University of New Hampshire
University of New Hampshire Scholars' Repository

Master's Theses and Capstones | Student Scholarship

Fall 2011

Accelerometer calibration for NASA's magnetospheric multiscale mission spacecraft

Benjamin M. Jenkins
University of New Hampshire, Durham

Follow this and additional works at: https://scholars.unh.edu/thesis

Recommended Citation
Jenkins, Benjamin M., "Accelerometer calibration for NASA's magnetospheric multiscale mission spacecraft" (2011). *Master's Theses and Capstones*. 659.
https://scholars.unh.edu/thesis/659

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.

ACCELEROMETER CALIBRATION FOR NASA's MAGNETOSPHERIC MULTISCALE MISSION SPACECRAFT

BY

BENJAMIN M JENKINS

B.S., University of New Hampshire, 2009

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Mechanical Engineering

September, 2011

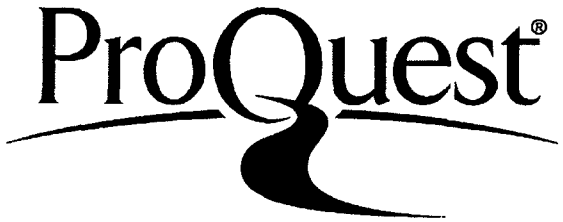
UMI Number: 1504951

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI 1504951
Copyright 2011 by ProQuest LLC.
All rights reserved. This edition of the work is protected against
unauthorized copying under Title 17, United States Code.

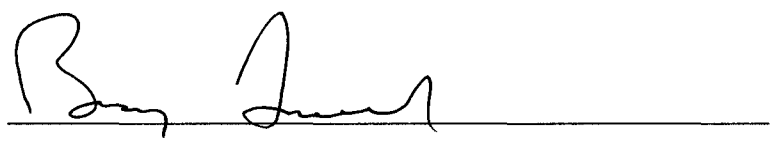


ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

This thesis has been examined and approved.

Thesis Director, Dr. May-Win L. Thein

Associate Professor of Mechanical Engineering



Dr. Barry Fussel

Professor of Mechanical Engineering

Dr. L. Gordon Kraft, III

Professor of Electrical and Computer Engineering

8/15/2011

Date

Acknowledgements

I would like to thank all those who have supported me through this thesis and my six years here at the University:

My thesis and academic advisor May-Win Thein, who introduced me to the field of spacecraft attitude estimation and control. Without her support and guidance this project would never have come to be.

Professor Barry Fussell and Professor Gordon Kraft for being part of my thesis committee, and for their outstanding teaching and advice for my future.

The NH Space Grant Consortium, Mechanical Engineering Department, Graduate School and NASA MMS Grants for their financial support over the past two years.

The MMS Attitude Control System Group of the Goddard Space Flight Center, specifically: Josephine K. San, Dean J. Chai, Samuel J. Placanica, David J. Mangus, Juan C. Raymond and Julie K. Thienel

My coaches Jim Boulanger and Robert Hoppler who guided me through five years of running and life.

Finally to my family for their constant support and encouragement, who, for better or for worse, made me who I am today.

I would like to dedicate this work to Aimee, who helped so much and put up with me throughout.

Table of Contents

Acknowledgements . . . . . **iii**

List of Tables . . . . . **viii**

Variables and Units . . . . . **ix**

List of Figures . . . . . **x**

Abstract . . . . . **xv**

INTRODUCTION **1**

1 Past Work 5

2 Sensors 7

3 Sources of Error 9

4 Methods 10

5 Thesis Outline 11

I SPACECRAFT ATTITUDE DYNAMICS AND KINEMATICS 13

1 Euler Angle Representation 13

2 Direct Cosine Matrix 15

2.1 Small Angle Approximations 15

3 Quaternion Representation 16

3.1 Quaternion Functions 17

4 Attitude Representation Conversions 19

5 Spacecraft Kinematics and Dynamics 20

6 Sensor Models 21

II LUMPED BIAS JUSTIFICATION 24

1 Analytical Justification 24

2 Numerical Validation of the Lumped Bias Model 26

III ESTIMATION TECHNIQUES 32

1 Cascading Filter 33

2 Dynamic Filters/Estimators 35

2.1 Kalman Filter 36

2.2 Extended Kalman Filter 38

2.3 H_∞ Filter 42

2.4 Sliding Mode Observer 45

3 Batch Calibration Routine 47

IV ANALYTICAL SIMULATIONS 50

1 Simulation Conditions 50

1.1 Simulation Inertia Tensor 52

1.2 Sensor Output and Simulation 53

1.3 Single and Multi-rate Filter Results 54

1.3.1 4Hz Measurement and Propagation 54

1.3.2 4Hz Measurement Updates and 400Hz Propagation Rate 56

2 Analytical Filter/Estimation Results 59

2.1 Cascading filter results 59

2.2 Known Inertia Tensor Results 62

2.2.1 Extended Kalman Filter for Known Inertia Tensor . 63

2.2.2 H_∞ Filter for Known Inertia Tensor 68

2.2.3 Sliding Mode Observer 72

2.2.4 Summary of Results for Known Inertia Tensor 77

2.3 Corrupt Inertia Tensor Results 78

2.3.1 EKF Results for Corrupt Inertia Tensor 79

2.3.2 H_∞ Filter Results for Corrupt Inertia Tensor 81

2.4 SMO Results for Corrupt Inertia Tensor 83

2.4.1 Summary of Filter Results for Corrupt Inertia Tensor 86

2.5 Batch Filter Results 87

2.5.1 Known Inertia Tensor Results 88

2.5.2 Corrupt Inertia Tensor Results 92

2.5.3 Batch Filter Tabulated Results 94

V EXPERIMENTAL VERIFICATION 95

1 NASA MMS TableSat Generation I 95

1.1 Experimental Hardware 97

1.2 Experimental Software 98

2 Experimental Results 100

2.1 Accelerometer Pre-Calibration 102
2.2 Experimental Lumped Accelerometer Bias Estimation 103
2.2.1 Neutral δ_y 105
2.2.2 Positive δ_y 108
2.2.3 Negative δ_y 110
2.3 Summary of Experimental Results 112
VI CONCLUSIONS 113
1 Future Work for Filters/Estimators 115
2 Future Work for the Experimental Test Bed 116
LIST OF REFERENCES 118
Appendix A 125
1 Additional Dynamic Filter Results 125
2 Euler Angle Rotation Matrices 133
3 Microcontroller Code 135
Appendix B - Matlab Code 139
1 Constants.m 139
2 gen_sim_data.m 140
3 file_00010_EKF09.m 142
4 run_efk46propR.m 147
5 run_hinf26_propR.m 149
6 run_avgekf43_smo05.m 151

List of Tables

4.1 Means of the error variances of 100 trials (known inertia) - EKF . . . 67

4.2 RMS of the mean errors for 100 trials (known inertia) - EKF 68

4.3 Means of the error variances of 100 trials (known inertia) - H_∞ 71

4.4 RMS of the mean errors for 100 trials (known inertia) - H_∞ 71

4.5 Means of the error variances of 100 trials (known inertia) - SMO . . . 76

4.6 RMS of the mean errors for 100 trials (known inertia) - SMO 76

4.7 Summary of the means of the error variances for filters with known inertia 77

4.8 Summary of the RMS of the mean errors for filters with known inertia 78

4.9 Mean of the error variances of 100 trials (uncertain inertia) 86

4.10 RMS of the mean errors for 100 trials (uncertain inertia) 87

4.11 Batch lumped bias estimation for all corruptions & no noise 90

4.12 Batch lumped bias estimation for all corruptions with noise 92

4.13 Batch lumped bias estimation for all corruptions with noise and inertia uncertainty 94

4.14 Batch lumped bias estimation summary 94

5.1 Lumped accelerometer bias trend for decreasing δ_y 112

Variables and Units

Variable Name	Symbol	Units
Full quaternion	$\bar{\mathbf{q}}$	-
Quaternion imaginary vector	$\mathbf{q}$	-
Quaternion scalar component	$\mathbf{q_0}$	-
Spacecraft angular rates	$\boldsymbol{\omega}$	rad/s
Time derivative of full quaternion	$\dot{\bar{\mathbf{q}}}$	-
Time derivative of spacecraft angular rates	$\dot{\boldsymbol{\omega}}$	rad/s^2
Angular rate augmented with zero for quaternion operators	$\bar{\omega}_{b/o}$	rad/s^2
True acceleration at location of accelerometer	$\boldsymbol{a}_{true}$	m/s^2
Acceleration where accelerometer is thought to be and oriented	$\boldsymbol{a}_{nom}$	m/s^2
Measured acceleration (a_{true} + noise)	$\boldsymbol{a}_m$	m/s^2
Acceleration at true accelerometer location, aligned to body axis	$\boldsymbol{a}_{shift}$	m/s^2
Acceleration at true accelerometer location, aligned to accelerometer axis	$\boldsymbol{a}_{mis}$	m/s^2
Accelerometer sensor bias	$\boldsymbol{a}_b$	m/s^2
Shift in accelerometer location from measured location	$\boldsymbol{\delta r}$	m
accelerometer axes orthogonal rotation defined by a small angle 1-2-3 Euler angle sequence	$\boldsymbol{\delta}$	rad
Rotation matrix defining accelerometer axes orthogonal rotation	$\boldsymbol{\Delta}$	-
Lumped bias correction term	$\boldsymbol{a}_{LB}$	m/s^2
Spacecraft true inertia tensor	$\boldsymbol{J}$	$kg-m^2$
Nominal (model) spacecraft inertia tensor	$\hat{\boldsymbol{J}}$	$kg-m^2$
Measured accelerometer location prior to launch	$\boldsymbol{r}$	m
Accelerometer white noise	$\boldsymbol{\nu}_{accelerometer}$	m/s^2
Torques on spacecraft	$\boldsymbol{T}$	$N-m$
State covariance matrix	$\boldsymbol{Q}$	-
Sensor noise covariance matrix	$\boldsymbol{R}$	-

List of Figures

1 Earth's magnetosphere [5] 1

2 Tetrahedron formation [7] 2

3 Rendering of MMS spacecraft with extended booms (to scale) [24] . . 3

4 Internal layout of spacecraft 7

1.1 Euler angles. 3-2-3 rotation $[\alpha, \beta, \gamma]^T$. 14

2.1 Numerically predicted bias from $a_{b,x}$ component 27

2.2 Numerically predicted bias from $\delta r_x = 1cm$ 28

2.3 Numerically predicted bias from $\delta r_y = 1cm$ 28

2.4 Numerically predicted bias from $\delta r_z = 1cm$ 29

2.5 Numerically predicted bias from $\delta_x = 20arcsec$ 30

2.6 Numerically predicted bias from $\delta_y = 20arcsec$ 30

2.7 Numerically predicted bias from $\delta_z = 20arcsec$ 31

3.1 Saturation function 46

3.2 Quaternion fit and derivative lines 48

4.1 Simulink dynamic simulation 51

4.2 Simulink subsystem: plant 51

4.3 EKF estimated quaternion for 1x propagation 55

4.4 EKF estimated body rates for 1x propagation 55

4.5 EKF estimated accelerometer bias for 1x propagation 56

4.6 EKF estimated quaternion for 100x propagation 57

4.7 EKF estimated body rates for 100x propagation 58

4.8 EKF estimated accelerometer bias for 100x propagation 58

4.9 RMS misalignment error due to RMS of accelerometer bias error . . . 60

4.10 Percent misalignment error due to RMS of accelerometer bias error . 60

4.11 RMS misalignment error due to RMS of center of mass error 61

4.12 Percent misalignment error due to RMS of center of mass error 61

4.13 EKF $\hat{q}$ subjected to bias errors and measurement noise (known inertia) 65

4.14 EKF $\hat{\omega}$ subjected to bias errors and measurement noise (known inertia) 65

4.15 EKF $\hat{a}_{LB}$ subjected to bias errors and measurement noise (known inertia) 66

4.16 H_∞ $\hat{q}$ subjected to bias errors and measurement noise (known inertia) 69

4.17 H_∞ $\hat{\omega}$ subjected to bias errors and measurement noise (known inertia) 70

4.18 H_∞ $\hat{a}_{LB}$ subjected to bias errors and measurement noise (known inertia) 70

4.19 SMO $\hat{q}$ subjected to bias errors and measurement noise (known inertia) 73

4.20 SMO $\hat{\omega}$ subjected to bias errors and measurement noise (known inertia) 74

4.21 SMO $\hat{a}_{LB}$ subjected to bias errors and measurement noise (known inertia) 75

4.22 EKF $\hat{q}$ subjected to bias errors and measurement noise (unknown inertia) 79

4.23 EKF $\hat{\omega}$ subjected to bias errors and measurement noise (unknown inertia) 80

4.24 EKF $\hat{a}_{LB}$ subjected to bias errors and measurement noise (unknown inertia) 81

4.25 H_∞ $\hat{q}$ subjected to bias errors and measurement noise (unknown inertia) 82

4.26 H_∞ $\hat{\omega}$ subjected to bias errors and measurement noise (unknown inertia) 82

4.27 H_∞ $\hat{a}_{LB}$ subjected to bias errors and measurement noise (unknown inertia) 83

4.28 SMO $\hat{q}$ subjected to bias errors and measurement noise (unknown inertia) 84

4.29 SMO $\hat{\omega}$ subjected to bias errors and measurement noise (unknown inertia) 84

4.30 SMO $\hat{a}_{LB}$ subjected to bias errors and measurement noise (unknown inertia) 85

4.31 BF quaternion estimation with 3-corruptions and noise (known inertia) 88

4.32 BF angular rate estimation with 3-corruptions and noise (known inertia) 89

4.33 BF lumped accelerometer bias estimation with 3-corruptions and noise (known inertia) 89

4.34 BF quaternion estimation with 3-corruptions and noise (known inertia) 90

4.35 BF angular rate estimation with 3-corruptions and noise (known inertia) 91

4.36 BF lumped accelerometer bias estimation with 3-corruptions and noise (known inertia) 91

4.37 BF quaternion estimation with 3-corruptions and noise (unknown inertia) 92

4.38 BF angular rate estimation with 3-corruptions and noise (unknown inertia) 93

4.39 BF lumped accelerometer bias estimation with 3-corruptions and noise (unknown inertia) 93

5.1 TableSat I 96

5.2 TableSat I components 96

5.3 TableSat for experimental accelerometer calibration - View 1 97

5.4 TableSat for experimental accelerometer calibration - View 2 101

5.5 TableSat for experimental accelerometer calibration - View 3 101

5.6 Settled model gyro output 102

5.7 Accelerometer signal for initial calibration 103

5.8 Experimental accelerometer neutral rotation about y-axis 104

5.9 Experimental accelerometer positive rotation about y-axis 104

5.10 Experimental accelerometer negative rotation about y-axis 105

5.11 Filtered quaternion from first spline 106

5.12 Angular rate estimate from first spline 107

5.13 Predicted and measured acceleration with error 107

5.14 Filtered quaternion from first spline 108

5.15 Angular rate estimate from first spline 109

5.16 Predicted and measured acceleration with error 109

5.17 Filtered quaternion from first spline 110

5.18 Angular rate estimate from first spline 111

5.19 Predicted and measured acceleration with error 111

6.1 $\hat{q}$ subjected to bias errors and no measurement noise (known inertia) 125

6.2 $\hat{\omega}$ subjected to bias errors and no measurement noise (known inertia) 125

6.3 $\hat{a}_{LB}$: subjected to bias errors and no measurement noise (known inertia) 126

6.4 $\hat{q}$ subjected to bias errors and 10x-measurement noise (known inertia) 126

6.5 $\hat{\omega}$ subjected to bias errors and 10x-measurement noise (known inertia) 127

6.6 $\hat{a}_{LB}$ subjected to bias errors and 10x-measurement noise (known inertia) 127

6.7 $\hat{q}$ subjected to bias errors and no measurement noise (known inertia) 128

6.8 $\hat{\omega}$ subjected to bias errors and no measurement noise (known inertia) 128

6.9 $\hat{a}_{LB}$ subjected to bias errors and no measurement noise (known inertia) 129

6.10 $\hat{q}$ subjected to bias errors and 10x-measurement noise (known inertia) 129

6.11 $\hat{\omega}$ subjected to bias errors and 10x-measurement noise (known inertia) 130

6.12 $\hat{a}_{LB}$ subjected to bias errors and 10x-measurement noise (known inertia) 130

6.13 $\hat{q}$ subjected to bias errors and no measurement noise (known inertia) 131

6.14 $\hat{\omega}$ subjected to bias errors and no measurement noise (known inertia) 131

6.15 $\hat{a}_{LB}$ subjected to bias errors and no measurement noise (known inertia) 132

6.16 $\hat{q}$ subjected to bias errors and 10x-measurement noise (known inertia) 132

6.17 $\hat{\omega}$ subjected to bias errors and 10x-measurement noise (known inertia) 133

6.18 $\hat{a}_{LB}$ subjected to bias errors and 10x-measurement noise (known inertia) 133

ABSTRACT

ACCELEROMETER CALIBRATION
FOR NASA's MAGNETOSPHERIC
MULTISCALE MISSION SPACECRAFT

by

Benjamin Jenkins

University of New Hampshire, September, 2011

This thesis presents several methods for the on-board and/or ground-based calibration of accelerometers for the spacecraft (s/c) of the NASA Magnetospheric Multi-Scale (MMS) Mission during mission operation. A lumped bias is estimated to correct for the total effect of the MMS accelerometer sensor bias, orthogonal misalignment and the shift in the s/c's center of mass.

Various estimation techniques are evaluated and compared, including both dynamically driven real-time filters/observers and post processing batch algorithms. Both methods are shown to accurately determine lumped bias, so long as the s/c inertia tensor is well known. If, however, there is any uncertainty in the inertia tensor, only post processing methods yield accurate lumped bias estimates. Analytical simulations show that these methods are able to correct accelerometer readings to within 1 micro-g of true acceleration. Preliminary experimental verification also shows proof of concept.

INTRODUCTION

The NASA Magnetospheric MultiScale (MMS) mission is expected to launch in 2014. The mission purpose is to explore three fundamental plasma processes in Earth's magnetosphere: magnetic reconnection, energetic particle acceleration and turbulence.

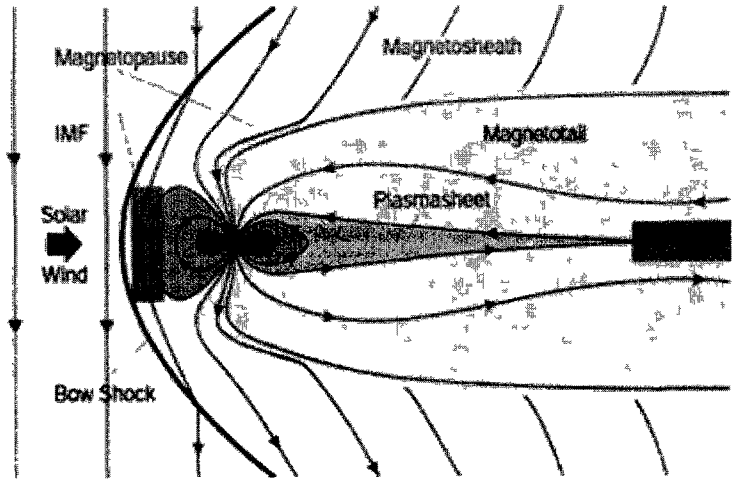


Figure 1: Earth's magnetosphere [5]

Since the plasma processes being investigated are four-dimensional (three spatial dimensions and a time dimension), in order to fully understand their physics, measurements must be made at a minimum of four locations at all times. To satisfy this requirement, the MMS mission will consist of a constellation of four identical spacecraft (s/c), flying in a precise tetrahedron formation.

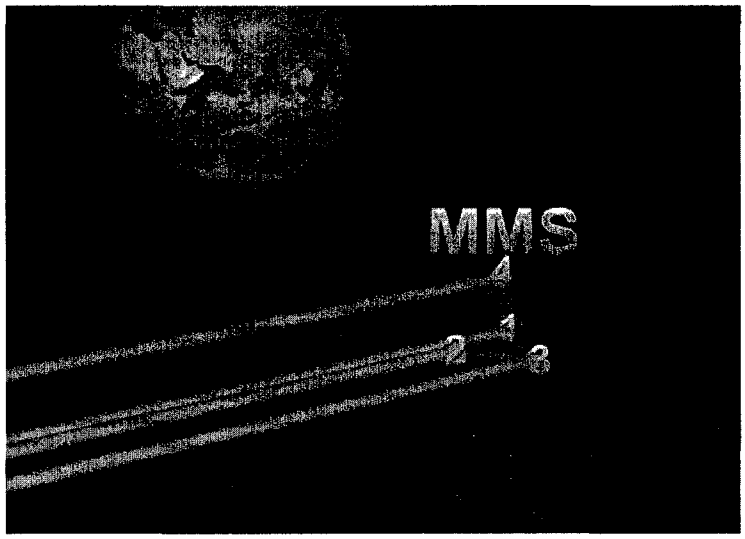


Figure 2: Tetrahedron formation [7]

The European Space Agency (ESA) already launched a similar mission named Cluster, which is investigating the Earth's magnetic environment and its interaction with the solar wind in three dimensions [1]. The MMS mission will build upon Cluster's success with substantially higher resolution data, both spatially and temporally. This higher resolution data will be collected by each of the spin-stabilized spacecraft, each having four 60m spin-plane Double-Probe (SDP) wire booms extending out radially and two 13m truss-type structure Axial-Plan Double-Probe (ADP) booms extending out axially.

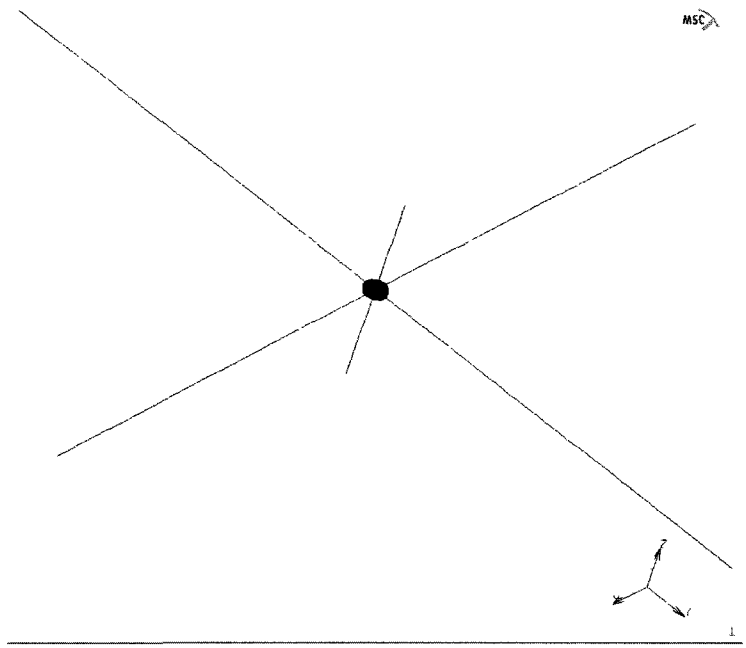


Figure 3: Rendering of MMS spacecraft with extended booms (to scale) [24]

A very large area of space can be monitored by four spinning spacecraft. Formations and spinning, however, add complications to the mission. Since four individual spacecraft will be used, the location of each must be known very accurately in order to compile data effectively. Additionally, the spin rate of each s/c must be precisely known as well. There are no on-board sensors (i.e., gyros) to directly measure spin rate. Estimation algorithms will be used to determine the spin rate from indirect measurements of attitude and acceleration, as well as knowledge of the system's dynamic/kinematic equations. State estimation for a similar spacecraft has been presented in [17] and for this specific MMS mission by Thienel in [39] and [40]. Recent developments, however, in stricter mission requirements and sensor findings necessitated the estimation of a total of three parameters. Orthogonal misalignment, in

addition to accelerometer sensor bias and accelerometer shift in location with respect to center of mass, is now required.

Accelerometer calibration is essential to allow accurate attitude estimation throughout the mission. The location of the accelerometer relative to the center of mass and its orthogonal alignment with the body axes are expected to shift due to launch vibrations and sun/shade transitions during orbit. The presented calibration methods intend to correct these errors that will develop. Formation missions necessitate extremely high precision controllers to minimize wasted fuel and obtain accurate scientific data and for these controllers to perform as they were designed they must have accurate sensor measurements and estimated states.

The addition of this new parameter to the two existing parameters contribute a total of nine augmented states that need to be estimated since each consists of a three-component vector. In combination with the six state variables to fully define attitude and angular rates, the estimation problem requires a 15^{th} order state estimator, with only six measured states. The questionable observability of these states and parameters has led to the investigation of the feasibility of using a single lumped accelerometer bias to replace the nine accelerometer calibration parameters with only three parameters. The result is a 9^{th} order state estimation algorithm.

This thesis will present the various methods used to calibrate the accelerometer in order to provide the most accurate measurements possible. These methods will be compared using simulated and experimental data. Additionally, both dynamic and batch estimators will be developed to estimate the 9 states and 3 parameters proposed to be necessary.

1 Past Work

A review of the literature surrounding this topic yielded many papers presenting portions of the work evaluated here or concepts applied to alternative missions. Although much research has been done on portions of concepts applied to alternative missions, there is no existing research found that studies the multiple techniques for lumped bias estimation or even a justification of practicality of determining the lumped bias.

Thienel and Markley's Paper on MMS state estimation [39] and their second paper written with Harman [40] present the original MMS accelerometer calibration approach. They presented an Extended Kalman Filter (EKF) for the estimation of attitude, angular rates, accelerometer bias and center of mass shift. Their work did not, however, account for accelerometer misalignment. This study strives to avoid some of the assumptions that are made in prior research. Additional work performed on MMS-specific observers includes the comparative study of various observer based control techniques by Mushaweh et. al. [23] [24]. A limitation of this paper was that the observers only estimated attitude and body rates. Koprubasi and Thein designed a Sliding Mode Observer based on the mission requirements of the CATSat satellite, presenting an EKF additionally [14] - [17]. CATSat was a very different spacecraft, with no accelerometer calibration requirement, however, many of fundamental observer concepts and design apply to this study.

A common method for ground based accelerometer calibration involves using cen-

trifuges [20]. Spinning an accelerometer at a known speed will induce a known radial acceleration on the accelerometer, allowing for it to be properly calibrated [27]. Since the MMS spacecraft will be spinning, a similar effect is produced which can be used to calibrate the accelerometer. Orthogonal misalignment is easily developed for accelerometers and other vector measurement devices. These vector measurements often play an important role in the spacecraft's state estimation, and require calibration to eliminate or reduce the unknown misalignment. Attitude estimation is both heavily dependent on the accelerometer yet used for the accelerometer's misalignment estimation in [27]. This was accomplished by using an off-line iterative algorithm using multiple accelerometers.

Shuster has authored several papers describing methods to determine sensor misalignment with and without knowledge of the spacecraft states [33] [42]. Many of the misalignment algorithms he presents involve aligning like sensors, such as attitude sensors with other attitude sensors, or accelerometers with accelerometers, which does not directly correlate to this work. It provides a solid foundation to build upon for misalignment determination. Shuster also published an extremely thorough work presenting nearly every attitude description possible [31].

If there is very little knowledge about the expected dynamics of a system, then spline fitting may provide an effective method to attenuate noise and/or determine the derivative at a point. Park et. al. use polynomial fitting to interpolate positions in [26] and Lee successfully used splines to smooth position and velocity information [19]. Although there is substantial knowledge of the model for this project, the lumped bias term has been found to have a very high sensitivity to the inertia tensor. Splines

are proposed to provide an alternative to the dynamic filters, avoiding all dependence of the state estimates on the inertia tensor.

2 Sensors

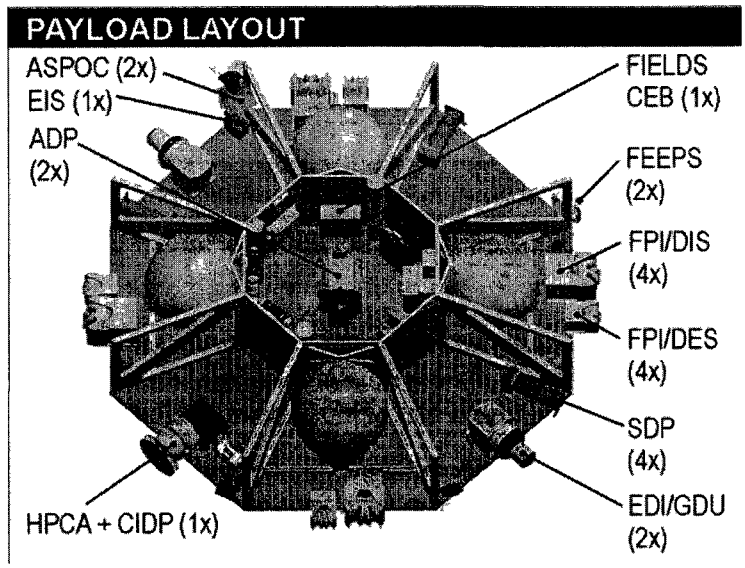


Figure 4: Internal layout of spacecraft

The three sensors typically used for state determination on spacecraft are star trackers, gyroscopes and accelerometers. The combination of these three sensors provide direct measurements to define the typical spacecraft states such as attitude and angular spin rate. Not all spacecraft will use all three of these measurements but a dynamic filter can be used to estimate the unmeasured states. Whether measured or not, knowledge of every state is required for control of the spacecraft.

A star tracker is an opto-electronic instrument used to provide the absolute three-

axis attitude of a spacecraft utilizing star observations. They employ a vision based system which takes a picture of the stars in its line of sight. It then compares this picture to a data bank of star maps to determine the attitude of the spacecraft with respect to a designated inertial frame. Star trackers are relatively accurate devices, however, they cannot provide measurements when pointing towards the Sun, Earth or Moon. In their paper presenting the calibration of a star tracker, Yang et. al. indicate that the uncertainties of the measured star direction vectors can be calibrated to within 4.0×10^{-5} rad of accuracy [4].

Gyroscopes directly measure the rotational velocity of the spacecraft. They are commonly manufactured in one of two ways: the mechanical gyro and the micro-electrical-mechanical system (MEMS) device. The mechanical gyro consists of a spinning disc or ring mounted on gimbals. The rotational inertia of the disk is substantially higher than the friction from the gimbals, allowing the disk to maintain its original attitude after its base has rotated. A second, more recent development of the gyro, is a MEMS device. These small electronics require much less electricity and are smaller and lighter. A drawback is that they are not as accurate or as reliable. Due to this, they are not typically used for spacecraft. If they are used, it is typically for smaller crafts with less stringent attitude constraints. A MEMS gyroscope has been used for the experimental verification in this thesis in Chapters V and ??. Although a gyro is included in the experimental test bed for this research, there will not be a gyro onboard the MMS spacecraft. Instead the angular rates will be estimated using the system model and star tracker and accelerometer measurements.

Accelerometers directly measure the acceleration of the spacecraft. Depending on

where they are located on the spacecraft they may be measuring only the translational acceleration of the spacecraft or may be measuring a combination of translational and rotational accelerations. Utilized by an inertial navigation system unit, accelerometers allow for "dead reckoning", which allows for the estimation of the position, attitude and velocity of an object without any external references.

Spinning spacecraft will often fly missions without gyros. This reduces launch costs since this cost is spacecraft is proportional to that of the spacecraft mass. It also reduces the power requirements of the spacecraft and eliminates a relatively high failure rate device from the design. NASA has chosen to fly the MMS mission without gyros, thus requiring an observer to estimate the spacecraft angular rates. Estimating angular rates does not present a large problem. Intensive accelerometer calibrations are usually not performed for spacecraft with such limited sensors.

3 Sources of Error

Multiple corruptions are included in the analytical simulations in this thesis to better represent the true MMS spacecraft. The corruptions tax each filter differently and sometimes violate the basic assumptions required for each particular filter. These corruptions test the robustness of each, albeit possibly degrading the performance of an optimally designed filter. Random numbers from a Gaussian distribution are included on the start tracker quaternion measurement and the accelerometer sensor output to simulate noise. Modeling uncertainty is incorporated by altering the inertia tensor. Three 3-component parameters are used to corrupt the accelerometer

output. A constant offset is used to simulate the sensors bias. The location of the accelerometer relative to the s/c center of mass is altered to simulate fuel usage, launch vibrations and thermal shifting. The three accelerometer measurement axes are assumed to always be orthogonal and a small orthogonal angle misalignment from the s/c body axes is introduced.

Various combinations of these corruptions are used to evaluate estimator performance and determining the robustness of each filter to these parametric uncertainties.

4 Methods

Three inherently different types of estimators are presented in this thesis. The first technique for accelerometer calibration is a cascading filter which estimates all nine terms of accelerometer calibration. The first six terms of accelerometer calibration as well as the attitude's six state variables were determined using an Extended Kalman Filter (EKF) designed by Thienel and presented in [39] and [40]. Using outputs from her filter and the equations required to describe the acceleration vector, the misalignment is calculated using a Least Squares Algorithm.

The second approach presented consists of dynamic filters/observers such as the Extended Kalman Filter, the H_∞ filter and the Sliding Mode Observer to estimate the six attitude states as well as a 3-parameter lumped bias accelerometer calibration term. Kalman and H_∞ filters both provide estimations based on some optimality criterion. The Sliding Mode Observer, although not defined as optimal, can be proven stable for bounded uncertainties. These dynamic filters are designed to propagate in

real-time onboard the spacecraft or via a ground station.

The last approach presented is designed to post-process measurement data (or run onboard a finite number of measurement samples behind the actual system). This approach is robust since it makes no assumptions about the system model other than the basic kinematic equations which describe a rotating body. It is assumed that all system state variables are smooth and continuous.

All methods are analytically tested. Only the post-process estimation routine is tested using experimental data.

5 Thesis Outline

This thesis is organized as follows:

- Chapter I, Spacecraft Attitude Dynamics and Kinematics - The equations describing s/c motion are presented and three attitude representations are described: Euler Angles, Direct Cosine Matrix and quaternions. Sensor models are also described.

- Chapter II, Lumped Bias Justification - Analytical and numerical justifications are made for the simplification of sensor bias, shift in center of mass and orthogonal misalignment to a single 3-element lumped bias.

- Chapter III, Estimation Techniques - 3 structurally independent filter categories are presented: a cascading filter built upon an existing EKF, three dynamic filters (EKF, H_∞ Filter and SMO), and a batch filter designed for robustness

when the inertia tensor is not known.

- Chapter IV, Analytical Simulations - The MatLab simulation setup and results from all filters described in Chapter III are presented.

- Chapter V, Experimental Verification - The batch filter is qualitatively verified using an experimental testbed.

- Chapter VI, Conclusions - The work presented is summarized and conclusions are drawn. Future work is suggested.

Chapter I

SPACECRAFT ATTITUDE DYNAMICS AND KINEMATICS

There are many representations that can be used to define a spacecraft's attitude. Common parameter selections can be divided into rank-3 definitions which are all susceptible to gimbal lock (singularities at specific attitudes) or 4-component tensor representations which include a redundant term but avoid singularity issues [21]. Shuster provides a detailed description of applicable attitude representations in [31]. The following section briefly introduces Euler Angles and the Direct Cosine Matrix but focuses on the description of quaternions, the attitude representation used for this work.

1 Euler Angle Representation

Euler angles are a common and intuitive description of attitude but are susceptible to gimbal-lock singularities. To be able to describe all attitudes accurately, at least two different Euler Angle sequences must be employed to maintain continuity if a singularity is encountered [31]. When two sets are used, more redundant data is produced than is produced when using quaternions.

A theorem by Euler states: [18] -

Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis.

Essentially, he showed that the rotation of any orthonormal coordinate frame onto another can be defined by three angles, representing a sequence of rotations about the coordinate frame's primary axes, so long as the same axis is not used twice in a row. Figure 1.1 shows this graphically [9].

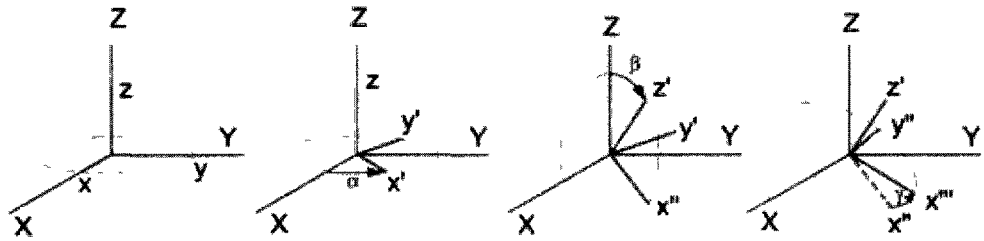


Figure 1.1: Euler angles. 3-2-3 rotation $[\alpha, \beta, \gamma]^T$.

Rotation of α about z: $x\ y\ z \rightarrow x'\ y'\ z$

Rotation of β about y': $x\ y\ z \rightarrow x''\ y'\ z'$

Rotation of γ about z': $x\ y\ z \rightarrow x'''\ y''\ z'$

The non-sequential restriction of axes yields 12 potential axes sequences. (3-2-3 is shown above). The sequence definition describes which axis each Euler angle corresponds to. The 3-2-3 rotation $[\alpha, \beta, \phi]^T$ requires a rotation of α about the z-axis, β about y', the new y-axis, and ϕ about z' the new z-axis.

2 Direct Cosine Matrix

In order to mathematically perform a rotation defined by Euler Angles, it is convenient to assemble it into a direct cosine matrix (DCM). This creates a 3x3 pure rotation matrix which need only be multiplied by a vector to yield the rotated equivalent vector. The DCM can be compiled by multiplying the three rotation matrices which define primary axis rotations in the corresponding order (equations 1.1, 1.2 and 1.3). All 12 Euler sequences' DCMs have been computed and presented in many resources including [34] and are shown in Appendix A.

$$R(1,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix} \tag{1.1}$$

$$R(2,\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{1.2}$$

$$R(1,\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

2.1 Small Angle Approximations

For small angles both the *sin* and *cos* functions can be simplified:

$$\begin{aligned} sin\theta &\simeq \theta \\ cos\theta &\simeq 1 \end{aligned} \tag{1.4}$$

$$\boldsymbol{R}(1,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \theta \\ 0 & -\theta & 1 \end{bmatrix} \tag{1.5}$$

$$\boldsymbol{R}(2,\theta) = \begin{bmatrix} 1 & 0 & -\theta \\ 0 & 1 & 0 \\ \theta & 0 & 1 \end{bmatrix} \tag{1.6}$$

$$\boldsymbol{R}(1,\theta) = \begin{bmatrix} 1 & \theta & 0 \\ -\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

$$\boldsymbol{R}^{123}_{\gamma\beta\alpha} = \begin{bmatrix} 1 & \alpha & -\beta \\ -\alpha & 1 & \gamma \\ \beta & -\gamma & 1 \end{bmatrix} \tag{1.8}$$

3 Quaternion Representation

The quaternion is another widely used attitude representation for spacecraft attitude and control. The quaternion, coined by Hamilton, has other common names such as Euler symmetric parameters or Rodrigues symmetric parameters [32]. One distinct advantage quaternions have over Euler angles is the absence of any singularities. The quaternion is a 4-component tensor that includes a redundant scalar term. Quaternions are hyper-complex numbers of rank 4 [17]. They consist of a scalar value as well as an imaginary vector in three-dimensional space. Obviously, as quaternions are tensors, ordinary linear algebra does not apply. Detailed explanations of quaternion algebra can be found in [18], [31] and [32]. Brief definitions for quaternion terminology are provided below.

3.1 Quaternion Functions

Quaternions consist of a scalar term, q_0, and a three-element imaginary vector $[q_1\boldsymbol{i} \;\; q_2\boldsymbol{j} \;\; q_3\boldsymbol{k}]^T$. In this thesis the scalar value is designated to be the first element.

$$\bar{\boldsymbol{q}} = \begin{bmatrix} q_0 \\ \underline{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1\mathbf{i} \\ q_2\mathbf{j} \\ q_3\mathbf{k} \end{bmatrix} \tag{1.9}$$

The four components of the quaternion must satisfy the following constraint:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{1.10}$$

Similar to computing vector cross products, the use of hyper-imaginary numbers is subject to the following identities, the only difference being unit vectors squared equal -1 rather than 0:

$$\begin{aligned} \mathbf{i^2} = \mathbf{j^2} = \mathbf{j^2} &= -\mathbf{1} \\ \mathbf{ij} = -\mathbf{ji} &= \mathbf{k} \\ \mathbf{jk} = -\mathbf{kj} &= \mathbf{i} \\ \mathbf{ki} = -\mathbf{ik} &= \mathbf{j} \end{aligned} \tag{1.11}$$

Quaternion multiplication is commonly denoted by $\otimes$ and is used to determine the error between two quaternions:

$$q_e = q_m \otimes \hat{q} \tag{1.12}$$

When referring to spacecraft attitudes, this error is known as "multiplicative error", as opposed to the standard error compilation using subtraction:

$$e = q_m - \hat{q} \tag{1.13}$$

which is referred to as "additive error". Quaternion multiplicative error is often used as an alternative to additive error because the resulting multiplicative error is a proper orthogonal rotation defined by q_e. Additive error does not constrain the error to be orthogonal. [21]

Given the quaternion form of Eq. (1.9), (i.e., $\bar{\boldsymbol{q}} = [q_0, \underline{\boldsymbol{q}}]^T$), the following notation can be used:

$$\bar{\boldsymbol{q}}_A \otimes \bar{\boldsymbol{q}}_B = \Omega(\bar{\boldsymbol{q}}_A)\bar{\boldsymbol{q}}_B \tag{1.14}$$

where $\Omega(\bar{\boldsymbol{q}}) \in \Re^{4\times 4}$ is defined as:

$$\Omega(\bar{\boldsymbol{q}}) = \begin{bmatrix} q_0 & -\underline{\boldsymbol{q}}^T \\ \underline{\boldsymbol{q}} & -[\underline{\boldsymbol{q}}\times] + I_{3x3}q_0 \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{1.15}$$

Here, $[\underline{\boldsymbol{q}}\times]$ is a skew-symmetric matrix and is defined as:

$$[\underline{\boldsymbol{q}}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{1.16}$$

Quaternion multiplication is required to formulate the spacecraft kinematic equation defining the time derivative of $\mathbf{q}$ for modeling and is as follows:

$$\dot{\bar{\boldsymbol{q}}} = \frac{1}{2}\bar{\boldsymbol{\omega}}_{b/o} \otimes \bar{\boldsymbol{q}}(t) \tag{1.17}$$

where $\bar{\boldsymbol{\omega}}_{b/o}$ is a vector that represents the spin rate of the spacecraft (s/c) body with respect to the s/c center of mass and is augmented with a zero to allow its use with quaternion multiplication:

$$\bar{\boldsymbol{\omega}}_{b/o} = \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{b/o} \end{bmatrix} = \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{1.18}$$

The quaternion multiplication equations can also be expressed similarly to Eq. (1.14),

$$\bar{\omega}_{b/o} \otimes \bar{q} = \Omega(\bar{\omega}_{b/o})\bar{q} \tag{1.19}$$

where,

$$\Omega(\bar{\omega}_{b/o}) = \begin{bmatrix} 0 & -\omega_{b/o}^T \\ \omega_{b/o} & -[\omega_{b/o}\times] \end{bmatrix} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{1.20}$$

A more detailed and thorough description of quaternion functions can be found in [11], [17], [18], [31] and [32].

4 Attitude Representation Conversions

Euler angles and quaternions can easily be converted to each other using some simple mathematical expressions and equations. To perform this conversion the DCM is required as a middle step. A quaternion can be used to formulate a DCM as follows:

$$\boldsymbol{A}(\boldsymbol{q}) = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2+2q_0q_3 & 2q_1q_3-2q_0q_2 \\ 2q_1q_2-2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3+2q_0q_1 \\ 2q_1q_3+2q_0q_2 & 2q_2q_3-2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \tag{1.21}$$

$$= \begin{bmatrix} 2q_0^2+2q_1^2-1 & 2q_1q_2+2q_0q_3 & 2q_1q_3-2q_0q_2 \\ 2q_1q_2-2q_0q_3 & 2q_0^2+2q_2^2-1 & 2q_2q_3+2q_0q_1 \\ 2q_1q_3+2q_0q_2 & 2q_2q_3-2q_0q_1 & 2q_0^2+2q_3^2-1 \end{bmatrix} \tag{1.22}$$

where Eq. (1.22) simplifies Eq. (1.21) using the quaternion constraint presented in Eq. (1.10). The reverse conversion of a DCM to quaternions is described by:

$$\begin{aligned} q_0 &= \pm\frac{1}{2}\sqrt{A_{11}+A_{22}+A_{33}+1} \\ q_1 &= \frac{1}{4q_0}(A_{23}-A32) \\ q_2 &= \frac{1}{4q_0}(A_{31}-A13) \\ q_3 &= \frac{1}{4q_0}(A_{12}-A21) \end{aligned} \tag{1.23}$$

the non-exclusivity of quaternions can be seen in this equation due to the "±". Since $\bar{q} = -\bar{q}$, a series of converted quaternions may not be continuous. A check for discontinuity and inversion of the quaternion can restore the quaternions expected continuous nature without effecting the quaternion's attitude definition.

When converting Euler angles to a DCM one of the twelve possible sequences must be chosen. For the 1-2-3 rotation sequence euler angle $\boldsymbol{e} = [\phi, \theta, \psi]^T$ the DCM is formulated as follows:

$$\boldsymbol{A}(\boldsymbol{e}) =$$

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{1.24}$$

DCM's for the remaining eleven sequences are defined in Appendix A. The conversion of the DCM to Euler Angles is performed by solving the following:

$$\begin{aligned} tan\ \psi &= \frac{A_{12}}{A_{11}} \\ sin\ \theta &= -A_{13} \\ tan\ \phi &= \frac{A_{23}}{A_{33}} \end{aligned} \tag{1.25}$$

5 Spacecraft Kinematics and Dynamics

As previously stated, Eq. (1.26) below defines the time derivative of attitude as 1/2 the multiplication of a quaternion assembled of the angular velocity and the attitude quaternion:

$$\dot{\bar{q}} = \frac{1}{2}\bar{\omega}_{b/o} \otimes \bar{q}(t) \tag{1.26}$$

The dynamic equations for angular rates are given as:

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(\sum \boldsymbol{T} - [\boldsymbol{J}\boldsymbol{\omega}\times]\boldsymbol{\omega}) \tag{1.27}$$

The above set of Euler's equations can be written more concisely and accurately if the entire inertia tensor, J, is known,

$$\begin{array}{l} \dot{w}_x = I_{xx}^{-1}(\sum T_x - (I_{zz} - I_{yy})\omega_y\omega_z) \\ \dot{w}_y = I_{yy}^{-1}(\sum T_y - (I_{xx} - I_{zz})\omega_x\omega_z) \\ \dot{w}_z = I_{zz}^{-1}(\sum T_z - (I_{yy} - I_{xx})\omega_x\omega_y) \end{array} \tag{1.28}$$

By expanding Eq. (1.26) and combining it with a simplified Eq. (1.28) with no input torques, one can derive the nonlinear equation $\boldsymbol{f}(t)$, which describes the system assuming the s/c body coordinate axes lie along the principle axes of inertia:

$$\boldsymbol{f}(x,t) = \begin{bmatrix} -\frac{1}{2}(\omega_x q_1 + \omega_y q_2 + \omega_z q_3) \\ \frac{1}{2}(\omega_x q_0 + \omega_z q_2 - \omega_y q_3) \\ \frac{1}{2}(\omega_y q_0 - \omega_z q_1 + \omega_x q_3) \\ \frac{1}{2}(\omega_z q_0 + \omega_y q_1 - \omega_x q_2) \\ -I_{xx}^{-1}(I_{zz} - I_{yy})\omega_y\omega_z \\ -I_{yy}^{-1}(I_{xx} - I_{zz})\omega_x\omega_z \\ -I_{zz}^{-1}(I_{yy} - I_{xx})\omega_x\omega_y \end{bmatrix} \tag{1.29}$$

If the spacecraft axes do not lie along the principle axes of inertia the nonlinear equation, accounting for torques on the body, becomes

$$\boldsymbol{f}(x,t) = \begin{bmatrix} \frac{1}{2}\bar{\boldsymbol{\omega}}_{b/o} \otimes \bar{\boldsymbol{q}}(t) \\ \boldsymbol{J}^{-1}(\sum \boldsymbol{T} - [\boldsymbol{J}\boldsymbol{\omega}\times]\boldsymbol{\omega}) \end{bmatrix} \tag{1.30}$$

6 Sensor Models

Simulating mission data requires models of the star tracker and accelerometer. The star tracker model is the attitude quaternion generated by the simulation with added noise. The accelerometer has a more complicated form, shown in Eq. (1.32).

To simulate a higher integrity accelerometer measurement, 4 vectors are used to artificially corrupt the measurement relative to the nominal acceleration model.

The quaternion measurement is of the form:

$$\boldsymbol{q_m} = \boldsymbol{q_{sim}} + \boldsymbol{\nu}_q \tag{1.31}$$

where $\boldsymbol{q}_{sim}$ is the simulated quaternion and $\boldsymbol{\nu}_q$ represents white noise.

The typical accelerometer model is:

$$\boldsymbol{a}_{nom} = [\dot{\boldsymbol{\omega}}\times]\boldsymbol{r} + [\boldsymbol{\omega}_{sim}\times]^2\boldsymbol{r} + \boldsymbol{a}_{sc/inertia} \tag{1.32}$$

where $\boldsymbol{\omega}_{sim}$ is the simulated body rate, $\dot{\boldsymbol{\omega}}$ is generated from $\boldsymbol{\omega}_{sim}$ using Eq. (1.27), $\mathbf{r}$ is the location of the accelerometer on the s/c relative to the center of mass, and $\boldsymbol{a}_{sc/inertia}$ is the acceleration of the s/c due to external forces or thrusters.

If the location of the accelerometer relative to the center of mass shifts, and the acceleration of the s/c relative to the inertial frame is neglected, the resulting accelerometer model is:

$$\boldsymbol{a}_{shift} = [\dot{\boldsymbol{\omega}}\times](\boldsymbol{r} + \boldsymbol{\delta r}) + [\boldsymbol{\omega}\times]^2(\boldsymbol{r} + \boldsymbol{\delta r}) \tag{1.33}$$

where $\boldsymbol{\delta r}$ is the location shift. If the accelerometer rotates relative to its initial alignment and maintains orthogonal alignment, this rotational misalignment can be accounted for using the following equation:

$$\boldsymbol{a}_{mis} = \boldsymbol{\Delta}[\boldsymbol{a}_{shift}] \tag{1.34}$$

where $\boldsymbol{\Delta}$ is the rotation matrix defining the misalignment. For MMS mission purposes one can safely assume small misalignments, allowing for small angle approximations

when constructing the rotation matrix. The symbol $\boldsymbol{\delta}$ denotes the 1-2-3 Euler rotation sequence such that

$$\boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \tag{1.35}$$

$$\boldsymbol{\Delta} = \boldsymbol{I}_{3x3} - [\boldsymbol{\delta}\times] \tag{1.36}$$

Here, $[\boldsymbol{\delta}\times]$ is a skew symmetric matrix defined previously in Section 3.1

To properly construct a higher integrity accelerometer model, sensor bias ($\boldsymbol{a}_b$) and white noise ($\boldsymbol{\nu}_{accelerometer}$) are also added such that

$$\boldsymbol{a}_{measured} = [\boldsymbol{a}_{mis} + \boldsymbol{a}_{bias}] + \boldsymbol{\nu}_{accelerometer} \tag{1.37}$$

This thesis proposes that for the MMS s/c the accelerometer can be adaquately represented by a constant lumped bias to correct for $\boldsymbol{\delta r}$, $\boldsymbol{\delta}$ and $\boldsymbol{a}_b$. That is,

$$\boldsymbol{a}_{nom} \approx [\boldsymbol{a}_m + \boldsymbol{a}_{LB}] \tag{1.38}$$

where $\boldsymbol{a}_{nom}$ is the nominal acceleration, $\boldsymbol{a}_m$ is the measured acceleration and $\boldsymbol{a}_{LB}$ is a lumped bias to account for all corruptions.

Chapter II

LUMPED BIAS JUSTIFICATION

This thesis proposes the estimation of a lumped bias to account for the accelerometer's bias, shift in location and orthogonal misalignment. Before investigating methods to estimate this lumped bias, a lumped bias assumption is confirmed to be adequate given two additional assumptions: (1) a near constant spin rate primarily about a single axis and (2) significantly larger inertia tensor terms on the primary axes than that of the secondary axes.

1 Analytical Justification

The nominal acceleration, $\boldsymbol{a}_{nom}$, can be represented and predicted by Eq. (2.1):

$$\boldsymbol{a}_{nom} = [\dot{\boldsymbol{\omega}}\times]\mathbf{r} + [\boldsymbol{\omega}\times]^2\mathbf{r} + \boldsymbol{a}_{\mathbf{sc/inertia}} \tag{2.1}$$

where $\boldsymbol{\omega}$ is the angular body rate, $\dot{\boldsymbol{\omega}}$ is the angular acceleration of the body, $\boldsymbol{r}$ is the location of the accelerometer relative to the s/c center of mass, and $\boldsymbol{a}_{sc/inertia}$ is the acceleration of the spacecraft due to external forces.

Given a small shift in the location of the accelerometer with respect to the center

of mass, denoted by $\boldsymbol{\delta r}$, the true acceleration is:

$$\boldsymbol{a}_{true} = \boldsymbol{a}_{nom} + \boldsymbol{a}_{LB} = [\dot{\boldsymbol{\omega}}\times](\boldsymbol{r} + \boldsymbol{\delta r}) + [\boldsymbol{\omega}\times]^2(\boldsymbol{r} + \boldsymbol{\delta r}) + \boldsymbol{a}_{sc/inertia} \tag{2.2}$$

where $\boldsymbol{a}_{LB}$ denotes the lumped bias.

Taking the difference between the nominal and true accelerations yields the resulting lumped bias as a function of $\boldsymbol{\delta r}$:

$$\boldsymbol{a}_{LB} = [\dot{\boldsymbol{\omega}}\times](\boldsymbol{\delta r}) + [\boldsymbol{\omega}\times]^2(\boldsymbol{\delta r}) \tag{2.3}$$

The $\dot{\boldsymbol{\omega}}$ term can be neglected if assumption (2) is made and there are no applied torques on the MMS s/c and, therefore, angular acceleration can be simplified to:

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}[\boldsymbol{J}\boldsymbol{\omega}\times]\boldsymbol{\omega} \approx \boldsymbol{0}_{3\times 1} \tag{2.4}$$

Given $[\boldsymbol{\omega}\times]$ is

$$[\boldsymbol{\omega}\times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.5}$$

Then $[\boldsymbol{\omega}\times]^2$ is such that

$$[\boldsymbol{\omega}\times]^2 = \begin{bmatrix} -\omega_z^2 - \omega_y^2 & \omega_x\omega_y & \omega_x\omega_z \\ \omega_x\omega_y & -\omega_z^2 - \omega_x^2 & \omega_y\omega_z \\ \omega_x\omega_z & \omega_y\omega_z & -\omega_y^2 - \omega_x^2 \end{bmatrix} \tag{2.6}$$

Furthermore, if ω_x and ω_y are sufficiently small, then

$$[\boldsymbol{\omega}\times]^2 \approx \begin{bmatrix} -\omega_z^2 & 0 & \omega_x\omega_z \\ 0 & -\omega_z^2 & \omega_y\omega_z \\ \omega_x\omega_z & \omega_y\omega_z & -\omega_y^2 - \omega_x^2 \end{bmatrix} \approx \begin{bmatrix} -\omega_z^2 & 0 & 0 \\ 0 & -\omega_z^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.7}$$

The resulting accelerometer lumped bias, $\boldsymbol{a}_{LB}$ is dependent only upon ω_z and the shift in center of mass in the lateral directions on the s/c spin plane. If these terms are constant then $\boldsymbol{a}_{LB}$ can safely be represented by a constant vector:

$$\boldsymbol{a}_{LB}(\boldsymbol{\delta r}) = [\boldsymbol{\omega}\times]^2\boldsymbol{\delta r} \approx \begin{bmatrix} -\omega_z^2 & 0 & 0 \\ 0 & -\omega_z^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta r_x \\ \delta r_y \\ \delta r_z \end{bmatrix} = \begin{bmatrix} -\omega_z^2\delta r_x \\ -\omega_z^2\delta r_y \\ 0 \end{bmatrix} \tag{2.8}$$

If the same approximations are made and only a misalignment is considered, then the true acceleration can be simplified to:

$$\boldsymbol{a}_{true} \approx [\boldsymbol{\omega}\times]^2\boldsymbol{r} \approx \begin{bmatrix} -\omega_z^2 & 0 & 0 \\ 0 & -\omega_z^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} -\omega_z^2 r_x \\ -\omega_z^2 r_y \\ 0 \end{bmatrix} \tag{2.9}$$

If an orthogonal small angle misalignment, represented by $[\boldsymbol{\delta}\times]$, is present, the measured acceleration, $\boldsymbol{a}_{mis}$, can be described as:

$$\boldsymbol{a}_{mis} \approx \boldsymbol{a}_{true} + \boldsymbol{a}_{LB} = (\boldsymbol{I}_{3x3} - \boldsymbol{\delta}\times) \begin{bmatrix} -\omega_z^2 r_x \\ -\omega_z^2 r_y \\ 0 \end{bmatrix} \tag{2.10}$$

The difference between the true acceleration and $\boldsymbol{a}_{mis}$, then, represents the lumped bias due to a small-angle orthogonal misalignment of the accelerometer:

$$\boldsymbol{a}_{LB}(\boldsymbol{\delta}) = \begin{bmatrix} 0 & \delta_z & -\delta_y \\ -\delta_z & 0 & \delta_x \\ \delta_y & -\delta_x & 0 \end{bmatrix} \begin{bmatrix} -\omega_z^2 r_x \\ -\omega_z^2 r_y \\ 0 \end{bmatrix} = \begin{bmatrix} -\delta_z r_y \omega_z^2 \\ \delta_z r_x \omega_z^2 \\ (\delta_x r_y - \delta_y r_x)\omega_z^2 \end{bmatrix} \tag{2.11}$$

Equations (2.8) and (2.11) each describe the lumped bias as a function of their respective accelerometer corruption, $\boldsymbol{\delta r}$ or $\boldsymbol{\delta}$. The approximated total lumped bias is, then,

$$\boldsymbol{a}_{LB}(\boldsymbol{\delta r}, \boldsymbol{\delta}) \approx \begin{bmatrix} -\delta_z r_y \omega_z^2 \\ \delta_z r_x \omega_z^2 \\ (\delta_x r_y - \delta_y r_x)\omega_z^2 \end{bmatrix} + \begin{bmatrix} -\omega_z^2 \delta r_x \\ -\omega_z^2 \delta r_y \\ 0 \end{bmatrix} \tag{2.12}$$

So long as the parameters in Eq. (2.12) are constant, the lumped bias approximation does in fact hold.

2 Numerical Validation of the Lumped Bias Model

In addition to an analytical inspection of the sensor and system equations to verify that the lumped bias is adequate, numerical simulations are run and accelerometer errors are computed under conditions of accelerometer bias, center of mass shift and

orthogonal misalignment. The results are consistent with those of the analytical results previously presented. For all simulations the measured acceleration is computed using the following:

$$\boldsymbol{a}_m = \boldsymbol{\Delta}[\dot{\boldsymbol{\omega}}\times](\boldsymbol{r}+\boldsymbol{\delta r}) + [\boldsymbol{\omega}\times]^2(\boldsymbol{r}+\boldsymbol{\delta r}) + \boldsymbol{a}_b + \boldsymbol{\nu}_{accelerometer} \tag{2.13}$$

The resulting error due to a static accelerometer bias, $\boldsymbol{a}_b$, is obviously apparent, but is shown, nonetheless, for the sake of visual comparison to other sources of accelerometer measurement error. A representative plot of the effects of static accelerometer bias is presented in Figure 2.1. Here, the x-component effect of $\boldsymbol{a}_m$ is shown for $a_{b\,x} = -1 \times 10^{-4}\ m/s^2$.

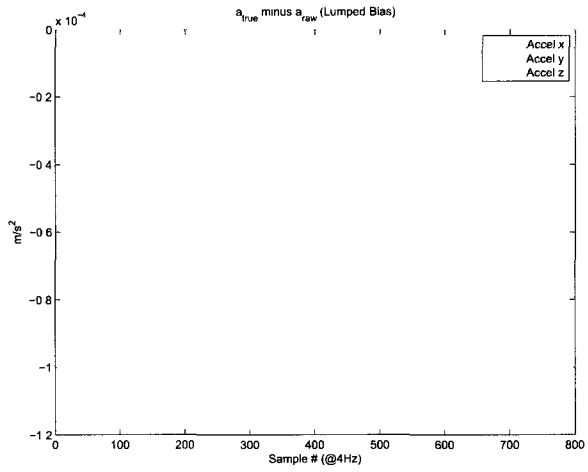


Figure 2.1: Numerically predicted bias from $a_{b.x}$ component

Figures 2.2 through 2.4 show resulting accelerometer measurement error due to a 1cm shift in accelerometer location, $\boldsymbol{\delta r}$, in the x, y and z axes respectively. Each shift along each individual axis is performed independently and no sensor bias or misalignment is introduced.

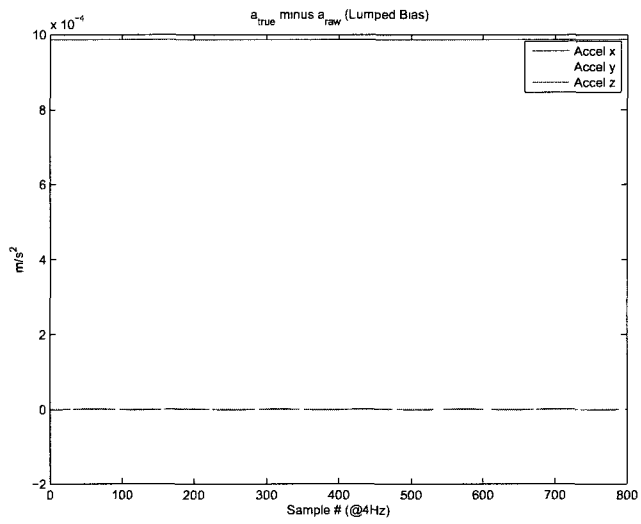


Figure 2.2: Numerically predicted bias from $\delta r_x = 1cm$

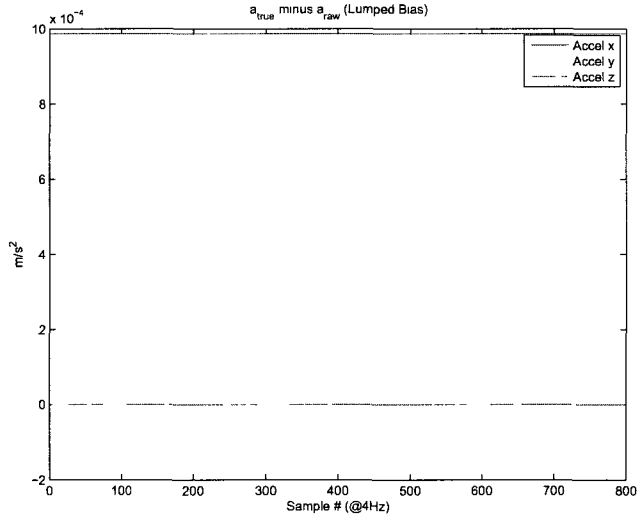


Figure 2.3: Numerically predicted bias from $\delta r_y = 1cm$

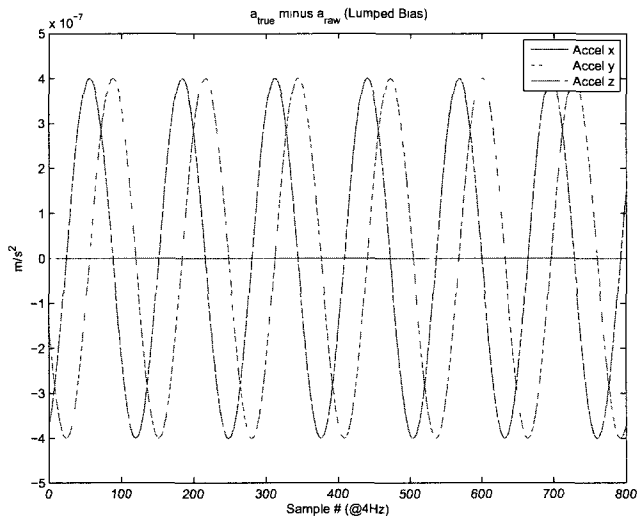


Figure 2.4: Numerically predicted bias from $\delta r_z = 1cm$

Uncertainties in δr_x and δr_y both yield a near constant bias error on the accelerometer. The oscillation error amplitudes, which cannot be seen above are on the order of $3 \times 10^{-6}\ m/s^2$. The magnitude and static nature both agree with the predicted analytical results of $\boldsymbol{a}_{LB} = \omega_z^2 \delta r_x = 9.8 \times 10^{-4} m/s^2$. An uncertainty on δr_z appears to yield an oscillating error, however, the scale is much smaller and the amplitudes of oscillations are within $\pm 0.05 \mu g$. In short, all oscillations for $\boldsymbol{\delta r}$ are considered negligible in light of the required mission accuracy for determining the true lumped bias to within $1 \mu g$.

Figures 2.5 through 2.7 show the accelerometer measurement errors, $\boldsymbol{a}_{LB}$, due to an orthogonal misalignment about each of the x, y and z-axes respectively. Each misalignment is 20arcsec, and no sensor bias or shift in accelerometer location are introduced.

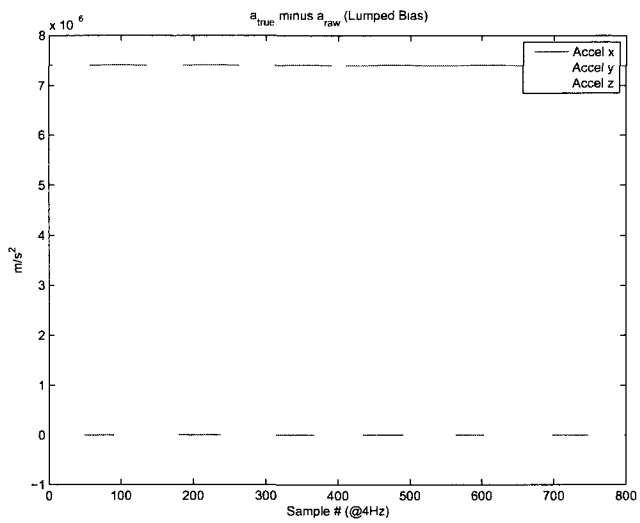


Figure 2 5: Numerically predicted bias from $\delta_x = 20arcsec$

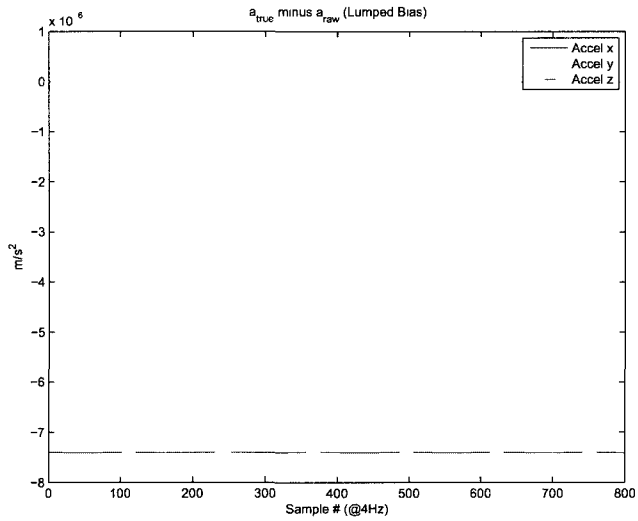


Figure 2 6. Numerically predicted bias from $\delta_y = 20arcsec$

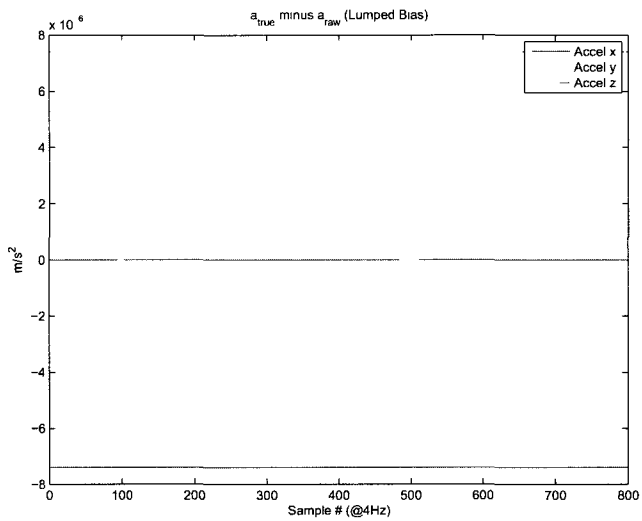


Figure 2.7: Numerically predicted bias from $\delta_z = 20arcsec$

The errors are again nearly constant, allowing them to be corrected for by a constant lumped bias. The magnitudes match those predicted by our analytical results which indicated $\boldsymbol{a}_{LB} = [0\ 0\ 7.12]^T \times 10^{-6}\ \ m/s^2$ for $\delta_x = 20arcsec$, $\boldsymbol{a}_{LB} = [0\ 0\ -7.12]^T \times 10^{-6}\ \ m/s^2$ for $\delta_y = 20arcsec$, and $\boldsymbol{a}_{LB} = [7.12\ -7.12\ 0]^T \times 10^{-6}\ \ m/s^2$ for $\delta_z = 20arcsec$.

From these results it is apparent that the errors due to $\boldsymbol{\delta r}$ and $\boldsymbol{\delta}$ are so similar to a static $\boldsymbol{a}_b$ that any filter would have difficulty distinguishing between the three Not only is a lumped bias adequate, but it also provides a more stable and accurate filter It should be noted that all the numerical results presented make none of the assumptions that are relied on for the analytical lumped-bias justification, such as only ω_z rotation and axes of inertia aligned with the center of mass.

Chapter III

ESTIMATION TECHNIQUES

Various estimation techniques are presented here, including both batch processes and real-time dynamically driven filtering methods. The first method is a cascading filter design, driven by an existing Extended Kalman Filter (EKF) designed by Thienel [39]. The second stage of the cascading filter design uses the method of least squares to determine the accelerometer's orthogonal misalignment. This design may have benefited from an iterative process. Some of the assumptions made by the EKF were inappropriate and therefore additional techniques are investigated. The remaining estimation routines involve the estimation of a 3-component lumped-bias parameter rather than the 9 components required to fully define the 3-component accelerometer's output bias, the 3-component location bias and three component orthogonal misalignment. This lumped bias is verified to be adequate using analytical, numerical and experimental methods. Three real-time filters/estimators are presented: an Extended Kalman Filter, an H_∞ filter and a Sliding Mode Observer. These filters/estimators are subjected to various model and measurement errors to test for robustness. In addition to these real-time filters, a post processing estimation routine is presented using a spline smoothing technique to smooth and differentiate measurements to solve the s/c kinematic equations and robustly determine the lumped bias.

1 Cascading Filter

The initial cascading filter was first proposed when NASA became concerned with the orthogonal misalignment of the accelerometer on board the MMS spacecraft. This accelerometer is crucial for driving state estimates at times when the s/c star tracker is unavailable or for performing precise thruster burns for crucial orbital maneuvers and maintaining formation. Previous substantial work has already been performed developing an Extended Kalman Filter for the estimation of the spacecraft's attitude, the spacecraft angular body rates, accelerometer bias, and the differential shift in the spacecraft's center of mass relative to the accelerometer [39] [40]. Concerns of the state's observability and filter stability if three additional states are added to the EKF lead to the concept of a cascading filter. Here the output of stable EKF estimates are used to determine the orthogonal misalignment via the method of least squares.

The full description of the MMS s/c acceleration is:

$$\boldsymbol{a}_m = \boldsymbol{\Delta}\left[[\dot{\boldsymbol{\omega}}\times](\boldsymbol{r}+\boldsymbol{\delta r}) + [\boldsymbol{\omega}\times]^2(\boldsymbol{r}+\boldsymbol{\delta r})\right] + \boldsymbol{a}_b + \boldsymbol{\nu}_{accelerometer} \tag{3.1}$$

where $\boldsymbol{a}_m$ is the measured acceleration, $\boldsymbol{\Delta}$ is the rotation matrix for the small angle orthogonal misalignment of the s/c accelerometer, $\boldsymbol{\omega}$ represents the angular body rates, $\boldsymbol{r}$ is the location of the accelerometer on the s/c relative to the s/c center of mass in s/c body coordinates, $\boldsymbol{\delta r}$ is the shift in accelerometer location, $\boldsymbol{a}_b$ is the accelerometer bias, and $\boldsymbol{\nu}_{accelerometer}$ represents white noise.

The misalignment corruption can be isolated using Eq. (3.2) which was first in-

troduced in Chapter I:

$$\boldsymbol{a}_{mis} = \boldsymbol{\Delta}\,[\boldsymbol{a}_{shift}] \tag{3.2}$$

where $\boldsymbol{a}_{mis}$ is the acceleration due to an orthogonal misalignment from $\boldsymbol{a}_{shift}$.

In Eq. (3.2), Δ is easy solved for using a least squares algorithm with several batch samples of $\boldsymbol{a}_{mis}$ and $\boldsymbol{a}_{shift}$. Solving this over-constrained equation allows attenuation of noise and other random uncertainties.

If $\boldsymbol{a}_{mis}$ and $\boldsymbol{a}_{shift}$ are computed as

$$\begin{aligned} \boldsymbol{a}_{mis} &= \boldsymbol{a}_m - \hat{\boldsymbol{a}}_b \\ \boldsymbol{a}_{shift} &= [\dot{\boldsymbol{\omega}}\times](\boldsymbol{r} + \hat{\boldsymbol{r}}_c) + [\boldsymbol{\omega}\times]^2(\boldsymbol{r} + \hat{\boldsymbol{r}}_c) \end{aligned} \tag{3.3}$$

Using N measurements, an augmented measurement vector may be obtained to form $\boldsymbol{A}_{mis}$ and $\boldsymbol{A}_{shift}$:

$$\begin{aligned} \boldsymbol{A}_{mis} &= [\boldsymbol{a}_{mis1}\boldsymbol{a}_{mis2}\ldots\boldsymbol{a}_{misN}] \\ \boldsymbol{A}_{shift} &= [\boldsymbol{a}_{shift1}\boldsymbol{a}_{shift2}\ldots\boldsymbol{a}_{shiftN}] \end{aligned} \tag{3.4}$$

The least-square algorithm is used to obtain the generic matrix $\boldsymbol{x}$ using $\boldsymbol{Y}$ and $\boldsymbol{H}$ which satisfy the equation below:

$$\begin{aligned} \boldsymbol{Y} &= \boldsymbol{x}\boldsymbol{H} \\ \boldsymbol{A}_{mis} &= (\boldsymbol{I}_{3\times3} - [\boldsymbol{\delta}\times])\boldsymbol{A}_{shift} \end{aligned} \tag{3.5}$$

The second relation in Eq. (3.5) replaces generic variables with parameters relevant to this research. Replacing $\boldsymbol{\Delta}$ in Eq. (3.2) by $(\boldsymbol{I}_{3x3} - [\boldsymbol{\delta}\times])$,

$$\boldsymbol{I}_{3\times3} - [\boldsymbol{\delta}\times] = \boldsymbol{A}_{mis}\boldsymbol{A}_{shift}^T(\boldsymbol{A}_{shift}\boldsymbol{A}_{shift}^T)^{-1} \tag{3.6}$$

The rotation matrix $(\boldsymbol{I}_{3\times3} - [\delta\times])$ is solved for by using Eq. (3.5) taking the pseudo-inverse and using the method of least squares. With $N \times 3$ equations and only 3×3

unknowns, the solution is over constrained, and the method of least squares provides a minimal-variance error solution.

The result is a small-angle rotation matrix, which in theory should be $(\boldsymbol{I}_{3\times3} - [\delta\times]) \in \Re^{3\times3}$. Findings show that the output of the method of least squares, due to errors in the accelerometer bias and location of the center of mass, result in an ill-conditioned rotation matrix output. Results are presented in the following chapter.

2 Dynamic Filters/Estimators

The MMS s/c requires 7 states to define its attitude and angular body rates.

$$\boldsymbol{x}' = \begin{bmatrix} \bar{\boldsymbol{q}} \\ \boldsymbol{\omega} \end{bmatrix} \tag{3.7}$$

Although the s/c is fully defined by these 7 states, in order to estimate the accelerometer lumped bias, the state vector $\boldsymbol{x}$ is augmented with three terms to represent this bias:

$$\boldsymbol{x} = \begin{bmatrix} \bar{\boldsymbol{q}} \\ \boldsymbol{\omega} \\ \boldsymbol{a}_{LB} \end{bmatrix} \tag{3.8}$$

The system model $\boldsymbol{f}$ defined in Eq. (1.30) must also be augmented with the description of the $\boldsymbol{a}_{LB}$ kinematics. It can be safely assumed that the lumped bias is constant. That is,

$$\dot{\boldsymbol{a}}_{LB} = \mathbf{0}_{3\times3} \tag{3.9}$$

The resulting augmented non-linear model is,

$$\boldsymbol{f}(\mathbf{x},\mathbf{t}) = \begin{bmatrix} \frac{1}{2}\bar{\boldsymbol{\omega}}_{b/o} \otimes \bar{\boldsymbol{q}}(t) \\ \boldsymbol{J}^{-1}(\sum \boldsymbol{T} - [\boldsymbol{J}\boldsymbol{\omega}\times]\boldsymbol{\omega}) \\ \mathbf{0_{3\times3}} \end{bmatrix} \tag{3.10}$$

Two sensors are used for filter updates: a star tracker and an accelerometer. The measurement equation relating these sensors to the augmented state vector $\boldsymbol{x}$ is

$$\boldsymbol{g}(x,t) = \begin{bmatrix} \bar{\boldsymbol{q}} \\ [\dot{\boldsymbol{\omega}}\times]\,\boldsymbol{r} + [\boldsymbol{\omega}\times]^2\,\boldsymbol{r} + \boldsymbol{a}_{LB} \end{bmatrix} \tag{3.11}$$

2.1 Kalman Filter

The Kalman Filter (KF) is an optimal filter provided certain conditions are met. The KF assumes that the model is linear and that the model and statistical data describing any model uncertainties and Gaussian measurement noise is known a priori. If these assumptions are not valid then the filter may still provide acceptable results. however, not necessarily optimal.

Optimality is accomplished by minimizing the expected variance of the estimation error. Utilizing knowledge of the system model uncertainty as well as the noise statistics for each sensor, the KF estimates future states using the known system model and then performs a weighted average of these estimates as well as the measurements. The Kalman algorithm provides optimal weightings for this process.

The discretized linear model is defined such that:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{B}_k\boldsymbol{u}_k + \boldsymbol{\Gamma}_k\boldsymbol{\eta}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{\nu}_k \end{aligned} \tag{3.12}$$

where $\boldsymbol{x}_k \in \Re^{n\times 1}$ is the vector of n states, $\boldsymbol{u}_k \in \Re^{p\times 1}$ represents the vector of p known inputs. Model uncertainties are lumped in to a single term $\boldsymbol{\Gamma}_k\boldsymbol{\eta}_k \in \Re^{n\times 1}$, where $\boldsymbol{\eta}_k$ represents model uncertainty (modeled as Gaussian process noise). $\boldsymbol{A}_k \in \Re^{n\times n}$ and

$\boldsymbol{B}_k \in \Re^{n\times m}$ represent the model in discrete state space form.

In the measurement equation, $\boldsymbol{y}_k \in \Re^{m\times 1}$ is the sensor output, $\boldsymbol{\nu}_k \in \Re^{m\times 1}$ represents the sensor noise and $\boldsymbol{C}_k \in \Re^{m\times n}$ is the state space measurement matrix.

The Kalman algorithm is described in [8] as:

$$\begin{aligned}
\boldsymbol{P}_{0|0} &= Var(\boldsymbol{x}_0) \\
\hat{\boldsymbol{x}}_{0|0} &= E(\boldsymbol{x}_0)
\end{aligned}$$

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{A}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} \\
\boldsymbol{P}_{k|k-1} &= \boldsymbol{A}_{k-1}\boldsymbol{P}_{k-1}\boldsymbol{A}_{k-1}^T + \boldsymbol{\Gamma}_{k-1}\boldsymbol{Q}_{k-1}\boldsymbol{\Gamma}_{k-1}^T
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{G}_k &= \boldsymbol{P}_{k|k-1}\boldsymbol{C}_k^T(\boldsymbol{C}_k\boldsymbol{P}_{k|k-1}\boldsymbol{C}_k^T + \boldsymbol{R}_k)^{-1} \\
\hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{G}_k(\boldsymbol{y}_k - \boldsymbol{C}_k\hat{\boldsymbol{x}}_{k|k-1}) \\
\boldsymbol{P}_{k|k} &= (\boldsymbol{I} - \boldsymbol{G}_k\boldsymbol{C}_k)\boldsymbol{P}_{k|k-1}
\end{aligned} \tag{3.13}$$

For use in the Kalman algorithm, $\boldsymbol{Q}_k \in \Re^{n\times n}$ is the covariance matrix corresponding to model process noise, $\boldsymbol{\eta}_k$. $\boldsymbol{R}_k \in \Re^{n\times n}$ is the covariance matrix corresponding to sensor noise, $\boldsymbol{\nu}_k$. $\boldsymbol{P}_k$ is an estimated covariance matrix corresponding to the KF's estimate error. The quantity $(\boldsymbol{y}_k - \boldsymbol{C}_k\hat{\boldsymbol{x}}_{k|k-1})$ is referred to as the innovation and is used by the Kalman gain, $\boldsymbol{G}_k$, to update the estimates after each measurement, $\boldsymbol{y}_k$.

The subscript $k|k-1$ represents a quantity estimated at time k using using the model and estimate from time $k-1$. Subscript $k|k$ represents an estimate at time k using the measurement at time k.

The five steps of the Kalman Filter are sometimes reduced to three where the propagation and measurement updates are combined.

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k+1} &= \boldsymbol{A}_k\hat{\boldsymbol{x}}_k + \boldsymbol{A}_k\boldsymbol{G}_k(\boldsymbol{y}_k - \boldsymbol{C}_k\hat{\boldsymbol{x}}_k) \\
\boldsymbol{G}_k &= \boldsymbol{P}_k(\boldsymbol{I} + \boldsymbol{C}_k^T\boldsymbol{R}_k^{-1}\boldsymbol{C}_k\boldsymbol{P}_k)^{-1}\boldsymbol{C}_k^T\boldsymbol{R}_k^{-1} \\
\boldsymbol{P}_{k+1} &= \boldsymbol{A}_k\boldsymbol{P}_k(\boldsymbol{I} + \boldsymbol{C}_k^T\boldsymbol{R}_k^{-1}\boldsymbol{C}_k\boldsymbol{P}_k)^{-1}\boldsymbol{A}_k^T + \Gamma_k\boldsymbol{Q}_k\Gamma_k^T
\end{aligned} \tag{3.14}$$

The KF uses the known variances of sensors and the system in addition to known system dynamics to form an optimal Kalman Gain, $\boldsymbol{G}_k$. This gain takes a weighted average of the dynamically expected states and a measurement of the states. By performing this weighted average at discrete sensor measurement times, the KF can accurately attenuate noise and estimate states not directly measured (assuming the system is observable). It also provides the covariance of the estimate error, indicating how accurate the state estimates are expected to be.

2.2 Extended Kalman Filter

The Kalman Filter is designed for linear systems [8]. A modified algorithm, the Extended Kalman Filter, was designed to accommodate nonlinear systems. The process is the same, except that linear matrices are replaced by either nonlinear functions or Jacobians of the nonlinear functions. A nonlinear discrete-time system is represented as

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}_k) + \Gamma_k\eta_k \\
\boldsymbol{y}_k &= \boldsymbol{g}_k(\boldsymbol{x}_k) + \boldsymbol{\nu}_k
\end{aligned} \tag{3.15}$$

In the propagation equation,

$$\hat{\boldsymbol{x}}_{k|k-1} = \mathrm{f}_{\mathrm{k-1}}(\hat{\mathrm{x}}_{\mathrm{k-1}}) \tag{3.16}$$

the nonlinear function, $\boldsymbol{f}$, is used to propagate states. The error covariance is propagated such that

$$\boldsymbol{P}_{k,k-1} = \left[\frac{\partial \mathrm{f_{k-1}}}{\partial \mathrm{x_{k-1}}}(\hat{\mathrm{x}}_{k-1})\right] \boldsymbol{P}_{k-1,k-1} \left[\frac{\partial \mathrm{f_{k-1}}}{\partial \mathrm{x_{k-1}}}(\hat{\mathrm{x}}_{k-1})\right]^T + \Gamma_{k-1}(\hat{\boldsymbol{x}}_{k-1})\boldsymbol{Q}_{k-1}\boldsymbol{\Gamma}_{k-1}^T(\hat{\boldsymbol{x}}_{k-1}) \tag{3.17}$$

where the Jacobian, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$, is used in place of $\boldsymbol{A}$. For the measurement update, the state estimates are propagated such that

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{G}_k(\boldsymbol{y}_k - \boldsymbol{g}_k(\hat{\boldsymbol{x}}_{k|k-1})) \tag{3.18}$$

Here the full nonlinear function $\boldsymbol{g}$ is used to form the innovation in place of $\boldsymbol{Cx}$. When updating the error covariance and Kalman gain after the measurement, the Jacobian $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$ is used in place of $\boldsymbol{C}$ and

$$\boldsymbol{P}_{k,k} = \left[\boldsymbol{I} - \boldsymbol{G}_k \left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right]\right] \boldsymbol{P}_{k,k-1} \tag{3.19}$$

$$\boldsymbol{G}_k = \boldsymbol{P}_{k,k-1} \left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right]^T \left[\left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right] \boldsymbol{P}_{k,k-1} \left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right]^T + \boldsymbol{R}_k\right]^{-1} \tag{3.20}$$

The EKF algorithm in its operating structure is defined as:

$$P_{0,0} = Var(x_0)$$
$$\hat{x}_0 = E(x_0)$$

$$P_{k,k-1} = \left[\frac{\partial \mathrm{f_{k-1}}}{\partial \mathrm{x_{k-1}}}(\hat{\mathrm{x}}_{k-1})\right] P_{k-1,k-1} \left[\frac{\partial \mathrm{f_{k-1}}}{\partial \mathrm{x_{k-1}}}(\hat{\mathrm{x}}_{k-1})\right]^T + \Gamma_{k-1}(\hat{x}_{k-1}) Q_{k-1} \Gamma_{k-1}^T(\hat{x}_{k-1})$$
$$\hat{x}_{k|k-1} = \mathrm{f_{k-1}}(\hat{\mathrm{x}}_{k-1}) \tag{3.21}$$

$$G_k = P_{k,k-1} \left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right]^T \left[\left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right] P_{k,k-1} \left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right]^T + R_k\right]^{-1}$$
$$P_{k,k} = \left[I - G_k \left[\frac{\partial \mathrm{g_k}}{\partial \mathrm{x_k}}(\hat{\mathrm{x}}_{k|k-1})\right]\right] P_{k,k-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + G_k(y_k - g_k(\hat{x}_{k|k-1}))$$

All Jacobians and matrices are evaluated at the most recent state estimate in each step.

For the spacecraft system in this research:

$$f = \begin{bmatrix} \dot{\bar{q}} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\bar{\omega}_{b/o} \otimes \bar{q} \\ J^-1\,(\sum T - [J\omega\times]\omega) \end{bmatrix} \tag{3.22}$$

The Jacobian can be divided into smaller Jacobians, many of which are null:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_q}{\partial q} & \frac{\partial f_q}{\partial w} & 0_{4\times3} \\ 0_{3\times4} & \frac{\partial f_w}{\partial w} & 0_{3\times3} \\ 0_{3\times4} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{3.23}$$

The three non-zero Jacobians are:

$$\frac{\partial f_q}{\partial q} = \begin{bmatrix} 0 & -\frac{1}{2}w_x & -\frac{1}{2}w_y & -\frac{1}{2}w_z \\ \frac{1}{2}w_x & 0 & \frac{1}{2}w_z & -\frac{1}{2}w_y \\ \frac{1}{2}w_y & -\frac{1}{2}w_z & 0 & \frac{1}{2}w_x \\ \frac{1}{2}w_z & \frac{1}{2}w_y & -\frac{1}{2}w_x & 0 \end{bmatrix} \tag{3.24}$$

$$\frac{\partial \boldsymbol{f_q}}{\partial \boldsymbol{w}} = \begin{bmatrix} -\frac{1}{2}q_1 & -\frac{1}{2}q_2 & -\frac{1}{2}q_3 \\ \frac{1}{2}q_0 & -\frac{1}{2}q_3 & \frac{1}{2}q_2 \\ \frac{1}{2}q_3 & \frac{1}{2}q_0 & -\frac{1}{2}q_1 \\ -\frac{1}{2}q_2 & \frac{1}{2}q_1 & \frac{1}{2}q_0 \end{bmatrix} \tag{3.25}$$

$$\frac{\partial \boldsymbol{f_w}}{\partial \boldsymbol{w}} = \begin{bmatrix} 0 & -\frac{5}{8}w_z & -\frac{5}{8}w_y \\ \frac{5}{8}w_z & 0 & \frac{5}{8}w_x \\ 0 & 0 & 0 \end{bmatrix} \tag{3.26}$$

Eq. (3.24) and (3.25) are independent of system parameters, Eq. (3.26) is simplified for only principal axes of inertia and includes constants which are dependent on the inertia tensor. The factor $\frac{5}{8}$ in Eq. (3.26) will need to change if the inertia tensor is changed.

For the measurement equation:

$$\boldsymbol{g} = \begin{bmatrix} \bar{\boldsymbol{q}} \\ [[\dot{\boldsymbol{\omega}}\times](\boldsymbol{r}) + [\boldsymbol{\omega}\times]^2(\boldsymbol{r})] + \boldsymbol{a}_{LB} \end{bmatrix} \tag{3.27}$$

The Jacobian can similarly be divided into simpler Jacobians:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{g_q}}{\partial \boldsymbol{q}} & 0_{4\times 3} & 0_{4\times 3} \\ 0_{3\times 4} & \frac{\partial \boldsymbol{g_a}}{\partial \boldsymbol{w}} & \frac{\partial \boldsymbol{g_a}}{\partial \boldsymbol{a}} \end{bmatrix} \tag{3.28}$$

The three non-zero Jacobians are:

$$\frac{\partial \boldsymbol{g_q}}{\partial \boldsymbol{q}} = I_{4x4} \tag{3.29}$$

$$\frac{\partial \boldsymbol{g_a}}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{13}{8}w_z\delta r_z + \delta r_y w_y & -2w_y\delta r_x + \delta r_y w_x & \frac{13}{8}\delta r_z w_x - 2\delta r_x w_z \\ w_y\delta r_x - 2\delta r_y w_x & \frac{13}{8}w_z\delta r_z + w_x\delta r_x & \frac{13}{8}\delta r_z w_y - 2\delta r_y w_z \\ \frac{3}{8}\delta r_x w_z - 2\delta r_z w_x & \frac{3}{8}\delta r_y w_z - 2\delta r_z w_y & \frac{3}{8}w_x\delta r_x + \frac{3}{8}\delta r_y w_y \end{bmatrix} \tag{3.30}$$

$$\frac{\partial \boldsymbol{g_a}}{\partial \boldsymbol{a}} = \mathbf{I_{3\times 3}} \tag{3.31}$$

The basic principles of the Extended Kalman Filter are the same as those of the Kalman Filter. They are redesigned and applied towards nonlinear systems. Due to

the linearization that occurs, the EKF is not an optimal filter More accurately it is the optimal first order Taylor series approximation of the minimum error variance filter.

2.3 H_∞ Filter

Due to the Kalman Filter's great success in the 1970s with aerospace applications, there were attempts to apply it to industrial processes. Many of the assumptions necessary for the success of the KF could not be made for industrial state estimation problems. The primary issue was that the system models are not nearly as accurate as those used in aerospace applications [35]. This led to the need for a more robust filter which could more effectively take into account modeling error and noise uncertainty. The product is the H_∞ Filter. Also known as the "minimax" filter, the H_∞ Filter is designed to minimize the worst case estimation error. This contrasts the KF's minimization of the expected value of the variance of the estimation error. In addition to minimizing a different error quantity, the H_∞ Filter makes no assumptions regarding the characterization of either of the process or measurement noise, Q and R, respectively. A thorough derivation of the H_∞ Filter is provided in [35].

Consider the cost function in Eq. (3.32):

$$\boldsymbol{J} = \frac{\sum_{k=0}^{N-1} \|\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k\|^2_{\boldsymbol{S}_k}}{\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\|^2_{\boldsymbol{P}_0^{-1}} + \sum_{k=0}^{N-1} \left(\|\boldsymbol{\eta}_k\|^2_{\boldsymbol{Q}_k^{-1}} + \|\boldsymbol{\nu}_k\|^2_{\boldsymbol{R}_k^{-1}} \right)} \tag{3.32}$$

By minimizing J, the worst case estimation error is also minimized. In practice, however, obtaining the function minimum is not practical. Instead, it is constrained by a user-determined set of bounds. Using Lagrange multipliers for this minimization

process results in the equations describing the H_∞ algorithm below. For the linear system described as:

$$\begin{aligned} \boldsymbol{x}_{x+1} &= \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{\eta}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}_k \boldsymbol{x}_k + \boldsymbol{\nu}_k \\ \boldsymbol{z}_k &= \boldsymbol{L}_k \boldsymbol{x}_k \end{aligned} \tag{3.33}$$

Where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the same state and measurement vectors as previously seen, $\boldsymbol{z} \in \Re^{10\times 1}$ is a linear combination of the states being estimated; for this research all states are required independently and thus $\boldsymbol{L} \in \Re^{10\times 10}$ is the identity matrix. The H_∞ algorithm is such that

$$\begin{aligned} \bar{\boldsymbol{S}}_k &= \boldsymbol{L}_k^T \boldsymbol{S}_k \boldsymbol{L}_k \\ \boldsymbol{G}_k &= \boldsymbol{P}_k[\boldsymbol{I} - \boldsymbol{\theta}\bar{\boldsymbol{S}}_k \boldsymbol{P}_k + \boldsymbol{C}_k \boldsymbol{R}_k^{-1} \boldsymbol{C}_k \boldsymbol{P}_k]^{-1} \boldsymbol{C}_k^T \boldsymbol{R}_k^{-1} \\ \hat{\boldsymbol{x}}_{k+1} &= \boldsymbol{A}_k \hat{\boldsymbol{x}}_k + \boldsymbol{A}_k \boldsymbol{G}_k (\boldsymbol{y}_k - \boldsymbol{C}_k \hat{\boldsymbol{x}}_k) \\ \boldsymbol{P}_{k+1} &= \boldsymbol{A}_k \boldsymbol{P}_k [\boldsymbol{I} - \boldsymbol{\theta}\bar{\boldsymbol{S}}_k \boldsymbol{P}_k + \boldsymbol{C}_k^T \boldsymbol{R}_k^{-1} \boldsymbol{C}_k \boldsymbol{P}_k]^{-1} \boldsymbol{A}_k^T + \boldsymbol{Q}_k \end{aligned} \tag{3.34}$$

The H_∞ Filter presented in [35] is similar to the Kalman Filter described in Eq. (3.14) with the addition of several filter parameters allowing for much greater design flexibility than for the EKF. Filter parameters added are:

- S_k which allows the weighting of states in the cost function
- L defines the linear combination of states being estimated
- $1/\theta$ defines the bound on the cost function

In order for the algorithm to provide the expected minimization, the following condition must be met:

$$\boldsymbol{P}_k^{-1} - \boldsymbol{\theta}\bar{\boldsymbol{S}}_k + \boldsymbol{H}_k^T \boldsymbol{R}_k^{-1} \boldsymbol{H}_k > 0 \tag{3.35}$$

The H_∞ Filter is an EKF modified for robustness. Q and R are no longer strictly prescribed as the system and sensor noise variances. Rather, if there is knowledge about them, this additional information can be incorporated, although this is not necessary. An additional term is added to the EKF equations to bound the cost function, where the limit is set to $1/\theta$. Too large a θ will result in a unrealistic bound and may cause the estimator to go unstable.

Although not commonly described as such, the H_∞ filter may also be similarly modified to accommodate nonlinear processes. Given the following nonlinear system:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}_k) + \boldsymbol{\eta}_k \\ \boldsymbol{y}_k &= \boldsymbol{g}_k(\boldsymbol{x}_k) + \boldsymbol{\nu}_k \end{aligned} \tag{3.36}$$

the H_∞ Filter can be modified to allow for nonlinear state estimation as follows:

$$\begin{aligned} \boldsymbol{P}_{0,0} &= Var(\boldsymbol{x}_0) \\ \hat{\boldsymbol{x}}_0 &= E(\boldsymbol{x}_0) \end{aligned}$$

$$\begin{aligned} \boldsymbol{P}_{k,k-1} &= \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right] \boldsymbol{P}_{k-1,k-1} \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right]^T + \boldsymbol{Q}_{k-1} \\ \hat{\boldsymbol{x}}_{k|k-1} &= \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}) \end{aligned} \tag{3.37}$$

$$\begin{aligned} \boldsymbol{P}_{k,k} &= \left[\boldsymbol{I} + \theta \bar{\boldsymbol{S}} \boldsymbol{P}_{k,k-1} + \left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T \boldsymbol{R}_k^{-1} \left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right] \boldsymbol{P}_{k,k-1}\right]^{-1} \\ \boldsymbol{G}_k &= \boldsymbol{P}_{k,k} \left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T \boldsymbol{R}_k^{-1} \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{G}_k(\boldsymbol{y}_k - \boldsymbol{g}_k(\hat{\boldsymbol{x}}_{k|k-1})) \end{aligned}$$

It should be noted that if the minimization constraint θ is set to zero, implying a constraint bound of infinity, the H_∞ filter is equivalent to that of the Kalman Filter.

2.4 Sliding Mode Observer

Although the standard form of the Sliding Mode Observer (SMO) is is not an optimized filter and is susceptible to measurement noise, the SMO does offer significant computational savings over that of optimal filters by using a constant correctional gain.

The SMO assumes the following nonlinear system:

$$\begin{aligned} \boldsymbol{x_{k+1}} &= \boldsymbol{f_k(x_k)} + \boldsymbol{B_k u_k} + \boldsymbol{\eta_k} \\ \boldsymbol{y_k} &= \boldsymbol{g_k(x_k)} + \boldsymbol{\nu_k} \end{aligned} \tag{3.38}$$

The Sliding Mode Observer algorithm is similar to previous filters, however, no error covariance matrix is computed:

$$\boldsymbol{\hat{x}_0} = E(\boldsymbol{x}_0)$$

$$\begin{aligned} \boldsymbol{\hat{x}_{k|k-1}} &= \boldsymbol{f(\hat{x}_{k-1})} \\ \boldsymbol{\tilde{y}} &= \boldsymbol{y_k} - \boldsymbol{g(\hat{x}_{k|k-1})} \\ \boldsymbol{1(S)} &= \boldsymbol{sat(\alpha \tilde{y})} \\ \boldsymbol{\hat{x}_{k|k}} &= \boldsymbol{\hat{x}_{k|k-1}} + \boldsymbol{G_{SMO}}\ \boldsymbol{\tilde{y}} - \boldsymbol{K_{SMO}}\ \boldsymbol{1(S)} \end{aligned} \tag{3.39}$$

Here, a switching term, $\boldsymbol{K_{SMO}1(S)}$, is incorporated for added robustness against model uncertainties and is a function of a sliding surface $\boldsymbol{S}$. The constant gain portion can be designed using a Luenberger observer, or a steady state Kalman Filter gain. The switching term can be defined by various functions, the simplest being the signum function, or more practically the saturation function. The saturation function allows significant reduction in chatter, as well as attenuation of noise since each correction

has a predefined maximum update magnitude.

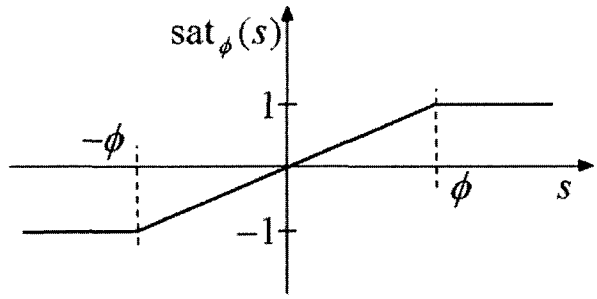


Figure 3.1: Saturation function

Although the equations for the algorithm are somewhat simpler than those of the Kalman or H_∞ filters, more work is performed designing the constant gain matrix, the switching term and the sliding surface. True sliding can occur in continuous-time systems due to the available infinite frequency of the switching function. The SMO in this research, on the other hand, is applied in discrete-time and, therefore, has limited switching frequency. This type of sliding is referred to as "quasi-sliding." The condition for quasi-sliding is given as [38]

$$\boldsymbol{S}(k)[\boldsymbol{S}(k+1) - \boldsymbol{S}(k)] < \boldsymbol{0} \tag{3.40}$$

Unlike optimal filters, convergence of the discrete-time SMO estimates can be guaranteed if

$$\boldsymbol{S}(k+1) < \boldsymbol{S}(k) \tag{3.41}$$

3 Batch Calibration Routine

Issues with the model uncertainty, specifically of the inertia tensor, lead to the design of an additional filter. If the inertia tensor is not perfectly known, then the dynamic filters' lumped bias estimates result in an offset error, thus yielding an incorrect lumped bias estimate. Although a solution is proposed for the dynamic filters in the future work chapter, this research also develops a method that is robust against this specific parametric uncertainty and is presented in this section. Since the MMS s/c attitude dynamics Eq. (3.42) is greatly affected by its inertia tensor, another dynamic relationship is used to calculate the s/c body rate:

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}\left(\sum \boldsymbol{T} - [\boldsymbol{J}\boldsymbol{\omega}\times]\boldsymbol{\omega}\right) \tag{3.42}$$

The proposed technique is to estimate the body rates using quaternion kinematic relations as shown below.

$$\dot{\bar{\boldsymbol{q}}} = \frac{1}{2}\bar{\boldsymbol{\omega}}_{b/o} \otimes \bar{\boldsymbol{q}} = \begin{bmatrix} -\frac{1}{2}(\omega_x q_1 + \omega_y q_2 + \omega_z q_3) \\ \frac{1}{2}(\omega_x q_0 + \omega_z q_2 - \omega_y q_3) \\ \frac{1}{2}(\omega_y q_0 - \omega_z q_1 + \omega_x q_3) \\ \frac{1}{2}(\omega_z q_0 + \omega_y q_1 - \omega_x q_2) \end{bmatrix} \tag{3.43}$$

Note that these relations are not directly affected by the errors in the inertia tensor or disturbance torques. As a result, using collected quaternion measurements, one can determine these angular rates by solving for them in Eq. (3.43), assuming the time derivative is known. Since the quaternion is a smooth function one can fit a polynomial spline to a collection of measurements to estimate the quaternion and its derivative while minimizing the effects of noise.

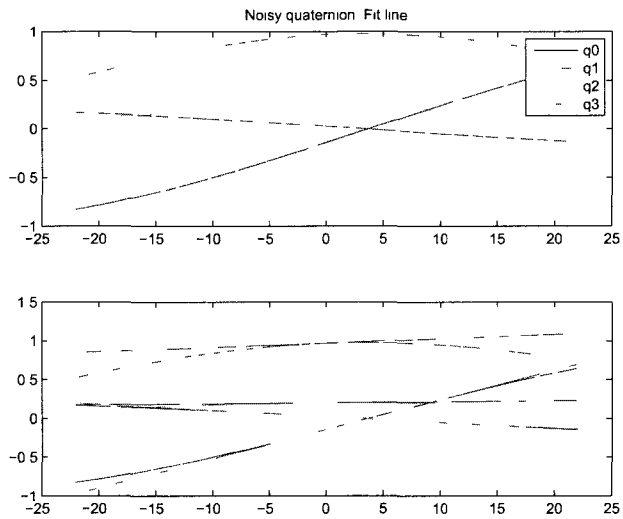


Figure 3.2: Quaternion fit and derivative lines

Figure 3.2 shows the spline on top of the measured quaternion in the top plot and the estimated slope at point 0 in the bottom plot. The n^{th} order fit is performed on $2M+1$ measurements using the method of least squares. The equation solved is compiled as,

$$\begin{bmatrix} (-M)^n & (-M)^{n-1} & \cdots & (-M)^0 \\ \vdots & \vdots & & \vdots \\ 0^n & 0^{n-1} & \cdots & 0^0 \\ \vdots & \vdots & & \vdots \\ M^n & M^{n-1} & \cdots & M^0 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} q_m(k=-M) \\ \vdots \\ q_m(k=0) \\ \vdots \\ q_m(k=M) \end{bmatrix} \tag{3.44}$$

The $2M+1$ measurements are indexed $-M$ through M and assembled to create the rightmost vector. For a given M and n the leftmost matrix will remain the same and must only be compiled once. The center vector contains the coefficients for the nth order fit. Since the indices are centered about the quaternion of interest, $q_m(k=0)$,

the filtered quaternion is a_0 and the slope at that point is a_1. The coefficient a_1 must take into account the equivalent δt per each unit index, k, to scale to $\dot{q}_m(k=0)$

Both the number of measurements used for each spline fit and the order of polynomial used can be adjusted for desired accuracy and noise attenuation. By performing this spline fit sequentially along the collected data, the body rate can be estimated. Since the time derivatives of the angular rates rely on the inertia tensor as well, and this matrix is being avoided, the same method of spline fitting can be applied to the angular rate estimate to smooth it as well as determine the slope at each point, thus providing an estimate of $\dot{\boldsymbol{\omega}}$, which is required for estimating the acceleration.

Using the nominal acceleration equation and the newly estimated $\hat{\boldsymbol{\omega}}$ and $\hat{\dot{\boldsymbol{\omega}}}$,

$$\boldsymbol{a}_{nom} = [\hat{\dot{\boldsymbol{\omega}}}\times]\boldsymbol{r} + [\hat{\boldsymbol{\omega}}\times]^2\boldsymbol{r} \tag{3.45}$$

An estimate of the nominal acceleration can be made, which when subtracted from the measured acceleration allows the calculation of the lumped bias.

Chapter IV

ANALYTICAL SIMULATIONS

1 Simulation Conditions

Using the dynamic equations presented in Chapter I, a simulation is created using numerical simulation software. The differential equation solution is found using a variable step Dormand-Prince solver. Both the calculated quaternion and body rate are recorded at a sample rate of 0.25Hz based on simulation time. The full, nonlinear dynamic equations are used,

$$\dot{\bar{q}} = \frac{1}{2}\bar{\omega}_{b/o} \otimes \bar{q}(t) \tag{4.1}$$

$$\dot{\omega} = \boldsymbol{J}^{-1}(\sum \boldsymbol{T} - [\boldsymbol{J}\boldsymbol{\omega}\times]\boldsymbol{\omega}) \tag{4.2}$$

Block diagrams representing the numerical solutions are:

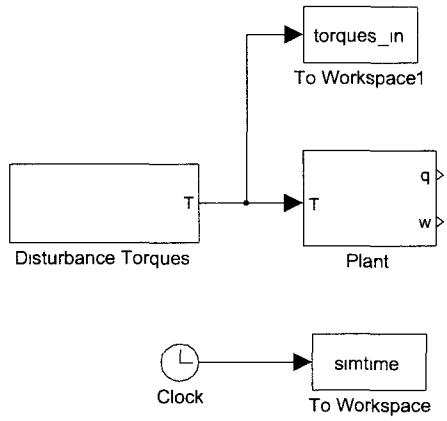


Figure 4.1: Simulink dynamic simulation

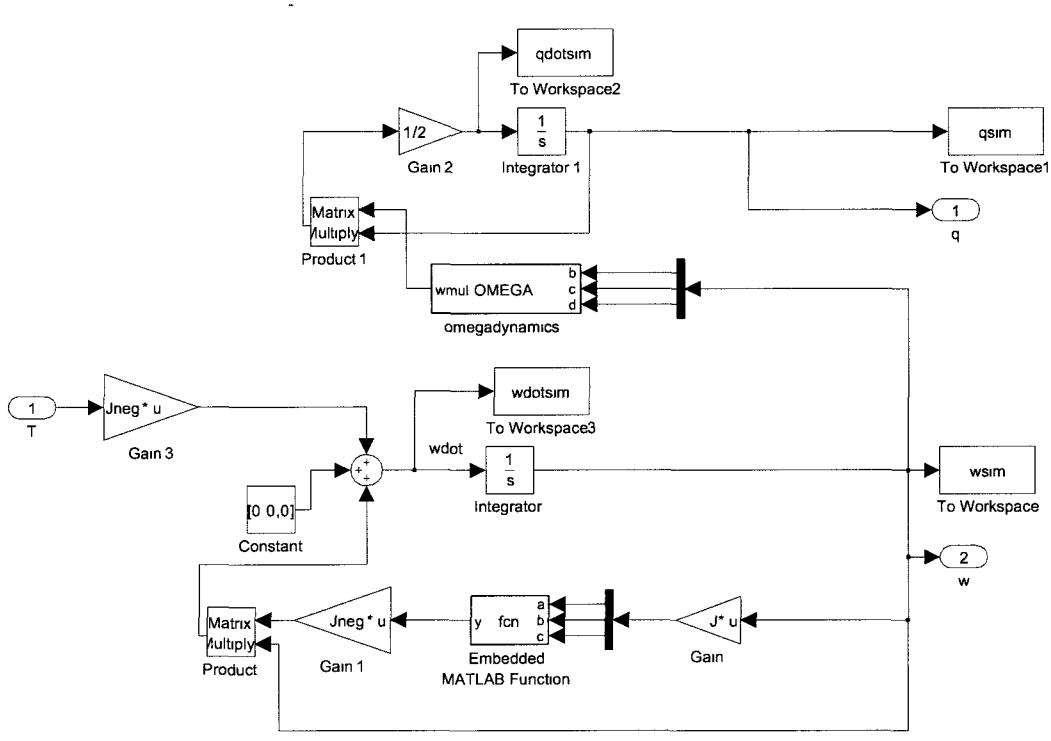


Figure 4.2: Simulink subsystem: plant

Per MMS mission requirements, the initial spin rates have a magnitude of three

revolutions per minute. If the angular velocity vector is not parallel to the major or minor axes of the s/c then the body is nutating and the angular offset from parallel is the coning angle. The magnitude of the nutation and coning angles are randomly generated within a set of bounds. The same bound definitions are used as were defined in [39] and [40]. Eq. (4.3) shows the construction of the initial body rate. To maintain as fair a comparison as possible, a random w_i is generated and used for all the filters/estimators.

$$\boldsymbol{\omega} = \begin{bmatrix} 3\,|sin\ \kappa|\boldsymbol{u}_{xy} \\ 3\ cos\ \kappa \end{bmatrix} rpm \tag{4.3}$$

Where κ is the coning angle selected from a normal distribution with zero mean and a standard deviation of 0.2 deg. The vector $\boldsymbol{u}_{xy}$ is the unit vector direction of the angular velocity in the x-y plane, constructed from a uniform distribution [39].

1.1 Simulation Inertia Tensor

Two different inertia tensors are defined; one for the compact spacecraft, and another for the spacecraft with SDP and ADP booms deployed. For the majority of the simulations the compact inertia tensor is used since it has a higher sensitivity to corruptions and disturbances, offering the more conservative of results.

The true compact inertia tensor is given as

$$\boldsymbol{J} = \begin{bmatrix} 783.35 & -12.28 & -4.84 \\ -12.28 & 803.79 & -7.67 \\ -4.84 & -7.67 & 1332.99 \end{bmatrix} kg \cdot m^2 \tag{4.4}$$

The s/c axes may not be aligned with the s/c principal axes of inertia. Any resulting cross products of inertia are considered as unknown parametric uncertainties.

Hence, the nominal inertia tensor, $\hat{\boldsymbol{J}}$, used by filters in this research is

$$\hat{\boldsymbol{J}} = \begin{bmatrix} 800 & 0 & 0 \\ 0 & 800 & 0 \\ 0 & 0 & 1300 \end{bmatrix} kg \cdot m^2 \tag{4.5}$$

True and nominal inertia tensors for the s/c with ADP and SDP booms deployed are

$$\boldsymbol{J_2} = \begin{bmatrix} 3160.32 & -225.92 & -4.82 \\ -225.92 & 3135.18 & -7.42 \\ -4.82 & -7.42 & 5475.89 \end{bmatrix} kg \cdot m^2 \tag{4.6}$$

$$\hat{\boldsymbol{J}}_{\boldsymbol{2}} = \begin{bmatrix} 3100 & 0 & 0 \\ 0 & 3100 & 0 \\ 0 & 0 & 5500 \end{bmatrix} kg \cdot m^2 \tag{4.7}$$

1.2 Sensor Output and Simulation

Sensor data is synthesized using the states generated by the numerical modeling software using the equations in section I:6. Four corruptions, which are unknown to the estimator, are present in the accelerometer model. Using definitions used by Markley et. al. in [40], each is randomly selected from a normal distribution with a mean of zero and standard deviations defined as:

$\boldsymbol{\nu_a}$ Gaussian noise, $\sigma = 1.465 \times 10^{-4}\ m/s^2$

$\boldsymbol{\delta r}$ Center of mass shift, $\sigma = 5\ cm$

$\boldsymbol{a_b}$ Accelerometer bias, $\sigma = 10^{-5}\ m/s^2$

$\boldsymbol{\delta}$ Orthogonal misalignment, $\sigma = 20\ arcsec$

Noise added to the sensor models is generated using a Gaussian description, with a mean of zero and the standard deviations below.

$$\boldsymbol{\sigma}_q = \begin{bmatrix} 0.0486 \\ 0.0485 \\ 0.1145 \\ 0.1143 \end{bmatrix} \times 10^{-3} \tag{4.8}$$

Standard deviations of the noise for the quaternion are defined for each component since the noise is actually defined as a misalignment of Euler angles. For the mission definition, the misalignment standard deviations convert to those shown in Eq. (4.16).

$$\sigma_a = 10^{-4} \ m/s^2 \tag{4.9}$$

1.3 Single and Multi-rate Filter Results

1.3.1 4Hz Measurement and Propagation

This section addresses limitations of the star tracker measurement sampling rate by propagating filters/estimators dynamic state estimate at a higher rate. Without loss of generality the EKF is used to exemplify the issues of the slow sampling rate as well as to show the resulting benefits of higher frequency propagation. The star tracker is the limiting sensor on the spacecraft, providing measurements at a rate of only 4Hz. The standard Extended Kalman Filter propagates its state estimates once between each measurement update. The EKF results without modeling errors/uncertianties or sensor noise are shown in Figures 4.3 through 4.5 for estimates of attitude, angular body rates and lumped biases, respectively. Each figure shows the estimated values, the actual values and the resulting estimate errors.

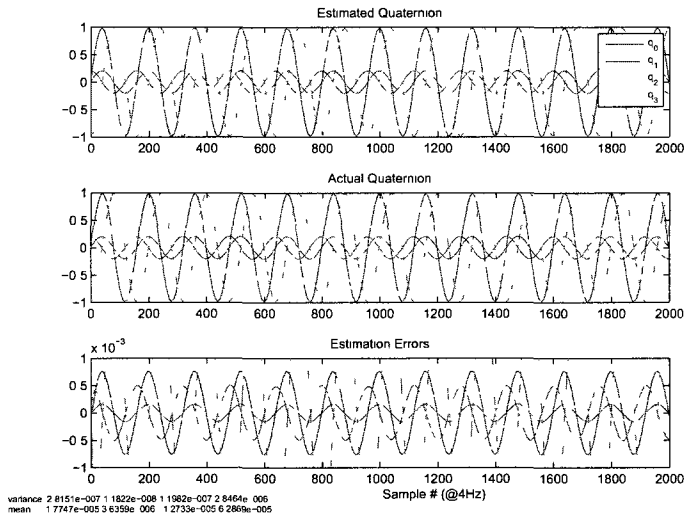


Figure 4.3: EKF estimated quaternion for 1x propagation

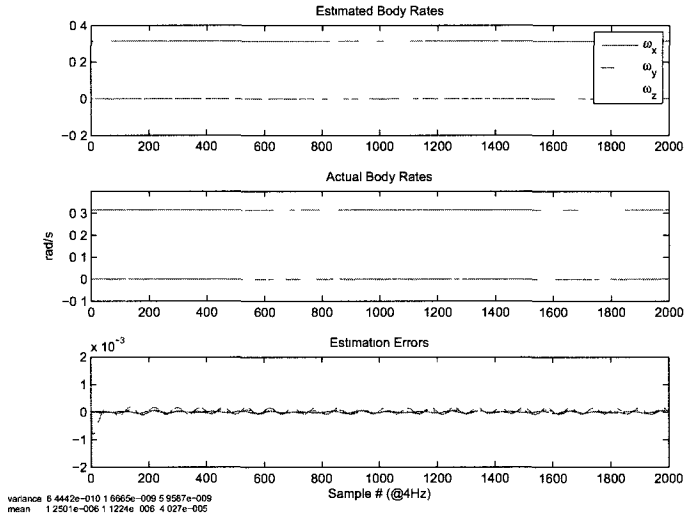


Figure 4.4: EKF estimated body rates for 1x propagation

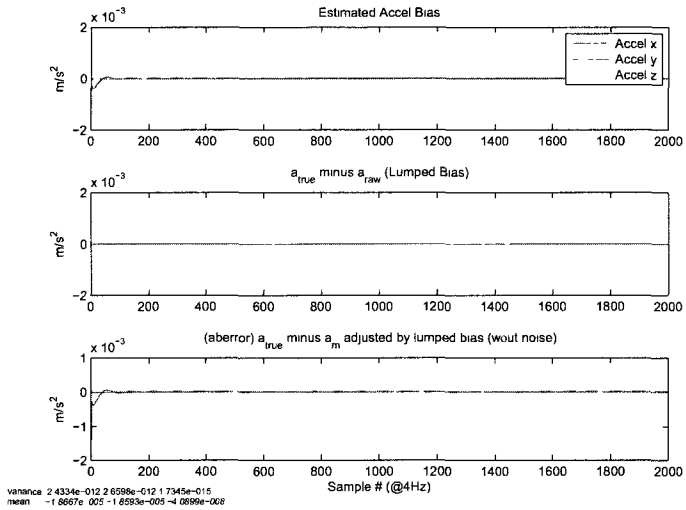


Figure 4.5: EKF estimated accelerometer bias for 1x propagation

In Figure 4.3 significant errors in the estimated quaternion are present even though the EKF has perfect knowledge of the system. This is due to the discretization of the nonlinear system model using a δt of 0.25 s. Errors in $\hat{\omega}$ and $\hat{a}_{LB}$ are also present. The accelerometer lumped bias estimate error is already at the limit of MMS mission specifications without any filter errors introduced.

1.3.2 4Hz Measurement Updates and 400Hz Propagation Rate

To address the issues of estimate errors caused by the slow propagation rate within the filter, the estimation propagation frequency is increased, thus significantly reducing estimation errors in the highly nonlinear s/c system. The following Kalman Filter

steps are calculated 100 times in between each measurement update:

$$\boldsymbol{P}_{k,k-1} = \left[\frac{\partial \mathbf{f}_{\mathbf{k-1}}}{\partial \mathbf{x}_{\mathbf{k-1}}}(\hat{\mathbf{x}}_{k-1})\right] \boldsymbol{P}_{k-1,k-1} \left[\frac{\partial \mathbf{f}_{\mathbf{k-1}}}{\partial \mathbf{x}_{\mathbf{k-1}}}(\hat{\mathbf{x}}_{k-1})\right]^{T} + \boldsymbol{Q}_{k-1}$$
$$\hat{\boldsymbol{x}}_{k|k-1} = \mathbf{f}_{\mathbf{k-1}}(\hat{\mathbf{x}}_{\mathbf{k-1}}) \tag{4.10}$$

The result is a multi-rate estimation algorithm propagating dynamic estimates at 400Hz and updating estimates with measurements at 4Hz.

Figures 4.6 through 4.8 show the improved accuracy in state and parameter estimates for the proposed multirate estimation technique with a propagation rate that is 100 times that of the star tracker measurement output.

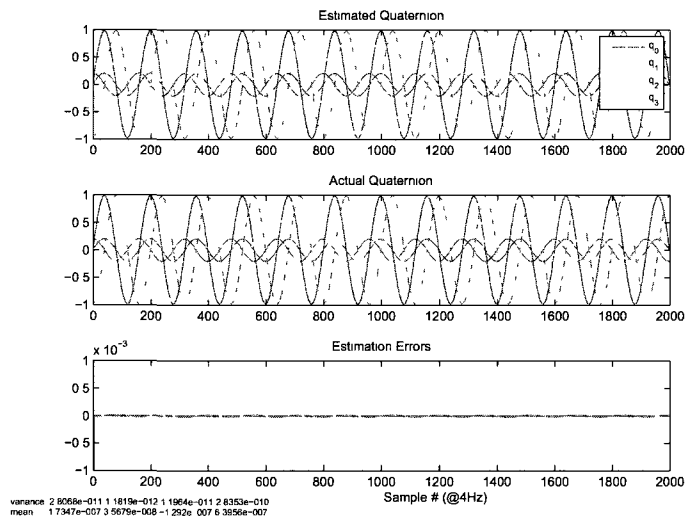


Figure 4.6: EKF estimated quaternion for 100x propagation

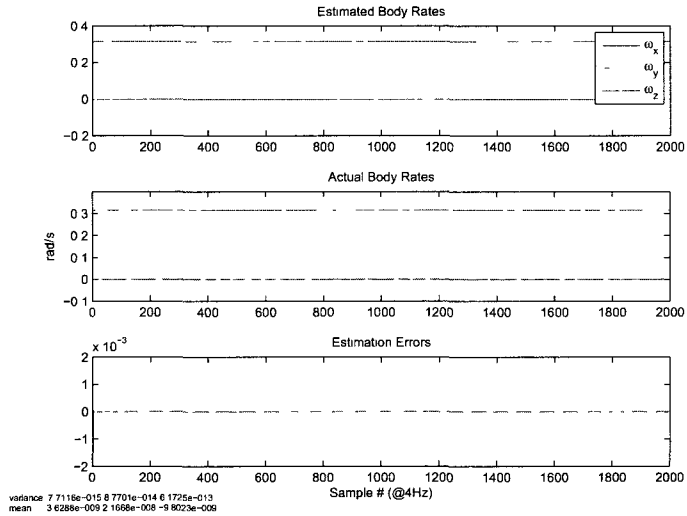


Figure 4.7: EKF estimated body rates for 100x propagation

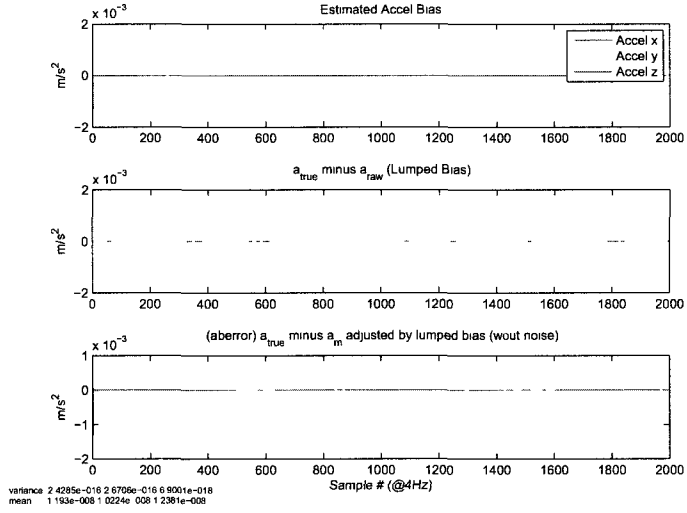


Figure 4.8: EKF estimated accelerometer bias for 100x propagation

Estimate error variances and means are both reduced by at least two orders of magnitude through the implementation of a 400Hz propagation rate allowing full

utilization of the system knowlege. Additional benefit from increased rates is investigated, however, errors did not noticeably improve. In addition to improving estimate accuracies by introducing such a small discretization timestep, it is not necessary to compute the discrete system uncertainty covariance matrix $\boldsymbol{Q}_k$,

$$\boldsymbol{Q}_k = E[\boldsymbol{\eta}_k \boldsymbol{\eta}_k^T] = \int_0^{\Delta t} \boldsymbol{A}_k(\Delta t - \tau)\boldsymbol{Q}_c \boldsymbol{A}_k(\Delta t - \tau)^T \delta\tau \tag{4.11}$$

where $\boldsymbol{A}_k$ is the discretized system matrix and $\boldsymbol{Q}_c$ is the continuous system uncertainty covariance matrix. This integration is now performed numerically within the filter estimate covariance propagation.

2 Analytical Filter/Estimation Results

2.1 Cascading filter results

The first stage in this study involves a feasibility study to observe the efficacy of using cascading filters to accurately estimate the accelerometer misalignment. Continuing work previously initiated by NASA engineers, an Extended Kalman Filter is used to obtain state estimates, which, in turn is used to determine the accelerometer misalignment via method of least squares. The resulting misalignment estimate errors are significant and do not meet NASA MMS requirements.

The preliminary tests were to determine the expected misalignment error, given an error in the EKF estimate. Numerous simulations were run using varying EKF output errors, the root-mean-square of the resulting misalignment errors were plotted against the root-mean-square of the input errors to find the largest allowable EKF

error.

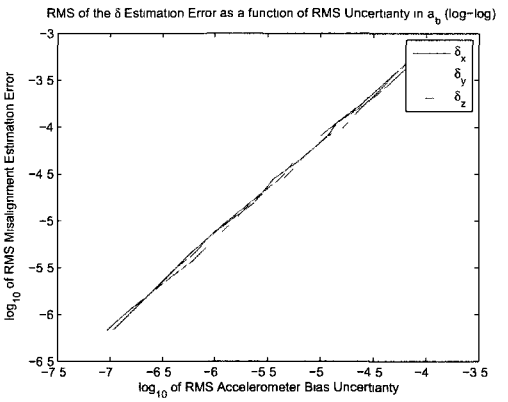


Figure 4.9: RMS misalignment error due to RMS of accelerometer bias error

Output accuracies of the EKF for the accelerometer bias estimation are on the order of 10^{-5}. Using Figure 4.9 the misalignment estimation error is expected to be on the order of 10^{-4} for δ_x and δ_y

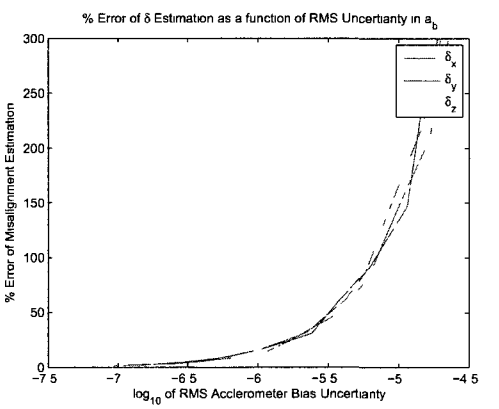


Figure 4.10: Percent misalignment error due to RMS of accelerometer bias error

Subject to the EKF estimate outputs predicted and the nominal misalignment used for simulations, the percent error of the accelerometer's orthogonal misalignment

estimation error will be over 300% of the nominal value.

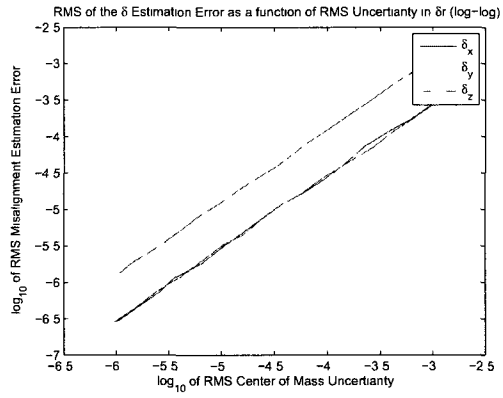


Figure 4.11: RMS misalignment error due to RMS of center of mass error

Output accuracies of the EKF for the accelerometer location shift are on the order of 10^{-4} for δr_x, δr_y and δr_z. The corresponding estimation errors for δ_x, δ_y and δ_z are 10^{-5}, 10^{-5} and 10^{-4} respectively.

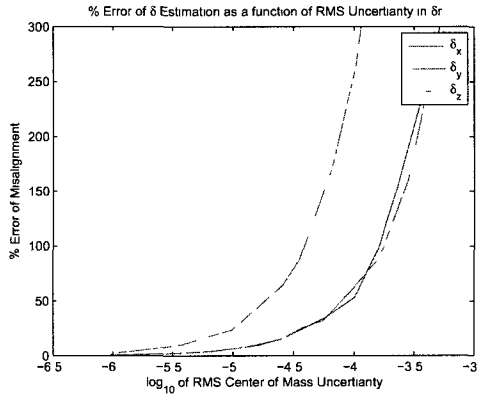


Figure 4.12: Percent misalignment error due to RMS of center of mass error

Subject to its inputs, the estimate of the least squares algorithm will have a 250% error for δ_z. Low sensitivity of δ_x and δ_y to $\boldsymbol{\delta r}$ allows estimates of these to within 5%

accuracy.

It should be noted that the percent error of the misalignment estimation error is entirely dependent on the size of the misalignment used in the numerical simulation. If the true misalignment is large, then the cascading filter may provide an estimate with a smaller percent error. If the true misalignment is small, however, the percent error will easily be larger than 100%.

An iterative procedure, allowing the EKF to account for the estimated misalignment, may have provided more accurate results. The EKF's assumption of a perfectly known inertia tensor may not be valid. Therefore, alternative methods are investigated that do not require this assumption.

2.2 Known Inertia Tensor Results

The first of many dynamic filter results are presented in this section. Each figure contains the variance and mean of the estimation errors on the bottom left. Although very small, these numbers are included to preserve the statistical analysis of the error for each individual simulation. All relevant statistics used for determining each filter's performance and comparison are included in tables following each different simulated condition. The tabulated statistics include mean or root-mean-square values of error variances or means respectively, from 100 simulations preformed with random initial conditions and corruptions selected from Gaussian distributions. For the plots presenting visual results the initial conditions, corruptions and noise vectors are all identical.

In this section of results, three general test scenarios are observed under assumed perfect knowledge of the MMS s/c inertia tensor:

Case 1 state and parameter estimation including all three sources of bias errors without sensor noise

Case 2 state and parameter estimation including all three sources of bias errors with sensor noise

Case 3 state and parameter estimation including all three sources of bias errors with 10 times the expected sensor noise

All three filters/observers are subjected to each of the three scenarios. The first scenario tests whether the filter is capable of the required task provided perfect measurements, the second scenario tests the filter under expected mission conditions, and the third scenario assesses the robustness of the filter/observer if additional noise is present.

2.2.1 Extended Kalman Filter for Known Inertia Tensor

For the EKF design the system uncertainty covariance matrix is:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_q & & \\ & \boldsymbol{Q}_w & \\ & & \boldsymbol{Q}_a \end{bmatrix} \tag{4.12}$$

$$\begin{aligned} \boldsymbol{Q}_q &= I_{4\times4} \times 10^{-11} \\ \boldsymbol{Q}_w &= I_{3\times3} \times 10^{-11} \\ \boldsymbol{Q}_a &= I_{3\times3} \times 10^{-20} \end{aligned} \tag{4.13}$$

and all other values for $\boldsymbol{Q}$ are 0.

The sensor noise covariance is:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_q & \\ & \boldsymbol{R}_a \end{bmatrix} \tag{4.14}$$

where $\boldsymbol{R}_q$ and $\boldsymbol{R}_a$ are both defined using the same sensor noise variance used to simulate the data and all other elements are 0. Such accurate knowledge of the noise statistics may not be known. This, however, provides the optimal EKF for comparison to the other filters:

$$\begin{aligned} \boldsymbol{R}_q &= I_{4\times 4} \times 10^{-11} \\ \boldsymbol{R}_a &= I_{3\times 3} \times 10^{-8} \end{aligned} \tag{4.15}$$

The expected noise standard deviations are,

$$\boldsymbol{\sigma}_q = \begin{bmatrix} 0.0486 \\ 0.0485 \\ 0.1145 \\ 0.1143 \end{bmatrix} \times 10^{-3} \tag{4.16}$$

$$\sigma_a = 10^{-4} m/s^2 \tag{4.17}$$

Figures 6.1 through 6.6 in Appendix A show the EKF results for Cases 1 and 3. Case 2 results are presented below,

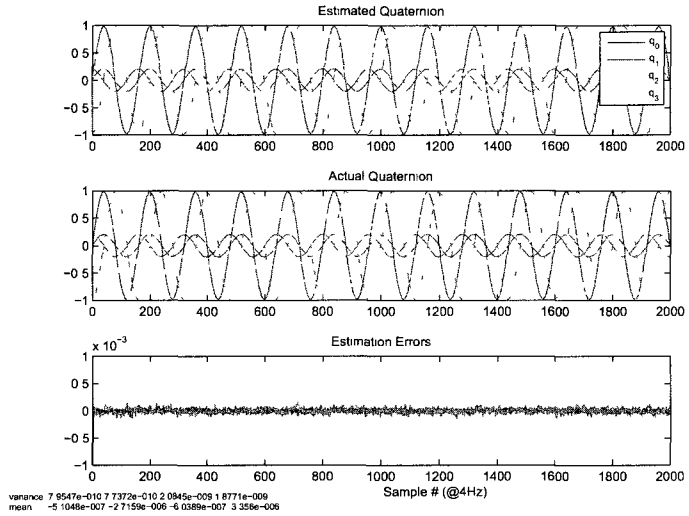


Figure 4.13: EKF $\hat{q}$ subjected to bias errors and measurement noise (known inertia)

Attitude estimation is expected to perform well since the star trackers are very accurate, this is the case and the error variance is reduced below sensor noise inputs.

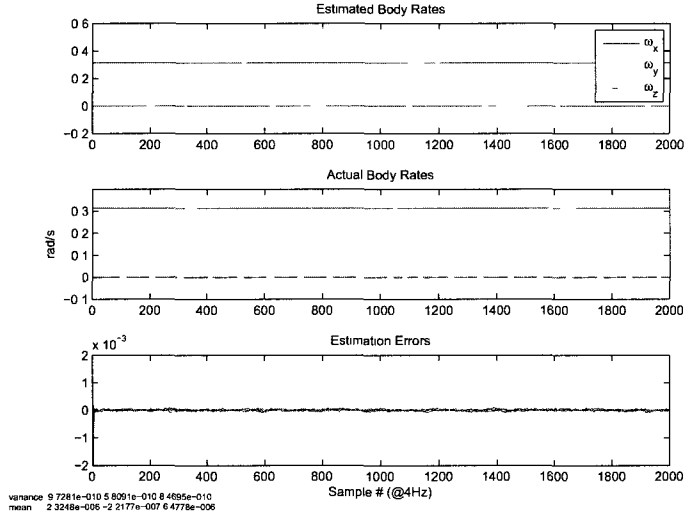


Figure 4.14: EKF $\hat{\omega}$ subjected to bias errors and measurement noise (known inertia)

Angular rate estimates have no sensor to directly compare to so a reduction in the error variance can not be evaluated. The errors do converge to zero quickly due to the variable Kalman Gain, and both error variance and mean are very small.

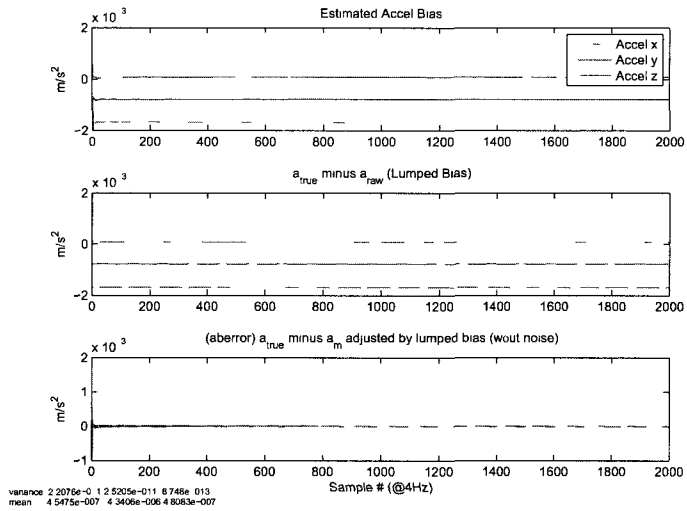


Figure 4.15: EKF $\hat{a}_{LB}$ subjected to bias errors and measurement noise (known inertia)

The most important state estimated for this research is the accelerometer bias, by modeling it as a constant with little uncertainty the accelerometer noise is nearly entirely attenuated.

The error variance of the later 50% of the filter output is calculated for 100 simulations with random initial conditions and accelerometer corruptions. The means of those 100 simulations are presented below to compare the average performance of each filter. Since the accelerometer lumped bias is the parameter of largest concern, and is expected to be constant, only the steady state results are analyzed.

Table 4.1: Means of the error variances of 100 trials (known inertia) - EKF

	q error variance	ω error variance $(rad/s)^2$	a_b error variance $(m/s^2)^2$
a_b r_c and Δ	$\begin{bmatrix} 0\ 0299 \\ 0\ 0050 \\ 0\ 0637 \\ 0\ 2918 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0\ 4379 \\ 0\ 2582 \\ 0\ 1577 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0\ 3293 \\ 0\ 3309 \\ 0\ 0100 \end{bmatrix} \times 10^{-9}$
a_b r_c and Δ with noise	$\begin{bmatrix} 0\ 0848 \\ 0\ 0845 \\ 0\ 2259 \\ 0\ 2207 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0\ 1703 \\ 0\ 1229 \\ 0\ 1260 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0\ 5802 \\ 0\ 5854 \\ 0\ 0207 \end{bmatrix} \times 10^{-9}$
a_b r_c and Δ with 10×noise	$\begin{bmatrix} 0\ 0810 \\ 0\ 0841 \\ 0\ 2085 \\ 0\ 1942 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0\ 9062 \\ 0\ 7200 \\ 0\ 9849 \end{bmatrix} \times 10^{-7}$	$\begin{bmatrix} 0\ 6462 \\ 0\ 6220 \\ 0\ 2172 \end{bmatrix} \times 10^{-9}$

The EKF error variances are very robust to sensor noise, the lumped bias specifically shows extremely small increases for 100× noise inputs.

Similarly to the error variances, the mean error is calculated from the later 50% of each simulations results. The RMS of the 100 mean errors is calculated and is used to show the filter's stead state performance. Mission tolerances are only defined for the accelerometer lumped bias for this work and are $10^{-5}\ m/s^2$.

Table 4.2: RMS of the mean errors for 100 trials (known inertia) - EKF

	mean error	ω mean error (rad/s)	a_b mean error (m/s^2)
a_b δr and Δ	$\begin{bmatrix} 0.1775 \\ 0.0477 \\ 0.1371 \\ 0.6292 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.6917 \\ 0.4426 \\ 0.4580 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.5163 \\ 0.5238 \\ 0.1161 \end{bmatrix} \times 10^{-6}$
a_b δr and Δ with noise	$\begin{bmatrix} 0.1612 \\ 0.1533 \\ 0.3544 \\ 0.3588 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.4013 \\ 0.2380 \\ 0.4609 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.4360 \\ 0.4010 \\ 0.3863 \end{bmatrix} \times 10^{-5}$
a_b δr and Δ with 10×noise	$\begin{bmatrix} 0.0011 \\ 0.0132 \\ 0.1021 \\ 0.0278 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.7737 \\ 0.4678 \\ 0.0053 \end{bmatrix} \times 10^{-3}$	$\begin{bmatrix} 0.0304 \\ 0.0618 \\ 0.3148 \end{bmatrix} \times 10^{-3}$

Both no noise and expected noise cases result in accelerometer lumped biases within the defined bounds. When the EKF designed for expected noise levels is subjected to 100× the noise power, estimate errors of all states suffer dramatically.

2.2.2 H_∞ Filter for Known Inertia Tensor

All common parameters of the H_∞ and EKF, such as Q and R, are defined identically. The additional H_∞ terms are designed to be tuned by the engineer and for this filter are defined as

$$k = \mathbf{10\times10} \tag{4.18}$$

L_k reduces the states required to be estimated as a linear combination of the filter states, since every state and parameter is required, L_k is set as an identity matrix.

$$\boldsymbol{Sk} = \begin{bmatrix} \mathbf{4\times4} & & \\ & \mathbf{10} \times \mathbf{3\times3} & \\ & & \mathbf{3\times3} \end{bmatrix} \tag{4.19}$$

Table 4.1: Means of the error variances of 100 trials (known inertia) - EKF

	q error variance	ω error variance $(rad/s)^2$	a_b error variance $(m/s^2)^2$
a_b δr and Δ	$\begin{bmatrix} 0.0299 \\ 0.0050 \\ 0.0637 \\ 0.2918 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0.4379 \\ 0.2582 \\ 0.1577 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0.3293 \\ 0.3309 \\ 0.0100 \end{bmatrix} \times 10^{-9}$
a_b δr and Δ with noise	$\begin{bmatrix} 0.0848 \\ 0.0845 \\ 0.2259 \\ 0.2207 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.1703 \\ 0.1229 \\ 0.1260 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.5802 \\ 0.5854 \\ 0.0207 \end{bmatrix} \times 10^{-9}$
a_b δr and Δ with 10×noise	$\begin{bmatrix} 0.0810 \\ 0.0841 \\ 0.2085 \\ 0.1942 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.9062 \\ 0.7200 \\ 0.9849 \end{bmatrix} \times 10^{-7}$	$\begin{bmatrix} 0.6462 \\ 0.6220 \\ 0.2172 \end{bmatrix} \times 10^{-9}$

The EKF error variances are very robust to sensor noise; the lumped bias specifically shows extremely small increases for 100× noise inputs.

Similarly to the error variances, the mean error is calculated from the later 50% of each simulations results. The RMS of the 100 mean errors is calculated and is used to show the filter's stead state performance. Mission tolerances are only defined for the accelerometer lumped bias for this work and are 10^{-5} m/s^2.

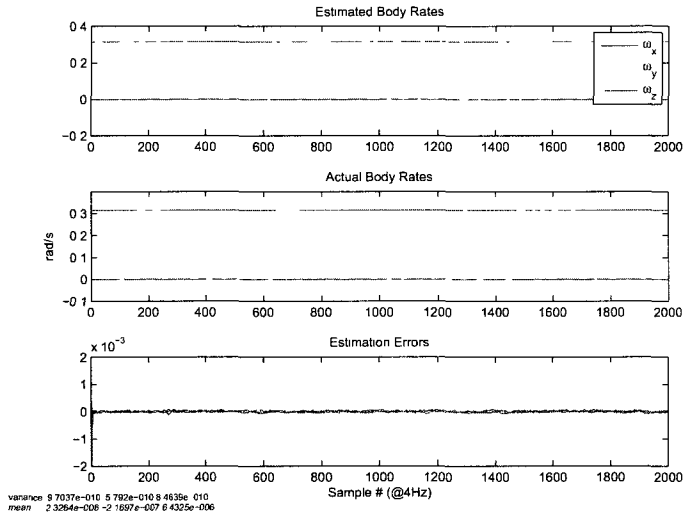


Figure 4.17: H_∞ $\hat{\omega}$ subjected to bias errors and measurement noise (known inertia)

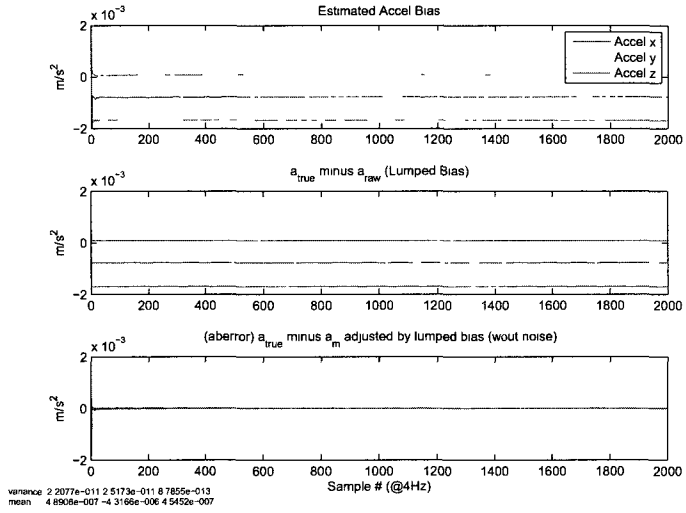


Figure 4.18: H_∞ $\hat{a}_{LB}$ subjected to bias errors and measurement noise (known inertia)

In the same fashion that EKF means of error variances and RMS of mean errors are calculated, values for the H_∞ are presented.

Table 4.3: Means of the error variances of 100 trials (known inertia) - H_∞

	q error variance	ω error variance $(rad/s)^2$	a_b error variance $(m/s^2)^2$
a_b δr and Δ	$\begin{bmatrix} 0.0298 \\ 0.0048 \\ 0.0598 \\ 0.2910 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0.4088 \\ 0.2384 \\ 0.1468 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0.3083 \\ 0.3089 \\ 0.0090 \end{bmatrix} \times 10^{-9}$
a_b δr and Δ with noise	$\begin{bmatrix} 0.0848 \\ 0.0848 \\ 0.2199 \\ 0.2186 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.1376 \\ 0.0994 \\ 0.1107 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.3531 \\ 0.3543 \\ 0.0208 \end{bmatrix} \times 10^{-9}$
a_b δr and Δ with 10×noise	$\begin{bmatrix} 0.0808 \\ 0.0828 \\ 0.2105 \\ 0.1933 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.0912 \\ 0.0719 \\ 0.1004 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.7291 \\ 0.7338 \\ 0.1871 \end{bmatrix} \times 10^{-9}$

Interestingly the error variances are most noticeably improved over the EKF results for Case 2, whereas, Case 1 and 3 show only minimal differences if not increases in error variance.

Table 4.4: RMS of the mean errors for 100 trials (known inertia) - H_∞

	mean error	ω mean error (rad/s)	a_b mean error (m/s^2)
a_b δr and Δ	$\begin{bmatrix} 0.1886 \\ 0.0591 \\ 0.1316 \\ 0.6576 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.6616 \\ 0.4843 \\ 0.3957 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.5399 \\ 0.4283 \\ 0.1169 \end{bmatrix} \times 10^{-6}$
a_b δr and Δ with noise	$\begin{bmatrix} 0.1584 \\ 0.1582 \\ 0.3528 \\ 0.3702 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.3821 \\ 0.2692 \\ 0.4510 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.3673 \\ 0.3830 \\ 0.3287 \end{bmatrix} \times 10^{-5}$
a_b δr and Δ with 10×noise	$\begin{bmatrix} 0.1568 \\ 0.1526 \\ 0.3763 \\ 0.3263 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.3162 \\ 0.2447 \\ 0.4714 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.3534 \\ 0.3804 \\ 0.3515 \end{bmatrix} \times 10^{-4}$

Similarly to the mean error variances, the RMS of the mean errors show the most noticeable decrease in error over the EKF for case 2. Both cases 1 and 2 allow estimation of the accelerometer lumped bias to values within mission tolerances.

2.2.3 Sliding Mode Observer

For the simulations performed, a steady state Kalman gain is used for $\boldsymbol{G}_{SMO}$:

$$\boldsymbol{G}_{SMO} = \begin{bmatrix} 0.6755 & -0.0103 & -0.0004 & -0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0103 & 0.7257 & -0.0001 & 0.0002 & -0.0000 & -0.0000 & -0.0000 \\ -0.0004 & -0.0001 & 0.7254 & 0.0110 & 0.0000 & -0.0000 & 0.0000 \\ -0.0000 & 0.0002 & 0.0110 & 0.6755 & 0.0000 & 0.0000 & 0.0000 \\ -0.0004 & 0.0016 & -0.0088 & -0.0019 & 0.0003 & -0.0000 & 0.0001 \\ -0.0019 & 0.0087 & 0.0016 & 0.0004 & -0.0000 & 0.0003 & 0.0001 \\ 0.0086 & 0.0018 & -0.0005 & 0.0020 & -0.0004 & -0.0004 & 0.0000 \\ 0.0133 & -0.0010 & 0.0053 & 0.0089 & 0.0088 & -0.0002 & -0.0000 \\ 0.0140 & -0.0042 & -0.0052 & 0.0067 & -0.0002 & 0.0088 & -0.0000 \\ 0.0008 & -0.0037 & 0.0011 & 0.0003 & -0.0000 & -0.0000 & 0.0091 \end{bmatrix} \tag{4.20}$$

The sliding gain is chosen after analysis of the system and measurement functions to be:

$$\boldsymbol{K}_{\boldsymbol{SMO}} = 0.075 \times [\boldsymbol{SK}_{\boldsymbol{q/q}}\ \boldsymbol{SK}_{\boldsymbol{q/w}}\ \boldsymbol{SK}_{\boldsymbol{a/ab}}] \tag{4.21}$$

Where $\boldsymbol{K}_{\boldsymbol{q/q}}$, $\boldsymbol{K}_{\boldsymbol{q/w}}$ and $\boldsymbol{K}_{\boldsymbol{a/ab}}$ are:

$$\boldsymbol{K}_{\boldsymbol{q/q}} = \boldsymbol{I}_{4\times 4} \tag{4.22}$$

$$\boldsymbol{K}_{\boldsymbol{q/w}} = \left[\frac{\partial \dot{\boldsymbol{q}}}{\partial \boldsymbol{\omega}}\right]^T = \begin{bmatrix} q_1 & -q_0 & -q_3 & q_2 \\ q_2 & q_3 & -q_0 & -q_1 \\ q_3 & -q_2 & q_1 & -q_0 \end{bmatrix} \tag{4.23}$$

$$\boldsymbol{K}_{\boldsymbol{a/ab}} = \boldsymbol{0}_{3\times 4} \tag{4.24}$$

The sliding surface $\boldsymbol{S}$ is the quaternion estimate error and allows the full angular body rate vector to be estimated even though a constant Kalman gain drives the

system.

$$\boldsymbol{S} = \tilde{\boldsymbol{y}} = \mathbf{0} \tag{4.25}$$

The switching function $\mathbf{1}(\boldsymbol{S})$ is chosen to be the saturation function with a boundary layer $\phi = 5 \times 10^{-7}$ such that

$$\mathbf{1}(\boldsymbol{S}) = sat\left(\frac{\boldsymbol{S}}{\phi}\right) \tag{4.26}$$

Figures 6.13 through 6.18 in Appendix A show the SMO results for Cases 1 and 3. Case 2 is presented below,

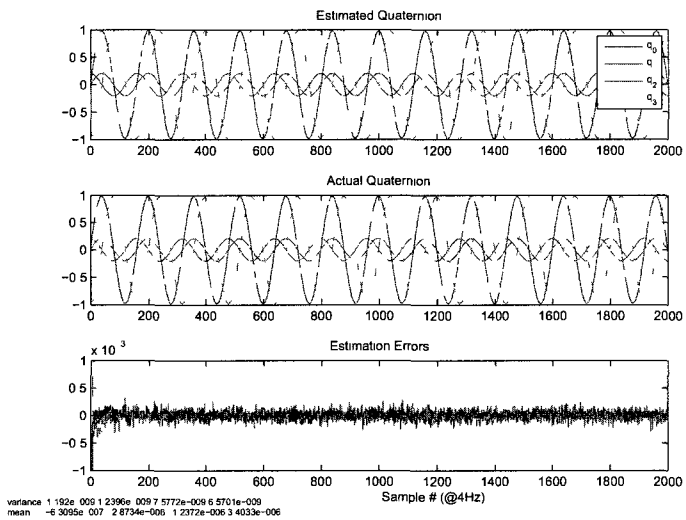


Figure 4.19: SMO $\hat{q}$ subjected to bias errors and measurement noise (known inertia)

The transient response until error convergence is longer than the EKF or H_∞ and the noise attenuations seems comparable.

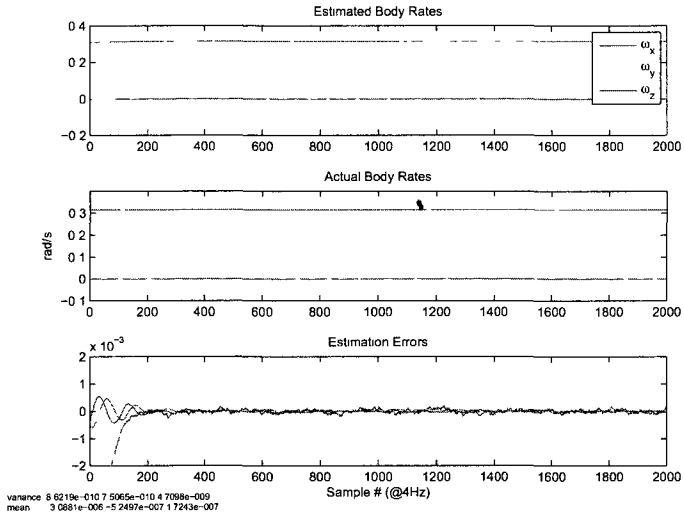


Figure 4.20: SMO $\hat{\omega}$ subjected to bias errors and measurement noise (known inertia)

Since there is no sensor directly measuring the body rates its time until convergence is significantly longer and could be of concern if required for s/c control. This filter is designed primarily for accelerometer lumped bias estimation, however, and the transient is acceptable.

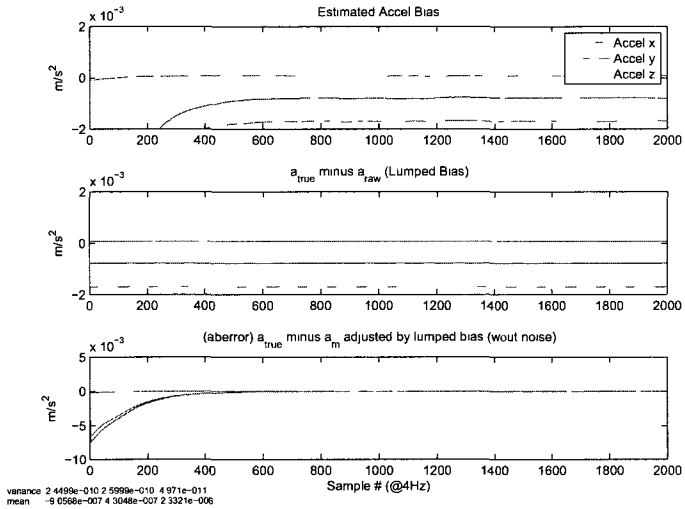


Figure 4.21: SMO $\hat{a}_{LB}$ subjected to bias errors and measurement noise (known inertia)

Due to the slow error convergence of the angular body rates, the accelerometer lumped bias exibits a similar transient portion of its response. The steady state lumped bias is of interest, however, and exhibits error variances and means similar to those of the optimal filters.

As is compiled for the EKF and H_∞ Filter, means or the SMO error variances and RMS of the mean errors are presented gathered from 100 random simulations.

Table 4.5: Means of the error variances of 100 trials (known inertia) - SMO

	q error variance	ω error variance $(rad/s)^2$	a_b error variance $(m/s^2)^2$
a_b δr and Δ	$\begin{bmatrix} 0\ 8628 \\ 0\ 0029 \\ 0\ 0030 \\ 0\ 8405 \end{bmatrix} \times 10^{-11}$	$\begin{bmatrix} 0\ 1390 \\ 0\ 0758 \\ 0\ 6686 \end{bmatrix} \times 10^{-9}$	$\begin{bmatrix} 0\ 3191 \\ 0\ 3270 \\ 0\ 0138 \end{bmatrix} \times 10^{-9}$
a_b δr and Δ with noise	$\begin{bmatrix} 0\ 1231 \\ 0\ 1342 \\ 0\ 7466 \\ 0\ 6696 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0\ 0834 \\ 0\ 0839 \\ 0\ 4619 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0\ 7772 \\ 0\ 7777 \\ 0\ 0978 \end{bmatrix} \times 10^{-9}$
a_b δr and Δ with 10×noise	$\begin{bmatrix} 0\ 1219 \\ 0\ 1330 \\ 0\ 7431 \\ 0\ 6636 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0\ 3190 \\ 0\ 3209 \\ 0\ 2632 \end{bmatrix} \times 10^{-7}$	$\begin{bmatrix} 0\ 1083 \\ 0\ 1025 \\ 0\ 0850 \end{bmatrix} \times 10^{-7}$

SMO error variances are on par with the EKF and H_∞ Filters for Cases 1 and 2, however, degrade significantly for Case 3.

Table 4.6: RMS of the mean errors for 100 trials (known inertia) - SMO

	mean error	ω mean error (rad/s)	a_b mean error (m/s^2)
a_b δr and Δ	$\begin{bmatrix} 0.1286 \\ 0.0043 \\ 0.0044 \\ 0.0743 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.0015 \\ 0.0024 \\ 0.1545 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.7639 \\ 0.7541 \\ 0.0109 \end{bmatrix} \times 10^{-5}$
a_b δr and Δ with noise	$\begin{bmatrix} 0.1638 \\ 0.1535 \\ 0.3724 \\ 0.3989 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.0220 \\ 0.0228 \\ 0.1204 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.6520 \\ 0.7106 \\ 0\ 4466 \end{bmatrix} \times 10^{-5}$
a_b δr and Δ with 10×noise	$\begin{bmatrix} 0.1773 \\ 0\ 1721 \\ 0.3762 \\ 0.3463 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.0578 \\ 0.0517 \\ 0.9480 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.6171 \\ 0.5976 \\ 0.3539 \end{bmatrix} \times 10^{-4}$

Although not as small as the EKF's and H_∞ Filter's, accelerometer lumped bias

mean errors are within the mission tolerances for Cases 1 and 2. Similarly to the optimal filters for Case 3 estimation errors are no longer within bounds.

2.2.4 Summary of Results for Known Inertia Tensor

The statistical results previously shown for Case 2 (expected conditions) are compiled together for direct comparison of the three filters in the tables below.

Table 4.7: Summary of the means of the error variances for filters with known inertia

	q error variance	ω error variance $(rad/s)^2$	a_b error variance $(m/s^2)^2$
EKF	$\begin{bmatrix} 0.0848 \\ 0.0845 \\ 0.2259 \\ 0.2207 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.1703 \\ 0.1229 \\ 0.1260 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.5802 \\ 0.5854 \\ 0.0207 \end{bmatrix} \times 10^{-9}$
H_∞	$\begin{bmatrix} 0.0848 \\ 0.0848 \\ 0.2199 \\ 0.2186 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.1376 \\ 0.0994 \\ 0.1107 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.3531 \\ 0.3543 \\ 0.0208 \end{bmatrix} \times 10^{-9}$
SMO	$\begin{bmatrix} 0.1231 \\ 0.1342 \\ 0.7466 \\ 0.6696 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.0834 \\ 0.0839 \\ 0.4619 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.7772 \\ 0.7777 \\ 0.0978 \end{bmatrix} \times 10^{-9}$

Although different filter/observer structures are used, the resulting error variance are all within 1 order of magnitude and substantially lower than the sensor noise variances input to the filters/observers.

Table 4.8: Summary of the RMS of the mean errors for filters with known inertia

	q mean error	ω mean error (rad/s)	a_b mean error (m/s^2)
EKF	$\begin{bmatrix} 0.1612 \\ 0.1533 \\ 0.3544 \\ 0.3588 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.4013 \\ 0.2380 \\ 0.4609 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.4360 \\ 0.4010 \\ 0.3863 \end{bmatrix} \times 10^{-5}$
H_∞	$\begin{bmatrix} 0.1584 \\ 0.1582 \\ 0.3528 \\ 0.3702 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.3821 \\ 0.2692 \\ 0.4510 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.3673 \\ 0.3830 \\ 0.3287 \end{bmatrix} \times 10^{-5}$
SMO	$\begin{bmatrix} 0.1638 \\ 0.1535 \\ 0.3724 \\ 0.3989 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.0220 \\ 0.0228 \\ 0.1204 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.6520 \\ 0.7106 \\ 0.4466 \end{bmatrix} \times 10^{-5}$

The numbers are primarily self explanatory, with a mission defined tolerance of 10^{-5} m/s^2 on a_{LB} every filter will provide adequate performance. The decrease in error mean for the H_∞ Filter over the EKF is clearly seen. While the SMO error mean is the highest, it offers significant computational savings which will be explained in the Conclusions.

2.3 Corrupt Inertia Tensor Results

The primary reason for investigating additional filtering techniques is to design a filter robust to a corrupt/unknown inertia tensor. Each dynamic filter/observer is tested with expected conditions (same as Case 2 previously described), however, the

true inertia tensor is reduced from

$$\boldsymbol{J} = \begin{bmatrix} 783.35 & -12.28 & -4.84 \\ -12.28 & 803.79 & -7.67 \\ -4.84 & -7.67 & 1332.99 \end{bmatrix} kg \cdot m^2 \tag{4.27}$$

to the diagonal nominal matrix

$$\hat{\boldsymbol{J}} = \begin{bmatrix} 800 & 0 & 0 \\ 0 & 800 & 0 \\ 0 & 0 & 1300 \end{bmatrix} kg \cdot m^2 \tag{4.28}$$

The same filters used with known inertia tensors are used here.

2.3.1 EKF Results for Corrupt Inertia Tensor

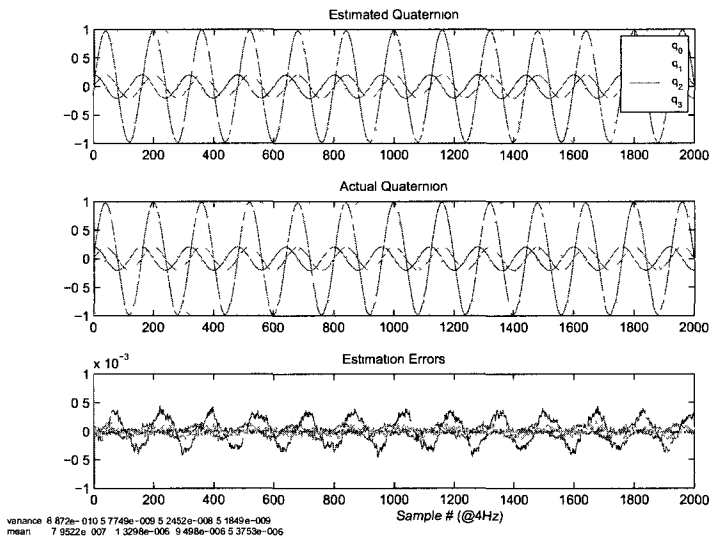


Figure 4.22: EKF $\hat{q}$ subjected to bias errors and measurement noise (unknown inertia)

Although the system dynamics describing quaternion propagation are not directly dependent on the inertia tensor, the errors developed in the angular rate estimation seen below cause significant errors in the quaternion estimation.

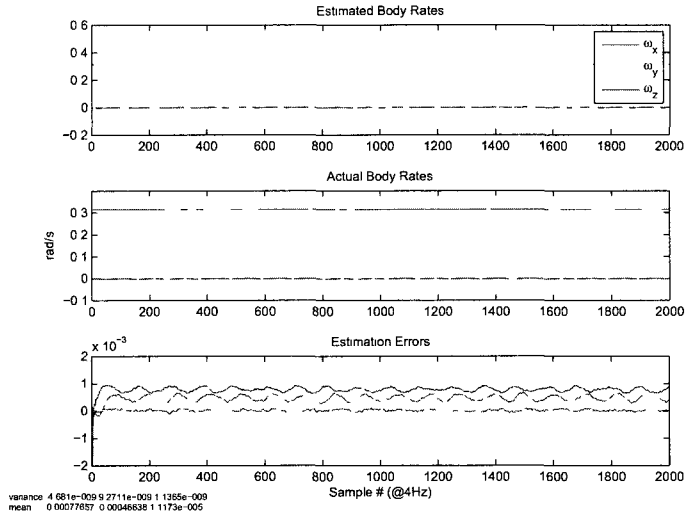


Figure 4.23: EKF $\hat{\omega}$ subjected to bias errors and measurement noise (unknown inertia)

Errors in the inertia tensor will cause a near constant error in the angular rate estimation. The offset is primarily proportional to the $J(1,3)$ and $J(2,3)$ terms which are multiplied by the dominant angular rate, ω_z, at each time step.

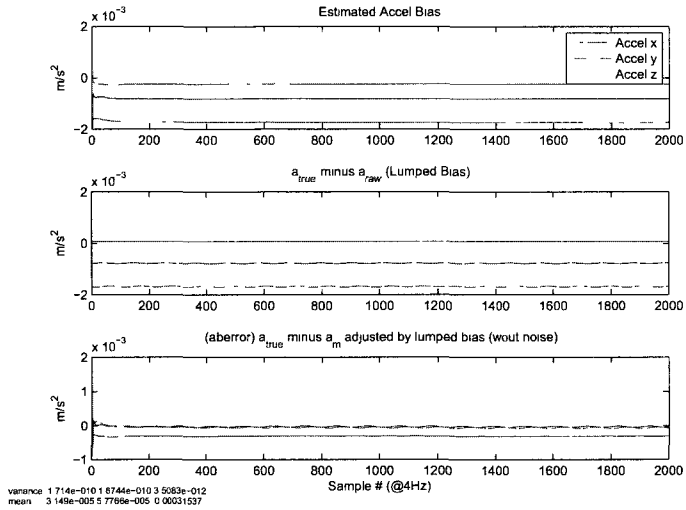


Figure 4.24: EKF $\hat{a}_{LB}$ subjected to bias errors and measurement noise (unknown inertia)

Since the error in $\boldsymbol{\omega}$ develops during the propagation stage, the measurement equation $\boldsymbol{g}(\hat{\boldsymbol{x}})$, used to formulate the innovations, will always contain errors and the resulting estimate lumped bias will converge to an incorrect value.

2.3.2 H_∞ Filter Results for Corrupt Inertia Tensor

The H_∞ Filter results are nearly identical, however, shown for the readers visual comparison.

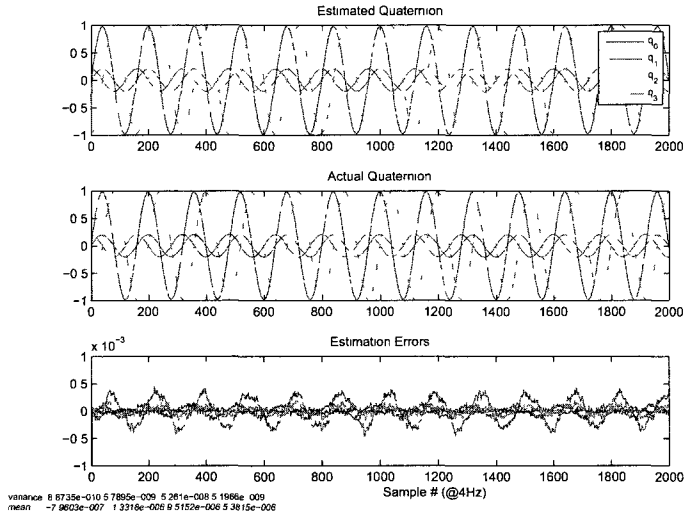


Figure 4.25: H_∞ $\hat{q}$ subjected to bias errors and measurement noise (unknown inertia)

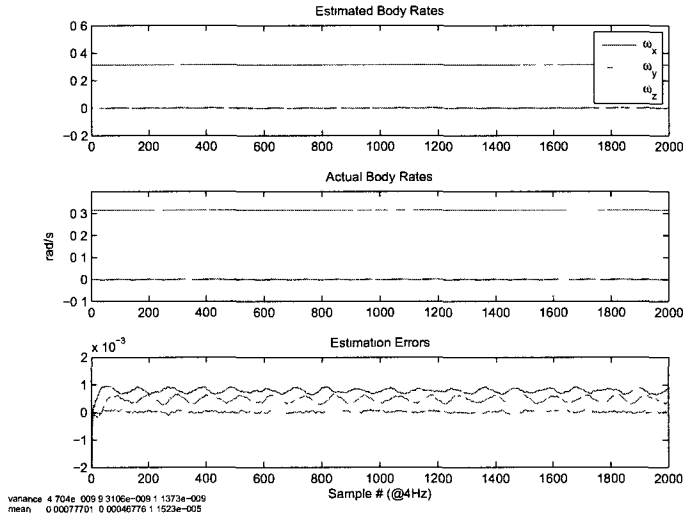


Figure 4.26: H_∞ $\hat{\omega}$ subjected to bias errors and measurement noise (unknown inertia)

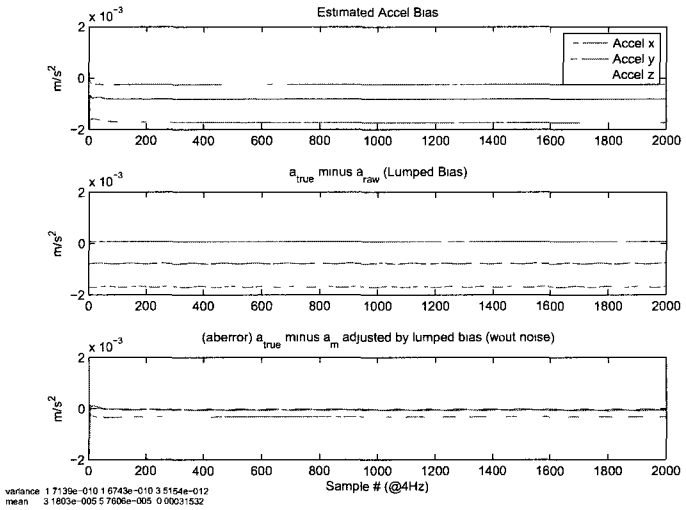


Figure 4.27: H_∞ $\hat{a}_{LB}$ subjected to bias errors and measurement noise (unknown inertia)

2.4 SMO Results for Corrupt Inertia Tensor

The SMO robustness can be seen in its corrupt inertia tensor results, the quaternion and angular rate mean estimate errors are smaller than those of the EKF or II_∞ Filter.

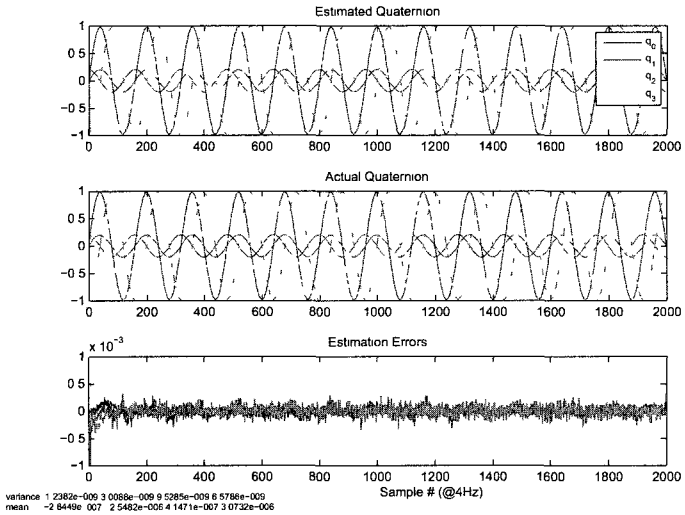


Figure 4.28: SMO $\hat{q}$ subjected to bias errors and measurement noise (unknown inertia)

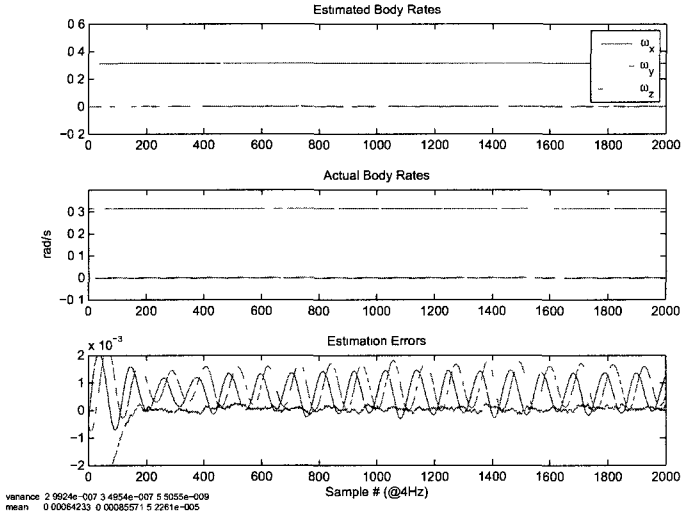


Figure 4.29: SMO $\hat{\omega}$ subjected to bias errors and measurement noise (unknown inertia)

The figure below shows the SMO $\mathbf{a_{LB}}$ error converging to zero. However, closer

inspection of the statistical data shows error means no better than the EKF and H_∞ Filter.

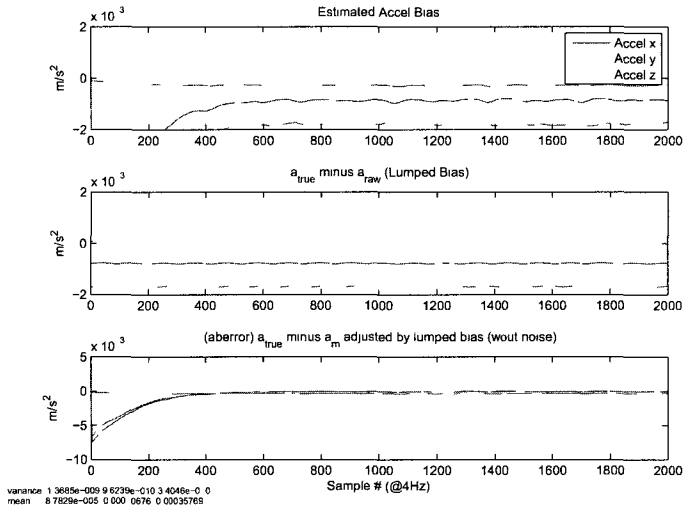


Figure 4.30: SMO $\hat{a}_{LB}$ subjected to bias errors and measurement noise (unknown inertia)

2.4.1 Summary of Filter Results for Corrupt Inertia Tensor

Table 4.9: Mean of the error variances of 100 trials (uncertain inertia)

	q error variance	ω error variance $(rad/s)^2$	a_b error variance $(m/s^2)^2$
EKF	$\begin{bmatrix} 0.0010 \\ 0.0517 \\ 0.4959 \\ 0.0407 \end{bmatrix} \times 10^{-7}$	$\begin{bmatrix} 0.3774 \\ 0.7305 \\ 0.0368 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.4857 \\ 0.4785 \\ 0.0340 \end{bmatrix} \times 10^{-9}$
H_∞	$\begin{bmatrix} 0.0010 \\ 0.0519 \\ 0.4977 \\ 0.0409 \end{bmatrix} \times 10^{-7}$	$\begin{bmatrix} 0.4206 \\ 0.8095 \\ 0.0423 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.5738 \\ 0.5641 \\ 0.0416 \end{bmatrix} \times 10^{-9}$
SMO	$\begin{bmatrix} 0.0089 \\ 0.1753 \\ 0.1767 \\ 0.0078 \end{bmatrix} \times 10^{-8}$	$\begin{bmatrix} 0.2102 \\ 0.2532 \\ 0.0343 \end{bmatrix} \times 10^{-6}$	$\begin{bmatrix} 0.2206 \\ 0.1702 \\ 0.0263 \end{bmatrix} \times 10^{-8}$

Table 4.10: RMS of the mean errors for 100 trials (uncertain inertia)

	q mean error	ω mean error (rad/s)	a_b mean error (m/s^2)
EKF	$\begin{bmatrix} 0.0011 \\ 0.0132 \\ 0.1021 \\ 0.0278 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.7737 \\ 0.4678 \\ 0.0053 \end{bmatrix} \times 10^{-3}$	$\begin{bmatrix} 0.0304 \\ 0.0618 \\ 0.3148 \end{bmatrix} \times 10^{-3}$
H_∞	$\begin{bmatrix} 0.0012 \\ 0.0134 \\ 0.1016 \\ 0.0278 \end{bmatrix} \times 10^{-4}$	$\begin{bmatrix} 0.7742 \\ 0.4689 \\ 0.0056 \end{bmatrix} \times 10^{-3}$	$\begin{bmatrix} 0.0308 \\ 0.0616 \\ 0.3148 \end{bmatrix} \times 10^{-3}$
SMO	$\begin{bmatrix} 0.0713 \\ 0.0505 \\ 0.1826 \\ 0.0497 \end{bmatrix} \times 10^{-5}$	$\begin{bmatrix} 0.6771 \\ 0.8364 \\ 0.0168 \end{bmatrix} \times 10^{-3}$	$\begin{bmatrix} 0.1280 \\ 0.1281 \\ 0.3557 \end{bmatrix} \times 10^{-3}$

Table 4.10 shows the RMS of the mean errors over 100 trials. From the third column it is apparent that the inertia tensor error that the filters/observers are subjected to causes errors in the a_{LB} estimation that are unacceptable for the MMS mission.

2.5 Batch Filter Results

In this section of results, the first two general test scenarios (Case 1 and 2) for filters/estimators are observed for that of the batch filter (BF), under the condition of a known inertia tensor:

For the condition of a corrupt inertia tensor as is described for the dynamic filters in Case 2 is also observed for the batch filter.

2.5.1 Known Inertia Tensor Results

Figures 4.31 through 4 33 present the batch filter results for Case 1

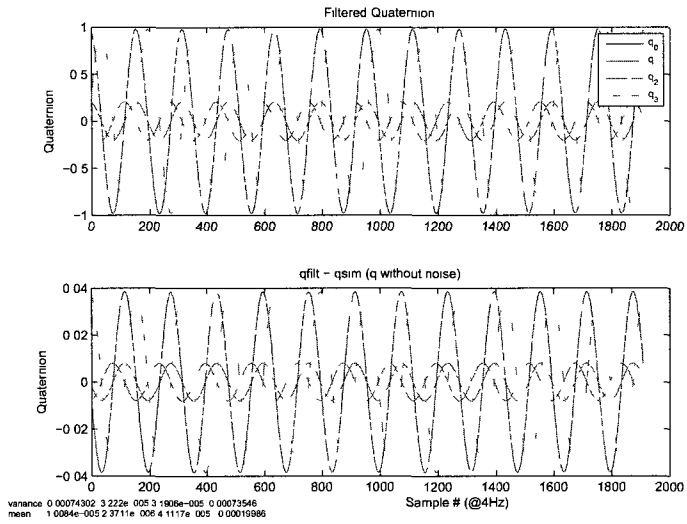


Figure 4.31: BF quaternion estimation with 3-corruptions and noise (known inertia)

Significant errors can be seen on the filtered quaternion, however, this does not affect the lumped bias estimation. Since the batch process does not estimate in real time, the s/c requires a dynamic filter on board to estimate states in real time. This dynamic filter could use a lumped bias parameter estimate provided by the batch process and not estimate one on board.

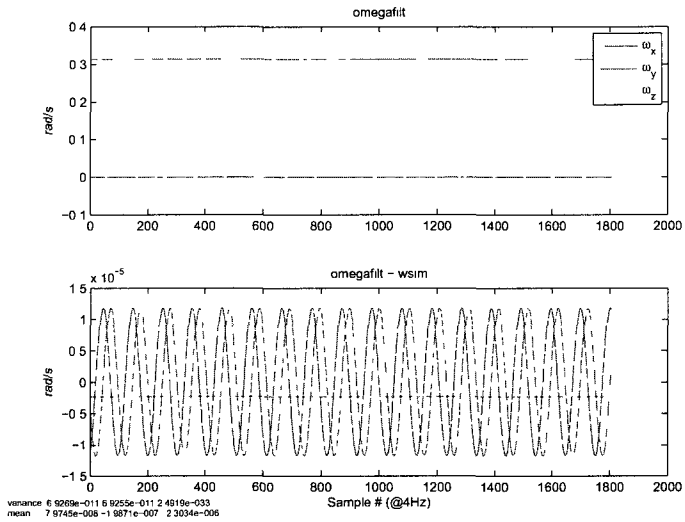


Figure 4.32: BF angular rate estimation with 3-corruptions and noise (known inertia)

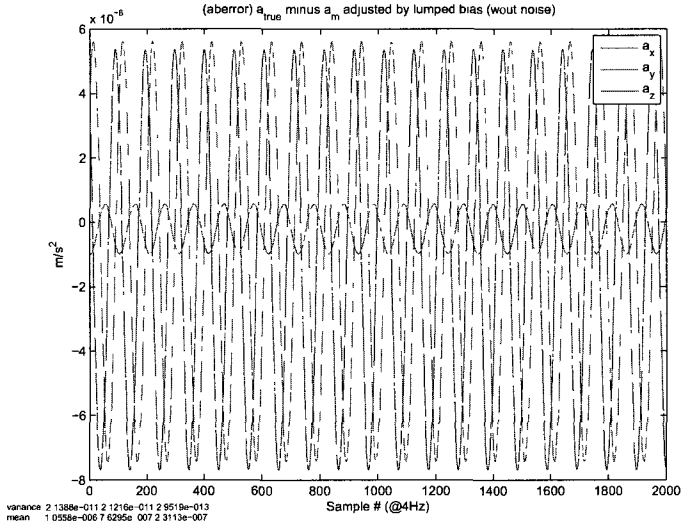


Figure 4.33: BF lumped accelerometer bias estimation with 3-corruptions and noise (known inertia)

Table 4.11: Batch lumped bias estimation for all corruptions & no noise

	$\hat{\mathbf{a}}_{\mathbf{LB}}$ error
True a_{LB}	$[0\ 0008 \quad 0.0017 \quad -0.0001]^T\ m/s^2$
$\hat{a}_{LB}$ Error	$[-0.1422 \quad -0.0759 \quad -0.0028]^T \times 10^{-5}\ m/s^2$
$\hat{a}_{LB}$ % Error	$[-0.1829 \quad -0\ 0449 \quad 0\ 0348]^T$ %

Figures 4.31 through 4.33 present the batch filter results for Case 2

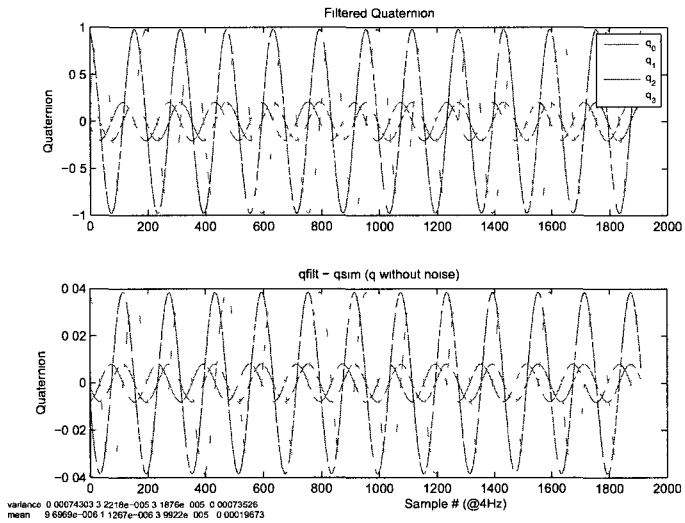


Figure 4.34: BF quaternion estimation with 3-corruptions and noise (known inertia)

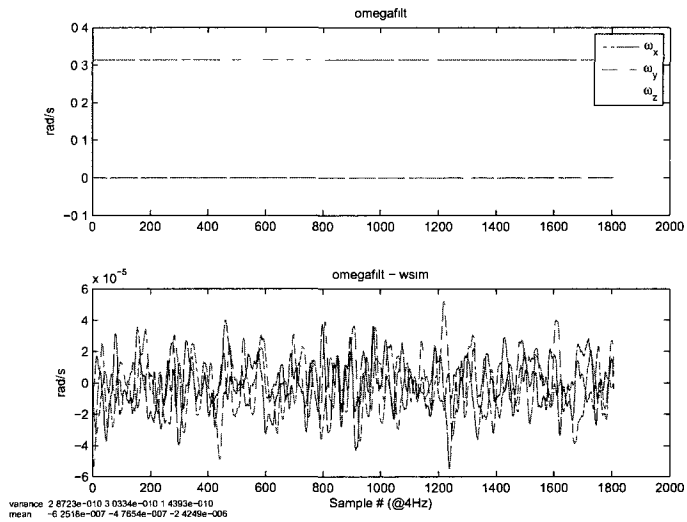


Figure 4.35: BF angular rate estimation with 3-corruptions and noise (known inertia)

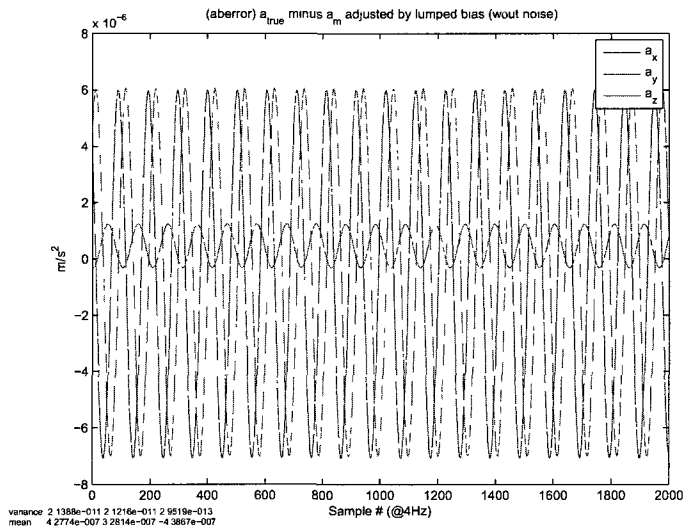


Figure 4.36: BF lumped accelerometer bias estimation with 3-corruptions and noise (known inertia)

Table 4.12: Batch lumped bias estimation for all corruptions with noise

	$\hat{\mathbf{a}}_{\mathbf{LB}}$ error
True a_{LB}	$[0\ 0008 \quad 0.0017 \quad -0.0001]^T\ m/s^2$
$\hat{a}_{LB}$ Error	$[-0.0831 \quad -0.1451 \quad -0.0164]^T \times 10^{-5}\ m/s^2$
$\hat{a}_{LB}$ % Error	$[-0\ 1066 \quad -0.0860 \quad 0.1961]^T$ %

2.5.2 Corrupt Inertia Tensor Results

Figures 4.31 through 4.33 present the batch filter results for scenario 2, however, with a corrupt inertia tensor used within the filter.

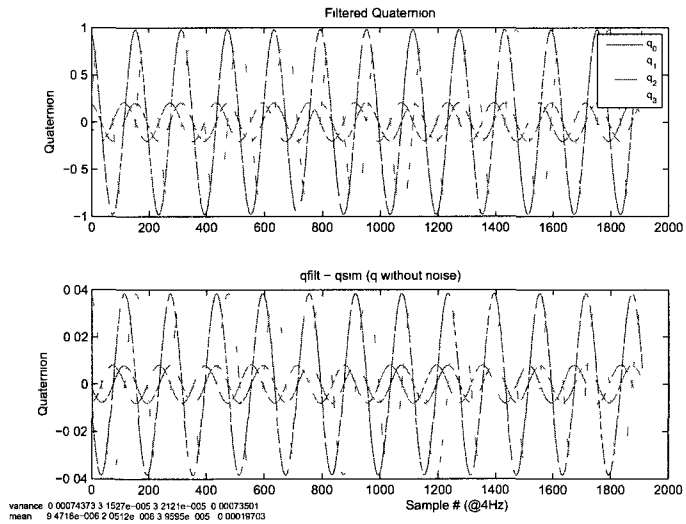


Figure 4.37: BF quaternion estimation with 3-corruptions and noise (unknown inertia)

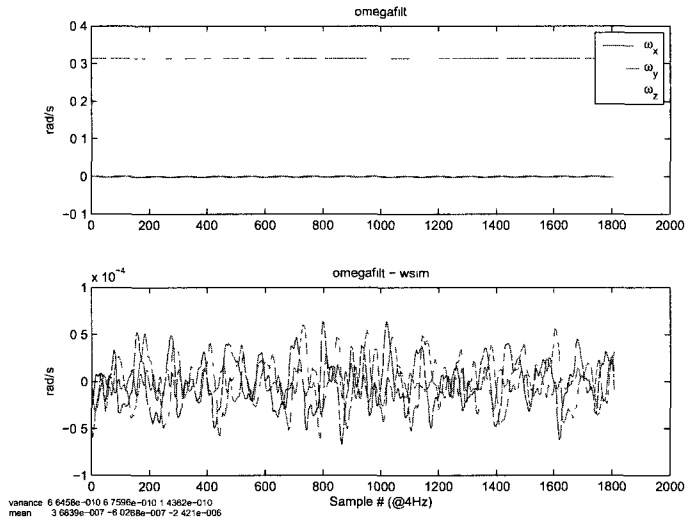


Figure 4.38: BF angular rate estimation with 3-corruptions and noise (unknown inertia)

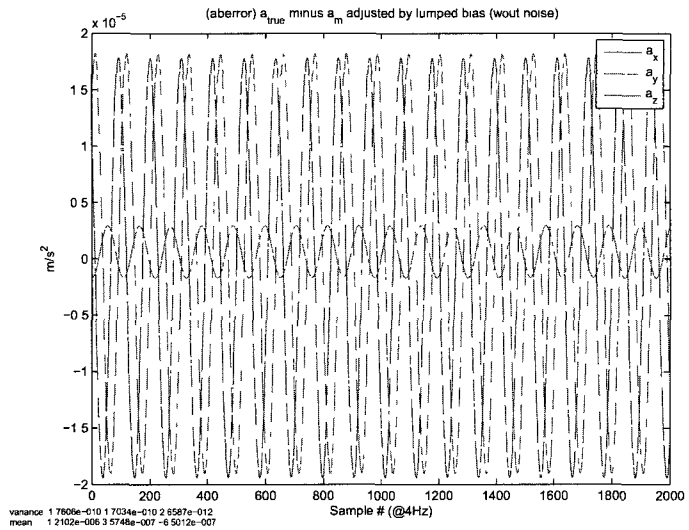


Figure 4.39: BF lumped accelerometer bias estimation with 3-corruptions and noise (unknown inertia)

Table 4.13: Batch lumped bias estimation for all corruptions with noise and inertia uncertainty

	$\hat{\mathbf{a}}_{\mathbf{LB}}$ error
True a_{LB}	$[0.0008 \;\; 0.0017 \;\; -0.0001]^T \;\; m/s^2$
$\hat{a}_{LB}$ Error	$[-0.1042 \;\; -0.0449 \;\; 0.0963]^T \times 10^{-5} \;\; m/s^2$
$\hat{a}_{LB}$ % Error	$[-0.1345 \;\; -0.0266 \;\; -1.2205]^T \;\; \%$

2.5.3 Batch Filter Tabulated Results

Analysis of the batch filter shows that there is no dependence on the inertia tensor. Varying errors are only due to random statistical differences between simulation cases.

Table 4.14: Batch lumped bias estimation summary

	$\hat{\mathbf{a}}_{\mathbf{LB}}$ error
True a_{LB}	$[0.0008 \;\; 0.0017 \;\; -0.0001]^T \;\; m/s^2$
$\hat{a}_{LB}$ Error (a_b, r_c, δ)	$[-0.1422 \;\; -0.0759 \;\; -0.0028]^T \times 10^{-5} \;\; m/s^2$
$\hat{a}_{LB}$ Error (a_b, r_c, δ, noise)	$[-0.0831 \;\; -0.1451 \;\; -0.0164]^T \times 10^{-5} \;\; m/s^2$
$\hat{a}_{LB}$ Error (a_b, r_c, δ, noise, J_{error})	$[-0.1042 \;\; -0.0449 \;\; 0.0963]^T \times 10^{-5} \;\; m/s^2$

Chapter V

EXPERIMENTAL VERIFICATION

1 NASA MMS TableSat Generation I

The proposed accelerometer bias estimation technique is implemented in an experimental testbed. The NASA MMS TableSat Generation I is a limited 3-DOF model of the MMS spacecraft, primarily designed for the analysis of the MMS s/c attitude dynamics and kinematics and the dynamic effects of the SDP and ADP flexible booms.

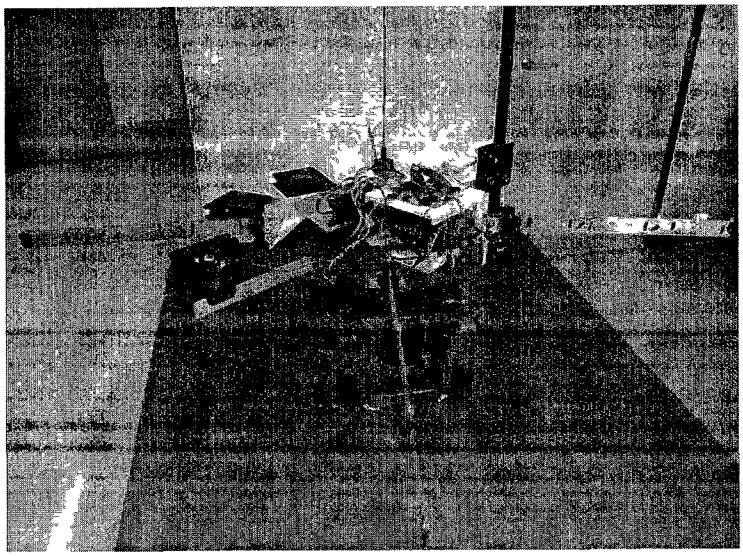


Figure 5.1: TableSat I

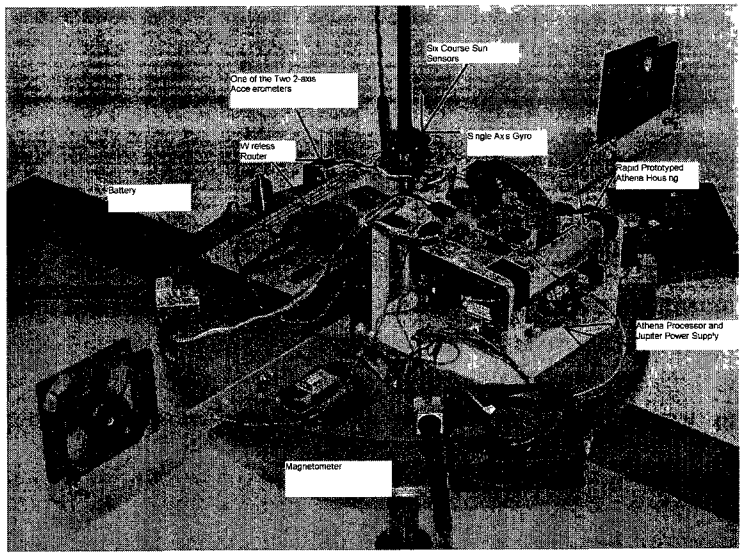


Figure 5.2: TableSat I components

For the purpose of this experimental analysis, a secondary TableSat I unit is used. Here, all electrical components, save one fan, have been removed from the TableSat

because of an unidentifiable electromagnetic field being emitted and corrupting the IMU magnetometer measurements. A new electronics package is designed for data collection, while rotation is accomplished by attaching a battery directly to a fan thruster. The new package includes a 6-DOF inertial measurement unit (IMU), an additional high-precision MEMS accelerometer, a microSD data-logging card and a microcontroller.

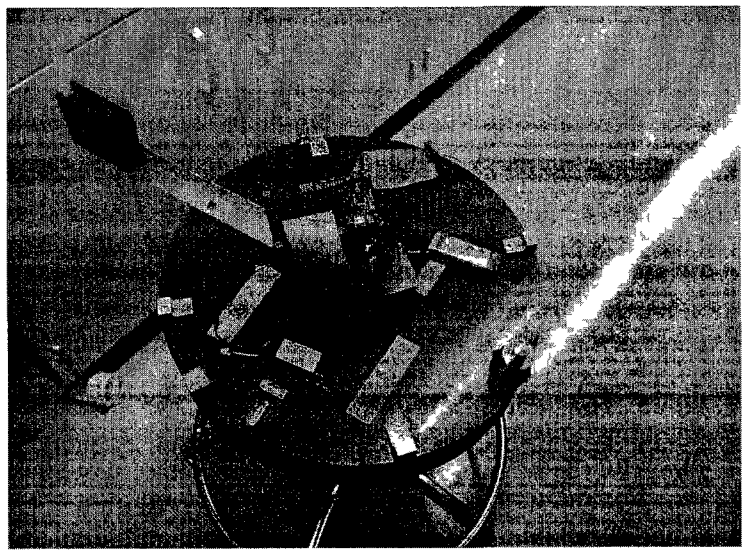


Figure 5.3: TableSat for experimental accelerometer calibration - View 1

1.1 Experimental Hardware

As previously mentioned, all existing hardware on the TableSat is removed and an entirely new electronics package is assembled.

Parallax Propeller Platform - 80MHz, 8-core 32-bit P8X32A microprocessor, with onboard USB communication for programming, 32 digital I/O pins, and integrated microSD card accessability through SPI communication

Sparkfun 9-DOF Razor IMU - 9 total combined measurements from a gyro, accelerometer and magnetometer utilized by an 8-bit microprocessor to provide attitude "measurements" as well as raw sensor data over UART serial communication

BMA180 Accelerometer Breakout Board - High sensitivity, 14-bit resolution $\pm$2g accelerometer with low-noise and SPI communication

2$\times$Lithium Ion batteries - 11.1 V for fan thrust and 7.4 V for powering the microcontroller, both 900mAh

80mm Computer Fan - Used for rotational thrust

1.2 Experimental Software

The propeller platform includes a substantial library of objects contributed by the company's engineers and the general public. Some existing open source objects are used to simplify the process of writing the required software code:

Software Serial - emulated UART serial port utilizing a single core, used for communication with the IMU

Software SPI - emulated Serial Peripheral Interface using a single core for microSD access and an additional core for BMA180 communication

Clock - emulated real-time clock for measurement time-stamping, uses a single core

FAT Library - collection of functions to access the microSD's FAT file structure, specifically creating text files and appending to them

All items are obtained through the Propeller Object Exchange [25].

The main program uses its own core to collect all measurements, obtain a time-stamp and store all information to the microSD card for analysis in a separate numerical simulation software package. To decrease the required computational effort all values are left in their raw format and converted afterwards to appropriate numerical values. The IMU firmware is customized to output all raw sensor data as well as the estimated Euler angles (a 1-2-3 rotation sequence) and Direct Cosine Matrix. All values are transmitted at 4800 baud as ASCII characters. The real-time clock output is a time-stamp with a resolution of 0.01 seconds in ASCII format. Communication with the accelerometer is performed with the SPI protocol, which uses a clock line to synchronize the two devices rather than enforcing a specified BAUD rate used by UART. Communication over SPI is performed one byte at a time and must be initialized by the host sending a command byte and address byte. The BMA180 responds with 6 bytes containing acceleration measurements in 3 axes and a measurement of the temperature for temperature compensation.

Measuring attitude typically requires an expensive and sophisticated vision system. Since such a system is not available for this research, an Inertial Measurement Unit (IMU) is used to estimate the attitude using a magnetometer, gyro and accelerometer, each providing measurements along all three orthogonal axes. The resulting attitude measurement is relatively accurate, albeit far from perfect, and is sufficient for preliminary experimental results. The accelerometer provides an inertial gravity measurement allowing correction for pitch and roll. The magnetometer allows corrections for drift from integrated gyro output about the body z-axis.

Integrating the gyro provides redundant information and allows for the combining and filtering of measurement output. This filtering is critical, as the system's MEMS sensors have excessive noise. The IMU utilizes open source firmware, which incorporates an EKF for attitude dead reckoning, using gyro and accelerometer measurements and an additional magnetometer update of rotation about the body z-axis.

2 Experimental Results

To further validate the performance of the Batch Process Filter in this study, experimental testing is performed. Unfortunately, it is not possible to measure the three accelerometer calibration parameters with enough accuracy to predict what the accelerometer lumped bias is. Instead, a qualitative analysis approach is attempted. Here, a trend is expected from the estimated lumped bias estimator, provided a specific input trend. Analytical calculations are performed to identify the trend correlations. In addition to inspecting the performance of the lumped bias estimation, an on-board gyro is used to evaluate the angular body rate estimates.

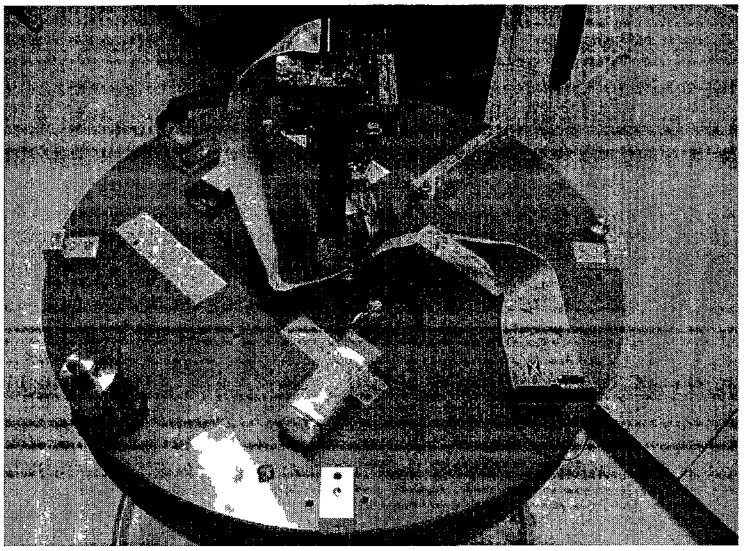


Figure 5.4. TableSat for experimental accelerometer calibration - View 2

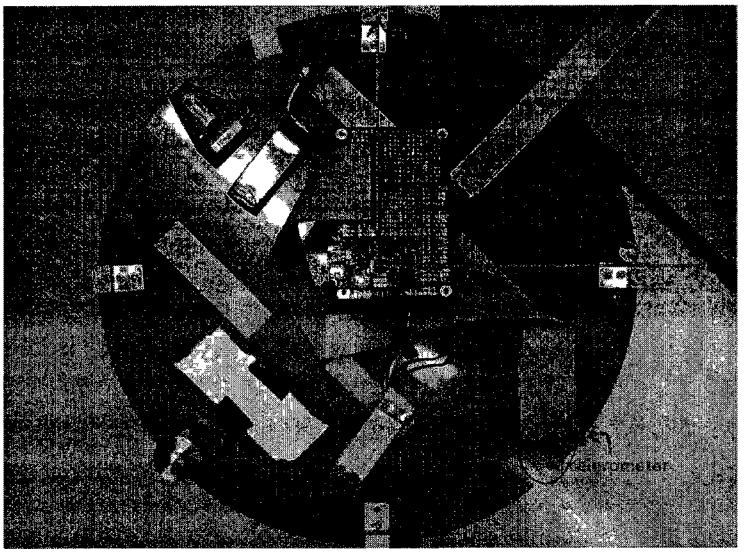


Figure 5.5: TableSat for experimental accelerometer calibration - View 3

Figure 5.5 shows an aerial view of the experimental testbed with standard x-y coordinates overlayed. The accelerometer can be seen on the bottom right quadrant

and the location relative to the center of mass is:

$$r = \begin{bmatrix} -11.69 \\ -14.35 \\ -2.13 \end{bmatrix} cm \tag{5.1}$$

The batch filter does not require the inertia tensor for its calculations and, therefore, it is not calculated.

2.1 Accelerometer Pre-Calibration

Prior to the accelerometer data being used, components due to gravity must be taken into account. These terms do not remain constant between tests since the accelerometer is rotated. This pre-calibration is performed using the first few seconds of data from each test while the model is still stationary. From the gyro output, the point in time of the TableSat's initial spin becomes obviously apparent.

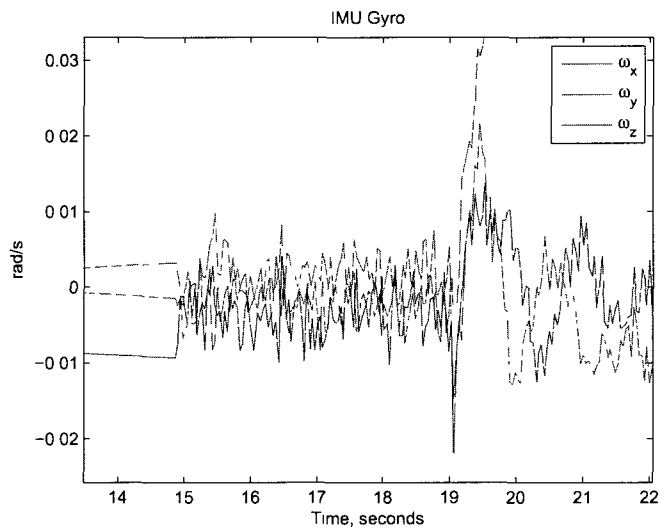


Figure 5.6: Settled model gyro output

To determine the gravity components, the acceleration prior to any rotational

motion/movement is used. Averaging the BMA180 accelerometer measurement data that correspond to this "pre-spin" time period serves to determine the pre-calibration bias which takes into account the sensor bias as well as gravitational acceleration. The resulting lumped bias that is to be estimated for these experiments is a result of any error in the accelerometer location between where it is measured to be and the orthogonal misalignment of the sensor, which is deliberately varied about the body y-axis.

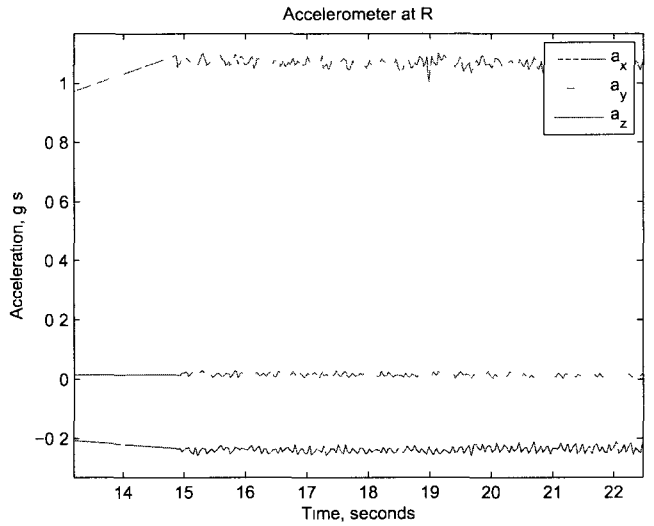


Figure 5.7: Accelerometer signal for initial calibration

2.2 Experimental Lumped Accelerometer Bias Estimation

The lumped accelerometer bias is estimated as the average error between the predicted acceleration and the measured acceleration for the last one-third of the data analyzed. Only this portion of the data is used to ensure a near constant angular rate which is required for accurate lumped bias estimation. Residual transients in the

angular rate are present and should show the robustness of the filter

For the experiment three test cases are analyzed

1 **"neutral" alignment:** $\delta_y \approx 0$

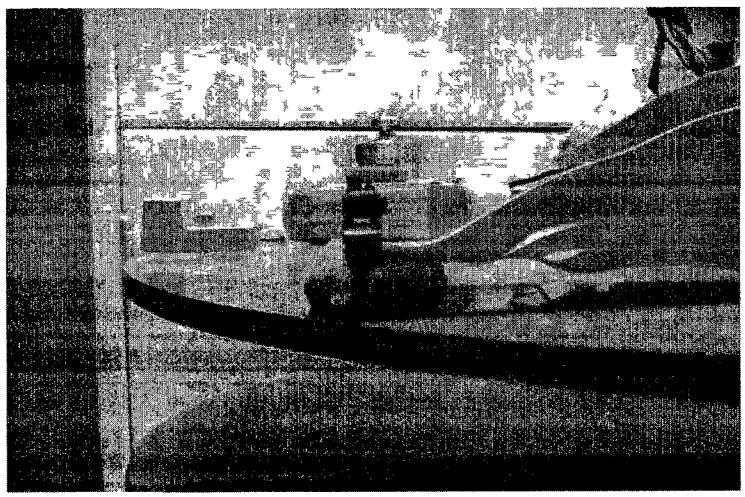


Figure 5 8 Experimental accelerometer neutral rotation about y-axis

2. "positive" alignment: $\delta_y > 0$

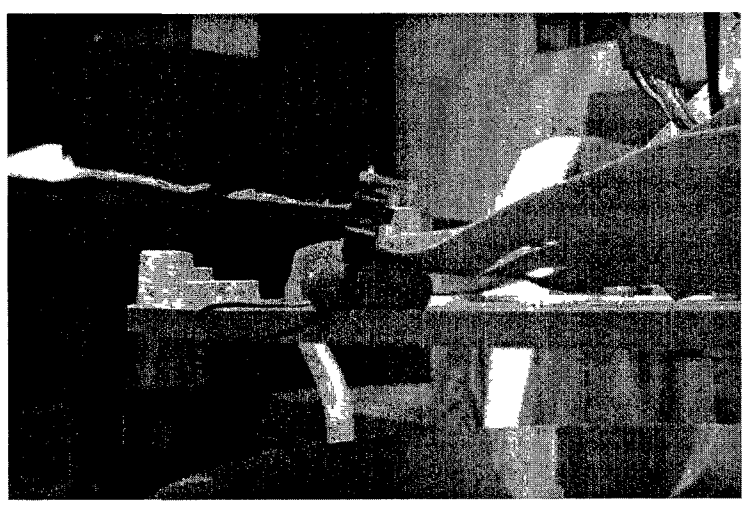


Figure 5 9 Experimental accelerometer positive rotation about y-axis

3. **"negative" alignment:** $\delta_y < 0$

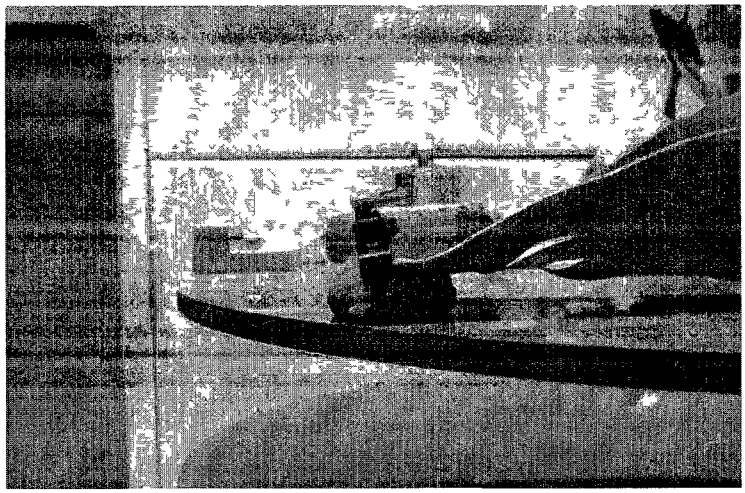


Figure 5 10 Experimental accelerometer negative rotation about y-axis

2.2.1 Neutral δ_y

Mounted at a neutral angle data is collected for processing through the batch filter

The batch process consists of two sequential spline fits The first is performed on the quaternion measurements, it provides a filtered quaternion measurement and an estimate of $\dot{\boldsymbol{q}}$.

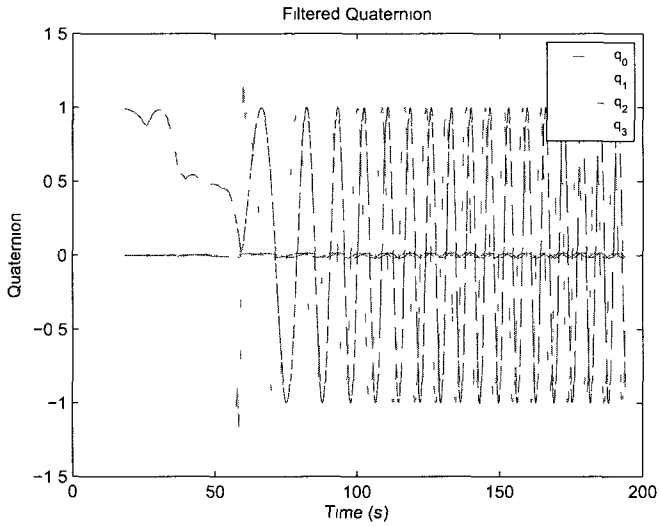


Figure 5.11: Filtered quaternion from first spline

The $\dot{\boldsymbol{q}}$ and thee filtered quaternion are used to estimate the angular body rates using Eq. (5.2).

$$\dot{\bar{\boldsymbol{q}}} = \frac{1}{2}\bar{\omega}_{b/o} \otimes \bar{\boldsymbol{q}}(t) \tag{5.2}$$

The gyro measurements and the angular body rate estimates are shown in Figure 5.12

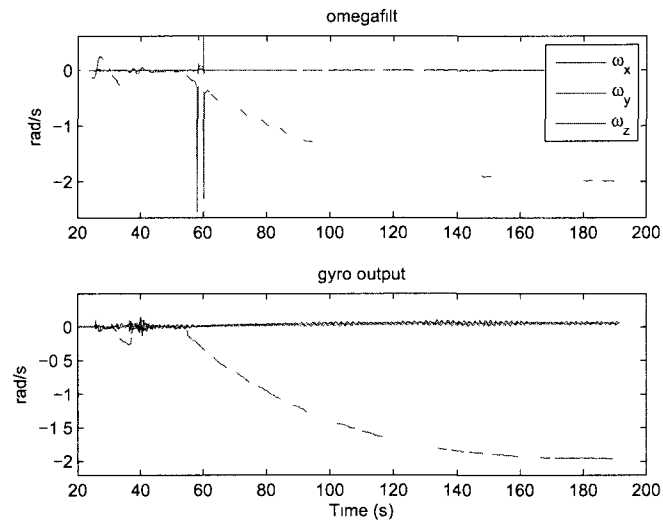


Figure 5 12: Angular rate estimate from first spline

A second spline fit is performed on the estimated body rates to filter then as well as estimate $\dot{\boldsymbol{\omega}}$. Using these two values the nominal acceleration can be estimated, it is shown along with the measured accelerations as,

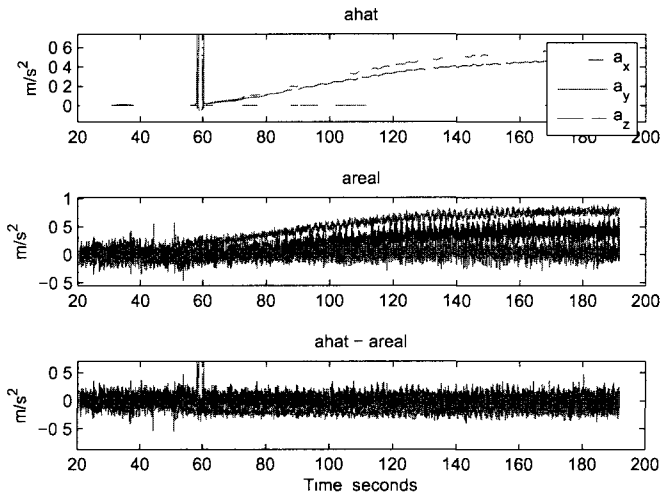


Figure 5 13 Predicted and measured acceleration with error

The resulting estimated lumped bias is:

$$\hat{a}_{LB} = \begin{bmatrix} 0.0240 \\ -0.2174 \\ -0.0198 \end{bmatrix} m/s^2 \tag{5.3}$$

2.2.2 Positive δ_y

The BMA180 accelerometer is rotated about the y-axis to an arbitrary positive angle relative to the neutral δ_y for the following results.

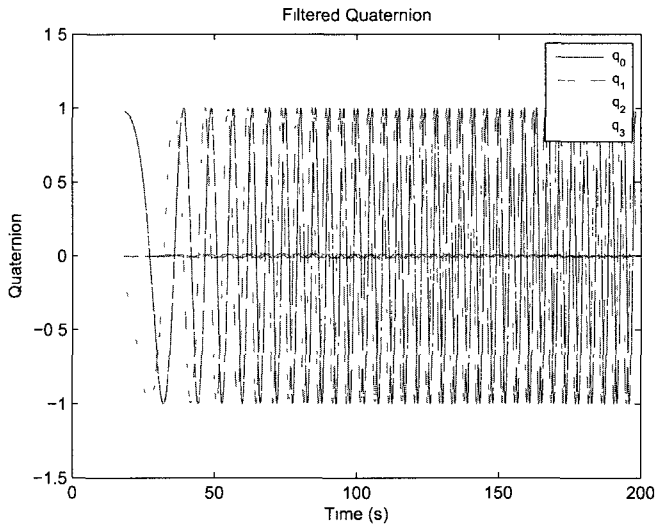


Figure 5.14: Filtered quaternion from first spline

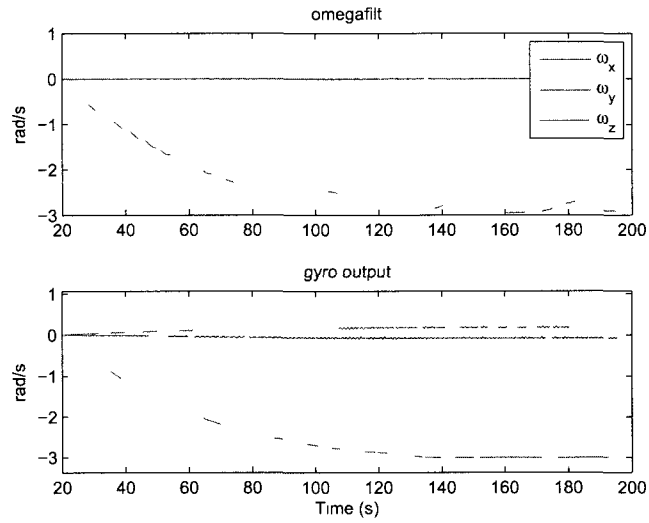


Figure 5 15: Angular rate estimate from first spline

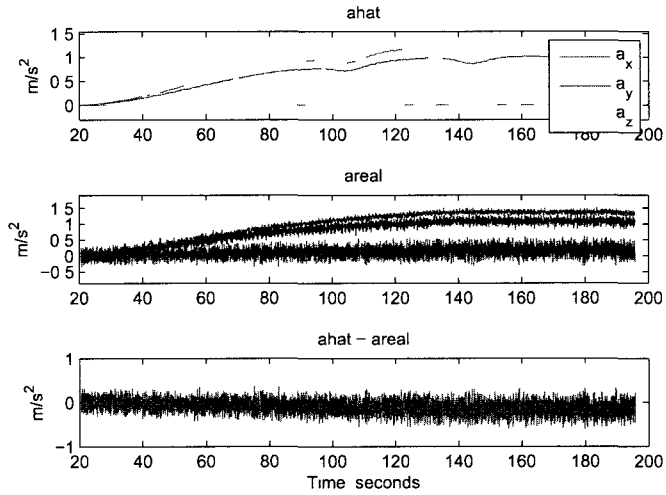


Figure 5.16: Predicted and measured acceleration with error

The predicted acceleration assumes that the BMA180 accelerometer axis are co-linear with the model's reference frame. This alignment is performed as closely as

possible for the x and z-axes, however, δ_x and δ_z will exist. δ_y is not known, but is positive for this test.

The estimated accelerometer bias for this first case of positive δ_y is:

$$\hat{a}_{LB} = \begin{bmatrix} -0.1008 \\ -0.1291 \\ -0.0953 \end{bmatrix} m/s^2 \tag{5.4}$$

2.2.3 Negative δ_y

For the third test δ_y is rotated negative from the neutral rotation.

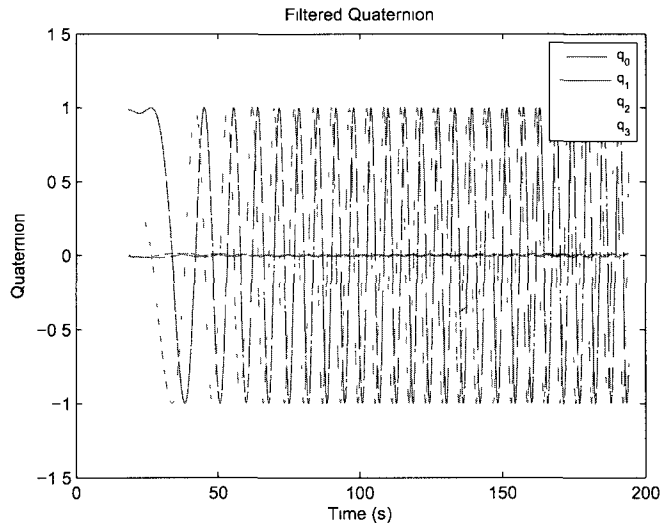


Figure 5.17: Filtered quaternion from first spline

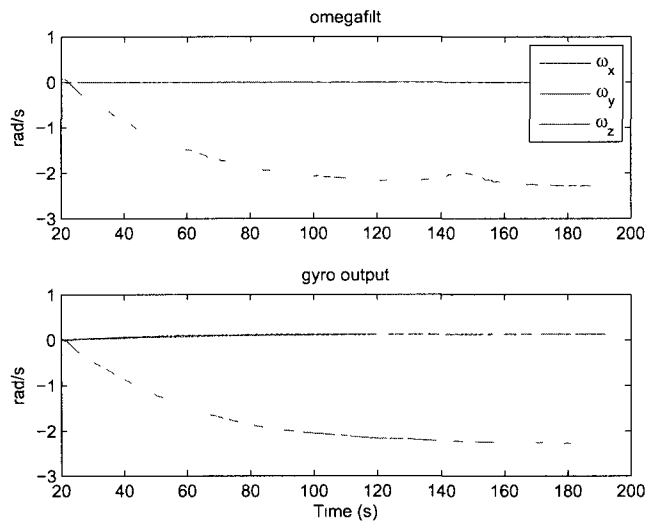


Figure 5.18: Angular rate estimate from first spline

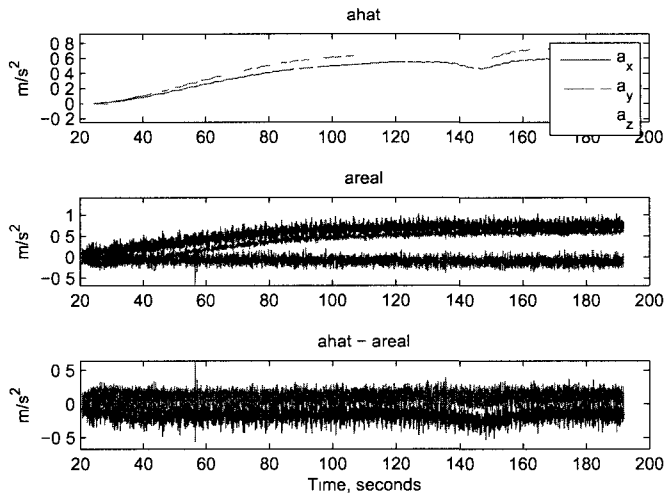


Figure 5.19: Predicted and measured acceleration with error

The resulting estimated lumped bias is:

$$\hat{a}_{LB} = \begin{bmatrix} -0.1657 \\ 0.1429 \\ 0.0832 \end{bmatrix} m/s^2 \tag{5.5}$$

2.3 Summary of Experimental Results

Changes in a_{LBx} and a_{LBy} can be attributed to rotations of the BMA180 about the z-axis while y-axis rotations were being made. Table 5.1 summarizes the three $\hat{a}_{LBz}$ and the expected trend of an increasing bias while δ_y decreases is seen. The fact that the increase is not constant is likely due to small changes in δ_x between tests. (Recall from the lumped bias justification that δ_x has an inverse relationship with a_{LBz}.)

Table 5.1: Lumped accelerometer bias trend for decreasing δ_y

	δ_{y+}	δ_{y0}	δ_{y-}
$\hat{a}_{LB}(m/s^2)$	-0.0953	-0.0198	0.0832

Although the true lumped bias value cannot be obtained for these experimental tests, the increasing trend of the lumped bias estimate with respect to the decreasing rotation of the accelerometer shows a qualitative confirmation of the batch filter method.

Chapter VI

CONCLUSIONS

The MMS spacecraft will be collecting high resolution (spatially and temporally) scientific data while flying in a precise tetrahedron formation. To maintain this formation the attitude observer and controller must both have accurate sensor measurements. Due to vibrations during launch, thermal shifting during sun-shade transitions while in orbit and fuel usage, three errors will develop in the accelerometer measurements: internal sensor bias, shift in location and orthogonal misalignment. The observer will have to operate without direct knowledge of the angular body rates since no gyro is on board.

This research intends to estimate a lumped bias parameter to correct for accelerometer bias, shift in accelerometer location and orthogonal misalignment. The lumped bias should be shown to be capable of such a correction and should be estimated using real-time or ground-based techniques. The lumped bias must be accurately estimated with or without perfect knowledge of the spacecraft inertia tensor and be able to correct the measured acceleration to within 1 μg of the nominal acceleration.

It is shown that an accurate determination of the accelerometer lumped bias can adequately and collectively correct for accelerometer bias, shift in the accelerometer's location with respect to the center of mass and an orthogonal misalignment. This

capability, however, assumes a near constant angular spin rate of the MMS spacecraft. Otherwise, the lumped bias may not accurately represent all the errors. In this research the lumped bias approximation is verified analytically, numerically and experimentally for mission conditions. For known inertia tensors, the lumped bias can be estimated in real time as a system parameter within an augmented state vector using a dynamic filter such as an Extended Kalman Filter, H_∞ Filter or Sliding Mode Observer.

Analytical simulations are run on a quad-core Intel i7 processor running at 2.4Ghz. Here, computational demand is estimated using MatLab's "tic" and "toc" commands to record the time required by each filter/observer. Using numerical simulations with conditions defined for the MMS mission, the dynamic filters are all capable of estimating the lumped bias within mission tolerances. The H_∞ Filter is the best performing filter of the three, on average estimating the accelerometer bias to within $1/3\ \mu g$ and with approximately 10% more accuracy than the EKF. This accuracy comes at the cost requiring 110% of the computational demand the EKF required. Although the SMO yields the least accurate lumped bias estimation, it is still within mission specifications and uses 50% of the EKF's computational demand. There are also many adaptations which could be applied to the SMO to improve its performance.

Uncertainties in the inertia tensor result in an incorrect lumped bias estimation primarily as a result of constant drift error arising from the angular rate propagation. Additionally, the function $\boldsymbol{g}(\boldsymbol{x})$ utilizes the inertia tensor when computing the nominal acceleration for the filter innovations. If there is not perfect knowledge of the inertia tensor, the predicted nominal acceleration will be incorrect and the resulting

lumped accelerometer bias will also be incorrect. None of the filters meet the mission specification of $1 \times 10^{-5}\ m/s^2$.

Alternatively, regardless of any existing uncertainties in the inertia tensor, the spline fitting batch filter provides an accurate estimate of the lumped accelerometer bias to within 1 μg. The batch filter can be run periodically offline, or online at a delay of 5 seconds. This method, however, requires a large amount of computational power. The batch filter is verified numerically and qualitatively confirmed experimentally.

The cascading filter initially analyzed does not reliably provide an accurate estimate of the misalignment. Its estimate is entirely dependent on the residual error of the estimated bias and center of mass. Since the spacecraft has very little nutation, the misalignment and center of mass shift are nearly undistinguishable, as seen by the fact that both sources of error can be modeled as a single lumped bias.

1 Future Work for Filters/Estimators

Two changes can be made to the initial cascading filter to potentially provide more accurate results. Further research is suggested in attempting to constrain the rotation matrix in the least-square algorithm to output the best-fit orthogonal matrix, rather than the best-fit matrix. Additionally, it is suggested that the filter be run iteratively, re-estimating the accelerometer bias and center of mass shift after each misalignment estimate.

Filter performance is adequate but an EKF using multiplicative quaternion error may provide improved accuracy. The angular rate estimates are highly dependent on

the quaternion estimates and would likely see improvements if the quaternion errors are decreased.

A parallel EKF scheme as is outlined in [8] may be better able to estimate the lumped bias if the inertia tensor is not known. The state vector would likely need to be augmented by the angular acceleration as well.

2 Future Work for the Experimental Test Bed

The BMA180 has a number of unique features which can be used to improve the measurement noise levels and accuracy. Most of these features were not used, but could be enabled to improve the measurements. Specifically, two improvements are suggested. The accelerometer includes an interrupt line to indicate when new accelerometer values have been computed. If used appropriately, the microprocessor can use this information to communicate with the BMA180 while the less important temperature conversion is being performed. This reduces noise on the accelerometer measurement due to communication. To utilize this feature, a dedicated core is needed to anticipate and obtain accelerometer values when they are available. Secondly, filters can be enabled onboard the BMA180. Since measurements are only taken at 25 Hz, the accelerometer can enable a low pass filter attenuating higher frequencies than this.

The ad-hoc addition of the magnetometer was an update to previous IMU's using only a gyro and accelerometer. If a true KF or EKF is designed for the IMU utilizing all three sensors, the output attitude could likely be improved. A realtime clock

would also improve the IMU estimation and the data collection, providing a higher resolution and more accurate time stamp for each measurement.

Acceleration measurements due to gravity are assumed to be constant for each experiment. To improve the accelerometer measurement the gravity magnitude should be determined from the static calibration and then the gravity vector should be rotated to account for changes in attitude. The new rotated gravity vector should be used to correct the measurement accordingly. Since the accelerometer lumped bias is calculated using an average error over many rotations this has no effect on the overall estimation accuracy.

LIST OF REFERENCES

[1] European Space Agency. Cluster. http://sci.esa.int/science-e/www/area/index.cfm, July 2011.

[2] I. Y. Bar-Itzhack. Matrix symetrization. *Journal of Guidance, Control and Dynamics*, 21(1):178–179, September 1997.

[3] Itzhack Y. Bar-Itzhack, Richard R. Harman, and Julie K. Thienel. Rigid body rate inference from attitude variation. *Journal of Guidance, Control and Dynamics*, 30(1):275–281, September 2006.

[4] Hai bo Liu, Xiu jian Li, Ji chun Tan, Jian kun Yang, Jun Yang, De zhi Su, and Hui Jia. Novel approach for laboratory calibration of star tracker. *Opt. Eng*, 49(7), Jul 2010.

[5] J. L. Burch and W. S. Lewis. Magnetospheric multiscale mission. http://mms.space.swri.edu/, Feb 2011.

[6] E. V. Buyanov. Method and apparatus for accurate determination of the inertia tensor of a solid body. *Measurement Techniques*, 31:1181–1184, 1989.

[7] Goddard Space Flight Center. Magnetospheric multiscale code 46.1. http://mms.gsfc.nasa.gov/, July 2011.

[8] Guanrong Chen Charles Chui. *Kalman Filtering*. Springer, 2009.

[9] Citizendium. Euler angles. http://en.citizendium.org/wiki/Euler_angles, July 2011.

[10] Christopher Edwards and Sarah K. Spurgeon. *Sliding Mode Control Theory and Applications.* Taylor and Francis, Bristol, 1998.

[11] Arthur Gelb. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[12] Michael E. Hough. Recursive bias estimation and orbit determination. *Journal of Guidance, Control, and Dynamics*, 32(2):645–652, March 2009.

[13] Marshall H. Kaplan. *Modern Spacecraft Dynamics and Control.* Wiley, 1976.

[14] K. Koprubasi and M. L. Thein. Spacecraft attitude and angular rate estimation using sliding mode observer. In *Proceedings of the IEEE 2003 International Conference on Recent Advances in Space Technologiesv (RAST)*, pages 156–161, 2003.

[15] K. Koprubasi and M. L. Thein. attitude and angular rate estimation using the sliding mode observer with additive quaternion corrections. In *2006 Preceedings of the American Control Conference*, pages 3302–3307, 2006.

[16] K. Koprubasi and M. L. Thein. attitude and angular rate estimation using the sliding mode observer with multiplicative quaternion corrections for the co-operative astrophysics technology satellite (catsat). In *2006 Preceedings of the American Control Conference*, pages 1736–1741, 2006.

[17] Kerem Koprubasi. Spacecraft attitude and angular rate estimation using sliding mode observers. Master's thesis, Middle East Technical University, 2002.

[18] Jack Kuipers. *Quaternions and Rotation Sequences.* Princeton University Press, 1999.

[19] Christopher H. Lee. A phase space spline smoother for fitting trajectories. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 34 of *1*, pages 346–356. Robotics Institute at Carnegie Mellon University, 2004.

[20] J. A. Macinante. Recent developments in accelerometer calibration techniques. pages 381–392, 1974.

[21] F. Landis Markley. Attitude error representations for kalman filtering. Technical report, NASA's Goddard Space Flight Center.

[22] Vitaly G. Melnikov. A new method for inertia tensor and center of gravity identification. *Nonlinear Analysis*, 63:1377–1382, 2005.

[23] N. Mushaweh, B. Jenkins, D. Castelli, and M. L. Thein. A comparative analsysis of body-rate estimation techniques for the nasa magnetospheric multiscale (mms) mission spacecraft. In *Proceedings of the 2009 AIAA Guidance, Navigation, and Control Congress and Exposition*, page 5949, Chicago, IL, August 2009. AIAA.

[24] Neil Mushaweh. An observer-based attitude and nutation control and flexible dynamic analysis for the nasa magnetospheric multiscale mission. Master of science, University of New Hampshire, Durham, NH, 2007.

[25] Parallax. Propeller object exchange. http://obex.parallax.com/, March 2011.

[26] Hee-Young Park, Won Tchon Oh, Dae-Hee Youn, and Chungyong Lee. Performance improvement of array shape estimation using spline interpolation. pages 1715–1718, Seoul, Korea. Yonsei University.

[27] Kourosh Parsa, Jorge Angeles, and Aurn K Misra. Attitude calibration of and accelerometer array. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, volume 1, pages 129–134, Washington, DC, May 2002.

[28] Joseph E. Sedlak Kalman filter estimation of spinning spacecraft attitude using markley variables. In *18th International Symposium on Space Flight Dynamics*, Munich, Germany, October 2004.

[29] Malcolm D. Shuster. Maximum likelihood estimation of spacecraft attitude. *The Journal of the Astronautical Sciences*, 37(1):79–88, March 1989.

[30] Malcolm D. Shuster. Kalman filtering of spacecraft attitude and the quest model *The Journal of the Astronautical Sciences*, 38(3):377–393, September 1990.

[31] Malcolm D. Shuster. A survey of attitude representations. *The Journal of Astronautical Sciences*, 41(4):439–517, October 1993.

[32] Malcolm D. Shuster. The nature of the quaternion. *The Journal of the Astronautical Sciences*, 57(3), 2007.

[33] Malcolm D. Shuster and Daniel S. Pitone. Consistent estimation of spacecraft sensor alignments. In *Proceedings of the American Control Conference*, volume 1, pages 1389–1395, 1990.

[34] Marcel J. Sidi. *Spacecraft Dynamics and Control: A Practical Engineering Approach.* Cambridge University Press, New York, 1997.

[35] Dan Simon. *Optimal State Estimation.* John Wiley & Sons, Hoboken, New Jersey, 2006.

[36] Jean-Jacques Slotine and Weiping Li. *Applied Nonlinear Control.* Prentice-Hall, London, 1991.

[37] Young Soo Suh. Orientation estimation using a quaternion-based indirect kalman filter with adaptive estimation of external acceleration. *IEEE Transactions on Instrumentation and Measurement*, 59(12):3296–3305, December 2010.

[38] May-Win L. Thein. A discrete time variable structure observer with overlapping boundary layers. Department of Mechanical Engineering, University of New Hampshire.

[39] Julie K Thienel and F Landis Markley. Extended kalman filter for mms state estimation. 2009.

[40] Julie K. Thienel, F. Landis Markley, and Richard R. Harman. Extended kalman filter for mms state estimation. *American Institute of Aeronautics and Astronautics*, 2009.

[41] Bong Wie. *Space Vehicle Dynamics and Control.* AIAA, 1998.

[42] Maria Cecilia Zanardi and Malcolm D. Shuster. Batch and filter approaches to spacecraft sensor alignment estimation. In *Proceedings of the 12th International*

Symposium on 'Space Flight Dynamics', pages 159–166, Darmstadt, Germany, June 1997. European Space Operations Center.

Appendix A - Miscellaneous Content

1 Additional Dynamic Filter Results

Figures 6.1 through 6.3 present the EKF results for case 1

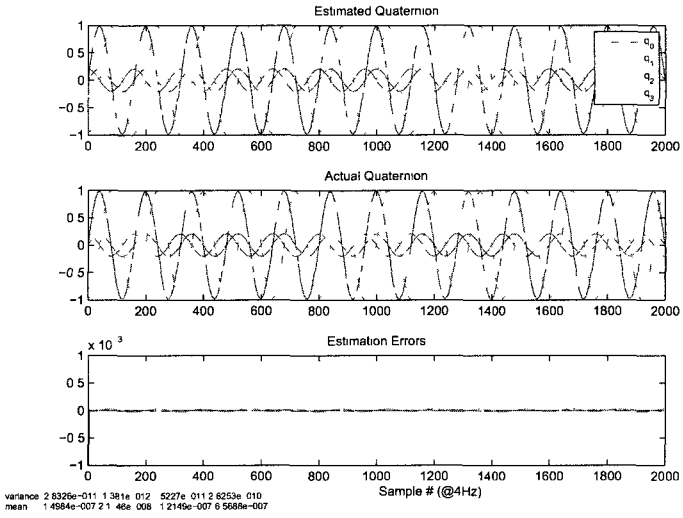


Figure 6.1: $\hat{q}$ subjected to bias errors and no measurement noise (known inertia)

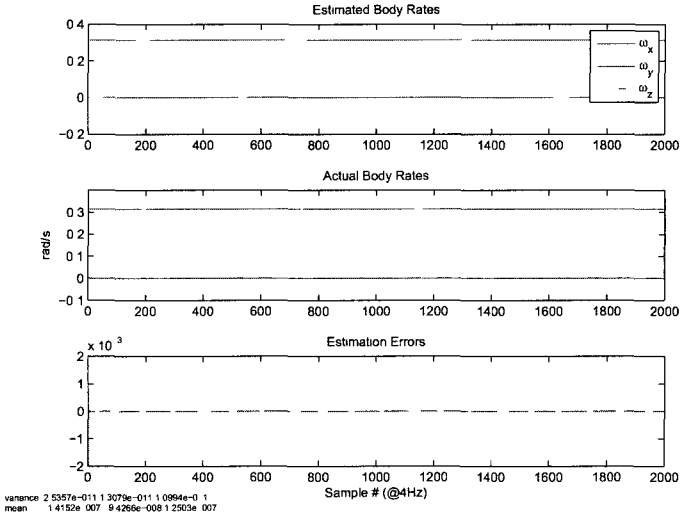


Figure 6.2: $\hat{\omega}$ subjected to bias errors and no measurement noise (known inertia)

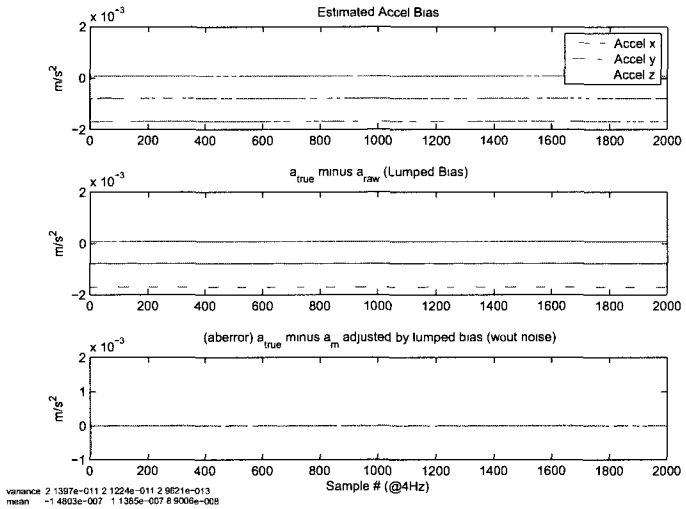


Figure 6.3: $\hat{a}_{LB}$: subjected to bias errors and no measurement noise (known inertia)

Figures 6.4 through 6.6 present the EKF results for case 3

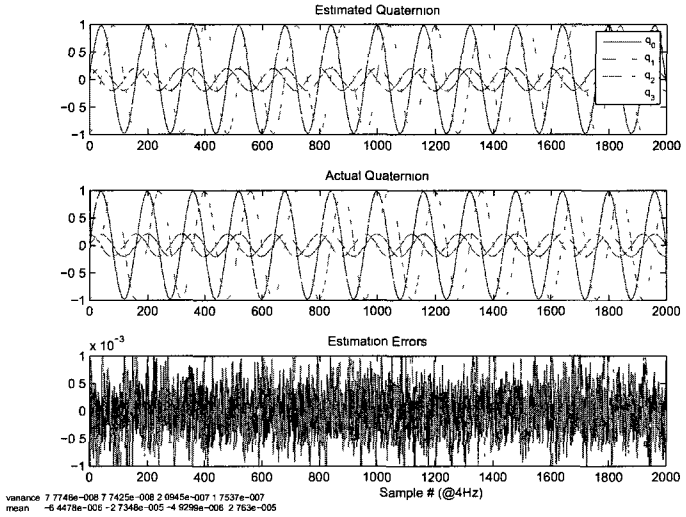


Figure 6.4: $\hat{q}$ subjected to bias errors and 10x-measurement noise (known inertia)

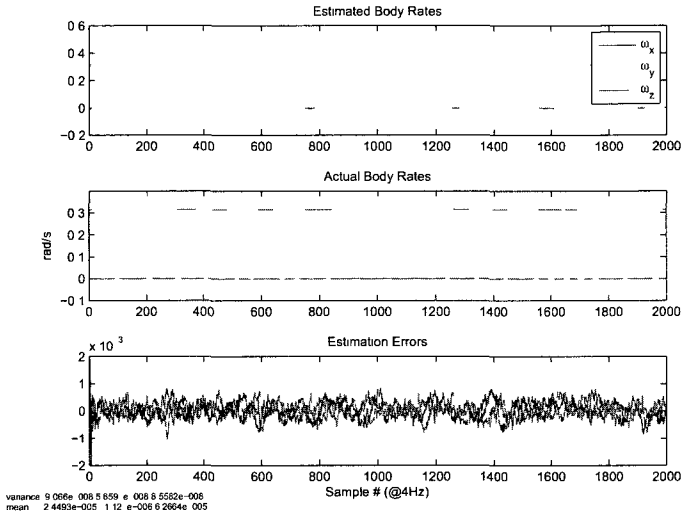


Figure 6.5: $\hat{\omega}$ subjected to bias errors and 10x-measurement noise (known inertia)

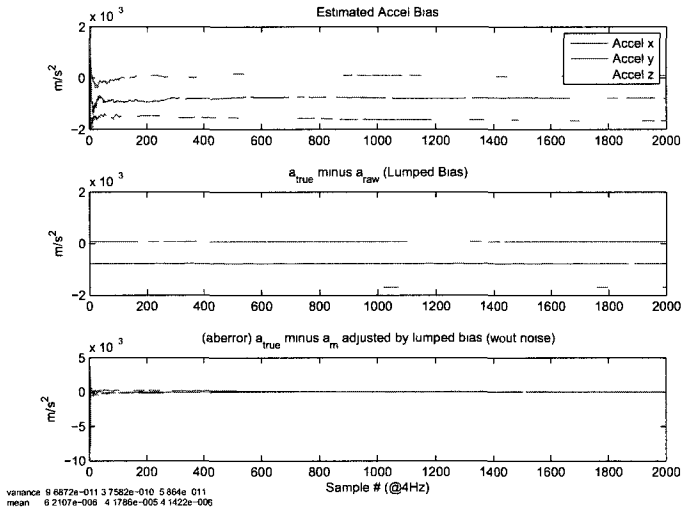


Figure 6.6: $\hat{a}_{LB}$ subjected to bias errors and 10x-measurement noise (known inertia)

Figures 6.7 through 6.9 present the H_∞ results for case 1

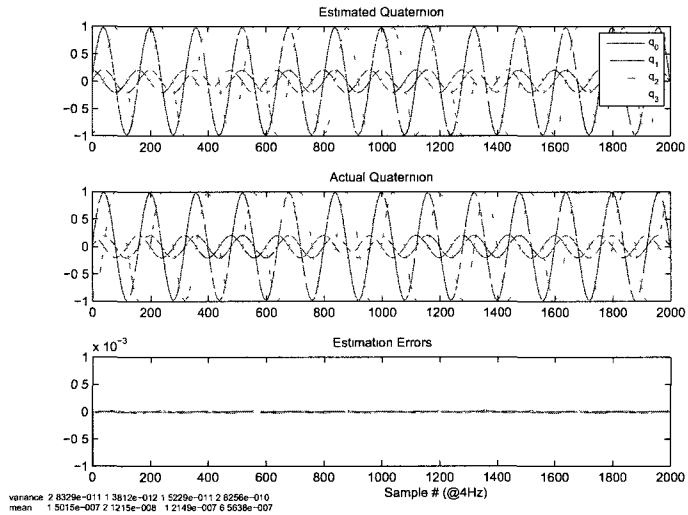


Figure 6.7: $\hat{q}$ subjected to bias errors and no measurement noise (known inertia)

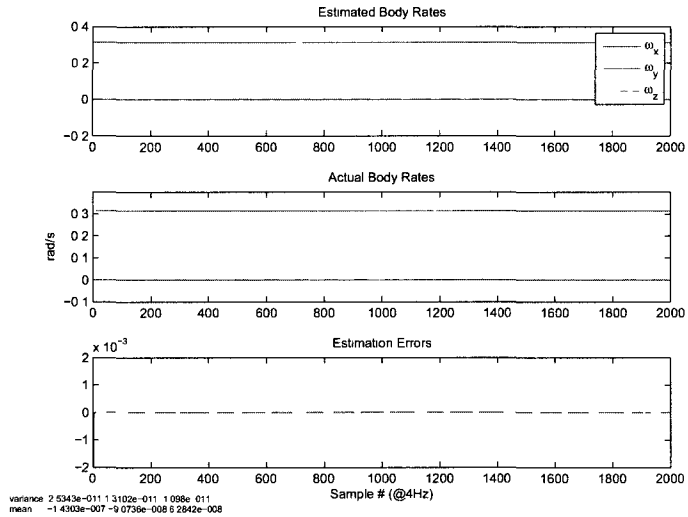


Figure 6.8: $\hat{\omega}$ subjected to bias errors and no measurement noise (known inertia)

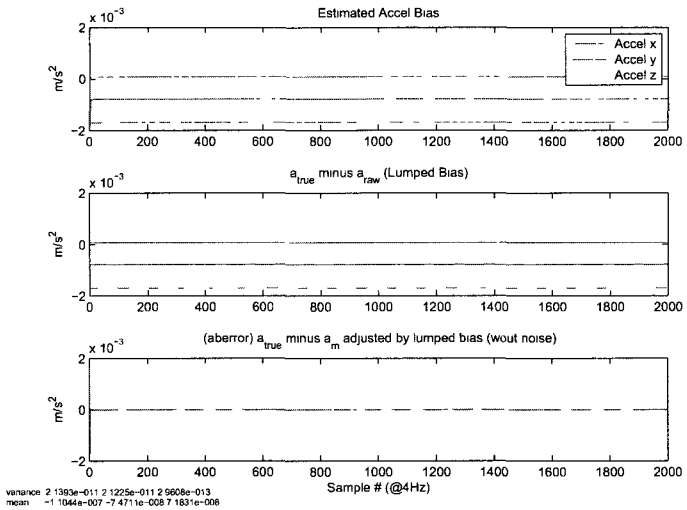


Figure 6.9: $\hat{a}_{LB}$ subjected to bias errors and no measurement noise (known inertia)

Figures 6.10 through 6.12 present the H_∞ results for case 3

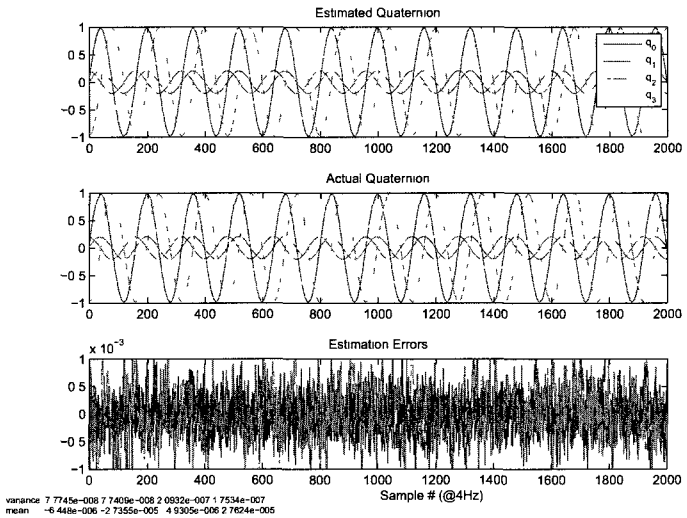


Figure 6.10: $\hat{q}$ subjected to bias errors and 10x-measurement noise (known inertia)

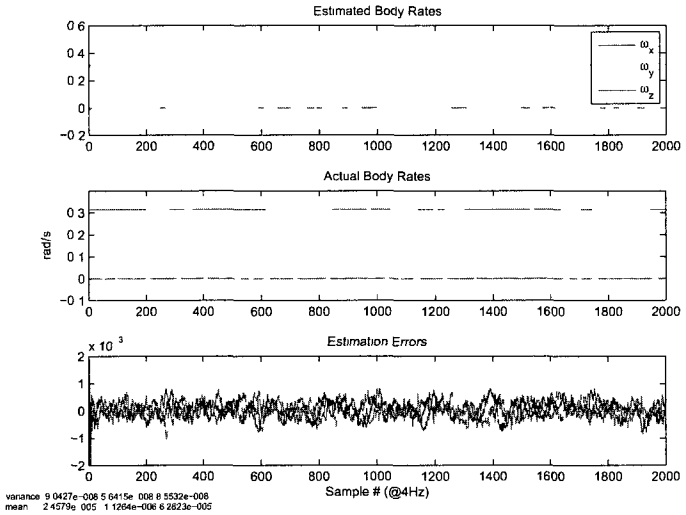


Figure 6.11: $\hat{\omega}$ subjected to bias errors and 10x-measurement noise (known inertia)

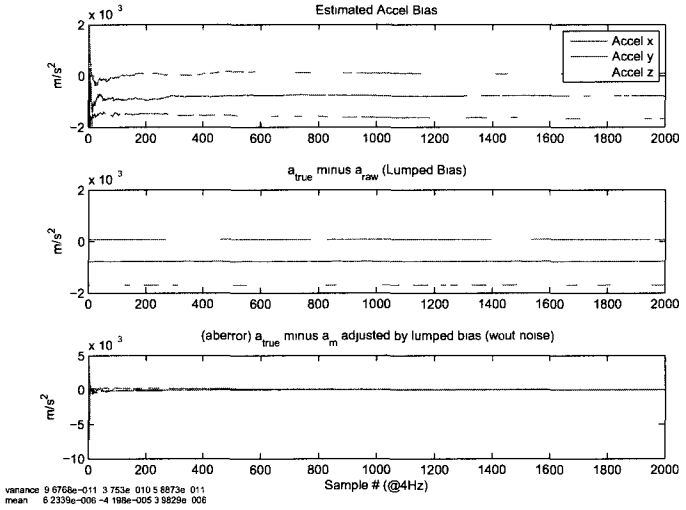


Figure 6.12: $\hat{a}_{LB}$ subjected to bias errors and 10x-measurement noise (known inertia)

Figures 6.13 through 6.15 present the SMO results for case 1

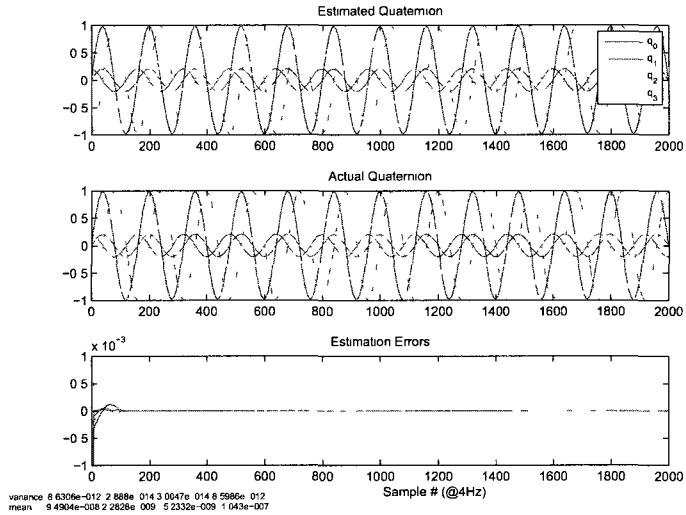


Figure 6.13: $\hat{q}$ subjected to bias errors and no measurement noise (known inertia)

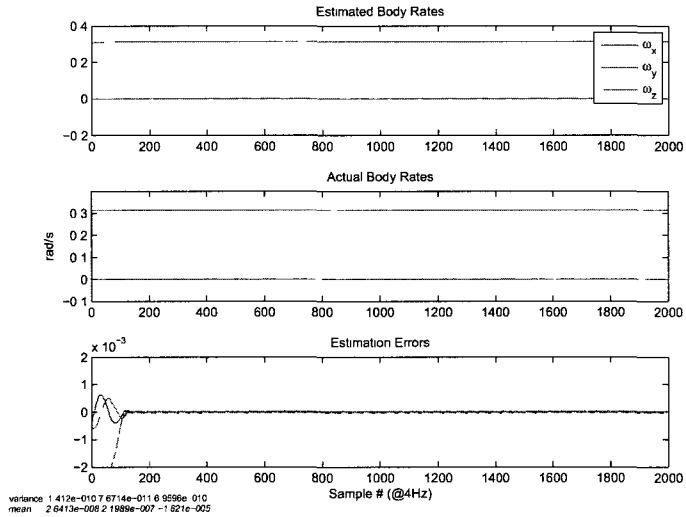


Figure 6.14: $\hat{\omega}$ subjected to bias errors and no measurement noise (known inertia)

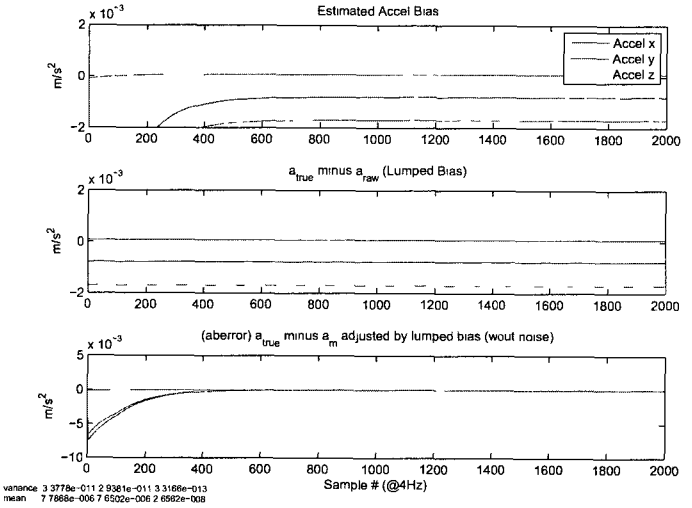


Figure 6.15: $\hat{a}_{LB}$ subjected to bias errors and no measurement noise (known inertia)

Figures 6.16 through 6.18 present the SMO results for case 3

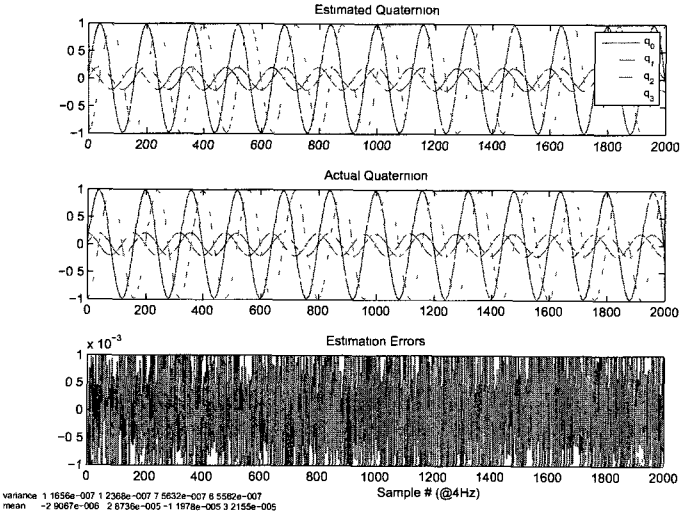


Figure 6.16: $\hat{q}$ subjected to bias errors and 10x-measurement noise (known inertia)

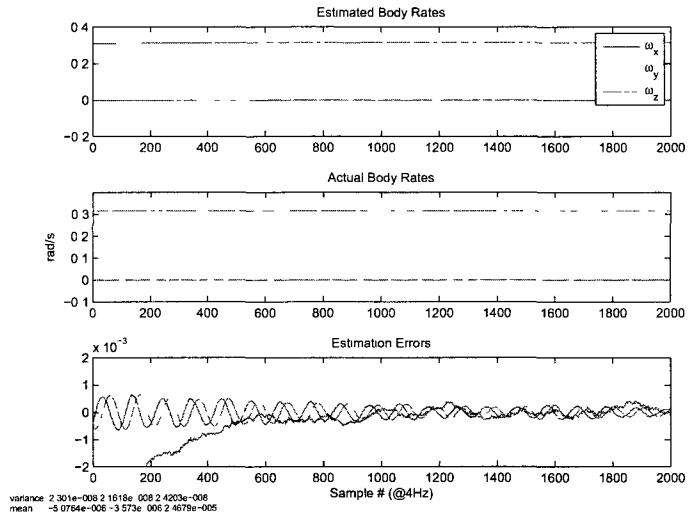


Figure 6.17: $\hat{\omega}$ subjected to bias errors and 10x-measurement noise (known inertia)

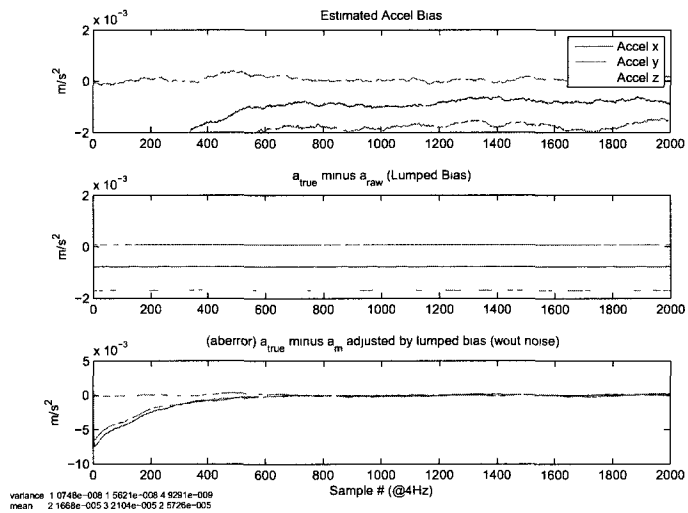


Figure 6.18: $\hat{a}_{LB}$ subjected to bias errors and 10x-measurement noise (known inertia)

2 Euler Angle Rotation Matrices

all in order of ϕ, θ, ψ
Roll - Pitch - Yaw (123)

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

RPR (121)

$$\begin{bmatrix} \cos(\theta) & \sin(\theta)\sin(\psi) & -\sin(\theta)\cos(\psi) \\ \sin(\phi)\sin(\theta) & -\sin(\phi)\cos(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta)\cos(\psi)+\cos(\phi)\sin(\psi) \\ \cos(\phi)\sin(\theta) & -\cos(\phi)\cos(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta)\cos(\psi)-\sin(\phi)\sin(\psi) \end{bmatrix}$$

RYR (131)

$$\begin{bmatrix} \cos(\psi) & \sin(\psi)\cos(\psi) & (\sin(\psi))^2 \\ -\cos(\phi)\sin(\psi) & \cos(\phi)(\cos(\psi))^2-\sin(\phi)\sin(\psi) & \cos(\phi)\cos(\psi)\sin(\psi)+\sin(\phi)\cos(\psi) \\ \sin(\phi)\sin(\psi) & -\sin(\phi)(\cos(\psi))^2-\cos(\phi)\sin(\psi) & -\sin(\phi)\cos(\psi)\sin(\psi)+\cos(\phi)\cos(\psi) \end{bmatrix}$$

RYP (132)

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\theta) & -\cos(\theta)\sin(\psi) \\ -\cos(\phi)\sin(\theta)\cos(\psi)+\sin(\phi)\sin(\psi) & \cos(\phi)\cos(\theta) & \cos(\phi)\sin(\theta)\sin(\psi)+\sin(\phi)\cos(\psi) \\ \sin(\phi)\sin(\theta)\cos(\psi)+\cos(\phi)\sin(\psi) & -\sin(\phi)\cos(\theta) & -\sin(\phi)\sin(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) \end{bmatrix}$$

PYR (231)

$$\begin{bmatrix} \cos(\phi)\cos(\theta) & \cos(\phi)\sin(\theta)\cos(\psi)+\sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) \\ -\sin(\theta) & \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) \\ \sin(\phi)\cos(\theta) & \sin(\phi)\sin(\theta)\cos(\psi)-\cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) \end{bmatrix}$$

PYP (232)

$$\begin{bmatrix} \cos(\phi)\cos(\theta)\cos(\psi)-\sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta) & -\cos(\phi)\cos(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) \\ -\sin(\theta)\cos(\psi) & \cos(\theta) & \sin(\theta)\sin(\psi) \\ \sin(\phi)\cos(\theta)\cos(\psi)+\cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta) & -\sin(\phi)\cos(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) \end{bmatrix}$$

PRP (212)

$$\begin{bmatrix} (\cos(\phi))^2-(\sin(\phi))^2\cos(\theta) & \sin(\phi)\sin(\theta) & -\cos(\phi)\sin(\phi)-\sin(\phi)\cos(\theta)\cos(\phi) \\ \sin(\phi)\sin(\theta) & \cos(\theta) & \cos(\phi)\sin(\theta) \\ \cos(\phi)\sin(\phi)+\sin(\phi)\cos(\theta)\cos(\phi) & -\cos(\phi)\sin(\theta) & -(\sin(\phi))^2+(\cos(\phi))^2\cos(\theta) \end{bmatrix}$$

PRY (213)

$$\begin{bmatrix} \cos(\phi)\cos(\psi)-\sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\sin(\psi)+\sin(\phi)\sin(\theta)\cos(\psi) & -\sin(\phi)\cos(\theta) \\ -\cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi) & \sin(\theta) \\ \sin(\phi)\cos(\psi)+\cos(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\sin(\psi)-\cos(\phi)\sin(\theta)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

YRP (312)

$$\begin{bmatrix} \cos(\phi)\cos(\psi)+\sin(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\cos(\theta) & -\cos(\phi)\sin(\psi)+\sin(\phi)\sin(\theta)\cos(\psi) \\ -\sin(\phi)\cos(\psi)+\cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) & \sin(\phi)\sin(\psi)+\cos(\phi)\sin(\theta)\cos(\psi) \\ \cos(\theta)\sin(\psi) & -\sin(\theta) & \cos(\theta)\cos(\psi) \end{bmatrix}$$

YRY (313)

$$\begin{bmatrix} -\sin(\phi)\cos(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta)\cos(\psi)+\cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta) \\ -\cos(\phi)\cos(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta)\cos(\psi)-\sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta) \\ \sin(\theta)\sin(\psi) & -\sin(\theta)\cos(\psi) & \cos(\theta) \end{bmatrix}$$

YPY (323)

$$\begin{bmatrix} \cos(\phi)\cos(\theta)\cos(\psi)-\sin(\phi)\sin(\psi) & \cos(\phi)\cos(\theta)\sin(\psi)+\sin(\phi)\cos(\psi) & -\cos(\phi)\sin(\theta) \\ -\sin(\phi)\cos(\theta)\cos(\psi)-\cos(\phi)\sin(\psi) & -\sin(\phi)\cos(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) & \sin(\phi)\sin(\theta) \\ \sin(\theta)\cos(\psi) & \sin(\theta)\sin(\psi) & \cos(\theta) \end{bmatrix}$$

YPR (321)

$$\begin{bmatrix} \cos(\phi)\cos(\theta) & \sin(\phi)\cos(\psi)+\cos(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\sin(\psi)-\cos(\phi)\sin(\theta)\cos(\psi) \\ -\sin(\phi)\cos(\theta) & \cos(\phi)\cos(\psi)-\sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\sin(\psi)+\sin(\phi)\sin(\theta)\cos(\psi) \\ \sin(\theta) & -\cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi) \end{bmatrix}$$

3 Microcontroller Code

```
{{
////////////////////////////////////////////////////
// SD3.01 FATEngine Demo
//
// Author: Kwabena W. Agyeman
// Updated: 1/2/2011
// Designed For: P8X32A
// Version: 1.0
//
// Copyright (c) 2011 Kwabena W. Agyeman
// See end of file for terms of use.
//
// Update History:
//
// v1.0 - Original release - 7/27/2010.
// v1.1 - Updated code for new file system driver - 1/2/2011.
//
// Type "help" into the serial terminal and press enter after
//        connecting to display the command list.
//
// Nyamekye,
////////////////////////////////////////////////////////////
}}
CON
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000
  _baudRateSpeed = 115_200
  _newLineCharacter = 13
  _clearScreenCharacter = 16
  _homeCursorCharacter = 1
  _NL = 10              ' works in wordpad but not notepad
  _CR = 13
  _comma = 44
  _semi = 59
  _receiverPin = 31
  _transmitterPin = 30
  _RTC_DAT = -1 ' -1 ifnot installed.
  _RTC_CLK = -1 ' -1 ifnot installed.
  _SD_DO = 0
  _SD_CLK = 1
  _SD_DI = 2
  _SD_CS = 3
  _SD_WP = -1 ' -1 ifnot installed.
  _SD_CD = -1 ' -1 ifnot installed.
  _imuRX = 13
  _imuTX = 12
  _imuBAUD = 57600
  _imuMODE = 0
  _accCS = 4
  _accSCK = 5
  _accSDO = 6
  _accSDI = 7
  _MSBFIRST = 5   ' for shiftOUT
  _MSBPOST = 2    ' for shiftIN
  _accXlsb = 130
OBJ
  fat: "SD3.01_FATEngine.spin"
'  taf: "SD3.01_FATEngine.spin"
'  rtc: "DS1307_RTCEngine.spin"
```

```
'  com: "RS232_COMEngine.spin"
  str: "ASCII0_STREngine.spin"
  pst:  "Parallax Serial Terminal"
  imu:  "FullDuplexSerial"
'  spi:  "SPI_Spin"
  spi:  "SPI_Asm"
  timer: "Timer2"       ' modified timer2 to have "," instead of :
VAR
  byte tbyte
  byte buf
  byte i, j
  byte imuOUT[256]
  byte accOUT[256]
  byte accTMP[8]
  byte teststring[256]
  byte value
  byte B1, B2, B3
  byte command
  long clockPTR
PUB main
  fat.FATEngineStart(_SD_DO, _SD_CLK, _SD_DI, _SD_CS,...
                _SD_WP, _SD_CD, _RTC_DAT, _RTC_CLK, -1)
  pst.start(_baudRateSpeed)
  imu.start(_imuRX,_imuTX,_imuMODE,_imuBAUD)
    pst.str(string("starting up",13))
    teststring := string("test string stuff2")
    pst.str(teststring)
    fat.mountPartition(0)
    pst.str(string("partition mounted"))
'    fat.openFile(string("TESTTXT4.txt"), "A")
    fat.deleteEntry(string("IMU010.txt"))
    pst.str(string("File Deleted"))
    fat.newFile(string("IMU010.txt"))
    fat.openFile(string("IMU010.txt"), "A")
    pst.str(string("file opened for write"))
    timer.start
    timer.run
  repeat 10000
    clockPTR := timer.showTimer
    readIMU(@imuOUT)
    readACC(@accOUT)
'    pst.str(clockPTR)
'    pst.str(@imuOUT)
'    pst.str(@accOUT)
    fat.writeString(clockPTR)
    fat.writeByte(_comma)
    fat.writeString(@imuOUT)
    fat.writeByte(_comma)
    fat.writeString(@accOUT)
'    fat.writeByte(_comma)
    fat.writeByte(_CR)
'    pst.char(13)
  pst.str(string("Closing File and Unmounting System"))
  fat.closeFile
  fat.unmountPartition
  pst.str(string("you may now shut off the uC"))
```

```
PRI readACC(strPtr)
'  SHIFTOUT(Dpin, Cpin, Mode, Bits, Value)
'  SHIFTIN(Dpin, Cpin, Mode, Bits)|Value
'  PUB start(_ClockDelay, _ClockState)
          ' clockdelay in uS and clock state 0or1
  dira[_accCS]~~
          ' make chip select pin output
  outa[_accCS] := 0
  i := 0
  spi.start(1,1)                ' 20 works
  command := _accXlsb
  spi.SHIFTOUT(_accSDI,_accSCK,_MSBFIRST,8,command)
'  pst.str(String("command sent"))
  repeat 7
    byte[@accTMP + i] := spi.SHIFTIN(_accSDO,_accSCK,_MSBPOST,8)
'    byte[@accTMP + i] := 123
'    pst.char(byte[@accTMP + i] )
    i++
  outa[_accCS] := 1
'  pst.str(String("done shiftin"))
  i := 0
  j := 0
  repeat 7
    value := byte[@accTMP + i]
    B1 := value/100
    B2 := (value // 100)/10
    B3 := (value // 10)
    byte[strPtr + j] := B1 + 48
    j++
    byte[strPtr + j] := B2 + 48
    j++
    byte[strPtr + j] := B3 + 48
    j++
    byte[strPtr + j] := ","
    j++
    i++
  byte[strPtr + j] := 0

PRI readIMU(strPtr)
  buf := imu.rx
  i := 0
  repeat until buf == "!"
    buf := imu.rx
  buf := imu.rx
  repeat until buf == "@"
    byte[strPtr + i] := buf
    buf := imu.rx
    i++
'  byte[strPtr + i] := buf
  byte[strPtr + i]  := 0

PRI multiplyDivide(dividen, divisor) | productHigh, productLow
  productHigh := clkfreq ** dividen
  productLow := clkfreq * dividen
  if(productHigh => divisor)
    return posx
  repeat 32
    dividen := (productHigh < 0)
    productHigh := ((productHigh << 1) + (productLow >> 31))
    productLow <<= 1
```

```
    result <<= 1
    if((productHigh => divisor) or dividen)
      productHigh -= divisor
      result += 1
{{
//////////////////////////////////////////////////////////////////
//                TERMS OF USE: MIT License                     //
//////////////////////////////////////////////////////////////////
// Permission is hereby granted, free of charge, to any person
// obtaining a copy of this software and associated documentation
// files (the "Software"), to deal in the Software without
// restriction, including without limitation the rights to use,
// copy, modify, merge, publish, distribute, sublicense, and/or
// sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following
// conditions:
//
// The above copyright notice and this permission notice shall
// be included in all copies or substantial portions of the
// Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
// KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
// PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
// COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
// USE OR OTHER DEALINGS IN THE SOFTWARE.
//////////////////////////////////////////////////////////////////
}}
```

Appendix B - MatLab Code

file_00010_EKF09.m - primary file used, calls all other .m files to perform simulations

constants.m - declares all the system constants required for simulations

gen_sim_data.m - formulates all the sensor measurements from the simulation quaternions and angular rates

1 Constants.m

```
% Constants .m file
% sets up the: Inertia, IC's
```

Declare Inertias

```
J = [783.35, -12.28, -4.84;...
     -12.28, 803.79, -7.67;...
     -4.84,  -7.67,  1332.99];
Jhat = [800, 0, 0;...
     0, 800, 0;...
     0,  0,  1300];
if Jknown == 1
    J = Jhat;                    % use a simple J for initial testing
end
Jneg = J^-1;
Ix0 = J(1,1);
Iy0 = J(2,2);
Iz0 = J(3,3);
% For the angular velocity estimatJn
Iwx0 = (Iy0-Iz0)/Ix0;
Iwy0 = (Iz0-Ix0)/Iy0;
Iwz0 = (Ix0-Iy0)/Iz0;
```

Declare Initial Conditions (attitude, body rate, torques, bias, COM, COM

error

Attitude note that the last element is the scalar (from Dr Thineil) whereas i use the FIRST element as the SCALAR

```
qi = [0.0183;0.2026;-0.9751;0.0880];
q0i = qi(4);
q1i = qi(1);
q2i = qi(2);
q3i = qi(3);
qi = [q0i; q1i; q2i; q3i];
% Body Rates
uxy = rand(2,1);
uxy = uxy/norm(uxy);
coning = normrnd(0,0.2*pi/180);
wi = [3*sin(coning)*uxy;3*cos(coning)]*2*pi/60
wxi = wi(1);
wyi = wi(2);
```

```
wzi = wi(3);
% Torques
Ti = [0;0;0];
% Accelerometer Bias
abi = normrnd(0,1e-5,3,1);
% Center of Mass
rci = normrnd(0,5/100,3,1);
% rci = normrnd(0,1/100,3,1);
% Measured Center of Mass
rAi = [0.75;0.75;0.5];
% rci = rci/10;       % REDUCED THE SIZE OF rc
ri = rAi + rci;
```

```
wi =
   -0.0011
   -0.0011
    0.3142
```

Calculate Initial Conditions (angular velocity)

```
widot = Jneg*fcross(J*wi)*wi + Jneg*Ti
```

```
widot =
  1.0e-003 *
    0.2202
   -0.2185
   -0.0000
```

Filter Initial Conditions

```
q_hati = [1;0;0;0];
w_hati = [0,0,0];
rc_hati = [0;0;0];
ab_hati = [0;0;0];
x0 = [qi;wi]
state_rate = 1
```

```
x0 =
    0.0880
    0.0183
    0.2026
   -0.9751
   -0.0011
   -0.0011
    0.3142
state_rate =
     1
```

2 gen_sim_data.m

```
% need to change this since we add noise directly to the quaternions
% qnoisesigma = 3.23209*10^-4; % converts to 3 sigma of 200 arcsec
qnoisevar = qnoisesigma^2;
% anoisesigma = 1*10^-4;
anoisevar = anoisesigma^2;
```

```
Error using ==> evalin
Undefined function or variable 'qnoisesigma'.
```

run simulation

```
sim('Plant2')
N = size(qsim,1) - 1;
rc = rci;
r = ri;
r1 = r(1);
r2 = r(2);
r3 = r(3);
rc1 = r(1);
rc2 = r(2);
rc3 = r(3);
RC = [rc1,rc2,rc3];
rA = rAi;
Jw = J;
% xi = [qi;wi];
```

account for corruptions

```
DELTA = eye(3) + fcross(delta);
qsim = qsim';
wsim = wsim';
if Jknown == 1;
    J = Jhat;  %%%%%%%%% this is for testing purposes
end
% preallocate measurement and true state arrays
anoise = zeros(3,N);
wdotsim = zeros(3,N);
areal = zeros(3,N);
amis = zeros(3,N);
asim = zeros(3,N);
a_m = zeros(3,N);
esim = zeros(3,N);
e2qsim = zeros(4,N);
r = ri;
for i = 1:N
%     esim(1:3,i) = Q2E_B(qsim(1:4,i),1);
%     e2qsim(1:4,i) = E2Q_B(esim(1:3,i),6);
    q = [qsim(2:4,i);qsim(1,i)];
    esim(1:3,i) = SpinCalc('QtoEA321',q',eps,0)';
    if esim(2,i) > 90
        esim(2,i) = esim(2,i) - 360;
    end
    qtemp = SpinCalc('EA321toQ', esim(1:3,i)',eps,1)';
    e2qsim(1:4,i) = [qtemp(4);qtemp(1:3)];
%     anoise(1:3,i) = random('norm',0,1,3,1)'*sqrt(anoisevar);
    anoise(1:3,i) = random('norm',0,anoisesigma,3,1)';
    wdotsim(1:3,i) = J\fcross(J*wsim(1:3,i))*wsim(1:3,i);
    areal(1:3,i) = fcross(wdotsim(1:3,i))*rA + fcross(wsim(1:3,i))^2*rA;
    areal2(1:3,i) = fcross(wdotsim(1:3,i))*r + fcross(wsim(1:3,i))^2*r;
    amis(1:3,i) = DELTA*areal2(1:3,i);
    asim(1:3,i) = amis(1:3,i) + ab;
    a_m(1:3,i) = asim(1:3,i) + anoise(1:3,i);
    q_m(1:4,i) = qsim(1:4,i) + random('norm',0,1,4,1).*qnoisesigmavec;
end
wdot_true = wdotsim;
a_true = areal;
a_raw = asim;   % accelerometer measurement minus the noise
```

```
% (accounts for uncerts)
```

3 file_00010_EKF09.m

```
clc
clear
tic
printplots = 1;
```

Declare simulation parameters

```
Jknown = 1;      % toggle J = Jhat
% Constants
constants_smallr
%disturbance torques on?
amp1 = 0*0.01;     % amplitudes of the disturbance torques
Tbias1 = 0;        % bias on sinusoidal torque
Tfreq1 = 0*2*pi/100; % freq for sinusoidal torque
amp2 = 0*0.01;
Tbias2 = 0;
Tfreq2 = 0;
amp3 = 0*0.01;
Tbias3 = 0;
Tfreq3 = 0;
% clear r
%rci  =[0.005;-0.005;0.005];
%ri = rAi + rci;


wi =
    0.0001
    0.0007
    0.3142
widot =
  1.0e-003 *
   -0.1421
    0.0231
   -0.0000
x0 =
    0.0880
    0.0183
    0.2026
   -0.9751
    0.0001
    0.0007
    0.3142
state_rate =
     1
```

Generate simulation data

```
%use first two lines to change simulation runtime and sensor rate
EndTime = 500;       % was 500  % 5000 averages out random
                     % means to close to zero
state_rate = 0.25;%0.25;   use for experimental 0.02
% specify noise levels (variances calculated in next m file)
qnoisesigma = 0*3.23209*10^-4;
qnoisesigmavec = 0*[0.0486;0.0485;0.1145;0.1143]*10^-3;
anoisesigma = 0*1.465*10^-4;
```

```
% qnoisesigmavec = [0.0486;0.0485;0.1145;0.1143]*10^-2;
% specify corruption levels here
ab = 0*[2;-2;1]*10^-4;        % m/sec^2
% ab = 1*[1;0;0]*10^-4;
% ab = 1*[0;1;0]*10^-4;
% ab = 1*[0;0;1]*10^-4;
delta = 0*[1;1;1]*10^-4;      % radians: 10^-4 >> 20 arcsec
% delta = 1*[1;0;0]*10^-4;
% delta = 1*[0;1;0]*10^-4;
% delta = 1*[0;0;1]*10^-4;
delta = 10*0.01745*[1;0;0];     % 1 degree >> 0.01745 rad
rci = 0*[1;1;-1]*10^-2;       % meters
% rci = 1*[1;0;0]*10^-2;        % 5cm
% rci = 1*[0;1;0]*10^-2;
% rci = 1*[0;0;1]*10^-2;
ri = rAi + rci;
gen_sim_data   %calls an m file
                % change "Plant2" to "Plant2_sinusoidaltorques"
                % in "gen_sim_data" for sinusoidal
                % torques


Error using ==> sim
Error using ==> targetman>throw_make_error at 517
(SLSF Diagnostic).


print('-sPlant2','-deps','-r300','-loose','Plant2print.eps')
print('-sPlant','-deps','-r300','-loose','Plant2subPlantprint.eps')
% importTdata
% importTdata02_const
% check_conv
```

run the EKF filter

% % declare the propagation timestep divisor (dT = state_rate/divisor)

```
divisor = 50;
% %
% % % comp_ekf(q_m,a_m,qsim,wsim,a_raw,a_true,...
% % % state_rate,divisor,anoisevar,RC,J)
% % % pause(1)
% % % comp_hinf(q_m,a_m,qsim,wsim,a_raw,a_true,...
% % % state_rate,divisor,anoisevar,RC,J)
% %
% % % run_hinf02
% % % run_smo01;
% % % run_ekf01
% % % batchab
% %
% %
% % %  run_ekf08_new02
% % % run_ekf08
% % % run_ekf09
% %
Jreal = J;
```

```
J = Jhat;
run_ekf46_multtest
% run_ekf46
% run_ekf43_sliding
% run_avgekf43_smo04
% run_hinf23
```

plot results

```
if exist('images','dir') == 0
    mkdir('images')
end
plot_ekf08ekf
%
% figure(8), plot(esim');
% legend('roll','pitch','yaw')
%
% figure(9), plot(e2qsim');
%
% % % % % plot_ekf08_new01
%
pause(1)
% clear Xhat
% run_hinf01
% plot_hinf01
```

fitting curve test

COMMENT HERE DOWN TO GET RID OF BATCH PROCESS

N = 1800; %Nexp; <– for experimental run M should be about 1/4 of a cycle worth of samples

```
M = 45; %101; %45  %21  % 45 works better for axis 1 and 2
                        % but worse for 3
% 35 is great     % must be odd
omegahat = zeros(3,N-M*2);
fitorder = 6   % 4 causes a bias in one of the terms % 6 works well
fitline = 0;  % determines if the fitline plots should be shown
if fitline == 1
    aviobj = avifile('qfitavi','compression','none','fps',10)
end
for s = 1:N-M*2
    point = s+M;
    qfit03                      % this calls the fit
                                % function for the quaternion
    qfilt(:,s) = qfilt_0';
    qdothat(:,s) = qdot';
    omegahat(:,s) = omega3(:);
    qhat2(:,s) = y(ceil(M/2),:)';
end
if fitline == 1
    aviobj = close(aviobj);
end
errorvarbatchqfilt = var(qfilt' - qsim(:,M:N-M-1)')
errormeanbatchqfilt = mean(qfilt' - qsim(:,M:N-M-1)')
figure(41), subplot(2,1,1), plot(qfilt')
```

```
% figure(12), subplot(2,1,1), plot(t2(M:N-M-1),omegahat')
title('Filtered Quaternion')
ylabel('Quaternion')
% ylim([-1,1]*10^-3)
subplot(2,1,2), plot(qfilt' - qsim(:,M:N-M-1)')
ylabel('Quaternion')
title('qfilt - qsim (q without noise)')
xlabel('                                                Sample # (@4Hz)')
% legend(num2str(errorvarbatchqfilt))
annotation('textbox',[0.01,0.075,0,0],'string',strvcat(...
['variance: ' num2str(errorvarbatchqfilt)],['mean:     '...
 num2str(errormeanbatchqfilt)]),'FitBoxToText','on',...
 'FontSize',7,'LineStyle','none')
if printplots == 1
    print -depsc -tiff images\omegahatbatch
    print -dpdf images\omegahatbatch
end
errorvarbatchwhat = var(omegahat' - wsim(:,M:N-M-1)')
errormeanbatchwhat = var(omegahat' - wsim(:,M:N-M-1)')
figure(12), subplot(2,1,1), plot(omegahat')
% figure(12), subplot(2,1,1), plot(t2(M:N-M-1),omegahat')
title('omegahat')
ylabel('rad/s')
% ylim([-1,1]*10^-3)
subplot(2,1,2), plot(omegahat' - wsim(:,M:N-M-1)')
ylabel('rad/s')
title('omegahat - wsim')
xlabel('Sample # (@4Hz)')
legend(num2str(errorvarbatchwhat))
annotation('textbox',[0.01,0.075,0,0],'string',strvcat(...
['variance: ' num2str(errorvarbatchwhat)],['mean:     '...
 num2str(errormeanbatchwhat)]),'FitBoxToText','on',...
 'FontSize',7,'LineStyle','none')
if printplots == 1
    print -depsc -tiff images\omegahatbatch
    print -dpdf images\omegahatbatch
end
% point = 70;
% M = 23
%
%
% qfit02
N2 = size(omegahat,2)
M = 51; %101; %51   %51
omegafilt = zeros(3,N2-M*2);
fitorder = 2
for s = 1:N2-M*2
    point = s+M;
    wfit01      % calls fit function for omega
    omegafilt(:,s) = what2;
    omegadot(:,s) = wdot;
end
begin = (N - (N2-2*M))/2
errorvarbatchwfilt = var(omegafilt' - wsim(:,begin+1:N-begin)')
errormeanbatchwfilt = mean(omegafilt' - wsim(:,begin+1:N-begin)')
figure(13), subplot(2,1,1), plot(omegafilt')
```

```
ylabel('rad/s')
title('omegafilt')
subplot(2,1,2), plot(omegafilt' - wsim(:,begin+1:N-begin)')
ylabel('rad/s')
title('omegafilt - wsim')
xlabel('Sample # (@4Hz)')
legend(num2str(errorvarbatchwfilt))
annotation('textbox',[0.01,0.075,0,0],'string',strvcat(...
['variance: ' num2str(errorvarbatchwfilt)],['mean:     '...
 num2str(errormeanbatchwfilt)]),'FitBoxToText','on',...
 'FontSize',7,'LineStyle','none')
if printplots == 1
    print -depsc -tiff images\omegafilt
    print -dpdf images\omegafilt
end
figure(14), subplot(2,1,1), plot(omegadot')
ylabel('rad/s^2')
title('omegadot, estimated from q')
subplot(2,1,2), plot(wdotsim(:,begin+1:N-begin)')
ylabel('rad/s^2')
title('wdotsim, real wdot')
xlabel('Sample # (@4Hz)')
% myfft2(omegahat(1,:),dT)
r0 = r;
N3 = size(omegafilt,2)
for i = 1:N3
    ahat(:,i) = fcross(omegadot(:,i))*r0 + fcross(omegafilt(:,i))^2*r0;
end
abreal = mean(areal(:,begin+1:N-begin)' - a_m(:,begin+1:N-begin)')
abhat = mean(ahat' - a_m(:,begin+1:N-begin)')
aberror = abreal - abhat
abpercent = aberror./abreal*100
figure(15), subplot(3,1,1), plot(ahat')
title('ahat')
ylabel('m/s^2')
subplot(3,1,2), plot(areal(:,begin+1:N-begin)')
title('areal')
ylabel('m/s^2')
subplot(3,1,3), plot(ahat' - areal(:,begin+1:N-begin)')
title('ahat - areal')
ylabel('m/s^2')
xlabel('Sample # (@4Hz)')
if printplots == 1
    print -depsc -tiff images\abiasbatch1
    print -dpdf images\abiasbatch1
end
figure(16), subplot(2,1,1), plot(areal(:,begin+1:N-begin)'...
 - asim(:,begin+1:N-begin)')
title('areal - asim (asim = am w/out noise)')
ylabel('m/s^2')
subplot(2,1,2), plot(ahat' - asim(:,begin+1:N-begin)')
title('ahat - asim (asim = am w/out noise)')
ylabel('m/s^2')
xlabel('Sample # (@4Hz)')
if printplots == 1
    print -depsc -tiff images\accelcorrectionsbatch1
```

```
    print -dpdf images\accelcorrectionsbatch1
end

for i=1:N
    abhatsim(:,i) = -abhat';
end
errormean2nd50ab = mean(-abhatsim(:,floor(N/2):N)'...
 - (a_true(:,floor(N/2):N)'- a_raw(:,floor(N/2):N)'));
errorvar2nd50ab = var(-abhatsim(:,floor(N/2):N)'...
 - (a_true(:,floor(N/2):N)'- a_raw(:,floor(N/2):N)'));
figure(17), plot(a_true' - (a_raw' - abhatsim'))
title('(aberror) a_{true} minus a_m adjusted by...
 lumped bias (w\out noise)')
ylabel('m/s^2')
xlabel('Sample # (@4Hz)')
% legend(strvcat(['variance: ' num2str(errorvar2nd50ab)]...
,['mean:     ' num2str(errormean2nd50ab)]))
annotation('textbox',[0.01,0.075,0,0],'string',...
strvcat(['variance: ' num2str(errorvar2nd50ab)],...
['mean:     ' num2str(errormean2nd50ab)]),...
'FitBoxToText','on','FontSize',7,'LineStyle','none')
if printplots == 1
    print -depsc -tiff images\acorrectedbatch1
    print -dpdf images\acorrectedbatch1
end
toc
```

4 run_efk46propR.m

```
%% ekf01 - estimate spin rate from just quaternions
dT = state_rate/divisor;
% checkQ
a = 1;
b = 1;
% a = 1/2;
% b = 1/2;
rc1 = rAi(1);
rc2 = rAi(2);
rc3 = rAi(3);
r = rAi;
%% Define P0 x0 Q0 and R0
P00 = eye(10);
abhat00 = zeros(3,1);
xhat00 = [0;0;0;0;0;0;0.31;abhat00];
Qq = eye(4)*10^-9% *5;   % 11
Qa = eye(3)*10^-10  %9 for J unknown; 11 for known
Qab = eye(3)*10^-20;  %13; %20;
Q = [Qq, zeros(4,6); zeros(3,4), Qa, zeros(3,3); zeros(3,7), Qab];
% Q = diag([Q_what,10^-20,10^-20,10^-20]);
Rq = [2,0,0,0;0,2,0,0;0,0,13,0;0,0,0,13]*10^-9;
% Rq = eye(4)*10^-11;%*100;   %15
Ra = eye(3)*anoisevar;
Ra = eye(3)*10^-8;%*100  %5;   %8; changed 3/23/2011
R = [Rq, zeros(4,3); zeros(3,4), Ra];
%% initialize values for Kalman Filter loop
Pk1k1 = P00;
```

```
Qk1 = Q;
Rk = R;
xhatk1k1 = xhat00;
abhatk1 = abhat00;
i = 1;
Xhat(:,i) = xhatk1k1;
%%
N = size(qsim,2) - 1;
Gkhist = zeros(10,7,N-1);
for i = 1:N-1
    xhatsub = xhatk1k1;
    Psub = Pk1k1;
%% Propagation loop is divided into smaller steps to improve errors
    for m = 1:divisor
        X = xhatsub;
        q0 = X(1);
        q1 = X(2);
        q2 = X(3);
        q3 = X(4);
        w1 = X(5);
        w2 = X(6);
        w3 = X(7);
        ab1 = X(8);
        ab2 = X(9);
        ab3 = X(10);
        %% formulate linearized A matrix
        Alin0 = [...
            a*[       0, -1/2*w1, -1/2*w2, -1/2*w3,...
                  -1/2*q1, -1/2*q2, -1/2*q3,0,0,0];...
            a*[  1/2*w1,       0,  1/2*w3, -1/2*w2,...
                    1/2*q0, -1/2*q3,  1/2*q2,0,0,0];...
            a*[  1/2*w2, -1/2*w3,       0,  1/2*w1,...
                    1/2*q3,  1/2*q0, -1/2*q1,0,0,0];...
            a*[  1/2*w3,  1/2*w2, -1/2*w1,       0,...
                   -1/2*q2,  1/2*q1,  1/2*q0,0,0,0];...
            b*[       0,       0,       0,       0,...
                       0, -5/8*w3, -5/8*w2,0,0,0];...
            b*[       0,       0,       0,       0,...
                  5/8*w3,       0,  5/8*w1,0,0,0];...
            b*[       0,       0,       0,       0,...
                         0,       0,       0,0,0,0];...
            zeros(3,10)];
        Alind = eye(10) + dT*Alin0 + (dT*Alin0)^2/2;
        Ad = Alind;
        Hcoeff = 1;
        Hcoeff2 = 1;
%         Hcoeff = 0.5;
%         Hcoeff2 = 0.5;
        Haw = [[    13/8*w3*rc3*Hcoeff+rc2*w2*Hcoeff,...
              -2*w2*rc1*Hcoeff2+rc2*w1*Hcoeff,...
                13/8*rc3*w1*Hcoeff-2*rc1*w3*Hcoeff2];
            [       w2*rc1*Hcoeff-2*rc2*w1*Hcoeff2,...
                13/8*w3*rc3*Hcoeff+w1*rc1*Hcoeff,...
                  13/8*rc3*w2*Hcoeff-2*rc2*w3*Hcoeff2];
            [   3/8*rc1*w3*Hcoeff-2*rc3*w1*Hcoeff2,...
               3/8*rc2*w3*Hcoeff-2*rc3*w2*Hcoeff2,...
                3/8*w1*rc1*Hcoeff+3/8*rc2*w2*Hcoeff]];
        Hq = eye(4);
```

```
        Hab = eye(3);
        Ck = [Hq, zeros(4,6); zeros(3,4), Haw, Hab];
        Ak1 = Ad;
        Psub = Ak1*Psub*Ak1' + Qk1/divisor;
%         xhatsub = Ak1*xhatsub;
        xhatsub = func_f(xhatsub,J,dT);
    end
%% back to measurement update portion of ekf loop
    xhatkk1 = xhatsub;
    Pkk1 = Psub;
    Gk = Pkk1*Ck'/(Ck*Pkk1*Ck' + Rk);
    Gkhist(:,:,i) = Gk;
    Pkk = (eye(10) - Gk*Ck)*Pkk1;
%     rhok = [q_m(1:4,i+1);a_m(:,i+1)] - Ck*xhatkk1;
    rhok = [q_m(1:4,i+1);a_m(:,i+1)] - func_h(xhatkk1,Jhat,r);
    rhokhist(:,i) = rhok;
    xhatkk = xhatkk1 + Gk*rhok;
    updatehist(:,i) = Gk*rhok;
    Xhat(:,i+1) = xhatkk;
    % added a plus one to the estimator storage
    X = xhatkk;
    xhatk1k1 = xhatkk;
    Pk1k1 = Pkk;
    qhat = xhatk1k1(1:4);
    what = xhatk1k1(5:7);
    %formulate ahat from w
% %     wdothat = J\fcross(J*what)*what;
% %     ahat(:,i+1) = fcross(wdothat)*r + fcross(what)^2*r;
end
```

5 run_hinf26_propR.m

```
%% ekf01 - estimate spin rate from just quaternions
dT = state_rate/divisor;
a = 1;
b = 1;
% a = 1/2;
% b = 1/2;
rc1 = rAi(1);
rc2 = rAi(2);
rc3 = rAi(3);
r = rAi;
%% Define P0 x0 Q0 and R0
P00 = eye(10);
abhat00 = zeros(3,1);
xhat00 = [0;0;0;0;0;0;0.31;abhat00];
Qq = eye(4)*10^-9% *5;   % 11
Qa = eye(3)*10^-10  %9 for J unknown; 11 for known
Qab = eye(3)*10^-20;
Q = [Qq, zeros(4,6); zeros(3,4), Qa, zeros(3,3); zeros(3,7), Qab];
% Q = diag([Q_what,10^-20,10^-20,10^-20]);
Rq = [2,0,0,0;0,2,0,0;0,0,13,0;0,0,0,13]*10^-9;
% Rq = eye(4)*10^-11;%*100;   %15
Ra = eye(3)*anoisevar;
Ra = eye(3)*10^-8;%*100  %5;   %8; changed 3/23/2011
```

```
R = [Rq, zeros(4,3); zeros(3,4), Ra];
Lk = eye(10);
% Sk = eye(10);
Sk = [eye(4),zeros(4,6);...
    zeros(3,4),10*[1,0,0;0,1,0;0,0,1],zeros(3);...
    zeros(3,7),eye(3)];
sigma = 0*50000;
sigma = 1000000;
% sigma = 0;
Sbar = Lk'*Sk*Lk;
%% initialize values for Kalman Filter loop
Pk1k1 = P00;
Qk1 = Q;
Rk = R;
xhatk1k1 = xhat00;
abhatk1 = abhat00;
i = 1;
Xhat(:,i) = xhatk1k1;
%%
N = size(qsim,2) - 1;
for i = 1:N-1
    xhatsub = xhatk1k1;
    Psub = Pk1k1;
%% Propagation loop is divided into smaller steps to improve errors
    for m = 1:divisor
        X = xhatsub;
        q0 = X(1);
        q1 = X(2);
        q2 = X(3);
        q3 = X(4);
        w1 = X(5);
        w2 = X(6);
        w3 = X(7);
        ab1 = X(8);
        ab2 = X(9);
        ab3 = X(10);
        %% formulate linearized A matrix
        Alin0 = [...
            a*[       0, -1/2*w1, -1/2*w2, -1/2*w3,...
                 -1/2*q1, -1/2*q2, -1/2*q3,0,0,0];...
            a*[  1/2*w1,       0,  1/2*w3, -1/2*w2,...
                   1/2*q0, -1/2*q3,  1/2*q2,0,0,0];...
            a*[  1/2*w2, -1/2*w3,       0,  1/2*w1,...
                   1/2*q3,  1/2*q0, -1/2*q1,0,0,0];...
            a*[  1/2*w3,  1/2*w2, -1/2*w1,       0,...
                  -1/2*q2,  1/2*q1,  1/2*q0,0,0,0];...
            b*[       0,       0,       0,       0,...
                        0, -5/8*w3, -5/8*w2,0,0,0];...
            b*[       0,       0,       0,       0,...
                  5/8*w3,       0,  5/8*w1,0,0,0];...
            b*[       0,       0,       0,       0,...
                    0,       0,       0,0,0,0];...
            zeros(3,10)];
        Alind = eye(10) + dT*Alin0 + (dT*Alin0)^2/2;
        Ad = Alind;
        Hcoeff = 1;
        Hcoeff2 = 1;
```

```
        Haw = [[    13/8*w3*rc3*Hcoeff+rc2*w2*Hcoeff,...
              -2*w2*rc1*Hcoeff2+rc2*w1*Hcoeff,...
                13/8*rc3*w1*Hcoeff-2*rc1*w3*Hcoeff2];
            [       w2*rc1*Hcoeff-2*rc2*w1*Hcoeff2,...
                13/8*w3*rc3*Hcoeff+w1*rc1*Hcoeff,...
                  13/8*rc3*w2*Hcoeff-2*rc2*w3*Hcoeff2];
            [   3/8*rc1*w3*Hcoeff-2*rc3*w1*Hcoeff2,...
               3/8*rc2*w3*Hcoeff-2*rc3*w2*Hcoeff2,...
                3/8*w1*rc1*Hcoeff+3/8*rc2*w2*Hcoeff]];
        Hq = eye(4);
        Hab = eye(3);
        Ck = [Hq, zeros(4,6); zeros(3,4), Haw, Hab];
        Ak1 = Ad;
        Psub = Ak1*Psub*Ak1' + Qk1/divisor;
%         xhatsub = Ak1*xhatsub;
        xhatsub = func_f(xhatsub,J,dT);
    end
%% back to measurement update portion of ekf loop
    xhatkk1 = xhatsub;
    Pkk1 = Psub;
    Pkk = Pkk1/(eye(10) + sigma*Sbar*Pkk1 + Ck'/Rk*Ck*Pkk1);
    Gk = Pkk*Ck'/Rk;
    Gkhist(:,:,i) = Gk;
    hinftest(:,i) = eig(Pkk^-1 - sigma*Sbar + Ck'/Rk*Ck);
    hinftestlogic(:,i) = hinftest(:,i) > zeros(10,1);
    rhok = [q_m(1:4,i+1);a_m(:,i+1)] - func_h(xhatkk1,Jhat,r);
    rhokhist(:,i) = rhok;
    xhatkk = xhatkk1 + Gk*rhok;
    Xhat(:,i+1) = xhatkk;
    % added a plus one to the estimator storage
    X = xhatkk;
    xhatk1k1 = xhatkk;
    Pk1k1 = Pkk;
    qhat = xhatk1k1(1:4);
    what = xhatk1k1(5:7);
    %formulate ahat from w
    wdothat = J\fcross(J*what)*what;
    ahat(:,i+1) = fcross(wdothat)*r + fcross(what)^2*r;
end
%%
 figure(111), set(111,'Position',[100, 100, 650, 200]),...
            plot(hinftestlogic')
 title('H_{\infty} Test Condition')
 xlabel('Sample # (4Hz)')
 if printplots == 1
    print -depsc -tiff images\hinftestcondition
    print -dpdf images\hinftestcondition
end
```

6 run_avgekf43_smo05.m

```
%% ekf01 - estimate spin rate from just quaternions
dT = state_rate/divisor;
a = 1;
b = 1;
```

```
% a = 1/2;
% b = 1/2;
rc1 = rAi(1);
rc2 = rAi(2);
rc3 = rAi(3);
r = rAi;
%% Define P0 x0 Q0 and R0
P00 = eye(10);
abhat00 = zeros(3,1);
xhat00 = [0;0;0;0;0;0;0.31;abhat00];
xhatk1k1 = xhat00;
abhatk1 = abhat00;
i = 1;
Xhat(:,i) = xhatk1k1;
%%
N = size(qsim,2) - 1;
Gk = [[0.6755   -0.0103   -0.0004   -0.0000...
    0.0000    0.0000    0.0000];...
  [ -0.0103    0.7257   -0.0001    0.0002...
     -0.0000   -0.0000   -0.0000];...
  [ -0.0004   -0.0001    0.7254    0.0110...
      0.0000   -0.0000    0.0000];...
  [ -0.0000    0.0002    0.0110    0.6755...
      0.0000    0.0000    0.0000];...
  [ -0.0004    0.0016   -0.0088   -0.0019...
      0.0003   -0.0000    0.0001];...
  [ -0.0019    0.0087    0.0016    0.0004...
     -0.0000    0.0003    0.0001];...
  [  0.0086    0.0018   -0.0005    0.0020...
     -0.0004   -0.0004    0.0000];...
  [  0.0133   -0.0010    0.0053    0.0089...
      0.0088   -0.0002   -0.0000];...
  [  0.0140   -0.0042   -0.0052    0.0067...
     -0.0002    0.0088   -0.0000];...
  [  0.0008   -0.0037    0.0011    0.0003...
     -0.0000   -0.0000    0.0091]];
SKwq = [0.25,-1,-1,0.25;...
    0.25,1,-1,-0.25;...
    1,-0.25,0.25,-1]*250*(1/250);
SKqq = eye(4);
SKab = zeros(3,4);
SK = 0.1*[SKqq;SKwq;SKab]*1;          % added *0.1 for known J
alpha = 1;
for i = 1:N-1
    xhatsub = xhatk1k1;
%     Psub = Pk1k1;
%% Propagation loop is divided into smaller steps to improve errors
    for m = 1:divisor
        X = xhatsub;
        q0 = X(1);
        q1 = X(2);
        q2 = X(3);
        q3 = X(4);
        w1 = X(5);
        w2 = X(6);
        w3 = X(7);
        ab1 = X(8);
```

```
        ab2 = X(9);
        ab3 = X(10);
        %% formulate linearized A matrix
        xhatsub = func_f(xhatsub,J,dT);
    end
%% back to measurement update portion of ekf loop
    xhatkk1 = xhatsub;
  SKwq = -1*[1*[q1,-q0,-q3,q2];...
    1*[q2,q3,-q0,-q1];...
    5*[q3,-q2,q1,-q0]]*250*0.75;
    SK = 0.075*[0.125*SKqq;SKwq;SKab];   % added 0.1* for known J
    cutoff = 5*10^-7;  %  7 works well;
    rhok = [q_m(1:4,i+1);a_m(:,i+1)] - func_h(xhatkk1,Jhat,r);
%     Surf = rhok(1:4);
%     Surf = sign(rhok(1:4))*10^-7*8;
    rhok_1 = rhok;
    rhok = alpha*rhok;
    for rhoi = 1:size(rhok,1)
        if abs(rhok(rhoi)) > cutoff
            rhok(rhoi) = cutoff*sign(rhok(rhoi));
        end
    end
    Surf = rhok(1:4);
%     SK = Gk;
%     Surf = rhok(1:4).^3;
    Surfhist(:,i) = Surf;
    xhatkk = xhatkk1 + Gk*rhok_1 + SK*Surf;   %*0.002;
    Xhat(:,i+1) = xhatkk;
    % added a plus one to the estimator storage
    X = xhatkk;
    xhatk1k1 = xhatkk;
    qhat = xhatk1k1(1:4);
    what = xhatk1k1(5:7);
end
```