REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	25.Oct.99	THESIS

4. TITLE AND SUBTITLE
APPLICATION OF OPTIMIZATION TECHNIQUES TO THE DESIGN AND MAINTENANCE OF SATELLITE CONSTELLATIONS

5. FUNDING NUMBERS

6. AUTHOR(S)
2D LT SMITH JAMES E

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
FY99-349

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS	15. NUMBER OF PAGES
	407
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

CSDL-T-1336

APPLICATION OF OPTIMIZATION TECHNIQUES TO THE DESIGN AND MAINTENANCE OF SATELLITE CONSTELLATIONS

by
James Earl Smith

June 1999

Master of Science Thesis
Massachusetts Institute of Technology

The Charles Stark Draper Laboratory, Inc.
555 Technology Square, Cambridge, Massachusetts 02139-3563

Application of Optimization Techniques to the Design and Maintenance of Satellite Constellations

by

James Earl Smith

B.S. Astronautical Engineering
United States Air Force Academy, 1997

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 1999

Signature of Author: _____
Department of Aeronautics and Astronautics
May 7, 1999

Certified by: _____
Dr. Ronald J. Proulx
Thesis Supervisor, CSDL

Certified by: _____
Dr. Paul J. Cefola
Thesis Supervisor, CSDL
Lecturer, Department of Aeronautics and Astronautics

Accepted by: _____
Jaime Peraire
Associate Professor
Chairman, Department Graduate Committee

[This Page Intentionally Left Blank]

Application of Optimization Techniques to the Design and Maintenance of Satellite Constellations

by

James Earl Smith

Submitted to the Department of Aeronautics and Astronautics on May 7, 1999 in
Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT

Optimization techniques were studied and applied to a variety of applications in both the design and maintenance of satellite constellations. Powell's method and parallel genetic algorithms were used in conjunction with precise orbit propagation schemes to develop robust orbit optimization tools.

Specifically, local and global optimization methods were used to design a 113:14 repeat ground track variant of the Ellipso™ inclined elliptical sub-constellation and a gear array variant of the Ellipso™ equatorial sub-constellation. The resulting optimal constellation designs were found to maintain stability, even when subjected to full perturbation analysis.

The global optimization technique of parallel genetic algorithms was also used to create an optimization approach capable of maintaining the designed orbits over specified lengths of time. Although the global method proved successful over short time periods, limitations of the approach eliminated longer time span optimizations and led to the creation of a more operational station-keeping optimization scheme. The more operational station-keeping implementation yielded similar station-keeping estimates while allowing for the study of longer time periods.

Thesis Supervisor: Dr. Ronald J. Proulx
Title: Principle Member of the Technical Staff, The Charles Stark Draper Laboratory

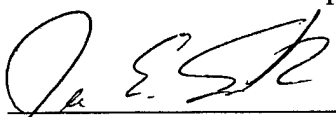
Thesis Supervisor: Dr. Paul J. Cefola
Title: Lecturer, Department of Aeronautics and Astronautics
Program Manager, The Charles Stark Draper Laboratory

[This Page Intentionally Left Blank]

ASSIGNMENT

Draper Laboratory Report Number T-1336

In consideration for the research opportunity and permission to prepare my thesis by and at The Charles Stark Draper Laboratory, Inc., I hereby assign my copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.



7 May 1999

James E. Smith, 2 Lt., USAF — Date

[This Page Intentionally Left Blank]

ACKNOWLEDGEMENTS

No major piece of work, such as this, is complete without mention of the many people who made its production possible.

First, I would like to express my appreciation to all those at the Charles Stark Draper Laboratory who provided assistance in so many ways. To my advisors, Dr. Ronald J. Proulx and Dr. Paul J. Cefola—your knowledge, inspiration, guidance, and extra motivational "push" when needed will not be forgotten. Thank you for giving me the chance as well as the tools to make the most of that chance.

To the members of the Education Office, John Sweeney, Loretta Mitrano, Arell Maguire, and George Schmidt, thank you for providing me with the opportunity and for paying the bills along the way.

To my office mates and fellow students, both past and present. To Scott Carter, Naresh Shah, Brian Kantsiper, and Joe Neelon—your association has provided me with valuable examples to emulate and your work paved the way for this work to be accomplished. To Jack Fischer and George Granholm who shared the recycled air of my office, as well as frustration with homework and FORTRAN, I couldn't have done it with out some one to complain to. To Geoff Billingsley and Mike Jamoom, it's been a long two years. Thanks for the shared experiences and insight.

To all others at the Draper Laboratory whose efforts, either directly or indirectly made this thesis possible: Tim Brand, Roger Medeiros, Linda Alger, Lee Norris, Neil Dennehy, Paul Johnson, Joe Sarcia and the reprographics department, and many others. Thank You.

Others, outside of the Draper Laboratory also had a significant impact on this work. To John Draim and the Ellipso Corporation, I express my deepest gratitude for everything from technical support to proofreading of papers. It has been a pleasure to work with you.

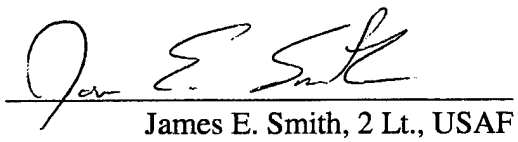
To all those at MIT whose association was brief but valuable. To Professor Eric Feron for providing me with academic guidance as well as optimization instruction. To Professors Battin, Van der Velde, Strang, and others who somehow let me pass their classes. To Liz Zotos and others in the graduate office for their behind the scenes efforts in behalf of all graduate students.

To the Air Force and members of AFIT staff, especially Capt. Rick Sutter. Thank you for allowing me this chance at civilian life and giving me the support to succeed.

To my parents for imparting to me the desire to do my best in life and for making many sacrifices to allow me to do so. To my brothers, sisters, nieces, and nephew, and to my recently acquired family—the Cromars, your e-mails and visits helped us remember there was life beyond Boston and your encouragement and prayers made all the difference.

Finally, I would like to thank the most important people in my life. My wife, Kristy, for providing support, friendship, love, and for catching all my stupid errors (both in this thesis and in life). And to my son James, whose entrance into the world in the middle of this whole project reminded me of what really is important in this life. I look forward to the eternities we can share together.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., with support from Draper Laboratory's DFY98 and 99 IR&D Programs under Astrodynamics IR&D Task 837.



James E. Smith, 2 Lt., USAF

Table of Contents

[This Page Intentionally Left Blank]

List of Figures

List of Tables

Chapter 1 Introduction

1-1 Statement of Objectives

In the field of astrodynamics, the areas of orbit design and on-orbit maintenance can have a significant impact upon the success of a given mission. Placing a satellite into an improperly designed orbit can greatly reduce the effectiveness of that satellite in accomplishing its given mission objectives. Similarly, an inability to maintain a spacecraft in the properly designed orbit can also have disastrous effects on the mission of that satellite.

All satellite orbit design is accomplished by first establishing orbit-related mission requirements. Requirement definition is the first step in orbit design as the choice of orbit typically defines not only the satellite's location in space, but also a number of other factors including the space mission lifetime, cost, environment, viewing geometry, and often also payload performance[1]. Table 1-1 lists a number of mission requirements along with parameters that can have an affect on these mission requirements. Due to the significant effect that the orbit design has on each of these mission requirement related aspects, finding the best or optimal design deserves special attention. By placing emphasis on finding the optimal orbit design, satellite designers can obtain corresponding gains in the overall mission performance of a spacecraft.

---

[1] Larson, Wiley J. and James R. Wertz. *Space Mission Analysis and Design,* Torrance, California: Microcosm, Inc., 1992, p. 157.

Table 1-1 Principal Mission Requirements and Design Parameters[2]

Mission Requirement	Influential Parameters
Coverage	Altitude Inclination Node
Sensitivity or Performance	Altitude
Environment and Survivability	Altitude Inclination
Launch Capability	Altitude Inclination
Ground Communications	Altitude Inclination
Orbit Lifetime	Altitude Eccentricity
Legal or Political Constraints	Altitude Inclination Longitude in GEO

In addition to design of the optimal orbit, finding optimal methods for performing satellite orbit maintenance or "station-keeping" also deserves added emphasis. Even with an optimally designed orbit, if that orbit and/or station within that orbit are not maintained, the performance of the satellite will degrade. Additionally, due to recent gains in the operational lifetime of a variety of satellite component technologies, satellites can maintain on-orbit operational capabilities for longer periods of time than previously estimated. Despite these gains in the operational lifetime of the component technologies, if the spacecraft cannot be maintained in the necessary orbit, there will be no corresponding gain in the operational lifetime of the satellite. Therefore, on-board fuel limitations have an increasingly important role in determining the overall lifetime of spacecraft. By finding optimal station keeping strategies, designers can not only decrease

---

[2] Larson, Wiley J. and James R. Wertz. *Space Mission Analysis and Design,* Torrance, California: Microcosm, Inc., 1992, p. 179.

the amount of required on-board fuel, but potentially increase both the satellite lifetime and corresponding system effectiveness metric as well.

The broad objective of this thesis is to explore ways in which both traditional and non-traditional optimization methods can be applied to find improvements in these two mission critical areas of satellite orbit design and on-orbit maintenance. Specifically, this thesis is a compilation of efforts to discover both orbit designs and station-keeping strategies capable of increasing the mission performance of multi-satellite constellations.

1-2 Optimization

Inherent to the efforts of this thesis is the concept of optimization. This concept of optimization can be defined in a number of ways. The American Heritage Dictionary[3] states that to optimize something is "to make the most effective use of" it. In many mathematical or other applications, this definition is applied very literally, and finding optimal solutions means obtaining the absolutely best solutions to a given problem. This interpretation is the conventional view of optimization as explained by Beightler, Phillips, and Wilde[4]:

> Man's longing for perfection finds expression in the theory of optimization. It studies how to describe and attain what is Best, once one knows how to measure and alter what is Good or Bad.... Optimization theory encompasses the quantitative study of optima and methods for finding them.

---

[3] *The American Heritage Dictionary of the English Language*, Houghton Mifflin Company, Boston, Massachusetts, 1981.

[4] Beightler, C. S., D. T. Phillips, and D. J. Wilde. *Foundations of Optimization*, Englewood Cliffs, New Jersey: Prentice-Hall, 1979, p. 1.

Although finding what is "Best" is the conventional view of optimization, it is not necessarily the only or natural definition. In human decision making, decisions are not usually made based on what is the perfect decision, as this "Best" solution is not typically available. Rather, human decision-makers take into account many factors and choose the solution that, at the time, appears better than any other available options. This more humanized view of optimization is a more natural definition and appears in many applications. In these applications, the goal of optimization shifts, from finding the best solution, to simply finding improvement.[5] Using this definition, optimal solutions are not those that give perfect performance, but instead are those that give better performance relative to other solutions. This concept of attempting to quickly find some good level of performance is known as "satisficing"[6] and is the view of optimization taken most frequently throughout this work.

1-3 Satellite Constellations

A recent trend in commercial satellite design has been the application of more than one satellite to a given mission. Due to the coordinated manner in which these satellites must perform in order to meet overall mission objectives, these multiple satellites are termed constellations. There are advantages which are evident when more than one satellite is applied to a given mission, but also a number of areas to which precise solutions (and hence optimization) becomes more important.

The main advantage gained when multiple space vehicles are applied to the same mission is in terms of coverage, or areas of the Earth that can see a satellite at any given

---

[5] Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989, p. 7.

time. With one satellite it is impossible to have coverage of more than one area of the globe at a given time. For most satellites, only a certain area of the Earth is covered and this area moves as the satellite orbits the Earth. However, if an appropriate number of satellites are placed in designated locations around the Earth, larger and larger portions of the Earth can be covered. In the late 1960s, Easton and Brecia of the United States Naval Research Laboratory in their 1969 report *Continuously Visible Satellite Constellations* analyzed coverage by satellites in two mutually perpendicular orbit planes and concluded that at least six satellites would be needed to provide full global coverage.[7] In the 1970s, J.G. Walker considered orbit types not previously considered by Easton and Brecia and concluded that continuous coverage of the Earth would require only five satellites[8]. Following this trend, John Draim, in the 1980s found and patented a constellation of four satellites in elliptical orbits that provide continuous Earth coverage.[9]

Achieving greater coverage through constellations is not without cost, however. Most noticeable of these costs is the cost to build additional satellites. In order to have multiple satellites in space, multiple satellites must first be built and launched at great expense. Improper design of satellite orbits which calls for a greater number of satellites than is actually needed can have a direct impact on a program's cost. Therefore, optimization is a useful tool in the design of these multi-satellite constellations.

The maintenance of multi-satellite constellations is also an area in which application of proper optimization techniques can lead to program cost savings. An

---

[6] Simon, H. A. *The Sciences of the Artificial,* Cambridge, Massachusetts: MIT Press, 1969.
[7]. Larson, Wiley J. and James R. Wertz. *Space Mission Analysis and Design,* Torrance, California: Microcosm, Inc., 1992, p. 189.
[8] Walker, J. G. "Satellite Constellations," *Journal of the British Interplanetary Society.* 1984, 37: 559-572.

optimally designed orbit is of little use if the satellite cannot be maintained in that orbit. An unfortunate fact of astrodynamics is that the orbits of satellites degrade. Therefore, in order to achieve mission objectives, small correctional maneuvers must often be performed such that the satellite is repositioned into the desired orbit. Each of these maneuvers, however, has an associated fuel cost. Through application of optimization techniques, the minimum fuel maneuvers can be found which allow the satellite to maintain the designed orbit and therefore, achieve the desired mission objectives at minimum cost. For constellations where the orbits of multiple satellites must be maintained and orbital maintenance costs are multiplied by the number of satellites, finding the minimum fuel maneuvers becomes even more of a priority.

1-3-1 Communication Constellations

An important mission to which constellations have been applied recently is the area of communications, specifically mobile communications. The gains in coverage through the application of multiple satellites to one mission are especially advantageous to the achievement of a communications mission. The goal of this type of mission is simple: provide a means whereby a user in one location on the globe can communicate with a user at an entirely different location on the globe. With only one satellite, achievement of this goal is impossible. However, by careful design and placement of the satellites in a constellation, coverage is increased and the goal of global mobile communications can become a reality.

---

[9] Draim, John. "Three- and Four-Satellite Continuous Coverage Constellations," *Journal of Guidance, Control, and Dynamics,* 1985, 6: 725-730.

Seeing the advantage that constellations present to the achievement of a communications mission, a number of companies have proposed systems to meet this goal. At the present time, one of these companies (Iridium) has succeed in creating an operating system[10] while the others are scheduled to begin operation within the next few years. The specific details about these constellations can be seen in Table 1-2. Note that the data presented in this table is only an approximation as precise orbital designs are often considered proprietary information.

Table 1-2 Comparisons of Personal Communications Satellite Systems[11]

	Ellipso Borealis/Concordia	Globalstar[12]	Iridium[13]	ICO[14]	Teledesic[15]
Orbit Type	SSFLA	LEO	LEO	MEO	LEO
Altitude (km)	520-7846 / 8063	1414	780	10390	1375
Eccentricity	0.33 / 0.0	0.0	0.0013	0.0	0.00118
Inclination (deg)	116.6 / 0.0	52.0	86.4	45.0	98.2
Period (hr)	3.0 / 4.67	1.9	1.7	6.0	1.9
Number of Sats	10 / 8	48	66	10	288
Number of Planes	2 / 1	8	6	2	12
Number of Sats Per Plane	5 / 8	6	11	5	24

NOTE: This table contains values that are more up to date than those available in the original reference. The more up to date values were taken from the home pages of the individual companies as contained in the footnotes.

---

[10] Swan, Peter A. "Iridium Gets Real," *Aerospace America*, Vol. 37, No. 2, February 1999, p. 23.
[11] Hulkover, Neal D., *A Reevaluation of Ellipso™, Globalstar, IRIDIUM™ and Odyssey™,* Presentation at Volpe Transportation Center, Cambridge, Massachusetts, 18 October 1994.
[12] Globalstar Corporation Internet Homepage. Available at www.globalstar.com. Accessed 28 April 1999.
[13] Iridium Corporation Internet Homepage. Available at www.iridium.com. Accessed 28 April 1999.
[14] ICO Internet Homepage. Available at www.ico.com. Accessed 28 April 1999.
[15] Teledesic Internet Homepage. Available at www.teledesic.com. Accessed 28 April 1999.

1-3-2 Ellipso™ Constellation

For all of the studies found in this thesis, the Ellipso™ constellation was used for analysis. As seen in Table 1-2, most of the designs for communications constellations are based on circular orbits. The only constellation that deviates from this circular standard is the Ellipso™ constellation. Although the optimization techniques discussed and implemented throughout this thesis would be applicable to the circular cases, as well, the non-circular nature of the Ellipso™ constellation presented a slightly more challenging case to which to apply and test the techniques.

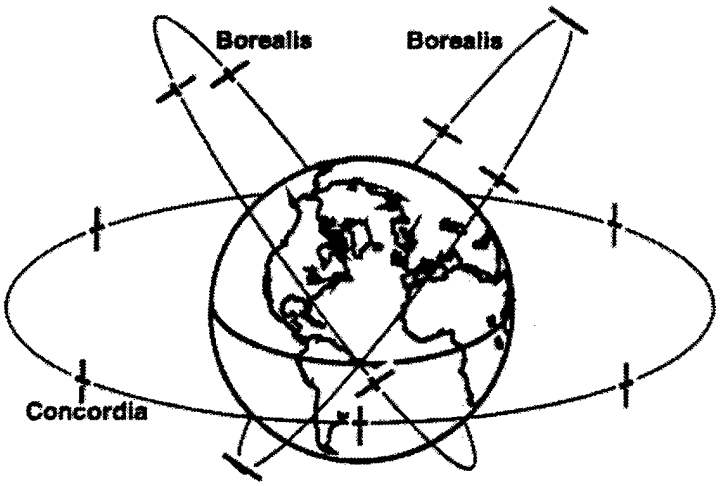


Figure 1-1 Ellipso Mobile Satellite System Orbits[16]

Figure 1-1 depicts the design of the Ellipso™ communications constellation designed by Ellipso, Inc. This constellation achieves near global coverage using two low/medium altitude sub-constellations operating in tandem. The concept of two

---

[16] Castiel, D. J. W. Brosius, and J. E. Draim. *Ellipso™: Coverage Optimization Using Elliptic Orbits,* Paper AIAA-94-1098-CP, 15th AIAA International Communications Satellite Systems Conference, San Diego, California, 28 February to 3 March 1994.

low/medium altitude sub-constellations operating in tandem, developed and patented by Castiel, Draim, and Brosius[17], is more efficient than traditional constellation designs in balancing the dual demands of global coverage and transmission power.[18]

The first of the two Ellipso™ sub-constellations is known as Borealis™. Borealis™ consists of two critically inclined, sun-synchronous eccentric orbit planes with a frozen line of apsides (SSFLA). These two orbit planes are aligned 180° apart, with one ascending node at noon and the other at midnight. This configuration provides 24-hour coverage of the Northern Hemisphere with four spacecraft and one on-orbit spare in each orbital plane.

Borealis™ is complemented by a second sub-constellation—a circular, equatorial sub-constellation known as Concordia™. Concordia™ is a medium-altitude circular equatorial orbit consisting of seven satellites and one on-orbit spare. It provides coverage around the tropical and southern latitudes. The altitude of the Concordia™ sub-constellation is approximately equal to the apogee height of the Borealis™ sub-constellation to insure that the same communications equipment can be used for all satellites in the constellation.

1-4 Thesis Overview

The remainder of this thesis is divided into two main sections: a discussion of the techniques and technologies that allow for optimization to be applied to constellation

---

[17] Castiel, D., J. E. Draim and J. W. Brosius. *Elliptical Orbit Satellite System and Deployment with Controllable Coverage Characteristics,* United States Patent Number 5,582,367, 10 December 1996.
[18] Draim, J. E. and T. J. Kacena. *Populating the Abyss—Investigating More Efficient Orbits,* Proceedings of 6th Annual AIAA/USU Conference on Small Satellites, Utah State University, Logan, Utah, 21-24 September 1992.

design and a discussion of the specific cases to which these optimization techniques were applied. Chapters 2 and 3 fall into the first section. Chapter 2 describes the basic enabling technologies and fundamentals required for performing orbit design. Specifically, orbit basics, orbit propagation, and computer aspects of orbit propagation are discussed. This discussion of astrodynamic fundamentals is followed by Chapter 3 in which specific optimization techniques and algorithms are presented. Chapters 4 and 5 discuss the application of these optimization techniques to the orbit design and maintenance applications. Chapter 4 presents an overview of the element/orbit design to which these optimization techniques were applied while Chapter 5 presents an application of the optimization techniques to a specific aspect of satellite constellation maintenance—station-keeping. Finally, Chapter 6 contains some observations resulting from the work completed for this thesis as well as some recommendations for future work.

Chapter 2 Enabling Techniques and Theories

As this thesis is based upon optimization of various applications relating to satellite constellations, a basic understanding of satellite motion (i.e. astrodynamics) is first required. Although it is impossible to explain astrodynamics in a single chapter, this chapter is an attempt to give the reader who is unschooled in astrodynamics a brief introduction to the basic concepts of this discipline. Specifically, the fundamental concepts used to describe a satellite's orbit and its motion along that orbit are briefly discussed. In an effort to provide a basis for the constellation maintenance applications, basic burn strategies are then presented. An explanation of satellite propagation techniques, and the use of parallel processing in satellite propagation applications, follows this description.

2-1 Fundamentals of Astrodynamics

The motion of bodies in space has been studied for centuries. As early as 300 BC Aristotle had developed a complex, mechanical model of the universe. Others, including Ptolemy, Copernicus, Brahe, Kepler, and Galileo added to his contributions.[19] These men, along with many others, helped to lay the foundation of modern astrodynamics, and astrodynamics, in turn, is necessary to lay the foundation for this work.

In order to apply optimization techniques to various satellite constellation applications, it first becomes necessary to understand how objects move in space and also

---

[19] Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics,* New York, New York: McGraw-Hill, Inc., 1994, p. 32.

to understand the conventions used to describe an orbit and to differentiate one orbit from another. This section is intended to impart some of that understanding to the reader. The discussions in this section are the author's compilation from the following excellent astrodynamic references:

I. *Fundamentals of Astrodynamics.* By Roger R. Bate, Donald D. Mueller, and Jerry E. White.[20]

II. *An Introduction to the Mathematics and Methods of Astrodynamics.* By Richard H. Battin.[21]

III. *Understanding Space: An Introduction to Astronautics.* By Jerry Jon Sellers.[22]

IV. *Fundamentals of Astrodynamics and Applications.* By David A. Vallado.[23]

The interested reader is directed to these sources for a more in depth discussion of any of the fundamental concepts discussed here.

2-1-1 Orbital Motion

Like all motion, the basis for orbital motion is found in Newton's three laws. Especially of interest in astrodynamics is Newton's second law that states that the time rate of change of an object's momentum is equal to the applied force.[24] For objects of constant mass, this law is often summarized as follows:

---

[20] New York, New York: Dover Publications, Inc, 1971.
[21] New York, New York: American Institute of Aeronautics and Astronautics, 1987.
[22] New York, New York: McGraw-Hill, Inc, 1994.
[23] New York, New York: McGraw-Hill, Inc, 1997.
[24] Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics,* New York, New York: McGraw-Hill, Inc., 1994, p. 105.

$$\sum \overline{F} = m \cdot \overline{a}$$ Equation 2-1

where:

$\overline{F}$ = applied force vectors

m = mass of the body

$\overline{a}$ = acceleration vector

By enumeration of the forces that act on a satellite orbiting the Earth, a general understanding of orbit motion can begin to be developed. Some of these forces include:

- The gravitational force of the Earth
- Drag from the upper atmosphere
- Third-Body gravitational effects from the Sun, the Moon, or other non-Earth bodies
- Solar radiation pressure
- Thrust from on-board rockets

By including these and other forces acting on a body in motion about the earth in Equation 2-1, the corresponding acceleration of a body can be computed and the motion of the satellite can then be described (this is the basis of the orbit propagation, see section 2-3). However, to gain an initial understanding of the type of motion to be expected, a number of simplifying assumptions, leading to the creation of the restricted two-body problem, are usually made.

2-1-1-1 Restricted Problem of Two-Bodies

Even though all of the forces enumerated above in section 2-1-1 do have an effect on the motion of satellites, the effect of all but the Earth's gravity can be eliminated with the proper assumptions. For example, by assuming that the satellite is traveling far above the Earth's atmosphere, the effect of drag can be ignored. In a similar manner, third-body effects can be ignored by assuming that the satellite is far enough away from any external bodies that their gravitational effects are negligible. It can also be assumed that the satellite is not thrusting and that the solar radiation pressure and other forces are also small enough to be negligible. The result of these assumptions is the elimination of all forces besides the Earth's gravity, where the Earth is assumed to be a point mass and the resulting gravitational field does not include non-spherical gravitational forces. By applying another of Newton's Laws, the Universal Law of Gravitation, the equation of motion of the satellite can now be expressed in a useful, analytic form:

$$\frac{G \cdot m_{Earth} \cdot m_{satellite}}{r^3} \bar{r} = m_{satellite} \cdot \ddot{\bar{r}} \qquad \text{Equation 2-2}$$

where:

G = universal gravitational constant (6.67×10^{-11} N*m^2/kg^2)

m_{Earth} = mass of the Earth

$m_{satellite}$ = mass of the satellite

$\bar{r}$ = position vector of the satellite

r = magnitude of the position vector

Simple algebraic manipulation allows for derivation of the restricted two-body equation of motion seen below:

$$\ddot{\bar{r}} + \frac{\mu}{r^3}\bar{r} = 0 \qquad \text{Equation 2-3}$$

where:

$\mu = G^*m_{Earth}$ (3.986005×10^{14} m^3/s^2)

Initially, it does not appear that much has been gained by representing the motion of a satellite in this form. Although the restricted two-body equation of motion is quite simple and elegant, it is a second-order, non-linear, vector differential equation from which it is difficult to gain any useful information about the motion of a satellite. However, if this equation is solved, the resulting solution provides insight into the expected motion of objects in orbit about the Earth. The solution process is not detailed here, but the resulting equation is presented below:

$$r = \frac{c_1}{1 + c_2 \cdot \cos\nu} \qquad \text{Equation 2-4}$$

where:

c_1 and c_2 = constants that depend on μ, position and velocity at some epoch time

ν = polar angle measured from a principle axis to the position vector

This result is quite significant. Not only does it describe the motion of the orbiting body about the Earth, it also represents a general relationship for any of the four conic sections: circle, ellipse, parabola, and hyperbola. Thus, through the restricted problem of two-bodies, it can be shown that any object moving in a gravitational field (note the gravitational field is a result of a point mass in this formulation) must follow one of these basic conic sections.

2-1-1-2 Orbit Perturbations

The restricted problem of two-bodies and corresponding results are a direct consequence of the simplifying assumptions made in regards to the forces acting on a satellite in orbit about the Earth. In the formulation of the restricted problem of two-bodies, the gravitational field of the Earth was assumed to resemble that formed by a point mass. This is not an entirely valid assumption, due to the non-homogenous nature of the Earth. Additionally, as discussed previously, though the Earth's gravity is the most significant force, it is not the only force acting on a satellite. The forces listed in section 2-1-1, along with some other smaller forces, can cause acceleration in the motion of the satellite. The accelerations that are not part of the two-body model are known as perturbing accelerations.

The perturbing accelerations will generally cause three types of variations in the orbit of a satellite: short period, long period, and secular variations. Figure 2-1 illustrates the difference between these types of effects:

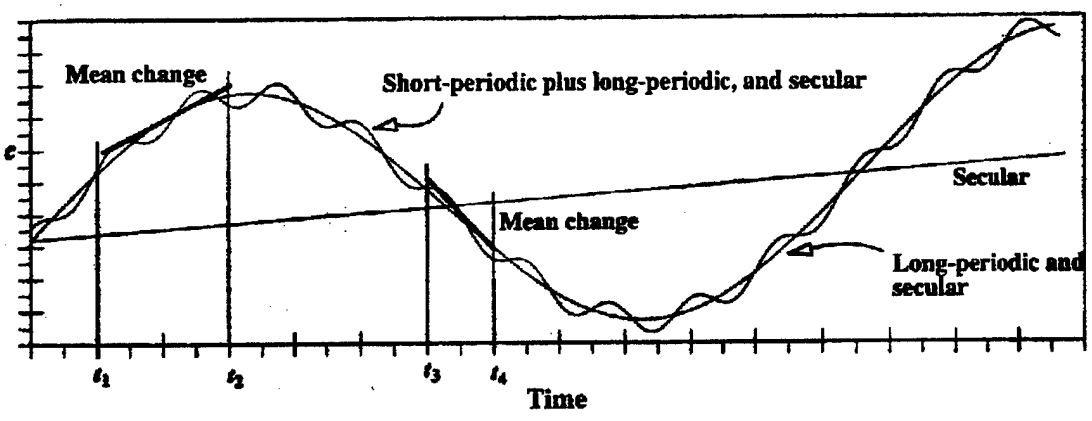


Figure 2-1 Perturbing Acceleration Effects[25]

---

[25] Vallado, David A. *Fundamentals of Astrodynamics and Applications,* New York, New York: McGraw-Hill, Inc., 1997, p. 545.

As seen in Figure 2-1, secular accelerations simply add a constant increasing or decreasing perturbation to an orbit. The effects of long period accelerations, on the other hand, are periodic. Short periodic accelerations are also periodic, but their amplitude is smaller than the amplitude of long periodic accelerations, and they recur at higher frequency.

Some important factors regarding these perturbing accelerations should be noted. First is the fact that the effects of most of these perturbations are dependent upon the orbit and/or physical characteristics of the satellite in question. Also, unlike the force of gravity of the Earth used in the two-body derivation, a number of these perturbing forces can be time varying. Both of these factors make modeling of satellite motion with perturbing accelerations more difficult than simple two-body modeling. It should also be noted that although the effects of most of these perturbations can cause undesirable behavior in the motion of the satellite (see Chapter 5), there are times when they can be utilized to one's benefit (see Chapter 4).

Regardless of whether they contribute positive or negative effects, the perturbing accelerations are one of the main reasons that optimization must be applied to astrodynamic applications. Optimization of design and maintenance applications is sometimes necessary in the two-body realm, but the addition of time-varying, satellite dependent perturbations to this realm complicates the optimization process and increases the need for robust optimization tools.

2-1-2 Orbital Elements

Besides having a basic understanding of orbital motion, an ability to describe the orbit of a given satellite in understandable terms is also a pre-requisite of attempting to optimize these orbits. As the size, shape and orientation of an orbit can vary drastically from the size, shape, and orientation of any other orbit, it is necessary to find parameters which allow for a complete description of these differences.

In order to uniquely describe an orbit for a given satellite, six parameters are required. The three components of position along with the three components of velocity could serve to distinguish the orbit of one satellite from another. However, these parameters are difficult to visualize. Instead, six parameters, known as classical or Keplerian orbital elements have been defined and are more commonly used to provide a descriptive parameterization of an orbit.

2-1-2-1 Keplerian Orbital Elements

The first two Keplerian orbital elements are used to define the size and shape of an orbit. They are the common geometric terms of semi-major axis and eccentricity as described below:

- Semi-major axis (a): The semi-major axis is a constant used to define the size of the orbit. It describes half the distance across the major axis of the orbit as seen in Figure 2-2. This figure also shows two other orbital values: apogee and perigee. Apogee is the point in the orbit furthest from the Earth and perigee is the closest point in the orbit to the Earth. These points become important in the definition of some of the remaining orbital elements.

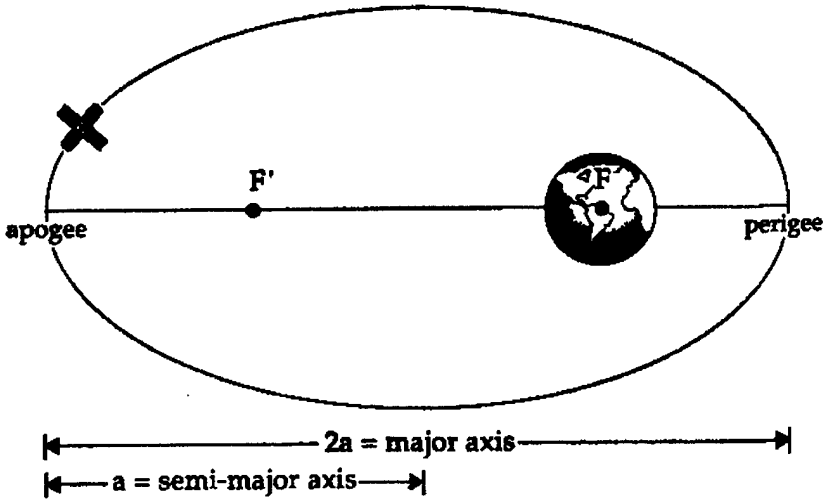


Figure 2-2 Semi-Major Axis Depiction[26]

- Eccentricity (e): Eccentricity is a constant that defines the shape of the orbit. For a circular case eccentricity is defined to be zero. As the orbit becomes more and more elliptical, the eccentricity then grows, reaching a value of one for parabolic orbits. Hyperbolic orbits are defined to have eccentricities greater than one. An example of eccentricities for a variety of conic sections is seen below:

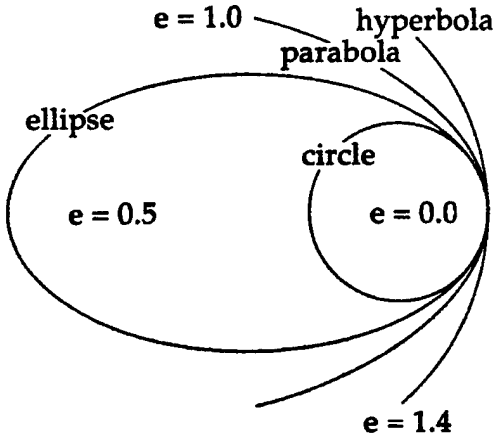


Figure 2-3 Sample Eccentricities of the Four Conic Sections[27]

---

[26] Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics,* New York, New York: McGraw-Hill, Inc., 1994, p. 141.

In addition to defining parameters that describe the size and shape of an orbit, it is also necessary to describe how that geometric shape is oriented in space. To do this, it is first necessary to define a reference frame in which to orient the orbit. For the Keplerian orbital elements, the Geocentric-Equatorial coordinate system is used. This reference frame is centered at the center of the Earth and the fundamental plane is defined to correspond to the Earth's equatorial plane. The principal direction (I) points to the vernal equinox direction, the out of plane component (K) is aligned with the North Pole of the Earth, and the third direction (J) is simply a result of the cross product of the other two directions to create a right handed coordinate system. This coordinate system is illustrated in Figure 2-4:

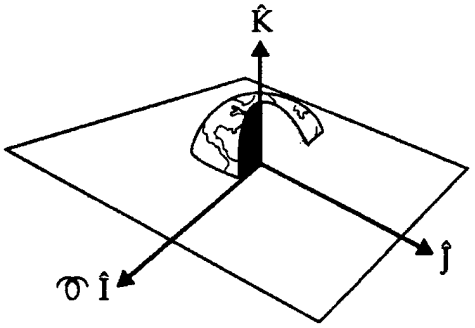


Figure 2-4 The Geocentric-Equatorial Coordinate System[28]

Using the Geocentric-Equatorial reference frame, it is possible to define three parameters that uniquely describe the orientation of a given orbit in that frame. These three parameters are inclination, longitude of the ascending node, and argument of perigee. A brief description of each is provided below:

---

[27] Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics,* New York, New York: McGraw-Hill, Inc., 1994, p. 142.
[28]Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics.* New York, New York: McGraw-Hill, Inc., 1994, p. 142.

- Inclination (i): Inclination is the angle between the out of plane component vector (K) and the angular momentum vector (the result of the cross product of position with velocity). It is used to describe the "tilt" of the orbit plane with respect to the equatorial plane. Its range of values is $0° \leq i \leq 180°$.
- Longitude/Right Ascension of the Ascending Node (Ω): The longitude of the ascending node is the angle between the principal direction (I) and the ascending node. The ascending node is defined as the location where the satellite crosses from the Southern Hemisphere into the Northern Hemisphere. The range of values for Ω is $0° \leq \Omega \leq 360°$.
- Argument of Perigee (ω): The argument of perigee is the angle from the ascending node to perigee measured in the direction of satellite motion. Its range of values is $0° \leq \omega \leq 360°$.

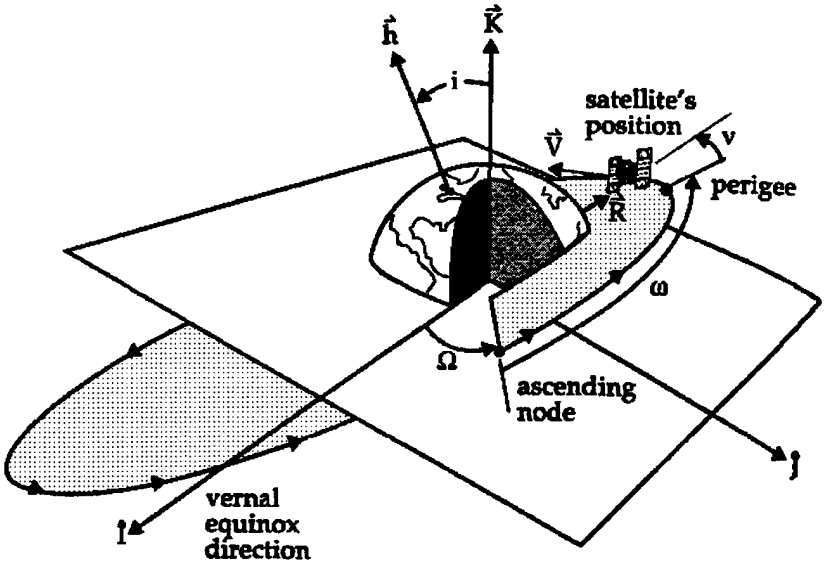


Figure 2-5 Four of the Keplerian Orbital Elements[29]

---

[29] Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics.* New York, New York: McGraw-Hill, Inc., 1994, p. 147.

Figure 2-5 shows the three orientation orbital elements as well as the final orbital element, shown as ν. This element is known as true anomaly and is used to define the position of the satellite in the orbital plane. It is defined as the angle between the argument of perigee and the actual position of the satellite. It should also be noted that because of the similarity between ν and v (the symbol for velocity), f is sometimes used to represent this element.

For many applications, including this project, it is often useful to define an additional element to take the place of true anomaly. The element, M, or Mean Anomaly is often used in its place. Mean anomaly is an angular expression of the average angular motion of the satellite in the orbit as opposed to a description of its true position. Mean anomaly cannot be geometrically defined and therefore cannot be included in a figure such as that used to describe the other orbital elements. Instead, the mathematical definition must be relied upon as seen below:

$$M = \sqrt{\frac{\mu}{a^3}}(t-\tau) \qquad \text{Equation 2-5}$$

where:

τ = time since passage of perigee

2-1-2-2 Equinoctial Orbital Elements

The Keplerian orbital elements are adequate to describe the majority of orbits. However, for some specific cases (i.e. circular and/or equatorial orbits) singularities can arise when the Keplerian elements are used. For example, for circular orbits, the distance from Earth is constant at all points in the orbit. Therefore the location of perigee is undefined. This lack of a properly defined perigee location leads to an inability to define

a corresponding argument of perigee or true anomaly for circular orbits. In a similar manner, for equatorial orbits whose inclination is 0° (the orbit and equatorial planes coincide), an ascending node cannot be defined and therefore, the longitude of ascending node also does not exist.

The singularities that arise from the use of Keplerian orbital elements can lead to complications for certain applications, specifically propagation of orbits with small inclinations and eccentricities. To overcome this problem, Broucke and Cefola[30], and later the Russians Lidov and Yurasov[31] derived the analytical equations required to build a propagator entirely in a non-singular element set. This second type of element set is known as the equinoctial element set. These elements are not as commonly used, but they do avoid the singularities present in circular and/or equatorial orbit element sets. The six equinoctial elements and their corresponding mathematical relationships to the Keplerian elements are defined as follows:

$$
\begin{aligned}
a &= a \\
h &= e\sin(\omega + I\Omega) \\
k &= e\cos(\omega + I\Omega) \\
p &= \tan^{I}\left(\frac{i}{2}\right)\sin\Omega \\
q &= \tan^{I}\left(\frac{i}{2}\right)\cos\Omega \\
\lambda &= M + \omega + \Omega
\end{aligned}
\qquad \text{Equation 2-6}
$$

where I is the retrograde factor which assumes the following values:

---

[30] Broucke, R. A. and Cefola, P. J., "On The Equinoctial Orbital Elements," *Celestial Mechanics*, 1972, 5: 303-310.

[31] Yurasov, V. *Universal Semi-analytic Satellite Motion Propagation Method,* US/Russian Space Surveillance Workshop, Poznan, Poland, 5 July 1996.

$$I = \begin{cases} 1, & for \quad 0 \le i < \pi \\ -1, & for \quad 0 < i \le \pi \end{cases}$$

If I = 1, the resulting element set is known as the direct equinoctial elements. If I = -1, then the element set is known as the retrograde equinoctial elements. This formulation was introduced because the direct elements experience a singularity at i = π and the retrograde elements experience a singularity at i = 0.

2-2 Burn Planning Techniques

Throughout the operational lifetime of a satellite, it is often necessary to change the orbital elements in some manner. The reason for these changes can be expected or unexpected. For example, when first placed into orbit, a satellite may not end up in the exact orbit for which it was designed. To correct this problem, "burns" (short firings of on-board rockets) are calculated and executed such that the satellite is repositioned into the orbit in which it can carry out the designed mission.

Unfortunately, a large percentage of on-orbit burns are a direct result of the negative effect of the perturbing accelerations described previously. Although, most orbits are designed to allow a satellite to perform a certain mission, due to the perturbing accelerations, the satellite will often drift from the desired orbit. In order to regain the desired state, small burns, known as "station-keeping" maneuvers, must be performed. A large portion of this thesis deals with the optimal way to perform these station-keeping maneuvers. Therefore, as background, this section contains some basic information regarding satellite maneuvers.

2-2-1 Fundamental Concepts of Burn Planning

In order to understand the details of burn planning, it is first necessary that one understand two very fundamental concepts: how burns are compared/measured and how to calculate these comparisons. The details of these two concepts are summarized in the following two sections.

2-2-1-1 Delta-V

A fundamental concept of satellite burn planning is that of delta-v or ΔV. For satellites, the amount of fuel that is required to complete a maneuver is directly proportional to the change in velocity required to complete that maneuver as shown in the rocket equation, below. Therefore, rather than dealing with amounts and equations related to fuel calculations, burn planners simply keep track of the required change in velocity. A solution which gives the smallest required velocity change (and hence the smallest ΔV) will also be the solution which requires the least amount of fuel.

$$\Delta V = c \ln\left(\frac{m_{initial}}{m_{final}}\right) \qquad \text{Equation 2-7}$$

where:

ΔV = velocity change (m/s)

c = effective exhaust velocity (m/s)

$m_{initial}$ = initial mass of vehicle before firing rocket (kg)

m_{final} = final mass of vehicle after firing rocket (kg)

2-2-1-2 Vis-Viva Integral

Because ΔV is a fundamental concept in the burn planning process, the ability to calculate changes in velocity between various orbits is essential. Battin and others have shown how this can be done using the concept of specific energy along with the eccentricity vector.[32] The basic concept behind this derivation is that the specific energy (ε) of an orbiting body can be expressed as a function of the gravitational constant and the semi-major axis as follows:

$$\varepsilon = \frac{-\mu}{2a} \qquad \text{Equation 2-8}$$

Additionally, the specific potential and kinetic energies of a satellite can be summed to yield a second relationship for the energy, which is a function of the desired quantity: velocity.

$$\varepsilon = \frac{v^2}{2} - \frac{\mu}{r} \qquad \text{Equation 2-9}$$

A combination of equations 2-7 and 2-8, followed by some algebraic manipulation, yields an expression that can be easily solved for velocity. This expression is known as the vis-viva integral and proves useful in maneuver planning calculations.

$$v^2 = \mu\left(\frac{2}{r} - \frac{1}{a}\right) \qquad \text{Equation 2-10}$$

---

[32] Battin, Richard H. *An Introduction to the Mathematics and Methods of Astrodynamics,* New York, New York: American Institute of Aeronautics and Astronautics, 1987, p. 116.

2-2-2 Hohmann Transfer

One of the most basic satellite maneuvers is the Hohmann transfer. This is a transfer between two coplanar orbits whose major axes are aligned. In 1925, Hohmann recognized that the smallest ΔV for this type of transfer is achieved by using a doubly tangent transfer ellipse.[33] In an effort to provide equations representative of maneuver calculations, a typical Hohmann transfer is detailed below.

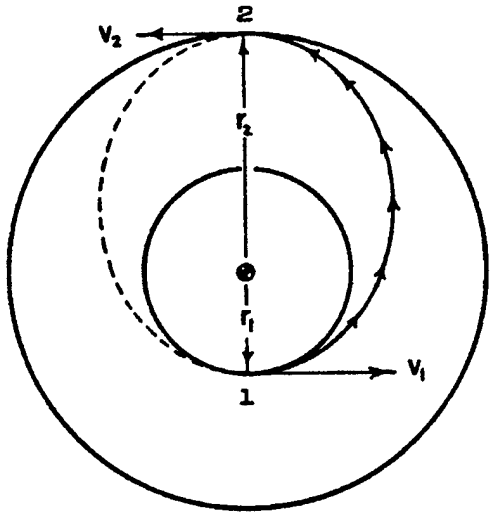


Figure 2-6 Typical Hohmann Transfer[34]

Figure 2-6 displays a typical Hohmann transfer. A satellite is initially assumed to be in orbit one (with radius r_1) and it is desired to move the satellite to orbit two (with radius r_2). For a Hohmann transfer, this change of orbit is accomplished via the elliptic transfer orbit whose perigee is tangent to orbit one and whose apogee is tangent to orbit

---

[33] Bate, Roger R., Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics,* New York, New York: Dover Publications, Inc., 1971, p. 163.
[34] Bate, Roger R., Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics,* New York, New York: Dover Publications, Inc., 1971, p. 164.

two. The calculations involved in determining the total ΔV for this transfer are outlined below.

Step I. Find the ΔV required to enter the transfer orbit from orbit one:

A. Find the velocity in orbit one

$$v_0 = \sqrt{2\left(\frac{\mu}{r_1} + \varepsilon_{orbit\_1}\right)} = \sqrt{2\left(\frac{\mu}{r_1} - \frac{\mu}{2r_1}\right)} = \sqrt{\frac{\mu}{r_1}} \qquad \text{Equation 2-11}$$

B. Find the velocity in the transfer orbit at point of tangency with orbit one

$$v_1 = \sqrt{2\left(\frac{\mu}{r_1} + \varepsilon_{transfer\_orbit}\right)} = \sqrt{2\left(\frac{\mu}{r_1} - \frac{\mu}{r_1 + r_2}\right)} \qquad \text{Equation 2-12}$$

C. Find the velocity difference between the two orbits

$$\Delta V_1 = |v_1 - v_0| \qquad \text{Equation 2-13}$$

Step II. Find the ΔV required to exit the transfer orbit and remain in orbit two

A. Find the velocity in the transfer orbit at point of tangency with orbit two

$$v_2 = \sqrt{2\left(\frac{\mu}{r_2} + \varepsilon_{transfer\_orbit}\right)} = \sqrt{2\left(\frac{\mu}{r_2} - \frac{\mu}{r_1 + r_2}\right)} \qquad \text{Equation 2-14}$$

B. Find the velocity in orbit two

$$v_f = \sqrt{2\left(\frac{\mu}{r_2} + \varepsilon_{orbit\_2}\right)} = \sqrt{2\left(\frac{\mu}{r_2} - \frac{\mu}{2r_2}\right)} = \sqrt{\frac{\mu}{r_2}} \qquad \text{Equation 2-15}$$

C. Find the velocity difference between the two orbits.

$$\Delta V_2 = |v_f - v_2| \qquad \text{Equation 2-16}$$

Step III. Find the total ΔV for the complete Hohmann transfer

$$\Delta V_{Hohmann} = \Delta V_1 + \Delta V_2 \qquad \text{Equation 2-17}$$

2-2-3 Gauss' Variational Equations

At times, circumstances arise which require certain elements of an orbit to be changed in a manner which cannot be accomplished by a Hohmann or any other predefined transfer sequence. In order to change certain elements without varying others, it is often useful to rely on a set of equations known as the Gauss Variational Equations. These equations represent the variation in the orbital elements as a function of the disturbing accelerations. The derivation of these equations is presented in Battin[35], and the resulting variation of each element with respect to disturbing accelerations is presented below.

$$\frac{da}{dt} = \frac{2a^2 v}{\mu} a_{dt}$$

$$\frac{de}{dt} = \frac{1}{v}\left[2(e+\cos f)a_{dt} - \left(\frac{r}{a}\sin f\right)a_{dn}\right]$$

$$\frac{di}{dt} = \frac{r\cos\theta}{h} a_{dh}$$

$$\frac{d\Omega}{dt} = \frac{r\cos\theta}{h\sin i} a_{dh}$$

$$\frac{d\omega}{dt} = \frac{1}{ev}\left[(2\sin f)a_{dt} + \left(2e + \frac{r}{a}\cos f\right)a_{dn}\right] - \frac{r\sin\theta\cos i}{h\sin i} a_{dh}$$

$$\frac{dM}{dt} = n - \frac{b}{eav}\left[\left(2\left(1+\frac{e^2 r}{p}\right)\sin f\right)a_{dt} + \left(\frac{r}{a}\cos f\right)a_{dn}\right]$$

Equation 2-18

where:

θ = argument of latitude = ω+f

p = parameter of the orbit = $a(1-e^2)$

b = semi-minor axis = $\sqrt{|a| \cdot p}$

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

h = magnitude of the angular momentum vector

It should be noted that these equations are written in the tangential-normal coordinate frame. To define this frame, let a_{dt} be the component of the disturbing acceleration in the plane of the orbit along the velocity vector. The second component, a_{dh} is then defined as the out of plane component of the acceleration in the direction of the angular momentum vector (h). Finally, a_{dn} is the component of the disturbing acceleration in the plane of the orbit, but perpendicular to the velocity and angular momentum vectors.

In order that the Gauss Variational Equations are useful in terms of ΔV analysis, it is also necessary to realize the relationship between acceleration and velocity as seen below:

$$\delta v = a \cdot \delta t \qquad \text{Equation 2-19}$$

By substituting this relationship (equation 2-19) into the variational equations, it is possible to use these variational equations in burn planning. For example, if it is desirable to change the eccentricity of the orbit, without changing the semi-major axis, it is easily seen from the variational equations that this can be done by a burn which is completely in the n (perpendicular to the velocity in the orbit plane) direction. Or, on the other hand, given a burn, with only out of plane components, the variational equations

---

[35] Battin, Richard H. *An Introduction to the Mathematics and Methods of Astrodynamics,* New York, New York: American Institute of Aeronautics and Astronautics, 1987, p. 488.

can be used to show that the only effect of this burn will be to change the ascending node, the inclination, and the argument of perigee.

2-3 Satellite Propagation Techniques

Inherent to the field of astrodynamics is the desire to successfully determine the future position and velocity of an orbiting body. If the real world were as easy to model and describe as it is assumed to be in the derivation of the two-body problem, finding the future state of a satellite would be a simple matter of solving equation 2-3. However, as discussed previously, many of the assumptions made to obtain the equations of motion for the two-body problem are not valid. In actuality, forces other than the point-mass modeled Earth's gravity act on the satellite, sometimes in an unpredictable manner. These other forces are known as perturbations and it is their presence that makes the determination of future orbit states difficult.

However, although accurate orbit propagation in the presence of a number of perturbations is more difficult than one might originally assume, a number of methods have been developed which incorporate the effect of the perturbing forces in predicting future motion of satellites. These methods can be broken into three main categories: General Perturbation Techniques, Special Perturbation Techniques, and Semi-Analytical Techniques. Each will be discussed briefly below. For a more detailed explanation the reader is referred to one of the astrodynamic sources described previously in section 2-1.

2-3-1 General Perturbation Techniques

General perturbation techniques are also known as analytical techniques. The basis of these methods is an analytic integration of the perturbing accelerations. These

methods replace the original equations of motion with an analytical approximation that captures the essential character of the motion over some limited time interval. Most often, the expression for the perturbing accelerations takes the form of a truncated series expansion.

The general perturbation technique has both advantages and disadvantages. First, by expressing the equations of motion analytically, a solution is obtained which is valid for any set of initial conditions. Other methods are often very specialized in their development and are only valid for only one set of initial conditions. And, although expressing the perturbing accelerations analytically can be a difficult and lengthy process, it leads to a better understanding of the source of the perturbation.

2-3-2 Special Perturbation Techniques

Unlike general perturbation techniques, special perturbation techniques rely very little on analytical information and instead, rely upon the ability to integrate the equations of motion, including all necessary perturbing accelerations, numerically. However, because of their reliance on numerical integration, they suffer from specificity. Applying the solution obtained for one problem to a similar, but slightly different problem is a non-trivial process. For most applications, all of the computations must be re-evaluated for each case, adding additional time to the solution technique.

One of the most often used special perturbation techniques is the Cowell method. This method was developed by P.H. Cowell in the early 20th century and was used to determine the orbit of the eighth satellite of Jupiter.[36] Since that time, Cowell's method

---

[36] Bate, Roger R., Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics,* New York, New York: Dover Publications, Inc., 1971, p. 387.

has become increasingly popular due to the increase in speed and capacity of computers which makes numerical integration a more feasible solution path.

The basis of Cowell's method is to rewrite the two-body equation of motion with the perturbing accelerations included.

$$\ddot{\bar{r}} + \frac{\mu}{r^3}\bar{r} = \bar{a}_{perturbed} \qquad \text{Equation 2-20}$$

The specific form of the perturbing accelerations on the right side of this equation depends upon the number and type of perturbations to be included. However, after successful inclusion of the perturbations into Equation 2-19, the propagation of the satellite then becomes a simple matter of numerically integrating a differential equation. Unfortunately, although this method is fairly straightforward and can be extremely accurate, it can also take large amounts of computing time.

2-3-3 Semi-Analytical Techniques

Semi-analytical techniques were designed to combine the best features of the general perturbation techniques with the best features of the special perturbation techniques. This combination was done in an attempt to get an optimal mix of the accuracy of special perturbation techniques with the efficiency of the general perturbation techniques. This section contains a brief description of semi-analytical methods, as well as an overview of the semi-analytical orbit propagator used in the completion of this study: the Draper Semianalytic Satellite Theory (DSST) Standalone Orbit Propagator.

2-3-3-1 Semi-Analytical Technique Description

The basis for semi-analytical methods lies in the de-coupling of the satellite equations of motion into two parts: one part which contains the secular and long periodic effects (see Figure 2-1) and one part which contains only the short periodic effects. This de-coupling is accomplished through a process known as the Generalized Method of Averaging or GMA (see Figure 2-7). GMA can be applied both through analytic and numerical methods.

Regardless of the method used for the separation, the advantage of semi-analytical methods lies in the separation of the perturbing effects. By removing all high frequency terms from the secular and long period components, the step size for integration of these components can be lengthened to equal the period of the shortest long period effect. In practice this step size is several orders of magnitude larger than that required, should the de-coupling not be accomplished. The short periodic variations are then solved for and added to the mean elements at the request times to produce the desired element history. For the specific details of this is process, the interested reader is referred to Wayne McClain's *A Semianalytic Artificial Satellite Theory: Vol. 1: Application of the Generalized Method of Averaging to the Artificial Satellite Program.*[37]

---

[37] Privately published, 1992. Copy available from author.

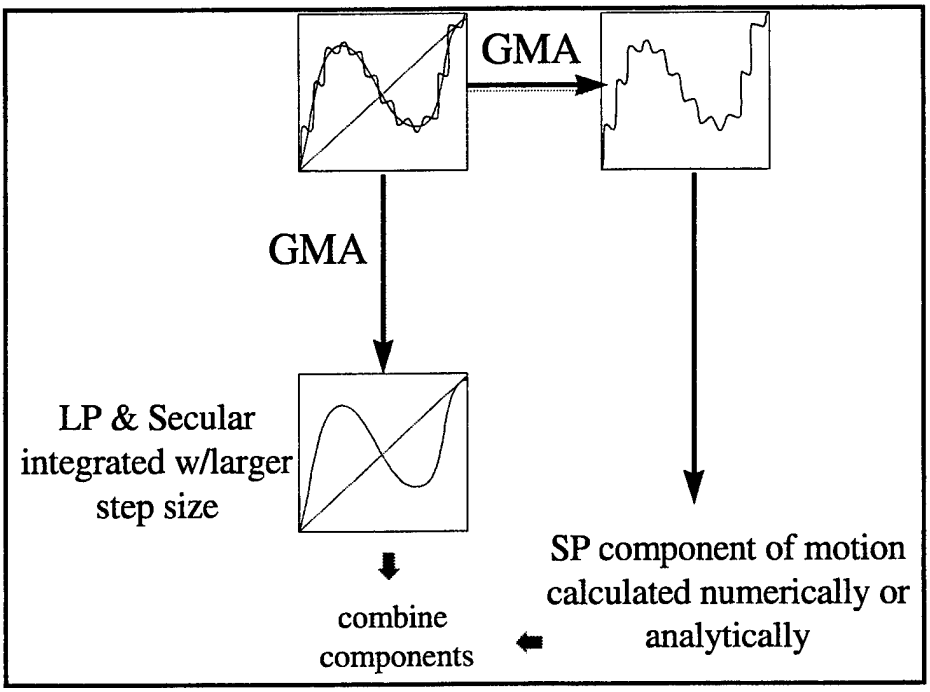


Figure 2-7 The Generalized Method of Averaging[38]

2-3-3-2 Draper Semi-Analytic Satellite Theory Standalone Orbit Propagator

By taking advantage of the benefits of semi-analytical methods, members of the staff at the Charles Stark Draper Laboratory and others have developed a highly accurate orbit propagator known as DSST (for a complete history of this propagator see Neelon, 59)[39]. A 1995 study conducted by Barker, Casali, and Wallner of the Kaman Sciences Corporation compared the accuracy and run times of this DSST propagator with several other propagation theories. The study concluded that of all the propagators considered, DSST contained the most complete perturbation models, and thus produced the most

---

[38] Fischer, Jack D. *The Evolution of Highly Eccentric Orbits,* CSDL-T-1310, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1998, p. 80.
[39] Neelon, Joseph G., Jr. *Orbit Determination for Medium Altitude Eccentric Orbits Using GPS Measurements,* CSDL-T-1330. Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February, 1999, p. 59.

accurate results for four different types of orbits studied.[40] As this thesis relied heavily upon orbit propagation, and since accuracy was a desirable element of the optimizations to be performed, DSST was chosen as the propagator for all simulations contained within this study.

Due to the reliance of this work upon DSST, it is imperative that the current models that make up the DSST package be outlined[41]. In the DSST package, the secular and long period effects are computed through the mean equations of motion. The perturbations included in this computation include:

- Central-body gravitational spherical harmonics of arbitrary degree and order (zonals and tesseral resonance) based on the 50 x 50 geo-potential
- J_2^2 second order effect. An explicit analytical expression, truncated to the first power of the satellite eccentricity, is used for the mean element rates of change.
- Third-body point mass effects (both single and double averaging theories)
- Atmospheric drag with J_2/drag coupling. Jacchia-Roberts, Harris Priester, and MSISE-90 atmospheric density models are available.
- Solar radiation pressure with eclipsing
- Integration Coordinate System based on the FK4 (B1950.0) and FK5 (J2000.0) coordinate frames
- Solar and lunar solid Earth tides

---

[40] Barker, W. N., S. J. Casali and R. N. Wallner. *The Accuracy of General Perturbation and Semianalytic Satellite Ephemeris Theories,* AAS Paper 95-432, AAS/AIAA Astrodynamics Specialist Conference, Halifax, Nova Scotia, August 1995.

The short-period variations are calculated separately and include the following perturbations:

- Central-body gravitational zonal harmonics of arbitrary degree based on the 50 x 50 geo-potential
- Central-body gravitational m-daily sectoral and tesseral harmonics of arbitrary degree and order based on the 50 x 50 geo-potential
- Central-body gravitational high-frequency sectoral and tesseral harmonics of arbitrary degree and order based on the 50 x 50 geo-potential
- J_2^2 and J_2/m-daily second order short-periodic variations
- Third-body point mass effects [both single (including Weak Time Dependence) and double averaging theories]
- Atmospheric drag
- Solar radiation pressure

2-4 Parallel Processing

The process of optimizing satellite constellations is one that can become extremely computationally intensive. For some of the applications in this study, it was necessary to use DSST to propagate a satellite's orbit forward in time 90 days more than 150,000 times. In order to reduce the computation time required for completion of these intensive optimization applications, parallel processing was implemented.

---

[41] Neelon, J., P. Cefola, and R. Proulx. "Current Development of the Draper Semi-Analytical Satellite Theory Standalone Orbit Propagator Package," *Advances in the Astronautical Sciences,* Volume 97 Part II, American Astronautical Society, 1998, p. 2037-2051.

2-4-1 Parallel Processing Description

The concept behind parallel processing is to use a group of processors to work simultaneously on different parts of the same problem as opposed to having one processor perform all calculations sequentially. By implementing a solution process in this manner, the time required to complete a computationally intensive simulation can be drastically reduced.

One of the most common parallel processing configurations, and the one used throughout this study, is known as the master-slave configuration. Under this arrangement, one process is designated as the master process. This master process then creates other processes, assigns these processes tasks, and monitors their progress. The slaves, in turn, perform the calculations required to complete their assigned task and return that information to the master. As a result of many slaves working on a variety of tasks, the overall computation load on a single processor is greatly reduced.

2-4-2 Message Passing Interface (MPI)

One of the limiting factors of parallel processing is the communication between the multiple processors. As the concepts of parallel processing developed, many significant applications were modeled into the parallel-programming paradigm. However, a different vendor created each application and these different vendors implemented their own variant of the parallel-programming paradigm. Over time, the importance of having a standardized method for implementing parallel processing into applications was recognized and the Message Passing Interface (MPI) effort was begun.

The goal of MPI is simply to develop a widely used standard for writing message-passing programs.[42] MPI is not, itself, an application, but rather a standard that has been defined for use in the development of message passing applications. By having an established standard message passing applications become much more portable and increase in ease of use. Furthermore, the definition of a message passing standard provides vendors with a clearly defined base set of routines that they can implement efficiently.

2-4-2-1 History of MPI[43]

The development of the MPI standard involved about 60 people from 40 organizations including most of the major parallel-processing vendors, as well as researchers from universities, government laboratories, and industry. The standardization effort was originally begun at the Workshop on Standards for Message Passing in a Distributed Memory Environment sponsored by the Center for Research on Parallel Computing, held April 29-30, 1992, in Williamsburg, Virginia. At this workshop, the basic features essential to a standard message-passing interface were discussed and a working group was established to continue the process.

Dongarra, Hempel, Hey, and Walker put forth the first preliminary draft, known as MPI-1 a few months after this workshop in November of 1992. Although not fully completed until February of 1993, the MPI-1 document embodied the main features that were identified at the Williamsburg workshop and was successful in its intent to promote interest in the area of standardization. In November of 1992, a meeting of the MPI

---

[42] Snir, Marc, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference,* Cambridge, Massachusetts: The MIT Press, 1996, p. 3.

working group was held in Minneapolis, Minnesota. At this meeting, the Message Passing Interface Forum (MPIF) was formalized and a goal of producing a draft MPI standard by the fall of 1993 was set. This goal was eventually met and the draft MPI standard was presented at the Supercomputing '93 conference in November of 1993. Following the 1993 conference, the Version 1.0 of the standard was released on May 5, 1994.

Beginning in March of 1995, the MPIF reconvened to correct errors and make clarifications to the Version 1.0 document. Version 1.1 that contained some minor changes was subsequently released in June of 1995. MPI-2 meetings began in April of 1995 and met every six weeks until April of 1997. In April of 1997, the MPI-2 document was unanimously accepted.

2-4-2-2 MPI Implementations and MPICH

After the MPI standard had been defined, a number of groups, both commercial and educational, began procedures to develop applications that implemented the MPI standard into working code. At the current time, more than thirteen such implementations have been developed for a variety of platforms (see http://www-unix.mcs.anl.gov/mpi/ implementations.html).

One such implementation is known as MPICH. This is a portable implementation of the full MPI specification for a wide variety of parallel-computing environments. The Argonne National Laboratory and Mississippi State University developed MPICH with the intent of promoting the adoption of the MPI Standard by providing users with a free, high-performance implementation on a diversity of platforms. The software is freely

---

[43] Hebert, Shane. *Message Passing Interface (MPI) FAQ,* Obtained online at

available and can be downloaded at the MPICH official web site: http://www.mcs.anl.gov/mpi/mpich/download.html.

The specific details of how to implement and use MPICH to perform parallel applications are too involved to enumerate in this thesis. Rather, the interested reader is referred to the *User's Guide for mpich, a Portable Implementation of MPI* by William Gropp and Ewing Lusk.[44]

---

http://www.erc.msstate.edu/mpi/mpi-faq.html on April 10, 1999.

[44] Mathematics and Computer Science Division, Argonne National Laboratory, ANL-96/6, 1996.

[This Page Intentionally Left Blank]

Chapter 3 Optimization Theories and Techniques

In addition to a general knowledge of the fundamental concepts of astrodynamics, a general knowledge of the fundamental concepts of optimization is also essential to an understanding of the applications and results presented in this thesis. In an effort to impart such an understanding, this chapter contains a discussion of a number of optimization techniques that were essential to this work.

3-1 Optimization Terms and Fundamentals

Prior to a detailed description of the specific optimization techniques used, it is first necessary to define some terms that are common to all optimization techniques. This section attempts to enumerate such terms.

- State Variables

  The state variables are variables that are used to describe the condition of the system at a given time. Throughout this work they will typically be labeled $x_1, x_2, \ldots x_n$.

- Control Variables

  The control variables are those variables that can be modified and which, when changed, will effect a corresponding change in the system. These are usually the variables for which one is attempting to solve. They are typically labeled $u_1, u_2, \ldots u_m$.

- Performance Measure or Objective Function

  The most fundamental part of an optimization problem is the objective function or performance measure. The objective function is the mathematical representation of the performance value one is trying to optimize, expressed as a combination of the state and control variables. Throughout this work, the objective function will be referenced as J.

- Constraints

  Restrictions on the values that the variables or objective function can take are called constraints.

It should be noted that the definitions presented above are for a certain class of optimization problems known as optimal control problems. There is a second class of simpler optimization problems in which the control variables are not present. In this case, the variables defined as the state variables become the variables to be optimized.

3-2 Analytical Optimization Theories

As the ability to find the optimal solution to a given problem is a highly desirable skill, the process that would allow one to do so has been studied for many years. The result of those years of study is a number of optimization techniques that can be applied to a variety of problems and yield the optimal solutions. These techniques can be broken into two different classifications: analytical optimization techniques and numerical optimization techniques. A number of examples of each of the techniques are presented in the following sections. This section presents concepts and examples relating to specific analytical techniques. A section detailing the concepts and theories relating to a variety of numerical optimization methods then follows this presentation.

3-2-1 Calculus Concepts

One of the most common analytical techniques is simply the application of calculus to optimization problems. This technique does not work for optimal control problems, but as it is the basis for most other techniques, it is presented briefly.

Given a function f(x), in order to find a maxima or minima of that function, it can easily be shown that a requirement for optimality is that the slope of f(x), evaluated at point x, must be equal to zero. This is best seen by contradiction and study of the cases where the derivative is not zero. If, for example, the first derivative of f(x) is positive ($df/dx > 0$), a small positive change in x will lead to a larger f(x) value, eliminating the possibility that x is a maximizing value. In a similar manner, a small negative change in x will lead to a smaller f(x) value, thereby eliminating the possibility that x is a minimizing value. The only way for x to be a minimizing value is for the slope to be zero. A similar result can be obtained when the first derivative of f(x) is negative. Therefore, a first requirement for optimality of x relative to f(x) is that the first derivative be equal to zero. This condition can be summarized as follows:

$$\frac{df(x)}{dx} = 0 \qquad \text{Equation 3-1}$$

If the first condition is met, it is also possible to determine whether the given solution is a minima or a maxima by using the second derivative. Although the proof is slightly more detailed (and therefore not included here), it can be shown that if the second derivative of f(x) with respect to x is positive when evaluated at point x, the value of f(x) at point x is a minima. On the other hand, if the second derivative is negative when evaluated at point x, the value of f(x) at point x is a maxima.

3-2-1-1 Functions of More than One Variable

The logic presented above in section 3-2-1 is also valid for functions that have more than one variable as an argument. However, the calculus for higher order problems changes slightly and this results in corresponding changes to the conditions listed above. The necessary condition for a maxima or minima for a function of two variables resulting from the change in problem order is listed below:

$$\frac{\partial f}{\partial x_1} = 0$$
$$\frac{\partial f}{\partial x_2} = 0$$

Equation 3-2

It becomes much more challenging to find a condition which allows one to definitively say whether or not the point (x_1, x_2) is a maximizing or minimizing value when more than one variable is involved. Although the logic is the same, the math involved is more complicated. Therefore, the details are avoided here and only the resulting sufficient conditions are presented. These conditions can be summarized as follows:

Point (x_1, x_2) is a relative maximum if:

$$\frac{\partial^2 f}{\partial^2 x_1} < 0 \text{ and } \frac{\partial^2 f}{\partial^2 x_1}\frac{\partial^2 f}{\partial^2 x_2} > \frac{\partial^2 f}{\partial x_1 \partial x_2}$$

Equation 3-3

Point (x_1, x_2) is a relative minimum if:

$$\frac{\partial^2 f}{\partial^2 x_1} > 0 \text{ and } \frac{\partial^2 f}{\partial^2 x_1}\frac{\partial^2 f}{\partial^2 x_2} > \frac{\partial^2 f}{\partial x_1 \partial x_2}$$

Equation 3-4

Point (x_1, x_2) a point of inflection or saddle point if:

$$\frac{\partial^2 f}{\partial^2 x_1}\frac{\partial^2 f}{\partial^2 x_2} < \frac{\partial^2 f}{\partial x_1 \partial x_2}$$

Equation 3-5

3-2-1-2 Lagrange Multipliers

Situations may also occur in which a function f depends upon variables which are not independent, but are related by one or more constraint conditions (i.e. $x_1 + x_2 = \alpha$). A concept developed by Lagrange becomes useful in this situation. The basis of this method is to introduce one new variable (λ) for each constraint condition which must be met. The introduction of these new variables allows for the formation of an entirely new objective function that can be created as follows:

$$J(x_1, x_2, \ldots x_n) = f(x_1, x_2 \ldots x_n) + \lambda_1 g_1(x_1, x_2 \ldots x_n) + \lambda_2 g_2(x_1, x_2 \ldots x_n) + \ldots \lambda_m g_m(x_1, x_2 \ldots x_n)$$

Equation 3-6

where:

n = number of state variables

m = number of constraints

$g(x_1, x_2, \ldots x_n)$ = constraint function

The function J is known as the adjoined objective function. After it has been formed, necessary conditions similar to those defined in section 3-2-1-1 can then be applied. The result of this application will be n equations and n + m unknowns. However, the m constraint functions can be used which results in a system of n + m equations and n + m unknowns. This system can then be solved for the variables in question as well as the newly introduced Lagrange multiplier values.

3-2-2 Variational Calculus

The calculus methods defined in the previous sections work well for functions of single or even multiple variables. However, situations often arise, specifically in the field of optimal control, in which the object to be optimized is not a set of variables, but rather a set of functions. For example, rather than finding the optimal input into a system at a given time, one might wish to find the optimal control input over an entire length of time. This modification to the desired result causes the objective function to become, in essence, a function of a function, or what is termed a functional. To handle problems of this sort, variational calculus has been developed.

Variational calculus is actually one of the oldest means of solving optimization problems. However, although its history dates back to the ancient Greeks, it was not until the 17th century that Sir Isaac Newton was able to make any substantial progress. Using principles of variational calculus, he was able to determine the shape of a body moving in air that encounters the least resistance.[45] Another infamous problem in the area of calculus of variations is known as the brachistochrone problem. This problem is to find the shape of a wire that causes a bead, under the influence of gravity, to move from point A to point B in minimum time. This problem was first posed by Johann Bernoulli in 1696 and the solution, a cycloid lying in the vertical plane, is credited to Johann Bernoulli, Newton, and L'Hospital.[46]

---

[45] Kirk, Donald E. *Optimal Control Theory: An Introduction,* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1970, p. 107.
[46] Kirk, Donald E. *Optimal Control Theory: An Introduction,* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1970, p. 107.

To understand the principles behind the calculus of variations, it is useful to look at a simple variational problem. To analyze a simple problem, an objective functional must first be defined. This is done as follows:

$$J(x) = \int_{t_0}^{t_f} g(x(t), \dot{x}(t), t)dt \qquad \text{Equation 3-7}$$

The notation used in the definition of the objective functional can be explained as follows. J(x) means that J is a functional of the function x. On the other hand, g is a function that assigns a real number to the point $(x(t), \dot{x}(t), t)$. It is assumed that g has continuous first and second partial derivatives with respect to all of its arguments and that t_0 and t_f are fixed.

If the objective were a simple function, as opposed to a functional, the next step would be simply to take the partial derivatives of J with respect to all of the variables and set them equal to zero as described previously in section 3-2-1-1. Despite the change from a function to a functional, finding the first derivatives is essentially what still must be done. However, since the variables in question are functions as opposed to variables, this process is termed finding the first variation of the functional as opposed to the first derivative. The first variation of J for the objective function presented above is:

$$\delta J = \int_{t_0}^{t_f} \left(\frac{\partial g}{\partial x}\delta x + \frac{\partial g}{\partial \dot{x}}\delta \dot{x} \right) dt \qquad \text{Equation 3-8}$$

Note that there are not any variations with respect to t_0 or t_f since they were specified to be fixed. In the more general case, they would also add a contribution to the first variation of J.

If the two variations δx and $\delta \dot{x}$ were completely independent, the corresponding coefficients of each variation could simply be set to zero and the necessary conditions for a maximum or minimum would have been derived. However, since both δx and $\delta \dot{x}$ are closely coupled variations, one of the terms must be rewritten in terms of the other. This can be done through integration by parts as detailed below:

$$\int_{t_0}^{t_f} \frac{\partial g}{\partial \dot{x}} \delta \dot{x} dt = \int_{t_0}^{t_f} \frac{\partial g}{\partial \dot{x}} d\delta x = \left(\frac{\partial g}{\partial \dot{x}} \delta x \right)_{t_0}^{t_f} - \int_{t_0}^{t_f} \delta x d\left(\frac{\partial g}{\partial \dot{x}} \right) \qquad \text{Equation 3-9}$$

Since there are not any variations associated with t_f or t_0 the first term of the integration by parts reduces to zero. The second term can then be substituted back into Equation 3-8 to yield the following modified equation of the first variation of J:

$$\delta J = \int_{t_0}^{t_f} \left(\frac{\partial g}{\partial x} - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}} \right] \right) \delta x \cdot dt \qquad \text{Equation 3-10}$$

Now, because there are not any constraints on δx, the only way that δJ can be equal to zero (and hence, an extreme value) is if the coefficient of δx is equal to zero. Setting the coefficient equal to zero results in the first necessary condition for an extreme value, also known as the Euler-Lagrange equation:

$$\frac{\partial g}{\partial x} - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{x}} \right] = 0 \qquad \text{Equation 3-11}$$

Since the Euler-Lagrange equation is a nonlinear, ordinary, time varying, second order-differential equation, it is typically difficult to solve. It does, however, provide a necessary condition that must be met for a solution to be considered optimal. For the objective function presented in this section, it was assumed that the initial and final times

as well as the initial and final states were fixed. It can be proven however, that regardless of the boundary conditions, the Euler-Lagrange equation must always be satisfied.[47]

Although the addition of degrees of freedom into the boundary conditions does not change the Euler-Lagrange equation, it does introduce additional necessary conditions into the solution process. However, unlike the Euler Lagrange equation that is constant between cases, these additional conditions vary, depending upon the case being studied. As the intent of this chapter is to provide a brief overview of a number of optimization techniques, the specific equations that result from each variation of the above problem are not presented. The results from the varying cases have been computed by a number of authors and can be found in a number of textbooks dealing with the subject.

3-2-2-1 Functionals of More than One Function

In a manner similar to that of 3-2-1-1, situations may arise where it is desirable to maximize or minimize a functional that depends upon several independent functions. The problem studied above would now be written as follows:

$$J(x_1, x_2, \ldots x_n) = \int_{t_0}^{t_f} g\left(x_1(t), \ldots, x_n(t), \dot{x}_1(t), \ldots, x_n(t), t\right) dt \qquad \text{Equation 3-12}$$

The preceding equation can also be written using vector notation that greatly simplifies future equations. This is done below:

$$J(\bar{x}) = \int_{t_0}^{t_f} g\left(\bar{x}(t), \dot{\bar{x}}(t), t\right) dt \qquad \text{Equation 3-13}$$

---

[47] Kirk, Donald E. *Optimal Control Theory: An Introduction,* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1970, p. 131.

Using this form of the objective function and vector algebra that parallels the derivation from the previous section, the matrix representation of the Euler-Lagrange equation can be produced:

$$\frac{\partial g}{\partial \bar{x}} - \frac{d}{dt}\left[\frac{\partial g}{\partial \dot{\bar{x}}}\right] = [0] \qquad \text{Equation 3-14}$$

Again, this equation becomes a necessary condition for all problems regardless of boundary conditions. However, the additional conditions that must be met are very dependent upon boundary conditions. Kirk has developed a table that summarizes the majority of these cases.[48]

3-2-2-2 Variational Approach to Optimal Control Problems

As was the case with general problems that could be solved through the application of calculus, the theory of calculus of variations as explained above is useful, but it would be even more useful if problems that are constrained in some manner could be solved using this technique. In fact, the solution process for constrained variational calculus problems is very similar to the Lagrange multiplier method used in non-variational calculus problems. This section briefly outlines the method.

3-2-2-2-1 Theory Description

The distinguishing factor between optimal control and simple variational calculus problems is the introduction of constraints, specifically state constraints. The functions for which one is attempting to solve are no longer arbitrary. Instead the states are usually constrained in the following manner:

---

[48] Kirk, Donald E. *Optimal Control Theory: An Introduction,* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1970, p. 151.

$$\dot{\bar{x}}(t) = \bar{a}(\bar{x}(t), \bar{u}(t), t) \qquad \text{Equation 3-15}$$

The objective then becomes to find the control vector (u) that minimizes some function J, and produces a state history (x(t)) that satisfies the state constraints detailed above. For this derivation, J will be given the following form:

$$J(\bar{u}) = h(\bar{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\bar{x}(t), \bar{u}(t), t)\,dt \qquad \text{Equation 3-16}$$

The only difference between this objective function and the functionals considered in section 3-2-2 is the term involving the final states and final time. Previously, it was assumed that the final state was fixed. For this application, the final state has become a measurable quantity in the objective function. The initial state and time, however, are still considered to be fixed values.

At this point the process necessary to solve this problem parallels the process used to solve simple functions with constraints. Lagrange multipliers must be introduced and the first variation must be taken and set equal to zero. Through this process, necessary conditions for an extremal to this problem can be found.

The augmented objective function is defined as follows:

$$J(\bar{u}) = h(\bar{x}(t_f), t_f) + \int_{t_0}^{t_f} \left(g(\bar{x}(t), \bar{u}(t), t) + \bar{\lambda}(t)^T \left[\bar{a}(\bar{x}(t), \bar{u}(t), t) - \dot{\bar{x}}(t) \right] \right) dt \qquad \text{Equation 3-17}$$

By defining a new function known as the Hamiltonian and applying the calculus of variations, the first variation can then be found. The Hamiltonian (H) and the first variation δJ_a are listed below:

$$H(\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t) = g(\bar{x}(t),\bar{u}(t),t) + \bar{\lambda}(t)^T \bar{a}(\bar{x}(t),\bar{u}(t),t) \qquad \text{Equation 3-18}$$

$$\delta J_a(\bar{u}) = \left[\frac{\partial h}{\partial \bar{x}}(\bar{x}(t_f),t_f) - \bar{\lambda}(t_f)^T\right]\cdot \delta x_f + \left[H(\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t) + \frac{\partial h}{\partial t}(\bar{x}(t_f),t_f)\right]\cdot \delta t_f$$
$$+ \int_{t_0}^{t_f}\left[\left(\frac{\partial H}{\partial x} + \dot{\bar{\lambda}}(t)^T\right)\cdot \delta(t)\bar{x} + \frac{\partial H}{\partial \bar{u}}(\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t)\cdot \delta\bar{u}(t) + \delta\bar{\lambda}(t)^T\left(\bar{a}(\bar{x}(t),\bar{u}(t),t) - \dot{\bar{x}}(t)\right)\right]dt$$

Equation 3-19

The first variation must vanish over the entire integral regardless of the boundary conditions, in order for the solution to be considered optimal. In order to vanish for all possible δ's, all of the coefficients on those δ's must go to zero. This results in the following conditions, all of which are necessary for optimality.

- State Equations

  By setting the coefficients of the δλ vector equal to zero, the original state equation constraints are recovered.

$$\left(\bar{a}(\bar{x}(t),\bar{u}(t),t) - \dot{\bar{x}}(t)\right) = 0 \qquad \text{Equation 3-20}$$

- Co-state Equations

  In a similar manner, setting the coefficients of the δx vector equal to zero gives a differential equation relationship for the λ vector. As this λ vector is often termed the co-state, this equation is also known as the co-state equation.

$$\dot{\bar{\lambda}}(t) = -\frac{\partial H}{\partial x}(\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t) \qquad \text{Equation 3-21}$$

- Control Condition

  In order for the δu term to vanish, the following constraint on the control must be met:

$$\frac{\partial H}{\partial \bar{u}}\left(\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t\right)=0 \qquad \text{Equation 3-22}$$

- Transversality Condition

  The transversality condition arises in problems in which the terminal time is not fixed (such as this one). It is found by setting the δt_f coefficient to zero:

$$H\left(\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t\right)+\frac{\partial h}{\partial t}\left(\bar{x}(t_f),t_f\right)=0 \qquad \text{Equation 3-23}$$

- Terminal Constraints

  The final condition is a direct result of the need for an adequate number of boundary conditions to make the problem solvable. Because the terminal state vector was allowed to vary, a condition has arisen in which the final components of the co-state vector are specified. If only portions of the state vector components are specified, the following relationship must hold for all of the components that are not specified.

$$\bar{\lambda}(t_f)=\frac{\partial h}{\partial \bar{x}}(\bar{x}(t_f),t_f) \qquad \text{Equation 3-24}$$

- Initial Conditions

  Although not a result of the derivation, it is also important to remember the fact that the initial conditions were specified as follows:

$$\bar{x}(t_0)=\bar{x}_0 \qquad \text{Equation 3-25}$$

The result of this section is a set of 2n+m+1 equations that can be used to solve for 2n+m+1 variables as summarized in Table 3-1

Table 3-1 Variable and Condition Summary for Optimal Control Problems

Variable to Be Solved For	Number	Corresponding Condition	Number
Constants of Integration from State Equation	n	Initial Conditions	n
Constants of Integration from the Co-state Equation	n	Terminal Constraints	n
Control Elements	m	Control Condition	m
Terminal Time	1	Transversality Condition	1

3-2-2-2-2 Primer Vector Theory: An Application of Optimal Control Techniques

The application of optimal control techniques to this thesis can be seen in the development of primer vector theory. Lawden first developed this theory in 1963 in his book *Optimal Trajectories for Space Navigation.*[49] In this work, Lawden used variational calculus techniques to find "the trajectory a rocket must follow if it is to accomplish some specified mission in an optimal manner as judged against some criterion of a quantitative nature."[50] One of his most important contributions was the development of four necessary conditions that must be met for the solution to an impulsive rendezvous problem to be optimal.

Through application of these conditions, Shah developed an optimization algorithm known as the Automated Station-Keeping Simulator (ASKS) which he used to compute optimal fuel budgets for the Ellipso™ Borealis sub-constellation.[51] This work becomes of interest to the present study as a similar problem is studied in Chapter 5. So that the reader might have a background to Shah's work, a brief derivation of the primer vector theory and the four Lawden conditions are presented here.

---

[49] Washington, D.C.: Butterworth, Inc., 1963.
[50] Lawden, D. F. *Optimal Trajectories for Space Navigation*, Washington, D.C.: Butterworth, 1963, p. 3.
[51] Shah, Naresh, H. *Automated Station-Keeping for Satellite Constellations,* CSDL-T-1288, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1997.

The equations of motion for a satellite are described by

$$\begin{aligned} \dot{R} &= V \\ \dot{V} &= \frac{\beta c}{m}\ell + G \\ \dot{m} &= \beta \end{aligned} \qquad \text{Equation 3-26}$$

where:

R = the position vector

V = the velocity vector

G = the gravity vector,

ℓ = the direction cosine vector of the thrust

β = the mass flow rate

c = the characteristic velocity

m = the mass of the satellite.

The corresponding constraints on the direction cosine vector and mass flow rate are:

$$\begin{aligned} \ell^T \ell &= 1 \\ \beta(\beta_{\max} - \beta) &= \alpha^2 \geq 0 \end{aligned} \qquad \text{Equation 3-27}$$

where:

α = a slack variable. A slack variable of α=0 implies that the mass flow rate will be zero or maximum. This is the equivalent of a bang-bang controller.

Using the methods of variational calculus, the Hamiltonian function for this problem can be formed:

$$H = \lambda^T \left(\frac{\beta c}{m} \ell + G \right) + \varphi^T V - \eta \beta - \mu_1 \left(\ell^T \ell - 1 \right) - \mu_2 \left[\beta \left(\beta_{\max} - \beta \right) - \alpha^2 \right]. \quad \text{Equation 3-28}$$

The vectors λ and φ and the scalars μ_1, μ_2, and η are time varying Lagrange multipliers (equivalent to the λ vector used previously). As has been shown, variational calculus states that the necessary conditions for the trajectory that optimizes the cost function J are

$$\begin{aligned}
\dot{\lambda}^T &= -\frac{\partial H}{\partial V} = -\varphi^T \\
\dot{\varphi}^T &= -\frac{\partial H}{\partial R} = -\lambda^T \frac{\partial G}{\partial R} \\
\dot{\eta} &= -\frac{\partial H}{\partial m} = \frac{\beta c}{m^2} \lambda^T \ell \\
0 &= -\frac{\partial H}{\partial \ell} = -\frac{\beta c}{m} \lambda^T + 2\mu_1 \ell^T \\
0 &= -\frac{\partial H}{\partial \beta} = -\frac{c}{m} \lambda^T \ell + \eta + \mu_2 \left(\beta_{\max} - 2\beta \right) \\
0 &= -\frac{\partial H}{\partial \alpha} = -2\mu_2 \alpha
\end{aligned} \quad \text{Equation 3-29}$$

The term primer vector was introduced by Lawden to describe the λ vector in these equations. Through a variety of assumptions and simplifications, (for which the interested reader is referred to Lawden's work), Lawden developed the following four necessary conditions, known today as Lawden's conditions[52].

---

[52] Lawden, D. F. *Optimal Trajectories for Space Navigation*, Washington, D.C.: Butterworth, 1963, p. 63.

1) The primer vector and its first time derivative must be continuous everywhere.

2) Whenever the rocket motor is operative, the thrust must be aligned with the primer which must have a certain constant magnitude P.

3) The magnitude p of the primer must not exceed P on any coasting arc.

4) The time derivative of the primer vector magnitude must be zero at all interior junction points separating coasting arcs.

3-3 Numerical Methods

The calculus concepts presented in the previous section are fundamental theories behind optimization, however in most cases they can not be directly implemented or programmed to solve an optimization problem. For situations in which direct programmable implementation is desired, a number of numerical optimization algorithms and techniques have been developed. Some of these techniques rely upon the fundamental theories presented above, while others do not.

For this thesis, a number of these techniques were applied to astrodynamic applications. Those optimization techniques that have relevance to this thesis are presented in this section. It should be noted however, that this list is in no way exhaustive. There are a large number of optimization techniques in existence, and new advances are made in the field on a routine basis. However, the techniques studied and presented below do represent a diverse slice of the available optimization techniques.

3-3-1 Dynamic Programming and Principle of Optimality

Dynamic programming is an optimization technique that is useful for making a sequence of interrelated decisions. It provides a systematic procedure for determining the

optimal combination of decisions.[53] Unlike some other methods, there does not exist a standard mathematical formulation of a dynamic programming problem. Rather, dynamic programming is made up of some general concepts, and the actual equations must be developed for each individual problem. For this thesis, dynamic programming was not actually used in the optimization of constellation design and/or maintenance. However, some of the fundamental concepts behind dynamic programming were relied upon heavily. These concepts are summarized in this section.

3-3-1-1 Greedy Algorithms

Inherent to optimization techniques that are used to make interrelated decisions is the concept of greediness. Greedy strategies are those that attempt to find the optimal path from a current state to a desired state without worrying about the effect of the chosen path on subsequent decisions. Very few optimization techniques can actually be solved through the application of greedy algorithms[54]. However, as their concept is important to this thesis as well as dynamic programming, they are briefly explained, by example, here.

Take for example, Figure 3-1 Example of Greedy Optimization Strategy. Assume the current state is point A, the objective is to move to point E, and the numbers are the costs to move from one state to another along a given path. It is easy to see that A-D-E is the optimal path. However, a greedy strategy will not find this path. Instead, starting at point A, a greedy strategy will attempt to find the optimal point from the current state, A, to one of the intermediate states (B, C, or D). Since the least costly choice is to move from A to B, this is the path the greedy algorithm will choose. No attention will be paid

---

[53] Hillier F. S. and G. J. Lieberman. *Introduction to Operations Research,* New York, New York: McGraw-Hill, Inc., 1995, p. 424.

to the fact that this choice forces the completion of the problem along route B-E: the most costly of the allowable paths.

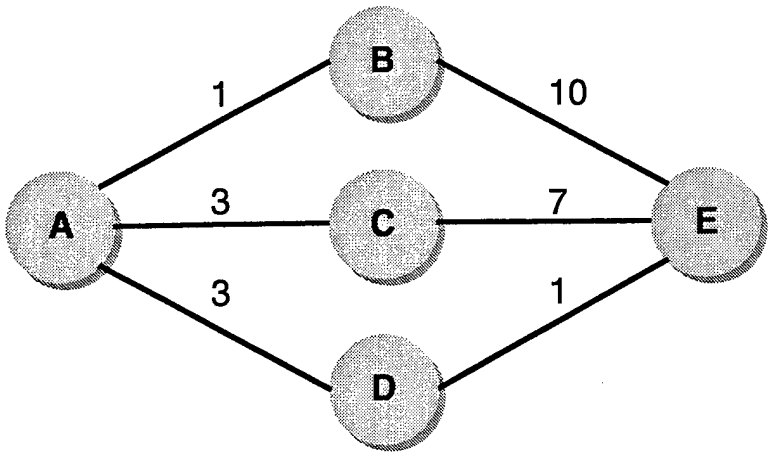


Figure 3-1 Example of Greedy Optimization Strategy

3-3-1-2 Principle of Optimality

For the problem in Figure 3-1, finding the optimal solution via trial and error is a fairly simple process. However, by increasing the number of nodes drastically, the number of possible routes will be increased and the difficulty of calculating all possible routes will also be increased. Rather than attempt trial and error on problems with a large number of nodes, dynamic programming and the principle of optimality have been found to be useful.

Dynamic programming starts with a small portion of the original problem and finds the optimal solution for this small portion. Then through the principle of

---

[54] Hillier F. S. and G. J. Lieberman. *Introduction to Operations Research,* New York, New York: McGraw-Hill, Inc., 1995, p. 363.

optimality, it is able to gradually enlarge the problem until a solution for the entire problem is found.

Exactly what is the principle of optimality that makes the dynamic programming solution path possible? Bellman, the developer of dynamic programming explained it this way:

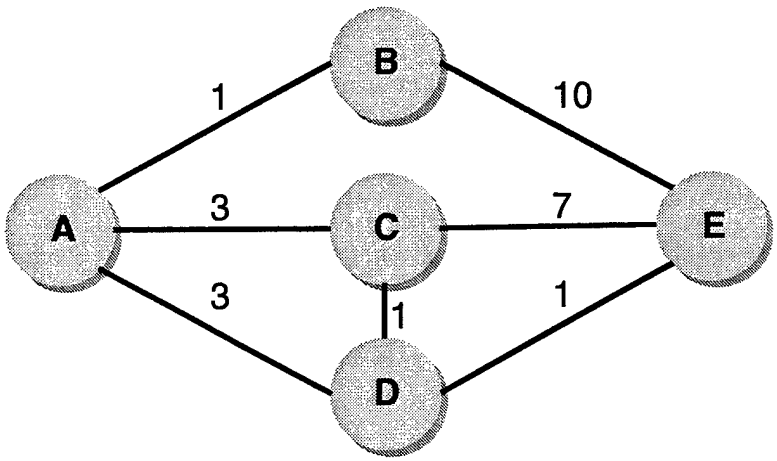
> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.[55]

In other words, the optimal decision at a current state is independent of the choices that led to arrival at that state.

As with greediness, the principle of optimality is best proven by example. Figure 3-2 contains the same problem as previously presented in Figure 3-1 with the addition of a path between C-D. It has already been shown that the optimal path from A to E is via D (the addition of the C-D link does not change that). The principle of optimality simply states that if A-D-E is the optimal path from A to E then D-E must be the optimal path from D to E. Other paths from D to E are possible (i.e. D-C-E), but if D-E were not the optimal path from D to E, then A-D-E would not be the optimal path from A to E.

---

[55] Kirk, Donald E. *Optimal Control Theory: An Introduction,* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1970, p. 54.



Cost A-E (4) = Cost A-D (3) + Optimal Cost D-E (1)

Figure 3-2 Problem for Which Principle of Optimality can be Demonstrated

The principle of optimality seems almost trivial, but recognition of it allows for important statements to be made which in turn form the basis of dynamic programming. First, due to the principle of optimality, the optimal policy for the remaining stages of a problem is independent of the decisions that forced arrival to a given state. In other words, the optimal path from the current state to the end is always the optimal path from the current state to the end, regardless of how one arrived at the current state. In the case above, D-E is always the optimal path from D-E no matter how one arrives at point D.

The second important result from the principle of optimality is that a recurrence relationship between various stages of a problem can be written.

$$C^*_{\alpha x_i h} = J_{\alpha x_i} + J^*_{x_i h} \qquad \text{Equation 3-30}$$

where:

$C^*_{\alpha x_i h}$ = the optimal cost to go from α to h via x_i

$J_{\alpha x_i}$ = the cost to go from α to x_i

$J^*_{x_i h}$ = the optimal cost to go from x_i to h (by any allowable path)

Also, the optimal decision at α can be found by:

$$J^*_{\alpha h} = \min\left\{C^*_{a x_1 h}, C^*_{\alpha x_2 h}, \ldots, C^*_{\alpha x_i h}\right\} \qquad \text{Equation 3-31}$$

These two equations, which are made possible by the principle of optimality, form the basis of dynamic programming. The dynamic programming process is then a simple iterative process. The decisions nearest the destination (h) are considered first. The optimal trajectories from these states (h-1) to the end state (h) are found and recorded. The process then moves backward one step and finds the optimal trajectories from the next states (h-2) to the end. This is done via the fact that the cost from these states to the end is simply the cost from the h-2 level states to a given h-1 state, plus the optimal cost from that h-1 state to the end (via the principle of optimality). The process can be iterated any number of times until an optimal path from the initial state to the desired state has been constructed.

3-3-2 Localized Methods

For problems in which the objective is to optimize a sequence of decisions or states, dynamic programming works well. However, the objective is often not to optimize a sequence of decisions, but rather to find the optimal value of a certain function. As a function maps a surface in n-dimensional space, the objective of this type of optimization problem can be seen as finding the extreme location on that mapped surface. To meet this objective, a number of methods, the majority of which can be classified as localized methods, have been developed.

Localized methods rely heavily upon the derivative or gradient information of a surface to find the maxima or minima of that surface. The optimization is accomplished by looking at points in the immediate vicinity of the current location and moving some distance in some direction that meets a predefined requirement, such as the direction of greatest descent. The process is then repeated at the next point (i.e. the direction of greatest decent from that point is calculated). By continually applying the same rule at a successive number of points, the optimal point is eventually reached.

The main problem with localized methods is their tendency to converge upon local rather than global optima. Figure 3-3 illustrates the difference. Point A, a local minima, is optimal when compared to the points immediately surrounding it. However, if the scope of the optimization is broadened, it can easily be seen that Point B is the true minima. If the starting point and/or step size for a localized method was not defined properly, there is a high probability that the solution produced would be the local rather than the global optimum.

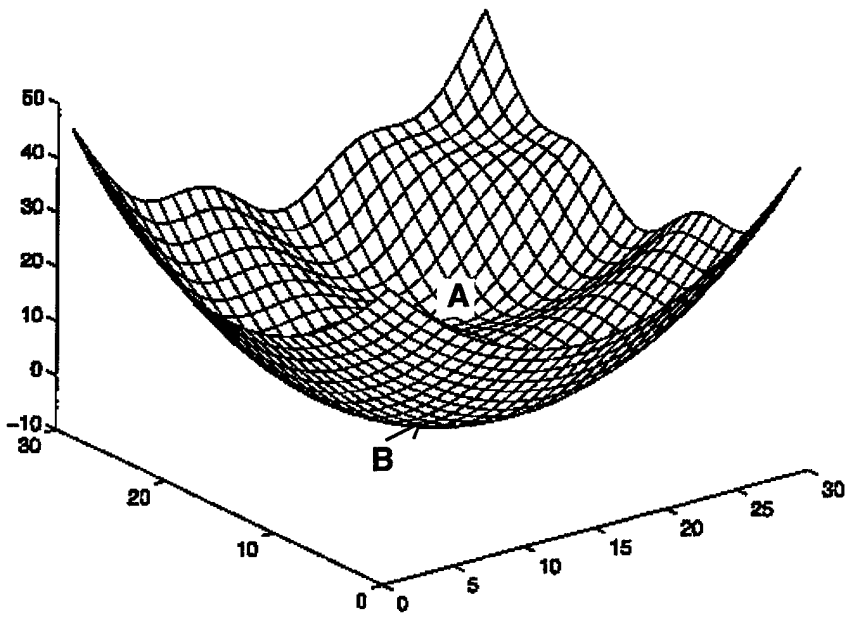


Figure 3-3 Local vs. Global Optimal Point[56]

A number of localized search algorithms have been developed and used successfully. A few of these are summarized in the following sections:

3-3-2-1 Gauss-Seidel Method

The basis of the Gauss-Seidel method is to minimize one coordinate axis at a time from the given starting point. The steps to the algorithm are presented below:

A. Initiate the algorithm at any point.

B. Minimize along the first coordinate axis.

C. Minimize along the next coordinate axis.

D. Minimize along remaining coordinate axes (one at a time until last direction has been minimized.)

---

[56] Feron, Eric. *Course Notes for Course 16.410: Introduction to Optimization and Decision Analysis*, Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 1998.

E. Iterate the process until the difference between resulting point and current point is less than a predefined tolerance.

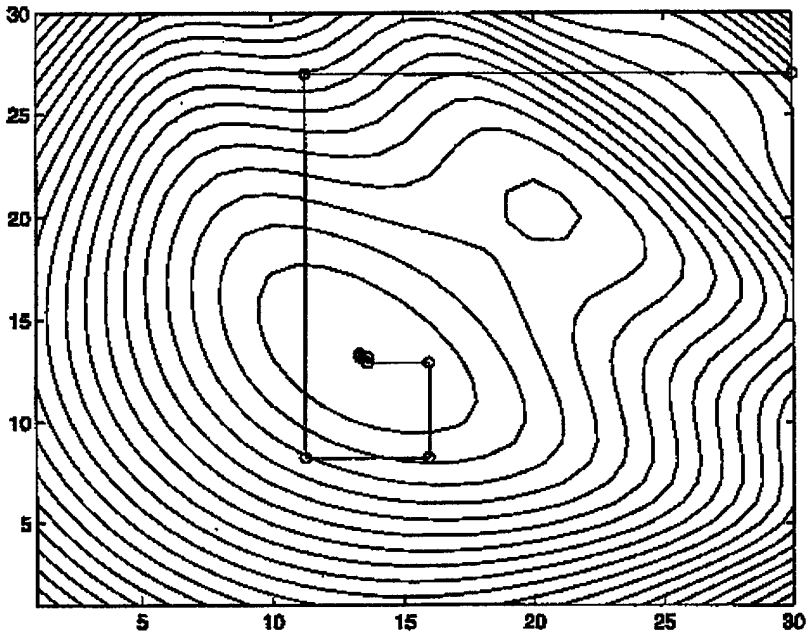


Figure 3-4 Application of the Gauss-Seidel Method[57]

A sample application of the Gauss-Seidel method to the surface presented in Figure 3-3 is shown in Figure 3-4. As this is a two-axis optimization, the algorithm continually switches from optimizing in the horizontal direction to optimizing in the vertical direction until it eventually converges at the global optimum. Note, however, that this solution is somewhat dependent upon the starting location. If the algorithm had started at a vertical value near the value of the local minima and minimized along the horizontal axis, the convergence would be to the incorrect solution.

3-3-2-2 Steepest Descent or Gradient Method

Rather than minimizing along one coordinate axis at a time, the gradient method first calculates the gradient or direction of steepest descent at the given point and then minimizes in that direction. As before, the steps to the algorithm are presented below along with Figure 3-5 which shows an application of the gradient method to the surface shown in Figure 3-3.

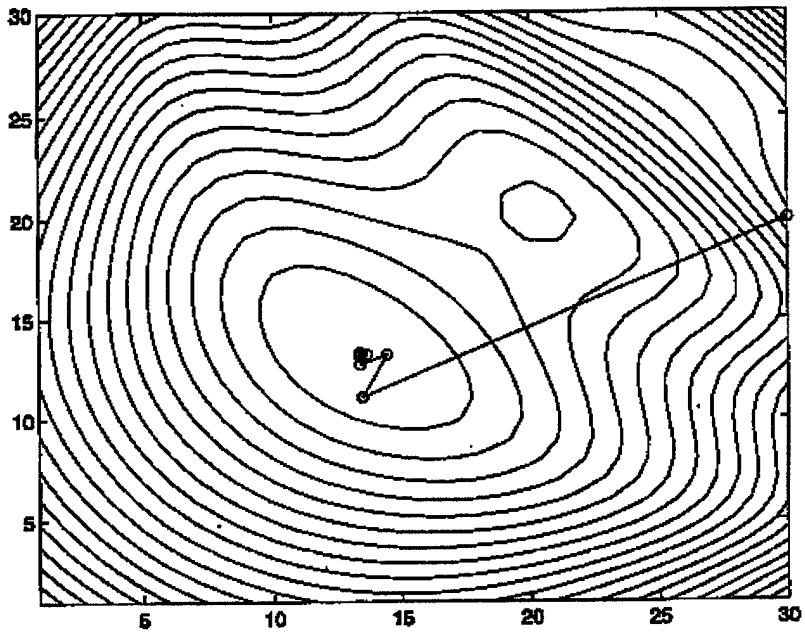


Figure 3-5 Application of the Method of Steepest Descent[58]

A. Initiate the algorithm at any point.

B. Calculate the direction of steepest descent from this point.

$$d = \nabla f(x) = -\frac{\partial f(x)}{\partial x} \qquad \text{Equation 3-32}$$

---

[57] Feron, Eric. *Course Notes for Course 16.410: Introduction to Optimization and Decision Analysis*, Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 1998.
[58] Feron, Eric. *Course Notes for Course 16.410: Introduction to Optimization and Decision Analysis*, Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 1998.

C. Minimize along the direction of steepest descent.

D. Iterate upon this process until a predefined tolerance is met.

Note that this method requires slightly fewer iterations than the Gauss-Seidel method, but this increase is due to the increase in available information. In order to apply this method, the gradient information must be available.

3-3-2-3 Newton's Method

By using even more information than just the gradient of the function, Newton's method is able to achieve even slightly faster convergence. This increase in convergence is accomplished by approximating the function to be minimized (f) by a vector Taylor series expansion, truncated at the quadratic terms:

$$f(x+d) = f(x) + \nabla f(x)^T d + \frac{1}{2} d^T \left(\nabla^2 f(x)\right) \cdot d \qquad \text{Equation 3-33}$$

where:

$$\nabla^2 f(x) = \frac{\partial^2 f(x)}{\partial x^2} = \text{Hessian Matrix}$$

Once the function is in quadratic form, the minimizing step (d) can easily be found through a simple partial derivative. The result, known as a Newton step is as follows:

$$d = -\nabla^2 f(x)^{-1} \nabla f(x) \qquad \text{Equation 3-34}$$

By using the direction and step defined by the Newton step, one pass will find the minimum of the quadratic form which was used to approximate f. For functions that are not exactly quadratic forms, the minimum of the quadratic approximation will vary from the actual minimum. However, by re-approximating the function quadratically at the

resulting point and recalculating the Newton step, the process will eventually converge to the minimum.

Figure 3-6 shows the results of application of Newton's method to the problem discussed previously. By comparing this figure with the gradient application shown in Figure 3-5, it can be seen that Newton's method does indeed reach the minimum in slightly fewer steps.

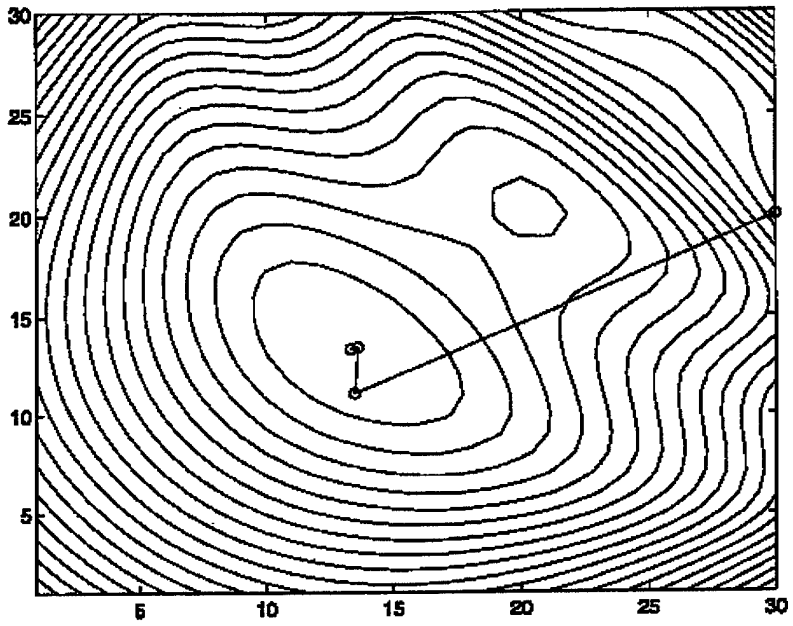


Figure 3-6 Application of Newton's Method[59]

3-3-2-4 Powell's Method

Although Newton's method appears to work well for minimizing functions, it has a major limiting factor. In order to successfully use the method, one must be able to successfully calculate not only the gradient, but also the Hessian matrix at a number of points. Powell's method, on the other hand, takes advantage of many of the same

principles of quadratic convergence, but is able to calculate the required step without requiring gradient or Hessian data. The basic steps to this method are as follows[60]:

A. Initialize the set of directions $\mathbf{u}_i$ to correspond to the principle axis vectors.

B. Save your starting position as $\mathbf{P}_0$

C. For i = 1,...,N (where N is the dimension of the search space), move $\mathbf{P}_{i-1}$ to the minimum along direction $\mathbf{u}_i$ and call this point $\mathbf{P}_i$.

D. For i = 1,...,N-1, set $\mathbf{u}_i$ to $\mathbf{u}_{i+1}$

E. Set $\mathbf{u}_N$ to $\mathbf{P}_N$ - $\mathbf{P}_0$, the average direction moved after trying all N possible directions

F. Minimize in this average direction and call this point $\mathbf{P}_0$.

G. Repeat the steps A through F until predefined convergence criteria are met.

This method achieves similar convergence characteristics as the Newton's method but does so without requiring explicit calculation of the Hessian matrix. This is a distinct advantage over Newton's method and therefore Powell's method was selected for application to many of the problems in this thesis (see section Chapter 4).

3-3-3 Non-localized Approaches

One of the biggest problems with localized approaches is the dependence upon starting point and step size definition. If these features are not chosen carefully, the localized algorithms may end up at local as opposed to global optimums. Figure 3-7 contains the results of the gradient techniques applied in the same manner as before to the

---

[59] Feron, Eric. *Course Notes for Course 16.410: Introduction to Optimization and Decision Analysis*, Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 1998.

identical problem (see Figure 3-5), however, the step size has been changed, slightly. The result is a convergence to the incorrect minimum.

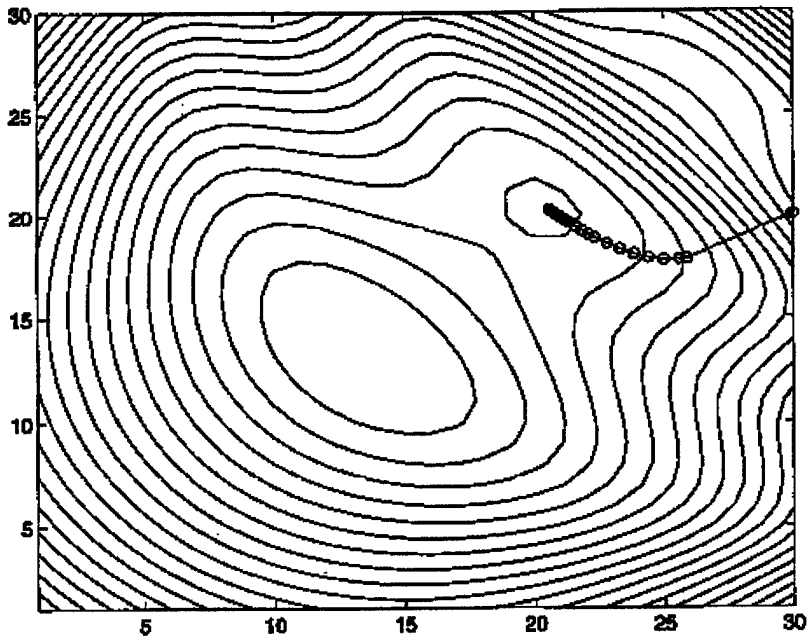


Figure 3-7 Gradient Method Convergence to Incorrect Solution[61]

One way to avoid convergence to the incorrect optimal point is to sample the surface at more than one point, or to perform the optimization from multiple starting points. This is the foundation, and also the advantage to non-localized or global optimization approaches.

A random search is one such global optimization technique. However, although sampling at multiple random locations may help avoid the problem of convergence to a local optimum, it introduces a new problem—the technique may never randomly choose the actual optimum.

---

[60] Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing—Second Edition,* New York, New York: Cambridge University Press, 1992, p. 409.

There is a compromise solution, however. That solution is to direct the randomness of the search in some manner. This idea, of directed randomness, is the foundation for a technique known as genetic algorithms. As the majority of the applications for this thesis relied heavily upon the use of genetic algorithms, the fundamentals behind this optimization scheme are discussed in some detail in the sections that follow.

3-3-3-1 Genetic Algorithms

Genetic algorithms were first introduced in the 1970's by John Holland and his students.[62] They have since been used to solve a number of optimization problems in a variety of fields. One of Holland's students, David Goldberg applied the technique to the gas-pipeline industry and since has created an excellent reference on genetic algorithms. This reference, *Genetic Algorithms in Search, Optimization, and Machine Learning*[63] forms the basis for the descriptions in this section and the interested reader is referred to it for further description of the concepts explained here.

3-3-3-1-1 Genetic Algorithm Differences from Other Optimization Methods

Before describing genetic algorithms in detail, it is first useful to study the differences (and corresponding advantages) between genetic algorithms and the more traditional optimization techniques outlined in the previous sections of this chapter. There are three main differences[64]:

---

[61] Feron, Eric. *Course Notes for Course 16.410: Introduction to Optimization and Decision Analysis*, Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 1998.
[62] Holland, J. H. *Adaptation in Natural and Artificial Systems,* Ann Arbor, Michigan: The University of Michigan Press, 1975.
[63] Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989.
[64] Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989, p. 7.

1. Genetic algorithms search from a population of points, not a single point. This is an advantage as it helps the algorithm avoid convergence to local minimum.

2. Genetic algorithms use objective function information, not derivatives or other auxiliary knowledge. Both the calculus methods and localized methods discussed previously require the ability to calculate the derivative in some manner. For simple problems, this is not an issue, but for problems which are difficult to express analytically, finding the derivative can be a challenge. As will be explained, the only information a genetic algorithm needs is the value of the objective function, evaluated at a given point, thus allowing for simpler implementation.

3. Genetic algorithms use probabilistic transition rules, not deterministic rules. In a manner similar to the advantage presented by maintaining a population of solutions rather than a single solution, by using probabilistic as opposed to deterministic rules, a genetic algorithm is able to avoid convergence to local minimum

These three differences combine to make the genetic algorithm a robust, global search tool.

3-3-3-1-2 Cycle of the Genetic Algorithm

Genetic algorithms are modeled after the Darwin natural selection principals of survival of the fittest. They imitate the natural selection process found in genetic evolution. This process is modeled in Figure 3-8:

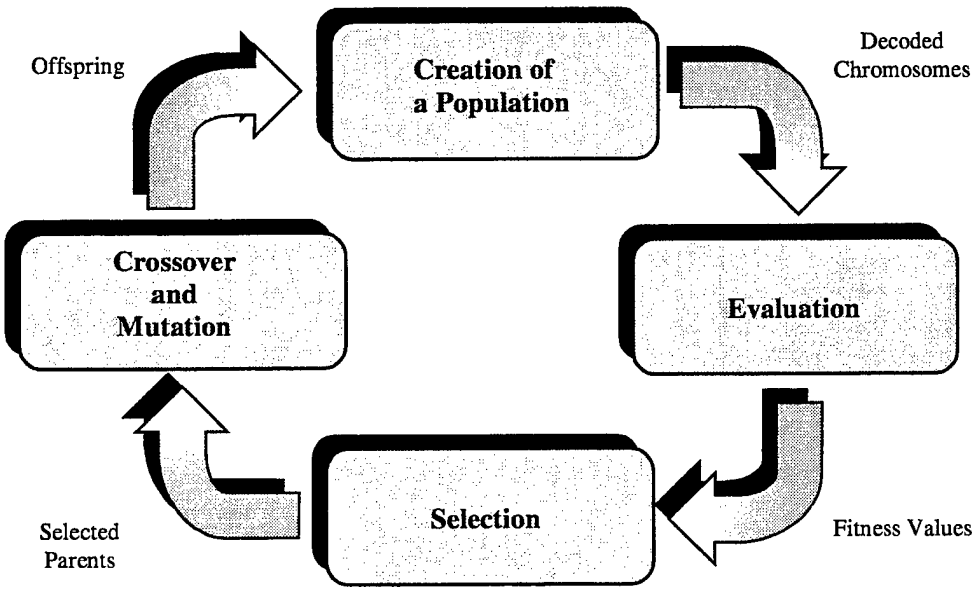


Figure 3-8 Cycle of Genetic Algorithm[65]

To gain a better understanding of genetic algorithms, it is useful to analyze each step of this cycle. The mechanics of the cycle are really very simple consisting of nothing more than copying strings and swapping partial strings. However, as the terminology can become slightly confusing, the following section provides details of each step of the cycle.

The first step of the cycle is the creation of all the members that form the first population. *Population* is simply the term used to describe a group of possible solution strings, while a *member* is the term used to describe an individual solution string from a population. Each member is a self-contained solution to the given problem as the values for all of the variables of interest have been coded in some manner (often binary) into that solution string. For the initial population, all members are generated randomly in order to

---

[65] Frayssinhes, E. *Investigating New Satellite Constellation Geometries with Genetic Algorithms,* Paper AIAA-96-3636, AIAA/AAS Astrodynamics Specialist Conference, San Diego, CA, 29-31 July 1996.

have a high amount of diversity in the initial population (to avoid convergence to local minima, as previously discussed.)

Following creation of the population, each member string is decoded to reveal the actual values of the variables of interest. These variables are then inserted into a predefined objective function and this objective function is evaluated using the variable contained in each string. The result of that evaluation is a numerical value of the objective function that can be associated with each string. Using the objective function values, a numerical value, termed a fitness value is assigned to each string. This *fitness value* can be thought of as a measure of the profit, utility, or goodness that one is attempting to optimize. These fitness values become important in the remaining steps of the cycle and in directing the randomness of the algorithm.

Using the fitness values for each member, a certain number of strings are selected for inclusion into the mating pool. This simply means that they are strings that have been chosen to make the members of the next generation or population. There are a number of ways in which this *selection* can be accomplished. Most often, however, these strings are selected in proportion to their fitness value (i.e. strings with higher fitness values are copied more times to the mating pool). By copying strings according to their fitness, strings with a higher value have a higher probability of contributing one or more offspring in the next generation. As the cycle is repeated, this allows for continual evolution to higher and higher fitness values.

Two other operators are also important contributors to the process of forcing evolution to higher and higher fitness values. These are the operators of *crossover* and *mutation* and they make up the fourth phase of the genetic algorithm cycle. Two

members of the mating pool are selected at random to enter this phase. After entering this phase they first undergo crossover. Crossover is simply an operator that switches some of the information from one string with the information in the corresponding locations of the other string. Randomness is preserved in this step as well, as the location for the crossover is randomly chosen each time that crossover is performed.

A number of crossover types have been developed, but two-point crossover is often used. In this form of crossover, a starting and ending location are both randomly generated with probability p_c. The variables that fall between these locations are then swapped and two new offspring for the next population are created. An example of two-point crossover can be seen in the following figure:

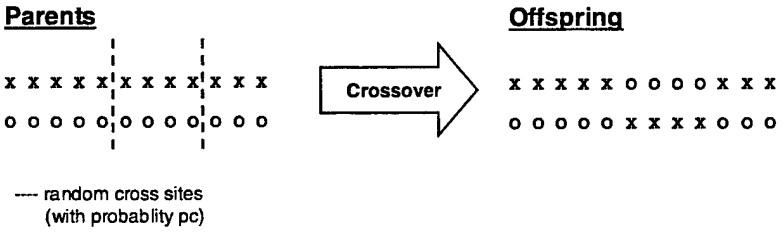


Figure 3-9 Two-Point Crossover Operation[66]

Before entering the next population and restarting the cycle, the new members must undergo one final operation: mutation. Mutation is simply the change of value of a random string position. It is applied independently to each element in each string with some probability p_m. It is useful in preventing the loss of some useful piece of genetic information during the crossover operation. Figure 3-10 illustrates this operation.

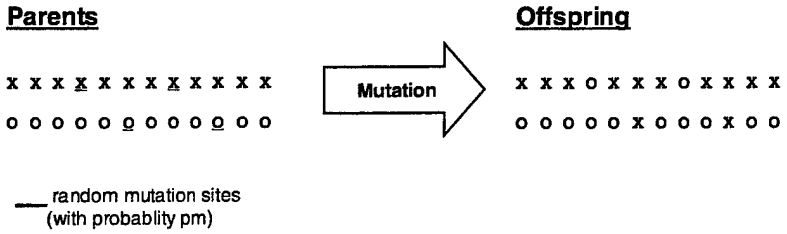


Figure 3-10 Mutation Operation[67]

3-3-3-1-3 Mathematical Foundations of the Genetic Algorithm

Although the operators that make up a genetic algorithm cycle seem random in definition and behavior, it can be shown that through continual application of these operators, the average fitness value of a population will converge to higher and higher values. However, in order to prove this point, a new term must be defined: schema.

A *schema* is a particular arrangement of bits at a particular location within a string. For example, if the strings which compose a population are binary in nature, and a * is used as a wildcard symbol, a string matches a particular schema if at every location a 1 in the string matches a 1 in the schema, a 0 in the string matches a 0 in the schema, and either value is found in the * locations. Therefore, a schema defined as 1*0* would be matched by the any of the following string: 1000, 1001, 1100, or 1101.

Schema can be thought of as the building blocks of a particular problem. By creating a string that is composed of all the correct building blocks (schema), the optimal

---

[66] Van Deventer, Paul G. *Flight Control Command Generation in a Real-Time Mission Planning System Using Constrained Genetic Optimization,* Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June, 1992, p. 28.

solution can be built. Therefore, by showing that the number of schemas with above average fitness values increases in each generation, the mathematical foundation of genetic algorithms can be demonstrated.

Two further definitions are required before proceeding with the derivation. These are the concepts of *schema order* and *schema defining length.* The order of a schema, denoted by o(H) is simply the number of fixed positions (in binary strings the number of 1's and 0's) present in the template. In the schema presented above (i.e. 1*0*) the order, o(H), is 2. The defining length $\delta(H)$ is then defined to be the distance between the first and last specified positions. For the 1*0* schema, the last specified position is 3 and the first is 1, therefore, the defining length $\delta(H) = 2$.

With all the required definitions in place, it is now possible to develop the fundamental theorem of genetic algorithms. This is accomplished by considering the combined effect of the selection, crossover, and mutation operators on the schemata contained within a population of strings.

First, it is necessary to start with selection. During selection, a string A_i is copied according to its fitness F_i, by being selected to the mating pool with probability $p_i = F_i/\Sigma F_j$. The expected number of copies of a string A_i in the mating pool is then given by np_i (where n = population size). If m(H,t) is defined to be the number of instances of a particular schema, H, contained in the population A(t) at time t, then the expected number of occurrences of the H schema in the next population A(t+1) can be written in a similar manner (if F(H) is the average fitness of the strings representing schema H at time t):

---

[67] Van Deventer, Paul G. *Flight Control Command Generation in a Real-Time Mission Planning System Using Constrained Genetic Optimization,* Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June, 1992, p 29.

$$m(H,t+1) = n \cdot m(H,t) \frac{F(H)}{\sum F_j}$$ Equation 3-35

By recognizing that $\Sigma F_j/n$ is simply the average fitness of the entire population, this representation can be reduced to the following form, known as the reproductive schema growth equation.

$$m(H,t+1) = m(H,t) \frac{F(H)}{\overline{F}}$$ Equation 3-36

From the reproductive schema growth equation, it can easily be seen that a particular schema grows as the ratio of the average fitness of the schema to the average fitness of the population. Therefore, schemata with fitness values above average fitness value of the population will proliferate as time goes on, while schemata with below average fitness values will die off.

Although Equation 3-36 shows the increase of above average schemata in subsequent populations, it does not take into account the effects of crossover and mutation. Simply copying the best portions of each existing string into future populations succeeds in raising the average fitness of the future populations, but does not promote exploration of new areas of the solution space, one of the main goals on non-localized methods. This is the function of crossover and mutation, but the effect they have on the propagation of above average schema must also be analyzed.

If crossover between two strings occurs at random with probability p_c, then a lower bound on the probability of survival p_s of a particular schema with defining length $\delta(H)$ can be given by:

$$p_s \geq 1 - p_c \frac{\delta(H)}{l-1}$$ Equation 3-37

where l is the length of the entire string.

The best way to understand this probability is through example. Consider a string of length l=7 and two representative schemata, one with $\delta(H) = 5$ and one with $\delta(H) = 1$:

A = 0 1 1 1 0 0 0

H_1= * 1 * * * * 0

H_2= * * * 1 0 * *

Now suppose one-point crossover is employed to generate a crossover site anywhere from position 1 to 7. Clearly the likelihood that that site will fall within the defining length of H_1 is higher than the likelihood that it will fall between positions 4 and 5, the only possible way to break up H_2. Therefore, as demonstrated by Equation 3-37, the schemata with shorter defining lengths have higher probability of surviving crossover. Combining the effects of selection and crossover, it can be seen that the number of occurrences of schemata that have above-average fitness values and short defining lengths will increase in future generations.

$$m(H,t+1) \geq m(H,t)\frac{F(H)}{\overline{F}}\left[1-p_c\frac{\delta(H)}{l-1}\right] \qquad \text{Equation 3-38}$$

Finally, the effects of mutation on the survival of a schema must be determined. The probability that any position in a string will be changed has been defined to be p_m. Therefore, the probability of survival of any single position is $(1 - p_m)$. Since there are o(H) positions specified in a schema, the probability that the entire schema will survive is $(1-p_m)^{o(H)}$. Since p_m is typically << 1, this probability can be approximated by $1-o(H)p_m$. The overall probability that a given schema survives selection, crossover, and mutation is given by the following equation (where small cross product terms have been ignored).

$$m(H,t+1) \geq m(H,t)\frac{F(H)}{\overline{F}}\left[1-p_c\frac{\delta(H)}{l-1}-o(H)p_m\right] \qquad \text{Equation 3-39}$$

This result is known as the fundamental theorem of genetic algorithms[68]. It proves that above-average schemata of low order with short defining lengths increase exponentially in successive populations.

3-3-3-1-4 A Computer Implementation of the Genetic Algorithm—PGAPack

For the applications studied in this thesis, it was necessary to obtain an implementation of the genetic algorithm which could interface well with the orbit propagation software DSST, while also providing important features such as parallel calculations and the ability to provide user input for certain genetic algorithm parameters. The Argonne National Laboratory Parallel Genetic Algorithm Package or PGAPack met these requirements and was used successfully in a number of the applications. A few of the important features of this software are summarized here.

PGAPack is a general-purpose, data-structure-neutral, parallel genetic algorithm library under development at the Argonne National Laboratory. It is freely available, including all source code, installation instructions, users guide and a collection of examples from ftp.mcs.anl.gov or http://www.mcs.anl.gov/pgapack.html.

According to the *Users Guide to the PGAPack Parallel Genetic Algorithm Library*[69] the intention of PGAPack is to provide most capabilities desired in a genetic

---

[68] Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989, p. 33.
[69] Levine, D. Argonne National Laboratory, ANL 95/18, 1996, Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html.

algorithm package, in an integrated, seamless and portable manner. Some of the key features listed in this user's guide are:

- Ability to be called from Fortran or C
- Executable on uni-processors, multiprocessors, multi-computers, and workstation networks.
- Binary, integer, real, and character valued native data types.
- Object-oriented data structure neutral design.
- Parameterized population replacement.
- Multiple choices for selection, crossover, and mutation operators
- Easy integration of hill climbing heuristics.
- Easy-to-use interface for novice and application users.
- Multiple levels of access for expert users.
- Full extensibility to support custom operators and new data types.
- Extensive debugging facilities.
- Large set of example problems.

Basic usage of the program is simple and is explained in detail in Chapter 5 of the *Users Guide*[70]. A cursory overview of the requirements is contained here:

Any file that uses a PGAPack function (all of which begin with PGA, i.e. PGACreate), must include the PGAPack header file, (in Fortran, this file is pgapackf.h). After including this file, the first call to PGAPack is always to PGACreate. PGACreate allocates space for the context variable, ctx, and returns its address. After the PGACreate

---

[70] Levine, D. *Users Guide to the PGAPack Parallel Genetic Algorithm Library,* Argonne National Laboratory, ANL 95/18, 1996. Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html.

call, the user may optionally set any non-default values desired. These are then followed by a call to PGASetUp to initialize the default values for any variables not specified by the user.

Finally, the PGARun command is used to execute the genetic algorithm. It is at this step that the objective function is tied into PGAPack. For this thesis, it is also at this step that the link to DSST was made. The second argument of the call to PGARun is the name of a user-supplied evaluation function. This is the only function that must be written by the user, and it is this function that defines the problem one is trying to optimize. Each string is passed into this evaluation function and the objective function value is passed out of this function, back to PGAPack that then uses this objective function value to assign the string a fitness value. The rest of the genetic algorithm process is transparent to the user who doesn't wish to modify the default values. For the applications in this thesis in which PGAPack was used, the specific parameters will be laid out in the sections detailing each application.

3-3-4 Hybrid Methods[71]

Although genetic algorithms are powerful tools for finding optimal points while avoiding local minima, they often require excessive amounts of computation time. An additional method of optimization which, although not used in this thesis, has proven to be successful in many applications is a hybrid optimization scheme. This type of scheme combines the ability of non-localized methods to avoid local minima with the ability of localized methods to converge quickly to a solution. An example of this type of scheme

---

[71] Levine, D. *Users Guide to the PGAPack Parallel Genetic Algorithm Library,* Argonne National Laboratory, ANL 95/18, 1996. Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html.

would be a genetic-algorithm/Newton method combination in which the genetic algorithm is used to obtain a rough estimate of a solution and this solution then becomes the starting point of a local optimization scheme. It is thought that for some of the applications, to which genetic algorithms were applied in this thesis, hybrid methods might work better, as pointed out in the future works section.

[This Page Intentionally Left Blank]

Chapter 4 Optimal Constellation Design

The optimization techniques and theories presented in Chapter 3 can be applied to the fundamentals of astrodynamics described in Chapter 2 to solve a variety of astrodynamic optimization problems. This chapter focuses on one class of those problems: the optimal design of satellite constellations. Specifically, this chapter focuses on the optimization of two Ellipso™ sub-constellation designs. The first of these applications is an attempt to find the optimal mean elements for the baseline 113:14 repeat ground track Ellipso™ Borealis sub-constellation. The second application studies the Ellipso™ Gear Array Design (U.S. Patent pending).

4-1 Ellipso™ Borealis 113:14 Repeat Ground Track Problem

As discussed in Chapter 1, the baseline Ellipso™ constellation consists of three orbit planes: one circular, equatorial orbit known as Concordia™ and two elliptical, inclined orbits known as the Borealis™ sub-constellation. Due to the circular, equatorial nature of the Concordia™ sub-constellation, its orbits are quite stable. A slight variation in any of the orbital elements does not drastically change the overall performance of this sub-constellation. However, the same cannot be said for the Borealis™ sub-constellation. Due to the eccentric, inclined nature of the Borealis™ orbits, slight variations in the orbital elements can have significant effects on the overall performance of the constellation. Thus, the problem is to find the optimal initial orbital elements which, when propagated over time, cause the least decay in the performance of the constellation.

The performance of the Borealis™ sub-constellation can be quantified through three desired behaviors. These desired behaviors are outlined below:

1. First, in order to best serve the populated areas of the Northern Hemisphere, the two Borealis™ planes were designed with apogees in the Northern Hemisphere. This apogee location allows for longer dwell times over the Northern Hemisphere and therefore increases the coverage in this hemisphere. In order to maintain high levels of coverage over the populated areas, it is necessary that apogee remains in the Northern Hemisphere for the entire lifetime of the satellite.
2. The second desired behavior of the satellites in the Borealis™ plane is a sun-synchronous behavior. By sun-synchronous it is meant that the ascending node of the orbital plane rotates around the Earth at the same rate as the sun (or approximately 1° per day). By forcing the orbit to move at the same rate as the Sun, the satellites in the orbit plane appear to be in the same location, relative to the ground each day. Specifically, the Borealis™ planes have been designed such that one orbital plane has an ascending node at noon and the other has an ascending node at midnight. For performance to avoid degradation, it is necessary that these nodal locations be maintained for the lifetime of the satellite, as well.
3. The final desired behavior of the satellites in the Borealis™ orbits is a repeat ground track behavior. The satellites in the constellation will have a repeating ground track if they have exactly an integer number of revolutions per integer number of Earth revolutions. This behavior allows for constant viewing angles and has been designed

into a number of existing satellite systems including SeaSat, LandSat, and GeoSat.[72] For the Ellipso™ application, the specific desired integer values are 113 orbital revolutions of the Borealis™ satellites for every 14 revolutions of the Earth. The choice of the 113:14 configuration was based on trade studies undertaken by MCHI during the second half of 1997.[73]

By quantifying the 113:14 repeat ground track problem in terms of fixed perigee, sun-synchronous, repeat ground track orbital planes, constraints are immediately introduced on two of the orbital elements and the order of the design problem is simultaneously reduced. First, in order that apogee remains constant over the Northern Hemisphere, it is necessary that perigee remains constant in the Southern Hemisphere (or 260°). Therefore, there is no need to design for the argument of perigee. It must be fixed at 260°. Additionally, if the orbit is truly sun-synchronous, with a repeat ground track, then the ascending node locations of noon and midnight must change at a constant rate. Therefore, the location of the ascending node can be determined based upon the location of the Sun on any given day. With these two orbital elements fixed, the objective of this optimization study then became the determination of the initial orbital elements, which were not fixed (a, e, and i), that would cause the smallest deviation from the desired values of the predetermined elements (ω and Ω) as well as from the 113:14 repeat ground track behavior.

---

[72] Larson, Wiley J. and James R. Wertz. *Space Mission Analysis and Design,* Torrance, California: Microcosm, Inc., 1992, p. 154.
[73] Draim, J.E., P. Cefola, R. Proulx, and D. Larsen. *Designing the Ellipso™ Satellites into the Elliptical Orbit Environment,* Paper IAF-98-A.4.03, 49th International Astronautical Congress, Melbourne, Australia, 28 September to 2 October 1998, p. 3.

4-1-1 113:14 Repeat Ground Track Problem Formulation

The objective function for this optimization can be created through a simple combination of each of the desired conditions described above: a fixed ω, a constant rate of change on Ω, and a 113:14 repeat ground track ratio. A linear combination of these conditions results in an objective function of the following form:

$$J(a_0, e_0, i_o) = \alpha K_1 + \beta K_2 + \gamma K_3 \qquad \text{Equation 4-1}$$

where:

- α,β, and γ are scale factors chosen arbitrarily to provide the desired combination of the desired behaviors. For this case these factors were empirically set to the following: $\alpha = 1$, $\beta = 10000$, and $\gamma = 1$.
- K_1, K_2, K_3 are functions which represent the desired behaviors

The desired behaviors can each be described mathematically as follows:

Fixed argument of perigee. The fixed argument of perigee behavior can be obtained by summing the differences between the actual argument of perigee (ω) and the desired argument of perigee (ω_0) over all sample times (S). In order to maintain apogee in the Northern Hemisphere, the desired behavior is for all deviations from the actual value of 260° to approach zero.

$$K_1 = \sum_S |\omega - \omega_0| \qquad \textit{Equation 4-2}$$

Sun Synchronous. The sun-synchronous behavior can be verified by checking that the actual rate of change of the ascending node is always approximately equal to the Earth's secular rotation about the sun (0.9865°/day). Once again, this behavior is

sampled over the orbital lifetime and the desired behavior is for all deviations to approach zero.

$$K_2 = \sum_S \left| \dot{\Omega} - 0.9865°/day \right| \qquad \text{Equation 4-3}$$

113:14 Repeat Ground track. One way to ensure the repeat ground track behavior is met, is to compare the desired time between nodal crossings (which can be calculated using the node rate and the rotation rate of the Earth, ω_e as outlined in Sabol's thesis[74]) with the actual time between nodal crossings of a given satellite in the plane. The result is an equation for K_3 and again the desired behavior is that, at all sample times, the deviations approach zero.

$$K_3 = \sum_S \left| \frac{2\pi}{\dot{M} + \dot{\omega}} - \frac{14}{113} \frac{2\pi}{\omega_e - \dot{\Omega}} \right| \qquad \text{Equation 4-4}$$

4-1-2 113:14 Repeat Ground Track Localized Solution Process

After creation of the objective function, a first attempt at finding the optimal semi-major axis, eccentricity, and inclination was made using localized methods. As search space was small (only three solve-for variables), and the gradient information was

---

[74] Sabol, Christopher. *Application of Sun-Synchronous, Critically Inclined Orbits to Global Personal Communications Systems,* CSDL-T-938, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, January 1987.

not easily available, it was decided that Powell's method would work well for attempting to solve this problem.

4-1-2-1 Powell's Method Program Structure

Powell's method was implemented in a simple Fortran program as shown in Figure 4-1.

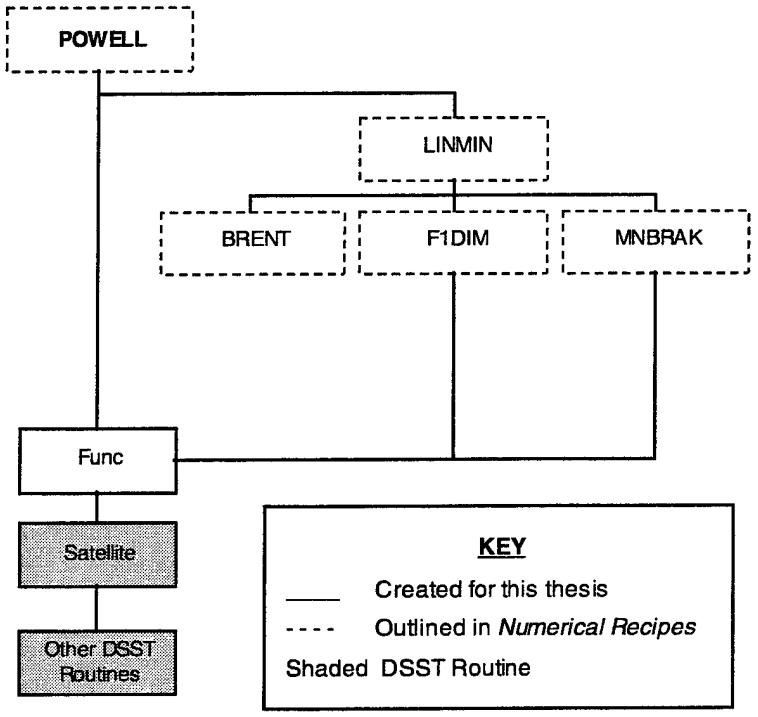


Figure 4-1 Program Structure for Powell's Method

In this structure, POWELL is the main program that performs the steps to the Powell algorithm outlined in section 3-3-2-4. The LINMIN and subsequent subroutines are simply used to perform the minimization in a given linear direction after POWELL has determined that direction. FUNC is a FORTRAN coding of the objective function for this optimization. Included in FUNC are calls to DSST, since orbit propagation is a

key component of the objective function. The details for all of the routines except for FUNC and those routines called by FUNC can be found in *Numerical Recipes in FORTRAN.*[75]

4-1-2-1-1 113:14 Repeat Ground Track Objective Function Code Structure

Since the majority of the routines created to perform this optimization are outlined in *Numerical Recipes in FORTRAN*[76], they will not be outlined here. These routines simply perform the necessary steps for Powell's method as described in Chapter 3. However, the FUNC routine is the exception. Rather than being a standardized routine for performing Powell's method, FUNC is a routine created especially for the implementation of the objective function into the solution of the 113:14 repeat ground track problem. More information regarding FUNC is presented here due to the specialized nature of its development.

Figure 4-2 contains an overview of the FUNC subroutine created for the solution of the 113:14 optimization problem. As can be seen from this figure, the FUNC routine is simply an implementation of the objective function detailed previously in section 4-1-1. Using DSST, the Borealis™ orbit is propagated for the desired length of time from certain specified initial values of a, e, and i. The variation of each of the three behaviors (fixed perigee, fixed node rate, and 113:14 repeat ground track) is checked and summed

---

[75] Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing—Second Editio,*. New York, New York: Cambridge University Press, 1992, p. 412.

[76] Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing—Second Edition,* New York, New York: Cambridge University Press, 1992, p. 412.

over time as the satellite is propagated. A weighted sum of these variations is then returned as the objective function value for the given a, e, and i combination.

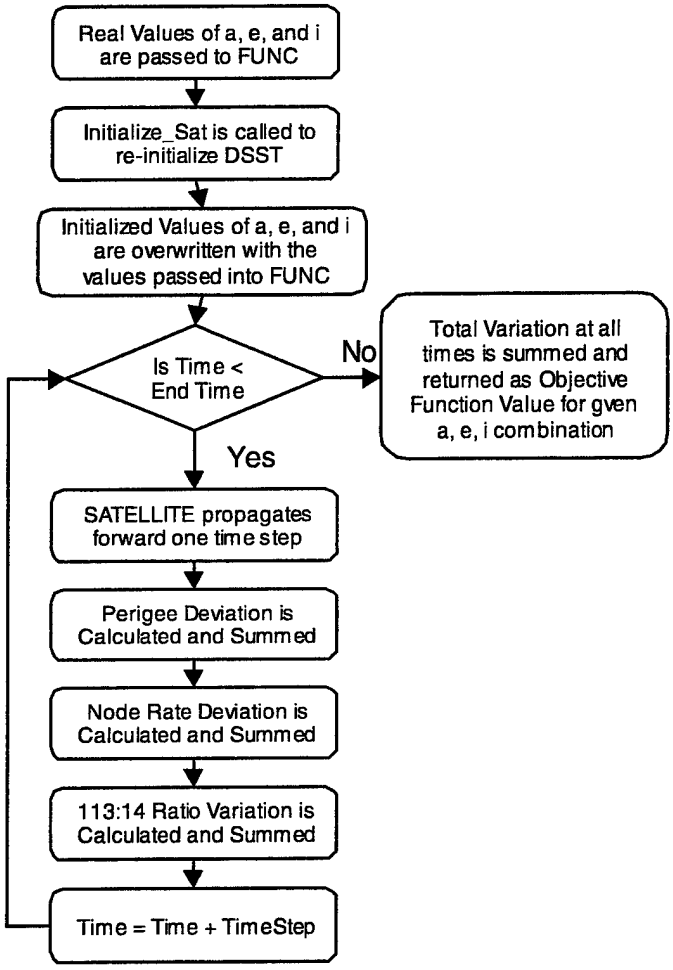


Figure 4-2 113:14 Repeat Ground Track FUNC Overview

4-1-2-1-2 113:14 Repeat Ground Track Propagation Input Parameters

In order to perform any type of calculations using the DSST software, it is necessary to first define certain parameters of the propagation. For example, one parameter that must be specified is whether or not to include the effect of third body point masses. The specific parameters to use are specified through an input deck. For this

optimization it was determined that only a 50 x 0 gravitational field and the Lunar/Solar Point mass perturbations should be included. The specific input deck used to convey this information to DSST for the 113:14 case can be found in Appendix A-1.

In addition to the perturbation parameters that must be specified, it was also necessary to specify certain features relating the satellite to be optimized. For this case one satellite from the Borealis™ node-at-noon orbit plane was chosen. To define this satellite, an argument of perigee of 260° and a right ascension of the ascending node value of 280° in the J2000 True of Date reference frame at an epoch time of 0 hr 0 min 00 sec on January 1, 1997 were specified. Additionally the satellite parameters of spacecraft mass and area were also specified as 1250 kg and 43.3 m^2, respectively.

4-1-2-2 Powell's Method Performance on 113:14 Repeat Ground Track Problem

Using the FORTRAN implementation of the Powell algorithm, along with the coded objective function and Borealis™ input deck described previously, attempts were made to find the optimal epoch elements. On each run of the Powell algorithm, convergence to a solution occurred rapidly. However, the result of that convergence was not always the same (a, e, and i) element set on successive runs of the algorithm. The majority of the time, the algorithm converged to the solution presented in Table 4-1. However, by varying the starting location, it was possible to cause the algorithm to converge to what has since been found to be an incorrect solution.

Figure 4-3 and Figure 4-4 show the reason for the inconsistency in convergence. These are surface plots of the eccentricity and inclination space plotted against the corresponding objective function values. To create these plots, the semi-major axis was fixed at the optimal value of 10496.8968 km, and both inclination and eccentricity were

allowed to vary over specified ranges. The result of this variation was a number of a, e, i combinations which could then be passed to FUNC to obtain objective function values for the specified variable combinations. As each objective function evaluation required a five-year propagation of the satellite orbit, MPI was used to reduce the total computation time, by dividing the many objective function evaluations across multiple processors. The resulting objective function values were then plotted against the two variable ranges. All possible combinations of these plots are presented in Appendix A-2. The two plots of most interest are presented here.

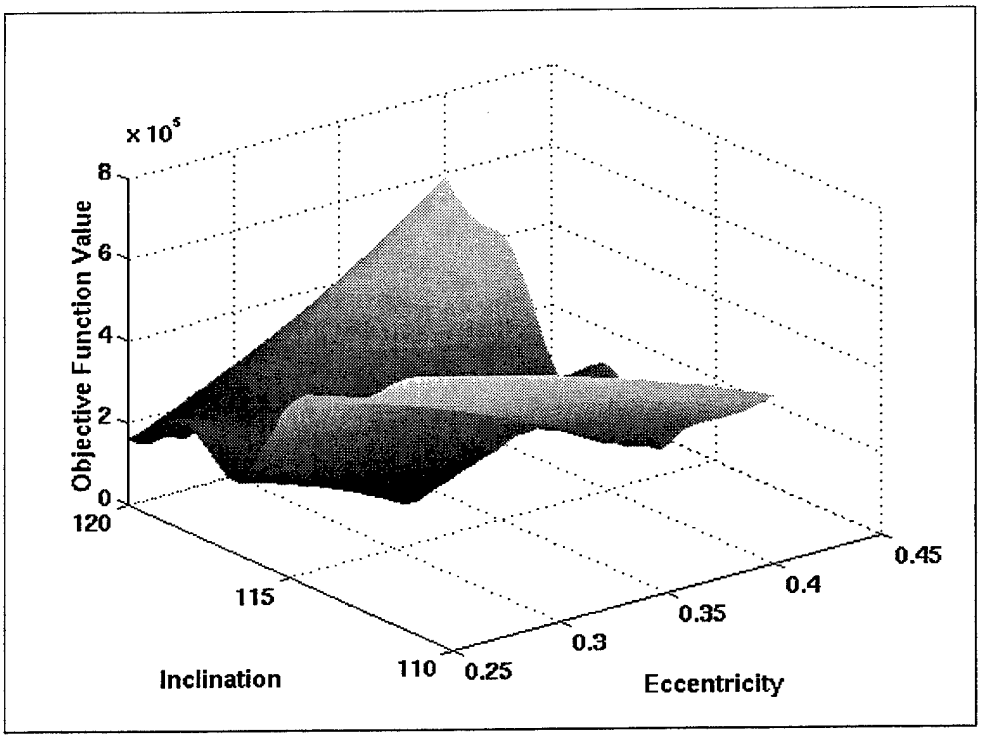


Figure 4-3 3-D Surface of 113:14 e/i Space (a = 10496.8968 km)

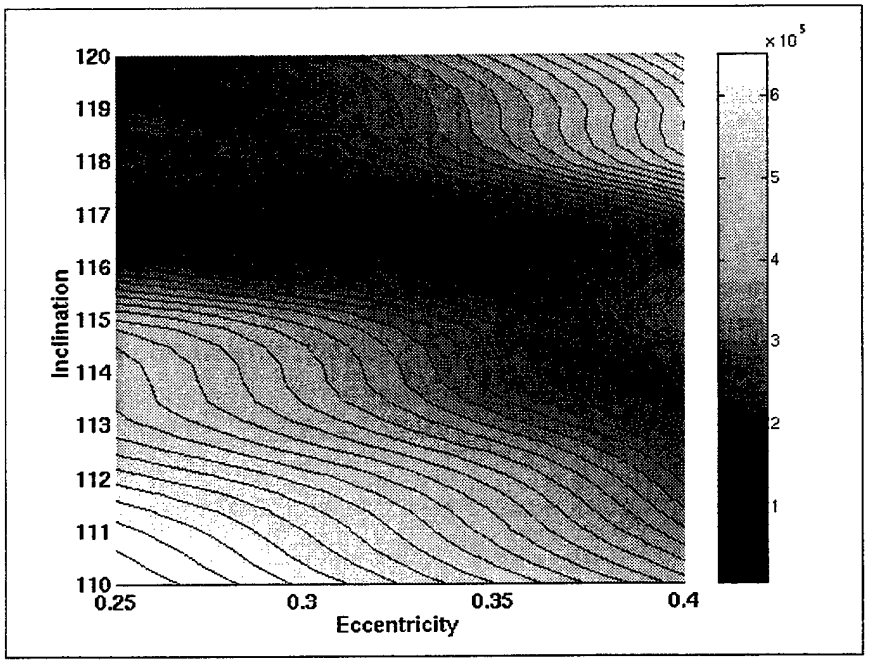


Figure 4-4 Contour Plot of 113:14 e/i Space (a = 10496.8968 km)

The interesting feature of the eccentricity and inclination space shown in Figure 4-3 and Figure 4-4 is the presence of multiple minima. The existence of one global minima is clear, but there are also at least two other points to which a localized method, such as the Powell algorithm could conceivably converge. In fact, after creating these plots, it was found that the non-optimal answers that were sometimes converged upon by Powell's method did in fact coincide with the local minima presented in these figures.

4-1-2-3 Results of Powell's Method for 113:14 Repeat Ground Track Optimization

The optimal results of a five-year optimization using the localized method was the a, e, and i combination shown in Table 4-1. When used as the epoch elements for the Borealis™ satellite, the elements presented in this table produced the smallest variation in each of the three desired behaviors over a five-year period. The details of the resulting behavior (see section 4-1-4) will be discussed after first discussing the optimization of the 113:14 case via a second optimization method: genetic algorithms.

Table 4-1 113:14 Results from Powell's Method Optimization

Orbital Element	Powell Determined Optimal Value
Semi-major axis (a)	10496.8968 km
Eccentricity (e)	0.32986
Inclination (i)	116.5782°

4-1-3 113:14 Repeat Ground Track Genetic Algorithm Solution Process

Although the localized Powell's method worked reasonably well in solving this design problem, attempts were also made to obtain the appropriate orbital elements using genetic algorithms. By using genetic algorithms to solve the same problem, it was hoped that the solution obtained through the local methods could be reproduced, and that by so doing, the optimality of this solution could be verified. Additionally, genetic algorithms were also chosen in the hopes of creating a more robust optimization algorithm that could avoid the local minima sometimes converged upon by the localized approach without requiring human intervention.

4-1-3-1 113:14 Repeat Ground Track Genetic Algorithm Code Structure

To solve this optimization problem using genetic algorithms, the genetic algorithm package known as PGAPack was used. Although PGAPack is a self-contained genetic algorithm software package, it was necessary that an interface between the DSST code and the various portions of the PGAPack code be created. This structure remained fairly constant throughout all applications in this thesis and therefore it is be presented in detail here. Future applications discussed in this thesis that also employed PGAPack will contain only descriptions of the changes necessary to the basic structure presented here.

In order to tie PGAPack into DSST, which would allow for propagation of the Borealis orbits necessary to calculate the desired objective function values, two main

pieces of code were required: a program shell which would execute PGAPack commands, and an objective function routine which would perform the propagations and return the objective function values to the genetic algorithm. PGA_SAT was created to be the program shell and the function FINDBEST was created as the objective function tie in to DSST. The structure of the entire program can be seen in Figure 4-5.

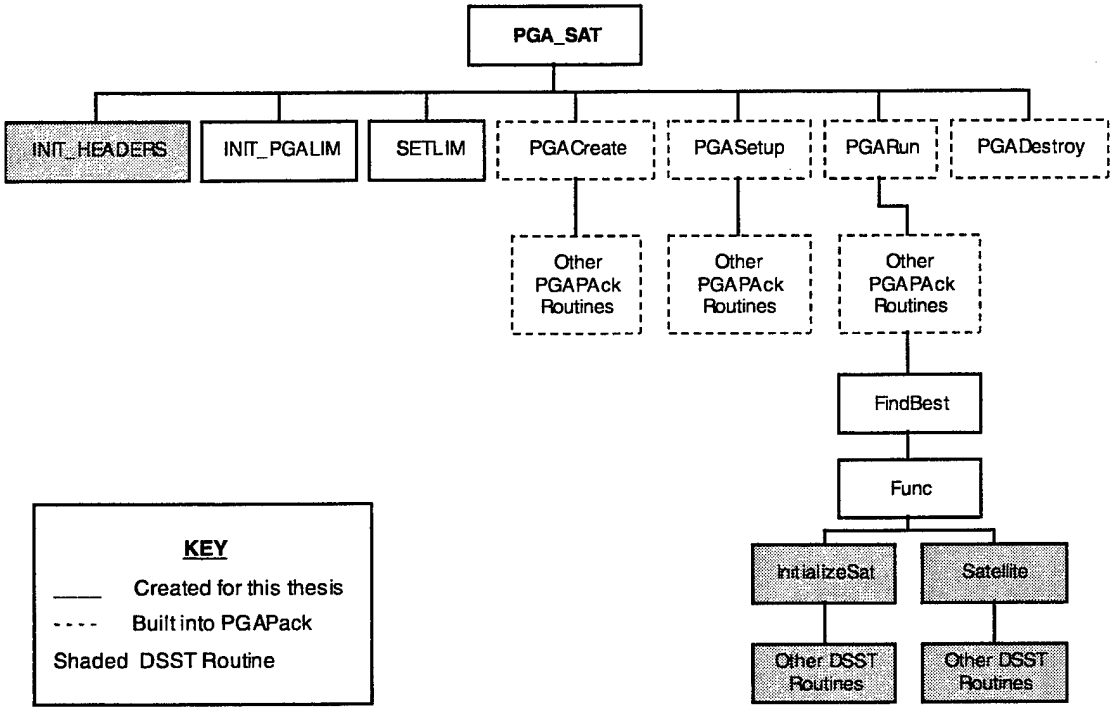


Figure 4-5 Software Overview for 113:14 Genetic Algorithm Solution Process

Although the flow through the software can be seen in Figure 4-5, the actual purpose of each routine is not readily evident. Therefore, a brief summary of each routine is provided below:

- PGA_SAT: As mentioned previously, PGA_SAT is the executable shell used to run the PGAPack program. It is first used to initialize the necessary values of a number of necessary variables. It then calls the PGAPack routines that perform the genetic algorithm optimization.

- INIT_HEADERS: INIT_HEADERS is a DSST routine used to initialize certain values necessary for the orbit propagator.
- INIT_PGALIM: The INIT_PGALIM routine is used to set a number of variables specific to the 113:14 optimization problem to predefined default values.
- SETLIM: The SETLIM routine is used to overwrite the default values of certain variables with user specified values.
- PGACreate: PGACreate is always the first PGAPack function called in a PGAPack program. It initializes the context variable, ctx that is necessary in calls to all other PGAPack routines. The parameters to PGACreate are the data-type to be used (Binary, Real, Character, or Integer), the string length, and the direction of optimization (maximization or minimization).
- PGASetUp: Following a call to PGACreate, a user can specify any number of PGAPack parameters through a series of commands as explained in the PGAPack Users Guide.[77] The call to PGASetUp initializes all parameters and function pointers not explicitly set by the user to the default values. The only required parameter is the ctx variable.
- PGARun: The PGARun call executes the genetic algorithm. It requires two arguments, the ctx variable, and the name of a user defined function that will be called to evaluate the strings (i.e. the objective function routine).
- FINDBEST: FINDBEST is the user-defined function that is used to evaluate each of the genetic algorithm strings. The parameters which are passed to it are the ctx

---

[77]Levine, D. *Users Guide to the PGAPack Parallel Genetic Algorithm Library,* Argonne National Laboratory, ANL 95/18, 1996. Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html.

variable, an index to the string to be evaluated (p), and an index to the population that contains that string (pop). The actual call to FINDBEST is accomplished internal to the PGAPack software.

- FUNC: In the software created for this application, FINDBEST is simply used to convert a string of any data-type to an array of real numbers which can then be evaluated against an objective function. The real number array is then passed to the next routine (FUNC), which actually performs the evaluation of the objective function. The FUNC used for the 113:14 optimization is the same routine that was used in the localized attempt. The only required parameter that must be passed to FUNC is the array of real numbers.
- INITIALIZE_SAT: Each objective function evaluation for these genetic algorithm satellite design problems involved a propagation of the orbit using DSST. In order to re-initialize the DSST software, each time FUNC was called upon to evaluate a string, INITIALIZE_SAT was created. It is simply a tool for initializing DSST given a predefined file name.
- SATELLITE: SATELLITE is the routine that actually performs the orbit propagation. It ties directly into DSST and returns the state of the specified satellite at any given time.

4-1-3-1-1 113:14 Repeat Ground Track Variable String Structure

As detailed in the section on genetic algorithms (section 3-3-3-1), genetic algorithms do not work directly with the variables of interest, but rather with strings that contain mappings of values for the variables to be solved for. For most genetic algorithm applications this mapping is a binary mapping, although PGAPack does allow users the

option of choosing between binary, integer, real, and character data-types. The choice of data-type is very dependent upon the type of problem being solved. For the solution of the 113:14 case, a binary mapping was found to work best.

As the only variables of interest for the 113:14 repeat ground track optimization were the orbital elements of a, e, and i, the creation of the string structure for this problem was quite simple. To achieve the greatest resolution of the search space allowable, each variable was simply coded into a 31-bit binary string (PGAPack has an upper limit of 31 bits). The result of this coding was a 93-bit string that PGAPack could then optimize. The structure of the resulting string is shown below in Figure 4-6:

[Binary Semi-major Axis] [Binary Eccentricity] [Binary Inclination]

Figure 4-6 String Structure for 113:14 Genetic Algorithm Optimization

4-1-3-1-2 113:14 Repeat Ground Track GA Objective Function Structure

Not only was it necessary to define the structure of the strings to be optimized, but it also was also necessary to properly define the objective function to be used in the evaluation of each string. An objective function for this problem was defined previously in Equation 4-1. In order for genetic algorithms to work properly, it is often necessary to modify the desired objective function to fit into a genetic algorithm framework. This problem, though, is quite simple and the objective function is acceptable in the form presented previously. In fact, the structure of the FUNC routine created previously for the localized Powell algorithm was such that it could be tied directly into the genetic algorithm application without any modification. The obvious result of this objective

function is that an a, e, and i combination which gives small variations from the desired behaviors will end up with small objective values and therefore high fitness values.

4-1-3-1-3 113:14 Repeat Ground Track Genetic Algorithm Parameters

As is the case with most attempts to solve problems with genetic algorithms, it is not enough simply to specify the desired string and objective function structure. Even with these definitions in place, the performance of a genetic algorithm is very dependent upon a number of other parameters. The mutation rate provides an excellent example of the need to choose the appropriate values for certain parameters. If the mutation rate is set too high, the population will maintain a high level of diversity, but will have difficulty converging to a solution. On the other hand, a mutation rate that is set too low will cause premature convergence to a solution that is more than likely not optimal.

Unfortunately, the best values for each of the genetic algorithm parameters change with each problem one attempts to solve. Therefore, a large portion of the time that must be invested into a genetic algorithm solution process must be invested into a process known as "tuning" the genetic algorithm. During the tuning process, one performs multiple runs of the genetic algorithm software and observes the behavior that changes in certain parameters have on the performance. Eventually, the parameters that give the best performance can be found.

The parameters that gave the best performance for solving the 113:14 problem are summarized in Table 4-2. They are also detailed briefly below:

- Stopping Rule: Since a genetic algorithm is an iterative process, some type of rule must be predefined as to when the algorithm should stop. PGAPack has three

different options for when to stop: If a maximum number of iterations has been met; If the best value of the objective function hasn't changed for a fixed number of iterations, or if all members of the population are too similar to one another. For this problem, setting a no-change-stopping rule of 100 iterations seemed to work well.

- Population Size: The population size parameter defines the number of strings to be maintained in each generation. The default PGAPack value is 100 which also seemed to work well for this optimization problem.
- Replaced Per Iteration Value: Two population replacement schemes are common in the literature. The first, the generational replacement genetic algorithm (GRGA) replaces the entire population each generation. The second, the steady-state genetic algorithm (SSGA) replaces only a few strings each generation and is a more recent development.[78] One advantage to the SSGA is that fewer function evaluations are required per iteration. As the function evaluations for this problem involved detailed orbit propagation, it proved advantageous to replace only a portion of the population each iteration as opposed to replacing all 100 strings. The number replaced each generation was 25.
- No Duplicates Flag: The no duplicates flag determines whether or not duplicate strings are allowed in a population. By setting it equal to true, the mutation operator is repeatedly applied to a duplicate string until the string no longer matches any other strings in the population. This requirement was found to be very advantageous as it

---

[78] Levine, D. *Users Guide to the PGAPack Parallel Genetic Algorithm Library,* Argonne National Laboratory, ANL 95/18, 1996. Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html, p. 19.

allowed the populations to maintain a high level of diversity, thereby avoiding premature convergence to local minima.

- Mutation and Crossover Flag: The default for PGAPack is to apply mutation only to strings that did not undergo crossover. However, as with the no duplicates flag, it was desirable to maintain high levels of diversity in solving the 113:14 repeat ground track problem. Setting the mutation and crossover flag to true allowed strings to undergo both mutation and crossover, thereby increasing the overall population diversity.
- Mutation Rate: The default PGAPack mutation probability is 1/L where L is the length of a string. This rule for defining the mutation rate was found to work well, and therefore the mutation rate was not modified from its default value.
- Crossover Type: As mentioned previously, there are a number of available crossover operators. The most common of these is the two-point crossover. PGAPack also allows for one-point or uniform crossover, but since two-point crossover was successful, neither of these options was selected.
- Crossover Probability: Like many of the other PGAPack parameters, the default probability of crossover of 0.85 also needed no modification.
- Selection Type: As described in section 3-3-3-1-2, the selection operator allocates reproductive trials to strings on the basis of their assigned fitness values. However, like crossover, a number of different ways of applying this operator are possible. PGAPack supports four selection schemes with tournament selection as the default.[79]

---

[79] Levine, D. *Users Guide to the PGAPack Parallel Genetic Algorithm Library,* Argonne National Laboratory, ANL 95/18, 1996. Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html, p. 22.

Table 4-2 113:14 Case Genetic Algorithm Parameters

Parameter	Type/Value
Stopping Rule	No Change
No Change Value	100
Population Size	100
Replaced Per Iteration	25
No Duplicates Flag	True
Mutation and Crossover Flag	True
Mutation Rate	1/String Length
Crossover Type	Two-Point
Crossover Probability	0.85
Selection Type	Tournament

4-1-3-1-4 113:14 Repeat Ground Track Genetic Algorithm Propagation Parameters

An input deck identical to the input deck used to implement the Powell algorithm was used (See 4-1-2-1-2) to perform the genetic algorithm optimization. Applying both methods to identical input decks allowed for a direct comparison of the results.

4-1-3-2 Performance of the GA on 113:14 Repeat Ground Track Optimization

Using the parameters and structures defined above, the genetic algorithm was able to successfully solve the 113:14 orbit design problem in a reasonable number of iterations. The time required to reach a converged state was longer than time required by the Powell method, but unlike the Powell algorithm, the genetic algorithm was able to avoid convergence to incorrect solutions. Both the best and the average objective function values as a function of iteration number are presented in the following figures. These figures clearly show that both the average and best objective function values are continually evolving to better and better values as predicted by the fundamental theorem of genetic algorithms. The fact that the best value remains constant for the last 150-200 iterations indicates that the number of iterations defined as the stopping rule is too large.

However, it was found that it was better to run for too many iterations, than to stop before actual convergence had been reached.

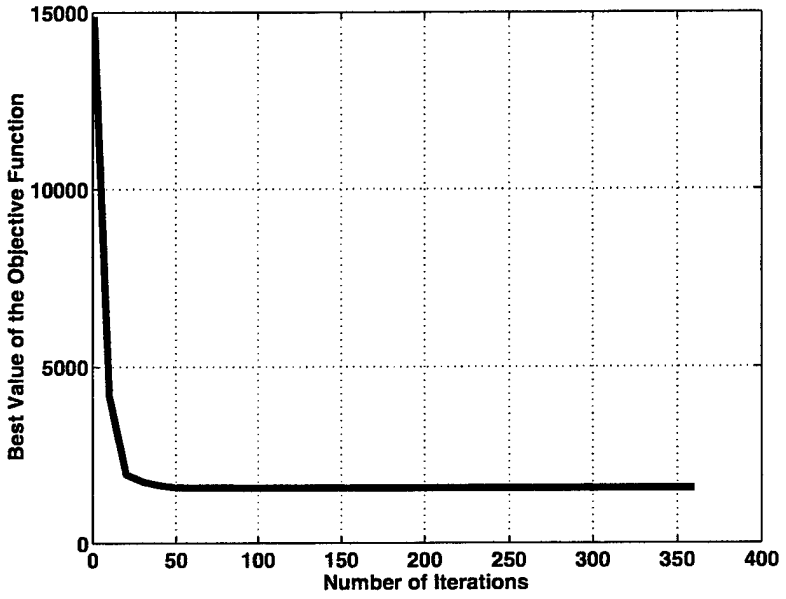


Figure 4-7 Best 113:14 Objective Function Value vs. Iteration Number

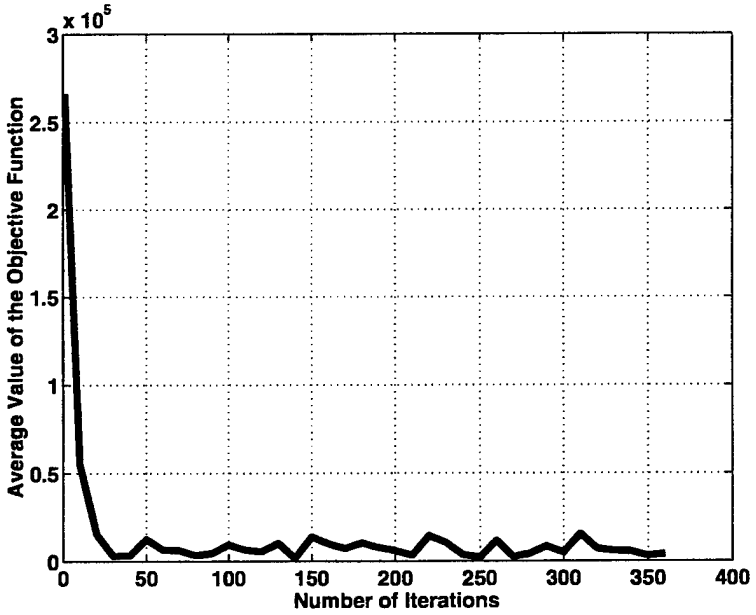


Figure 4-8 Average 113:14 Objective Function Value vs. Iteration Number

4-1-3-3 Genetic Algorithm Results for 113:14 Repeat Ground Track Optimization

As was the case with Powell's method, the result of the genetic algorithm optimization was a three-variable optimized set of the orbital elements (a, e, and i) that gave the minimum decay from the initially specified values of argument of perigee, node rate, and repeat ground track. The optimal elements from Powell's method have already been presented in Table 4-1. Table 4-3 shows the results from the genetic algorithm optimization along with the difference between these results and the results from the Powell optimization. Notice that the answers generated by both techniques are nearly identical.

Table 4-3 GA Derived Optimal Elements for the Borealis Node at Noon 113:14 Case

Element	Value	Difference from Powell
Semi-major Axis (a)	10496.8969 km	0.0001 km
Eccentricity (e)	0.32985	0.00001
Inclination (I)	116.5782°	0.0000°

4-1-4 Performance of Borealis™ With Optimally Designed Elements

After establishing (both through localized and genetic algorithm approaches) the optimal initial elements for the Borealis™ node-at-noon orbit plane, it was necessary to analyze the behavior of the orbit with those elements specified. To understand the behavior, DSST was again used to propagate the orbit. The result of that propagation was a series of plots that show the design accuracy of the orbit.

Of most interest among the elements are the accuracy of the argument of perigee and the ascending node. One of the three desired behaviors was a constant argument of perigee of 260°. Although, due to the zonal and third body perturbations, it is impossible

to meet this objective exactly, Figure 4-9 shows that the set of optimal elements does provide a trajectory whose mean is maintained near zero.

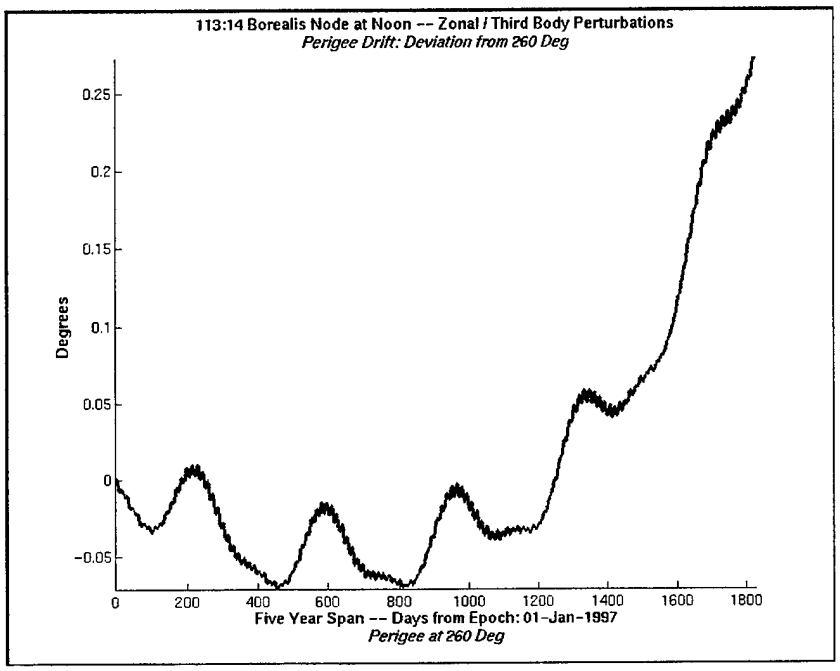


Figure 4-9 113:14 Repeat Ground Track Argument of Perigee Design Accuracy

The other desired behavior that can easily be plotted is the behavior of the longitude of the ascending node. As mentioned previously, the desire for this element was for it to have a constant rate that would allow the orbit to be sun-synchronous. Since the desired rate is a constant, the desired node can be easily computed as a function of time. The difference between the propagated ascending node and the desired ascending node is shown in Figure 4-10. Once again it is impossible to maintain the desired orbit exactly, but the optimal elements found are successful in zeroing out the mean of the variation.

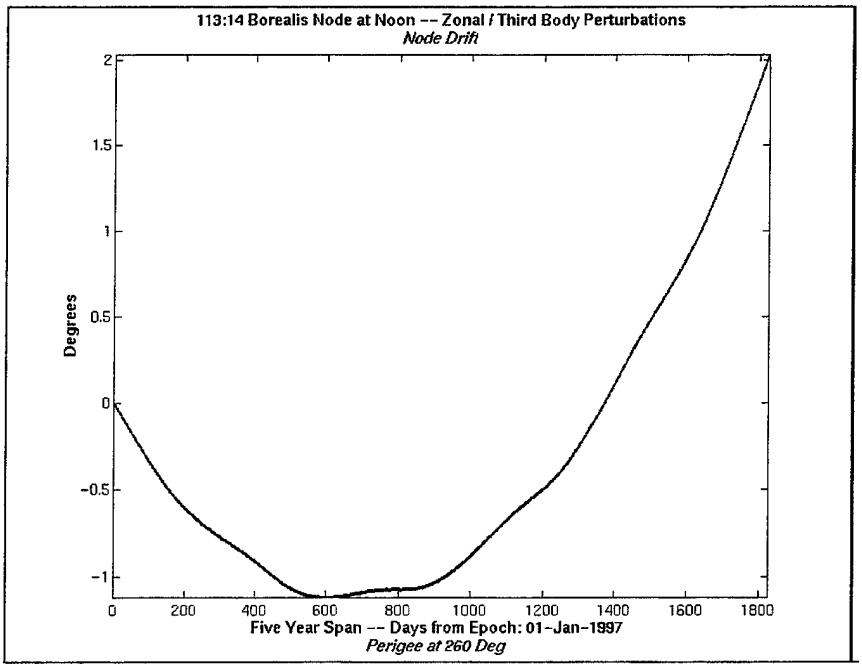


Figure 4-10 113:14 Right Ascension of the Ascending Node Design Accuracy

The behavior of the other elements is not nearly of as much interest as there were not any predefined desires for their behavior. Plots similar to those for the argument of perigee and the node can, however, be found for all the elements in Appendix A-4-1. Also, the maximum and minimum variations of these elements over the five-year life span are summarized in Table 4-4.

Table 4-4 Borealis Node at Noon 113:14 Design Accuracy over 5 Year Life Span

Oscillation/Variation	Maximum	Minimum
Semi-major axis (km)	0	0
Eccentricity	6.0 e-03	-1.7e-04
Inclination (deg)	1.5 e-02	-1.4 e-02
Argument of Perigee (deg)	2.7 e-01	-2.8 e-02
Ascending Node (deg)	2.0e+00	-1.1e+00

Although the optimally designed elements seem to perform well when propagated under zonal and third-body perturbations, there is an additional requirement to check their behavior when subjected to real world perturbations. Some additional decay is expected when the orbits are subjected to full perturbation models, but it must be verified that this additional decay is small enough that, through station keeping, the spacecraft could be controlled to desired levels of decay. In order to analyze the decay, it was necessary to propagate the satellite with more than just zonal and third body point mass effects included. The effects of these "real-world" perturbations were created using atmospheric drag, solar radiation pressure, tesseral harmonics, third-body, and solid Earth tides. A Jacchia-Roberts atmospheric density model was used in the determination of the drag effects.

The element history plots resulting from these full-perturbation propagations are included in appendix A-4-2. The argument of perigee and ascending node plots are included here in Figure 4-11 and Figure 4-12 for comparison to the results presented previously. It can readily be seen from the element decay figures that the behavior of the orbit under full perturbation models is not as stable as the behavior under only zonal and third-body effects. The decrease in stability is simply because the additional perturbations have effects on the orbit which were not taken into account during the optimization. However, as the decay under full perturbations appears small enough to be controllable through station keeping (see Chapter 5), the design can be considered adequate. Table 4-5 summarizes the decay under the effect of these perturbations.

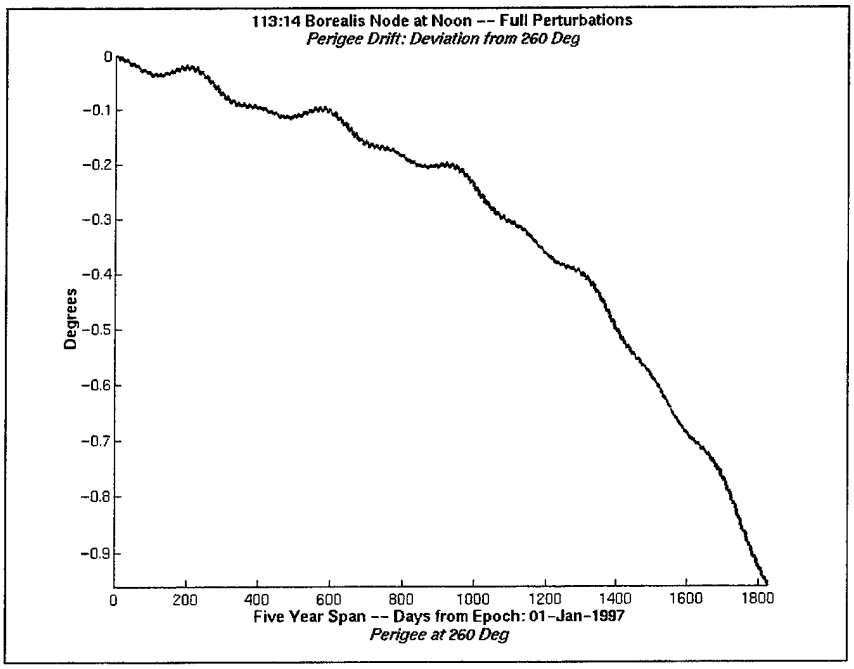


Figure 4-11 113:14 Argument of Perigee Decay Under Full Perturbations

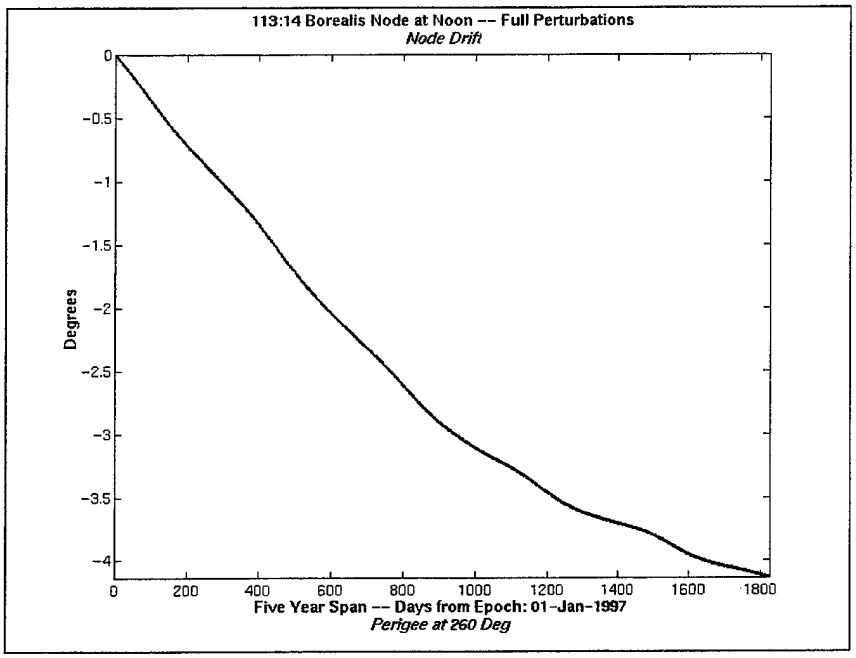


Figure 4-12 113:14 RAAN Decay Under Full Perturbations

Table 4-5 Borealis™ Node-at-Noon 113:14 Design Decay over 5 Year Life Span under Full Perturbations

Oscillation/Variation	Maximum	Minimum
Semi-major axis (km)	7.7e-01	-1.4e+01
Eccentricity	6.2e-04	-1.7e-03
Inclination (deg)	9.0e-03	-2.9e-02
Argument of Perigee (deg)	1.2e-03	-9.6e-01
Ascending Node (deg)	0	-4.1e+00

4-2 Ellipso™ Gear Array Design Optimization

Following successful application of genetic algorithms to the 113:14 repeat ground track design problem, attempts were made to solve a similar problem, that of the gear array, using similar techniques. The following sections contain a discussion of the gear array problem, the techniques used to solve the problem, and the resulting orbital behavior observed.

4-2-1 Gear Array Description.

Draim[80] and Turner[81] have described the use of elliptical orbits in the equatorial plane, whose apogees always remain on the sunlit side of the Earth. These types of orbits appear well suited to the field of satellite communications since they provide extra Earth coverage, as well as extra redundancy due to the "bunching" of the satellites near apogee.

---

[80] Draim, J. E. *Optimization of the ELLIPSO and ELLIPSO 2g Personal Communications Systems,* International Workshop on Mission Design and Implementation of Satellite Systems, Toulouse, France, 17-19 Novmeber 1997.

[81] Turner, A. E. *New non-Geosynchronous Orbits for Communications Satellites to Off-Load Daily Peaks in Geostationary Traffic*, AAS Paper 87-547, AAS/AIAA Astrodynamics Specialists Conference, Kallispell, Montana, 10-13 August 1987.

The concept of the "Gear Array" (US Patent Pending) is under development at Ellipso, Inc.[82] This gear array concept is a two-orbit hybrid elliptical/circular constellation consisting of two sub-constellations as shown in Figure 4-13:

- An elliptical sub-constellation of Apogee Pointing to the Sun (APTS) orbits containing n satellites.
- A circular sub-constellation whose motion is commensurable with that of the APTS orbits containing m satellites.

The apogee of the elliptical orbit approximately (or exactly, in some cases) matches the altitude of the circular orbit. On the daylight side, they are phased with an approximately equal spread between the circular satellites and the elliptical satellites.

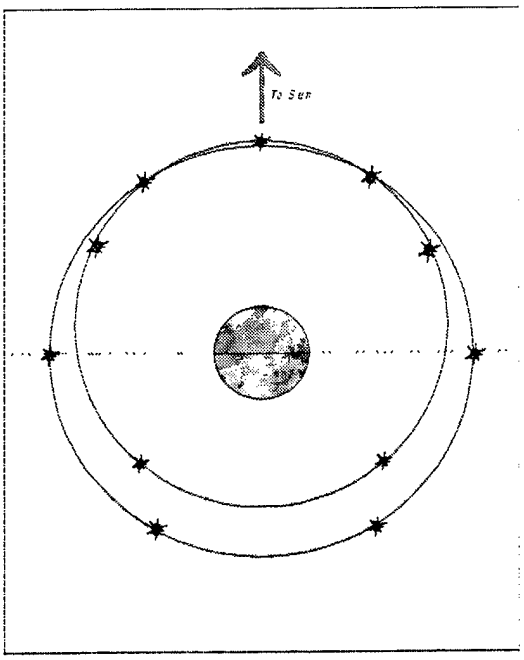


Figure 4-13 5:6 Gear Array Viewed from North Pole

---

[82] Proulx, Ronald J, James E. Smith, John E. Draim, and Paul J. Cefola. *Ellipso™ Gear Array: Coordinated Elliptical/Circular Constellations*, AIAA Paper 98-4383, AIAA/AAS Astrodynamics Specialist Conference, Boston, Massachusetts, 10-12 August 1998.

The choice of the term "gear array" appeared naturally, as the satellites resemble the teeth in a mechanical epicyclical gear train. The satellites appearing in the elliptical component of the gear array have shorter periods than those appearing in the circular component of the gear array. The ratio of the number of elliptical satellites to circular satellites is approximately the same as the ratio of their periods, thus as they rotate, the orbits periodically repeat their alignment in a manner similar to the teeth of a gear.

4-2-2 Gear Array Problem Formulation

Like the optimization problem for the 113:14 repeat ground track case, the optimization problem for the design of the gear array is to find the optimal initial orbital elements that give the desired behavior over some specified length of time. The objective function, from which the optimal elements are to be determined, is, once again, a simple linear combination of the desired behaviors. For the gear array there are two basic desirable behaviors: APTS and commensurability. These two behaviors are described by three mathematical expressions (K_1, K_2, and K_3) which can be combined to form the desired objective function (J).

$$J(a_e, a_c, e) = \alpha K_1 + \beta K_2 + \gamma K_3 \qquad \text{Equation 4-5}$$

where:

α, β, and γ are scale factors chosen to normalize the individual constraints. For this application α was emipirically set to 1000, β to 10, and γ to 1000.

In addition to defining the objective function, the desired behaviors also force all of the initial orbital elements except for those which are the arguments of the objective function J: the eccentric orbit semi-major axis (a_e), the circular orbit semi-major axis (a_c), and the eccentric orbit eccentricity (e). Since both orbits are in the equatorial plane, their

inclinations are defined to be zero and their ascending nodes are undefined. Additionally, for circular orbits, eccentricity is defined to be zero and the argument of perigee is undefined. The only remaining variable, the initial argument of perigee for the eccentric orbit is forced by virtue of the fact that apogee must always point towards the Sun. The initial Sun vector can be computed and the initial argument of perigee can then be set to this value.

4-2-2-1 APTS behavior

Having defined the solve-for variables as well as the form of the objective function, it is now necessary to mathematically define each of the desired behaviors which make up that objective function. The simplest of the behaviors to define mathematically is the APTS behavior. This is similar to the sun-synchronous behavior desired of the 113:14 orbits. However, since the elliptical orbit in the gear array is in the equatorial plane (i = 0°), it is not enough for the node (Ω) to move at the same rate as the Sun. Instead, a combination of the node and the argument of perigee (ω) must move at this rate. This desired combination is expressed below:

$$\dot{\omega} + \dot{\Omega} = .9865^{\circ} / day \qquad \text{Equation 4-6}$$

For equatorial orbits, the right ascension of the ascending node, Ω, is not well defined, and the APTS constraint must be recast on the line of apsides motion in terms of equinoctial elements. In this formulation, the APTS constraint takes on the following form:

$$K_1 = \sum \left| \frac{k\dot{h} - h\dot{k}}{h^2 + k^2} - .9865^{\circ} / day \right| \qquad \text{Equation 4-7}$$

Figure 4-14 shows values of the APTS constraint, K_1, as a function of the eccentric orbit semi-major axis and eccentricity. The reason for the choice of these two variables will be discussed in detail later in section 4-2-3-1. However, if a desired offset between the circular semi-major axis and the eccentric orbit apogee height, Δ, is defined ahead of time, the gear problem can be reduced to a two-variable optimization problem and plots of the behavior of each constraint can be generated. All of the plots in this section were created with a Δ of 0 km (i.e. the eccentric and circular orbits are tangent at apogee) and an n to m ratio of 4:5 (i.e. 4 satellites in the APTS array and 5 satellites in the circular array).

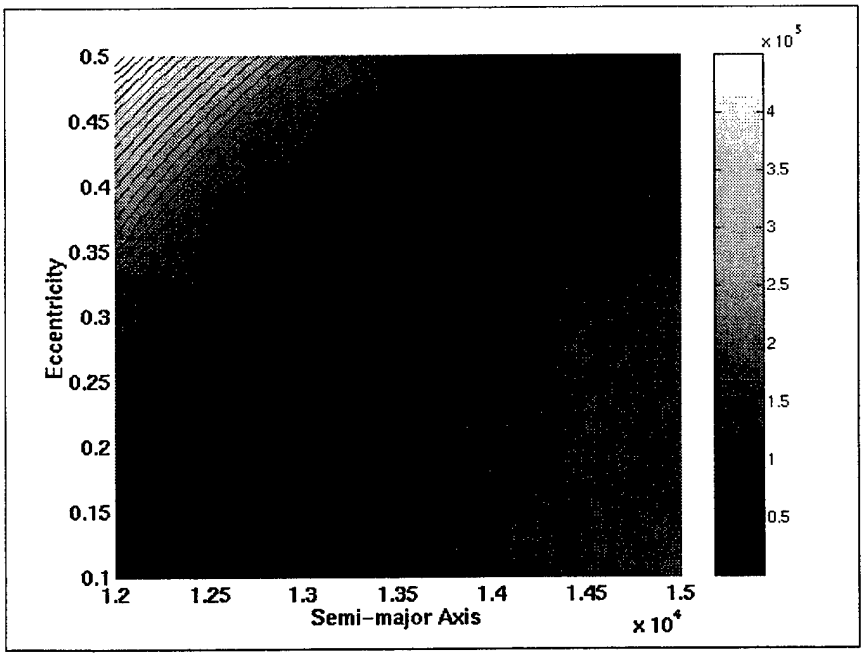


Figure 4-14 APTS Constraint Behavior Contour Plot

With sufficient graphical resolution, Figure 4-14 shows exactly what is expected. For a given eccentricity, there is a corresponding semi-major axis such that apogee is

always offset from the Sun at a fixed angle. Therefore, for any eccentricity, a corresponding semi-major axis produces an APTS orbit.

4-2-2-2 Commensurability Behavior

In the previous section, the desired APTS behavior has been framed mathematically and plotted. The result of that discussion, however, is that for any eccentricity, there is a corresponding semi-major axis that will minimize K_1. Therefore, with only a constraint on the APTS behavior, the problem is not sufficiently constrained to be solvable. In order to further define the problem, it is necessary to also include the effect of the desired commensurability relationship.

The commensurability relationship between the APTS and Circular components of the gear array is actually both a geometric and a rate control. That is, the gear array should have a stroboscopic character, its structure repeating cyclically; and the periods of the APTS and circular arrays should also be approximately commensurable. Each of these behaviors can be expressed mathematically as discussed below:

4-2-2-2-1 Geometric or Stroboscopic Constraint

To define the desired geometric behavior, let $\hat{u}_e(t)$ be the unit vector pointing at a satellite in the APTS sub-constellation and let $\hat{u}_c(t)$ be the unit vector pointing at a satellite in the circular sub-constellation of the GEAR array at time t. Also define P_e to be the anomalistic period of the APTS array and P_c to be the anomalistic period of the circular array. The stroboscopic constraint then requires that the dot product between the two vectors be periodic every m periods of the elliptical array as follows:

$$\hat{u}_e(t+mP_e)\cdot\hat{u}_c(t+mP_e)=Q(t) \qquad \text{Equation 4-8}$$

This constraint geometrically connects the motion of the circular array to the motion of the APTS array. Putting the APTS satellite at apogee at $t=0$, and setting

$$Q_0=Q(0)=\cos(\pi/m) \qquad \text{Equation 4-9}$$

the initial position of the satellite in the circular orbit is seen to be offset from the apogee direction of the APTS orbit at epoch by π/m radians. In order to sample the deviation from this constraint at many times during the constellation lifetime, this stroboscopic constraint can be recast as:

$$K_2=\sum_s\left|\hat{u}_e(smP_e)\cdot\hat{u}_c(smP_e)-Q_0\right| \qquad \text{Equation 4-10}$$

The surface formed by this constraint is much more complex than the surface formed by the APTS constraint as shown in Figure 4-14. Rather than having a one to one correspondence between a and e, this stroboscopic constraint appears to be largely independent of a and has multiple minimizing values of e as seen in Figure 4-15.

An edge-on view of the behavior along the semi-major axis is depicted in Figure 4-16. Note that the value of the objective function depends only on the choice of eccentricity. In the range selected, there are two minimizing values of eccentricity: 0.16 and 0.41. Also note that although 0.16 appears to be the best choice of eccentricity, both choices actually cause objective values of approximately zero. The differences present in

the figure are due to the refinement of the a/e grid over which the objective function was plotted.

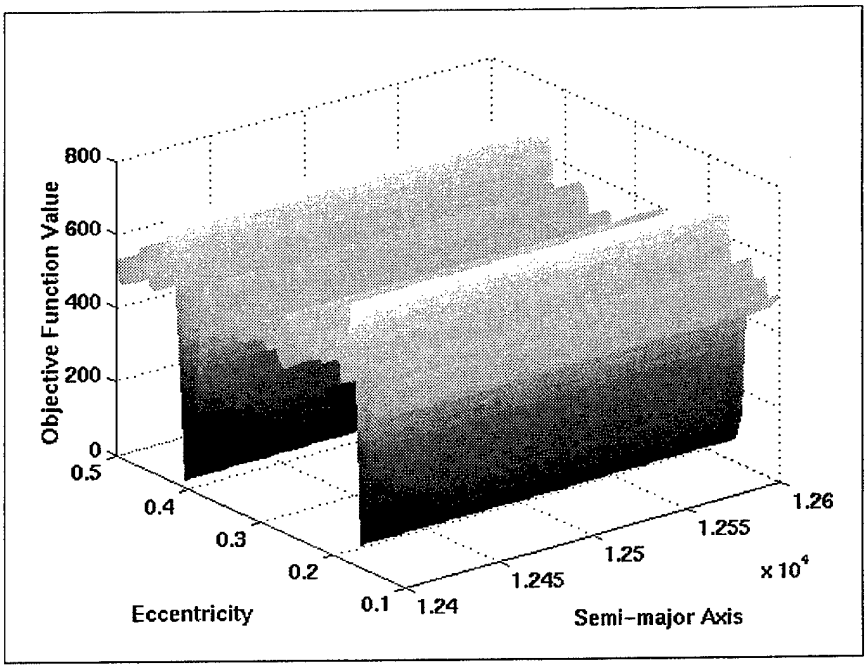


Figure 4-15 Stroboscopic Constraint Behavior

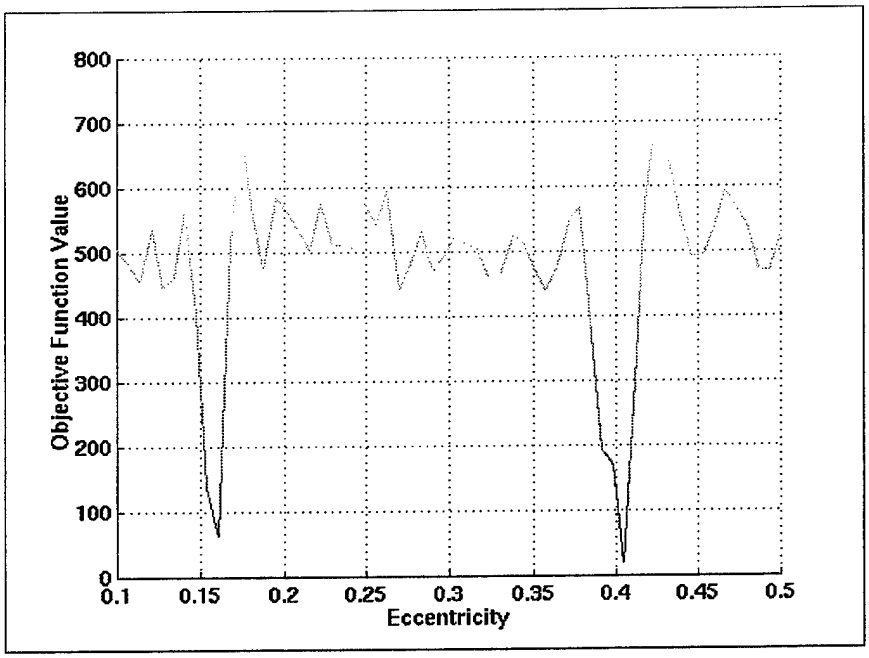


Figure 4-16 Stroboscopic Constraint Behavior: Edge-on View

4-2-2-2-2 Period Commensurability or Ratio Constraint

As shown in Figure 4-16, the stroboscopic constraint alone cannot force a unique eccentricity. Instead, several eccentricities provide a stroboscopic solution for the gear array. In order to further define the problem, a second commensurability constraint, was defined. Rather than relying on the desired geometry, this constraint instead frames the commensurability relationship in terms of the periods of the two orbits. This constraint requires that the anomalistic period of the APTS array and the anomalistic period of the circular array be approximately commensurable to the ratio of the number of satellies in each array: n/m. Sampling through the period of performance, we obtain the following constraint:

$$K_3 = \sum \left| \frac{P_e}{P_c} - \frac{n}{m} \right| \qquad \text{Equation 4-11}$$

Proceeding in a manner similar to that followed for the discussion of the other two constraints, the ratio constraint can be plotted. Like the other commensurability condition, the plot shows that this periodic constraint is also quite dependent on eccentricity. The variation of the constraint with respect to eccentricity can be seen in Figure 4-17.

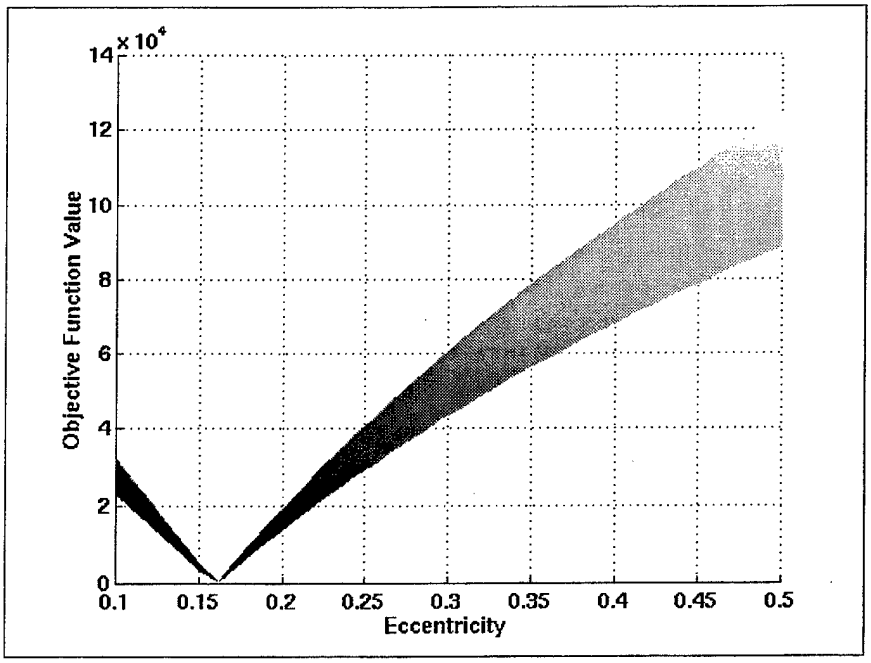


Figure 4-17 Ratio Constraint Behavior

Over the wide range of eccentricity values analyzed in Figure 4-16 and Figure 4-17, it appears that the minimum values of the stroboscopic and ratio constraints are aligned near an eccentricity of 0.16. However, there is actually a slight shift, due to changes in the period of both orbits caused by non-two body effects. With an appropriately complex level of analysis, the ratio constraint could be constructed such that the minimum occurred at a point corresponding exactly to one of the minima of the stroboscopic constraint. An advantage to the approach used here, however, is that this complexity is not necessary. The purpose of the ratio constraint is only to locate the general region over which to search the stroboscopic space. The stroboscopic constraint can then be used to locate the exact value of eccentricity that will cause the desired repetition between orbits.

4-2-3 Gear Array Solution Process

This section discusses the various steps that were taken to find a solution to the objective function outlined above. Initial attempts were made using the localized Powell's method, but as with the 113:14 case, Powell's method often converged to incorrect solutions. Therefore, Powell's method was abandoned, and the gear array design problem was instead solved using the parallel genetic algorithm approach. The steps taken in this approach are presented in this section.

4-2-3-1 Gear Array Problem Parameterization

Like many optimization problems, solutions to the gear array problem may be parameterized in several different ways. In attempting to find a solution three of these parameterization methods were considered.

- Fix a desired offset, Δ, between the apogee heights, and construct the APTS array which will meet the desired constraints. For example two tangent orbits at apogee could be achieved with a Δ of 0 km. By using the relationship $a_c = a_e(1+e)-\Delta$, where a_c is the semi-major axis of the satellite in the circular array, and a_e is the semi-major axis of the satellite in the APTS array, one of the solve for variables could be eliminated.
- Fix the semi-major axis of the circular array, and find the semi-major axis and eccentricity of the APTS array which meets the constraints. This approach also

reduces the problem to a two-variable optimization problem, but it assumes that a desired semi-major axis is known.

- Fix the desired apogee height of the elliptical array, and find the semi-major axis of both the ellipitcal and circular arrays which minimize the objective function. Again the problem becomes a two-variable optimization, but like the previous parameterization, it assumes that something about the design is pre-defined.

These three parameterization methods have been labeled the offset method, the fixed circular semi-major axis method, and the fixed apogee height method, respectively. All three are viable options for solution of the gear array problem, but a study of each method found that a careful choice of the parameterizatoin method led to an easiser solution path.

The best parameterization was found to be the offset method. Not only did this method allow for the most flexibility in the three design parameters (a_e, a_c, and e), but the solution space was also determined to be the easiest to navigate. The following three figures were created by formulating the problem in each of the three parameterization methods described above and allowing the free variables to vary over specified ranges. The result of these variations was an array of objective function values corresponding to a variable pair. Since each of the parameterization methods reduced the problem to a two-variable problem, the surfaces shown below are the actual surfaces over which optimization for the 4:5 gear array with 0 km offset is desired.

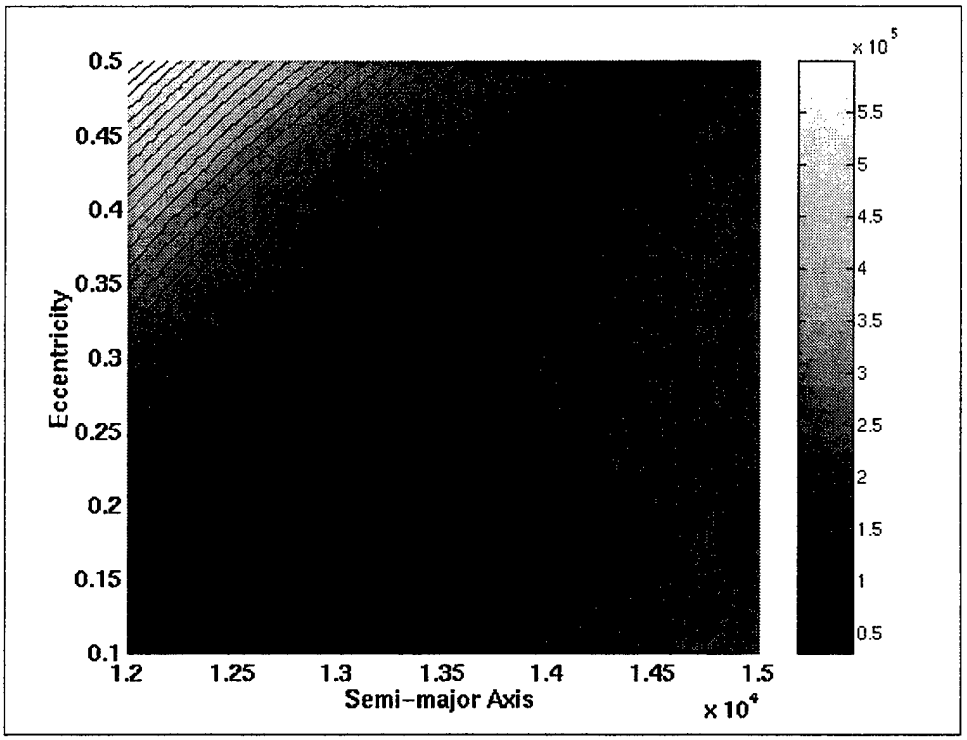


Figure 4-18 Offset Method Contour Plot

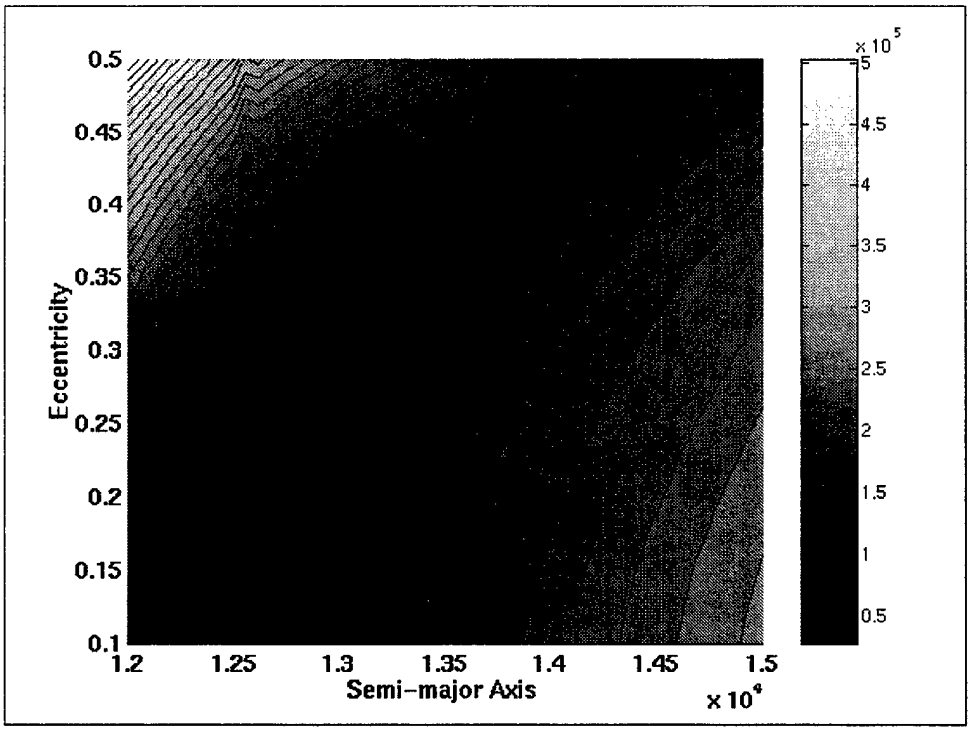


Figure 4-19 Fixed Circular Semi-Major Axis Method Contour Plot

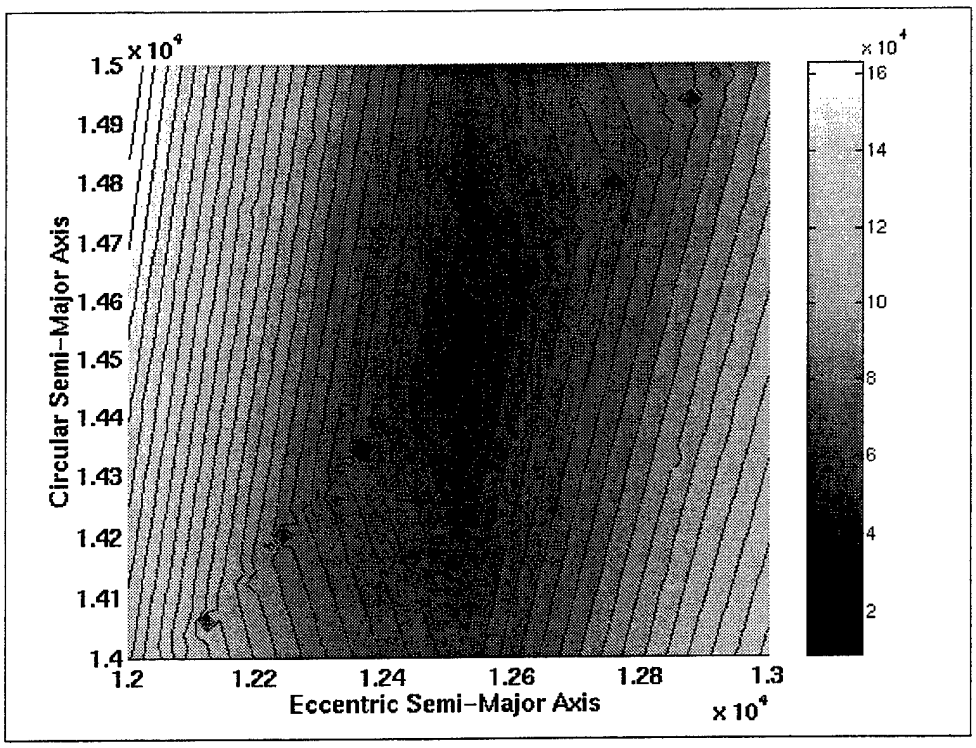


Figure 4-20 Fixed Apogee Height Method Contour Plot

A careful analysis of the three surfaces shows why the offset method is the parameterization method of choice. Although both the offset and fixed circular semi-major axis methods create smooth surfaces, the surface created by the offset method has a much clearer minimum. Both surfaces have regions of minimum values, but the region created by the offset method is a small sink, as opposed to the large region of minimum values created by the fixed semi-major axis method. For optimization purposes, it was found to be much easier to find the minimum of the sink than to find the minimum of the large region in Figure 4-19.

The selection of the offset method over the fixed apogee height method is much more obvious. A simple comparison between the two surfaces of Figure 4-18 and Figure 4-20 reveals that the offset surface is much easier to minimize. The fixed apogee height

surface does have clearly defined minima, but they are located at the bottom of very narrow spikes, making optimization quite difficult.

4-2-3-2 Gear Array Optimization Code Structure

Having settled upon a parameterization method, it was necessary to create the code that would allow implementation of that method. Since the code for the implementation of the 113:14 case was created in a fairly general manner, the code structure presented in Figure 4-5 could also be used to solve the gear optimization problem. The only required changes were in the SETLIM routine to specify the different solve-for variables, in FUNC to define the new objective function, and in PGA_SAT to define new genetic algorithm parameters. Additionally, a new DSST input deck was defined. Each of these modifications is discussed in more detail in the following sections.

4-2-3-2-1 Gear Array Genetic Algorithm Variable String Structure

In order to implement the fixed offset method, the strings in the genetic algorithm were defined to be 62 bit binary strings with 31 bits representing the eccentric semi-major axis and 31 bits representing the eccentric orbit eccentricity. There was no need to include the circular orbit semi-major axis in the solve-for string as it was simply a function of these other two variables and the predefined offset.

4-2-3-2-2 Gear Array Objective Function Structure

The objective function routine, FUNC, was the only piece of code that required significant modification for the gear array optimization to be successful. Since FUNC

contains the objective function information and the gear problem dealt with an entirely different objective function than the 113:14 case, it was necessary to write an entirely new FUNC to perform the objective function evaluations for each string. An overview of the evaluation process can be seen in Figure 4-21.

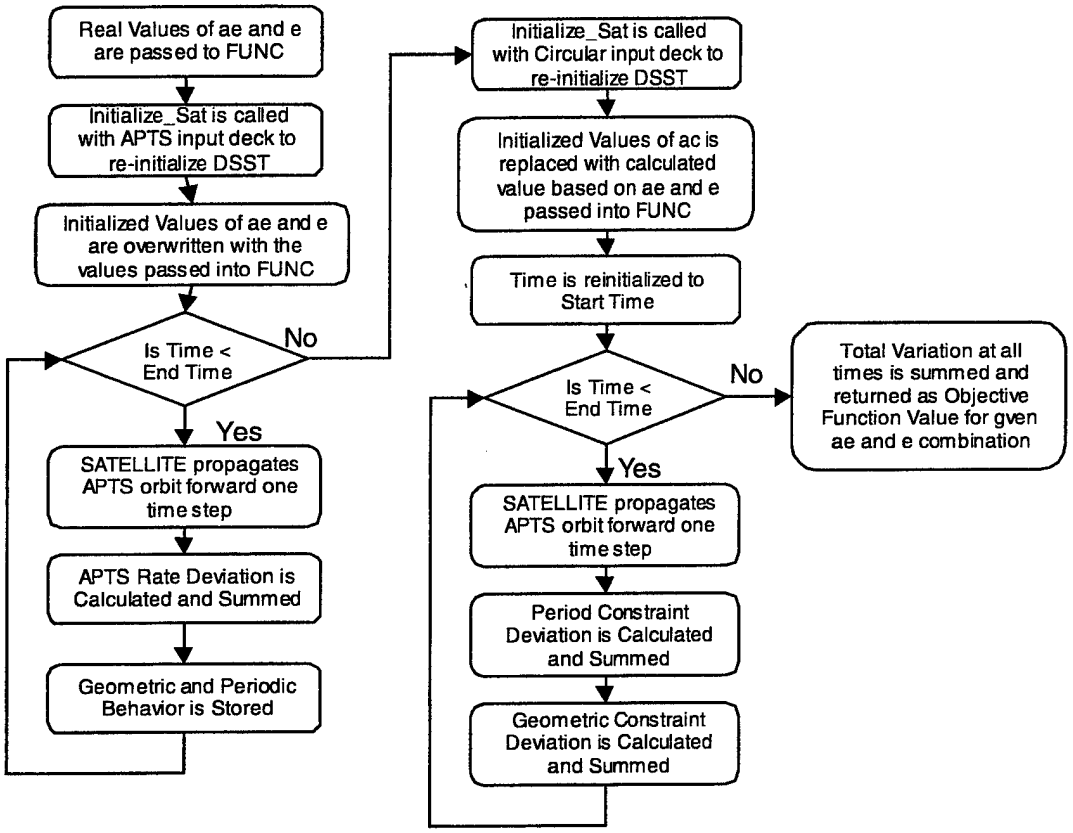


Figure 4-21 Gear Array Optimization FUNC Overview

As can be seen by a simple comparison between the FUNC flowchart seen here and the FUNC flowchart for the 113:14 case shown in Figure 4-2, the main difference between the two objective functions is the addition of a second loop through the DSST propagation. This additional loop is necessary for the gear array optimization because

there are now two separate orbits of interest: the elliptical APTS orbit and the circular orbit. The APTS orbit is propagated first with the values of a_e and e that are contained in the GA string to be evaluated. From this propagation the APTS rate deviation is calculated and the information necessary to calculate the geometric and periodic behaviors is stored.

However, since the geometric and periodic behavior is a relationship between the APTS and circular orbit, the deviation from the desired values cannot be calculated without first determining the motion of the circular component. Therefore, the necessary information for the APTS orbit is stored and the DSST propagator is restarted with the circular orbit information, including the circular semi-major axis that is calculated from the a_e and e values contained in the string. After propagation of the circular orbit, it is then possible to compare the motion of each orbit at each time step and calculate the deviations from desired behavior at each step. The total deviation of all three constraints is then combined in a weighted sum and returned to the genetic algorithm as the objective function for the given string. As was the case with the 113:14 case, the objective function has been designed so that strings that give small variations from the desired behavior will end up with small objective function values and therefore high fitness values.

4-2-3-2-3 Gear Array Genetic Algorithm Parameters

With two exceptions, the specific genetic algorithm parameters used for the 113:14 case and presented in Table 4-2 were also found to work well for the gear array optimization. The two exceptions were the number of iterations to repeat without changing before stopping and the mutation rate. The reason for both of these changes

was related to premature convergence at slightly incorrect answers. It was found, through test runs of the genetic algorithm, that the GA had a tendency to settle upon an answer that was near the optimum, but was not exactly the optimum. By forcing the genetic algorithm to perform more iterations (by increasing the No Change Value) and to maintain more diversity (by raising the mutation rate) it was found that this premature convergence could be avoided. The specific genetic algorithm parameters used for the gear array optimization are summarized in Table 4-6.

Table 4-6 Gear Array Genetic Algorithm Parameters

Parameter	Type/Value
Stopping Rule	No Change
No Change Value	150
Population Size	100
Replaced Per Iteration	25
No Duplicates Flag	True
Mutation and Crossover Flag	True
Mutation Rate	0.02
Crossover Type	Two-Point
Crossover Probability	0.85
Selection Type	Tournament

4-2-3-2-4 Gear Array Propagation Parameters

The design of the gear array was accomplished in the presence of the Zonal Geopotential, employing the J_2 through J_{50} harmonics. All other perturbations were not included in the design process (although they were included in an analysis of the optimal designs). The epoch for all cases was January 1, 1997 and all cases were designed over a one-year time span.

Two different cases were run, both employing the offset method. The first case was a 5:6 gear with an offset of 285 km. This case was chosen as it produced an apogee height of 8050 km. This apogee height is the same as the original seven satellite

Concordia™ equatorial ring of the Ellipso™ constellation and therefore could be used for direct comparison between the original baseline Ellipso™ and the gear design. A nine-satellite 4:5 gear array with a 0-km offset was also investigated as an alternative with two fewer satellites in the array. The input decks for the elliptic and circular arrays of both cases can be found in Appendix B-4.

4-2-4 Genetic Algorithm Performance on Gear Array Design Optimization

After tuning the genetic algorithm with the correct parameters, the genetic algorithm approach proved successful in finding the elements to create optimally designed gear arrays. The required number of iterations also proved to be quite reasonable—on the order of 300-500. Figure 4-22 and Figure 4-23 show how the best objective function of each population slowly evolved to the optimal value for both the 5:6 and 4:5 case. The same plots showing the average value evolution can be found in Appendix B-5.

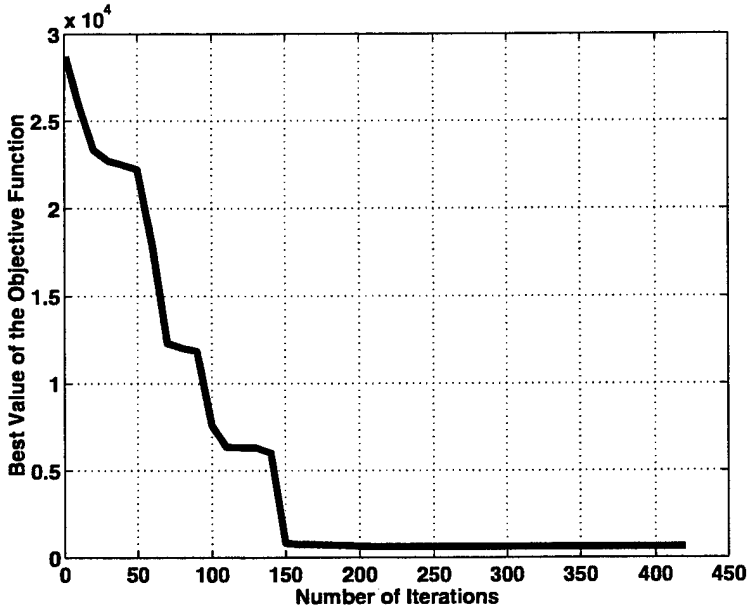


Figure 4-22 5:6 Case Gear Design Genetic Algorithm Convergence Plot

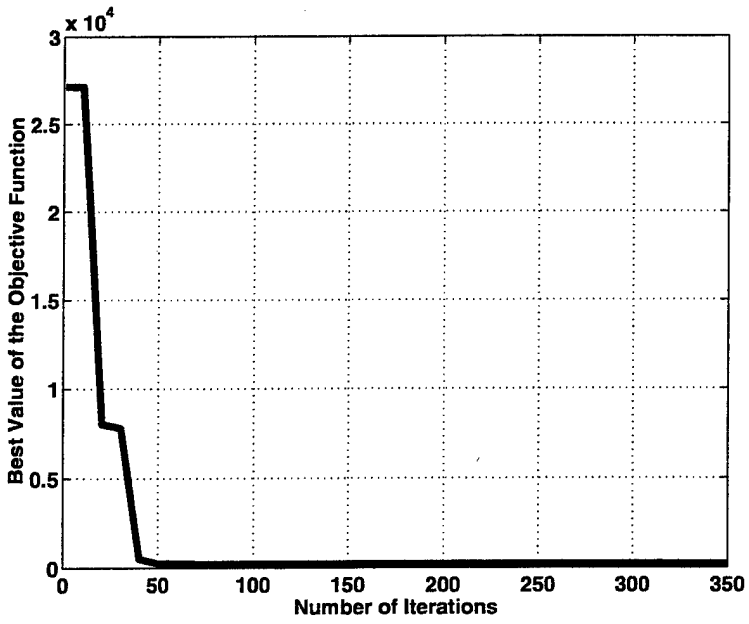


Figure 4-23 4:5 Case Gear Design Genetic Algorithm Convergence Plot

4-2-5 Results of Gear Array Optimization

Successful optimization of the gear problem using genetic algorithms resulted in a simple semi-major axis/eccentricity pair which, when used as the epoch elements of an orbit propagation (under a zonal 50 x 0 field), produced the smallest total deviation from the desired behaviors of periodicity and apogee pointing to the Sun. The optimally designed elements for both the 5:6 gear 8050 apogee and the 4:5 gear 0 km offset are presented in Table 4-7.

Table 4-7 Optimal Gear Array Design Parameters

	5:6 GEAR 8050 APOGEE		4:5 GEAR 0 KM OFFSET	
Array Type	APTS	Circular	APTS	Circular
Number of Satellites	5	6	4	5
Semi-Major Axis (Km)	12537.37	14143.57	12546.57	14555.45
Eccentricity	0.151172	0.0	0.160114	0.0
Apogee Height	8050.	7765.	8177.	8177.
Perigee Height	6149.	7765.	4159.	8177.
Phasing in Mean Anomaly	72	60	90.	72.
Anomalistic Period (secs)	13947.98	16737.58	13980.01	17475.07

4-2-6 Gear Array Performance

Two separate analyses regarding the performance of the optimal orbital elements shown in Table 4-7 can be performed. The first is an analysis into the performance of the orbits in relation to the objective function and the desired behaviors. The second is a comparison of the overall performance of these gear arrays in terms of communications coverage capabilities, when compared to the baseline Ellipso™ constellation.

4-2-6-1 Gear Array Performance Objective Function Analysis

Two separate areas must also be studied in terms of the design behavior with regard to the objective function. First, in an effort to show that the designs achieved by the genetic algorithm are optimal, it should be shown that the APTS, commensurability, and geometric behaviors are all minimized in the 50 x 0 field in which they were designed. Then, since the design optimization was performed only in the presence of zonal fields, it is also necessary to study the behavior of the constellations under full perturbations to ensure the desired behaviors are still met.

4-2-6-1-1 Gear Array Design Accuracy

The design accuracy of the optimized elements is best assessed by analyzing the deviation in each of the three desired behaviors: APTS pointing, gearing ratio, and gearing phase. As these were the three items to be directly optimized, it is expected that their error will be small. For a well designed orbit, it is also desirable that the elements remain fairly constant over time. If the elements decay drastically from their initial values, the gear behavior of the arrays will also decay. Therefore, in assessing the design accuracy, the decay of the orbital elements was also studied.

Table 4-8 and Table 4-9 contain a summary of the statistics for the desired behaviors as well as the orbit elements for the two designed arrays propagated five years from a January 1, 1997 epoch under a zonal 50 x 0 field. The corresponding plots can also be found in Appendix B-6.

Table 4-8 5:6 Gear 8050 Apogee Design Accuracy

	Maximum	Minimum	Mean	St. Dev.
Nominal Pointing Error (Deg)	7.07e-05	-3.43e-05	2.00e-05	2.46e-05
Gearing Ratio Error	3.43e-03	-1.11e-03	-5.53e-06	1.80e-04
Gearing Phase Error (Deg.)	3.63e-05	-6.75e-03	-3.13e-03	2.50e-03
APTS SMA Deviation (km)	0.00e-00	0.00e-00	0.00e-00	0.00e-00
APTS Eccentricity Deviation	3.00e-08	-1.09e-07	-4.65e-08	4.38e-08
APTS Inclination Deviation (Deg)	1.13e-02	3.69e-15	7.20e-03	3.28e-03
APTS Mean Anomaly Deviation (Deg)	7.61e-02	0.00e-00	3.54e-02	2.83e-02
Circular SMA Deviation (km)	0.00e-00	0.00e-00	0.00e-02	0.00e-00
Circular Eccentricity Deviation	4.53e-08	0.00e-00	2.33e-08	1.48e-08
Circular Inclination Deviation (Deg)	2.84e-03	3.69e-15	1.62e-03	8.29e-04

Table 4-9 4:5 Gear 0 km Offset Design Accuracy

	Maximum	Minimum	Mean	St. Dev.
Nominal Pointing Error (Deg)	2.30e-05	-2.92e-04	-1.15e-04	9.31e-05
Gearing Ratio Error	3.69e-03	-1.12e-03	1.13e-06	1.90e-04
Gearing Phase Error (Deg.)	2.49e-13	-5.46e-03	-2.56e-03	2.00e-03
APTS SMA Deviation (km)	0.00e-00	0.00e-00	0.00e-00	0.00e-00
APTS Eccentricity Deviation	3.00e-08	-1.15e-07	-4.90e-08	4.57e-08
APTS Inclination Deviation (Deg)	1.18e-02	3.70e-15	7.56e-03	3.46e-03
APTS Mean Anomaly Deviation (Deg)	7.60e-02	0.00e-00	3.53e-02	2.82e-02
Circular SMA Deviation (km)	0.00e-00	0.00e-00	0.00e-00	0.00e-00
Circular Eccentricity Deviation	4.99e-08	0.00e-00	2.70e-08	1.61e-08
Circular Inclination Deviation (Deg)	3.16e-03	3.69e-15	1.92e-03	8.71e-04

The data found in the above tables and the plots found in Appendix B-6 show that the elements converged upon by the genetic algorithm optimization approach do indeed produce stable orbits that meet the gear criterion specified in the problem. The furthest deviation in either case of any of the three constrained behaviors is less than 0.004 (the Gearing Ratio Deviation). The largest value of the mean error is also on this same order (less than 0.004) and occurs in the gearing phase error in both cases. Analysis of the element decay over the five-year period also shows that under the 50 x 0 zonal field, the orbits are quite stable and experience only very small drift over the entire five-year period.

4-2-6-1-2 Gear Array Orbit Decay Under Full Perturbations

Although the designed orbits appear to meet the gearing criterion and to maintain their stability over the desired five-year lifetime, it is also necessary to study the decay of the orbits under all perturbations during the same five-year period. It is possible that when subjected to third-body point mass effects, atmospheric drag, solar radiation pressure, tesseral harmonics, and solid Earth tides, that the gear behavior seen in the 50 x

0 design space might disappear. Appendix B-6 as well as Table 4-10 and Table 4-11 detail the results of the five-year propagation under full perturbation models.

Table 4-10 5:6 Gear 8050 Apogee Decay Under Full Perturbations

	Maximum	Minimum	Mean	St. Dev.
Nominal Pointing Error (Deg)	8.63e-03	-1.75e-00	-8.72e-01	5.05e-01
Gearing Ratio Error	4.40e-03	-4.99e-04	-3.11e-04	3.32e-04
Gearing Phase Error (Deg.)	5.95e-01	-5.79e-00	-8.80e-01	1.75e-00
APTS SMA Deviation (km)	6.43e-02	-6.30e-03	1.89e-02	1.92e-02
APTS Eccentricity Deviation	2.21e-04	-4.49e-05	5.71e-05	5.60e-05
APTS Inclination Deviation (Deg)	4.82e-02	3.69e-15	2.34e-02	1.42e-02
APTS Mean Anomaly Deviation (Deg)	5.09e-04	-14.65e-00	-4.75e-00	4.16e-00
Circular SMA Deviation (km)	0.00e-00	-2.20e-03	-1.15e-03	7.42e-04
Circular Eccentricity Deviation	1.28e-03	0.00e-00	8.41e-04	3.72e-04
Circular Inclination Deviation (Deg)	8.11e-02	3.69e-15	4.63e-02	2.37e-02

Table 4-11 4:5 Gear 0 km Offset Decay Under Full Perturbations

	Maximum	Minimum	Mean	St. Dev.
Nominal Pointing Error (Deg)	8.12e-03	-1.90e-00	-9.44e-01	5.48e-01
Gearing Ratio Error	7.17e-03	-5.10e-04	-2.54e-04	5.49e-04
Gearing Phase Error (Deg.)	6.21e-01	-6.08e-00	-9.16e-01	1.87e-00
APTS SMA Deviation (km)	7.16e-02	-6.60e-03	2.12e-02	2.14e-02
APTS Eccentricity Deviation	2.52e-04	-4.86e-05	6.61e-05	6.44e-05
APTS Inclination Deviation (Deg)	4.88e-02	3.69e-15	2.35e-02	1.43e-02
APTS Mean Anomaly Deviation (Deg)	5.72e-04	-15.74e-00	-5.01e-00	4.47e-00
Circular SMA Deviation (km)	0.00e-00	-1.70e-03	-8.15e-04	5.75e-04
Circular Eccentricity Deviation	1.12e-03	0.00e-00	6.75e-04	3.53e-04
Circular Inclination Deviation (Deg)	9.40e-02	3.69e-15	5.74e-02	2.60e-02

The designed gear arrays are actually quite stable in the presence of full perturbation models. As expected, the decay of all elements increased when additional perturbations were included, but that increase was not enough to destroy the structure of the array and eliminate the desired behaviors. Probably the most serious decay occurred in the expected mean anomaly of the APTS orbit (measured at each anomalistic period). This value decayed approximately 15° in both cases. This decay was a direct result of the semi-major axis decay due to the addition of solar radiation pressure. And although this semi-major axis decay was small (on the order of 70 m), its corresponding effect on mean

anomaly led directly to error in the gearing phase. In both cases, the phase error decays approximately 6°. This decay is not enough to destroy the gear-like behavior, but it is larger than originally expected.

Two possible explanations exist for the decay seen under the full perturbation model. First is simply the fact that the design was created under a simpler model than it is now being analyzed in. This fact alone is enough to introduce error into the orbit design. Ideally, the orbit would be designed in the presence of all modeled perturbations, not just the zonal field. However, due to the extremely iterative process of genetic algorithms, the choice was made to avoid the increase in computer time that would be incurred through the addition of a full perturbation model.

The second possible way to avoid the increase in decay under the full-perturbation model also was not implemented in this study due to the computer processing time requirements. Due to the large amount of time required for each period of satellite propagation, the optimization of the gear array elements was performed in both cases using only one-year propagations. However, the decay models studied in the tables above presents the total decay over a five-year period (the lifetime of the satellites in question). It is entirely feasible that by performing the optimization using five-year propagations that more optimal (in terms of five-year decay) elements would be the result.

Figure 4-24 demonstrates this fact using the semi-major axis decay history for the APTS orbit of the 4:5 gear 0 km offset case. It can clearly be seen that for approximately the first 400 days, the motion is centered on zero. As 366 days is the length of time for which the design was created, this centered behavior is to be expected, even under the

full perturbation model. However, after the first 400 days, the semi-major axis begins to diverge quite rapidly. Although some of this divergence can be attributed to an increase in drag during this second and third years, by including the entire five year period of interest in the optimization it is thought that this divergence can be further minimized.

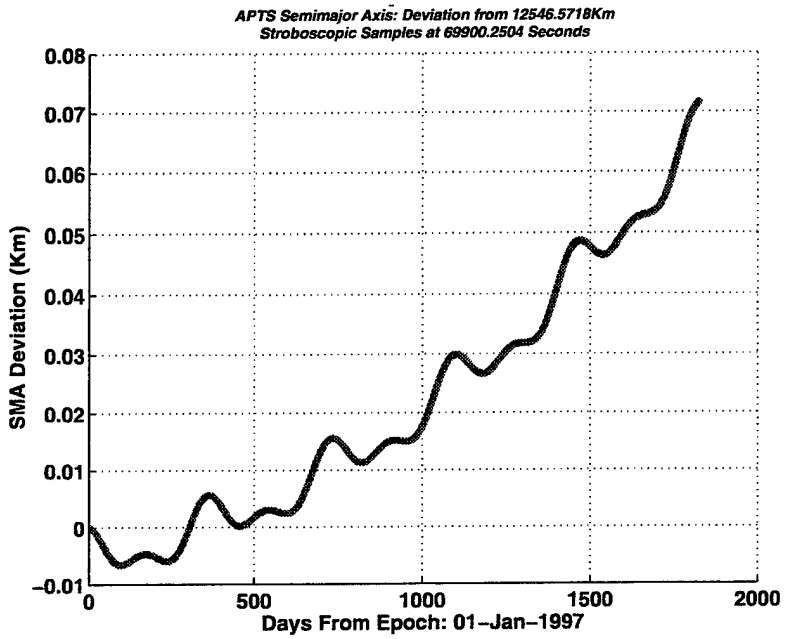


Figure 4-24 4:5 Gear 0 km Offset 5-Year APTS SMA Divergence (Full Pert.)

4-2-6-2 Gear Array Coverage Analysis

A final area of the gear array that is necessary for analysis is its performance relative to the current Ellipso™ baseline constellation consisting of Concordia™ and Borealis™ sub-constellations. To perform this analysis a coverage comparison was prepared for both of the designed gear arrays and the Ellipso™ Concordia constellation which is an equatorial, circular constellation at 8050-km altitude as well.

The coverage analysis performed identifies the minimal elevation angle, the average elevation angle, and the average number of satellites in view at any given time. The data is collected over a two-week time interval and is presented as a function of

latitude at a set of local times (+/- 1-½ hours). The data for all three constellations is presented on the same "wedge" plots, which allows for a simple comparison between the designs.

A sample of one of these wedge plots can be seen in Figure 4-25. This figure contains a comparison of the minimum elevation angles as a function of latitude. It is clear that both the 11-satellite 5:6 gear array and the 9-satellite 4:5 gear array provide better elevation angles than the baseline 7-satellite Concordia™ array. The "wedge" plots for the other areas of interest (average elevation angle and number of satellites in view) at both noon and 3 P.M. local time can be found in Appendix B-7. All figures show that the gear arrays out-perform the Concordia™ array.

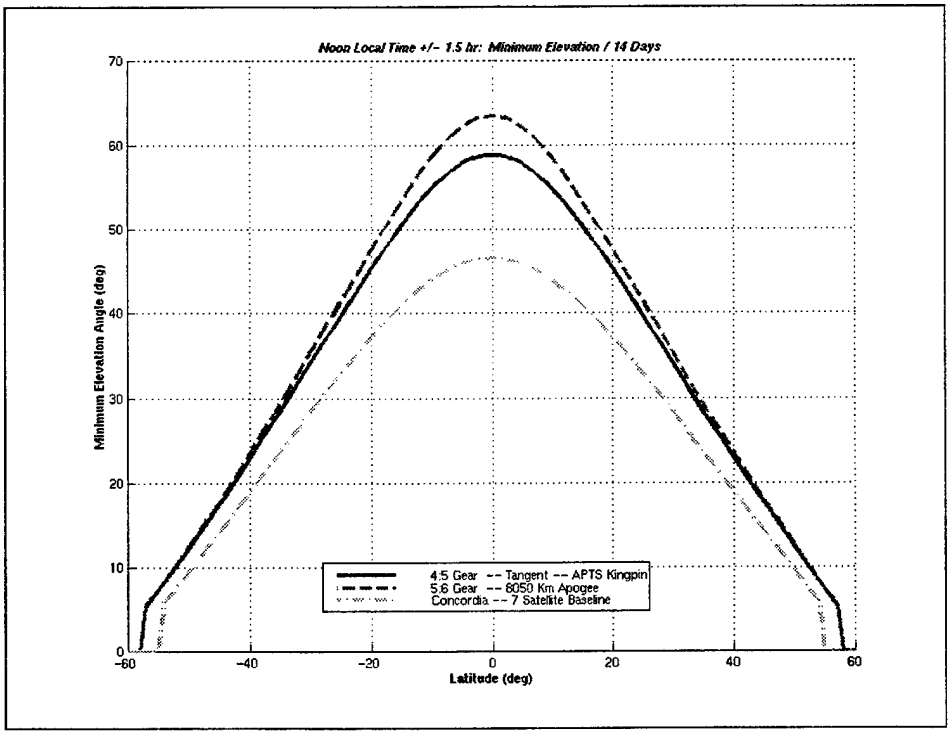


Figure 4-25 Gear Array Minimum Elevation Angle Comparison—Noon Local Time

[This Page Intentionally Left Blank]

Chapter 5 Optimal Constellation Maintenance

This chapter investigates the use of optimization techniques (specifically genetic algorithms) to generate optimal station-keeping strategies, under a variety of constraints, with highly accurate orbit propagation. The basic problem is first formulated, followed by a discussion of previous attempts to solve the station-keeping problem. Genetic algorithms and the two specific methods studied are then described in detail. The first approach is a global approach where all burns required to maintain the orbit for a specified period of time are found. Due to limitations that arose under this approach, a second, more "operational" approach was implemented and tested. The results of these tests, as well as observations stemming from these results, are included.

5-1 Station Keeping Problem Formulation

The objective for station keeping problems is simple: minimize the fuel required to maintain the orbit for a given period of time. Mathematically this can be expressed as seen below:

$$J = \sum_{i=1}^{n} \left\| \Delta \bar{v}_i \right\| \qquad \text{Equation 5-1}$$

where:

i = index variable

n = number of burns used

J = Cost/Objective to be minimized

Δv_i = delta-v required for the i^{th} burn

Without any constraints this problem is quite trivial. The satellite will propagate forward in time and nothing will be required to happen. Therefore, it is necessary to constrain the orbit in such a manner that burns will be required to occur in order that the constraints are met. For this investigation, a box form of constraints was chosen. Each of the elements of the actual orbit was constrained to lie within a given error distance from the reference orbit. The net effect of each of these error distances becomes a box around the reference orbit. As long as the satellite stays within this box, the constraints are met.

A simple two-dimensional rendering of this type of constraint can be seen in Figure 5-1. Here the dot represents the state of the satellite at a given time and the dashed line represents the reference orbit. If state one is taken to be the semi-major axis then it is easily seen that an error limit on this state will translate into an upper and lower limit on the position which the two solid lines represent. In a similar manner, if state two is taken to be the mean anomaly, then the corresponding error limit will translate into a forward and backward limit on the position of the satellite. The combined effect of the constraints on the two states translates into a box. The satellite is now constrained in both directions. In a similar manner, additional constraints will only add dimensionality to or change the shape of this box.

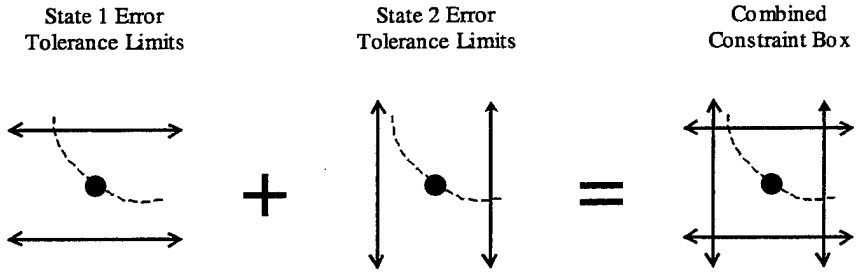


Figure 5-1 Box Constraint Depiction

This type of constraint results in the following mathematical expression:

$$\left|\overline{\alpha}_{ref}(t) - \overline{\alpha}_{actual}(t)\right| \leq \overline{\varepsilon}(t) \qquad \text{Equation 5-2}$$

where:

$\alpha_{ref}(t)$ = vector of reference states at time t

$\alpha_{actual}(t)$ = vector of actual states at time t

$\varepsilon(t)$ = vector of tolerances

Although for this study the constraints were defined to be in terms of the mean orbital elements, there is nothing about this general method that requires this exact selection of constraints. Any quantity that can be measured as a function of time and then compared to a measurable reference quantity should be acceptable. It should also be noted that although they are not detailed here, the system is further constrained by the state equations, which depend upon the system modeling.

5-2 Previous Station-Keeping Attempt Limitations

As the station-keeping problem is a fundamental problem to satellite on-orbit operations, a number of attempts have been made to find efficient solutions. However, each of the schemes studied to date suffers from one or more of the following three limitations.

5-2-1 Simplified Orbit Propagation

In order to lower the complexity of the station-keeping problem to an order that can be solved through traditional optimization methods, many attempts use simplified orbit propagation techniques. This simplification often means that any effects due to higher order orbital dynamics are neglected. Even in cases where higher order orbital

dynamics are included, the dynamics are usually linearized, thereby reducing their true effects. In order to find truly accurate optimal solutions, it is necessary to include as many effects as possible.

5-2-2 Localized/Greedy Strategy

A number of previous attempts also suffer from what is termed greediness. These greedy strategies were discussed previously in section 3-3-1-1. Basically, greedy strategies are those which attempt to find the optimal path from a current state to a desired state, without worrying about the effect the chosen path will have on subsequent decisions.

In terms of the station-keeping problem, a greedy strategy is one that waits until a violation of the constraints is about to occur and that then finds the optimal burns that are necessary to return the spacecraft to a non-violating state. By waiting until a violation is approached and then finding the optimal burns necessary to return the spacecraft to a non-violating state, it is often the case that a need to perform more expensive burns later on is created. For greedy algorithms, however, this fact is ignored. On the other hand, a non-greedy strategy is one that calculates all the burns required to maintain the satellite within the given constraints for the entire length of time specified.

5-2-3 Requirement of Pre-defined Targeting Scheme

A direct result of the localization of previous methods is that a targeting scheme must be defined prior to attempting the optimization. As mentioned previously, upon a spacecraft's approach of a violation, previous methods attempt to find the optimal way to return a spacecraft to a non-violating state. However, as an infinite number of non-

violating states are available, there must be some pre-determined manner of defining which of the available states should be targeted. Without a well-defined knowledge of the state space, it is very difficult to define the proper state to label as the desired target. An incorrect choice of targeted state can have a drastic effect on the amount of fuel required, making this an important aspect of the optimization problem.

5-3 Global Station Keeping Approach

The global station keeping approach was an attempt to find a way to overcome the three limitations of previous studies listed above. It was thought that by solving for all of the burns required over the lifetime of a specified satellite, that the greediness of previous approaches could be overcome. Additionally, if this solution was accomplished using accurate orbit propagation techniques, the accuracy limitation could be overcome, as well. In order to overcome these limitations, however, an optimization method that would allow very accurate orbit propagation while at the same time providing a global, non-greedy perspective to the optimization was required. A parallel genetic algorithm proved to be a viable tool for accomplishing this objective.

Due to the nature of the genetic algorithm operators of selection, crossover, and mutation, genetic algorithms have distinct advantages over other optimization techniques in overcoming the limitations of previous station-keeping methods. First, because these operators do not rely in any manner upon the gradient or derivative information of the problem, they allow for utilization of very accurate orbit propagation methods. In analytic and other attempts at solving the minimum fuel, path-constrained problem, it is often necessary to calculate some level of derivative information. The accuracy of the model to be applied to the solution method is dependent upon the accuracy of these

derivatives. However, in a genetic algorithm solution process, derivative information is not necessary, thereby allowing for much easier application of more detailed system models.

The lack of derivative information along with the robust, global nature of genetic algorithms also allows the other two limitations of previous studies to be overcome. Unlike some methods which are forced to break the problem into various sections (i.e. coasting arcs, impulsive arcs, etc.), genetic algorithms are able to arrive at a solution to the problem over entire pre-defined intervals. This ability allows for a non-greedy approach to be taken, which in turn eliminates the need for a targeting scheme to be defined. During the course of operation, the genetic algorithm defines its own targeting scheme such that the total effect of all the burns over the period of time specified is optimal.

5-3-1 Global Station Keeping Implementation[83]

After deciding upon the optimization technique to apply to the global station-keeping problem, some sort of reasonable implementation scheme was necessary. The implementation was accomplished through the use of three main software libraries: the Draper Laboratory Semi-analytical Satellite Theory standalone orbit propagator package (DSST); the Mississippi State implementation of Message Passing Interface (MPI) known as MPICH; and the Argonne National Laboratory parallel version of genetic algorithms known as Parallel Genetic Algorithm Package or PGAPack. DSST was used as the orbit propagator, MPICH was used to provide parallelism and to decrease

computation times, and PGAPack was used to provide the optimization capabilities. As PGAPack is a complete genetic algorithm package, implementing it with the DSST software was simply a matter of specifying the variables to be encoded into each string, the objective function to optimize, and the values of certain genetic algorithm parameters.

5-3-1-1 Global Station Keeping Variable Description

In order to find the optimal burn strategy required to maintain the desired orbit of a given satellite, it is necessary to solve for three different things: the time of each burn, the direction of each burn, and the magnitude of each burn. For this particular investigation these variables to be optimized were specified as shown in Figure 5-2 and as detailed in Table 5-1.

> **[Burn 1 Time] [Burn 2 Time]...[Burn m Time]** [Burn 1 Tangential Component] [Burn 2 Tangential Component]...[Burn m Tangential Component] **[Burn 1 Normal Component] [Burn 2 Normal Component]...[Burn m Normal Component]** [Burn 1 Radial Component] [Burn 2 Radial Component]...[Burn m Radial Component] **[Burn 1 On/Off Flag] [Burn 2 On/Off Flag]...[Burn m On/Off Flag]**

Figure 5-2 Global Station Keeping Approach Genetic Algorithm String Structure

One variable was created for the time of each burn, each of the three burn components and a flag for each burn to indicate whether or not the effect of the given burn should be included in the propagation. The flag variable was necessary to allow for n-burn solutions where n was not known ahead of time and was some number less than

---

[83] Smith, James E., Ronald J. Proulx, Paul J. Cefola, and John E. Draim. *Optimal Station-Keeping Strategies via Parallel Genetic Algorithms.* Paper AAS 99-123, AAS/AIAA Space Flight Mechanics Meeting, Breckenridge, Colorado, 7-10 February 1999.

the total allowable m burns. This structure led to a total of five variables per burn. Thus, for a typical case in which a maximum of ten burns was allowed, the resulting string to be optimized by the genetic algorithm would be 50 variables.

Table 5-1 GA Global Station Keeping Solution Process Variable Allocation

Variable	Number/Burn	Lower Limit	Upper Limit	Units
Elapsed Time from Epoch	1	0.00	Maximum Time	Seconds
Magnitude of Burn Components	3	- Maximum Component Magnitude	+ Maximum Component Magnitude	m/s
On/Off Flag	1	0.00	1.00	N/A

5-3-1-2 Global Station Keeping Objective Function

Although an objective function for the general station-keeping problem was defined previously in section 5-1, this objective function is not sufficient for application to a genetic algorithm. Genetic algorithms work best with unconstrained objective functions. Therefore, the constraints must be adjoined to the objective function in a manner that yields an acceptable objective function for solution by genetic algorithms. A first attempt at this was a simple linear combination as seen below:

$$J = \sum_{i=1}^{n} \left\| \Delta \bar{v}_i \right\| + \int_0^t \bar{\lambda}^T \left\{ \left| \bar{\alpha}_{ref}(t) - \bar{\alpha}_{actual}(t) \right| - \varepsilon(t) \right\} dt \qquad \text{Equation 5-3}$$

where:

λ = vector of scale factors

It can readily be seen that using this objective function will attempt to force the difference between the reference state and the actual state to be equal to the defined tolerance in order that the resulting effect on the objective function is equal to zero. This is not the desired behavior. As long as the difference between the actual and the reference states is less than the defined tolerance, the contribution to the objective function should be negligible.

In order to create this behavior, it was necessary to define an objective function that is in some sense two objective functions. The effect of the deviation from the reference state is only included in the objective function if the deviation is greater than the defined tolerance. Otherwise, the objective function is reduced to the unconstrained minimum fuel problem objective defined previously. The following is the resulting mathematical formulation of the objective function that was incorporated into the genetic algorithm solution process.

$$J = \sum_{i=1}^{n} \left\| \Delta \bar{v}_i \right\| + \int_0^t C(\mathrm{t})\mathrm{dt} \qquad \text{Equation 5-4}$$

$$\text{If } \left| \bar{\alpha}_{ref}(\mathrm{t}) - \bar{\alpha}_{actual}(\mathrm{t}) \right| \geq \bar{\varepsilon}(\mathrm{t}) \text{ then } C(\mathrm{t}) = \bar{\lambda}^T \left(\left| \bar{\alpha}_{ref}(\mathrm{t}) - \bar{\alpha}_{actual}(\mathrm{t}) \right| - \bar{\varepsilon}(\mathrm{t}) \right) \quad \text{otherwise C(t) = 0}$$

5-3-1-3 Global Station Keeping Genetic Algorithm Parameters

As with the optimal design cases presented in the previous chapter, since the performance of the genetic algorithm is very dependent upon a number of predefined parameters, tuning of the approach was necessary to determine appropriate values of a number of genetic algorithm parameters. The specific genetic algorithm parameter values that were found to work best can be seen in Table 5-2.

Table 5-2 Global Station Keeping Approach Genetic Algorithm Parameters

Parameter	Type/Value
Stopping Rule	No Change
No Change Value	2000
Population Size	100
Replaced Per Iteration	25
No Duplicates Flag	True
Mutation and Crossover Flag	True
Mutation Type	Gaussian
Real Mutation Constant	0.5
Mutation Rate	1/String Length
Crossover Type	Two-Point
Crossover Probability	0.85
Selection Type	Tournament

These parameters are the same as those used in the previous cases with the exception of the mutation parameters. In the two design cases, the strings being optimized were simply a binary representation of the variables to be optimized. For the station-keeping problem, however, the required accuracy eliminated a binary representation as a possible choice. Instead each allele (i.e. numerical value) in the string was represented by the real value of the variable it was representing. The genetic algorithm operators of selection and crossover were unaffected by this change in string structure, but the mutation operator required modification.

In a binary representation, if an allele is randomly selected for mutation, it simply undergoes a "bit-flip" operation where its value is changed from 0 to 1 or vice versa. However, for a real valued allele, the possible values that it can take are infinite. Therefore, a simple bit-flip operation is impossible. Instead, the mutation operator takes the following form:

$$v \leftarrow v \pm p \times v \qquad \text{Equation 5-5}$$

where:

v = existing allele value

p = a percentage operator

Thus, if a real valued allele is selected for mutation, the operation it undergoes is simply the addition of a positive or negative percentage of its current value to its current value.

In the PGAPack software there are three possible options for how p is selected. The first option is to simply define p to be a constant percentage. The second is to define a range from which p is selected via a uniform distribution. The final option is to select p from a Gaussian distribution with zero mean and predefined standard deviation. This last option, with a predefined standard deviation of 0.5 is the option that was found to provide the best convergence rate for this application.

5-3-1-4 Useful Modifications to the Global Station Keeping Implementation

Although the formulation of the station keeping problem presented in the previous sections can be used successfully to find near optimal station-keeping strategies, a number of modifications were found to be useful in helping the genetic algorithm to arrive at near-optimal solutions in a more efficient manner. These changes fall into one of two categories: modifications to the objective function or modification to the variable structure. An explanation of these modifications follows.

5-3-1-4-1 Modifications to the Global Station Keeping Objective Function

In order to force faster convergence of the genetic algorithm, a number of changes were made to the objective function. Section 5-3-1-2 presents the problem in a very straightforward manner, but the genetic algorithm has some difficulty in navigating the resulting solution space. After a number of trials, the following three part objective function demonstrated the best behavior in forcing the genetic algorithm to the optimal solution.

J = DeltaV Contribution + Deviation Contribution + Time Contribution Equation 5-6

Delta-V Contribution: Previously, the delta-v contribution to the objective function was simply a sum of the magnitudes of the burns used (n). However, the inclusion of an on/off flag in the variable string caused difficulty with this formulation. If the flag for a given burn was in the off position, the genetic algorithm did not receive any feedback on whether or not the components of that burn should be large or small. This made it very difficult for new burns to ever be turned on as their magnitudes were usually too large to maintain the trajectory within the desired box. Instead, it was found that a more appropriate function was to minimize the sum of all allowable burns (m) as shown below:

$$\text{Delta V Contribution} = \sum_{i=1}^{m} \left\| \Delta \bar{v}_i \right\| \qquad \text{Equation 5-7}$$

Deviation Contribution: The deviation constraint described previously is sufficient to obtain the desired results. It was found, however, that it was useful to have some sort of reward for having more burns on than off. It was easier for the genetic algorithm to zero out a burn that was on but unnecessary than it was for the genetic algorithm to turn on a

needed burn. For example, if a four-burn strategy is the optimal solution, it was seen to be easier to zero out one burn in a five-burn solution than to add a burn to a three-burn solution. Therefore, to present some sort of reward for having more burns turned on, the deviation contribution was changed as shown below:

$$\text{Deviation Contribution} = \left(\int_0^t C(t)dt \right) \Big/ \sqrt{n} \qquad \text{Equation 5-8}$$

With the number of burns, n, in the denominator of the deviation contribution, increasing the number of burns has the same effect as lowering the weighting on the deviation constraint and therefore rewards solutions that have a higher number of burns turned on.

Time Contribution: Although not present in the original objective function, another useful modification was found to be the inclusion of a time of first deviation parameter. This parameter was simply a measure of the time until the first violation of the box constraints occurred. If the system did not exit the box until later on in the time period of interest, this was considered better than a solution which caused an earlier exit. Early runs of the algorithm without this addition to the objective function were found to converge to solutions in which the last half of the trajectory was maintained within the box, but the first half had violations. However, as the early burns have an impact on the entire trajectory it was very difficult to change the early burns in a manner that eliminated the early violation. By adding the time until first deviation parameter to the objective function, strings that violated near the beginning of the time period were quickly eliminated.

Time Contribution = Maximum Time – Time Until First Deviation Equation 5-9

5-3-1-4-2 Modifications to the Global Station Keeping Variable Structure

Two changes to the variable structure presented in Table 5-1 were found to be very useful in causing faster convergence of the genetic algorithm. The first of these changes relates to the time until first deviation parameter that was added to the objective function as discussed above. It was noted that under all circumstances, if the orbit is to be maintained within the desired box, at least one burn must occur before the first deviation of the uncontrolled orbit occurs. This led to a new constraint on the time of the first burn. Rather than being constrained to occur somewhere between time zero and the maximum time, the first burn was constrained to occur before the time of the first deviation.

The second change to the variable structure did not stem from the nature of the station-keeping problem but rather from the nature of genetic algorithms, themselves. In the initial formulation detailed in Table 5-1, the variable which the genetic algorithm sees for crossover and mutation can have any value, as long as this value falls within the limits listed in the table. For example a time variable can have the value 20,000 seconds while the burn component might be of the form 0.05 m/s. This formulation led to the genetic algorithm operators not having as significant an effect as expected.

To combat this problem, the variables were scaled so that the maximum and minimum allowable value for each type of variable was the same. This was done by setting the minimum value of each variable to zero and the maximum to two. For example, for the burn components a value of zero was made to correspond to the negative of the maximum allowable magnitude, a value of one was set to correspond to a zero magnitude, and a value of two was set to correspond to the positive maximum allowable

magnitude. A similar scale was created for the time and flag variables. As all variables in the string were now identically scaled, this allowed crossover and mutation operators to function more efficiently. The variables were then converted to the actual values prior to the objective function evaluation.

5-3-1-4-3 Effectiveness of the Global Station Keeping Modifications

Prior to implementation of the modifications described in this section, the genetic algorithm required between 30,000 and 50,000 iterations to arrive at near-optimal solutions. Even after this large number of iterations, 30% or more of the time the solution to which the genetic algorithm converged did not maintain the satellite within the desired box constraints. After the modifications, all cases have been found to converge in 10,000 to 20,000 iterations, and the majority of the time, the station-keeping strategies generated, meet the desired constraint conditions

5-3-1-5 Computer Implementation of the Global Station Keeping Approach

The implementation of the solution process for the station-keeping problem followed closely the structure used to implement the 113:14 and gear design optimization problems. However, as the objective function to be optimized in the station keeping problem was more involved than the two design objective functions, it was necessary to make some significant changes to that program structure. These changes are illustrated in Figure 5-3 and discussed below.

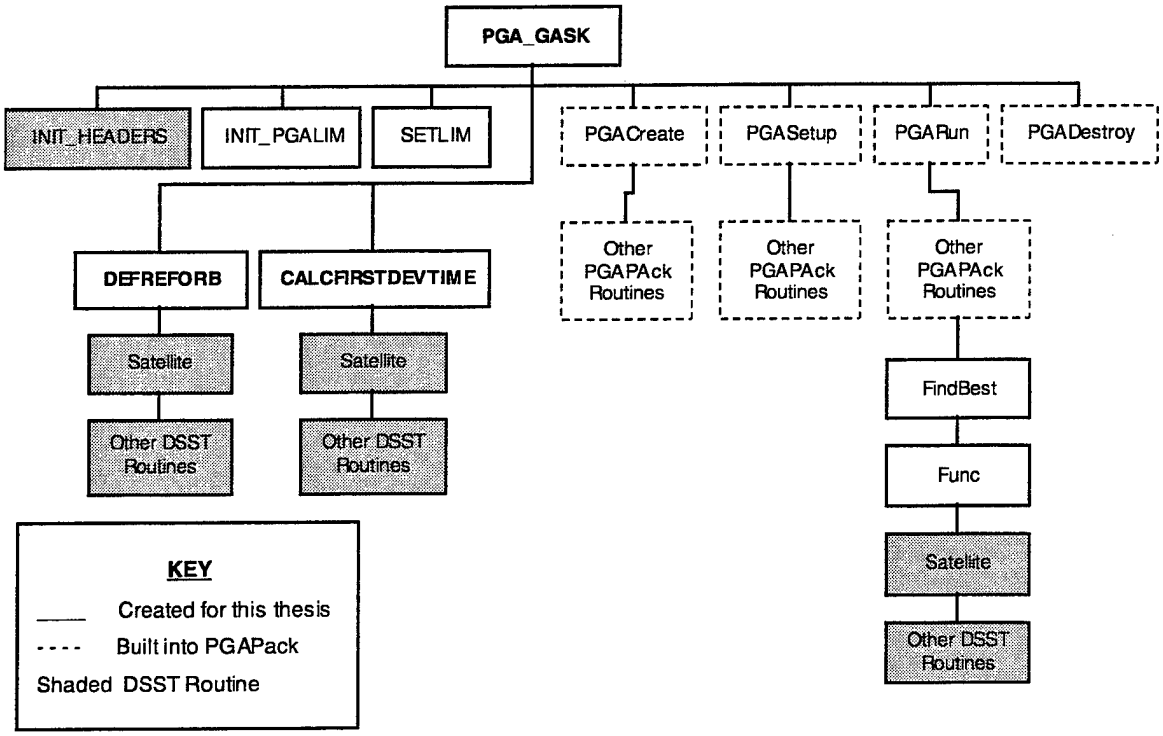


Figure 5-3 Global Station Keeping Genetic Algorithm Software Overview

5-3-1-5-1 Code for Reference Orbit Definition

The first modification to the PGA_SAT structure was the addition of a call from the PGA routine to a newly created subroutine known as DEFREFORB for DEFine REFerence ORBit. This routine simply reads in the reference orbit input file and propagates the reference orbit forward a predetermined number of time steps, storing the orbital elements at each step. Since the necessary data is stored in a global array, it is only necessary to call this routine one time at the beginning of the genetic algorithm optimization.

5-3-1-5-2 Code for Time of First Deviation Calculation

A second routine that also had to be created for this application of genetic algorithms was the routine known as CALCFIRSTDEVTIME for CALCulate FIRST DEViation TIME. As discussed in section 5-3-1-4-2, the performance of the genetic algorithm was greatly improved by modifying the variable structure such that the time of the first burn was constrained. Because at least one burn must occur before the time of the first violation of the constraints or deviation outside of the constraint box, the upper limit on the first time variable was set to be equal to the time of the first deviation. However, this time had to be calculated after the reference orbit had been defined, but before the variable strings were created. The subroutine CALCFIRSTDEVTIME was created and called from the PGA program in order that this objective might be met.

5-3-1-5-3 Objective Function Code Structure (FUNC)

The final required code modification/creation was the formulation of the desired objective function into a routine through which each string could be evaluated. As with all PGAPack objective function routines, the variables to be passed to the objective function routine are simply those variables that make up the strings (in this case burn times, magnitudes and on/off flags). The desired result of the call to the objective function is then simply a numerical value of the performance of the given string against the objective function. The process by which the numerical objective function value was generated for the global station keeping optimization problem is illustrated below in Figure 5-4.

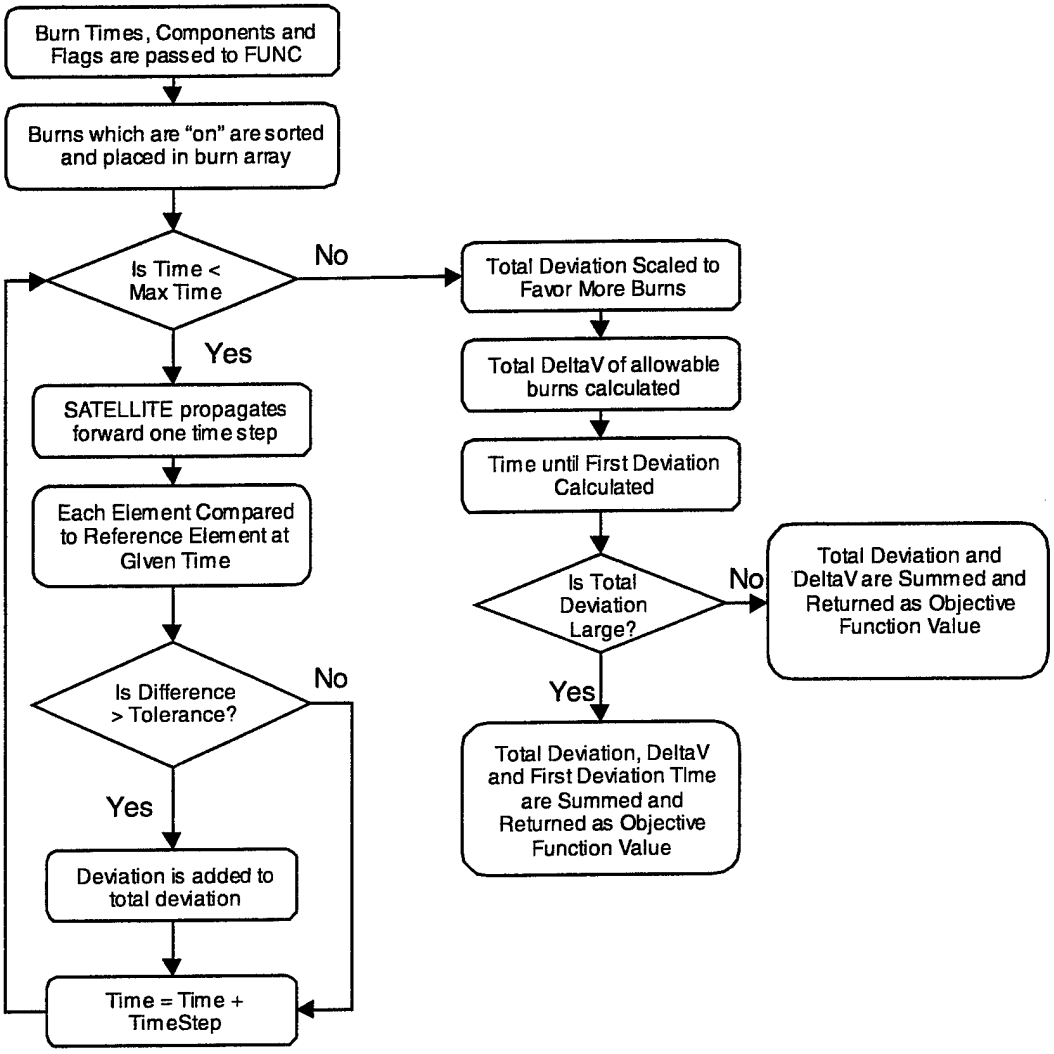


Figure 5-4 Global Station Keeping Optimization FUNC Routine Overview

The process depicted in Figure 5-4 is simply a coded representation of the modified objective function as discussed in section 5-3-1-4. For a given string to be evaluated, the information from that string, namely burn times, components, and on/off flags are passed to the objective function routine, FUNC. FUNC then determines which burns are "on" and sorts those burns into an array to be used by DSST. Following that sorting, DSST is used to perform a full propagation of the orbit, incorporating the burns

at the specified times. The resulting trajectory is then compared to the previously stored reference trajectory and for those times at which the trajectory is outside of the allowable tolerance limits, the deviations of the elements are summed. The result of the propagation is a numerical value that represents the total deviation of the trajectory that the given burns create. For trajectories that cause large deviations, the deviation value is then summed with the total ΔV and also a time parameter. As discussed in section 5-3-1-4-2, this technique helps with convergence as strings that cause violations early in the time period of interest are quickly eliminated. For those strings whose violation value is below a predetermined limit, only the violation and the ΔV are summed and returned as the objective function. Ideally, the violation value goes to zero and all that is seen in the objective function is the ΔV value which can then be minimized.

5-3-2 Global Station Keeping Approach Test Cases

This global genetic algorithm station-keeping solution process was evaluated through a number of test cases. Each case was accomplished using propagations of the Ellipso^{TM} Borealis sub-constellation. This constellation was chosen because of the tight element limits that are imposed on the orbits in order to maintain certain desired constellation characteristics. Using the Borealis sub-constellation for testing of this method also allowed for direct comparison to the results of previous studies that also used this sub- constellation as a test case.

5-3-2-1 Global Station Keeping Case Descriptions

Although a number of trial cases were run to test the genetic algorithm software, only two cases are presented here as a sample of the type of problem this method was successful in solving. Both of the cases presented are 90-day runs of the Ellipso™ Borealis satellite propagated as detailed in the following sections.

Note that although the original intent of this optimization approach was to find the optimal burns for the entire lifetime of a given satellite (in this case the 5-year lifetime of the Borealis™ sub-constellation), computational issues prevented this from actually being accomplished (see section 5-3-4-1). Instead of a full five-year optimization, a ninety day time frame was found to be long enough to provide complexity while still maintaining a problem that was computationally feasible. Therefore, both cases presented in the following sections are ninety-day rather than 5-year optimizations.

5-3-2-1-1 Global Station Keeping Reference Orbit Definition

As described previously in section 5-3-1-5-3, the objective function for this solution process takes into account the deviation of the propagated orbit using a given set of burns, from a predefined reference orbit. Therefore, an important aspect of this solution process is the definition and propagation of the reference orbit.

For all cases presented in this thesis, the reference epoch was set to the vernal equinox of March 21, 2000. The genetic algorithm design process of the 113:14 case was then applied to obtain the initial elements that gave the smallest deviation from the 113:14 desired behaviors over a 90-day period. The resulting elements in a J2000 True of Date reference frame and corresponding Ellipso, Inc. defined tolerances for the chosen

satellite (a node at noon Borealis satellite) are as listed in Table 5-3. A spacecraft mass of 1250 kg and area of 43.3 m^2 are also assumed in all cases

Table 5-3 90-Day Optimized Ellipso Borealis™ Node at Noon Epoch Elements and Tolerances

Element	Reference Epoch State	Tolerance
Semi-major axis (km)	10496.8839	+/- 1.0000
Eccentricity	0.3328	+/- 0.0003
Inclination (deg)	116.5577	+/- 0.0500
Right Ascension of Ascending Node (deg)	0.0000	+/- 0.5000
Argument of Perigee (deg)	260.0000	+/- 1.0000
Mean Anomaly (deg)	0.0000	+/- 1.0000

The reference orbit for both cases was then created using the DSST software with the epoch elements from Table 5-3 propagated using a 50 x 0 gravitational field. The result was an orbit which gave the best performance to the desired 113:14 Sun-synchronous behavior. This orbit was stored and the burns under real perturbations were computed which would maintain the difference between the actual orbit and this reference orbit within the given tolerances. The input deck used in the creation of this reference orbit can be found in Appendix C-2-1.

5-3-2-1-2 Global Station Keeping Case 1—Epoch Aligned with Reference Elements

Case one was perhaps the simplest of all cases to run, but despite its simplicity, was a useful case for demonstrating the feasibility of the genetic algorithm method. It also proved to be a helpful case from which to gain an initial understanding about the behavior of the algorithm. For this case, the propagation was begun with both the actual and reference orbits aligned. In terms of the box constraints described previously, this was equivalent to starting the satellite in the center of the box.

The fully perturbed orbits were propagated using the epoch elements which were aligned with the reference epoch elements and a 21 x 21 gravitational field with drag, solar-radiation pressure, solar and lunar third-body point mass disturbances, and solid-earth tide effects also included. A Jacchia-Roberts atmospheric density model was used in the determination of the drag effects. The input deck that specified all of these perturbations can be found in Appendix C-2.

The behavior of the uncontrolled orbit is detailed in Table 5-4 and plotted in Appendix C-3-1. It can clearly be seen that by including the full perturbations on the orbit for a 90-day period that the semi-major axis, eccentricity, and Mean Anomaly all drift beyond the allowable range. The largest of these drifts is the Mean Anomaly drift of 37.7° as shown in Figure 5-5. Due to this large drift the mean anomaly was found to be the driving factor in this case.

Table 5-4 Global Case 1 Epoch Elements and Uncontrolled Deviations

Element	Epoch State	Max. Deviation	Violation?
Semi-major axis (km)	10496.8839	1.68922	Yes
Eccentricity	0.3328	0.00047	Yes
Inclination (deg)	116.5577	0.00594	No
RAAN (deg)	0.0000	0.05282	No
Argument of Perigee (deg)	260.0000	0.18047	No
Mean Anomaly (deg)	0.0000	37.73677	Yes

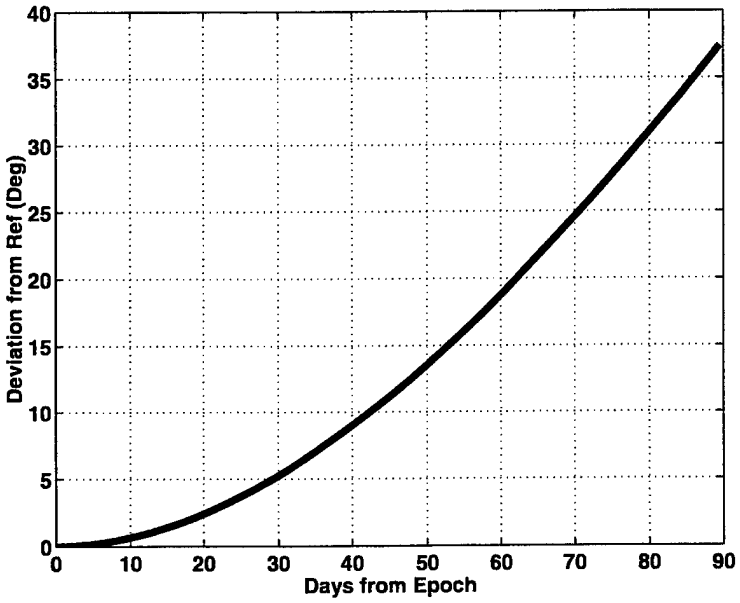


Figure 5-5 Global Case 1 Uncontrolled Mean Anomaly Drift (Limit = 1°)

5-3-2-1-3 Global Station Keeping Case 2—Epoch Elements at Extreme Limits

In order to offset the simplicity of the first case, the second case was designed to be as difficult as possibly could be expected. After proving feasibility with case one, it was desirable to prove the robustness of the genetic algorithm method with case two. In order to do this, a large percentage of the tolerance of each limit was added to or subtracted from the reference epoch state such that the actual orbit was almost as far away from the reference as the tolerances would allow. The direction of motion of each element was also determined such that a violation of each element was almost certain to occur within the first few days of propagation. In terms of the box constraints, this was equivalent to placing the satellite on the corner of the box with a velocity that will force it to leave the box almost immediately.

The epoch elements and corresponding deviations can be seen in Table 5-5 and are plotted in Appendix C-4-1. This table and corresponding plots show that all elements do indeed drift beyond the allowable states if the orbit is not controlled. As with case one, mean anomaly has the largest deviation but semi-major axis and argument of perigee deviations are also quite large. The argument of perigee and mean anomaly histories were found to be the most difficult to control. Their uncontrolled history plots are presented in appendix C-4-1 for comparison to the future controlled histories.

Table 5-5 Global Case 2 Epoch Elements And Uncontrolled Deviations

Element	Epoch State	Max. Deviation	Violation?
Semi-major axis (km)	10495.9739	2.66704	Yes
Eccentricity	0.3331	0.00078	Yes
Inclination (deg)	116.5082	0.05574	Yes
RAAN (deg)	359.5020	0.534322	Yes
Argument of Perigee (deg)	259.0200	1.49870	Yes
Mean Anomaly (deg)	0.9000	73.44632	Yes

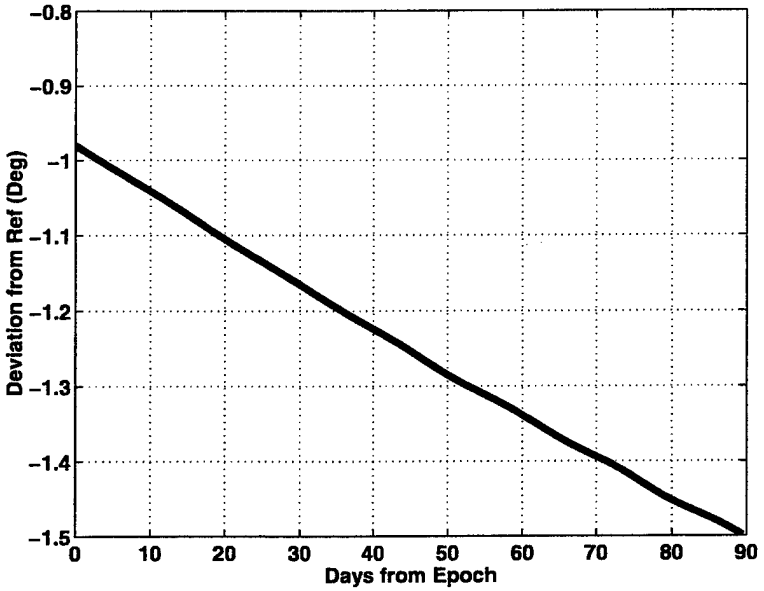


Figure 5-6 Global Case 2 Uncontrolled Argument of Perigee Drift (Limit = 1°)

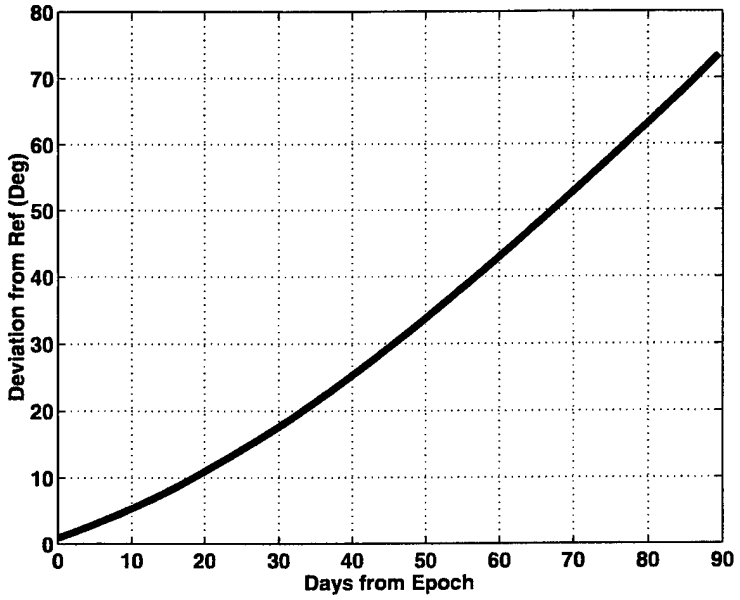


Figure 5-7 Global Case 2 Uncontrolled Mean Anomaly Drift (Limit = 1°)

5-3-2-2 Global Station Keeping Results

The result of the genetic algorithm optimization performed for each of the two cases described above is an n-burn solution that maintains all six of the actual orbital elements within the desired tolerances from the reference orbit. The resulting element history plots for Case 1 and Case 2 can be found in Appendix C-3-2 and Appendix C-4-2, respectively. Some observations regarding these results as well as some details of the results are presented below.

5-3-2-2-1 Case 1 Results—Epoch Elements Aligned with Reference Elements

Based on the uncontrolled history plots for this case (see Appendix C-3-1), it was determined that no more than five burns should be necessary to maintain the orbit within the desired tolerances over the specified 90-day period. Of the allowable five burns, the genetic algorithm converged on the four-burn solution displayed in Table 5-6.

Table 5-6 Global Case 1 Burn Times and Components

Burn Time (Since Epoch)	Mean Anomaly at Burn Time	Radial (m/s)	Along-Track (m/s)	Cross-Track (m/s)	Magnitude (m/s)
07d 23h 43m 44.98s	173.33°	0.0002	0.2572	0.0001	0.2572
19d 06h 30m 50.77s	196.52°	0.0001	0.0324	0.0000	0.0324
20d 12h 41m 33.55s	250.21°	0.0000	0.0107	0.0000	0.0107
46d 00h 38m 01.39s	179.99°	0.0000	0.2690	0.0000	0.2690
				Total DeltaV	**0.5693**

A number of observations can be made regarding this solution. First is the fact that the solution contains only along-track components. Additionally, the mean anomalies of the two burns with largest magnitudes reveal that these two burns occur very close to apogee. Both of these observations are useful in showing that the solution is at least a near-optimal solution.

For this case, the two elements that require controlling are mean anomaly and eccentricity. Based on astrodynamics, it can be shown that the most efficient way to control these two elements is to perform burns near perigee or apogee and to burn in such a manner that the full effect of the burns goes toward controlling these two elements. By burning at apogee, in an along-track direction, no fuel is wasted in controlling inclination, ascending node, or argument of perigee trajectories. The solution in Table 5-6 clearly meets these two requirements for efficiency.

An additional observation which further suggests near-optimality of this solution is that one or more of the controlled element histories ends at the edge of the designated tolerance box. In this case, both the final eccentricity and mean anomaly deviations are equal to the defined tolerances at the end of the run (see Figure 5-8 and Figure 5-9). Although this behavior introduces problems if this method is to be used for long-term station keeping (see section 5-3-4-2), it does help to show optimality of the solution. By burning just enough to be exactly at the defined tolerance at the end of the predetermined

time period, the minimum amount of fuel is consumed. Given the locations of the four burns, any decrease in the amount of fuel expended will cause a violation of the constraints. On the other hand, any increase in the amount of fuel expended could possibly cause the end point of the trajectory to move away from the tolerance limit, but there is no benefit to this change. Instead, it will only serve to increase the delta-v.

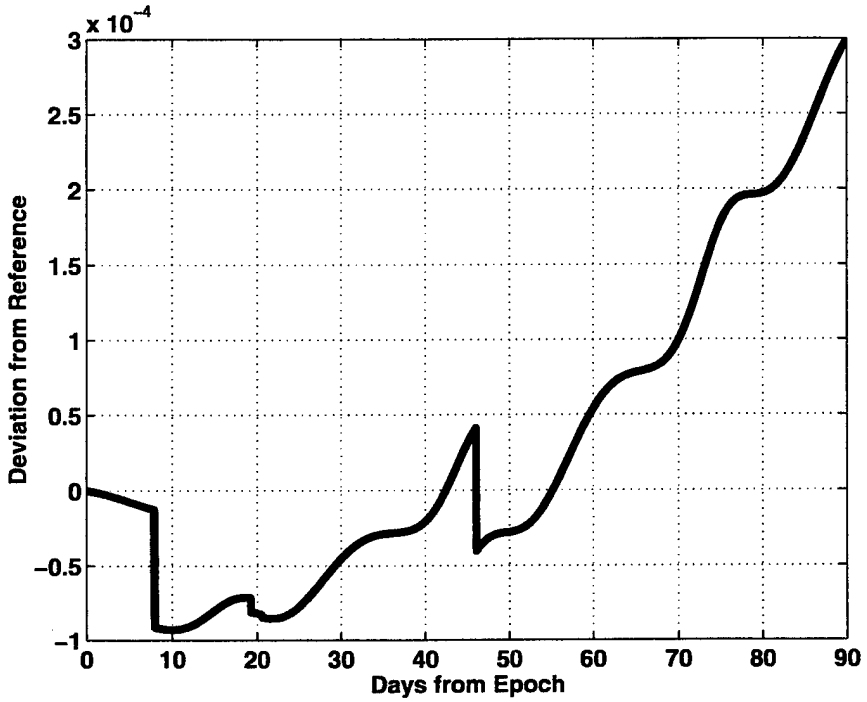


Figure 5-8 Global Case 1 Controlled Eccentricity Deviation (Limit = 0.0003)

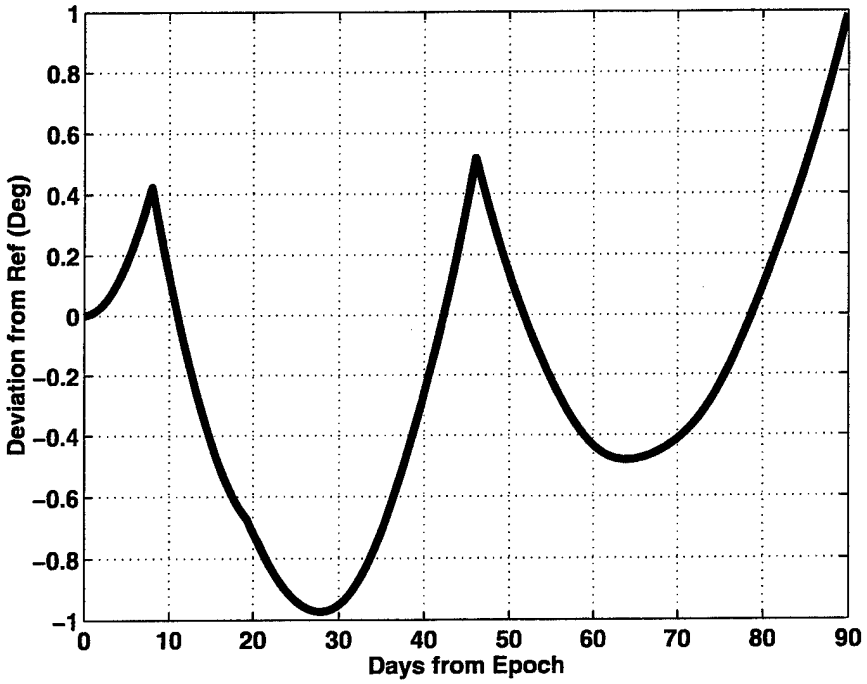


Figure 5-9 Global Case 1 Controlled Mean Anomaly Deviation (Limit = 1°)

5-3-2-2-2 Case 2 Results—Epoch Elements at Extreme Limits

With all of the epoch elements at the allowable limits, a greater number of burns appears to be required to maintain the case two uncontrolled trajectories (see Appendix C-4-2) within the desired tolerances. Application of the genetic algorithm optimization software resulted in the following 12-burn solution (see Table 5-7) which maintained the trajectory within the desired tolerances for the desired 90-days.

Table 5-7 Global Case 2 Burn Times and Components

Burn Time (Since Epoch)	Mean Anomaly at Burn Time	Radial (m/s)	Along-Track (m/s)	Cross-Track (m/s)	Magnitude (m/s)
00d 07h 09m 38.70s	147.97	0.5174	0.6092	2.5290	2.6523
00d 13h 52m 55.81s	241.62	0.2854	0.0836	-0.0642	0.3042
01d 20h22m 14.32s	332.49	-0.0108	-0.1526	-0.5102	0.5326
07d 23h 40m 32.91s	166.92	0.2504	0.1357	0.2054	0.3511
11d 19h 36m 18.69s	136.55	-0.0136	0.1626	2.7394	2.7443
12d 00h 11m 27.14s	331.70	-0.1610	-0.2072	-0.0861	0.2762
15d 11h 59m 23.28s	037.62	-0.0131	0.0017	0.0044	0.0139
18d 12h 00m 21.34s	117.07	0.1100	0.2494	2.5528	2.5673
44d 12h 49m 08.31s	162.84	0.0234	0.0534	0.0030	0.0584
55d 04h 31m 06.52s	161.21	0.2431	0.1472	0.0505	0.2886
77d 02h 53m 35.52s	169.44	0.0134	0.0196	0.0006	0.0238
85d 22h 01m 08.98s	170.93	0.0002	0.1773	-0.0011	0.1773
				Total DeltaV	**9.9900**

The most noticeable characteristic of this result is its complexity. Unlike case one, where the solution could be verified for near-optimality, it is difficult to determine whether or not this result is an optimal one. The burns are occurring at random locations and in all three possible directions. There are, however, some features of this solution that warrant discussion.

The first of these is that all of the elements are maintained within the desired constraint box. As this case was purposely designed to be the most difficult case imaginable, the fact that the genetic algorithm determined a solution which met all the constraints is, of itself, a significant feature of this solution.

Second, three of the six element trajectories end on the edge of the constraint box (see Figure 5-10 and Figure 5-11). As discussed with case one, this shows that no extra fuel is being expended, at least for the given times and positions of the twelve burns. Therefore, although optimality cannot be proven, this shows that at least to some degree this solution is optimal.

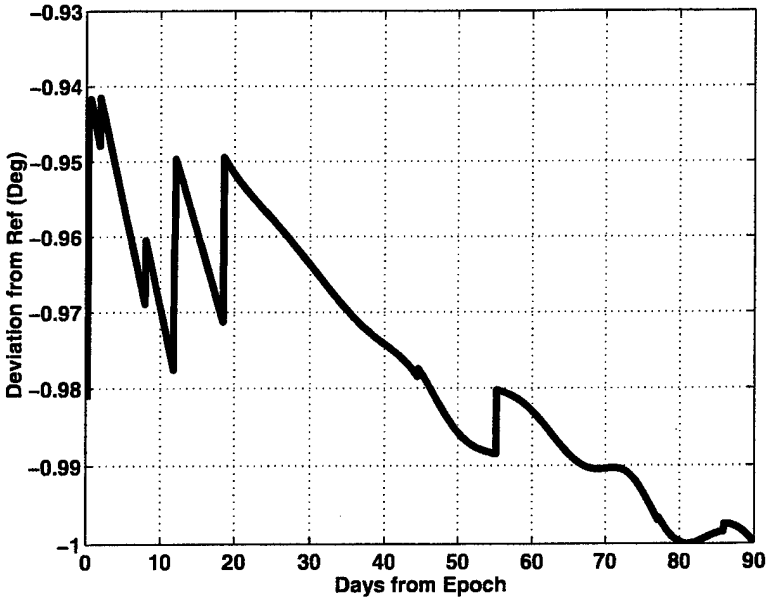


Figure 5-10 Global Case 2 Controlled Argument of Perigee Deviation (Limit = 1°)

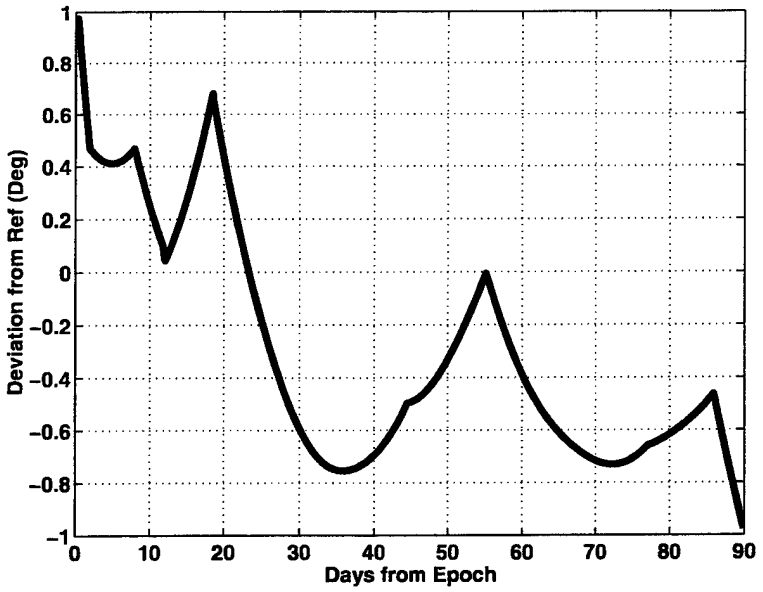


Figure 5-11 Global Case 2 Controlled Mean Anomaly Deviation (Limit = 1°)

One especially noticeable feature of this result is the large amount of ΔV that is required in the out of plane or cross track direction. Returning to Table 5-7, it can be seen that three of the burns contain out of plane components greater than two m/s,

whereas none of the other components of any of the burns exceed one m/s. The large variation of these components from the others calls into question the optimality of the solution.

The three burns in question occur on day 0, 11, and 18. Figure 5-10 shows that there are significant changes in the argument of perigee history at these three times. It can also be seen in that there are significant inclination changes that occur at these three times as well (see Figure 5-12).

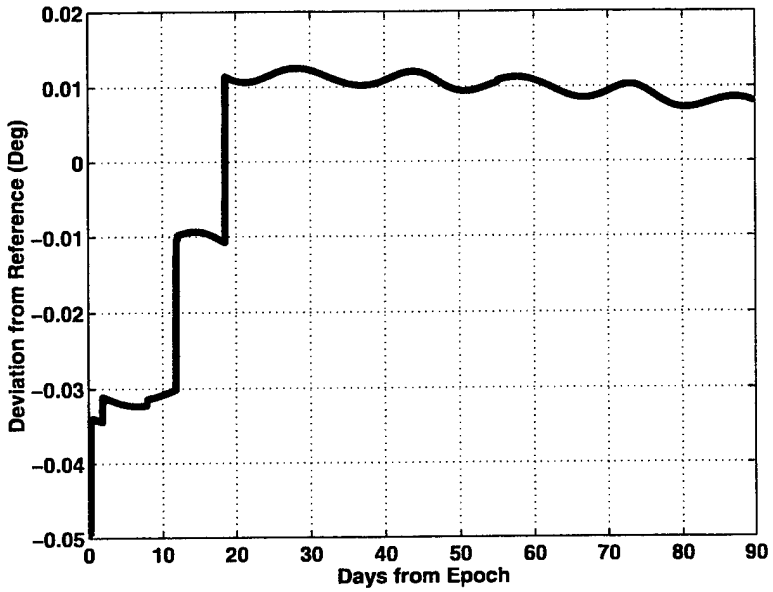


Figure 5-12 Global Case 2 Controlled Inclination Deviation (Limit = 0.05°)

Since the argument of perigee ends at its tolerance limit, the initial assumption is that these large out of plane burns are required to keep the argument of perigee from violating. However, the inclination history plot shows that these three components are larger than needed to control the inclination. The inclination ends far away from its tolerance limit of 0.05°. Since it is also possible to control the argument of perigee using radial or along track burns (which would not affect the inclination), it seems that it would

be more optimal to control the system in a different manner. By using smaller out of plane burns, the inclination could be controlled closer to its limit and the overall ΔV could even be lowered.

Unfortunately, although it would save fuel if the orbit could be controlled in this more logical manner, the constraints imposed on the system limit the possibilities. To better understand the presence of the large out of plane components in the case two solution, a simple test case was created and run for two different scenarios: with only out of plane burns allowed, and with no out of plane burns allowed[84]. For these two scenarios the same initial conditions were used as for case two, but rather than constraining all six of the orbital elements, only the argument of perigee was constrained. Additionally, rather than attempting to control the argument of perigee for all 90 days, only 20 day runs were performed. The results of this case make it painfully obvious why the large out of plane components are necessary.

When attempting to control the argument of perigee using only out of plane components, a two-burn solution with a ΔV of 6.49 m/s was required. On the other hand, controlling the argument of perigee with only radial and along-track burns led to a required ΔV of only 1.97 m/s. This is a nearly 70 percent reduction in required ΔV and seems to support the conclusion that a significant fuel savings could be achieved in the case two solution by reducing the out of plane component magnitudes.

However, when all six elements are constrained as was done in case two, it is impossible to control argument of perigee through radial and along-track burns while still

---

[84] The author would like to acknowledge Tim Brand of The Charles Stark Draper Laboratory for his recommendations in this section.

meeting the other constraints. This is especially true when the semi-major axis and mean anomaly are constrained.

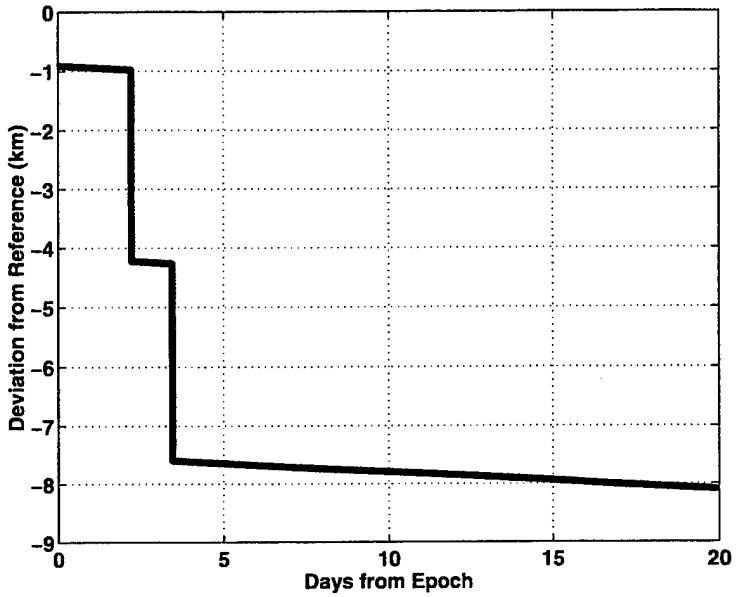


Figure 5-13 Semi-Major Axis History for Controlling Argument of Perigee with Radial and Tangential Burns

Figure 5-13 shows the semi-major axis deviation history for the simple case above in which the argument of perigee was controlled entirely by radial and along-track burns. Both of the required burns can be easily seen on this plot. In order that the argument of perigee be changed enough to avoid violation, it is necessary to change the semi-major axis by more than three kilometers. With only a two kilometer range to work in (from –1 to +1 km), it is clear that using this method, it will be impossible to control both the argument of perigee and semi-major axis simultaneously. The only solution then is some combination of burns which relies heavily on the out of plane components which fail to change the semi-major axis significantly, but greatly affect the argument of perigee. This

is exactly what is seen in the case two solution and therefore, despite its non-intuitive nature, the solution seems to be more optimal than originally thought.

5-3-3 Global Station Keeping Comparison to ASKS Results

In previous work, Shah[85], along with Proulx, Kantsiper, Cefola, and Draim[86], developed a station-keeping method that employed primer vector techniques in conjunction with Lambert's theorem to calculate the optimal burns required to maintain a constellation, given a set of orbit tolerances of the type defined in Table 5-3. Like the current study, the primer vector technique utilized precise orbit propagation tools and was tested on the Ellipso™ constellation. These similarities allowed for somewhat direct comparisons between the two methods.

However, unlike the current study, the primer vector method was very dependent upon a pre-defined targeting scheme. Specifically, the targeting schemes developed by Shah were tailored to an Ellipso™ 8:1 orbit design (i.e. future target locations were determined based on expected behavior of the 8:1 orbit). The current study utilizes the more recent Ellipso™ 113:14 design that contains variations in the orbital elements of the satellites in the Borealis orbit plane. As a result of the resonance change between the two orbit designs, the secular decay of the orbit also changed. Therefore, the targeting schemes for the Automated Station-Keeping Simulator (ASKS) developed by Shah which were developed under the expectation of certain values of secular drift were not as accurate when they were applied to the current case as they were when applied to the 8:1

---

[85] Shah, Naresh, H. *Automated Station-Keeping for Satellite Constellations,* CSDL-T-1288, Master of Science Thesis, Massachusetts Institute of Technology, June 1997.

constellation design. However, in an effort to show the advantage the genetic algorithm method possesses in not requiring a pre-defined targeting scheme, the two cases studied previously were solved using the ASKS software and a variety of available targeting schemes. A comparison between the best ASKS results and the best genetic algorithm (GA) results can be found in Table 5-8.

Table 5-8 Global Station Keeping GA and ASKS Results Comparison

Case	ASKS Burns	ASKS Delta V (m/s)	GA Burns	GA Delta V (m/s)	% Reduction
Case 1	20	2.3941	4	0.5693	76.22
Case 2	22	78.721	12	9.9900	87.31

As shown in Table 5-8, the genetic algorithm method was much more successful at calculating minimum fuel station-keeping trajectories than the ASKS approach. This is a direct result of the lack of a correctly defined targeting scheme for the 113:14 constellation design in the ASKS package. It is expected that if a correct targeting scheme for 113:14 design could be defined, then both the GA and ASKS methods would generate similar results. However, the process of defining the appropriate targeting scheme is difficult, and as can be seen, the result of that effort is a very specialized result. The advantage to the genetic algorithm approach lies in its generalized view of the problem. There is no need for a predefined targeting scheme. This allows for any satellite to be studied with minimal change in the implementation of the approach.

Besides the obvious advantage which comes from the lack of a pre-defined targeting scheme in the genetic algorithm approach, an additional advantage of this

---

[86] Shah, N., R. J. Proulx, B. Kantsiper, P. J. Cefola, and J. E. Draim. *Automated Station-Keeping for Satellite Constellations,* Paper #C-7, International Workshop on Mission Design and Implementation of Satellite Constellations, Toulouse, France, 17-19 November 1997.

method over the ASKS method arises from the fact that the converged solution always maintains the fully perturbed trajectory entirely inside the constraint box. This is not true of the ASKS approach. Depending upon the targeting scheme chosen, a number of the solutions generated by this method were unable to maintain all six elements inside the desired tolerance limits over the entire 90-day period.

5-3-4 Limitations of the Global Station Keeping Approach

Despite the success with which the genetic algorithm method is able to find near-optimal station-keeping strategies, there are a number of difficulties with this technique that should be noted.

5-3-4-1 Computational Limitations

The first of these limitations is not necessarily a problem with the methodology, but rather an issue with the current state of computational capabilities. The cases for this study were run on a high-end graphics workstation—a Silicon Graphics Origin server with eight 195 MHz processors and 2 GB of RAM. On average six processors were used in parallel, but the time required to arrive at a solution was still between 12 and 24 hours. This time can mostly be attributed to the inclusion of very accurate orbit propagation tools, but is also due to the large number of iterations (an average of about 10,000) that were necessary for the genetic algorithm to converge to a solution. An increase in computing power could help to eliminate this limitation, but as of 1999, computational time issues are a main limitation to the use of this approach in an operational manner.

5-3-4-2 Limitation of Ending at Near-Violations

A second limitation to using this technique in any type of operational manner has been touched on previously in analyzing the results of the two cases and is also an indirect result of the computational limitations. As discussed in some detail previously, the solutions generated by the genetic algorithm to the station-keeping problem all exhibited the behavior of causing at least one of the element histories to end at the edge of its tolerance box. This is not a problem if the time period for which the optimal burns are being calculated is the entire time period of interest.

However, as is the case with the Ellipso^{TM} system, the operational lifetime is longer than the three-month cases run here. When this occurs the only way to handle the next time period is to start optimizing where the previous solution left off. For example, if days 90 to 180 were to be run for Case 1 of this study, the starting elements would have to be the same as the ending elements of the first 90 days and therefore, both eccentricity and mean anomaly would immediately reach a point of violation. Although this technique should be able to optimize the resulting 90-day trajectory, continually starting at a point of violation does not make sense from an operational standpoint.

Additionally, continually starting at a point of violation fails to overcome the greediness of previous techniques. The algorithm is finding the optimal way to handle the first 90 days without taking into account the effect that the first 90-day solution might have on the next 90 days. In fact, the combined 0 to 180-day delta-v solution that would be generated through separate runs of this algorithm, would be more than the delta-v determined by one 180-day run which eliminated the intermediate stopping point. Originally, the intent of this study was to apply the genetic algorithm technique to the

entire lifetime of the satellites of interest, but once again the computational capabilities available at the current time proved to be a limiting factor.

5-3-4-3 Non-repeatable Limitation

An additional limitation to any type of operational implementation of this approach stems from the non-repeatable nature of this method. Due to the random nature of genetic algorithms and the jagged nature of the solution space, it is extremely difficult to obtain the same solution to an identical problem two or more times in a row. The most common behavior observed was that multiple runs of the same problem provided similar values for the objective function, but very different values for the variables (i.e. burns) necessary to arrive at these objective function values. For example, a second and third run of Case 1 provided objective function values (i.e. delta-v values) of 0.5630 m/s and 0.5632 m/s. These values are comparable to the original value of 0.5693 m/s arrived at on the first run. However, the burn strategies that allowed for arrival at these delta-v values for the second and third runs of Case 1 are much different than the burn strategy originally obtained and presented in Table 6.

5-3-4-4 Dependence on Propagation Techniques

A final operational limitation of this technique lies in the dependence of the overall solution on the accuracy of the propagation tools. Even if the computational limitations could be overcome and the optimal trajectory could be calculated for the entire lifetime of a given satellite, the optimal trajectory would only be valid as long as the true trajectory of the satellite corresponded exactly to the trajectory predicted by the model. If, due to atmospheric or other modeling error, the true flight of the satellite

differed even slightly from that predicted by the DSST code, the entire solution from that point on would no longer be valid.

5-4 Operational Station Keeping Approach

Although the global station keeping method described up to this point in this chapter was successful in generating near-optimal station-keeping strategies, the limitations discussed in section 5-3-4 limit the actual usefulness of this method. It has been found that the global approach could be used successfully to create baseline station-keeping fuel budget estimates, but it could not be used in any sort of operational station-keeping algorithm.

In an effort to overcome the operational limitations of the previous global optimization approach, an attempt was made to develop a method which would be faster, more repeatable, and allow for easy implementation of the solution into an operational station-keeping scheme. This method builds upon many of the concepts developed for the global station keeping approach, but rather than focusing only on maintaining the orbit within the desired state constraints, this method focused on ways to maintain near-optimality in the station keeping maneuvers, while also maintaining operational characteristics.

The following sections discuss the modifications that were made to the global station keeping approach in order to make it more operational. The concepts behind these modifications are first discussed. This discussion is followed by specific details regarding the modifications to the code and implementation. Finally, a number of cases are presented and analyzed.

5-4-1 Definition of Operational Features

Although the objective function for this operational approach is the same as the objective function for the global station-keeping approach discussed in the previous section (i.e. minimize the required fuel), there are some additional objectives which will be used to evaluate the solution and the solution process. These are the objectives of repeatability, speed of convergence and implementation ease and are used to define the "operationality" of a given process. As it is through these parameters that this approach will be compared to the more global approach, they are each defined briefly in this section.

Repeatability: For station-keeping processes that could be used in actual satellite operations schemes, it is desirable that a given solution process generate identical or nearly identical solutions to the same station-keeping problem. The global approach discussed earlier in this chapter did not meet this requirement. If the global approach were exercised twice on the same problem, the resulting burn sequence was found to be quite varying. Although the overall ΔV estimates were of the same magnitude, the times and magnitudes of the burns that made up these ΔV estimates varied widely. For an approach to be considered operational, it is highly desirable that the same results can be generated on a repeatable basis.

Speed of Convergence: As a satellite progresses along its orbit, it experiences many perturbations. The effects of some of these perturbations are entirely predictable, while the effects of others, such as drag, continually vary and are therefore impossible to

predict. In order to be able to maintain the satellite's orbit despite the changes caused by these unpredictable perturbations, it is necessary that the station-keeping strategy can be developed in an operational (e.g. daily planning) mode. Methods for generating these station-keeping strategies that require an extreme number of iterations and long convergence times are clearly not operationally feasible.

Ease of Implementation: This objective refers not to how easily the solution process can be implemented and station-keeping strategies generated, but rather to how easily the solution itself can be applied to satellite operations. The solutions from the global approach clearly did not lend themselves to easy implementation. They relied heavily upon the accuracy of the orbit propagation software. Any deviation of the satellite from the expected position renders the entire burn list (from that time on) useless. A more operational station keeping approach should remove this dependency on the accuracy of the orbit propagator, thereby providing easier implementation of the resulting burns

5-4-2 Operational Approach Overview

As the basic objective of this operational approach and the previously discussed global approach are the same (to minimize required fuel), the basic features of the solution process did not change. The same objective function, complete with useful modifications and the same variable structure were utilized. However, in order to make the approach more operational, the ways in which these basic components were implemented had to be modified. This section looks at the modifications that were made and the motivation behind these modifications.

One of the primary objectives of this approach was to reduce the time required for the genetic algorithm to converge to an optimal station keeping strategy. Through experimentation, it was found that the convergence time was tied directly to two things: the length of time of the propagation and the number of variables to be optimized. It was found that by reducing one or both of these parameters, the convergence time could also be reduced.

However, given the global nature of the previous approach, neither the number of allowable burns nor the length of time over which to solve can be easily reduced. In fact, in order to apply the method to the entire lifetime of a satellite, it is actually desirable that both of these parameters be increased. This is a clear contradiction to the operational desire of fast convergence and therefore it is necessary to create a new way of looking at the problem before faster convergence can be achieved.

To define a new way of looking at the problem, it was necessary to relinquish some of the global nature of the problem and return to a somewhat greedy strategy. The greedy operational strategy can be summarized as follows. The satellite flies along unhindered until a constraint violation is predicted to occur. In the region of this predicted deviation, an optimal burn strategy is developed. By this it is meant that the burns which make up this optimal burn strategy are only allowed to occur during a short time period on either side of the expected constraint violation. Also, rather than meeting the objective of maintaining the satellite inside the constraint box for some specified period of time, the optimal burn strategy is developed to meet the objective of allowing the satellite to drift for maximum time without violation of the constraints. The resulting

burn strategy can then be implemented and the satellite can be left alone until another constraint violation is predicted.

This approach has many operational advantages. First it allows for a decrease in the number of variables and in the propagation time. Since the burns must be contained within a short time period in the region of the constraint violation, the required number of burns is small. Additionally, since burns are only allowed in the region of the deviation, it is only possible to prevent future constraint violations for fairly short periods of time. Both of these factors lead to shorter convergence times.

This approach also has the advantage of being more easily implemented. The burns are all performed within a short period of a deviation that is expected to occur quite soon. After that short period has passed, no more burns are performed until another deviation is expected, at which time the optimization is repeated. This has a distinct advantage in terms of implementation, as the reliance on the accuracy of the orbit propagator has been reduced. Notice that all burns occur within the neighborhood of the current time. It is much easier to model with accuracy the current environment than it is to predict future environmental changes. Also, should some unexpected error enter the system after the burns have been performed, this perturbation does not have any effect on future burns, as it would have previously. Instead, once the burns have been performed, the satellite is left to drift until a future deviation is expected. If this deviation occurs before the propagator predicts it will occur, this has no effect on the burn strategy. The optimization can simply be executed at this time and the process repeated. Table 5-9 details the differences between the global and operational approaches and section 5-4-3 discusses the implementation of the operational approach.

Table 5-9 Global vs. Operational Optimization Approach Comparison

Parameter	Global Optimization Approach	Operational Approach
Burn Number	Enough to Handle Multiple Constraint Violations Over Lengthy Propagation Time	Enough to Handle One Violation in Constrained Time Period
Burn Times	Occur Anywhere in Time Period of Interest	Occur in Short Time Period on Either Side of Violation
Propagation Time	Predetermined Lengthy Period of Time	Algorithm Determined Maximum Drift Time
Follow On Burns	Very Reliant on Predicted Satellite State	Independent of Current Predictions--Determined by Future Optimizations.

5-4-3 Implementation of Operational Approach

Although the basic structure of this more operational optimization approach was quite similar to the global station-keeping approach, some changes were still required in order to implement this technique. These changes are outlined briefly in the following sections.

5-4-3-1 Operational Approach Variable Description

The variable structure outlined in Table 5-1, along with the modifications outlined in section 5-3-1-4-2, was essentially adequate for this problem as well. The desired solve for variables are still the time of each burn, the magnitude of each burn component and an on/off flag for each burn. The only modification required was a change in the allowable times for each burn. Rather than being allowed to vary anywhere between zero and the maximum time, the burn times were now constrained to lie within a certain predefined range of the expected deviation time (see section 5-3-1-5-2). It should also be noted that scaling all variables between 0 and 2 as described in section 5-3-1-4-2 was found to be effective in this problem, as well.

Table 5-10 GA Operational Station Keeping Solution Process Variable Allocation

Variable	Number /Burn	Pre-Scaling Lower Limit	Pre-Scaling Upper Limit	Units
Elapsed Time from Epoch	1	Time Of Expected Violation Minus Burn Window	Time of Expected Violation Plus Burn Window	Seconds
Magnitude of Burn Components	3	- Maximum Component Magnitude	+ Maximum Component Magnitude	m/s
On/Off Flag	1	0.00	1.00	N/A

5-4-3-2 Operational Approach Objective Function

The largest modifications to the implementation were required in the modification of the objective function. Due to the new approach of attempting to find the optimal burns that allowed for maximum drift time between burns, it was necessary to modify the objective function to include some information about the drift time in order that it might be maximized.

Initially, attempts were made to create this modification by simply adjoining an additional term to the modified objective function of Equation 5-6 as follows:

$$J = \Delta V + \text{Deviation Contribution} + \text{Time Contribution} - \text{Drift Time} \qquad \text{Equation 5-10}$$

Upon first glance, it appears that this is a sufficient objective function. Since the other three terms are being minimized, the negative of the drift time has simply been added onto the objective function. Therefore, larger values of the drift time should lead to smaller objective function values.

However, there are problems with this formulation. Notice that when optimizing J, the goal is to minimize ΔV. In order for this to happen effectively, the other terms in

the objective function must go to zero. If they do not go to zero, the problem then depends heavily on the weights that are associated to each term. By placing weights that are too large on the deviation or time contributions, one can essentially eliminate all minimization of fuel. Solutions that provide non-violating trajectories will be generated, but they will not be fuel optimal.

In the global approach, reducing the objective function to a complete fuel minimization function was not a problem. Since the deviation contribution is simply a function of the total deviation from the constraints, as soon as a solution is found which does not violate the constraints, the deviation contribution goes to zero and is effectively removed from the objective function. Also, recall that the time contribution is simply a measure of the difference between a fixed end time and the time until the first deviation. For the global case the end time was known and so once again, solutions which created trajectories that were entirely contained with in the constraint box forced this term to zero. As the global case progresses, strings that do not cause violations of the constraints survive, and the objective function reduced entirely to $J = \Delta V$ which allows for minimization of the ΔV.

However, if the same objective function is used to solve the operational approach, it is impossible to reduce the objective function to the simple form $J = \Delta V$. First, since the end time is no longer known (it is equal to the maximum drift time which is being solved for), it is impossible to force the difference between the end time and the time of the first deviation to zero. It can, however, be argued that having both the maximum drift time and the time contribution both in the objective function is an unneeded redundancy. The purpose of both is to favor solutions that produce trajectories that avoid violation for

long periods of time. However, even removing the time contribution from the objective function above does not eliminate the problem. Since the maximum drift time will always be a real value, it is still impossible to reduce the objective function to $J = \Delta V$ in order to minimize the required fuel, as desired. Even modifying this term to be the reciprocal of the drift time (i.e. –1/drift time) does not allow for complete minimization of the ΔV. The objective function will still rely heavily upon randomly chosen weighting factors.

To eliminate this difficulty, a two-step solution process was used. The first step of the solution process was used to maximize the drift time. For this portion of the optimization the objective function was simply J = Drift Time. The genetic algorithm, with the same string structure described in section 5-4-3-1 of burn times, magnitudes and on/off flags was then used to find a string of burns that allowed for maximum drift time before another series of burns was required. Once this portion of the optimization had completed, the resulting maximum drift time was saved as the end time. A second optimization was then accomplished, using the techniques from the previous global approach, along with the burn time constraints, to minimize the required ΔV over the previously solved for fixed period of time. This process is summarized in Figure 5-14. Note that even though the global genetic algorithm approach is used to perform a portion of this optimization, because the number of burns and total propagation time has been reduced, the time to perform the optimization has also been reduced.

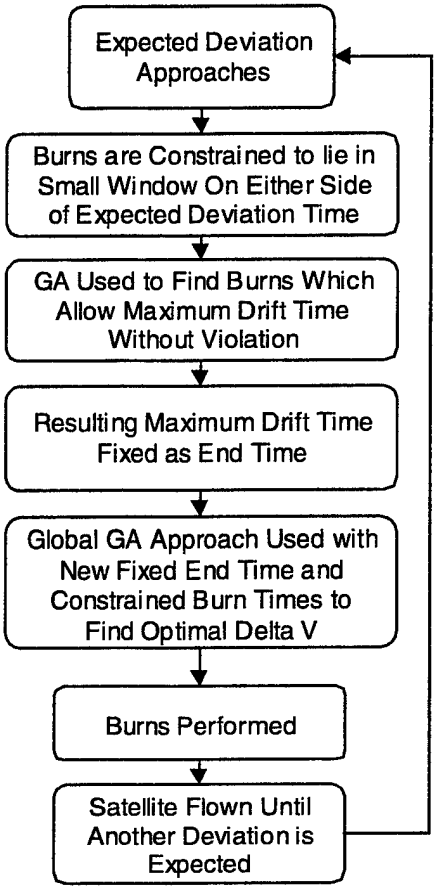


Figure 5-14 Operational GA Optimization Approach Overview

5-4-3-3 Operational Approach Genetic Algorithm Parameters

Since the genetic algorithm is used twice in the operational station-keeping approach, it is necessary to tune the genetic algorithm with two different sets of genetic algorithm parameters. For the most part, the parameters used in the global approach and presented in Table 5-2 were also good values for this optimization approach. However, it was necessary to modify some of the parameters. Those which worked best for the maximum time optimization loop are presented below in Table 5-11, while those that worked best for the ΔV portion of the optimization can be found in Table 5-12.

Table 5-11 Operational Station Keeping Approach Maximum Drift Time Optimization Genetic Algorithm Parameters

Parameter	Type/Value
Stopping Rule	No Change
No Change Value	100
Population Size	100
Replaced Per Iteration	25
No Duplicates Flag	True
Mutation and Crossover Flag	True
Mutation Type	Gaussian
Real Mutation Constant	0.2
Mutation Rate	1/String Length
Crossover Type	Two-Point
Crossover Probability	0.85
Selection Type	Tournament

Table 5-12 Operational Station Keeping Approach ΔV Optimization Genetic Algorithm Parameters

Parameter	Type/Value
Stopping Rule	No Change
No Change Value	2000
Population Size	100
Replaced Per Iteration	25
No Duplicates Flag	True
Mutation and Crossover Flag	True
Mutation Type	Gaussian
Real Mutation Constant	0.5
Mutation Rate	1/String Length
Crossover Type	Two-Point
Crossover Probability	0.85
Selection Type	Tournament

The main difference between the parameters used for the two different portions of the optimization can be found in the no change value required before the genetic algorithm is allowed to stop. For the maximum time portion of the optimization, only 100 iterations with the same value were required before stopping as opposed to 2000 required for the ΔV optimization. There are two reasons for this difference. First, the

genetic algorithm seemed to converge to the maximum drift time much faster than it could converge to the optimal ΔV and it usually did so without settling on intermediary solutions. Therefore, any repetition of the same solution was a clear signal that a near optimal had been reached.

The second reason stems from the difference in the required accuracy of the solution from each portion of the optimization. There is a clear desire to have the optimal ΔV be as accurate as possible. Therefore, the genetic algorithm is forced to continue, even after converging to a solution, to ensure that no future decrease can be obtained. On the other hand, there is no clear need for an extremely accurate drift time. Even if the maximum drift time is not found, the genetic algorithm will still produce an optimal burn strategy for whatever time is determined. This burn strategy can then be implemented and satellite operations continued, regardless of any error in the maximum drift time.

5-4-3-4 Operational Approach Computer Implementation

The implementation of the operational approach required very little modification from the previous software developed for the global approach. In fact, the code structure presented in Figure 5-3 and used for the global station keeping case is identical to the code structure used for this operation approach. However, although the structure of the code is the same, two modifications were required to the details of both PGA_GASK and FUNC.

The change to both of these routines involved the creation of a flag that was used to signal which portion of the optimization the genetic algorithm was in at any given time. If the flag indicated that the drift time maximization was being performed, then

FUNC calculated the maximum drift time for a given string and returned it to the genetic algorithm as the objective function value. Otherwise, FUNC again performed the steps illustrated in Figure 5-4. This two-layered FUNC approach to the optimization did cause some difficulty with the genetic algorithm, but this difficulty was overcome by simply killing the entire population at the end of the time maximization and restarting the genetic algorithm with a fresh population for the ΔV minimization.

5-4-4 Operational Approach Feasibility Test

In order to test the feasibility and robustness of this operational approach, attempts were made to control Ellipso Borealis™ node at noon satellite given the same epoch time and elements as were used in the case two optimization for the global operational method. These elements along with the corresponding uncontrolled deviations can be found in Table 5-5. These elements were designed to represent the worst possible state of the satellite with all elements at or near their limits and with rates which will soon cause each element to violate. The global approach was successful in controlling the satellite even with these extreme starting conditions, but the solution was composed of 12 burns spread out over the 90 day period (see Table 5-7). For this case, only five burns were allowed and they were constrained to occur within a 48-hour period (24 hours on either side of the first expected deviation).

5-4-4-1 Feasibility Test Results

Using the localized optimization approach, the genetic algorithm was first used to calculate burns that allowed for maximum drift between the last burn and the time of the next deviation. This step of the optimization was surprisingly fast requiring only 330

iterations. The result was a total time of 38 days with a 3-burn solution comprised of burns totaling 15.12 m/s in ΔV. Clearly, stopping the optimization at this point would be a serious error. The previous application of the global approach to this problem resulted in a solution of 9.99 m/s for a 90-day period. This approach exceeded that amount over only a 38-day interval.

In order to arrive at a more reasonable solution, the second portion of the operational approach was implemented. The genetic algorithm was again used in conjunction with DSST to find ΔV solutions. However, rather than solving for maximum drift time, the time was set to the 38 day limit solved for previously and the optimization simply attempted to find minimum fuel solutions which avoided violation of the constraints. This portion of the optimization required a slightly greater number of iterations (2670), but the resulting ΔV of 6.26 m/s was less than half that originally estimated during the first half of the optimization. The four-burn solution that resulted from the optimization is presented below in Table 5-13.

Table 5-13 Operational Approach Feasibility Test Case Burn Times and Components

Burn Time (Since Epoch)	Mean Anomaly at Burn Time	Radial (m/s)	Along-Track (m/s)	Cross-Track (m/s)	Magnitude (m/s)
6d 37m 55.03s	83.94°	0.0136	-0.0351	3.2910	3.2912
6d 46m 01.15s	100.30°	0.2899	0.3992	2.6092	2.6554
9d 00m 41.36s	12.00°	-0.2058	0.0046	0.0086	0.2060
15d 58m 53.60s	135.84°	0.0194	0.0958	0.0526	0.1110
				Total DeltaV	**6.2636**

Both the mean anomaly and argument of perigee deviation history plots are presented below in Figure 5-15 and Figure 5-16. These two deviation histories are presented as these two elements had the most significant impact on the solution. They clearly show that the time of 38 days arrived at by the first pass through the optimization

is optimal. It is impossible for the mean anomaly to drift any longer without violation of the constraints. It is possible to allow the argument of perigee to drift longer than this 38-day limit. However, doing so would require more fuel. Since the entire system is constrained by the mean anomaly time limit, more motion in the argument of perigee would be wasteful. The optimal solution is then to burn enough in the argument of perigee to drift right to the limit at the end of the 38 days as shown in Figure 5-16. The element deviation history plots of all six elements can be found in appendix D-2.

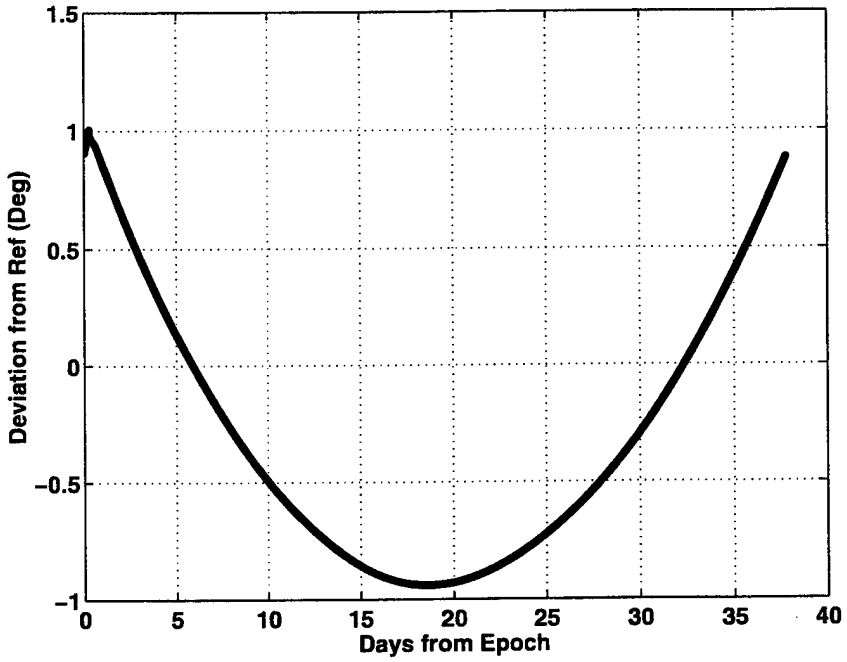


Figure 5-15 Operational Feasibility Test Controlled Mean Anomaly Deviation (Limit = 1°)

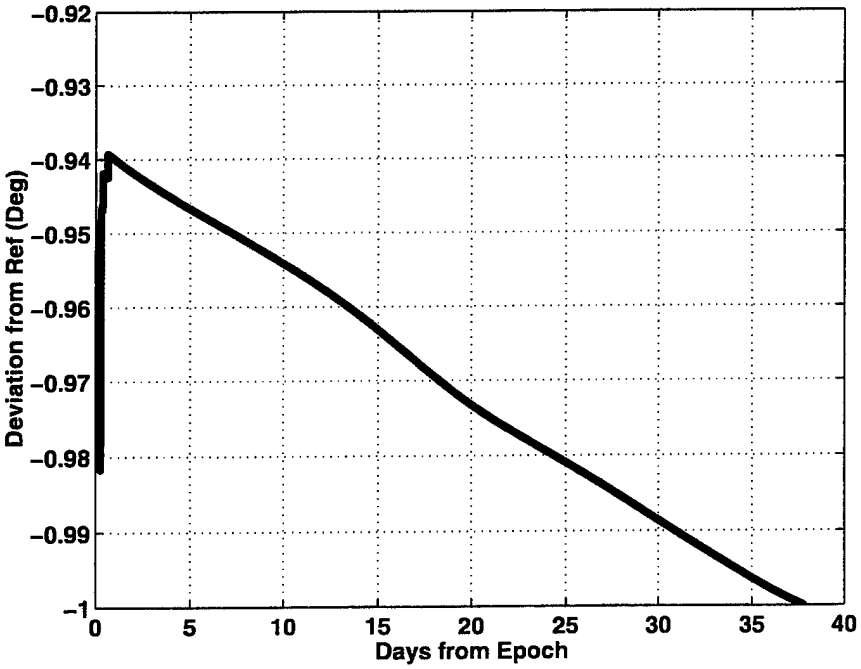


Figure 5-16 Operational Feasibility Test Controlled Argument of Perigee Deviation (Limit = 1°)

5-4-4-2 Feasibility Test Results Evaluation

In order to qualify the results presented in the previous section (5-4-4-1) as a successful operational approach, it is necessary to evaluate the solution and the solution process in terms of the pre-defined operational characteristics of speed of convergence, ease of implementation and repeatability. Specifically, by comparing the previous global solution with the solution resulting from the operational approach, it can be determined whether or not any gains in the operational feasibility of the genetic algorithm approach have been made.

The difference between both approaches in terms of speed of convergence is quite distinguishing. Although both optimizations were accomplished using an SGI workstation with five processors, the operational approach was able to converge in a total

number of iterations less than 3000 which required less than 2 hours of computer time. The global approach, on the other hand, required over 10,000 iterations and 14 hours of computer time. For actual implementation into an operational scheme, the two hours that are required by this new approach is definitely a more reasonable value.

The steps required for implementation of the solution resulting from this new approach into an operational scheme are more reasonable, as well. As discussed previously, in order to implement the 12-burn global approach solution, it is necessary that the satellite maintain a trajectory that exactly matches the trajectory predicted by the DSST software. Any variation, no matter how slight, will invalidate the solution and require the entire optimization be re-accomplished with an updated state as an input.

The solution from the new approach, on the other hand, has only a very slight dependence on the accuracy of the DSST prediction. The predicted trajectory must be maintained only long enough that all of the burns can be performed. Ideally, the trajectory will be maintained after the burns, as well, and the entire predicted 38-day drift can be accomplished without violation. However, a deviation from the DSST predicted trajectory does not invalidate the solution. Rather, the satellite can be allowed to drift on any actual trajectory until a future violation approaches. If the DSST model is exactly right, this violation will occur on day 38. If not, the only change to the process is that the next iteration must be accomplished sooner. The solution arrived at from the previous optimization is unchanged and can be fully implemented.

Finally, in order to qualify as a more operational approach than the global method, the new optimization scheme must be more repeatable than the global approach. This is one area in which the difference between the approaches is not entirely clear.

Both methods can be run repeatedly and will generate similar ΔV values for each successive optimization. However, neither method will generate identical burn lists on two successive runs of the software. Due to the fact that the solution of the operational approach is dependent upon fewer burns than the global approach, the resulting burn lists from successive optimizations via the operational approach are much more similar than burn lists resulting from successive optimizations via the global approach. Due to the fact that burns cannot occur after the first few days in the operational approach, the trajectories resulting from successive runs of the operational approach are much more similar than successive global approach solution trajectories.

5-4-5 Operational Approach as a Planning Tool

Although the approach presented in this section was developed with an operational intent, it can also serve well as a planning tool. If used in an operational context, the method would be used to find the optimal burn strategies to avoid violation (such as that shown in Table 5-13), each time that a violation of the constraints is suspected. Note, however, that the end result of the optimization is always an end state that is nearing a violation. Therefore, by simply looping through the process, with the end state of one iteration becoming the epoch state of the next, it is possible to use this approach to obtain operational estimates of the required ΔV over any length of time.

5-4-5-1 Operational Approach Planning Test

Using the operational approach as a planning tool, it was desirable to estimate the ΔV required to maintain an Ellipso™ Borealis Node at Noon satellite for the entire lifetime of five years. The following sections describe the steps necessary to use the operational approach as a planning tool to accomplish this goal.

5-4-5-1-1 Planning Test Epoch Elements Definition

In order to estimate the required ΔV for the Borealis satellite in question, it was first necessary to use the tools from Chapter 4 to define the optimal epoch elements for this case. For comparison sake with the global approach and other cases presented in this thesis, an epoch date of March 21, 2000 was chosen. However, even though the global cases are also a Borealis Node at Noon satellite with this identical epoch date, the elements used for the reference orbit in those cases (presented in Table 5-3) were optimized for the 113:14 repeat ground track behavior over only a 90-day period. Using these elements as epoch elements for a five-year period will not yield optimal behavior in the reference orbit. Instead, the 113:14 repeat ground track optimization was again performed using March 21, 2000 as the epoch state and the deviations from the desired repeat ground track behavior were minimized over a five year period. The result was a set of elements which gave minimal deviation from the 113:14 repeat ground track behavior over the entire five-year period starting in March 2000. These elements (seen in the J2000 True of Date reference frame in Table 5-14) were then used as the epoch elements for both the reference and actual trajectories in the exercise of the operational approach as a planning tool.

Table 5-14 5-Year Optimized Ellipso Borealis™ Node at Noon Epoch Elements and Tolerances

Element	Epoch State	Tolerance
Semi-major axis (km)	10496.8757	+/- 1.0000
Eccentricity	0.3330	+/- 0.0003
Inclination (deg)	116.5614	+/- 0.0500
Right Ascension of Ascending Node (deg)	0.0000	+/- 0.5000
Argument of Perigee (deg)	260.0000	+/- 1.0000
Mean Anomaly (deg)	0.0000	+/- 1.0000

As with the global cases, the reference orbit was created using the DSST software with the epoch elements from Table 5-14 propagated using a 50 x 0 gravitational field. The actual orbit was propagated using the same elements and a 21 x 21 gravitational field with drag, solar-radiation pressure, solar and lunar third-body point mass disturbances, and solid-earth tide effects also included. A Jacchia-Roberts atmospheric density model was used in the determination of the drag effects. A spacecraft mass of 1250 kg and area of 43.3 m^2 was also assumed in all cases.

5-4-5-1-2 Planning Test One Year Validation Case

Prior to attempting to run the operational approach software to plan for an entire five-year lifetime of the Borealis™ satellite, a shorter one-year run was first attempted. By continually looping through the operational approach software with updated epoch elements corresponding to the end state of a previous optimization, the approach was able to successfully maintain the Borealis™ satellite within the required limits for 404.8 days using 14 burns and a ΔV of 14.88 m/s. However, despite the successful completion of the one-year validation attempt, a number of needed improvements were evident. When implemented, these improvements allowed for even better operation of the approach as a planning tool.

Allowable Number of Iterations: The most noticeable feature of the one-year optimization using the operational approach was the lengthy time required for convergence. Using five processors on an SGI machine, the optimization still required more than 24 hours of computer time. Assuming linear growth, more than five days of computer time would be required to complete the desired five-year estimation.

The driving factor behind the required time seemed to be the number of iterations required by some of the optimization loops. Although the majority of the genetic algorithm optimizations were able to converge in approximately 1000 iterations, a few required approximately 5000 iterations. Inspection of the difference revealed that those optimizations requiring 5000 iterations could be attributed to initial convergence to an incorrect solution. Although these populations were able to eventually converge to the correct solution, the time and iterations required to move from the incorrect solution were higher than expected.

It was found that restarting the optimization allowed for quicker convergence than waiting for the partially converged solution to move from the incorrect point. This was done by simply limiting the allowable number of iterations to a lower number (such as 1000). Most of the time, the optimization is completed before this limit is reached. However, if the allowable limit of iterations is reached, the optimization is stopped and the end state is checked against the reference state. If the resulting deviations are larger than the allowable (i.e. the trajectory is not in the box), the current population is destroyed and the algorithm is restarted from the last known epoch state. Using this approach, it was found that, at most, three restarts were needed. Thus, at most 3000 iterations were required, greatly reducing the required amount of time for convergence.

Size of the Time Grid: Since this approach is trying to approximate a continuous time process with a discrete number of computations, it is necessary to define a grid over which the reference and actual orbits can be compared. For the global case, this comparison was performed four times every day without problem (once every six hours). However, for this more operational approach, all the burns are constrained to fall within a two-day interval. Thus, the probability that two burns will occur within a six-hour grid point is much higher. Although the net effect of those two burns will still be calculated by the DSST propagator (which has its own pre-defined step size), it is possible for a violation of the constraints to occur in between the six hour checks (i.e. one burn causes a constraint violation and the other fixes it, before a check for violations is performed.)

This undesirable behavior is quite evident in both the semi-major axis and eccentricity history plots for the one year planning case (see Figure 5-17 and Figure 5-18). Twice during the year history of the semi-major axis a burn occurs which forces the deviation from the reference down to greater than three kilometers, only to have a second burn occur which forces the deviation back less than the 1 km limit. Similar behavior can be seen in the eccentricity deviation plot in which the deviation is forced up beyond 4×10^{-4} only to be almost immediately driven back below the 3×10^{-4} limit. In all cases the burns in question occur less than six hours apart, and therefore, the deviations are not seen by the objective function which is only checked on a six hour grid.

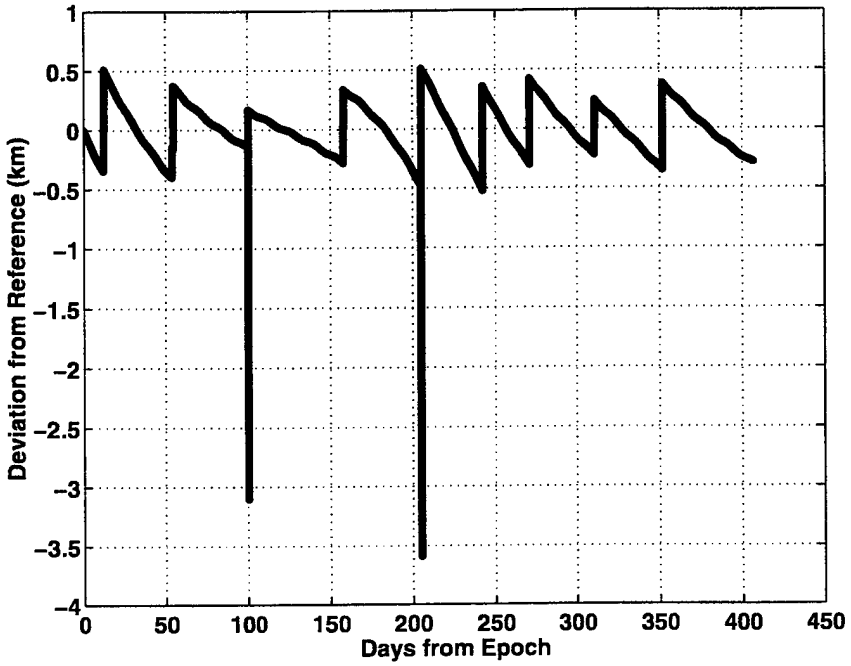


Figure 5-17 One Year Operational Approach Planning Test Semi-major Axis Deviation (Limit = 1 km)

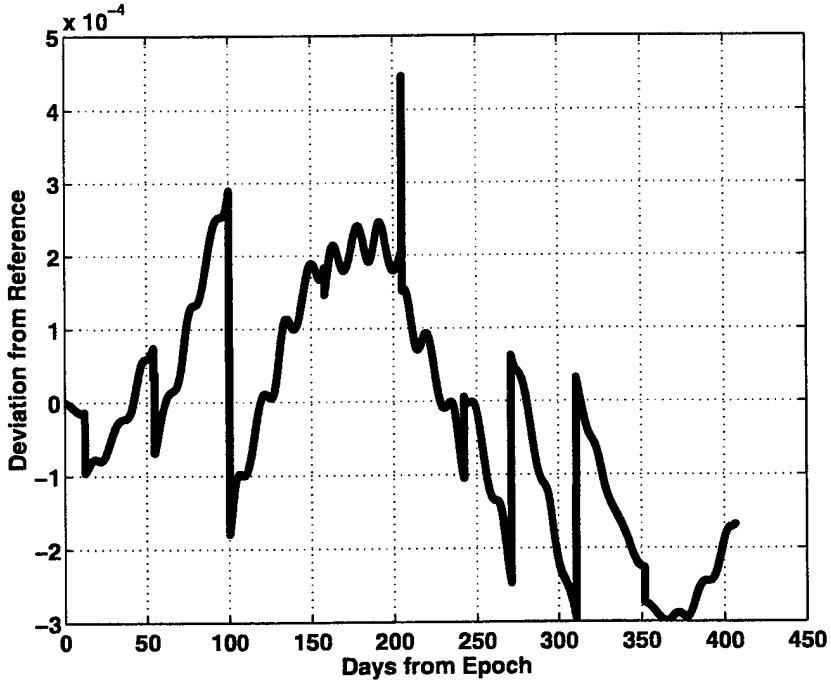


Figure 5-18 One Year Operational Approach Planning Test Eccentricity Deviation (Limit = $3x10^{-4}$)

It seems that simply checking the deviations over a denser time grid can solve this problem. However, this solution goes counter to the desired behavior of fast convergence. By checking even eight as opposed to four times per day, the number of computations for each iteration are doubled, and the problem still might not be eliminated as any deviations which occur within a three hour window still would not be seen. Ideally, the grid should be defined to be less than the orbit period of 1.5 hours, but with the objective function defined as it is, checking this often is not feasible.

A more reasonable solution was discovered by noticing that the only time a dense grid is required is during the allowable window for burns to occur (in this case, a two-day window). During the time that burns cannot occur, it is impossible for any element to drive extremely far out of its allowable range and back in before being checked (this scenario can only be created by burns). Therefore, by modifying the code to check over a one-hour grid during the two-day burn window and only an eight-hour grid over the rest of the optimization time period, it was hoped that the undesirable deviations could be eliminated while still maintaining relatively low times for convergence.

It should also be noted that despite the undesirable nature of the deviations in this case, it is conceivable that lower ΔV values could be achieved by redefining the problem to allow violations during the length of the burn window. It is often the case that burns of the type found in Figure 5-17 and Figure 5-18 are useful for changing certain elements at lower cost than other types of burns. By allowing deviations to occur during the short length of the burn window, some satellites may be able to be controlled at lower cost than when even the burns are constrained to stay within the constraint box.

5-4-5-1-3 Planning Test 2.5 Year ΔV Estimation

Even with the improvements in convergence time gained by implementation of the changes discussed in the previous section, the time required to complete a five-year ΔV estimation was still felt to be longer than desirable. Therefore, despite the desire for a five-year ΔV estimation, the planning tool was only used to estimate approximately a 2.5-year period of the Borealis™ Node-At-Noon satellite's lifetime. Although not entirely accurate, approximate estimates of the five-year period can be gained by scaling the solution over the entire five-year period.

This 2.5-year ΔV estimation was created using the 5-year optimized elements listed in Table 5-14 along with the actual and reference orbits discussed in the accompanying section (section 5-4-5-1-1). The resulting optimization required less than 24 hours of computer time (using 5 processors) and resulted in the solution summarized in Table 5-15.

Table 5-15 2.5-Year Borealis Node-at-Noon ΔV Planning Test Results

Days Forward	995.07 days
Number of Burns	65 burns
Total Delta-V	24.35 m/s

The deviation histories of all six orbital elements for this 2.5-year case can be found in appendix D-4-3. The semi-major axis deviation history is also presented below (see Figure 5-19) as it contains behavior that merits discussion. Note that unlike previous solutions, even during the time that the satellite is maneuvering to correct its state, no violations of the semi-major axis are present. Due to a failure to check over a dense enough grid, previous optimizations had found solutions that drove the semi-major axis

beyond its one-kilometer limit and back in a short enough time span that the deviation was not noticed by the algorithm. However, prior to this 2.5-year optimization, the solution to this problem discussed in section 5-4-5-1-2 was implemented. This implementation allowed the algorithm to sample more often during the times of possible burns, but less often during the drift time. By sampling more often during the times of possible burns, strings which contained burns that caused "out and back" violations were quickly eliminated. Due to the absence of any of this type of deviations in the solution to the 2.5-year optimization, it can be reasonably concluded that the methodology and implementation for eliminating these deviations was successful.

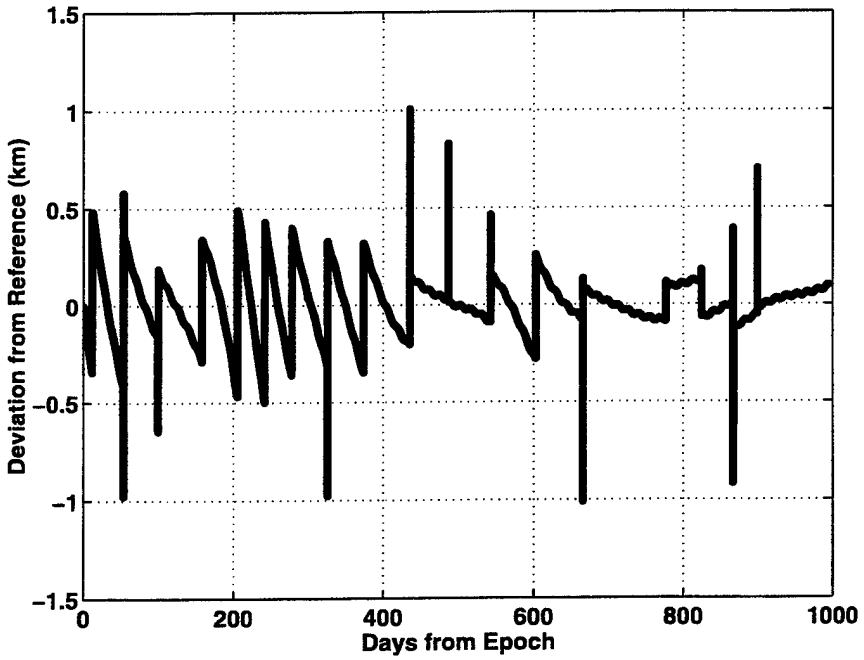


Figure 5-19 2.5-Year Operational Planning Approach Semi-Major Axis Deviation (Limit = 1 km)

5-4-5-1-4 Limitations of the Operational Approach as a Planning Tool

Despite the success of the operational approach as a planning tool over the 1000-day region from March 21, 2000, there is a significant limitation of the approach which must be discussed. This limitation is the inability to change the parameters of the optimization for each loop when the optimizations are strung together in the manner used by the operational planning tool approach.

The 2.5-year planning tool test case provides an illustrative example of this limitation. Note that over the entire 1000 day region for the Borealis™ optimization each cycle of the optimization entails making changes to essentially the same elements: a, e, and M. Only occasionally do the argument of perigee, inclination, or ascending node require change (see Appendix D and the ascending node deviation history in Figure 5-20). Due to the similarity of each step, the weighting factors, genetic algorithm parameters, number of burns, etc. can remain fairly constant and the algorithm will still be successful. However should the required optimization change drastically, (i.e. should additional elements violate the constraints on a routine basis), it is entirely likely that the optimization will fail if forced to use the initial set of parameters.

For example, due to the nature of the 2.5 year Borealis™ optimization presented in the previous section, it was possible to achieve successful results by limiting the algorithm to four allowable burns, each with a maximum component magnitude of 3 m/s. Additionally, an increase in speed of convergence was obtained by realizing that, due to the simplicity of the problem, near maximum time could be determined in less than 500 iterations. However, although these parameters worked well for this 2.5-year optimization, they did not work well if times beyond the 1000 days were attempted.

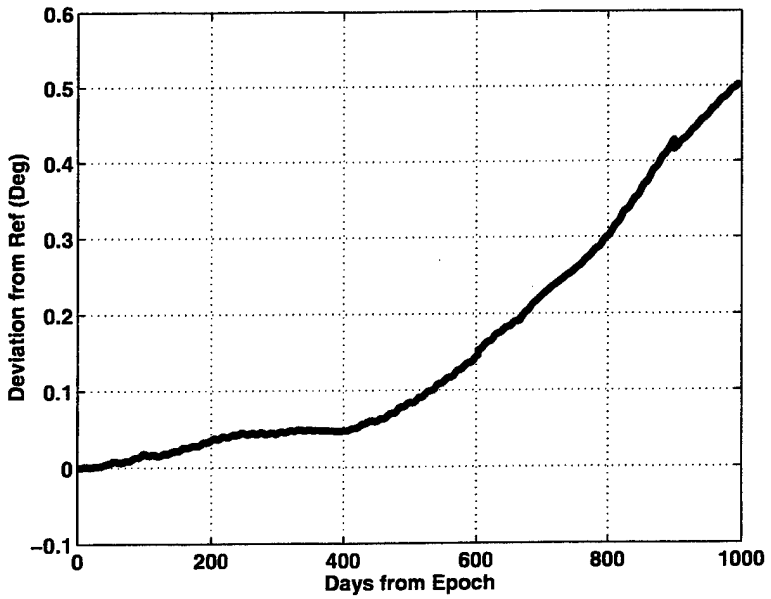


Figure 5-20 2.5-Year Operational Planning Approach RAAN Deviation (Limit = 0.5°)

Note the ending location of both the argument of perigee and the ascending node at the end of the 1000-day optimization. Both end at/or near their tolerance limit. This ending location makes optimizations beyond this point more difficult than those which preceded it. Rather than only dealing with eccentricity and Mean Anomaly violations, optimizations beyond this point must find a way to handle the secular violation of the ascending node and argument of perigee as well. The control of these new violations requires more burns as well as burns of larger magnitude than the four 3 m/s burns previously allowed. Additionally, due to the more difficult nature of the optimization, iterations beyond the 500 allowable may be required to find the maximum drift time. The result of these changes is that the parameters that worked well previously are no longer adequate, and due to the unpredictable effect of previous burns on the trajectory, it is impossible to predict these changes ahead of time.

The ability to predict the required weights is not an issue when the algorithm is run in an operational rather than a planning sense. When run in a non-planning mode, the algorithm is only run one step at a time and an operator would be present to interpret the results. Using previous experience, the operator could then input the required parameters to allow for a successive step to be completed. However, when run in the planning mode, the operator is not present at each step and the inability to input the necessary weights becomes a serious limitation to the usefulness of the method as a planning tool.

5-4-5-2 Greediness of the Operational Approach

Despite the low ΔV estimate that the operational approach was able to generate for the 2.5 year Borealis orbit analyzed in this study, it is important to remember that the operational approach is a greedy strategy. As a greedy algorithm, the approach cares only about finding the optimal ΔV strategy for the here and now, it cares nothing about the future. Due to this failure to take the future into consideration, continual application of the operational method over long periods of time can lead to high fuel estimates and undesirable behavior, particularly if secular motion of some of the elements is involved.

5-4-5-2-1 Example of the Operational Approach Greediness

The Borealis™ satellite studied again provides a useful example of this phenomenon. As has been discussed, the 1000-day period studied above posed no problems, as the secular motion of the ascending node did not reach the tolerance limit until only slightly before the end of the 1000 days. However, by performing an optimization for a slightly longer period of time (approximately 1200 days) the greediness of the algorithm becomes evident.

Figure 5-21 contains the deviation of the ascending node from the reference trajectory resulting from the approximately 1200-day optimization. The behavior of the orbit during the first 1000 days is, as expected, unperturbed by the optimization. However, upon reaching the limit of 0.5° at about the 1000-day point, the behavior changes quite significantly. As the goal of each step of the optimization is to simply find a way to move forward some specified time step (typically about 30-45 days in this region), the burns resulting from each step are only enough to barely maintain the ascending node inside the tolerance region. As this process repeats itself for each time period, the cumulative result is a trajectory that seems to "crawl" along the specified tolerance limit.

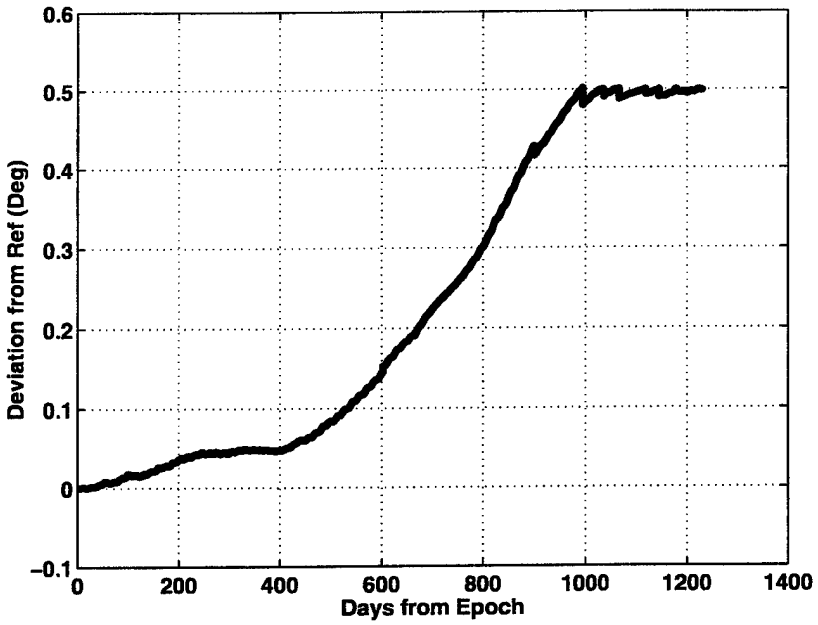


Figure 5-21 RAAN Deviation of 1200 Day Optimization Showing Greedy Behavior

Although this behavior meets the requirement of maintaining the satellite's orbit inside the constraint box, it is clear that this is a greedy and more costly solution than a global optimal solution would be. The greediness is even more evident when the ΔV for the first 1000 days is compared to the ΔV required for the last 200. Table 5-15 shows that for the first 1000 days a total of 24.35 m/s was required to maintain the trajectory inside the specified constraints. However, once control of the secular behavior of the ascending node was introduced, an additional 15.35 m/s was required for the last 200 days. This is an increase of over 63 percent for only a 20 percent increase in time.

However, a significant increase in ΔV alone is not necessarily the result of a greedy strategy. It is possible, that a significant increase in ΔV would be required to control the violation of the ascending node that occurs around the 1000^{th} day, even in a non-greedy approach. The nature of the increase, however, is a more significant indicator of the greediness of the approach.

Figure 5-22 and Figure 5-23 show this nature, graphically. Figure 5-22 contains a fifth order polynomial fit to the total amount of ΔV required as a function of time. Figure 5-23 is similar, but shows the total number of burns required as a function of time. Note, in both figures, the significant increase in slope of the functions after the 1000^{th} day. It is this increase in slope that signals a greedy result. As the algorithm progresses over time without regard to future ramifications of current choices, it is forcing higher ΔV choices in the future as evidenced by the increase in slope. A similar plot of a non-greedy algorithm would not have the continual increase in slope shown here. Rather, the overall slope would be fairly constant with discontinuities at certain locations corresponding to

large ΔV requirements to return to secular motion of certain elements (such as the ascending node) to near nominal states.

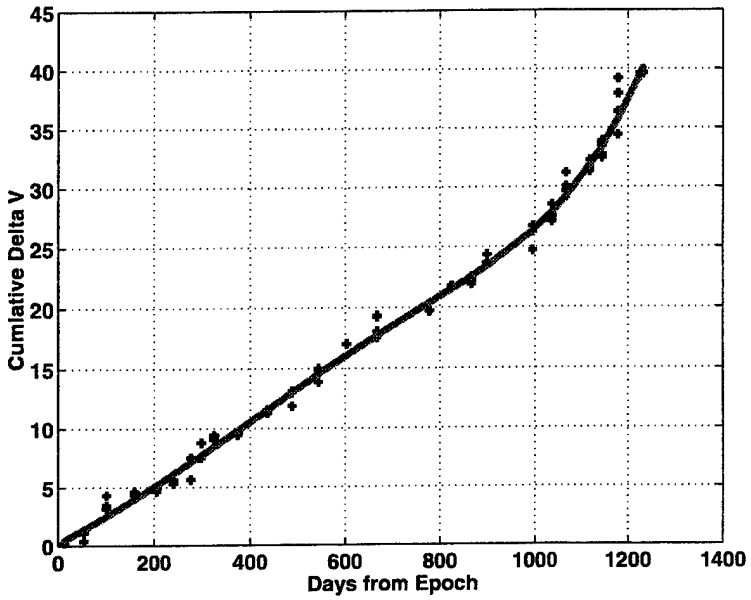


Figure 5-22 Cumulative Delta-V vs. Time for Greedy Case

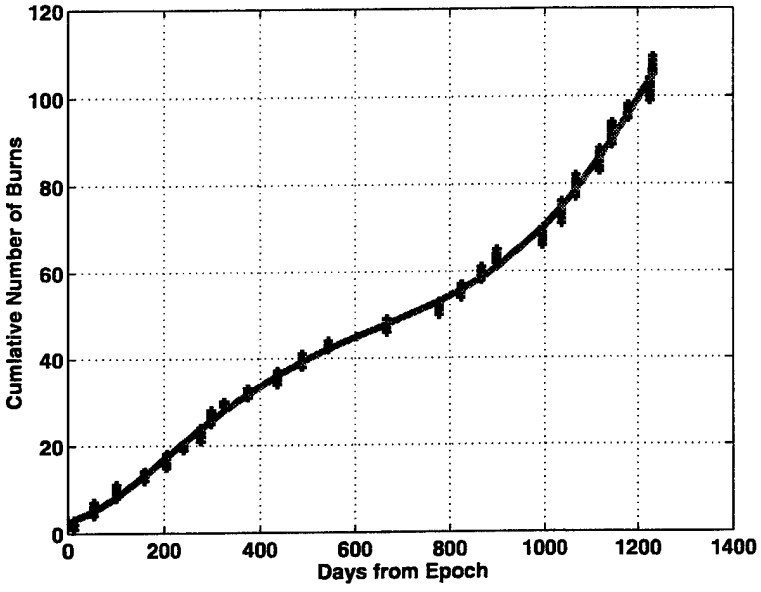


Figure 5-23 Number of Burns Required vs. Time for Greedy Case

5-4-5-2-2 Possible Solutions to Minimize Greediness of the Operational Approach

The fundamental limitation of the greediness of the operational approach is that it allows the algorithm to ignore secular variation of non-violating elements until the variations reach a violating state. At this violating state the system is much more unstable and difficult to control than if extra effort had been expended early on to maintain the secular variation near nominal values.

Returning again to the ascending node deviation history in Figure 5-21, note the rate of change of the deviation in the first 500 days as compared to the last 500 days. With all elements starting at their nominal values, the system is quite stable and the secular rate of change of the ascending node is small. However, as the overall stability of the system is allowed to degrade, the tendency toward instability also increases. As, the nominal elements have been designed with maximum stability in mind (see Chapter 4), a direct correlation can be found between deviation from these nominal values and instability of the system.

The result of this correlation is the need to find an algorithm that maintains the final solution closer to the nominal values than to the artificially defined tolerances. The best solution would be to implement the global algorithm. This approach would notice the increase in ΔV required by allowing the secular motion to continue unchecked during the first few hundred days. However, current limitations in computing power (see section 5-3-4) make this solution path infeasible. Therefore, a number of other paths are proposed as possible solutions to eliminate some of the negative effects of the greediness of the operational approach. These proposed solutions are summarized below.

Tighten Constraints: Probably the most easily implemented of the proposed solutions is to simply modify the constraint box such that the secular violations of the ascending node and other elements are not allowed to vary enough to destabilize the system. By tightening the constraints, the algorithm would be forced to perform adjustments to the ascending node and other secular violations earlier in the process, thereby, eliminating the ability to wait on these adjustments and also eliminating some of the greediness of the algorithm. The difficulty of this implementation lies in choosing the appropriate constraint values. Excess tightening of the constraints may lead to an increase in stability, but may also lead to an unnecessary increase in frequency and number of maneuvers.

Implement a Restart Protocol: A second possible solution that would allow the system to remain more stable is the implementation of some sort of restart protocol. By defining either certain intervals or states at which the solution process must be "restarted" or returned to nominal, overall stability of the system could be increased. For example, the restart protocol could be defined such that any time the solution is seen to "crawl" along the tolerance limit for some fixed number of iterations, the operational approach is paused and the burns necessary to return the system to nominal are computed. This return to nominal would return stability to the system and the operational approach could be continued as before. The main limitation to this solution is, again, the need for specific insight into the problem before the restart protocol can be defined. Implementation of the improper protocol may indeed eliminate the greediness, but at the expense of large amounts of ΔV.

Integral Objective Function: The most reasonable of all solutions to the greediness resulting from the secular behavior of some of the elements is to modify the objective function such that the integral of the deviation from nominal is minimized over time. This would have the effect of averaging out the secular variation around nominal and therefore would maintain average stability. Unlike the other two suggested solutions, the implementation of this solution requires little insight into the specifics of the case at hand. In some sense, the algorithm itself would determine the proper limits and restart protocol.

The only difficulty to this implementation would be the balancing of the ΔV and deviation contributions to the objective function. In the current formulation, only deviations outside of the constraint box contribute to the objective function. In feasible solutions (those which meet all constraints), the deviation contributions are zero and minimization can be performed only on the ΔV. However, with an integral formulation, even feasible solutions will have both deviation and ΔV contributions to the objective function. Proper balancing of the weighting of these contributions will have a significant effect on the solution. Too much weight on the integral portion will produce solutions that are maintained very close to the nominal value, but at the expense of ΔV. On the other hand, too much weight on the ΔV portion will produce solutions with low ΔV, but which fail to meet the desired behavior of minimum deviation from nominal.

[This Page Intentionally Left Blank]

Chapter 6 Conclusions and Future Work

This work developed optimization methods that proved successful in both designing and maintaining satellite constellations. Two design applications were developed and tested along with two methods for performing orbit maintenance. Initial optimizations were performed using Powell's localized method, but convergence to local minimums required the introduction of genetic algorithms into the optimization process.

A robust genetic algorithm astrodynamic optimization tool was created using the genetic algorithm package PGAPack in conjunction with the Charles Stark Draper Laboratory implementation of DSST and the Mississippi State University development of MPICH. These three software tools were tied together through a series of FORTRAN routines to create an optimization tool general enough to handle both design and maintenance applications with minimal modification.

The genetic algorithm optimization tool developed was first used successfully in the refinement of two Ellipso™ constellation designs. The Borealis™ sub-constellation initial elements were refined to provide minimum drift away from the three desired behaviors of 113:14 repeat ground track, fixed argument of perigee and constant nodal rate. Based on the success of the tool in optimizing the repeat ground track design, it was also used to refine the initial orbital elements of the recently conceived gear array. The optimal design for the gear array was found to provide better coverage and elevation characteristics than the previous Borealis™/Concordia™ design of the Ellipso™ constellation.

The combination PGAPack, DSST, MPICH optimization tool was also used successfully to generate optimal station-keeping strategies for the Borealis™ sub-constellation using two different optimization approaches. The first approach viewed the problem from a global perspective and attempted to find all burns necessary to maintain the desired constellation over an entire satellite lifetime. Although the method was successful over short time frames, computational and other limitations prevented it from being successful over an entire satellite lifetime, as originally desired. Additionally, the method lacked a number of desirable operational characteristics such as repeatability and speed of convergence.

To overcome the limitations of the global station-keeping approach, a second method was implemented. This second method can be considered a more operational approach. This operational approach did prove successful in generating near-optimal station keeping strategies, and although, due to greediness, the required ΔV's are slightly higher than the ΔV's required by the global approach, the method can be more easily implemented into a satellite operations concept.

The next sections list specific conclusions and recommendations for future work for each of the four design and maintenance problems detailed in this thesis.

6-1 113:14 Repeat Ground Track Design

Optimization techniques were first used to find the optimal orbital elements that minimized the total deviation of an Ellipso Borealis™ node-at-noon satellite from the three desired design behaviors of 113:14 repeat ground track, fixed argument of perigee, and sun synchronous node behavior. Both Powell's method and genetic algorithms were used to find the optimal epoch elements for a five year optimization starting on January

1st, 1997 in a 50 x 0 zonal field with third-body effects also included. The result of the optimization was a set of elements that proved extremely stable in the design space as well as relatively stable in the presence of full perturbations. The optimal elements for this design are presented below in Table 6-1.

Table 6-1 Optimal Epoch Elements for the Borealis Node at Noon 113:14 5-Year Optimization (Epoch = January 1, 1997)

Element	Value
Semi-major Axis (a)	10496.8969 km
Eccentricity (e)	0.32985
Inclination (i)	116.5782°
RAAN (Ω)	280.00°
Argument of Perigee (ω)	260°
Mean Anomaly (M)	0.00°

6-1-1 113:14 Repeat Ground Track Design Conclusions

One important conclusion gained from the 113:14 repeat ground track design application was the inadequacy of the localized methods to avoid convergence to incorrect solutions. This behavior was evident even for this fairly simple problem with only three solve-for variables. Although the localized methods were able to find the optimal solution on the majority of the trials, there were also a number of trials in which they failed. Without some sort of external feedback, such as plots or a second optimization technique, determining which of the solutions is truly the global optimum can be difficult.

A second conclusion relates to the importance of designing in the same environment that the actual operation of the satellite will be performed. For this 113:14 repeat ground track design problem, the optimal elements were designed in a 50 x 0 zonal/third body field as opposed to a field which included all possible perturbations.

The result was a set of initial elements that gave optimal behavior under the 50 x 0/third-body field, but which were not necessarily optimal under the full perturbation model.

6-1-2 Areas of Future Work for the 113:14 Repeat Ground Track Design

Two areas are recommended for future pursuit. First, despite the significant increase required in computer time, it is recommended that the design be optimized under the full perturbation model and the results compared to the optimal orbital resulting from the 50 x 0/third body optimization. It is expected that the elements resulting from the full perturbation optimization will vary slightly from those presented in Table 6-1 and that corresponding changes in the drift from nominal under full perturbation models will also be seen. To reduce computation time in performing the optimization under full perturbations, it might be possible to create a hybrid optimization scheme where the result from the genetic algorithm simple model optimization is used as the starting point for a localized method with full perturbations included.

The second area of recommended future work deals with the amount of allowable drift in the elements. Even with an optimization of the elements under full perturbation analysis, it will still be impossible to exactly meet the desired conditions at all times. Some amount of drift will still be present in the element histories of the final solution. If the amount of drift present is larger than acceptable values, it may be necessary to perform the optimization in a manner that would solve for more than just the initial elements. Instead, it may be necessary to solve for the optimal elements at various times throughout the lifetime of the satellite. This type of optimization would require a significant reformulation of the problem.

6-2 Gear Array Design

The design of the gear array was an attempt to use the same optimization tools that were successful in refining an existing constellation concept to refine an entirely new constellation design. In this new design, the two orbit planes are aligned at the equator and the orbit of one sub-constellation is designed to have a period that is a certain ratio of the period of the second sub-constellation. This type of design forces motion that is similar to the teeth of a gear assembly.

In order to design a gear array, the problem was decomposed into three desired behaviors: an apogee pointing to the Sun constraint, a ratio constraint, and a geometric constraint. Various methods of parameterizing the problem were also studied. Using a fixed offset method of parameterization, two types of gear array were constructed. The design of these arrays was constructed using a 50 x 0 zonals only field over the course of one year with an epoch date of January 1, 1997. The resulting design parameters for the two arrays are summarized in Table 6-2.

Table 6-2 Optimal Gear Array Design Parameters

	5:6 GEAR 8050 APOGEE		4:5 GEAR 0 KM OFFSET	
Array Type	APTS	Circular	APTS	Circular
Number of Satellites	5	6	4	5
Semi-Major Axis (Km)	12537.37	14143.57	12546.57	14555.45
Eccentricity	0.151172	0.0	0.160114	0.0
Apogee Height	8050.	7765.	8177.	8177.
Perigee Height	6149.	7765.	4159.	8177.
Phasing in Mean Anomaly	72	60	90.	72.
Anomalistic Period (secs)	13947.98	16737.58	13980.01	17475.07

6-2-1 Gear Array Design Conclusions

The first conclusion regarding the gear array design optimization is quite similar to one of the conclusions drawn from the repeat ground track optimization. Specifically, it can be concluded that the resulting solution would be more optimal if designed in the desired operation space (a fully perturbed 5-year optimization) as opposed to a zonals only 1-year optimization space. However, like the 113:14 case, computer limitations prevented this ideal optimization from actually occurring.

Further conclusions relating to optimization techniques can also be drawn from the gear array design process. For example, in attempting to find the optimal design for the gear array, three parameterizations of the problem were studied. Although each parameterization contained the same information, the resulting solution spaces were drastically different. This leads to the conclusion that for many optimization problems, the formulation of the problem can have a significant impact on the ability to find an optimal solution.

The gear array design also allows for conclusions regarding coverage to be made. Due to the nature of the optimally designed gear array constellations, all communication constellation characteristics exceeded those for the baseline Ellipso™ Concordia™ sub-constellation including satellites in view and minimum and average elevation angles. Although some of this increase can be attributed to the increase in the number of satellites in the gear array design, a significant portion of the increase can also be attributed to the phasing and design of the gear array.

6-2-2 Areas of Future Work for the Gear Array Design

For the future it is recommended that the gear array designs be refined in a full perturbation field. The differences in decay can then be compared with the current design to see if the increase in computer time for the fully perturbed optimization is worth the expected decrease in decay.

Additionally, a variety of gear combinations beyond the 4:5 and 5:6 cases implemented for this thesis should be studied. It is quite possible that an array that exhibits better coverage than either of the arrays studied to date exists. It is also possible that additional solve-for variables, such as the number of satellites in each array, and additional constraints, such as coverage constraints, could be built into the optimization process. If properly implemented the addition of such solve-for variables and coverage parameters could make possible the design of a more optimal array.

6-3 Global Station Keeping Optimization

Genetic algorithms were also used successfully to maintain the Borealis™ node-at-noon satellite within certain pre-defined tolerances in relation to the designed reference orbit. Using the 113:14 repeat ground track optimization process with an epoch date of March 21, 2000 and a total time of 90 days, the optimally designed elements were found and used to create a reference orbit in a 50 x 0 zonal field which exhibited the desired characteristics. These elements and tolerances are summarized in Table 6-3.

Table 6-3 90-Day Optimized Ellipso Borealis™ Node at Noon Epoch Elements and Tolerances (Epoch = March 21, 2000)

Element	Reference Epoch State	Tolerance
Semi-major axis (km)	10496.8839	+/- 1.0000
Eccentricity	0.3328	+/- 0.0003
Inclination (deg)	116.5577	+/- 0.0500
Right Ascension of Ascending Node (deg)	0.0000	+/- 0.5000
Argument of Perigee (deg)	260.0000	+/- 1.0000
Mean Anomaly (deg)	0.0000	+/- 1.0000

Two different sets of initial elements were then used to test the application of the genetic algorithm to the station-keeping problem. The first case aligned the initial elements with the reference elements and propagated the actual orbit in a fully perturbed field. The resulting four-burn solution was successfully able to maintain the difference between the actual and reference orbits within the predefined constraints. The second case moved the actual epoch elements to a location such that all elements were near the predefined tolerances in an effort to test the robustness of the system. Although 12 burns were required to control this trajectory, the method was still successful and provides an upper bound on the ΔV required to control the satellite. The total number of burns and delta-v for the two cases used to test this approach are summarized in Table 6-4.

Table 6-4 Results of Two Global Station Keeping Optimization Test Cases

Case	Number of Burns	Delta_V (m/s)
1-Elements Aligned With Reference	4	0.5693
2-Elements at Extreme Limits	12	9.9900

6-3-1 Global Station Keeping Optimization Conclusions

An important aspect of the implementation of genetic algorithms into the global station keeping optimization was the change from binary to real allele values. Despite their usefulness in the design optimization schemes, binary variable representations were found to be inadequate to handle the refinement necessary for the station-keeping approaches. It was also found that a number of other genetic algorithm parameters required modification before the genetic algorithm code created for the design applications could be successfully used to optimize the maintenance applications.

The need for modification of the genetic algorithm parameters is an important conclusion as it shows the need for proper tuning of the genetic algorithm for each application. It is impossible to define the "perfect" genetic algorithm parameters that will work for all problems, even when the problems are somewhat related. As proof of this point, the parameters that were used for each optimization in this thesis are summarized in Table 6-5. Note that although there are similarities among these parameters, significant differences also exist.

Table 6-5 Genetic Algorithm Parameter Comparison

Parameter	113:14	Gear	Global	Operational Time Opt.	Operational ΔV Opt.
Variable Type	Binary	Binary	Real	Real	Real
Stopping Rule	No Change	NC	NC	NC	NC
No Change Value	100	150	2000	100	2000
Population Size	100	100	100	100	100
Replaced Per Iteration	25	25	25	25	25
No Duplicates Flag	True	True	True	True	True
Mutation & Crossover Flag	True	True	True	True	True
Mutation Type	Binary	Binary	Gaussian	Gaussian	Gaussian
Real Mutation Constant	N/A	N/A	0.5	0.2	0.5
Mutation Rate	1/L	0.02	1/L	1/L	1/L
Crossover Type	Two-Point	2-Pt	2-Pt	2-Pt	2 Pt
Crossover Probability	0.85	0.85	0.85	0.85	0.85
Selection Type	Tournament	Tournament	Tournament.	Tournament.	Tournament

Even with proper tuning, the global station-keeping algorithm was unable to reach the original goal of defining station-keeping strategies for the entire lifetime of a given satellite. Instead it was found that the method was extremely limited by the amount of available computing power. This leads to the conclusion that although a given optimization technique might be possible, it is not always feasible. There is not anything about the approach presented in this thesis which should keep it from working over longer time frames, it is just not feasible to attempt to do so.

6-3-2 Areas of Future Work for the Global Station Keeping Optimization

One of the areas of future work which is expected have a significant impact on the global station keeping optimization scheme is the creation of a hybrid genetic algorithm/localized approach. One of the major limitations of the global station keeping approach is the number of iterations and corresponding amount of time required to exercise the algorithm. A large percentage of the iterations are required simply to refine the optimal solution (i.e. zero out unneeded burns, etc.). By incorporating a localized algorithm into the refinement stage of the global station keeping optimization, it is expected that required run times could be greatly reduced.

An additional area to which future efforts could be directed is the definition of the reference orbit. For all cases studied in this thesis, the reference orbit was defined as the orbit in a zonal 50 x 0 field that gave the minimum drift from the 113:14 repeat ground track characteristics. At times near epoch, burning to meet this designed orbit is desirable. However, as time elapses, even the designed orbit begins to decay from the desired behavior. After this decay has begun, burning to meet the designed orbit is not necessarily the best thing to do. After some time has elapsed from epoch, it might be

possible to burn to a set of elements different from the reference trajectory that will cause smaller future drift from the 113:14 repeat ground track characteristics than the previously designed orbit. By performing a re-optimization of the 113:14 repeat ground track orbit design at regular intervals, it may be possible to define a reference that is more stable, relative to the 113:14 characteristics than the orbit resulting from only one optimization performed at epoch.

It may also be desirable to perform an analysis into the actual dependence of the global station keeping approach on the orbit propagation. In discussing the limitations of the global approach, the point was made that should the satellite even slightly deviate from the trajectory laid out by the global station keeping algorithm, all future burns would have to be recalculated. Although this statement is true, it is possible that some slight error in the propagation could be ignored. Through a series of tests, it might be possible to determine the sensitivity of the global solution process to the expected propagator error.

Drawing on the lessons learned from the gear array design process, it might be useful to also perform an analysis into the different ways to parameterize the station-keeping problem. In the design of the gear array, it was found that various parameterizations yielded solutions with varying levels of difficulty. It is possible that the current parameterization of the station-keeping problem is leading to some of the difficulty in generation of optimal solutions. Study into other ways to parameterize the problem could therefore prove very useful.

Along similar lines, the effect of the objective function formulation is also an area into which further efforts could be directed. This thesis focused solely upon controlling

the spacecraft based upon orbital element constraints. As other methods of constraining the trajectory are possible, these methods should be implemented and the effect on the solution process analyzed. For example, rather than constraining the orbital elements, it may be desirable to constrain a function of these elements such as ground track or cross-track and along-track. The implementation of this type of control is thought to be fairly simple, however the effect on the optimization tool's performance is unknown and should be pursued.

6-4 Operational Station Keeping Optimization

In an effort to overcome many of the limitations present in the global station keeping approach, a second application of genetic algorithms to the station-keeping problem was created. Due to the faster convergence, easier implementation, and increase in repeatability (all desirable operational characteristics), this application was termed the operational station keeping approach.

The main difference between the global and operational station keeping approaches was the introduction of a two-layered optimization scheme. In the operational approach, prior to solving for the optimal burns, the maximum drift times between constraint violations is first computed. Due to the addition of a restriction that burns can only occur in the region of an expected deviation, the maximum drift time is quite easily solved for. Using the maximum drift time, the optimal burns that allow for that drift time can then be solved for in the second step of the optimization. The entire process requires many fewer iterations than the global approach.

The operational station keeping optimization tool was first tested using epoch elements identical to the global case two epoch elements. This case was successful in showing the viability of this approach.

Attention was then turned to the application of the operational approach to ΔV estimations over extended periods of time. By restarting the algorithm with the final elements from a previous run as the epoch elements for the next, the operational approach was found to be a viable way of making reasonable ΔV estimations. Using this methodology, the approach was used to compute a one-year test case as well as a two and a half-year ΔV estimation. The results of these cases are summarized in Table 6-6.

Table 6-6 Results of Three Operational Station Keeping ΔV Planning Test Cases

Case	Actual Days Forward	Number of Burns	Delta_V (m/s)
1 Year Test Case	404.8	14	14.88
2.5 Year ΔV Estimation	995.1	65	24.35

6-4-1 Operational Station Keeping Approach Conclusions

One important conclusion relating to the operational station keeping approach is in regards to the effectiveness of greedy algorithms. Although over some time frames, the operational approach was successful, it was found that the greediness of the approach eventually led to difficulty. This supports the original thought that a global approach is often a better choice than a greedy one for performing optimizations.

6-4-2 Operational Station Keeping Approach Areas of Future Work

The most important area of future work for the operational station keeping approach is the incorporation of some strategy to contain the secular motion near the nominal value as discussed in section 5-4-5-2-2. Three possible solutions are presented

in this section which might be successful at accomplishing this goal. However, further study is required to determine which of the possible solutions is indeed the best for eliminating some of the greediness of the operational station keeping approach.

A second area of possible future study that was not pursued in this thesis is the optimal value of the restriction on the burn times. For this thesis the burns were constrained to lie within either one day or one half day on either side of the expected deviation. However, these time limits were simply chosen at random. It is thought that forcing the burns to lie within one orbit period might have a significant impact on the performance of the algorithm, as the number of burn locations will be greatly reduced. However, exactly what that impact will be is yet to be determined.

Appendix A 113:14 Repeat Groun d Track Optimization Data

This appendix contains data necessary to perform the optimization of the 113:14 repeat ground track case, as well as various plots representing the results of that optimization. Specifically, this appendix contains the DSST input deck, 3-D surface plots of the surface to be optimized, genetic algorithm FORTRAN code specific to this case, and the resulting element decay plots.

A-1 113:14 Repeat Ground T rack DSST Input Deck

This section contains the input deck to the DSST propagator for the Borealis™ node-at-noon, argument of perigee at 260° orbit used in the optimization of the 113:14 repeat ground track orbit design process. Proper flag and keyword values were determined from personal communications with Dr. Ronald Proulx, The Charles Stark Draper Laboratory.

```
C
C
C      PMEF FILE FOR 113:14 OPTIMIZATION
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.1997010100000000D+08 PME_DATE          1
   0.0000000000000000D+06 PME_TIME          2
   0.1049688200000000D+05 ELS_KEP(1)        3
   0.3320880821578410D-00 ELS_KEP(2)        4
   0.1165837080726980D+03 ELS_KEP(3)        5
   0.2800000000000000D+03 ELS_KEP(4)        6
   0.2600000000000000D+03 ELS_KEP(5)        7
   0.0000000000000000D+03 ELS_KEP(6)        8
   0.0000000000000000D+00 ELS_EQUIN(1)      9
   0.0000000000000000D+00 ELS_EQUIN(2)     10
   0.0000000000000000D+00 ELS_EQUIN(3)     11
   0.0000000000000000+00 ELS_EQUIN(4)      12
   0.0000000000000000D+00 ELS_EQUIN(5)     13
   0.0000000000000000D+00 ELS_EQUIN(6)     14
   0.0000000000000000D+00 POSVEL(1)        15
   0.0000000000000000D+00 POSVEL(2)        16
   0.0000000000000000D+00 POSVEL(3)        17
```

```
0.0000000000000000D+00 POSVEL(4)       18
0.0000000000000000D+00 POSVEL(5)       19
0.0000000000000000D+00 POSVEL(6)       20
0.2000000000000000D+01 PME_CD          21
0.0000000000000000D+00 PME_RHO_ONE     22
0.5000000000000000D-04 SMA_SIGMA       23
0.5000000000000000D-04 INC_SIGMA       24
0.5000000000000000D-04 ASC_SIGMA       25
0.1250000000000000D+04 PME_SCMASS      26
0.4330000000000000D-04 PME_SCAREA      27
0.8640000000000000D+05 PME_STEPSIZE    28
0.1400000000000000D+02 DP_SPARE1       29
0.1130000000000000D+03 DP_SPARE2       30
0.0000000000000000D+00 DP_SPARE3       31
0.0000000000000000D+00 DP_SPARE4       32
0.0000000000000000D+00 DP_SPARE5       33
0.0000000000000000D+00 DP_SPARE6       34
         1             PME_RETRO       35
        12             PME_KEP_SYS     36
        12             POS_VEL_SYS     37
         1             GEN_METHOD      38
         1             ATMOS_MODEL     39
    840401             JACRB_DATE      40
       123             JACRB_SSS       41
    840401             SLP1950_DATE    42
       456             SLP1950_SSS     43
    840401             SLPTOD_DATE     44
       789             SLPTOD_SSS      45
    840401             TIMECF_DATE     46
       123             TIMECF_SSS      47
         2             HARRIS_MODEL    48
        10             POTNTL_MODEL    49
        50             PME_NMAX        50
         0             PME_MMAX        51
         1             PME_IZONAL      52
         1             PME_IJ2J2       53
         0             PME_NMAXRS      54
         0             PME_MMAXRS      55
         1             PME_ITHIRD      56
         2             PME_INDDRG      57  2 = DRAG OFF
         2             PME_ISZAK       58
         2             PME_INDSOL      59  2 = SOLRAD OFF
         2             PME_JSHPER      60
         1             PME_JZONAL      61
         1             PME_JMDALY      62
         2             PME_INP_TYPE    63
        12             PME_EQUI_SYS    64
        11             INTEG_FRAME     65
        19             OUTPUT_FRAME    66
         0             PME_NSTATE      67
         1             PME_SPSHPER     68
         2             PME_KSPCF       69
         4             PME_INDSET      70
         0             INT_SPARE1      71
         0             INT_SPARE2      72
         0             INT_SPARE3      73
         0             INT_SPARE4      74
         0             INT_SPARE5      75
         0             INT_SPARE6      76
         0             INT_SPARE7      77
         0             INT_SPARE8      78
         0             INT_SPARE9      79
         0             INT_SPARE10     80
```

A-2 113:14 Repeat Ground Track Surface Plots

This section contains plots of the surface to be optimized for the 113:14 repeat ground track case. The optimization was actually performed in four-dimensional space. However, since four dimensions could not be plotted, these plots were generated by fixing one of the variables at the optimal value and allowing the other two values of interest to vary. These plots can therefore not be used to validate the optimality of all three variables, but they are useful in gaining an understanding of the types of surfaces the constraints in this problem generate.

For all sections, the order of the plots is the same. The first plot is a three-dimensional view of the surface. The second plot is a top view of the surface with contour lines drawn. The final two plots in each section are edge on views along one of the two variable axes. These edge on views allow for the optimal value of each variable to be easily distinguished.

For Section A-2-1, inclination was fixed at the optimal value 116.5782° and semi-major axis and eccentricity were allowed to vary. The same process was followed in A-2-2, with semi-major axis fixed at its optimal value of 10496.8968 km. Finally, Section A-2-3 contains the surface plots with eccentricity fixed at 0.32986 and semi-major axis and inclination taking a range of values. The most notable of these sections is section A-2-2. The plots in this section clearly show regions of local minima in the (e-I) space, into which a localized method might unknowingly converge.

A-2-1 113:14 Repeat Ground Track Semi-major Axis/Eccentricity Plots

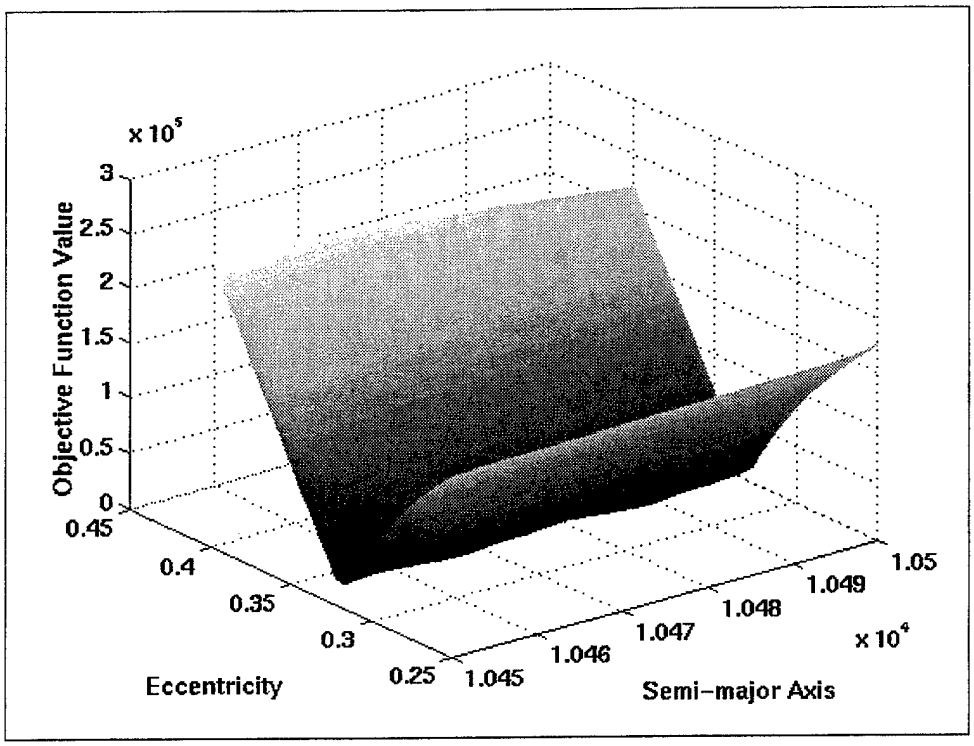


Figure A-1 3-D Surface of 113:14 a/e Space (i = 116.5782°)

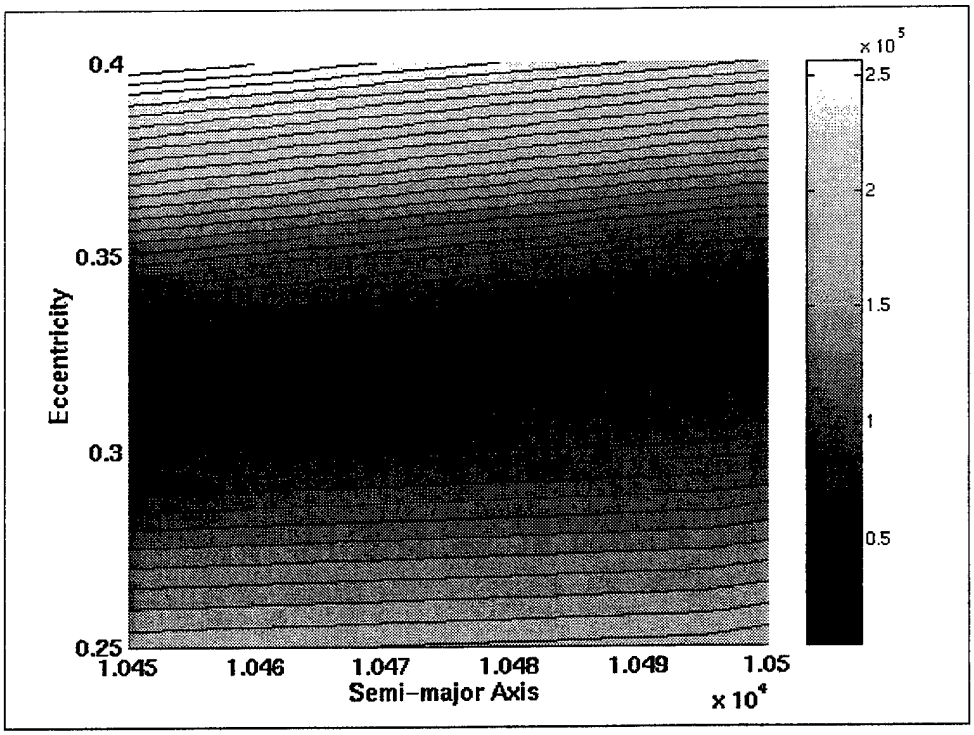


Figure A-2 Contour Plot of 113:14 a/e Space (i = 116.5782°)

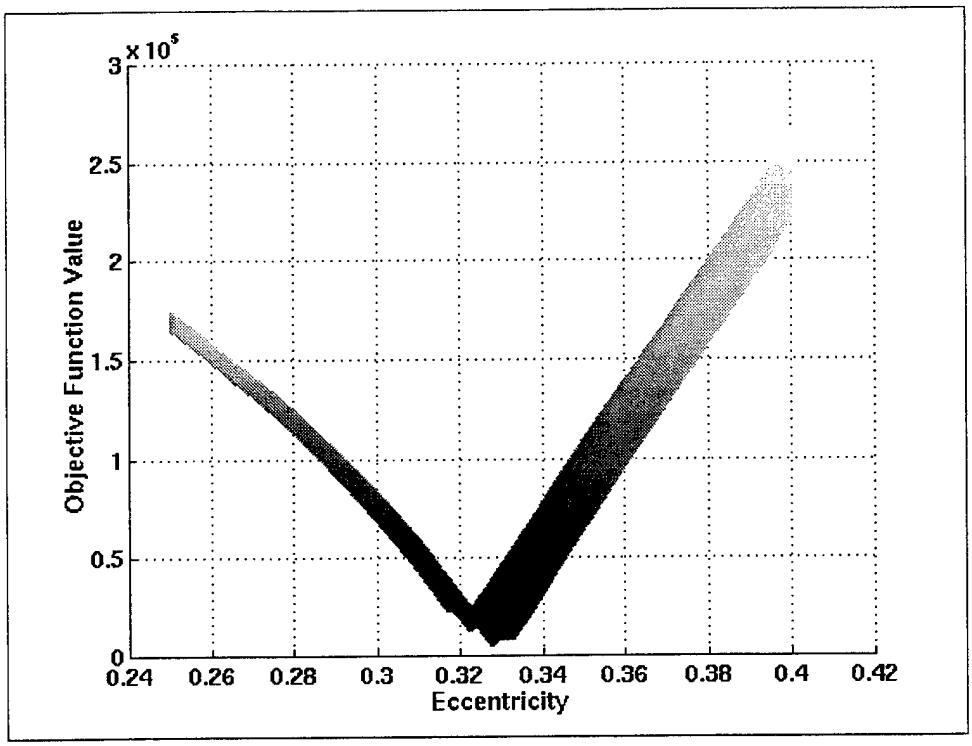


Figure A-3 Eccentricity Edge-on View of 113:14 a/e Space (i = 116.5782°)

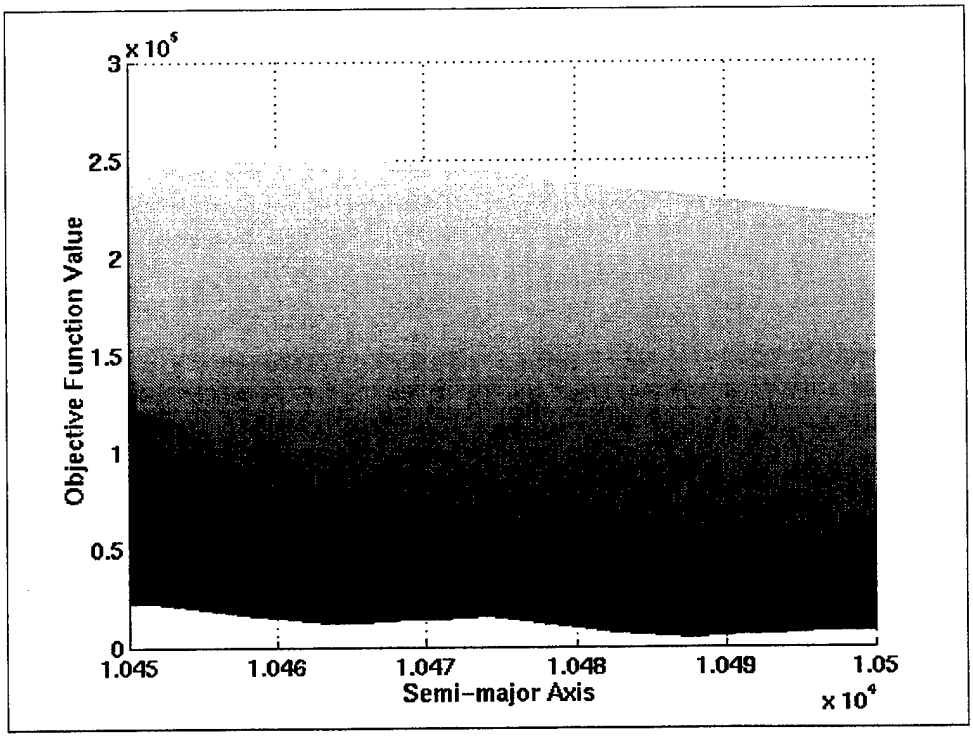


Figure A-4 Semi-Major Axis Edge-on View of 113:14 a/e Space (i = 116.5782°)

A-2-2 113: 14 Repeat Ground Track Eccentricity/Inclination Plots

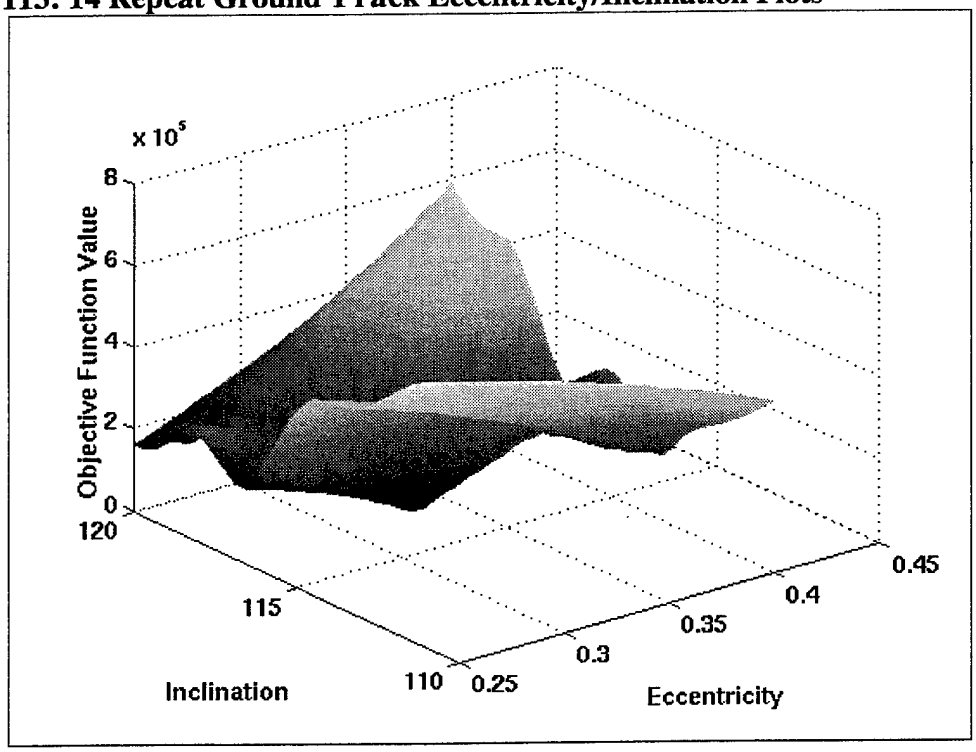


Figure A-5 3-D Surface of 113:14 e/i space (a = 10496.8968 km)

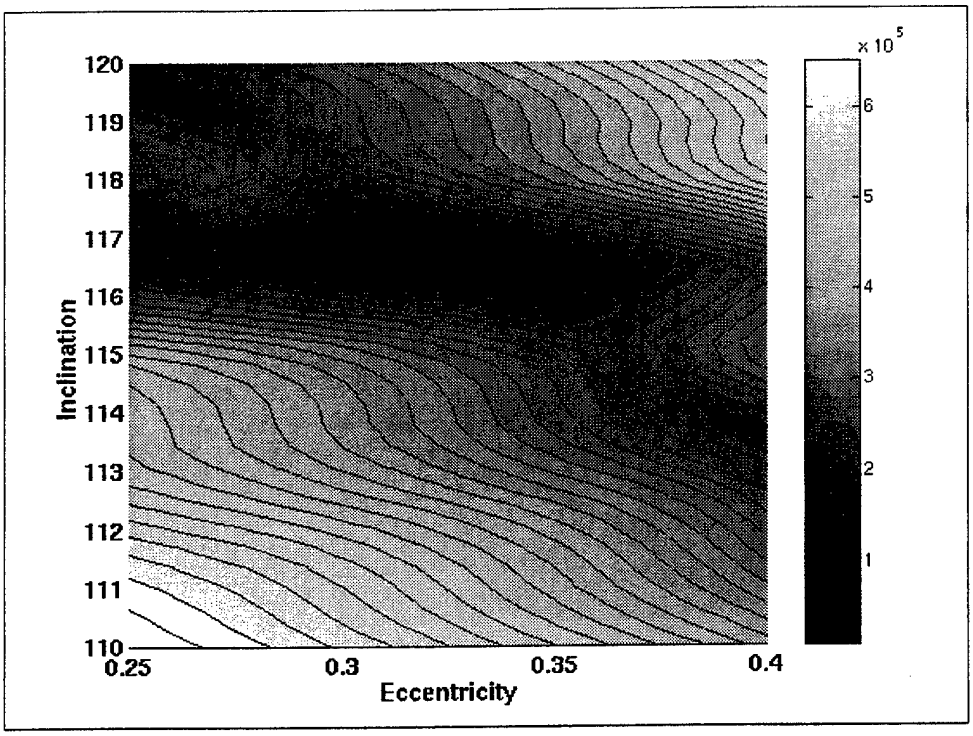


Figure A-6 Contour Plot of 113:14 e/i space (a = 10496.8968 km)

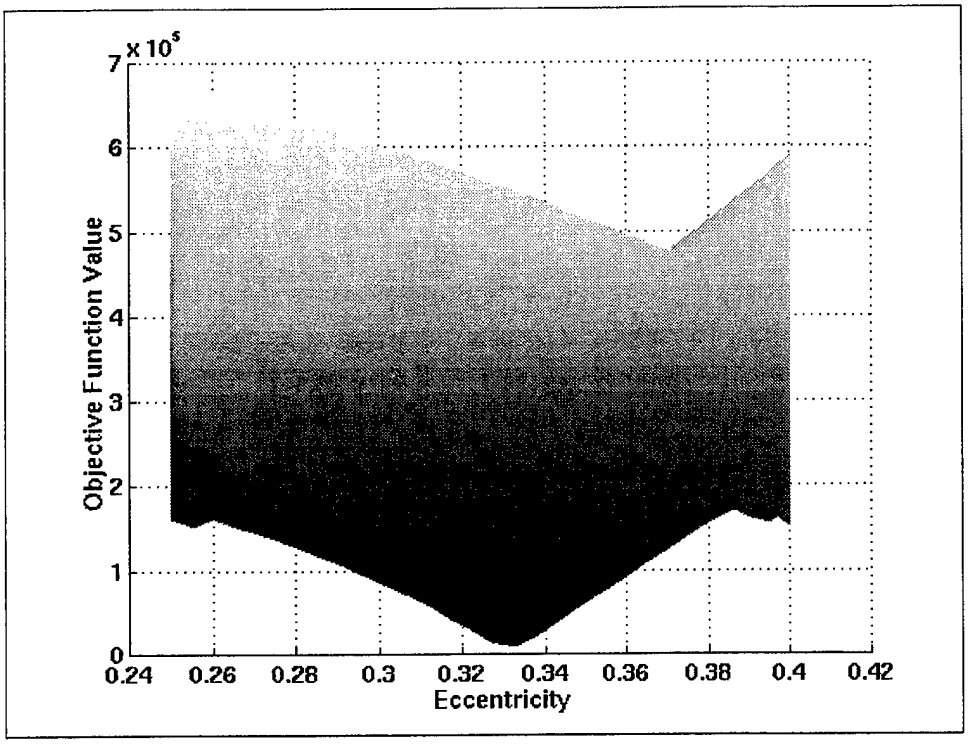


Figure A-7 Eccentricity Edge-On View of 113:14 e/i space (a = 10496.8968 km)

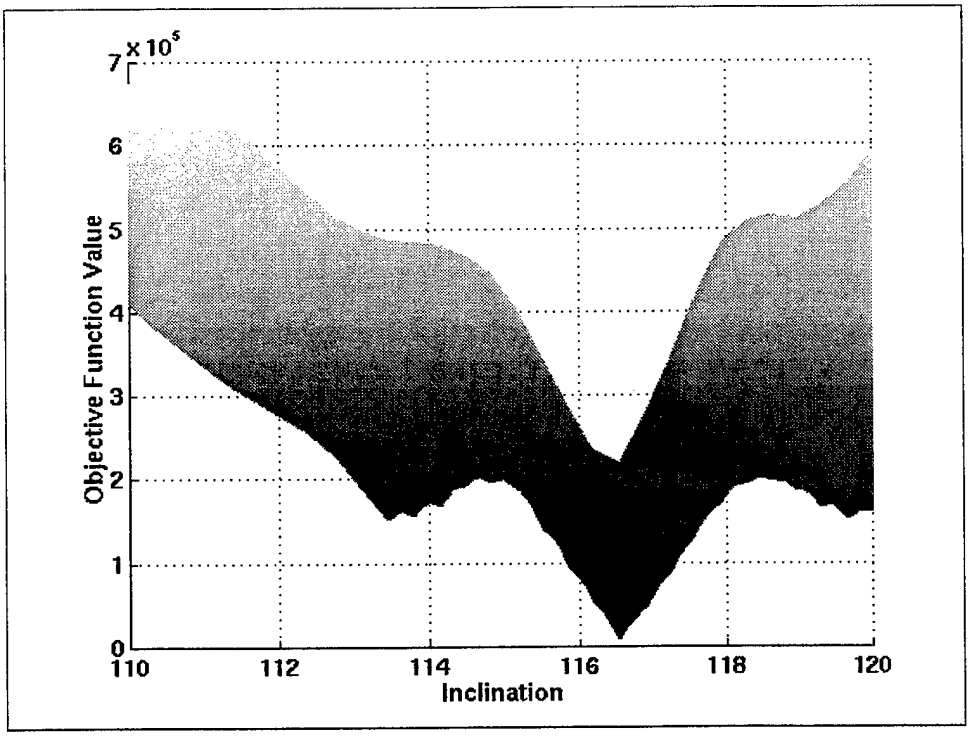


Figure A-8 Inclination Edge-On View of 113:14 e/i space (a = 10496.8968 km)

A-2-3 113:14 Repeat Ground Track Semi-major Axis/Inclination Plots

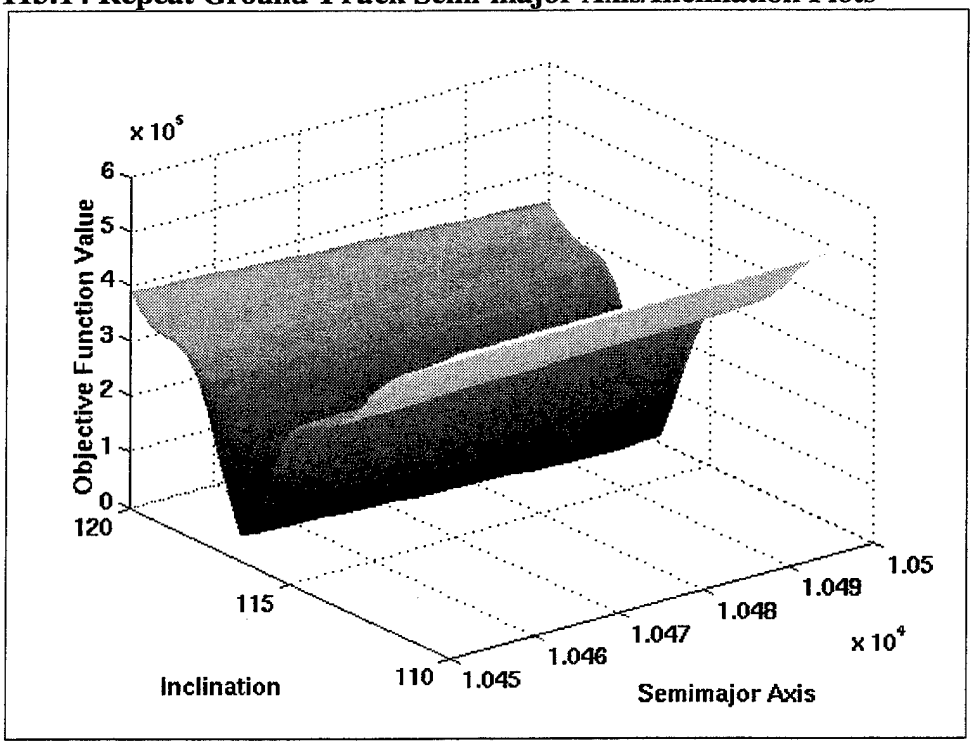


Figure A-9 3-D Surface of 113:14 a/i Space (e = 0.32986)

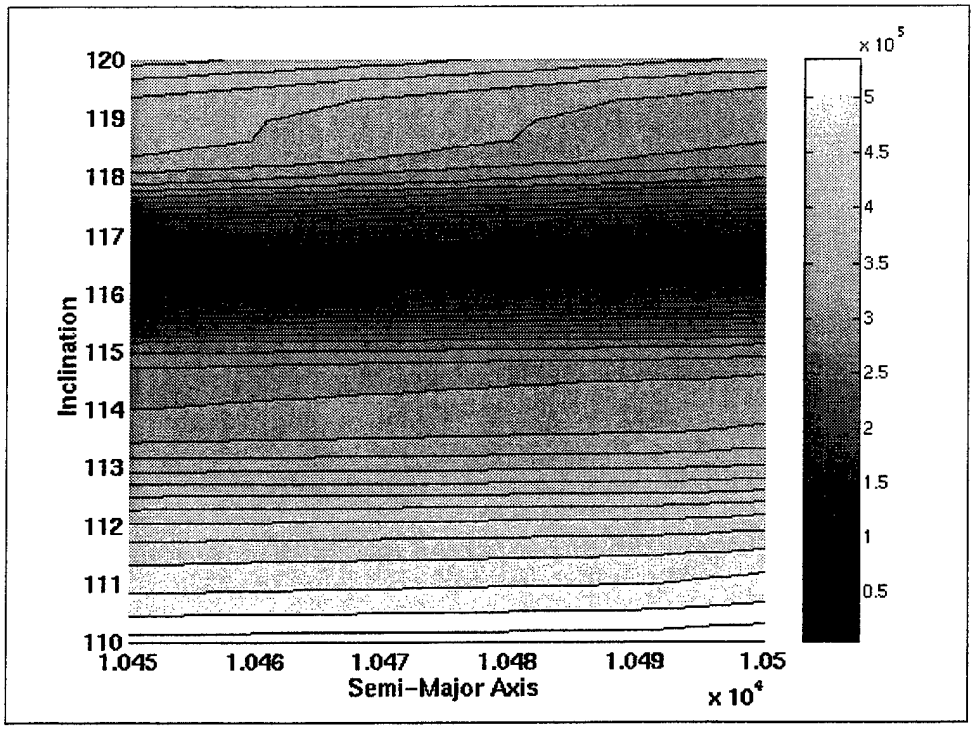


Figure A-10 Contour Plot of 113:14 a/i Space (e = 0.32986)

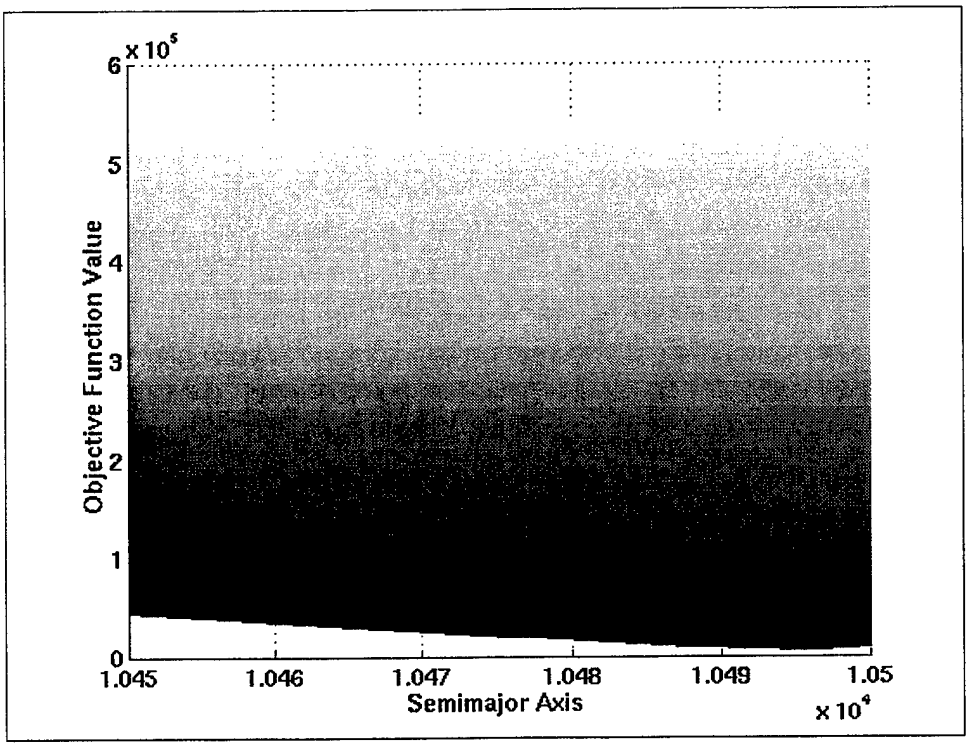


Figure A-11 Semi-major Axis Edge-on View of 113:14 a/i Space (e = 0.32986)

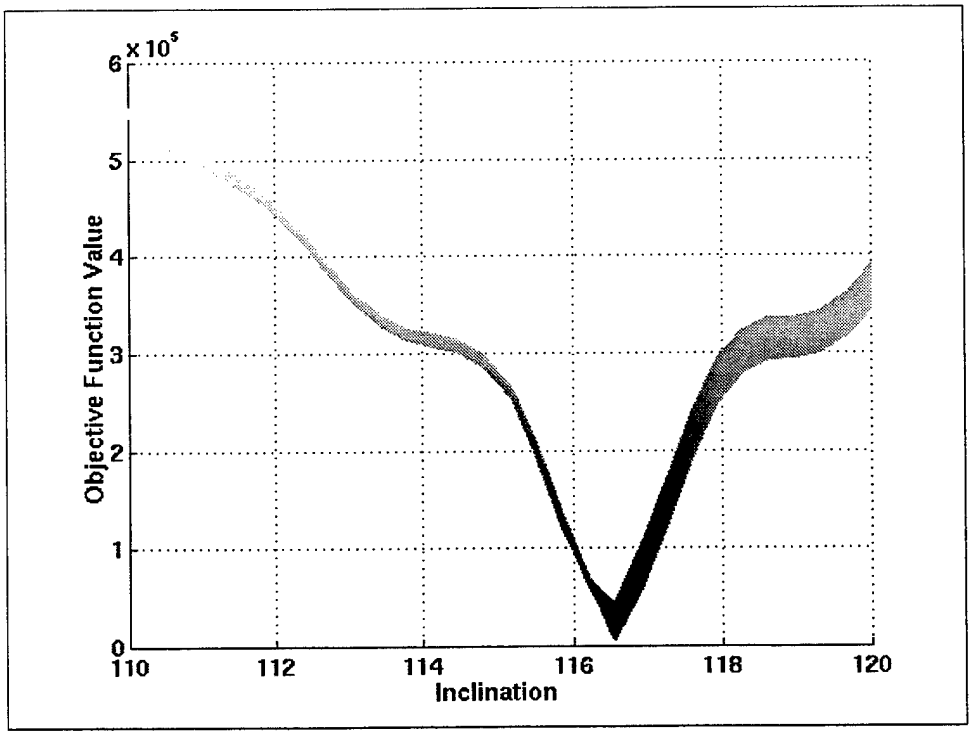


Figure A-12 Inclination Edge-on View of 113:14 a/i Space (e = 0.32986)

A-3 113:14 Repeat Ground Track Genetic Algorithm Code

This section contains the FORTRAN code that was developed specifically to tie DSST to the PGAPack software in the solution of the 113:14 repeat ground track problem. For a description of the function of each routine see Chapter 4.

A-3-1 PGA_113_14

```
Program pgasat
C-----------------------------------------------------------------------
C
C                              Program PGASAT
C
C  This file contains Genetic Algorithm code for optimizing any number
C  of orbital elements in conjunction with DSST orbit propagator.
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C  Modifications:
C  Ver  Date      Author                Description
C  ---  --------  ------------------    -----------------------------
C  1.0  01/15/98  J. Smith/R. Proulx    Original
C  1.1  01/23/98  J. Smith              Moved limit initialization to
C                                       setlim
C  1.2  02/03/98  J. Smith/R. Proulx    Moved call to initialize_sat to
C                                       func
C
C  Parameters:
C     CTX          PGAPack context variable
C     P            Chromosome Index in Population
C     POP          Which Population to refer to
C     MYID         MPI process ID variable
C     IERROR       MPI error code
C     BESTINDEX    Index of the best string
C     I            Counter Variable
C     BEST         Array containing the optimized values
C
C  Subroutines Called:
C
C  MPI Routines:
C     MPI_Init                          Initialize MPI
C     MPI_Comm_rank                     Determine ID of all processes
C     MPI_Bdcast                        Broadcast message to all processes
C     MPI_Finalize                      Finalize MPI
C
C  PGA Pack Routines:
C     PGASetStoppingRuleType            Define stopping Rule
C     PGASetMaxNoChangeValue            Set number of iterations to
C                                       continue without change
C     PGASetNumReplaceValue             Set number of string to change in
C                                       each population
C     PGASetNoDuplicatesFlag            Allow/Disallow duplicate strings
C     PGASetMutationAndCrossoverFlag    Mutation and Crossover performed
C                                       separately or together
C     PGASetUp                          Setup the PGA routines
C     PGASetPrintOptions                Allows for various output options
```

```
C     PGARun                          Run the Genetic Algorithm code
C     PGADestroy                      End the PGA Code
C
C  Other Routines:
C     init_headers
C     init_pgalim                     Creates the pgalim common block
C     setlim                          Use to set number of variables to
C                                     be optimized and limits
C
C  External Functions:
C      findbest                        Converts binary string to real
C                                     and returns fitness values
C
C
C-----------------------------------------------------------------------
      implicit none

      include 'pgapackf.h'
      include 'mpif.h'
      include 'pgalim.h'

      double precision findbest
      external         findbest

C----------------- Variable Declarations -----------------------------

      integer ctx, p, pop
      integer myid, ierror

      integer bestindex, i
      double precision best(MAXVAR)

      character*24 fdate
      external fdate

C------------------ Main Program -------------------------------------

C     Initialize MPI processes

      write(*,*) fdate()

      call MPI_Init(ierror)
      call MPI_Comm_rank(MPI_COMM_WORLD, myid, ierror)

C     Initialize DSST code and PGALIM for PGA code

      call init_headers
      call init_PGALIM

C     Limits on the variables to be optimized must be included here

      call setlim

C     Calculate the string lengths and the location of each variable
C     within string.
C
      PGALIM.LEN = PGALIM.NUMVAR*PGALIM.BPN

      PGALIM.BITLIML(1)=1
      PGALIM.BITLIMU(1)=PGALIM.BPN

      I = 2
      DO WHILE (I . LE. PGALIM.NUMVAR)
         PGALIM.BITLIML(I)=PGALIM.BITLIML(I-1)+PGALIM.BPN
```

```
         PGALIM.BITLIMU(I)=PGALIM.BITLIMU(I-1)+PGALIM.BPN
         I = I+1
       ENDDO

       ctx = PGACreate(PGA_DATATYPE_BINARY, PGALIM.LEN, PGA_MINIMIZE)

C      This section contains all the non-default PGA Pack modifications
C      the user wishes to make

C      This defines the stopping rule and sets the number of iterations
C      in which no change is allowed before stopping

       call PGASetStoppingRuleType(ctx,PGA_STOP_NOCHANGE)
       call PGASetMaxNoChangeValue(ctx,100)

C      This call sets the number of strings to replace in each population

       call PGASetNumReplaceValue(ctx,25)

C      This call allows or disallows duplicate strings in the population.
C      NOTE:  If string length is short this may cause a failure as it
C      is impossible to generate enough combinations so that there aren't
C      any duplicates.

       call PGASetNoDuplicatesFlag(ctx, PGA_TRUE)

C      If set to true, this call allows for mutation to occur on strings
C      which are produced through crossover.  If false, mutation can
C      only occur on strings which did not experience crossover.

       call PGASetMutationAndCrossoverFlag(ctx,PGA_TRUE)

C      This call prints the average of each population as well as the
C      best.

       call PGASetPrintOptions(ctx,PGA_REPORT_AVERAGE)

C      This section runs PGA Pack and finds the "best" values of the
C      parameters based on a metric defined by "findbest"

       call PGASetUp(ctx)

       call PGARun(ctx, findbest)

C      This final section converts the best values to real numbers,
C      prints them and ends the program

       If (myid .eq. 0) then
          bestindex = PGAGetBestIndex(ctx,pop)
          i = 1
          do while (i .le. PGALIM.NUMVAR)
            best(i) = PGAGetRealFromGrayCode(ctx,bestindex,pop,
     1                  PGALIM. bitliml(i),PGALIM. bitlimu(i),
     2                  PGALIM.lowlim(i),PGALIM.uplim(i))
            print *, best(i )
            i = i + 1
          enddo
       endif

       call PGADestroy(ctx)
       call MPI_Finalize(ierror)

       write(*,*) fdate()
```

```
      stop
      end
```

A-3-2 SETLIM

```
SUBROUTINE setlim
C
C
C     Revision History ***********************************************
C
C     Rev     Date     Who/Where            Comments
C     1.0  01/23/98    J.E. Smith/CSDL     Initial routine
C
C     Description ****************************************************
C
C     Set the nondefault values of variables for use by PGASAT
C
C     CALLING SEQUENCE ***********************************************
C
C          CALL setlim
C
C     PARAMETERS *****************************************************
C
C          No parameters
C
C
C     INPUTS FROM INCLUDED MODULES ***********************************
C
C          None
C
C     OUTPUTS FROM INCLUDED MODULES **********************************
C
C     NUMVAR
C     UPLIM
C     LOWLIM
C
C***************** DECLARATIONS ***************************************
C
      IMPLICIT NONE
C
C     HEADER FILES ***************************************************
C
      INCLUDE         'pgalim.h'
C
C     LOCAL VARIABLES  ***********************************************
C
      INTEGER*4    I
C
C***************** BEGIN PROGRAM **************************************
C
C
C Specify the number of variables to optimize

        PGALIM.NUMVAR = 3

C Specify the number of bits to code each variable with

        PGALIM.BPN = 20

C Specify the upper and lower limit on the variables to be optimized

        PGALIM.LOWLIM(1) = 10490.0D0
        PGALIM.UPLIM(1) = 10500.0D0
```

```
      PGALIM.LOWLIM(2) = 0.30D0
      PGALIM.UPLIM(2) = 0.45D0

      PGALIM.LOWLIM(3) = 110.0d0
      PGALIM.UPLIM(3) = 130.0D0


      RETURN
      END
```

A-3-3 FINDBEST

```
double precision function findbest(ctx,p,pop)
C-----------------------------------------------------------------------
C
C                              Function findbest
C
C
C     This function converts the binary strings to real numbers and
C     sends them to an "evaluation" function
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C  Modifications:
C  Ver  Date      Author                 Description
C  ---  --------  -----------------      -----------------------------
C  1.0  01/15/98  J. Smith/R. Proulx     Original
C
C  Parameters:
C     CTX         PGAPack context variable
C     P           Chromosome Index in Population
C     POP         Which Population to refer to
C     I           Counter Variable
C     X           Array of variables being evaluated
C
C  Subroutines Called:
C
C  PGA Pack Routines:
C     PGAGetRealfromGrayCode     Coverts binary strings to real numbers
C
C  External Functions:
C      func               Does the satellite propagation and calculates
C                         a fitness value
C
C-----------------------------------------------------------------------

      implicit none

      include 'pgapackf.h'
      include 'pgalim.h'

      real*8 func
      external func

C----------------- Variable Declarations -----------------------------

      integer ctx, p, pop,  i
      double precision x(MAXVAR)

C------------------ Main Program -------------------------------------
```

```
C     This steps converts the binary string into 'numvar' real numbers.

      i = 1
      do while (i .le. PGALIM.numvar)
        x(i) = PGAGetRealFromGrayCode(ctx,p,pop
     1   ,PGALIM.bitliml(i), PGALIM. bitlimu(i),
     2    PGALIM.lowlim(i),PGALIM.uplim(i))
        i = i + 1
      enddo

C     This call to func returns an fitness value for each string

      findbest=func(x)

      return
      end
```

A-3-4 FUNC

```
real*8  function func (p)
C-----------------------------------------------------------------------
C
C                              Function Func
C
C  This file contains the objective function for the 113_14 optimization
C  problem
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C          Ronald J. Proulx
C          Draper Laboratory
C
C
C  Parameters:
C     CTX          PGAPack context variable
C     P            Chromosome Index in Population
C     POP          Which Population to refer to
C
C  Subroutines Called:
C
C  External Functions:
C      satellite      Link to DSST Orbit Propagator
C
C
C-----------------------------------------------------------------------
      implicit none

      include 'pmern.h'
      include 'frc.h'

C-----  VARIABLE DECLARATIONS ------------------------------------------

C----- Define variables for call to Satellite ------------------

      integer*4       satellite
      external        satellite

      integer*4       max_list_length
      parameter      (max_list_length  = 1)
```

```
      integer*4       status
      integer*4       burn_number
      Integer*4       iatmos_preburn / 1 /
      integer*4       iatmos_postburn / 1 /

      real*8          time
      real*8          burn_delta_v(4,max_list_length)
      REAL*8          RHO_ONE_HIGH      / 0.0 D0 /
      REAL*8          RHO_ONE_LOW       / 0.0 D0 /
      real*8          epoch_ymd
      real*8          epoch_hms
      real*8          posvel(6)
      real*8          elements (17)

      character*(6)   name  /'pmern1'/
      CHARACTER*(72)  MESSAGE
      CHARACTER*(12)  FILENAME

C ------- Define Other Variables

      integer*4       i

      real*8          n_days
      real*8          n_revs
      real*8          ratio_nom
      real*8          ratio_cur
      real*8          lambda_rate
      real*8          ratio_dif
      real*8          ratio_var
      real*8          xp
      real*8          xq
      real*8          pdot
      real*8          qdot
      real*8          radians
      real*8          degrees
      real*8          p(*)
      real*8          omega_earth
      real*8          node_rate_nom
      real*8          node_rate
      real*8          node_init
      real*8          node_expected
      real*8          node_prev
      real*8          node_nom
      real*8          node_cur
      real*8          node_dif
      real*8          node_var
      real*8          perigee_init
      real*8          perigee_cur
      real*8          perigee_nom
      real*8          perigee_dif
      real*8          perigee_var

      PARAMETER   (  RADIANS        = 57.295779513082321    D00 )
      PARAMETER   (  DEGREES        =  0.017453292519943296 D00 )
      parameter      (node_rate_nom = .98564736d0/86400.d0) ! deg/sec


C     ********** BEGIN PROGRAM ********************************

C     *** Initialize the DSST Code ***************************

      call initialize_sat(name)
```

```
C     Set defaults for request time and burn list ************

      TIME              = 0.D0
      BURN_NUMBER       = 0
      DO I=1,MAX_LIST_LENGTH
        BURN_DELTA_V(1,I) = 0.D0
        BURN_DELTA_V(2,I) = 0.D0
        BURN_DELTA_V(3,I) = 0.D0
        BURN_DELTA_V(4,I) = 0.D0
      END DO

C  Set the DSST elements equal to the values from the GA string ***

      pmern.els_kepler(1) = p(1)
      pmern.els_kepler(2) = p(2)
      pmern.els_kepler(3) = p(3)

C *** Initialize the variations to zero ************************

      node_var = 0.d0
      perigee_var = 0.d0
      ratio_var    = 0.d0

c     Find the variations at each time ***************************

      DO i = 1,1827
         time = dble(i-1) * 86400.d0

C       CALL SATELLITE to obtain state at request time **********

        STATUS = SATELLITE ( name ,           TIME,
     2                       BURN_DELTA_V,    BURN_NUMBER,
     3                       IATMOS_PREBURN,  IATMOS_POSTBURN,
     4                       RHO_ONE_HIGH,    RHO_ONE_LOW,
     5                       EPOCH_YMD,       EPOCH_HMS,
     6                       POSVEL,          ELEMENTS,
     7                       MESSAGE,         FILENAME )

C **** Initialize Certain Values the First time through ************

        if(i.eq.1) then

           node_init = pmern.els_kepler(4)
           node_init = mod(node_init,360.d0)
           if(node_init.lt.0.d0) node_init = node_init + 360.d0
           node_prev = node_init - 86400.d0*node_rate_nom

           n_days = pmern.dp_spare(1)
           n_revs = pmern.dp_spare(2)
           omega_earth = frc.omega(1)
           ratio_nom = n_days/n_revs

           perigee_init = pmern.els_kepler(5)
           perigee_init = mod(perigee_init,360.d0)
           if(perigee_init.lt.0.d0) perigee_init = perigee_init + 360.d0

        endif

C     Find the Deviation from the Desired Node Behavior ***************

      node_nom    = node_init +    time  *node_rate_nom
      node_cur    =     elements(4)
      node_expected = node_prev + node_rate_nom*86400.d0
```

```
      do while (abs(node_cur - node_expected).gt.180.d0)
         if(node_cur .lt. node_expected) then
            node_cur = node_cur + 360.d0
         else
            node_cur = node_cur - 360.d0
         endif
      end do

      node_prev = node_cur
      node_dif = node_nom-node_cur

      node_var = node_var + abs(node_dif)

C     Find the Deviation from the Desired Argument of Perigee Behavior

      perigee_nom = perigee_init
      perigee_cur = elements(5)
      perigee_dif = perigee_nom - perigee_cur
      if(abs(perigee_dif).gt.180.d0) then
         if(perigee_dif.lt.0.d0) then
            perigee_dif=perigee_dif+360.d0
         else
            perigee_dif=perigee_dif-360.d0
         endif
      endif

      perigee_var = perigee_var + abs(perigee_dif)

C     Find the Deviation from the Desired Repeat Ground Track Behavior *

      xp = elements(9)
      xq = elements(10)
      pdot = elements(15)
      qdot = elements(16)
      lambda_rate = elements(17)
      node_rate  =  (xq*pdot - xp*qdot)/(xp*xp+xq*xq)
      ratio_cur = ( omega_earth - node_rate ) / ( lambda_rate*degrees -
      node_rate )
      ratio_dif = n_days/n_revs - ratio_cur
      ratio_var  = ratio_var + abs(ratio_dif)

      ENDDO

C ****Calculate the Total FUNC value through a weighted sum of the three
C     variations ***

      func = node_var + perigee_var + 10000.d0*ratio_var

      return
      end
```

A-4 113:14 Repeat Ground Track Element History Plots

This section of Appendix A consists of two sections, both of which contain the history plots of all five orbital elements (a, e, i, Ω, and ω). They were created by propagating the orbits with DSST using the optimal orbital elements. For all but Ω, the resulting element histories were then compared with the optimal (initial values) to create the deviation plots presented below. Since the desired behavior of the ascending node was a fixed rate, rather than a fixed value, the desired history had to be calculated based on a linear relationship that used the desired fixed rate. The actual trajectory was then compared with the calculated trajectory and this is the Ω deviation presented in the figures below. Section A-4-1 contains the decay plots for the elements in the design space of a 50 x 0 zonal field with third-body effects, while Section A-4-2 is the decay plots for the orbit in the presence of full perturbation models. All elements are presented in a J2000 True of Date reference frame.

A-4-1 113:14 Repeat Ground Track Zonal and Third Body Decay Plots

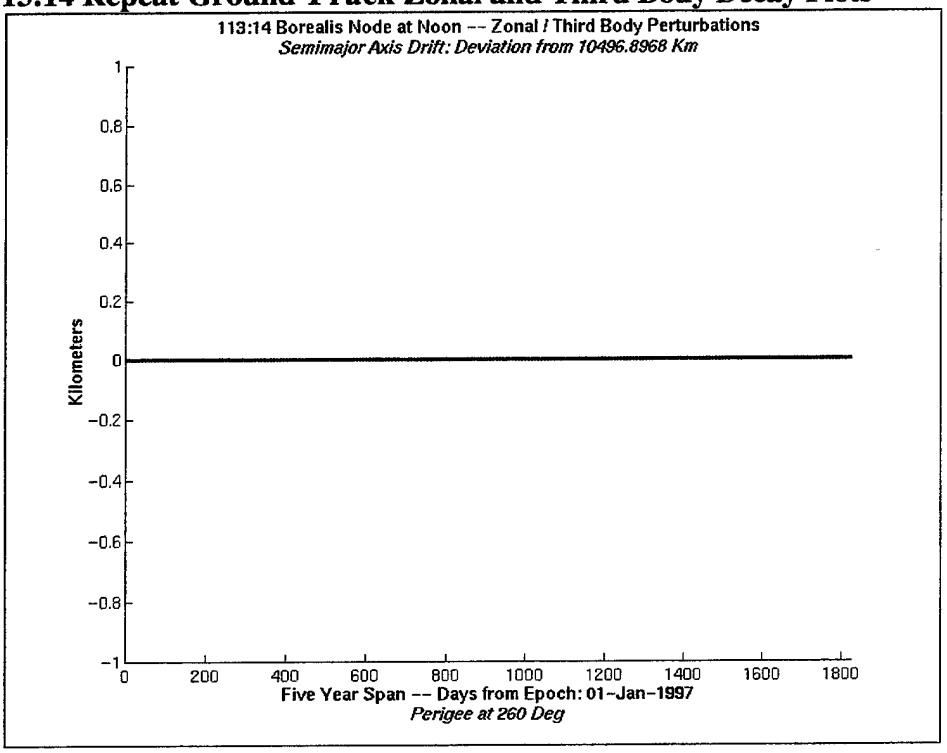


Figure A-13 113:14 SMA Deviation from 10496.8968 km (Zonals/3B Pert.)

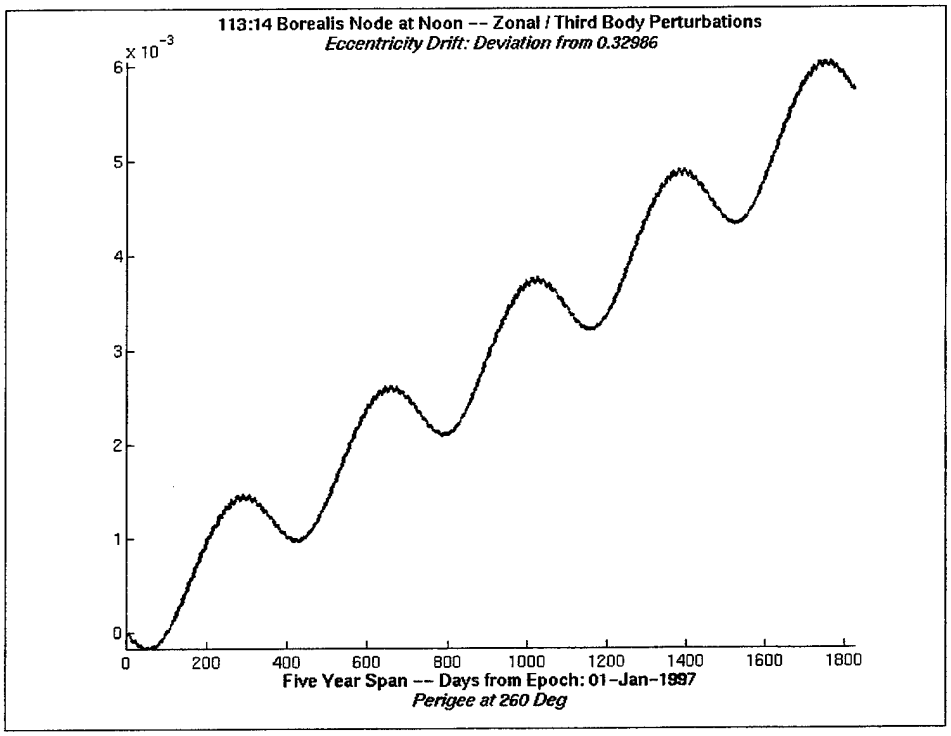


Figure A-14 113:14 Eccentricity Deviation from 0.32986 (Zonals/3B Pert.)

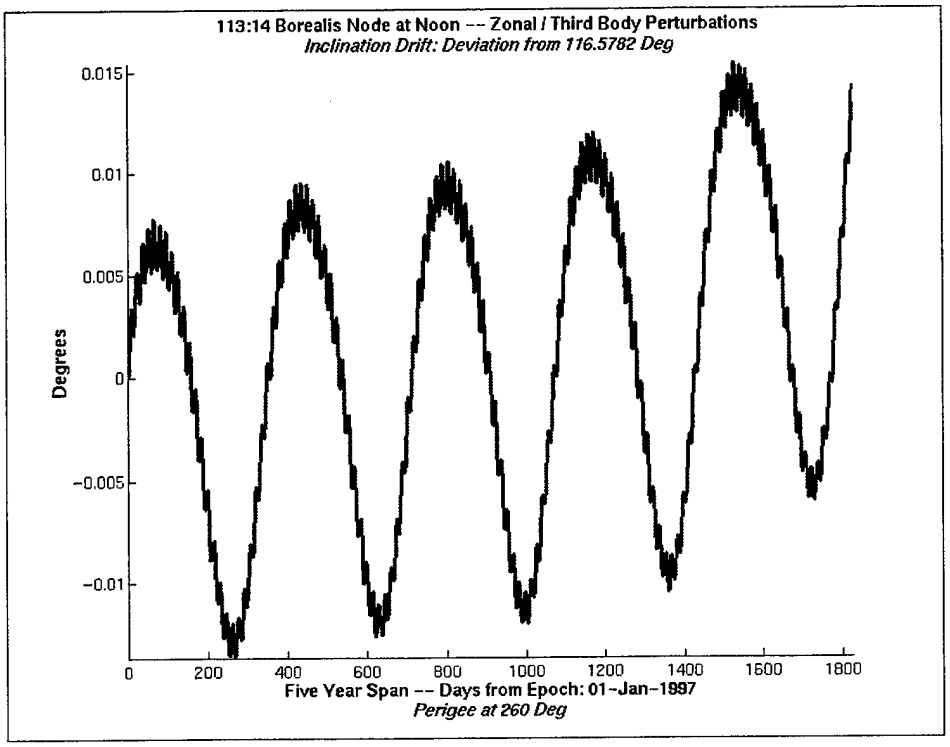


Figure A-15 113:14 Inclination Deviation from 116.5782° (Zonals/3B Pert.)

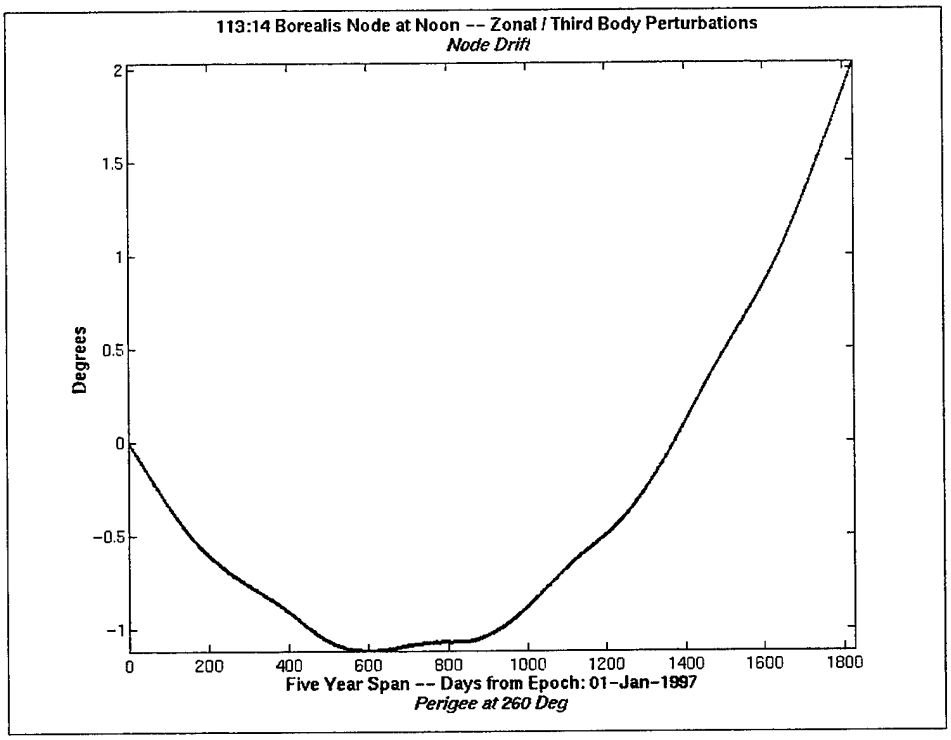


Figure A-16 113:14 Node Deviation from Calculated Values (Zonals/3B Pert.)

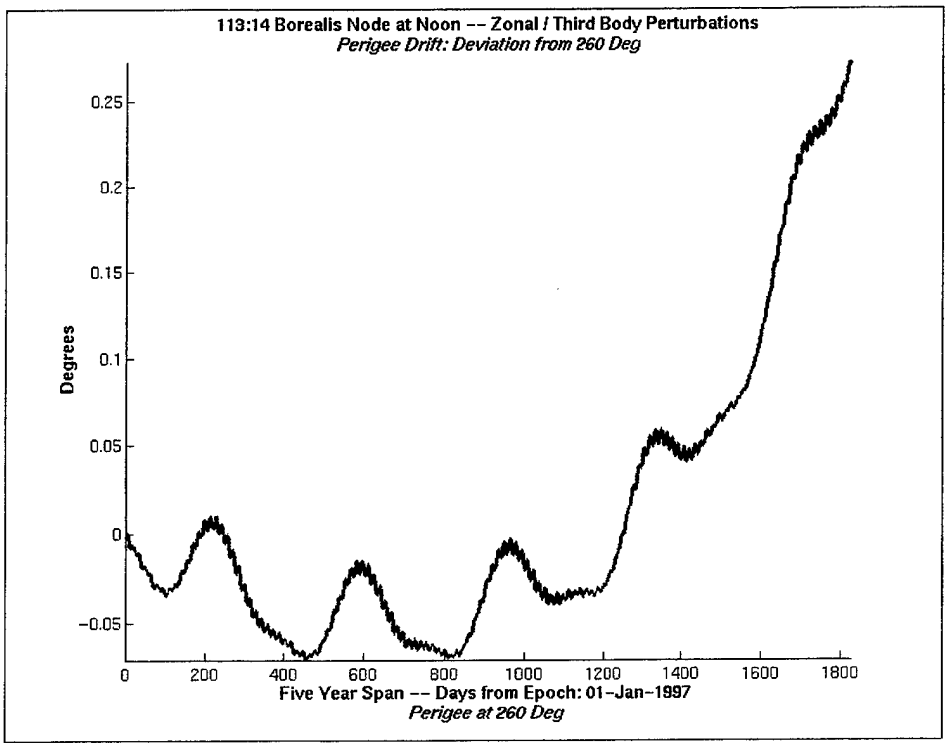


Figure A-17 113:14 Perigee Deviation from 260° (Zonals/3B Pert.)

A-4-2 113:14 Repeat Ground Track Full Perturbation Decay Plots

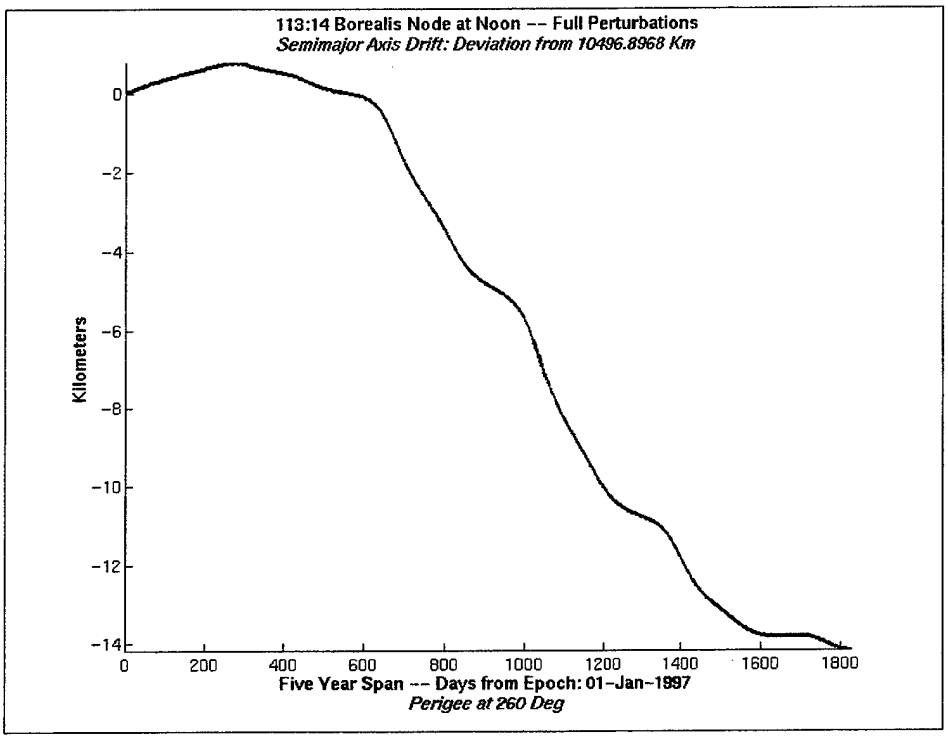


Figure A-18 113:14 Semi-Major Axis Deviation from 10496.8968 km (Full Pert.)

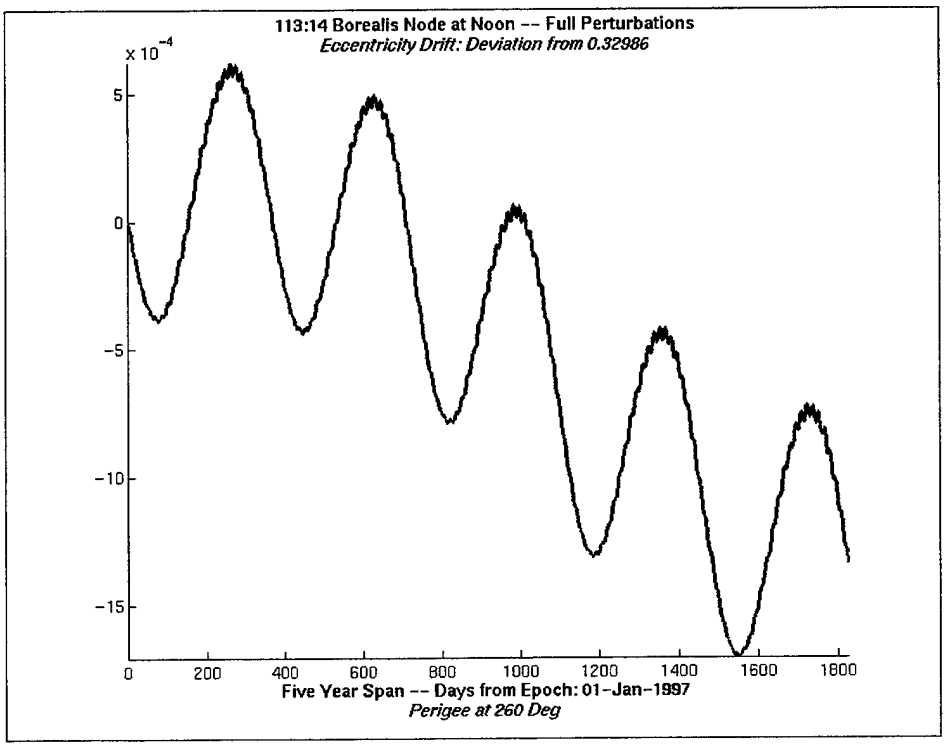


Figure A-19 113:14 Eccentricity Deviation from 0.32986 (Full Pert.)

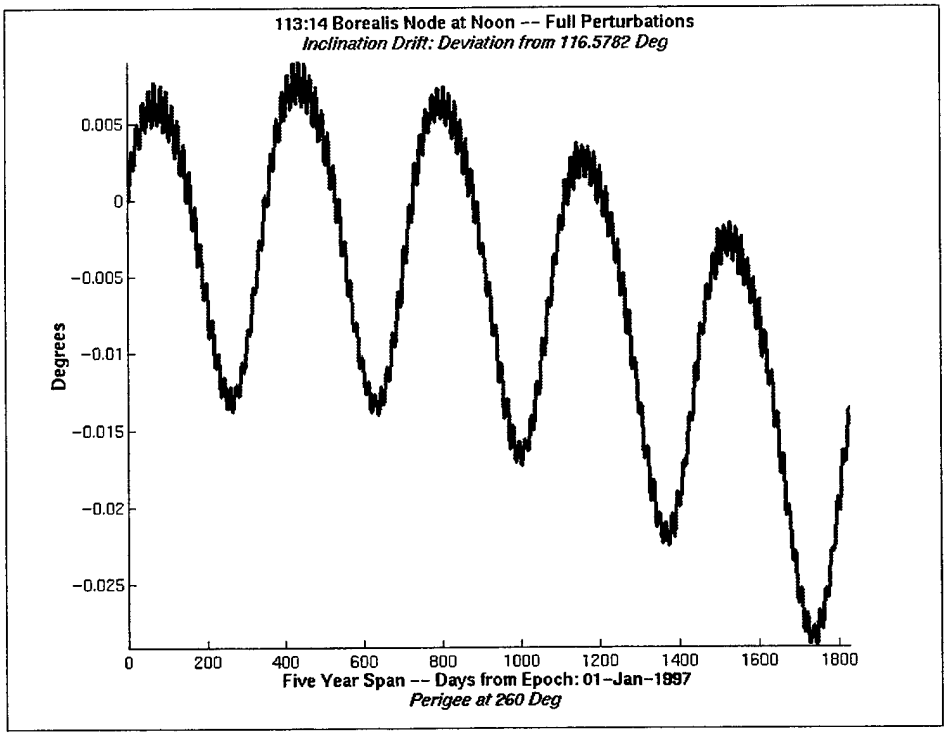


Figure A-20 113:14 Inclination Deviation from 116.5782° (Full Pert.)

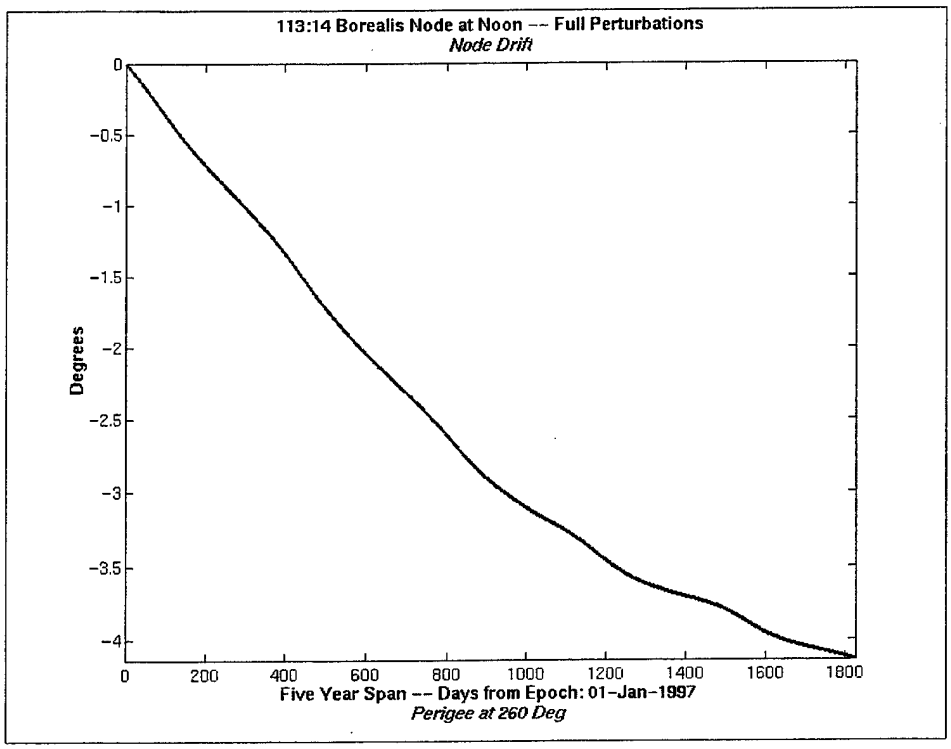


Figure A-21 113:14 Node Deviation from Calculated Values (Full Pert.)

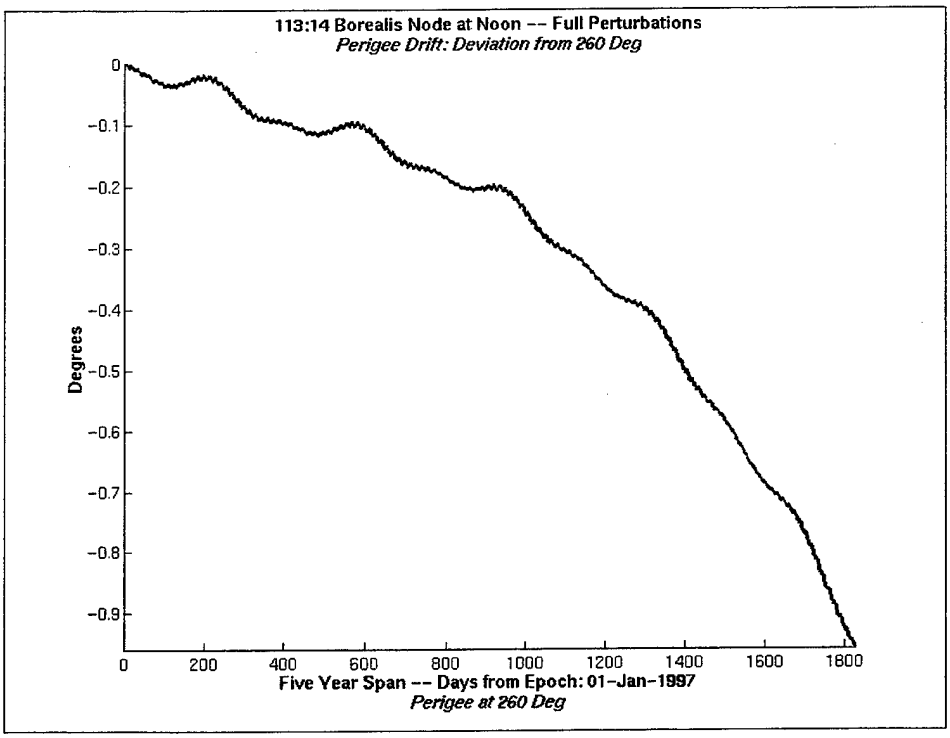


Figure A-22 113:14 Perigee Deviation from 260° (Full Pert.)

Appendix B Gear Array Design Problem Data

This appendix contains data necessary to perform—as well as plots representing the setup and results of—the gear array optimization problem. The various constraint surface plots are first presented. Similar surface plots produced through three different problem formulations follow these plots. FORTRAN code specific to the gear solution is then included, along with corresponding DSST input decks. Finally, the results are presented in three forms: genetic algorithm convergence plots, element decay plots (under zonal and full perturbations), and coverage analysis plots.

B-1 Gear Array Constraint Surface Plots

This first section of Appendix B contains four views of each of the constraint surfaces. These plots were created by setting the scale factors on the other two constraints to zero and plotting the objective function vs. the two solve for variables (a_e and i). All of the plots were created for the 4:5 gear array with an offset of 0 km. Also note that all of the plots were created in a 50 x 0 zonal field over the course of one year, except for the stroboscopic constraint plots. Due to the extremely varying nature of this constraint's surface, the plot resulting from one-year propagations could not be plotted successfully without extremely dense sampling. In order to sample at a more reasonable level, the stroboscopic constraint surface was created using only a one week propagation.

The four views presented for each constraint are a 3-D view, a contour plot or top view, and an edge on view along both the a_e and i axis to show the location of the minima.

B-1-1 APTS Constraint

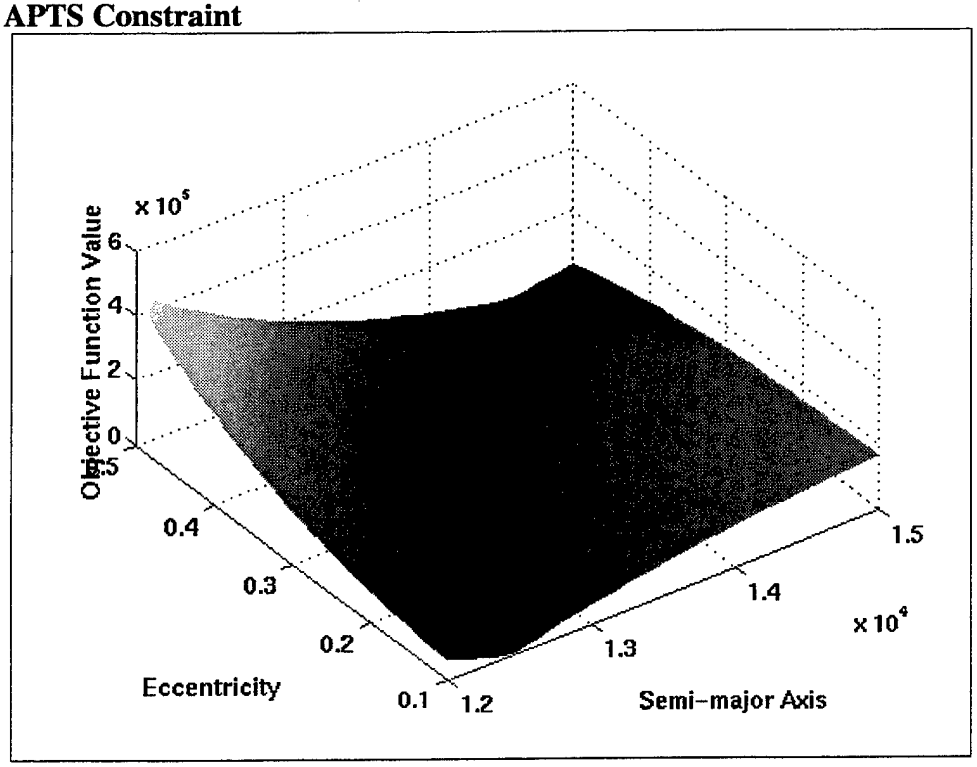


Figure B-1 3-D View of APTS Constraint Surface

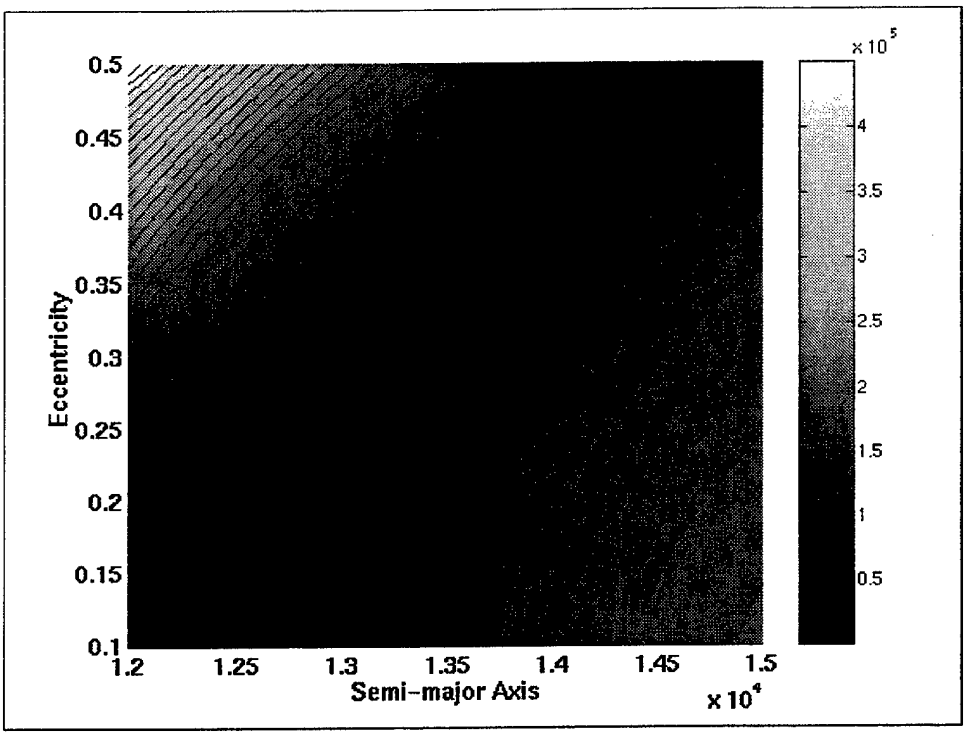


Figure B-2 Contour Plot of APTS Constraint Surface

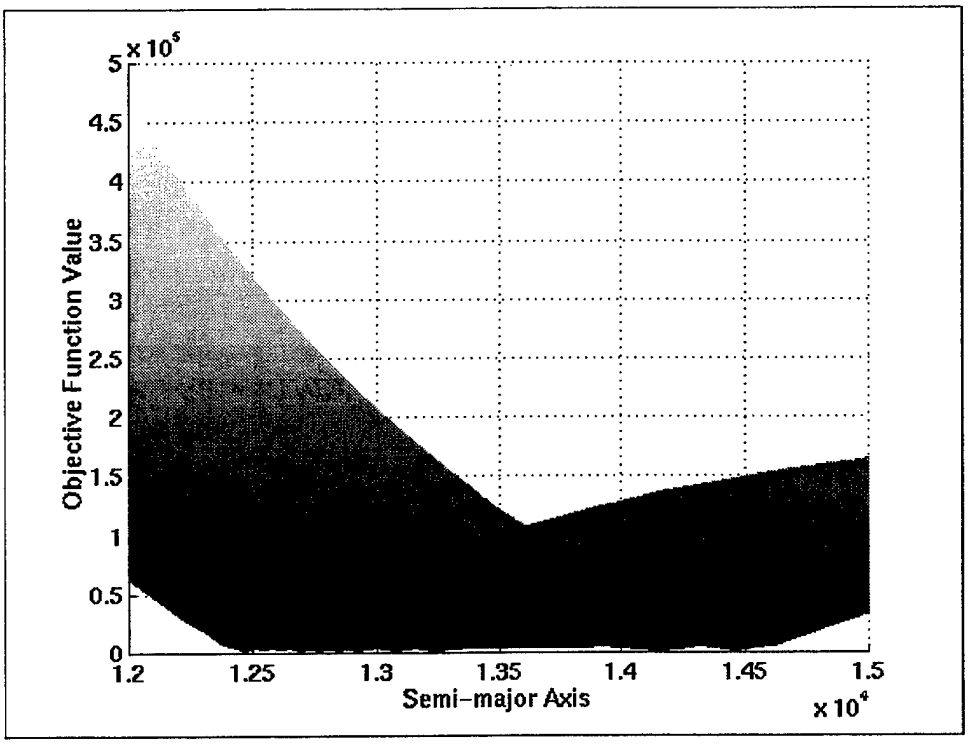


Figure B-3 Semi-Major Axis Edge on View of APTS Constraint Surface

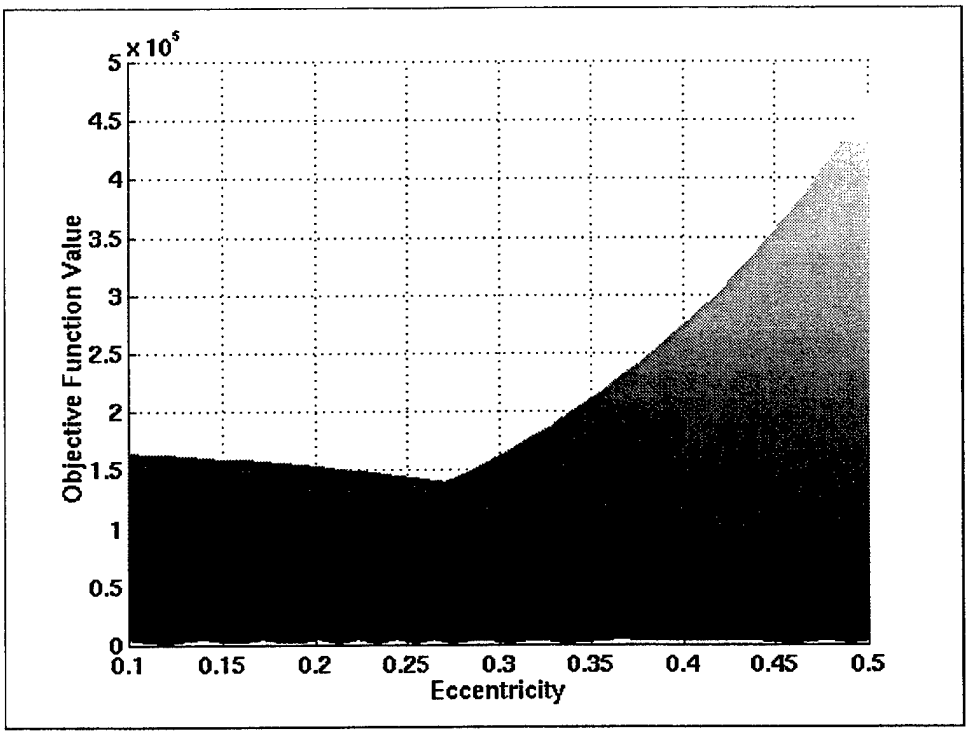


Figure B-4 Eccentricity Edge-on View of APTS Constraint Surface

B-1-2 Stroboscopic Constraint

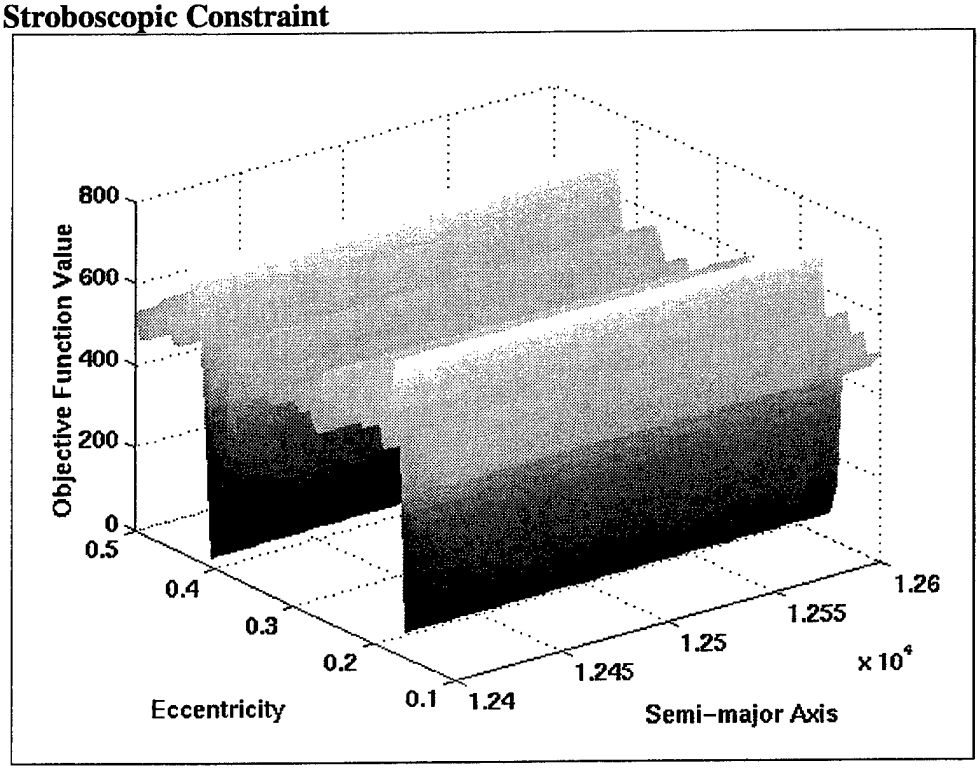


Figure B-5 3-D View of Stroboscopic Constraint Surface

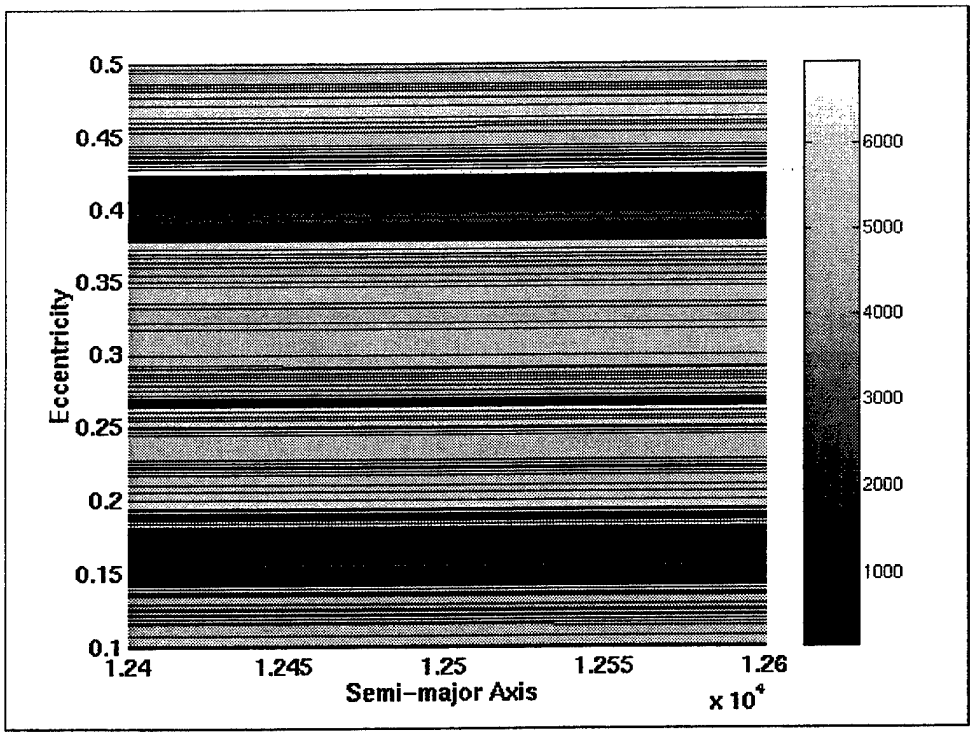


Figure B-6 Contour Plot of Stroboscopic Constraint Surface

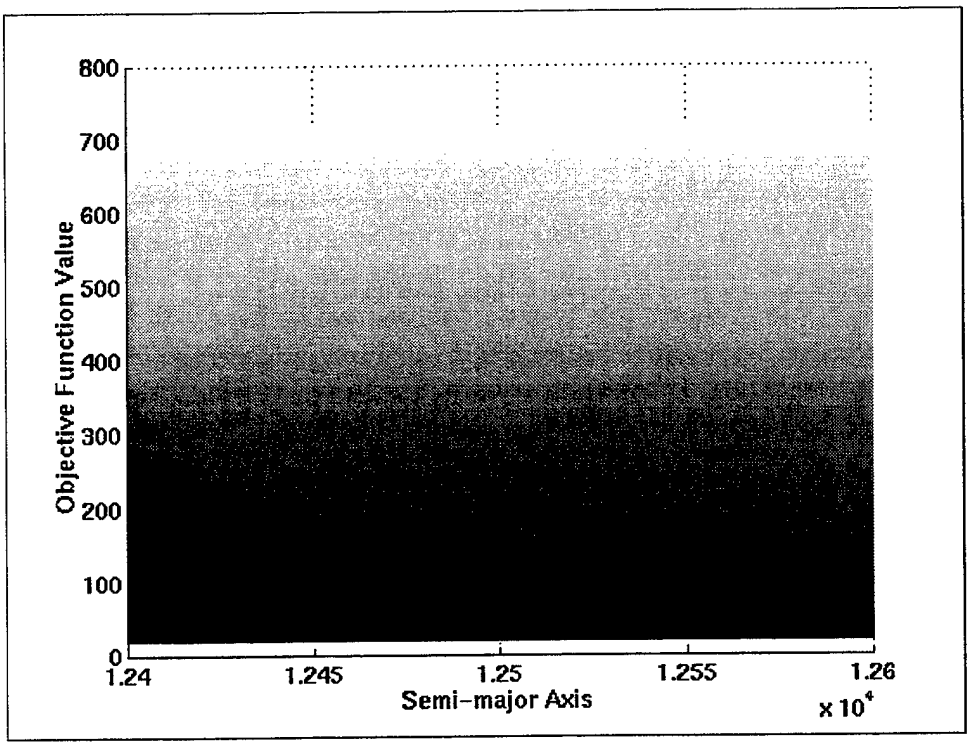


Figure B-7 Semi-Major Axis Edge-on View of Stroboscopic Constraint Surface

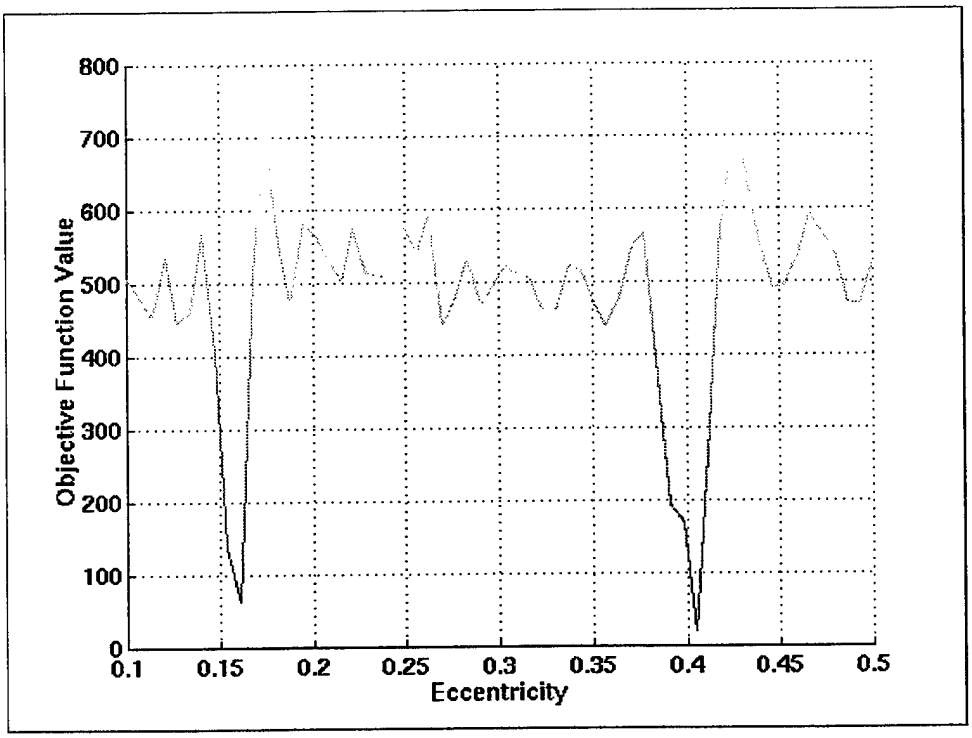


Figure B-8 Eccentricity Edge-on View of Stroboscopic Constraint Surface

B-1-3 Ratio Constraint

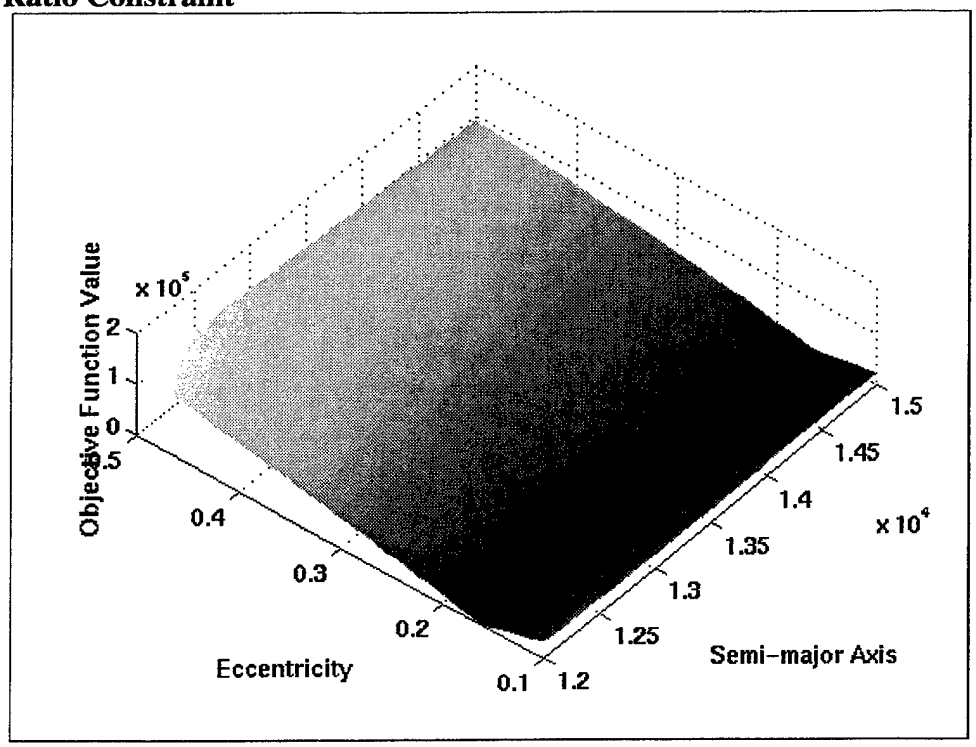


Figure B-9 3-D View of Ratio Constraint Surface

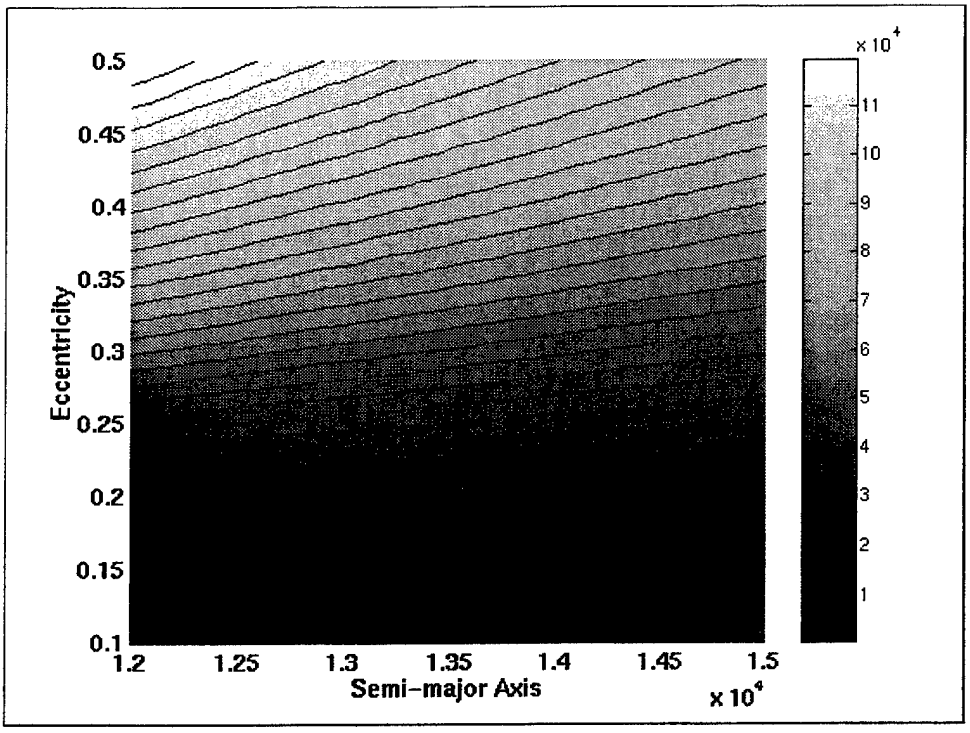


Figure B-10 Contour Plot of Ratio Constraint Surface

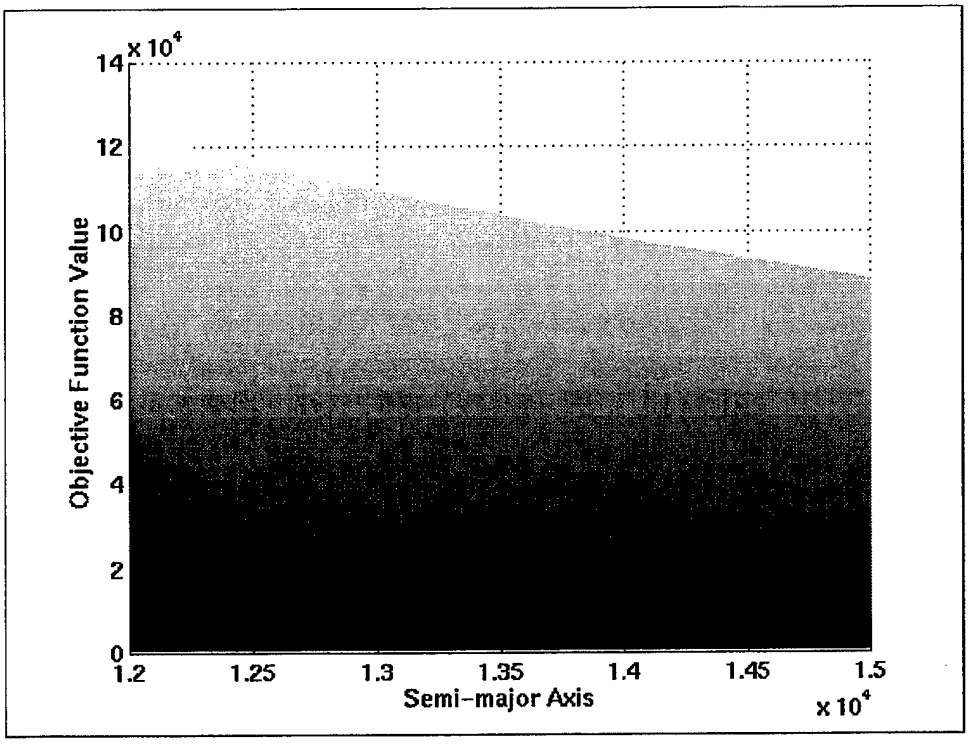


Figure B-11 Semi-Major Axis Edge-on View of Ratio Constraint Surface

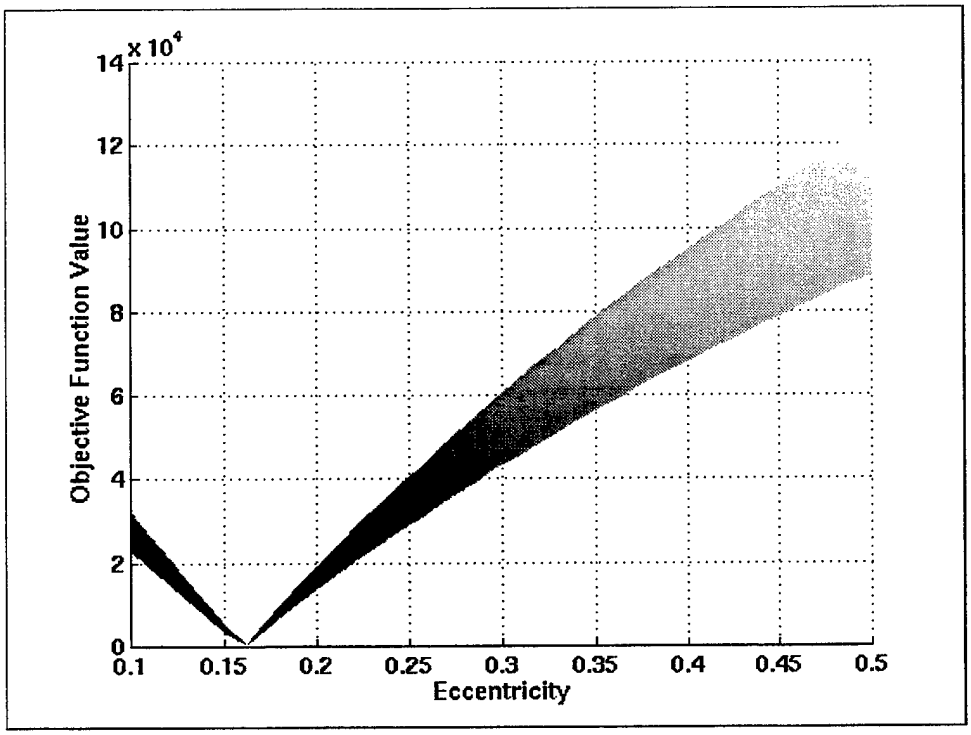


Figure B-12 Eccentricity Edge-on View of Ratio Constraint Surface

B-2 Gear Array Parameterization Options Surface Plots

This section contains the surface plots that resulted from formulating the gear array optimization problem in three different ways. In actuality, there are three solve for variables for the gear array problem: a_e, a_c, and i. However, it is likely that some information (such as desired offset or apogee height) about the gear array will be known prior to the optimization. Therefore, the relationship between the known information and the solve for variables can be taken advantage of and the problem can be reduced to a two variable optimization. This reduction in variables allows for a variety of ways to parameterize the problem. For this study, three of those parameterization methods were analyzed. An important aspect of that analysis was an inspection of the optimization surfaces that resulted from the various parameterizations. The plots contained in this section present that information.

The fixed offset surface is first presented, followed by the fixed circular semi-major axis surface which is then followed by the fixed apogee height surface. As in previous appendices, four views are presented: a 3-D view, a contour plot, and an edge-on view of each axis. Each plot was created via a one year propagation in a 50 x 0 zonals only field.

B-2-1 Offset Method

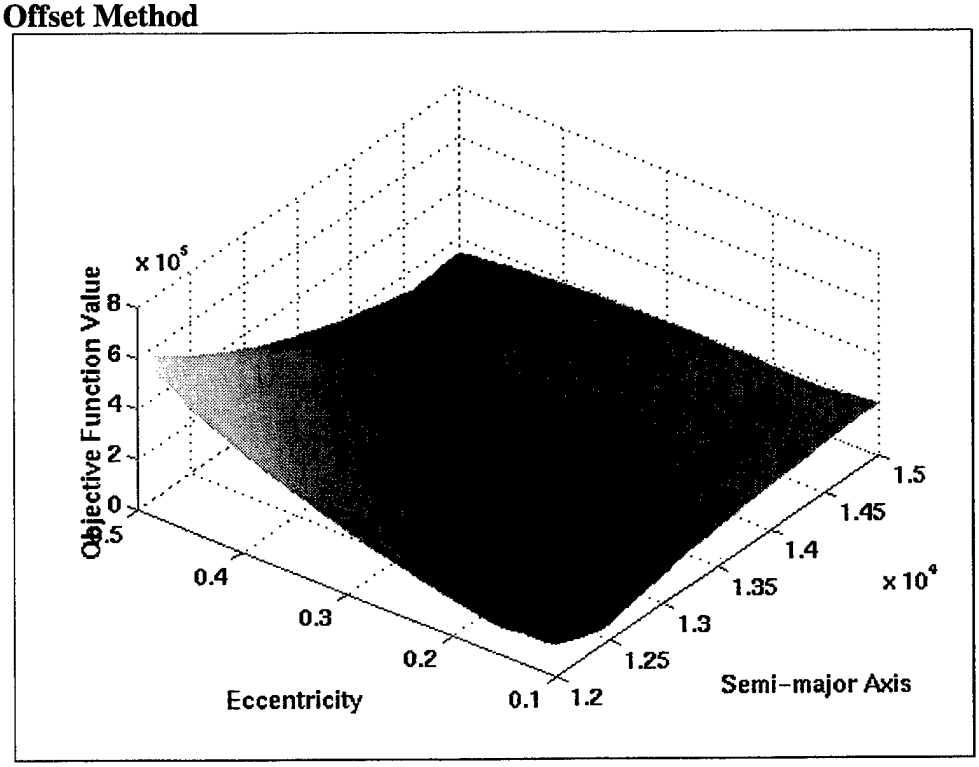


Figure B-13 3-D View of Offset Method Parameterization Surface

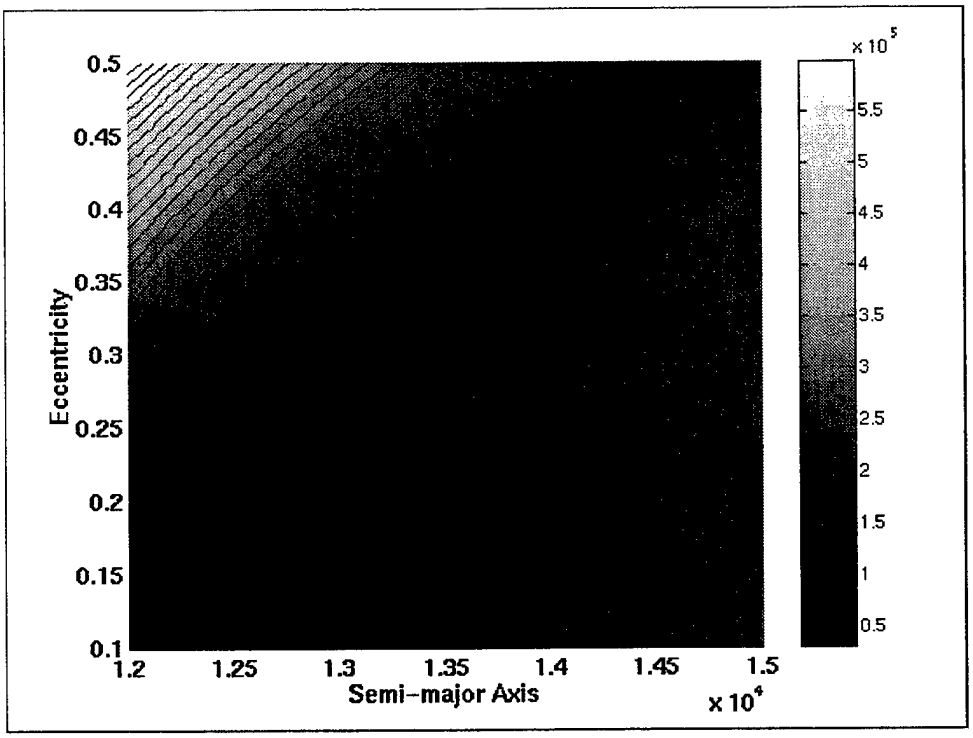


Figure B-14 Contour Plot of Offset Method Parameterization Surface

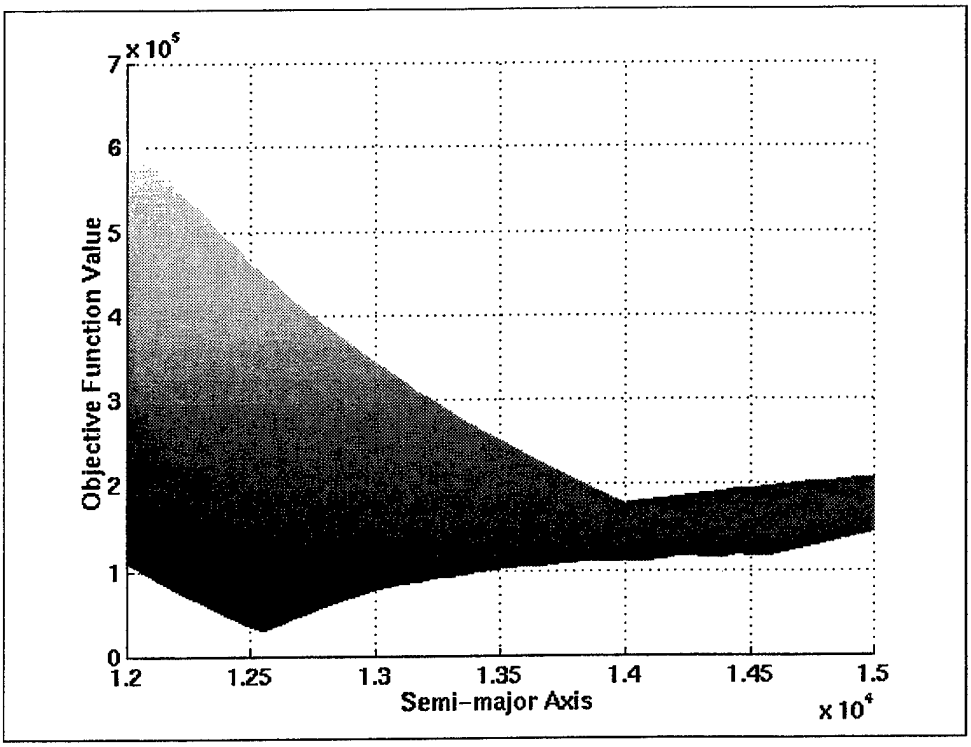


Figure B-15 Semi-Major Axis Edge-on View of Offset Method Parameterization

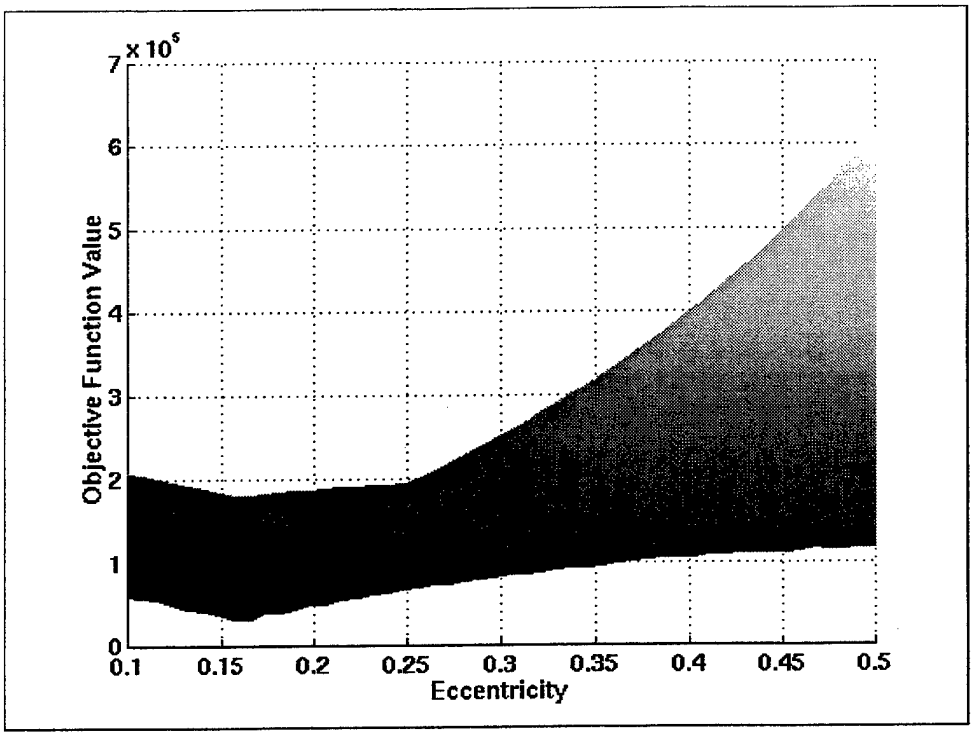


Figure B-16 Eccentricity Edge-on View of Offset Method Parameterization Surface

B-2-2 Fixed Circular Semi-Major Axis Method

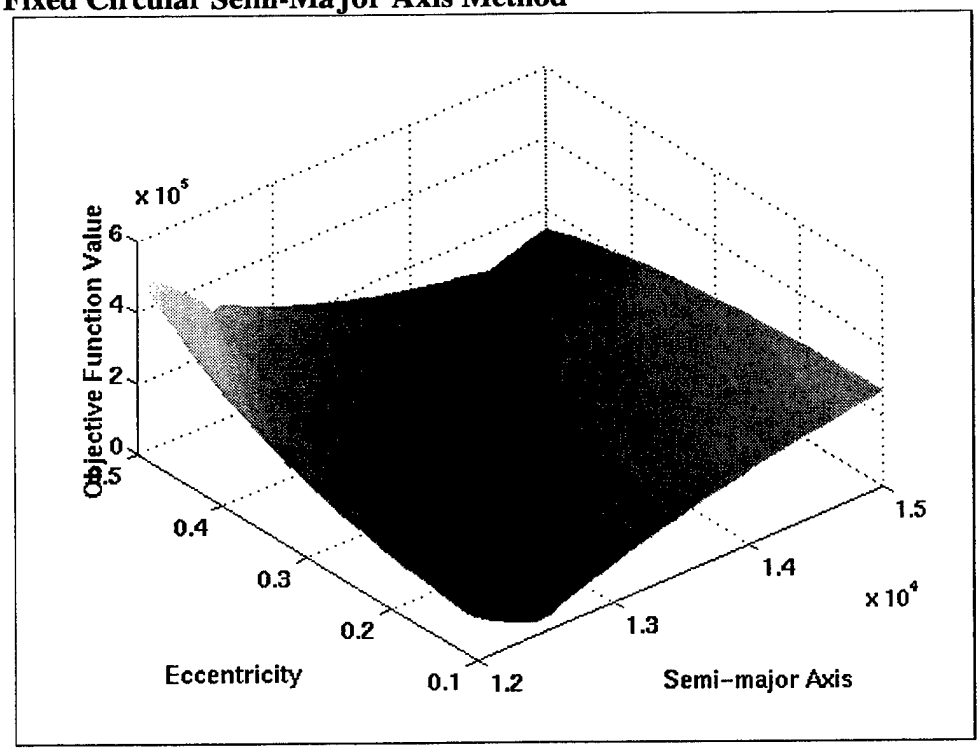


Figure B-17 3-D View of Fixed Circular SMA Parameterization Surface

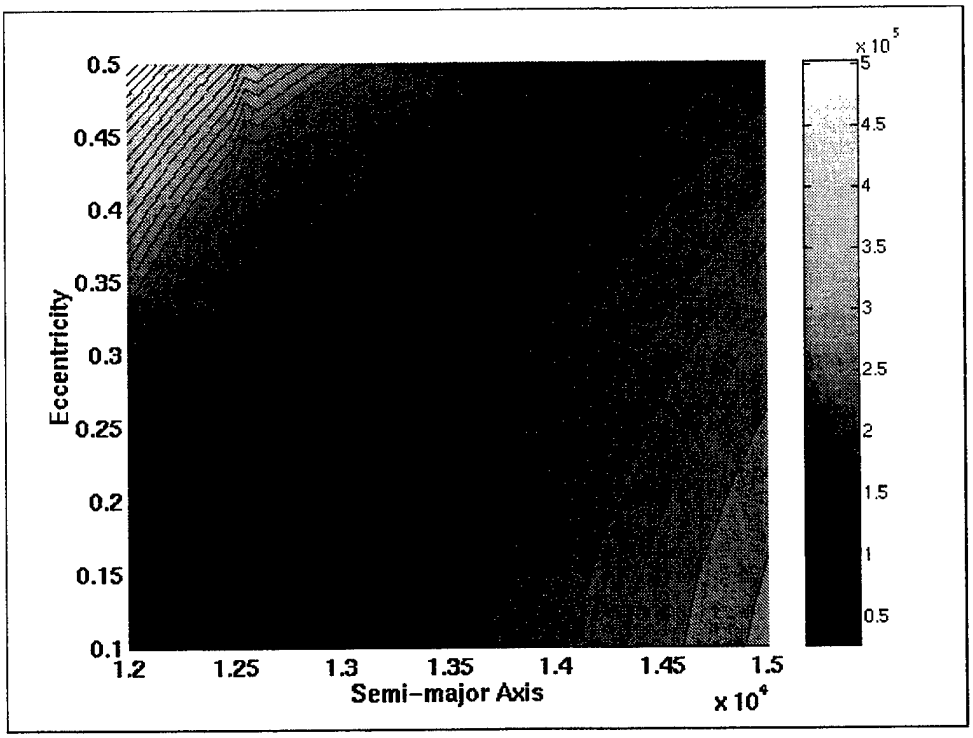


Figure B-18 Contour Plot of Fixed Circular SMA Parameterization Surface

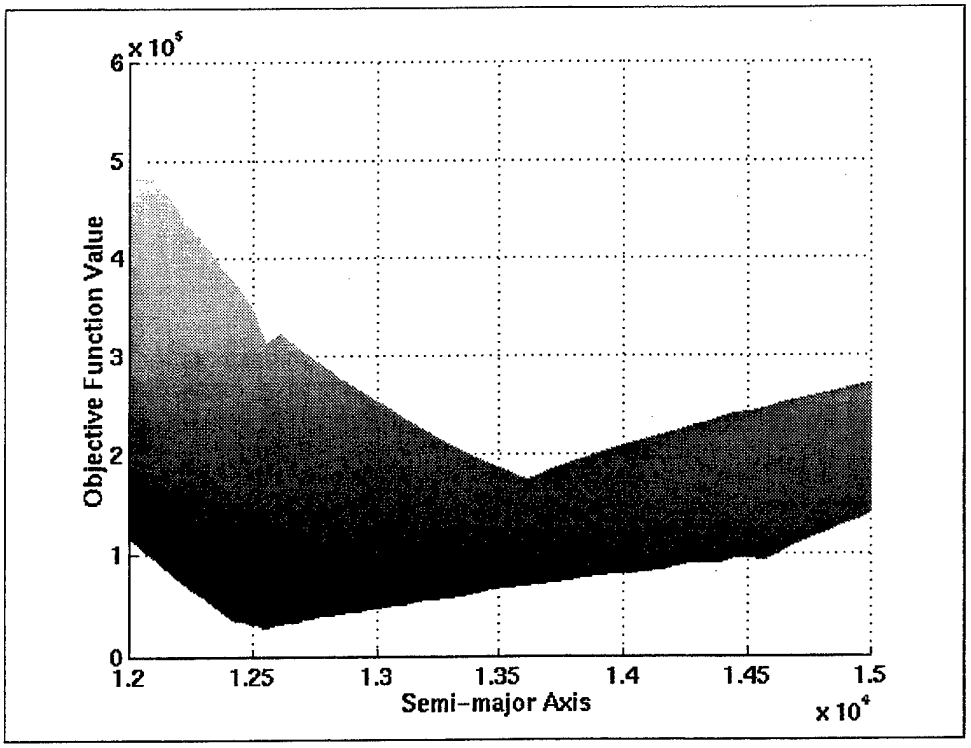


Figure B-19 SMA Edge-on View of Fixed Circular SMA Parameterization

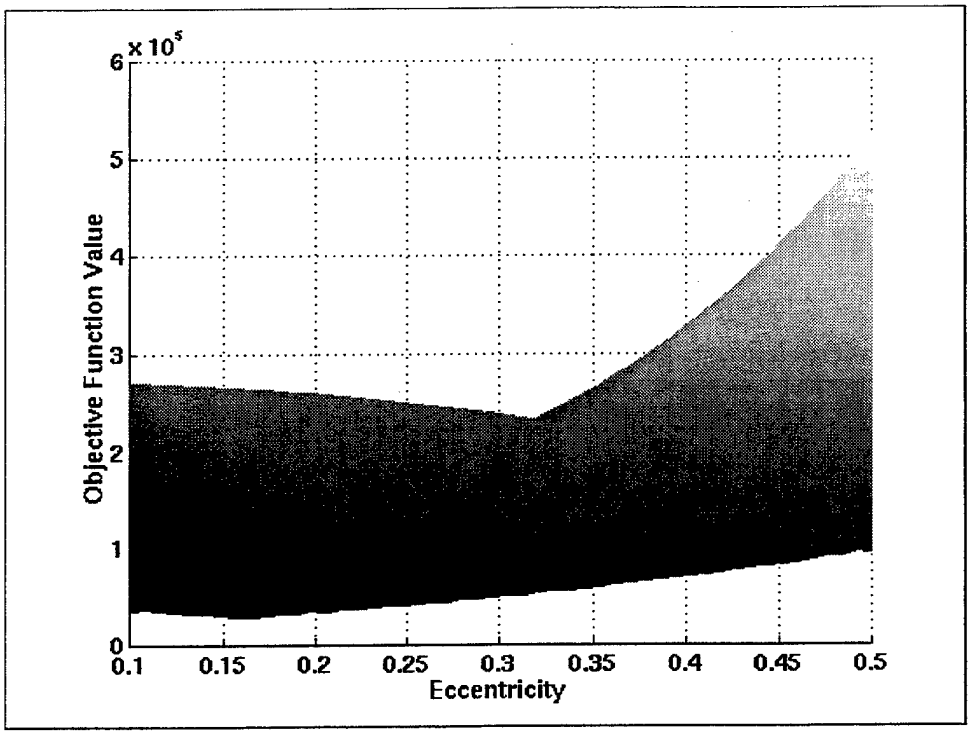


Figure B-20 Eccentricity Edge-On View of Fixed Circular SMA Parameterization

B-2-3 Fixed Apogee Height Method

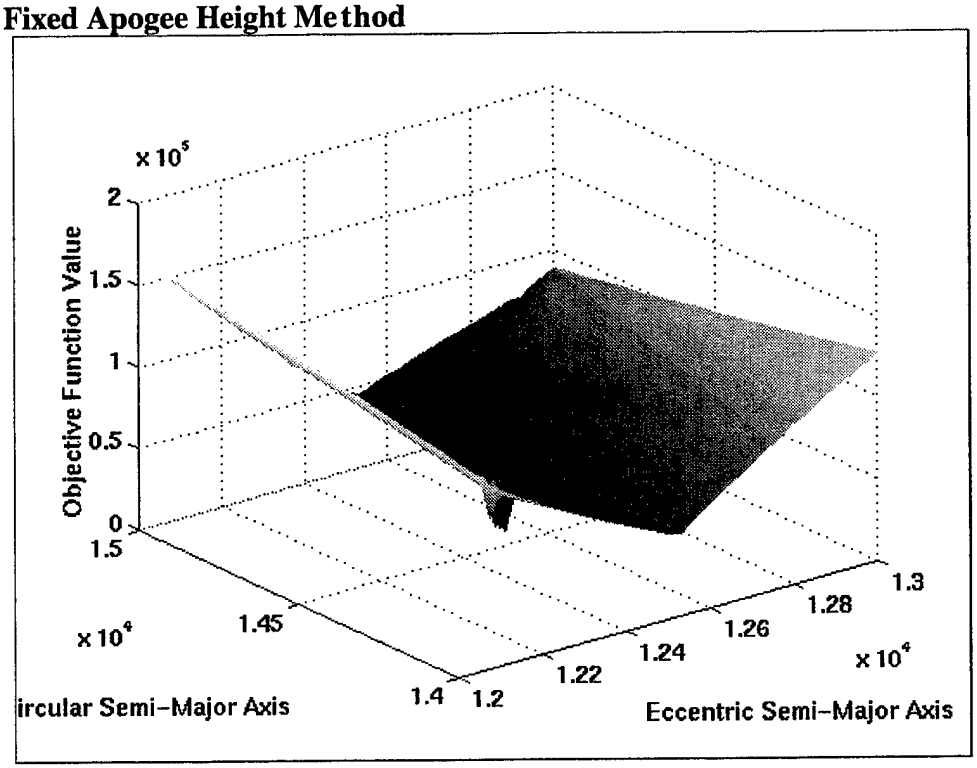


Figure B-21 3-D View of Fixed Apogee Height Parameterization Surface

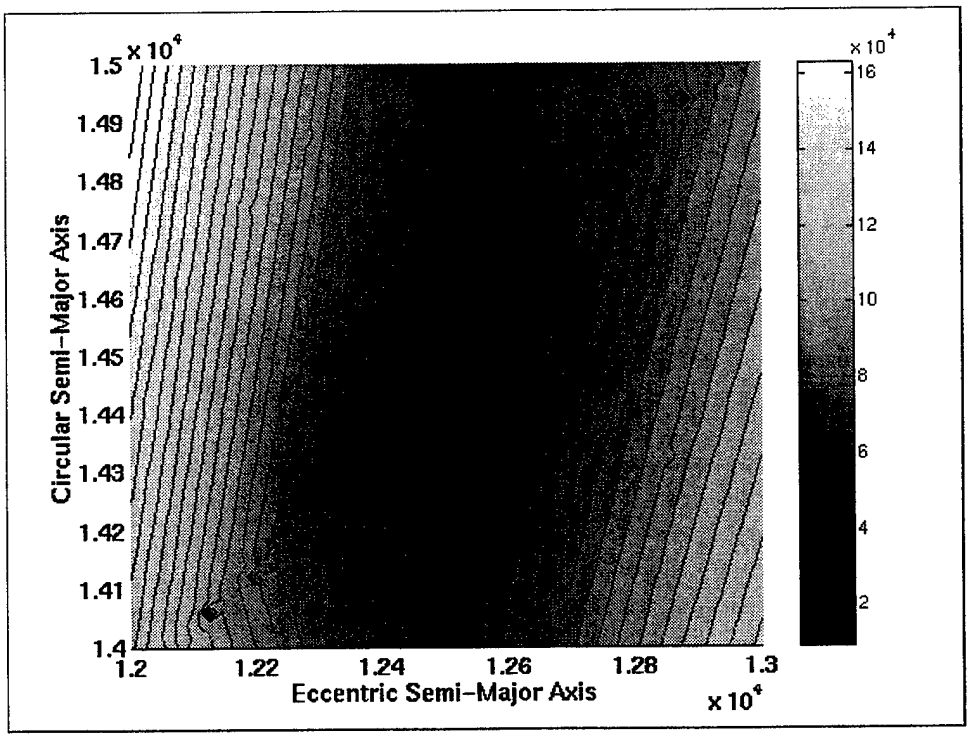


Figure B-22 Contour Plot of Fixed Apogee Height Parameterization Surface

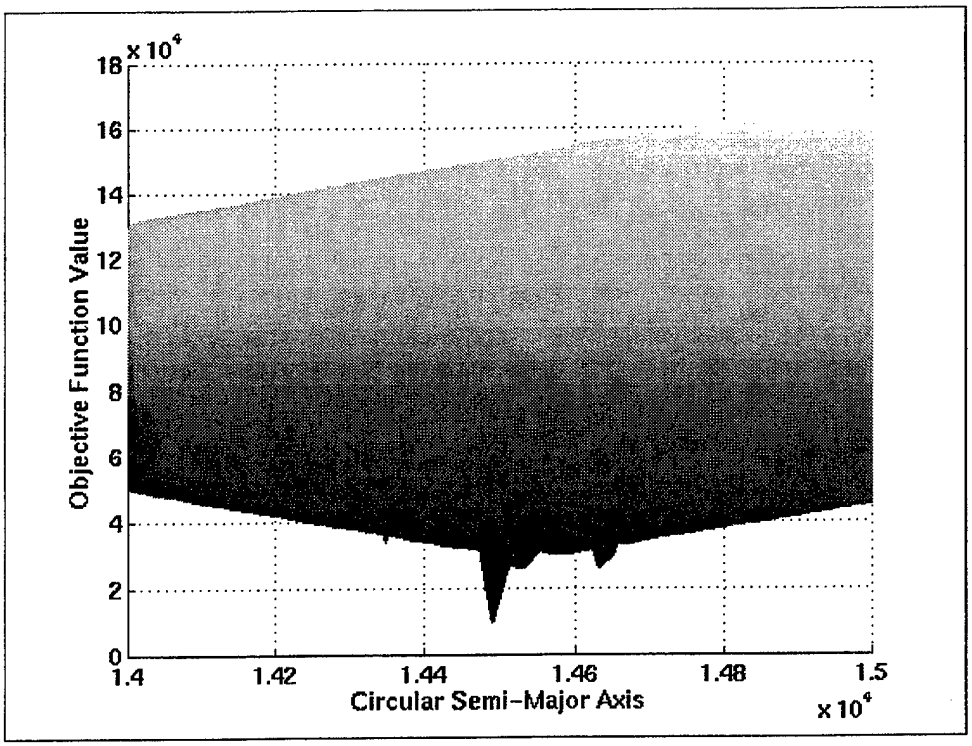


Figure B-23 Circular SMA Edge-on View of Fixed Apogee Height Parameterization

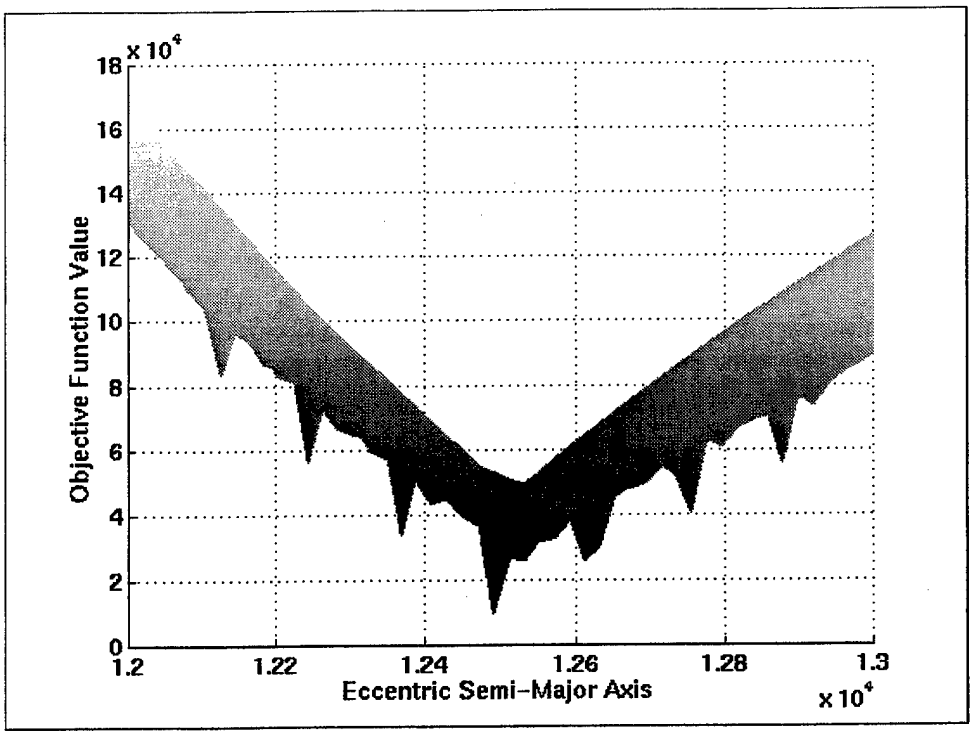


Figure B-24 Eccentric SMA Edge-on View of Fixed Apogee Height Parameterization Surface

B-3 Gear Array Objective Function Code (FUNC)

Included in this section is the objective function code required to perform the optimization of the gear array using PGAPack in conjunction with DSST and MPICH. The code presented here is for the fixed offset parameterization. Similar functions were created for the other two methods of formulating the problem

```
real*8  function func (p)
C-----------------------------------------------------------------------
C
C                              Function Func
C
C  This file contains the objective function for the GEAR fixed offset
C  parameterization optimization problem
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C          Ronald J. Proulx
C          Draper Laboratory
C
C  External Functions:
C      satellite     Link to DSST Orbit Propagator
C
C-----------------------------------------------------------------------

      implicit none

      include 'pmern.h'
      include 'frc.h'
      include 'switch.h'
      include 'matrix.h'
      include 'satelm.h'

C-----  VARIABLE DECLARATIONS ----------------------------------------

C----- Define variables for call to Satellite -----------------

      integer*4       satellite
      external        satellite

      integer*4       max_list_length
      parameter      (max_list_length  = 1)

      integer*4       status
      integer*4       burn_number
      Integer*4       iatmos_preburn / 1 /
      integer*4       iatmos_postburn / 1 /
      integer*4       month,hour,minute
      real*8          day(maxvar)

      real*8          time
      real*8          burn_delta_v(4,max_list_length)
      REAL*8          RHO_ONE_HIGH      / 0.0 D0 /
      REAL*8          RHO_ONE_LOW       / 0.0 D0 /
```

```
      real*8          epoch_ymd
      real*8          epoch_hms
      real*8          posvel(6)
      real*8          elements (17)

      character*(6)   name
      CHARACTER*(72)  MESSAGE
      CHARACTER*(12)  FILENAME

C ------- Define Other Variables

      integer*4       i

      real*8          p(*)
      real*8          pi

      real*8          n_apts
      real*8          n_circ
      real*8          radians
      real*8          degrees

      real*8          h
      real*8          k
      real*8          hhkk
      real*8          dhdt
      real*8          dkdt
      real*8          dldt
      real*8          aptsrate_nom
      real*8          aptsrate_cur
      real*8          aptsrate_dif
      real*8          aptsrate_var
      real*8          M_apts
      real*8          P_apts(12000)
      real*8          mag_x
      real*8          UX_apts(12000)
      real*8          UY_apts(12000)
      real*8          UZ_apts(12000)
      real*8          M_circ
      real*8          P_circ
      real*8          gear_ratio_nom
      real*8          gear_ratio_cur
      real*8          gear_ratio_dif
      real*8          gear_ratio_var
      real*8          UX_circ
      real*8          UY_circ
      real*8          UZ_circ
      real*8          gear_phase_nom
      real*8          gear_phase_cur
      real*8          gear_phase_dif
      real*8          gear_phase_var
      real*8          sma_circ
      real*8          offset
      real*8          numdays
      real*8          height_apts


      parameter   (  radians        = 57.295779513082321    D00 )
      parameter   (  degrees        =  0.017453292519943296 D00 )


      common /offsetcom/ offset,sma_circ,numdays,height_apts


C     ********** BEGIN PROGRAM ********************************
```

```
      pi = 4.0d0*atan2(1.d0,1.d0)

C     Set defaults for request time and burn list

      BURN_NUMBER      = 0
      DO I=1,MAX_LIST_LENGTH
         BURN_DELTA_V(1,I) = 0.D0
         BURN_DELTA_V(2,I) = 0.D0
         BURN_DELTA_V(3,I) = 0.D0
         BURN_DELTA_V(4,I) = 0.D0
      END DO

C     Initialize APTS component of Gear Constellation

      name = 'pmerna'
      call initialize_sat (name)

      n_apts = pmern.dp_spare(1)
      n_circ = pmern.dp_spare(2)

      gear_ratio_nom = n_apts/n_circ
      gear_phase_nom = pi/n_circ
      aptsrate_nom   = .98564736d0*degrees/86400.d0
      aptsrate_var = 0.d0

c     Set sma and ecc to input values

      pmern.els_kepler(1) = p(1)
      pmern.els_kepler(2) = p(2)

      time = 0.d0
      i    = 1

      DO WHILE (time .lt.numdays*86400.d0)

         STATUS = SATELLITE ( name, TIME,
     2         BURN_DELTA_V,    BURN_NUMBER,
     3         IATMOS_PREBURN, IATMOS_POSTBURN,
     4         RHO_ONE_HIGH,    RHO_ONE_LOW,
     5         EPOCH_YMD,       EPOCH_HMS,
     6         POSVEL,          ELEMENTS,
     7         MESSAGE,         FILENAME )


C     Compute the period of the anomalistic period of the satellite

         h = elements(7)
         k = elements(8)
         hhkk = h*h+k*k
         dhdt = elements(13)
         dkdt = elements(14)
         dldt = elements(17)*degrees ! radians/sec

         aptsrate_cur = (k*dhdt-h*dkdt)/hhkk ! radians/sec
         aptsrate_dif = aptsrate_cur - aptsrate_nom ! radians/sec
         aptsrate_var = aptsrate_var + abs(aptsrate_dif) ! radians/sec

         M_apts     = dldt - aptsrate_cur ! radians/sec
         P_apts (i) = 2.d0*pi / M_apts ! seconds

         mag_x = sqrt(posvel(1)**2+posvel(2)**2+posvel(3)**2) ! Km

         UX_apts(i) = posvel(1)/mag_x
```

```
         UY_apts(i) = posvel(2)/mag_x
         UZ_apts(i) = posvel(3)/mag_x

         time = time + n_circ*P_apts(i)
         i = i+1
      end do

      name = 'pmernc'

C     Initialize circular component of Gear Constellation

      call initialize_sat(name)

      pmern.els_kepler(1) = p(1)*(1.d0 + p(2)) - offset


      gear_ratio_var = 0.d0
      gear_phase_var = 0.d0

      open(unit=39)
      open(unit=40)

      time = 0.d0
      i = 1
      DO While (time.lt. numdays*86400.d0)

         STATUS = SATELLITE ( name, TIME,
     2         BURN_DELTA_V,    BURN_NUMBER,
     3         IATMOS_PREBURN,  IATMOS_POSTBURN,
     4         RHO_ONE_HIGH,    RHO_ONE_LOW,
     5         EPOCH_YMD,       EPOCH_HMS,
     6         POSVEL,          ELEMENTS,
     7         MESSAGE,         FILENAME )

         h = elements(7)
         k = elements(8)
         hhkk = h*h+k*k
         dhdt = elements(13)
         dkdt = elements(14)
         dldt = elements(17)*degrees ! radians/sec

         if (hhkk.gt.0.d0) then
            M_circ =  dldt - (k*dhdt-h*dkdt)/hhkk ! radians/sec
         else
            M_circ =  dldt      ! radians/sec
         endif
         P_circ = 2.d0*pi/M_circ

         gear_ratio_cur = P_apts(i)/P_circ
         gear_ratio_dif = gear_ratio_cur - gear_ratio_nom
         gear_ratio_var = gear_ratio_var + abs(gear_ratio_dif)

         mag_x = sqrt(posvel(1)**2+posvel(2)**2+posvel(3)**2) ! Km

         UX_circ = posvel(1)/mag_x
         UY_circ = posvel(2)/mag_x
         UZ_circ = posvel(3)/mag_x

         gear_phase_cur =
     *         acos(
     *         UX_circ*UX_apts(i) +
     *         UY_circ*UY_apts(i) +
     *         UZ_circ*UZ_apts(i)
     *         )*radians
```

```
      gear_phase_dif = gear_phase_cur - gear_phase_nom*radians

      gear_phase_var = gear_phase_var + abs(gear_phase_dif)
      time = time + n_circ*P_apts(i)
      i = i+1

    ENDDO

    close(39)
    close(40)

    func =
   *     10.d0*gear_phase_var
   *     + 1000.0d0*gear_ratio_var
   *     + 1000.0d0*aptsrate_var*86400.d0*radians ! degrees/day

    return
    end
```

B-4 Gear Array Input Decks

Also important to the computer implementation of the gear optimization are the DSST input decks that specify which parameters and perturbations should be used in the propagation of the orbits. Since the gear array is composed of two separate orbital arrays (a circular and elliptical component), two separate input decks were required. The input deck for the circular component of the 5:6 8050 km apogee height gear array is first presented followed the by elliptical component input deck for the same 5:6 array. Proper flag and keyword values were determined from personal communications with Dr. Ronald Proulx, The Charles Stark Draper Laboratory.

B-4-1 Gear Array Circular Orbit DSST Input Deck

```
C
C
C      PMEF FILE FOR 5:6 GEAR ARRAY CIRCULAR ORBIT
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.1997010100000000D+08 PME_DATE          1
   0.0000000000000000D+06 PME_TIME          2
   0.1414357000000000D+05 ELS_KEP(1)        3
   0.0000000000000000D-00 ELS_KEP(2)        4
   0.0000000000000000D-00 ELS_KEP(3)        5
   0.0000000000000000D+00 ELS_KEP(4)        6
   0.1010000000000000D+03 ELS_KEP(5)        7
   0.1500000000000000D+03 ELS_KEP(6)        8
   0.0000000000000000D+00 ELS_EQUIN(1)      9
   0.0000000000000000D+00 ELS_EQUIN(2)     10
```

```
0.0000000000000000D+00 ELS_EQUIN(3)    11
0.0000000000000000+00 ELS_EQUIN(4)     12
0.0000000000000000D+00 ELS_EQUIN(5)    13
0.0000000000000000D+00 ELS_EQUIN(6)    14
0.0000000000000000D+00 POSVEL(1)       15
0.0000000000000000D+00 POSVEL(2)       16
0.0000000000000000D+00 POSVEL(3)       17
0.0000000000000000D+00 POSVEL(4)       18
0.0000000000000000D+00 POSVEL(5)       19
0.0000000000000000D+00 POSVEL(6)       20
0.2000000000000000D+01 PME_CD          21
0.0000000000000000D+00 PME_RHO_ONE     22
0.5000000000000000D-04 SMA_SIGMA       23
0.5000000000000000D-04 INC_SIGMA       24
0.5000000000000000D-04 ASC_SIGMA       25
0.7000000000000000D+03 PME_SCMASS      26
0.2500000000000000D-04 PME_SCAREA      27
0.8640000000000000D+05 PME_STEPSIZE    28
0.5000000000000000D+01 DP_SPARE1       29
0.6000000000000000D+01 DP_SPARE2       30
0.0000000000000000D+00 DP_SPARE3       31
0.0000000000000000D+00 DP_SPARE4       32
0.0000000000000000D+00 DP_SPARE5       33
0.0000000000000000D+00 DP_SPARE6       34
         1             PME_RETRO       35
        12             PME_KEP_SYS     36
        12             POS_VEL_SYS     37
         1             GEN_METHOD      38
         1             ATMOS_MODEL     39
    840401             JACRB_DATE      40
       123             JACRB_SSS       41
    840401             SLP1950_DATE    42
       456             SLP1950_SSS     43
    840401             SLPTOD_DATE     44
       789             SLPTOD_SSS      45
    840401             TIMECF_DATE     46
       123             TIMECF_SSS      47
         2             HARRIS_MODEL    48
        10             POTNTL_MODEL    49
        50             PME_NMAX        50
         0             PME_MMAX        51
         1             PME_IZONAL      52
         1             PME_IJ2J2       53
         0             PME_NMAXRS      54
         0             PME_MMAXRS      55
         3             PME_ITHIRD      56
         2             PME_INDDRG      57  2 = DRAG OFF
         2             PME_ISZAK       58
         2             PME_INDSOL      59  2 = SOLRAD OFF
         2             PME_JSHPER      60
         2             PME_JZONAL      61
         2             PME_JMDALY      62
         2             PME_INP_TYPE    63
        12             PME_EQUI_SYS    64
        11             INTEG_FRAME     65
        19             OUTPUT_FRAME    66
         0             PME_NSTATE      67
         1             PME_SPSHPER     68
         2             PME_KSPCF       69
         4             PME_INDSET      70
         0             INT_SPARE1      71
         0             INT_SPARE2      72
         0             INT_SPARE3      73
         0             INT_SPARE4      74
```

```
           0               INT_SPARE5      75
           0               INT_SPARE6      76
           0               INT_SPARE7      77
           0               INT_SPARE8      78
           0               INT_SPARE9      79
           0               INT_SPARE10     80
```

B-4-2 5:6 Gear Array APTS Elliptical Orbit DSST Input Deck

```
C
C
C      PMEF FILE FOR 5:6 GEAR ARRAY ELLIPTICAL ORBIT
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.1997010100000000D+08 PME_DATE         1
   0.0000000000000000D+06 PME_TIME         2
   0.1252737131888390D+05 ELS_KEP(1)       3
   0.1517200478453748D+00 ELS_KEP(2)       4
   0.0000000000000000D+00 ELS_KEP(3)       5
   0.0000000000000000D+00 ELS_KEP(4)       6
   0.1010000000000000D+03 ELS_KEP(5)       7
   0.1800000000000000D+03 ELS_KEP(6)       8
   0.0000000000000000D+00 ELS_EQUIN(1)     9
   0.0000000000000000D+00 ELS_EQUIN(2)    10
   0.0000000000000000D+00 ELS_EQUIN(3)    11
   0.0000000000000000+00 ELS_EQUIN(4)     12
   0.0000000000000000D+00 ELS_EQUIN(5)    13
   0.0000000000000000D+00 ELS_EQUIN(6)    14
   0.0000000000000000D+00 POSVEL(1)       15
   0.0000000000000000D+00 POSVEL(2)       16
   0.0000000000000000D+00 POSVEL(3)       17
   0.0000000000000000D+00 POSVEL(4)       18
   0.0000000000000000D+00 POSVEL(5)       19
   0.0000000000000000D+00 POSVEL(6)       20
   0.2000000000000000D+01 PME_CD          21
   0.0000000000000000D+00 PME_RHO_ONE     22
   0.5000000000000000D-04 SMA_SIGMA       23
   0.5000000000000000D-04 INC_SIGMA       24
   0.5000000000000000D-04 ASC_SIGMA       25
   0.7000000000000000D+03 PME_SCMASS      26
   0.2500000000000000D-04 PME_SCAREA      27
   0.8640000000000000D+05 PME_STEPSIZE    28
   0.5000000000000000D+01 DP_SPARE1       29
   0.6000000000000000D+01 DP_SPARE2       30
   0.0000000000000000D+00 DP_SPARE3       31
   0.0000000000000000D+00 DP_SPARE4       32
   0.0000000000000000D+00 DP_SPARE5       33
   0.0000000000000000D+00 DP_SPARE6       34
          1                PME_RETRO       35
         12                PME_KEP_SYS     36
         12                POS_VEL_SYS     37
          1                GEN_METHOD      38
          1                ATMOS_MODEL     39
     840401                JACRB_DATE      40
        123                JACRB_SSS       41
     840401                SLP1950_DATE    42
        456                SLP1950_SSS     43
     840401                SLPTOD_DATE     44
        789                SLPTOD_SSS      45
     840401                TIMECF_DATE     46
        123                TIMECF_SSS      47
          2                HARRIS_MODEL    48
         10                POTNTL_MODEL    49
```

```
50              PME_NMAX        50
 0              PME_MMAX        51
 1              PME_IZONAL      52
 1              PME_IJ2J2       53
 0              PME_NMAXRS      54
 0              PME_MMAXRS      55
 3              PME_ITHIRD      56
 2              PME_INDDRG      57  2 = DRAG OFF
 2              PME_ISZAK       58
 2              PME_INDSOL      59  2 = SOLRAD OFF
 2              PME_JSHPER      60
 2              PME_JZONAL      61
 2              PME_JMDALY      62
 2              PME_INP_TYPE    63
12              PME_EQUI_SYS    64
11              INTEG_FRAME     65
19              OUTPUT_FRAME    66
 0              PME_NSTATE      67
 1              PME_SPSHPER     68
 2              PME_KSPCF       69
 4              PME_INDSET      70
 0              INT_SPARE1      71
 0              INT_SPARE2      72
 0              INT_SPARE3      73
 0              INT_SPARE4      74
 0              INT_SPARE5      75
 0              INT_SPARE6      76
 0              INT_SPARE7      77
 0              INT_SPARE8      78
 0              INT_SPARE9      79
 0              INT_SPARE10     80
```

B-5 Gear Array Genetic Algorithm Performance Plots

Contained in this section are the convergence plots of the genetic algorithm for both the 4:5 0 km offset and the 5:6 8050 apogee height optimizations. As the genetic algorithm performed the optimization, the best and average value of the strings in each population were stored and plotted here. The resulting plots demonstrate the fundamental theorem of genetic algorithms: that the average of a population slowly converges to better and better values.

B-5-1 5:6 Gear 8050 km Apogee Case

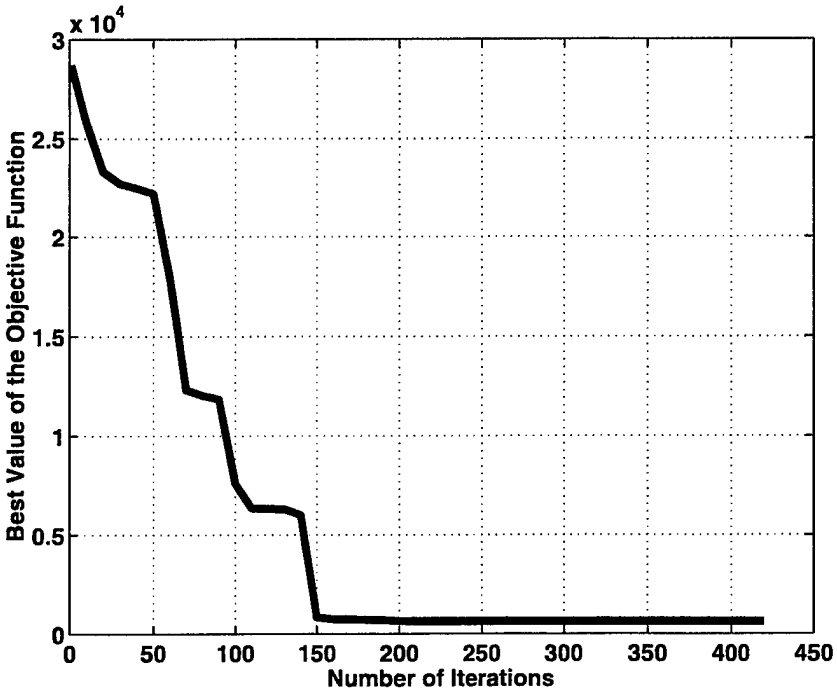


Figure B-25 Best 5:6 Gear Array Objective Function Value Convergence

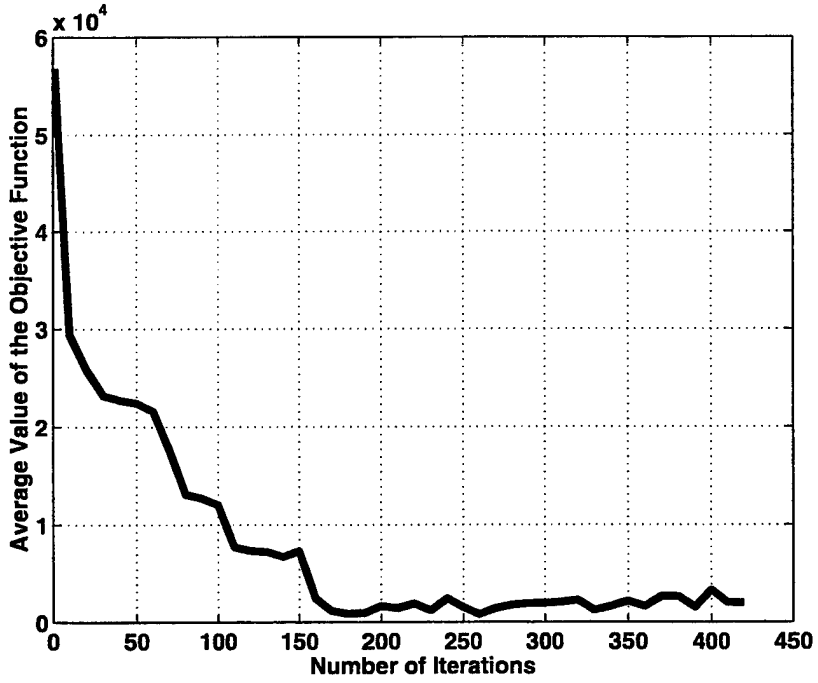


Figure B-26 Average 5:6 Gear Array Objective Function Value Convergence

B-5-2 4:5 Gear 0 km Offset Case

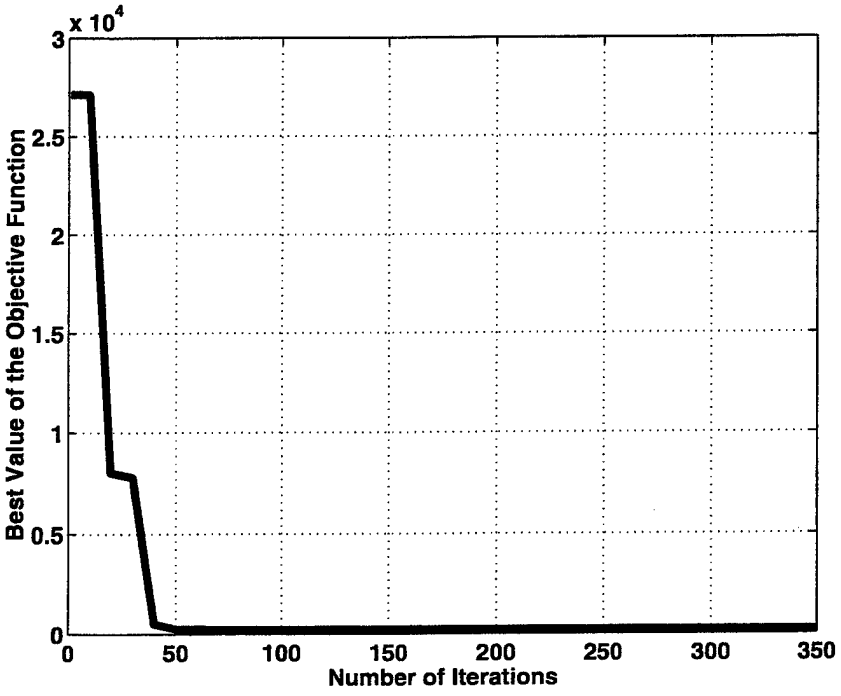


Figure B-27 Best 4:5 Gear Array Objective Function Value Convergence

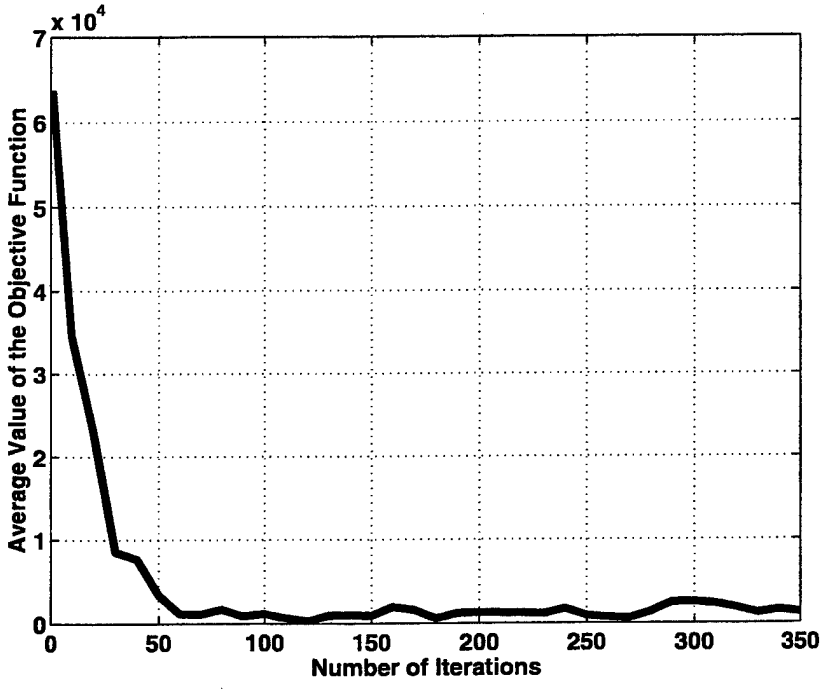


Figure B-28 Average 4:5 Gear Array Objective Function Value Convergence

B-6 Desired Gear Behavior and Element Decay Plots

Important to the optimization of the gear array is an understanding of how stable the resulting orbits are, or in other words, how the orbits decay over time. The following four sections present plots that detail this decay. The decay of the 5:6 8050 km apogee height array under a zonal 50 x 0 field is first presented. The decay for the same case is then shown under a fully perturbed field. The remaining two sections present identical plots for the 4:5 0 km offset case.

Ten plots are presented in each section. The first three show the deviation of the design from the three desired constraints: apogee pointing to the sun, gearing phase, and gearing ratio. The element decay plots of the APTS orbit and the element decay plots of the circular array then follow.

B-6-1 5:6 Gear 8050 km Apogee Height Zonal 50 x 0 Field Decay Plots

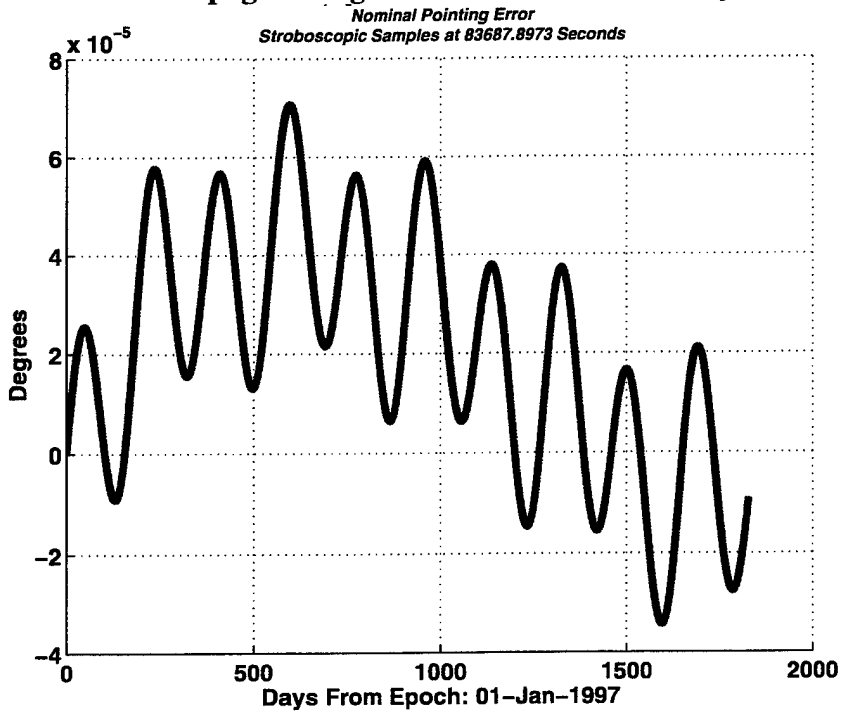


Figure B-29 5:6 Gear APTS Constraint Error (Zonals Only)

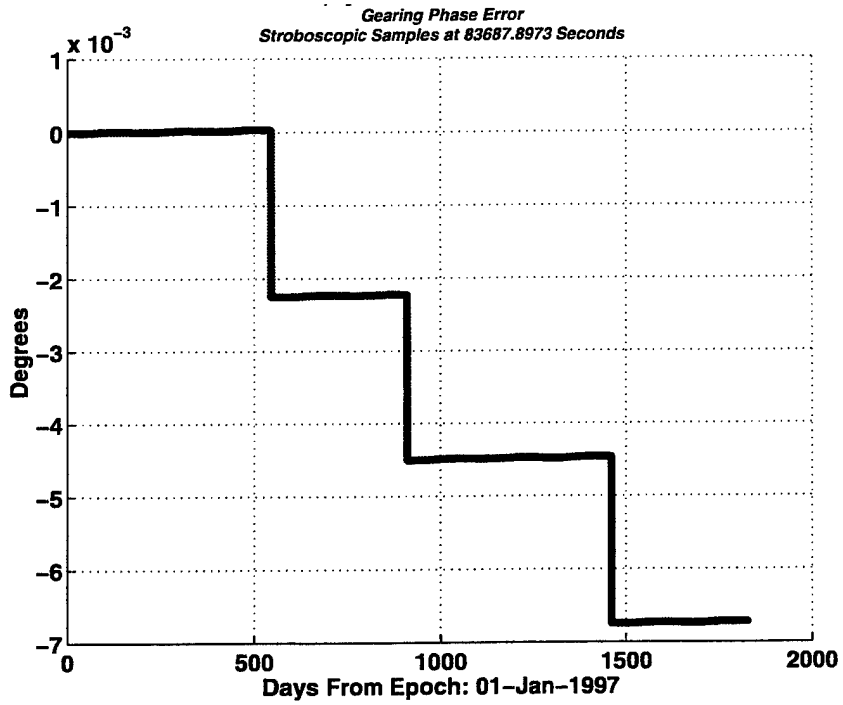


Figure B-30 5:6 Gear Stroboscopic Constraint Error (Zonals Only)

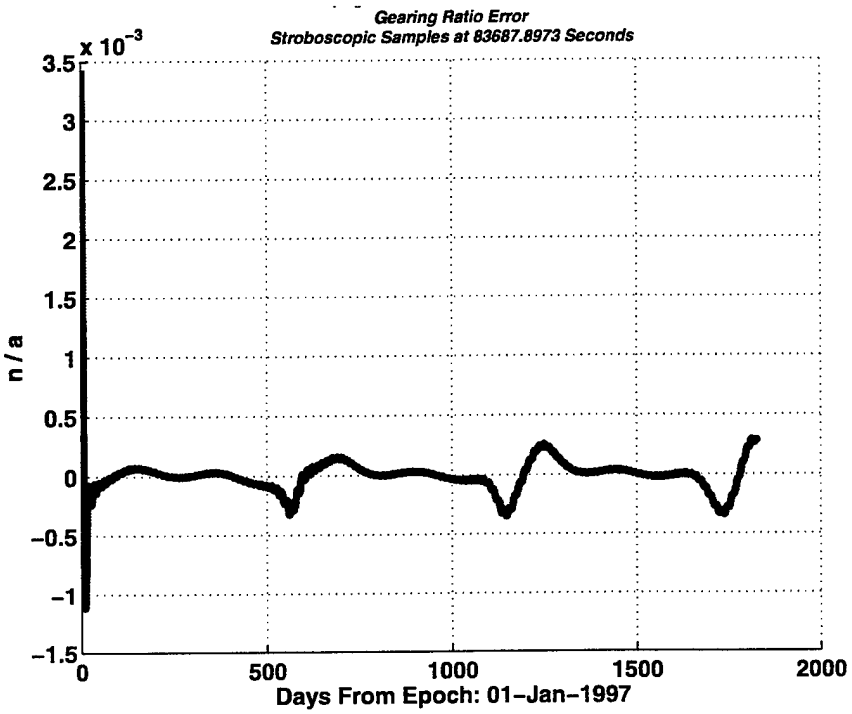


Figure B-31 5:6 Gear Ratio Constraint Error (Zonals Only)

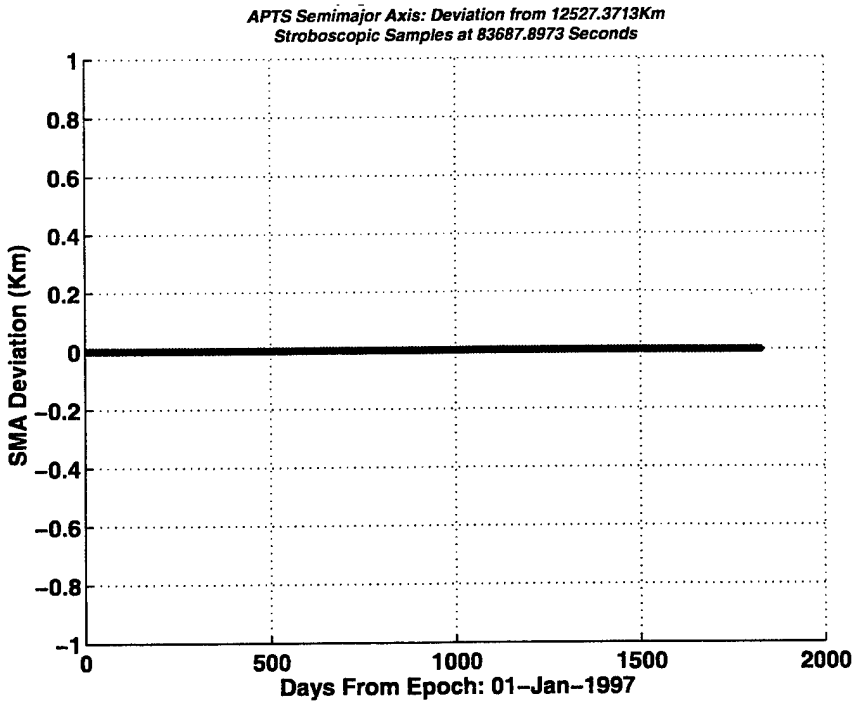


Figure B-32 5:6 Gear APTS Semi-major Axis Deviation from 12527.3713 km (Zonals Only)

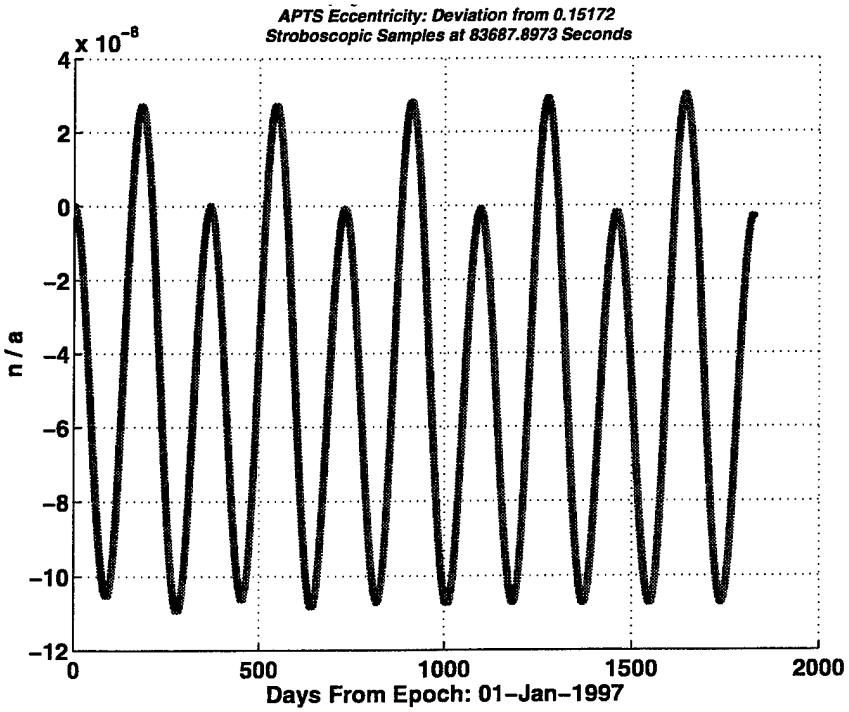


Figure B-33 5:6 Gear APTS Eccentricity Deviation from 0.15172 (Zonals Only)

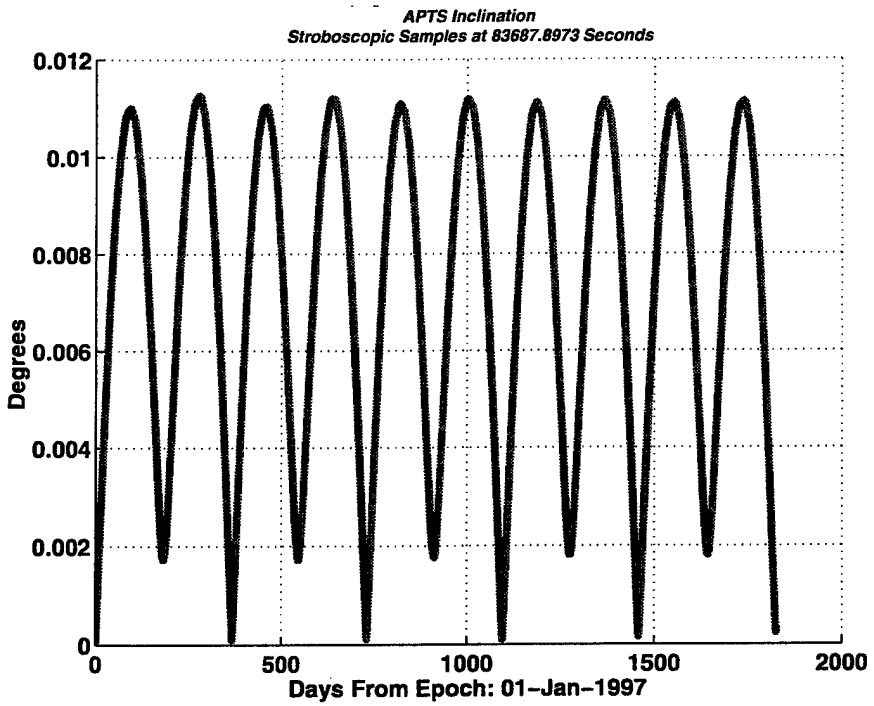


Figure B-34 5:6 Gear APTS Inclination Deviation from 0.0° (Zonals Only)

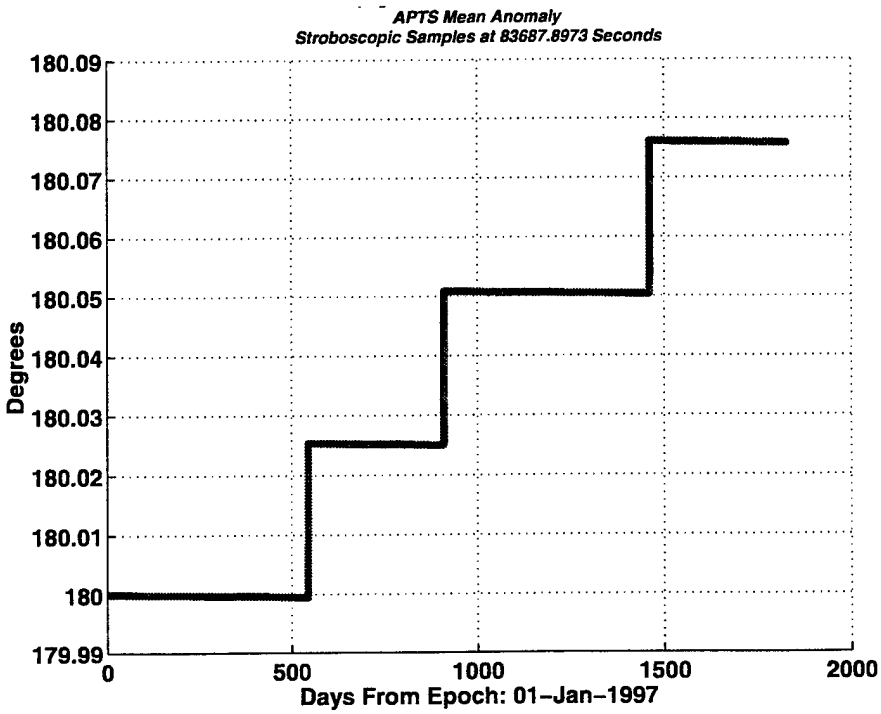


Figure B-35 5:6 Gear APTS Mean Anomaly History (Zonals Only)

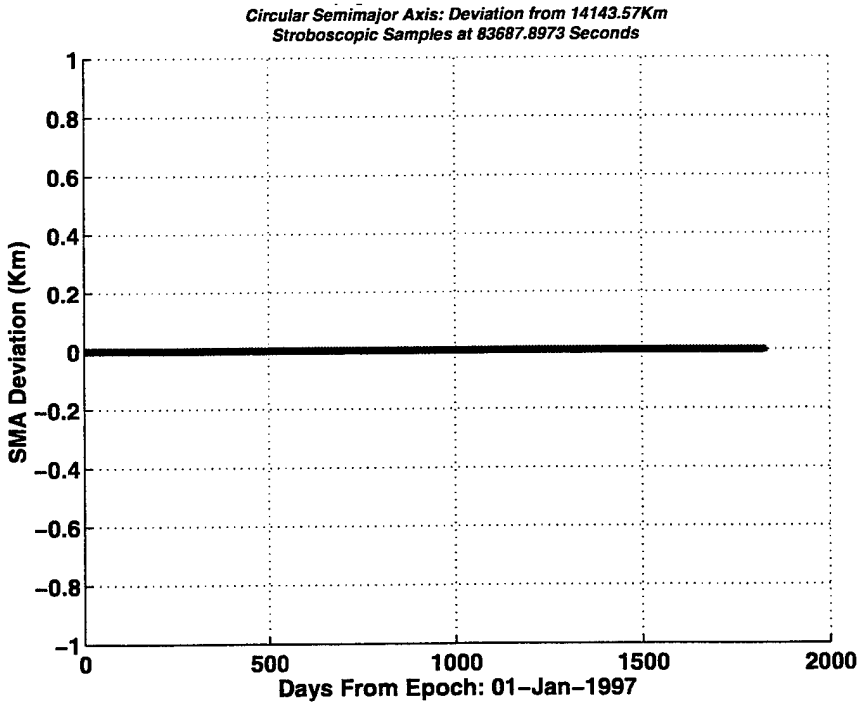


Figure B-36 5:6 Gear Circular Semi-major Axis Deviation from 14143.57 km (Zonals Only)

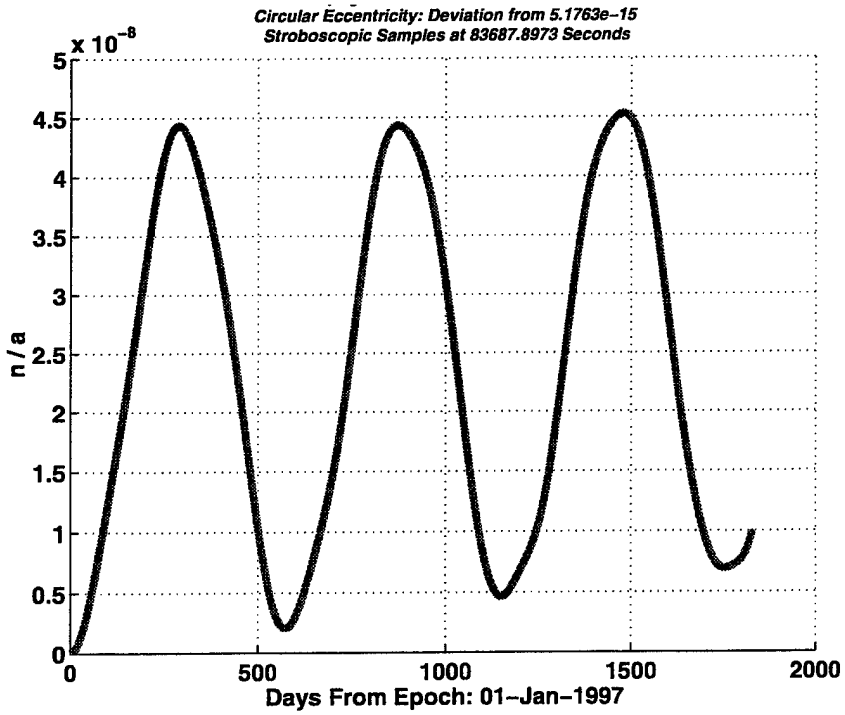


Figure B-37 5:6 Gear Circular Eccentricity Deviation from 0.0 (Zonals Only)

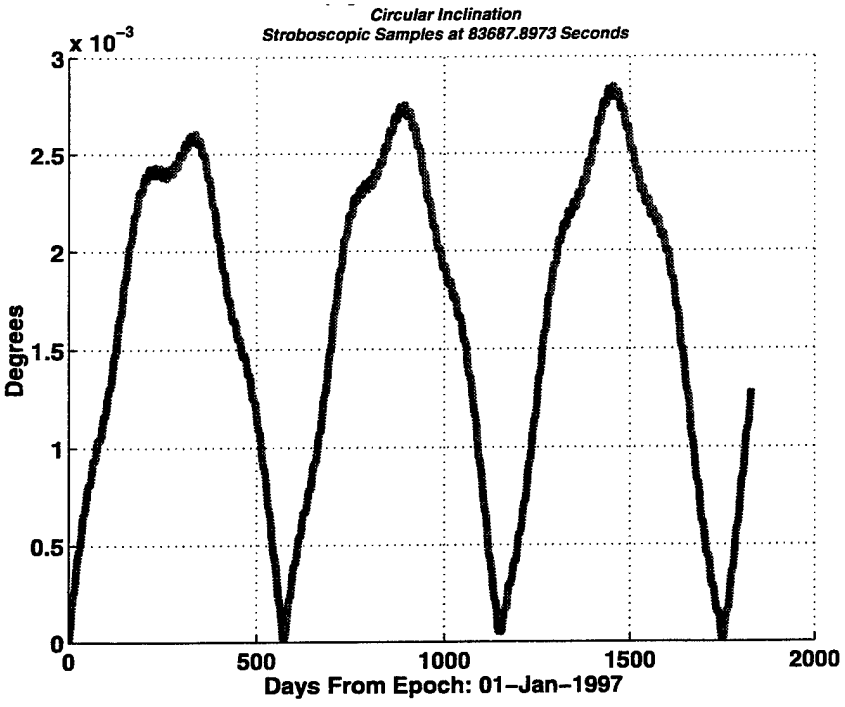


Figure B-38 5:6 Gear Circular Inclination Deviation from 0.0° (Zonals Only)

B-6-2 5:6 Gear 8050 km Apogee Height Full Perturbation Decay Plots

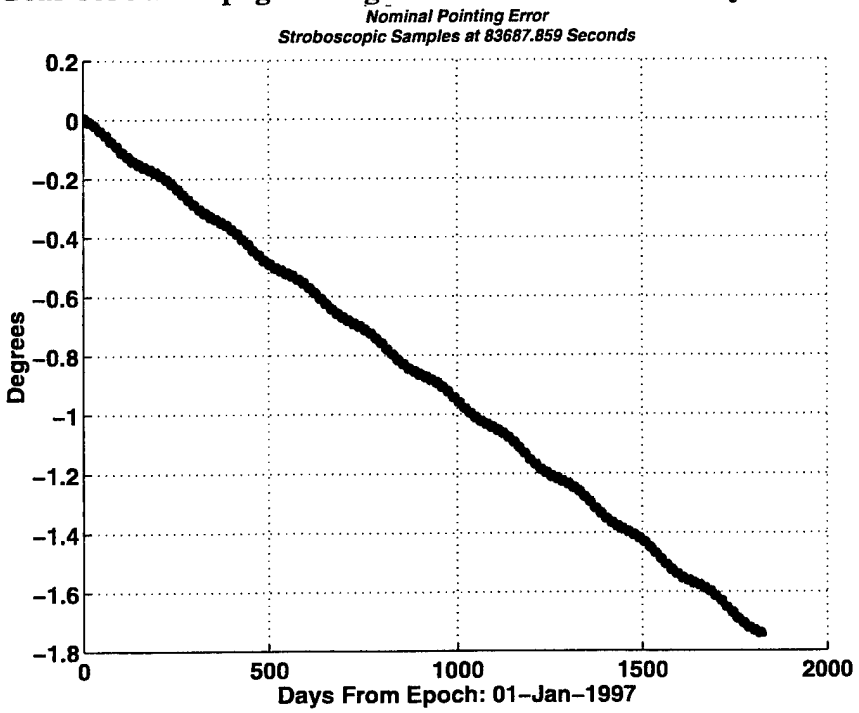


Figure B-39 5:6 Gear APTS Constraint Error (Full Pert.)

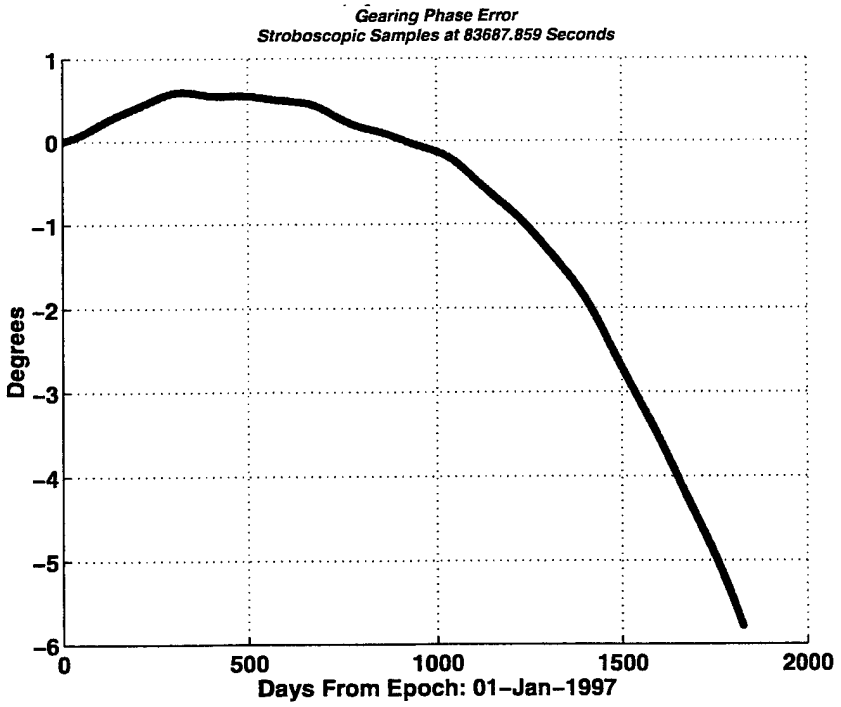


Figure B-40 5:6 Gear Stroboscopic Constraint Error (Full Pert.)

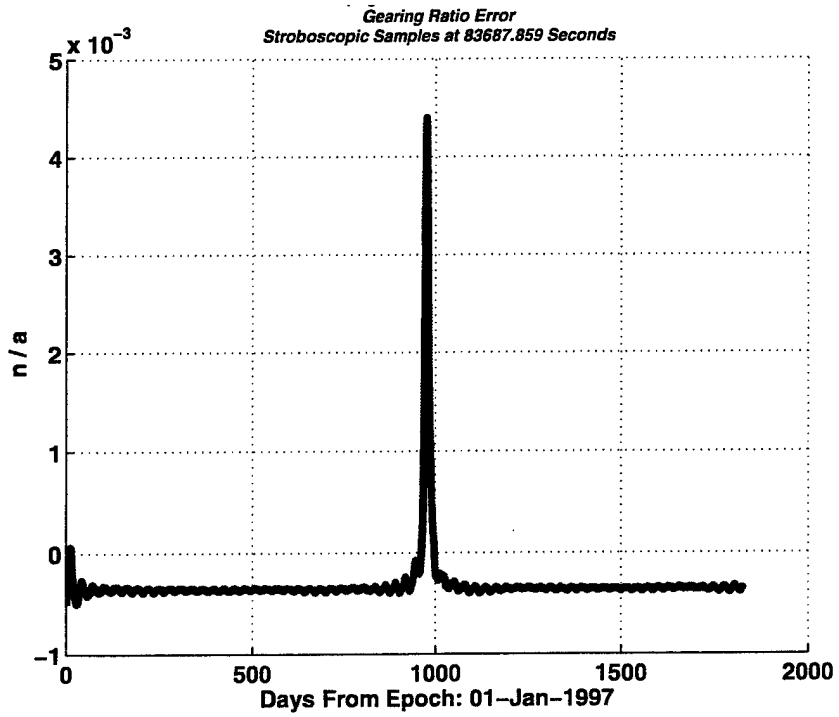


Figure B-41 5:6 Gear Ratio Constraint Error (Full Pert.)

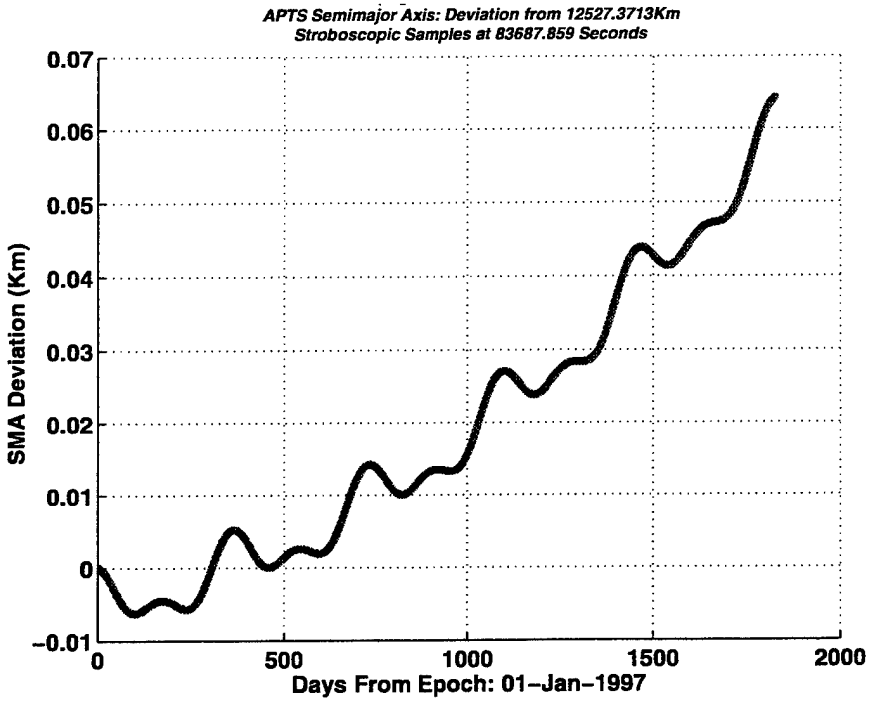


Figure B-42 5:6 Gear APTS Semi-major Axis Deviation from 12527.3713 km (Full Pert.)

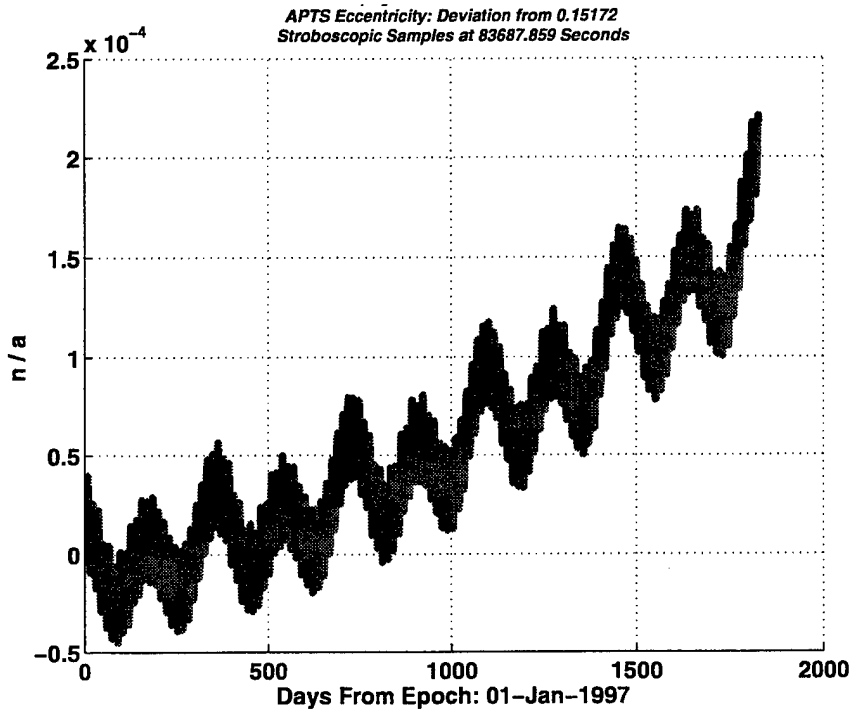


Figure B-43 5:6 Gear APTS Eccentricity Deviation from 0.15172 (Full Pert.)

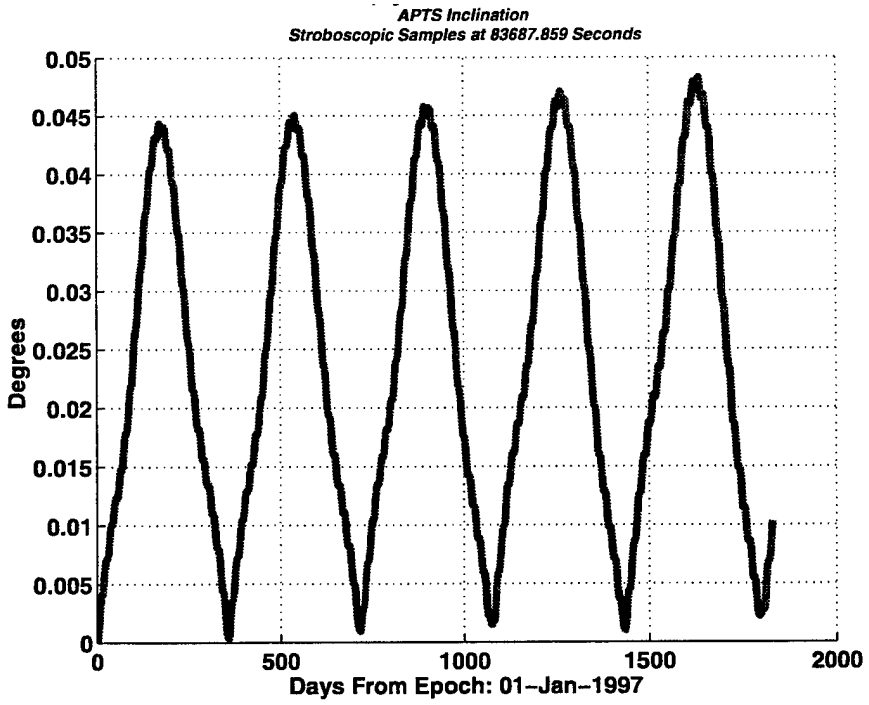


Figure B-44 5:6 Gear APTS Inclination Deviation from 0.0° (Full Pert.)

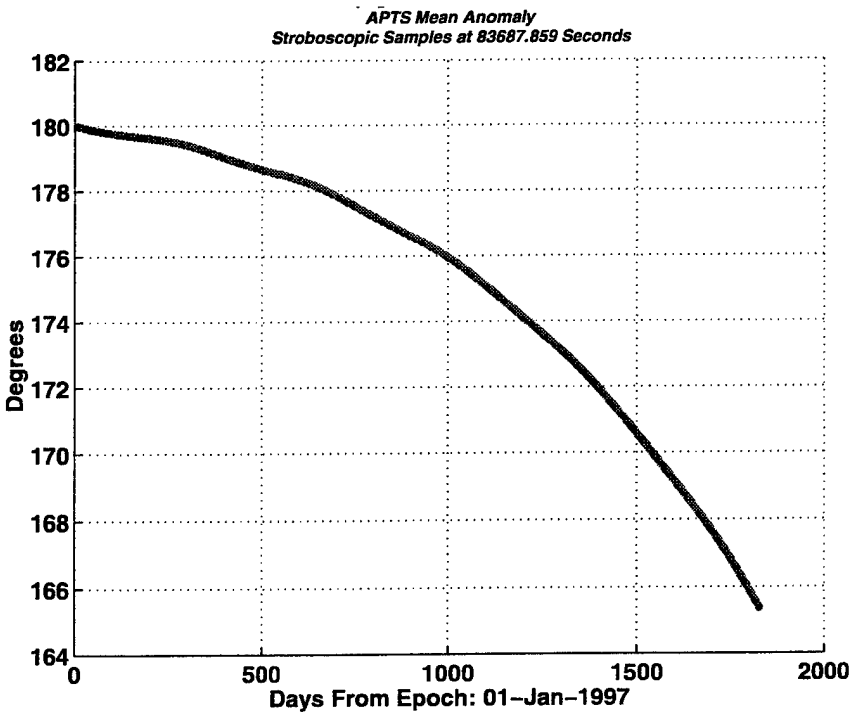


Figure B-45 5:6 Gear APTS Mean Anomaly History (Full Pert.)

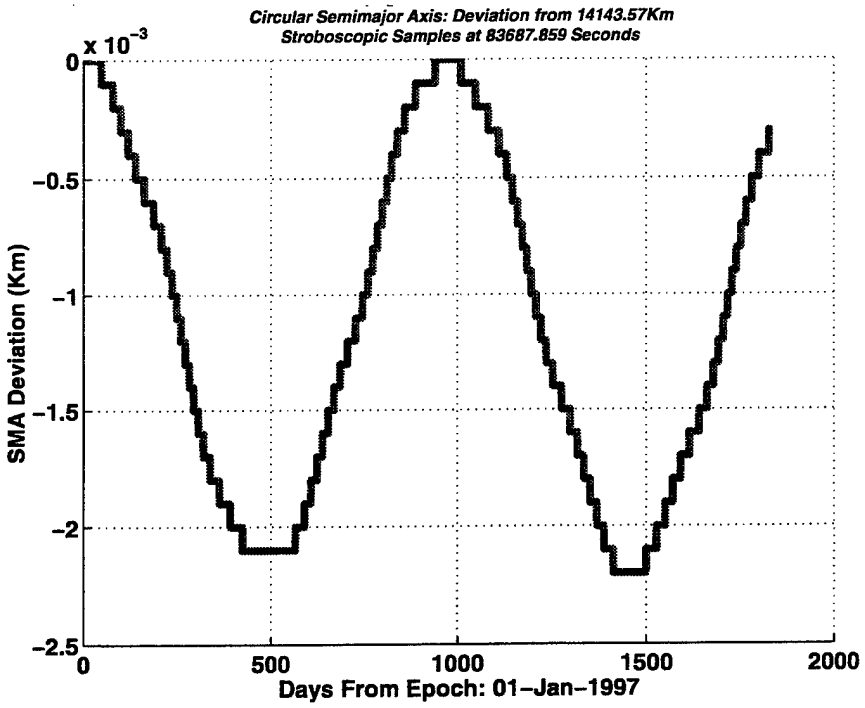


Figure B-46 5:6 Gear Circular Semi-major Axis Deviation from 14143.57 km (Full Pert.)

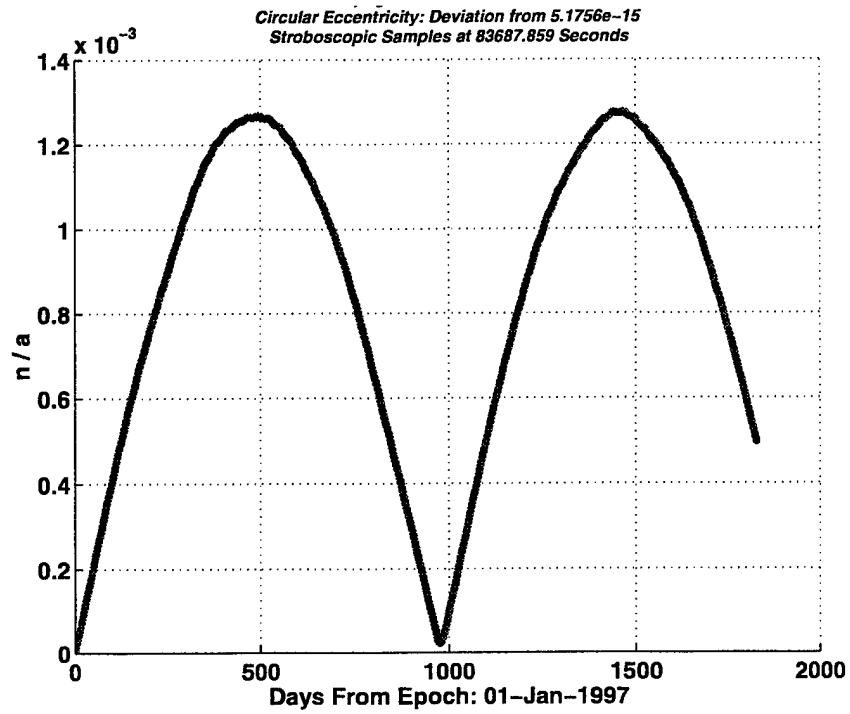


Figure B-47 5:6 Gear Circular Eccentricity Deviation from 0.0 (Full Pert.)

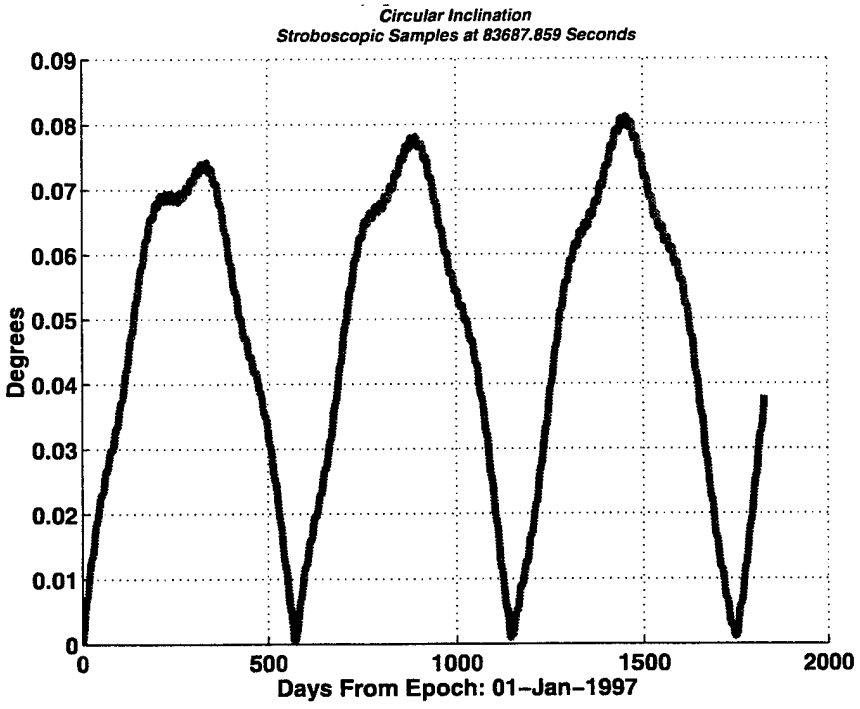


Figure B-48 5:6 Gear Circular Inclination Deviation from 0.0° (Full Pert.)

B-6-3 4:5 Gear 0 km Offset Zonal 50 x 0 Field Decay Plots

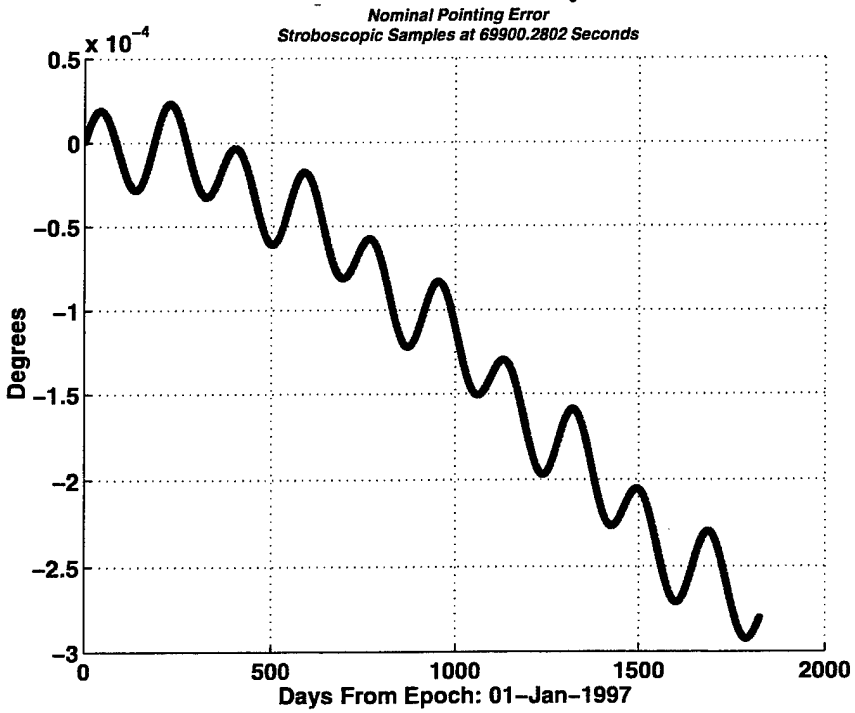


Figure B-49 4:5 Gear APTS Constraint Error (Zonals Only)

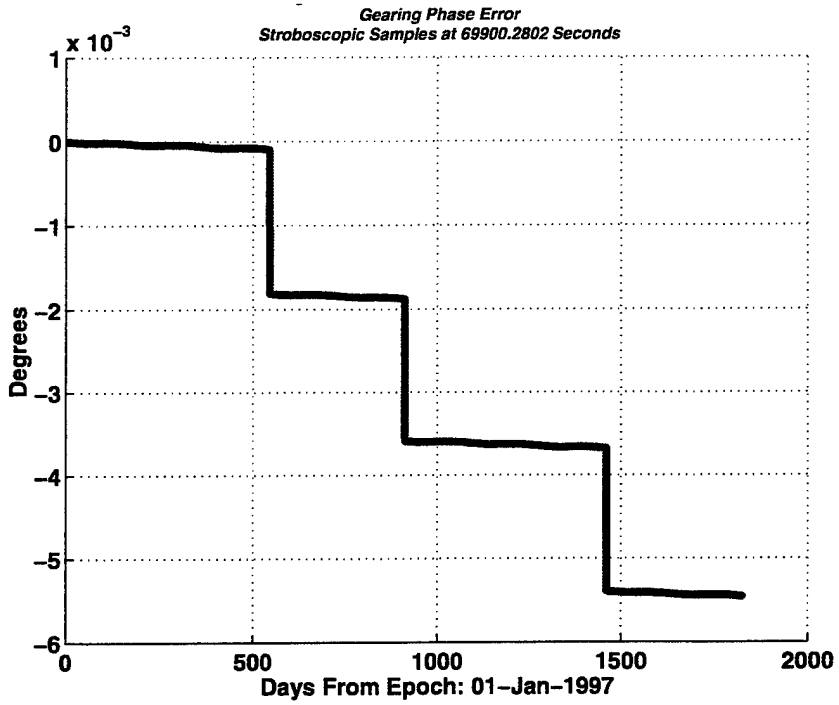


Figure B-50 4:5 Gear Stroboscopic Constraint Error (Zonals Only)

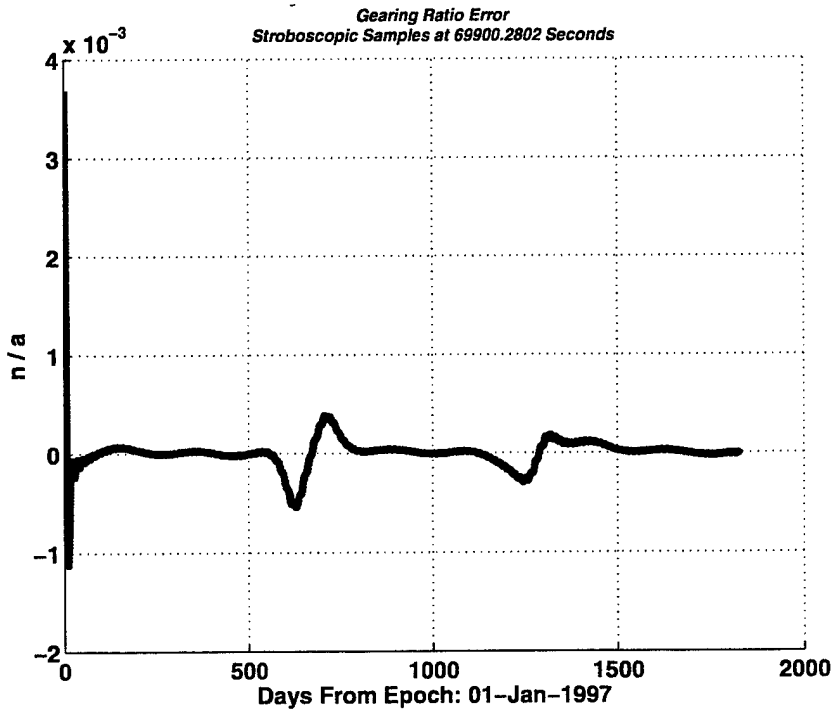


Figure B-51 4:5 Gear Ratio Constraint Error (Zonals Only)

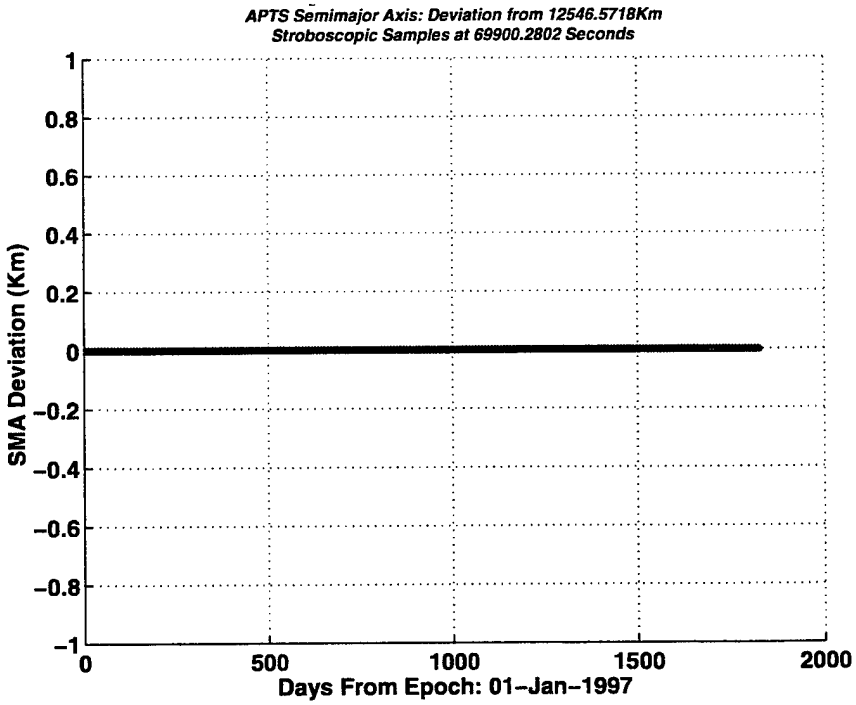


Figure B-52 4:5 Gear APTS Semi-major Axis Deviation from 12546.5718 km (Zonals Only)

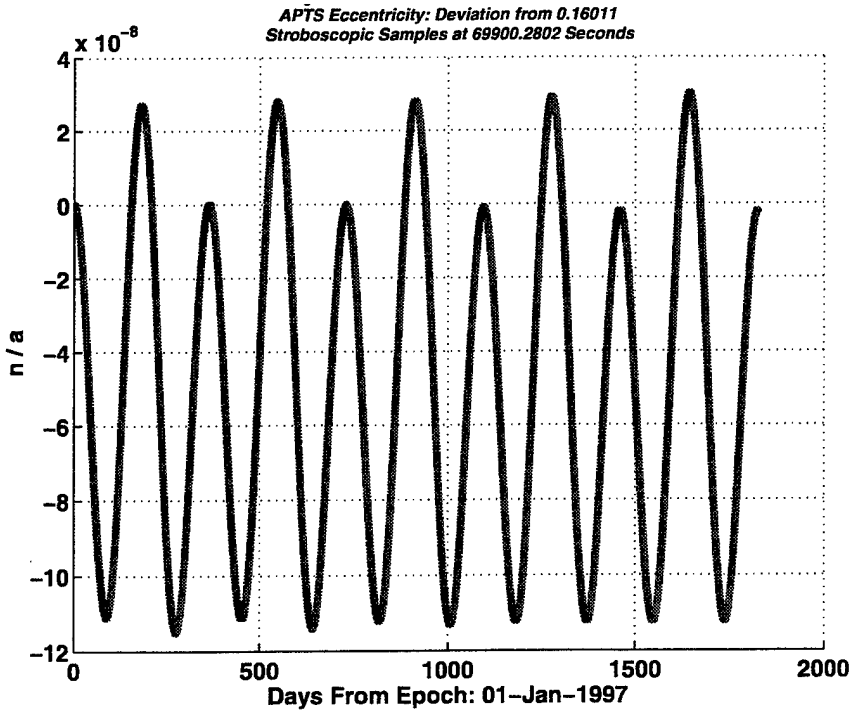


Figure B-53 4:5 Gear APTS Eccentricity Deviation from 0.16011 (Zonals Only)

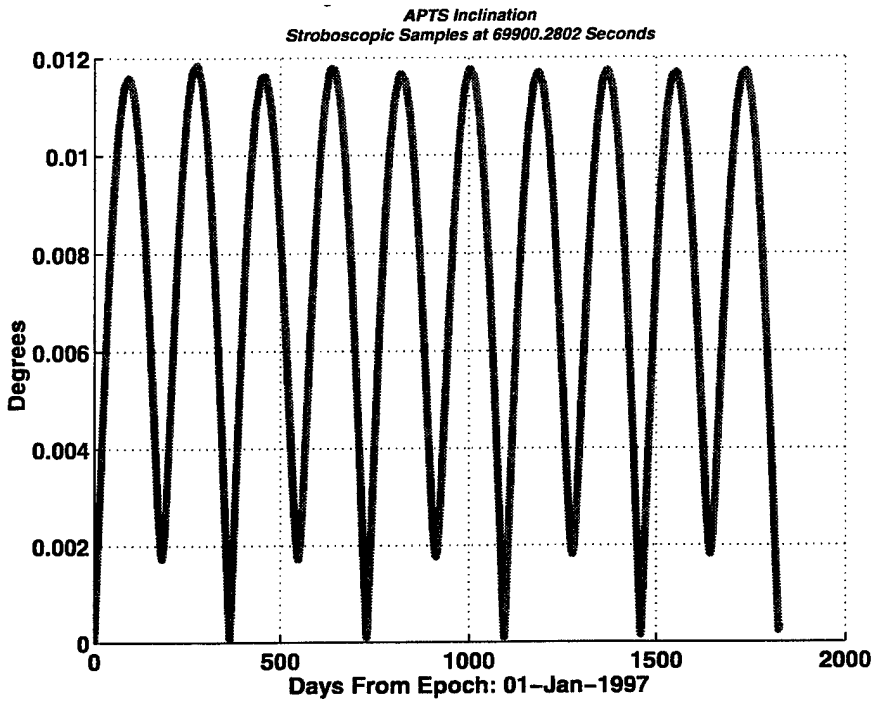


Figure B-54 4:5 Gear APTS Inclination Deviation from 0.0° (Zonals Only)

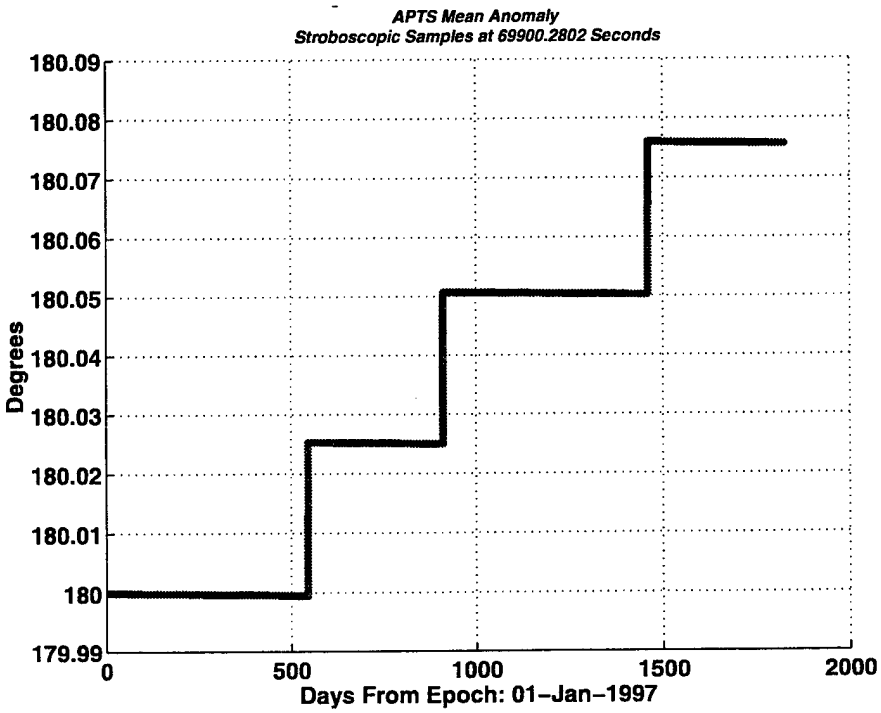


Figure B-55 4:5 Gear APTS Mean Anomaly History (Zonals Only)

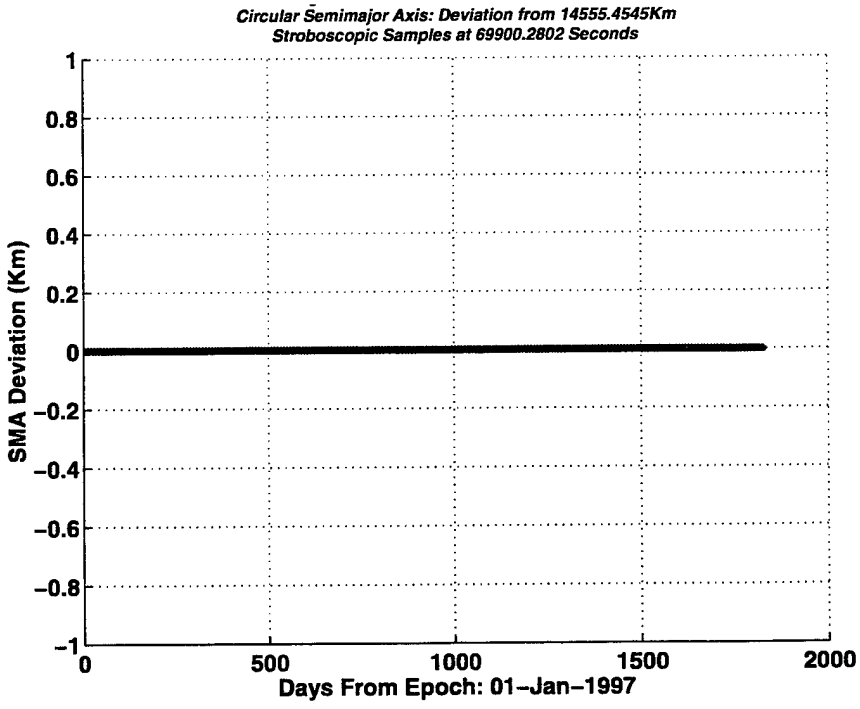


Figure B-56 4:5 Gear Circular Semi-Major Axis Deviation from 14555.4545 km (Zonals Only)

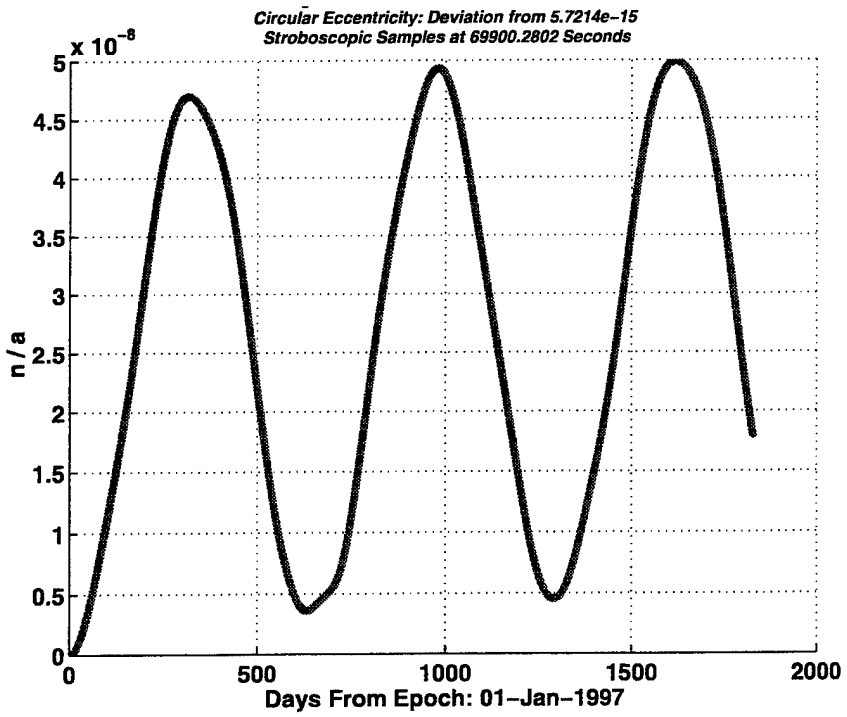


Figure B-57 4:5 Gear Circular Eccentricity Deviation from 0.0 (Zonals Only)

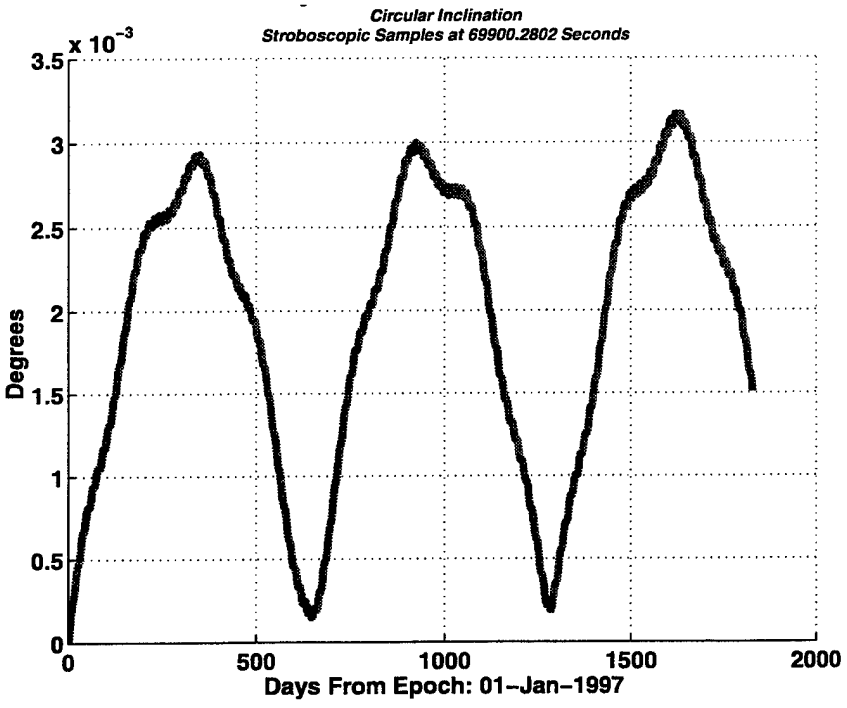


Figure B-58 4:5 Gear Circular Inclination Deviation from 0.0° (Zonals Only)

B-6-4 4:5 Gear 0 km Offset Full Perturbation Decay Plots

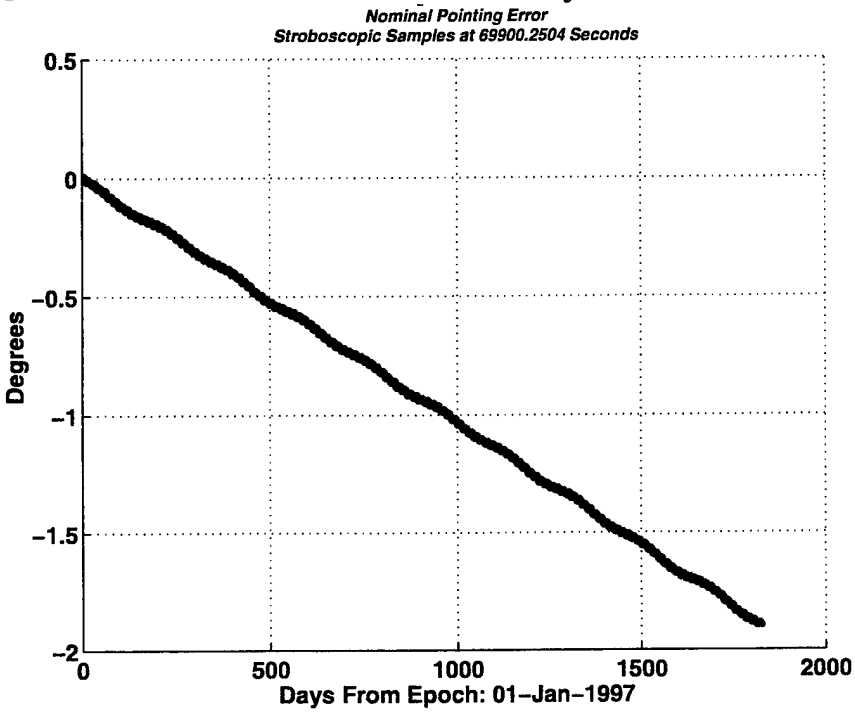


Figure B-59 4:5 Gear APTS Constraint Error (Full Pert.)

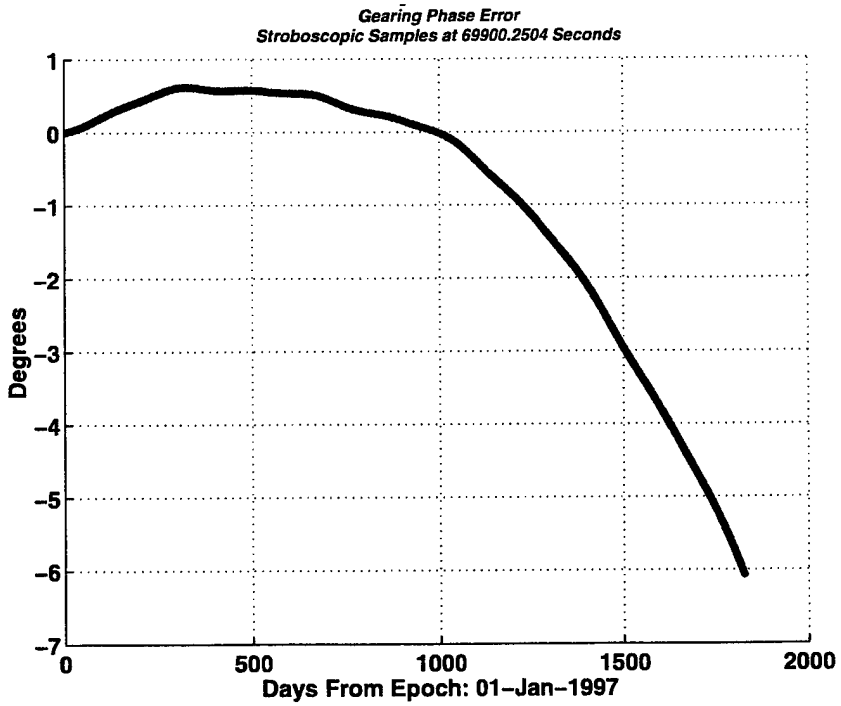


Figure B-60 4:5 Gear Stroboscopic Constraint Error (Full Pert.)

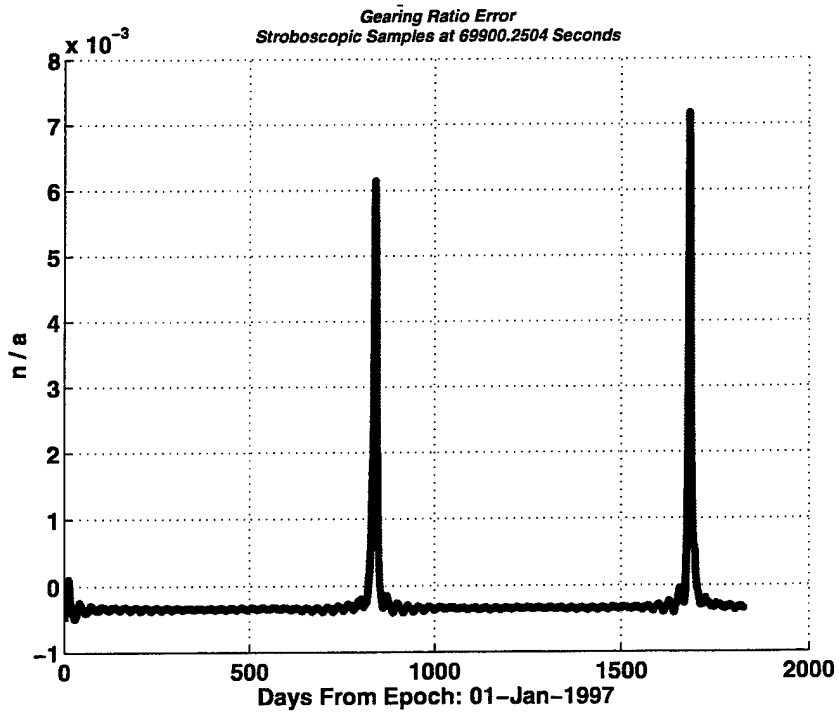


Figure B-61 4:5 Gear Ratio Constraint Error (Full Pert.)

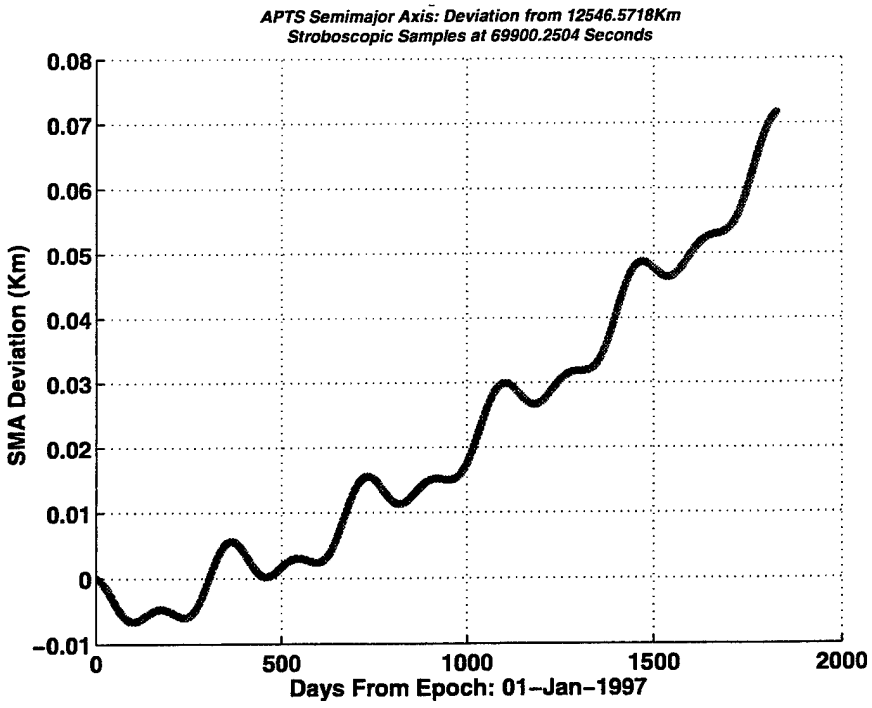


Figure B-62 4:5 Gear APTS Semi-Major Axis Deviation from 12546.5718 km (Full Pert.)

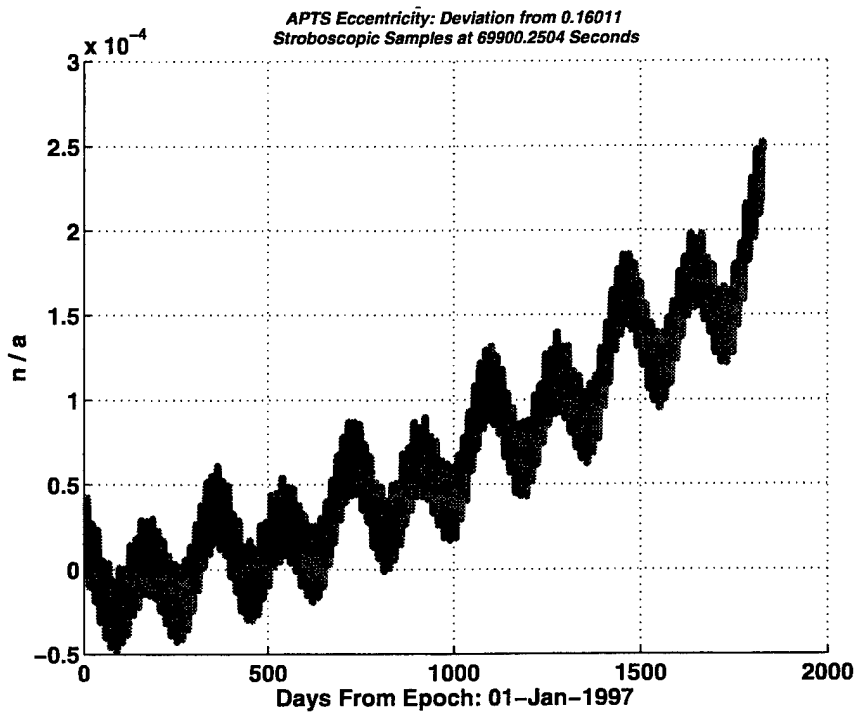


Figure B-63 4:5 Gear APTS Eccentricity Deviation from 0.16011 (Full Pert.)

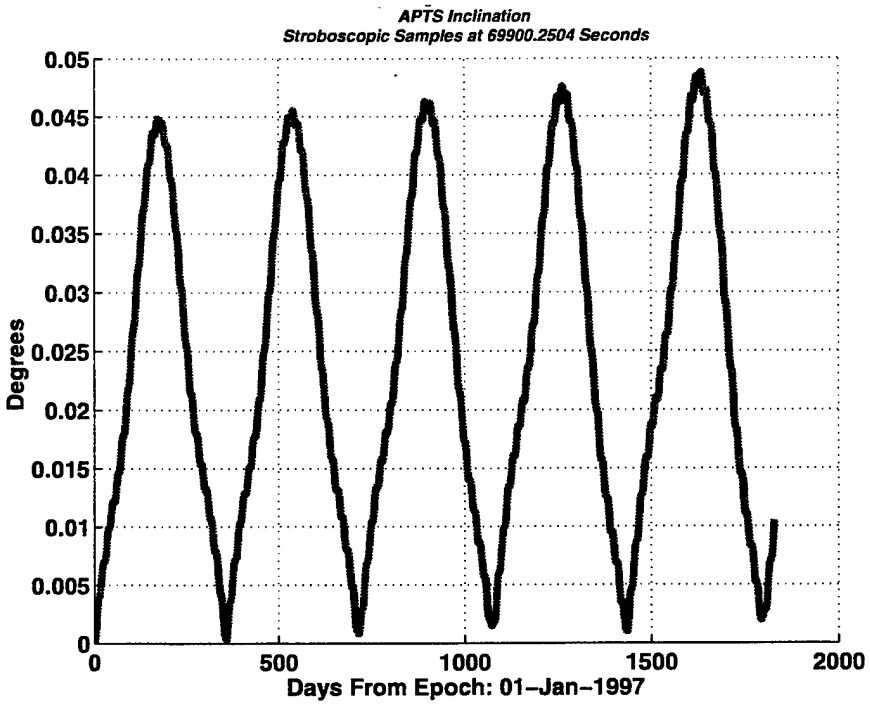


Figure B-64 4:5 Gear APTS Inclination Deviation from 0.0° (Full Pert.)

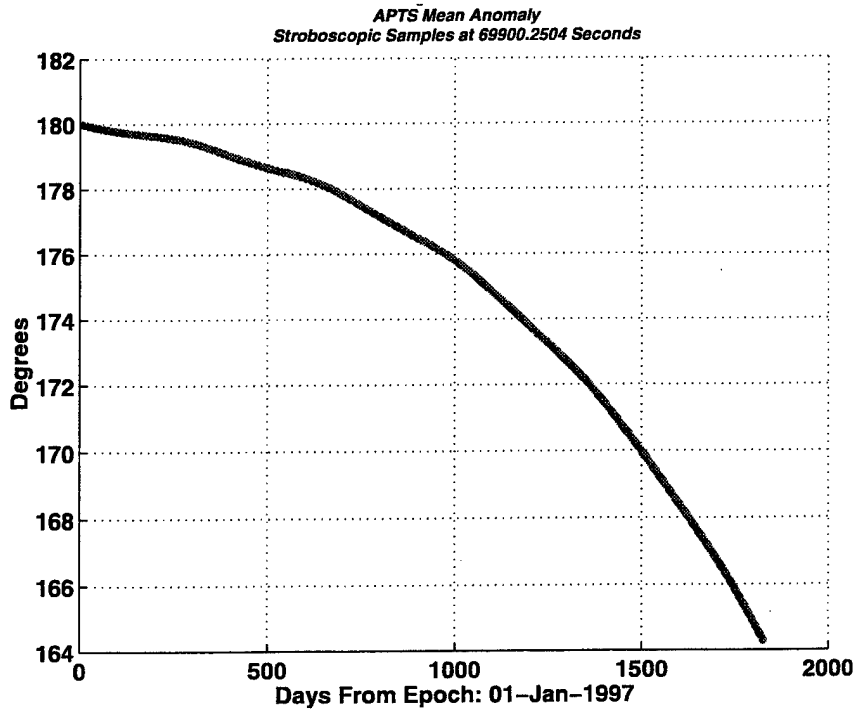


Figure B-65 4:5 Gear APTS Mean Anomaly History (Full Pert.)

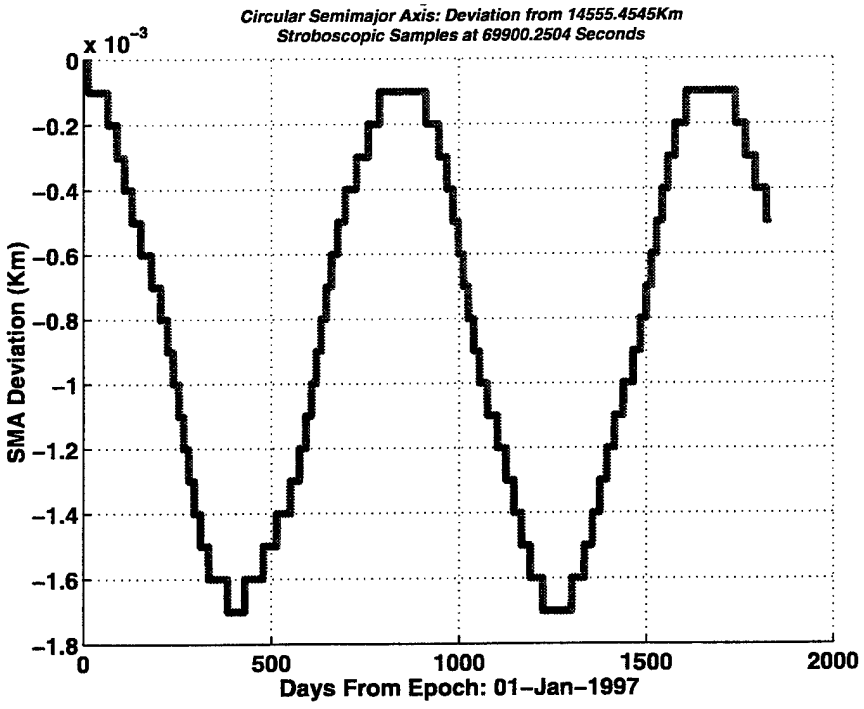


Figure B-66 4:5 Gear Circular Semi-Major Axis Deviation from 14555.4545 km (Full Pert.)

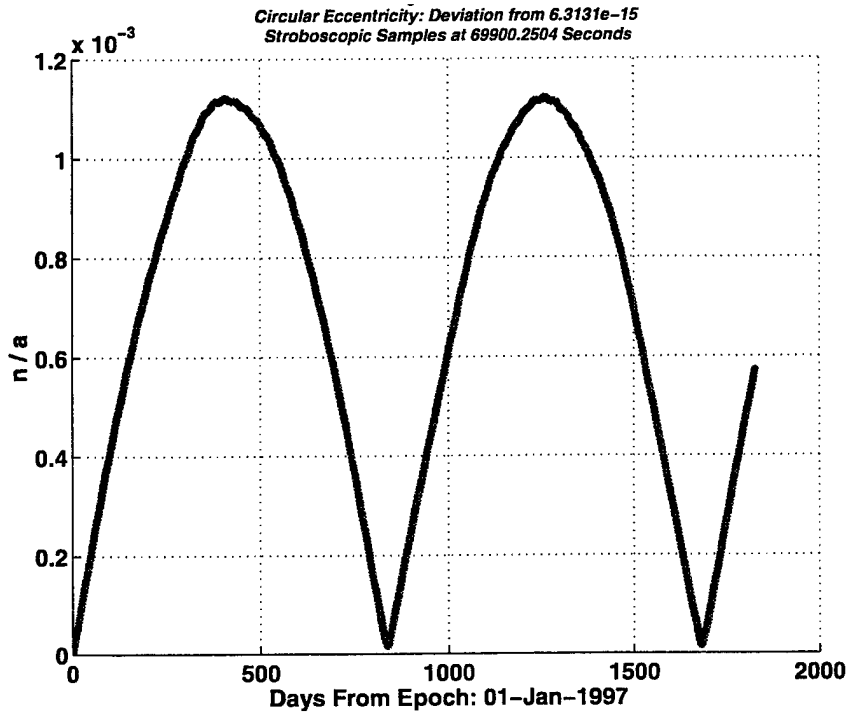


Figure B-67 4:5 Gear Circular Eccentricity Deviation from 0.0 (Full Pert.)

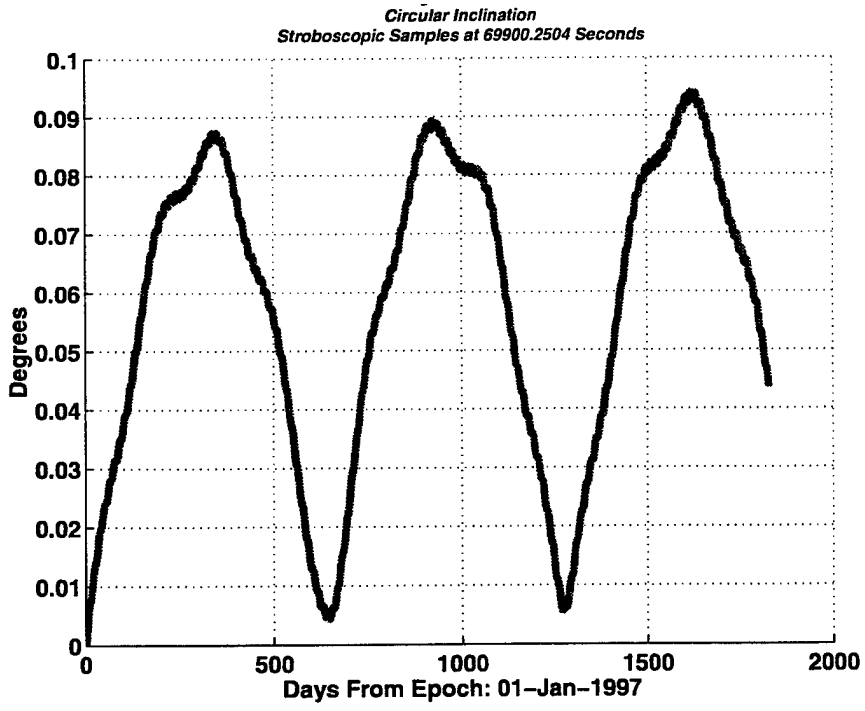


Figure B-68 4:5 Gear Circular Inclination Deviation from 0.0° (Full Pert.)

B-7 Gear Array Coverage Analysis Plots

In order to compare the performance of the optimally designed gear arrays with the baseline Ellipso™ array, the designs were used to simulate the coverage seen at various latitudes at various times of the day. The simulation was used to collect data over a two-week time interval which could then be used to create the "wedge" plots found in the following sections.

The wedge plots from a local time of noon and the wedge plots from a local time of 3 PM are presented in the following sections. For each time, data corresponding to the minimum elevation angle, average elevation angle, minimum number of satellites in view, maximum number of satellites in view, and average number of satellites in view is presented.

B-7-1 Local Time of Noon Coverage Analysis Plots

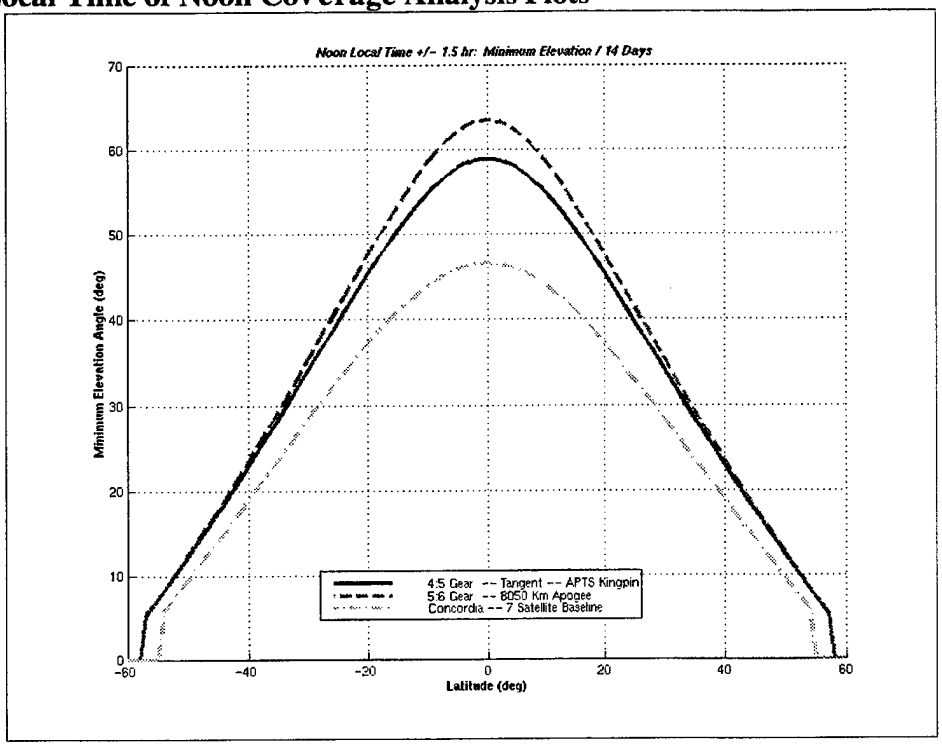


Figure B-69 Minimum Elevation Angle Comparison—Noon Local Time

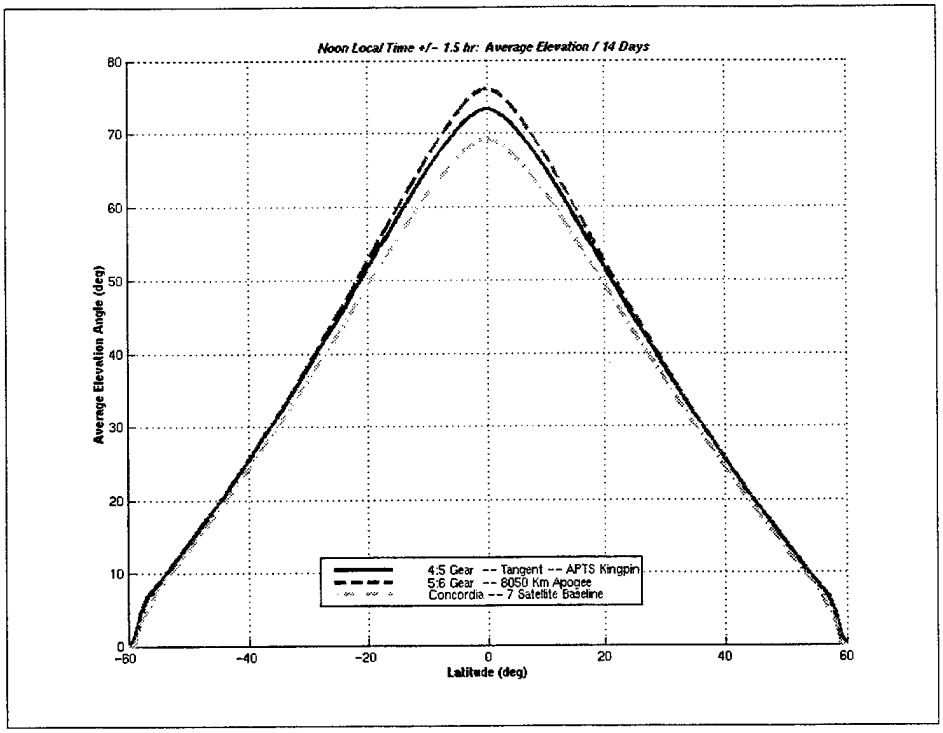


Figure B-70 Average Elevation Comparison—Noon Local Time

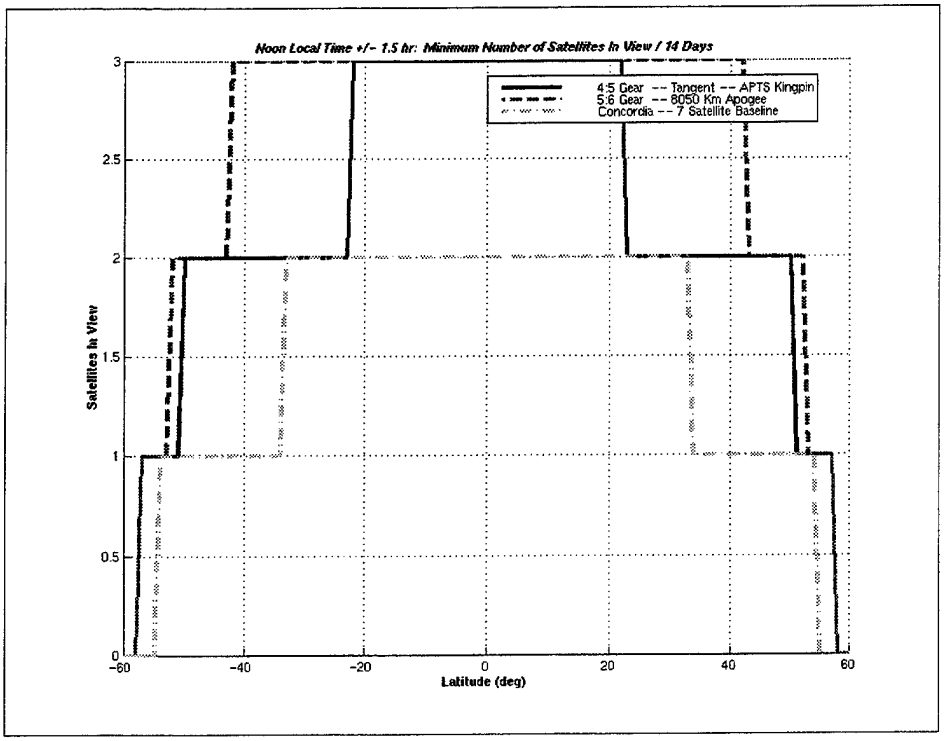


Figure B-71 Minimum Number of Satellites in View Comparison—Noon Local Time

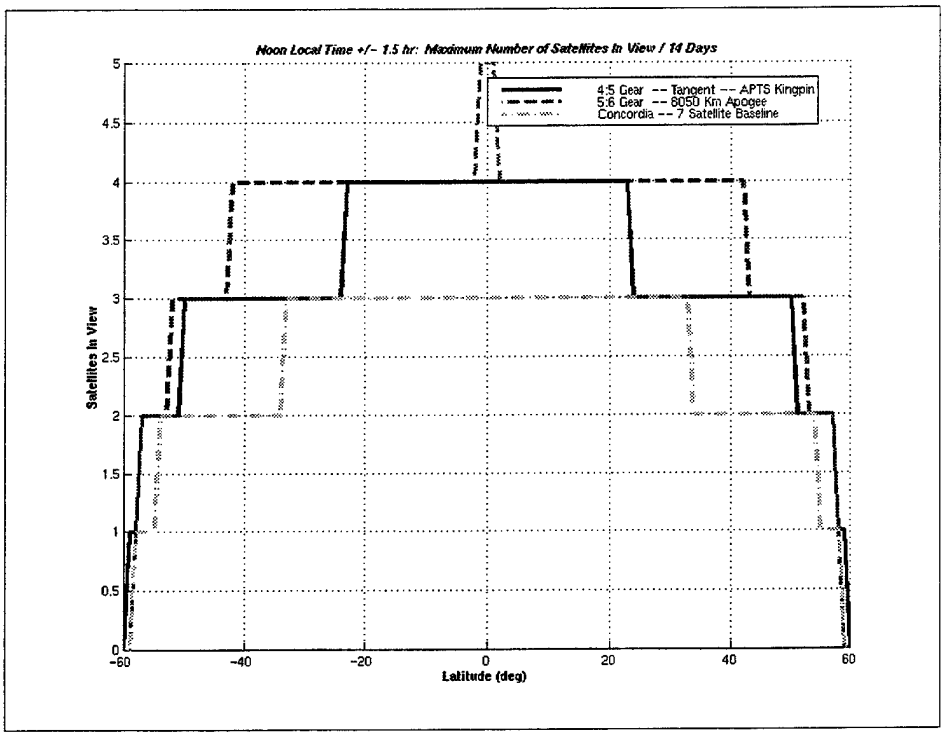


Figure B-72 Maximum Number of Satellites in View Comparison—Noon Local Time

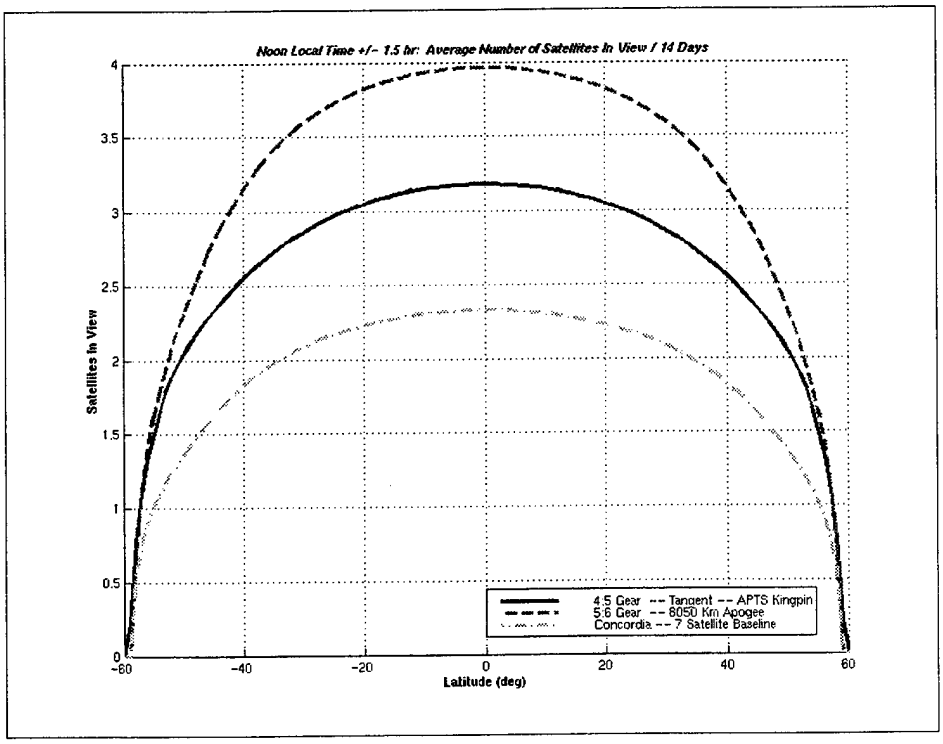


Figure B-73 Average Number of Satellites in View Comparison—Noon Local Time

B-7-2 Local Time of 3:00 P.M. Coverage Analysis Plots

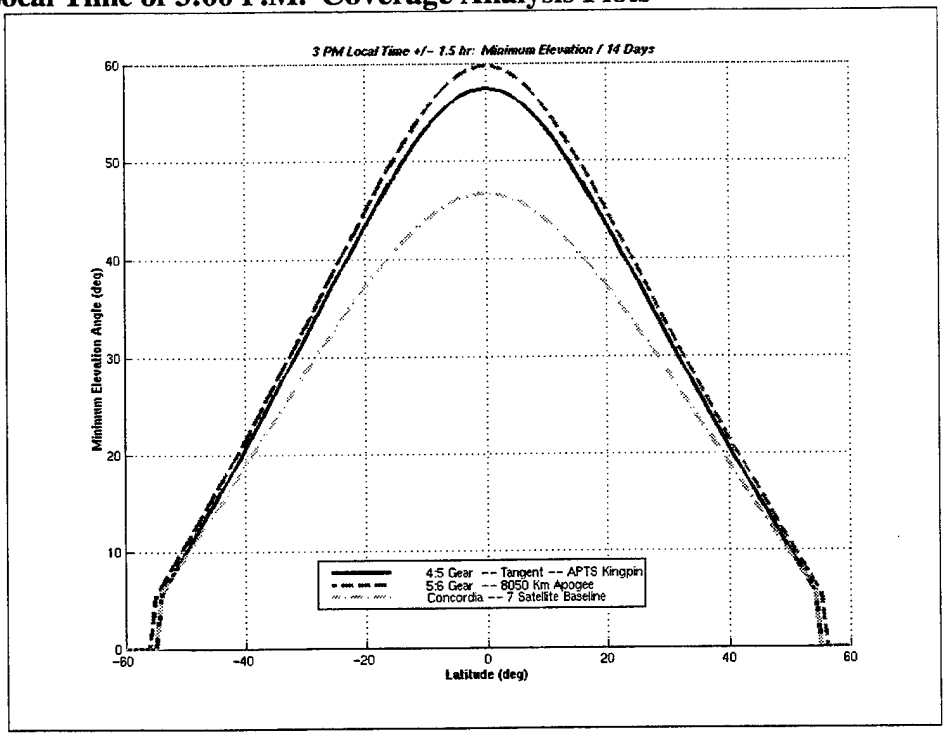


Figure B-74 Minimum Elevation Comparison—3 P.M. Local Time

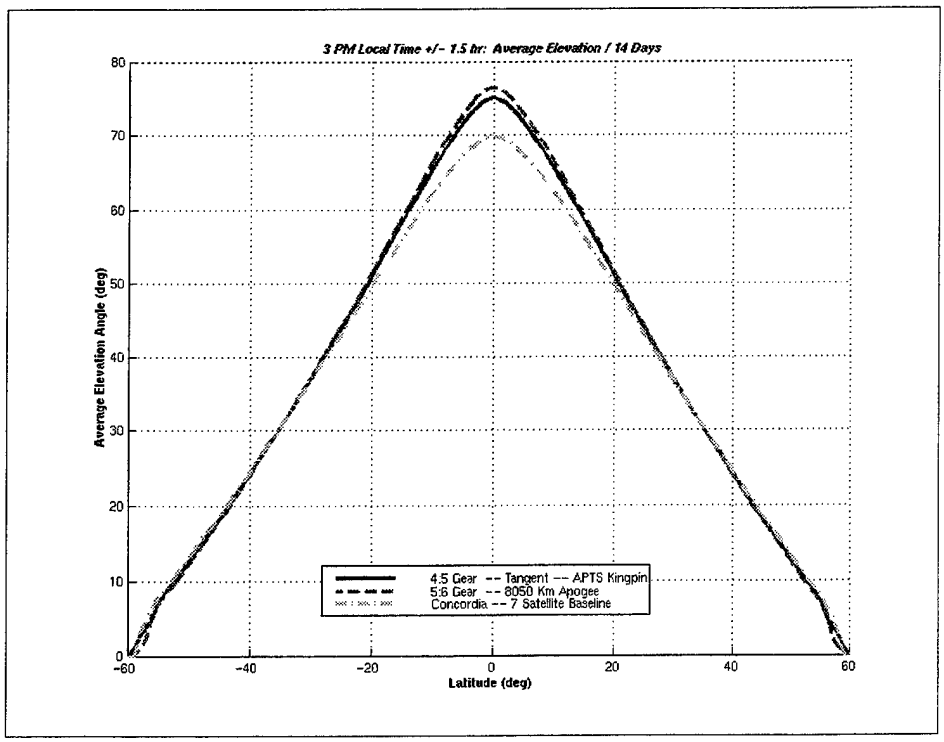


Figure B-75 Average Elevation Comparison—3 P.M. Local Time

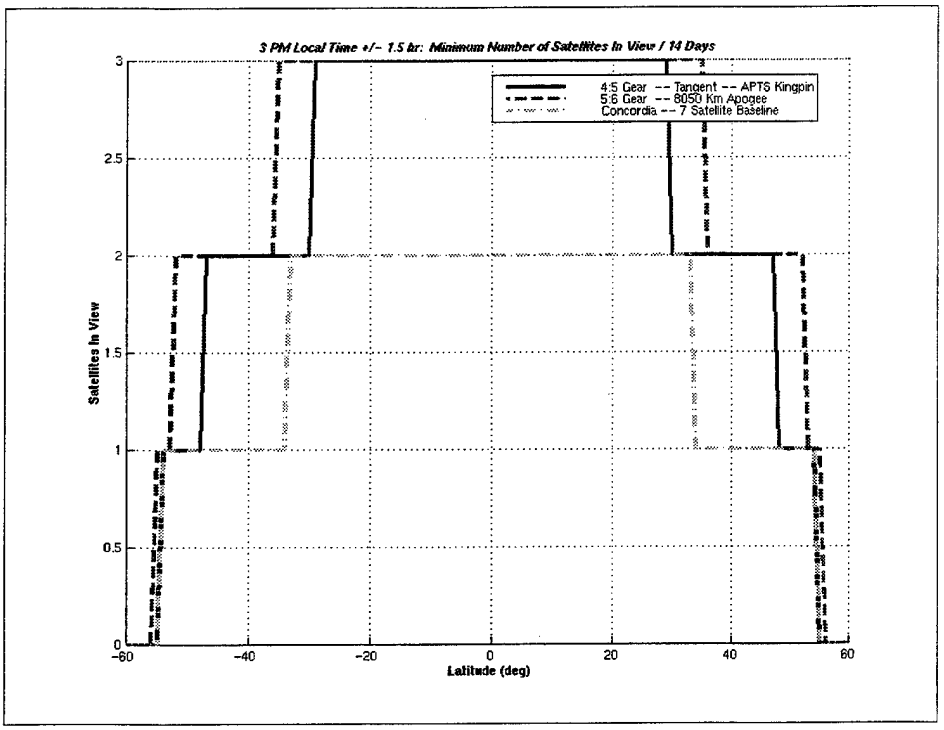


Figure B-76 Minimum Number of Satellites in View Comparison—3 P.M. Local Time

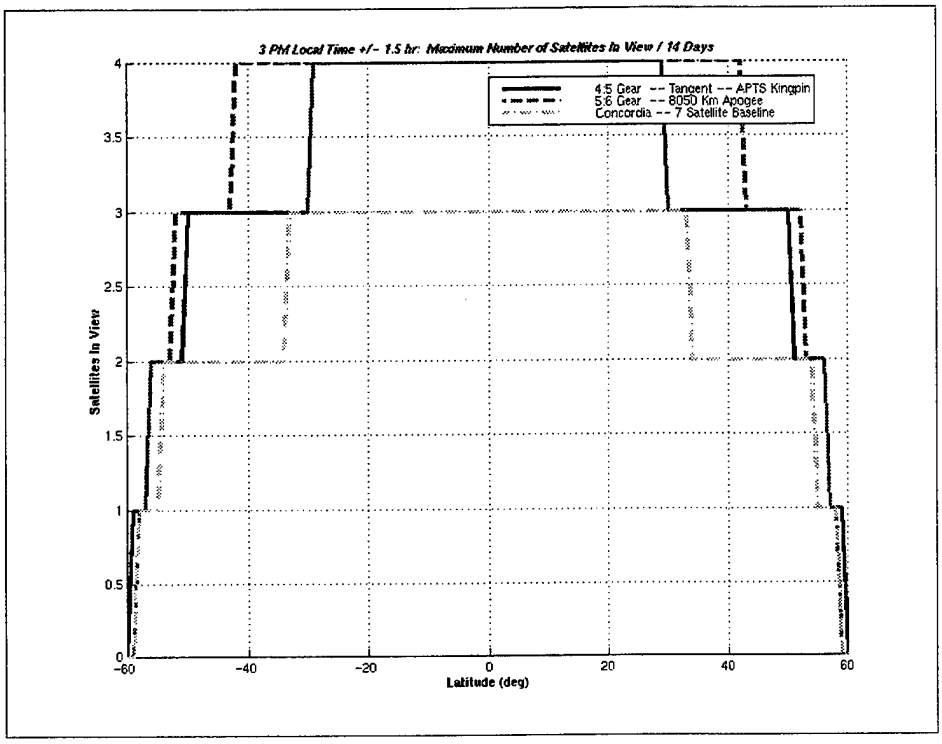


Figure B-77 Maximum Number of Satellites in View Comparison—3 P.M. Local Time

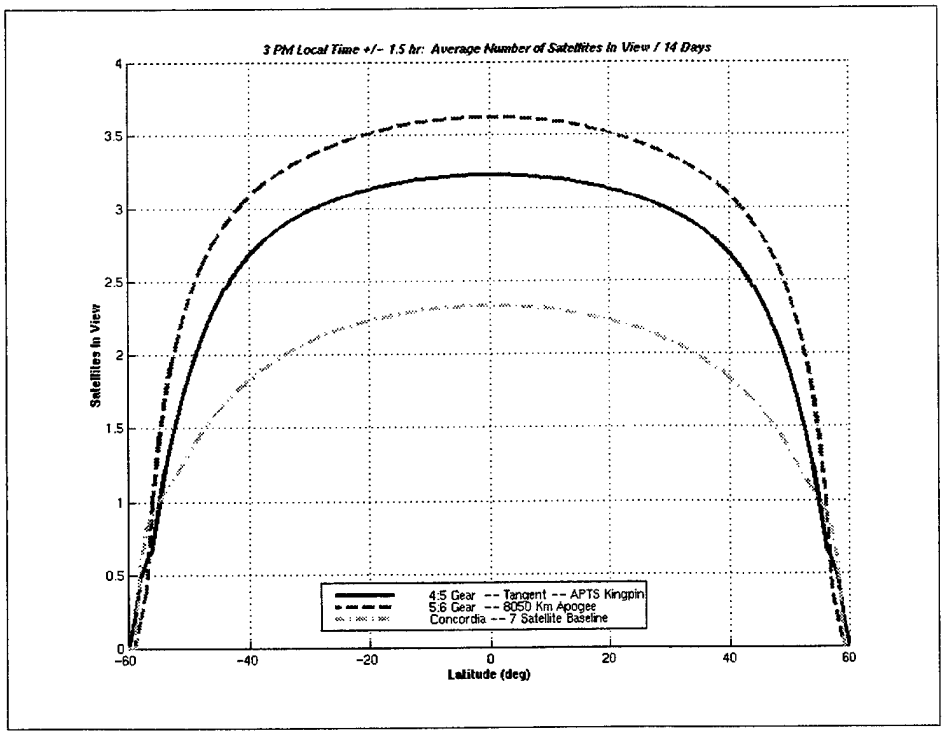


Figure B-78 Average Number of Satellites in View Comparison—3 P.M. Local Time

[This Page Intentionally Left Blank]

Appendix C Global Station Keepi n g Approach Data

This appendix contains the input data and related results for the global station keeping approach optimization cases. The modifications required to the previously used PGAPack/DSST code are first presented. These modifications are followed by the presentation of the two required input decks: one for a reference orbit definition, and one for propagation of the actual orbit. Finally, the uncontrolled and controlled element histories for both the center of the box case (Case 1) and the more difficult edge of the box case (Case 2) are presented.

C-1 Code Modifications for G lobal Station Keeping Implementation

As discussed in Chapter 5, some slight modifications to the PGAPack/DSST code setup was required to convert the software tool from an orbit design optimization package to an orbit maintenance optimization package. The most obvious of these modifications is the definition of a completely different objective function (FUNC) subroutine. Inherent to the problem is also the requirement for a reference orbit definition and a calculation of the time of the first deviation. These requirements each led to the creation of the DEFREFORB and CALCFIRTDEVTIME subroutines, respectively.

C-1-1 FUNC

```
real*8  function func(p)
C-----------------------------------------------------------------------
C
C                              Function Func
C
C  This file contains the objective function for the global
C  station keeping optimization problem.
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C          Ronald J. Proulx
C          Draper Laboratory
C
C-----------------------------------------------------------------------

      implicit none

C-------- Include necessary modules -------------------

      include 'pmern.h'
      include 'frc.h'
      include 'pgalim.h'

C----- Define variables for call to Satellite -----------------

      integer*4       satellite
      external        satellite

      integer*4       max_list_length
      parameter      (max_list_length  = 52)

      integer*4       status
      integer*4       burn_number
      Integer*4       iatmos_preburn / 1 /
      integer*4       iatmos_postburn / 1 /
      integer*4       month,hour,minute

      real*8          day(maxvar)
      real*8          time
      real*8          burn_delta_v(4,max_list_length)
      REAL*8          RHO_ONE_HIGH      / 0.0 D0 /
      REAL*8          RHO_ONE_LOW       / 0.0 D0 /
      real*8          epoch_ymd
      real*8          epoch_hms
      real*8          posvel(6)
      real*8          elements (17)
      real*8          second(maxvar)
      real*8          btime(max_list_length)

      character*(6)   name  /'pmern1'/
      character*(6)   refname /'pmernr'/
      CHARACTER*(72)  MESSAGE
      CHARACTER*(12)  FILENAME

C--------  Define other variables ----------------------

      integer*4       i
      integer*4       index
      integer*4       j,k
      integer*1       onoff(maxvar)
      integer*4       usedburn
```

```
      real*8          p(*)
      real*8          mag
      real*8          burn(maxvar)
      real*8          magburn(maxvar)
      real*8          deltavtot
      real*8          deltavsum(maxvar)
      real*8          burntime(maxvar)
      real*8          bcconst(maxvar)
      real*8          bcconsttot
      real*8          vconst
      real*8          Mcheck
      real*8          pi
      real*8          comp
      real*8          two_n
      real*8          bitzero(3)
      real*8          deltat
      real*8          maxbcconst(maxvar)
      real*8          totbcconst(maxvar)
      real*8          timeweight
      real*8          timeconst
      real*8          endconsttot
      real*8          endconst(maxvar)
      real*8          order(maxvar)
      real*8          same(maxvar)

      external        mag

C     ********** BEGIN PROGRAM *********************************

      pgalim.eflag = .false.

c     Initialize everything to 0.0d0 -------------------------

      deltat = 86400.0d0/PGALIM.TIMESPERDAY

      deltavtot = 0.0d0
      vconst = 0.0d0
      bcconsttot = 0.0d0
      endconsttot = 0.0d0
      do i = 1, maxvar
         bcconst(i) = 0.0d0
         deltavsum(i) = 0.0d0
         maxbcconst(i) = 0.0d0
         totbcconst(i) = 0.0d0
         day(i) = 0.0d0
         endconst(i) = 0.0d0
         order(i) = 0.0d0
         onoff(i) = 0.0d0
         same(i) = 0.0d0
      enddo
      PGALIM.firstdev = 0.0d0

      TIME              = 0.D0
      BURN_NUMBER       = 0
      DO I=1,PGALIM.MAXBURN
         BURN_DELTA_V(1,I) = 0.D0
         BURN_DELTA_V(2,I) = 0.D0
         BURN_DELTA_V(3,I) = 0.D0
         BURN_DELTA_V(4,I) = 0.D0
      ENDDO

      pi = acos(-1.0d0)
```

```
C     See if burns are on or off and find total number of burns

      do i = 1,PGALIM.MINBURN
         onoff(i) = 1
      enddo

      k = PGALIM.MINBURN+1
      do i = PGALIM.NUMVAR+1,PGALIM.LEN
         if (p(i).gt.1.0d0) then
            onoff(k) = 1
         else
            onoff(k) = 0
         endif
         k = k + 1
      enddo

      do i = 1, PGALIM.MAXBURN
         BURN_NUMBER = onoff(i) + BURN_NUMBER
      enddo

C     Get burn components from the string

      usedburn = 1
      do i = 1, PGALIM.MAXBURN
         if (onoff(i).eq.1) then
            btime(usedburn) = p(i)
            usedburn = usedburn+1
         endif
      enddo


      do i = 1, BURN_NUMBER
         order(i) = 1
         do j = 1, BURN_NUMBER
            if (btime(i).gt.btime(j)) then
               order(i) = order(i)+1
            endif
         enddo
        enddo

      k =1
      do i = 1,BURN_NUMBER
         do j = 1,BURN_NUMBER
            if (btime(i).eq.btime(j).and.(i.ne.j)) then
               same(k) = (i)
            endif
         enddo
         if (same(k).ne.0) then
            k = k + 1
         endif
      enddo

      k = k-1
      do i=1,k
         order(same(i))=order(same(i))+(i-1)
      enddo


      usedburn = 1
      DO I=1,PGALIM.MAXBURN
         if (onoff(i).eq.1) then
            BURN_DELTA_V(1,order(usedburn)) = p(i)
            BURN_DELTA_V(2,order(usedburn)) = p(i+PGALIM.MAXBURN*1)
            BURN_DELTA_V(3,order(usedburn)) = p(i+PGALIM.MAXBURN*2)
```

```
            BURN_DELTA_V(4,order(usedburn)) = p(i+PGALIM.MAXBURN*3)
            usedburn = usedburn + 1
         endif
      ENDDO

c     CALL SATELLITE to obtain state at request time

      time = 0.0d0
      index = 1

      do while (time.le.PGALIM.MAXTIME)

         STATUS = SATELLITE ( name ,           TIME,
     2                        BURN_DELTA_V,    BURN_NUMBER,
     3                        IATMOS_PREBURN,  IATMOS_POSTBURN,
     4                        RHO_ONE_HIGH,    RHO_ONE_LOW,
     5                        EPOCH_YMD,       EPOCH_HMS,
     6                        POSVEL,          ELEMENTS,
     7                        MESSAGE,         FILENAME )

         if(status.ne.0) WRITE(*,*) 'STATUS = ', STATUS

c     This is the check  for elements 1 through 2
            do i = 1,2
               PGALIM.target(i) = PGALIM.reforb(i,index)
            If(DABS(PGALIM.target(i) - elements(i)).gt.PGALIM.tol(i)) then
             bcconst(i)=DABS((PGALIM.target(i)-elements(i)))-PGALIM.tol(i)
                  if (PGALIM.firstdev .eq. 0.0d0) then
                     PGALIM.firstdev = time
                  endif
               else
                  bcconst(i) = 0.0d0
               endif
               totbcconst(i) = bcconst(i)*timeweight + totbcconst(i)
               if (bcconst(i) .gt. maxbcconst(i)) then
                  maxbcconst(i) = bcconst(i)
               endif
            enddo

c     This is the constraint check for elements 3 through 6

            do i = 3,6
               PGALIM.target(i) = PGALIM.reforb(i,index)
                  Mcheck = dacos(dcos((PGALIM.target(i)-
                       elements(i))*pi/180.0d0))*(180.0d0/pi)
               IF (Mcheck.gt.PGALIM.tol(i)) then
                  bcconst(i) = DABS((Mcheck-PGALIM.tol(i)))
                  if (PGALIM.firstdev .eq. 0.0d0) then
                     PGALIM.firstdev = time
                  endif
               else
                  bcconst(i) = 0.0d0
               endif
               totbcconst(i) = bcconst(i)*timeweight + totbcconst(i)

               if (bcconst(i) .gt. maxbcconst(i)) then
                  maxbcconst(i) = bcconst(i)
               endif
            enddo

         time = time + deltat
         index = index + 1
      enddo
```

```
      do i = 1,6
       totbcconst(i) = totbcconst(i)*PGALIM.targetweight(i)
                        *PGALIM.targeton(i)
         bcconsttot = totbcconst(i)+bcconsttot
      enddo
      bcconsttot = bcconsttot+endconsttot


c     Scale Total Deviation to favor more burns

      bcconsttot = bcconsttot/sqrt(dble(BURN_NUMBER))
      bcconsttot = bcconsttot/(PGALIM.MAXTIME/86400.0d0)

C     Find the total Delta V

      do j = 1, Burn_number
         do i = 1, 3
            BURN(i) = BURN_DELTA_V(i+1,j)
         enddo
         MAGBURN(J) = MAG(BURN)
      enddo

      do i = 1, BURN_NUMBER
         deltavtot = MAGBURN(i)+deltavtot
      enddo


C     Find the sum of the Delta_V used for func
C     Use this formulation if outside the box

      if (bcconsttot .gt. 1.0d0) then

         usedburn = 1
         do i = 1, PGALIM.MAXBURN
            if (onoff(i) .eq.1) then
               do j = 2,4
                  comp = BURN_DELTA_V(j,usedburn)**2.0d0
                  deltavsum(i) = comp + deltavsum(i)
               enddo
               usedburn = 1 + usedburn
            else
               do j = 1,3
                  comp = (p(i+j*PGALIM.MAXBURN)-0.0d0)**2.0d0
                  deltavsum(i) = deltavsum(i) + comp
               enddo
            endif
               vconst = dsqrt(deltavsum(i))+vconst
         enddo

C     Use this formulation if inside the box

      else

         usedburn = 1
         do i = 1, PGALIM.MAXBURN
            do j = 1,3
               comp = (p(i+j*PGALIM.MAXBURN))**2.0d0
               deltavsum(i) = comp+deltavsum(i)
            enddo
            vconst = dsqrt(deltavsum(i))+vconst
         enddo

      endif
```

```
      vconst = vconst * PGALIM.targetweight(7)

      timeconst = (PGALIM.MAXTIME-PGALIM.firstdev)/86400.0d0

C     FINALLY, calculate func associated with given string

      if (pgalim.eflag .eq. .false.) then
         if (bcconsttot.lt.4000.0d0) then
            func = (bcconsttot)*4.0d-2 + vconst*4.0d-2
         else
            func = bcconsttot*0.000001d0 + vconst*0.5d0 + timeconst
         endif
      else
         func = PGALIM.MINERROR
      endif

C     Write the output the last time through func ---------------

      if ((pgalim.end .eq. .true.) .or. (pgalim.plot .eq. .true.)) then

         write(*,*) 'func=',func

         do i = 1,6
            if (PGALIM.targeton(i) .eq. 1) then
               write(*,100)
i,PGALIM.target(i),elements(i),PGALIM.target(i)-elements(i)
            else
               write(*,101) i,PGALIM.target(i),elements(i)
            endif
         enddo

         write(*,*) 'First Deviation Occurs at time=',
                    PGALIM.FIRSTDEV/86400.0d0

         do i = 1,6
            write(*,*) 'Max Deviation of element',i,maxbcconst(i)
         enddo
         do i = 1,6
            write(*,*) 'Total Deviation of element', i, totbcconst(i)
         enddo

         deltavtot = 0.0d0
         do i = 1, BURN_NUMBER
            deltavtot = MAGBURN(i)+deltavtot
         enddo

         write(*,*) 'Total DeltaV=',deltavtot
         write(*,*) 'Number of Burns=',BURN_NUMBER

         write(*,*) 'BURN TABLE------------------------------'

         do i = 1, BURN_NUMBER
            write(*,*)      i,'      Time=',BURN_DELTA_V(1,i),'Magnitude
=',MAGBURN(i)
         enddo

         write(*,*) 'BURN COMPONENTS ------------------------'
         write(*,*) '          TIME    TANGENTIAL     NORMAL       RADIAL
BURN NUMBER'

         usedburn = 1
         do i = 1, PGALIM.MAXBURN
            if (onoff(i).eq.1) then
            write(*,50) p(i),p(i+PGALIM.MAXBURN*1),p(i+2*pgalim.maxburn)
```

```
                              ,p(i+3*pgalim.maxburn),order(usedburn)
               usedburn = usedburn + 1
               else
               write(*,51)
p(i),p(i+PGALIM.MAXBURN*1),p(i+2*pgalim.maxburn),p(i+3*pgalim.maxburn)
               endif
            enddo
         endif

         if (PGALIM.plot .eq. .true.) then
c             call plots(BURN_NUMBER,BURN_DELTA_V,PGALIM.MAXTIME)
         endif

 50    FORMAT(F14.4,F12.6,F12.6,F12.6,F10.0)
 51    FORMAT(F14.4,F12.6,F12.6,F12.6)
 100   Format(I4,F16.8,F16.8,F16.8)
 101   Format(I4,F16.8,F16.8)
 150   Format(I4,F16.8,F16.8)

         return
         end
```

C-1-2 DEFREFORB

```
Subroutine DefineRefOrb
C-------------------------------------------------------------------------
C
C                              Subroutine DefineRefOrb
C
C  This file performs the steps necessary to define the reference orbit
C  for the GA optimal satellite maintenance program.
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C          Ronald J. Proulx
C          Draper Laboratory
C
C
C-------------------------------------------------------------------------
      implicit none

C-------- Include necessary modules --------------------

      include 'pmern.h'
      include 'frc.h'
      include 'pgalim.h'

C----- Define variables  ------------------

      integer*4      satellite
      external       satellite

      integer*4      max_list_length
      parameter     (max_list_length  = 52)

      integer*4      status
      integer*4      burn_number
      Integer*4      iatmos_preburn / 1 /
      integer*4      iatmos_postburn / 1 /
      integer*4      month,hour,minute
      integer*4 index,i
```

```
      real*8          day(maxvar)
      real*8          time
      real*8          burn_delta_v(4,max_list_length)
      REAL*8          RHO_ONE_HIGH      / 0.0 D0 /
      REAL*8          RHO_ONE_LOW       / 0.0 D0 /
      real*8          epoch_ymd
      real*8          epoch_hms
      real*8          posvel(6)
      real*8          elements (17)
      real*8          second(maxvar)
      real*8          btime(max_list_length)
      real*8 deltat

      character*(6)   refname /'pmernr'/
      CHARACTER*(72) MESSAGE
      CHARACTER*(12) FILENAME


c     CALL SATELLITE to define reference array ------------------

      deltat = 86400.0d0/PGALIM.TIMESPERDAY

      index = 1

      TIME              = 0.D0
      BURN_NUMBER       = 0
      DO I=1,PGALIM.MAXBURN
         BURN_DELTA_V(1,I) = 0.D0
         BURN_DELTA_V(2,I) = 0.D0
         BURN_DELTA_V(3,I) = 0.D0
         BURN_DELTA_V(4,I) = 0.D0
      ENDDO


      if (PGALIM.REFFLAG .eq. .false.) then

         do while (time.le.PGALIM.MAXTIME)

            STATUS = SATELLITE ( refname ,           TIME,
     2                           BURN_DELTA_V,    BURN_NUMBER,
     3                           IATMOS_PREBURN, IATMOS_POSTBURN,
     4                           RHO_ONE_HIGH,    RHO_ONE_LOW,
     5                           EPOCH_YMD,       EPOCH_HMS,
     6                           POSVEL,          ELEMENTS,
     7                           MESSAGE,         FILENAME )

            if(status.ne.0) WRITE(*,*) 'STATUS = ', STATUS

            do i = 1,6
               PGALIM.reforb(i,index) = elements(i)
            enddo

            time = time + deltat
            index = index + 1

         enddo
         PGALIM.REFFLAG = .true.

      endif


      return
      end
```

C-1-3 CALCFIRSTDEVTIME

```
subroutine calcfirstdevtime
C-----------------------------------------------------------------------
C
C                              Subroutine Calcfirstdevtime
C
C  This file determines the time of first violation of the given box
C  Constraints for the GA optimization approach
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C          Ronald J. Proulx
C          Draper Laboratory
C
C
C
C-----------------------------------------------------------------------
      implicit none

C-------- Include necessary modules -------------------

      include 'pmern.h'
      include 'frc.h'
      include 'pgalim.h'

C----- Define variables ------------------

      integer*4       satellite
      external        satellite

      integer*4       max_list_length
      parameter      (max_list_length  = 52)

      integer*4       status
      integer*4       burn_number
      Integer*4       iatmos_preburn / 1 /
      integer*4       iatmos_postburn / 1 /
      integer*4       month,hour,minute
      integer*4 index,i

      real*8          day(maxvar)
      real*8          time
      real*8          burn_delta_v(4,max_list_length)
      REAL*8          RHO_ONE_HIGH      / 0.0 D0 /
      REAL*8          RHO_ONE_LOW       / 0.0 D0 /
      real*8          epoch_ymd
      real*8          epoch_hms
      real*8          posvel(6)
      real*8          elements (17)
      real*8          second(maxvar)
      real*8          btime(max_list_length)
      real*8          firstdev,deltat,pi,Mcheck

      character*(6)  name  /'pmern1'/
      CHARACTER*(72) MESSAGE
      CHARACTER*(12) FILENAME

C Initialize variables ************8

      deltat = 86400.0d0/PGALIM.TIMESPERDAY

      firstdev = 0.0d0
```

```
      TIME              = 0.D0
      BURN_NUMBER       = 0
      DO I=1,PGALIM.MAXBURN
         BURN_DELTA_V(1,I) = 0.D0
         BURN_DELTA_V(2,I) = 0.D0
         BURN_DELTA_V(3,I) = 0.D0
         BURN_DELTA_V(4,I) = 0.D0
      ENDDO

      index = 1

C Until a violation occurs, call satellite to obtain current state **

      do while (firstdev.eq.0.0d0)

         STATUS = SATELLITE ( name ,           TIME,
     2                        BURN_DELTA_V,    BURN_NUMBER,
     3                        IATMOS_PREBURN, IATMOS_POSTBURN,
     4                        RHO_ONE_HIGH,    RHO_ONE_LOW,
     5                        EPOCH_YMD,       EPOCH_HMS,
     6                        POSVEL,          ELEMENTS,
     7                        MESSAGE,         FILENAME )

         if(status.ne.0) WRITE(*,*) 'STATUS = ', STATUS

C Check if the current state causes a violation ******

         do i = 1,2
            PGALIM.target(i) = PGALIM.reforb(i,index)
            If  (DABS(PGALIM.target(i)  -  elements(i)).gt.PGALIM.tol(i))
then
               firstdev = time
            endif
         enddo


         do i = 3,6
            pi = acos(-1.0d0)
            PGALIM.target(i) = PGALIM.reforb(i,index)
            Mcheck           =          dacos(dcos((PGALIM.target(i)-
elements(i))*pi/180.0d0))*(180.0d0/pi)
            IF (Mcheck.gt.PGALIM.tol(i)) then
               firstdev = time
            endif
         enddo

         time = time+deltat
         index = index + 1

      enddo

      PGALIM.UPLIM(1) = (firstdev)/(PGALIM.MAXTIME/2.0d0)

      return
      end
```

C-2 Global Station Keeping Approach DSST Input Decks

In order to estimate the station keeping requirements for a given satellite, the definition of two orbits is required: a reference or desired orbit and an actual or perturbed

orbit. For the PGAPack/DSST simulation of these orbits it was necessary to define DSST input decks which could simulate both the desired and actual behaviors. These input decks are presented in the following sections. The reference orbit input deck (which was the same for both cases) is first presented. This is then followed by the actual orbit input deck used in the first case and the input deck used to simulate the actual orbit for case two. Proper flag and keyword values were determined from personal communications with Dr. Ronald Proulx, The Charles Stark Draper Laboratory.

C-2-1 Reference Orbit

```
C
C
C       PMEF FILE FOR GLOBAL STATION KEEPING REFERENCE ORBIT
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.2000032100000000D+08 PME_DATE          1
   0.0148000000000000D+06 PME_TIME          2
   0.1049688389359363D+05 ELS_KEP(1)        3
   0.3327635613329539D+00 ELS_KEP(2)        4
   0.1165576926449117D+03 ELS_KEP(3)        5
   0.0000000000000000D+03 ELS_KEP(4)        6
   0.2600000000000000D+03 ELS_KEP(5)        7
   0.0000000000000000D+03 ELS_KEP(6)        8
   0.0000000000000000D+00 ELS_EQUIN(1)      9
   0.0000000000000000D+00 ELS_EQUIN(2)     10
   0.0000000000000000D+00 ELS_EQUIN(3)     11
   0.0000000000000000+00 ELS_EQUIN(4)      12
   0.0000000000000000D+00 ELS_EQUIN(5)     13
   0.0000000000000000D+00 ELS_EQUIN(6)     14
   0.0000000000000000D+00 POSVEL(1)        15
   0.0000000000000000D+00 POSVEL(2)        16
   0.0000000000000000D+00 POSVEL(3)        17
   0.0000000000000000D+00 POSVEL(4)        18
   0.0000000000000000D+00 POSVEL(5)        19
   0.0000000000000000D+00 POSVEL(6)        20
   0.2000000000000000D+01 PME_CD           21
   0.0000000000000000D+00 PME_RHO_ONE      22
   0.5000000000000000D-04 SMA_SIGMA        23
   0.5000000000000000D-04 INC_SIGMA        24
   0.5000000000000000D-04 ASC_SIGMA        25
   0.1250000000000000D+04 PME_SCMASS       26
   0.4330000000000000D-04 PME_SCAREA       27
   0.8640000000000000D+05 PME_STEPSIZE     28
   0.1400000000000000D+02 DP_SPARE1        29
   0.1130000000000000D+03 DP_SPARE2        30
   0.0000000000000000D+00 DP_SPARE3        31
   0.0000000000000000D+00 DP_SPARE4        32
   0.0000000000000000D+00 DP_SPARE5        33
   0.0000000000000000D+00 DP_SPARE6        34
          1               PME_RETRO        35
         12               PME_KEP_SYS      36
         12               POS_VEL_SYS      37
          1               GEN_METHOD       38
          1               ATMOS_MODEL      39
```

```
    840401              JACRB_DATE      40
       123              JACRB_SSS       41
    840401              SLP1950_DATE    42
       456              SLP1950_SSS     43
    840401              SLPTOD_DATE     44
       789              SLPTOD_SSS      45
    840401              TIMECF_DATE     46
       123              TIMECF_SSS      47
         2              HARRIS_MODEL    48
        10              POTNTL_MODEL    49
        50              PME_NMAX        50
         0              PME_MMAX        51
         1              PME_IZONAL      52
         1              PME_IJ2J2       53
         0              PME_NMAXRS      54
         0              PME_MMAXRS      55
         3              PME_ITHIRD      56
         2              PME_INDDRG      57  2 = DRAG OFF
         2              PME_ISZAK       58
         2              PME_INDSOL      59  2 = SOLRAD OFF
         2              PME_JSHPER      60
         1              PME_JZONAL      61
         1              PME_JMDALY      62
         2              PME_INP_TYPE    63
        12              PME_EQUI_SYS    64
        11              INTEG_FRAME     65
        19              OUTPUT_FRAME    66
         0              PME_NSTATE      67
         1              PME_SPSHPER     68
         2              PME_KSPCF       69
         4              PME_INDSET      70
         0              INT_SPARE1      71
         0              INT_SPARE2      72
         0              INT_SPARE3      73
         0              INT_SPARE4      74
         0              INT_SPARE5      75
         0              INT_SPARE6      76
         0              INT_SPARE7      77
         0              INT_SPARE8      78
         0              INT_SPARE9      79
         0              INT_SPARE10     80
```

C-2-2 Case 1 Actual Orbit

```
C
C
C      PMEF FILE FOR GLOBAL STATION KEEPING CASE 1
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.2000032100000000D+08 PME_DATE         1
   0.0148000000000000D+06 PME_TIME         2
   0.1049688389359363D+05 ELS_KEP(1)       3
   0.3327635613329539D+00 ELS_KEP(2)       4
   0.1165576926449117D+03 ELS_KEP(3)       5
   0.0000000000000000D+03 ELS_KEP(4)       6
   0.2600000000000000D+03 ELS_KEP(5)       7
   0.0000000000000000D+03 ELS_KEP(6)       8
   0.0000000000000000D+00 ELS_EQUIN(1)     9
   0.0000000000000000D+00 ELS_EQUIN(2)    10
   0.0000000000000000D+00 ELS_EQUIN(3)    11
   0.0000000000000000+00 ELS_EQUIN(4)     12
   0.0000000000000000D+00 ELS_EQUIN(5)    13
   0.0000000000000000D+00 ELS_EQUIN(6)    14
```

```
0.0000000000000000D+00 POSVEL(1)       15
0.0000000000000000D+00 POSVEL(2)       16
0.0000000000000000D+00 POSVEL(3)       17
0.0000000000000000D+00 POSVEL(4)       18
0.0000000000000000D+00 POSVEL(5)       19
0.0000000000000000D+00 POSVEL(6)       20
0.2000000000000000D+01 PME_CD          21
0.0000000000000000D+00 PME_RHO_ONE     22
0.5000000000000000D-04 SMA_SIGMA       23
0.5000000000000000D-04 INC_SIGMA       24
0.5000000000000000D-04 ASC_SIGMA       25
0.1250000000000000D+04 PME_SCMASS      26
0.4330000000000000D-04 PME_SCAREA      27
0.8640000000000000D+05 PME_STEPSIZE    28
0.1400000000000000D+02 DP_SPARE1       29
0.1130000000000000D+03 DP_SPARE2       30
0.0000000000000000D+00 DP_SPARE3       31
0.0000000000000000D+00 DP_SPARE4       32
0.0000000000000000D+00 DP_SPARE5       33
0.0000000000000000D+00 DP_SPARE6       34
         1             PME_RETRO       35
        12             PME_KEP_SYS     36
        12             POS_VEL_SYS     37
         1             GEN_METHOD      38
         1             ATMOS_MODEL     39
    840401             JACRB_DATE      40
       123             JACRB_SSS       41
    840401             SLP1950_DATE    42
       456             SLP1950_SSS     43
    840401             SLPTOD_DATE     44
       789             SLPTOD_SSS      45
    840401             TIMECF_DATE     46
       123             TIMECF_SSS      47
         2             HARRIS_MODEL    48
        10             POTNTL_MODEL    49
        21             PME_NMAX        50
        21             PME_MMAX        51
         1             PME_IZONAL      52
         1             PME_IJ2J2       53
        21             PME_NMAXRS      54
        21             PME_MMAXRS      55
         1             PME_ITHIRD      56
         1             PME_INDDRG      57  2 = DRAG OFF
         1             PME_ISZAK       58
         1             PME_INDSOL      59  2 = SOLRAD OFF
         2             PME_JSHPER      60
         1             PME_JZONAL      61
         1             PME_JMDALY      62
         2             PME_INP_TYPE    63
        12             PME_EQUI_SYS    64
        11             INTEG_FRAME     65
        19             OUTPUT_FRAME    66
         0             PME_NSTATE      67
         1             PME_SPSHPER     68
         2             PME_KSPCF       69
         1             PME_INDSET      70
         0             INT_SPARE1      71
         0             INT_SPARE2      72
         0             INT_SPARE3      73
         0             INT_SPARE4      74
         0             INT_SPARE5      75
         0             INT_SPARE6      76
         0             INT_SPARE7      77
         0             INT_SPARE8      78
```

```
           0                INT_SPARE9      79
           0                INT_SPARE10     80
```

C-2-3 Case 2 Actual Orbit

```
C
C
C       PMEF FILE GLOBAL STATION KEEPING CASE 2
C
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.2000032100000000D+08 PME_DATE          1
   0.0148000000000000D+06 PME_TIME          2
   0.1049597389359363D+05 ELS_KEP(1)        3
   0.3330605613329539D+00 ELS_KEP(2)        4
   0.1165081926449117D+03 ELS_KEP(3)        5
   0.3595020000000000D+03 ELS_KEP(4)        6
   0.2590200000000000D+03 ELS_KEP(5)        7
   0.0009000000000000D+03 ELS_KEP(6)        8
   0.0000000000000000D+00 ELS_EQUIN(1)      9
   0.0000000000000000D+00 ELS_EQUIN(2)     10
   0.0000000000000000D+00 ELS_EQUIN(3)     11
   0.0000000000000000+00 ELS_EQUIN(4)      12
   0.0000000000000000D+00 ELS_EQUIN(5)     13
   0.0000000000000000D+00 ELS_EQUIN(6)     14
   0.0000000000000000D+00 POSVEL(1)        15
   0.0000000000000000D+00 POSVEL(2)        16
   0.0000000000000000D+00 POSVEL(3)        17
   0.0000000000000000D+00 POSVEL(4)        18
   0.0000000000000000D+00 POSVEL(5)        19
   0.0000000000000000D+00 POSVEL(6)        20
   0.2000000000000000D+01 PME_CD           21
   0.0000000000000000D+00 PME_RHO_ONE      22
   0.5000000000000000D-04 SMA_SIGMA        23
   0.5000000000000000D-04 INC_SIGMA        24
   0.5000000000000000D-04 ASC_SIGMA        25
   0.1250000000000000D+04 PME_SCMASS       26
   0.4330000000000000D-04 PME_SCAREA       27
   0.8640000000000000D+05 PME_STEPSIZE     28
   0.1400000000000000D+02 DP_SPARE1        29
   0.1130000000000000D+03 DP_SPARE2        30
   0.0000000000000000D+00 DP_SPARE3        31
   0.0000000000000000D+00 DP_SPARE4        32
   0.0000000000000000D+00 DP_SPARE5        33
   0.0000000000000000D+00 DP_SPARE6        34
           1              PME_RETRO        35
          12              PME_KEP_SYS      36
          12              POS_VEL_SYS      37
           1              GEN_METHOD       38
           1              ATMOS_MODEL      39
      840401              JACRB_DATE       40
         123              JACRB_SSS        41
      840401              SLP1950_DATE     42
         456              SLP1950_SSS      43
      840401              SLPTOD_DATE      44
         789              SLPTOD_SSS       45
      840401              TIMECF_DATE      46
         123              TIMECF_SSS       47
           2              HARRIS_MODEL     48
          10              POTNTL_MODEL     49
          21              PME_NMAX         50
          21              PME_MMAX         51
           1              PME_IZONAL       52
```

```
 1              PME_IJ2J2       53
21              PME_NMAXRS      54
21              PME_MMAXRS      55
 1              PME_ITHIRD      56
 1              PME_INDDRG      57   2 = DRAG OFF
 1              PME_ISZAK       58
 1              PME_INDSOL      59   2 = SOLRAD OFF
 2              PME_JSHPER      60
 1              PME_JZONAL      61
 1              PME_JMDALY      62
 2              PME_INP_TYPE    63
12              PME_EQUI_SYS    64
11              INTEG_FRAME     65
19              OUTPUT_FRAME    66
 0              PME_NSTATE      67
 1              PME_SPSHPER     68
 2              PME_KSPCF       69
 1              PME_INDSET      70
 0              INT_SPARE1      71
 0              INT_SPARE2      72
 0              INT_SPARE3      73
 0              INT_SPARE4      74
 0              INT_SPARE5      75
 0              INT_SPARE6      76
 0              INT_SPARE7      77
 0              INT_SPARE8      78
 0              INT_SPARE9      79
 0              INT_SPARE10     80
```

C-3 Global Station Keeping Case 1 Element Deviation Plots

This section contains plots that show the actual orbit's difference from the reference orbit over the entire 90 day period of interest for case one. The uncontrolled deviations are first presented to show that control is indeed necessary to meet the desired constraints, specifically on elements a, e, and M. The controlled deviations resulting from the optimization are then presented. These plots show that the optimization technique was indeed able to maintain the trajectory within the desired constraints over the entire period of interest.

C-3-1 Global Case 1 Uncontrolled Deviations

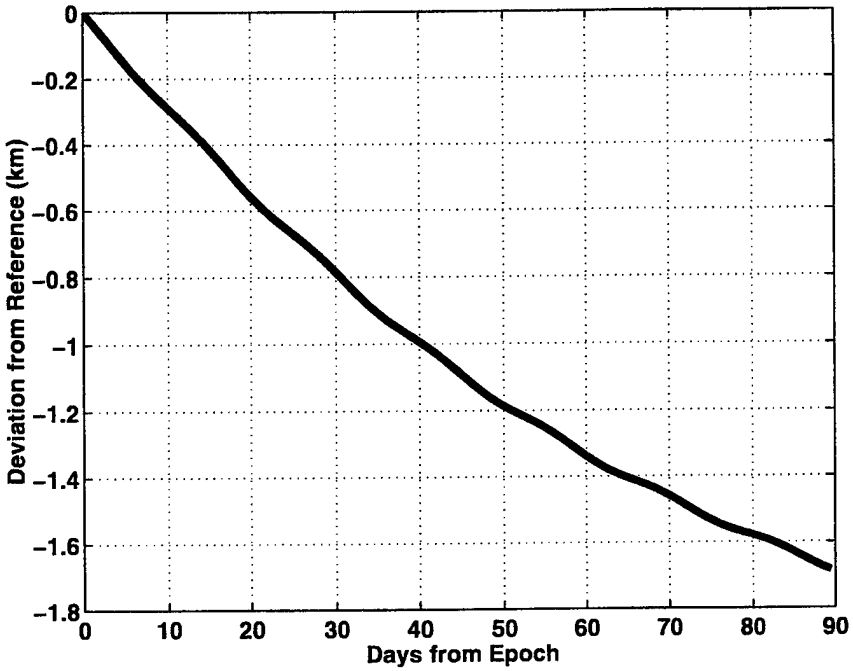


Figure C-1 Global Case 1 Uncontrolled SMA Deviation (Limit = 1°)

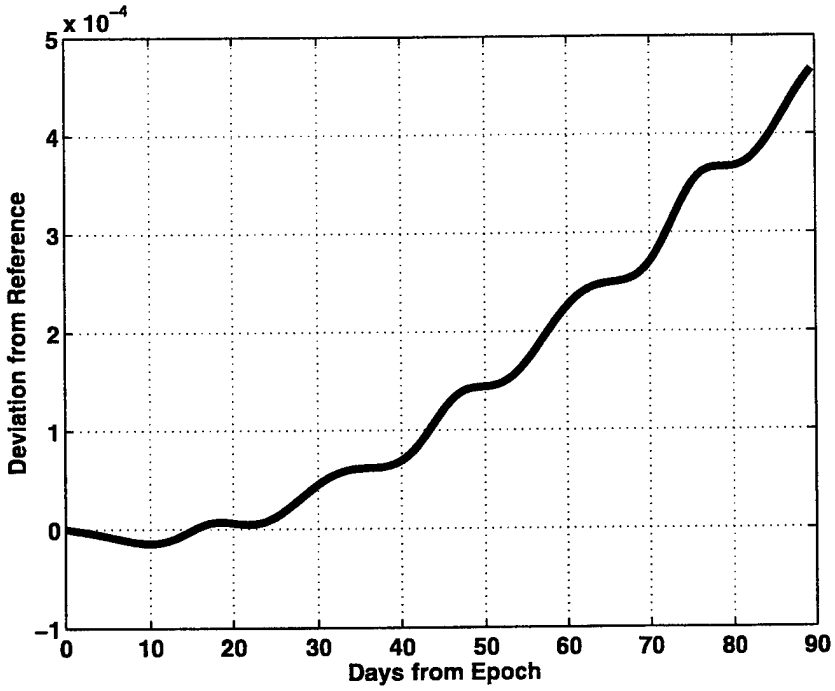


Figure C-2 Global Case 1 Uncontrolled Eccentricity Deviation (Limit = 0.0003)

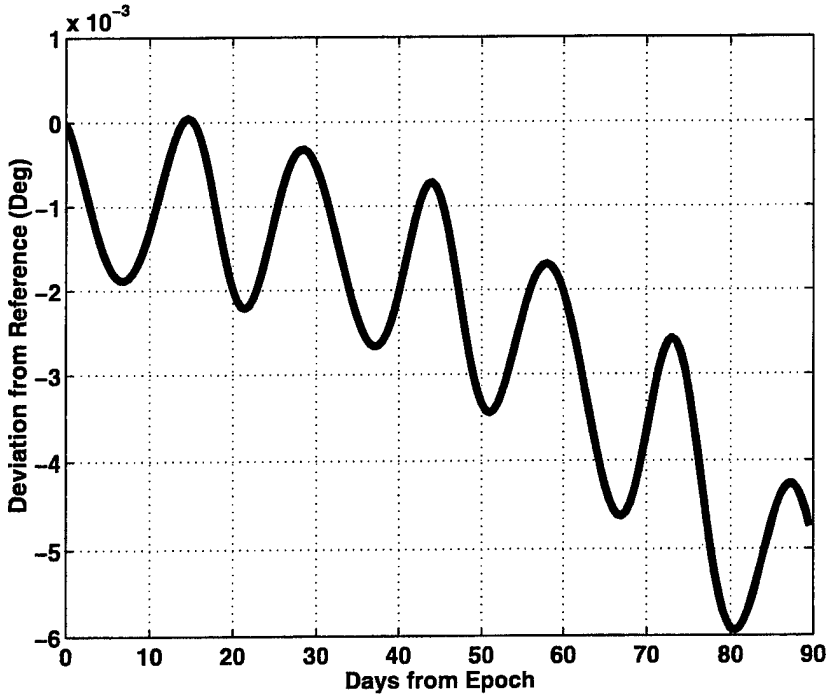


Figure C-3 Global Case 1 Uncontrolled Inclination Deviation (Limit = 0.05°)

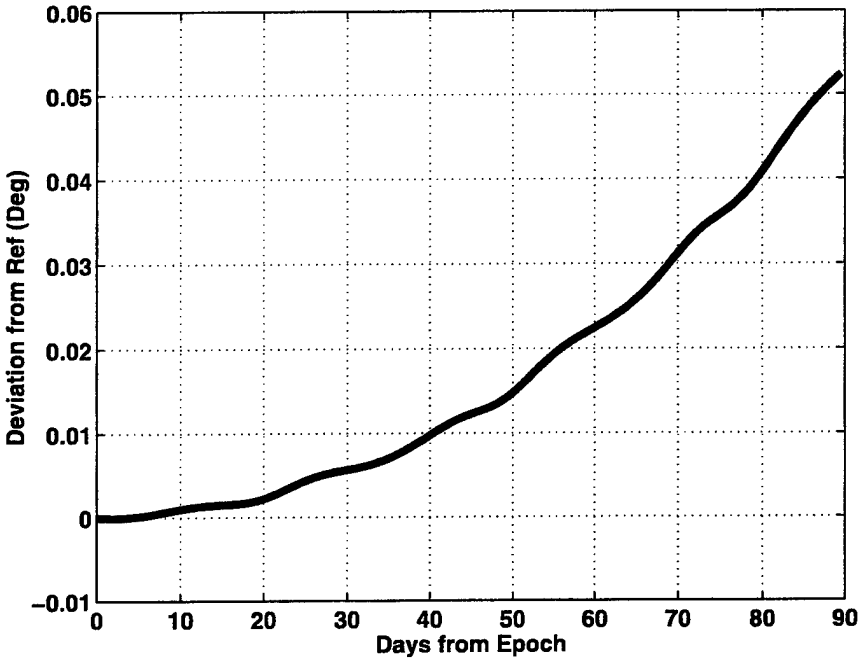


Figure C-4 Global Case 1 Uncontrolled RAAN Deviation (Limit = 0.5°)

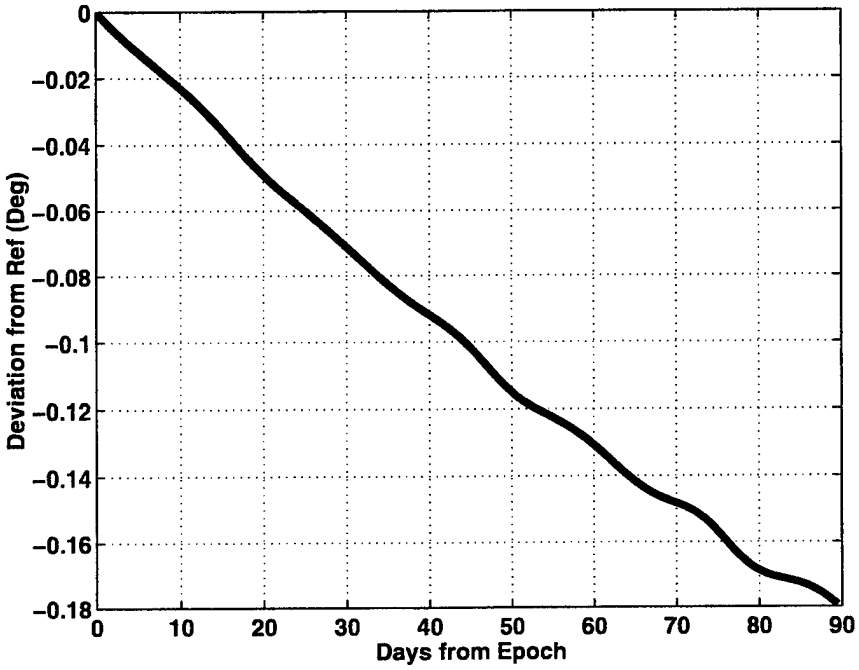


Figure C-5 Global Case 1 Uncontrolled Argument of Perigee Deviation (Limit = 1°)

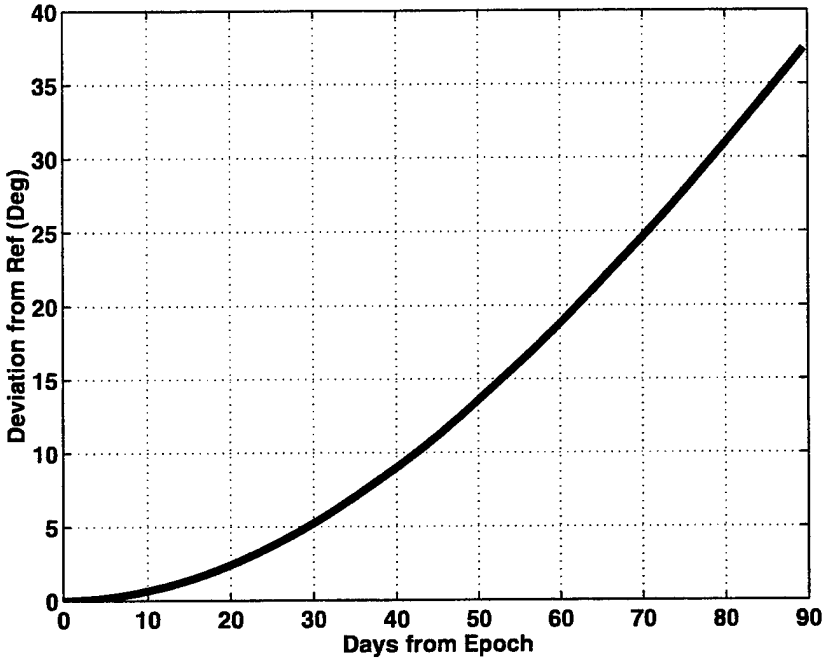


Figure C-6 Global Case 1 Uncontrolled Mean Anomaly Deviation (Limit = 1°)

C-3-2 Global Case 1 Controlled Deviations

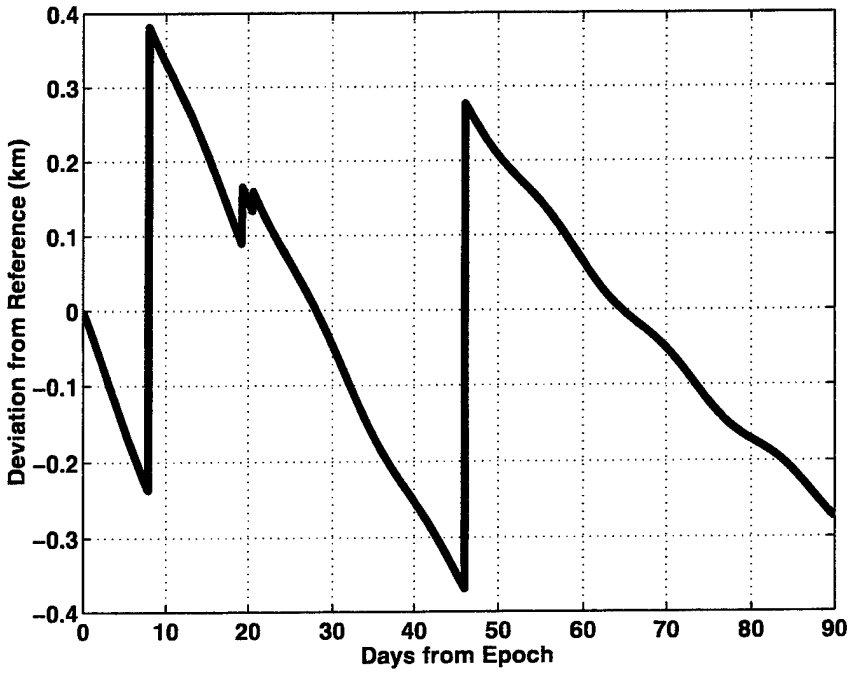


Figure C-7 Global Case 1 Controlled SMA Deviation (Limit = 1 km)

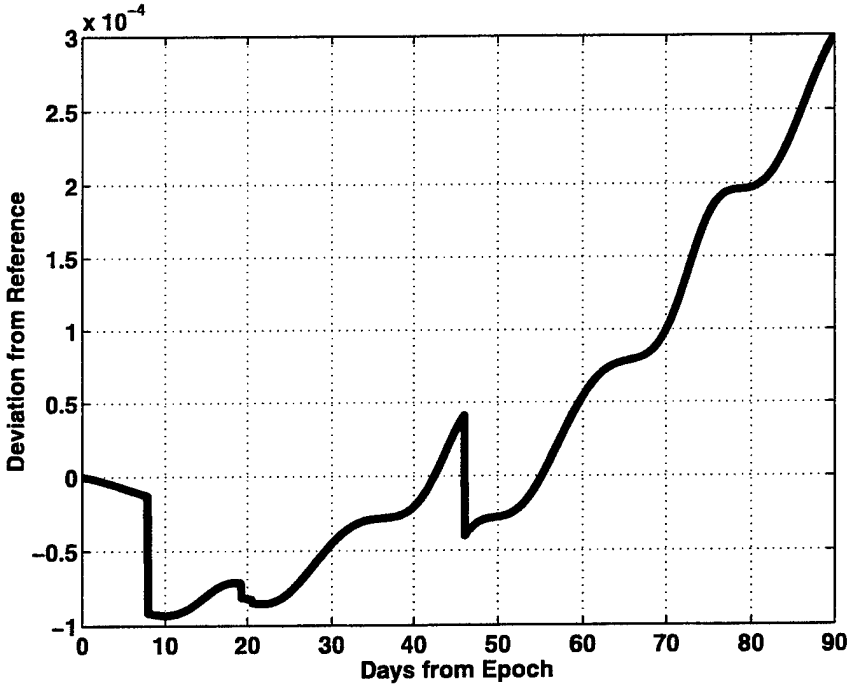


Figure C-8 Global Case 1 Controlled Eccentricity Deviation (Limit = 0.0003)

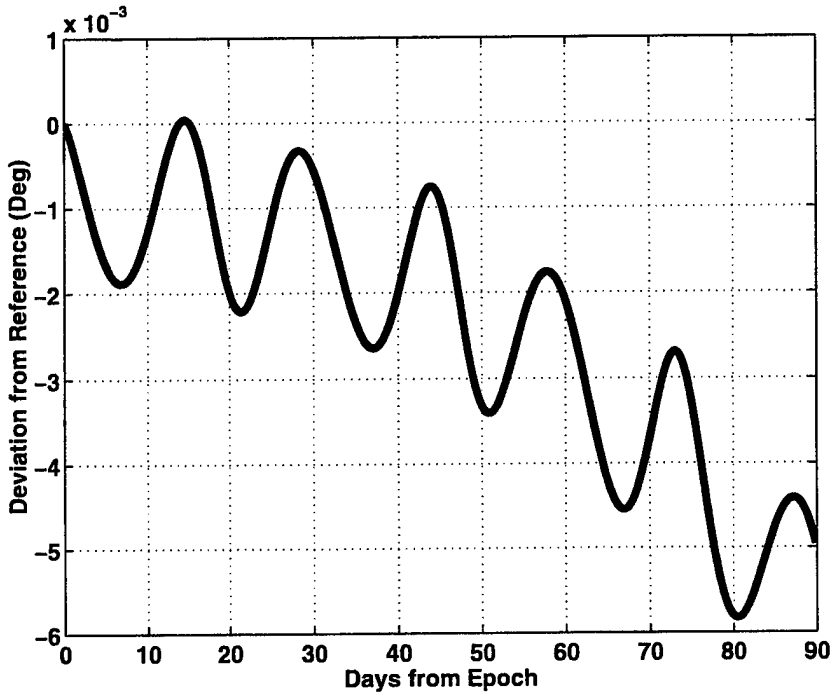


Figure C-9 Global Case 1 Controlled Inclination Deviation (Limit = 0.05°)

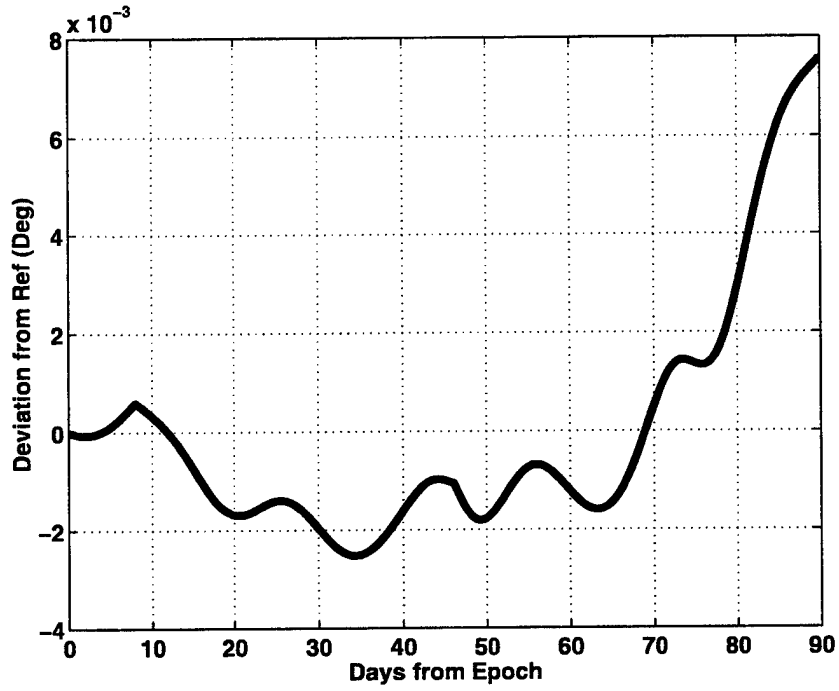


Figure C-10 Global Case 1 Controlled RAAN Deviation (Limit = 0.5°)

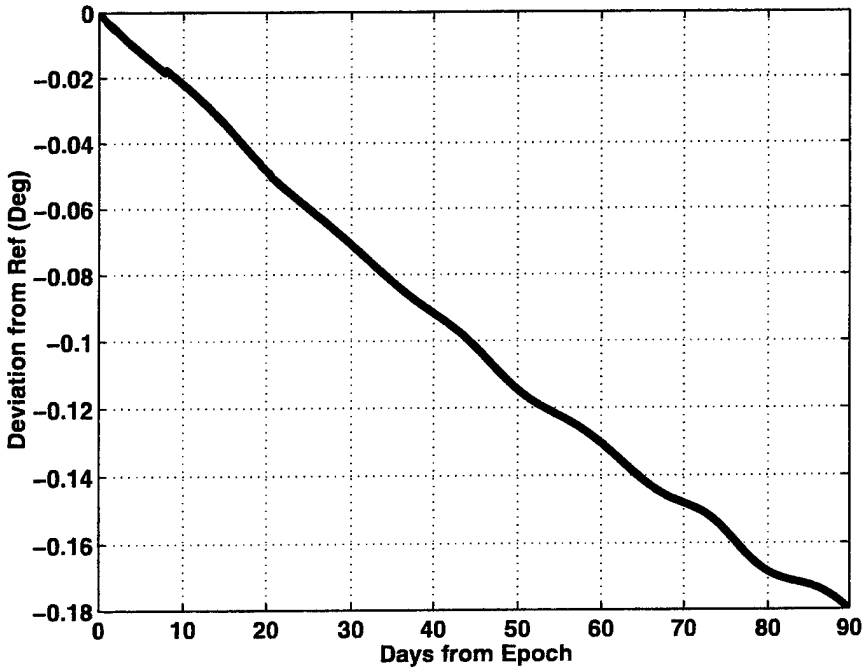


Figure C-11 Global Case 1 Controlled Argument of Perigee Deviation (Limit = 1°)

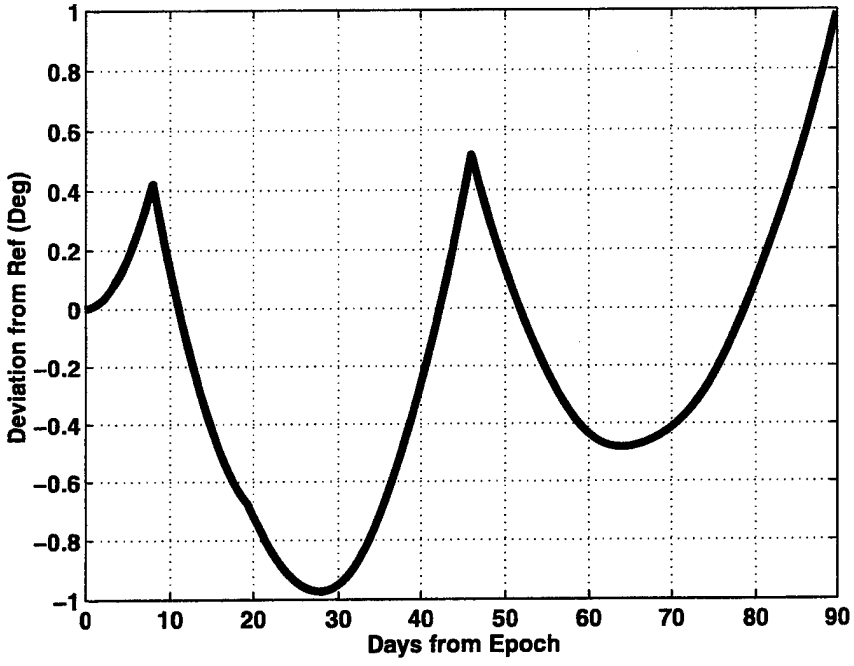


Figure C-12 Global Case 1 Controlled Mean Anomaly Deviation (Limit = 1°)

C-4 Global Station Keeping Case 2 Element Deviation Plots

This section contains plots that show the actual orbit's difference from the reference orbit over the entire 90-day period of interest for case two. The uncontrolled deviations are first presented to show that control is indeed necessary to meet the desired constraints for all elements. The controlled deviations resulting from the optimization are then presented. These plots show that the optimization technique was indeed able to maintain the trajectory within the desired constraints over the entire period of interest, even despite the difficult starting conditions which were imposed on the system.

C-4-1 Global Case 2 Uncontrolled Deviations

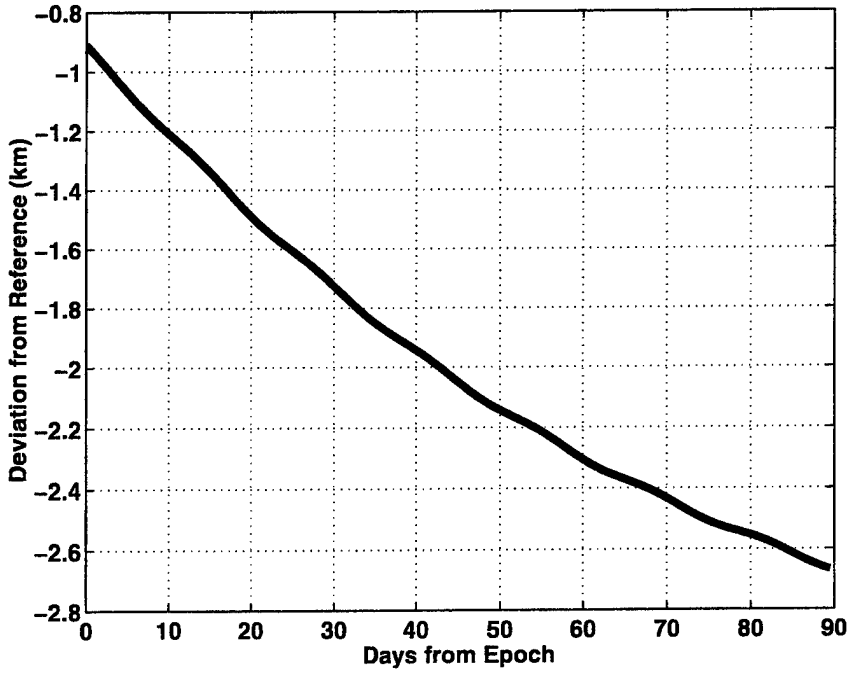


Figure C-13 Global Case 2 Uncontrolled SMA Deviation (Limit = 1 km)

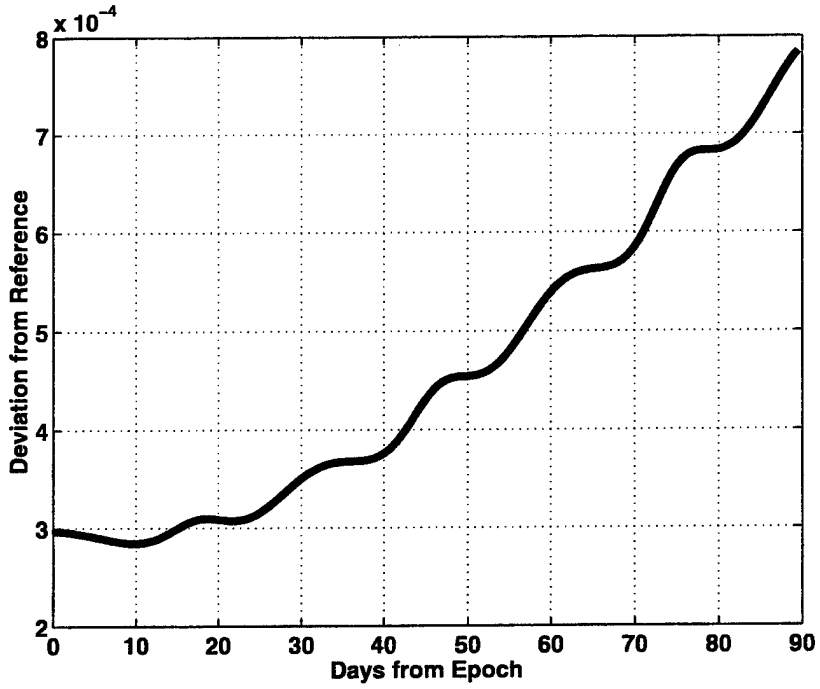


Figure C-14 Global Case 2 Uncontrolled Eccentricity Deviation (Limit = 0.0003)

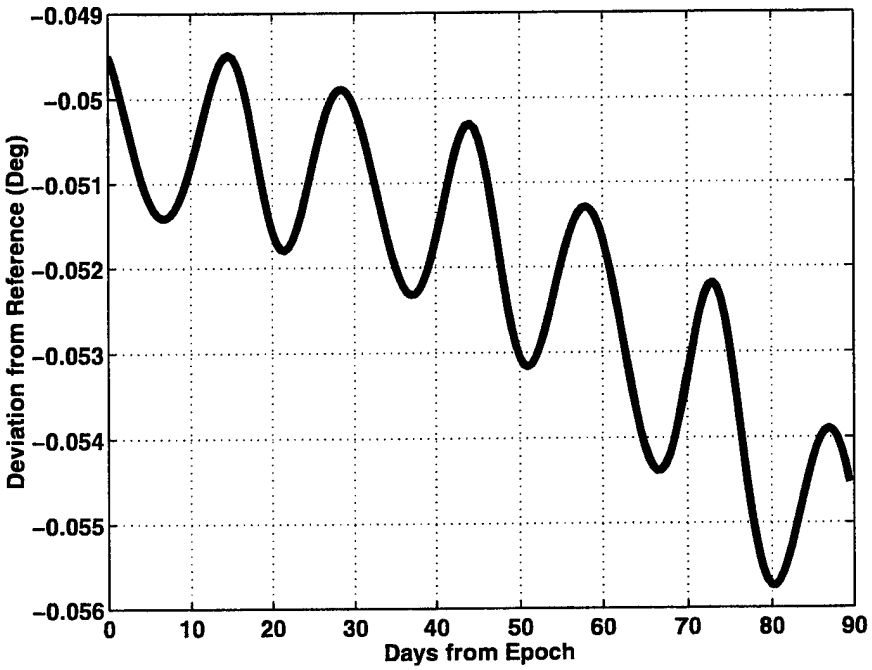


Figure C-15 Global Case 2 Uncontrolled Inclination Deviation (Limit = 0.05°)

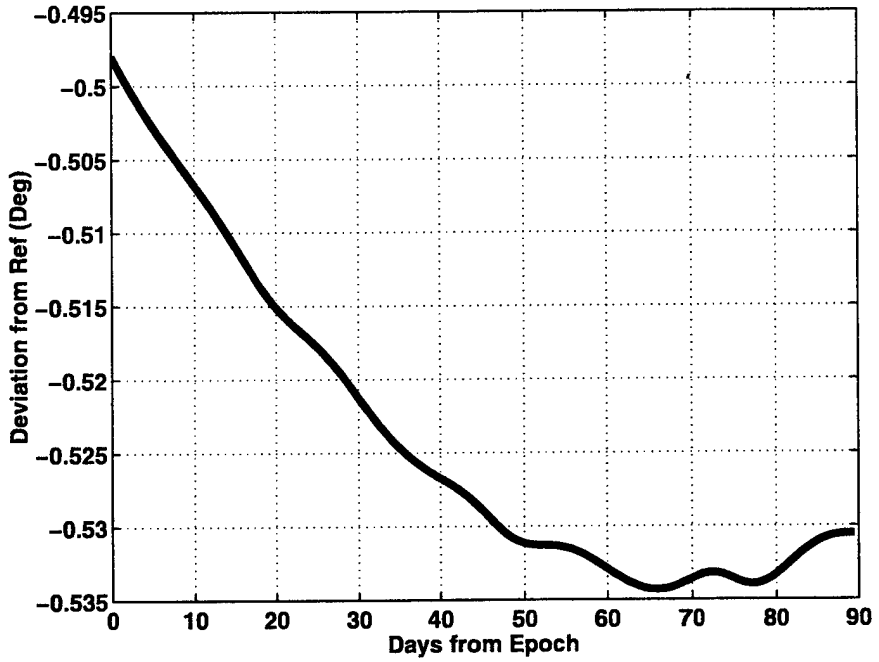


Figure C-16 Global Case 2 Uncontrolled RAAN Deviation (Limit = 0.5°)

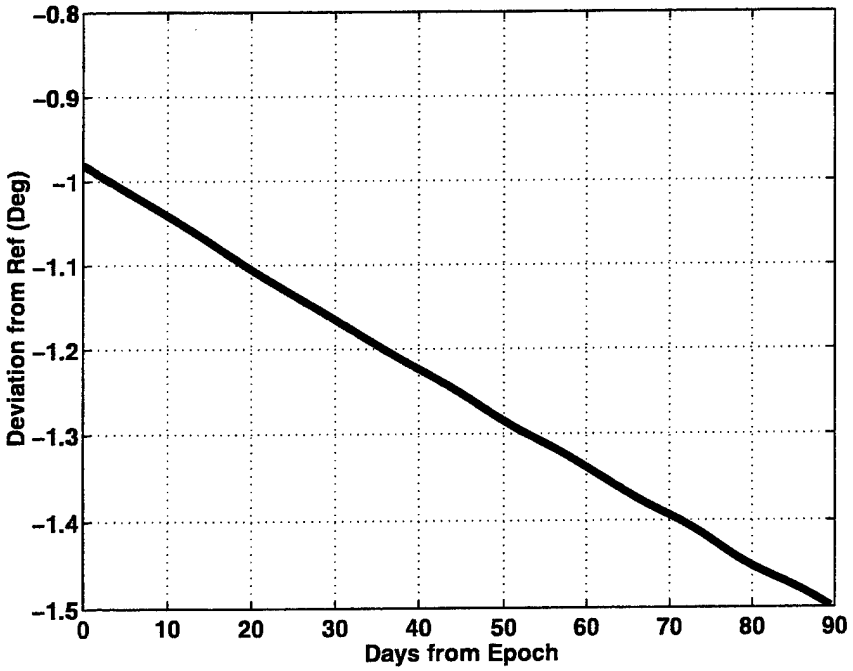


Figure C-17 Global Case 2 Uncontrolled Argument of Perigee Deviation (Limit = 1°)

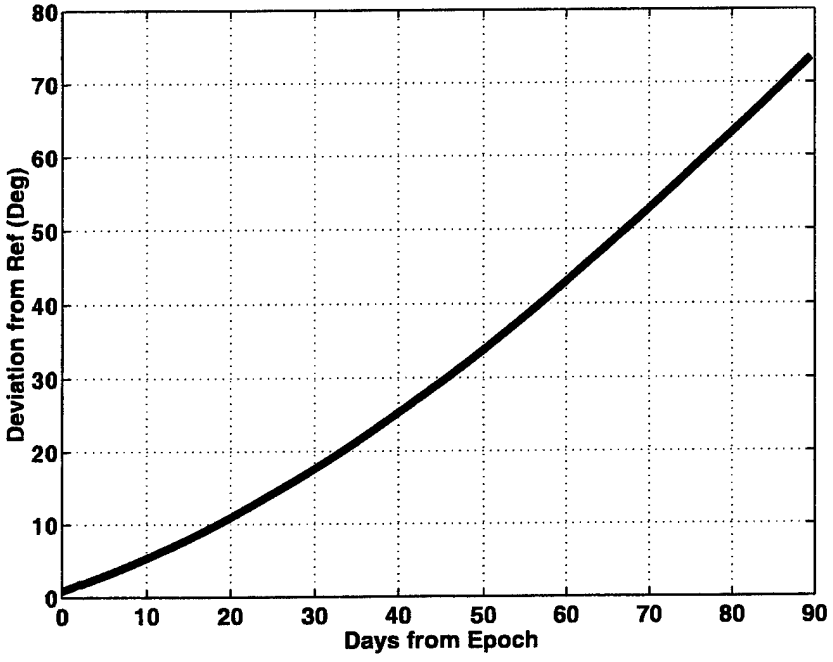


Figure C-18 Global Case 2 Uncontrolled Mean Anomaly Deviation (Limit = 1°)

C-4-2 Global Case 2 Controlled Deviations

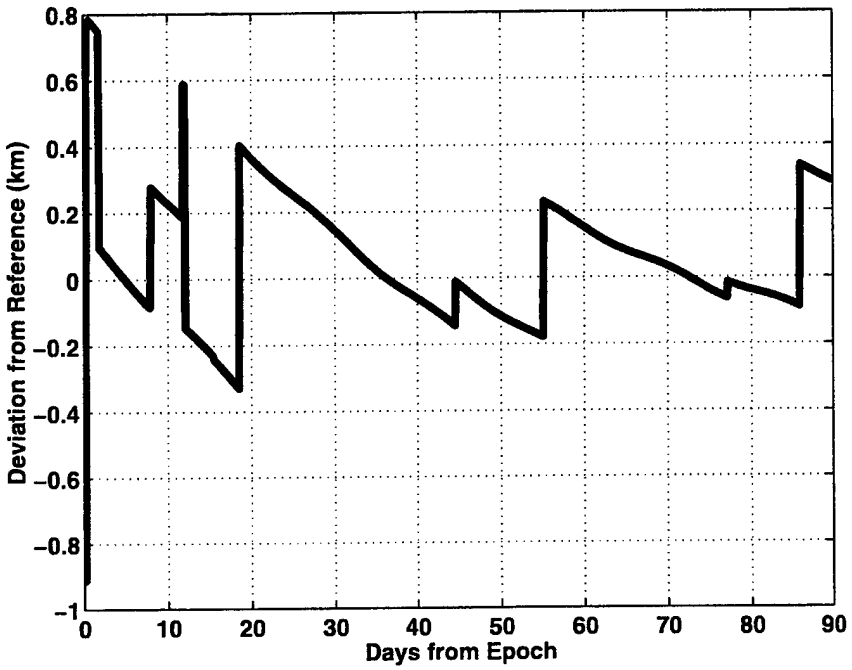


Figure C-19 Global Case 2 Controlled SMA Deviation (Limit = 1 km)

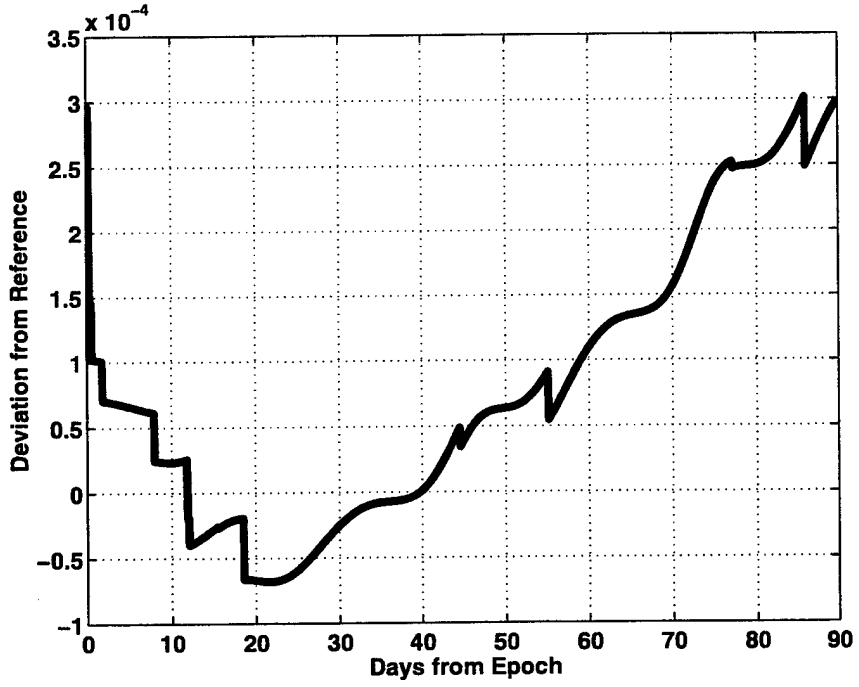


Figure C-20 Global Case 2 Controlled Eccentricity Deviation (Limit = 0.0003)

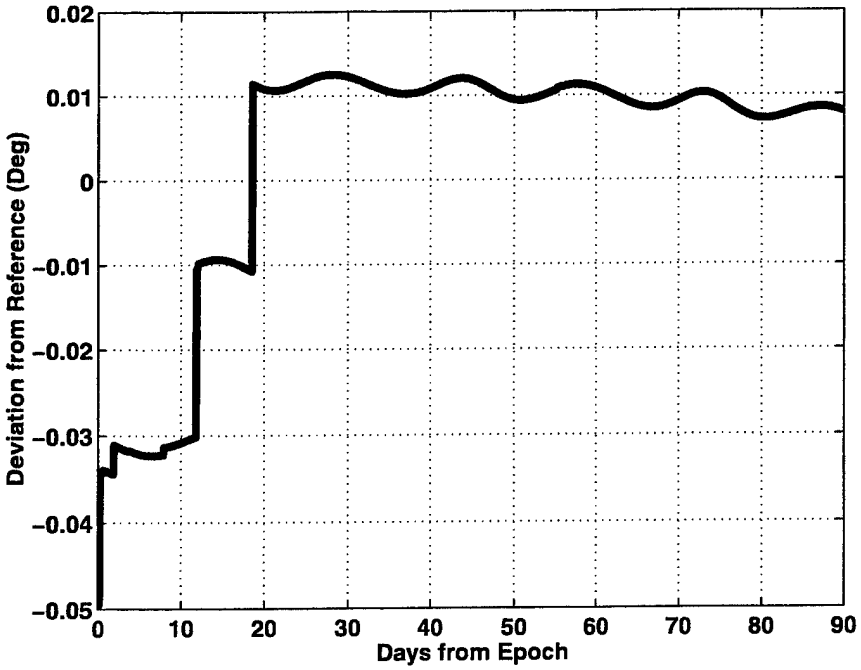


Figure C-21 Global Case 2 Controlled Inclination Deviation (Limit = 0.05°)

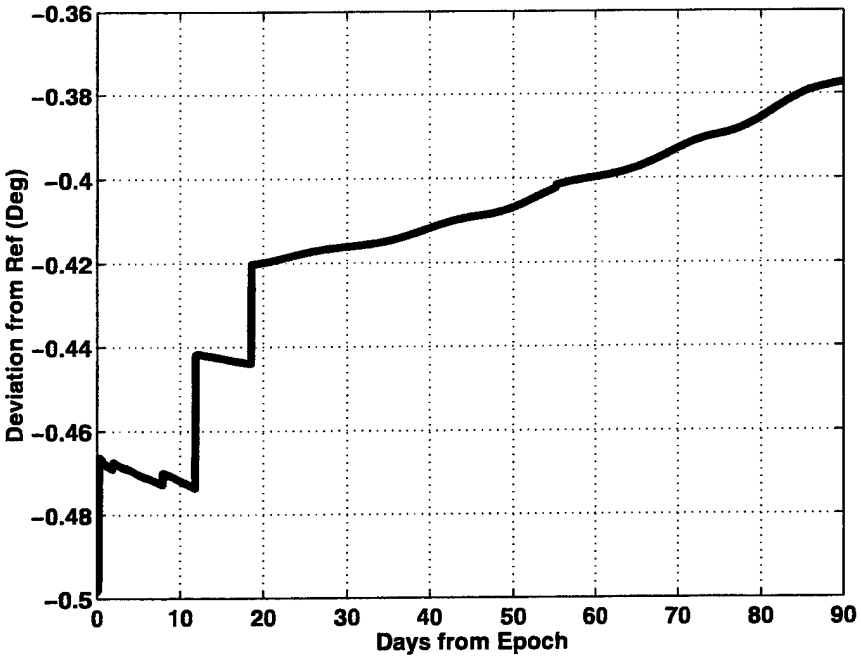


Figure C-22 Global Case 2 Controlled RAAN Deviation (Limit = 0.5°)

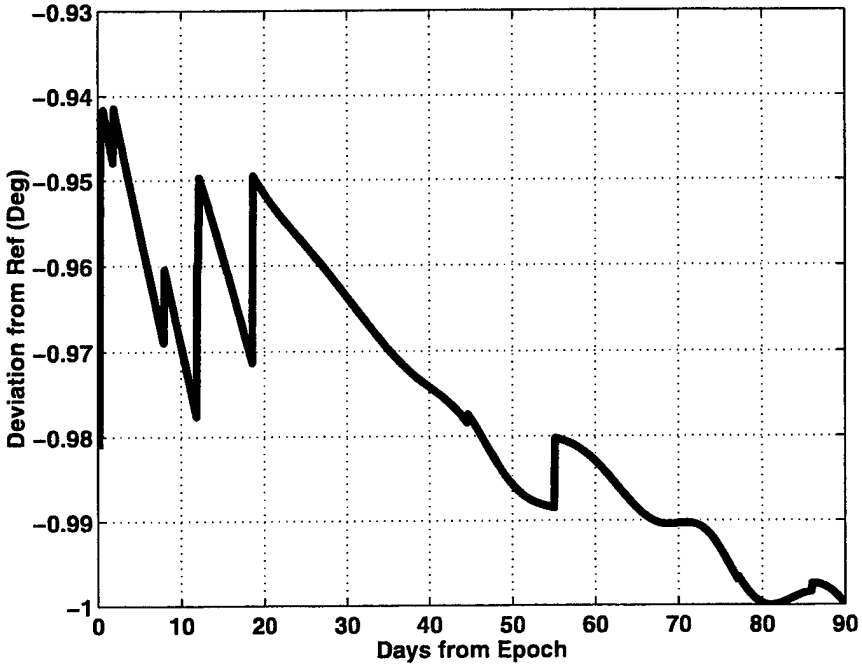


Figure C-23 Global Case 2 Controlled Argument of Perigee Deviation (Limit = 1°)

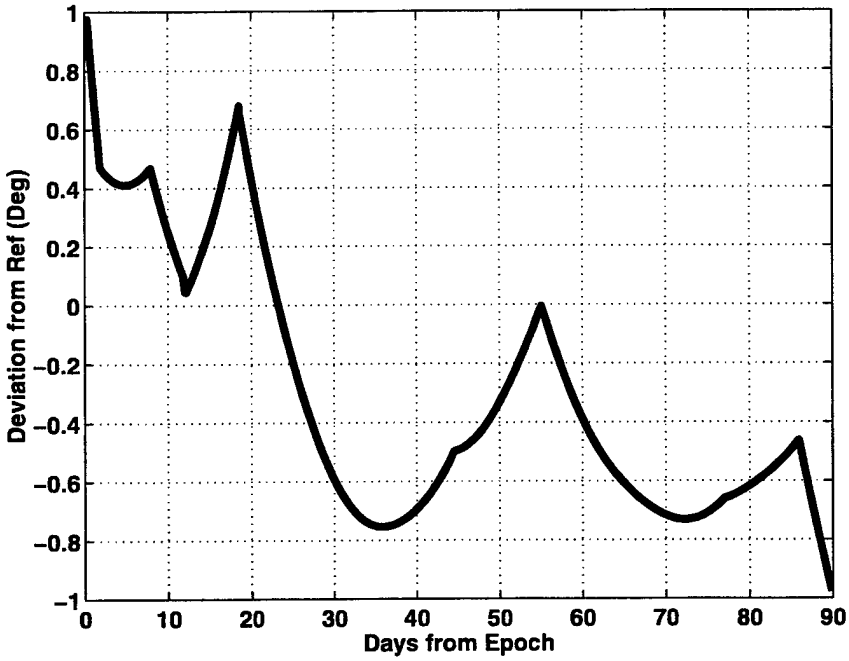


Figure C-24 Global Case 2 Controlled Mean Anomaly Deviation (Limit = 1°)

[This Page Intentionally Left Blank]

Appendix D Operational Station K eeping Approach Data

This appendix contains the code modifications required for the implementation of the operational station keeping approach as well as the results of a number of test cases to which this approach was applied. The code modifications are first presented. These modifications are followed by the results of the application of the operational approach to the global case elements. Finally, results from one year, 2.5 year and 1200 day applications of the approach as a planning tool are included.

D-1 Code Modifications Req uired for Operational Approach Implementation

This section contains printouts of the genetic algorithm and objective function code necessary to implement the operational station keeping approach. Both the genetic algorithm code (PGA_GASK) and the objective function code (FUNC) required modification from the similar routines used to implement the global approach. The most important of these changes was the conversion to a two-layered objective function as discussed in section 5-4-3-4. The specific modifications to each routine can be seen below:

D-1-1 Operational Approach M odifications to PGA_GASK

```
Program pga_opgask
C-----------------------------------------------------------------------
C
C                              Program PGA_OPGASK
C
C  This file contains Genetic Algorithm code for finding optimal number
C   of burns in conjunction with DSST orbit propagator using the
C   operational approach.
C
C  Author: James E. Smith, 2Lt, USAF
C          MIT/ Aero-Astro Dept./ Draper Fellow
C
C
C-----------------------------------------------------------------------
      implicit none
```

```
      include 'pgapackf.h'
      include 'mpif.h'
      include 'pgalim.h'
      include 'pmern.h'

      double precision findbest_nburn
      external         findbest_nburn

      double precision findbest_last
      external         findbest_last

C---------------- Variable Declarations -------------------------------
      integer ctx, p, pop
      integer myid, ierror
      integer number, counter, step
      integer bestindex, i, locnumvar

      integer*4 BURN_NUMBER,repeat
      real*8 BURN_DELTA_V(4,MAXVAR),FIRSTDEV,looptime,lasttime

      double precision best(MAXVAR)

      character*24 fdate
      external fdate

      character*6 name /'pmern1'/

C------------------ Main Program ----------------------------------------

C     Initialize MPI processes

      write(*,*) fdate()

      call MPI_Init(ierror)
      call MPI_Comm_rank(MPI_COMM_WORLD, myid, ierror)

C     Initialize The Variables to Default Values

      PGALIM.totalburns = 0
      PGALIM.totaldeltav = 0.0d0
      PGALIM.epochtime = 0.0d0
      repeat = 0
      lasttime = -10.0d0
      looptime = 1.25d0

      call init_headers
      call init_PGALIM

C     Perform Maximum Drift Time Optimization

      do while (PGALIM.epochtime.le.looptime)

         PGALIM.OPFLAG = 1

         call setlim_nburn
         PGALIM.END = .FALSE.

         PGALIM.FIRSTDEVZEROB = PGALIM.MAXTIME

         call DefineRefOrb

         BURN_NUMBER      = 0
         DO I=1,PGALIM.MAXBURN
            BURN_DELTA_V(1,I) = 0.D0
```

```
            BURN_DELTA_V(2,I) = 0.D0
            BURN_DELTA_V(3,I) = 0.D0
            BURN_DELTA_V(4,I) = 0.D0
         ENDDO

         call initialize_sat(name)

         if (PGALIM.epochtime.gt.0.0d0) then
            pmern.date = pgalim.nextdate
            pmern.time = pgalim.nexttime
            do i = 1,6
               pmern.els_kepler(i) = PGALIM.kepels(i)
            enddo
         endif

         call CalcFirstDevTime(BURN_NUMBER,BURN_DELTA_V,0)

         ctx = PGACreate(PGA_DATATYPE_REAL, PGALIM.LEN, PGALIM.FLAG)

C     This defines the stopping rule and sets the number of iterations
C     in which no change is allowed before stopping.  A number of other
C     PGA parameters are also set.

         call PGASetRealInitRange(ctx,PGALIM.LOWLIM,PGALIM.UPLIM)
         call PGASetStoppingRuleType(ctx,PGA_STOP_NOCHANGE)
         call PGASetStoppingRuleType(ctx,PGA_STOP_MAXITER)
         call PGASetMaxGAIterValue(ctx, 2000)
         call PGASetMaxNoChangeValue(ctx,500)
         call PGASetPrintOptions(ctx,PGA_REPORT_STRING)
         call PGASetPrintOptions(ctx,PGA_REPORT_AVERAGE)
         call PGASetPrintFrequencyValue(ctx,5)

         call PGASetMutationRealValue(ctx,0.5d0)
         call PGASetMutationBoundedFlag(ctx,PGA_TRUE)

         call PGASetPopSize(ctx,100)
         call PGASetNumReplaceValue(ctx,25)

C     This call allows or disallows duplicate strings in the population.
C     NOTE:  If string length is short this may cause a failure as it is
C     impossible to generate enough combinations so that there aren't
C     any duplicates.

         call PGASetNoDuplicatesFlag(ctx, PGA_TRUE)

         call PGASetRestartFlag(ctx,PGA_TRUE)
         call PGASetRestartFrequencyValue(ctx,50)

C     If set to true, this call allows for mutation to occur on strings
C     which are produced through crossover.  If false, mutuation can
C     only occur on strings which did not experience crossover.

         call PGASetMutationAndCrossoverFlag(ctx,PGA_TRUE)

C     This section runs PGA Pack and finds the "best" values of the
C     parameters based on a metric defined by "findbest_nburn"

         call PGASetUp(ctx)

         call PGARun(ctx, findbest_nburn)

C     This final section converts the best values to real numbers,
C     prints them and ends the first half of the optimization.
```

```
            PGALIM.END = .TRUE.
            If (myid .eq. 0) then
               bestindex = PGAGetBestIndex(ctx,pop)
               best(1) = findbest_nburn(ctx,bestindex,pop)
               write(*,*) 'final eval = ',best(1)
            endif

            do i = 1, PGALIM.LEN+1
               call MPI_BCAST(PGALIM.BEST(i),1,MPI_DOUBLE_PRECISION,0,
                     MPI_COMM_WORLD,ierror)
            enddo
            call MPI_BCAST(PGALIM.OPFLAG,1,MPI_INTEGER,0,
                     MPI_COMM_WORLD,ierror)

            PGALIM.MAXTIME = PGALIM.BEST(PGALIM.LEN+1)

            call PGADestroy(ctx)

c     loop 2-----------------------------------------------------------
C     Perform the Delta V optimization ------------------------


            call setlim_nburn


            PGALIM.END = .FALSE.

            if (PGALIM.epochtime.eq.lasttime) then
               repeat = repeat + 1
            else
               repeat = 0
            endif

            write(*,*) 'repeat = ', repeat

            lasttime = PGALIM.epochtime

            call DefineRefOrb

            BURN_NUMBER      = 0
            DO I=1,PGALIM.MAXBURN
               BURN_DELTA_V(1,I) = 0.D0
               BURN_DELTA_V(2,I) = 0.D0
               BURN_DELTA_V(3,I) = 0.D0
               BURN_DELTA_V(4,I) = 0.D0
            ENDDO

            call initialize_sat(name)

            if (PGALIM.epochtime.gt.0.0d0) then
               pmern.date = pgalim.nextdate
               pmern.time = pgalim.nexttime
               do i = 1,6
                  pmern.els_kepler(i) = PGALIM.kepels(i)
               enddo
            endif

            call CalcFirstDevTime(BURN_NUMBER,BURN_DELTA_V,0)


            ctx = PGACreate(PGA_DATATYPE_REAL, PGALIM.LEN, PGALIM.FLAG)

C     This defines the stopping rule and sets the number of iterations
C     in which no change is allowed before stopping and other
```

```
C     PGAPack Parameters

         call PGASetRealInitRange(ctx,PGALIM.LOWLIM,PGALIM.UPLIM)
         call PGASetStoppingRuleType(ctx,PGA_STOP_NOCHANGE)
         call PGASetStoppingRuleType(ctx,PGA_STOP_MAXITER)
         call PGASetMaxGAIterValue(ctx, 3000)
         call PGASetMaxNoChangeValue(ctx,200)

         call PGASetPrintOptions(ctx,PGA_REPORT_STRING)
         call PGASetPrintOptions(ctx,PGA_REPORT_AVERAGE)
         call PGASetPrintFrequencyValue(ctx,5)

         call PGASetMutationRealValue(ctx,0.2d0)
         call PGASetMutationBoundedFlag(ctx,PGA_TRUE)

         call PGASetPopSize(ctx,100)
         call PGASetNumReplaceValue(ctx,25)


C     This call allows or disallows duplicate strings in the population.
C     NOTE:  If string length is short this may cause a failure as it is
C     impossible to generate enough combinations so that there aren't
C     any duplicates.

         call PGASetNoDuplicatesFlag(ctx, PGA_TRUE)
         call PGASetRestartFlag(ctx,PGA_TRUE)

C     If set to true, this call allows for mutation to occur on strings
C     which are produced through crossover.  If false, mutuation can
C     only occur on strings which did not experience crossover.

         call PGASetMutationAndCrossoverFlag(ctx,PGA_TRUE)

C     This section runs PGA Pack and finds the "best" values of the
C     parameters based on a metric defined by "findbest_nburn"

         call PGASetUp(ctx)

         call init_string(ctx,1,PGA_OLDPOP)

         if (repeat.eq.1) then
            PGALIM.TARGETWEIGHT(7) = 15.0d0
c     d          PGALIM.TARGETWEIGHT(7) = 10.0d0
         elseif (repeat.ge.1) then
            PGALIM.TARGETWEIGHT(7) = PGALIM.TARGETWEIGHT(7)
                                     0.25d0*PGALIM.TARGETWEIGHT(7)
c     d          if (repeat.ge.3) then
c     d          if (repeat.ge.2) then
c     call init_string(ctx,1,PGA_OLDPOP)
c     d          endif
         endif

         call PGARun(ctx, findbest_nburn)

C     This final section converts the best values to real numbers,
C     prints them and ends the program

         PGALIM.END = .TRUE.
         If (myid .eq. 0) then
            bestindex = PGAGetBestIndex(ctx,pop)
            best(1) = findbest_nburn(ctx,bestindex,pop)
            write(*,*) 'final eval = ',best(1)
         endif
```

```
         do i = 1, 6
            call MPI_BCAST(PGALIM.kepels(i),1,MPI_DOUBLE_PRECISION,0,
                  PI_COMM_WORLD,ierror)
         enddo
         call MPI_BCAST(PGALIM.epochtime,1,MPI_DOUBLE_PRECISION,0,
                  MPI_COMM_WORLD,ierror)
         call MPI_BCAST(PGALIM.nextdate,1,MPI_DOUBLE_PRECISION,0,
                  MPI_COMM_WORLD,ierror)
         call MPI_BCAST(PGALIM.nexttime,1,MPI_DOUBLE_PRECISION,0,
                  MPI_COMM_WORLD,ierror)

         call PGADestroy(ctx)


      enddo

      if (myid.eq.0) then
         write(*,*)
         write(*,*) 'SUMMARY STATS ----------------------------------`
         write(*,*) 'Days forward = ',PGALIM.epochtime/86400.0d0
         write(*,*) 'Total number of burns =',PGALIM.totalburns
         write(*,*) 'Total Delta V = ',PGALIM.totaldeltav
         write(*,*)
      endif

      call MPI_Finalize(ierror)
      write(*,*) fdate()

      stop
      end
```

D-1-2 Operational Approach Modifications to FUNC

```
real*8  function func(p)
C-----------------------------------------------------------------------
C
C     Function Func
C
C     This file contains the objective function for the operational
C     station keeping approach.
C
C     Author: James E. Smith, 2Lt, USAF
C     MIT/ Aero-Astro Dept./ Draper Fellow
C
C     Ronald J. Proulx
C     Draper Laboratory
C
C-----------------------------------------------------------------------

      implicit none

C--------Include necessary modules -------------------

      include 'pmern.h'
      include 'frc.h'
      include 'pgalim.h'

C-----Define variables for call to Satellite ------------------

      integer*4      satellite
      external       satellite
```

```
      integer*4       max_list_length
      parameter      (max_list_length  = 52)

      integer*4       status
      integer*4       burn_number
      Integer*4       iatmos_preburn / 1 /
      integer*4       iatmos_postburn / 1 /

      real*8          time
      real*8          burn_delta_v(4,max_list_length)
      REAL*8          RHO_ONE_HIGH      / 0.0 D0 /
      REAL*8          RHO_ONE_LOW       / 0.0 D0 /
      real*8          epoch_ymd
      real*8          epoch_hms
      real*8          posvel(6)
      real*8          elements (17)
      real*8          btime(max_list_length)

      character*(6)   name  /'pmern1'/
      character*(6)   refname /'pmernr'/
      CHARACTER*(72)  MESSAGE
      CHARACTER*(12)  FILENAME

      real*8 dayjul0,secjul0,year,second,month,day,hour,minute
      real*8 dayjul,secjul,nexttime,nextdate,checktime


C--------Define other variables ----------------------

      integer*4       i
      integer*4       index
      integer*4       j
      integer*4       k
      integer*4       onoff(maxvar)
      integer*4       usedburn

      real*8          p(*)
      real*8          mag
      real*8          sumdelta
      real*8          burn(maxvar)
      real*8          magburn(maxvar)
      real*8          deltavtot
      real*8          deltavsum(maxvar)
      real*8          burntime(maxvar)
      real*8          bcconst(maxvar)
      real*8          bcconsttot
      real*8          vconst
      real*8          Mcheck
      real*8          pi
      real*8          comp
      real*8          two_n
      real*8          bitzero(3)
      real*8          deltat
      real*8          maxbcconst(maxvar)
      real*8          totbcconst(maxvar)
      real*8          timeweight
      real*8          timeconst
      real*8          endconsttot
      real*8          endconst(maxvar)
      real*8          endtime
      real*8          firstdev
      real*8          temp
      real*8          order(maxvar)
      real*8          same(maxvar)
```

```
      integer*4        skip

      external       mag

C     ********** BEGIN PROGRAM ********************************

      pgalim.eflag = .false.
      skip = 0

c     Initialize everything to 0.0d0 -------------------------


      deltavtot = 0.0d0
      vconst = 0.0d0
      bcconsttot = 0.0d0
      endconsttot = 0.0d0
      do i = 1, maxvar
         bcconst(i) = 0.0d0
         deltavsum(i) = 0.0d0
         maxbcconst(i) = 0.0d0
         totbcconst(i) = 0.0d0
         endconst(i) = 0.0d0
         order(i) = 0.0d0
         onoff(i) = 0.0d0
         same(i) = 0.0d0
      enddo

      TIME             = 0.D0
      BURN_NUMBER      = 0
      DO I=1,PGALIM.MAXBURN
         BURN_DELTA_V(1,I) = 0.D0
         BURN_DELTA_V(2,I) = 0.D0
         BURN_DELTA_V(3,I) = 0.D0
         BURN_DELTA_V(4,I) = 0.D0
      ENDDO


C     See if burns are on or off and find total number of burns --------

      if (PGALIM.OPFLAG.eq.1) then
         j = 0
      else
         j = 0
      endif

      do i = 1,PGALIM.MINBURN
         onoff(i) = 1
      enddo

      k = PGALIM.MINBURN+1
      do i = PGALIM.NUMVAR+1,PGALIM.LEN-j
         if ((p(i).gt.0.1d0).and.(p(i).le.0.2d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.0.3d0).and.(p(i).le.0.4d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.0.5d0).and.(p(i).le.0.6d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.0.7d0).and.(p(i).le.0.8d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.0.9d0).and.(p(i).le.1.0d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.1.1d0).and.(p(i).le.1.2d0)) then
```

```
            onoff(k) = 1
         elseif ((p(i).gt.1.3d0).and.(p(i).le.1.4d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.1.5d0).and.(p(i).le.1.6d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.1.7d0).and.(p(i).le.1.8d0)) then
            onoff(k) = 1
         elseif ((p(i).gt.1.9d0).and.(p(i).le.2.0d0)) then
            onoff(k) = 1
         else
            onoff(k) = 0
         endif
         k = k + 1
      enddo

      BURN_NUMBER = 0
      do i = 1, PGALIM.MAXBURN
         BURN_NUMBER = onoff(i) + BURN_NUMBER
      enddo

C     Get burn components from the string

      usedburn = 1
      do i = 1, PGALIM.MAXBURN
         if (onoff(i).eq.1) then
            btime(usedburn) = p(i)
            usedburn = usedburn+1
         endif
      enddo


      do i = 1, BURN_NUMBER
         order(i) = 1
         do j = 1, BURN_NUMBER
            if (btime(i).gt.btime(j)) then
               order(i) = order(i)+1
            endif
         enddo
      enddo

      k =1
      do i = 1,BURN_NUMBER
         do j = 1,BURN_NUMBER
            if (btime(i).eq.btime(j).and.(i.ne.j)) then
               same(k) = (i)
            endif
         enddo
         if (same(k).ne.0) then
            k = k + 1
         endif
      enddo

      k = k-1
      do i=1,k
         order(same(i))=order(same(i))+(i-1)
      enddo

      usedburn = 1
      DO I=1,PGALIM.MAXBURN
         if (onoff(i).eq.1) then
            BURN_DELTA_V(1,order(usedburn)) = p(i)
            BURN_DELTA_V(2,order(usedburn)) = p(i+PGALIM.MAXBURN*1)
            BURN_DELTA_V(3,order(usedburn)) = p(i+PGALIM.MAXBURN*2)
            BURN_DELTA_V(4,order(usedburn)) = p(i+PGALIM.MAXBURN*3)
```

```
            usedburn = usedburn + 1
         endif
      ENDDO


c     If this is the time optimization,
C     calculate the time of first devation--------------------------

      if (PGALIM.OPFLAG.eq.1) then
         call calcfirstdevtime(BURN_NUMBER,BURN_DELTA_V,1)
      else

C     Otherwise call satellite to perform deltav optimization --------
c     CALL SATELLITE to obtain state at request time

         call calcfirstdevtime(BURN_NUMBER,BURN_DELTA_V,1)

         endtime = PGALIM.MAXTIME
         time = 0.0d0
         index = 1

         do while (time.le.endtime)

            STATUS = SATELLITE ( name ,           TIME,
     2           BURN_DELTA_V,    BURN_NUMBER,
     3           IATMOS_PREBURN,  IATMOS_POSTBURN,
     4           RHO_ONE_HIGH,    RHO_ONE_LOW,
     5           EPOCH_YMD,       EPOCH_HMS,
     6           POSVEL,          ELEMENTS,
     7           MESSAGE,         FILENAME )

            if(status.ne.0) WRITE(*,*) 'STATUS = ', STATUS

c     This is for elements 1 through 2
            do i = 1,2
               PGALIM.target(i) = PGALIM.reforb(i,index)
               If (DABS(PGALIM.target(i) - elements(i)).gt.
                                    PGALIM.tol(i)) then
                  bcconst(i)=DABS((PGALIM.target(i)-elements(i)))-
                             PGALIM.tol(i)
               else
                  bcconst(i) = 0.0d0
               endif
               totbcconst(i) = bcconst(i) + totbcconst(i)
               if (bcconst(i) .gt. maxbcconst(i)) then
                  maxbcconst(i) = bcconst(i)
               endif

            enddo

c     This is for elements 3 through 6

            pi = acos(-1.0d0)
            do i = 3,6
               PGALIM.target(i) = PGALIM.reforb(i,index)
               Mcheck = dacos(dcos((PGALIM.target(i)-
                              elements(i))*pi/180.0d0))*(180.0d0/pi)
               IF (Mcheck.gt.PGALIM.tol(i)) then
                  bcconst(i) = DABS((Mcheck-PGALIM.tol(i)))
               else
                  bcconst(i) = 0.0d0
               endif
               totbcconst(i) = bcconst(i) + totbcconst(i)
```

```
               if (bcconst(i) .gt. maxbcconst(i)) then
                  maxbcconst(i) = bcconst(i)
               endif
            enddo

            if (time.ge.(PGALIM.FIRSTDEVZEROB+PGALIM.BURNWINDOW)) then
               deltat = 86400.0d0/(PGALIM.TIMESPERDAY)
               index = index + (24)
            else
               deltat = 86400.0d0/(PGALIM.TIMESPERDAY*24.0d0)
               index = index + 1
            endif
            time = time + deltat

         enddo


         do i = 1,6
            totbcconst(i) = totbcconst(i)*PGALIM.targetweight(i)
                            *PGALIM.targeton(i)
            bcconsttot = totbcconst(i)+bcconsttot
         enddo

c     Scale Total Deviation to favor more burns

         bcconsttot = bcconsttot/sqrt(dble(BURN_NUMBER))
         bcconsttot = bcconsttot/(PGALIM.MAXTIME/86400.0d0)


      endif

C     Find the total Delta V

      do j = 1, BURN_NUMBER
         do i = 1, 3
            BURN(i) = BURN_DELTA_V(i+1,j)
         enddo
         MAGBURN(J) = MAG(BURN)
      enddo

      do i = 1, BURN_NUMBER
         deltavtot = MAGBURN(i) + deltavtot
      enddo

C     Find the sum of the Delta_V used for func
C     Use this formulation if outside the box

      if (bcconsttot .gt. 1.0d0) then

         usedburn = 1
         do i = 1, PGALIM.MAXBURN
            if (onoff(i) .eq.1) then
               do j = 2,4
                  comp = BURN_DELTA_V(j,usedburn)**2.0d0
                  deltavsum(i) = comp + deltavsum(i)
               enddo
               usedburn = 1 + usedburn
            else
               do j = 1,3
                  comp = (p(i+j*PGALIM.MAXBURN)-0.0d0)**2.0d0
                  deltavsum(i) = deltavsum(i) + comp
               enddo
            endif
            vconst = dsqrt(deltavsum(i))+vconst
```

```
         enddo

C        Use this formulation if inside the box

         else

            usedburn = 1
            do i = 1, PGALIM.MAXBURN
               do j = 1,3
                  comp = (p(i+j*PGALIM.MAXBURN))**2.0d0
                  deltavsum(i) = comp+deltavsum(i)
               enddo
               vconst = dsqrt(deltavsum(i))+vconst
            enddo

         endif

         vconst   = vconst * PGALIM.targetweight(7)

         if (PGALIM.firstdev.ne.0.0d0) then
            timeconst = (PGALIM.MAXTIME-PGALIM.firstdev)/86400.0d0
         else
            timeconst = 0.0d0
         endif

C        FINALLY, calculate func associated with given string

         if (pgalim.eflag .eq. .false.) then
            if (pgalim.opflag.eq.1) then
               func = PGALIM.firstdev/86400.0d0
            else if (bcconsttot.lt.3000.0d0) then
               func = bcconsttot*(4.0d-1)+vconst*(4.0d-1)
            else
               func = bcconsttot*0.000001d0 + vconst*0.5d0+timeconst
            endif

         else
            func = PGALIM.MINERROR
         endif

C        Write the output the last time through func ---------------

         if ((pgalim.end .eq. .true.) .or. (pgalim.plot .eq. .true.)) then

            if(PGALIM.OPFLAG.eq.1) then

               do i = 1,PGALIM.LEN
                  PGALIM.BEST(i) = p(i)
               enddo
               PGALIM.BEST(PGALIM.LEN+1) = PGALIM.firstdev-
                           (86400.0d0/PGALIM.TIMESPERDAY)
               PGALIM.OPFLAG = 2


               write(*,*) 'func=',func
               write(*,*) 'Endtime = ',PGALIM.firstdev,'seconds'

            else

               write(*,*) 'func=',func
               do i = 1,6
                  if (PGALIM.targeton(i) .eq. 1) then
                     write(*,100) i,PGALIM.target(i),elements(i),
                           PGALIM.target(i)-elements(i)
```

```
                  else
                     write(*,101) i,PGALIM.target(i),elements(i)
                  endif
               enddo

               do i = 1,6
                  write(*,*) 'Max Deviation of element',i,maxbcconst(i)
               enddo
               do i = 1,6
                  write(*,*) 'Total Deviation of element', i, totbcconst(i)
               enddo

               write(*,*) 'Total DeltaV=',deltavtot
               write(*,*) 'Number of Burns=',BURN_NUMBER
               write(*,*) 'End Time=',endtime/86400.0d0


               write(*,*) 'BURN TABLE------------------------------'
               do i = 1, BURN_NUMBER
                  write(*,*) i,' Time=',BURN_DELTA_V(1,i),
                                 'Magnitude =',MAGBURN(i)
               enddo

C     ------ Update for next time through loop

               if (bcconsttot.gt.5.0d0) then
                  write(*,*) '*****************************************'
                  write(*,*) '*         ERROR--SOLUTION NOT IN BOX       *'
                  write(*,*) '*           REOPTIMIZING LAST LOOP         *'
                  write(*,*) '*****************************************'
               else
                  checktime = endtime - 86400.0d0*2.0d0

                  time = checktime

                  STATUS = SATELLITE ( name, TIME,
     2                 BURN_DELTA_V,   BURN_NUMBER,
     3                 IATMOS_PREBURN, IATMOS_POSTBURN,
     4                 RHO_ONE_HIGH,   RHO_ONE_LOW,
     5                 EPOCH_YMD,      EPOCH_HMS,
     6                 POSVEL,         ELEMENTS,
     7                 MESSAGE,        FILENAME )

                  do i = 1,6
                     write(*,*) 'Next Starting Element',i,elements(i)
                  enddo

                  call julpak(dayjul0,secjul0,pmern.date,pmern.time)
                  call calndr (year,month,day,hour,minute,second,
     1                 dayjul0,secjul0 + checktime)
                  call julian (dayjul,secjul,year,month,day,
                           hour,minute,second)
                  call calpak (nextdate,nexttime,dayjul,secjul)

                  write(*,*) 'Next Start Date',nextdate
                  write(*,*) 'Next Start Time',nexttime

                  PGALIM.totalburns = PGALIM.totalburns + BURN_NUMBER
                  PGALIM.totaldeltav = PGALIM.totaldeltav + deltavtot


                  do i = 1,6
                     PGALIM.kepels(i) = elements(i)
                  enddo
```

```
            do i = 1,BURN_NUMBER
               write(91,*) BURN_DELTA_V(1,i)+ PGALIM.epochtime
               write(91,*) BURN_DELTA_V(2,i)
               write(91,*) BURN_DELTA_V(3,i)
               write(91,*) BURN_DELTA_V(4,i)
            enddo

            PGALIM.epochtime = checktime + PGALIM.epochtime
            PGALIM.nextdate = nextdate
            PGALIM.nexttime = nexttime

         endif



      endif

      write(*,*) 'BURN COMPONENTS ------------------------'
      write(*,*) '          TIME    TANGENTIAL
                NORMAL       RADIAL     BURN NUMBER'

      usedburn = 1
      do i = 1, PGALIM.MAXBURN
         if (onoff(i).eq.1) then
            write(*,50) p(i),p(i+PGALIM.MAXBURN*1),
         p(i+2*pgalim.maxburn),p(i+3*pgalim.maxburn),order(usedburn)
            usedburn = usedburn + 1
         else
            write(*,51) p(i),p(i+PGALIM.MAXBURN*1)
                ,p(i+2*pgalim.maxburn),p(i+3*pgalim.maxburn)
         endif
      enddo


   endif


50    FORMAT(F14.4,F12.6,F12.6,F12.6,F10.0)
51    FORMAT(F14.4,F12.6,F12.6,F12.6)
100   Format(I4,F16.8,F16.8,F16.8)
101   Format(I4,F16.8,F16.8)
150   Format(I4,F16.8,F16.8)

      return
      end
```

D-2 Feasibility Test of the Operational Approach Deviation Plots

In order to verify that the operational approach was indeed a viable method for producing station-keeping strategies, the operational approach was applied to the global approach case two epoch elements. As described in section 5-3-2-1-3 these epoch elements were created such that a violation of all six orbital elements is guaranteed in the first few days of operation. Due to all six orbital elements violating, this choice of elements provides a robust test of the station-keeping algorithm and was therefore used to determine the feasibility of employing such an approach. This section contains the element history plots of each of the six orbital elements resulting from the application of the operational station keeping approach to the case two epoch elements.

Figure D-1 Operational Feasibility Test SMA Deviation (Limit = 1 km)

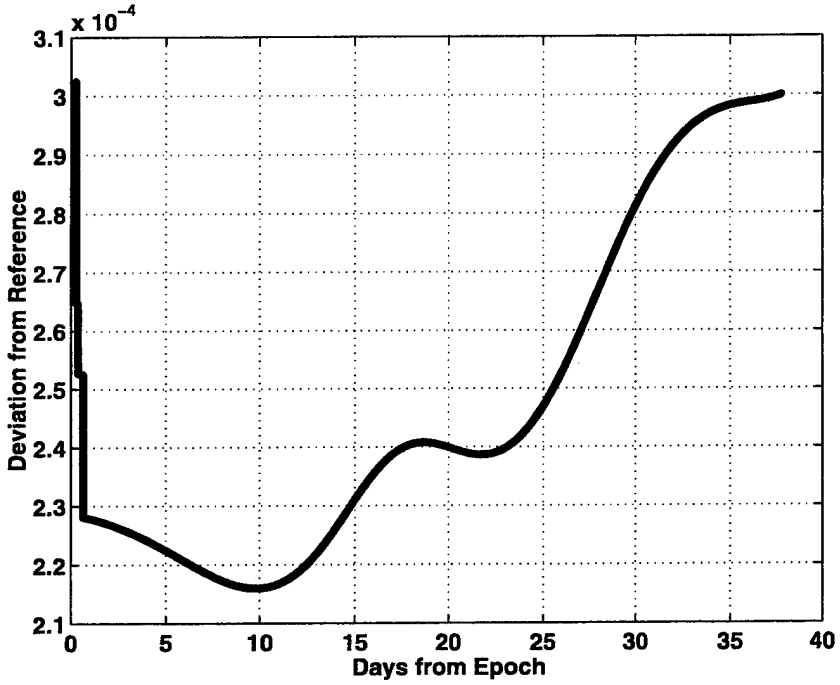


Figure D-2 Operational Feasibility Test Eccentricity Deviation (Limit = 0.0003)

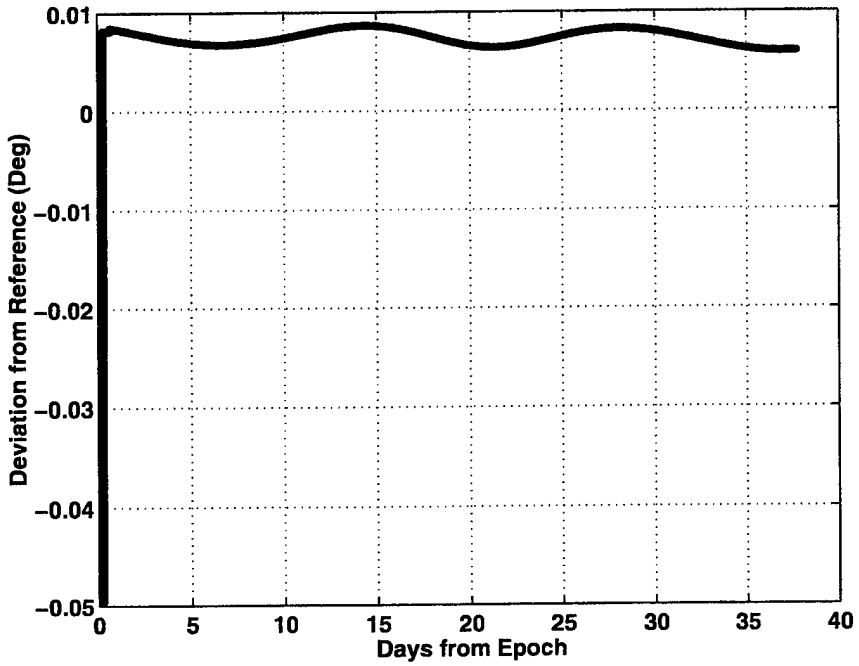


Figure D-3 Operational Feasibility Test Inclination Deviation (Limit = 0.05°)

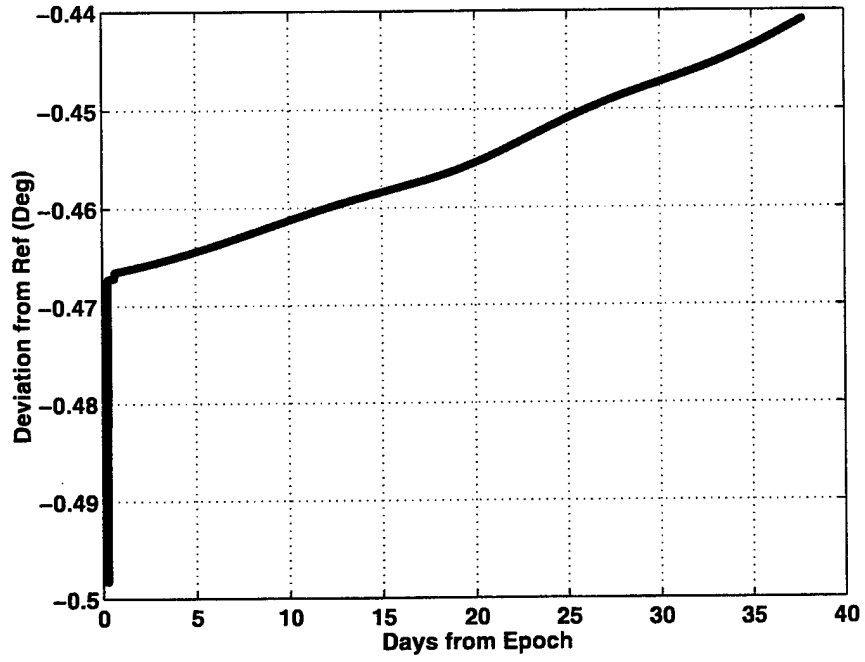


Figure D-4 Operational Feasibility Test RAAN Deviation (Limit = 0.5°)

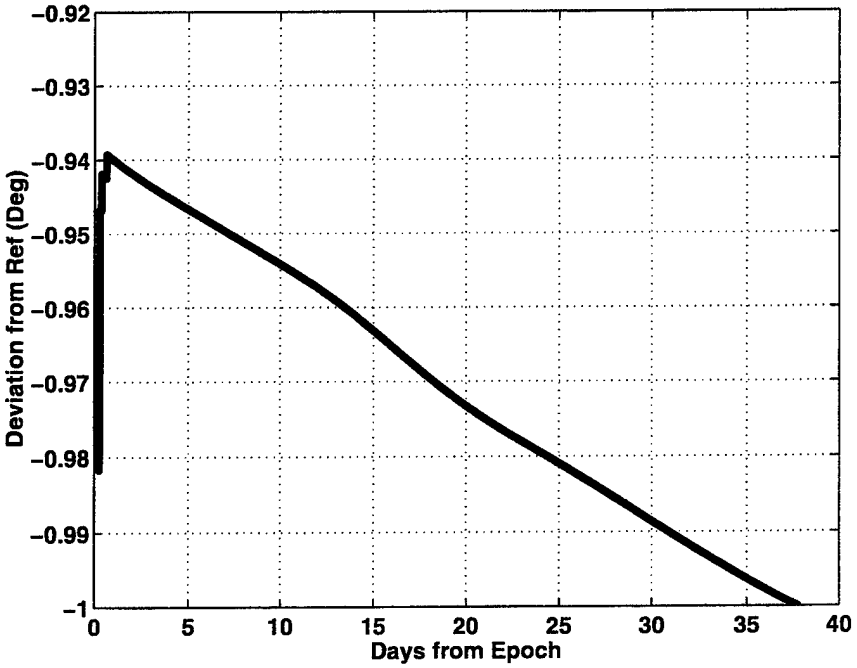


Figure D-5 Operational Feasibility Test Arg. of Perigee Deviation (Limit = 1°)

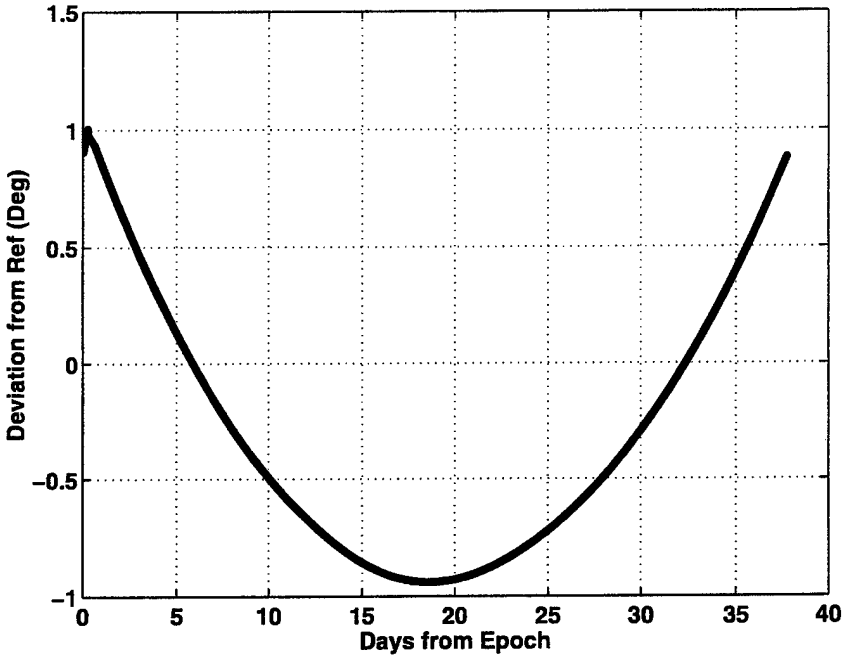


Figure D-6 Operational Feasibility Test Mean Anomaly Deviation (Limit = 1°)

D-3 Input Decks for Operational Approach as Planning Tool

This section contains the input decks utilized to define both the reference and actual orbits for testing the application of the operational approach as a planning tool. The first input deck was used to create the reference orbit in DSST and the second was used to create the actual orbit. The elements contained in both input decks are a result of a five-year 113:14 optimization as explained in section 5-4-5-1-1. Proper flag and keyword values were determined from personal communications with Dr. Ronald Proulx, The Charles Stark Draper Laboratory.

D-3-1 Reference Orbit

```
C
C
C      PMEF FILE FOR PLANNING TEST REFERENCE ORBIT DEFINITION
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.2000032100000000D+08 PME_DATE          1
   0.0148000000000000D+06 PME_TIME          2
   0.1049687570948029D+05 ELS_KEP(1)        3
   0.3330325737341785D+00 ELS_KEP(2)        4
   0.1165614506209416D+03 ELS_KEP(3)        5
   0.0000000000000000D+03 ELS_KEP(4)        6
   0.2600000000000000D+03 ELS_KEP(5)        7
   0.0000000000000000D+03 ELS_KEP(6)        8
   0.0000000000000000D+00 ELS_EQUIN(1)      9
   0.0000000000000000D+00 ELS_EQUIN(2)     10
   0.0000000000000000D+00 ELS_EQUIN(3)     11
   0.0000000000000000+00 ELS_EQUIN(4)      12
   0.0000000000000000D+00 ELS_EQUIN(5)     13
   0.0000000000000000D+00 ELS_EQUIN(6)     14
   0.0000000000000000D+00 POSVEL(1)        15
   0.0000000000000000D+00 POSVEL(2)        16
   0.0000000000000000D+00 POSVEL(3)        17
   0.0000000000000000D+00 POSVEL(4)        18
   0.0000000000000000D+00 POSVEL(5)        19
   0.0000000000000000D+00 POSVEL(6)        20
   0.2000000000000000D+01 PME_CD           21
   0.0000000000000000D+00 PME_RHO_ONE      22
   0.5000000000000000D-04 SMA_SIGMA        23
   0.5000000000000000D-04 INC_SIGMA        24
   0.5000000000000000D-04 ASC_SIGMA        25
   0.1250000000000000D+04 PME_SCMASS       26
   0.4330000000000000D-04 PME_SCAREA       27
   0.8640000000000000D+05 PME_STEPSIZE     28
   0.1400000000000000D+02 DP_SPARE1        29
   0.1130000000000000D+03 DP_SPARE2        30
   0.0000000000000000D+00 DP_SPARE3        31
   0.0000000000000000D+00 DP_SPARE4        32
   0.0000000000000000D+00 DP_SPARE5        33
```

```
0.0000000000000000D+00 DP_SPARE6       34
             1         PME_RETRO       35
            12         PME_KEP_SYS     36
            12         POS_VEL_SYS     37
             1         GEN_METHOD      38
             1         ATMOS_MODEL     39
        840401         JACRB_DATE      40
           123         JACRB_SSS       41
        840401         SLP1950_DATE    42
           456         SLP1950_SSS     43
        840401         SLPTOD_DATE     44
           789         SLPTOD_SSS      45
        840401         TIMECF_DATE     46
           123         TIMECF_SSS      47
             2         HARRIS_MODEL    48
            10         POTNTL_MODEL    49
            50         PME_NMAX        50
             0         PME_MMAX        51
             1         PME_IZONAL      52
             1         PME_IJ2J2       53
             0         PME_NMAXRS      54
             0         PME_MMAXRS      55
             3         PME_ITHIRD      56
             2         PME_INDDRG      57  2 = DRAG OFF
             2         PME_ISZAK       58
             2         PME_INDSOL      59  2 = SOLRAD OFF
             2         PME_JSHPER      60
             1         PME_JZONAL      61
             1         PME_JMDALY      62
             2         PME_INP_TYPE    63
            12         PME_EQUI_SYS    64
            11         INTEG_FRAME     65
            19         OUTPUT_FRAME    66
             0         PME_NSTATE      67
             1         PME_SPSHPER     68
             2         PME_KSPCF       69
             4         PME_INDSET      70
             0         INT_SPARE1      71
             0         INT_SPARE2      72
             0         INT_SPARE3      73
             0         INT_SPARE4      74
             0         INT_SPARE5      75
             0         INT_SPARE6      76
             0         INT_SPARE7      77
             0         INT_SPARE8      78
             0         INT_SPARE9      79
             0         INT_SPARE10     80
```

D-3-2 Actual Orbit

```
C
C      PMEF FILE FOR ACTUAL ORBIT FOR PLANNING TEST CASES
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
   0.2000032100000000D+08 PME_DATE         1
   0.0148000000000000D+06 PME_TIME         2
   0.1049687570948029D+05 ELS_KEP(1)       3
   0.3330325737341785D+00 ELS_KEP(2)       4
   0.1165614506209416D+03 ELS_KEP(3)       5
   0.0000000000000000D+03 ELS_KEP(4)       6
```

```
0.2600000000000000D+03 ELS_KEP(5)       7
0.0000000000000000D+03 ELS_KEP(6)       8
0.0000000000000000D+00 ELS_EQUIN(1)     9
0.0000000000000000D+00 ELS_EQUIN(2)    10
0.0000000000000000D+00 ELS_EQUIN(3)    11
0.0000000000000000+00 ELS_EQUIN(4)     12
0.0000000000000000D+00 ELS_EQUIN(5)    13
0.0000000000000000D+00 ELS_EQUIN(6)    14
0.0000000000000000D+00 POSVEL(1)       15
0.0000000000000000D+00 POSVEL(2)       16
0.0000000000000000D+00 POSVEL(3)       17
0.0000000000000000D+00 POSVEL(4)       18
0.0000000000000000D+00 POSVEL(5)       19
0.0000000000000000D+00 POSVEL(6)       20
0.2000000000000000D+01 PME_CD          21
0.0000000000000000D+00 PME_RHO_ONE     22
0.5000000000000000D-04 SMA_SIGMA       23
0.5000000000000000D-04 INC_SIGMA       24
0.5000000000000000D-04 ASC_SIGMA       25
0.1250000000000000D+04 PME_SCMASS      26
0.4330000000000000D-04 PME_SCAREA      27
0.8640000000000000D+05 PME_STEPSIZE    28
0.1400000000000000D+02 DP_SPARE1       29
0.1130000000000000D+03 DP_SPARE2       30
0.0000000000000000D+00 DP_SPARE3       31
0.0000000000000000D+00 DP_SPARE4       32
0.0000000000000000D+00 DP_SPARE5       33
0.0000000000000000D+00 DP_SPARE6       34
         1             PME_RETRO       35
        12             PME_KEP_SYS     36
        12             POS_VEL_SYS     37
         1             GEN_METHOD      38
         1             ATMOS_MODEL     39
    840401             JACRB_DATE      40
       123             JACRB_SSS       41
    840401             SLP1950_DATE    42
       456             SLP1950_SSS     43
    840401             SLPTOD_DATE     44
       789             SLPTOD_SSS      45
    840401             TIMECF_DATE     46
       123             TIMECF_SSS      47
         2             HARRIS_MODEL    48
        10             POTNTL_MODEL    49
        21             PME_NMAX        50
        21             PME_MMAX        51
         1             PME_IZONAL      52
         1             PME_IJ2J2       53
        21             PME_NMAXRS      54
        21             PME_MMAXRS      55
         1             PME_ITHIRD      56
         1             PME_INDDRG      57  2 = DRAG OFF
         1             PME_ISZAK       58
         1             PME_INDSOL      59  2 = SOLRAD OFF
         2             PME_JSHPER      60
         1             PME_JZONAL      61
         1             PME_JMDALY      62
         2             PME_INP_TYPE    63
        12             PME_EQUI_SYS    64
        11             INTEG_FRAME     65
        19             OUTPUT_FRAME    66
         0             PME_NSTATE      67
         1             PME_SPSHPER     68
         2             PME_KSPCF       69
         1             PME_INDSET      70
```

```
0               INT_SPARE1      71
0               INT_SPARE2      72
0               INT_SPARE3      73
0               INT_SPARE4      74
0               INT_SPARE5      75
0               INT_SPARE6      76
0               INT_SPARE7      77
0               INT_SPARE8      78
0               INT_SPARE9      79
0               INT_SPARE10     80
```

D-4 Results of Operational Approach as Planning Tool

As discussed in section 5-4-5, by stringing together a number of applications of the operational approach, estimates of the amount of fuel required for an extended period of time can be obtained. This section contains the results from a number of these planning applications. The uncontrolled five-year element deviation histories are first presented for later comparison with the controlled elements. The results of three separate cases are then presented: a 1 year test case, a 2.5 year ΔV estimation, and 1200 day case used to show the greediness of the strategy.

D-4-1 Uncontrolled Deviation with Five-year Optimized Epoch Elements

Due to the change in length of optimization times (from 90 days to one year or more), it was necessary to redefine the optimal epoch elements for a five year case with a March 21, 2000 epoch (see section 5-4-5-1-1). The definition of new elements also changed the behavior of the uncontrolled orbit. For comparison to controlled results which are presented in subsequent sections, the five-year uncontrolled deviations from the reference orbit are presented here.

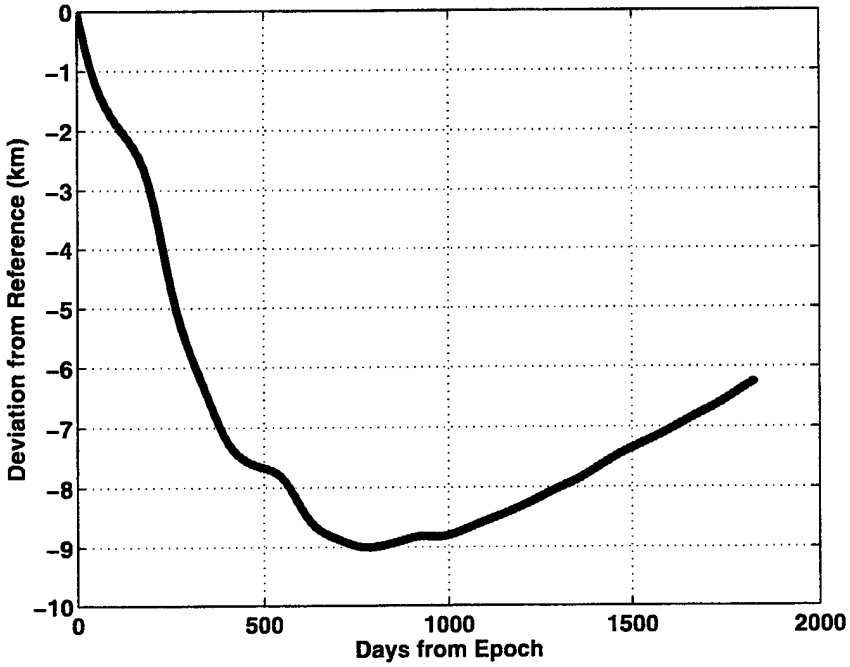


Figure D-7 Operational Planning Case Uncontrolled SMA Deviation (Limit = 1 km)

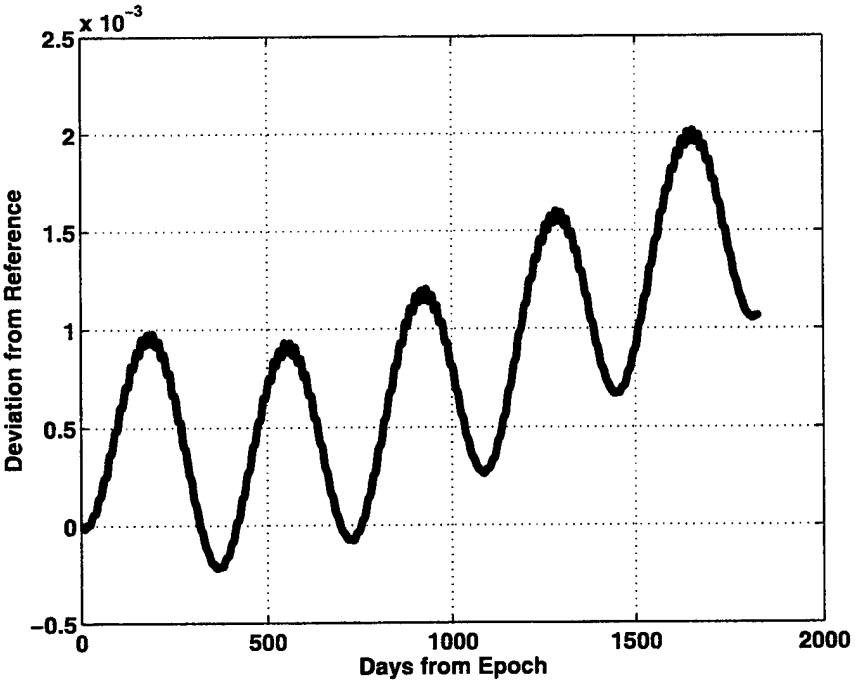


Figure D-8 Operational Planning Case Uncontrolled e Deviation (Limit = 0.0003)

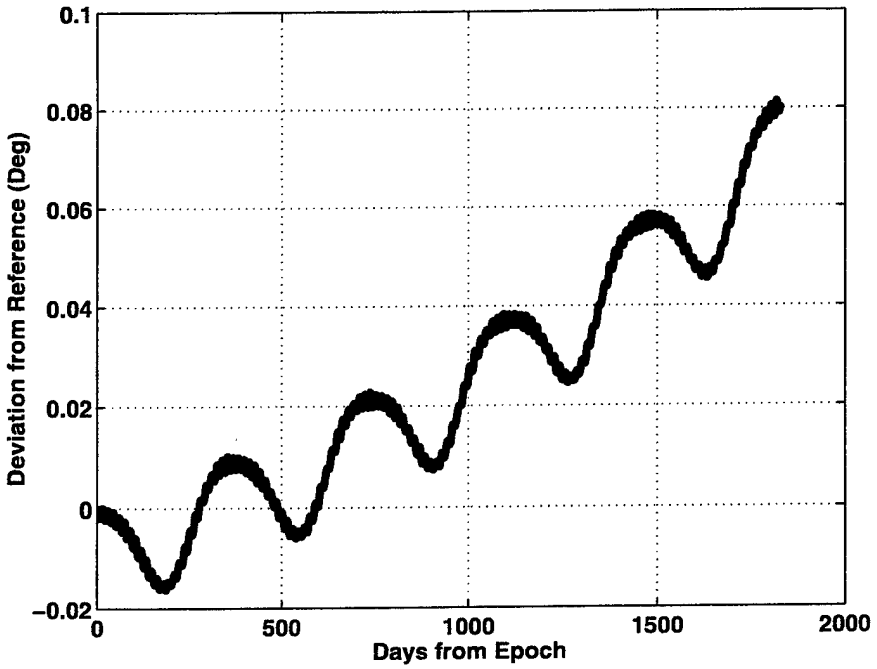


Figure D-9 Operational Planning Case Uncontrolled i Deviation (Limit = 0.05°)

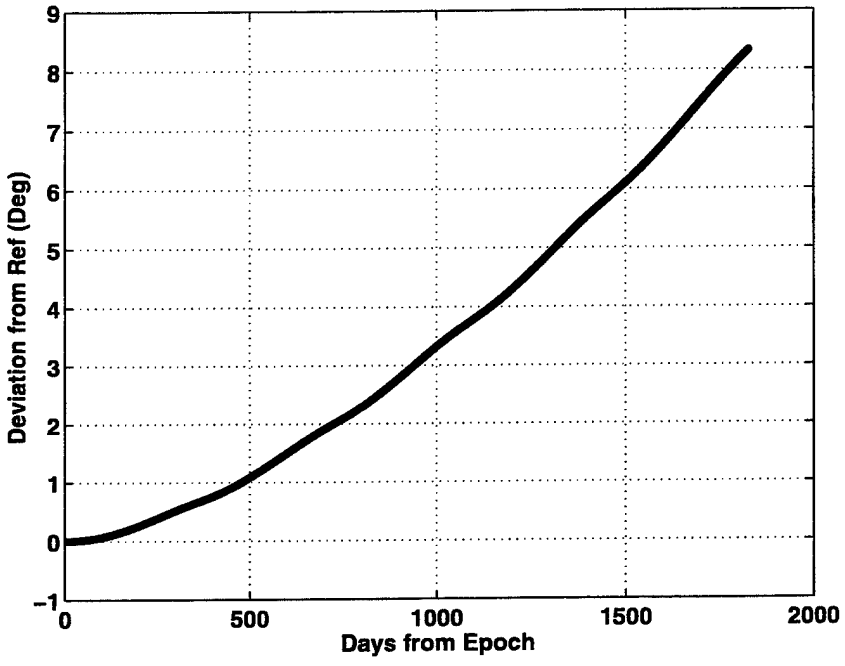


Figure D-10 Operational Planning Case Uncontrolled Ω Deviation (Limit = 0.5°)

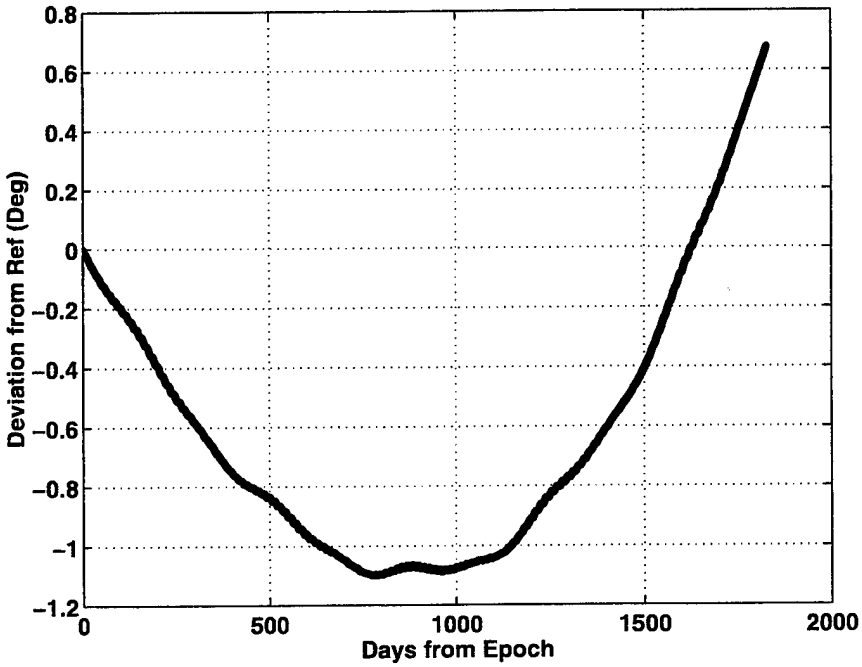


Figure D-11 Operational Planning Case Uncontrolled ω Deviation (Limit = 1°)

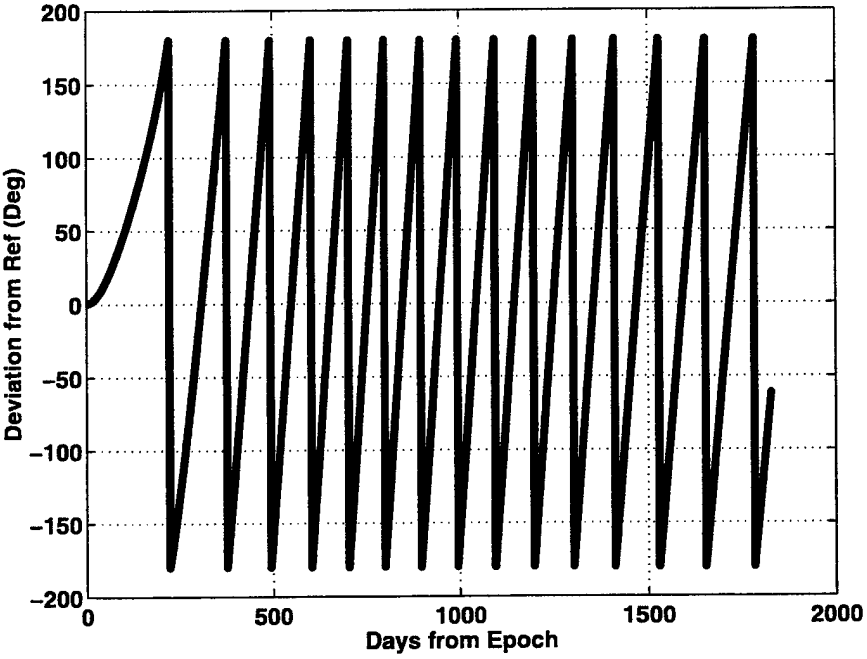


Figure D- 12 Operational Planning Case Uncontrolled M Deviation (Limit = 1°)

D-4-2 Results of One Year Test of Operational Planning Tool

This section contains the results of the repeated application of the operational approach to the Borealis™ node at noon orbit for one year. This case was used to verify the use of the operational approach as a planning tool.

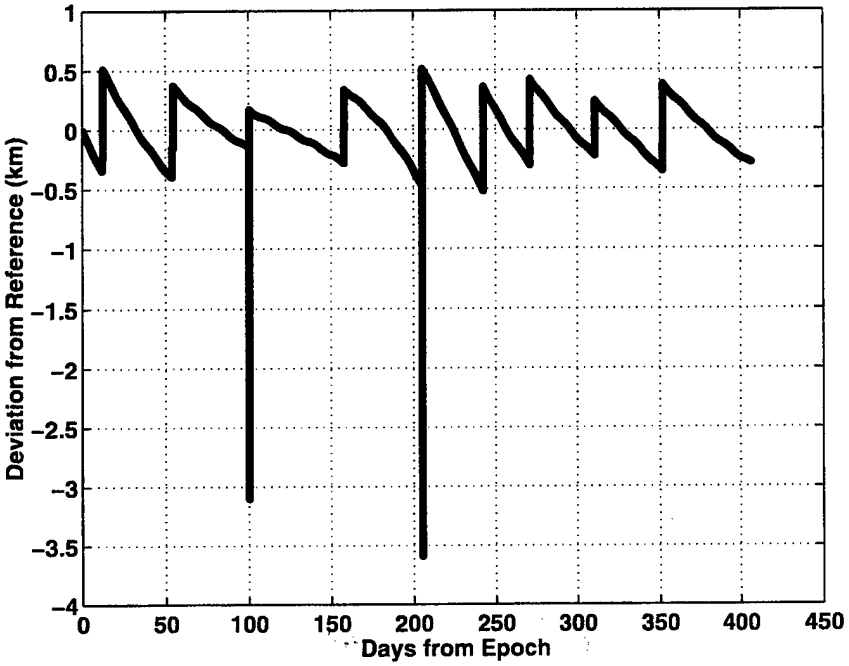


Figure D-13 1-Year Operational Planning Approach SMA Deviation (Limit = 1 km)

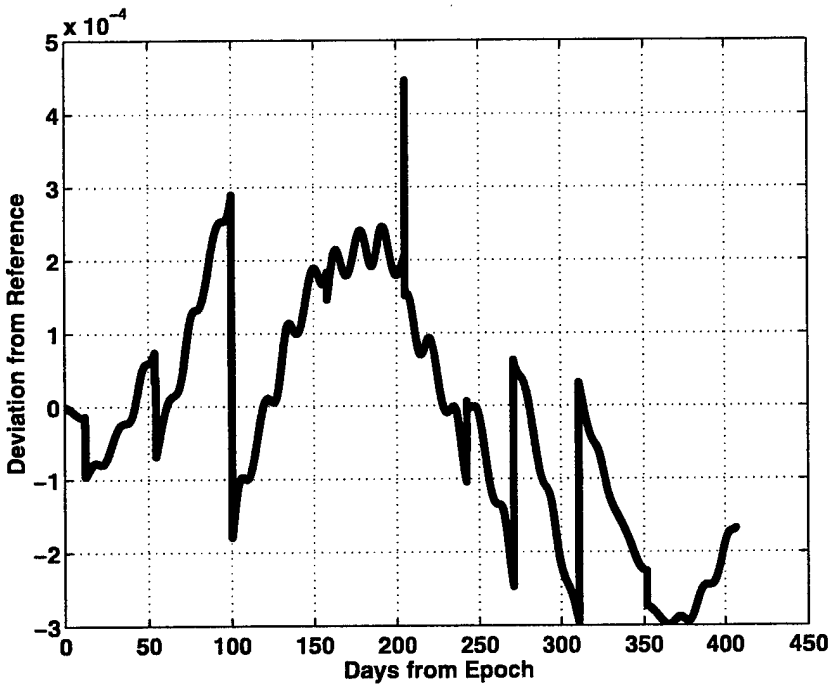


Figure D-14 1-Year Operational Planning Approach e Deviation (Limit = 0.0003)

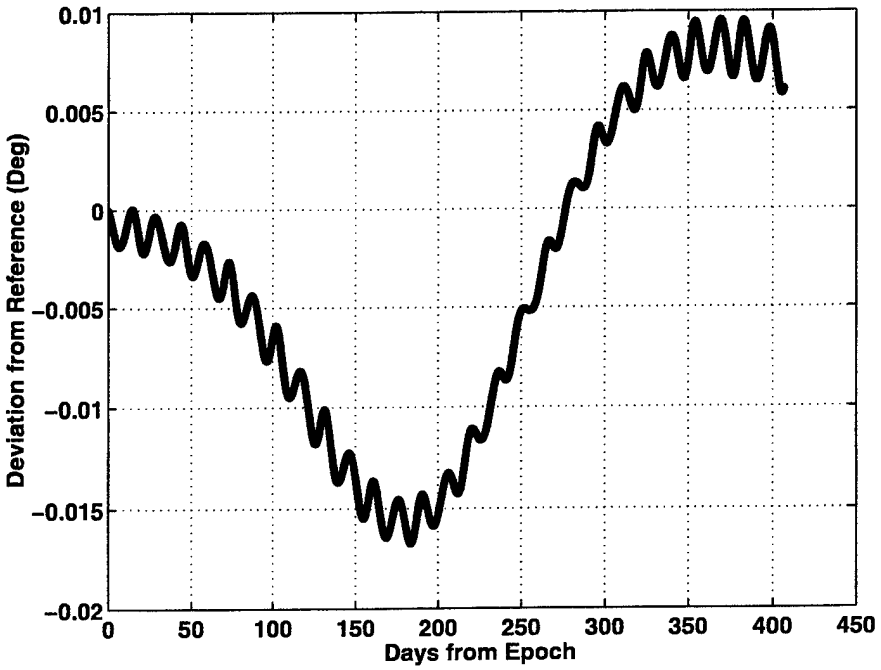


Figure D-15 1-Year Operational Planning Approach i Deviation (Limit = 0.05°)

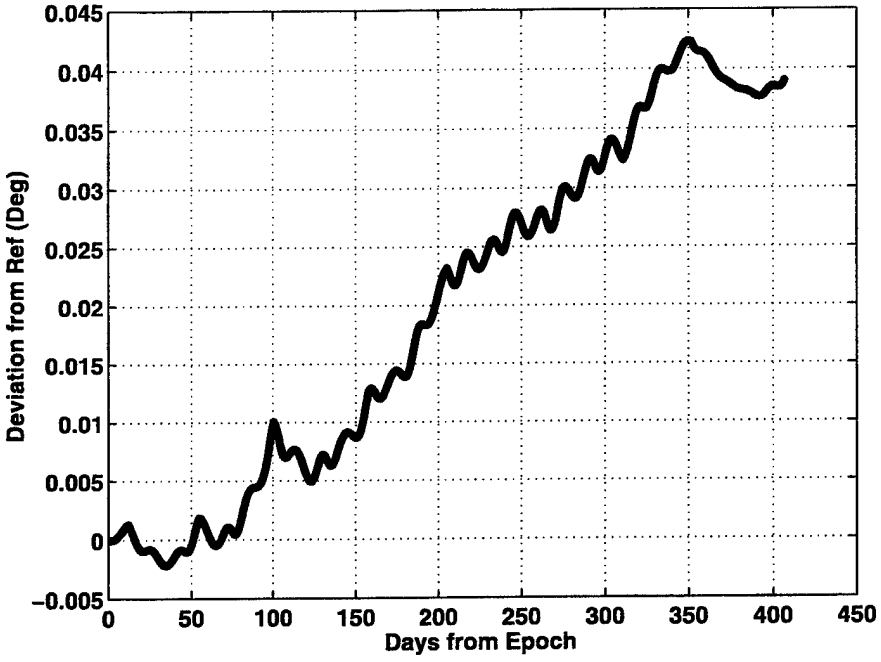


Figure D-16 1-Year Operational Planning Approach Ω Deviation (Limit = 0.5°)

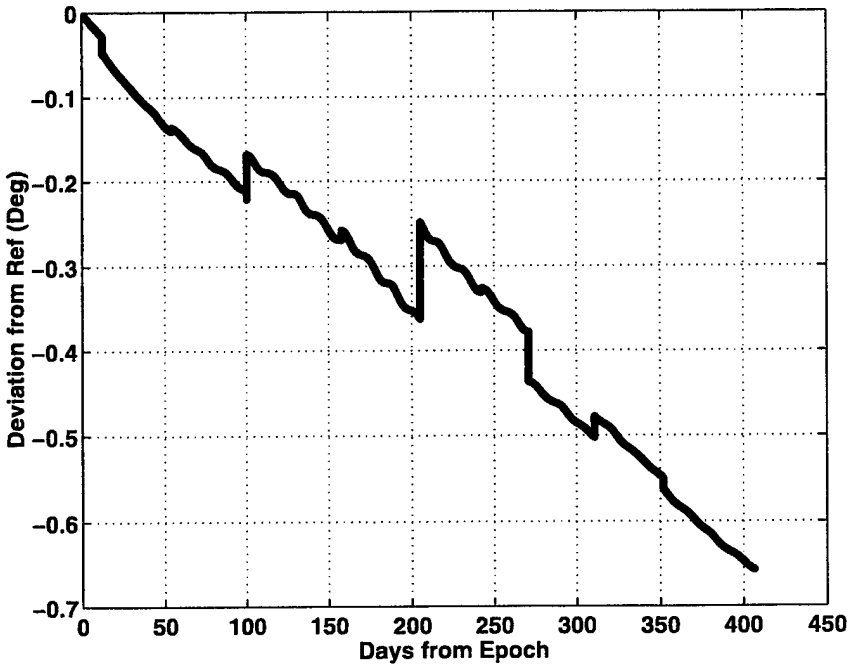


Figure D-17 1-Year Operational Planning Approach ω Deviation (Limit = 1°)

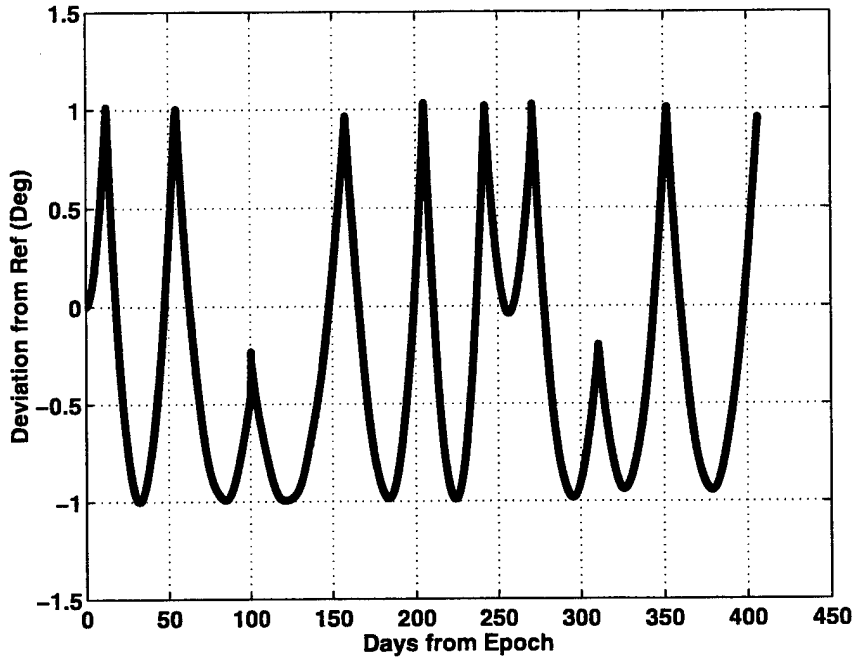


Figure D-18 1-Year Operational Planning Approach M Deviation (Limit = 1°)

D-4-3 Results of 2.5 Year ΔV Estimation from Operational Planning Tool

After slight modifications, found to be necessary as a result of the one-year case, the operational approach was applied to the Borealis™ node at noon orbit to estimate the required ΔV for 2.5 years. This section contains the element deviation history plots resulting from that optimization.

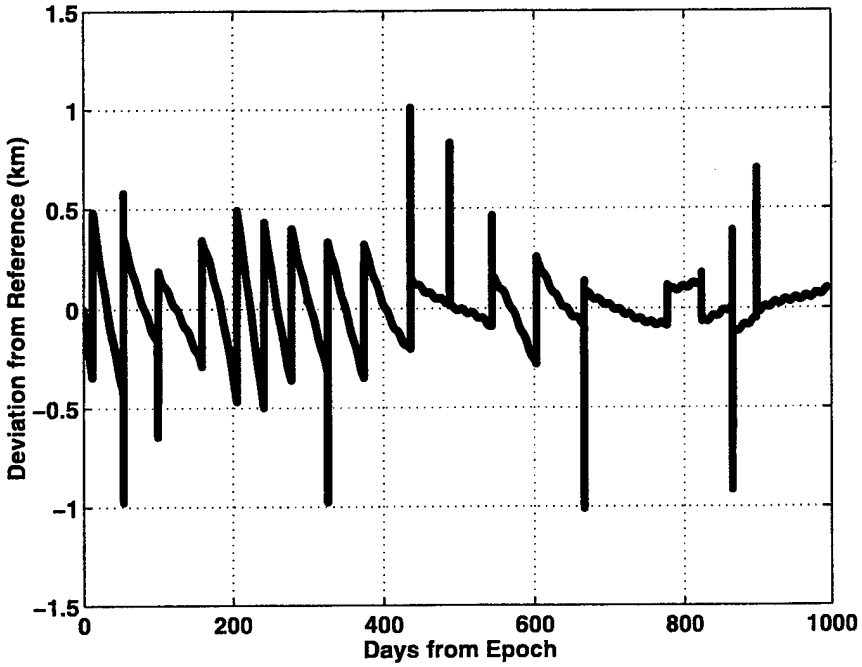


Figure D-19 2.5-Year Operational Planning Case SMA Deviation (Limit = 1 km)

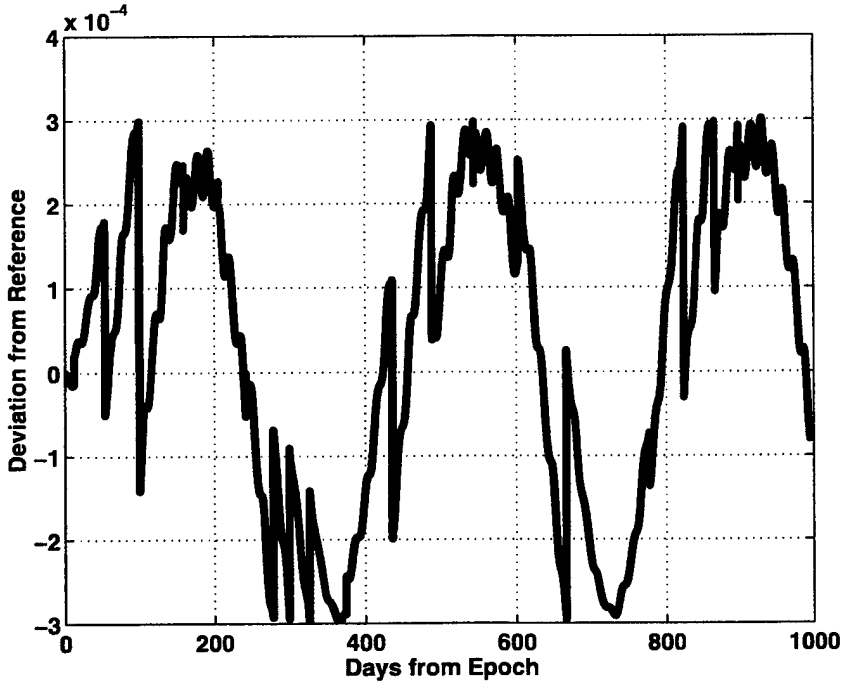


Figure D-20 2.5-Year Operational Planning Case e Deviation (Limit = 0.0003)

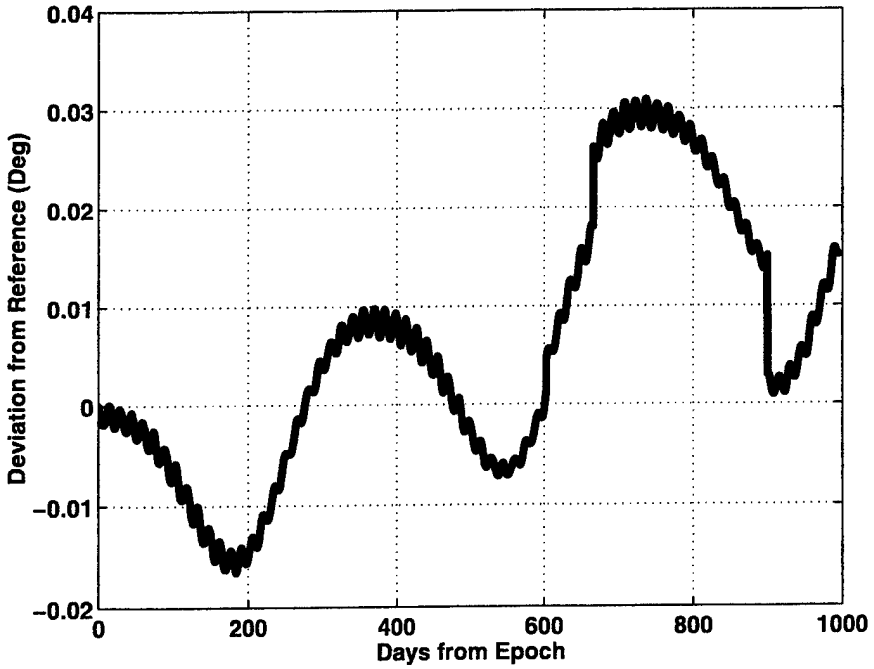


Figure D-21 2.5-Year Operational Planning Case i Deviation (Limit = 0.05°)

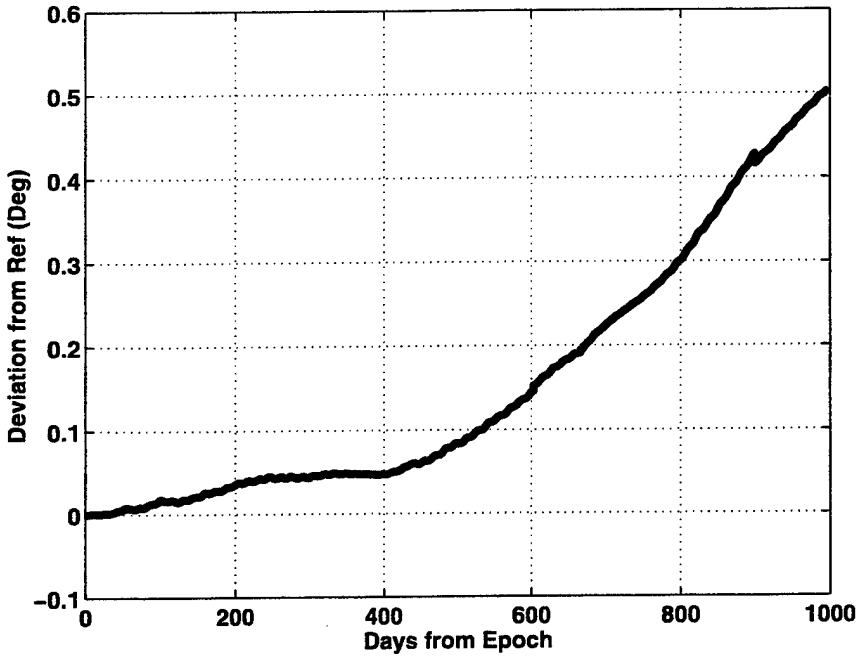


Figure D-22 2.5-Year Operational Planning Case Ω Deviation (Limit = 0.5°)

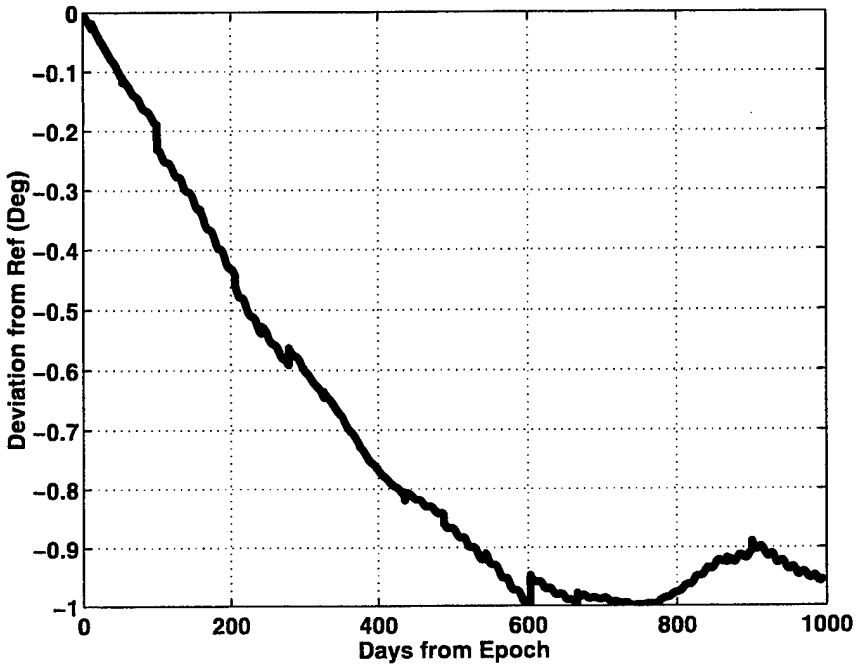


Figure D-23 2.5-Year Operational Planning Case ω Deviation (Limit = 1°)

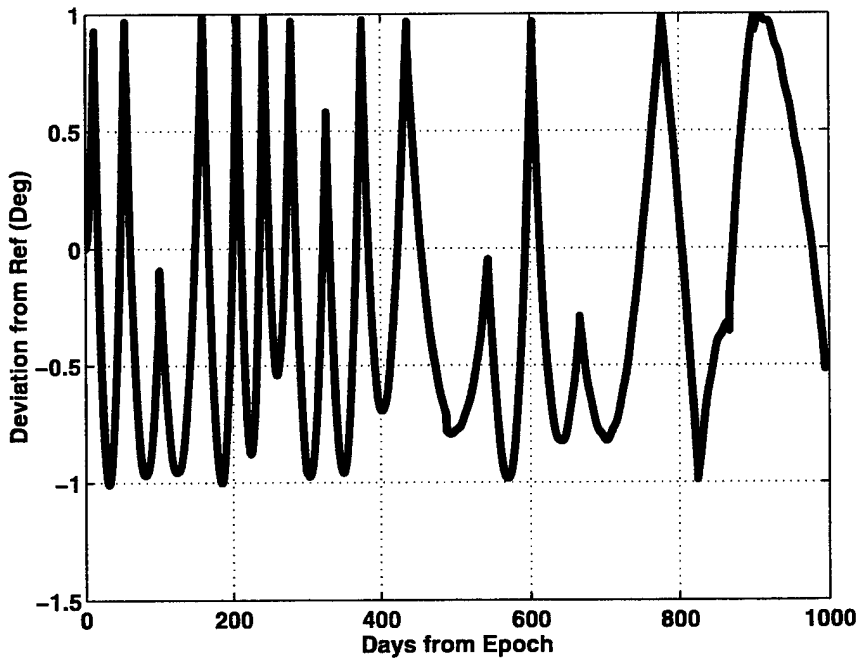


Figure D-24 2.5-Year Operational Planning Case M Deviation (Limit = 1°)

D-4-4 1200 Day Optimization Using Operational Approach Results

This section contains the plots resulting from running the operational approach as a planning tool for an additional 200 days past the end of the 2.5-year results. Both the argument of perigee and ascending node trajectories end up "crawling" along the edge as a result of the greediness of the operational approach (see section 5-4-5-2).

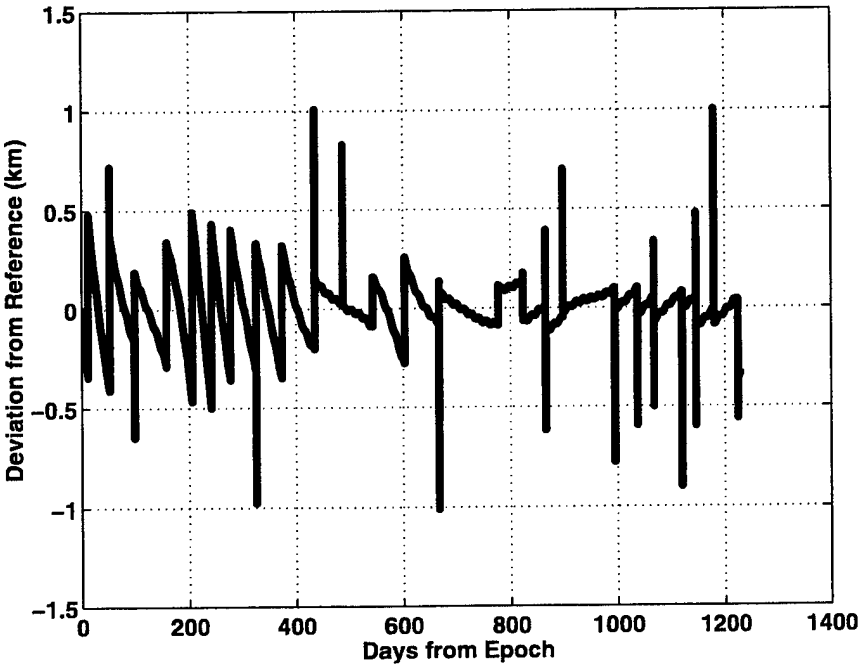


Figure D-25 1200 Day Operational Planning Case SMA Deviation (Limit = 1 km)

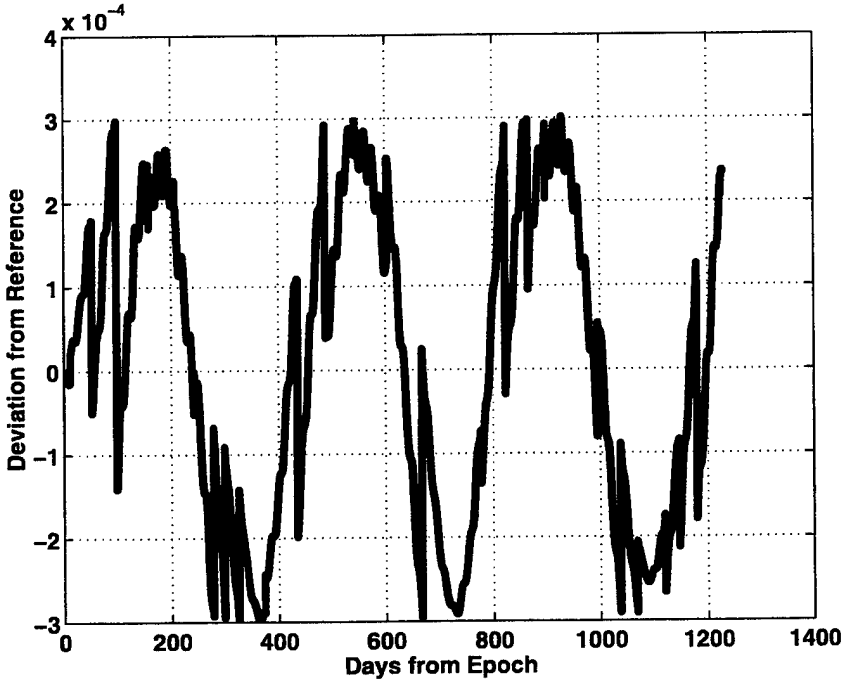


Figure D-26 1200 Day Operational Planning Case e Deviation (Limit = 0.0003)

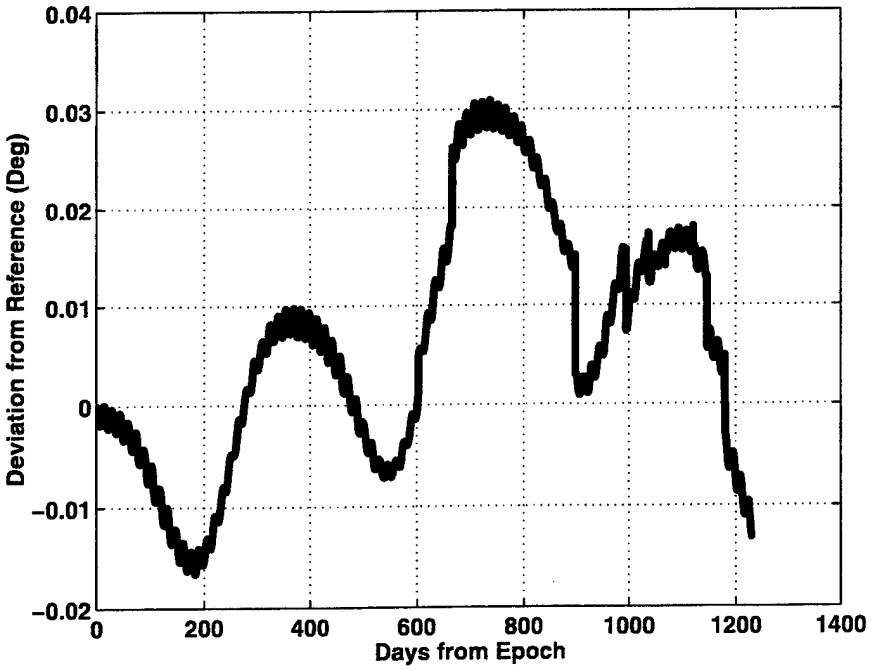


Figure D-27 1200 Day Operational Planning Case i Deviation (Limit = 0.05°)

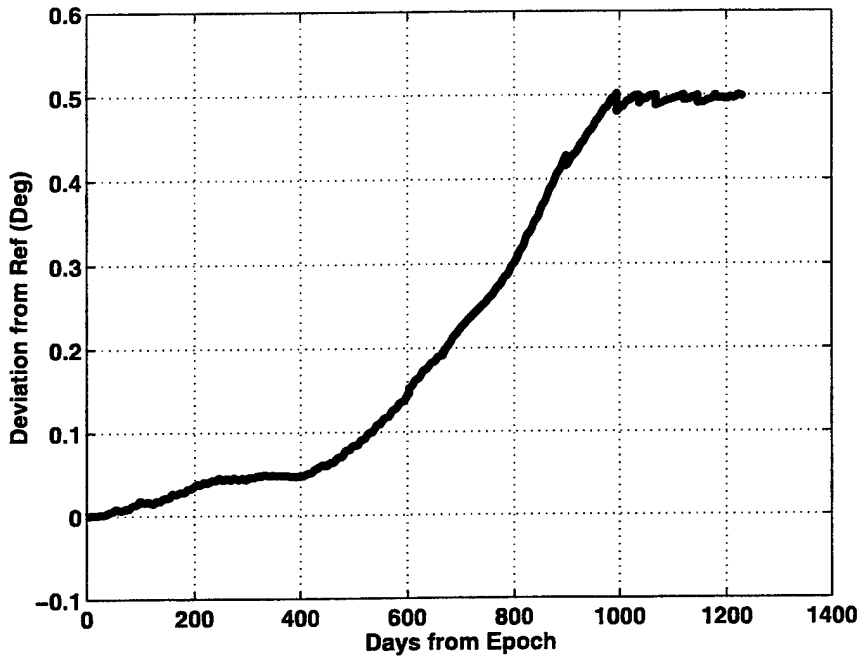


Figure D-28 1200 Day Operational Planning Case Ω Deviation (Limit = 0.5°)

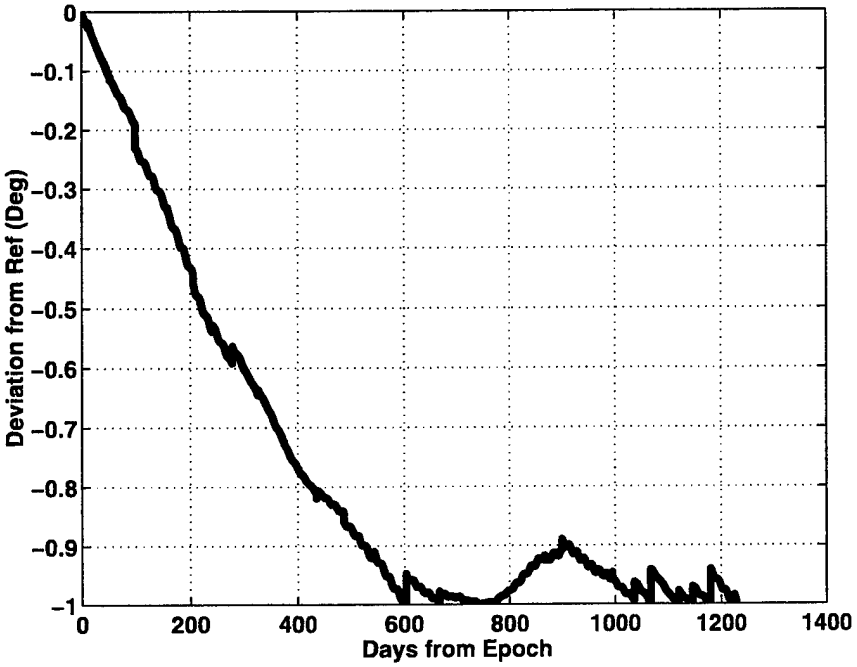


Figure D-29 1200 Day Operational Planning Case ω Deviation (Limit = 1°)

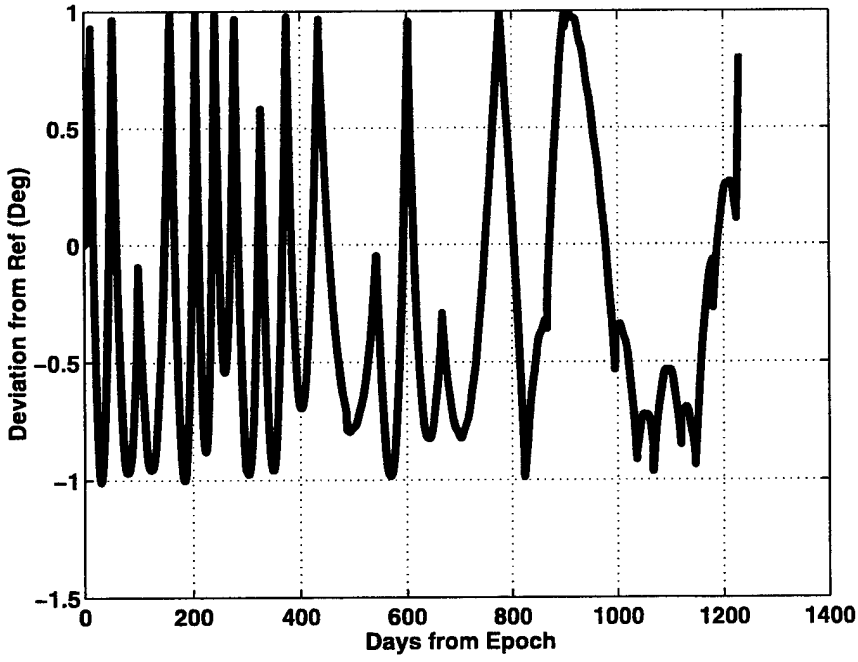


Figure D-30 1200 Day Operational Planning Case M Deviation (Limit = 1°)

[This Page Intentionally Left Blank]

References

1) *The American Heritage Dictionary of the English Language*, Houghton Mifflin Company, Boston, Massachusetts, 1981.

2) Barker, W. N., S. J. Casali and R. N. Wallner. *The Accuracy of General Perturbation and Semianalytic Satellite Ephemeris Theories,* AAS Paper 95-432, AAS/AIAA Astrodynamics Specialist Conference, Halifax, Nova Scotia, August 1995.

3) Bate, Roger R., Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics,* New York, New York: Dover Publications, Inc., 1971.

4) Battin, Richard H. *An Introduction to the Mathematics and Methods of Astrodynamics,* New York, New York: American Institute of Aeronautics and Astronautics, 1987.

5) Beightler, C. S., D. T. Phillips, and D. J. Wilde. *Foundations of Optimization,* Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

6) Broucke, R. A. and Cefola, P. J. "On The Equinoctial Orbital Elements," *Celestial Mechanics*, 1972, 5: 303-310.

7) Castiel, D., J. E. Draim and J. W. Brosius. *Elliptical Orbit Satellite System and Deployment with Controllable Coverage Characteristics,* United States Patent Number 5,582,367, 10 December 1996.

8) Castiel, D., J. W. Brosius, and J. E. Draim. *Ellipso™: Coverage Optimization Using Elliptic Orbits,* Paper AIAA-94-1098-CP, 15th AIAA International Communications Satellite Systems Conference, San Diego, California. 28 February to 3 March 1994.

9) Draim, J. E. *Optimization of the ELLIPSO and ELLIPSO 2g Personal Communications Systems,* International Workshop on Mission Design and Implementation of Satellite Systems, Toulouse, France, 17-19 November 1997.

10) Draim, J. E. "Three- and Four-Satellite Continuous Coverage Constellations," *Journal of Guidance, Control, and Dynamics,* 1985, 6: 725-730.

11) Draim, J. E., P. Cefola, R. Proulx, and D. Larsen. *Designing the Ellipso™ Satellites into the Elliptical Orbit Environment,* Paper IAF-98-A.4.03, 49th International Astronautical Congress, Melbourne, Australia, 28 September to 2 October 1998.

12) Draim, J. E. and T.J. Kacena. *Populating the Abyss—Investigating More Efficient Orbits,* Proceedings of 6th Annual AIAA/USU Conference on Small Satellites, Utah State University, Logan, Utah, 21-24 September 1992.

13) Feron, Eric. *Course Notes for Course 16.410: Introduction to Optimization and Decision Analysis,* Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 1998. In possession of author.

14) Fischer, Jack D. *The Evolution of Highly Eccentric Orbits,* CSDL-T-1310, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1998.

15) Frayssinhes, E. *Investigating New Satellite Constellation Geometries with Genetic Algorithms,* Paper AIAA-96-3636, AIAA/AAS Astrodynamics Specialist Conference, San Diego, California, 29-31 July 1996.

16) Globalstar Corporation Internet Homepage. Available at www.globalstar.com. Accessed 28 April 1999.

17) Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989.

18) Gropp, William and Ewing Lusk. *User's Guide for mpich, a Portable Implementation of MPI,* Mathematics and Computer Science Division, Argonne National Laboratory, ANL-96/6, 1996.

19) Hebert, Shane. *Message Passing Interface (MPI) FAQ,* Obtained online at http://www.erc.msstate.edu/mpi/mpi-faq.html on April 10, 1999.

20) Hillier F. S. and G. J. Lieberman. *Introduction to Operations Research,* New York, New York: McGraw-Hill, Inc., 1995.

21) Holland, J. H. *Adaptation in Natural and Artificial Systems,* Ann Arbor, Michigan: The University of Michigan Press, 1975.

22) Hulkover, Neal D. *A Reevaluation of Ellipso™, Globalstar, IRIDIUM™ and Odyssey™,* Presentation at Volpe Transportation Center, Cambridge, Massachusetts, 18 October 1994.

23) ICO Internet Homepage. Available at www.ico.com. Accessed 28 April 1999.

24) Iridium Corporation Internet Homepage. Available at www.iridium.com. Accessed 28 April 1999.

25) Kirk, Donald E. *Optimal Control Theory: An Introduction,* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1970.

26) Larson, Wiley J. and James R. Wertz. *Space Mission Analysis and Design,* Torrance, California: Microcosm, Inc., 1992.

27) Lawden, D. F. *Optimal Trajectories for Space Navigation,* Washington, D.C.: Butterworth, Inc., 1963.

28) Levine, D. *Users Guide to the PGAPack Parallel Genetic Algorithm Library,* Argonne National Laboratory, ANL 95/18, 1996. Online at http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html.

29) McClain, Wayne. *Semianalytic Artificial Satellite Theory: Vol. I: Application of the Generalized Method of Averaging to the Artificial Satellite Program,* privately published, 1992. Copy available from the author.

30) Neelon, Joseph G., Jr. *Orbit Determination for Medium Altitude Eccentric Orbits Using GPS Measurements,* CSDL-T-1330, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February, 1999.

31) Neelon, J., P. Cefola, and R. Proulx. "Current Development of the Draper Semi-Analytical Satellite Theory Standalone Orbit Propagator Package," *Advances in the Astronautical Sciences,* Volume 97 Part II, American Astronautical Society, 1998.

32) Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing—Second Edition,* New York, New York: Cambridge University Press, 1992.

33) Proulx, Ronald J, James E. Smith, John E. Draim, and Paul J. Cefola. *Ellipso™ Gear Array: Coordinated Elliptical/Circular Constellations,* Paper 98-4383, AIAA/AAS Astrodynamics Specialist Conference, Boston, Massachusetts, 10-12 August 1998.

34) Sabol, Christopher. *Application of Sun-Synchronous, Critically Inclined Orbits to Global Personal Communications Systems,* CSDL-T-938, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, January 1987.

35) Sellers, Jerry Jon. *Understanding Space: An Introduction to Astronautics,* New York: McGraw-Hill, Inc., 1994.

36) Shah, Naresh, H. *Automated Station-Keeping for Satellite Constellations,* CSDL-T-1288, Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1997.

37) Shah, N., R. J. Proulx, B. Kantsiper, P. J. Cefola, and J. E. Draim. *Automated Station-Keeping for Satellite Constellations,* Paper #C-7, International Workshop on Mission Design and Implementation of Satellite Constellations, Toulouse, France, 17-19 November 1997. CSDL-P-3605.

38) Simon, H. A. *The Sciences of the Artificial,* Cambridge, Massachusetts: MIT Press, 1969.

39) Smith, James E., Ronald J. Proulx, Paul J. Cefola, and John E. Draim. *Optimal Station-Keeping Strategies via Parallel Genetic Algorithms,* Paper AAS 99-123, AAS/AIAA Space Flight Mechanics Meeting, Breckenridge, Colorado, 7-10 February 1999.

40) Snir, Marc, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference,* Cambridge, Massachusetts: The MIT Press, 1996.

41) Swan, Peter A. "Iridium Gets Real," *Aerospace America*, Vol. 37, No. 2, February 1999.

42) Teledesic Internet Homepage. Available at www.teledesic.com. Accessed 28 April 1999.

43) Turner, A.E. *New non-Geosynchronous Orbits for Communications Satellites to Off-Load Daily Peaks in Geostationary Traffic*, AAS Paper 87-547, AAS/AIAA Astrodynamics Specialists Conference, Kallispell, Montana, 10-13 August 1987.

44) Vallado, David A., *Fundamentals of Astrodynamics and Applications.* New York, New York: McGraw-Hill, Inc., 1997.

45) Van Deventer, Paul G. *Flight Control Command Generation in a Real-Time Mission Planning System Using Constrained Genetic Optimization,* Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1992.

46) Walker, J. G. "Satellite Constellations," *Journal of the British Interplanetary Society,* 1984, 37: 559-572.

47) Yurasov, V. *Universal Semi-analytic Satellite Motion Propagation Method,* US/Russian Space Surveillance Workshop, Poznan, Poland, 5 July 1996.

Biography

James Earl Smith was born on 10 January 1973 in Aurora, Colorado as the second of six children. Following short stints in Utah and California, his family settled in Boise, Idaho where he was raised. After graduating as the valedictorian from Meridian High School in Meridian, Idaho in 1991, he applied and was accepted for entrance into the United States Air Force Academy Class of 1995. After completion of one year at the Air Force Academy he resigned his commission to serve for two years as a volunteer missionary in the Japan Osaka Mission of the The Church of Jesus Christ of Latter-day Saints.

Following the two-year mission, he reapplied and was again accepted to the Air Force Academy, this time to the Class of 1997. He was commissioned as a Second Lieutenant in the Air Force and graduated from the Air Force Academy as a distinguished graduate with a B.S. in Astronautical Engineering in May of 1997. At that time he was also honored as the top cadet in a Far Eastern Language, the top cadet in Astronautical Engineering, the top cadet in the Engineering Divisions, the top cadet in Academic Performance, and as top overall cadet in the graduating Class of 1997. Additionally the American Institute of Aeronautics and Astronautics honored him as the 1997 AIAA Rocky Mountain Section Student Engineer of the Year.

Following graduation from the Air Force Academy, Lieutenant Smith was awarded a Draper Fellowship by The Charles Stark Draper Laboratory and his first assignment as an Air Force officer was to attend graduate school at the Massachusetts Institute of Technology in Cambridge, Massachusetts. He served as a student at M.I.T. from May 1997 to May of 1999. During that time he also married Kristine Cromar of Colorado Springs, Colorado and they were blessed with the birth of their first son, James Nathan Smith. Upon completion of the S.M. degree in Aeronautics and Astronautics in May of 1999, Lt. Smith was promoted to First Lieutenant and received assignment to the 2nd Space Operations Squadron at Schriever Air Force Base, Colorado to serve as a GPS spacecraft systems engineer.

Lt. Smith is a member of the American Institute of Aeronautics and Astronautics, Tau Beta Pi, Sigma Gamma Tau, and Sigma Xi.