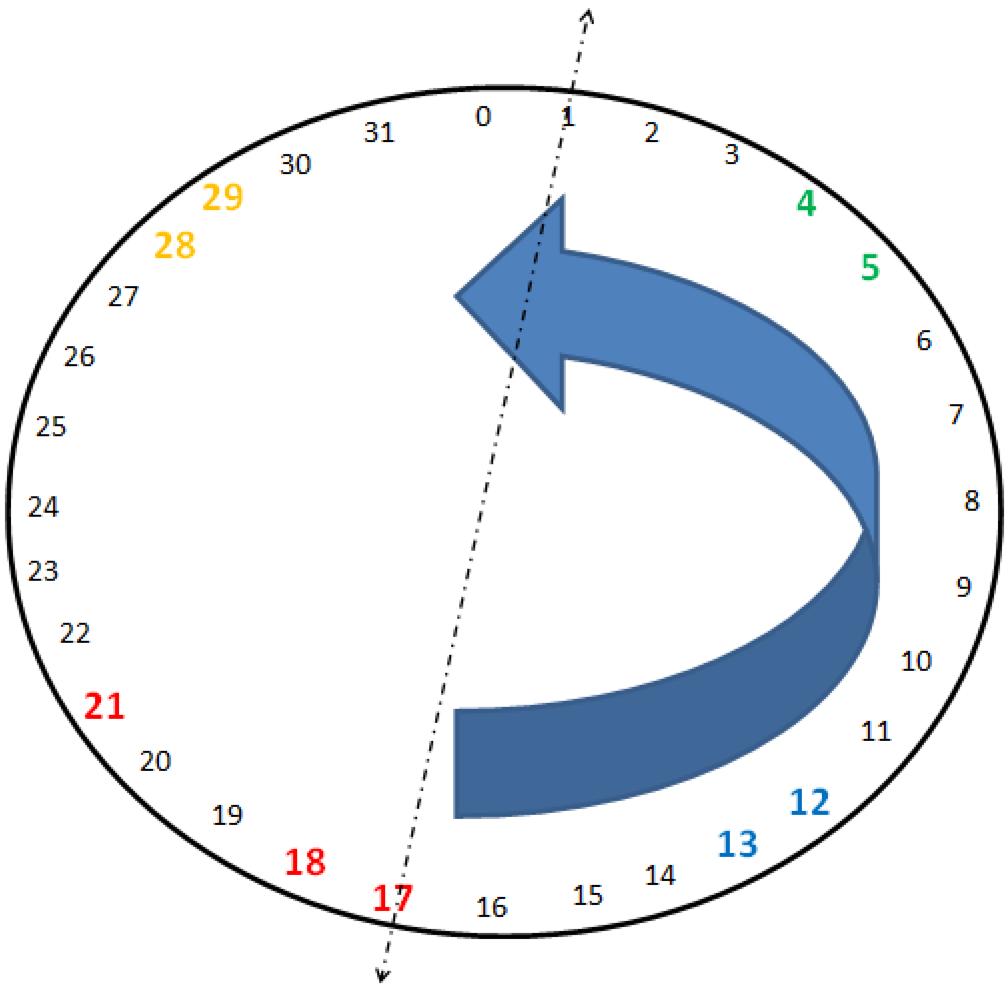


Figure 5: Previous sample computation

If the number of samples required for prediction is N=1, then X consists of DFC Torque data (@Frame 17), Wheel Input Torque (@Frame 13), Wheel Speed (@Frame 12).).

	DFC Torque	Wheel Input Torque	Wheel Speed
Current data	**F17**	**F21**	**F18**
Previous sample1	**F17**	**F13**	**F12**

If the number of samples required for prediction is N=2, then X consists of DFC Torque data (@Frame 17), Wheel Input Torque (@Frame 13), Wheel Speed (@Frame 12) as first set of samples and DFC Torque data (@Frame 17), Wheel Input Torque (@Frame 5), Wheel Speed (@Frame 4) as second set of samples.

	DFC Torque	Wheel Input Torque	Wheel Speed
Current data	**F17**	**F21**	**F18**
Previous sample1	**F17**	**F13**	**F12**
Previous sample2	**F17**	**F5**	**F4**

Similarly when N=3,

	DFC Torque	Wheel Input Torque	Wheel Speed
Current data	**F17**	**F21**	**F18**
Previous sample1	**F17**	**F13**	**F12**
Previous sample2	**F17**	**F5**	**F4**
Previous sample3	**F17′**	**F29**	**F28**

Where F17′ is the previous DFC torque sample.

When N=4,

	DFC Torque	Wheel Input Torque	Wheel Speed
Current data	**F17**	**F21**	**F18**
Previous sample1	**F17**	**F13**	**F12**
Previous sample2	**F17**	**F5**	**F4**
Previous sample3	**F17′**	**F29**	**F28**
Previous sample4	**F17′**	**F13′**	**F12′**

Where F17′, F13′, F12′ are from previous master frame.

When N=5,

	DFC Torque	Wheel Input Torque	Wheel Speed
Current data	**F17**	**F21**	**F18**
Previous sample1	**F17**	**F13**	**F12**
Previous sample2	**F17**	**F5**	**F4**
Previous sample3	**F17′**	**F29**	**F28**
Previous sample4	**F17′**	**F13′**	**F12′**
Previous sample5	**F17′**	**F5′**	**F4′**

Where F17′, F13′, F12′, F5′, F4′ are from previous master frame.

When N=6,

	DFC Torque	Wheel Input Torque	Wheel Speed
Current data	**F17**	**F21**	**F18**
Previous sample1	**F17**	**F13**	**F12**
Previous sample2	**F17**	**F5**	**F4**
Previous sample3	**F17′**	**F29**	**F28**
Previous sample4	**F17′**	**F13′**	**F12′**
Previous sample5	**F17′**	**F5′**	**F4′**
Previous sample6	**F17′′**	**F29′**	**F28′**

Where F17′, F13′, F12′, F5′, F4′, F29′, F28′ are from previous master frame. F17′′ is from previous to previous master frame.

And so on.

Wheel speed has a range of ±5000 rpm, Wheel input torque has a range of ±0.05 Nm, DFC Torque has a range of ±0.01Nm. Therefore these parameters are normalized to ±1 range before training the model.

5. MODELING THE WHEEL USING ARTIFICIAL NEURAL NETWORKS

The scope of this dissertation is to develop the model of the spacecraft actuator (*reaction wheel*) that is capable of early prediction of the actuator behaviour using supervised machine learning approach.

Machine learning (ML) is an application of Artificial Intelligence (AI). It is the study of algorithms that provides systems the ability to automatically learn from examples and experiences instead of hard coded rules. There are numerous machine learning approaches. Decision Tree Learning, Artificial Neural Networks, Deep Learning, Support Vector Machines, Bayesian Networks are some of them.

Out of the different ML approaches, Artificial Neural Network (ANN) is chosen. ANNs are one of the commonly applied machine learning algorithms. ANN can be used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables. It is a computational structure based on the biological neural network. ANN can be viewed as directed graphs in which artificial neurons are nodes and directed edges with weights are the connections between the neurons.

ANNs are mainly of two topologies. Feed Forward networks and Recurrent (feedback). Learning process in an ANN can be viewed as the problem of updating network architecture and connection weights so that network can effectively perform a given task. There are three main learning paradigms: supervised learning, unsupervised learning, reinforcement learning. In supervised learning, the network is provided with a correct answer for every input pattern. Unsupervised Learning explores the underlying structure in the data or correlation between patterns in the data and organizes patterns into categories from this correlation. Reinforcement learning allows machines to determine the ideal behavior within a specific context to maximize performance.

The most common family of feed forward network, called the Multi-Layer Perceptron MLP) is the architecture chosen for modeling the Reaction Wheel. MLP uses supervised learning. The wheel model is derived from the training set containing input vectors and the corresponding output vectors. MLP has at least 3 layers of nodes (one input layer, one output layer, one or more hidden layers).

MLP is ideal for modeling the reaction wheel because it can distinguish data that is not linearly separable, it can estimate complex pattern mappings, it can transform arbitrary regions from one pattern space to another and it can learn non-linear models.

5.1Implementation details:

This project is implemented using

IDE	:	Anaconda Spyder 3.1.4
Language	:	Python 3.6.1
Libraries	:	scikit-learn

6. MLPREGRESSOR

6.1 Scikit Learn:

Scikit learn is a free machine learning library in Python programming language. It features various regression, clustering and classification algorithms. It is built on NumPy, SciPy, matplotlib. Multi-Layer Perceptron (MLP) is a supervised learning algorithm that learns a function that is the relationship between input and output by training on a dataset. The multilayer perceptron is an artificial neural network structure and is a nonparametric estimator that can be used for classification and regression.

Scikit Learn provides two neural network models for supervised learning based on MLP: MLPClassifier and MLPRegressor. MLPClassifier classifies the input data into different classes. It supports multi class classification. MLPRegressor outputs a set of continuous values. It also supports multi output regression where a sample can have more than one target.

The problem statement in this dissertation involves developing a wheel model for predicting the wheel behavior. The wheel exhibits nonlinear characteristics. Therefore a nonlinear regression model is required for modeling. Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model. Scikit Learn's MLPRegressor[4] module is the multi-layer feedback neural network model chosen for training the wheel model.

6.2 MLPRegressor Parameters:

MLPRegressor can approximate nonlinear functions of the input. This implementation works with data represented as dense and sparse numpy arrays of floating point values. MLPRegressor trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. Learning occurs by changing connection weights after each piece of data is processed based on error between output and expected result. Learning carried out through back propogation. Features of MLPRegressor are as follows:

6.2.1 Loss Function:

Scikit Learn's MLPRegressor uses Squared Error as the Loss Function. It can also have a regularization term (alpha) added to the loss function to prevent overfitting by shrinking the model parameters. L2 is the sum of squares of the weights and is computationally efficient. Alpha can penalize weights with large magnitudes. This in turn helps the model to predict better.

$$Loss(\hat{y}, y, W) = \frac{1}{2}||\hat{y} - y||_2^2 + \frac{\alpha}{2}||W||_2^2$$

Where $\hat{y}$ is the predicted value, y is the true value, ɑ is L2 regularization term that penalizes complex models. ɑ is non negative parameter that controls the magnitude of the penalty.

Starting from initial random weights, MLP aims at minimizing the loss function by repeatedly updating these weights. After computing the loss, a backward pass propagates it from the output layer to the previous layers, providing each weight parameter with an update value meant to decrease the loss.

6.2.2 Weight Optimization Algorithm:

Scikit learn supports the following solvers for weight optimization ***(solver)***: Stochastic Gradient Descent (sgd), Adaptive Moment Estimation (Adam) (a variation of gradient descent for large datasets), Limited Memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm(for small datasets).

6.2.2.1 ADAM

Gradient Descent is one of the most popular methods to optimize neural networks. Adam[3] is a gradient descent optimization algorithm that is suited for large datasets. It works well with sparse gradients and non-stationary settings. This method computes adaptive learning rates for each parameter. It stores the exponentially moving averages of gradient (M_0) and its squares (R_0).

The updates will be proportional to $\frac{\text{Average gradient}}{\sqrt{\textit{Average squared gradient}}}$

Stochastic update (gradient) for subset of training examples is given by

$$L(W) = \frac{1}{b}\sum_{j=1}^{b} l(W\,; x_{ij}, y_{ij}) + \alpha r(W)$$

where

$\{(x_{ij}, y_{ij})\}^{b}_{j=1}$	: random mini batch chosen at iteration t. x_{ij} are the training instances and y_{ij} are the corresponding labels
W	: network parameters to learn
$l(W; x_{ij}, y_{ij})$	: loss of network parameterized by W wrt (x_{ij}, y_{ij})
r(W)	: regularization function $\|\|W\|\|^2$
α	: regularization weight ***(alpha)*** (typical choice : 0.0001)

Algorithm:

```
$M_0 = 0, R_0 = 0$ (Initialization)

For t = 1. . . T:
    $M_t = \beta_1 M_{t-1} + (1 - \beta_1)\nabla L_t(W_{t-1})$ (1st moment estimate)
    $R_t = \beta_2 R_{t-1} + (1 - \beta_2)\nabla L_t(W_{t-1})^2$ (2nd moment estimate)
    $\widehat{M}_t = M_t / (1 - (\beta_1)^t)$ (1st moment bias correction)
    $\widehat{R}_t = R_t / (1 - (\beta_2)^t)$ (2nd moment bias correction)
    $W_t = W_{t-1} - \eta \frac{\widehat{M}t}{\sqrt{\widehat{R}t + \varepsilon}}$ (Update)
Return $W_T$
```

Hyper-parameters:

$\eta > 0$	– constant learning rate (typical choice: 0.001) ***(learning_rate_init & learning_rate)***
$\beta_1 \in [0, 1)$	– 1st moment decay rate (typical choice: 0.9) ***(beta_1)***
$\beta_2 \in [0, 1)$	– 2nd moment decay rate (typical choice: 0.999) ***(beta_2)***
$\varepsilon > 0$	– numerical term (typical choice: 10^{-8}) ***(epsilon)***

The algorithm stops when it reaches a preset maximum number of iterations (***max_iter)***; or when the improvement in loss is below a certain, small number ***(tol)***.

6.2.2.2 LBFGS

Limited-memory BFGS approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm using a limited amount of computer memory. For small datasets lbfgs can converge faster and perform better. Quasi Newton methods approximate the Hessian from past gradients. L-BFGS[5] like BFGS, uses an estimation to the inverse Hessian matrix to search through variable space. Instead of the inverse Hessian Hk, L-BFGS maintains a history of the past m updates of the position x and gradient ∇ f(x), where generally the history size m can be small (often m<10).

6.2.3 Mini batches:

Gradient descent carried out in mini batches is a tradeoff between computing the gradient of whole datasets and computing the gradient of each training example to perform one update. The size of mini-batches ***(batch_size)*** ranges between 50 and 256. Samples can be shuffled ***(shuffle)*** in each iteration. Random mini batches are chosen for each iteration. The seed of the random state generator ***(random_state)*** determines the batches chosen.

L-BFGS solver does not use mini batches.

6.2.4 Activation function for the hidden layer:

Multilayer perceptron can have one or more hidden layer. The number of neurons in the hidden layer ***(hidden_layer_sizes)*** should be suitable selected inorder to avoid overfitting or underfitting.

Scikit learn provides the following activation functions for neurons ***(activation)***: identity function, logistic sigmoid function, hyperbolic tan function and rectified linear unit function. Hyperbolic tangent which output in the range [-1,1] and exhibits gives nonlinear response. The neurons in the wheel neural network will have hyperbolic tangent activation function.

6.2.5 Early Stopping:

Early stopping ***(early_stopping)*** is another technique to avoid overfitting. It prevents poor generalization while training a model by gradient descent optimization. In order to halt the optimizer, the dataset is divided into training dataset and a small validation dataset ***(validation_fraction)*** to get an estimate of the validation process. The validation score is computed and the training is terminated when the validation score is not improving within the tolerance defined for two consecutive epochs.

6.2.6 Score:

The model is evaluated based on the score. The score of MLPRegressor returns the coefficient of determination R^2 of the prediction. The coefficient R^2 is defined as (1 - u/v), where u is the regression sum of squares ((y_true - y_pred) ** 2).sum() and v is the residual sum of squares ((y_true - y_true.mean()) ** 2).sum(). Best possible score is 1.0. A constant model that gives same output disregarding input gives a score of 0. A bad model has a negative score.

7. IMPLEMENTATION

The telemetry data of *Satellite_XYZ* launched on 25th February 2013 is used for training the wheel model. One of the reaction wheels (Wheel 1) of this satellite had failed. This happened on 6th October 2014. The wheel configuration had switched from 4RW to 3RW.

The data for training and testing the model was chosen during the time the wheels exhibited nominal behavior and there was no maneuvering. During this time, the satellite was in Normal mode (3 axis SKF), Wheel Mode Flag was 'TOC' where wheels are the control actuators, DFC was enabled and Orbit Maneuver Sequencer status= 'Over'.

The parameters of the four wheels were extracted from telemetry data and processed for the durations given below:

Training data: 28th April to 14th May 2013

Testing data set 1: 14th to 31st May 2013

Testing data set 2: 07th to 12th March 2013

Testing data set 3: 28th to 29th April 2013

Testing data set 4: 04th to 05th October 2014 (data just before wheel failure on 06th October 2014).

7.1 Approach-1

In the first approach the models for the four wheels are developed separately going by their theoretical assumption of being independent.

Let 'N' be the number of previous samples required to predict the current sample.

Training Data:

Inputs: x_1 = Wheel Input Torque (previous sample 1)
x_2 = Wheel Speed (previous sample 1)
x_3 = DFC torque (previous sample 1)

x_4 = Wheel Input Torque (previous sample 2)
x_5 = Wheel speed (previous sample 2)
x_6 = DFC Torque (previous sample 2)

"
"

x_{3N-2} = Wheel Input Torque (previous sample N)
x_{3N-1} = Wheel Speed (previous sample N)
x_{3N} = DFC Torque (previous sample N)

Outputs: y_1 = Wheel Input Torque (current sample)
y_2 = Wheel Speed (current sample)
y_3 = DFC Torque (current sample)

Number of neurons in input layer = 3N

where N is the number of previous samples required for prediction.

Number of neurons in the output layer = 3

Number of neurons in hidden layer = h (to be determined)

Total number of neurons = 3N * h* **3**

The number of hidden layers in the multilayer perceptron is kept at 1.

The number of neurons in the hidden layer = 2*n +1 where n is the number of input neurons. [Hecht-Nielsen (1987)][2]

Consider any one neuron in the hidden layer.

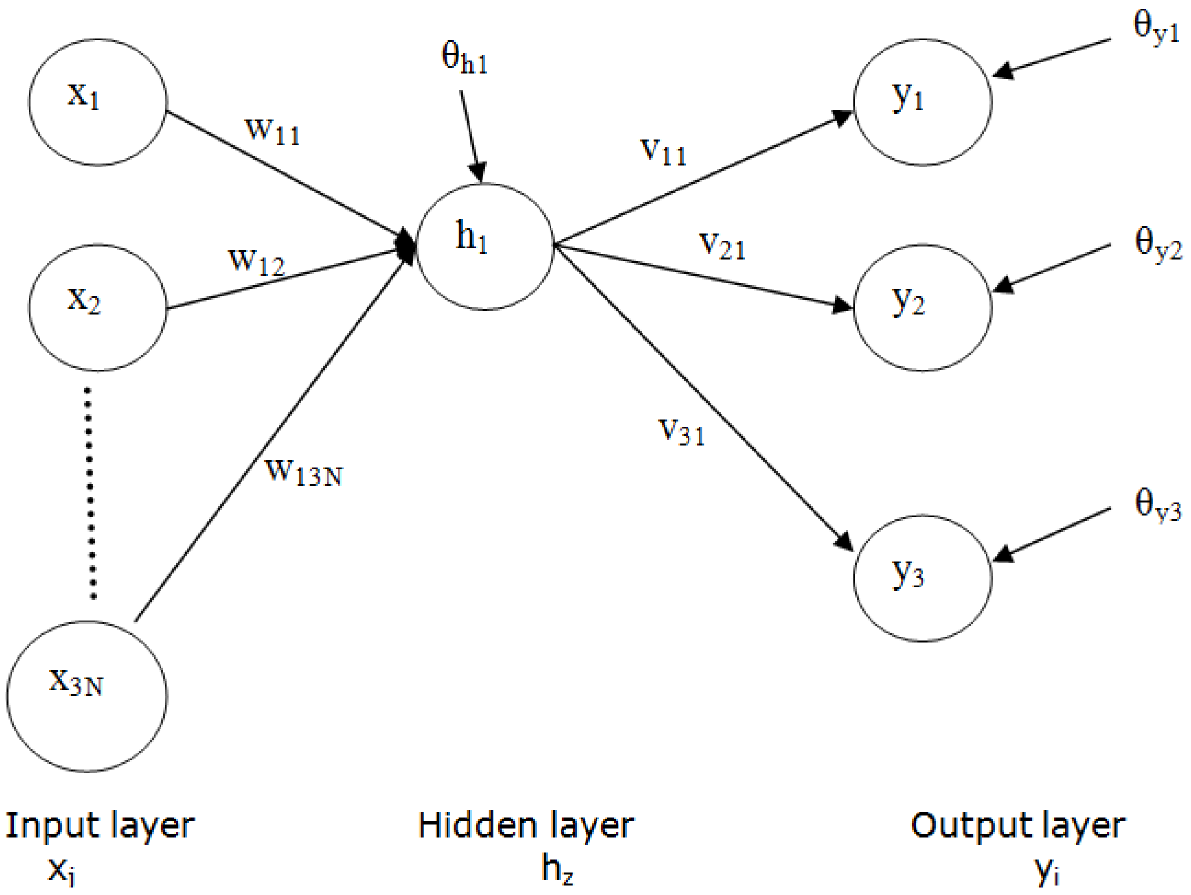


Figure 6: A node in the hidden layer of individual wheel NN

Input to hidden neuron z1 $= \sum_{i=1}^{3N} xiw1i + \theta h1$

where N is the number of previous samples required to make prediction.

The feature set is normalized to [-1,1]. Therefore the activation function chosen for the hidden layer is a sigmoid non linear bias function that is the hyperbolic tangent which ranges from -1 to 1.

Therefore the output of the hidden neuron z_1 $= \tanh(x)$

$= \tanh(\sum_{i=1}^{3N} xiw1i + \theta_{h1})$

The nodes in the output layer of MLPRegressor do not have activation function

Output of y1 $= v_{11} \tanh(\sum_{i=1}^{3N} xiw1i + \theta_{h1}) + \theta_{y1}$

Output of y2 $= v_{21} \tanh(\sum_{i=1}^{3N} xiw1i + \theta_{h1}) + \theta_{y2}$

Output of y3 $= v_{31} \tanh(\sum_{i=1}^{3N} xiw1i + \theta_{h1})+ \theta_{y3}$

Considering 'h' neurons in the hidden layer

Output of y1 $= \sum_{z=1}^{h} \mathrm{v1z} \tanh(\sum_{i=1}^{3N} xiwzi + \theta hz) + \theta y1$

Output of y2 $= \sum_{z=1}^{h} \mathrm{v2z} \tanh(\sum_{i=1}^{3N} xiwzi + \theta hz) + \theta y2$

Output of y3 $= \sum_{z=1}^{h} \mathrm{v3z} \tanh(\sum_{i=1}^{3N} xiwzi + \theta hz) + \theta y3$

After every iteration the value of the weights and bias are updated inorder to minimize the error between the actual and predicted value.

7.1.1 Wheel 1

The score of wheel1 model was computed for different values of N and different values of MLPRegressor parameters.

Five previous samples is found to be the best choice for predicting using wheel1 model.

Number of samples required for early prediction, N = 5

Number of hidden layers = 1.

The model becomes more complex with more number of hidden layers.

Number of input neurons 3N = 15

Number of neurons in hidden layer = 2*3*N + 1 = 31

Number of neurons in output layer = 3

7.1.1.1 MLPRegressor Parameters:

MLPRegressor model parameters arrived at for early predictions are as follows:

- *activation = 'tanh'*
- *alpha = 0.0001*
- *batch_size = 50*
- *beta_1 = 0.9*
- *beta_2 = 0.999*
- *early_stopping = True*
- *epsilon=1e-08*
- *hidden_layer_sizes=((2*3*N+1),)*
- *learning_rate_init=0.001*
- *max_iter=200*
- *random_state=1*

- *shuffle=True*
- *solver='adam'*
- *tol=0.0001*
- *validation_fraction=0.1*
- *verbose=False*

7.1.1.2 Training Results:

Enter number of previous samples required for prediction : 5

Size of feature set= 15

Training set score (28Apr-14May2013) = 0.976541633262

Best validation score= 0.977950560779

No. of iterations= 16

Loss= 1.76656146848e-05

7.1.1.3 Loss curve:

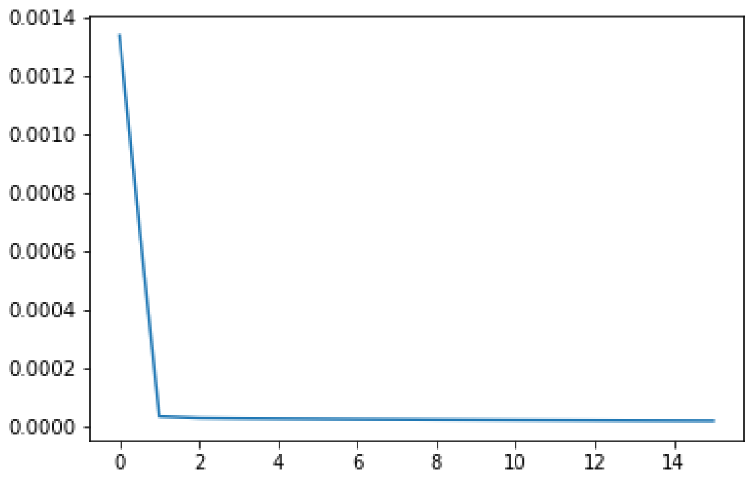


Figure 7: Loss Curve of Wheel 1 NN

7.1.1.4 Validation scores:

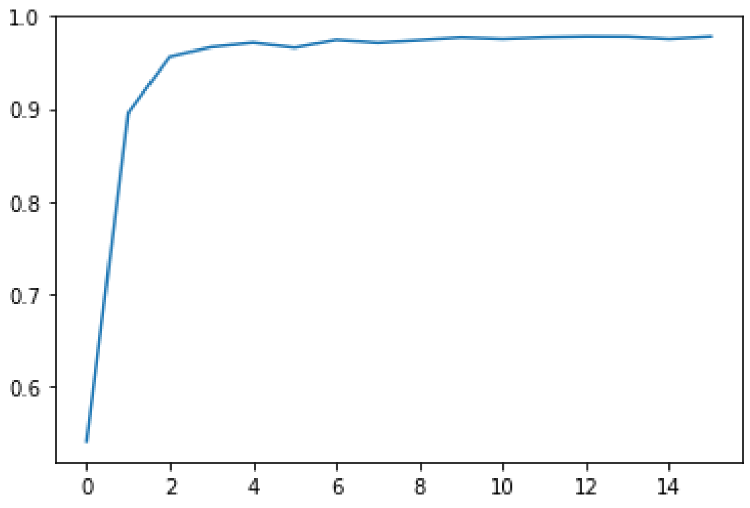


Figure 8: Validation score curve of Wheel 1 NN

7.1.1.5 Network weights:

Weights of the first layer

```
([[-0.01036857,  0.01411874, -0.12133806, ..., -0.0783444 ,
         0.05649789, -0.17331806],
       [-0.08284929,  0.34814834,  0.04809844, ...,  0.11443806,
        -0.25631459, -0.08747799],
       [ 0.14451131, -0.06360895, -0.28958579, ..., -0.21852794,
         0.26115393, -0.02447946],
       ...,
       [-0.06538596,  0.01483883, -0.12005918, ...,  0.12658711,
         0.05320511, -0.20725232],
       [ 0.12256083,  0.00146815,  0.34390179, ..., -0.06414128,
         0.01116676, -0.04444648],
       [ 0.20679279, -0.06908627,  0.26692723, ..., -0.20974518,
        -0.16315138,  0.22831985]])
```

Weights of the second layer

```
([[ 0.36223365,  0.26107837,  0.33901077],
       [-0.25391225, -0.3226769 , -0.20320175],
       [-0.13083948, -0.28235665,  0.06408927],
       ...,
       [ 0.31703944, -0.23965684,  0.18469217],
       [-0.15141796,  0.20173471, -0.08500699],
       [ 0.23284478, -0.10515857, -0.21387251]])]
```

7.1.1.6 Biases:

Biases of the neurons in hidden layer

-0.161489180063, 0.374484686484, -0.142801381247, -0.00398028940947, -0.15847422245, 0.285677991355, -0.121025256213, -0.022424188803, -0.0108162292247, 0.291583153626, -0.313210956024, 0.126100042141, 0.25743763319, -0.367323222285, 0.0415385213389, -0.150561201386, 0.285812237103, 0.0585151764188, 0.16330752923, -0.0591221483426, 0.117393972344, -0.12602983778, -0.378812504667, 0.0845864643995, -0.161133189945, -0.0131508279646, 0.275231045865, -0.0788808303759, -0.207827186061, -0.0687549402685, 0.286554660286

Biases of neurons in the output layer

0.19917702, -0.14318973, 0.22206825

7.1.1.7 Testing Results:

Nominal Data

The model was tested with data during nominal wheel behavior (Normal mode). The scores obtained are as follows:

Testing set1 score (14-31May 2013) = 0.619855028202

Testing set2 score (07-12Mar2013) = 0.754883869318

Testing set3 score (28-29Apr2013) = 0.934567375467

The model was tested with data corresponding to the time just before the wheel had failed.
Testing set4 score (04-05Oct2014) = 0.920655953234

When faulty data was fed to wheel1 model, it was not able to perceive any failure and returned a good score. Here the score of the model does not give any indication of the failure that is to happen.

7.1.2 Wheel 2

The score of wheel2 model was computed for different values of N and different values of MLPRegressor parameters. Five previous samples is found to be the best choice for predicting using wheel2 model.

Number of samples required for early prediction, N = 5

Number of hidden layers = 1.

The model becomes more complex with more number of hidden layers.

Number of input neurons 3N = 15

Number of neurons in hidden layer = 2*3*N + 1 = 31

Number of neurons in output layer = 3

7.1.2.1 MLPRegressor Parameters:

MLPRegressor model parameters arrived at for early predictions are as follows:

- *activation = 'tanh'*
- *alpha = 0.0001*
- *batch_size = 50*
- *beta_1 = 0.9*
- *beta_2 = 0.999*
- *early_stopping = True*
- *epsilon=1e-08*
- *hidden_layer_sizes=((2*3*N+1),)*
- *learning_rate_init=0.001*
- *max_iter=200*
- *random_state=1*
- *shuffle=True*
- *solver='adam'*
- *tol=0.0001*
- *validation_fraction=0.1*
- *verbose=False*

7.1.2.2 Training Results:

```
Enter number of previous samples required for prediction : 5
Size of feature set= 15
Training set score (28Apr-14May2013) = 0.990626502841
Best validation score= 0.991259515605
No. of iterations= 17
Loss= 1.58692835966e-05
```

7.1.2.3 Loss curve:

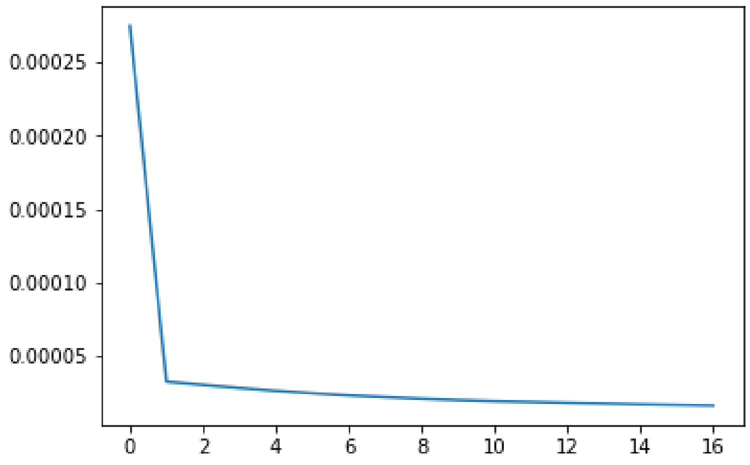


Figure 9: Loss Curve of Wheel 2 NN

7.1.2.4 Validation scores:

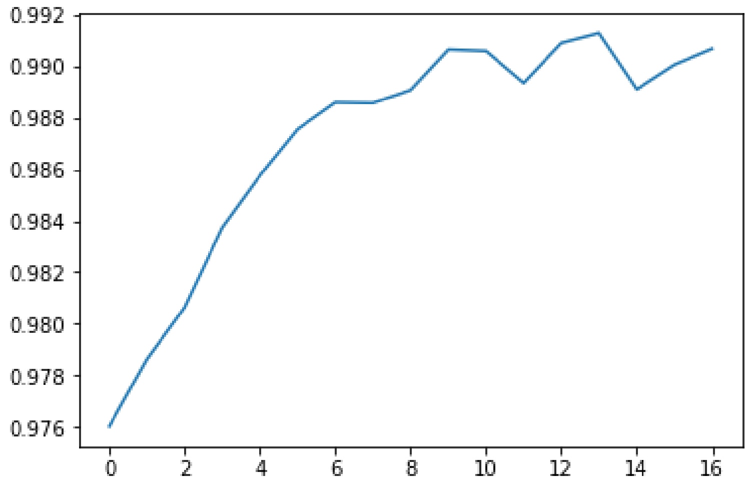


Figure 10: Validation score curve of Wheel 2 NN

7.1.2.5 Network weights:

Weights of the first layer

```
([[  7.63833366e-03,   9.54487066e-03,  -2.48502824e-02, ...,
       7.90783602e-03,   3.24348538e-03,   9.80243940e-03],
     [ -5.68432996e-02,   3.13951795e-01,   1.88062608e-02, ...,
       1.21242011e-01,  -2.59865930e-01,  -7.81199031e-02],
     [  1.24957216e-01,  -4.98993871e-02,  -1.65085383e-01, ...,
      -1.20456830e-01,   1.44581229e-01,  -4.22322915e-02],
     ...,
     [ -5.99372697e-03,   1.25501296e-02,  -2.11717304e-02, ...,
       8.13218874e-02,   1.67245724e-03,   1.20902800e-04],
     [  1.40405755e-01,  -2.81071628e-02,   3.09963596e-01, ...,
      -5.38812932e-02,  -2.76205735e-04,  -3.80053528e-02],
     [  1.30876959e-01,  -7.01450417e-02,   1.93904766e-01, ...,
      -2.05791883e-01,  -1.19987263e-01,   1.15695241e-01]])
```

Weights of the second layer

```
([[ 0.36369987,  0.29890719,  0.34138402],
     [-0.27405144, -0.35593607, -0.19915778],
     [-0.12160085, -0.25481057,  0.0650494 ],
     ...,
     [ 0.30340201, -0.24128728,  0.19072039],
     [-0.00946958,  0.20068504, -0.10523336],
     [ 0.23410552, -0.13507482, -0.22063798]])
```

7.1.2.6 Biases:

Biases of the neurons in hidden layer

-0.146272290869, 0.360146071342, -0.153232683332, 0.00915243150165, -0.128573509826, 0.306681791632, -0.118526992608, -0.0277735201927, 0.0270964328449, 0.298741555751, -0.32209217283, 0.142397281781, 0.270453697003, -0.356451311238, 0.0355777068679, -0.152175123424, 0.276289000569, 0.0282941679208, 0.158937415684, -0.0261693491004, 0.12031742843, -0.136593494759, -0.342488195955, 0.071541448154, -0.125219652678, -0.00338854285189, 0.280316434504, -0.069052851327, -0.171080328335, -0.106383466695, 0.286059455148

Biases of neurons in the output layer

0.20204885, -0.17350146, 0.214453

7.1.2.7 Testing Results:

Nominal Data

The model was tested with data during nominal wheel behavior (normal mode). The scores obtained are as follows:

Testing set1 score (14-31May2013) = -0.469015088261

Testing set2 score (07-12Mar2013) = 0.975387064518

Testing set3 score (28-29Apr2013) = 0.848105552426

The model was tested with data corresponding to the time just before the wheel had failed.
Testing set4 score (04-05Oct2014) = 0.448218468252

Here the score of the model does not give any indication of the failure that is to happen.

7.1.3 Wheel 3

The score of wheel3 model was computed for different values of N and different values of MLPRegressor parameters. Five previous samples is found to be the best choice for predicting using wheel3 model.

Number of samples required for early prediction, N = 5

Number of hidden layers = 1.

The model becomes more complex with more number of hidden layers.

Number of input neurons 3N = 15

Number of neurons in hidden layer = 2*3*N + 1 = 31

Number of neurons in output layer = 3

7.1.3.1 MLPRegressor Parameters:

MLPRegressor model parameters arrived at for early prediction are as follows:

- *activation = 'tanh'*
- *alpha = 0.0001*
- *batch_size = 50*
- *beta_1 = 0.9*
- *beta_2 = 0.999*
- *early_stopping = True*
- *epsilon=1e-08*
- *hidden_layer_sizes=((2*3*N+1),)*
- *learning_rate_init=0.001*
- *max_iter=200*
- *random_state=1*
- *shuffle=True*
- *solver='adam'*
- *tol=0.0001*
- *validation_fraction=0.1*
- *verbose=False*

7.1.3.2 Training Results:

Enter number of previous samples required for prediction: 5

Size of feature set= 15

Training set score (28Apr-14May2013) = 0.931036808591

Best validation score= 0.933469491292

No. of iterations= 20

Loss= 1.62222148751e-05

7.1.3.3 Loss curve:

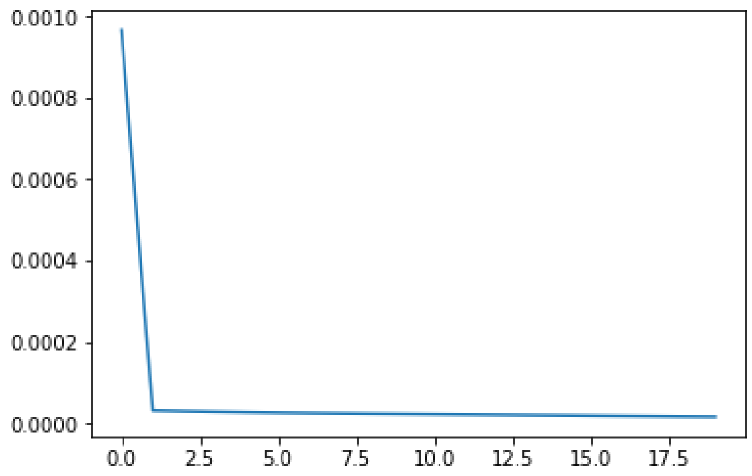


Figure 11: Loss Curve of Wheel 3 NN

7.1.3.4 Validation scores:

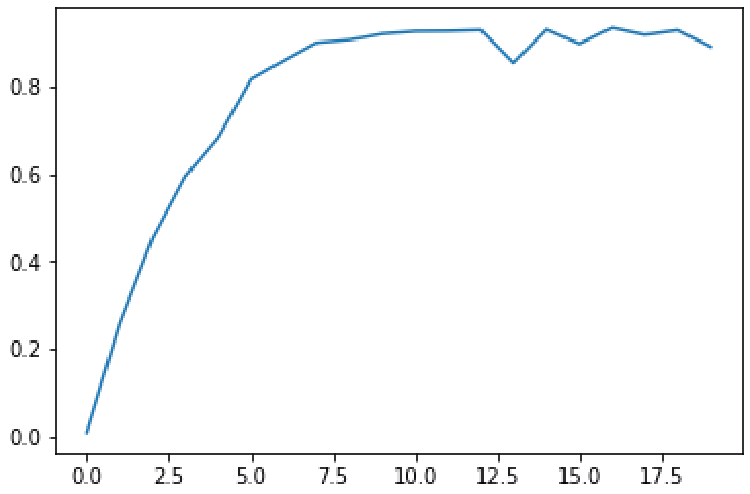


Figure 12: Validation score curve of Wheel 3 NN

7.1.3.5 Network weights:

Weights of the first layer

```
([[ 0.00887959,  0.01704371, -0.17421089, ..., -0.07669604,
         0.07309901, -0.1936124 ],
       [-0.07835569,  0.32476968,  0.04568621, ...,  0.1116654 ,
        -0.24662246, -0.08500789],
       [ 0.14885841, -0.06655063, -0.27572394, ..., -0.20846562,
         0.2441058 , -0.03098378],
       ...,
       [-0.09075843,  0.02940715, -0.16845412, ...,  0.17126823,
         0.07615029, -0.2337606 ],
       [ 0.11977724, -0.00083396,  0.33355659, ..., -0.06167519,
         0.00998291, -0.04260176],
       [ 0.18999464, -0.06623224,  0.25047898, ..., -0.21709382,
        -0.13607633,  0.21716618]])
```

Weights of the second layer

```
([[ 0.36528056,  0.17029067,  0.33055023],
    [-0.25304403, -0.31792885, -0.20307192],
    [-0.13125304, -0.25668866,  0.06291955],
    ...,
    [ 0.31211536, -0.22707687,  0.18553414],
    [-0.15146995,  0.18949462, -0.0835823 ],
    [ 0.23045297, -0.09904481, -0.21107748]])
```

7.1.3.6 Biases:

Biases of the neurons in hidden layer

-0.161021003963, 0.373876758909, -0.14317505663, -0.00311528490981, -0.159442265447, 0.283708276224, -0.121718329799, -0.022042031354, -0.00521166390706, 0.2913541453, -0.313537887632, 0.131085601443, 0.257808310828, -0.367011923574, 0.0417603027296, -0.150914542139, 0.285847387569, 0.0526552811128, 0.163720027976, -0.0615711743315, 0.11827437045, -0.126487241922, -0.380263557775, 0.0839501318479, -0.162607204926, -0.0129503386174, 0.275153804854, -0.0787120693086, -0.209192307388, -0.0679757616602, 0.286301398525

Biases of neurons in the output layer

0.19890628, -0.14387666, 0.2225139

7.1.3.7 Testing Results:

Nominal Data

The model was tested with data during nominal wheel behavior (Normal mode). The scores obtained are as follows:

Testing set1 score (14-31May2013) = 0.889355535859

Testing set2 score (07-12Mar2013) = -0.761460883688

Testing set3 score (28-29Apr2013) = 0.753692440339

The model was tested with data corresponding to the time just before the wheel had failed.

Testing set4 score (04-05Oct2014) = -0.527643641247

Wheel 3 model predictions were inaccurate as it returned a negative score with nominal as well as faulty data.

7.1.4 Wheel 4

The score of wheel4 model was computed for different values of N and different values of MLPRegressor parameters. Eight previous samples is found to be the best choice for predicting using wheel4 model.

Number of samples required for early prediction, N = 8

Number of hidden layers = 1.

The model becomes more complex with more number of hidden layers.

Number of input neurons 3N = 24

Number of neurons in hidden layer = 2*3*N + 1 = 49

Number of neurons in output layer = 3

7.1.4.1 MLPRegressor Parameters:

MLPRegressor model parameters arrived at for early prediction are as follows:

- *activation = 'tanh'*
- *alpha = 0.0001*
- *batch_size = 50*
- *beta_1 = 0.9*
- *beta_2 = 0.999*
- *early_stopping = True*
- *epsilon=1e-08*
- *hidden_layer_sizes=((2*3*N+1),)*
- *learning_rate_init=0.001*
- *max_iter=200*
- *random_state=1*
- *shuffle=True*
- *solver='adam'*
- *tol=0.0001*
- *validation_fraction=0.1*
- *verbose=False*

7.1.4.2 Training Results:

```
Enter number of previous samples required for prediction : 8
Size of feature set= 24
Training set score (28Apr-14May2013) = 0.910150592158
Best validation score= 0.949700239111
No. of iterations= 36
Loss= 2.34302417188e-05
```

7.1.4.3 Loss curve:

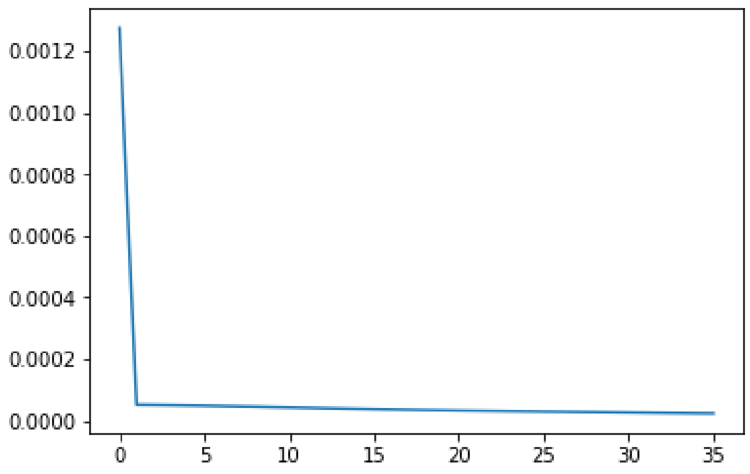


Figure 13: Loss curve of Wheel 4 NN

7.1.4.4 Validation scores:

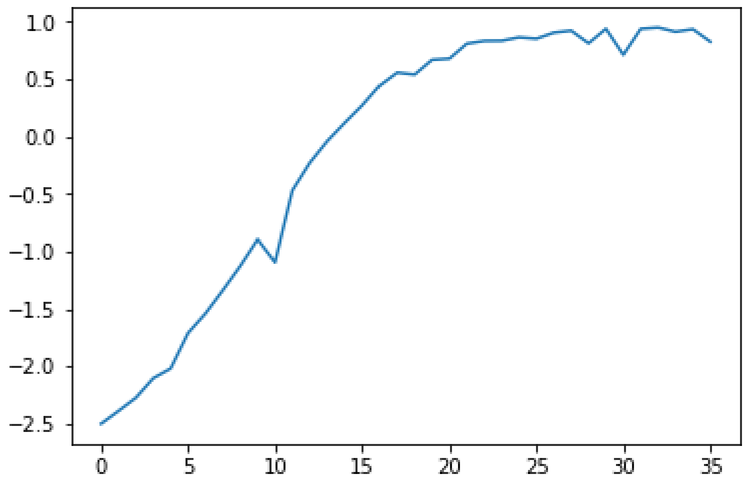


Figure 14: Validation score curve of Wheel 4 NN

7.1.4.5 Network weights:

<u>Weights of the first layer</u>

```
([[-0.02202537,  0.01293854, -0.11994587, ...,  0.15929396,
        -0.08323931, -0.08948255],
       [-0.20616226, -0.19396516,  0.0704956 , ..., -0.12657764,
         0.22722771,  0.04851371],
       [-0.21679069,  0.04921019, -0.05785439, ..., -0.21045106,
        -0.1887446 , -0.157551  ],
       ...,
       [ 0.09683262, -0.11501295, -0.06196458, ..., -0.00370807,
         0.05055639,  0.02045487],
       [ 0.03425618,  0.20388857, -0.12685538, ..., -0.09602572,
         0.24101386,  0.01134839],
       [-0.01128804,  0.08296134, -0.1376722 , ..., -0.25823743,
         0.11091581,  0.10738227]])
```

Weights of the second layer

```
([[-0.04974976, -0.242852  ,  0.16857574],
     [-0.30637188,  0.23620266, -0.19337889],
     [ 0.07525747, -0.13100597,  0.23303871],
     ...,
     [-0.10387579,  0.32046053,  0.17794016],
     [ 0.08011272,  0.25036052,  0.27652803],
     [ 0.28007138,  0.30992985, -0.24201664]])
```

7.1.4.6 Biases:

Biases of the neurons in hidden layer

-0.149132787604, 0.222701661131, -0.0325966885808, 0.182757088271, 0.0766044732115, 0.0154550489905, -0.14155532607, 0.0586091142956, 0.196478414298, -0.142751603396, 0.192496962125, 0.200763138013, -0.0701353183518, 0.181519547881, 0.194230502109, -0.200490414141, 0.181556635264, -0.207911252697, 0.252089059422, 0.0501958463519, -0.125907631026, -0.154504787535, 0.0918115283376, 0.237807469265, 0.22412478755, 0.0272197153332, 0.268020903086, 0.0801247337754, -0.218713407991, -0.0766168936433, -0.0663874663907, 0.310957014999, -0.0387853763393, 0.252786001927, 0.00717946431872, -0.215820801052, -0.271901147155, -0.0898223088074, -0.186918099144, 0.0374378323751, -0.0231869807782, 0.339139240103, 0.206825628319, -0.169849370024, -0.0641095507245, 0.0131785717863, 0.0788071220221, 0.196162771698, 0.0230610449225

Biases of neurons in the output layer

-0.27231753, -0.25231765, -0.30017846

7.1.4.7 Testing Results:

Nominal Data

The model was tested with data during nominal wheel behavior (normal mode). The scores obtained are as follows:

Testing set1 score (14-31May2013) = 0.798770315857

Testing set2 score (07-12Mar2013) = 0.708944403582

Testing set3 score (28-29Apr2013) = 0.96011476634

The model was tested with data corresponding to the time just before the wheel had failed.

Testing set4 score (04-05Oct2014) = -35.7455740807

When faulty data was fed to wheel4 model, it returned a negative score.

The individual wheel models are tested with wheel telemetry data during nominal and failure conditions. The response of the models to the pre-failure data is not uniform. It is seen that the individual wheel models are not accurate in early prediction of failures.

7.2 Approach-2

Even though the four wheels are considered to be independent theoretically, practically there exists coupling between the four reaction wheels of a spacecraft. The reaction wheel system comprising of all the four wheels needs to be modeled. Therefore the feature set for developing the wheel model should have the parameters of all four wheels.

Let ‘N’ be the number of previous samples required to predict the current sample.

Training Data:

Inputs:

x_1 = Wheel1 Input Torque (previous sample 1)
x_2 = Wheel1 Speed (previous sample 1)
x_3 = Wheel1 DFC torque (previous sample 1)
x_4 = Wheel2 Input Torque (previous sample 1)
x_5 = Wheel2 Speed (previous sample 1)
x_6 = Wheel2 DFC Torque (previous sample 1)
x_7 = Wheel3 Input Torque (previous sample 1)
x_8 = Wheel3 Speed (previous sample 1)
x_9 = Wheel3 DFC Torque (previous sample 1)
x_{10} = Wheel4 Input Torque (previous sample 1)
x_{11} = Wheel4 Speed (previous sample 1)
x_{12} = Wheel4 DFC Torque (previous sample 1)

x_{13} = Wheel1 Input Torque (previous sample 2)
x_{14} = Wheel1 Speed (previous sample 2)
x_{15} = Wheel1 DFC torque (previous sample 2)
x_{16} = Wheel2 Input Torque (previous sample 2)
x_{17} = Wheel2 Speed (previous sample 2)
x_{18} = Wheel2 DFC Torque (previous sample 2)
x_{19} = Wheel3 Input Torque (previous sample 2)
x_{20} = Wheel3 Speed (previous sample 2)
x_{21} = Wheel3 DFC Torque (previous sample 2)
x_{22} = Wheel4 Input Torque (previous sample 2)
x_{23} = Wheel4 Speed (previous sample 2)
x_{24} = Wheel4 DFC Torque (previous sample 2)
"
"
x_{12N-11} = Wheel1 Input Torque (previous sample N)
x_{12N-10} = Wheel1 Speed (previous sample N)
x_{12N-9} = Wheel1 DFC torque (previous sample N)
x_{12N-8} = Wheel2 Input Torque (previous sample N)
x_{12N-7} = Wheel2 speed (previous sample N)
x_{12N-6} = Wheel2 DFC Torque (previous sample N)
x_{12N-5} = Wheel3 Input Torque (previous sample N)
x_{12N-4} = Wheel3 Speed (previous sample N)
x_{12N-3} = Wheel3 DFC Torque (previous sample N)
x_{12N-2} = Wheel4 Input Torque (previous sample N)
x_{12N-1} = Wheel4 Speed (previous sample N)
x_{12N} = Wheel4 DFC Torque (previous sample N)

Outputs:

y_1 = Wheel1 Input Torque (current sample)
y_2 = Wheel1 Speed (current sample)
y_3 = Wheel1 DFC torque (current sample)
y_4 = Wheel2 Input Torque (current sample)
y_5 = Wheel2 Speed (current sample)
y_6 = Wheel2 DFC Torque (current sample)
y_7 = Wheel3 Input Torque (current sample)
y_8 = Wheel3 Speed (current sample)
y_9 = Wheel3 DFC Torque (current sample)
y_{10} = Wheel4 Input Torque (current sample)
y_{11} = Wheel4 Speed (current sample)
y_{12} = Wheel4 DFC Torque (current sample)

Number of neurons in input layer = 12*N where N is the number of previous samples required for prediction.

Number of neurons in the output layer = 12

Number of neurons in hidden layer = h (to be determined)

Total number of neurons = 12N * h* 12

The number of hidden layers in the multilayer perceptron is kept at 1.

The number of neurons in the hidden layer = 2*n +1 where n is the number of input neurons. [Hecht-Nielsen (1987)][2]

Consider any one neuron in the hidden layer.

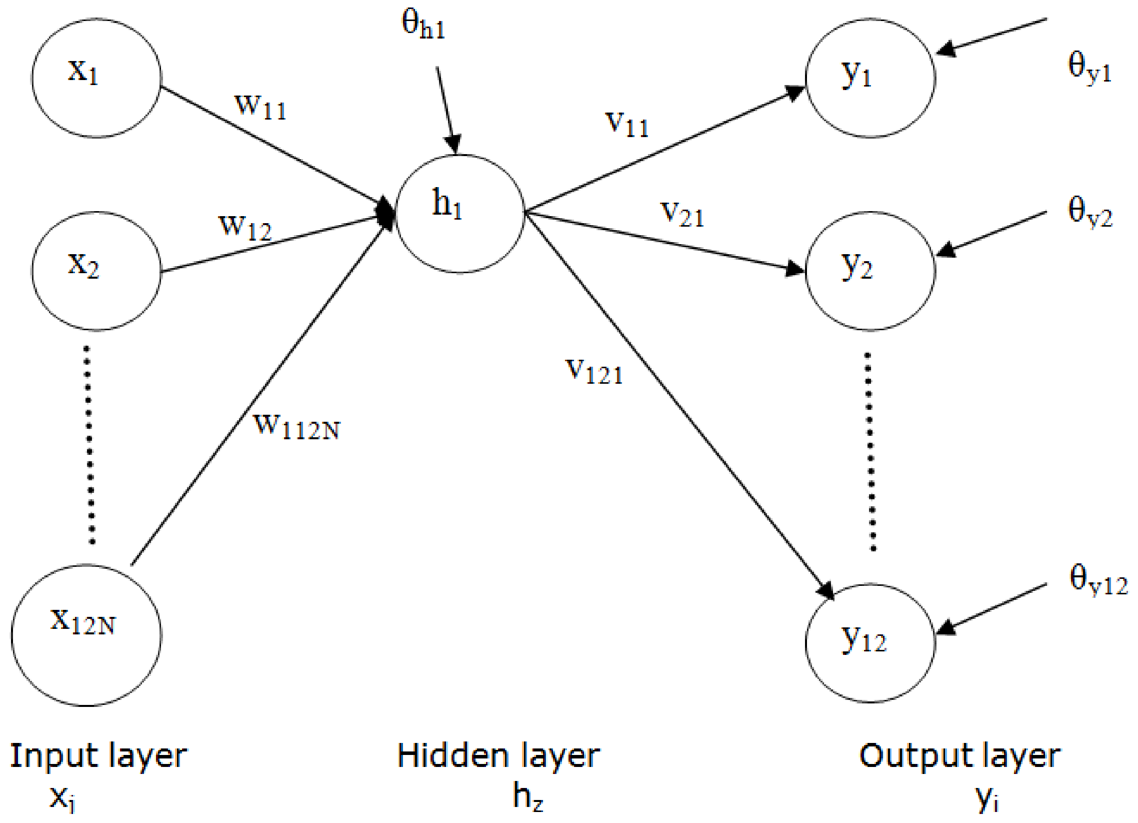


Figure 15: A node in the hidden layer of all wheel NN

Input to hidden neuron z_1 $= \sum_{i=1}^{12N} xiw1i + \theta_{h1}$

where N is the number of previous samples required to make prediction.

The feature set is normalized to [-1,1]. Therefore the activation function chosen for the hidden layer is a sigmoid non linear bias function that is the hyperbolic tangent which ranges from -1 to 1.

Therefore the output of the hidden neuron z_1 $= \tanh(x)$

$= \tanh(\sum_{i=1}^{12N} xiw1i + \theta_{h1})$

The nodes in the output layer of MLPRegressor do not have activation function

Output of y1 $= v_{11} \tanh(\sum_{i=1}^{12N} xiw1i + \theta_{h1}) + \theta_{y1}$

Output of y2 $= v_{21} \tanh(\sum_{i=1}^{12N} xiw1i + \theta_{h1}) + \theta_{y2}$

"

"

Output of y12 $= v_{121} \tanh(\sum_{i=1}^{12N} xiw1i + \theta_{h1}) + \theta_{y12}$

Considering 'h' neurons in the hidden layer

Output of y1	$= \sum_{z=1}^{h} \mathrm{v1z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y1}$
Output of y2	$= \sum_{z=1}^{h} \mathrm{v2z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y2}$
Output of y3	$= \sum_{z=1}^{h} \mathrm{v3z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y3}$
Output of y4	$= \sum_{z=1}^{h} \mathrm{v4z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y4}$
Output of y5	$= \sum_{z=1}^{h} \mathrm{v5z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y5}$
Output of y6	$= \sum_{z=1}^{h} \mathrm{v6z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y6}$
Output of y7	$= \sum_{z=1}^{h} \mathrm{v7z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y7}$
Output of y8	$= \sum_{z=1}^{h} \mathrm{v8z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y8}$
Output of y9	$= \sum_{z=1}^{h} \mathrm{v9z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y9}$
Output of y10	$= \sum_{z=1}^{h} \mathrm{v10z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y10}$
Output of y11	$= \sum_{z=1}^{h} \mathrm{v11z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y11}$
Output of y12	$= \sum_{z=1}^{h} \mathrm{v12z} \tanh(\sum_{i=1}^{12N} xiwzi + \theta\mathrm{hz}) + \theta\mathrm{y12}$

After every iteration the value of the weights and bias are updated inorder to minimize the error between the actual and predicted value.

7.2.1 Wheel model during Normal mode without maneuvering

The score of this wheel system model was computed for different values of N and different values of MLPRegressor parameters. Five previous samples is found to be the best choice for predicting using this wheel model.

Number of samples required for early prediction, N =5

Number of hidden layers = 1.

The model becomes more complex with more number of hidden layers.

Number of input neurons 12N = 60

Number of neurons in hidden layer = 2*12*N + 1 = 121

Number of neurons in output layer = 12

7.2.1.1 MLPRegressor Parameters:

MLPRegressor model parameters arrived at for early predictions are as follows:

- *activation = 'tanh'*
- *alpha = 0.0001*
- *batch_size = 100*
- *beta_1 = 0.9*
- *beta_2 = 0.999*
- *early_stopping = True*
- *epsilon=1e-08*
- *hidden_layer_sizes=((2*12*N+1),)*
- *learning_rate_init=0.001*
- *max_iter=200*
- *random_state=0*
- *shuffle=True*
- *solver='adam'*
- *tol=0.0001*
- *validation_fraction=0.1*
- *verbose=False*

7.2.1.2 Training Results:

Enter number of previous samples required for prediction: 5

Size of feature set= 60

Training set score (28Apr-14May2013) = 0.953641009969

Best validation score = 0.954249087289

No. of iterations = 14

Loss = 7.69761741885e-05

7.2.1.3 Loss curve:

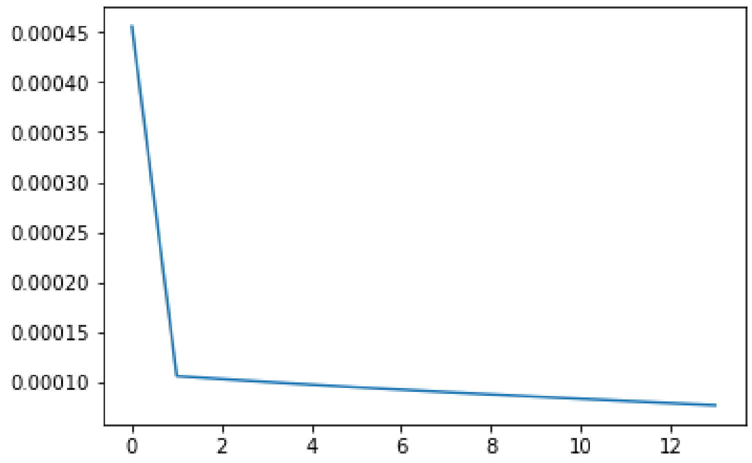


Figure 16: Loss Curve of all wheels NN

7.2.1.4 Validation scores:

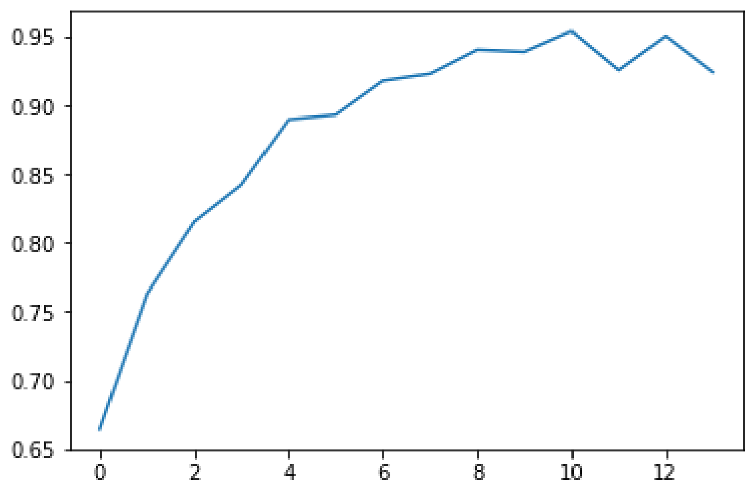


Figure 17: Validation score curve of all wheel NN

7.2.1.5 Network weights:

Weights of the first layer

```
([[-0.00363164,  0.06572265,  0.02198438, ...,  0.0710488 , 0.01472886,  0.05335785],
   [ 0.00670966,  0.17462197,  0.05950975, ...,  0.02188538, 0.13704337, -0.12773354],
   [ 0.02670024, -0.12020445,  0.07954184, ..., -0.12923572, 0.13523344,  0.02782365],
      ...,
  [ 0.15028025, -0.02125047,  0.06773059, ..., -0.08719965, -0.14437286,  0.04602296],
  [-0.09667753,  0.00643976, -0.13079567, ..., -0.02742947,  0.08538127, -0.00652794],
  [-0.15307552,-0.18630394,-0.03430641, ..., -0.04188962,  0.07496583, -0.14080431]])
```

Weights of the second layer

```
([[-0.11751063, 0.14513255,  0.12319875, ...,  0.03271961,  -0.07199406,  0.0780566 ],
   [ 0.0728453 ,  0.12145791,  0.19864291, ..., -0.13738579, -0.11359139, -0.169303  ],
   [ 0.10295924,  0.06147273,  0.13587354, ...,  0.14168167, 0.05110766, -0.02148167],
      ...,
   [ 0.08689843,  0.07846768, -0.19516427, ...,  0.01482349, -0.20644008,  0.1059399 ],
   [0.15032848, -0.01635745,  0.13140829, ...,  0.09460374, -0.02235696, -0.01295069],
   [-0.1267516, -0.1014467 ,  0.06649714, ..., -0.06126668,  0.06805263, -0.17242644]])
```

7.2.1.6 Biases:

Biases of the neurons in hidden layer

0.0677652172999, 0.00511795328917, -0.0181107576061, -0.0919534982591, 0.168531804192, -0.128017808356, -0.0991665701809, -0.0452117466067, -0.14213607237, 0.166584319712, 0.141902871182, -0.0303881949289, 0.00485201303202, -0.150279271861, 0.129312662459, 0.0935342887772, -0.0292666809809, -0.0168272957765, 0.0638361660927, -0.0178489248455, -0.0887475630246, -0.0673740826277, 0.0132357670185, 0.182163154104, -0.0121211299954, -0.169318324543, 0.150769688537, -0.0668748740683, -0.157633697532, -0.146040470475, -0.126910641424, 0.0373024173262, -0.0403791102403, 0.140511177181, 0.100941392756, -0.056826800516, 0.184626652902, 0.0624584108029, -0.181803015063, -0.0217367785649, 0.157772262229, -0.0658923049886, 0.146967448151, 0.0492811369368, 0.0381502643761, -0.0418866633071, 0.158087684058, -0.0153165260088, 0.0404228156932, 0.0422516080828, 0.0523455691096, -0.061899930636, 0.125065761023, 0.0409298633078, -0.0447522467974, -0.0723377719165, -0.133092489164, 0.0317904997835, 0.0963176997894, -0.0342271467331, 0.161153522292, -0.127803853959, -0.0119381774661, 0.0407704288013, 0.0324144401283, 0.149787567815, 0.0594417199048, 0.126019639921, 0.0655050916235, -0.0466350326748, -0.0740013323249, -0.0633190645783, -0.0447968883681, 0.113684870906, 0.067568122927, 0.12592430118, 0.146085879349, 0.106994829877, 0.152792044811, 0.168397624027, -0.0590469347736, -0.0269536365994, 0.0229770725435, 0.0613445445135, -0.0198296094514, 0.111522038176, 0.0398277301113, 0.0374854234391, -0.13483569424, -0.118764370676, -0.0595366845142, 0.158765708503, -0.145918425854, 0.158535358818, -0.014586874458, -0.014713412093, 0.134141260777, 0.11222181903, -0.129540355728, 0.103547892324, -0.00378159585263, -0.0717109910472, -0.14963569543, 0.0392142964149, 0.0944677362548, 0.146780242408, 0.161509209881, -0.12721229177, -0.0302756369291, -0.171623818196, 0.0423052137551, 0.132911168302, 0.0598357990267, -0.0475677617374, 0.101545567976, 0.188072842877, -0.173408706511, 0.111134804893, -0.0708485249307, 0.0950819289031, -0.086963226543

Biases of neurons in the output layer

0.157226872718, 0.0615293972732, -0.181424152228, -0.111567764617, -0.155907480093, 0.0415044674155, 0.0246600418141, -0.0574116557309, 0.102106661342, -0.182156824819, 0.134433859958, 0.157122227182

7.2.1.7 Testing Results:

Nominal Data

The model was tested with data during nominal wheel behavior (normal mode). The scores obtained are as follows:

Testing set1 score (14-31May2013) = 0.677915765695

Testing set2 score (07-12Mar2013) = 0.627092878958

Testing set3 score (28-29Apr2013) = 0.885870490827

The model was tested with data corresponding to the time wheel had failed.

Testing set4 score (04-05Oct2014) = -3.03848315826

Therefore combined wheel model returned a positive score when tested with wheel telemetry data during normal mode. The model returned a negative score when tested with telemetry data prior to the occurrence of Wheel 1 failure. The neural network's score is therefore providing an indication of failure that is to happen. Therefore this model considering all the four wheels is chosen for early prediction rather than individual wheel model. And, MLPRegressor's score is the criteria with which the failures are predicted.

7.2.2 Wheel model during Normal mode with maneuvering

Neural Network was developed to model the reaction wheel system during maneuvering in Normal model. The dataset for training and testing this model was extracted from the telemetry data of Satellite_PQR. Satellite_PQR undergoes frequent maneuvering due to its imaging requirement. **The data for training and testing the model was chosen during the time the wheels had undergone maneuvering in Normal mode.** During this time, the satellite was in Normal mode (3 axis SKF), Wheel Mode Flag was 'TOC' where wheels are the control actuators, DFC was enabled and Step And Stare (SnS) Flag= 'True'. Each maneuvering instance is of short duration (<10 minutes). Therefore different maneuvering instances were combined to generate the training data set.

Training data:	10th October 2017 02:57:51:068 to 03:04:24:287
	10th October 2017 04:31:30:811 to 04:36:58:493
	11th October 2017 04:13:39:157 to 04:20:28:761
	12th October 2017 03:57:09:432 to 04:03:26:266
	12th October 2017 05:33:33:016 to 05:38:44:313
	13th October 2017 03:40:06:936 to 03:46:07:385
Testing data set-1:	13th October 2017 05:16:14:135 to 05:21:25:433
Testing data set-2:	17th October 2017 04:04:47:542 to 04:11:37:146

The details of the wheel parameters in Satellite_PQR telemetry is as given below. There are 64 sub frames and 128 channels/sub frame in Satellite_PQR master frame.

Table 2: Telemetry Details (Satellite_PQR)

Sl No	Parameter	Sampling Rate	Frame ID	Range	Resolution	Type
1	Wheel-1,2,3,4 Input Torque	16s	58	±0.3 Nm	9.15555e-6 Nm	16 bit 2 words
5	Wheel 1,2,3,4 speed	2s	1, 9, 17, 25, 33, 41, 49, 57	±6000 rpm	1.8311E-01 rpm	16 bit 2 words
9	DFC Torque Wheel 1,2,3,4	4s	11, 27, 43, 59	±0.05Nm	1.525E-06 Nm	16 bit 2 words

The feature set for training and testing is generated using the logic described in section 4.2.

	Wheel Input Torque	**DFC Torque**	**Wheel Speed**
Current data	**F58**	**F59**	**F57**
		N=1	
Current data	**F58**	**F59**	**F57**
Previous sample1	**F58**	**F43**	**F41**
		N=2	
Current data	**F58**	**F59**	**F57**
Previous sample1	**F58**	**F43**	**F41**
Previous sample2	**F58**	**F27**	**F25**
		N=3	
Current data	**F58**	**F59**	**F57**
Previous sample1	**F58**	**F43**	**F41**
Previous sample2	**F58**	**F27**	**F25**
Previous sample3	**F58′**	**F11**	**F09**
		N=4	
Current data	**F58**	**F59**	**F57**
Previous sample1	**F58**	**F43**	**F41**
Previous sample2	**F58**	**F27**	**F25**
Previous sample3	**F58′**	**F11**	**F09**
Previous sample4	**F58′**	**F43′**	**F41′**
		N=5	
Current data	**F58**	**F59**	**F57**
Previous sample1	**F58**	**F43**	**F41**
Previous sample2	**F58**	**F27**	**F25**
Previous sample3	**F58′**	**F11**	**F09**
Previous sample4	**F58′**	**F43′**	**F41′**
Previous sample5	**F58′**	**F27′**	**F25′**
		N=6	
Current data	**F58**	**F59**	**F57**
Previous sample1	**F58**	**F43**	**F41**
Previous sample2	**F58**	**F27**	**F25**
Previous sample3	**F58′**	**F11**	**F09**
Previous sample4	**F58′**	**F43′**	**F41′**
Previous sample5	**F58′**	**F27′**	**F25′**
Previous sample6	**F58′′**	**F11′**	**F09′**

Where F58′, F43′, F41′, F27′, F25′, F11′, F09′ are from previous master frame. F58′′ is from previous to previous master frame.

And so on.

Wheel speed has a range of ±6000 rpm, Wheel input torque has a range of ±0.3 Nm, DFC Torque has a range of ±0.05 Nm. Therefore these parameters are normalized to ±1 range before training the model.

The size of the dataset for modeling the wheel during maneuvering condition is small. For small datasets, 'lbfgs' solver for weight optimization can converge faster. It is an optimizer in the family of quasi-Newton methods.

7.2.2.1 MLPRegressor Parameters:

The score of this wheel system model was computed for different values of N and different values of MLPRegressor parameters. Two previous samples is found to be the best choice for predicting using this wheel model.

Number of samples required for early prediction, N =2

Number of hidden layers = 1.

The model becomes more complex with more number of hidden layers.

Number of input neurons 12N = 24

Number of neurons in hidden layer = 12*N = 24

Number of neurons in hidden layer is kept same as that in input layer

Number of neurons in output layer = 12

MLPRegressor model parameters arrived at for early predictions are as follows:

- *activation = 'tanh'*
- *alpha = 0.0001*
- *hidden_layer_sizes=((12*N),)*
- *max_iter=200*
- *random_state=0*
- *solver='lbfgs'*
- *tol=0.0001*

7.2.2.2 Training Results:

Enter number of previous samples required for prediction: 2

Size of feature set= 24

Training set score = 0.999695380262

No. of iterations = 177

Loss = 4.91546617576e-05

7.2.2.3 Network weights:

Weights of the first layer

```
([[-0.07322437,  0.11902873, -0.20233761, ...,  0.30044089,
         0.01703599,  0.11639699],
       [ 0.11837667,  0.23008752, -0.19150185, ...,  0.02530049,
         0.22752096,  0.08901146],
       [-0.11976597, -0.22796987, -0.33771448, ...,  0.04180342,
         0.28315928, -0.28030327],
       ...,
       [ 0.09753176, -0.12957281,  0.28900468, ..., -0.36330848,
        -0.3073663 , -0.30981081],
       [-0.05908745, -0.08905146,  0.32353034, ..., -0.24965307,
        -0.1325823 ,  0.05324447],
       [ 0.18541311, -0.06856703, -0.27676433, ...,  0.09562638,
         0.10026593,  0.03166648]])
```

Weights of the second layer

```
([[ 0.08626996,  0.19495787, -0.17489431, ...,  0.05784491,
         0.09665944,  0.2912304 ],
       [ 0.51125056,  0.11482469, -0.12336681, ...,  0.24975561,
        -0.13215996,  0.1043145 ],
       [ 0.00538439, -0.21337465,  0.03742395, ...,  0.10487098,
         0.1115148 , -0.25606438],
       ...,
       [ 0.07168343, -0.29358794,  0.03180913, ..., -0.14731548,
        -0.18422158,  0.03025684],
       [ 0.49027344,  0.23323747,  0.17137094, ..., -0.29256026,
         0.20970722, -0.29413859],
       [ 0.02729893, -0.2423445 ,  0.11073375, ..., -0.00102432,
         0.00292775, -0.26232638]])
```

7.2.2.4 Biases:

Biases of the neurons in hidden layer

-0.252837386943, -0.0666725075512, 0.116649137253, -0.0238043202682, 0.0527459549111, -0.248184661718, 0.0437912742701, 0.0645854254081, 0.163877407767, -0.141044062593, -0.0339897493239, -0.109081296923, -0.142643429233, 0.238020756469, -0.103911221302, -0.160767008078, 0.17015287332, -0.1121735444, -0.169600390201, -0.0578292726241, -0.321998755734, -0.0817208678768, -0.278741107336, -0.233732509747

Biases of neurons in the output layer

0.0878468376162, 0.174037716449, -0.0654543205759, 0.0940845080677, -0.0345955296423, -0.00289930054019, 0.153771102287, -0.0820481962988, 0.0539497168476, -0.0656038353887, 0.144965310145, -0.0830429429164

7.2.2.5 Testing Results:

Testing set1 score = 0.914587553216

Testing set2 score = 0.990595107207

8. FEASIBILITY STUDY FOR ONBOARD IMPLEMENTATION

Onboard implementation of neural networks for early fault detection of actuators involves two main aspects.

1. Ground Development of reaction wheel model
2. Onboard implementation of wheel model

8.1 Ground Development of Reaction Wheel Model:

Wheel parameters are available at different frames in the satellite telemetry data with different sampling rates. The inaccuracies in the neural network due to the non-availability of these parameters at the same instant of time can be avoided if these parameters can be captured at the same instant of time for generating the training data.

For this an onboard facility can be developed to collect Wheel Speed, TCS, DFC Torque of all the four wheels at the *same instant of time at a higher sampling rate* (say 32ms).

The wheel neural network can be developed using this data as the training data during ground testing of OBC.

8.2 Onboard implementation of wheel model

The neural network developed after ground training and exhaustive validation is then implemented as part of satellite's onboard software for real time testing and residue generation.

The neural network parameters which include number of samples required for early fault prediction (N), number of neurons in the hidden layer (h), weights of the first layer (w[z][i]), biases of hidden neurons (theta_h[z]), weights of the second layer (v[j][z]), biases of output neurons (theta_y[j]) are arrived at after elaborate testing and validation.

The neural network logic to be implemented onboard is as follows. (Refer section 7.2)

Pseudocode:

```
N, h, i, j, z         : INTEGER
x                     : FLOAT ARRAY(1..12N)
y                     : FLOAT ARRAY(1..12)
w                     : 2D FLOAT ARRAY (1..h, 1..12N)
v                     : 2D FLOAT ARRAY (1..12, 1..h)
theta_h               : FLOAT ARRAY(1..h)
theta_y               : FLOAT ARRAY(1..12)
```

```
FOR j IN 1..12 LOOP
    Temp2 = 0
    FOR z IN 1..h LOOP
            Temp1 = 0
            FOR i IN 1..12N LOOP
                    Temp1 = Temp1 + x[i]*w[z][i]
            END LOOP
            Temp2 = Temp2 + v[j][z] * tanh(Temp1 + Theta_h[z])
    END LOOP
    y[j] = Temp2 + Theta_y[j]
END LOOP
```

The input to the Reaction Wheel model, x, is implemented as a FIFO array. At any instant of time t, the input vector x consists of N previous sample sets of wheel data. Each previous sample set contains wheel speed, TCS, DFC Torque of all the four wheels. The first previous sample set corresponds to x_{t-1}, the second previous sample set corresponds to x_{t-2}, the third previous sample set corresponds to x_{t-3} and so on.

The model output y_{t_pred} , is the current value of wheel speed, TCS, DFC Torque of all the four wheels. y_t is the true value of wheel speed, TCS, DFC Torque of all four wheels acquired by OBC software. The output vector y_{t_pred} is compared with y_t to generate the residual error. The residual error is used for analyzing the wheel performance and detecting any tendency of the reaction wheel system towards failure.

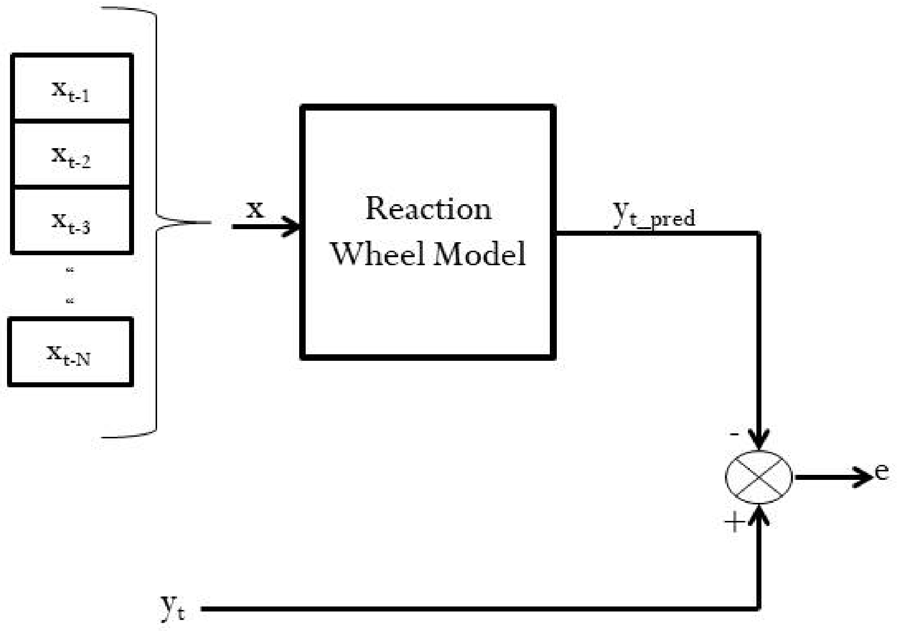


Figure 18: Model inputs at time t

At the next instant of time t+1, the Nth sample in the previous cycle in the input array is pushed out. The input vector now consists of x_t which is the first previous sample set, x_{t-1} which is the second previous sample set and so on till x_{t-N+1} which is the Nth previous sample set. The model outputs y_{t_pred+1} which is compared with y_{t+1} to generate the error.

Therefore the input vector can be implemented as a FIFO array where the Nth sample in the input array is pushed out in each cycle. The error generated is used for deciding the further course of action.

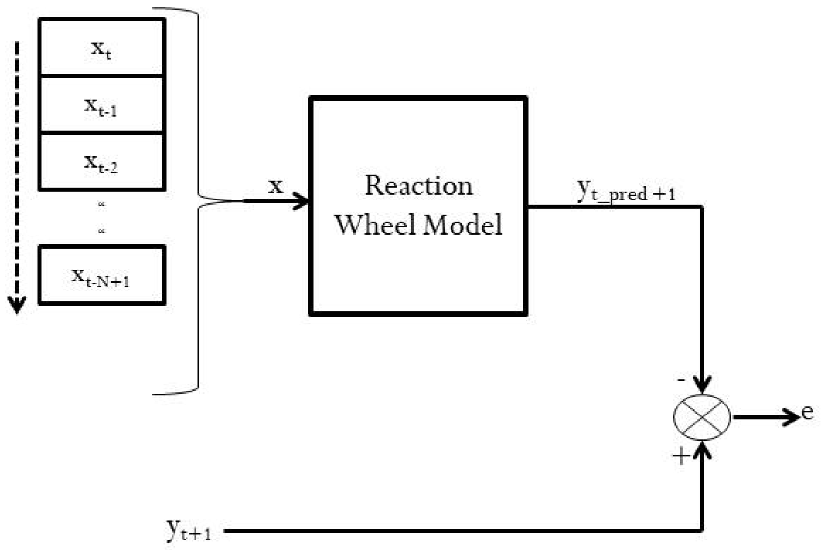


Figure 19: Model inputs at time t+1

9. SUMMARY

Two approaches are identified for developing reaction wheel model. In the first approach, four neural networks for the four Reaction wheels are developed. In the second approach, a combined model for the four wheels is developed. The telemetry data of Satellite_XYZ is used for training the wheel models in both the approaches. Based on the number of samples required for early prediction (user input), the telemetry data has to be extracted and processed to derive the feature set required for training the neural networks.

The networks are trained with wheel telemetry data during Normal mode when the wheel behavior is nominal. Further, they are tested with normal mode wheel telemetry data and pre-failure wheel telemetry data. It is seen that the individual wheel models are not responding to pre-failure data uniformly. However the combined wheel model gave a clear indication of failure with a negative MLPRegressor score, which is the coefficient of determination R^2, when fed with wheel telemetry data prior to actual failure. Therefore the combined wheel model is chosen for further refinement for early prediction of wheel failure. Also, the score of neural network is the criteria chosen for early failure detection.

Based on the second approach, the reaction wheel model is developed for prediction while maneuvering. The telemetry data of Satellite_PQR which undergoes frequent maneuvering due to its imaging requirement is used for training this model. Since each maneuvering is of short duration, telemetry data from several such maneuvering instances are combined to train the model. This model is further validated with test data.

Feasibility of onboard implementation is also worked out in this report. Two steps are proposed as part of onboard implementation- Development of the model during ground testing and implementing this model in onboard software for real time prediction.

Details of work done:

Sl No	Description of work	Status
1	Literature Study	Done
2	Develop feature set generation logic from telemetry data	Done
3	Develop and test neural network model for Reaction Wheel1 in Normal mode	Done
4	Develop and test neural network model for Reaction Wheel2 in Normal mode	Done
5	Develop and test neural network model for Reaction	Done

	Wheel3 in Normal mode	
6	Develop and test neural network model for Reaction Wheel4 in Normal mode	Done
7	Develop and test neural network model for combined wheel system in Normal mode	Done
8	Develop wheel model for prediction during maneuvering	Done
9	Feasibility study for onboard implementation	Done

10. BIBLIOGRAPHY

[1] Bhekisipho Twala, “Predicting software faults in Large Space Systems using Machine Learning Techniques,” *Defence Science Journal*, Vol. 61, No. 4, July 2011

[2] Dan W. Patterson, *Artificial Neural Networks Theory and Applications,* Singapore, Prentice Hall, 1996

[3] Ethem Alpaydın, Introduction to Machine Learning, Second Edition, MIT Press, 2010

[4] Takehisa Yairi, Yoshinobu Kawahara, Ryohei Fujimaki, Yuichi Sato, and Kazuo Machida, “Telemetry-mining: A Machine Learning Approach to Anomaly Detection and Fault Diagnosis for Space Systems,” in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, 2006, paper.

11. REFERENCES

[1] Amy McGovern and Kiri L. Wagstaff, *Machine learning in space: extending our reach*, Springer, 30 April 2011, editorial published online

[2] Hecht-Nielsen, R., 1987, Kolmogorov's mapping neural network existence theorem. In IEEE First Annual International Conference on Neural Networks, 3, pp. 11–13.

[3] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014)

[4] sklearn.neural_network.multilayer_perceptron.MLPRegressor documentation

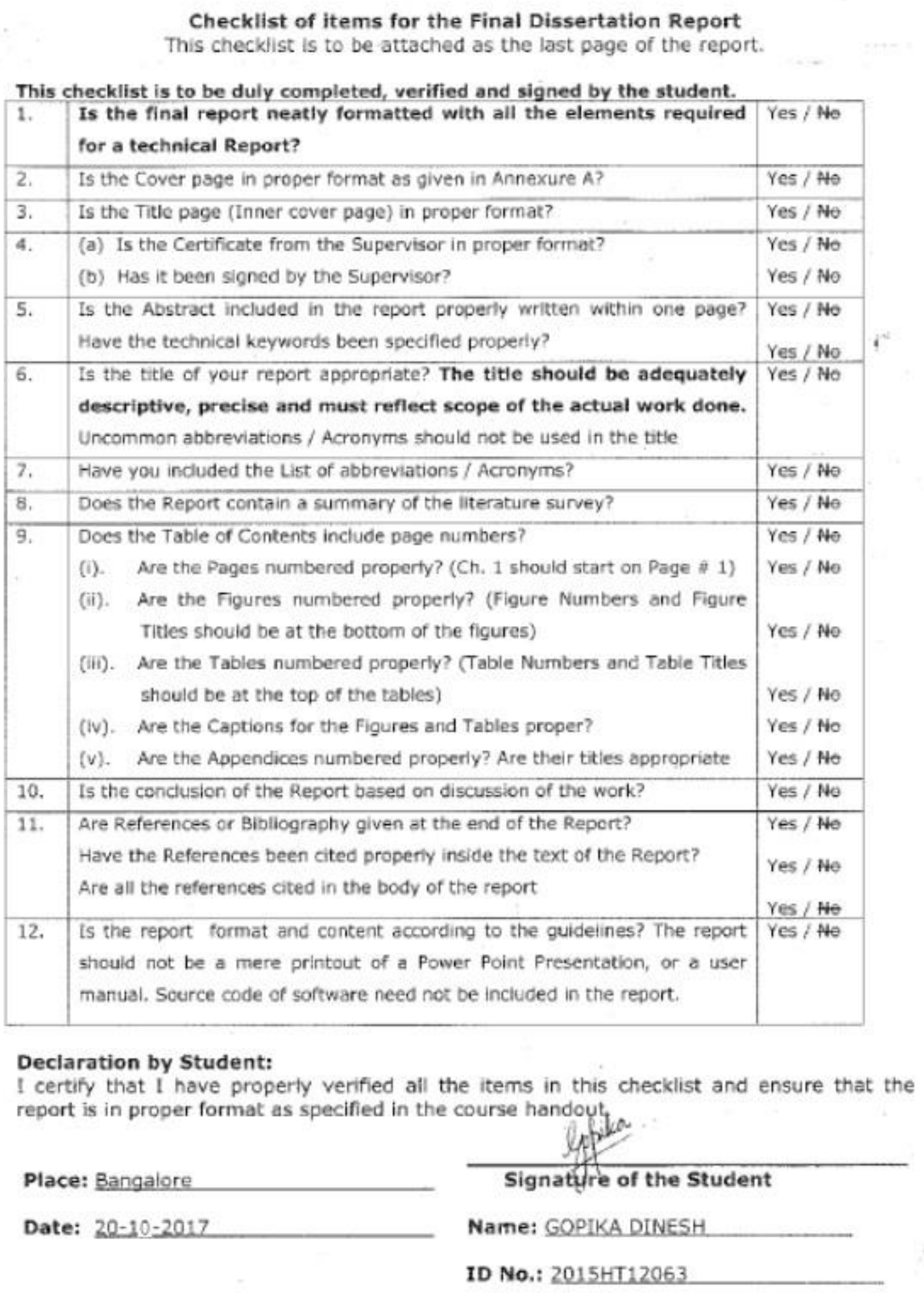
[5] Wikipedia, *Limited-memory_BFGS*, Retrieved from https://en.wikipedia.org/wiki/Limited-memory_BFGS

12. APPENDIX

Table 3: List Of Abbreviations

Abbreviation	**Description**
ML	Machine Learning
ANN	Artificial Neural Networks
NN	Neural Networks
AOCS	Attitude and Orbit Control System
OBC	Onboard Computer
RW	Reaction Wheels
MLP	Multi Layer Perceptron
FDIR	Fault Detection Isolation Recovery
WDE	Wheel Drive Electrons
TCS	Torque Command Signal
DFC	Dynamic Friction Compensation
LPF	Low Pass Filter
TOC	Torque Output Control
NSC	Nominal Speed Control
AI	Artificial Intelligence
ADAM	Adaptive Moment Estimation
SGD	Stochastic Gradient Descent
LBFGS	Limited Memory Broyden–Fletcher–Goldfarb–Shanno

13. CHECKLIST



Checklist of items for the Final Dissertation Report

This checklist is to be attached as the last page of the report.

This checklist is to be duly completed, verified and signed by the student.

1.	**Is the final report neatly formatted with all the elements required for a technical Report?**	Yes / ~~No~~
2.	Is the Cover page in proper format as given in Annexure A?	Yes / ~~No~~
3.	Is the Title page (Inner cover page) in proper format?	Yes / ~~No~~
4.	(a) Is the Certificate from the Supervisor in proper format?	Yes / ~~No~~
	(b) Has it been signed by the Supervisor?	Yes / No
5.	Is the Abstract included in the report properly written within one page?	Yes / ~~No~~
	Have the technical keywords been specified properly?	Yes / No
6.	Is the title of your report appropriate? **The title should be adequately descriptive, precise and must reflect scope of the actual work done.** Uncommon abbreviations / Acronyms should not be used in the title	Yes / No
7.	Have you included the List of abbreviations / Acronyms?	Yes / ~~No~~
8.	Does the Report contain a summary of the literature survey?	Yes / ~~No~~
9.	Does the Table of Contents include page numbers?	Yes / ~~No~~
	(i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1)	Yes / No
	(ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures)	Yes / ~~No~~
	(iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables)	Yes / ~~No~~
	(iv). Are the Captions for the Figures and Tables proper?	Yes / No
	(v). Are the Appendices numbered properly? Are their titles appropriate	Yes / ~~No~~
10.	Is the conclusion of the Report based on discussion of the work?	Yes / No
11.	Are References or Bibliography given at the end of the Report?	Yes / ~~No~~
	Have the References been cited properly inside the text of the Report?	Yes / ~~No~~
	Are all the references cited in the body of the report	Yes / ~~No~~
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes / ~~No~~

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Gopika

Signature of the Student

Place: Bangalore

Date: 20-10-2017

Name: GOPIKA DINESH

ID No.: 2015HT12063