OPTIMAL THRUSTER SELECTION WITH ROBUST ESTIMATION FOR FORMATION FLYING APPLICATIONS

By

Trent Yang

B.S. Aerospace Engineering
University of Colorado – Boulder, 2000

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2003

Signature of Author: _____
Department of Aeronautics and Astronautics
May 21, 2003

Certified by: _____
Jonathan How
Associate Professor of Aeronautics and Astronautics
Thesis Advisor

Certified by: _____
Bruce A. Persson
Charles Stark Draper Laboratory
Thesis Supervisor

Accepted by: _____
Edward M. Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Abstract

OPTIMAL THRUSTER SELECTION WITH ROBUST ESTIMATION FOR FORMATION FLYING APPLICATIONS

By

Trent Yang

Submitted to the Department of Aeronautics and Astronautics on May 21, 2003 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Aeronautics and Astronautics

The research goal was to develop a computationally fast mapper that can be easily configured to any spacecraft with various types of actuators. The estimation process must be compatible to the mapper and have a fast yet robust fault detection algorithm. A robust fault detection system must be sensitive to failures without raising any false alarms.

The linear program (LP) mapping system presented in this thesis was first introduced by Crawford in 1968 (Ref. 42) and has been since used on a number of vehicle control systems (Ref. 9, Ref. 41). In this paper, several new innovations are developed as extensions to the basic LP. First, a solution scheme is presented to handle thruster performance degradation due to fuel flow loss from multiple thruster usages. Second, new techniques for solving linear programming problems with uncertain data are explored. In particular, a robust LP (RLP) formulation is developed here to deal with uncertainty in either the thruster performance or the velocity and positional measurements.

The estimation process is a combination of a Kalman filter and a Generalized Likelihood Ratio (GLR) test for actuator fault detection. A new model-comparison (MC) approach is introduced in conjunction with the GLR test to quickly and reliably determine the exact nature of the failure, once a malfunction has been detected.

Finally, the simulations of the estimator and thrust mapper system are performed on the SPHERES and Orion formation flying test beds and results show improved control capabilities along with significant fuel saving.

Technical Supervisor: Bruce Persson,
Title: Senior Member of the Technical Staff at Draper Laboratory

Thesis Advisor: Jonathan How
Title: Associate Professor of Aeronautics and Astronautics

Intentionally Left Blank

Acknowledgement

5/20/2003

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Individual Research and Development funding, Project 3028.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas

Trent Yang 20 May 2003

I would also like to take this opportunity to thank all the people who have made this an enjoyable experience and helped me learn, grow and enjoy life during my time here in the MIT aerospace department. In fact, I have enjoyed this so thoroughly that I have decided to write a second thesis for my Technology and Policy degree (coming to a library near you!).

Andy, John, Will and Mike – Thanks for being such great roommates and friends. It is comforting to know that I could always count on you guys for being there and not burning down the house!

Kacey – You are my best friend and thanks for being there always!

Josh, Andy, Tony, Rob, Phil, Mike, and Arthur – thanks for all the support and help. I enjoyed working with each and every one of you.

And to all my friends in and around MIT and Boston area – its been great fun, I hope you are not reading through this for fun ☺

Of course, I have to give thanks to my advisors Jon How at MIT and Bruce Persson at Draper. Without your insightful knowledge and willingness to help, none of this would have happened. I truly appreciate having the opportunity to work and learn from you in those two years. Furthermore, I would like to thank Draper for their financial support and having me as a Draper fellow for two years.

Finally, and most importantly, I would like to thank my parents for all their love, support, guidance, trust and encouragements throughout my life. Nothing I can say here can convey my love and respect for you.

Intentionally Left Blank

Table of Contents

Abstract 3

Acknowledgement 5

Assignment 6

Table of Contents 7

List of Figures 10

List of Tables 12

Chapter 1 Introduction 13

1.1 Move towards multiple autonomous satellites 13

1.2 Relevance of this research to autonomous satellites 15

1.3 Background and Previous Research 16

1.4 Contribution of this Thesis 17

1.4.1 Thruster Mapper 17

1.4.2 Estimation 19

1.5 Laboratory Demonstration of Algorithms 20

1.6 Thesis Organization 20

Chapter 2 Thruster Mapper 23

2.1 Introduction 23

2.2 Problem Formulation 24

2.2.1 Linear Program Formulation 25

2.3 Special Tailored Simplex Algorithm for Spacecraft 27

2.3.1 Testing on SPHERES 33

2.4 Thruster Mapping with Degradation 34

2.4.1 Iterative Method I 35

2.4.2 Iterative Method II 39

2.4.3 MILP algorithm for degradation 42

2.4.4 Performance of Degradation Algorithms 43

Chapter 3 Robust Linear Programming For Uncertainties in the Thruster Mapping Problem 49

3.1 Introduction 49

3.2 Uncertainty in *b* 50

3.2.1 Symmetric Errors in b 53

3.3 Uncertainty in A 56

3.4 Other Ways of Dealing with Uncertainties in A and b 62

3.5 Conclusions 63

Chapter 4 Estimation 65

4.1 Introduction 65

4.2 Kalman Filter and Estimation 66

4.2.1 State Dynamics 66

4.2.2 Propagation of Statistics and Measurement Incorporation 67

4.2.3 Kalman Filter Summary Equations 70

4.2.4 Filter Simplifications 71

Chapter 5 Fault Detection and Re-Organization 73

5.1 Introduction 73

5.2 Fault Detection and Isolation 76

5.2.1 Problem Definition 76

5.2.2 Simple Chi-Squared Tests 80

5.2.3 Generalized Likelihood Ratio (GLR) 84

5.2.4 Comparison of Fault Detection Tests 86

5.3 Fault Estimation and Compensation 92

5.3.1 A Model-Comparison Approach 93

Chapter 6 Simulation on the ORION Testbed 97

6.1 ORION Mission Background 97

6.2 Hill's Equations 99

6.3 Thrust Mapper and Degradation for Orion 101

6.4 Estimation and Fault Detection 105

6.4.1 Orion Spacecraft Operation Details 106

6.4.2 States of Orion 107

6.4.3 Simulation Results 110

6.4.4 Summary of Estimation and Fault Detection Equations and Variables for Orion 116

Chapter 7 Conclusions 119

APPENDIX A 121

APPENDIX B 129

APPENDIX C 135

APPENDIX D 137

APPENDIX E 139

References 145

List of Figures

Fig. 1 Block diagram of how the thruster mapper and estimation fits into the overall picture.....15
Fig. 2 Detailed picture of the estimation block 19
Fig. 3 Control strategy of having a "control module" and a "thrust mapper". The "control module" is only responsible for determining the optimal accelerations at each step while the thrust mapper finds the on-times for the available actuators that meets the desired accelerations. 23
Fig. 4 Thruster force degradation due to multiple on jets.....34
Fig. 5 Iterative method 1 for dealing with multiple thruster on degradation 36
Fig. 6 Diagram of Iterative method II.....40
Fig. 7 Average error in achieving desired acceleration due to degradation.....44
Fig. 8 Time comparison of the two iterative loops vs. no iteration 45
Fig. 9 Number of iterations before convergence using the different methods 46
Fig. 10 Average MILP algorithm solution time compared to iterative algorithms.....47
Fig. 11 Estimation and Fault Detection 65
Fig. 12 Implementation of known degradation model 77
Fig. 13 One possibility of detecting long-term degradations in actuator system that are hard to model 78
Fig. 14 Detection vs. False Alarm Ratios for different measurement and actuator firing standard deviations.....88
Fig. 15 Detection vs. False Alarm Ratios for different Detection Levels and R,Q values in the Kalman Filter.....88
Fig. 16 Time to Detection and Isolation of failure for different measurement and actuator uncertainties 90
Fig. 17 Time to Detection and Isolation for different Detection Levels and R,Q values in the Kalman Filter.....90
Fig. 18 Multiple-model failure estimation.....94
Fig. 19 Orion Formation Flying Mission 97
Fig. 20 Sample Optimal Trajectory.....98
Fig. 21 Hill's (LVLH) reference frame.....99
Fig. 22. Actual vs. design trajectory + lost acceleration due to multiple thruster firings 103
Fig. 23. Thruster Usage with Degradation Mapper Algorithm and resulting trajectory 104

Fig. 24. Orion operations procedure for each time period 106
Fig. 25. Percentage of false alarms as a function of GLR threshold for Orion 108
Fig. 26. Detection probability for a 50% degradation at GLR threshold = 0.00008. Note: It is impossible to have the case of a failed thruster on-time of over 1 second and occupying less than 8% of total on-times. 109
Fig. 27. Detection rate for a single thruster full-on failure at GLR threshold = 0.00008 110
Fig. 28. Actual vs. designed trajectory with one thruster 50% failure in the radial direction 111
Fig. 29. Actual vs. designed trajectory with feedback control when actual trajectory deviates 25 m from planned path. 112
Fig. 30. Expected vs. Measured Velocity Changes from Actuator Firings with Thruster 1 failure (complete off). 113
Fig. 31. GLR values for each thruster after failure to thruster one. 114
Fig. 32. Actual vs. designed trajectory 115
Fig. 33. Actual vs. designed trajectory with fault detection algorithm implemented 115
Fig. 34. Fuel savings from using fault detection to guarantee successful mission completion ... 116

List of Tables

Table 1 Example of a matrix (A) relating the effecting of each actuator on the spacecraft movement 26

Table 2, Linear Programming Algorithm Speed Comparison. * processor used on the ORION spacecraft mission is a 200Mhz StrongARM processor (Pentium class). 32

Table 3, Mapping algorithm comparison between LP method vs. SPHERES mapper. Avg. % error is defined as: error = (actual d_v – expected d_v)/(expected d_v) 33

Table 4, degradation convergence example using differential convergence. Note: this example uses the ORION spacecraft's thruster alignment and degradation factors. 38

Table 5, degradation convergence example using differential convergence. Note: this example uses the ORION spacecraft's thruster alignment and degradation factors. 41

Table 6, Percentage of correct solutions for different types of variations in *A*. Note: the values of the A and b matrices in all simulations were completely random to give an idea of the general conservatism of the algorithms 61

Table 7, Decisions errors for fault detection 74

Table 8 Example of two actuators that cannot be distinguished using the set chi-squared method 83

Table 9 average computation times for various detection algorithms 91

Table 10 Orion's Thruster Mapper 101

Chapter 1
Introduction

1.1 Move towards multiple autonomous satellites

Beginning in the early 1990s, NASA (National Aeronautics and Space Administration) started the concept of smaller, faster and cheaper satellites for future space satellite missions. Instead of the old monolithic space vehicles that carried multiple instruments for various scientific needs, NASA envisioned a fleet of smaller satellites having dedicated capabilities onboard to autonomously perform individual tasks while working in concert with the other satellites. Several reasons made this approach to future space missions appealing. First, large monolithic satellites usually take a long time to build and test due to different lead times on all the instruments and electronics involved. When the entire satellite is finally assembled and readied for flight, many of the technologies onboard could be outdated and inefficient. Furthermore, there is extreme pressure for the launch to be successful. A single failure would mean jeopardizing not only a billion dollar project but also many years of labor from thousands of scientists and engineers.

By using multiple satellites, many of the above problems are alleviated. Since not all instruments on a mission need to be available at the same time, the most urgent ones can be built first and made operational while they wait for the rest to be made and launched. Each satellite, by shortening its manufacturing time, can also be outfitted with the latest electronic and computer technologies to guarantee performance and reliability. By spreading the instruments across various satellites and using different launch times, one could also reduce the risk of complete mission failure due to single satellite or launch failures. Even if one or more of the individual satellites failed, the overall mission could still continue,

albeit at a reduced capacity. This is a much better scenario than losing an entire mission from one single failure on a monolithic satellite.

While the concept of using multiple satellites is good, there are many barriers to be crossed and technologies validated before we can realize the full potentials and benefits of such a vision [Ref. 39]. One of the main challenges in the realization of fully autonomous space vehicles is in optimizing the control architecture of these micro-satellites. Humans currently perform almost all mission planning and spacecraft diagnostics remotely from mission control centers. In order to reduce the possibility of increased human errors from working with multiple satellites, the spacecraft need to be fully autonomous to plan and execute their own activities in accordance to high-level goals stated by ground controllers. Rather than have humans do the detailed planning necessary to carry out the desired tasks, the "smart" satellites will formulate their own plans, using high-level goals provided by the operations team. Each spacecraft will then devise their own plan by combining those goals with its detailed knowledge of both the condition of the spacecraft and how to control it. It then executes that plan, constantly monitoring its progress. By transferring the normal human functions of mission operation and monitoring to artificial intelligence, a spacecraft would be more agile to respond to unexpected circumstances such as sudden failures or malfunctions. Furthermore, decreasing human operational responsibilities should also significantly decrease operating costs, which is always a large portion of any space mission budget [Ref. 40]. Also, for future deep space missions, it is imperative that most mission operation and diagnostics to be performed onboard to ensure real-time planning and maneuvering without taking up valuable resources on the Deep Space Network.

1.2 Relevance of this research to autonomous satellites

In this Thesis, the control algorithms developed are part of a modular architecture where each function can be easily replaced without affecting other tasks (see Fig. 1).

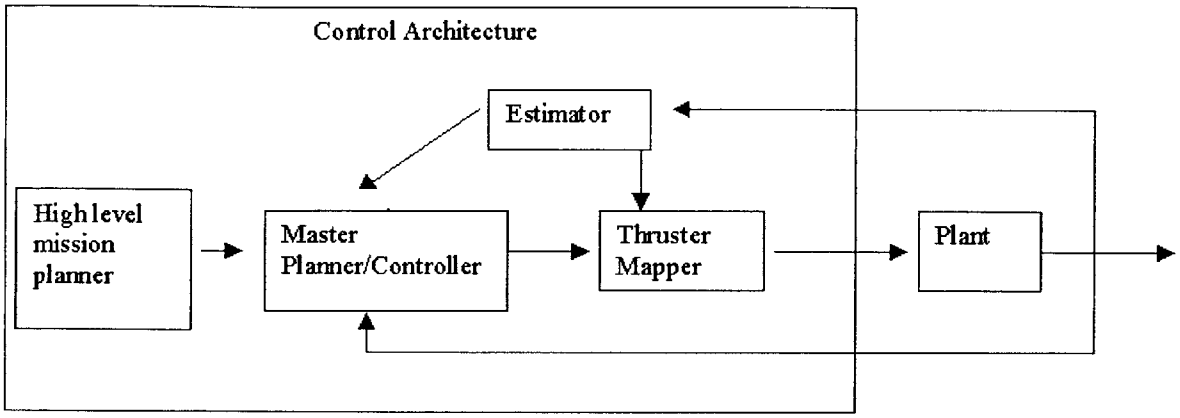


Fig. 1 Block diagram of how the thruster mapper and estimation fits into the overall picture

A modular control scheme allows individual functions to have very specific and defined tasks that can be specifically tailored to multiple spacecrafts. By dividing the main control task into several pieces, it is also easier to optimize each function to a specific need.

Each function in the control architecture serves an important role that would be critical to the autonomous control of any spacecraft. The high-level mission planner is designed to translate human-mandated goals to specific targets that the spacecraft can achieve (i.e. rendezvous at location x,y,z) [Ref. 2]. Next, the spacecraft optimizes the best control/trajectory to achieve the goal under certain requirements such as minimizing fuel or time [Ref.1]. Once the mission planning is complete, then a resource allocation tool (thruster mapper) is required to determine which actuators to use for different acceleration needs [Ref. 41]. Along with this thruster mapper system, an estimation algorithm is needed to monitor the performance of all actuators. This will enable failures to be automatically detected and the updated actuator performance data can be fed

back to both the mapper and mission planning systems. This thesis focuses on the last two functions of the autonomous control architecture – the thruster mapper and estimation algorithms.

1.3 Background and Previous Research

In recent years, there has been a large body of research that have focused on the various parts of the modularized control architecture for autonomous spacecraft. Although most of the effort has been focused on the high-level planning and control algorithms [Ref.1, Ref. 2, Ref. 14, Ref. 34, Ref. 35], there has been some research into a mapping and estimation/fault detection system [Ref. 23, Ref. 24, Ref. 25, Ref. 41].

Different ways of approaching the mapping function have been proposed and shown to work. Some researchers [Ref. 24] have used a neural network approach. Although this process is fairly complicated and requires heavy computing power, it was argued that optimizing the control of on-off spacecraft thrusters is a nonlinear constrained problem that is extremely hard to solve [Ref. 24]. But, as shown recently by Buffington [Ref. 41] and previously by Crawford and Bergman [Ref. 3 and Ref. 42], the mapping problem can be solved through a Linear Program (LP) that is both efficient and scalable to any number and type of control surfaces. The LP technique has also been utilized and applied with great success by Draper Laboratory in the Space Shuttle flights [Ref. 3].

Estimation and fault detection has long been an important part of any control theory. The most common and widely used optimal filtering technique since the 1960s has been the Kalman filter, which is based on the theories of N. Weiner [Ref. 15]. The Kalman filter allows recursive processing of noisy measurement data which makes it readily implementable for most real-time, multiple-input/multiple-output applications. Previous research have used the

Kalman filter with great success in estimating various properties of a spacecraft including mass, and thruster performance (Ref. 17).

1.4 Contribution of this Thesis

The purpose of this research is to build upon previous knowledge of mapping and estimation techniques and find optimization methods that can be applied to different types of micro-satellites. The desired characteristics of the thruster mapper and estimation systems is that they must be generic, computationally fast and robust. The algorithms must be generic in the sense that they are compatible with the high-level control algorithms and the actuators typically found in small autonomous spacecraft. They also need to be robust to various actuator failures that may occur and adapt accordingly. Finally, the algorithms must be small enough to operate in real-time on a micro-satellite that typically has limited computation capability and memory space.

For both the thruster mapper and the estimation process, several appropriate techniques are considered and their strengths and weaknesses are analyzed. One complete model of the mapper and estimation modules is then applied to the Orion formation-flying mission [Ref. 8]. The mission provides an excellent test scenario where the micro-satellites not only have to operate autonomously but also have many constraints not usually seen on larger, more expensive spacecraft. These constraints include noisier measurement systems, less redundancy in system components, minimum amount of fuel and tougher control requirements for new missions such as formation flying and station keeping.

1.4.1 Thruster Mapper

The thruster mapper approach used in this thesis is based on a linear programming (LP) technique that has been previously investigated in Ref. 3 and

Ref. 41. The LP formulation is quite simple and can be adapted to accommodate various types of actuators. Due to its high computation and memory requirements, the neural network approach will not be considered here.

Several new innovations to the LP process are developed in this Thesis. A special tailored simplex method is first devised to speed up the computation time for the linear program. Other contributions to this research area include dealing with degradation effects from multiple thrusters firing at once and uncertainty in the parameters of the linear program, namely the actuator performance and desired acceleration.

Degradation can result from multiple thruster firings due to limited pressure levels in the propulsion system. Testing of a propulsion system developed for the Orion micro-satellite [Ref. 5] has shown non-trivial degradation effects that depend on the design of the mechanical system and the mode of operation.

Uncertainty exists in the models of all physical systems. The purpose of a robust algorithm is to find solutions that are good for all possible situations. In the thruster mapper system, uncertainty can arise in two areas – the actuator performance level and the desired acceleration. Uncertainty in the propulsion system can be tested on the ground prior to flight. On the other hand, the desired acceleration level is usually a function of the onboard measurement instruments' sensitivity. For example, if the high-level trajectory planner is working in the Local Vertical Local Horizontal (LVLH) frame, then the thruster mapper would have to convert the required acceleration into the body frame using current attitude information. This extra coordinate transformation is necessary because the actuators' performance levels are stored in the body-frame. Any errors from onboard sensors used for attitude measurements will directly translate into uncertainty for the desired acceleration level in the body-frame. A robust formulation of the LP thrust mapper is used to handle these uncertainties and the results are compared to the non-robust solutions.

1.4.2 Estimation

The estimator for this system (shown in Fig. 2) performs three tasks. First, it runs a real-time filter to track the performance of each actuator. Residuals, calculated from the difference between the expected and measured accelerations, are used in the detection algorithm to quickly determine any unexpected failures and isolate the faulty actuator. When a failure is detected, the identification algorithm is responsible for determining the exact level of failure for the actuator in question and updating the new actuator performance levels. That information is then passed onto both the planner and mapper algorithms.

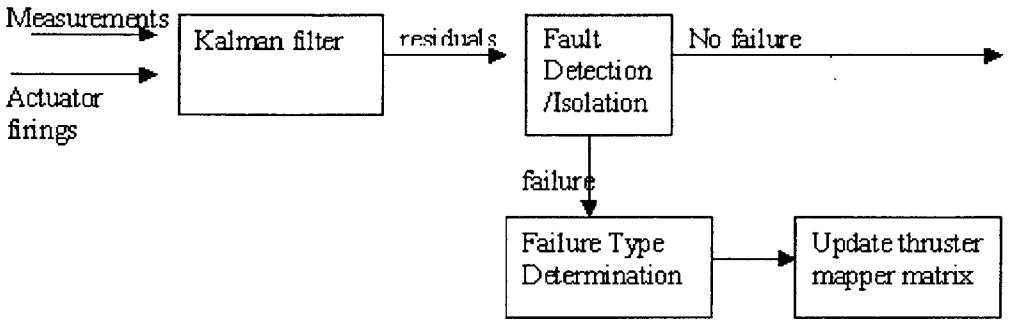


Fig. 2 Detailed picture of the estimation block

The estimation filter is based on a static Kalman filter with small changes designed to reduce computation time and incorporate any degradations due to multiple thruster firings. Three different failure detection algorithms are analyzed in this thesis. The Chi squared (χ^2) and General Likelihood Ratio (GLR) tests use the innovation (residual) values from the Kalman filter while the threshold detection method checks the estimated values against a set of predetermined values for fault detection. All three approaches are analyzed and compared in terms of total computation time, probability of detecting a failure, and number of false alarms.

1.5 Laboratory Demonstration of Algorithms

The Orion formation-flying mission [Ref. 6] provides an excellent opportunity to test out the algorithms on a fully autonomous multiple spacecraft mission. Unfortunately, any real data will have to wait until its launch in 2003 (one year after the completion of this thesis). However, two laboratory testbeds are available that can be used to simulate critical aspects of the mission such as satellite communication, real computational limits, and different high-level control structures.

The SPHERES testbed consists of miniature satellites that can be tested either in 1-g laboratory environments or on NASA's reduced gravity KC-135 flights [Ref. 7]. The purpose of SPHERES is to test and debug various control algorithms associated with autonomous spacecraft flight. Testing on the system has shown that the thruster mapper developed in this Thesis could be seamlessly inserted into the existing control system and at the same time provide far superior actuation control than their previous system (see section 2.3.1).

The second environment involves an innovative hardware-in-the-loop testbed developed at MIT's Space Systems Lab for the specific purpose of simulating multiple autonomous spacecrafts [Ref. 8]. Estimation and fault detection procedures are tested and shown to perform as expected. Although this simulation testbed does not include any hardware except the computers, it provides a realistic interface between the software algorithms and the communication between satellites.

1.6 Thesis Organization

There are three main sections in this thesis. The first section, Chapter 2 and Chapter 3, deals with outlining the basic principals of the thruster mapper

along with the theories necessary for dealing with multiple firing degradation and uncertainty in the linear program. The second section, Chapter 4 and Chapter 5, deals with the estimation block, which includes the Kalman filter, fault detection and variable updates. Both chapters contain a theoretical section along with a validation section where the algorithms developed are tested in either of the two simulation environments described above.

Finally, the last section, Chapter 6, discusses how the two functions, mapper and estimation, work together especially in the context of the Orion mission. Various uncertainties and failures are simulated on the testbed and the performance of the spacecraft with and without the mapping and estimation process are compared and analyzed.

Chapter 2
Thruster Mapper

2.1 Introduction

The thrust-mapping task is an essential link between the main software controller and the actuators on any spacecraft that have decoupled the control from the mapping process (see figure below).

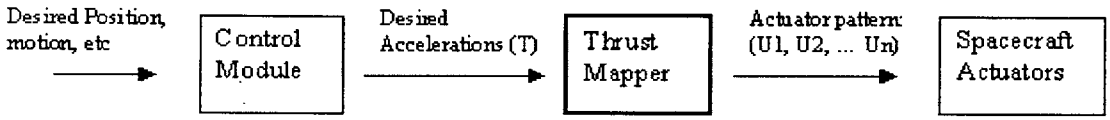


Fig. 3 Control strategy of having a "control module" and a "thrust mapper". The "control module" is only responsible for determining the optimal accelerations at each step while the thrust mapper finds the on-times for the available actuators that meets the desired accelerations.

The main controller is responsible for calculating how much acceleration or velocity change is needed in each direction to achieve a desired motion. It is then the job of the mapper to convert those requirements into specific on-off times for the actuators that are available. The actuators could vary from different types of gas-propulsion systems to momentum wheels or even simple drag plates. An effective mapping program must be able to work with all the actuator types and be fast enough to have a very short delay time between the controller and the execution of the desired accelerations.

Since most spacecraft have redundant actuators for protection against failures, there is usually more than one set of actuator on-time solutions that can satisfy the desired accelerations. A good thrust mapper design should find the solution that minimizes total fuel or actuator usage. These are especially

important parameters for any small spacecraft that can only carry a limited amount of fuel. Other considerations in the design of a thruster mapper include accounting for thrust reduction from multiple thruster firings and uncertainty in actuator output or desired motions. The thrust degradation is a result of the limitations in mass flow rate when several thrusters fire at the same time [Ref. 5]. This phenomenon can be well characterized during ground testing and tabulated ahead of time. Uncertainty arises in several different forms. One uncertainty could be the performance of the thrusters. For example, the jets on the Orion satellite have been shown to have a +/- 10% variation in thrust [Ref. 5]. The second type of error evolves from uncertainty in the desired acceleration levels as determined by the main controller. This could be due to either uncertainty in the final target or errors in the on-board measurement instruments.

2.2 Problem Formulation

Considering all the above requirements, the thruster mapper, shown in Fig. 3, must take a generic set of desired rate changes and find the output vector of discrete thruster values (on/off times) that minimizes a specified cost function. The desired forces and torques comes from the main control module (e.g. a PID controller) and is assumed to be in the body coordinate frame. If some other reference frame were used then a transformation to the body frame would be necessary.

The output of the thrust mapper is the on-times for all of the available actuators to obtain the desired acceleration or rate change, T. Each element of the on-time vector, U, is a number between zero and the maximum on time during a particular control period. For example, if the controller for a certain spacecraft is running at 0.2 Hz, then at each iteration, the thruster mapper solution could vary between zero and five seconds, the maximum on-time for that period. Alternatively, the vector of U's could also be designed to represent the

percentage on-time for the actuators. The fraction between zero and one would then represent the percent on-time for an actuator during each control period.

On most spacecraft propulsion systems, the total combined actuator effect is a linear combination of all the actuators with very little additional non-linear coupling effects from multiple actuator firings. This scenario lends itself to using a linear program (LP) for solving the problem [Ref. 3 and Ref. 4] if the effects of using each actuator are known. In some instances, the performance of a system is degraded by a coupling effect that is discussed in section 2.4. The LP solution can also be modified to handle this non-linearity.

2.2.1 Linear Program Formulation

Consider a spacecraft with four actuators. Turning on each actuator will affect the spacecraft motion in a way that depends upon the actuator strength, its location on the spacecraft and the spacecraft's mass properties. Table 1 shows an example matrix that summarizes the effects of each actuator on the motions of the spacecraft.

	Actuator 1	Actuator 2	Actuator 3	Actuator 4
x-dir (cm/sec^2)	5	-3	-2	0
y-dir (cm/sec^2)	0	2	4	-6
z-dir (cm/sec^2)	1	0.5	-2	0
θ-dir (rad/sec^2)	3.5	2	-5	0
ϕ-dir (rad/sec^2)	0	0	-3	4
φ-dir (rad/sec^2)	0	2	0	-1

Table 1 Example of a matrix (A) relating the effecting of each actuator on the spacecraft movement

Each column of the *A* matrix (Table 1) describes the spacecraft acceleration due to turning on that particular actuator. Using the above matrix, a linear program can be set up to find the on-times for all the actuators to satisfy any desired rate change, $\hat{T}$.

LP Objective: $$\text{minimize: } \begin{pmatrix} C & K \end{pmatrix} \begin{pmatrix} U \\ S \end{pmatrix} \quad (1)$$

LP Constraint: $$\hat{T} = AU + GS \quad (2)$$

LP Limits: $$0 \le U \le U_{upperbnd} \qquad S \ge 0 \quad (3)$$

A is a *m* x *n* sized actuator performance matrix, where *m* is the total degrees of freedom (normally 6) and *n* is the total number of actuators available (see Table

1 for example); U is the on-times for each actuator; $U_{upperbnd}$ is the maximum allowed time to turn on a thruster per control period, and C is the cost of using each actuator. In cases where the actuators are not able to completely satisfy the desired accelerations, a series of slack variables, S, are utilized. The slack variables should be a length of 2m x 2m for satisfying both the positive and negative directions for each axis of motion. G represents an identity matrix of size 2m x 2m and K is the cost of using the slack variables.

Preferences for using certain actuators can also be accommodated through a cost scheme for each of the thrusters. The relative cost, C_i, of using each actuator is determined by the designer's preference in using the actuators. For example, if both momentum wheels and thrusters were available, one would likely assign a higher cost to use the thrusters in order to save fuel usage. Otherwise, if all actuators on a spacecraft are equally preferred, then the cost vector, C, would have a constant and equal value for each element. Next, the cost of using the slack variables needs to be determined.

If the objective is to obtain as much of the desired forces/torques, $\hat{T}$, as possible, one would assign higher costs, K, to using the slack variables than the actuators ($K >> C$). However, there are times when saving fuel is just as important as accomplishing the desired spacecraft movements. In those circumstances, the cost ratio between the actuators and the slack variables (C/K) can be adjusted to the desired trade-off level.

2.3 Special Tailored Simplex Algorithm for Spacecraft

There are many commercial and off-the-shelf products that contain linear program solvers. But most of these solvers are written to be very generic, which tends to increase valuable computation time. A thrust mapper on any spacecraft

must be computationally fast to avoid any unnecessary delays in executing the control commands.

There are two common methods to solve any linear programming problem – the Simplex and Interior-Point methods. For the thruster mapping problem, the simplex method is selected as the base algorithm. In most spacecraft applications, there are usually redundant actuators in all degrees of motion; this leads to multiple optimal solutions that would satisfy the minimum fuel usage requirement. The Simplex solution, since it traverses the boundaries of the solution space, will always pick the minimum number of thrusters to fire. Whereas an Interior-Point algorithm may find a solution faster but would not always minimize the total number of actuators used. There are several reasons for wanting to minimize the number of actuators being used at once. First of all, one can reduce degradation resulting from multiple thruster firings if individual gas tanks serve multiple thrusters (see discussion in chapter 2.4). Second, having too many actuators working at once will slow down the estimation process that is used to determine the performance and failure of the actuators (see estimation section in Chapter 4). Third, in many future space missions, there will be many close proximity maneuvers between satellites and in some cases, jet plumes from thruster firings can negatively affect or even damage close-by satellites. By reducing the total number of jets firing, one can reduce the restrictions on close proximity maneuvering.

The simplex algorithm developed here is an extension of a Draper Laboratory algorithm [Ref. 9]. The simplex algorithm works by looping through each degree of freedom in the problem (six for most spacecraft problems) and finding the best combination of actuators to satisfy the LP constraint at the lowest cost (equation 1). There are five basic parts to a Simplex algorithm: Setup, Invite, Exclude, Decision, and Return Solution (see APPENDIX A for full algorithm). Setup determines an initial solution and is executed only once per call to Simplex. Invite finds an unused actuator to bring into the basis in order to

reduce the cost. The basis, *B*, is the set of actuators (or slack variables) that is selected to satisfy the LP constraint:

$$Bu = b$$

where *b* is the vector of desired acceleration or rate change. *B*, the basis, is an *m x m* matrix whose columns correspond to the performance levels of selected thrusters. And *u* is the *m* x 1 vector of on-times for the selected thrusters to satisfy the constraint.

Exclude makes the decision on which column of the basis should be replaced with the new incoming actuator. *Decision* determines if the replaced basic variable goes to zero or its upper bound. *Return Solution* computes the cost of the solution and determines whether or not to terminate the simplex algorithm. The process is terminated if it is determined that no columns of the basic matrix, *B*, can be replaced with a new actuator to reduce cost.

In most Simplex algorithms, the *Setup* function determines an initial sub-optimal solution to the problem. The algorithm developed here avoids that initial calculation by building the set of required slack variables into the solver. The initial slack variables and solution to the LP problem is set to the desired acceleration vector, $\hat{T}$. In summary, the LP problem under the new algorithm appends the following initial conditions to the problem.

LP initial solution:

$$S = \hat{T}$$
$$B = \mathrm{diag}(sign(\hat{T})) \tag{4}$$

Where $\hat{T}$ is the vector of desired acceleration and the rest of the formulation is kept the same:

LP Constraint:

$$\hat{T} = AU + GS \tag{5}$$

LP Objective:

$$\text{minimize: } \begin{pmatrix} C & K \end{pmatrix} \begin{pmatrix} U \\ S \end{pmatrix} \tag{6}$$

LP Limits:

$$\begin{aligned} 0 \le U \le U_{upperbnd} \\ S \ge 0 \end{aligned} \tag{7}$$

The second difference is how the algorithm stores and updates the cost of inviting a non-basic actuator to replace a column in the basis, *B*. In most simplex solvers including the Draper algorithm [Ref. 9], an inverse basis matrix is utilized in evaluating which non-basic actuator to invite. In this algorithm, the relatively computational-heavy inverse matrix is replaced with a linear combination matrix. The variables and decision process involved are as follows:

The linear combination matrix, *Y (mxn)*, contains n m-dimensional coefficient vectors which relate the actuators to the basis. It is initialized to:

Initialization of variable Y:

$$Y = B^{-1}A \tag{8}$$

Where *A* is the actuator performance matrix and $\text{diag}(sign(T))$ is the initial basis (see equation 4). Next, we find the appropriate actuator to bring into the basis that would have the greatest impact on the total cost. The *z*-vector contains the cost change of bringing in each actuator that's not already in the basis and is initialized to:

Initialization of jet evaluators:

$$\begin{aligned} z &= kY - C \\ k &= K_{1xm} \end{aligned} \tag{9}$$

Where *k* is a 1xm vector of Ks, the cost associated with using the slack variables, and *C* is the vector of costs of using the available actuators. The maximum element that is greater than zero in the vector *z* is the invited actuator:

Search for max. evaluator:

$$[z_{\max}, i_{\max}] = \max(z > 0) \tag{10}$$

Where i_{max} is the index of the invited actuator that corresponds to the maximum z-element, z_{max}.

After determining which basic column to exclude (see APPENDIX A), the Y matrix and z vector are updated as follows:

$$
\begin{aligned}
Y_{jex,i} &= \frac{Y_{jex,i}}{Y_{jex,i_{\max}}} \forall i = 1...n \\
Y_{j,i} &= Y_{j,i} - \frac{Y_{jex,i}}{Y_{jex,i_{\max}}} Y_{j,i_{\max}} \forall i = 1..n; j = 1..m; j \neq jex \\
z_i &= z_i - \frac{Y_{jex,i}}{Y_{jex,i_{\max}}} z_{i_{\max}}
\end{aligned}
\tag{11}
$$

where the index *jex* is the basis vector to be excluded. By directly updating the linear combination matrix, Y, instead of computing a new B^{-1} matrix, significant computation time can be saved. Simulations show a 30% decrease in computation time for sample spacecraft mapper solutions.

The third difference from general conventional simplex algorithms is that this algorithm is designed to handle upper bounds on the decision variable (eqn 7). If there were no upper bound constraints, the number of non-zero decision variables in the solution will be, at most, equal to the number of equality constraints [Ref. 9]. When this upper bound is lowered, the solution may require more than m non-zero decision variables where m is the number of degrees of freedom in the problem. This is due to the fact that the actuators being selected to fulfill the desired rate change could reach the maximum on time (upper bound) before completely satisfying one of the constraint objectives. Furthermore, the linear combination coefficient matrix, Y, and the evaluator vector, z, are updated differently if either the incoming actuator or the one exiting the basis goes to its upper bound:

$$Y_{j,i} = -Y_{j,i}; \forall j = 1...m$$
$$z_i = -z_i \tag{12}$$

This operation is performed in the place of equation 11 when the incoming actuator, i = i_{max}, goes to its upper bound. If *i* is the index of the outgoing vector then equation 12 is performed in addition to equation 11. Switching the polarity signals any further changes to the usage of that thruster will involve it coming down from its upper bound instead of increasing from zero.

Finally, this algorithm also differs from the Draper algorithm in that it does not consider negative constraints on the decision variables. Since all jets in this mapping problem have a non-negativity requirement (i.e. cannot have an on-time of less than zero seconds), it is not necessary to consider negative decision variable solutions, which will save valuable storage and processing requirements. See APPENDIX A for the entire algorithm.

Testing of this algorithm was done using the system configuration of the Orion spacecraft. In all, the spacecraft configuration has twelve actuators (see APPENDIX C) and the desired accelerations are in the full 6-degrees of freedom. The computation speed of this algorithm is compared with several off-the-shelf solvers and the results are summarized in Table 2.

Algorithm Used	Avg # of flops for solution	Time equivalent on a spacecraft computer*
Algorithm in Thesis	2,200	0.0122 sec
Draper Algorithm	4,200	0.0233 sec
Matlab LP	28,000	0.35 sec

Table 2, Linear Programming Algorithm Speed Comparison. * processor used on the ORION spacecraft mission is a 200Mhz StrongARM processor (Pentium class).

2.3.1 Testing on SPHERES

The thruster mapping algorithm developed in the previous sections was implemented and tested on the SPHERES satellite (see Ref. 7 for details on the SPHERE project). Regretfully, no concrete test data can be presented due to problems with the measurement system at the time. But running this algorithm on SPHERES demonstrated two very important aspects of this process. First, it was easily compatible with the control system currently employed on SPHERES without any major changes either to the mapping algorithm or the SPHERES control architecture. Second, even though SPHERES ran their control loops at 20 Hz, the simplex algorithm was able to complete all calculations within that extremely short time period. This was an extremely important point because in order to save computation time, the SPHERES control team had implemented a very crude mapping algorithm where at each iteration, it just turned on the thrusters that are pointed in the direction of the desired motion. This crude algorithm was used instead of a more exact solution due to its short computation time and since the controller was running at such a fast speed it was not absolutely necessary to obtain the exact accelerations at each step. But, the simplex algorithm proved that not only was it computationally efficient in calculating the solution but gave a much better solution than the original method. Table 3 shows performance of both mappers in terms of the average percentage deviation from the desired control command for each control iteration.

Mapping method	Avg. % error
LP Mapper in Thesis	0
SPHERES' crude mapper	15.18

Table 3, Mapping algorithm comparison between LP method vs. SPHERES mapper. Avg. % error is defined as: error = (actual d_v – expected d_v)/(expected d_v)

2.4 Thruster Mapping with Degradation

For many small autonomous space vehicles, only a minimal number of fuel tanks are carried in an effort to save space and weight. At the same time, in order to increase reliability and guard against single failures jeopardizing an entire mission, the number of thrusters is usually high (around 10-12). This means that one tank and pressure valve usually serves several thrusters; leading to a performance degradation when multiple thrusters are fired at once. This phenomenon can be easily tested and calibrated on the ground. An example of a spacecraft propulsion system that displays this degradation is the ORION spacecraft.

On the ORION spacecraft, two constant pressure tanks are used to serve twelve individual thrusters. Testing of the hardware model show that the force outputs from the individual thrusters decline rapidly as the number of on thrusters increase (Fig. 4).

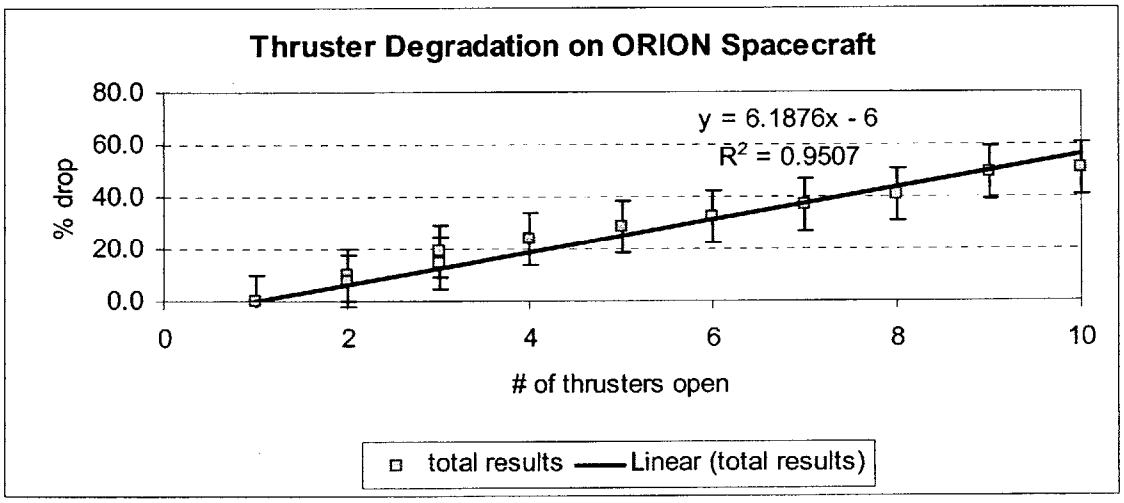


Fig. 4 Thruster force degradation due to multiple on jets

On average, turning on six thrusters at the same time would cause a 30% drop in the expected power output of each actuator. If left uncorrected, the actuation loss will directly translate into unattained spacecraft movements. This

would lead to inefficient control systems or worse yet, false failure alarms in the estimation process.

Three different ways of dealing with this phenomenon are explored in the next two sections. The first two methods involve an iterative scheme for converging on the correct solution. The third algorithm uses a mixed integer linear programming (MILP) method that is more simplistic but not as accurate.

2.4.1 Iterative Method 1

The basic ideas of this algorithm are to keep the LP mapping algorithm as before but add an outer loop that will differentially adjust the commanded (desired) rate change vector, T, in response to the errors caused by degradation. Fig. 5 shows a graphical representation of the process:

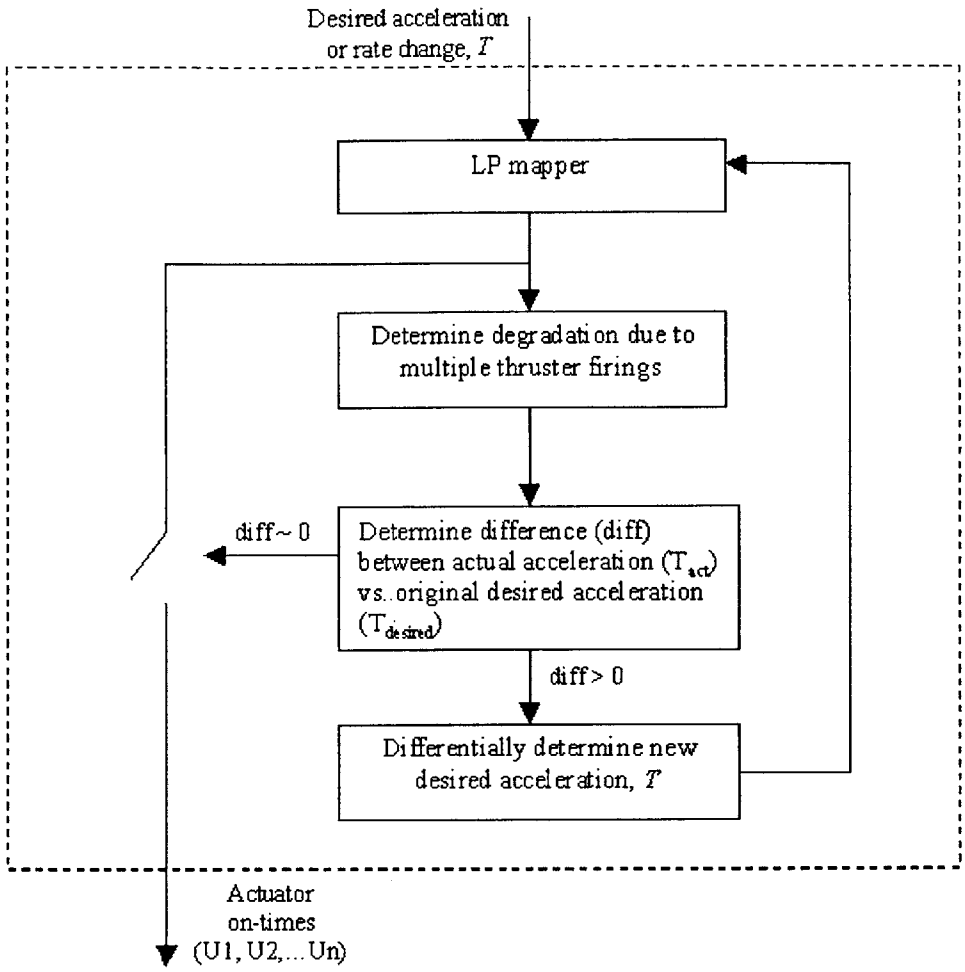


Fig. 5 Iterative method 1 for dealing with multiple thruster on degradation

The LP mapper (see section 2.2.1) returns a vector of on-times for all the thrusters that satisfy the desired acceleration, *T*. Using these on-times, one can determine which thrusters will be on at the same time at any given moment. For example, given a on-time vector of:

$$U=\begin{bmatrix}1\\2\\0\\0\end{bmatrix};$$

Thrusters one and two will be on at the same during the first second of operation, which would result in a degradation of:

$$degradation = \%\deg * \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

where *%deg* is the known degradation percentage due to two thrusters operating at the same time. Next, the true expected acceleration due to the current thruster on-times can be found along with the difference from the original desired acceleration as given by the controller:

$$T_a = Au - degradation$$

$$diff = T - T_a$$

If the difference is small (i.e. diff ~ 0) then one can exit the thrust mapping module with the current set of on-times, *U*. Otherwise, the LP mapper is called again with a higher desired acceleration.

$$T_{i+1} = T_i + diff \frac{\partial T_i}{\partial T_a}$$

$$\frac{\partial T_i}{\partial T_a} \approx \begin{bmatrix} \frac{T_{i1}}{T_{a1}} & 0 & 0 \\ 0 & .... & 0 \\ 0 & 0 & \frac{T_{im}}{T_{am}} \end{bmatrix}$$

where *m* is the degree of freedom in the problem or the length of the b-vector. The derivative is intended to increase the desired acceleration in the directions that are most deficient due to actuator degradations. An approximation for the

partial derivative is used because the true partial differential equation is difficult to calculate due to the upper-bound limit for the actuator on-times. If more than *m* actuators are needed, the derivative matrix would be non-square. (see APPENDIX E).

Table 4 shows an example which converged to the desired acceleration level in only two iterations of this algorithm.

	Force required: $T_{desired}$	[5;1;1;0;0;0]	
Iterations	T_i (Desired rate change)	T_{act} (Actual rate change due to degradation)	Degradation
1	$\begin{bmatrix} 5 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 4.52 \\ 0.82 \\ 0.82 \\ 0 \\ -0.18 \\ 0.18 \end{bmatrix}$	$\begin{bmatrix} 0.48 \\ 0.18 \\ 0.18 \\ 0 \\ 0.18 \\ -0.18 \end{bmatrix}$
2	$\begin{bmatrix} 5.531 \\ 1.2195 \\ 1.2195 \\ 0 \\ 0.18 \\ -0.18 \end{bmatrix}$	$\begin{bmatrix} 4.97 \\ 1.0 \\ 1.0 \\ 0 \\ -0.03 \\ 0.03 \end{bmatrix} \sim T_{desired}$	$\begin{bmatrix} 0.6 \\ 0.22 \\ 0.22 \\ 0 \\ 0.21 \\ -0.21 \end{bmatrix}$

Table 4, degradation convergence example using differential convergence. Note: this example uses the ORION spacecraft's thruster alignment and degradation factors.

By using a differential convergence scheme, the method is fast and usually very accurate. But due to the numerical approximation for the differential equation, $\frac{\partial T_i}{\partial T_a}$, the algorithm under certain conditions will degrade into an infinite loop. Although this phenomenon can be detected and corrected during operation, it would still cause extra computation time – usually on the order of three extra iterations and the solution would not be optimal.

With these drawbacks in mind, a second iterative method is developed that asymptotically converges to the desired accelerations. By avoiding the

numerical instabilities involved with the differential equation, simulations show that the second method provided numerical stability but at a cost of longer convergence time.

2.4.2 Iterative Method II

The central idea of this algorithm is to use an iterative method to find the extra thruster firing necessary to make up the losses due to degradation (Fig. 6). During each iteration, one finds the difference between the desired acceleration, $T_{desired}$, and the actual acceleration after degradation, T_a. The difference is then fed back into the LP mapper to find the optimal thruster firings to make up that loss (see Fig. 6). After each iteration, the upper-bound time vector, $U_{upperbnd}$, must be updated to reflect each thrusters usage during this iteration:

$$U_{i+1_{upperbnd}} = U_{i_{upperbnd}} - U_i$$

To speed up the convergence of this algorithm, the cost vector, C, of using the actuators needs be updated after each iteration. The cost of previously selected thrusters should be lowered to ensure that the LP mapper would not needlessly select new thrusters to satisfy the degradation difference.

$$C = C - U * factor$$

where the variable, *factor*, can be tested to provide the desired level of cost reduction compared to amount of thruster usage. Numerical simulations show that for the Orion spacecraft, a *factor* value of greater than 0.1 is more than sufficient to achieve the desired effect.

By giving the thrusters that are already being used a lower cost, the LP mapper would be more likely to use those thrusters to make up the acceleration deficit instead of picking new thrusters. Reducing the number of total thrusters

being used should reduce total degradation, which in turn lowers the overall time for convergence.

Fig. 6 is a graphical representation of this iteration scheme.

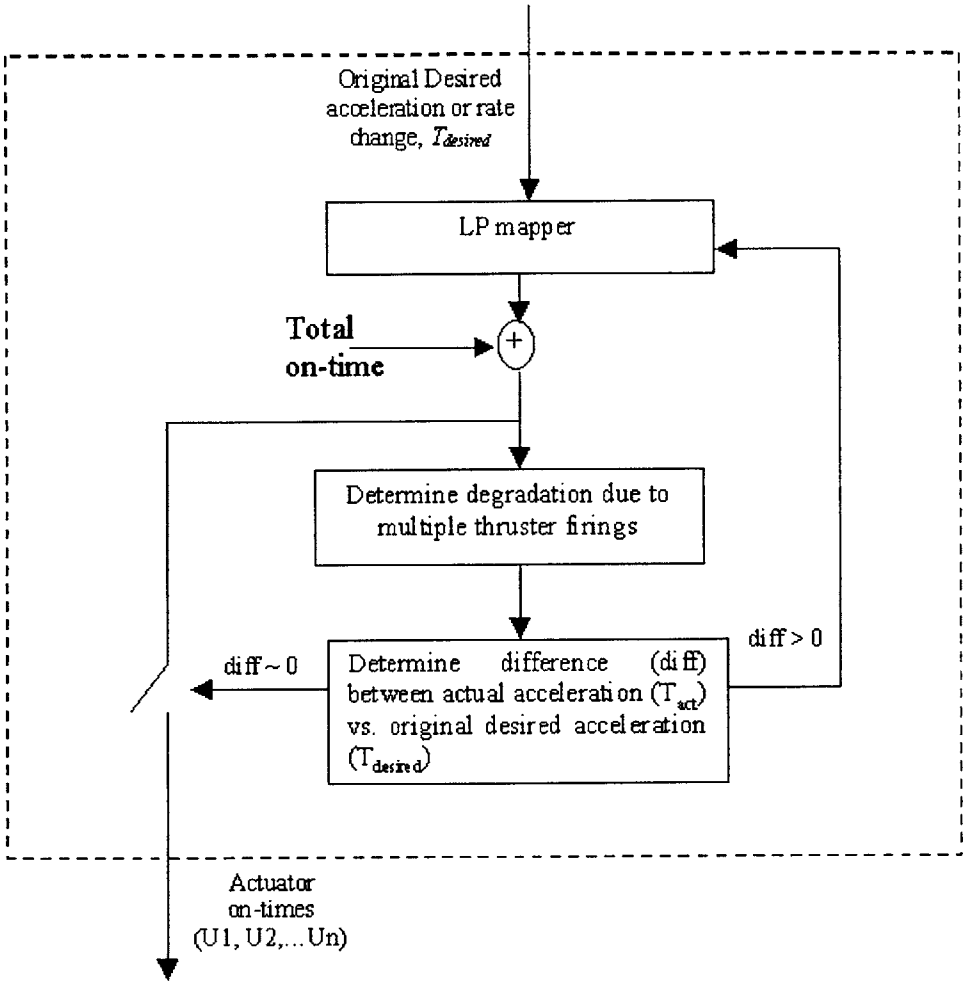


Fig. 6 Diagram of Iterative method II

A detailed algorithm for this method along with the algorithm for determining the degradation due to multiple thruster can be found in the Appendix B.

Table 5 shows an example of the mapping problem using this algorithm (Note: compare to Table 4 for differences in convergence between the two iterative methods):

	Force desired: $T_{desired}$	[5;1;1;0;0;0]
Iterations	T_i (acceleration wanted)	T_{act} (acceleration after degradation
1	$\begin{bmatrix} 5 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 4.52 \\ 0.82 \\ 0.82 \\ 0 \\ -0.18 \\ 0.18 \end{bmatrix}$
2	$\begin{bmatrix} 0.48 \\ 0.18 \\ 0.18 \\ 0 \\ 0.18 \\ -0.18 \end{bmatrix}$	$\begin{bmatrix} 4.928 \\ 0.9676 \\ 0.9676 \\ 0 \\ -0.033 \\ 0.033 \end{bmatrix}$
3	$\begin{bmatrix} 0.072 \\ 0.032 \\ 0.032 \\ 0 \\ 0.03 \\ -0.03 \end{bmatrix}$	$\begin{bmatrix} 4.99 \\ 0.99 \\ 0.99 \\ 0 \\ -0.01 \\ 0.01 \end{bmatrix} \sim T_{desired}$

Table 5, degradation convergence example using differential convergence. Note: this example uses the ORION spacecraft's thruster alignment and degradation factors.

The second column shows the desired acceleration given to the LP solver at each iteration, T_i, and the last column is the total achievable acceleration, T_{act}, from adding together all the actuator on-times from each iteration.

The convergence speed of this algorithm hinges entirely on the total number of thrusters, their alignment and the degradation factors of the particular spacecraft in question. Using the ORION spacecraft, which has twelve thrusters, and a degradation factor of approximately 6% for each additional thruster used (Fig. 4), the maximum number of iterations to converge to within 1% of the desired acceleration level is less than seven.

Section 2.4.4 provides a more detailed comparison of the performance of the different iterative methods.

2.4.3 MILP algorithm for degradation

In this section a new programming algorithm is explored that may provide a simpler, non-looping, method of solving for the degradation problem. The mixed integer linear programming technique is employed here with limited success.

Mixed integer linear programming is different from ordinary linear programming in that some of the variables can be constrained to be integers only. This feature can be exploited to approximate the degradation factor due to multiple thruster firings. Several new variables are added to the original LP formulation. The binary variable vector, *use,* signals whether each of the actuators is being used or not. Each element in the vector is either the number 1 if that particular thruster is used or 0 otherwise. The variable *num_on* represents the total number of actuators used during the firing period. The degradation factor due to multiple thruster firings is given by *Degrade_f.* For ORION, the degradation factor is 6.2% (see Fig. 4). The variable, *degrade,* is the total degradation due to all the thruster firings during each iteration. The MILP programming was written in the AMPL mathematical programming language [Ref. 10] and the following equations show the implementation of the additional variables within the LP program.

Extra MILP constraints:

$$U(i) \le u_{upperbnd}(i)use(i) \tag{13}$$

$$num\_on = sum(use) \tag{14}$$

$$degrade = (num\_on - 1)degrade\_f \tag{15}$$

Old LP equation:

$$\hat{T} = AU + BS \tag{16}$$

New MILP equation:

$$\hat{T} = AU(1 - degrade) + BS \tag{17}$$

Although this method seems much simpler than the looping LP algorithm, there are two severe drawbacks. First, by using one degradation factor, *degrade_f*, the designer is assuming that the degradation level from multiple thruster firings is fairly linear. Although testing found this linear phenomenon on the ORION propulsion system, see Fig. 4, this cannot be assumed for all other spacecrafts. The shape of the degradation curve is dependent upon the hardware engineering of each unique propulsion system. Second, this algorithm does not differentiate between thrusters that are on for different amount of time in a control period. All the thrusters that will be fired during each iteration are considered to be overlapping even though one may fire for four seconds while the second fires for only one second. This would increase the total degradation factor and cause over usage of the thrusters and waste fuel.

Finally, due to the use of integer variables within the program, a different algorithm that can solve mixed integer linear programs (MILP) must be used to solve the problem other than Simplex. Currently there are several popular methods including [Ref. 11]: Branch and Bound, plane cutting and if only binary variables were used, the Balas/additive algorithm. All of these methods are more complex and time consuming than the simplex LP thus they provide no guarantee of faster solution times. In the following section the branch and bound approach is utilized and the computation speed is compared to the looping-LP algorithm.

Future work in this area could explore other more intelligent programming methods for dealing with the above-mentioned drawbacks.

2.4.4 Performance of Degradation Algorithms

First, we shall compare the accuracy of the algorithms in obtaining the desired acceleration. The test simulation uses the ORION spacecraft data (see APPENDIX C for full ORION system layout with relevant data information).

Three different conditions were tested – rotational only acceleration, translational only acceleration, and full 6-DOF movements. Fig. 7 shows the performance of all three algorithms in terms of providing the desired acceleration compared with using the original simplex method that had no consideration for possible degradation effects.

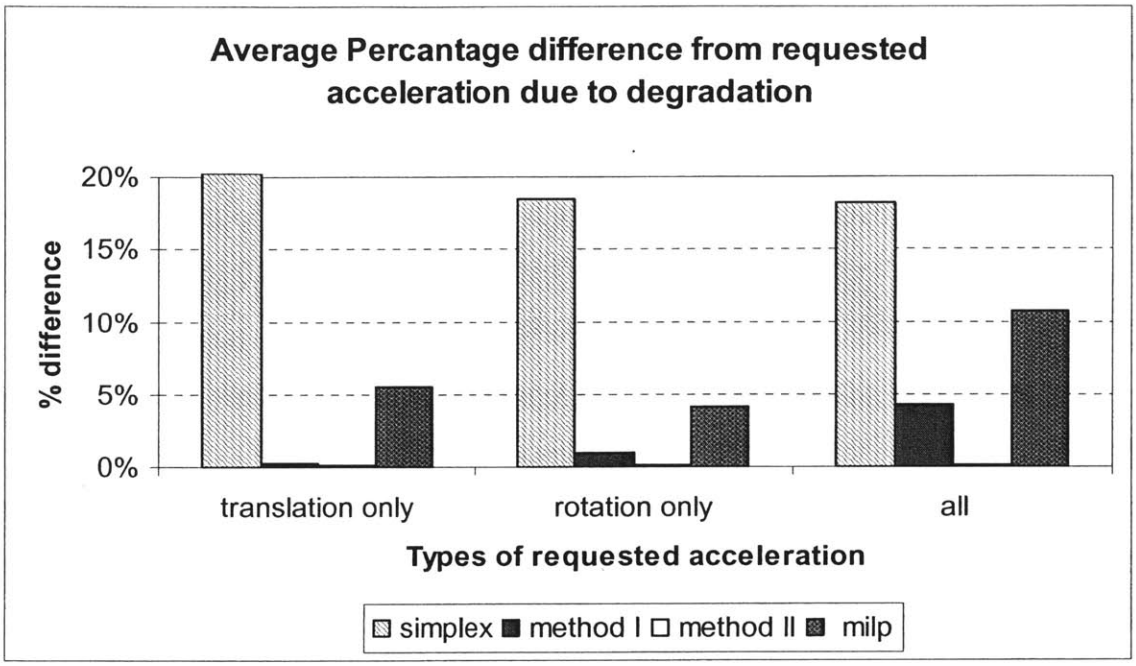


Fig. 7 Average error in achieving desired acceleration due to degradation

Fig. 7 shows that the second iterative method guarantees a true convergence while the differential iteration algorithm (method 1) performs well in the rotational or translation only requirements but failed to converge reliably in the full 6-DOF situations due to the numerical inaccuracies discussed in section 2.4.1. The MILP solutions were consistently worse because of its assumption that all thrusters will degrade by the same amount no matter how long they are on for (as explained in section 2.4.3).

Data for Fig. 7 were compiled through testing the various algorithms under a large number of different randomly selected acceleration requirements. Each algorithm individually came up with the best thruster on times to obtain the

desired velocity change and then those results were post analyzed utilizing the known degradation percentages.

Although accuracy is important, the algorithm would be useless if it took a large number of loops to converge. Using the ORION spacecraft data, the following figures show the performance of the two iterative algorithms under different conditions.

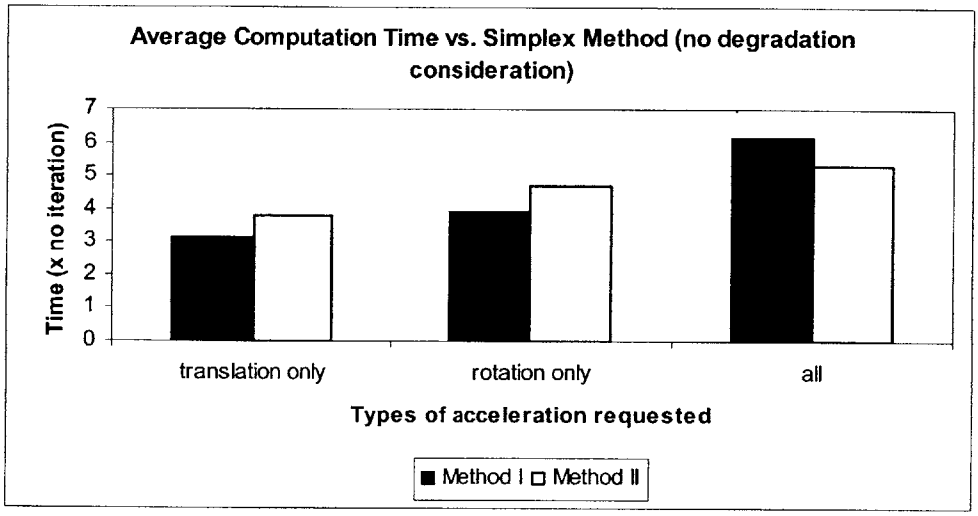


Fig. 8 Time comparison of the two iterative loops vs. no iteration

Fig. 8 shows the average time to compute the mapping algorithm using the iterative methods compared with the time it would have taken with no iterations (i.e. no consideration for degradation). On average, it took four iterations (Fig. 9) on either algorithm to converge to the desired acceleration. Overall that equates to using five times as much time as the original Simplex method that did not compensate for the degradation phenomenon.

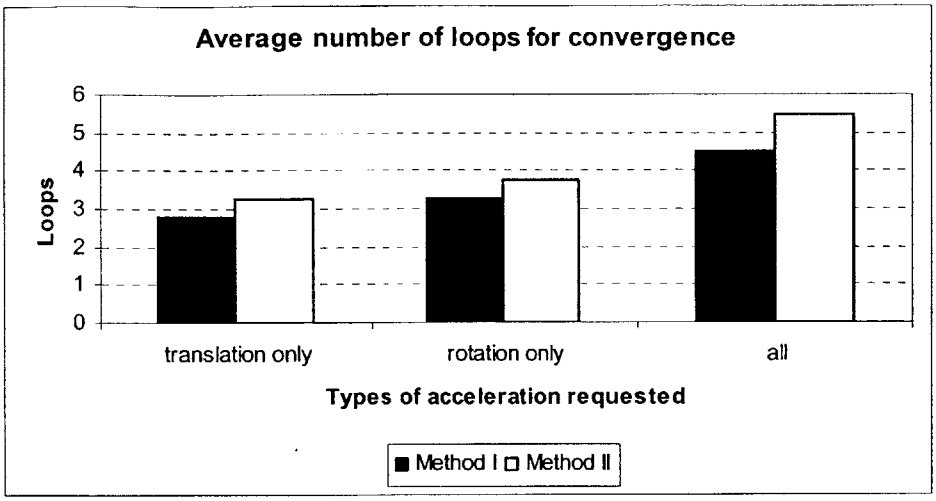


Fig. 9 Number of iterations before convergence using the different methods

In general, method 1, using the differential convergence algorithm, takes fewer iterations to converge than method 2 but due to its higher computation intensity per iteration, the total time required for each method is approximately equal. Method 1 has a slight advantage if the requested accelerations were either translation or rotation only but if combined, method 2 was faster. In the worst-case scenarios, both methods took up to nine times longer to find a solution than the original simplex LP calculation. Although these times seem high at first, one must keep in mind that the average computation time of one simplex calculation is less than 0.01 sec (see section 2.3).

The MILP method although not as accurate as the iterative methods, did have the advantage of having a faster computation time. Fig. 10 shows the average computation time of using the MILP algorithm vs. the iterative methods.

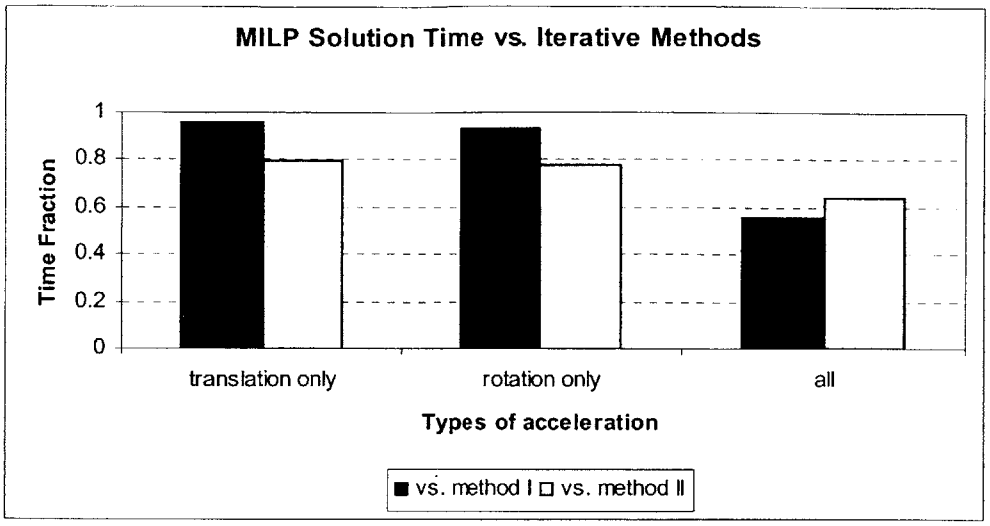


Fig. 10 Average MILP algorithm solution time compared to iterative algorithms

While the time savings for acceleration requirements of translation and rotation only were limited (at most 20%), the MILP algorithm significantly outperforms the other methods when both translation and rotational movements are required. These data suggest that the MILP methodology should be further explored to determine whether a more accurate algorithm could be found while still retaining the time savings.

Chapter 3

Robust Linear Programming For Uncertainties in the Thruster Mapping Problem

3.1 Introduction

In the above sections, the linear programming method used to determine the on-times for each thrusters assumes perfect knowledge of the desired spacecraft motion, and each thruster's performance. In reality, nothing is known precisely. In fact, all knowledge of a satellite's performance will be based on noisy instrument measurements and even the desired spacecraft movement vectors are computed based on imprecise knowledge of the states of the system.

In this chapter, new techniques for solving linear programming problems with uncertain data are explored. The two major categories of uncertainty are in the A matrix and b vector for the general $Ax=b$ LP problem. (Note the b vector plays the same role as the desired rate change vector, T, in the LP mapper. And the solution, x, would be the same as the on-times vector, u). For symmetric and non-symmetric uncertainties in the b vector, different techniques of solving the problem are presented in this Chapter. For the A matrix, a robust formulation is developed that deals with column-wise uncertainty. Column-wise uncertainty assumes that uncertainty in each column is independent. The column wise uncertainty is a specific case of the more general row-wise uncertainty problem, and can still be formulated into a Linear Programming problem while the general row-wise problem is no longer an LP program [Ref. 12].

Theoretical and numerical test results show that under special cases of error distribution, where the errors in b are symmetric and the errors in A are also symmetric and are a small percentage (<50%) of the original values, the robust

LP problem collapses down to the original *Ax=b* problem. In fact, this Thesis shows that this special case of error distribution is the most likely of the errors to be encountered, thus for most spacecraft problems, the original linear programming formulation is the best method even in the face of uncertainty.

3.2 Uncertainty in *b*

Uncertainty in the *b* vector can arise from many different situations. In the spacecraft thrust mapping problem, the most common uncertainty or errors are due to noisy measurements. For example, if the desired acceleration values are received from the controller in terms of one reference frame (i.e. LVLH), then it is necessary to perform a transformation to the body-frame to accomplish the thruster mapping problem. The mapping problem is usually done in the body frame because the thruster performance levels (matrix A) are usually tested and stored in that frame. If the attitude of the spacecraft is not known exactly then the transformed acceleration value will have an uncertainty factor attached to it (see example in second half of this section).

One robust formulation with uncertainty in the *b* vector is to find the solution that minimizes the maximum error due to all possible *b's* [Ref. 13]:

Robust LP Formulation:
$$\min_{x}(\max_{i=1..L}(b_i - Ax)) \tag{18}$$

Satisfying this constraint means finding the solution, *x*, that will minimize the maximum possible deviation from all possible desired states, b_i.

One solution to this problem involves forming another, slightly larger, linear program for the uncertainties. Let us begin by manipulating equation 18 into the following equivalent form:

$$\| Ax - b_i \| \leq S; \forall i = 1...L \tag{19}$$

$$\begin{aligned} Ax - b_i &\le S \\ Ax - b_i &\ge -S \end{aligned} \tag{20}$$

The variable S is the maximum difference between *Ax* and any of the possible *b's*. Now, combining the two inequalities and still maintaining the desire to minimize *S*, we have the following linear program:

LP formulation w/ b vector uncertainties:

$$\text{minimize}(HS + cx)$$

$$\begin{bmatrix} A & I \\ -A & -I \end{bmatrix} \begin{bmatrix} x \\ S \end{bmatrix} \le B \tag{21}$$

$$B = \begin{bmatrix} b_1 & b_2 & \dots & b_L \\ -b_1 & -b_2 & \dots & -b_L \end{bmatrix}$$

where *H* >> *c* to ensure that we are first minimizing *S*, and the total on-times second. Now, if the number of uncertainties in *b* is small, then we can just calculate the new LP formulation, equation 21, for each set of *b's*, $\begin{bmatrix} b_i \\ -b_i \end{bmatrix}$, and find the smallest combination of *x* and *S's*. But if the uncertainty set is large or continuous, it would be extremely time consuming if not impossible to solve equation 21 for every case and post analyze the solutions.

One method of solving equation 21 without considering every case of b is by minimizing each row of the matrix B [Ref. 14]:

$$b_{\min_i} = \min_i (B_{i,j}); \forall j = 1 \dots 2m \tag{22}$$

where *m* is the length of the original vector *b*. Now the uncertainty problem can again be solved with one LP calculation [Ref. 14]:

$$\text{minimize}(HS + cx)$$

robust formulation w/ uncertainty in b: $$\begin{bmatrix} A & I \\ -A & -I \end{bmatrix}\begin{bmatrix} x \\ S \end{bmatrix} \leq b_{\min} \quad (23)$$

This formulation minimizes the maximum error, S, if the uncertainties for each element of *b* are independent. Because the b_{min} vector is formed individually from each row of *B* (eqn 22), the robust formulation considers all possible combinations of the elements in *B*. For the thrust mapping problem, although one may encounter situations where the uncertainties in each element of the desired state vector, *b*, are independent, the most common uncertainties have specific characteristics that limit the total number of possible *b's*.

The most common uncertainty in the desired state vector, *b*, are from uncertainty in the measurement system. One example that is encountered by every spacecraft involves uncertainty in the measured attitude state. When the rate change requested by the master controller is not in the body frame, one has to make a reference frame-transformation:

Reference frame rotation to body: $$b_{body} = R_{obody} b_o$$

where b_o is the reference frame the controller is working in. R_{obody} is the transformation matrix to the body frame. Since the transformation matrix, R_{obody}, depends upon accurate knowledge of the attitude of the spacecraft, any uncertainties in those parameters can be modeled as an extra rotation away from the body-axis:

Rotation w/ uncertainty in attitude: $$b = R_{error} b_{body} = R_{error} R_{obody} b_o$$

where *b* is the desired state vector used in the mapping equation *Ax=b* and R_{error} is any extra rotation due to uncertainty in the attitude information. In general the R_{error} matrix is a combination of the rotational uncertainty in all three axis:

$$R_{error} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ, ϕ and ψ are the measurement errors in each of the three angular axes. If the errors are small then we can apply the small angle theorem and R_{error} becomes:

$$R_{error} = \begin{bmatrix} 1 & \psi & -\theta \\ -\psi & 1 & \phi \\ \theta & -\phi & 1 \end{bmatrix}$$

Under this type of error, the uncertainty in each element of the desired state vector, *b*, are dependent on each other.

By taking the minimum across each row, the robust algorithm as defined by equations 22 and 23 considers a much larger set of uncertainties than the given problem. Unfortunately, there are no simple methods to limit the solution space without sacrificing the speed and ease of this robust formulation. The trade-off is between efficiency and quality. This robust formulation is a one step process that gives a quick and all-inclusive answer. While it may not be the "best" solution for all types of uncertainties, it does guarantee a fast and feasible solution that adequately considers all the uncertainties. Numerical testing find that the robust formulation on average gives answers that are 20% closer to the true desired *b* than purely solving the original $Ax=b_o$ problem without considering the uncertainties.

3.2.1 Symmetric Errors in b

The uncertainties in *b* need not be considered if its errors are symmetric. If the errors in *b* are symmetric, the robust solution breaks down to the original

problem of $Ax=b_o$ and that is the exact solution to the problem. Symmetric errors in b are very common. For example, continuing the attitude measurement example in the above section.

Reference frame rotation to body: $b_{body} = R_{obody} b_o$

Rotation w/ uncertainty in attitude: $b = R_{error} b_{body} = R_{error} R_{obody} b_o$

$$R_{error} = \begin{bmatrix} 1 & \psi & -\theta \\ -\psi & 1 & \phi \\ \theta & -\phi & 1 \end{bmatrix}$$

If the measurement system is not biased towards any of the axis, then the uncertainty will be the same for all three angles ($\theta=\phi=\psi$). In that case, the uncertainties in *b*, Δb, due to measurements will be:

$$\Delta b = \begin{bmatrix} 1 & \theta & -\theta \\ -\theta & 1 & \theta \\ \theta & -\theta & 1 \end{bmatrix} b_{body} - b_{body}$$

If the angle errors, θ, are symmetric and unbiased, then the uncertainty for each element in *b* will also be symmetric:

$$\Delta b_{max} = abs(\min(\Delta b)) = abs(\max(\Delta b))$$

The following proof shows that under the above conditions of symmetric uncertainties in the elements of *b*, the robust solution is always the same as the default solution ($Ax=b_o$).

First, consider the constraint of minimizing the maximum error (b-Ax) due to uncertainty in *b*, Δb. The formulation can be rewritten as two simultaneous equations:

$$
\begin{aligned}
Ax &\le b_o + \Delta b_i + S \\
-Ax &\le -b_o + \Delta b_i + S \\
\forall i &= 1 \ldots L
\end{aligned}
\tag{24}
$$

Since the Δbs are symmetric and we want to minimize the positive slack variables, *S*, the above equations can be reduced to the following:

$$
\begin{aligned}
Ax &= b_o + \Delta b_{\max} + S \\
-Ax &= -b_o + \Delta b_{\max} + S
\end{aligned}
\tag{25}
$$

Only the equivalence case needs to be considered to minimize *S* because all other cases would only increase *S*. If b_o is in the solution space of *Ax*, then regardless of what symmetric uncertainties, Δb, are present, the best solution is always the original problem $Ax=b_o$ and the maximum error is $S=\Delta b_{\max}$.

A second proof of the above problem is based on the convex optimization theory as presented in [Ref. 13]. The structured worst-case pertubation function, ε, can be computed as:

$$
\begin{aligned}
\varepsilon_{s\infty}(x) &= \max_{i=1,\ldots,m} \left\| \tilde{A}_i x - \tilde{b}_i \right\|_1 \\
\tilde{A}_i^T &= [a_{0,i}, \rho a_{1,i}, \ldots\ldots, \rho a_{L,i}] \\
\tilde{B}_i^T &= [b_{0,i}, \rho b_{1,i}, \ldots\ldots, \rho b_{L,i}]
\end{aligned}
\tag{26}
$$

where the ρa_* and ρb_* are the errors associated with *A* and *b*. The robust solution is solved as [Ref. 13]:

$$
\min_x \max_{i=1,\ldots,m} \left\| \tilde{A}_i x - \tilde{b}_i \right\|_1 \tag{27}
$$

or

$$\min_{x} \max_{i=1,\ldots,m} \left\| \begin{matrix} A_{0,i}x - b_{0,i} \\ \rho A_{1,i}x - \rho b_{1,i} \\ \ldots. \\ \rho A_{L,i}x - \rho b_{L,i} \end{matrix} \right\|_1 \tag{28}$$

but if there is no disturbance in *A* then the problem reduces to:

$$\min_{x} \max_{i=1,\ldots,m} \left\| \begin{matrix} A_{0,i}x - b_{0,i} \\ 0 - \rho b_{1,i} \\ \ldots. \\ 0 - \rho b_{L,i} \end{matrix} \right\|_1 \tag{29}$$

Since one cannot change the uncertainties in the *b* terms (ρb) then the best solution is to make $A_{0,i}x - b_{0,i} = 0$. Collecting together all *n* elements, the solution becomes $A_0 x = b_0$, which is the original problem without the uncertainties.

3.3 Uncertainty in A

Uncertainty in the *A* matrix for the thruster mapping problem (Ax=b) arise from to the variability of each actuator's output. Every time an actuator is used, its actual thrust power may vary according to the reliability of the propulsion hardware. For example, on the ORION spacecraft, ground testing of the propulsion system shows that the thruster performance is only guaranteed to $\pm 10\%$ of the pre-calibrated levels. Of course, one would like to find solutions that are insensitive to these errors when calculating the on-times for each actuator. Again, we can set up the robust problem similar to the uncertainty in the *b*-vector

by looking for the solution, x, that minimizes the maximum possible error (b-Ax) from all the possible As:

$$\min_{x}(\max_{i=1..L}(b - A_i x)) \tag{30}$$

where L is the number of total possible A's. Following the robust formulation with variations in b, we can form a new LP constraint equation as follows:

$$\| A_i x - b \| \leq S; \forall i = 1...L \tag{31}$$

$$\begin{aligned} A_i x - b &\leq S \\ A_i x - b &\geq -S \end{aligned} \tag{32}$$

The slack variable S is the difference between the desired rate change, b, and the effective rate change from using the actuators, Ax. Now, combining the two inequalities with the desire to minimize the maximum S, we have the following set of linear objective and constraints:

$$\begin{aligned} &\text{minimize}(HS + cx) \\ &A_i^* \begin{bmatrix} x \\ S \end{bmatrix} \leq b \\ &A_i^* = \begin{bmatrix} A_i & -1 \\ -A_i & -1 \end{bmatrix}; \forall i = 1...L \end{aligned} \tag{33}$$

If the uncertainty in each column of A^* is independent (i.e. each actuator's firing uncertainty is independent) and are "column wise" then the above system of linear constraints (eqn. 33) is equivalent to the following robust LP formulation (Ref. 12):

Robust LP for variations in A:

$$\begin{aligned} &A^* x^* \leq b^*, x^* \geq 0; \\ &a_{ij}^* = \sup_{a_i \in K_i} (a_i)_j \end{aligned} \tag{34}$$

where K_i represents the total set of possible *A*'s and

$$A^* = \begin{bmatrix} A & -1 \\ -A & -1 \end{bmatrix}$$
$$x^* = \begin{bmatrix} x \\ S \end{bmatrix} \qquad (35)$$
$$b^* = \begin{bmatrix} b \\ -b \end{bmatrix}$$

It should be noted here that the case of column-wise uncertainty is conservative: the constraints of the robust formulation correspond to the case when every entry in the constraint matrix is as "bad" (as large) as it could be [Ref. 12]. For the mapping problem, this "conservatism" is not a problem when the values in each column of the *A* matrix all have the same sign (i.e. either all positive or all negative). If the signs vary for each element then the robust formulation (eqn 34) will give answers that are more conservative (see discussion on conservatism in 3.2). This conservatism is introduced because the algorithm considers the elements in each column of A to vary independently. To understand why the mixing of signs in the *A* matrix is a problem, lets compare two different actuators of a sample problem:

The sample spacecraft operates in three-degrees of freedom:

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & 1 \end{bmatrix}; \; b = \begin{bmatrix} 3 \\ 1 \\ 3 \end{bmatrix}; \; x \le 5$$

If a +/- 10% uncertainty is present in both actuators, the robust formulation the problem would be:

$$A^* x^* \le b^*, x^* \ge 0;$$
$$a_{ij}^* = \sup_{a_i \in K_i} (a_i)_j$$

$$A^* = \begin{bmatrix} A & -1 \\ -A & -1 \end{bmatrix} = \begin{bmatrix} 1.1 & 1.1 & -1 & & \\ -0.9 & 1.1 & & -1 & \\ 1.1 & 1.1 & & & -1 \\ -0.9 & -0.9 & -1 & & \\ 1.1 & -0.9 & & -1 & \\ -0.9 & -0.9 & & & -1 \end{bmatrix}$$

This formulation assumes that the "worst" (largest) possible *A's* are:

Robust Formulation's worst-case scenario:

$$A^+_{worst} = \begin{bmatrix} 1.1 & 1.1 \\ -0.9 & 1.1 \\ 1.1 & 1.1 \end{bmatrix}; \; A^-_{worst} = \begin{bmatrix} -0.9 & -0.9 \\ 0.9 & -0.9 \\ -0.9 & -0.9 \end{bmatrix};$$

But because each element in a column cannot vary by itself (i.e. the elements of a column vector are not independent), the real "worst" case scenarios are:

Actual worst-case scenarios:

$$A^+_{worst} = \begin{bmatrix} 1.1 & 1.1 \\ -1.1 & 1.1 \\ 1.1 & 1.1 \end{bmatrix}; \; A^-_{worst} = \begin{bmatrix} -0.9 & -0.9 \\ 0.9 & -0.9 \\ -0.9 & -0.9 \end{bmatrix};$$

The second column of the actual worst-case scenario is equivalent to the robust formulation but the first column is not. The example shows that for the case when all the elements in a particular actuator column do not have the same sign, the robust formulation will consider a constraint space that is larger than the actual constraint space.

Numerical testing can be used to test the conservatism of the robust formulation. The test is set up to determine if one can find a solution, x, that gives a lower objective value (eqn. 33) than the robust solution. A sample set of

possible solutions, x, can be found by considering different column combinations of A's:

$$A_i x_i = b; \forall i = 1...L$$

where L is the total number of different combinations of A. Using the above example, some possibilities of A are:

$$A_i = \begin{bmatrix} 1.02 & 0.95 \\ -1.02 & 0.95 \\ 1.02 & 0.95 \end{bmatrix}; \begin{bmatrix} 1.1 & 1.1 \\ -1.1 & 1.1 \\ 1.1 & 1.1 \end{bmatrix}; \begin{bmatrix} 0.91 & 1 \\ -0.91 & 1 \\ 0.91 & 1 \end{bmatrix} ....$$

Again, note that each column's variation is independent but all the elements within a particular column must fluctuate together.

For each A_i, there is a unique solution, x_i, that satisfies the desired rate change. The post analysis to determine the best x_i involves finding the maximum possible error from using each set of solutions:

$$\min_{j=1..L}[\max(A_i x_j - b)]; \forall i = 1...L$$

Table 6 shows the percentage of correct solutions using robust formulation under various types of variations in the columns of the thruster mapper, A.

Types of variation in the columns of thruster mapper, A	Ax=b solution No consideration for variations in A	Robust LP formulation $A^*x^*<= b^*$
• Symmetric errors • Errors less than 10% of original values • All columns had same max variations • Entries in A are all positive or all negative	100%	100%
• Non-symmetric errors • Errors less than 10% of original values • All columns had same max variations • Entries in A are all positive or all negative	0%	100%
• Symmetric errors • Errors greater than 100% of original values • All columns had same max variations • Entries in A are all positive or all negative	0%	100%
• Symmetric errors • Errors less than 10% of original values • Columns had different max variations • Entries in A are all positive or all negative	32%	100%
• Symmetric errors • Errors less than 10% of original values • All columns had same max variations • Entries in A are either positive or negative	90%	92%

Table 6, Percentage of correct solutions for different types of variations in *A*. Note: the values of the A and b matrices in all simulations were completely random to give an idea of the general conservatism of the algorithms

From the numerical tests based upon the above formulations, it is found that the robust LP formulation finds the best solution under all circumstances unless the elements in the thrust mapper matrix, *A*, has both positive and negative values. One further interesting result is that the default solution, x_o, which does not consider any variations in *A* was found to be the best solution if the variations in *A* were symmetric and small (<10%) (see Table 6).

One way to combat the "conservatism" of the robust formulation is to consider the actuator uncertainties as "row-wise" uncertainties. In the case of "row-wise" uncertainty, a robust program can be found to reflect the fact that the

coefficients of the constraints cannot simultaneously be as bad as every one of them could be [Ref. 12]. Unfortunately, the general "row-wise" uncertainty problem can only be transformed into a conic quadratic program (CQP) [Ref. 12] that is much more computationally complex than an LP problem. Due to this drawback, the conic quadratic solution was not pursued in this Thesis but future work could help to define the trade-off between computational complexities of a CQP vs. the "conservatism" of the robust LP solution.

3.4 Other Ways of Dealing with Uncertainties in A and b

The previous sections 3.2 and 3.3 considered uncertainties in both the matrix A and vector b to be hard constraints that must be satisfied. An alternative method of solving for this class of uncertainties is through stochastic programming [Ref. 32]. A sample formulation for the thruster mapping using stochastic programming is:

$$\begin{aligned}&\text{minimize}: cx\\&\text{constraint}: prob(\|b - Ax\| \le S) \ge 1 - alpha\\&x, S \ge 0\end{aligned}$$

In this formulation, x is still the unknown solution but S is a predetermined error that one is willing to tolerate and *alpha* is the probability of failure. This formulation requires the knowledge of the probability distributions of A and b.

To transform the above probabilistic constraints into an equivalent deterministic equation is considered in Ref. 32 and will not be discussed here. The main point is that if the distributions of A and b are continuous then the deterministic equations become a set of nonlinear constraints that are much more computationally complex to solve for than the robust formulation developed in sections 3.2 and 3.3. But, if the distribution is discrete, the new constraints

can be transformed into a linear mixed-integer program [Ref. 32]. Further research is needed to determine if that is a viable alternative in terms of computational burden.

3.5 Conclusions

This chapter analyzed the thrust mapping problem under uncertainties in both the *A* and *b* matrices. Uncertainties in the *A* matrix arise from variations in the actuator performance or uncertainties in the measurement system. The *b* vector can also be inexact due to poor measurements or uncertainties in the control system. In both cases, robust formulations of the original LP problem were derived along with a performance comparison with the original Ax=b solution that did not consider the uncertainties in the system. The robust solutions designed in this thesis focused on computational speed with some sacrifices in precision. Although the robust solutions may be conservative in some specific cases, it did guarantee a better solution than the original thruster mapper under all conditions.

For the *b* vector, if the uncertainties are symmetric, it was shown that the robust formulation reduces to the original $Ax=b_o$ solution. If the uncertainties are skewed, then the robust solution is on average 20% better than the original solution.

For the *A* matrix, the most common and likely uncertainties are column-wise uncertainties. This assumes that each actuator is independent and could have varying degrees of deviations. Again, the robust algorithm developed here opted for speed with some sacrifice in the final solution. These flaws appear when the elements in each column of *A* do not have the same sign (i.e. some values of the column are positive while the others are negative). This "conservatism" is negligible when the uncertainties are small or extremely large.

Chapter 4

Estimation

4.1 Introduction

In order to guarantee the continued success of the thruster mapper model developed in the above sections, it is essential to have a continued monitoring system on the performance of the thrusters. A sound monitoring system will not only provide the best estimate of the performance of the thrusters but also allow fast detections of failures.

Fig. 11 shows the basic structure of the estimation and fault detection algorithm.

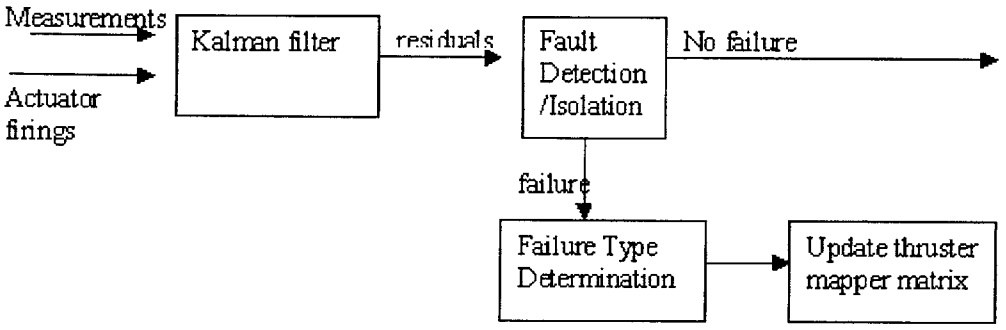


Fig. 11 Estimation and Fault Detection

This chapter describes how estimates of the thruster performance values can be generated using knowledge of the thruster on-times and measured velocities. A basic Kalman filter system is used for continuous monitoring of the thruster performance levels. This filter was selected both for its ease of implementation and statistical reliability to provide the best estimate at each step.

4.2 Kalman Filter and Estimation

This section provides a simple and straightforward method to formulate the estimation problem of determining the performance of each of the actuators onboard a satellite. The estimation states are the elements that form the thruster mapper matrix (Table 1 in section 2.2.1). In that example, there are four actuators and six degrees of freedom. This leads to a state vector, x, of length 24 x 1 by arranging each column under the previous ones.

4.2.1 State Dynamics

First, it is assumed that the actuator's performance levels are stable and any failures and degradations happen in an unpredictable fashion. The deterministic model for the state dynamics can be described as:

State dynamics $$\dot{\underline{x}}(t) = \underline{0} \tag{36}$$

This continuous equation can be rewritten in discrete form for computer processing as:

Discrete state dynamics $$\underline{x}_{k+1} = \Phi x_k \tag{37}$$

where the single-step state transition matrix, Φ is simply the identity matrix of size L x L where L is the length of the state vector x and is also the product of the number of actuators (n) and degrees of freedom (m=6).

At each discrete step, an additive noise term is added to the discrete dynamics equation to account for unmodeled system variations:

State dynamics w/ noise $$\underline{x}_{k+1} = \Phi x_k + w_k \tag{38}$$

where w_k is a L-dimensional hypothesized white gaussian sequence that's independent of x_k with the following properties:

$$E(\underline{w}_k) = \underline{0}$$
$$E(\underline{w}_k \underline{w}_k^T) = Q_k \geq 0 \tag{39}$$

The noise in the system, w_k, serves two purposes in this setting. First it is an attempt to describe some of the unmodeled variations in the system. For example, sometimes there are inherent uncertainties in the thruster output levels due to inexact mechanical or valve control techniques. Second, it prevents the filter gains from becoming overly optimistic after a period of time. If they do become optimistic, the filter will tend to ignore recent data and thus fail to track variations in the state estimates, *x*, as they occur [Ref. 17]. Furthermore, the values of Q_k can be tuned to attain desired response properties in the filter.

Finally, if unknown changes occur in the performance properties of the actuators, either due to failures or large degradatation, then the state dynamic equation will become:

State dynamics w/ failure $$\underline{x}_{k+1} = \Phi \underline{x}_k + \underline{w}_k + \delta_{k+1;\theta} \underline{\upsilon} \tag{40}$$

Where υ is the unknown failure jump, θ is the jump interval and $\delta_{k;\theta}$ is the kronenecker delta function defined to equal 0 except when $k=\theta$, then it takes the value of 1. This definition will be of importance when failure detection is discussed in the following sections.

4.2.2 Propagation of Statistics and Measurement Incorporation

First, we must define the statistics associated with measurement states, Z. In most spacecraft applications, especially small satellite operations, no instruments exist for the sole purpose of measuring the performance of each of the actuators. But, some type of velocity measurement system must exist to

perform vehicular control. This information can be used readily to help in the estimation of the actuator output levels.

At each discrete step, measurements of the impulsive change in the translational and angular velocities are taken as:

Measurement equation:
$$\underline{Z}_k = \underline{u}_k - \underline{u}_{k-1} \tag{41}$$

Where U is a mx1 vector that captures both the angular, ω, and translation, v, velocities:

$$\underline{u}_k = \begin{bmatrix} \underline{v}_k \\ \underline{\omega}_k \end{bmatrix} \tag{42}$$

Usually, instruments take these velocity measurements with known degrees of uncertainty. This noise, u^d, can be usually approximated as a white gaussian sequence with statistics of:

Measurement Uncertainty
$$\begin{aligned} E(\underline{u}_k^d) &= 0 \\ E(\underline{u}_k^d \underline{u}_k^{dT}) &= R_k \geq 0 \end{aligned} \tag{43}$$

It is assumed that we have an initial estimate of the actuator performance levels when the satellite begins operation. This prior estimate may come from ground testing before flight and will be denoted as $\hat{x}_k^-$ where the "hat" denotes the estimate, and the "super minus" indicates this is the best estimate prior to incorporating any measurement at time step k. It is now helpful to assume that we know the error covariance matrix associated with $\hat{x}_k^-$. First, the estimation error is defined to be the difference between the actual values, x_k, and the current best estimate, $\hat{x}_k$:

Estimation error:
$$e_k^- = x_k - \hat{x}_k^- \tag{44}$$

Assuming that the estimation error has zero mean, the associated error covariance matrix is:

Error covariance $$P_k^- = E(e_k^- e_k^{-T}) = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] \quad (45)$$

In most cases, the estimation problem begins with no prior measurements and the error covariance matrix, P_k, would equal the system error covariance, Q_k [Ref. 15].

We are now ready to determine the new estimate, $\hat{x}_k$, by incorporating the most recent noisy measurement with the prior estimate. Using Bayes theorem and conditional density properties [Ref. 15], the optimal new estimate based on minimum mean-square error can be found through blending the new measurement with the prior estimates:

Measurement Update: $$\hat{x}_k = \hat{x}_k^- + K_k(Z_k - H_k\hat{x}_k^-) \quad (46)$$

Where H_k is a m x L matrix that contains the on-time information for all the actuators at time step k and K_k is the blending factor that is known as the Kalman gain.

Substituting equation 46 into equation 45, the updated covariance matrix is found to be:

$$P_k = (I - K_k H_k)P_k^- \quad (47)$$

Note, there are several other expressions of P_k that would yield the identical answers and should be considered if equation 47 does not produce satisfactory arithmetic solutions due to digital round-off errors [Ref. 15].

Lets now consider propagating the error covariance matrix. The error covariance associated with $\hat{x}_{k+1}^-$ is found through first determining the priori error equation:

Priori error equation: $$e_{k+1}^{-} = x_{k+1} - \hat{x}_{k+1}^{-} = \Phi_k e_k + w_k \quad (48)$$

The propagation of the error covariance matrix, P, is the expected covariance of the estimation error, e. In our particular case, the propagated error covariance, p_{k+1}^{-}, is just the last covariance estimate, P_k, plus the system uncertainty, Q_k, since Φ in a static Kalman filter is the identity matrix.

Propagated Err Cov: $$P_{k+1}^{-} = E(e_{k+1}^{-} e_{k+1}^{-T}) = \Phi_k P_k \Phi_k^T + Q_k \quad (49)$$

Finally, the optimal Kalman gain, K_k, that minimize the mean-square estimation error is:

Kalman gain: $$K_k = P_k^{-} H_k^T (H_k P_k^{-} H_k^T + R_k)^{-1} \quad (50)$$

4.2.3 Kalman Filter Summary Equations

The previous sections gave the mathematical background for the formulation of the equations in using the Kalman filter to estimate the performance of the thrusters. The estimates from the filter are optimal in the sense that the mean square of the estimation error is minimized. Several assumptions were made in using this filter and they are: the dynamics and measurement equations are linear in terms of the state and that the additive process and measurement noises are zero-mean, white, gaussian and independent of each other and the state.

In summary the filter equations are as follows:

The propagation equations in between measurements are:

State propagation: $$\underline{\hat{x}}_{k+1}^{-} = \Phi \underline{\hat{x}}_k \quad (51)$$

Err covariance propagation: $$P_{k+1}^{-} = \Phi_k P_k \Phi_k^T + Q_k \quad (52)$$

where Φ is the identity matrix.

The update equations across measurements are:

State Update: $$\hat{x}_k = \hat{x}_k^- + K_k \varepsilon_k \quad (53)$$

Covariance update: $$P_k = (I - K_k H_k) P_k^- \quad (54)$$

Where

Residual: $$\varepsilon_k = Z_k - H_k \hat{x}_k^- \quad (55)$$

Kalman gain: $$K_k^T = P_k H_k^T Y_k^{-1} \quad (56)$$

Residual covariance $$Y_k = H_k P_k H_k^T + R_k \quad (57)$$

The vector ε_k is the zero-mean residual and Y_k is its covariance. Because the residual term from the Kalman filter is well characterized, it will be exploited later for fault detection (Chapter 5).

Other notes of interest is that the Kalman gain is independent of the state but depend upon the applied impulses, H_k. This means that the gain will always be time varying even when *R* and *Q* are constant and the error covariance reaches a steady-state operation.

4.2.4 Filter Simplifications

One standard approach to speeding up the Kalman filter is by determining the steady-state Kalman gain, K, so one would not have to calculate it at every step. The Kalman gain vector, K, is independent of the state but is closely related with the applied thruster firings, H. Since in the actuator estimation problem, the applied actuator firings are changing at every time step, even when the residual covariance of the estimate reaches a steady-state value, K will be

time varying. Thus, one cannot precompute a steady-state K value beforehand to save computation time on-board.

Because of the possible large number of states (degree of freedom [m] times number of actuators [n]), one method of saving computation time would be estimate only the values of the state that is used in the current iteration. At each step only a few of the total actuators will be required and since all thrusters are independent of each other, only the states that relate to the currently used thrusters need to be estimated. At every time step, the number of floating point calculations depend heavily upon the size of the error covariance, *P*. If instead of passing in the values of all twelve actuators but only the ones that are used are processed, then the number of floating point calculations saved per time step based on a full 6-DOF spacecraft is over $(n*6)^3$ where n is the number of idle actuators. Furthermore, since this is a static Kalman filter and Φ is the identity matrix, the propagation equations can be reduced to an addition of the last best estimate and the related noise. Under these simplifications the Kalman filter equations are:

State propagation: $$\hat{\underline{x}}_{k+1}^{-} = \hat{\underline{x}}_k \tag{58}$$

Err covariance propagation: $$P_{k+1}^{-} = P_k + Q_k \tag{59}$$

The update equations are the same as equations 53 – 57 except that the states, x, and associated error covariance matrix, P, will be:

Estimated States and Error Cov: $$\hat{x}_k(i), P_k(i,i) \tag{60}$$

where i is a vector that contains the indices of the states that are associated with the actuators that were fired in this step. For example, if actuators one and three of a four actuator spacecraft fired at step k, the i vector for that case would be:

$$i = [1,2, \ldots, 6, 13, 14, \ldots 18] \tag{61}$$

Chapter 5

Fault Detection and Re-Organization

5.1 Introduction

Beyond maintaining a good estimate of the actuator performance levels, it is also essential to have a fast fault detection system that can quickly and reliably detect and isolate any failures to the actuation system. Speed and robustness are both important components of this system. When a failure occurs, one can always wait for the estimation to converge to the new values but that usually wastes a lot of fuel as it will take several firings before the estimator can converge close to the new value. The reason for this is if estimation occurs during multiple actuator firings, then the error would likely be spread to the other actuators that were also on and it would take the estimator a long time to completely contribute all the errors to the failed thruster. Thus it is important to develop an algorithm that not only can quickly isolate a failure but also identify the exact failure nature. But an algorithm that is extremely fast in detecting any failures would also be useless if it routinely misdiagnoses the failure. There are two types of errors that can occur in detecting failures. The type I error is defined as not detecting an actual failure while type II error is when a failure is falsely detected. Table 7 illustrates the two failure types:

Decision	Failure Present	No Failure
Failure detected	Ok	Type II error
Failure NOT detected	Type I error	Ok

Table 7, Decisions errors for fault detection

Sometimes a type I error is referred to as the significance level of a statistical test [Ref. 18]. The higher that error, the more likely the test would not determine a failure when it occurs. Unfortunately, if the detection levels are set low so as to minimize type I error, it is usually at the expense of having more type II errors. Type II errors are especially intolerable because a wrong fault detection would risk permanently losing a working actuator as the compensation algorithm may shut down any actuators that are not working correctly. Thus a system that is designed to respond quickly to failures must necessarily be sensitive to certain high frequency effects and this in turn will tend to increase the sensitivity of the system to noise and the occurrence of false alarms [Ref. 19]. This fundamental tradeoff between these design issues must be grounded in the actual system in question. That is, if the spacecraft has many redundant actuators then it may be tolerable to have a high false alarm rate vs. a system without substantial back-up capabilities. Other considerations include on-board computing power and mission requirements on control precision. A complex detection system may reliably find and isolate any failures to guarantee performance, but it probably would come at an extremely high computation cost that most small spacecrafts cannot provide.

Many fault detection algorithms have been developed in the past 30 years due to increases in computing power and improvement in on-board instrumentation. The different methods include 'failure-sensitive' filters, innovation or residual based detection, model-hypothesis testing and neural networks [Ref. 19]. The 'failure-sensitive' filter refers to filters that have state

estimates which are more sensitive to failures than normal filters. Examples of such filters include using a limited memory filter, increasing noise covariance or simply fixing the filter gains [Ref. 27 and Ref. 28].

The innovation or residual based detection system is the most popular of the aforementioned group. The fact that the monitoring system can be attached to the Kalman filter is particularly appealing to most applications since that should cut down on redundant computation costs. A number of statistical tests can be performed on the Kalman filter residuals including chi-squared testing, and generalized likelihood ratio approach (GLR) [Ref. 16, Ref. 17, Ref. 19, Ref. 21, and Ref. 22]. The chi-squared tests draw on the fact that the residuals should be white, zero-mean and independent. Deviations from any of those properties can be detected using the chi-squared tests. The GLR method is slightly more complicated because it tries to isolate different failures by using knowledge of the different effects such failures would have on the system innovations. In the thruster mapper system, the model effects of each actuator failure is just the acceleration characteristic of each thruster (columns of Table 1).

Trying to detect and isolate the failures to certain pre-computed models motivates the model-hypothesis approach. These methods have the benefit of doing both system identification and state estimation. Usually a large "bank" of linear filters based on different model hypotheses are run simultaneously and the innovations from the various filters are used to compute the conditional probability that each system model is the correct one. Although proven to be fairly accurate [Ref. 19], this method usually carries a high computation burden to accommodate all possible, or at least the most likely, failure modes.

Finally, much work has been done on implementing a neural network-learning algorithm to accommodate any type of failure. The idea is that the estimation procedure is coupled with a learning algorithm that analyzes all incoming measurements to determine the best estimate of the performance of the systems. Several detailed studies have been done in this area and the

results have been extremely positive, especially when dealing with unanticipated faults, which are hard to characterize using any other fault detection method [Ref. 29]. The drawbacks to this method are that detecting anticipated faults may take relatively longer time than using some of the other methods [Ref. 24, Ref. 23] and it has high computation and storage requirements.

For this thesis, we picked several of the above methods that were thought to be more suitable for small satellite applications. Small satellites are usually characterized by relatively low computing power, noisier instruments, and very few or no redundancies in system parameters. Also, the detection method must work well with the above-described thruster mapper and estimation systems described in Chapter 2 and Chapter 4. With those requirements in mind, three different innovations based tests using the chi-squared method, a GLR approach and an estimator threshold test were compared under different failure and system conditions, such as noise and actuator performance levels. The estimator threshold test basically relies on the estimates of the Kalman filter to detect failures. If the state estimates become significantly different than the original value, a failure would be signaled.

Once a failure has been detected, it is usually then necessary to isolate the failure to be able to update the system model for the control system. A model comparison technique is introduced here that can work with any of the detection systems. The technique involves relatively low computation time and can be fine tuned to any system depending upon noise levels and accuracy requirements.

5.2 Fault Detection and Isolation

5.2.1 Problem Definition

The overall performance of the thruster mapper system greatly depends upon its ability to adapt to changes in the actuator system. One of the most

basic yet important tasks in estimation is the ability to detect failures rapidly and update the system to the post-failure model. There are many types of failures that could affect the performance of any system: initial misalignment, long-term degradation and sudden abrupt failures.

Initial misalignment refers to the possibility of actuators being misaligned during its flight to space. This type of failure is fairly uncommon and will not be studied in-depth in this thesis (although in Chapter 6, we will test the effectiveness of the method described below). First of all, it would be nearly impossible to identify the true behavior of the actuators if one is to solely base the values on the estimation process (section 4.2) during initial normal operations after launch. This is because of the high number of estimated states (number of actuators x DOF) and the limited observable space (6 degrees of freedom). One possibility of testing for such failures is to use each actuator at least once after launch to test for any mishaps during its space travel. The resulting estimates of the performance of each actuator can be easily used to measure the likelihood of failure compared to default (expected) values (see Chapter 6 for results).

The second type of failure, long-term degradation, is much more common and can result from many different hardware issues including decreased pressure from the propulsion valve or clogging of the thruster plumbing. If such long-term degradation is known and can be modeled through ground-testing, it is then a simple case of attaching such a model to the estimation system:

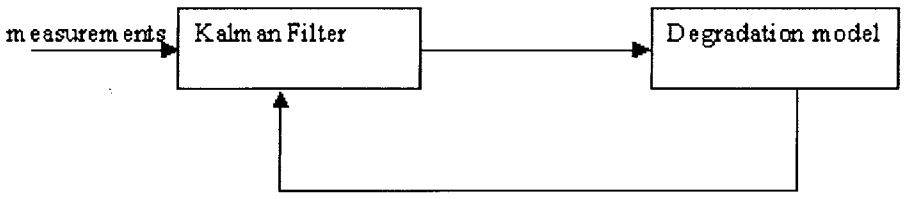


Fig. 12 Implementation of known degradation model

But, in some cases the exact nature of the degradation history is not known and must be determined in-flight. Two possibilities immediately arise as possible

methods in calibrating for this degradation. If the system noise levels (actuator performance and measurement accuracy) are low, then the Kalman filter can be designed to have a fast response time through either high Q/R ratio or artificially keeping the state covariance factors high. A "fast" Kalman filter will respond to new measurements more readily and be better able to track long-term degradation factors.

The second method involves using the fault detection algorithms developed in the next section for abrupt changes. The intuition behind this method is that if the inherent system noise is high, then one cannot rely on the Kalman filter to accurately track small degradation levels. But over time, as the actuator performance is significantly degraded by such forces, then one should be able to detect the large deviation as a single-time failure and update the model as such. Fig. 13 shows the behavior of the model vs. the actual system for this scenario.

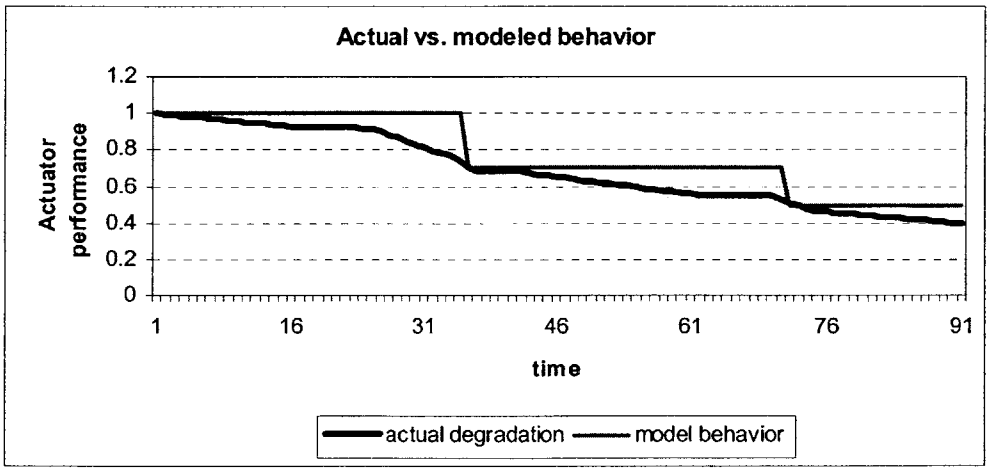


Fig. 13 One possibility of detecting long-term degradations in actuator system that are hard to model

The robustness of this system and the speed with which it updates degradation will greatly depend upon the noise level of the system and the designer's willingness to tolerate false alarms (see following sections).

The third type of failures, abrupt on/off failures, is much more likely to happen than the initial misalignment failures and cannot be modeled unlike the long-term degradations. Luckily, extensive studies have been done on this class of failures and many solutions have been demonstrated to work in the aerospace industry [Ref. 16, Ref. 19, Ref. 20, Ref. 21, Ref. 22, Ref. 30, Ref. 29]. The design of failure detection systems involves many factors of consideration. First of all, the detection system must work with the rest of the system structure including the thruster mapper and the Kalman filter. Second, there is the trade-off between the responsiveness of the detection system and the willingness to tolerate false alarms which can significantly degrade the performance of the overall spacecraft. Third, hardware issues such as sensor redundancy must be considered. Finally, the trade-off between computational complexity and performance is especially crucial for small autonomous spacecrafts that typically do not be carry the state-of-the-art computers and have limited storage capacity.

Although there are many different fault detection methods, our system with the limited measurement capabilities (measurements are six degrees of freedom only – not individual thrusters), no sensor redundancy, relatively high measurement noise and poor computing power make most of the systems impractical. For example, the "failure-sensitive" filters, which have relatively low computation time, require extremely accurate measurements or else not only does the fault detection performance degrade, but the estimator becomes sub-optimal as well [Ref. 19].

The simplest detection method would be to rely on estimates of the actuator performance from the Kalman filter (Estimator threshold test). When the estimated values cross certain thresholds (such as 80% or 60% of original value) then one can declare a possible failure in that actuator. Although simple, this method has many drawbacks. First, the filter could become very insensitive to new measurements after prolonged operations. If a failure occured, the estimates would change very slowly, thus making a fast detection nearly

impossible. If one tried to sensitize the filter by lowering the measurement noise value in the filter, R, one may run the risk of having an erratic system that is extremely sensitive to the inherent noise factors in the system. Also, in our particular problem, the high ratio of estimated states vs. number of actual observable space limit the performance of the Kalman estimation process. For the above reasons, in most real-life situations, one cannot simply rely on the filter for failure detection.

In our system where one must make the decision on failures from single measurements because thruster firings change continuously over time while the measurements are of the same variables, the most robust systems to use are the innovation-based detection systems. Two classes of innovation-based detection systems are compared here and both are based on the chi-squared test. The first test is a generic chi-squared test on the innovations and is based on the work of Mehra and Peschon [Ref. 22] who determined many statistical tests to be performed on the innovations. The second innovations-based approach is called the generalized likelihood ratio (GLR) and is motivated by the shortcomings of the simpler chi-squared approach and can be used to isolate different failures by using knowledge of the effects each failures have on systems.

5.2.2 Simple Chi-Squared Tests

The innovations (or residuals, ε) of the Kalman filter are defined as the difference between the actual system output and the expected output based on the model. Under normal conditions, the error signal is "small" and corresponds to random fluctuations that match the variations present in both the system and the measurement devices. It is well known that a Kalman filter generates innovation values that are white, zero-mean and Gaussian in nature [Ref. 22]. In this case the chi-squared test value at step k is a chi-squared random variable with m degrees of freedom.

Chi-squared test value

$$l_k = \sum_{j=1}^{m} \varepsilon'(j) Y^{-1}(j) \varepsilon(j) \tag{62}$$

In the equation, ε , the residual, and Y its corresponding covariance, are both precalculated values from the Kalman filter; m is the number of degree of freedom in the system – usually six. If a system abnormality occurs, the statistics of ε change and the detection rule takes the form of:

chi-squared detection rule:

$$\begin{aligned} l_k > \delta &\Rightarrow FAILURE \\ l_k \leq \delta &\Rightarrow NOFAILURE \end{aligned} \tag{63}$$

with the aid of chi-squared tables, one can find the desired level of δ that gives high detection rate at an acceptable rate of false alarms.

The implementation of the above chi-squared test is extremely simple and, not surprisingly, it has rather severe limitations in performance. The method is basically an alarm method – i.e. the system makes no attempt to isolate failures – and only those failure modes that have dramatic effects on the residual are detectable by this method [Ref. 19]. By considering the components of ε separately, it might be possible to increase the detection rate and at the same time attempt to do some failure isolation also. Two variations of this method are considered below.

The first variation involves just looking at each component of the residual, ε. If the test detection is based on the last available measurement, the SDOF (single degree of freedom) chi-squared test is:

Single Chi-squared test

$$l_k(i) = \varepsilon'(i) Y^{-1}(i) \varepsilon(i); \forall i = 1..m \tag{64}$$

$$\begin{aligned} l_k(i) > \delta &\Rightarrow i^{th} component FAILURE \\ l_k(i) \leq \delta &\Rightarrow i^{th} component NOFAILURE \end{aligned} \tag{65}$$

At each step, one or more components of the residual may have an abnormally large deviation, which leads to a fault detection. Since each actuator is known to only affect the spacecraft in a limited number of directions (based on its thruster mapper), one can then make an educated guess as to which thrusters could have failed to produce the fault detection in the individual component of *l*. For example, if the only actuators on a spacecraft are completely described by Table 1 and the individual chi-squared test finds a failure in the 5th element, *l*(5), then the only thrusters that could have caused this would be actuators 3 and 4. This variation on the chi-squared test will give some knowledge for failure isolation but since on most spacecrafts there will be multiple actuators affecting one direction of movement, it will still be extremely hard to pin-point the exact failure. The third (or second variation of) chi-squared method described below tries to further refine the process.

The second variation in the chi-squared model involves looking at combinations of the components of *l* in determining failure. The intuition behind this method is that while there may be multiple thrusters that affect each direction of motion, there will be at most two or three thrusters that will affect the spacecraft exactly. Again using Table 1 as a sample set of actuators on a spacecraft, it is clear that all four actuators affect the spacecraft uniquely in that no two of the actuators affect the same set of directions. Thus a chi-squared test may be of the form:

Set Chi-Squared Test:
$$l_k(i) = \sum_{setofdirections,j} \varepsilon'(j) Y^{-1}(j) \varepsilon(j) \tag{66}$$

An example set of the j would be: j = [1, 5, 6]. This case corresponds to the detection of failures for the actuators described in Table 8.

This method provides a fast yet robust method of isolating the failures along with fault detection. The Set Chi-squared test can correctly identify the failure actuator as long as there are not two or more thrusters aligned in the same

direction. For example, if a spacecraft had two actuators such as [Table 8], it would not be able to distinguish between which of the two thrusters actually failed by having only the Set Chi-Squared test.

	x-dir	y-dir	z-dir	θ-rot	ϕ-rot	φ-rot
Actuator 1	3 m/sec	0	0	0	-2 rad/sec	1 rad/sec
Actuator 2	-1 m/sec	0	0	0	4 rad/sec	-2 rad/sec

Table 8 Example of two actuators that cannot be distinguished using the set chi-squared method

Even though actuators 1 and 2 had opposite directions, they still affected the same axes of movement. Since the chi-squared test is direction blind due to the residual term multiplying by itself (eqn 66), it is impossible to know which actuator failed in the above example using only the Set Chi-Squared test. One method of resolving this difficulty is to increase the estimation filter's sensitivity to new measurements for the actuators in question once a failure is detected. This can be achieved by increasing the covariance of the estimates and waiting for the estimates to converge to the new values. If possible, it is desirable to be able to fire the actuators in question alone for a couple of time steps to speed up the convergence time. But if the actuators failed in the middle of an experiment, or satellite maneuver then it is not likely that one would have the luxury of using the actuators in question to speed up the estimation process.

It is then desirable to form a detection test that will automatically isolate the failure without relying on the slower filter estimate process. The following section describes a much more robust detection process that not only is as sensitive to the failures as the generic chi-squared tests, but can also isolate the failures to specific actuators.

5.2.3 Generalized Likelihood Ratio (GLR)

The GLR test was in part designed to overcome the shortcomings of the simpler chi-squared test and can be applied to a wide range of actuator failures with the ability to isolate the failed actuator by using knowledge of the different effects of such failures on the spacecraft. The GLR test also relies on the measurements and residual values of the Kalman filter based on the no failure model [Chapter 4]. The hypothesis of the test can be expressed in terms of the innovation/residual values:

GLR Hypothesis

$$\begin{aligned} Ho &: \varepsilon_k = \varepsilon_{ok} \\ H1 &: \varepsilon_k = \varepsilon_{ok} + G_{k;\theta}\upsilon \end{aligned} \tag{67}$$

Where ε_{ok} is the expected residual value without failures at time step k, G_k is the pre-computed failure signature (i.e. columns of the thruster mapper matrix), θ is the time of failure, which in our problem is always assumed to be the current step, k, and ν is the magnitude between 0 and 1. To determine if there was a failure in the last step, we must compute the maximum likelihood estimates (MLE) of ν and θ. From the MLEs, one can then compute the log-likelihood ratio for failure vs. no failure.

Since all the relevant densities of the problem are gaussian, the MLE of ν is [Ref. 16, Ref. 31]:

MLE of ν:

$$\hat{\upsilon} = S^{-1}(k)\chi(k) \tag{68}$$

Where S is deterministic:

$$S(k) = G^T(k)Y^{-1}(k)G(k) \tag{69}$$

and χ is the linear combination of the residuals:

$$\chi(k) = G(k)Y^{-1}(k)\varepsilon(k) \tag{70}$$

v can be interpreted as a least squares estimate of the failure magnitude assuming that the failure happened on step k and that we have no a priori information about the value of v. Furthermore, S^{-1} can then be viewed as the error covariance of the estimate of v.

It follows then that the generalized log likelihood ratio for the decision rule is [Ref. 31]:

MLE of θ=k:

$$l_k = \frac{\chi_k^2}{S_k} \tag{71}$$

The decision rule for the hypothesis test (eqn 67) is then:

GLR decision rule:

$$l_k \begin{array}{c} H1 \\ > \\ < \\ Ho \end{array} \delta$$

where l_k has a chi-squared statistic and δ is a threshold value that is chosen to give the desired tradeoff between false and missed alarms. Typical values of δ depend on the values of the actuator's acceleration levels and would vary from spacecraft to spacecraft.

Since everything is linear, at each step one can detect for multiple failure signatures, G, where each one is the effect of an actuator on the spacecraft. For example if a spacecraft had four actuators (Table 1 in section 2.2.1) then there would be four separate GLR tests and each G would correspond to the actuator values in the mapper (columns of Table 1 in section 2.2.1). Since this test was designed for abrupt failures that occur one at a time, the largest likelihood ratio will be compared against the threshold.

Multiple Failure Signature GLR Test:

$$\max_{i=1..n}(l_k(i)) \begin{matrix} H1 \\ > \\ < \\ Ho \end{matrix} \delta$$

This will guarantee that only one actuator is declared to have failed at any given time step. This is a useful analysis especially when multiple actuators have similar effects on the spacecraft. In that case, one failure will likely lead to multiple likelihood ratios, *l*, that are above the given threshold, δ.

5.2.4 Comparison of Fault Detection Tests

In this section, the fault detection methods described in the above sections – estimator threshold, chi-squared tests, GLR test - are applied to a sample spacecraft. The objective is to determine which method is best suited for small spacecraft missions where one has limited computation time and relatively noisy measurement systems. Three important characteristics are examined for the various detection algorithms – Robustness (detection/false alarm ratio), average time to detection and isolation after a failure has occurred, and the computation burden of the algorithms. There are two important timing issues for each method. The first issue is the raw computation time at each time step. An extremely robust system would be completely useless if it takes minutes or even seconds to compute at each time step. For example, on the ORION spacecraft mission, the control system time step is less than three seconds and all calculations must be done during that time period. The second important timing issue is the total time required for detecting a failure and isolating that failure to the right actuator. The GLR test should outperform all of the other algorithms in this department since it combines both the detection and isolation step in one. The chi-squared tests only identify possible failure suspects and one has to wait for the estimation to change significantly to isolate a failure. And the estimate

threshold should be slowest but the most robust. The robustness of a detection system can be measured in terms of detection rate vs. false alarm rate. As previously discussed, to increase detection rates, one must typically also tolerate a higher number of false alarms. The better method should always have a higher detection to false alarm ratio under any system properties such as measurement, system noise levels, Kalman filter values (R,Q values) and detection thresholds.

The sample spacecraft is modeled after the ORION satellite and it has twelve thrusters that will provide at least one redundant control thruster for each direction of the six degrees of freedom. Three different types of failures were applied to the tests – full on, full off and partial degradation. The results show that the GLR test is the superior choice. It always performed as well or better than any of the other algorithms in terms of its robustness to false alarms vs. detection rate, and vastly outperformed everyone in the time to detect and isolate a failure under various system properties. The GLR did have a higher computation burden, but it was still well within the system requirements (see Table 9).

The first set of tests compared the robustness of the various detection systems under various noise levels and system properties.

Detection vs. false alarm ratios for 75% degradation level (other types of failure had similar results):

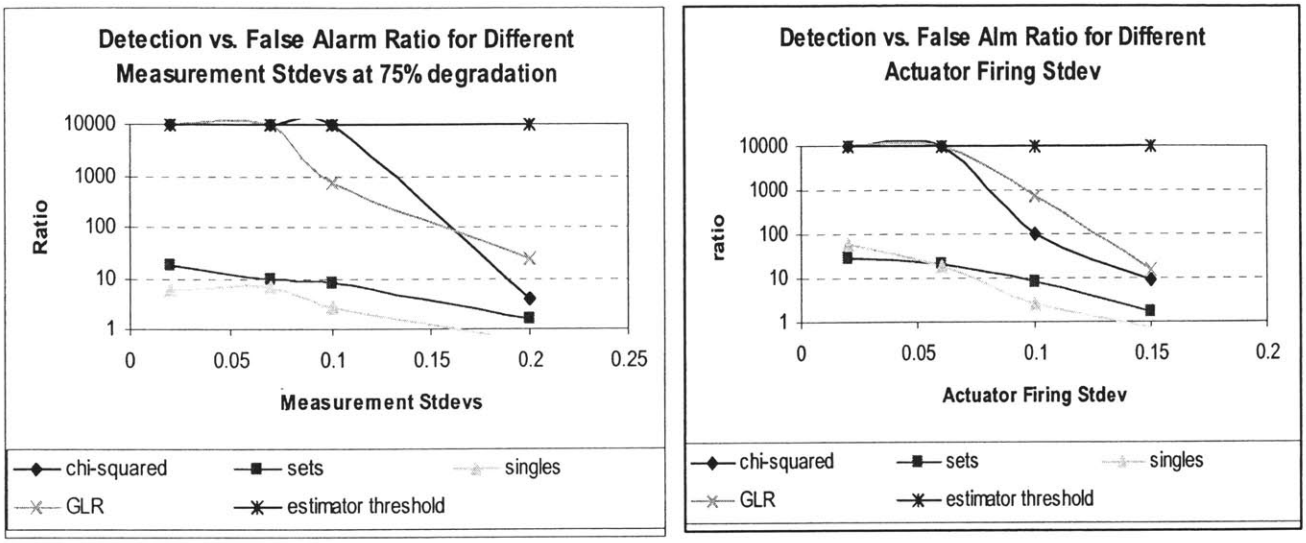


Fig. 14 Detection vs. False Alarm Ratios for different measurement and actuator firing standard deviations.

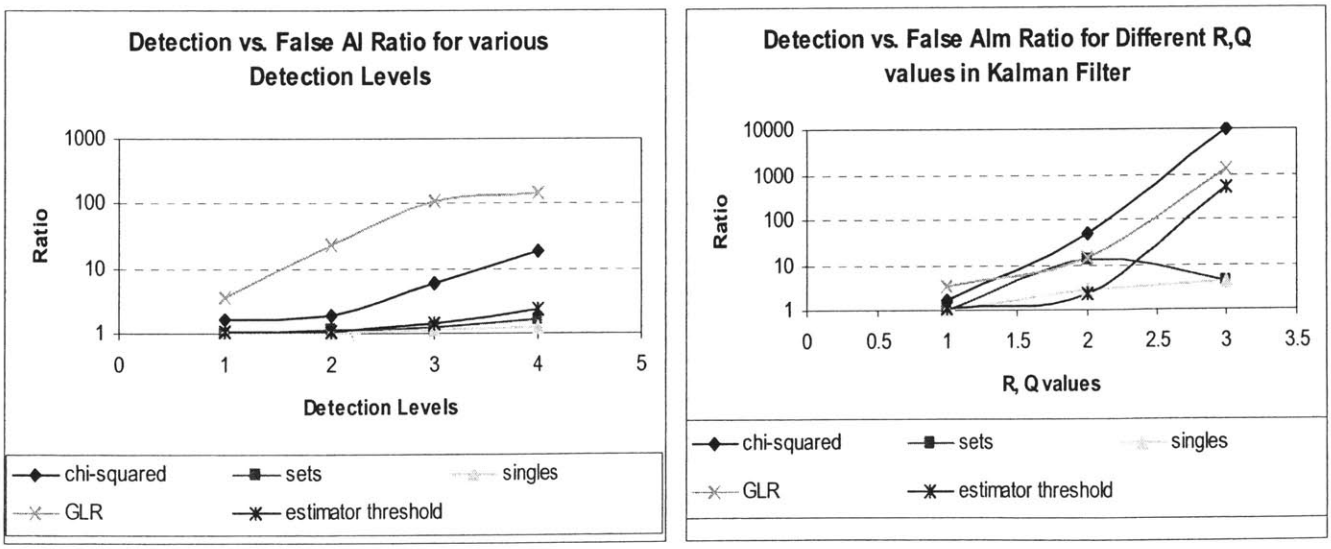


Fig. 15 Detection vs. False Alarm Ratios for different Detection Levels and R,Q values in the Kalman Filter.

Fig. 14 and Fig. 15 show the detection/false alarm ratio for the five different detection algorithm for a single actuator degradation to 25% of original level. In Fig. 14, the x-axis for both graphs are the standard deviation fractions for the measurement system and the actuator firing uncertainty. As expected, the detection rate/false alarm rate decrease significantly as the uncertainty in the system parameters increase. In the first graph in Fig. 15, detection/false alarm ratios were gathered for various detection levels for the different methods. In general, raising the detection threshold will increase the detection/false alarm ratios. But this comes at a cost of lowered detection rates for actual failures. The second graph shows that varying the R/Q ratio in the Kalman filter will also have a significant effect on the detection algorithms robustness. If one increases the R/Q ratio (an assumption of noisy measurement values), the algorithms tend to have a higher detection/false alarm ratio.

In general, the GLR method performed as well or nearly as well as any other method under all the test variations. Although the threshold on estimator method matched the GLR's detection/false alarm rate for different measurement and system noise levels, it lagged in terms of time to detection.

The second set of tests compared the time to detection and isolation of the failure after its occurrence.

Total Detection + Isolation time:

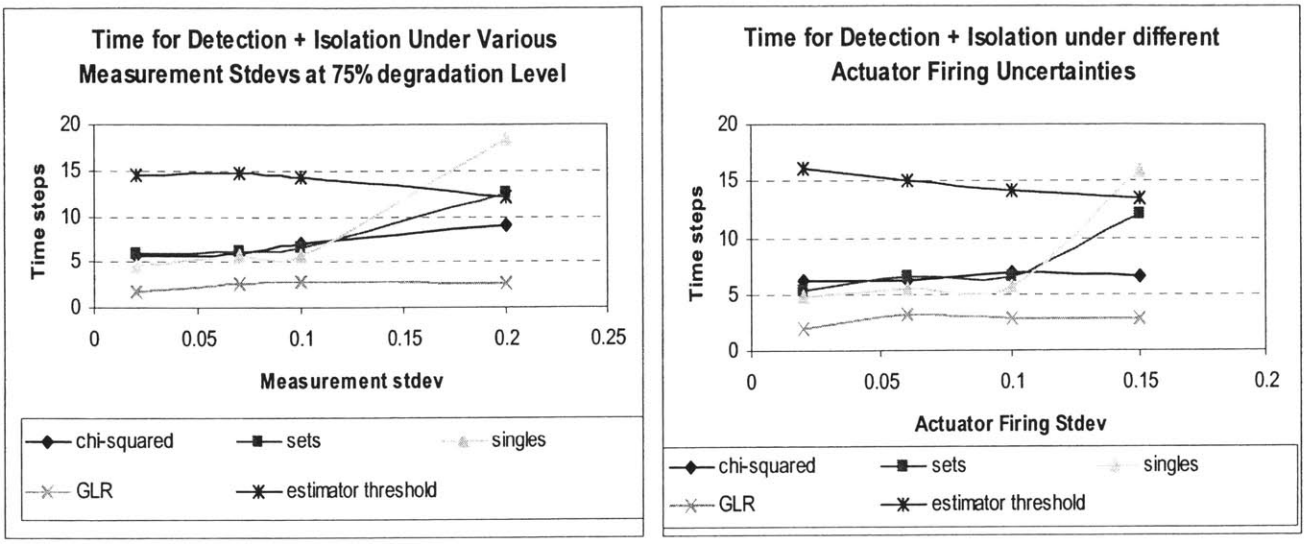


Fig. 16 Time to Detection and Isolation of failure for different measurement and actuator uncertainties

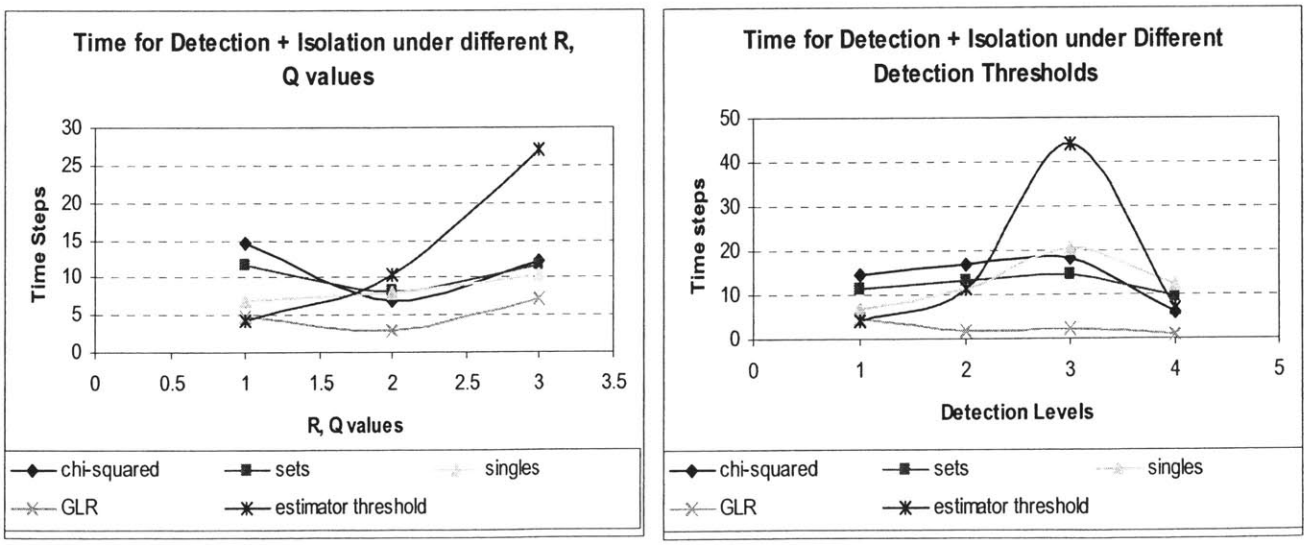


Fig. 17 Time to Detection and Isolation for different Detection Levels and R,Q values in the Kalman Filter.

Fig. 16 and Fig. 17 explored the time for each algorithm to successfully detect and isolate a single 75% degradation failure. The different conditions tested are the same as the graphs in Fig. 14 and Fig. 15. In general, the time to detection variable was determined to be not nearly as elastic as the detection/false alarm ratio. Overall, the GLR algorithm consistently outperformed the other methods by at least a factor of three and sometimes as high as ten or more.

As mentioned in the previous sections, the overall "goodness" of a algorithm is a combination of its detection speed and robustness in failure detection. The above figures clearly show that the GLR method is superior to the other algorithms under almost every variation in system and environmental conditions.

Finally, one must test to see if the GLR test is feasible in terms of the computation power available on small satellites. To test, the algorithms were ran on a Pentium class processor – the average processor used on current small satellites. While the GLR test does take relatively longer computation time than any other method, it is not significant because all the algorithms finished its computations in milliseconds:

	# of floating point calculations	~ time on pentium processor (msec)
generic chi squared test	1800.5	9.0025
single chi square test	1794.5	8.9725
set chi-squared test	1803.5	9.0175
GLR test	8202	41.01

Table 9 average computation times for various detection algorithms

The number of floating point calculations counts the total number of arithmetic operations that are on average performed in each algorithm. Each basic operation (addition, subtraction, multiplication, division) on any two real numbers counts as one calculation. The average total computation time on a Pentium-

class processor is indicative of average processing time on a current small satellite, since computing power on spacecrafts usually lag real-world computing standards by at least 4-5 years.

Modeling of a sample small autonomous spacecraft have shown that the GLR technique is far superior than the other methods in detecting and isolating actuator failures on spacecrafts that have limited computation power and limited measurement/on-board diagnostic capabilities. On average, the GLR method had a detection/false alarm ratio that was several magnitudes better than the other methods. Its detection speed was also consistently two to three times faster than its nearest competitor.

5.3 Fault Estimation and Compensation

Following the detection and isolation of the failed thruster, one would like to know how the thruster failed and be able to update that information back to the thruster mapper as fast as possible. Complementing the GLR detection method, several approaches have been developed to re-estimate the state once a failure has been detected. The different approaches involve directly adding the failure level, ν, to the estimates or simply increasing the covariance of the estimate filter or a combination of both [Ref. 31]. However, in many cases the GLR estimate, ν, may be quite inaccurate and simply updating the state estimate based on it could lead to instabilities [Ref. 31]. Intuitively, this increase in uncertainty can be reflected through a higher error covariance. But as shown above, waiting for the filter to converge through a higher covariance may take longer than desirable. Furthermore, under the above conditions, it is not possible to fully distinguish between a full "on" or full "off" failure. A full "on" failure is characterized by an actuator that is stuck in the on position while a full "off' failure is characterized by a thruster's failure to turn on.

Under these conditions, a new model comparison method is introduced here that has shown a great deal of robustness under various empirical tests.

5.3.1 A Model-Comparison Approach

The standard approach to adaptive reconfiguration in the event of a failure is an indirect method where the controller waits for the estimation to converge to the new value before updating the parameters. However, such an approach has been shown to be relatively slow and can handle only small to moderate uncertainties [Ref. 25]. Furthermore, for our particular problem of actuator performance identification with a high number of states to be estimated (number of actuators x degrees of freedom), the Kalman filter, estimation process can only be relied upon as an approximate calibration device. In fact, on most small spacecraft missions where one of the primary controller objectives is to conserve fuel, a thruster may not fire for more than one or two times in a row before a long period of rest. It would take an extremely long time for the estimation process to obtain enough information for the estimation to converge to the new correct values. Thus, a method is desired for immediately estimating the new actuator performance once a failure has been detected.

In this section, a model-comparison estimation process is described where multiple failure possibilities are applied to the identified failed actuator and the most likely of the set is chosen as the best failure estimation.

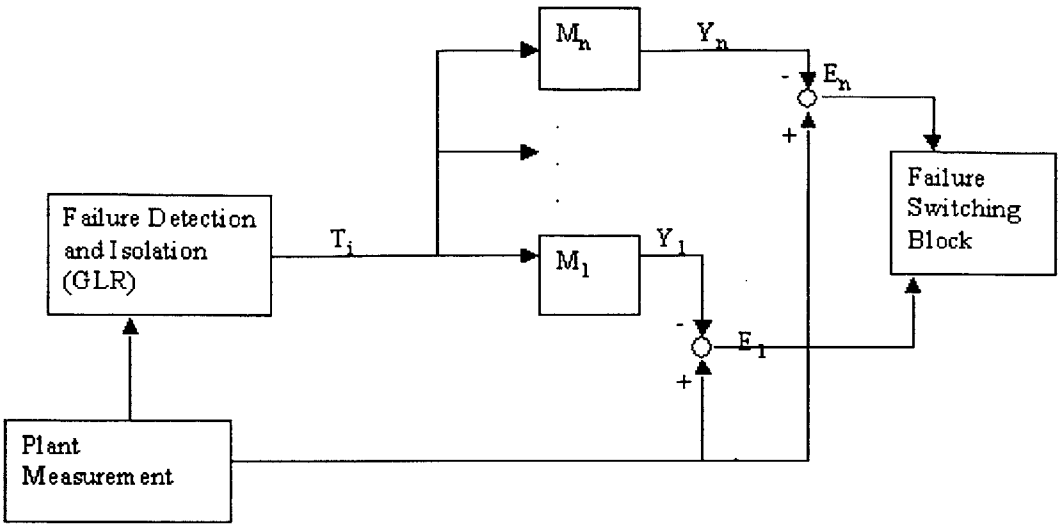


Fig. 18 Multiple-model failure estimation

In Fig. 18, T_i represents the failed actuator as determined by the failure detection and isolation algorithm. The *M*s are the various anticipated failure modes that may affect an actuator. The three most normal failure modes are: full on (actuator cannot be turned off), full off (actuator cannot be turned on), and partial degradation (thruster only operating at a percentage of its original level). Several models to describe the level of partial degradation of the actuator can be accommodated. For example, five different partial degradation models may be used where the first describes the case where the actuator has degraded in performance by 20%, the second 40% and so on.

To model a full on failure, one can assume that the actuator was turned on for the maximum on time during the last measurement period:

Full-on Failure Model: $$u(i) = max\_on\_time$$

Where *u* is the vector of on-times and *i* is the failed actuator in question.

Modeling full-off and partial degradations can be easily achieved through changing the known quantities of the actuator performance values within the thruster mapper matrix, *A*:

Full-off Failure Model: $$A(:,i) = 0$$

Partial Degradation model: $$A(:,i) = A(:,i)d$$
$$0 < d = degradationlevel < 1$$

The expected results from each of the failure models can be quickly calculated since that only depends upon the mapper matrix and the on-times of the actuators, *u*:

Failure Model Result: $$y = Au$$

The results from each failure model is then compared to the actual measurement results:

Model Residual: $$E = Z - y$$

Where Z is the actual rate changes recorded by the on-board measurement system. If one assumes that the characteristic of the model residuals are similar to the original residuals, ε, in the Kalman filter then the same chi-squared test using the filter residual variance as described in section 5.2.2 can be used to test for which failure model was the most likely:

Failure Model Likelihood Test: $$l_i = \sum_{j=1}^{n} E_i'(j) Y^{-1}(j) E_i(j)$$

Where l_i is the chi-squared value of the i^{th} failure model and *Y* is the residual variance from the Kalman filter. Note that even if the model residuals here do not have the same statistics as the filter residuals, it still makes sense to use the above chi-squared test using the residual variance from the filter. The chi-squared test is basically the norm of the residual with each term normalized by

the inverse of the residual variance. The running residual variance from the Kalman filter contains the variability of information of each residual, which over time is basically how noisy the system is in that particular degree of freedom. If the variance is high then one should discount any errors in that element and vice versa if the variance is low.

Once all the chi-squared values, *l*, are calculated from the different failure models, the model with the minimum l value is selected as the most likely failure estimate. If the failure is found to be a full-on failure then it would be desirable to completely turn off the thruster and notify both the main controller and the thruster mapper so that its usage will never be required. Otherwise, the failure was either a partial or full degradation and that can easily be updated to the system through the matrix mapper.

Chapter 6

Simulation on the ORION Testbed

6.1 ORION Mission Background

The main objective of the Orion [Ref. 33] mission is to demonstrate formation-flying capabilities using optimal autonomous control algorithms and precise relative-navigation based on carrier phase differential GPS (CDGPS). Slated for launch in mid-2004, the mission will have 3 spacecraft performing relative control using GPS signals (see Fig. 19):

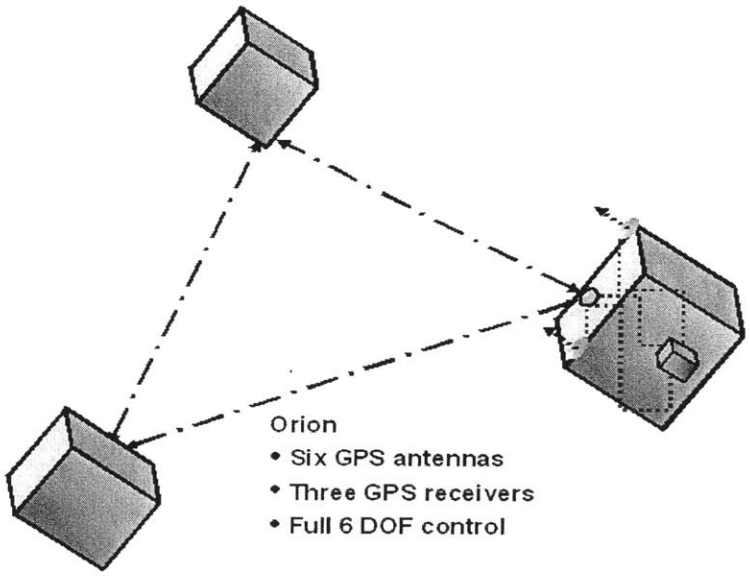


Fig. 19 Orion Formation Flying Mission

The twelve thrusters on each spacecraft provide fully redundant 6-DOF control. Relative position and velocity information between the spacecraft are calculated using the CDGPS solutions (see APPENDIX C for full Orion spacecraft statistics).

Optimal trajectories for station-keeping or formation-flying have been developed using a convex optimization method that can be solved via linear programming [Ref. 34]. There are several constraints inn the problem can including: fuel minimization, time to rendezvous, maximum acceleration, actuator rate limits, and differential disturbances such as drag and J2 effects. Given these constraints, the LP controller finds a set of rate/velocity changes at optimal points of the orbit that will take one spacecraft to a desired location in the given time period. Most of these control sequences are "bang-off-bang" meaning that the actuators are on at the beginning, off during cruise, and then on again during a final deceleration at the end. An example of an optimized trajectory that minimizes fuel usage is shown in Fig. 20:

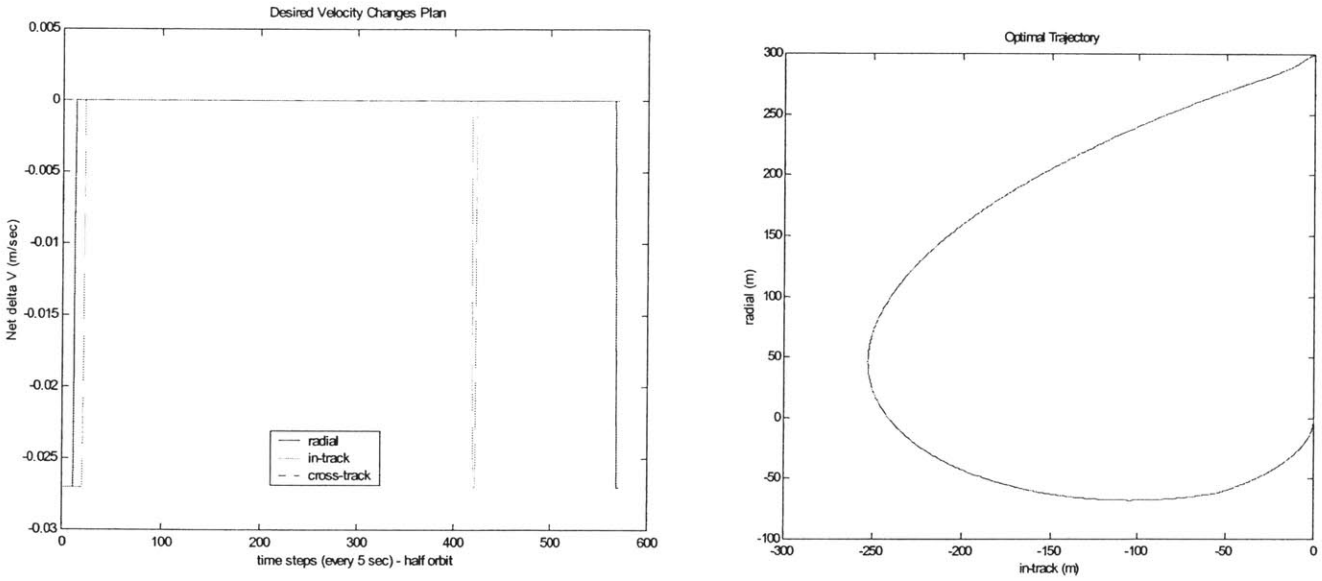


Fig. 20 Sample Optimal Trajectory

The simulation in Fig. 20 was done using the linearized Hill's dynamics (see section 6.2) and shows an optimal trajectory for one spacecraft to rendezvous with another (located at the origin). The initial separation between the spacecraft is 300 meters in the radial direction and the time for rendezvous is half an orbit.

In this chapter, the thruster mapper and fault detection algorithms developed in this thesis will be applied to the Orion spacecraft. Simulation results show that degradation effects from multiple thruster firings can drastically change the course of a planned maneuver if it is not properly controlled. The degradation compensation algorithm developed in section 2.4.2 is applied here with great success. Thruster failures are also shown to have mission-crippling effects if not properly diagnosed. Section 6.4 provides a detailed study of how the estimation and fault detection algorithms developed in Chapter 4 and Chapter 5 is applied to a real spacecraft mission. Finally, the simulations show improved control capabilities along with significant fuel savings when the estimation and fault detection algorithms are used along with the thrust mapper.

6.2 Hill's Equations

The Hill's equations of motion for any spacecraft are its relative movements relative to a target reference orbit, as shown in Fig. 21:

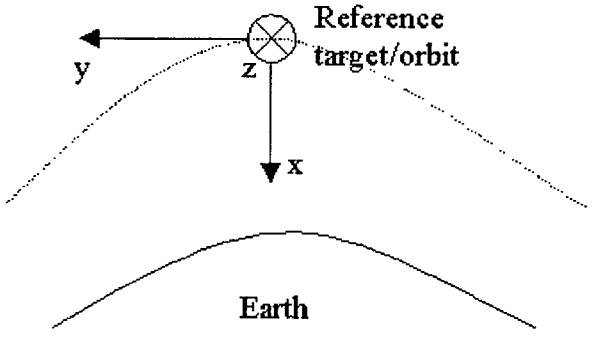


Fig. 21 Hill's (LVLH) reference frame

Where the x (radial) direction is measured along the radius vector to the center of the earth; the y direction (in-track) is measured along the velocity vector of the target orbit with positive direction corresponding to positive velocity; and the z axis (cross-track) is normal to the orbit plane, forming a right handed coordinate

system (Fig. 21). The Hill's reference frame will also be referred to as the Local Vertical Local Horizontal (LVLH) frame in the rest of the Thesis.

The continuous Hill's equations are derived from Kepler's Third Law (Ref. 35):

Hill's Equations

$$\ddot{x} = 2n\dot{y} + 3n^2 x + F_x$$
$$\ddot{y} = -2n\dot{x} + F_y$$
$$\ddot{z} = -n^2 z + F_z$$

Where *n* is the orbital rate or mean motion of the reference orbit and the *Fs* are the applied impulses.

Orbital Rate

$$n = \sqrt{\frac{\mu}{a^3}}$$

Where μ is the gravitational parameter of the Earth (3.986E5 km^3/s^2) and *a* is the semi major axis of the reference orbit.

The assumptions in Hill's equations are (Ref. 35):

- $\sqrt{x^2 + y^2}$ is small compared to the target orbit's radius
- The applied forces, F, are small.
- The target orbit has a small eccentricity (i.e. nearly circular orbit).

Since the spacecraft separation for the Orion mission will be relatively small compared to its orbit radius, the Hill's equations are a good approximation for simulation testing. The mean orbital rate is simulated at n=0.0011 which corresponds to an orbit altitude of 600 km – a typical low earth orbit.

6.3 Thrust Mapper and Degradation for Orion

Once the main controller has found the optimal rate changes for a certain maneuver, it is up to the thrust mapper (Chapter 2) to provide them with the available thrusters. Spacecraft properties including total mass and moment of inertias (APPENDIX C), plus ground testing of the propulsion system provide the needed knowledge of the acceleration available from each of the twelve thrusters on Orion (see Table 10).

	THRUSTERS											
directions	1	2	3	4	5	6	7	8	9	10	11	12
x-translation (mm/sec)	-0.003	0	0	0.003	0	0	0.003	0	0	-0.003	0	0
y-translation (mm/sec)	0	0	0.003	0	0	-0.003	0	0	0.003	0	0	-0.003
z-translation (mm/sec)	0	-0.003	0	0	-0.003	0	0	0.003	0	0	0.003	0
x-rotation (rad/sec)	0	-0.0417	0.0417	0	0.0417	-0.0417	0	0.0417	-0.0417	0	-0.0417	0.0417
y-rotation (rad/sec)	0.0412	-0.412	0	-0.0412	0.0412	0	0.0412	-0.0412	0	-0.0412	0.0412	0
z-rotation (rad/sec)	0.0449	0	-0.0449	0.0449	0	-0.0449	-0.0449	0	0.0449	-0.0449	0	0.0449

Table 10 Orion's Thruster Mapper

The twelve-thruster configuration on the Orion spacecrafts allows it to have redundant control in all directions. In fact, this configuration guarantees full 6-DOF control even with one full failure to any of the jets.

As discussed in section 2.4 of this thesis, ground testing of the Orion propulsion system displayed significant degradation if multiple thrusters are turned on at the same time. The average degradation for every extra thruster turned on is 6% (Fig. 4). Applying this degradation to the example given in Fig. 20, one can appreciate the magnitude of its effects. In the first time period, the master controller issues a rate change requirement:

$$v\_DESIRED = \begin{bmatrix} -0.027 & -0.027 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note, since no attitude data were simulated, it was assumed that the spacecraft started out where the control frame (LVLH) was aligned perfectly with the body frame. Any subsequent velocity change requirements also included a zero attitude change constraint. Using the known thruster acceleration levels, Table 10, and the LP mapper (Ax = b), the optimal on-times that minimizes fuel usage is:

$$\text{ON\_TIMES} = \lfloor 4.5 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 4.5 \;\; 0 \;\; 0 \;\; 0 \;\; 4.5 \;\; 0 \;\; 4.5 \rfloor \text{seconds}$$

With all four thrusters turned on at once, the total degradation would be approximately 18% (3 X 6%). Thus, instead of acquiring the desired rate change, *v_desired*, the actual velocity change in that time period is only:

$$v\_actual = \lfloor -0.002214 \;\; -0.02214 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \rfloor .$$

Because of this degradation, the actual trajectory of the spacecraft is completely different than the planned maneuver. Fig. 22 shows the actual trajectory of the spacecraft due to losses from multiple thruster firings:

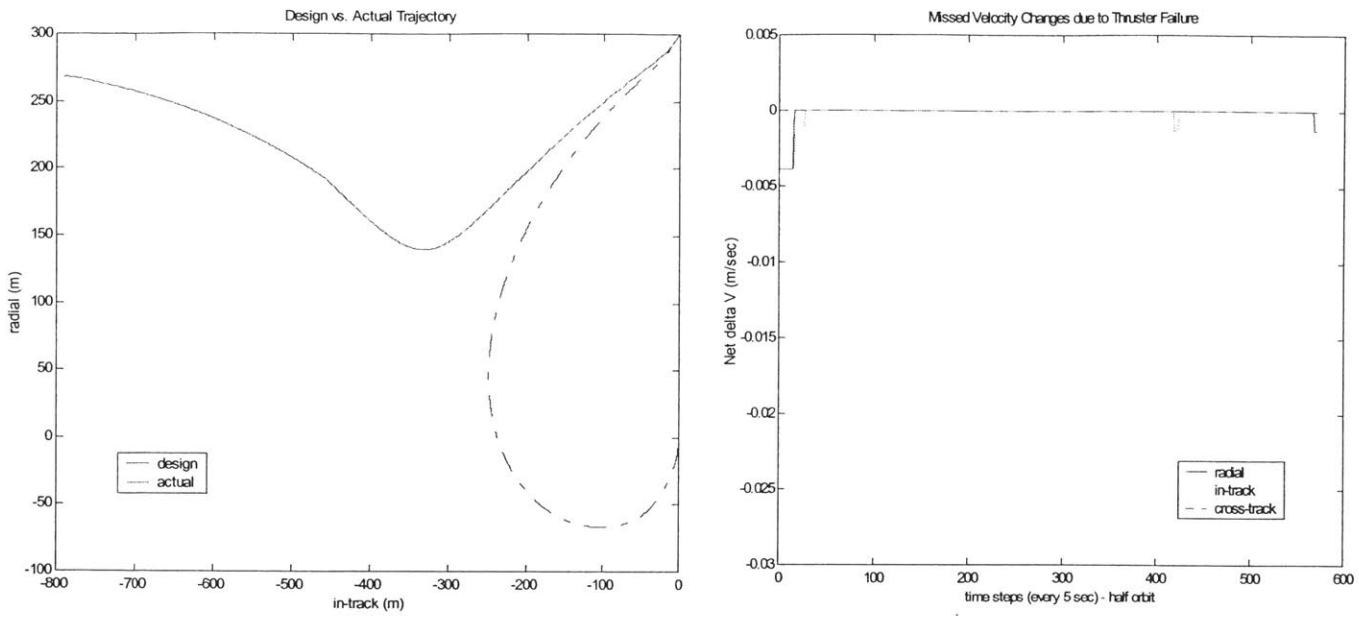


Fig. 22. Actual vs. design trajectory + lost acceleration due to multiple thruster firings

Even in this example, where a small number of thrusters (4) are needed to satisfy the required velocity changes, the losses due to multiple thruster firings are large enough to completely alter the course of the spacecraft. However, the iterative thrust mapping solution as described in section 2.4.2 can be used to fix this problem. The mapper is set to continuously increase thruster firings until the total thruster output after degradation is within 0.1% of the desired acceleration. Fig. 23 shows the results of the rendezvous with the iterative mapping scheme. Because of performance degradation, more fuel usage is required to satisfy the accelerations needed in the maneuver.

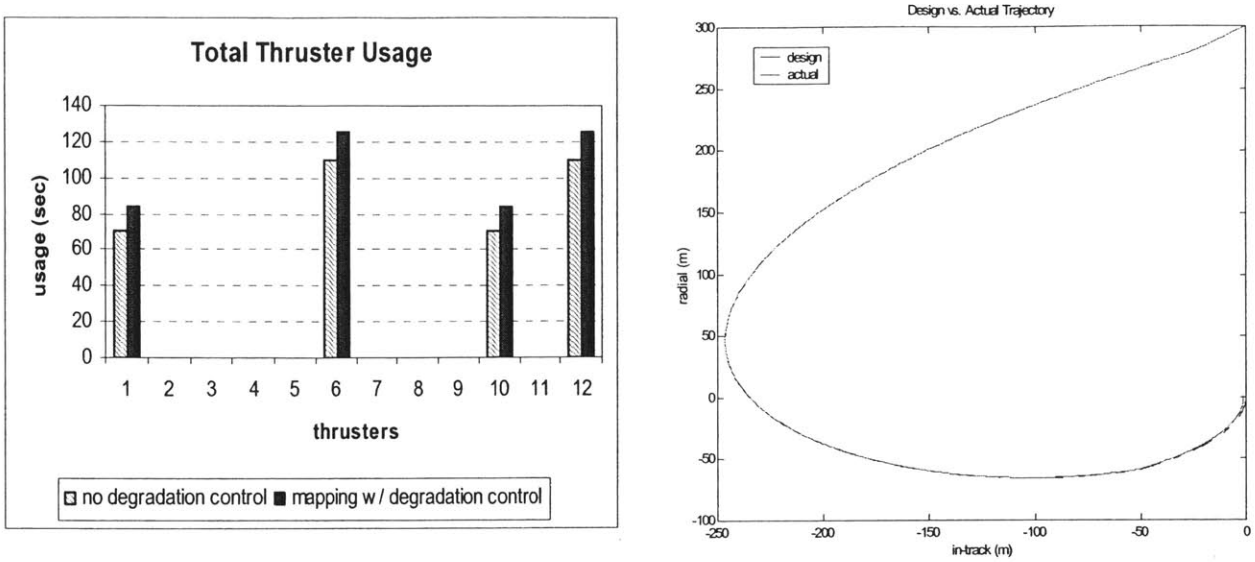


Fig. 23. Thruster Usage with Degradation Mapper Algorithm and resulting trajectory

Overall, when taking the multiple-thruster-firing degradations into consideration, it required the thrusters to be turned on for approximately 16% longer than before. Most of additional thruster usage is in the early stages of the flight, where both radial and in-track accelerations are required (see Fig. 20 and Fig. 22). All four thrusters were used in that period, resulting in 18% degradation. The last two firings only had two thrusters on at the same time to provide acceleration in one translational direction (no rotations), which would correspond to approximately a 12% thrust shortfall.

Besides thruster degradation, uncertainties are also present in the Orion thruster mapping problem. In the Orion experiment, all attitude information will be provided by the GPS system. If the control reference frame (LVLH) is not lined up with the body-frame, a transformation must be done based on the attitude information. The basic transformation equation is:

$$b_{body} = R_{obody} b_o$$

where b_o is the desired velocity changes in the control reference frame and R_{obody} is the rotational matrix to translate b_o into the body frame. Uncertainties in the

CDGPS measurements cause an extra rotation away from the true state. The error in the rotation would then show up as the uncertainty in the *Ax=b* LP problem:

$$b = R_{error} b_{body} = R_{error} R_{obody} b_o$$

Section 3.2 provides a detailed illustration of this process. Ground testing of the GPS attitude information shows an average uncertainty of $\pm 1^{\circ}$ in each euler angle with no extra bias towards any axis [Ref. 36]. This uncertainty leads to symmetric errors in the desired state, *b,* and under those conditions, solving the original Ax=b mapping solution gives the best robust solution to minimize the maximum possible error ($b_{body} - Ax$) (see section 3.2.1).

The other uncertainty in the thrust mapping problem is due to the random fluctuations in the thruster output. Ground testing show individual thruster firings on Orion can vary between $\pm 10\%$ (Fig. 4) but the deviations are independent for each thruster. Again due to the symmetry in the errors, solving the original Ax=b solution gives the best robust solution to minimize the maximum possible error ($b - A_{real}x$) (see section 3.3 for full a discussion).

6.4 Estimation and Fault Detection

Precise knowledge of the thruster performance and fast detection of failures will be a key to the performance of the Orion formation-flying mission. In addition to the thruster mapper, the estimation and GLR fault detection methods discussed in Chapter 4 and Chapter 5 will be applied. In summary, the estimation procedure will be a static Kalman filter estimating the thruster performance values given in Table 10. It is assumed that the properties of the thrusters are static and large errors between the measured and expected velocities can be attributed to a failure. Failures, whether full on/off or partial

degradation, are assumed to happen randomly and are unpredictable and independent. The GLR (Generalized Likelihood Ratio) detection algorithm (section 5.2.3) is used for detecting and isolating a failure, while the model-comparison approach (section 5.3.1) is used for determining the exact type of failure.

6.4.1 Orion Spacecraft Operation Details

When a particular maneuver is desired (i.e. move one spacecraft to 500 m away from the "master" satellite in one orbit time), the controller pre-determines the optimal acceleration level for each time period for the "slave" craft to achieve the desired state. Fig. 24 summarizes the activities involved for each time period (5 seconds):

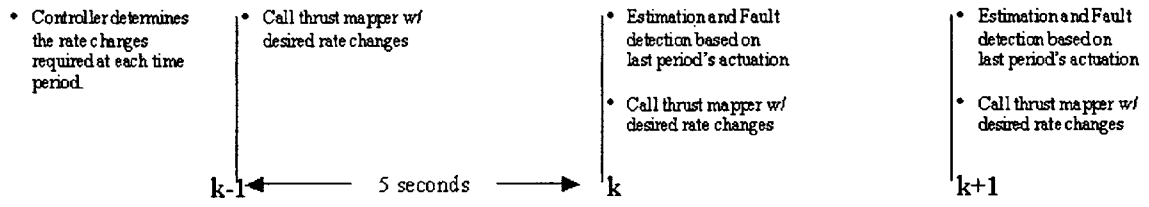


Fig. 24. Orion operations procedure for each time period

At the beginning of each time period (except the very first one), the estimation and fault detection procedure uses the CDGPS measurements to determine the velocity changes due to last period's actuation. Each Orion spacecraft is equipped with six GPS receivers and three coupled receivers for CDGPS measurements. By measuring the Doppler shift in the signals received between two spacecraft in formation, it is possible to solve for the relative positions and velocities of the two vehicles [Ref. 38]. Current ground testing show that using CDGPS, relative velocity and position measurements are accurate to ± 0.5 mm/sec and 2 cm respectively [Ref. 43].

On the Orion mission, like other formation-flying or station-keeping operations, a typical maneuver involves one master spacecraft that remains stationary in its orbit while the slave satellite(s) move into the correct relative position. In this scenario, the performance of the thrusters on the "slave" spacecraft is directly measured by the relative velocity data between it and the master satellite:

Velocity change due to actuation: $$V_{thrust} = V_k - V_{k-1}$$

Where V_k and V_{k-1} are the relative velocity measurement between the "slave" and the master spacecraft at each step and the difference, V_{thrust}, is due to the thruster firings of the slave satellite on step k.

6.4.2 States of Orion

The Kalman filter is used to estimate all 72 states of the thruster mapper (Table 10). Due to the high number of estimation states, relative to the low number of measurements, 6, the estimation/Kalman filter process is designed to have a long response time for more smoothing of the measurement and system noises. This is achieved through a high R/Q (10/0.01) ratio. Having a relatively larger noise value for the measurements, R_k, vs. the system noise, Q_k, makes the estimation more smooth and less responsive to random noises but at the expense of a slower response to failures. The estimation process thus will be used more as a monitoring system for long-term changes in the system characteristics while the GLR test is used for detections of abrupt failures.

While it would be desirable to have a fast response in the fault detection algorithm, setting the detection threshold too low would likely result in a high number of false alarms (see section 5.2.1 for full discussion). On the Orion spacecraft, since there is only one thruster redundancy in each direction, losing even one thruster to a false alarm would greatly reduce its capabilities.

Therefore, the first requirement for the threshold value would be an extremely low false alarm ratio. Fig. 25 shows the percentage of false alarms under different system characteristics. Note, the expected thruster firing error is ± 10% and the measurement standard deviations are 0.5 mm(deg)/sec.

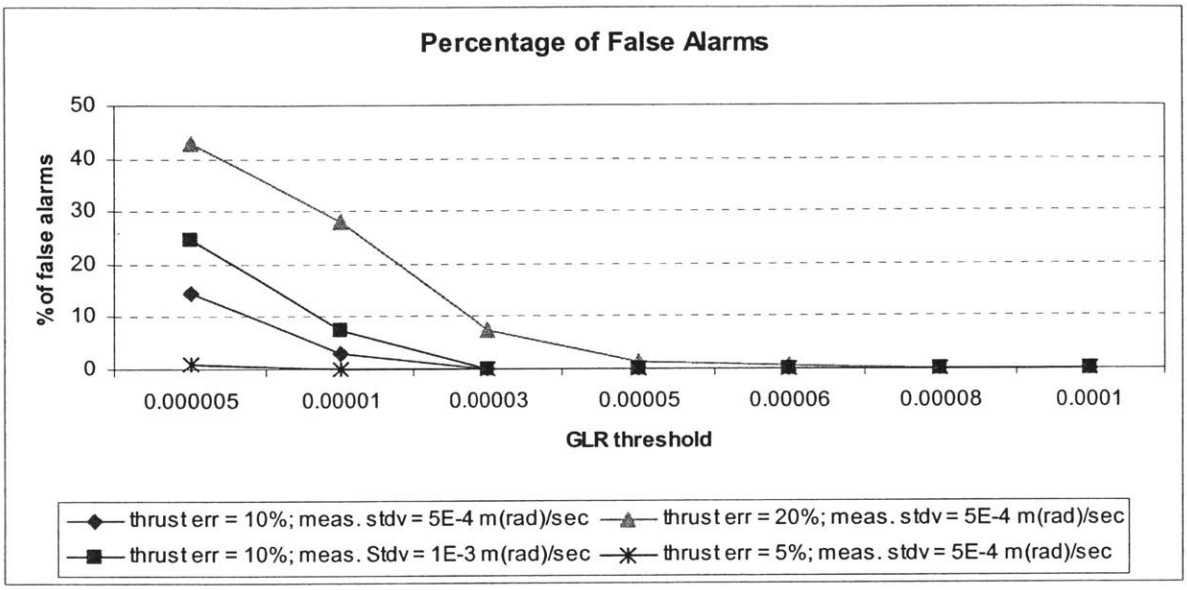


Fig. 25. Percentage of false alarms as a function of GLR threshold for Orion

Data for Fig. 25 were created by running 100,000 randomly generated thruster firings. For the default case of ± 10% thruster firing error and 0.5 mm/sec measurement standard deviation, a GLR threshold of over $3x10^{-5}$ guarantees no false alarms. Simulations also showed that the number of false alarms were also more sensitive to the fluctuations in the thruster firing than the measurement errors. Doubling the thruster firing standard deviation (▲ vs. ♦) caused much more false alarms than doubling the measurement errors (■ vs. ♦). The GLR threshold level of $5.5x10^{-5}$ was selected for the Orion spacecraft because no false alarms were detected in any of the simulations at that level.

The next set of simulations was designed to determine the probability of detecting a real failure at the $5.5x10^{-5}$ threshold level. Several factors can affect the detection of a failure – the type of failure (full-on, full-off or partial

degradation), the on-time of the failed thruster, and the failed thruster on-time as a percentage of the total on-times for all the thrusters during that period. Fig. 26 and Fig. 27 show the detection probability distribution for a full-on failure and a partial degradation. Fig. 26 shows that using a threshold of 5.5×10^{-5}, the GLR algorithm can detect a 50% degradation virtually 100% of the time if the failed thruster's on-time is at least 1.5 seconds and it is 10% of the total actuation on-time for all the thrusters combined in the control period.

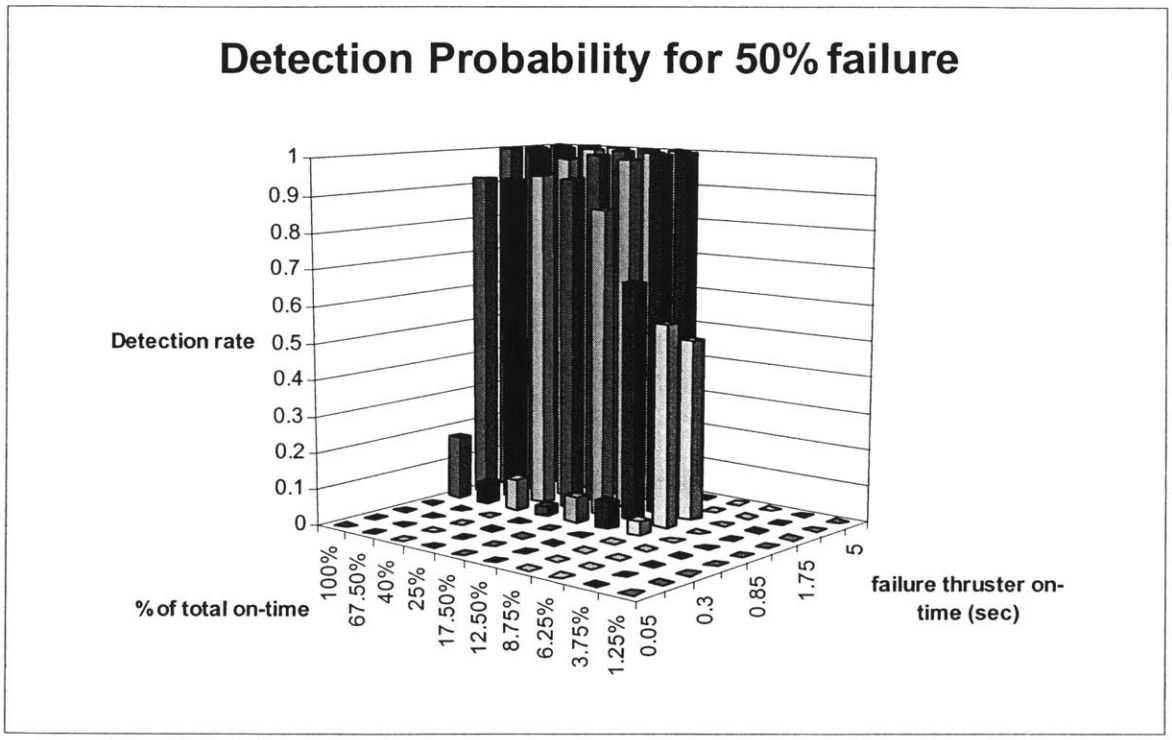


Fig. 26. Detection probability for a 50% degradation at GLR threshold = 0.000055. Note: It is impossible to have the case of a failed thruster on-time of over 1 second and occupying less than 8% of total on-times.

At below 0.8 seconds of on-time, it is virtually impossible to detect the failure because the failure signature would still be within the uncertainty in the thruster firing and measurement errors. Similar graphs for full-off failure and partial degradations to 25% and 75% of original levels can be found in APPENDIX D. Low detection levels for short thruster firings are not a big concern because for

most optimal trajectory maneuvers, the thrusters tend to have a bang-off-bang or full-on, off and full-on profile (see Fig. 20 for example).

A full-on failure is when a thruster becomes stuck on when it should be off. Assuming the thruster is stuck on for the full time period of five seconds, simulations show that the fault detection algorithm will correctly identify the failure every time at a GLR threshold of 5.5×10^{-5} (see Fig. 27).

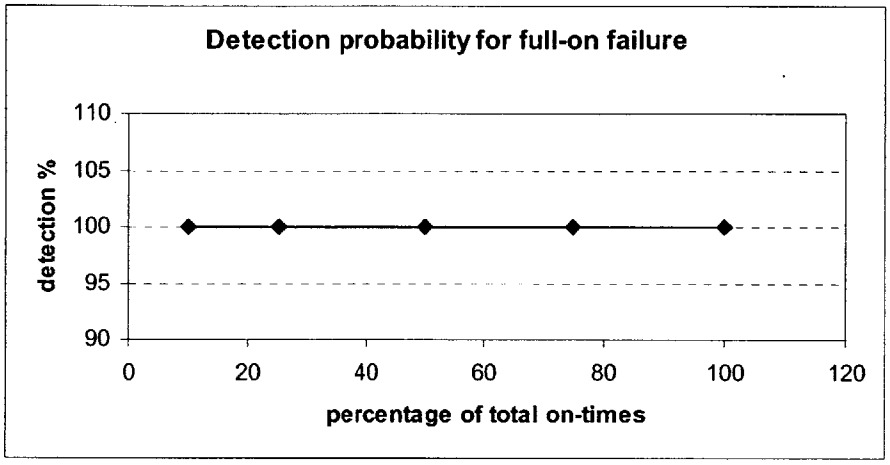


Fig. 27. Detection rate for a single thruster full-on failure at GLR threshold = 0.000055

6.4.3 Simulation Results

The full benefit of the fault detection algorithms is readily apparent when we compare orbit maneuvers with and without detecting a thruster failure. When a failure is detected, the thruster mapper is updated and the master controller re-plans the accelerations needed at every time period with the updated information. Since the smallest of deviations from the planned accelerations can cause large deviations from the planned maneuver, an undetected thruster failure will have large effects on the planned trajectory. Using the orbit maneuver example given in section 6.1, the slave satellite is required to rendezvous with the master spacecraft in half an orbit when its initial separation is 300 meters. In the first 10-15 time periods, full thrust power is required in both the in-track and

radial directions. If during the fifth step, thruster one, which provides half of the needed in-track acceleration, suddenly fails and no compensation plan is made, the slave craft will be further away from the master satellite at the end of the half orbit than when it started (see Fig. 28).

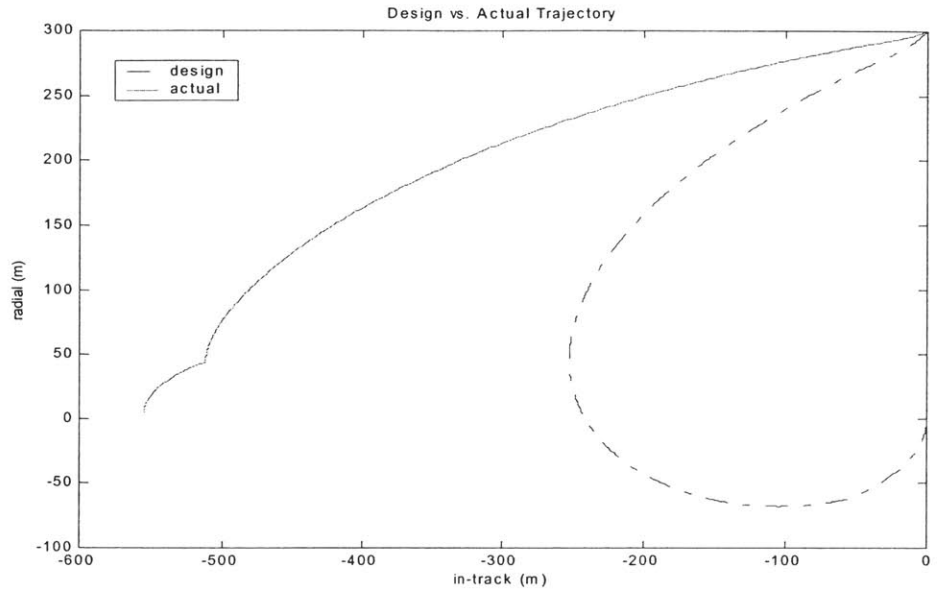


Fig. 28. Actual vs. designed trajectory with one thruster 50% failure in the radial direction

One possible solution method, which does not require failure detection, is to re-plan the design trajectory whenever the actual trajectory deviates significantly from the planned path. The results using this approach are shown in Fig. 29. The dotted lines are the design trajectories which are re-planed when the actual trajectory of the spacecraft deviates at least 25 meters from it. In all, over five total control re-plans were necessary to maneuver the spacecraft close to its final target.

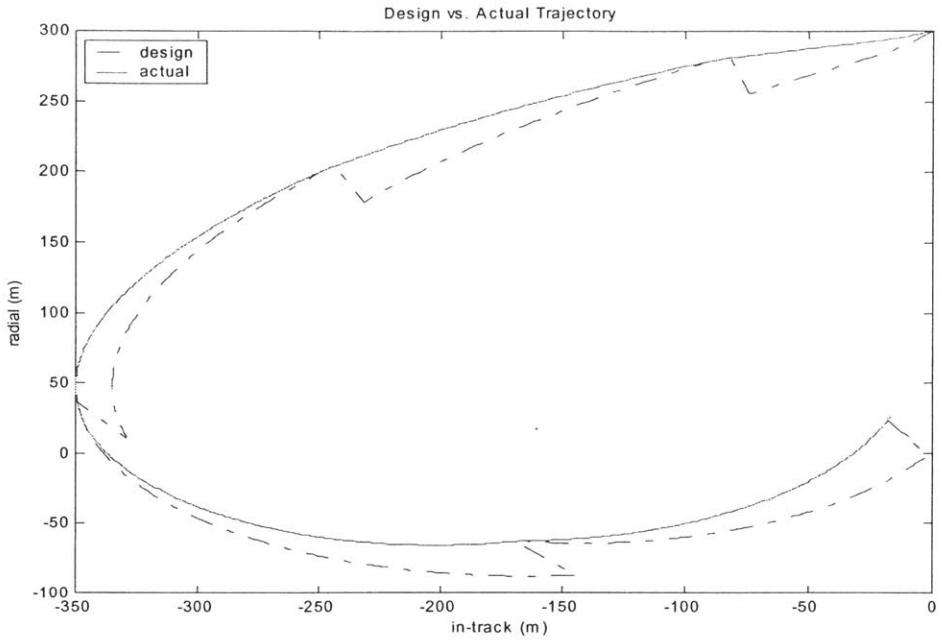


Fig. 29. Actual vs. designed trajectory with feedback control when actual trajectory deviates 25 m from planned path.

Redesigning the trajectory without identifying the root cause of the failure may help the spacecraft end up closer to the intended target, but at a cost of fuel usage.

By implementing the fault detection algorithm as outlined in this thesis, the thruster failure is detected when the measured velocity change is significantly different than the expected values (see Chapter 5). In this particular simulation, thruster one fails on step five when it was expected to be on for 4.5 seconds. The fault detection algorithm, running at 1 Hz, was able to isolate and determine the cause of failure after the first second of operation. Fig. 30 shows that the actual measured velocity/rate changes were significantly different than the expected rates from the known commanded on-times:

commanded on-times: $U=\lfloor 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1 \rfloor$ seconds

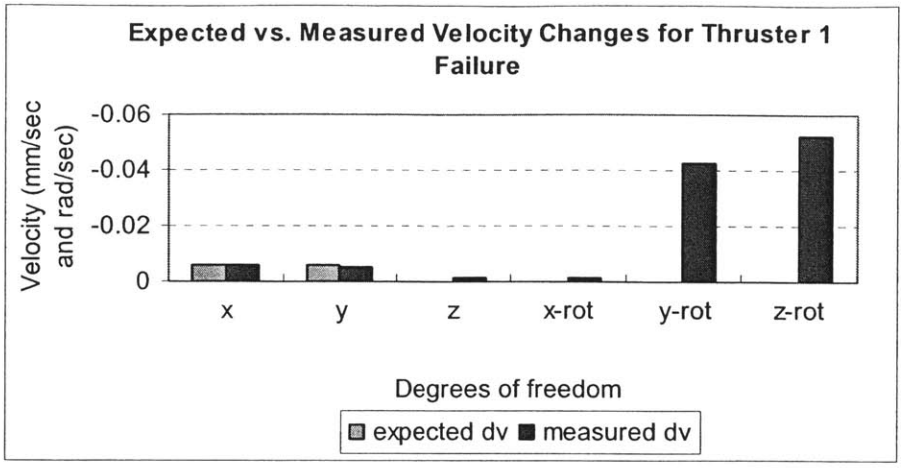


Fig. 30. Expected vs. Measured Velocity Changes from Actuator Firings with Thruster 1 failure (complete off).

The expected velocity changes are found by multiplying the on-times, *U*, with the current thrust mapper, *A* (see Table 10) and the measured velocities come from the CDGPS measurements.

The difference between the expected and measured velocity change is then used by the GLR failure detection algorithm (see sections 5.2.3 and 6.4.1) to determine if a failure has occurred. The GLR value for each thruster is the likelihood of that thruster failing with the given velocity differences. The thruster with the highest GLR value (thruster one) greater than the threshold is identified as the failure.

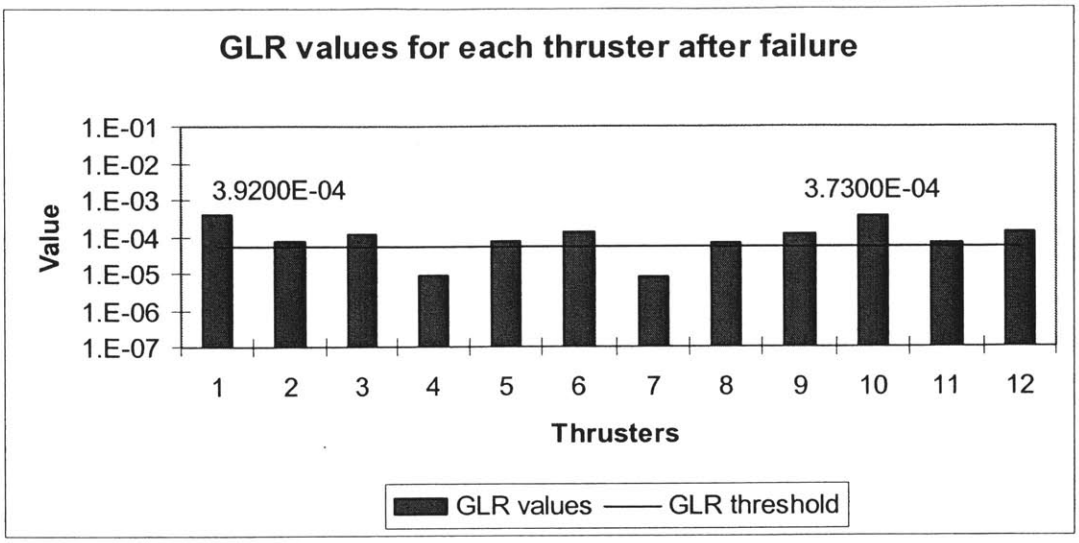


Fig. 31. GLR values for each thruster after failure to thruster one.

Thruster one and ten have close GLR values because their acceleration vector (see Table 10) are very similar. Although the difference between GLR values of two thrusters are very small, $2x10^{-5}$, it is statistically very significant because the average GLR value due to measurement uncertainties and no failures is one magnitude lower at $1x10^{-6}$. In fact, numerical simulations show that the GLR approach is accurate in isolating the correct failed thruster 99.9% of the times.

After the identification of the failed thruster, the model-comparison (MC) method (see section 5.3.1) is utilized to determine the exact type of failure. The MC method utilizes the chi-squared statistics on the likelihood of the failure corresponding to any of the given failure models. The true failure model corresponds to the one with the smallest chi-squared statistic.

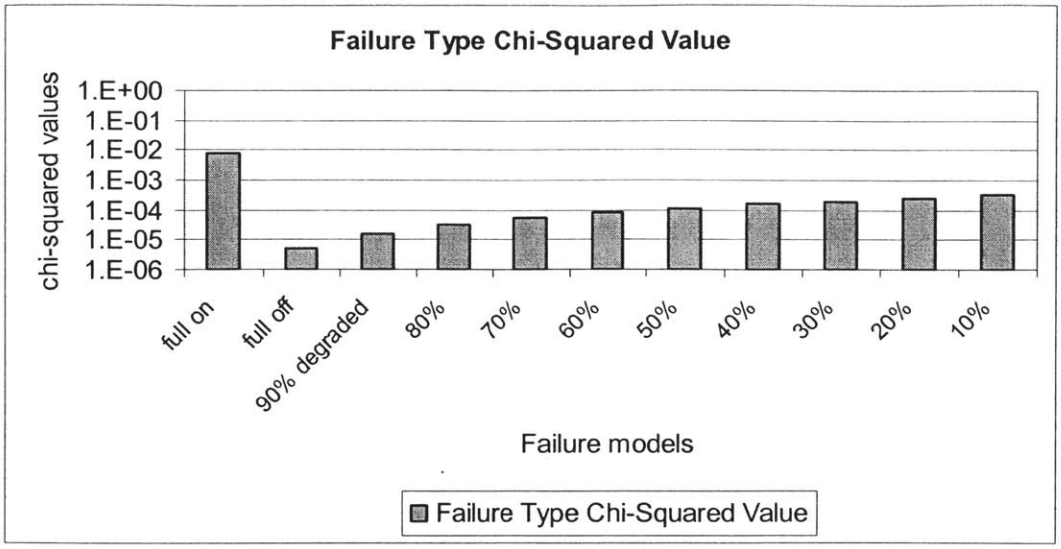


Fig. 32. Actual vs. designed trajectory

The full-off failure for thruster one is selected for having the minimum chi-squared value (see section 5.3.1). Once the failure magnitude is determined the thrust mapper is updated and the master rendezvous control re-plans with the new information (see Fig. 33):

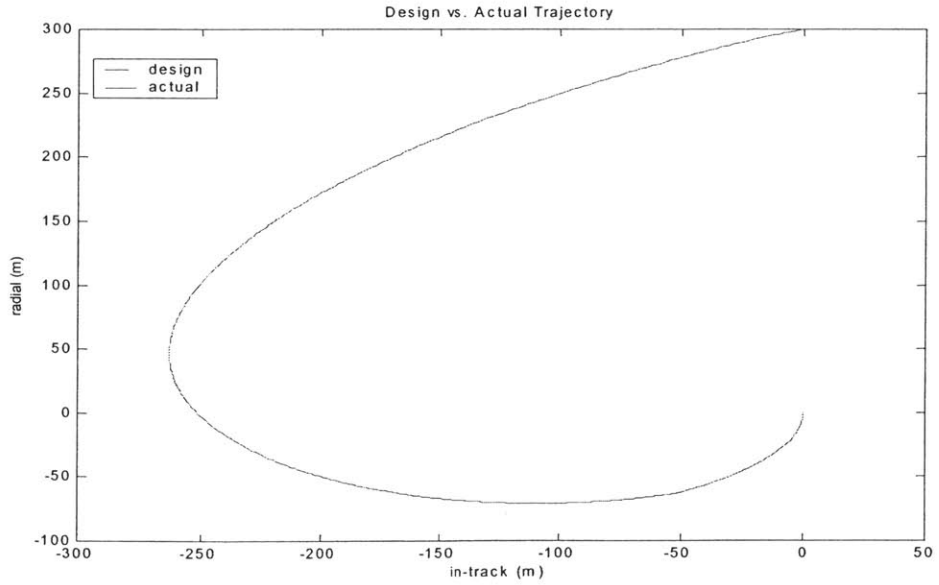


Fig. 33. Actual vs. designed trajectory with fault detection algorithm implemented

Not only does the new plan guarantee the successful completion of the rendezvous but it also saves fuel compared with making multiple re-plans without the knowledge of the actual failure:

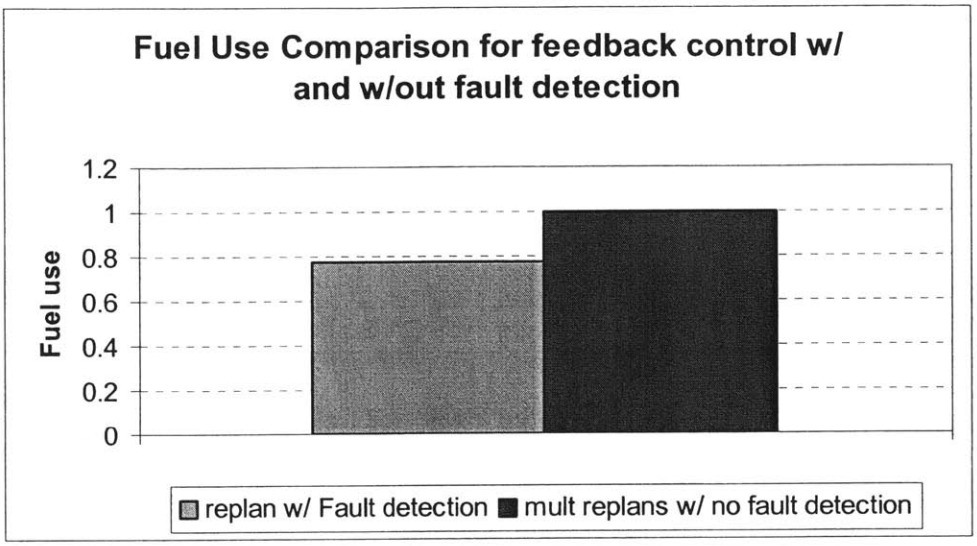


Fig. 34. Fuel savings from using fault detection to guarantee successful mission completion

Simulation results comparing other types of failures including multiple simultaneous thruster failures and partial degradations can be found in APPENDIX E.

6.4.4 Summary of Estimation and Fault Detection Equations and Variables for Orion

Estimation (Kalman Filter) :

State: $\underline{\hat{x}} = 72\text{x}1$ vector – from 12 thrusters and 6 DOF (72)

State propagation: $\underline{\hat{x}}_{k+1}^{-} = \underline{\hat{x}}_k$ (73)

Err covariance propagation:

$$P_{k+1}^{-} = P_k + Q_k \qquad Q_k = 0.01 \tag{74}$$

The update equations across measurements are:

State Update:

$$\hat{x}_k = \hat{x}_k^{-} + K_k \varepsilon_k \tag{75}$$

Covariance update:

$$P_k = (I - K_k H_k) P_k^{-} \tag{76}$$

Residual:

$$\varepsilon_k = Z_k - H_k \hat{x}_k^{-} \tag{77}$$

Kalman gain:

$$K_k^T = P_k H_k^T Y_k^{-1} \tag{78}$$

Residual covariance

$$Y_k = H_k P_k H_k^T + R_k \qquad R_k = 10 \tag{79}$$

Fault Detection Using GLR:

GLR Hypothesis

$$Ho: \varepsilon_k = \varepsilon_{ok} \qquad H1: \varepsilon_k = \varepsilon_{ok} + G_{k,i}\upsilon \tag{80}$$

Where *G* is the failure signature and can be any of the columns in the thruster mapper (Table 10).

GLR decision rule:

$$\max_{i=1..n}(l_{k,i}) \begin{matrix} H1 \\ > \\ < \\ Ho \end{matrix} \delta$$

MLE of θ=k:

$$l_{k,i} = \frac{\chi_{k,i}^2}{S_{k,i}} \tag{81}$$

$$\chi_{k,i} = G_{k,i} Y_k^{-1} \varepsilon_{k,i} \tag{82}$$

$$S_{k,i} = G_{k,i}^T Y_k G_{k,i} \tag{83}$$

Once a failure is detected, it is run through the model-comparison test to determine the exact type of failure (note, j = 1… total number of possible failure types):

Failure Type:
$$failure\_type = \underset{j}{\mathrm{MIN}}(l)$$

Failure Model Likelihood Test:
$$l_j = E_j^T Y_k^{-1} E_j$$

Failure Model Residual:
$$E_j = Z_k - y_j$$

Failure Model:
$$y_j = A_j u_j$$

Full off failure Model:
$$A_j(:,i) = 0$$
$$u_j = u_i$$

Partial Degradation:
$$A_j(:,i) = A_k(:,i) * degradation\_factor$$
$$u_j = u_i$$
$$0 \le degradation\_factor \le 1$$

Full on Failure:
$$A_j(:,i) = A_k(:,i)$$
$$u_j = 5 = \max\_ontime$$

Note: A_k is the thruster mapper model and u_i is the failed thruster on-time given by the thruster-mapping function at step k.

Chapter 7 Conclusions

New thrust mapper and estimation algorithms were developed in this thesis that can be used with on small autonomous spacecraft systems. The thrust mapper employs the Linear Programming methodology with additional improvements in its ability to deal with thruster degradation and system uncertainty. The estimation procedure utilized a static Kalman filter along with the generalized likelihood ratio (GLR) test to quickly identify and isolate any unexpected failures in the actuators. After the initial detection, a model-comparison (MC) estimation process is developed to accurately assess the type of failure experienced by the actuator.

These new algorithms are modular, adaptable to various types of spacecrafts and control systems and robust to uncertainties in the system. Both the thrust mapper and the estimation process require minimal computation burden and can be easily added to any autonomous spacecraft missions.

Simulations on SPHERE test bed using the thrust mapper only showed an improvement of over 15% in performance when compared with an older allocation method (section 2.3.1). When both the thrust mapper and the estimation algorithms were tested on the Orion spacecraft test bed, dramatic improvements in both fuel usage and control robustness were achieved. Unexpected thruster failures were routinely identified and corrected in the mapper within five iterations (section 6.4.3).

APPENDIX A

Full Simplex Upper Bound Algorithm – Matlab version

```
function [J, F] = simplex_upbnd(W, A, c, K,u)
%
% This implementation of the Simplex linear programming algorithm
% finds the fuel optimal set of jets and on-times for the commanded rate change.
%
% other functions called:
% xsign.m
%
% Output
% J - m-vector of jet identifiers
% F - m-vector of jet on-times
%
% Input
% W - m-vector of commanded rate change
% A - m x n matrix whose columns are the activity vectors of the n jets
% c - n-vector of costs associated with each jet
% K - a large cost
% u - n-vector of upper bnds
%
% Local
% Y - matrix representing the linear comb. of each jet vs. the basis
% y - local vector indicating the rate of change of the basic decision var. due to the value of
the invited var.
```

```
% z   -          cost of gradients of replacing the basis jets
% u_b  - upper bound of the variables in the basis
%        J_nb -  n+m-vector of jets not in the basis (0-not in soln, 1-at boundary)
% case2 - boolean for if the current step corresponds to case #2
% m   - # of thrust directions
% n   - # of thrusters
% lctr -  local counter of how many times we've gone through the loop
% zmax -  current evaluator maximum
% imax -  current index of zmax
% d_p  - pivot ratio
% d_u  - upper bnd ratio
% d    - pivot/upper bnd ratio used in exchange
% j_p  - index of smallest pivot ratio
% j_u  - index of smallest upper bnd ratio
% jex  - index of basis to exclude
% i, j -  loop counters
% x    - temporary ratio memory for comparison to d_p or d_u
% temp -  for storing temporary variables
% j_out - index of basis going out (for case 2 only)



case2 =0;
[m,n] = size(A);


% setup the initial solution
Y = diag(xsign(W))*A;                % linear combination coefficients
```

```
z = K*ones(1,m)*diag(xsign(W))*A - c';    % evaluators
J = [n+1:n+m]';                  % jet list in the basis
J_nb = zeros(n+m,1);                                    % jet list not in the
basis (either zero or at upper bnd)
F = abs(W);                      % on-times
u_b = 1E6*ones(size(W));                                % set high upper-bnds
for slack var. (curr basis)
u = [u; u_b];


% loop until the evaluators are non-positive
lctr = 1;
[zmax, imax] = max(z);
while(zmax > 0.0 & lctr < 3*m+1)
   % find the basis vector to replace
   y = Y(:,imax);
   d_p = inf;  % pivot ratio value
   d_u = inf;  % upper bnd ratio value
   jex = 0;          % index of basis to exchange
   j_p = 0;          % index of smallest pivot value
   j_u = 0;          % index of smallest upper bnd ratio value
   % determine the best pivot
   % and best upper bnd index
   for i=1:m  % loop through all basic variables
     if y(i) > 0.0
       % the basic variable will go down (lower bnd)
       x = abs(F(i))/y(i);
       if x < d_p
```

```
      d_p = x;
      j_p = i;
    end
  elseif y(i) < 0.0
    % basic variable will go up (upper bnd)
    x = (F(i)- u_b(i))/y(i);
    if x < d_u
      d_u = x;
      j_u = i;
    end;
  end;
end;


% now determine how the incoming variable should be incorporated
% and how the basic variables should be dropped.
% case 1:  one basic variable goes to zero (dropped) while a non-basis
%                    variable gets added.
% case 2:  the non-basic variable enters the basis while the one it replaces
%                              goes to its upper bnd.
% case 3:     the non-basic variable goes to its upper bnd.
if ((d_u >= u(imax)) & (d_p >= u(imax)))
  % case 3
  % update the non-basis vector
  J_nb(imax) = u(imax);
  % update the on times for the jets in the basis
  F = F - u(imax)*y;
```

```
  % update the cost/evaluator vector & combination coefficient matrix
  Y(:,imax) = -Y(:,imax);
  z(imax) = -z(imax);

else
  if ((d_u >= d_p) & (u(imax) >= d_p))
    % case 1
    jex = j_p;
    d = d_p;
    % update jet boundary value for jet going out of the basis
    J_nb(J(jex)) = 0;
  elseif ((d_p > d_u) & (u(imax) >= d_u))
    % case 2
    jex = j_u;
    d = d_u;
    %update jet boundary value for jet going out of the basis
    J_nb(J(jex)) = u(J(jex));  % assuming slack var. will not goto upper bnd

                        case2 = 1;
  end;
  % update the solution
  F = F - d*y;
  % update jet vectors
  J_out = J(jex);
  J(jex) = imax;
  % update upper bnd vector
```

```
u_b(jex) = u(imax);



% check to see if incoming var is decreasing from upper bnd
if (J_nb(imax) ~= 0)
  F(jex) = J_nb(imax) - d;
  J_nb(imax) = 0;
else
  F(jex) = d;
end;

% transform the linear combination coefficients and evaluators
for i=1:n
  temp = Y(jex,i)/y(jex);
  z(i) = z(i) - zmax*temp;
  Y(jex,i) = temp;


  for j=1:m
    if j~=jex
      Y(j,i) = Y(j,i) - Y(jex,i)*y(j);
    end;
  end;

end;% end of for i=1:n

if (case2 == 1)
```

```
        % update the cost/evaluator vector & combination coefficient matrix
        % for the jet going out of the basis and at its upbnd
        Y(:,J_out) = -Y(:,J_out);
        z(J_out) = -z(J_out);
                                case2 = 0;
    end;


  end;% end of different cases of replacing the basis


  % find the maximum evaluator
  [zmax, imax] = max(z);
  lctr = lctr + 1;
end

% check to see if there are non-basic var that are at its upper bnd
ind = find(J_nb ~= 0);
J = [J; ind];
F = [F; J_nb(ind)];
```

APPENDIX B

```
function [thrusters_on_off, T_act] = thrust_mapper_deg(T_des, D, u_ub)


% This function is the iterative method II described in the Thesis for
% dealing with multiple-thruster on degradations.
%
% degradation factor = (num_thrusters_on-1)*0.06; based upon orion testing values



% variables
% output:
%  thrusters_on_off:  variable of on, off time percentage for each of the thrusters
%  T_act:  expected rate change from using the set of thrusters_on_off.


% input
% T_des: desired delta_v (6x1 - vel + w)
%  D:  mapper of each thrusters firing capabilities
%      m x n where m is # of acc directions + any torque (eg x,y,torq) and n is # of thrusters
%  u_ub: maximum on time


% local
%  num_t: number of thrusters
%  num_d: number of thrust directions
%  u_c:  cost of thruster use
%  degradation_factor:  degradation factor 6% per extra thruster on
%  count:  number of loops for convergence
```

```
% times:  all the on times of the thrusters, sorted in ascending order
% T_c;  this is the max rate change we can get without degradation
% J_u:  simplex returned thrusters in basis
% F_u:  simplex returned on-times for the thrusters in J_u



% first get the size of the dimensions
[num_d, num_t] = size(D);

% setup cost
u_c = ones(num_t,1);  % cost of each thruster use in nominal condition
K = 1000; % high cost for slack variables

% bounds on u (thrusters on off)
uub_x = u_ub*ones(num_t,1);

% setup LP loop values
thrusters_on_off = zeros(num_t,1);
diff = T_des;
last_diff = zeros(num_d,1);
degradation_factor = 0.06;
T_act = zeros(num_d,1);
T_c = zeros(num_d,1);
count = 0;
count2 = 0;
```

```
% loop until we cannot provide any more thrust (last_diff - diff ~ 0)
% or we are close to the wanted rate change,t_des, (diff ~ 0)
while ( (max(abs(diff)) > 1E-5) & max(abs(last_diff-diff)) > 1E-5 )
  count = count + 1;
        [J_u, F_u] = simplex_upbnd(diff, D, u_c, K, uub_x);
  % determine the degradation level due to multiple firings
  % go through all the thrusters and determine which ones are on at the same time

  % find all the on times
  % and add to pervious on times
  t_ons = find(J_u <= num_t);
  thrusters_on_off(J_u(t_ons)) = thrusters_on_off(J_u(t_ons))+F_u(t_ons);

  t_ons = find(thrusters_on_off>0);
  times = sort(thrusters_on_off(t_ons));

  %loop through all the times
  last_time = 0;
  count2= 0;
        T_act = zeros(num_d,1);
 for i = 1:length(times)
   if (times(i) ~= last_time)
     count2 = count2+1;
     curr_per = times(i) - last_time;
```

```
    % determine # of thrusters on in this period
    num_on = max(find(times >= times(i))) - (i-1);
                %last_time_num_t = max(find(times == times(i)));
    degrade = 1-(num_on-1)*degradation_factor;



    for z = 1:num_t;
                if (thrusters_on_off(z) >= times(i))
                if count ==1
          % reset W_c to what we can get maximumly with no degradation
          % note this is only done on the first iteration of the entire program
                T_c = T_c + D(:,z)*curr_per;
                end;
                        T_act = T_act + D(:,z)*curr_per*degrade;
                        end;
    end;



  end;
  last_time = times(i);
end;


% find difference
last_diff = diff;
diff = T_c - T_act; %difference between what we have now vs. what we can get maximumly
                                        % with no degradation
```

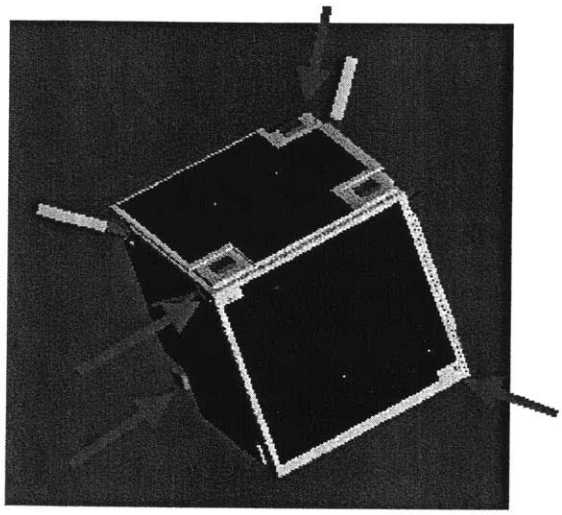
```
  % update the upperbnd vector and also the cost vector
  new_t_cost = inv(1-(length(t_ons) -1)*0.06);
  for i = 1:num_t
    uub_x(i) = u_ub - thrusters_on_off(i);
    if thrusters_on_off(i) == 0
      u_c(i) = new_t_cost;
    end;

  end;

end;
```

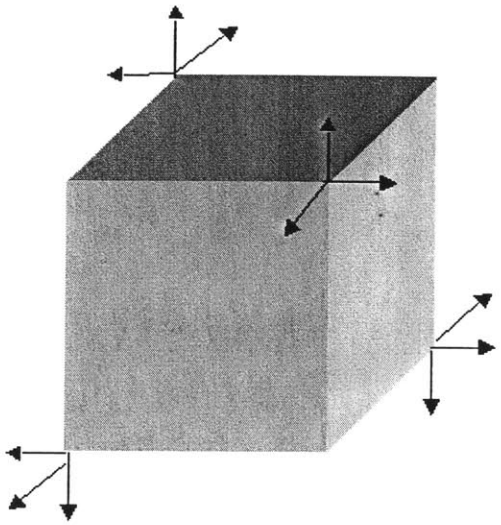
APPENDIX C

Orion Spacecraft Data

Propulsion System:

- Three axis control with 12 thrusters. 4 Clusters of 3 as shown below:

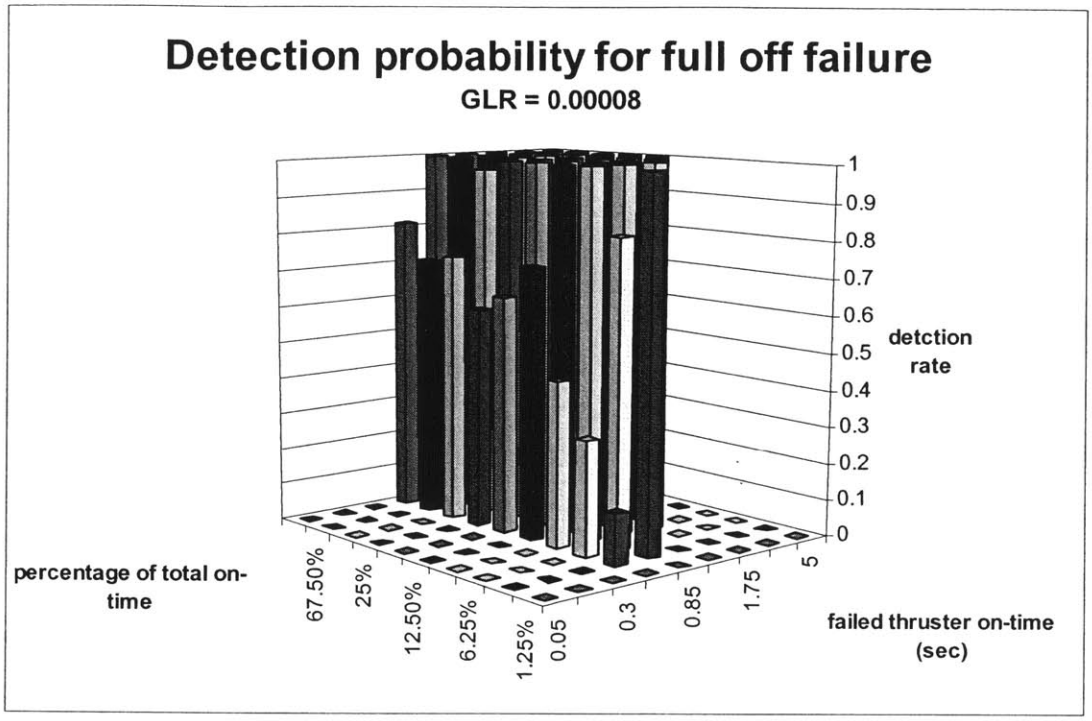




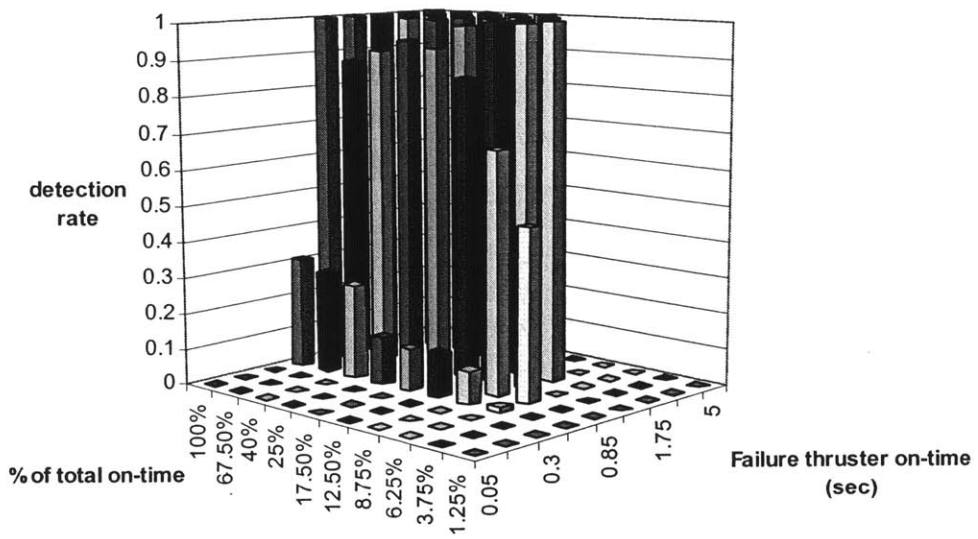
- Thruster power ~ 60 mN / thruster
- Moment of Inertia (Ref. 36):
  - I_{xx} = 0.575 kg m^2
  - I_{yy} = 0.582 kg m^2
  - I_{zz} = 0.534 kg m^2
- Carrier Differential GPS Integrity
  - Relative Position: accurate to ~ 2 cm and 2°
  - Relative Velocity: accurate to ~ 0.5 mm/sec and 1 deg/sec

APPENDIX D

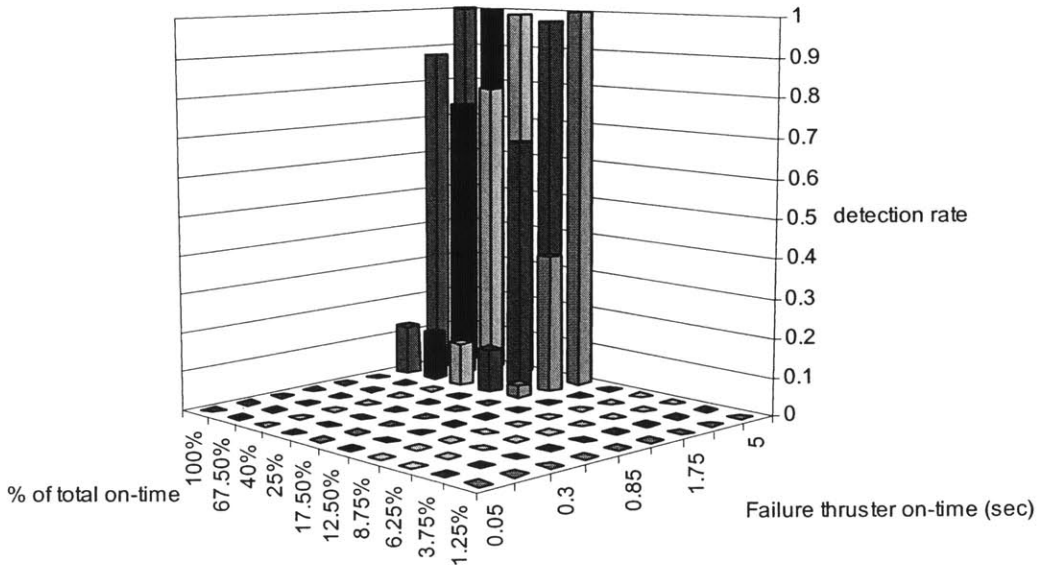
Orion Failure Detection Data



Probability of Detection for Failure to 25% level



Detection rate for Failure to 75% level



APPENDIX E

Orion Simulation Results with Different Thruster Failures

Case one:

Start location (300, 0, 0) m ; start velocity (0, 0, 0) m/sec

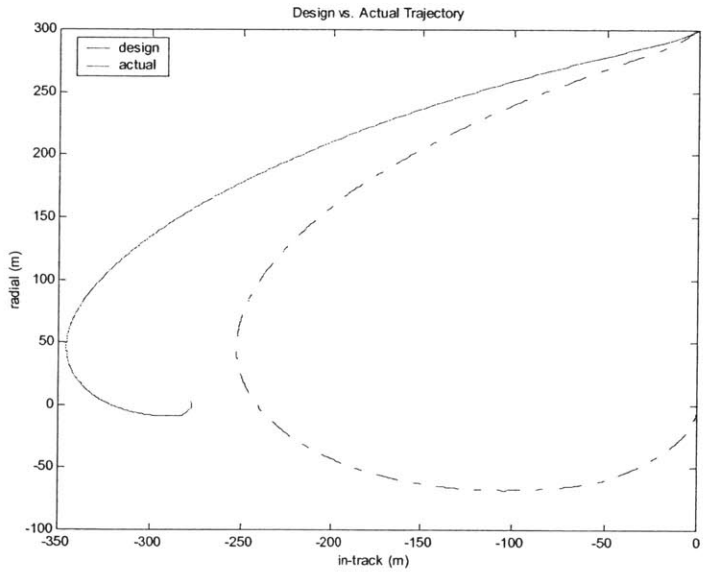
End location (0, 0, 0) m; end velocity (0, 0, 0) m/sec

Manuever time: 0.5 orbits

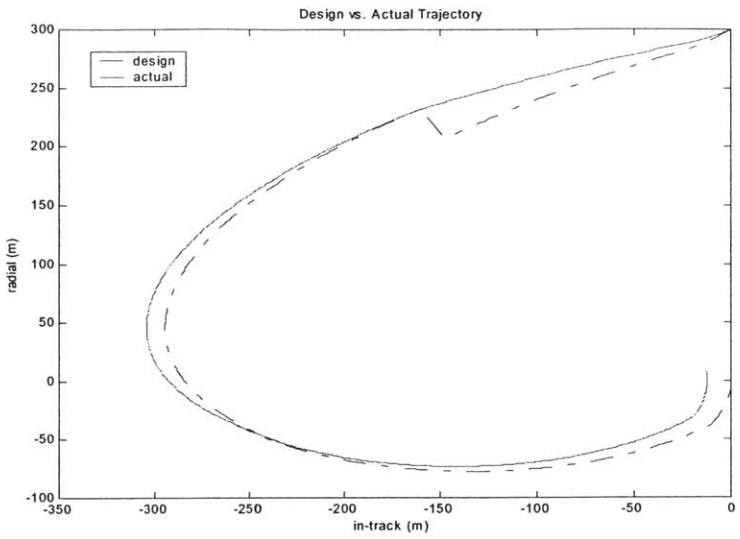
Orbital Rate: 0.0011 1/sec

Thurster failure: #1 degraded to 50% of original level at step 1.

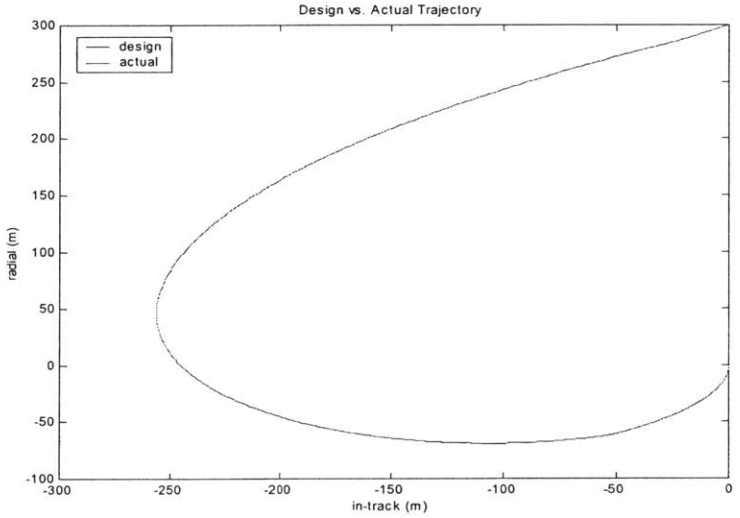
Without Fault Detection or Feedback Control:



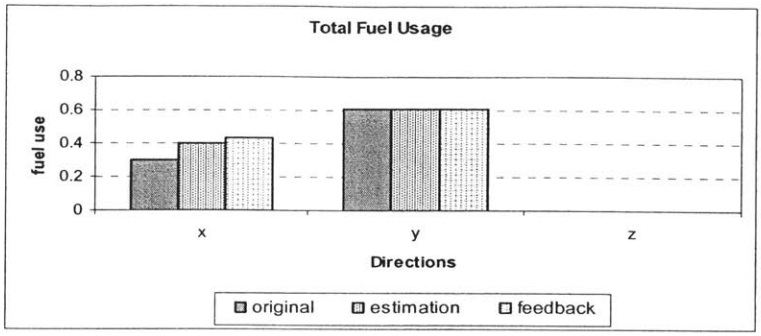
With Feedback Control:



With Fault Detection:



Fuel Usage Comparison:



Case 2:

Start location (300, 0, 0) m ; start velocity (0, 0, 0) m/sec

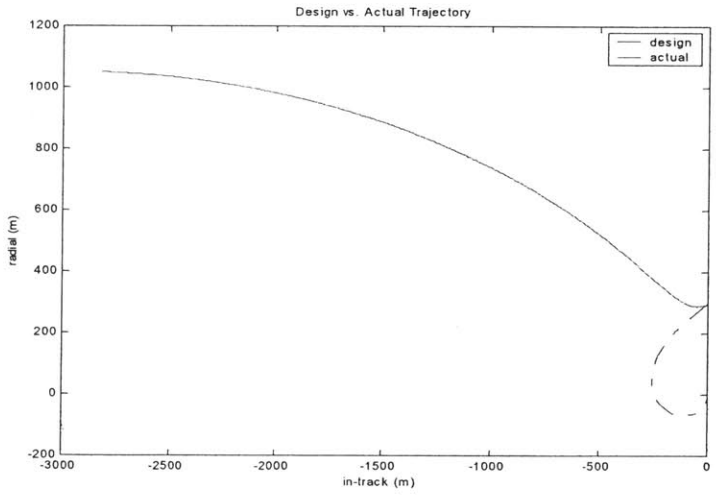
End location (0, 0, 0) m; end velocity (0, 0, 0) m/sec

Manuever time: 0.5 orbits

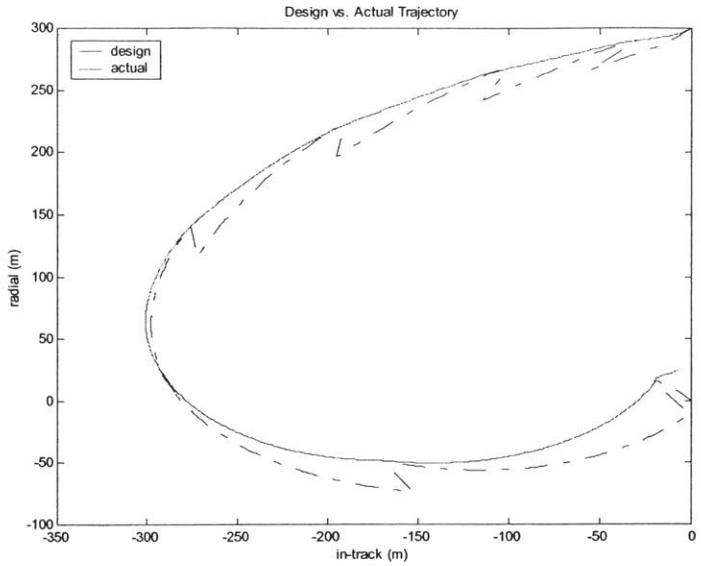
Orbital Rate: 0.0011 1/sec

Thurster failure: #1 and #2 simultaneously failed completely at step 1.

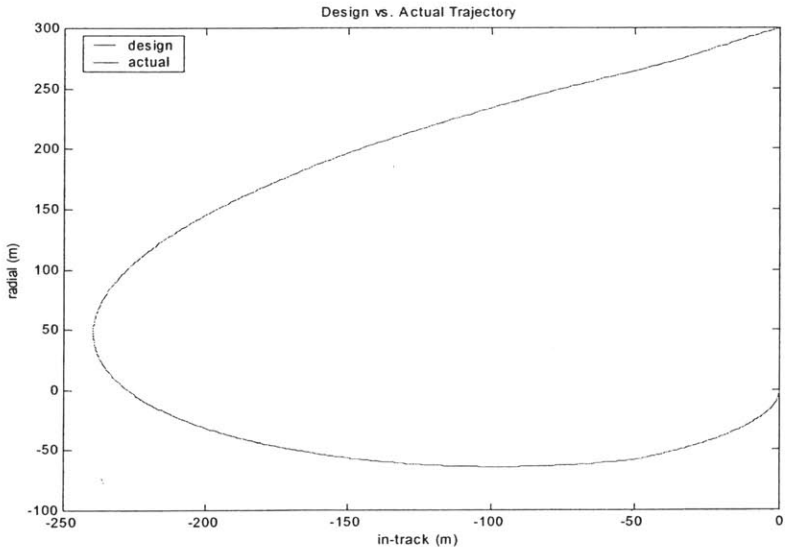
Simulation without Fault Detection or Feedback:



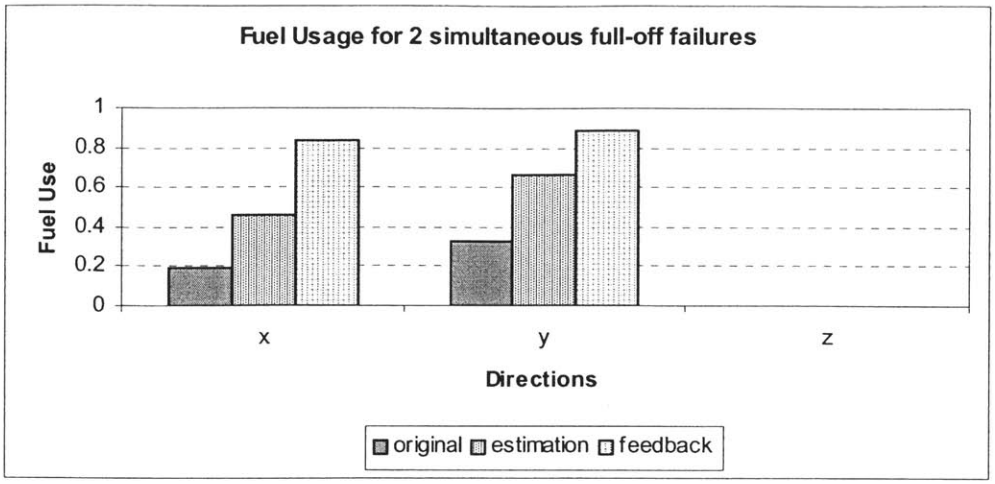
With feedback:



With Fault Detection (note failure 1 was found and corrected after first step while failure 2 was found and corrected after second step):



Fuel Usage Comparison:


Fuel Usage for 2 simultaneous full-off failures
Fuel Use
1
0.8
0.6
0.4
0.2
0
x
y
z
Directions
original
estimation
feedback

References

Ref.1 Tillerson, Mike. "Coordination and Control of Multiple Spacecraft using Convex Optimization Techniques", Master of Science Thesis, MIT, June 2002.

Ref. 2 Myers, Karen and Smith, Stephen. "Issues in the Integration of Planning and Scheduling for Enterprise Control." *DARPA-JFACC Symposium on Advance in Enterprise Control*; San Diego, CA, November 1999.

Ref. 3 Bergmann, et al, "An Advanced Spacecraft Autopilot Concept," Journal of Guidance and Control, Vol. 2, No. 3, May-June 1979, p. 161.

Ref. 4 Bodson, Mark. Evaluation of Optimization Methods for Control Allocations, AIAA-2001-4223.

Ref. 5 Thruster Testing Stanford, 2001

Ref. 6 How, Jonathan, http://www.mit.edu/people/jhow/orion/

Ref. 7 Otero, Alvar Saenz. The SPHERES Satellite Formation Flight Testbed: Design and Initial Control. EECS thesis at MIT, 8/31/2000

Ref. 8 Philip Ferguson, Chan-woo Park, Michael Tillerson, and Jonathan P. How, New Formation Flying Testbed for Analyzing Distributed Estimation and Control Architectures

Ref. 9 J.A. Paradiso, "A Highly Adaptable Steering/Selection Procedure for Combined CMG/RCS Spacecraft Control", Detailed Report, CSDL-R-1835, Draper Laboratory, Cambridge, MA march 1986

Ref. 10 Fourer, R, Gay, David M., Kernighan, Brian W. AMPL – A modeling Language for Mathematical Programming, 1993

Ref. 11 Rao, S.S., Optimization: Theory and Applications, 2nd edition, 1984.

Ref. 12 Ben-Tal, A and Nemirovski, A. Robust Solutions of Uncertain Linear Programs via Convex Programming, dec 1995.

Ref. 13 Hindi, Haitham and Boyd, Stephen. Robust solutions to L1, L2, and L_∞ uncertain linear approximation problems using convex optimization.

Ref. 14 Tillerson, M and How, J. Analysis of Sensor Noise on Formation Flying Control. Submitted to ACC journal 2001.

Ref. 15 Brown, Robert Grover and Hwang, Patrick Y.C. Introduction to Random Signals and Applied Kalman Filtering. 3rd edition, 1997.

Ref. 16 Mangoubi, Rami. Robust Estimation and Failure Detection, Springer 1998.

Ref. 17 Levy, David Retlaw. Mass Property Estimation with Jet Failure Identification for Control of Asymmetrical Satellites. MIT/Draper thesis, 1983.

Ref. 18 Montgomery, Douglas and Runger, George. Applied Statistics and Probability for Engineers. John Wiley & Sons, Inc, 1994.

Ref. 19 Willsky, Alan. A survey of Design Methods for Failure Detection in Dynamic Systems. Automatica Vol. 12 pp. 601-611, 1976.

Ref. 20 Deyst, John and Deckert, James. RCS Jet Failure Identification for the Space Shuttle. Proc. IFAC 75, 8/1975.

Ref. 21 Willsky, Alan, Deyst, John, and Crawford, Bard. Adaptive Filtering and Self-Test Methods for Failure Detection and Compensation. Proc of the 1974 JACC, June 19-21, 1974.

Ref. 22 Mehra and Peschon. An innovations approach to fault detection and diagnosis in dynamic systems. Automatica 7. 637-640. 1969.

Ref. 23 Farrell, Jay, Appleby, Brent, and Berger, Torsten. On the Detection and Accommodation of Unanticipated Faults, AIAA-92-4537-CP

Ref. 24 Wilson, Edward. Experiments in Neural Network Control of a Free Flying Space Robot. Stanford Thesis for Department of Mechanical Engineering. 1995.

Ref. 25 Boskovic, Jovan, Li, Sai-Ming and Mehra, Raman. On-Line Failure Detection and Identification (FDI) and Adaptive Reconfigurable Control (ARC) in Aerospace Applications. Proceedings of the ACC 6/25-27, 2001.

Ref. 26 Boskovic, Jovan, Li, Sai-Ming and Mehra, Raman. Intelligent Control of Spacecraft in the Presence of Actuator Failures. Proceedings of the 38th CDC, 12/1999.

Ref. 27 Jazwinski, A.H. Limited Memory Optimal Filtering. IEEE Trans. Aut. Control 13. 558-563 1963

Ref. 28 Jazwinski, A.H. Stochastic Process and Filtering Theory. Academic Press, New York, 1970.

Ref. 29 Farrell, Jay, Appleby, Brent, and Berger, Torsten. On the Detection and Accomodation of Unanticipated Faults. AIAA-92-4537-CP

Ref. 30 Boskovic, Jovan D, Li, Sai-Ming, and Mehra, Raman. On-Line Failure Detection and Identification (FDI) and Adaptive Reconfigurable Control (ARC) in Aerospace Applications, Proceedings of the American Control Conference. June25-27, 2001. pg 2625 –2626.

Ref. 31 Willsky, Alan, and Jones, Harold. A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems. IEEE Transactions on Automatic Control, 2/1976. pg 108 – 112.

Ref. 32 Beasley, J.E. Operation Research Notes – Stochastic programming conversions. http://mscmga.ms.ic.ac.uk/jeb/or/spconvert.html

Ref. 33 Orion Mission Home Page. http://www.mit.edu/people/jhow/orion/.

Ref. 34 Tillerson, M., Inalhan G., and How, J. "Coordination and Control of Distributed Spacecraft Systems Using Convex Optimization Techniques". International Journal of Robust and Non-linear Control, Nov. 2000.

Ref. 35 Leonard, Carolina. Formationkeeping of Spacecraft via Differential Drag. Master of Science Thesis, MIT, 7/1/1986.

Ref. 36 Richards, Arthur. Orion Attitude Stabilization System. Internal Orion Document, MIT, 2000

Ref. 37 Orion Critical Design Review (CDR), 10/2000

Ref. 38 Park, Chanwoo. Precise relative Navigation using Augmented CDGPS. MIT Doctoral thesis, 6/2001.

Ref. 39 Leitner, Jesse. Bauer, Frank. Folta, David. Moreau, Michael. Carpenter, Russell. How, Jonathan. Distributed Spacecraft Systems Develop New GPS Capabilities, GPS World Magazine, 2/1/2002

Ref. 40 Rasmussen, Amy. Cost Models for Large versus Small Spacecrafs. SPIE proceedings. Vol. 3439, 1998, p.14-22.

Ref. 41 Buffington, James M. "Tailless Aircraft Control Allocation"

Ref. 42 Crawford, B.S., "Operation and Design of Multi-Jet Spacecraft Control Systems", M.I.T. Doctoral Thesis, CSDL-T-509, September 13, 1968.

Ref. 43 F.D. Busse, J.P. How, J. Simpson, J. Leitner, Orion-Emerald: Carrier differential GPS for Formation Flying, Proc. of IEEE Aerospace Conference 2001, Big Sky,MT, Mar 2001.