

**ORBIT DETERMINATION FOR MEDIUM ALTITUDE ECCENTRIC
ORBITS USING GPS MEASUREMENTS**

by

Joseph G. Neelon, Jr.

B.S. Aerospace Engineering
The Pennsylvania State University
(1996)

B.S. General Studies
Kutztown University
(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

February 1999

© 1998 Joseph G. Neelon, Jr. All rights reserved.

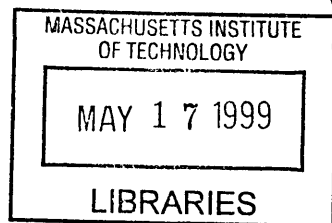
The author hereby grants to MIT permission to reproduce and to distribute publicly paper
and electronic copies of this thesis document in whole or in part.

Signature of Author _____
Department of Aeronautics and Astronautics
December 1998

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



Aero

(This page intentionally left blank.)

ORBIT DETERMINATION FOR MEDIUM ALTITUDE ECCENTRIC ORBITS USING GPS MEASUREMENTS

by

Joseph G. Neelon, Jr.

Submitted to the department of Aeronautics and Astronautics on December 9, 1998 in partial fulfillment of the requirements for the Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

Recent satellite constellation developers have increasingly considered non-traditional medium altitude circular and elliptical orbits. These projects will track their satellites using the GPS system, which has been used on LEO satellites for several years. But can GPS be used as the sole input for the orbit determination of satellites in these medium altitude eccentric orbits? To determine the feasibility of such a process, a program was constructed to produce simulated GPS navigation solutions. These simulated navigation solutions were used in place of actual solutions that do not exist. The simulation included realistic error models for the GPS satellite clocks and for ionospheric delays, and it evaluated the visibility between the GPS satellites and their target. The navigation solutions were computed using all GPS signals received by the target satellite. To increase the speed of calculation, the simulation used a modified version of the self-scheduling architecture for parallel processing and implemented it using the MPI Standard. The orbits used in this simulation were the Ellipso Borealis, Ellipso Concordia, and Molniya orbits. The navigation solutions produced from these orbits were used as the input to an orbit determination process that employed the Goddard Trajectory Determination System (GTDS) program. This orbit determination process was performed using the differential corrections function of GTDS. The determined orbit was compared to the orbits used in the production of the simulated navigation solutions. The results of this comparison implied that the orbits could be reproduced from GPS navigation solutions if the precise force models were used in the determination process. Testing was conducted to examine the effects of force mismodeling on the orbit determination process.

Thesis Supervisor: Dr. Ronald J. Proulx

Title: Principle Member of the Technical Staff, The Charles Stark Draper Laboratory, Inc.

Thesis Supervisor: Dr. Paul J. Cefola

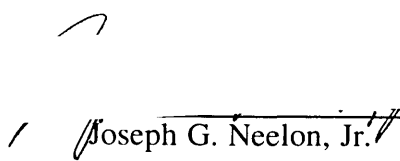
Title: Lecturer, Department of Aeronautics and Astronautics

Program Manager, The Charles Stark Draper Laboratory, Inc.

ASSIGNMENT

Draper Laboratory Report Number T - 1330

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.


/ Joseph G. Neelon, Jr. /

12/10/98
Date

Acknowledgements

Although I am the one who gets credit for this work, credit is also due to several people who helped me along the way. Their advice and support throughout this project was invaluable.

I would like to thank Dr. Paul Cefola and Dr. Ronald Proulx for giving me the opportunity to work for them and for putting up with me for two and a half years. They taught me a lot and pushed me to do better than I thought possible.

I want to extend my appreciation to the USAF Phillips Laboratory for their financial support during the my first year at MIT, with special thanks to Lt. Col. David Vallado.

To Neil Dennehy, for his input and his offers of assistance at various times in the past six months. To John Draim from Ellipso, Inc., for his insights and interest in this work, and for his enthusiasm about its completion.

To John Sweeney, Joan Schiffer, and Arel Maguire from the education office for their assistance in administrative matters with MIT.

To Dr. William Blume and the people at JPL, for their assistance with information on VSOP and for the incentive to complete this work as quickly as possible.

To Capt. Scott Carter, USAF, for paving the way for this work in his Master's thesis. To Miranda Barrows, for her assistance in the development of the MPI application.

To Jack and Elizabeth Fischer, for always having confidence in my abilities and for letting me know it. To Heidi Perry, for her words of encouragement. To Jim Smith and George Granholm, for their friendly ribbing which caused me to put in that much more effort.

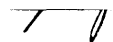
I would also like to thank my family. To my father, who was always interested in my progress, even if he was a little pushy sometimes. To my mother, who is always willing to exercise her bragging rights. To my grandparents, for their generosity and their concern about my future. To my grandfather who has been watching me from above for the past six years, for his pride in my accomplishments.

Lastly, I would like to thank God for the gifts He has given me and the opportunities that He has presented me. I will do my best to reach the potential He has planted in me.

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

Publication of this thesis does not constitute approval by the Charles Stark Draper Laboratory or the Massachusetts Institute of Technology of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Permission is hereby granted by the Charles Stark Draper Laboratory, Inc., to the Massachusetts Institute of Technology to reproduce any or all of this thesis.

 Joseph G. Neefon, Jr.

(This page intentionally left blank.)

Contents

Chapter 1	Introduction.....	19
1.1	The GPS System.....	21
1.1.1	GPS System Segments.....	22
1.1.2	Operation.....	23
1.1.3	GPS Related Errors.....	28
1.1.3.1	Environmental errors.....	28
1.1.3.2	Inherent errors.....	31
1.1.3.3	Intentional errors.....	34
1.2	Eccentric Orbits and GPS.....	34
1.2.1	Visibility Issues.....	36
1.2.2	Ionospheric Interference.....	40
1.3	GPS and Orbit Determination.....	42
1.3.1	Review of Past Concepts.....	42
1.3.2	Current Project.....	45
1.4	Thesis Overview.....	46
Chapter 2	The Computation of Satellite Constellation Motion via Parallel Processing.....	47
2.1	Parallel Processing.....	47
2.1.1	Parallel Virtual Machine (PVM).....	49
2.1.2	Message Passing Interface (MPI).....	51
2.1.2.1	MPI functions.....	53
2.2	Applications of Parallel Processing to Constellation Motion.....	55
2.2.1	Applications at Draper Laboratory.....	55

2.2.2	Applications External to Draper Laboratory	56
2.2.3	Current Application	58
2.3	Improved Draper Semianalytic Satellite Theory (DSST) 58Standalone Orbit Propagator	58
2.3.1	History.....	59
2.3.2	Current Functions	61
Chapter 3	Construction of GPS Receiver Navigation Solutions including Realistic Errors - Algorithms	65
3.1	Generation of the Target Satellite's Orbit.....	66
3.2	The Simulation.....	68
3.2.1	Determination of the "Truth" Positions	69
3.2.2	Identification of GPS Satellites in View	73
3.2.3	Simulation of Signal Errors	79
3.2.3.1	Clock errors.....	79
3.2.3.2	Ionospheric effects.....	82
3.2.3.3	Ephemeris errors	88
3.2.4	Producing the Navigation Solutions	90
3.2.4.1	A priori estimate of position	90
3.2.4.2	Satellite selection	92
Chapter 4	Construction of GPS Receiver Navigation Solutions including Realistic Errors - Simulation Architecture.....	95
4.1	Self-Scheduling Architecture.....	95
4.1.1	Advantages of this Architecture.....	98
4.1.2	Communications.....	100
4.2	Interaction between Architecture and Placement of Computational Segments	101

4.2.1	The Master's Task.....	103
4.2.2	The Slaves' Work	105
4.2.3	The Navslave's Job	108
4.2.4	Structured Data Files.....	110
4.3	Graphic Processing	111
4.3.1	Stripper Utility.....	113
Chapter 5	Orbit Determination Error Analysis Results.....	115
5.1	Validation of the Simulation.....	116
5.1.1	The Satellite in View Algorithm	117
5.1.2	The Satellite Clock Algorithm.....	123
5.1.3	The Navigation Solution Algorithm.....	123
5.1.4	The Entire Simulation.....	125
5.2	Orbit Determination (OD).....	129
5.2.1	Validation of the Deterministic Analysis Process.....	130
5.2.2	Ellipso	133
5.2.2.1	Borealis.....	134
5.2.2.2	Concordia.....	144
5.2.3	Molniya.....	149
5.3	Effects of Mismodelling	153
5.3.1	Borealis.....	154
5.3.2	Concordia	162
5.3.3	Molniya.....	162
5.4	Disposition of OD Related Data.....	165

Chapter 6	Conclusions and Future Work.....	167
6.1	Conclusions.....	169
6.2	Recommendations.....	171
6.3	Future Work	172
6.3.1	GPS Ephemeris Errors	173
6.3.2	Improving the Ionospheric Model.....	173
6.3.3	Relativistic Effects.....	174
Appendix A	Software Creation and Modification.....	177
A.1	New Programs	178
A.1.1	gps_pred.for.....	178
A.1.2	master.for.....	179
A.1.3	navslave.for	180
A.1.4	slave.for.....	183
A.2	Stubs	189
A.3	Modified Subroutines	190
A.4	Source Code.....	191
A.4.1	gps_pred.for.....	191
A.4.2	master.for.....	193
A.4.3	navslave.for	196
A.4.4	slave.for.....	199
Appendix B	The Goddard Trajectory Determination System (GTDS).....	203
B.1	The History of GTDS/DSST	203
B.2	Orbit Propagators	207

B.2.1	Special Perturbations.....	207
B.2.1.1	Cowell.....	208
B.2.1.2	VOP	209
B.2.2	General Perturbations.....	214
B.2.3	Semianalytic Methods.....	215
B.2.3.1	DSST.....	215
B.3	Operations	222
Appendix C	Space Qualified GPS Receivers.....	227
References	231

(This page intentionally left blank.)

Table of Figures

Figure 1.1	Pseudorandom Noise Coding.....	25
Figure 1.2	Phase Measurement of PRN Code.....	25
Figure 1.3	Layers of the Ionosphere.....	29
Figure 1.4	Ranging Errors from Satellite Clock Dither.....	35
Figure 1.5	Examples of GPS Visibility.....	37
Figure 1.6	Visible Surface Area.....	38
Figure 1.7	Paths through Ionosphere (ground-based).....	41
Figure 1.8	Worst Case Ionospheric Delay.....	41
Figure 3.1	Deterministic GPS Navigation Solution Analysis Data Flow.....	65
Figure 3.2	Data Flow of “Truth” Orbit Generation.....	67
Figure 3.3	“Truth” Positions.....	72
Figure 3.4	Earth Shadowing Test.....	74
Figure 3.5	Geometry for Verification.....	75
Figure 3.6	Lost Time vs. Antenna Angle (within plane).....	77
Figure 3.7	Lost Time vs. Antenna Angle (out of plane).....	78
Figure 3.8	Example of Graze Radius.....	84
Figure 3.9	Determination of Graze Radius.....	86
Figure 3.10	Subcases of $r_{\text{graze}} < r_{\text{ionmax}}$	87
Figure 3.11	Subcases of $r_{\text{graze}} < r_{\text{ionmin}}$	88
Figure 4.1	Architecture #1.....	96
Figure 4.2	Architecture #2.....	97
Figure 4.3	Program Structure.....	99
Figure 4.4	Master Subroutine Structure.....	105
Figure 4.5	Slave Subroutine Structure.....	107

Figure 4.6	Navslave Subroutine Structure	111
Figure 5.1	Deterministic Analysis Data Flow	115
Figure 5.2	Boundaries for 70° Antennas	118
Figure 5.3	Number of Satellites in View for Borealis, Node at Noon (zenith antenna).....	121
Figure 5.4	Number of Satellites in View for Borealis, Node at Noon (nadir antenna).....	121
Figure 5.5	Number of Satellites in View for Borealis, Node at Noon (both antennas)	122
Figure 5.6	Ranging Errors from Satellite Clock Dither	124
Figure 5.7	Simulated Ranging Errors.....	124
Figure 5.8	Total Errors in Navigation Solutions with No Simulated Errors	126
Figure 5.9	Comparison of Total Errors with GDOP	126
Figure 5.10	Comparison between Simulated Navigation Solutions and TOPEX “Truth”	128
Figure 5.11	Comparison between Actual Navigation Solutions and POE	128
Figure 5.12	Data Flow of the Orbit Determination Process.....	130
Figure 5.13	Errors in the Deterministic Analysis Process (Fit Span).....	132
Figure 5.14	Errors in the Deterministic Analysis Process (Predict Span).....	132
Figure 5.15	Borealis NAN Navigation Solution Errors - Two Antennas, Blocked Ionosphere.....	135
Figure 5.16	OD Errors in Borealis, Node at Noon - Two Antennas, 1 Minute Density, Maximum Length (Fit Span)	137
Figure 5.17	OD Errors in Borealis, Node at Noon - Two Antennas, 1 Minute Density, Maximum Length (Predict Span)	137
Figure 5.18	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 1 Minute Density, Maximum Length (Fit Span).....	138
Figure 5.19	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 1 Minute Density, Maximum Length (Predict Span).....	138

Figure 5.20	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Fit Span).....	140
Figure 5.21	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span).....	140
Figure 5.22	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Fit Span).....	142
Figure 5.23	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 1 Week of Obs. (Fit Span).....	142
Figure 5.24	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Predict Span).....	143
Figure 5.25	OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 1 Week of Obs. (Predict Span).....	143
Figure 5.26	OD Errors in Borealis, Node at Midnight - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span).....	146
Figure 5.27	OD Errors in Borealis, Node at Midnight - Zenith Antenna, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span).....	146
Figure 5.28	OD Errors in Concordia - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Fit Span)	148
Figure 5.29	OD Errors in Concordia - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Predict Span).....	148
Figure 5.30	Number of Satellites in View for the Molniya (both antennas)	150
Figure 5.31	Crosstrack Errors in Molniya - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Fit Span).....	151
Figure 5.32	OD Errors in Molniya - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Fit Span).....	152
Figure 5.33	OD Errors in Molniya - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span).....	152
Figure 5.34	Prediction Errors in Borealis NAN - Mismodeling Solar Radiation Pressure, Solar Reflectivity (-25%).....	155

Figure 5.35	Prediction Errors in Borealis NAN - Mismodeling Solar Radiation Pressure, Solar Reflectivity (+25%)	155
Figure 5.36	Prediction Errors in Borealis NAN - Mismodeling Drag, Drag Coefficient (-25%).....	156
Figure 5.37	Prediction Errors in Borealis NAN - Mismodeling Drag, Drag Coefficient (+25%).....	156
Figure 5.38	OD Errors in Borealis NAN - Mismodeling Drag, Harris-Priester (Fit Span).....	157
Figure 5.39	OD Errors in Borealis NAN - Mismodeling Drag, Harris-Priester (Predict Span).....	157
Figure 5.40	OD Errors in Borealis NAN - Mismodeling Drag, '92 Epoch and Harris-Priester (Fit Span)	159
Figure 5.41	OD Errors in Borealis NAN - Mismodeling Drag, '92 Epoch and Harris-Priester (Predict Span)	159
Figure 5.42	OD Errors in Borealis NAN - Mismodeled Geopotential Size, 21 x 21 (Predict Span).....	160
Figure 5.43	Difference between JGM2 21 x 21 and GEMT3 21 x 21 (Fit Span).....	161
Figure 5.44	Difference between JGM2 21 x 21 and GEMT3 21 x 21 (Predict Span).....	161
Figure 5.45	Prediction Errors in Concordia -Mismodeled Solar Radiation Pressure, Solar Reflectivity (-25%).....	163
Figure 5.46	Prediction Errors in Concordia -Mismodeled Solar Radiation Pressure, Solar Reflectivity (+25%).....	163
Figure 5.47	OD Errors in Molniya - Mismodeled Geopotential Size 12 x 12 (Predict Span).....	164
Figure B.1	Orbital Motion Decomposition.....	216
Figure B.2	Construction of a Keyword Card.....	225

Table of Tables

Table 3.1	Orbital Elements of the GPS Constellation.....	71
Table 5.1	Error Characteristics for Borealis NAN (in meters).....	135
Table 5.2	Borealis NAN OD Position Errors (in meters) - 4 Minute Density.....	139
Table 5.3	Borealis NAN OD Velocity Errors (in millimeters/sec) - 4 Minute Density.....	139
Table 5.4	Borealis NAN OD Position Errors (in meters) - 15 Minute Density, Two Antennas.....	141
Table 5.5	Borealis NAN OD Velocity Errors (in millimeters/sec) - 15 Minute Density, Two Antennas.....	141
Table 5.6	Error Characteristics for Borealis NAM (in meters).....	144
Table 5.7	Borealis NAM OD Position Errors (in meters) - 4 Minute Density.....	145
Table 5.8	Borealis NAM OD Velocity Errors (in millimeters/sec) - 4 Minute Density.....	145
Table 5.9	Concordia OD Position Errors (in meters) - Two Antennas, 15 Minute Density.....	147
Table 5.10	Concordia OD Velocity Errors (in millimeters/sec) - Two Antennas, 15 Minute Density	147
Table 5.11	Error Characteristics for Molniya (in meters).....	149
Table 5.12	Molniya OD Position Errors (in meters) - 4 Minute Density, 2 Weeks of Obs.	151
Table 5.13	Molniya OD Velocity Errors (in millimeters/sec) - 4 Minute Density, 2 Weeks of Obs.	151
Table 5.14	Borealis NAN OD Position Errors (in meters) - '92 Epoch.....	158
Table 5.15	Borealis NAN OD Velocity Errors (in millimeters/sec) - '92 Epoch.....	158
Table 5.16	Molniya OD Position Errors (in meters) - Mismodeling Geopotential Field.....	164
Table 5.17	Molniya OD Velocity Errors (in millimeters/sec) - Mismodeling Geopotential Field.....	164
Table C.1	Spaceborne GPS Receivers	228

(This page intentionally left blank.)

Chapter 1

Introduction

Until the 1990's, most space vehicles were the property of governments [34,41]. The U.S. and Russia each competed to accomplish space milestones before the other could. Military and government organizations used satellites for exploration, surveillance, and a secure means of communication. However, there were a significant number of commercial space applications in operation at that time.

The commercial market for space has increased dramatically in the past decade. The primary commercial application has been the field of communication [41]; more communication satellites are in orbit than ever before, and the number is continuing to grow. Private companies have also become interested in space exploration, and are launching vehicles to pursue the knowledge they desire.

All satellites must have a place to reside. A satellite's orbit is selected based on the mission requirements, the benefits and hazards of various altitudes, and the availability of space at its desired altitude. The most popular orbits to date have been low Earth (LEO) and geosynchronous (GEO) orbits. Of these two, LEO is more frequently chosen for at least one of the following three reasons: first, a launch vehicle can place a more massive satellite into a lower orbit; second, there is a lower cost associated with placing a satellite into LEO; and third, having a satellite in LEO allows the possibility of maintenance and repair. The main appeal of the GEO orbits is that the satellite remains above the same area of the Earth at all times, with a secondary benefit of being able to view an entire hemisphere.

Early satellites were sent up into LEO to perform experiments and allowed to die on orbit with no concern about de-orbiting the vehicles. Some of these satellites have been

damaged and have had pieces break off. There have also been accidents throughout the history of space launches and satellite operations, including explosions and collisions with debris or other satellites. These accidents produce large amounts of debris, which in turn can cause more collisions. Due to the minute size of most of the debris fragments, most collisions go unnoticed. Other contributions to this debris field are the rocket bodies of multistage launch vehicles, which were ejected from their vehicles above the altitude where they would fall back to Earth. With the current populous of objects (debris and satellites, active or dead) in the LEO altitudes and the advent of new systems that use large numbers of satellites, like Globalstar(48) [61], Iridium(66) [59], and Teledesic(288) [60], there are some serious concerns about the increased likelihood of satellites colliding with other satellites [65] or debris [66].

Eccentricity is another important issue in the selection of an appropriate orbit. All orbits are eccentric, just to different extents. Most satellites travel in nearly circular, low eccentricity orbits. But higher eccentricity can be used to accomplish specific mission objectives. For example, the Russian Molniya constellation used a highly eccentric orbit design to produce extended duration passes over Russia and North America on a daily basis [19,34]. The new Ellipso constellation is also using an elliptical orbit for two of its three planes to economically provide increased coverage of the northern hemisphere [14,15]. These are just two examples of systems that used or will use elliptical orbits. Another potential use of eccentric orbits is to allow a satellite to perform its required operations in LEO while limiting its exposure to the high debris environment.

As more new systems are developed, designers may want to use higher and more eccentric orbits than the current norm. Their reasoning may be debris avoidance, satellite interference, system cost, or other higher-level mission requirements. Regardless of their reasons, the operators of the systems will still need to track their satellites.

In recent years, the Global Positioning System (GPS) has seen a tremendous surge in popularity. GPS receivers (GPSR) can be purchased by anyone who wants one. They

are available in hand-held units or in units that can be mounted on cars, boats and airplanes. They have even been mounted on satellites and the Space Shuttle. In fact, satellites equipped with GPS receivers were flown as early as 1982 [8]. These satellites all flew in LEO orbits. The earliest of these satellites were used to test GPS and to evaluate how well GPS receivers could determine the satellites' positions. Using GPS onboard a user satellite basically gives that user satellite some ability to track itself, and from this data, the satellite's orbit can be determined and predicted.

Until now, GPS receivers have not flown in medium (MEO) or high Earth orbits (HEO), but they are being tested for performance on GEO satellites. Also, they have flown in very few elliptical orbits [17]. GPS performance in these orbital cases is questionable. Is it possible to use GPS for the determination and prediction of these types of orbits? That is the question behind this thesis.

1.1 The GPS System

GPS is the global positioning system that was developed by the U.S. Armed Forces [46]. It is a high-precision navigational system that works on the concept of radio ranging. The following sections will describe the operational segments of the GPS system, how the system operates, and the errors associated with its operation. Throughout these descriptions, the term satellite will refer only to a GPS satellite. This discussion of GPS focuses on the civilian user. In particular, the discussion does not include information about the precision (P) code, only the coarse acquisition (C/A) code.

1.1.1 GPS System Segments

The GPS system consists of three operational segments [53]: the Control segment, the Space segment, and the User segment. If either of the first two segments are not operating properly, the performance of GPS is degraded. A failure in the User segment would only affect the individual user and could be corrected easily.

The Control segment [23,53] consists of a series of ground-based monitoring stations and a master control station. The monitoring stations measure the pseudoranges and carrier phases of the GPS satellites and their transmitted signals. The data is sent to the master control station to verify accurate operation. The pseudorange and carrier information are processed to remove known sources of error. The smoothed data is used as input to a Kalman filter that predicts the GPS satellites' positions using a linearized model of their motion. This linearized model uses the sum of the reference states and their residuals in place of the current state. The reference states are taken from the reference trajectories, which are produced by propagating from known states with the non-linear model of motion. To prevent the reference states from deviating too far from the actual states, these reference trajectories are updated when they fail linearity tests. The Kalman filter uses the residuals from one-half hour before the upload time and the partial derivatives from the reference trajectories to produce the ephemeris estimates by propagating through the entire time span of the upload. The residuals between the ephemeris estimates and the reference trajectories are divided into segments that correspond to the fit intervals from the time of upload. These segments of the residuals are processed through a weighted least squares fit to generate corrections that are applied to the previously predicted Keplerian elements. The new Keplerian elements are converted to the coefficients that the GPS satellites transmit in their navigational messages [23,51]. These values are used by the GPS receiver to calculate the position of the GPS satellite that sent the signal. On at least a

daily basis, the master control station sends updates to the GPS satellites that include this data.

The Space segment [53] is a constellation of 24 semisynchronous satellites that are disbursed in six evenly-spaced orbital planes; within the planes, the satellites are unevenly spaced to reduce the effects of satellite outages. These satellites continuously transmit a coded signal toward the Earth. The coding of these signals indicates which satellite sent the transmission. This coding can also be used to determine the time of signal transmission. The signal contains a message with data that the users' receivers need to calculate their positions.

The User segment [53] was originally limited to the military, but it has expanded to the general public. Anyone with a GPS receiver can make use of the navigational signals. However, the signals received by the general public differ from those received by military users. Based on their level of access, the GPS receivers take the messages received from the satellites and process the data to determine the users' positions.

1.1.2 Operation

Each satellite transmits a signal toward Earth. The signal is encoded using a pseudorandom noise (PRN) code that is unique to the satellite [30]. The code changes the phase of the carrier by 180° (Figure 1.1) and is relatively easy to isolate from the carrier. The signal's message is the navigational data required for a GPS receiver to determine the location of the source of the signal and the corrections to the clock and range measurements [51]. This data includes the coefficient values for: the clock correction polynomial; the ephemeris of the sending satellite that are accurate for 2-3 hours from time of transmission [51]; the satellite almanac, for satellite selection purposes; translation of GPS time to UTC time; and

corrections for ionospheric delay. The GPS satellite receives an update, on at least a daily basis, for the data it transmits in the message.

A GPS receiver is capable of receiving multiple signals simultaneously; its signal capacity is dependent on its construction. The receiver identifies the satellites through their signals' PRN codes and estimates the time of signal transmission based on the phase of the PRN code (Figure 1.2). Each signal is isolated and processed individually. The signals are decoded and demodulated to retrieve the data. The transmission time is corrected using the clock correction polynomial equation and the coefficients contained in the message [51]. The equation is:

$$\Delta t_{SV} = a_{f0} + a_{f1}(t - t_{oc}) + a_{f2}(t - t_{oc})^2 + \Delta t_R \quad (1.1)$$

where

$t = t_{SV} + \Delta t_{SV}$	= GPS time of transmission
$\Delta t_R = Fe\sqrt{A}\sin E_k = 2 \frac{\mathbf{R} \cdot \mathbf{V}}{c^2}$	= the relativistic correction
t_{SV}	= the PRN phase time at time of transmission
$a_{f0}, a_{f1},$ and a_{f2}	= the correction coefficients
t_{oc}	= the reference time for clock correction
$F = -2 \frac{\sqrt{\mu}}{c^2}$	= $-4.442807633 \times 10^{-10} \text{ s/m}^{1/2}$
μ	= Earth universal gravitational parameter
c	= speed of light
\mathbf{R}	= instantaneous position vector of satellite
\mathbf{V}	= instantaneous velocity vector of satellite
e	= eccentricity of satellite orbit
A	= semimajor axis of satellite orbit
and E_k	= eccentric anomaly of satellite orbit

The receiver also calculates the position of each satellite from known constants and the data it received. The GPS orbit is modeled as an elliptical orbit with correction terms

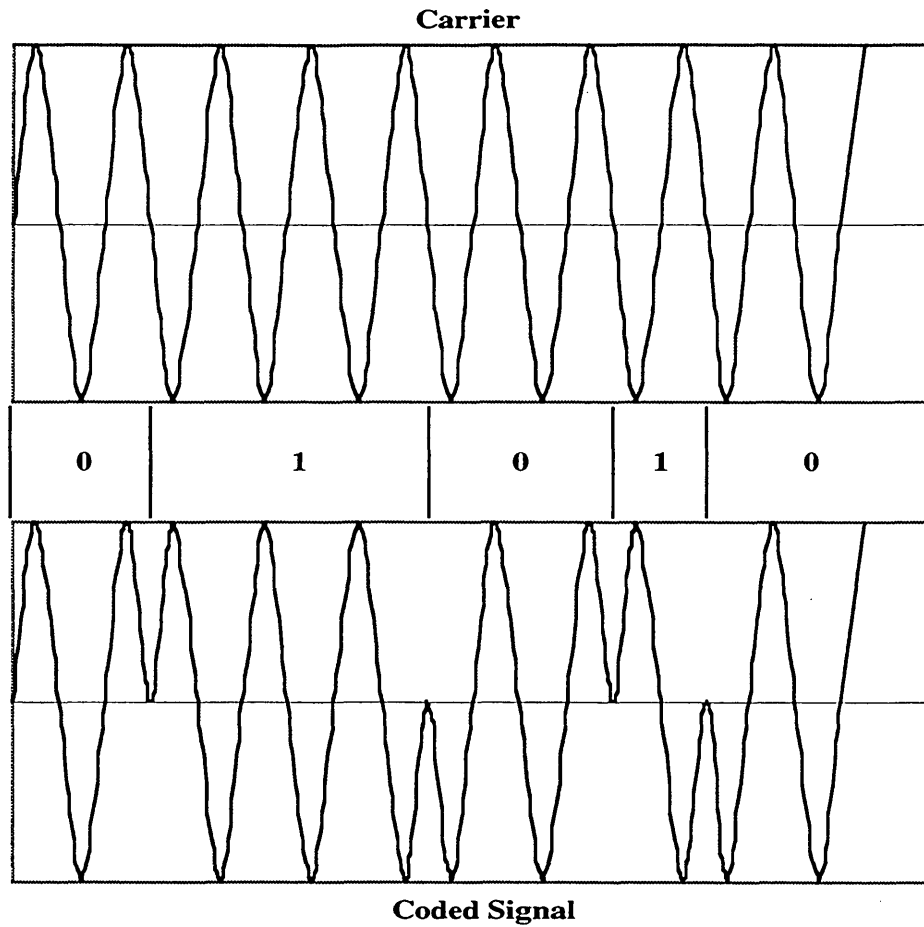


Figure 1.1. Pseudorandom Noise Coding

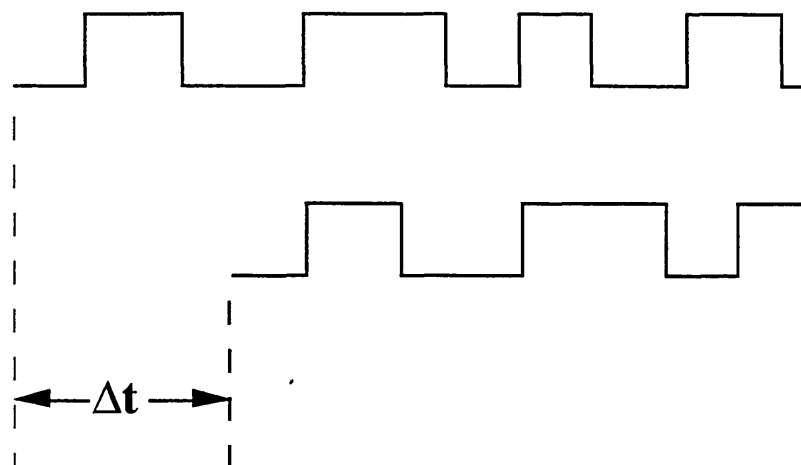


Figure 1.2 Phase Measurement of PRN Code

for perturbations related to radius, argument of latitude, inclination angle, and right ascension. The data in each GPS signal contains the coefficients for the modified Keplerian equation [51] for its orbit, and they consist of:

M_0	Mean anomaly in semicircles
Δn	Mean motion difference from computed value in semicircles
e	Eccentricity of the orbit
$(A)^{1/2}$	Square root of semimajor axis in $m^{1/2}$
$(\text{OMEGA})_0$	Longitude of ascending node in semicircles
i_0	Inclination angle in semicircles
ω	Argument of perigee in semicircles
OMEGADOT	Rate of right ascension in semicircles/second
IDOT	Rate of inclination change in semicircles/second
C_{uc}	Amplitude of argument of latitude cosine correction term in radians
C_{us}	Amplitude of argument of latitude sine correction term in radians
C_{rc}	Amplitude of orbit radius cosine correction term in meters
C_{rs}	Amplitude of orbit radius sine correction term in meters
C_{ic}	Amplitude of inclination angle cosine correction term in radians
C_{is}	Amplitude of inclination angle sine correction term in radians
and T_{0e}	Reference time for ephemeris in seconds (subscript 0 refers to this time)

Included with these coefficients is the issue of data-ephemeris (IODE) parameter. This parameter is used to verify data consistency within the GPS message. Table 8 in [51] presents the equations used to compute the GPS ephemeris from this data.

After estimating the satellites' positions and transmission times, the pseudoranges are calculated. The base value of a pseudorange is computed by multiplying the time of transit and the speed of light. This value may be corrected for ionospheric effects [36,51]. One equation that models ionospheric delay with a half cosine approximation is:

$$I(t) = \left\{ \begin{array}{l} F(E) \left[5 \times 10^{-9} + \text{AMP}(L) \cos \left[\frac{2\pi(t - 50,400)}{\text{PER}(L)} \right] \right] \Big| \text{day} \\ F(E) (5 \times 10^{-9}) \Big| \text{night} \end{array} \right\} \quad (1.2)$$

where

F(E)	= Obliquity factor as a function of elevation angle, E
AMP(L)	= Amplitude of half cosine
PER(L)	= Period of half cosine

The amplitude and period of the half cosine approximation are both cubic functions of the geodetic latitude of the user. The satellites include the coefficients for these cubic functions in their messages. The receiver can correct the pseudoranges for ionospheric delay if the equations are included in its program, but the manufacturer may have opted to ignore the ionosphere.

Once the pseudoranges of at least four satellites have been calculated, the receiver has all the information it needs to determine its position. There are a number of different algorithms for determining the position of a GPS receiver [1]. Some of these are filtering processes that require knowledge of the dynamics of the user. The most elementary algorithm for position determination is a least squares solution to the linearized observation equation:

$$\Delta \rho = G \Delta \mathbf{x} + \Delta \epsilon_p \quad (1.3)$$

where

$$\Delta \rho \equiv \begin{bmatrix} \Delta \rho_1 \\ \Delta \rho_2 \\ \vdots \\ \Delta \rho_n \end{bmatrix} \quad G \equiv \begin{bmatrix} -\hat{\mathbf{i}}_1^T & 1 \\ -\hat{\mathbf{i}}_2^T & 1 \\ \vdots & \vdots \\ -\hat{\mathbf{i}}_n^T & 1 \end{bmatrix} \quad \Delta \mathbf{x} \equiv \begin{bmatrix} \Delta \mathbf{r}_u \\ c \cdot \Delta b_u \end{bmatrix} \quad \Delta \epsilon_p \equiv \begin{bmatrix} \Delta \epsilon_{\rho 1} \\ \Delta \epsilon_{\rho 2} \\ \vdots \\ \Delta \epsilon_{\rho n} \end{bmatrix}$$

and

$$\hat{\mathbf{i}}_i \equiv \frac{\mathbf{r}_i - \hat{\mathbf{r}}_u}{|\mathbf{r}_i - \hat{\mathbf{r}}_u|} \quad \Delta \mathbf{r} \equiv \hat{\mathbf{r}}_u - \mathbf{r}_u \quad \Delta b \equiv \hat{b}_u - b_u \quad \Delta \epsilon_{\rho i} \equiv \hat{\epsilon}_{\rho i} - \epsilon_{\rho i}$$

The caret marks indicate the predicted values for satellite position, clock bias, and estimated error contributions. The values to calculate are $\Delta \mathbf{x}$, the vector of the change in

position and time bias from the a priori values. The change in estimated errors vector, $\Delta\epsilon_p$, is assumed to be zero mean and is, therefore, neglected from the least squares solution. The change in pseudorange, $\Delta\rho$, is difference between the predicted and measured pseudorange. G is the geometry matrix of the satellite positions, which consists of the line-of-sight unit vectors ($\hat{\mathbf{1}}$'s) between the satellites and the predicted position of the user and $\mathbf{1}$'s for the clock bias term. Usually, G is not a square matrix, so it does not have a true inverse. But the Moore-Penrose generalized inverse can be used to solve this equation.

The accuracy of this algorithm is dependent on the quality of the measurements and the geometry of the satellite positions. The measurement quality depends on the variance of the measurement error; the variance is influenced by the error conditions and can vary from 0.3 to 30 meters [1]. The satellite geometry can cause dilution of precision (DOP), which will be described in the next section.

1.1.3 GPS Related Errors

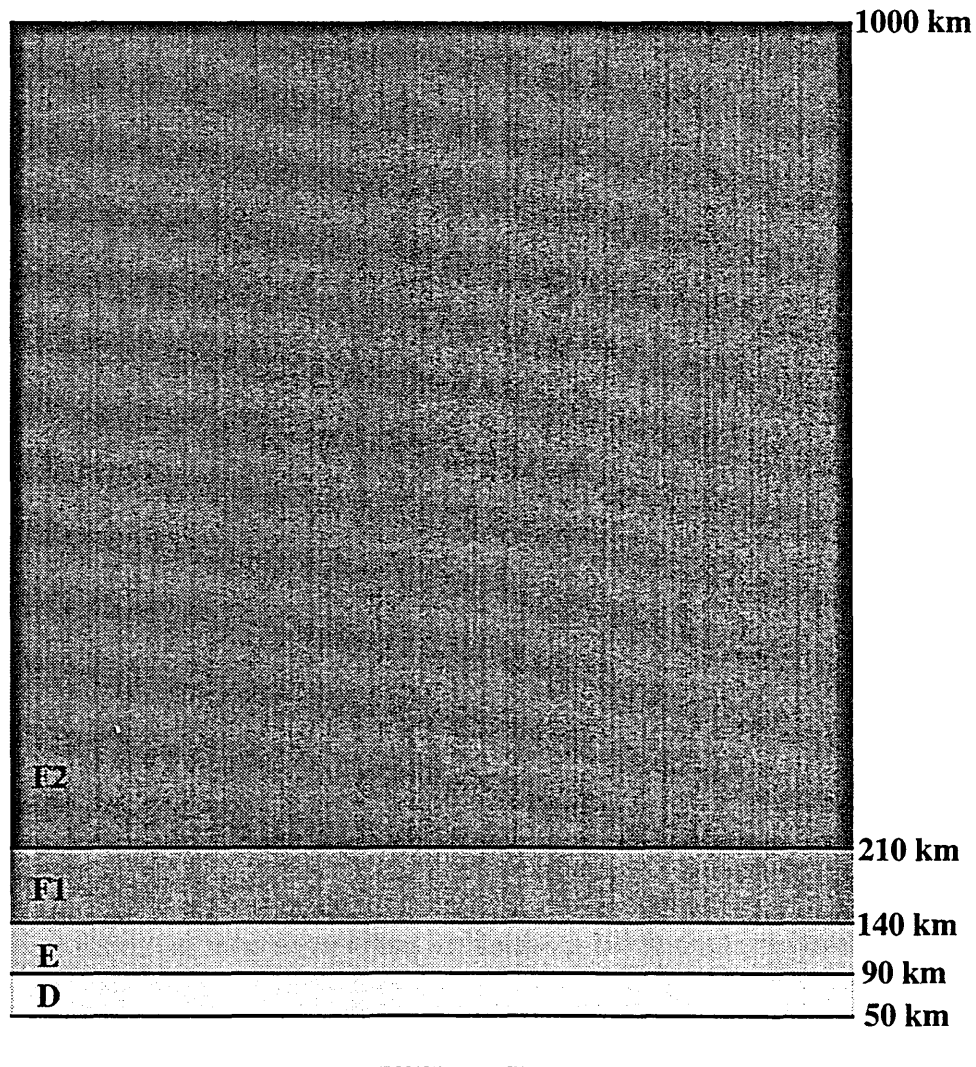
GPS is not a perfect system. There are errors associated with the operation of the system. These errors can be grouped in three classifications: environmental, inherent, and intentional. These error classifications and their members are described below.

1.1.3.1 Environmental errors

The environmental errors are the delays caused when the signals pass through different layers of the atmosphere. The layers of the atmosphere that are the main contributors to these errors are the ionosphere and the troposphere, in that order [47]. The errors caused by both of these layers of the atmosphere are intensified for lower elevation angles, because the signals have a longer path through these regions to reach their destination.

The ionosphere is the layer of the atmosphere that is populated by higher concentrations of free electrons and ions [36]. The ionosphere is broken down into four regions (D, E, F1, and F2), based on increasing altitude and electron density (Figure 1.3); the highest region's electron density rises to a maximum value, then its density decreases with altitude. The lower three regions are strongly dependent on ionizing radiation from

Protonosphere



Earth

Figure 1.3 Layers of the Ionosphere

the sun and do not exist at night while F2 decreases in density and raises its altitude of maximum density during the night. Each region has a different effect on radio wave propagation. Region D has no impact on GPS frequencies; region E has negligible effects on these frequencies. The time delay caused by region F1, with the small contribution from E, is approximately 10% of the entire ionospheric delay experienced by GPS. The F2 region, which is primarily composed of atomic oxygen and its ions, is responsible for most of the radio propagation problems at GPS frequencies. Above region F2 is the protonosphere, a region of ionized hydrogen that extends to approximately GPS orbital height. It has an unknown electron density and can contribute 10% to the time delay during the day or 50% at night. These effects cause errors in pseudorange that vary from a few meters to several tens of meters. The pseudorange can be corrected for this time delay from the coefficients sent in the GPS message, if the receiver's program includes ionospheric correction.

The troposphere is the collection of dry gases and water vapor that exist close to the Earth's surface [52]. This layer of the atmosphere can affect GPS signals through attenuation, scintillation, and signal delay. These effects are caused mainly by the nitrogen and oxygen in the atmosphere, but the water vapor has a noticeable effect on the signal delay. The first two effects change the strength of the signal, and they are relatively minor unless the elevation angle is less than 5° . The signal delay is caused by refraction of the signal by the atmosphere. This refraction is dependent on the local temperature, pressure, and humidity. This effect changes the speed of the radio wave from its speed in vacuum, and extends the transit time of the signal. This delay corresponds to a pseudorange error of 2.3 - 3.1 meters [52]. The dry atmosphere portion of this effect can be modeled and accounts for a majority of the error, but this model for error correction may not be included in the GPS receiver program.

1.1.3.2 Inherent errors

Since perfection is unattainable, every system that has ever been built has some inherent errors in its operation. Some of these errors are due to design constraints while others are natural limitations of physics. For the GPS system, these errors occur in five aspects of operation: ephemeris data, satellite clock, multipath reception, receiver operation, and geometric dilution of precision. Each of these sources contributes a small amount of error.

The ephemeris data that is transmitted by the GPS satellites is produced from a least squares fit over a 4-6 hour period of ephemeris residuals described in Section 1.1.1 [23]. This ephemeris compression process was devised in order to 1) limit the amount of data to be transmitted from the GPS Master Control Station to the GPS satellites, 2) limit the amount of data to be transmitted by the GPS satellites to the user, and 3) simplify the ephemeris calculations performed in the user's GPS receiver. A satellite could not send enough information in its message to include the parameters required for the precise position, but it could handle the reduced number of parameters for the fit of the precise position. The parameters for two weeks worth of fits are sent to the satellites in their regular updates; the two-week period was selected for emergency situations when the satellites cannot be contacted. The fits are less accurate, but the error stays reasonably close to the precise solution.

The error in the satellite clock is dependent on the quality of the clock. All the satellites use atomic clocks that have an accuracy rating of approximately 1 part per 10^{13} . These clocks are the most precise tools for time measurement. At this accuracy, the error accumulated in a day's time (86400 seconds) would be 8.64×10^{-9} seconds or about 2.6 meters. In 1984, the error was reported to be 4.1 meters for 24-hour predictions. The clocks are synchronized daily or more frequently, which prevents the further accumulation of error. Because the standard deviation's growth is reported to be quadratic with time, the expected average error is on the order of 1-2 meters for 12-hour updates [47].

Multipath reception arises from the fact that electromagnetic waves and beams reflect off surfaces [4]. A reflected GPS signal will continue to travel and may be received by a GPS receiver that has already received the same, unreflected signal. This occurrence distorts the signal and degrades the positioning accuracy. The effects of multipath can be mitigated in a number of ways. First, the receiver's antenna can be mounted on an rf-absorbing surface; this option is not really feasible for a mobile receiver. Another option is to polarize the antenna to receive right-hand circularly polarized signals; this technique does not block signals of other polarization, but it does reduce their strength. A third way to deal with multipath is to design the antenna's gain pattern to have low gain in the directions that are most likely to produce reflected signals. Other methods are being devised to cope with this problem. In extreme cases, this phenomenon can contribute 15 meters of ranging error. The effects of multipath reception cannot be eliminated, but they can be minimized.

Receivers have seen a tremendous improvement from the time of their original design, but they still have limitations. Their clocks are high quality, but not as good as the satellite clocks. The microprocessors have the ability to produce the desired precision for positioning, but the output is only as good as the software onboard. The receiver can also introduce range errors due to thermal noise. These limitations do not contribute much to the total error, only 0.5 meters in bias and 0.2 meters in noise [47].

The geometry of the GPS satellites that are used for determining the user's position can have a greater impact on the accuracy of the solution than the ranging errors [1,47]. A poor selection of satellites can result in a dilution of the accuracy. This quantity is measured through a number of different dilution factors, of which GDOP and PDOP are the most common. PDOP is the position dilution of precision factor and it measures the dilution of the position accuracy. GDOP is the geometric dilution of precision and it incorporates the dilution of the position and the clock bias. The calculation of GDOP and PDOP begins with Equation 1.3. The solution of this equation for Δx is:

$$\Delta \hat{\mathbf{x}} \equiv (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \Delta \rho \quad (1.4)$$

where $(\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T$ is the equivalent of the Moore-Penrose generalized inverse that was mentioned in the discussion of Equation 1.3. To get the error covariance of the solution, the expectation operator E can be applied to the product of the solution and its transpose as follows:

$$\begin{aligned} E[\Delta \hat{\mathbf{x}} \quad \Delta \hat{\mathbf{x}}^T] &= E\left[(\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \Delta \rho \quad \Delta \rho^T \mathbf{G} (\mathbf{G}^T \mathbf{G})^{-1} \right] \\ &= (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T E[\Delta \rho \quad \Delta \rho^T] \mathbf{G} (\mathbf{G}^T \mathbf{G})^{-1} \end{aligned} \quad (1.5)$$

The geometry matrix G has no random components, so all instances of G can be move out of the expectation operation. Under the assumption that the range errors are uncorrelated and have equal variance, the expectation operation $E[\Delta \rho \quad \Delta \rho^T]$ becomes $\sigma^2 \mathbf{I}$ and Equation 1.5 reduces to:

$$E[\Delta \hat{\mathbf{x}} \quad \Delta \hat{\mathbf{x}}^T] = \sigma^2 (\mathbf{G}^T \mathbf{G})^{-1} \quad (1.6)$$

The diagonal of this covariance matrix corresponds to the variances in the components of the solution. GDOP is the square root of the trace of $(\mathbf{G}^T \mathbf{G})^{-1}$, and PDOP is the square root of the sum of the first three diagonal elements. The error magnitudes are estimated as the products of the dilution factor and σ . For example, the total position error magnitude is estimated to be $\sigma \times \text{PDOP}$. Because of the assumption that the measurement errors are uncorrelated and are statistically alike, this value is only an approximation. Dilution can be reduced by using more satellites. This tactic improves the satellite geometry for position estimates.

1.1.3.3 Intentional errors

Because GPS is a military asset, the U.S. Government has restricted access to the more precise navigational data. To accomplish this, GPS has been seeded with intentional errors, which are referred to as Selective Availability (SA) [47,54]. These errors are built into the data that is transmitted by the satellites. The two potential recipients of these errors are the ephemeris data and the satellite clock.

The ephemeris data may be modified from the best available predicted values. The errors would be embedded in the coefficients that the satellite transmits for the purpose of calculating the satellite's position. By adding more errors to the satellite positions, the navigational algorithm produces a solution that is less accurate. Ephemeris errors change slowly with time, so the effectiveness of SA on the ephemeris is limited [47].

The satellite clocks are definitely being altered for SA. The clocks are being dithered to vary the range errors (Figure 1.4). The user position computed from these modified ranges wanders around the true location, and the position errors become uncorrelated in a matter of minutes. There is no correlation between the errors produced by different satellites. The only way to determine the behavior of SA is through direct observation. Models can be developed that are statistically similar, but there is no model that reproduces this dither exactly. Therefore, these models cannot be used by "unauthorized" users to remove SA from the satellite clock.

1.2 Eccentric Orbits and GPS

Why is GPS performance questionable for elliptical orbits? As stated earlier, GPS receivers have been flown mainly on satellites with nearly circular orbits at LEO altitudes [17]. These orbits experience very little variation in their altitudes and velocities. In

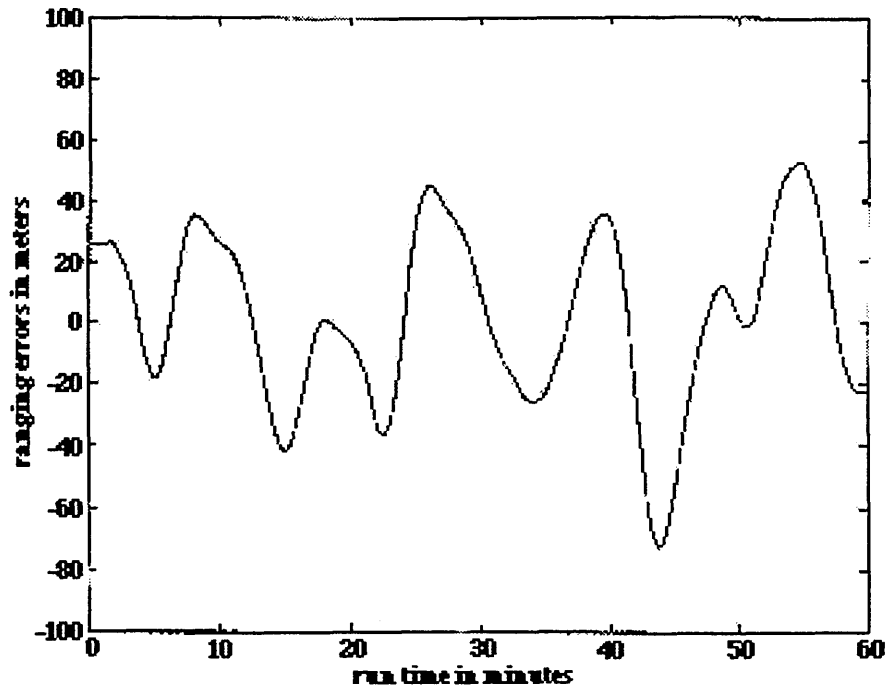


Figure 1.4 Ranging Errors from Satellite Clock Dither [54]

eccentric orbits, the altitudes and velocities are continuously changing; the more eccentric the orbit, the greater the range of variation. These variations could have a significant impact on GPS performance. For instance, as the altitude of the receiving satellite increases, visibility between the receiver and GPS satellites may become a problem. In addition, if a downward facing antenna is used at these higher altitudes, the signals received by this antenna may be affected by ionospheric interference to a much higher extent than is experienced by upward facing antennas. These are some of the problems that can seriously impact the performance of GPS.

1.2.1 Visibility Issues

As a satellite's altitude rises in an elliptical orbit, visibility between the GPS satellites and the receiving satellite can become a problem. This problem arises from the design of the GPS transmission array and the limitation on the GPS receiver's antenna.

The GPS transmission array was designed to transmit a signal in a beam with a 21.3° half-beam angle from the nadir direction [50]. This signal is transmitted at a distance of 26,561 kilometers from the center of the Earth. With this beam configuration and the placement of the satellites, the GPS constellation provides full coverage of an invisible sphere centered at the Earth with a radius of 9650 kilometers. This coverage is more than adequate for performance at any Earth location, which is why the system was designed. If the previously mentioned sphere were increased in size beyond the radius of 9650 kilometers, the beam from each individual satellite no longer covers the entire surface of the larger sphere it faces. As the radius of the sphere increases, the surface coverage on this sphere by an individual beam reduces in size. At each location on this invisible sphere, the number of GPS satellite contacts begins to drop and gaps begin to develop in the four-way coverage.

Figure 1.5 shows an example of the gaps in coverage from the target's point of view. In other words, the figure looks at the target and asks, "Where can a GPS satellite be and still be able to see the target?" The pictures in this figure are a planar cut through the target and the center of the Earth. Rotating these pictures about the position vector of the target produces the sphere of visibility. Figure 1.5A shows a surface target. This target can be seen by any GPS satellite that is above the dotted line. Figure 1.5B shows a target satellite at an altitude of 6,000 kilometers. Since the geometry is symmetric about the target's position vector, only one half of the plane is presented. In this figure, the dotted lines indicate the edges of the GPS transmission cone or the line of sight between the target

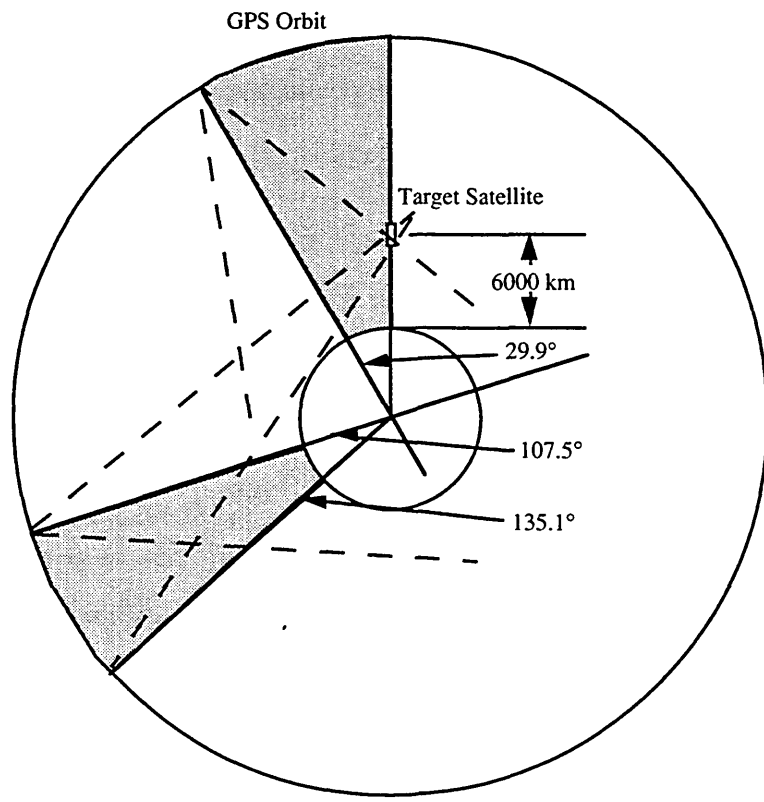
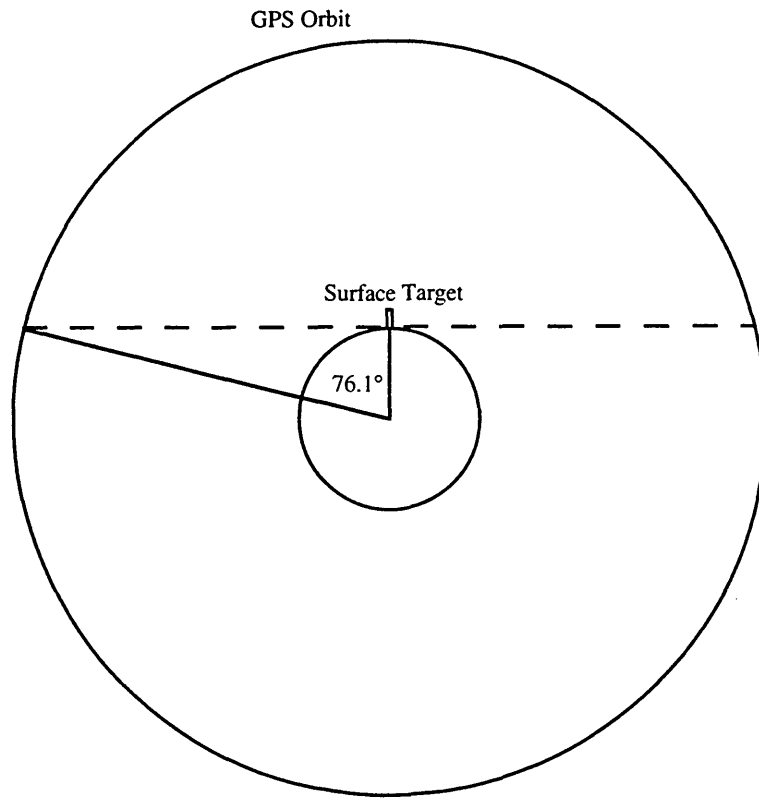


Figure 1.5 Examples of GPS Visibility

and GPS satellites at the edge of the Earth's shadow. The GPS satellite boundary positions are connected to the Earth's center to produce the shaded areas. All angles presented in Figure 1.5B are measured from the target satellite's position vector. After rotating the figure, the shaded regions become a circle above the target satellite and a ring on the far side of the Earth. Any GPS satellite on the surface of the shaded regions in this picture can see the target satellite.

The visibility regions described above vary with altitude. Figure 1.6 was created by repeating the examination of a target satellite for other altitudes. It depicts how these regions of visibility vary, starting from an altitude of 3270 kilometers. The sizes of the regions are scaled as a percentage of the surface area of a sphere with the same radius as the GPS orbit. As the target rises, the shaded regions become smaller. For the upper region, the circle's radius decreases and approaches zero as the target gets closer to GPS altitude. The ring's radii each become smaller for the lower region, but the region of visibility approaches 2% of the GPS sphere as altitude approaches infinity. There is no overlap between these regions above 3270 kilometers altitude, so the total visible surface area is the sum of the areas of the two regions.

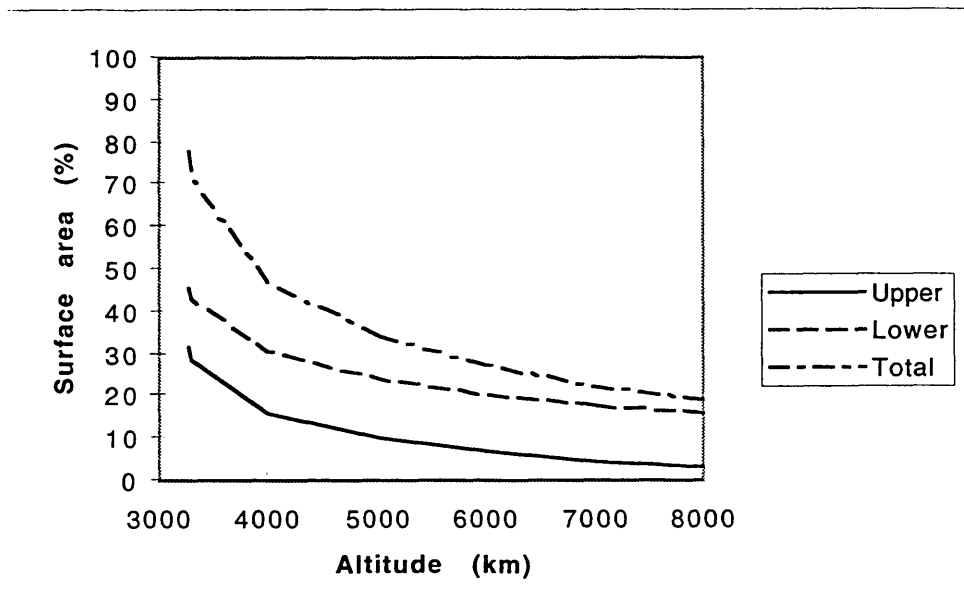


Figure 1.6 Visible Surface Area

Reception is not constrained solely by the location of the transmission's source. The target's reception is also dependent on its receiving antenna's orientation. For example, the target in Figure 1.5A can receive a signal from any GPS satellite in the indicated region, provided that its receiver is equipped with a hemispherical antenna and its view is not blocked by terrain. In Figure 1.5B, the target has three choices for antenna orientation. If the antenna faces nadir, the only signals that the target could receive are those transmitted by GPS satellites on the surface of the ring. For a zenith facing antenna, the same condition holds for GPS satellites on the surface of the circle. The third alternative for antenna positioning is to face it normal to the radial direction. This alternative results in the ability to receive signals from a hemisphere of the sum of the two previous options.

As stated earlier, the visibility problem is also affected by the limitations of the receiving antenna. The discussion of Figure 1.5 assumed that the antenna being used was hemispherical. However, there are no space-rated GPS receiver antennas that cover an entire hemisphere. The best antenna in this class can receive signal up to 70° from the antenna's axis, or 140° across the diameter of the reception cone. In Figure 1.5B, this decrease in reception angle does not impact the first two options, but it affects the third. If the antenna is left normal to the radial direction, the target satellite would lose 20° from the edges of the hemisphere. This effect can be altered by reorienting the antenna. The final orientation of the antenna depends on which of the two regions has more value for the application.

The discussion of antenna orientation to this point has looked at the pictures in Figure 1.5 as static cases in altitude, but the altitude of an elliptical orbit is constantly changing. As the target satellite approaches perigee, a nadir facing antenna would be useless; the Earth would block almost the entire cone of reception. As it approaches apogee, a zenith facing antenna would not be blocked by the Earth, but the circle of

visibility may become so small that the receiver cannot obtain four signals. The third alternative cannot provide the maximum number of available satellites at any time as the other two options can, but it may provide a more balanced number of satellites than either of the others.

1.2.2 Ionospheric Interference

When operating a GPS receiver on the Earth's surface or in LEO, the incoming GPS signals are delayed by ionospheric interference. If a receiver is operated above the ionosphere, the effects of the ionosphere could be increased or decreased, depending on antenna orientation. An antenna pointing upward will see none of the ionosphere, but a downward-facing antenna can look through the full depth of the ionosphere twice.

For a ground-based GPS receiver, the ionosphere slows the signal as it descends. The delay associated with the ionosphere is dependent on the electron density and the length of the path through the ionosphere. The shortest path through the ionosphere is in the radial direction from the Earth's center and through the receiver's position, and the longest path is tangent to the Earth's surface at the receiver's location (Figure 1.7). These paths have include ionospheric path lengths of 790 km and 2059 km, respectively.

The worst case of ionospheric delay is experienced when the GPS signal grazes the inner edge of the ionosphere (Figure 1.8). This path through the ionosphere has a length of 6643 km, better than three times longer than the worst path for a ground based receiver. The delay resulting from this pass can have an effect of up to 400 m in the pseudorange for that one signal. This error, compounded with the other known errors, may degrade the performance of GPS to the point where it is unsuitable for use in orbit determination.

There are three ways to deal with the higher level of ionospheric effect associated with looking past the Earth: ignore the effects and live with the degradation; compute a correction for the ionospheric delay; or dismiss GPS signals that have made the long

journey through the ionosphere. The first option is the simplest, but may result in an unacceptably high error. The second option is the most complex and computationally

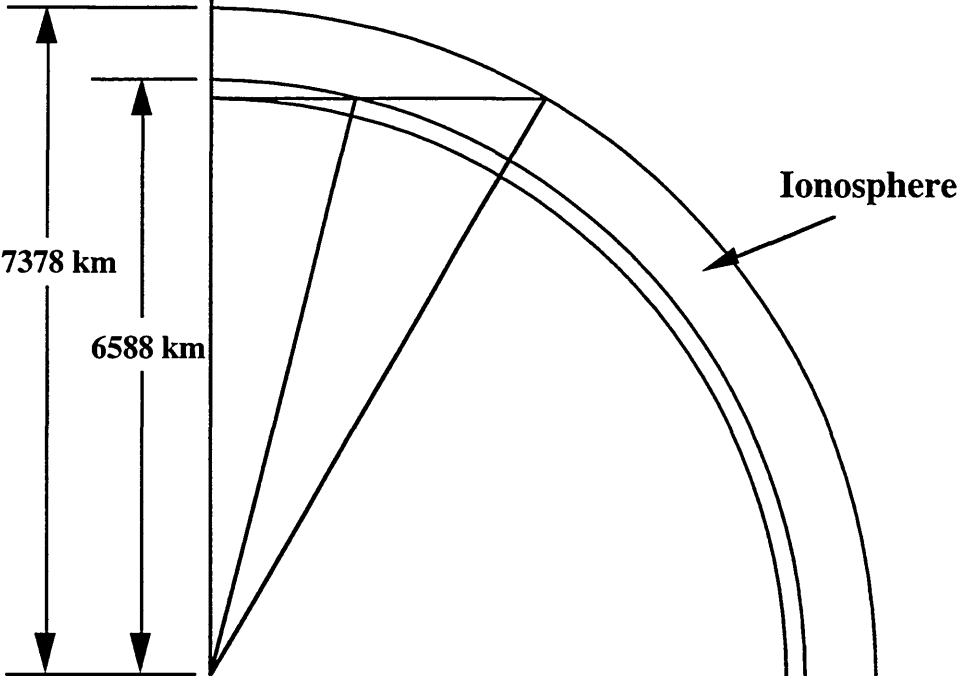


Figure 1.7 Paths through Ionosphere (ground-based)

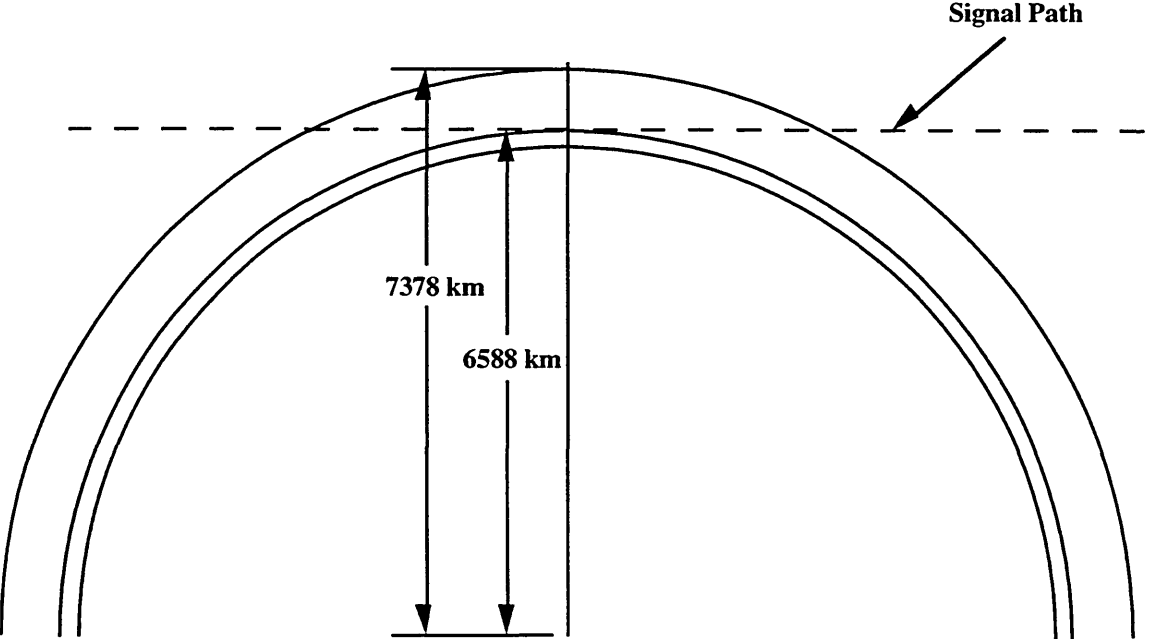


Figure 1.8 Worst Case Ionospheric Delay

demanding. The third option removes the increased error and only requires the method to identify the signals that will not be used. The first and third options are investigated in this thesis.

1.3 GPS and Orbit Determination

The concept of using GPS for orbit determination is not new. In fact, there have been several suggestions on how to use GPS data in orbit determination processes. Some methods use GPS data as a portion of their input, and some require the assistance of ground stations for the corrections used in differential GPS (DGPS) or for data processing. Some of these methods are presented here, followed by a description of the orbit determination method that was used in this project.

1.3.1 Review of Past Concepts

In 1990, Stephen Lichten and Jeffrey Estafan proposed a method of orbit determination for elliptical HEO satellites [38]. This method was specifically targeted for use on spaceborne very long baseline interferometry (VLBI) stations that were scheduled to launch in the mid to late 90's; in particular, the stations of interest were the Japanese VLBI Space Observatory Program (VSOP) MUSES-B and the International VLBI Satellite (IVS). This method uses a combination of two-way Doppler tracking, GPS tracking from the ground, and GPS tracking from the satellite. The GPS ground tracking data is used to calibrate the tropospheric delay experienced by the two-way Doppler signals. The GPS satellite tracking data and the calibrated two-way Doppler data are processed using a Kalman filter.

The VSOP satellite, which was renamed HALCA, was launched on February 12, 1997. To date, most of the orbit determination for HALCA has been handled by the Jet

Propulsion Laboratory (JPL). The orbital solutions have been produced using just two-way Doppler. Due to recent HALCA trouble, the testing of the on-board GPS receiver has not started yet and no GPS data is available [3]. As a result, this method has not been tested.

In 1995, R. Fennessey, et al. conducted an evaluation to see if GPS navigation solutions could be used as the input for orbit determination for LEO satellites [17]. This evaluation revealed that an orbit determination technique using GPS navigation solutions and the TRACE computer program [67] produced as good an orbit, or better, as one determined from the Space Ground Link System (SGLS) [68] range, azimuth, elevation, and range-rate (RAER), which had been the standard source of data at the time. It also indicated that, for higher accuracy, additional software modifications would be necessary to implement an orbit determination capability using differential GPS, and that additional studies should be conducted to determine the costs and performance of orbit determination using GPS.

In his thesis in 1996, Scott Carter also investigated the possibility of using GPS navigation solutions as input for orbit determination, but the approach was slightly different [8]. With a minor amount of preprocessing, the navigation solutions were, again, used directly in the orbit determination process. But the orbit determination was performed using the Draper version of the Goddard Trajectory Determination System (GTDS) software, which could perform the process using either precise mean or osculating elements.

To achieve the desired accuracy, modifications to the GTDS were required. The first change was the inclusion of two previous improvements to GTDS that had been developed separately and had not been merged into a common version of GTDS. These improvements were: changes associated with the Radarsat project, and the ability to increase the size of the gravity potential field model to degree and order of 50. On top of

these modifications, a model for lunar and solar solid Earth tides and a more modern inertial reference frame were introduced into GTDS.

After these modifications were completed, the GTDS was able to take in the GPS navigation solutions and perform the orbit determination process. The output produced from the orbit determination of TOPEX indicated a fit accuracy of 9.5-10.5 meters over five days, and predictions with an error growth rate of 10-15 meters per day over the following two days. This result showed that GPS solutions could be used to accurately determine the orbit of a LEO satellite without the high complexity and cost associated with the differential GPS process.

In an AIAA paper at the 1997 Astrodynamics Specialist Conference, S. Lichten, et al. presented two methods for orbit determination of higher altitude satellites using GPS as input [39]. These methods were devised to provide alternatives to the traditional zenith-facing GPS receivers, which have a more difficult time acquiring enough GPS signals as their altitude increases. The main focus of this paper is on geosynchronous satellites.

The first method, known as GPS-enhanced tracking (GET), uses a satellite-mounted beacon that sends a signal similar to those sent from GPS satellites. This beacon is received, along with GPS signals, by ground-based GPS receivers. The data that was simultaneously acquired by these receivers is used to determine the positions of the GPS satellites and, with some modifications, the remaining satellite in question. This method has been tested on two Tracking and Data Relay Satellites (TDRS).

The second method employs a downward looking GPS receiver instead of the traditional upward looking. For geosynchronous satellites, nadir-facing GPS receivers would only receive signals from 1 or 2 GPS satellites. This amount is inadequate for the standard position calculations. By using a more precise model of orbital motion at geosynchronous altitudes, a solution is calculable using fewer GPS satellites. The performance of this method has not been tested yet, but an experiment using this technique is due to be launched in 1999.

In the MOMS/Priroda Camera Experiment, the mission requirements forced the implementation of a high accuracy orbit determination process [37]. Standard GPS positions provided adequate input data to meet most of the requirements, but the high resolution graphics processing required more accurate inputs for the orbit determination. To provide this higher accuracy, a differential GPS (DGPS) method was implemented using a ground-based GPS receiver that received GPS signals from the same satellites simultaneously. With the aid of precise GPS ephemerides and clock offsets, the data from the ground receiver and the receiver on Priroda could be processed to produce the required accuracy.

In 1998, Peter Melvin presented a refinement of GPS orbit determination for LEO satellites [42]. It used pseudorange corrections and double differencing between a LEO satellite and one or more ground stations to produce smaller residuals in the least-squares solution to the orbit. The solution was processed in a Kalman filter and generated positions that were consistently accurate to within ten meters.

1.3.2 Current Project

The work presented in this thesis builds on the work of Scott Carter [8]. However, Carter had empirical data from several spacecraft (TOPEX, TAOS, and EUVE) to test his processes against. There is no GPS data for MEO satellites in eccentric orbits, so the data must be simulated. The simulation designed in this project includes the GPS errors that contribute the most to the navigation solution errors. The output of the simulation is ECEF position only. As in Carter's work, only the batch least-squares method of data processing is investigated, but the software designed for this project is flexible enough to allow for filtering or other methods.

This project examines several antenna facings for the GPS receiver, including nadir facing. It also looks at the benefits of using more than one GPS antenna. Three antenna

configurations are investigated to characterize their performance in different orbital situations.

After evaluating the performance of the orbit determination process for these orbits using perfect force modeling, an investigation is conducted on the effects of force mismodeling on this performance. The forces that are mismodeled and the level of mismodeling are selected based on the characteristics of the orbit being determined.

1.4 Thesis Overview

The purpose of this chapter was to present the rationale behind the question posed in this thesis, and to provide information on the problems associated with the pursuit of its answer. This chapter also describes previous means of orbit determination using GPS data and how the proposed process differs. The remainder of this document will provide a description of the approach taken to address the question, a history of the tools used to facilitate its completion, a detailed explanation of the simulation developed for this task, and an analysis of the data collected from that simulation.

Chapter 2 presents the concept of parallel processing, and explains how it has been used in previous applications for computing the motion of constellations of satellites and how it is used in this project. Chapter 3 describes the navigation solution simulation and data generation processes, and provides a detailed description of the algorithms used in the simulation. Chapter 4 explains the simulation architecture and its unique features. Chapter 5 displays the results of the data analysis and explains the observed performance. Chapter 6 presents the conclusions and topics for future research.

Chapter 2

The Computation of Satellite Constellation Motion via Parallel Processing

The planning and operation of multi-satellite constellations results in more stringent requirements for orbit propagation. One approach to solving this problem is the implementation of parallel processing.

This chapter discusses the principle of parallel processing. It describes the two methods of parallel processing that have been employed by Draper Laboratory and some past implementations of these methods. Finally, it explains how parallel processing was applied to the current problem.

2.1 Parallel Processing

In the past decade, scientists and engineers have begun to perform more tests through the use of simulations. These simulations can accurately reproduce phenomena that cannot be predicted correctly through analytical theory and are prohibitively expensive or hazardous for laboratory experimentation. Traditionally, these simulations have been operated on supercomputers, but the demand for increased capacity in these machines has driven up their cost dramatically.

An alternative to the use of single-processor supercomputers is parallel processing [28]. This concept started as an experiment, but it has seen great acceptance recently. The concept behind parallel processing is to use a group of processors working simultaneously on different parts of the same problem instead of one large processor performing all the calculations sequentially. Teamwork between processors working in parallel can result in

more tasks being accomplished than the sum of the tasks that each processor could complete individually. And as mentioned before, larger single-processor computers increase in cost, but at a higher rate than they increase in power. Computer manufacturers realized the power of parallel processing and began to make parallel supercomputers that provide a large number of processors within a single machine. But the price of these parallel supercomputers is still prohibitively high. So rather than purchasing a supercomputer, many sites have implemented programming that allows them to use their networks of workstations as a parallel computer. The success and cost effectiveness of this practice have resulted in the purchase of networks of workstations that are dedicated to the parallel jobs that once ran on a supercomputer.

But there are obstacles to parallel processing in the areas of hardware, algorithms, and software. The main hardware problem is communication; the higher-speed computers need a communication environment that can keep up with them. Advances in this area have developed a better balance between computation and communication in parallel computers, and should soon be extended to workstations as well. The difficulty with algorithms comes from trying to identify the levels of independence of individual tasks from the rest of simulation, based on physics, mathematics, and programming ingenuity. Software is the biggest obstacle to parallel processing. Part of the trouble with software is that automatically parallelizing compilers do not work well in all instances; the best results in parallel software still come from programs that provide their own parallel algorithms [28]. Other problems with parallel software come from the compromises between expressiveness, efficiency, and portability in the mechanism for communicating the algorithm to the computer.

There are a number of different computational models for parallel processing. These models represent the operation of the program, but do not limit the implementation to a specific machine or programming language. All of these models could be implemented on any hardware through the use of the correct software. Some of these models are:

Data parallelism
Shared memory
Message passing
Remote memory operations

Data parallelism is based on one concept; the parallelism is limited to the data. This model processes multiple sets of data through the same sequential program. In the shared memory model, all processors have access to one shared address space and memory operations are controlled through a locking system; this means that an individual memory location is locked by the processor accessing it to prevent other processors from changing its value. For message passing, all processors have only a local address space, but they can communicate with the other processors and relay information to them; to accomplish this communication, each processor must perform a send or receive operation. The controlled access employed by the shared memory model and the cooperative local memory transfer of the message passing model are the two extremes, and the remote memory operation model resides somewhere between them; it allows one processor to explicitly access the memory location of another without the other's cooperation. These models are not the only ones that exist, but they are representative of the major concepts in parallel processing.

The requirement for cooperative communication in the message passing model provides a level of memory control that is very desirable. For this reason, message passing has been the selected parallel model for a number of different constellation applications. The next two sections present descriptions of two approaches to the message passing model: the Parallel Virtual Machine (PVM), and the Message Passing Interface (MPI).

2.1.1 Parallel Virtual Machine (PVM)

PVM is one approach to the message passing model for parallel processing [24]. It is a software framework that allows a collection of computers to be viewed as a single machine;

these computers need not be of the same design. It was designed to operate primarily on Unix operating systems, but it is capable of running on systems with similar functionality. Its construction is based on the following concepts [24]:

- User-configured host pool: The user can define which machines will be used to execute a particular program. The pool of machines may be modified during program operation.
- Translucent access to hardware: Programs may view the hardware as a collection of similar processors, or they can take advantage of the unique characteristics of each processor for enhanced performance.
- Process-based computation: Work is broken down into tasks that operate independently and alternate between communication and computation. A single processor is not limited to execution of a single task.
- Explicit message-passing model: Tasks communicate cooperatively by sending and receiving messages explicitly.
- Heterogeneity support: PVM supports the interaction between heterogeneous machines and networks. During communications, PVM performs data conversion for message compatibility.
- Multiprocessor support: To make best use of the hardware, PVM utilizes the internal message-passing structure in multiprocessor machines.

PVM consists of two parts: the library and the daemon. The library contains the functional routines required for the interactions between the different processors. These routines are user-callable and perform message-passing functions, spawn new processes, control the synchronization of the tasks, and alter the configuration of the virtual machine. The daemon resides in each processor that is a component of the virtual machine. It is basically the path of communication between an individual processor and the rest of the virtual machine.

PVM has been used in past applications for parallel processing of satellite constellations, but it was not used during this project. It was the first widely released software package for parallel processing with the message passing model and is used in applications around the world. However, PVM's flexible program structure reduces the

portability of its applications. These applications are usually customized to the environments in which they were developed, taking advantage of the unique functionality of these environments. With a more controlled program structure, applications would not rely on the functions of an individual environment and would be more portable. The Message Passing Interface (MPI) appeared to employ some of that control.

2.1.2 Message Passing Interface (MPI)

MPI is another example of the message passing model for parallel processing [28]. Its initial development was completed over a two-year period, from April of 1992 to May of 1994. The development began as a result of a workshop on message passing standards that was sponsored by the Center for Research in Parallel Computation. This workshop spawned the realization that many people had good ideas for implementing the message passing model, and that people were willing to coordinate efforts to define a standard. A committee was formed to undertake this task at the Supercomputing '92 conference [69].

The goals of this committee were:

- Define a portable standard for message passing. This standard would not be official, like the ANSI standard, but it would be attractive to implementors and users.
- Maintain an open-door policy. The development process would be open to anyone to offer ideas, participate in discussions, attend meetings, or monitor e-mail.
- Complete the standard in one year.

The participants in this development effort came from computer vendors, software producers, and academic institutions. The MPI committee met every six weeks until the standard was completed.

Since MPI is an example of the message passing model, it has all the characteristic of such a model. It has the ability to work well with both the parallel supercomputers and

the workstation networks. It is a useful model for expressing parallel algorithms and provides the control that is lacking in data-parallel and compiler-based models. The communication process has an anthropomorphic flavor to it, which can cause some people to visualize workers performing the tasks and talking to each other; this visualization can be very helpful in the development of a parallel algorithm [28]. The controls on communication in the message passing model facilitate the detection of faulty reads and writes that cause unexpected overwrites of data, a major cause of errors in these programs. Message passing also provides high performance by associating specific data with processes and allowing the memory-management hardware to operate at full capacity.

MPI is still a compromise between expressiveness, efficiency, and portability, but its development was influenced by the knowledge of these goals. Because it was designed to be the standard for message passing, MPI is portable. There is nothing intrinsic to the design that causes reduced efficiency. And as was mentioned in the advantages of the message passing model, it is quite convenient for expressing parallel algorithms. For a compromise, it has struck an excellent balance.

MPI is not a wholly new construction, but a compilation of best features from preexisting message passing systems that have been modified when necessary and standardized. It is not a language, but a library where all of the subroutines and functions for its operation are stored. When a program that uses MPI is compiled, it is linked to the MPI library. MPI's features are limited to those that address its computational model.

The MPI standard establishes a level of control on the structure of the parallel program; this feature is nonexistent in PVM. The standard was also constructed with the implementors and programmers in mind. It provides complex functions as utilities for the programmers that are not available in PVM. And it provides a better balance between efficiency, portability, and expressiveness than PVM. In 1995, Scott Wallace identified these benefits of MPI over PVM [56]. But MPI was relatively new, and there was no

guarantee that it would receive wide acceptance. PVM was the more widely accepted application, so he opted to use it in his application.

2.1.2.1 MPI functions

MPI has a series of functions to perform tasks during its operation. The basic six functions are:

```
MPI_INIT  
MPI_COMM_SIZE  
MPI_COMM_RANK  
MPI_SEND  
MPI_RECV  
MPI_FINALIZE
```

These are the only six functions that are really necessary for most programs. The other functions all improve performance in some way, but are not needed for simple operation.

The two functions that are mandatory for every MPI program are `MPI_INIT` and `MPI_FINALIZE`. `MPI_INIT` initializes the MPI environment and can only be called once during the operation of a program. It returns an error code, which will either be `MPI_SUCCESS` or an error code that is defined in the implementation. At the completion of the program, `MPI_FINALIZE` must be called by all processes to terminate the MPI environment. No more MPI functions may be called after this one.

The two functions that include the string `COMM` are communicator related functions. The communicator is responsible for relaying the messages within a group of processes. The two `COMM` functions are used to identify the size of the group and the identity of each process within the group. `MPI_COMM_SIZE` returns the value `numprocs`, which is equal to the number of processes in the group. `MPI_COMM_RANK` returns `myid`, which is the process identifier of the process that called the function; this function is normally called by all processes toward the beginning of the program as an initialization step.

The remaining two functions are the heart and soul of MPI. Without these functions, the program might as well be written sequentially. These functions are `MPI_SEND` and `MPI_RECV`. The `MPI_SEND` function is called in Fortran with the following sequence:

```
call MPI_SEND(buf, count, datatype, dest, tag, comm, ierr)
```

where `buf` is the memory location of the data being sent, `count` is the number of data, `datatype` is self-explanatory, `dest` is the rank of the process that is the messages destination, `tag` is a nonnegative integer of arbitrary value that is used to restrict receipt of the message, `comm` is the identifier of the communicator, and `ierr` is an error code variable. The matching `MPI_RECV` call is:

```
call MPI_RECV(buf, count, datatype, source, tag, comm, stat, ierr)
```

where `buf` is the memory location where the message will be stored, `count` is the maximum number of data that this call is allowed to receive, `source` is the rank of the process that is sending the message, `stat` is the status array of the received message, and `datatype`, `tag`, `comm`, and `ierr` are the same as above. The `MPI_SEND` call must have specific values for all inputs, but `MPI_RECV` can use wildcard values for `source` and `tag` (`MPI_ANY_SOURCE` and `MPI_ANY_TAG`). Using this method allows a repetitive function to receive messages from multiple sources as the function returns to this call. The `source`, `tag`, and actual length of the message received can be retrieved from the status array.

Some of the advanced functions provide convenient ways of performing collective operations. These functions can broadcast a message for all processes to receive, or gather data from all process, or compare the values in the processes' buffers. Other advanced functions improve the flexibility, modularity, and accuracy of the MPI programs that use them. These functions can be used for more advanced applications, but are not used in this thesis.

2.2 Applications of Parallel Processing to Constellation Motion

In the past five years, a number of parallel processing applications for the orbit propagation of constellations have been developed. Three such applications were produced by Draper Fellows: Scott Wallace, Naresh Shah, and Brian Kantsiper. But other people who are not connected with Draper Laboratory were also working to construct similar applications.

At the present time, there are two parallel processing applications for constellations in development at Draper Laboratory. One of these applications is being developed by Draper Fellow James Smith. The other is the topic of this thesis.

The next three sections present brief descriptions of the past and present parallel processing applications. The first of these sections describes the parallel processing applications developed at Draper Laboratory. The second section presents a couple of past applications developed external to Draper Laboratory. The last section discusses how parallel processing is used in the current project.

2.2.1 Applications at Draper Laboratory

In 1995, Scott Wallace developed a parallel orbit propagator using the Draper Semianalytic Satellite Theory (DSST) Standalone Orbit Propagator in conjunction with PVM [55,56]. PVM was selected for its portability and its ability to accept the preexisting code with few changes. The PVM/DSST was used to model the evolution of the Teledesic constellation, which was projected to consist of 840 satellites at that time. In addition, genetic algorithms were incorporated into the application to produce an optimization tool that could be used in orbit design.

In 1997, the Automated Station-Keeping Simulator (ASKS) was developed by Naresh Shah [49,70]. This simulator uses the MPI Standard as the basis for its parallel

processing. It propagates all satellites in the constellation simultaneously using the DSST Standalone Orbit Propagator. It assigns one process to each satellite and one to the “ground station,” which is responsible for monitoring these satellites. When the orbital element variations reach an established level of error, the simulation processes a correction to them. It can use several targeting methods to determine how much correction the elements need.

ASKS was used in a station-keeping evaluation of the Ellipso constellation. In this evaluation, ASKS demonstrated its capability to be an autonomous maneuver planner. It was also able to optimize the maneuvers based on its method of targeting. Through the use of this maneuver optimization and some coverage algorithms designed by Brian Kantsiper [35,71], ASKS was able to provide a lifetime analysis of the fuel consumption and coverage for the Ellipso constellation.

In the past year, Proulx, et al. employed MPI coupled with the Parallel Genetic Algorithm Package (PGAPack) [72] to refine the orbits and evaluate the coverage for the Ellipso Gear Array [73]. More recently, J. Smith is employing MPI/PGAPack to develop globally optimal path-constrained satellite station-keeping maneuver strategies [74].

2.2.2 Applications External to Draper Laboratory

The Navy and the Air Force have facilities that are responsible for maintaining a catalog of the objects in orbit. As part of this maintenance, these facilities monitor the space objects as they pass overhead. The observations obtained in this activity are processed to update existing objects in the catalog or to find new ones.

In 1996, S. Coffey, et al. altered the Search and Determine (SAD) program, an orbit determination program developed by Naval Space Command that is used to determine the orbits of unknown objects from uncorrelated observations [12]. This alteration was undertaken to reduce the computational resources required to process large numbers of

observations. SAD processes pairs of points to create candidate orbits for the unknown objects, and refines these candidate orbits through an iterative process of performing differential correction fits of the observations, predicting the locations of additional observations, and acquiring observations near these predictions. The fits and predictions are performed using the PPT2 analytical propagator. The alteration to the program consisted of constructing a parallel processing framework that could be used with the algorithm. Because of its portability to a large number of systems, PVM was selected for the parallel processing implementation. The result of this modification was that SAD could process twelve times more data than before in a period of hours instead of days, and it increased its production of acceptable orbits by twenty-five percent.

In 1997, H. Neal, et al. performed a similar alteration on SPeCIAL-K, a special perturbations numerical propagator used by the Naval Research Laboratory [43]. This alteration was accomplished to demonstrate the feasibility of using a numerical propagator as the tool for maintaining the entire space catalog, not just a few select satellites. Prior to this effort, the space catalog was maintained through the use of the SGP4 and PPT3 analytic propagators, and SPeCIAL-K's operation was limited to the few objects that required more accurate orbit predictions, like weather satellites and objects in the decay phase of their orbits. Through the implementation of a parallel processing environment, SPeCIAL-K now has the capability of performing orbit determination on several objects simultaneously. Tests on the new SPeCIAL-K application showed a measurable improvement in the accuracy of the orbits over those produced using SGP4. The parallel processing environment reduced the required computer resources to the point where it is reasonable to use SPeCIAL-K to maintain the entire catalog.

Although these two applications do not specifically handle constellations of satellites, they do process a large number of objects in orbit. In fact, the objects that they process could be considered a very loosely structured constellation. An extension of their functions to include more structured constellations would not be difficult.

2.2.3 Current Application

In the current project, the application of parallel processing involves the simulation of the GPS navigation solutions for satellites in medium altitude, elliptical orbits. These navigation solutions are produced by simulating the GPS constellation and the target satellite. In the original concept, the pseudoranges and positions of the GPS satellites would be computed in parallel and combined to calculate the navigation solution. However, the final implementation processes the entire constellation for each time step, and the time steps are evaluated in parallel. This application has the ability to use the DSST, just like the other Draper applications.

2.3 Improved Draper Semianalytic Satellite Theory (DSST) Standalone Orbit Propagator

The DSST Standalone Orbit Propagator has been the main propagator for use in the parallel processing environments at Draper Laboratory. It has the advantage of rapid operation without the loss of accuracy associated with the purely analytical propagators. Its rapid performance and the ease with which it could be integrated into these parallel environments made it a good candidate for use in these applications.

There have been changes to the DSST Standalone since it was first used in a parallel processing application. Presented here is a brief history of the DSST Standalone, and its current capabilities.

2.3.1 History

The DSST was created during the advanced development of GTDS. It has existed in two different forms: as a component of GTDS, and as the Standalone Orbit Propagator Package. The history of the DSST is contained within the histories of GTDS and the Standalone. GTDS's history is presented in Appendix B, so the remainder of this section will describe the Standalone.

The Standalone DSST program was created in 1984. Its functionality was limited to the functions that existed in the DSST section of GTDS at the time of its derivation. The program has been revised several times to improve its performance within specific projects.

The Standalone was initially developed for use at Aerospace Corporation and was delivered in 1984 [9]. This initial version included the complete DSST models for the mean motion and a portion of the DSST short-periodic models (the zonals and the tesseral m-dailies). Aerospace adapted the Standalone to meet their needs. First, the program had to be modified to allow a better interface with the CYBER mainframe [27]. This alteration process was documented by G.B. Green; the changes are identifiable and reversible. Then, R.G. Hopkins developed two program shells that utilized the Standalone for mission analysis and stationkeeping applications. These shells are the Aerospace Astroynamics Department Mean Element Orbit Propagation and Stationkeeping program (MEANELT) [31] and SATPROP [32]; SATPROP does not have stationkeeping capabilities. These enhancements enabled the engineers at Aerospace to use the Standalone for their mission planning projects.

While these modifications were being made at Aerospace Corp., the Standalone was employed at Draper Laboratory in a navigation study for satellite constellations [29]. Portability of the Standalone to different computer environments was also considered. The Standalone was ported to VAX, PC [5], and the Sun workstation [6] in the 1989-90 timeframe. In order to improve the accuracy of these ported versions, the program was

altered slightly to compensate for the peculiarities of each machine type. While not all of the concepts developed during the navigation study and porting exercises were implemented, all of this experience found application in later efforts. Draper's 1984 proposal to make the DSST Standalone available in the USAF Colorado Springs computing environment also contributed to this technology [33].

In 1990-92, Draper Laboratory technical staff undertook and completed the development of a Mission Support Program (L6MSN) for the Landsat 6 program [18]. This program included a Maneuver Planning capability and the DSST Standalone was selected to be the orbit propagator for this function. Because L6MSN employed a near real-time, multi-tasking architecture, the top level interface for the Standalone was re-worked to be file driven. The functionality in the Standalone was also expanded to include finite rocket burn models compatible with the DSST and the atmospheric density options were expanded to include the Jacchia-Roberts model [21]. In 1995, the Radarsat Flight Dynamics System [10] employed a similar version of the Standalone which utilized an impulsive burn model instead of the finite burn model.

In addition to these modifications, the Standalone has also been upgraded to improve the maintainability of the source code. This upgrade is the same process that was suggested for GTDS [7] and has brought the Standalone into conformance with the coding standard [11]. One key aspect of this upgrade was the elimination of COMMON blocks and their replacement with structured records and modules; the structuring of the records establishes unique variable names to prevent confusion within the subroutines. The overall goal of this effort was to prepare for a transition to FORTRAN 90. The improvements outlined in the upgrade proposal were performed realizing that new computational paradigms are available and offer opportunities to take advantage of the analytical structure of DSST.

The computational capabilities of the DSST began to differ between the GTDS implementation and the Standalone. These differences arose from the fact that DSST

development continued in GTDS from the time of the Standalone's creation while modifications to the Standalone focused on individual project needs.

A new version of the Standalone has recently been developed. This version incorporates the major functional changes that have been implemented in the DSST in GTDS over recent years. The work of D. Fonte and S. Carter was a major portion of the functional improvement to the DSST. Because Carter's work built on the changes made during Radarsat development, Radarsat's changes were also a concern. By implementing these changes, the Standalone gained access to the 50 x 50 geopotential model, the solid Earth tides models, and the J2000 coordinate systems [44]. Additionally, the DSST Tesseral Linear Combination model was installed into the Standalone to improve the performance of the short-periodic model. These modifications make the Standalone capable of supporting high accuracy osculating to mean element conversion for a broad range of orbit configurations.

2.3.2 Current Functions

The current DSST capabilities in GTDS are based on the initial physical models and the modifications it has undergone since its creation. They include the following models:

Mean Element Equations of Motion

- Central-body gravitational spherical harmonics of arbitrary degree and order (zonals and tesseral resonance) based on the 50 x 50 geopotential
- J_2 -squared second order effect (model based on eccentricity truncation)
- Third-body point mass effects (both single and double averaging theories)
- Atmospheric drag with J_2 /drag coupling
- 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

Short-Periodic Motion

- Central-body gravitational zonal harmonics of arbitrary degree based on the 50 x 50 geopotential
- Central-body gravitational m-daily sectoral and tesseral harmonics of arbitrary degree and order based on the 50 x 50 geopotential
- Central-body gravitational high-frequency sectoral and tesseral harmonics of arbitrary degree and order based on the 50 x 50 geopotential
- J_2 -squared 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

State Transition Matrix

- General concept based on the Generalized Method of Averaging

Mean Element Orbit Determination

- Weighted Least Squares (Batch)
- Kalman Filter

General(applicable to both the mean element and short periodic motion)

- Numerical interpolation techniques
- Modified representations of the Hansen coefficients
- Jacchia-Roberts atmospheric density models
- Multiple output reference frames (including B1950.0 and J2000.0)

Prior to its most recent improvements, the Standalone contained a majority of the functions in the DSST for GTDS. The major exceptions to this statement were:

- Solar and lunar solid Earth tides model
- Integration using the FK5 (J2000.0) coordinate frame
- Geopotential size is still limited to a degree and order of 21
- Short-periodic tesseral linear combination terms (a subset of the high-frequency sectoral and tesseral harmonics)
- J_2 -squared short periodic variations

With the completion of the new modifications, only the J_2 -squared short periodic variations have not been applied to the Standalone [45]. However, two rocket burn models

(finite and impulsive) are present in the Standalone that are absent from GTDS. These burn models were added to the Standalone during the Landsat 6 and Radarsat projects, respectively.

(This page intentionally left blank.)

Chapter 3

Construction of GPS Receiver Navigation Solutions including Realistic Errors - Algorithms

The goal of this project is to determine if GPS navigation solutions can be used to determine and predict higher altitude elliptical orbits. The deterministic error analysis approach is employed. This approach requires actual navigation solutions. But these navigation solutions do not exist for the eccentric orbits of interest. Therefore, they must be simulated.

A program was written to perform this simulation. In order to produce the navigation solutions, this program requires a representation of the true path of the target satellite. This path can be generated using the DSST Standalone [45] or by producing the precise osculating elements for that orbit via special perturbations.

Figure 3.1 presents the functional data flow that is employed in this project. The orbital path of the target is generated using special perturbations. This “truth” is used for two purposes: input to the simulation of the navigation solutions, and a basis of comparison for the simulated navigation solutions and the determined orbit that is produced from them. The simulation takes the “truth” and produces simulated C/A navigation solutions that include realistic errors from selective availability and ionospheric delay. These navigation solutions are used as observations for the orbit determination process. These navigation solutions are used as observations for the orbit determination process.

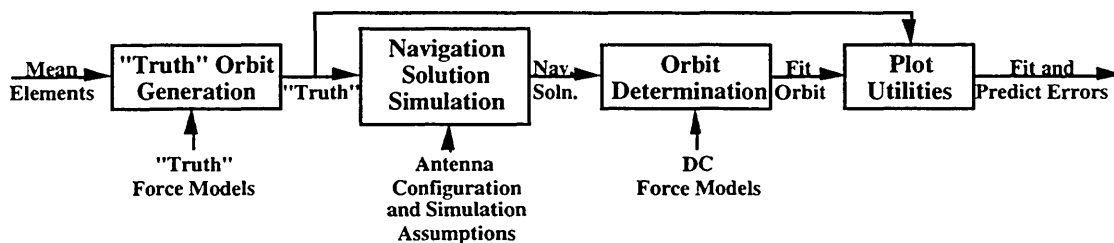


Figure 3.1 Deterministic GPS Navigation Solution Analysis Data Flow

This chapter explains the processes involved in the “truth” orbit generation and the navigation solution simulation displayed in Figure 3.1. It also provides a detailed description of the algorithms implemented in the simulation program.

3.1 Generation of the Target Satellite’s Orbit

The target orbit that is used in the simulation is assumed to be a true representation of the target satellite’s orbital motion. To produce the best approximation of this motion, the orbit is generated using a special perturbations propagator. The target orbit is represented by a file containing the positions and velocities of a precise osculating orbit on a specified time grid. The production of such a file is a three step process (Figure 3.2). This process allows the user to characterize the epoch conditions of the target orbit in terms of mean elements. The process is performed using the Draper version of the GTDS computer program, which is presented in Appendix B. The special perturbations propagator in this version of GTDS produces extremely accurate orbits compared to real world conditions.

The first step of the process is to perform an ephemeris generation using the precise mean elements (PME) of the target’s orbit at the desired epoch. This ephemeris generation uses the DSST propagator and produces the orbit from the sum of the mean elements and the associated short periodics. The force models used in this ephemeris include:

- Atmospheric drag using the Jacchia-Roberts atmospheric density model
- The JGM2 gravitational potential field with degree and order of 50
- Solar and lunar point masses
- Solar radiation pressure
- Solar and lunar solid Earth tidal effects

These same models are also used through the remainder of the input generation process. This data is stored in an ORB1 file [75].

That ORB1 file is used as the input to a differential corrections operation. This process is run with the Cowell special perturbations propagator. This process and the previous one form a combination known as the precise conversion of elements (PCE)¹. The output of this operation is the set of initial osculating elements for the orbit at the same epoch.

The osculating elements are used in the input file for the last step of the input generation. This step is the Cowell ephemeris generation. The output of this operation is a highly accurate approximation of the osculating orbit. An ORB1 file is generated; this file stores the positions, velocities, and other necessary data for the acquisition of the target's "truth" position and velocity for each time step in the simulation.

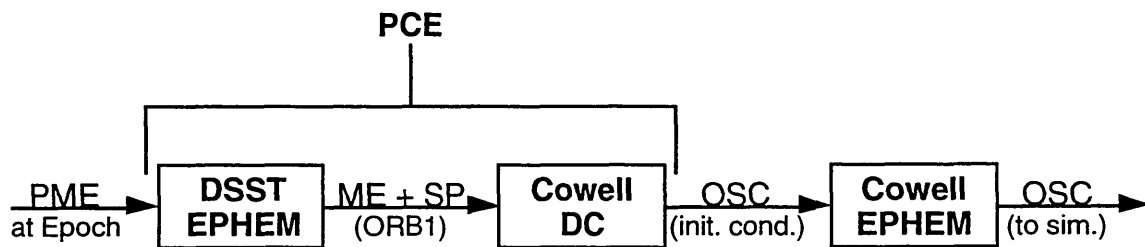


Figure 3.2 Data Flow of "Truth" Orbit Generation

This process is performed for each target orbit and for each epoch in this project. During an examination of the effects of mismodelling drag, the input data was recreated using the Jacchia model for atmospheric density with much noisier data due to the high geomagnetic instability during that epoch.

Because the GTDS version with the most functionality resides on the VAX, the "truth" orbit is generated on the VAX/VMS system. But in order to make use of parallel processing, the simulation was constructed on Draper's Data Center 1 (DC1); DC1 is a Silicon Graphics Origin 2000 computer that uses the IRIX 6.4 operating system and is

¹ This is an unconventional use of the PCE nomenclature. Usually, PCE refers to the generation of a mean element set from an osculating trajectory.

accessible through an Ethernet connection. So the ORB1 file is created in ASCII format to facilitate its porting from the VAX to DC1 for use in the simulation.

3.2 The Simulation

The simulation was developed to produce reasonable approximations of GPS navigation solutions for any target satellite. It takes in the target satellite's orbit for a given period of time, identifies the GPS satellites that are capable of transmitting to the target's location for each moment, and, if there are enough GPS signals at that moment, computes the navigation solution of the target's receiver. To produce the navigation solutions, the simulation must determine the target's time of reception, the time of transmission for each GPS satellite that the target can receive, and the position of each GPS satellite at its time of transmission. A minimum of four GPS signals is required to calculate the navigation solution.

To get an accurate reading of the time of transit from the GPS satellites to the target satellite, the "truth" positions of all satellites are used in the early portions of the simulation. The transit time is subtracted from the reception time to determine the time of transmission.

As a part of the simulation, the ionospheric delay and the errors in the satellite clock are mimicked. These errors result in a realistic level of error in the navigation solutions. To closely match the effects of these errors, they are added to the GPS times of transmission at different points during the simulation. The new times correspond to the time that the target believes to be the time of transmission. These perceived times of transmission are tagged with the number of the GPS satellite that transmitted the signal. When the times have been computed for each of the acceptable GPS satellites, the group is time-tagged for the time of signal reception and passed on to the navigational fix algorithm.

Once the data has been passed to the navigation solution algorithm, the solution can be computed. It uses the time data to compute the pseudoranges. These pseudoranges and the positions of the GPS satellites at the perceived time of transmission are processed to find the target's position. The target's position is time-tagged with the time of signal reception and placed in a file.

The next several sections contain detailed descriptions of the algorithms used in this simulation. The software required to implement these algorithms is listed in Appendix A.

3.2.1 Determination of the “Truth” Positions

The navigational solution algorithm does not need the actual positions of the target and GPS satellites. In fact, its job is to compute the target's position. So, why are these positions determined in the simulation?

The times of signal transmission are required for the navigation solution algorithm. The “truth” positions of the target and the GPS satellites must be known to determine these times. From these positions, the transit times can be calculated; these transit times are subtracted from the time of reception to produce the times of transmission. The target's “truth” positions can also be used to measure the accuracy of the navigation solutions and the accuracy of the solution from the eventual orbit determination process.

To determine the “truth” positions of the target and GPS satellites, their orbital paths must be established. These paths are realized in different ways for the target and for the GPS constellation. The target's orbital path can be generated by using the DSST Standalone or by performing the “truth” orbit generation described in Section 3.1; at the current time, the orbit is produced by the latter method. The positions and velocities of the target satellite are stored in a file with the indicated step size between each state, and are processed for each time of reception requested in the simulation. For the GPS constellation, the orbit can be generated from actual information or through an

approximation of the motion. For this simulation, a mean element approximation of the GPS constellation is used to compute the satellites' orbital paths, but the program can also support the use of the actual positions; this mean motion approximation includes the effects of J_2 zonals. The orbital elements used for the approximation are listed in Table 3.1 [62], and the epoch for these elements is the epoch of the current run; in this table, ΔM is the phase to the adjacent satellite. These elements were applied to arbitrary epochs to produce a representative distribution of the GPS constellation. Since GPS was designed to provide longitudinally independent coverage of the Earth, this distribution should provide an adequate representation of GPS coverage regardless of the epoch used. In addition, the primary epoch of interest is in the future, and there is no way to know what the exact locations of the GPS satellites will be.

Nodal motion was not included in the GPS approximation. The average right ascension node rate of the actual GPS orbits is approximately -7.99×10^{-9} radians per second. Over a period of two weeks, the node changes position by about -0.55 degrees. Since two weeks is the maximum length of a simulation run for this project, this small change in the position of the right ascension node should have little impact on the results.

But the time for the target's "truth" position is not the same as those for the GPS positions (Figure 3.3). The target's position is evaluated at its time of signal reception (t_1); the GPS positions are determined at their times of signal transmission (t_0), which differ from satellite to satellite. Their positions are computed individually through an iterative process that also determines the time of transmission. The process starts from the time of reception, and computes the GPS satellite's position at that time. The range from the GPS satellite to the target is computed, as is the first estimate for the time of transmission for the signal, using the following equations:

$$\mathbf{d} = \mathbf{r}_G(t_1) - \mathbf{r}_t \quad (3.1)$$

$$d = |\mathbf{d}| \quad (3.2)$$

$$t_0^1 = t_1 - \frac{d}{c} \quad (3.3)$$

where

- $r_G(t)$ = the GPS position vector at time t
- r_s = the target's position at time of reception
- d = the difference vector between GPS position and target's position
- c = the speed of light
- t_1 = the time of reception by target
- t_0^1 = the first estimate of the time of signal transmission

Table 3.1 Orbital Elements of the GPS Constellation

Sat	ID	a (km)	i (deg)	Ω (deg)	M(deg)	ΔM (deg)
1	A3	26561.75	55.0	272.847	11.676	103.55
2	A4	26561.75	55.0	272.847	41.806	31.13
3	A2	26561.75	55.0	272.847	161.786	119.98
4	A1	26561.75	55.0	272.847	268.126	106.34
5	B1	26561.75	55.0	332.847	80.956	130.98
6	B2	26561.75	55.0	332.847	173.336	92.38
7	B4	26561.75	55.0	332.847	204.376	31.04
8	B3	26561.75	55.0	332.847	309.976	105.60
9	C1	26561.75	55.0	32.847	111.876	100.08
10	C4	26561.75	55.0	32.847	241.556	129.68
11	C3	26561.75	55.0	32.847	339.666	98.11
12	C2	26561.75	55.0	32.847	11.796	32.13
13	D1	26561.75	55.0	92.847	135.226	100.07
14	D4	26561.75	55.0	92.847	167.356	32.13
15	D2	26561.75	55.0	92.847	265.446	98.09
16	D3	26561.75	55.0	92.847	35.156	129.71
17	E1	26561.75	55.0	152.847	197.046	130.98
18	E2	26561.75	55.0	152.847	302.596	105.55
19	E4	26561.75	55.0	152.847	333.686	31.09
20	E3	26561.75	55.0	152.847	66.066	92.38
21	F1	26561.75	55.0	212.847	238.886	103.54
22	F2	26561.75	55.0	212.847	345.226	106.34
23	F3	26561.75	55.0	212.847	105.206	119.98
24	F4	26561.75	55.0	212.847	135.346	30.00

But the GPS signals experience time delays from environmental and mechanical effects. (The computations of the signal delays are described in section 3.2.3.) These effects need not be included in the initial estimates for position and time of transmission, but they must be included in all subsequent estimates. The GPS position is reevaluated at the time of the time estimate, and the second and subsequent estimates for the time of transmission are computed from the following equations:

$$\mathbf{d} = \mathbf{r}_G(t_0^{n-1}) - \mathbf{r}_s \quad (3.4)$$

$$t_0^n = t_1 - \left(\frac{d}{c} + \Delta t_d \right) \quad (3.5)$$

where

$\mathbf{r}_G(t_0^n)$ = the GPS position for the n^{th} time estimate

t_0^n = the n^{th} estimate of the time of signal transmission

Δt_d = the signal delays

This process is continued until the difference from one time estimate to the next is less than the established tolerance, the time for light to travel one millimeter. The GPS position evaluated for the final estimate of transmission time is accepted as the “truth” position for that GPS satellite and the given time of signal reception.

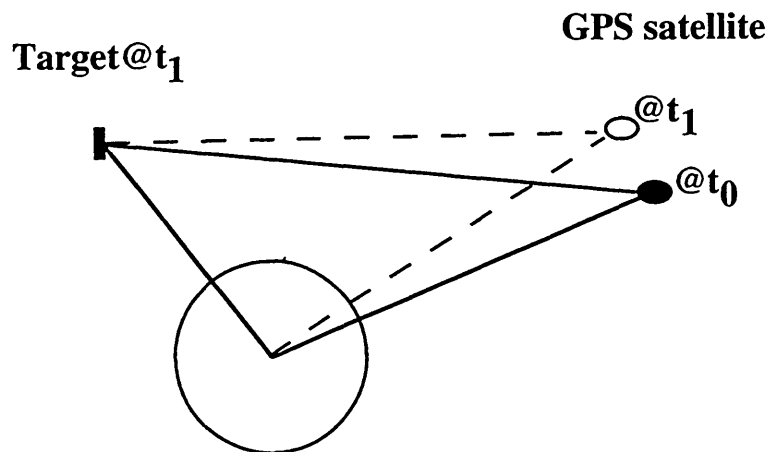


Figure 3.3 “Truth” Positions

3.2.2 Identification of GPS Satellites in View

Once the “truth” positions of all satellites have been determined, the GPS satellite positions must be tested. The test is to verify that a GPS satellite in that location could send a signal that would contact the target at its location of reception. Conditions that could prevent the signal from reaching the target are:

- Earth shadowing
- The target is outside of the satellite’s cone of transmission
- The GPS satellite is outside of the reception cone of the target’s GPS receiver antenna.

If any of these conditions occur, the target receives no signal from that GPS satellite.

To test these conditions, the geometry of the two locations must be evaluated; the third condition also depends on the orientation of the GPS receiver’s antenna. The Earth shadowing is checked by determining the angle between the two position vectors and comparing it to a maximum. For the transmission cone, the angle between the GPS position vector and the difference in position vector is measured. The reception cone condition is tested by evaluating the angle between the difference vector and the antenna’s main axis.

The passing conditions for the Earth shadowing test must be determined for each time of reception and for each individual GPS satellite because the conditions are dependent on the magnitudes of the target and GPS satellite position vectors. The passing condition looks at a triangle with the vertex of vectors with these magnitudes placed at the center of the Earth. The angle between these vectors is set so that the third side of the triangle is tangent to the surface of the Earth (Figure 3.4). The angles between the third side and each of the two vectors are calculated, and their sum is subtracted from 180° . The remainder is

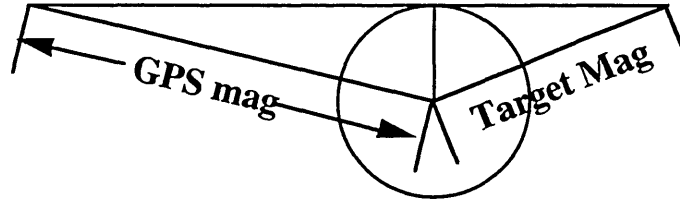


Figure 3.4 Earth Shadowing Test

the maximum allowable angle between the position vectors without Earth shadowing, and the actual geometry should pass the following condition:

$$\arccos\left(\frac{\mathbf{r}_s \cdot \mathbf{r}_G}{|\mathbf{r}_s||\mathbf{r}_G|}\right) \leq 180^\circ - \arcsin\frac{r_E}{|\mathbf{r}_G|} - \arcsin\frac{r_E}{|\mathbf{r}_s|} \quad (3.6)$$

where

\mathbf{r}_s = the target's position vector at time of reception
 \mathbf{r}_G = the GPS position vector at its time of transmission
 r_E = the Earth's radius

The condition for the GPS transmission cone is a fixed value that corresponds to its half-beam angle, which is currently 21.3° [50]. The angle between the difference vector and the GPS position vector (Figure 3.5) is compared to this value, and the inequality is expressed as follows:

$$\arccos\left(\frac{\mathbf{d} \cdot \mathbf{r}_G}{|\mathbf{d}||\mathbf{r}_G|}\right) \leq \alpha_{\max} \quad (3.7)$$

where

\mathbf{d} = the difference vector, as defined earlier
 α_{\max} = the half-beam angle of the GPS transmission

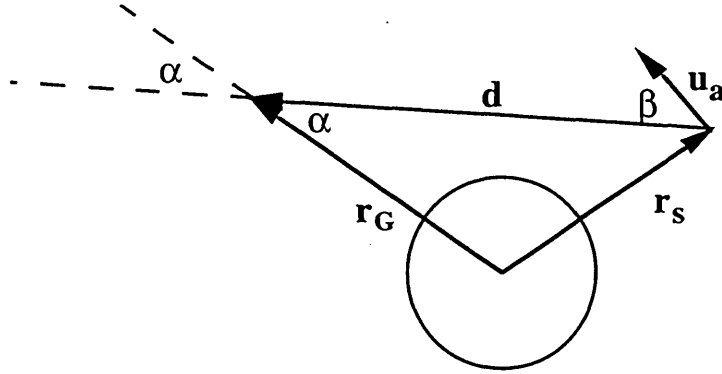


Figure 3.5 Geometry for Verification

The remaining test, the reception cone, requires knowledge of the orientation of the GPS receiver's antenna. For the purposes of this simulation, the receiver antenna can be placed in a variety of different orientations. Due to the motion of the target satellite, the antenna's main axis vector must be calculated for each time of reception. The first step in calculating the antenna's axis vector is to establish the target's orbital plane. This step is simplified by using vectors from an Earth-centered, inertia (ECI) reference frame. This is the only computation in this simulation of GPS data that is affected by reference frame, so all position and velocity vectors in these portions of the simulation are in ECI. A cross-product of the target's position and velocity vectors from this reference frame produces the angular momentum vector, which is normal to the orbital plane. A cross-product of this normal vector and the position vector produces a vector in the orbital plane that is normal to and leading the position vector. Using the position vector, the normal to the plane, and the normal within the plane as principle axes, the antenna can easily be placed into the desired orientation. For the initial orientation, the antenna's unit vector is mapped on the normal vector within the orbital plane. From this orientation, a simple rotation about the normal to the plane or the position vector provides the desired orientation. The equations used to establish this direction are:

$$\mathbf{r}_s \times \mathbf{v}_s = \mathbf{h}_s \quad (3.8)$$

$$\mathbf{h}_s \times \mathbf{r}_s = \mathbf{a}_i \quad (3.9)$$

$$\mathbf{u}_a = R_n(\alpha_a) \frac{\mathbf{a}_i}{a_i} \quad (3.10)$$

where

- \mathbf{r}_s = the position vector of the target satellite
- \mathbf{v}_s = the velocity vector of the target satellite
- \mathbf{h}_s = the angular momentum vector of the target satellite
- \mathbf{a}_i = the intermediate axis vector of the antenna
- \mathbf{u}_a = the unit vector of the antenna's axis
- R_n = the matrix for a rotation about the n^{th} principle vector
 - 1) position vector
 - 2) normal vector within plane
 - 3) normal vector to the orbital plane
- α_a = the desired angle of rotation

After the orientation is established, the test itself is very simple. The angle between the antenna's axis vector and the difference vector is compared to the reception half-beam angle, which for these purposes is 70° . The condition is tested as follows:

$$\arccos\left(\frac{\mathbf{u}_a \cdot \mathbf{d}}{|\mathbf{d}|}\right) \leq \beta_{\max} \quad (3.11)$$

where

- β_{\max} = the reception half-beam angle

If any one of these tests fail, the GPS satellite is unacceptable for the processing of transmission time. The simulation continues, ignoring this satellite and all others that fail these tests for the remainder of the individual time step.

Some testing was required to determine the desired orientation of the antenna. The orientation was selected based on the volume of solutions it could acquire during an orbit. The first series of tested orientations resulted from rotating the antenna about the normal to the orbital plane in ten degree increments. The simulation operated for each of these

orientations for a three hour orbit. The navigation solutions were tabulated to determine the orientations that had the least amount of time when solutions could not be processed. Figure 3.6 shows the results of this test. In this figure, a positive antenna angle indicates a rotation about the plane's normal toward the position vector. The data indicates that the least time is lost in the near-nadir and the most is lost toward zenith.

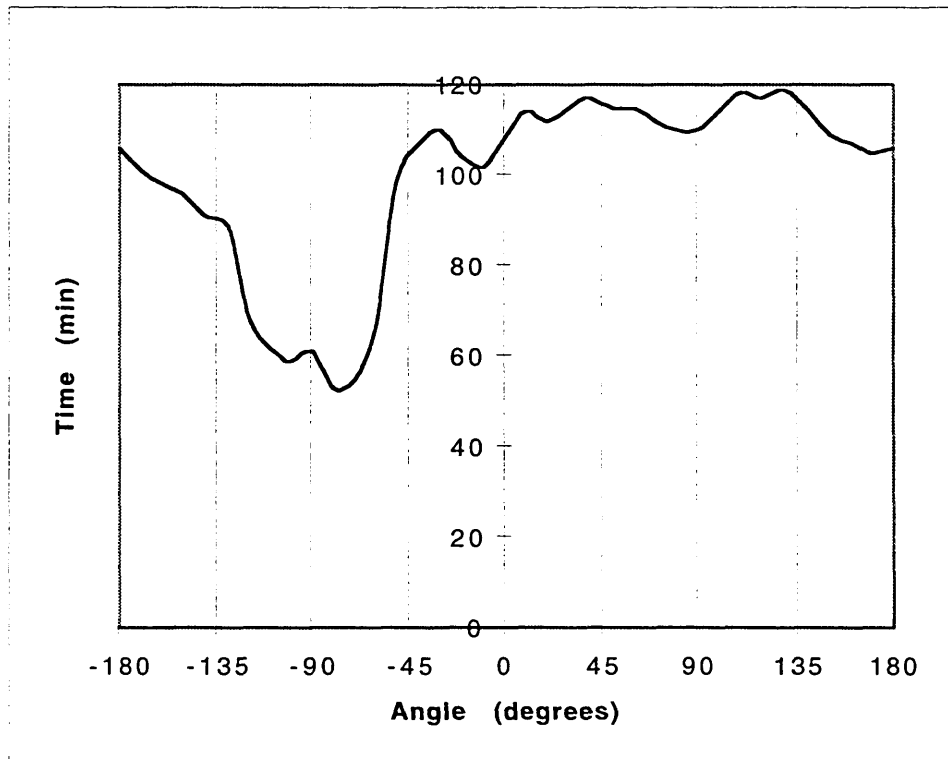


Figure 3.6 Lost Time vs. Antenna Angle (within plane)

After evaluating orientations within the orbital plane, the out of plane orientations were examined. The antenna started in the same initial direction as for in-plane, but the rotation was about the position vector itself. The angle increments were the same, as was the length of the runs. Figure 3.7 displays these results. In this case, a positive angle means a rotation to the target's right. There is a less than thirty minute variation in times out of plane, as opposed to the two hour variation in-plane, but the least time loss appears to occur in the region trailing the target's right side.

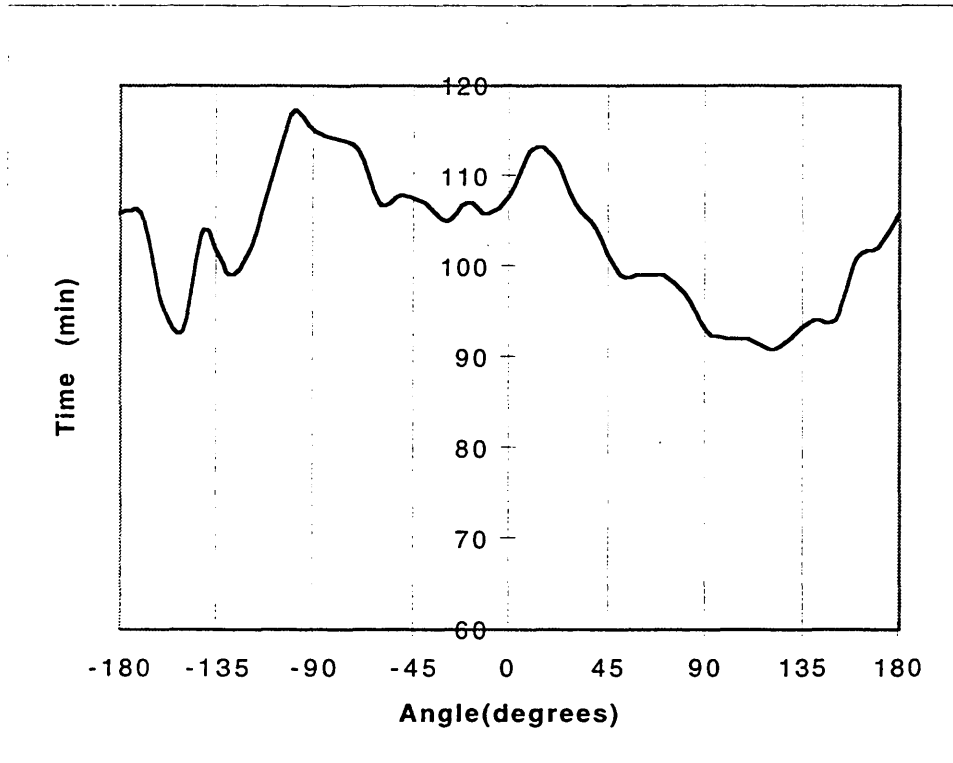


Figure 3.7 Lost Time vs. Antenna Angle (out of plane)

In hopes that the lost time could be reduced further, the in-plane and out of plane angles that resulted in the minimum time loss were combined into a new orientation. However, this new orientation had benefits over the near-nadir case.

So, for the higher altitude elliptical orbits, the optimum antenna position appears to be approximately ten degrees off of nadir. Because the direction is easier to maintain and the variation for the optimum is so small, the nadir orientation was selected for the antenna configuration. But a nadir facing antenna provides little or no data near perigee, which is the more difficult portion of the orbit to determine. Conversely, the zenith facing antenna acquires several solutions near perigee, and few to no solutions at higher altitudes. For this reason, three antenna configurations were considered throughout the remainder of the analyses: nadir facing, zenith facing, and a two-antenna configuration including one of each.

3.2.3 Simulation of Signal Errors

The errors associated with the GPS process are evaluated at different stages in the simulation. Delays from atmospheric and mechanical effects are calculated during the estimation of the GPS satellite “truth” positions. The satellite clock errors from SA and bias are computed after the “truth” time of transmission has been determined and are applied to that time to produce the time that the target satellite believes the signal was transmitted. And the GPS ephemeris errors are computed during the calculation of the GPS positions to be used in the navigation solution algorithm.

Not all GPS related errors are incorporated into this simulation. The inclusion of errors started with the ones that have the greatest impact on pseudorange and worked down. This simulation includes the errors due to satellite clock (with SA) and ionospheric effects. The errors due to tropospheric effects were not included because these errors have little effect on the pseudorange and the GPS signals will not usually dip that far into the atmosphere for this application. The multipath errors were left out, partially for its low effect and partially for the complexity of modeling. The receiver errors have the smallest impact of all the errors, so their effects are usually ignored. The ephemeris errors, the third largest error when SA is in operation, were intended to be included, but the question of how to implement the errors was not addressed in time. Any of these error models could be added or improved in the future.

3.2.3.1 Clock errors

The explanation of errors in the introduction presented two different types of errors connected to the satellite clock, the inherent errors in the clock itself and the intentional dithering of the clock for SA. When SA is in operation, the satellite clock is the most significant source of error by an order of magnitude. Without SA, the satellite clock has as

much of an effect on range as the ephemeris data and less than the ionosphere. An assumption was made that SA is always on. This assumption is true at the moment, and if it were not, SA would still be applied in this simulation for the worst case scenario.

The two types of satellite clock error can be combined into a single model, and have been in this simulation. The range error effects of the satellite clock are modeled using the second-order Gauss-Markov model [54], which uses the following equation:

$$\ddot{x}_p + 2\beta\omega_0\dot{x}_p + \omega_0^2x_p = c \times w \quad (3.12)$$

where

- ω_0 = the natural frequency
- β = the damping factor, which is less than 1
- w = white Gaussian noise

Deriving the discrete-time state-space equation from equation 3.12 and the equation $x_v = \dot{x}_p$ produces:

$$\begin{bmatrix} x_p \\ x_v \end{bmatrix}_{i+1} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} x_p \\ x_v \end{bmatrix}_i + \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_i \quad (3.13)$$

where w_1 and w_2 are Gaussian white noise functions with zero mean and a variance of 1.

The elements of the state transition matrix are calculated from the equations:

$$\begin{aligned} \phi_{11} &= e^{-\beta\omega_0\Delta T} \left[\cos(\omega_1\Delta T) + \beta \left(\frac{\omega_0}{\omega_1} \right) \sin(\omega_1\Delta T) \right] \\ \phi_{12} &= \left(\frac{1}{\omega_1} \right) e^{-\beta\omega_0\Delta T} \sin(\omega_1\Delta T) \\ \phi_{21} &= -\omega_0^2\phi_{12} \\ \phi_{22} &= e^{-\beta\omega_0\Delta T} \left[\cos(\omega_1\Delta T) - \beta \left(\frac{\omega_0}{\omega_1} \right) \sin(\omega_1\Delta T) \right] \end{aligned} \quad (3.14)$$

where

$$\begin{aligned}\omega_1 &= \omega_0 \sqrt{1 - \beta^2} \\ \Delta T &= \text{the update period}\end{aligned}$$

The u matrix is the square root of the error covariance matrix of the white noise such that:

$$\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} \begin{bmatrix} u_{11} & 0 \\ u_{12} & u_{22} \end{bmatrix} \quad (3.15)$$

where the elements of the u matrix are calculated as:

$$u_{11} = \sqrt{Q_{11} - \frac{Q_{12}^2}{Q_{22}}} \quad u_{12} = \frac{Q_{12}}{\sqrt{Q_{22}}} \quad u_{22} = \sqrt{Q_{22}} \quad (3.16)$$

and the elements of the covariance matrix are calculated as:

$$\begin{aligned}Q_{11} &= \frac{c^2}{4\beta\omega_0^3} \left[1 - \frac{\omega_0^2}{\omega_1^2} e^{-2\beta\omega_0\Delta T} \left(1 - \beta^2 \cos(2\omega_1\Delta T) + \beta \frac{\omega_1}{\omega_0} \sin(2\omega_1\Delta T) \right) \right] \\ Q_{12} = Q_{21} &= \frac{c^2}{4\omega_1^2} \left[e^{-2\beta\omega_0\Delta T} (1 - \cos(2\omega_1\Delta T)) \right] \\ Q_{22} &= \frac{c^2}{4\beta\omega_0} \left[1 - \frac{\omega_0^2}{\omega_1^2} e^{-2\beta\omega_0\Delta T} \left(1 - \beta^2 \cos(2\omega_1\Delta T) - \beta \frac{\omega_1}{\omega_0} \sin(2\omega_1\Delta T) \right) \right]\end{aligned} \quad (3.17)$$

where

$$\omega_0 = \frac{\sigma_v}{\sigma_x} \quad c^2 = 4\beta \frac{\sigma_v^3}{\sigma_x} \quad (3.18)$$

$$\text{range sigma: } \sigma_x = \sqrt{\frac{c^2}{4\beta\omega_0^3}} \quad \text{velocity sigma: } \sigma_v = \sqrt{\frac{c^2}{4\beta\omega_0}} \quad (3.19)$$

The damping factor β has a constant value in the Gauss-Markov process. For this simulation, the values of ω_0 and c^2 are taken from the Radio Technical Commission for Aeronautics (RTCA) recommendation [54]. These three values are listed below:

$$\beta = \frac{1}{\sqrt{2}} \quad \omega_0 = 0.012\text{rad/s} \quad c^2 = 0.002585\text{m}^2 \quad (3.20)$$

Gaussian white noise generators are used for two purposes in the simulation of the clock errors. The first purpose is to provide input to the noise term of the state-space equation as discussed earlier in this section. The second is to produce the initial values of x_p and x_v . These values are Gaussian with zero mean and variances of σ_p and σ_v respectively.

The result of this simulation is a representation of the range error and range rate error on pseudorange caused by the satellite clock. This particular model was selected because it produces noisier errors than those measured from the actual GPS satellites, so it can be considered a worst case for SA. However, this representation is statistically similar to the actual clock errors. How these errors are usually applied to the pseudorange in other simulations is unclear. For this project, the range error is converted to a time difference by division by the signal's velocity (the speed of light), and subtracted from the time of transmission that was computed in the iterative process.

3.2.3.2 Ionospheric effects

Ionospheric interference is caused by free electrons disturbing the propagation of the signal which results in an increase in the transit time. The degree of interference is dependent on the density of free electrons in the space through which the signal is traveling [36]. For ground-based GPS receivers, the signal will always pass through the ionosphere to reach them and the distance of travel through the ionosphere is only affected by the elevation angle to the GPS satellite; the lower the elevation angle, the greater the distance through the ionosphere. For a GPS receiver mounted on a satellite, these constraints do not apply.

The GPS signal may not pass through the ionosphere at all, or it may travel through this region for a distance greater than the longest path to a receiver on Earth.

The calculation of the ionospheric delay for GPS signals is based on the following equation [36]:

$$\Delta t = \frac{40.3}{cf^2} \int Ndl \quad (3.21)$$

where

- $\int Ndl$ = the total electron content (TEC) of the signal path
- l = the length of the path
- f = the frequency of the signal
- c = the speed of light

This equation requires the length of the path through the ionosphere. To determine this value, the graze radius must be known. The graze radius is distance of the signal's closest pass to the center of the Earth. Assuming the ionosphere is a perfect sphere, the length of the path from the edge of the ionosphere to the graze radius is the same whether the signal is approaching the graze radius or leaving it (Figure 3.8). This is due to the fact that the signal path is tangent to a sphere with a radius equal to the graze radius and concentric with the ionosphere. Figure 3.8 is not drawn to scale for the Earth and the ionosphere, but is simply a geometric example. The remaining figures in this section are constructed the same way.

From the graze radius, the length of the path through the ionosphere and its derivative with respect to the angle from the graze radius are determined as follows:

$$\cos \theta = \frac{r_{\text{graze}}}{r_{\text{ion}}} \quad (3.22)$$

$$l = 2r_{\text{ion}} \sin \theta \quad (3.23)$$

$$dl = 2r_{\text{ion}} \cos \theta d\theta \quad (3.24)$$

where

- θ = the angle between the graze radius and point where the signal intersects the maximum ionospheric radius
- r_{graze} = the graze radius
- r_{ion} = the maximum radius of the ionosphere
- l = the length of the path through the ionosphere

Substituting this expression into Equation 3.21 produces:

$$\Delta t = \frac{40.3}{cf^2} \int_{\theta_i}^{\theta_r} N \cos \theta d\theta \quad (3.25)$$

where

θ = the angle between the graze radius and the current position

$N = N(r)$ and $r = r(\theta)$

$\therefore N = N(\theta)$

This equation requires a function for the electron density N as a function of height. This function is not currently available to this work.

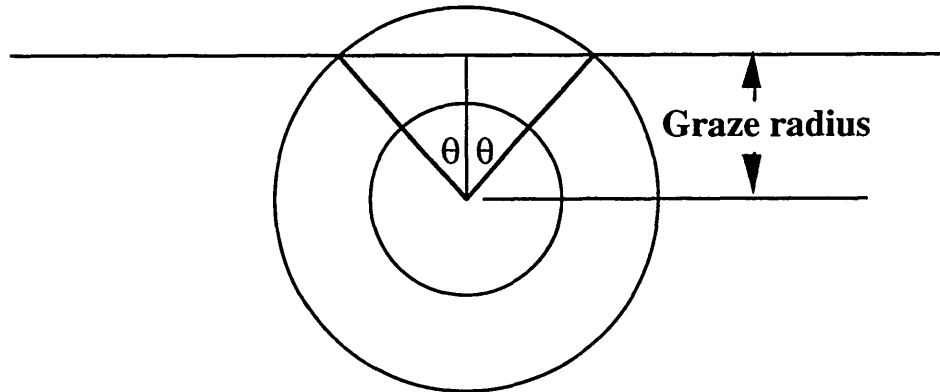


Figure 3.8 Example of Graze Radius

A secondary method of calculating ionospheric delay was devised that models the ionosphere as having uniform electron density. This model is not very accurate, but it does provide a starting point from which improvements can be made. This model uses the worse case scenario for TEC, giving it a value of 10^{19} electrons/m² [36]. This TEC

corresponds with a signal at zero elevation angle passing through the ionosphere during maximum solar activity. The ionospheric delay is scaled based on path length and this value of TEC in the following equation:

$$\Delta t = \frac{40.3}{cf^2} \text{TEC}_{\max} \frac{l}{l_{\max}} \quad (3.26)$$

where

- TEC_{\max} = the maximum value of total electron content
- l_{\max} = the maximum ionospheric path length for a ground-based GPS receiver
- l = the actual ionospheric path length of the sign

The last piece of information necessary for this model is the actual path length through the ionosphere. The positions of the target and the GPS satellite and the value of the graze radius must be known to compute this length.

The graze radius is calculated from the satellites' positions. The signal path is basically the difference vector that has been discussed in previous sections. The value of the graze radius is determined from a right triangle with a hypotenuse equal to the GPS radius at the given time (Figure 3.9). The opposite angle from the graze radius is the angle between the difference vector and the GPS position vector. Simple trigonometry provides the solution:

$$r_{\text{graze}} = r_G \sin \alpha \quad (3.27)$$

where

$$\alpha = \arccos \left(\frac{\mathbf{r}_G \cdot \mathbf{d}}{|\mathbf{r}_G| |\mathbf{d}|} \right)$$

The size of the final angle can be gained through simple subtraction.

There are three possible value ranges for the graze radius: greater than the outer radius of the ionosphere, within the range of ionospheric radii, and less than the lower

radius of the ionosphere. If the graze radius is greater than the ionosphere's top radius, the signal does not travel through it and experiences no ionospheric delay. To determine the length of travel in the remaining conditions, they must be broken down into multiple cases.

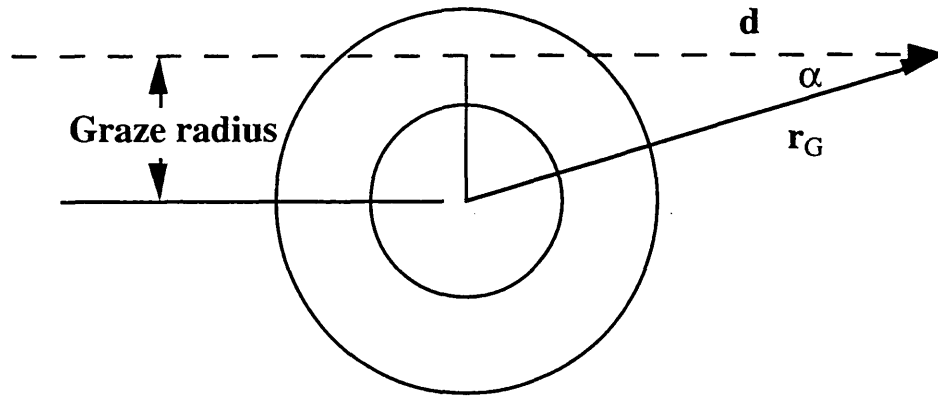


Figure 3.9 Determination of Graze Radius

The first major case is: the graze radius is less than the ionosphere's greatest radius. This case is divided into four subcases, based on the geometry of the satellites. These cases are depicted in Figure 3.10. These four subcases can be broken into two classes, target above ionosphere and target within ionosphere. For the target above condition, a test of the angle between the target and GPS position vectors will determine if the target is at C_1 or C_4 . If the angle between the position vectors is less than the angle between the graze radius and the GPS vector, the target is in C_1 and the signal does not pass into the ionosphere. If the angle is greater, the target is in C_4 and the path length is given by Equation 3.23. For the target within condition, there is no difference in the equations to determine their path lengths. The equation is:

$$l = r_{ion} \sin \theta + r_s \sin(\gamma - \beta) \quad (3.28)$$

where

- r_s = the magnitude of the target satellite's position vector
- γ = the angle between the satellites' position vectors

β = the angle between the graze radius and the GPS vector

Since the quantity $(\gamma - \beta)$ is negative in subcase C_2 and positive in C_3 , the equation produces the correct length for both cases.

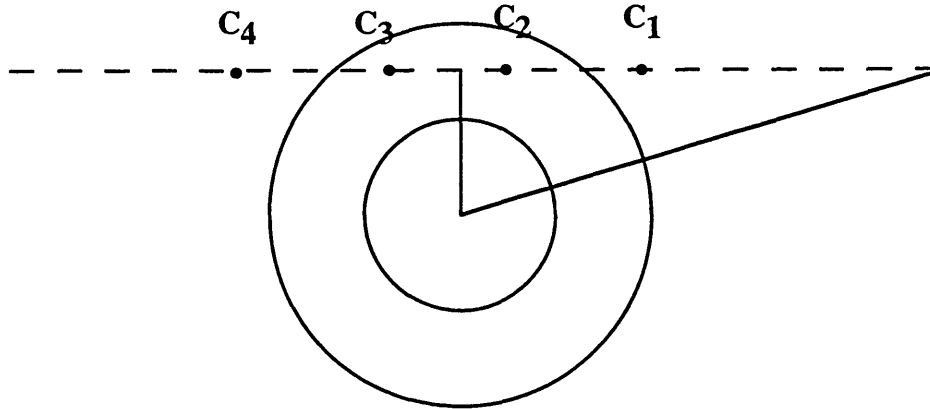


Figure 3.10 Subcases of $r_{\text{graze}} < r_{\text{ionmax}}$

The second major case is the graze radius is less than the lowest ionospheric radius. If the signal passes into this region, some portion of the path length may not be affected by the ionospheric delay. If the path length is zero when this case is tested, this condition is ignored. Again, there are four sub cases to this major case, and they are presented in Figure 3.11. There are also two classes to these cases, target above lower radius and target within lower radius. The conditions to test between C_5 and C_8 are the same as those for C_1 and C_4 , but the value being calculated is the portion of the previous length that is unaffected by ionospheric delay and will be subtracted. The portion excluded from C_8 is calculated using a modified version of Equation 3.23, but the magnitude and angle used in this case are different. The equation is:

$$l_r = 2r_{\text{ionmin}} \sin \theta_{\text{in}} \quad (3.29)$$

where

$$\theta_{in} = \arccos \frac{r_{graze}}{r_{ionmin}}$$

l_r = the portion of the length to be subtracted

r_{ionmin} = the minimum radius of the ionosphere

θ_{in} = the angle between the graze radius and point where the signal intersects the inner ionospheric radius

The length reduction for cases C_6 and C_7 can be calculated from the same equation, just as with cases C_2 and C_3 . In fact, their equations differ in the same way as the equations for C_4 and C_8 do. Their equation for length reduction is:

$$l_r = r_{ionmin} \sin \theta_{in} + r_s \sin(\gamma - \beta) \quad (3.30)$$

The major cases are checked in the order presented above. Upon completion of these tests, the path length has been computed and can be dropped into Equation 3.26 to find the ionospheric time delay. This delay is added to the transit time computed in the iterative calculation for the time of transmission.

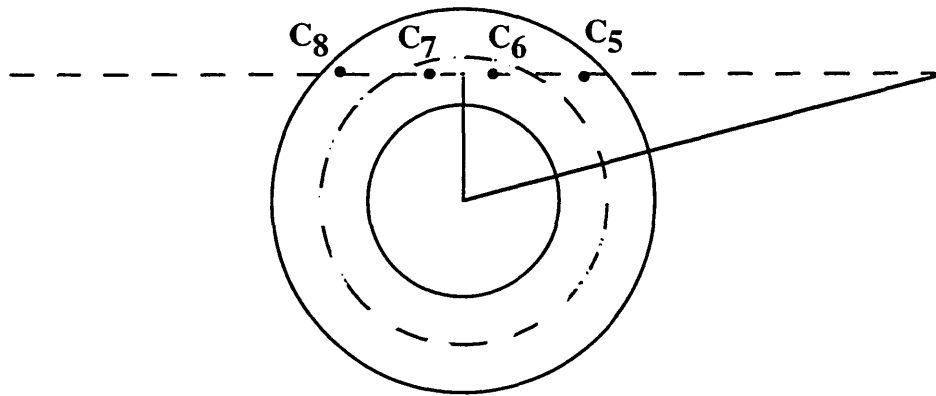


Figure 3.11 Subcases of $r_{graze} < r_{ionmin}$

3.2.3.3 Ephemeris errors

This simulation allows for three different ways of computing the GPS ephemeris positions that are used in the navigation solution. They can be calculated from the ephemeris

coefficients that are listed in tables. They can also be computed from coefficients that are determined during the processing of the program. Or they can be computed as the “truth” positions were. In the first two cases, the errors are worked into the coefficients for the ephemeris positions. For the third case, errors must be added to the “truth” positions.

There are 17 pieces of data in the GPS signal for use in determining the ephemeris position, the 16 coefficients and the IODE parameter [51]. The number of coefficients was limited to 16 by the size of the GPS message. These coefficients are derived from a least squares fit of the reference trajectory and are updated on at least a daily basis. The most frequently used fits are processed for a 4-hour period and have a 3-hour overlap with the next fit. If the daily update does not occur, the fits for day 2 through day 14 are based on a 6-hour fit of the orbit and an overlap time of 2 hours with the next interval. The accuracies of these fits are within 0.35 meters and 1.5 meters for a period of 3 and 2 hours from time of data transmission, respectively [51]. If SA includes intentional ephemeris errors, the errors are worked into the fit before the coefficients are determined.

For the case of coefficients being drawn from a table, there are two possibilities for the source of these coefficients. They could be taken from the actual GPS data that was sent by the satellites. These values are available through sites on the Internet. They could also be derived from a simulated GPS constellation. These coefficients could be evaluated beforehand and store in a file that the navigation solution algorithm can access. A similar method could be used to simulate the coefficients within the program and produce the second case mentioned in the first paragraph of this section.

Currently, the third case, computing the “truth” and adding errors, was the intended method for determining the ephemeris positions of the GPS satellites. However, the ephemeris “truth” is not the same as the “truth” described in Section 3.2.1. The ephemeris “truth” is computed in the same manner, but it is evaluated at the time that the target satellite believes the GPS signal was sent rather than the actual time of transmission. This perceived time includes the effects of the ionosphere, which actually delay the signal, but it

also includes the variations associated with the clock dither. The process introduces errors without adding anything to the position, but additional errors will be included.

At present, the generation of the additional ephemeris errors is not included in the simulation. The subroutine that will contain this generation has been created as a stub. The program also includes all the connections necessary to add the coefficient functionality to the simulation at some future time.

3.2.4 Producing the Navigation Solutions

All of the data required for a receiver to determine its positions has now been simulated. This information is presented to the navigational algorithm for processing.

The navigational algorithm in this simulation is a single-point, least squares solution to the linearized observation equation (1.3) as described in section 1.1.2. The Moore-Penrose generalized inverse is used to determine the inverse of the geometry matrix G . The solution is iterated until the change in estimate is smaller than the tolerance.

But this process requires an initial estimate of the target's position. This a priori position is used to determine the line-of-sight unit vectors for the G matrix and the predicted pseudorange for the calculation of $\Delta\rho$. Without these data, the solution cannot be started

3.2.4.1 A priori estimate of position

The basic pseudorange equations are reexamined to determine an initial estimate for the target's position. These equations are [1]:

$$\begin{aligned}\rho_i &= |\mathbf{r}_i - \mathbf{r}_u| + c \cdot b_u + \varepsilon_{\rho_i} \\ &= \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + c \cdot b_u + \varepsilon_{\rho_i}\end{aligned}\quad (3.31)$$

where i indicates the number of the GPS satellite and u the user. For the initial estimate, the error ε_p is assumed to be zero for all satellites. To produce a set of equations that are linear with respect to the unknowns, Equation 3.31 is rearranged and squared to produce:

$$(\rho_i - c \cdot b_u)^2 = (x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2 \quad (3.32)$$

These equations are subtracted from each other to eliminate the squares of the unknown terms. After simplification the equations now look like:

$$\begin{aligned}(x_i - x_j)x_u + (y_i - y_j)y_u + (z_i - z_j)z_u - (\rho_i - \rho_j)c \cdot b_u \\ = \frac{1}{2}(x_i^2 - x_j^2 + y_i^2 - y_j^2 + z_i^2 - z_j^2 - (\rho_i^2 - \rho_j^2))\end{aligned}\quad (3.33)$$

Since the minimum number of satellites for a navigational solution is 4, the initial estimate must be made with only 4 GPS ephemeris position. This limitation results in only 3 linearly independent equations from Equation 3.33. For this reason, the clock bias is assumed to be zero. This simplification produces a matrix of 3 equations with 3 unknowns.

$$\begin{bmatrix} (x_1 - x_2) & (y_1 - y_2) & (z_1 - z_2) \\ (x_1 - x_3) & (y_1 - y_3) & (z_1 - z_3) \\ (x_1 - x_4) & (y_1 - y_4) & (z_1 - z_4) \end{bmatrix} \begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = \frac{1}{2} \begin{bmatrix} r_1^2 - r_2^2 - (\rho_1^2 - \rho_2^2) \\ r_1^2 - r_3^2 - (\rho_1^2 - \rho_3^2) \\ r_1^2 - r_4^2 - (\rho_1^2 - \rho_4^2) \end{bmatrix} \quad (3.34)$$

where

$$r_i^2 = x_i^2 + y_i^2 + z_i^2$$

The initial estimate of the target's position is the solution to this equation, which can be found by inverting the matrix and multiplying it on both sides of the equation. This

solution can be used as the a priori value to compute the initial geometry matrix and the vector for change in pseudorange.

3.2.4.2 Satellite selection

There is a satellite selection process performed in all GPS receivers [62]; that selection process chooses the GPS satellites to be used in the navigation solution. In this simulation, all satellites in view are used in the navigation solution, but only four are used to determine the a priori estimate of position. This satellite selection process determines which satellites should be used for this a priori estimate.

At first, there was no real selection process. Satellites were taken on a first come, first served basis. This method resulted in a significant problem. On several occasions, the first four satellites produced a set of equations for the a priori matrix that were linearly dependent. Subsequently, the erroneous values calculated for the a priori estimate caused the navigation solution to fail.

To eliminate this problem, a selection process was established. A matrix for a priori calculation is constructed for each combination of accepted satellites. The determinant of each matrix is computed and all combinations with a determinant of zero are discarded. That step alone should eliminate the problem described in the last paragraph.

Rather than just eliminating the combinations that would cause failures, the selection process should also help the performance of the a priori estimation. The best a priori estimate would be derived from the best satellite geometry as input. GDOP is a measure of satellite geometry; smaller GDOP values indicate a better satellite geometry. Therefore, the satellite combination with the best GDOP would be selected.

Instead of directly computing the GDOP for all of the combinations, an alternative measure was determined. For a square matrix, the GDOP of the matrix is inversely proportional to the magnitude of its determinant. The determinant is already being

computed to throw out the bad selections. So, the magnitudes of the determinants are stored by combination identifier during the elimination process. All satellite combinations that survive the first test are compared to find the largest determinant magnitude. That combination is selected for the a priori estimate.

This solution was not entirely successful. On a rare occasion, similar failures crept into the simulation. The source was identified as a situation where the best determinant in the selection process was still remarkably small, but non-zero. A new approach was taken to eliminate the problem. The satellite selection from the previous process is tested for its geometric distribution. The matrix is separated by rows into three vectors. These vectors are reduced in magnitude to unit vectors. The normal to the plane of the first two vectors is determined by taking their cross product. The angle between this normal and the third vector is calculated from their dot product. If the angle between them is 90° , the a priori estimate and the navigation solution are not computed. To eliminate navigation solutions with a high GDOP, a tolerance for the proximity of this angle to 90° was selected.

These selection processes have eliminated the problems associated with unacceptable a priori estimates. They also remove navigation solutions with undesirable levels of GDOP from consideration.

(This page intentionally left blank.)

Chapter 4

Construction of GPS Receiver Navigation Solutions including Realistic Errors - Simulation Architecture

While developing the algorithms for the navigation solution simulation, the idea of using parallel processing was introduced. The base algorithms could be operated in a sequential program, which would perform the same function, but it would take longer to process the large number of time steps that were intended to be used. So, parallel processing was the way to go, but what architecture should be used and in what environment?

The selection of the environment was relatively easy. Past astrodynamics projects at Draper Laboratory have performed parallel processing using PVM and MPI. Another current project uses MPI, which means that the required software and some assistance in learning its operation were readily available. In addition, the structure prescribed in the MPI standard made the implementation of the program easier.

As for the architecture, a survey of information on parallel processing in general and MPI in particular revealed an architecture which seemed well suited to this application. It is the self-scheduling architecture [28]. However, during the design of the simulation, changes were made to this basic architecture.

4.1 Self-Scheduling Architecture

The self-scheduling, or master-slave, algorithm is one of the most common parallel algorithms [28]. This selection of algorithm may be popular for the ease of visualizing the operations. One can picture a manager or foreman with a fist full of tasks that need to be completed. His workers come up to him at the beginning of the shift to get their initial assignments. As each worker finishes his task, he returns to the foreman who gives him

the next assignment on the stack. No tasks are reserved for specific workers; they are simply given the next task. This process continues until the stack is completed or the shift ends. This description can easily be designed into an MPI program.

This self-scheduling architecture seemed to be a good framework for the navigation solution simulation. The work could be broken down into smaller tasks and distributed to the processes.

The original concept for the organization of the simulation was to have the master process operate the target satellite and the slaves would work on the GPS satellites individually and for a single time step (Figure 4.1). Once a slave completed the processing of its satellite, it would return the data to the master. The master would give the slave the next satellite for processing; if the 24th satellite was just given out, the master advanced to the next time step and assigned satellite number 1. After the master had received the GPS data for all 24 satellites in a given time step, it would calculate the navigation solution for the target satellite. This organization appeared to have a significant problem. The whole point of using the master-slave architecture was to keep the slaves occupied as much as possible, but when the master began to process the navigation solution, the slaves would end up waiting for the master to complete its task.

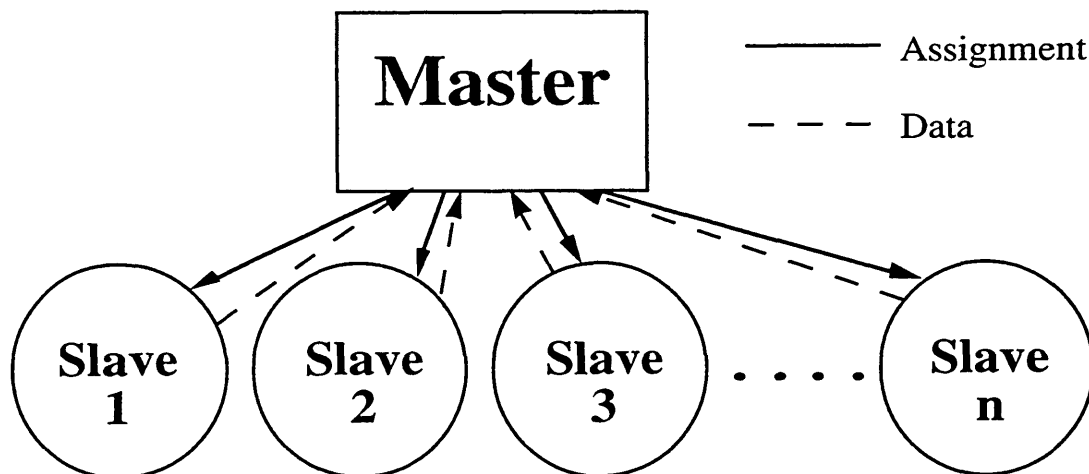


Figure 4.1 Architecture #1

The solution to this problem was the incorporation of a new, special slave (Figure 4.2), one whose tasks are restricted to the processing of the navigation solutions. So, the master continued to operate the target satellite and to assign the satellites in the same manner. When a slave completed its satellite, it would not send the data to the master, but to the special slave; then the regular slave would report to the master for its next assignment. The special slave collected the data from the slaves and stored them in an array based on their time step. Once it had the data for all 24 satellite for the next time step, it would process the navigation solution. A new problem arose with this design. The regular

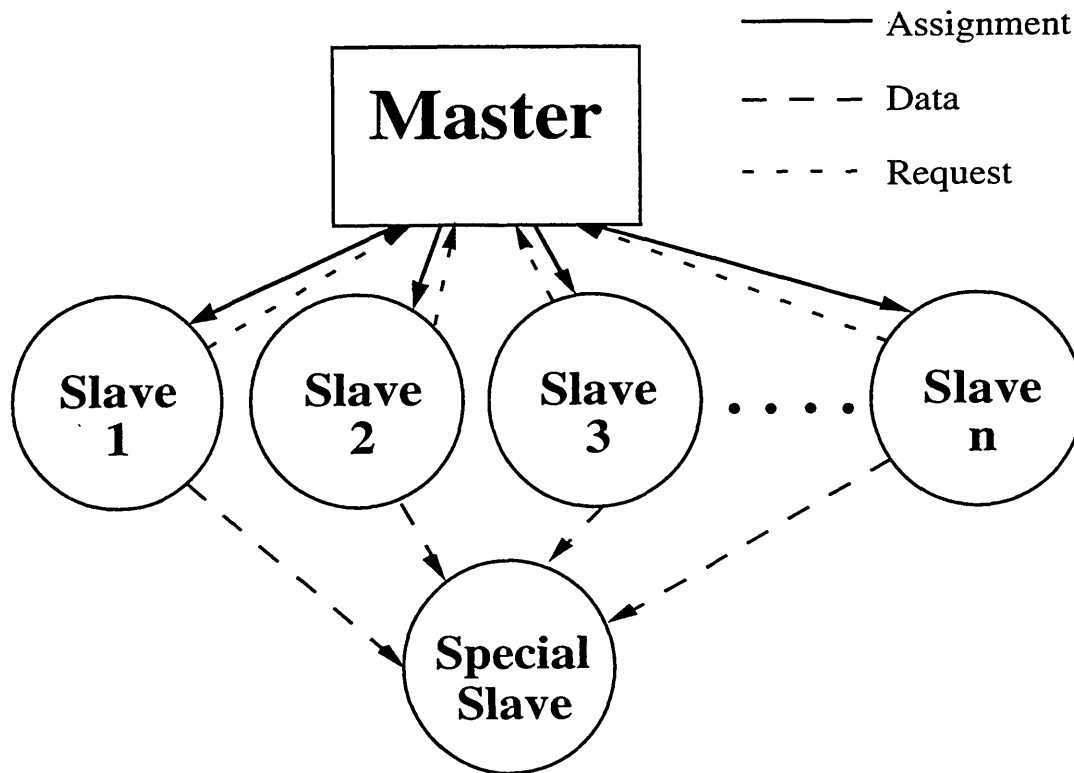


Figure 4.2 Architecture #2

slaves do not complete their work at the same pace. The array was not wide enough to compensate. The data would wrap around to the incomplete time step, and begin to fill that section of the array with data from the new time step. This situation would cause the special slave to compute a navigation solution using data from two different time steps,

which is unacceptable. Increasing the width of the array delayed the onset of this condition, but it did not solve the problem.

The next modification was inspired by the fact that the number of satellites is small in comparison to the number of time steps that this simulation will process in a run. This change did not affect the architecture of the program, only the task assignments. The master is still operating the target satellite, but now the assignments it gives out are time steps. Each slave takes its time step and produces the data for all 24 GPS satellites. When it's finished, the slave passes its data to the special slave and reports for its next assignment. The special slave waits for the data for the next time step on its list, and processes the navigation solution as the data comes in. The navigation solution is time coded and placed in a file, and the special slave goes back to find the next time step. This change eliminated the wrap-around problem. There was some thought to passing the navigation solution responsibility on to the regular slaves, but they would not be able to deposit the solutions into the same file. Also, keeping the navigation solution algorithm separate makes it easier to debug or to modify the algorithm.

4.1.1 Advantages of this Architecture

The architecture presented above has several advantages in construction and usage. These advantages fall into two categories, simplicity and flexibility.

Simplicity is first seen in the construction of this architecture. The main structure consists of a program and three subroutines (Figure 4.3). The program initializes the MPI environment and establishes the identities of the individual processes and allows the processes to begin their respective subroutines [28]. The three subroutines correspond to the three types of processes that exist in this architecture: the master, the regular slave, and the special slave. Each process chooses its subroutine based on its identity; note that multiple processes may choose the slave subroutine while only one may choose either the

master or special slave subroutine. The communication is built into the subroutines and it's kept to a minimum. When execution of the subroutines is completed, the processes return to the main program, which closes the MPI environment and ends.

Modification of this architecture is also very simple. This feature is seen in the modification from Architecture #1 to Architecture #2 (Figures 4.1 and 4.2). To accomplish this change, the subroutine for the special slave was constructed and the new communication paths were established. This ease of modification also makes this architecture extremely flexible.

The flexibility of this architecture is not limited to the structure, but is also seen in its application. The computations required for the simulation in this thesis could be easily removed from the architecture and replaced with those needed for a different project. Or the current computations could be modified to increase the accuracy of the simulation. These computations would not affect the operation of this architecture, only the processing of the input data.

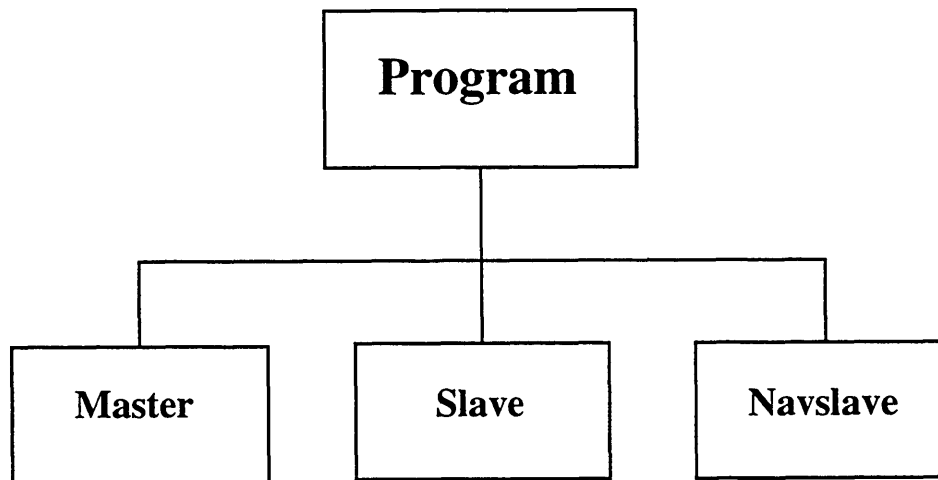


Figure 4.3 Program Structure

Another advantage of this architecture is the ease of operation and testing. Because this architecture was constructed to perform repetitive tasks in the master, slave, and special slave subroutines, these subroutines were each constructed based on a simple loop. The

structure can be easily tested before any of the computational elements have been added to the program.

4.1.2 Communications

As mentioned in the previous section, communication within the subroutines is kept to a minimum. By limiting the amount of communication, most of the run time is devoted to computation. All communications in this simulation are the simple, straight-forward, point-to-point communications described in the MPI functions [28]. The following paragraphs describe the communications from the point-of-view of each subroutine.

The master sends the epoch time and the total number of time steps to the special slave. Then, it sends the initializing data and the initial assignments to the other slaves, and begins its loop. Next, it waits to receive a request for tasking from one of the slaves. The master sends the next task, or a shutdown command if the tasks have all been assigned. Then, it waits for the next request. If the master has distributed all assignments and commanded all slaves to shut down, it returns to the main program.

The regular slave accepts the initializing data, begins its loop, and waits to receive its message. Upon receiving the message, the slave reads it to determine if it is an assignment or a shutdown command. If it is an assignment, the slave performs its task, sends two messages to the special slave, sends a request to the master for its next task, and waits for the response. If the message is a shutdown command, the slave returns to the main program.

The special slave waits to receive the epoch time and the total number of time steps from the master. Next, it begins its loop and waits for a message from a regular slave that matches its current time step; this first message tells the special slave how much data will be arriving in the second message. After receiving the first message, it waits for the second message with an appropriately sized array. When the second message arrives, the special

slave processes the data, deposits the results in a file, and begins to wait for the first message for the next time step. When it has completed the final time step, it returns to the main program.

In an earlier iteration of the simulation, the master subroutine made use of the broadcast function to transmit the initial time and time step size to all other processes. This course was originally taken based on the assumption that the simulation would be processing evenly spaced time steps during every run. This assumption need not be true, so the broadcast function was removed to allow greater flexibility in the time assignments.

4.2 Interaction between Architecture and Placement of Computational Segments

The placement of the computational segments of the simulation is partially dictated by the architecture, but the reverse is also true. The final architecture of the program was not selected until some of the basic computational segments were installed. These few segments gave an idea of the performance characteristics of the structure, which were for the most part satisfactory. But these characteristics also steered the evolution of the architecture to its final stage.

The placement process began with the selection of the master-slave concept for the architecture [28]. For this type of architecture, the most desirable results come from keeping the slaves as busy as possible. Because the master is the one that assigns the tasks, it should be ready to provide the next assignment as soon as the slave makes the request. For this reason, the master should have as few computational responsibilities as possible.

So, how much responsibility should be given to the slaves? That was one of the first questions to arise in the placement process. The answer did not come easily. As stated in the previous sections, the initial tasking for the master was assigning the tasks and

processing the navigation solutions. The delay caused by computing these solutions would have meant a bottleneck as the slaves completed their tasks and came back to find the master unavailable. The slaves would not be able to start work again until the master had finished its tasks and started assigning tasks again. The master can only process one request at a time, so each slave would have to wait its turn. This performance was unacceptable from the standpoint of keeping slaves busy and the computational segment for the navigation solution had to be reassigned.

At this point in the development, the slaves were responsible for the processing of individual GPS satellites and would not be capable of taking on the navigation solution task. So, the special slave was designed to perform this task and remove the responsibility from the master.

The next question is basically a subquestion of the first: which subroutine should process the target satellite's position? The two alternatives were the master and the slaves. If the master processed the satellite, the calculations would be performed once for each time step and the data would be passed to the slave in its assignment. This procedure might have caused the master to be detained from assigning tasks. If the slaves processed the target, the calculations would be performed each time a task was started and no satellite data would need to be passed to the slaves. But the multiple processings for the same result seemed to be a waste of computer time. At that time, the target processing was given to the slaves.

The next step in the evolution occurred as a result of the wrap-around problem. This situation was anticipated and the special slave was given an array for storage of these premature inputs. But the array was not wide enough and the data began to wrap around to the incomplete time steps. The array was made wider, but this only delayed the problem. The solution to this problem, changing the assignments from individual GPS satellites to complete time step, reinforced the decision to place the target satellite's calculations into the slaves. This change also reduced contact with the master to once per time step. It also

simplified the data acceptance for the special slave. This increase in assignment size does not affect the architecture of the program, but it does cause a minor change within the slave process, the inclusion of a loop.

The following sections present the responsibilities and the internal structure of the individual processes as they appear in the completed simulation. They describe the roles of each process in the master-slave-special slave architecture, and briefly mention the programs used in their construction. In these sections, the special slave is referred to by the name of its program, navslave. The programs that were created for these structures were designed to be as simple and straightforward as possible; they were written in Fortran to make them compatible with the preexisting subroutines that are used in the simulation.¹ These programs are described in detail in Appendix A.

4.2.1 The Master's Task

The master subroutine has an extremely light load to handle. By limiting its responsibilities to a bare minimum, the master has no excuse for any delays in providing new assignments to the slaves. Its responsibilities consist of:

- acquiring the data for initializing the simulation
- relaying the necessary data to the other processes
- assigning the initial tasks to the slaves
- monitoring the assignments load
- waiting for an assignment request from any slave
- replying to that request

¹ MPI supports programs in Fortran and C computer languages.

At the current time, the time distribution is restricted to uniform time steps from an initial start time, so the master has the added task of computing the time that is sent in the assignment.

The first three responsibilities of the master are its initialization procedure and only occur once in the course of a run. It acquires the initializing data from a subroutine called STARTUP. The data consists of the Julian date of the epoch of the simulation, the time since epoch of the first signal reception, the length of a time step (in seconds), the total number of time steps, and the input type identifier. The Julian date and the total number of time steps are sent to the navslave; the Julian date is used in the time tagging of the observations, and the number of time steps indicates the number of messages to be received. The time of first signal reception and input identifier are sent to the remaining slaves; the time of initial reception is needed for the operation of the satellite clock subroutine, and the input identifier specifies the means by which the target satellite's states will be calculated (DSST or Cowell). The initial assignments present the time steps to the slaves in numerical order, which is different from the first come, first served policy in the loop. These assignments are made before the master enters the loop, and then the policy takes over.

A majority of the master's time is taken up with waiting. It waits for the slave to send its request so that the slave does not wait. Once it receives a request, it checks to see how many assignments have been given out. If there are still some assignments left, the master computes the next time for the assignment, and sends the assignment. If no tasks are left, the master sends a message to shut down. Then, it updates the records and returns to its waiting.

As stated, the master is not terribly taxed. In fact, it does not contain any of the computational segments that were created for this simulation. In the course of its operation, it only calls three subroutines and two of those are the MPI communication functions, MPI_SEND and MPI_RECV (Figure 4.4). The remaining subroutine,

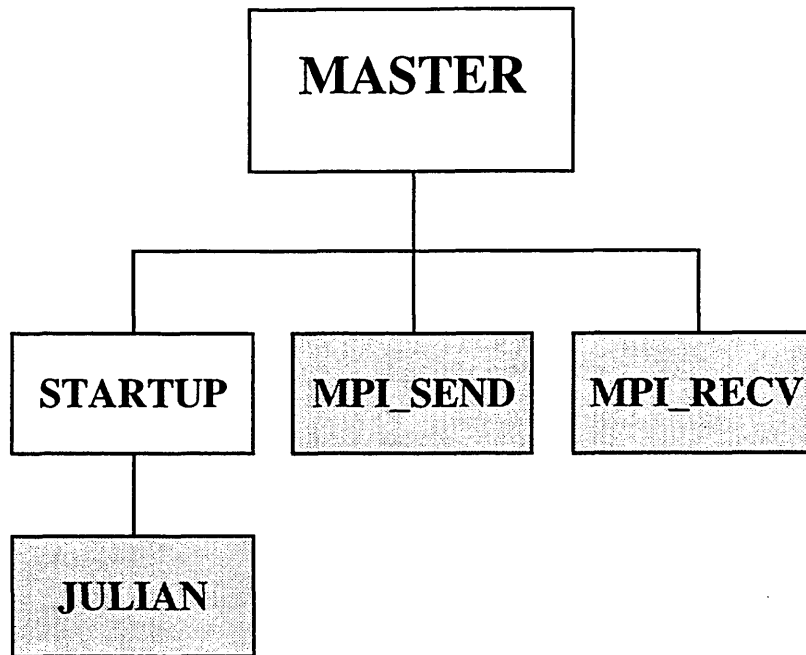


Figure 4.4 Master Subroutine Structure

STARTUP, establishes the data described earlier. Within STARTUP, there is only one subroutine called. That subroutine, JULIAN, converts the epoch to its Julian date for use in the special slave. The only new programs in this structure are the MASTER and STARTUP subroutines, indicated by the white boxes in Figure 4.4. The shaded boxes indicate preexisting subroutines, which may be connected to other subroutines from GTDS.

4.2.2 The Slaves' Work

The slaves' work is never done. The slave subroutine is the most heavily tasked subroutine in the simulation. It is responsible for the entire process of producing the perceived times of transmission. The slave's responsibilities include:

- initializing its variables
- accepting the initialization data
- receiving its assignment from the master
- determining the target's position and velocity and the position of each GPS satellite
- testing each GPS satellite's position at its time of transmission for acceptability
- computing the satellite clock errors for each acceptable GPS satellite
- modifying the times of transmission to include the effects of the clock errors
- notifying the navslave of how much data it is about to receive
- passing the data to the navslave
- resetting its memory locations
- requesting its next assignment

Basically, the slave is responsible for operating all of the computational subroutines necessary for the simulation of GPS signals. It is quite apparent that the slave has a very full plate.

Most of the slave's responsibilities are addressed through the calling of subroutines (Figure 4.5). The main jobs of the slave subroutine are to call the computational subroutines, monitor the loops for completion, store the appropriate data, and communicate with the other major subroutines.

The operation of the slave is straightforward. Before beginning its primary loop, it initializes the necessary variables and data structures with `INIT_CLOCK` and `INIT_SLAVEINFO`, and it accepts the input identifier and initial reception time with the `MPI_RECV` subroutine. Within the main loop, it receives its message via `MPI_RECV`. If the tag on this message is zero, the slave shuts down. Otherwise, it takes the time from the message and starts its task. First, it determines the target's position at this time (`TARPOS`) and uses this time and position to find the time of transmission for each GPS satellite

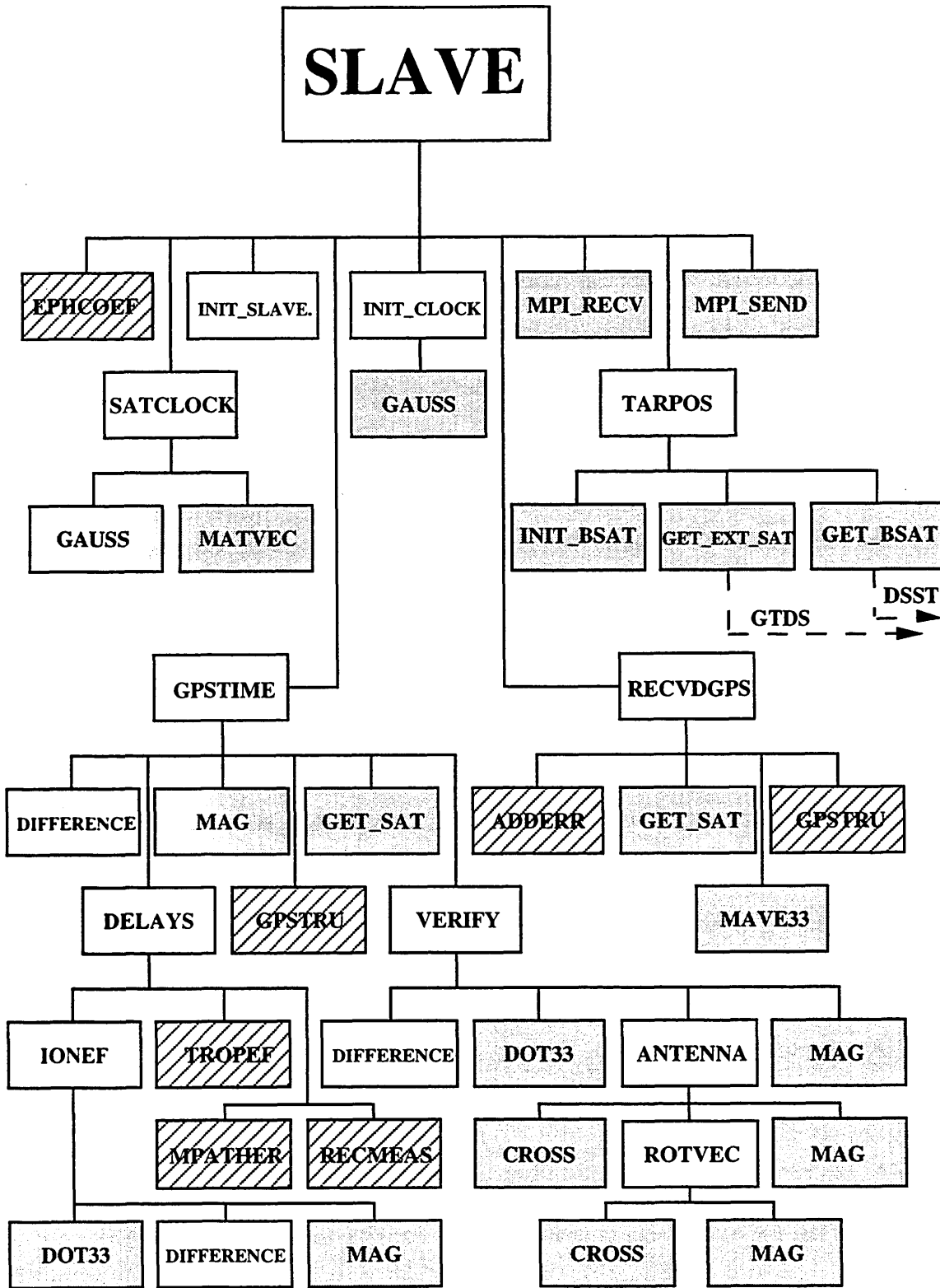


Figure 4.5 Slave Subroutine Structure

(GPSTIME). As part of this calculation of transmission time, each GPS satellite is evaluated for two-way visibility with the target satellite (VERIFY). The slave computes the satellite clock error (SATCLOCK) for each GPS satellite that is accepted and alters the transmission times accordingly. Depending on the ephemeris case, the slave either calculates the GPS ephemeris position (RECVDGPS) and adds errors to it (ADDERR) or determines the coefficients necessary to calculate it in the navigation solution algorithm (EPHCOEF). Once this sequence is complete for all 24 GPS satellites, the slave sends the navigation solution data to the navslave subroutine, resets its variables, sends a request for the next task, and returns to the beginning of the main loop.

The subroutine structure is quite involved. In fact, not all of the subroutines within the slave subroutine have been mentioned here, but they are all displayed in Figure 4.5. Markings are consistent between the figures. The boxes with diagonals represent subroutines that do not exist yet, but will be created during an upgrade of the simulation.

4.2.3 The Navslave's Job

The last process is the navslave. This slave is also kept quite busy, but only with the operations associated with the calculations and output of the navigation solutions. Its name was derived from navigation slave. It is responsible for the following tasks:

- initializing its variables
- preparing a working file
- accepting the epoch time and number of time steps from the master
- receiving the navigation data from the slaves
- if there is a sufficient amount of data, processing the solutions and storing them in a file

This is not a significant load in comparison with that of the other slaves, but if several regular slaves are operating, the navslave is still kept very busy.

Once again, the first few responsibilities involve the setup of the process and are handled before the navslave enters its main loop. First the variables are initialized using the subroutines INIT_HEADERS and INIT_NAVINFO. Then, a working file that is required in the file storage process is opened by using GETENV. And just before entering the loop, the navslave receives (MPI_RECV) the epoch data and number of time steps. The epoch data is needed to produce the time tags for the output file, and the number of time steps sizes the main loop. These actions are very important to the performance of the navslave, but they only need to be done once.

The navslave is required to do some waiting. The length of the wait is determined by the performance of the other slaves. Unlike the first come, first served policy followed by the master, the navslave must process its steps in sequence. It has to wait for the next task on the list, even if other tasks are available. So if an individual slave is operating particularly slowly, later time steps could be piling up before its current task is complete.

Within the loop, the navslave receives (MPI_RECV) the data from a slave. It tests the data to determine if a sufficient number of visible GPS satellites are available for the computation a navigation solution. If there are enough, it identifies how the a priori estimate of position should be performed and passes the data to the navigation algorithm (NAVFIX). When the algorithm finishes, it advances the loop and starts again.

The NAVFIX subroutine is the primary functional component of the navslave. It takes the navigation data and, if necessary, computes the ephemeris positions of the GPS satellites. Then, it selects the four GPS satellites (SELECTSAT and GEOCHECK) or extrapolates from past data to determine the a priori estimate for the navigation solution (GMATINV). This a priori estimate is used, along with all of the navigation data, to make the geometry matrix. A vector of expected changes in pseudorange is also constructed. The pseudo-inverse of the geometry is calculated using the Moore-Penrose Generalized

Inverse (GMATINV). The change in the position and time bias vector is the product of the pseudo-inverse and the vector of pseudorange changes (MATVEC). This process is repeated for the construction of the geometry matrix until magnitude of the position and time bias change is below a specified tolerance. Upon completion of the navigation solution, it is time coded and stored in a file (OUTNAV). Additional data, including the solutions range, time bias, and GDOP, are stored in a second file for statistical purposes.

The structure of the navslave is shown in Figure 4.6. A majority of this structure consists of the NAVFIX substructure, which should be expected from the previous discussion. The box shaded in the diamond pattern, GMATINV, is not really a new or preexisting program. It is a modified version of the program MATINV. The modifications to this program are presented in Appendix A.

4.2.4 Structured Data Files

In addition to the programs that were created, a small set of structured data files was also constructed. These files were designed to assist with the internal data management of the processes. Because the master was so limited in its computational load, it did not appear to need one of these files. The slaves and navslave each have a data file for the frequently used variables in their lower subroutines.

Access to these files is achieved by using an INCLUDE statement for the desired file within the subroutine and calling the variable name preceded by the file name and a period. For example, in the SLAVEINFO.H file there is a variable, INPUTYPE, which can be retrieved by typing SLAVEINFO.INPUTYPE in a subroutine that includes the file. For constants, the process is much simpler. Constants are saved as parameters and can be called by their name alone if the file is included in the subroutine.

An additional structured data file was included in the construction of the slave process. The CLOCK.H file is used strictly for the maintenance of the 24 GPS satellite

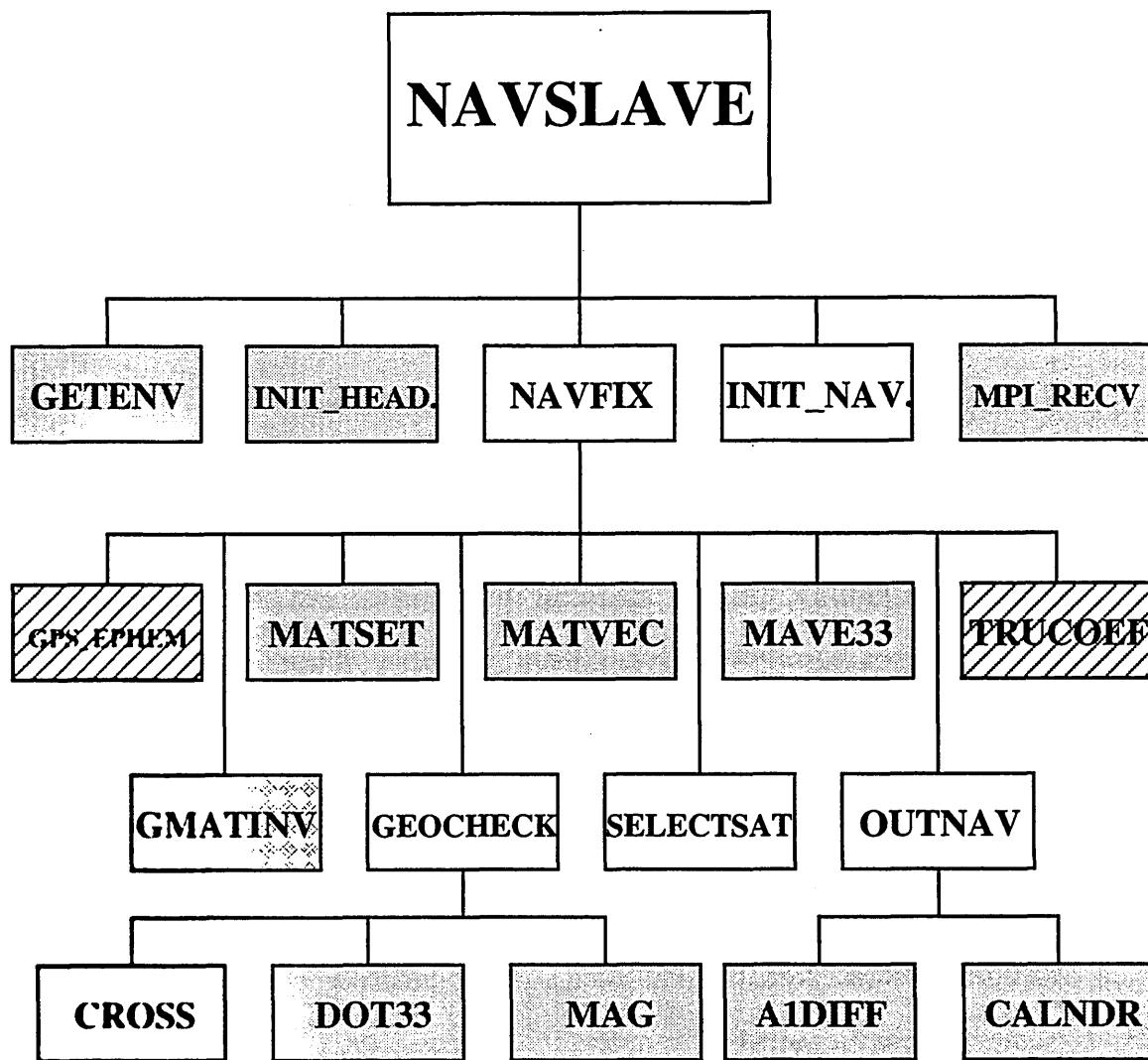


Figure 4.6 Navslave Subroutine Structure

clocks. The design of these files is a convenient way to store and update the variables associated with the clocks' upkeep. The SATCLOCK and INIT_CLOCK subroutines are the only programs that have access to this file.

4.3 Graphic Processing

The architecture presented above is used as the framework for the simulation, which produces simulated navigation solutions from a spaceborne GPS receiver. These

navigation solutions are used as the input for an orbit determination process. The behavior of navigation solutions and the determined orbit both must be examined.

To better interpret the output from the simulation and the orbit determination process, it is desirable to have the data presented graphically. The primary tools available for the production of graphics from this data are Microsoft Excel and Matlab.

Excel was primarily used in the early stages of simulation testing. A smaller amount of data was being produced and could be easily ported into Excel. In addition, the calculation capabilities of Excel were used for some minor processing of data for the included images.

Matlab is the primary tool for producing the graphics used for interpreting the results. One of the main reasons for its use is that it is capable of handling the volumes of data being produced in the simulation and orbit determination processes. Another reason is its programmability. Short programs can be written that will perform calculations on the data and output the results in sets of figures. This feature is important for the comparisons between the simulation input and the output of the orbit determination.

Most of the output from the simulation is stored in tabular form in a collection of files. The data in these files include: the number of satellites in view at all times during the simulation, the target satellite's actual position, the times of accepted navigation solutions, their times since epoch, the number of satellites used in the navigation solution, the radial distance to the navigation solution, the calculated time biases, the GDOPs, and the geometry parameter used in the selection of the a priori position estimate. Tabular information can easily be presented for interpretation; plotting on Matlab is a simple process of identifying the columns of a table to be plotted, which Matlab perceives as the X and Y components of a graph. So, any functional relationships between these values can be visualized very quickly.

The output from the orbit determination process and its comparison with the simulation input is a different story. First of all, the files containing this data are

incompatible with Matlab. Before the files can be loaded into Matlab, all text must be removed from them. This task is accomplished through the use of a stripper utility that was designed for this purpose. Once the files have been loaded, a Matlab program compares them and produces figures that show the component level differences between the input and the output over a period of time. Depending on the program, these components could be Cartesian components, orbital elements, or radius, cross-track, and along-track (RCA) components. The period of time may be continuous through the all the data, or it may be broken down into the fit and predict spans. These graphics provide a simple means of evaluating the quality of the orbit determination from the simulated navigation solutions.

4.3.1 Stripper Utility

The stripper utility was designed to remove the POSITION and VELOCITY tags from the ASCII files containing the simulation input and the orbit determination output. While removing the tags, the utility also reorganizes the data for more convenient use by Matlab.

To operate the utility, the desired ASCII file must be copied into a generic file identified in the stripper's allocation process. Once there, the file must be trimmed until only the lines with POSITION and VELOCITY tags are present. Finally, a file END marker must be placed after the last VELOCITY entry. Now, the file is ready for the stripper.

The stripper utility reads the first line of the file and enters a loop. Upon entering the loop, the stripper tests the alphanumeric field of the line to see if it is the END marker. Satisfied that the END has not been reached, the utility reads the next line. The alphanumeric fields are disregarded and the position and velocity vectors are written, in that order, on a single line of another generic file. The stripper reads the following line and returns to the beginning of the loop. This procedure continues until the END is reached.

As stated in the preceding paragraph, the position and velocity are stored on the same line of the output file. By storing them in this manner, the array does not need to be resized within Matlab and operations can be performed immediately.

This utility was written to process ASCII files with a specific format. It will only work properly on files that have a series of double lines of related data, with the line format of a length-eight alphanumeric field following by three numerical fields of size 25.

Chapter 5

Orbit Determination Error Analyses Results

The goal of this project is to evaluate if GPS navigation solutions can be used to as the sole input for orbit determination of medium altitude, eccentric orbits. To compensate for the lack of actual navigation solutions for these orbits, a program was constructed to simulate these navigation solutions.

Figure 5.1 presents the complete data flow for the simulation and orbit determination process. The “truth” orbit for the target satellite is generated using the process described in Section 3.1. This “truth” is used as input to the simulation of the navigation solution and as the actual satellite orbit for comparison with the orbit determined from the simulated navigation solutions and with the navigation solutions themselves. The algorithms and assumptions used in the simulation are explained in Section 3.2. The orbit determination process and the comparison of the determined orbit with the “truth” are presented in this chapter. The performance of the orbit determination process is examined for different target orbits: Ellipso Borealis (both planes), Ellipso Concordia, and Molniya. The performance is tested using the ideal force models, and with the forces mismodeled; this is done to test the process’s sensitivity to these model errors.

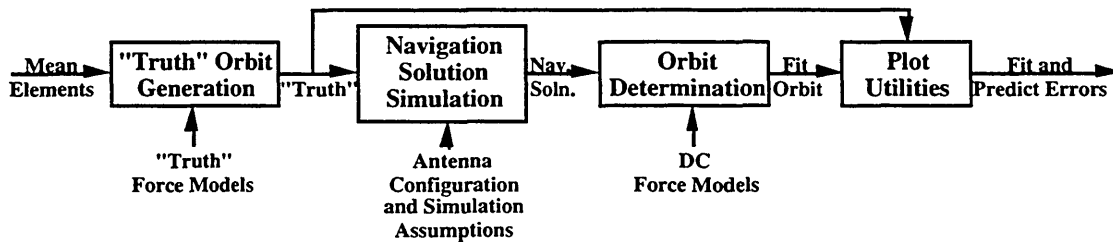


Figure 5.1 Deterministic Analysis Data Flow

Before performing the evaluation of the orbit determination process, the methods used to simulate these navigation solutions must be validated. This validation is accomplished in two steps: validation of the component algorithms and validation of the complete simulation. The validation process is also presented in this chapter.

5.1 Validation of the Simulation

Before using the navigation solutions in the orbit determination process, the simulation was tested to verify proper performance. This testing was conducted on the component level, on the simulation as a whole, and over the full analysis process.

The first set of tests focused on the performance of the individual algorithms. If these algorithms do not produce the proper results, the simulation would be of little value. The algorithms that were validated are the satellites in view algorithm, the satellite clock algorithm, and the navigation solution algorithm. To provide a wide range of altitudes to the simulation, the Ellipso Borealis orbit with its ascending node at noon was used as the target orbit while performing these early tests [14,15]. The Borealis orbit is a critically inclined, sun-synchronous orbit with a semimajor axis of 10496.882 km and an eccentricity of 0.332088. The epoch selected for the simulation of the Borealis orbit was 1:48 AM April 21, 1999.

After the algorithms have been evaluated, the performance of the entire simulation is tested. The simulation's performance is tested by comparing the simulated navigation solutions, with all errors included, to actual navigation solutions. To perform this test, the simulation was run using TOPEX as the target satellite in a time interval for which actual navigation solutions were available [8].

The following sections describe the validation process on the component algorithm, followed by the validation of the whole simulation. Each section provides a description of the criteria used in its validation.

5.1.1 The Satellite in View Algorithm

Section 3.2.2 describes the algorithm that identifies the GPS satellites in view. Since a minimum number of GPS satellites is required for the purposes of the navigation solution, it is important to know that this algorithm is performing as intended.

At different altitudes, there is an expected performance of the satellite in view algorithm. These expectations are based on the geometry of the situation, the number of GPS receiver antennas, their orientation on the target satellite, and their reception angles.

Using the standard single antenna configuration, facing in the zenith direction, the target satellite can expect to see the most GPS satellites at low altitudes. Also, at these lower altitudes, any GPS satellite within the line of sight of the target can send a signal toward the target. As the target gains altitude, it reaches a point (approximately 3270 km altitude) where the previous statement no longer applies (Figure 5.2). Beyond this altitude, a band develops in the sphere of the GPS constellation; any GPS satellite within this band cannot transmit a signal to the target satellite because the target is outside of the transmission cone. This band expands as the target continues to climb. At one point in this expansion, the only GPS satellites above the target that are capable of sending a signal to it are the ones within the target's reception cone; for an antenna with a reception cone half-angle of 70° , this point occurs at an altitude of approximately 3890 km, only 620 km above the line of sight boundary (Figure 5.2). But signal reception requires a two-way view. The locations from which the target can receive signals are restricted to within the projection of the reception cone onto the sphere of the GPS constellation. This region

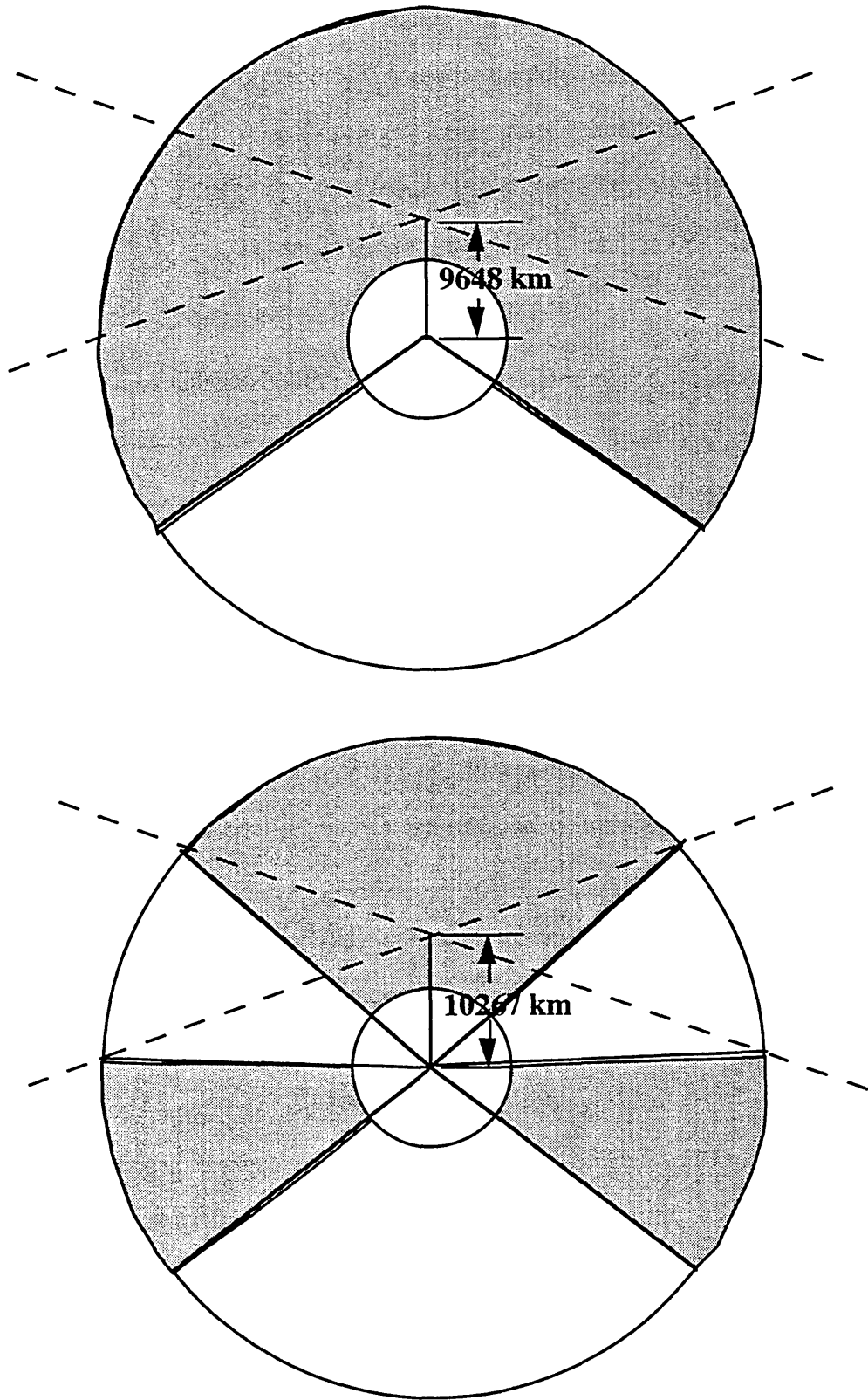


Figure 5.2 Boundaries for 70° Antennas

shrinks as the satellite climbs and approaches the GPS sphere, but at a slower rate than the band grows. Both of these behaviors contribute to the decline in the number of GPS signals received as the target climbs.

For a nadir facing antenna, low altitude means a lack of GPS contact. The Earth is in the way of the GPS signals. Nadir facing antennas only start to become effective once the satellite has climbed high enough for the reception cone to clear the limb of the Earth. For the 70° reception cone half-angle, this translates to about 410 km altitude. The same characteristic behavior of GPS viewing the target occurs as with the zenith facing antenna, but the region of reception for the nadir facing antenna expands as the target climbs. The vertex of the cone is moving away from the GPS sphere and the shadowing Earth, so the area of the projection grows. As the target continues to climb, the reception area reaches the edge of the band described in the previous paragraph (Figure 5.2). Again, the only GPS satellites capable of sending a signal from below are within the cone of reception. Using the same antenna, this transition occurs at the same altitude as it does for the zenith facing antenna, 3890 km; this is the best altitude for GPS reception for a nadir facing antenna. As the target satellite continues to climb, the region of reception begins shrink. But the rate of area reduction is slowed by the reduction of the size of the Earth's shadow.

Using two antennas, with one facing in each of the zenith and nadir directions, results in a combination of the two effects. The poor performance of the nadir antenna at lower altitudes is compensated by the zenith antenna's best performance, and vice versa for higher altitudes. At medium and higher altitudes, teamwork between the antennas may eliminate instances when neither of the antennas could acquire enough GPS signals for the navigation solution individually. Throughout all the altitudes of interest in this thesis, navigation solutions available through the use of one antenna can be improved by using additional signals from the other. Between the teamwork and occasions when each could produce a navigation solution individually, the combination of the two antennas may actually provide more or less navigation solutions than the sum of the solutions from each

antenna. For this discussion, the two antennas are assumed to be collocated. Considering the size of satellites and the large distances being described, this assumption is reasonable.

From this analysis of the factors affecting the expectations, the satellite in view algorithm should produce the following results:

- Zenith facing - Larger numbers of GPS satellites in view at the lower altitudes with a steady decrease in number as altitude increases
- Nadir facing - Least number at low altitudes, increasing as altitude rises, cresting at approximately 4000 km, and decreasing beyond
- Both - Sum of other two, with higher altitude for last possible navigation solution

The results of the satellite in view algorithm are presented in Figure 5.3 for zenith, Figure 5.4 for nadir, and Figure 5.5 for the combination of the two. The visibility of the GPS satellite in the algorithm matches the predicted behavior. The maximum number of satellites for the nadir and combination cases are achieved at an approximate radius of 10300 km from the Earth's center, which is very close to the height predicted in the analysis. Ellipso Borealis was an excellent choice for this test because its range of altitudes covered the location of maximum satellite reception for all three antenna configurations, as well as their decline after the maximum.

The Ellipso Concordia is another orbit that could be considered in this analysis, but the orbit has such a small variance in altitude that it would produce little data. Concordia is a nearly circular, nearly equatorial orbit with a semimajor axis of 14428.138 km and an eccentricity of approximately 0. This orbit cannot use a zenith facing antenna for computing the navigation solutions unless it is accompanied by a nadir antenna. The zenith antenna receives only one or two signals from GPS satellites at this altitude, but these additional satellites could provide the last signal necessary for the determination of the navigation solution or improve the navigation solution from the nadir antenna markedly.

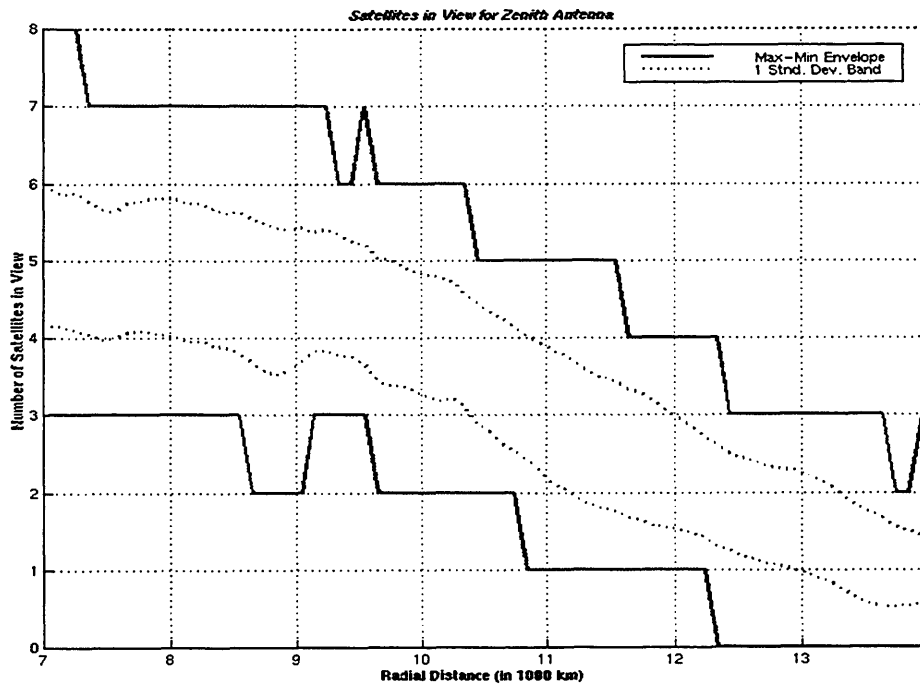


Figure 5.3 Number of Satellites in View for the Borealis, Node at Noon (zenith antenna)

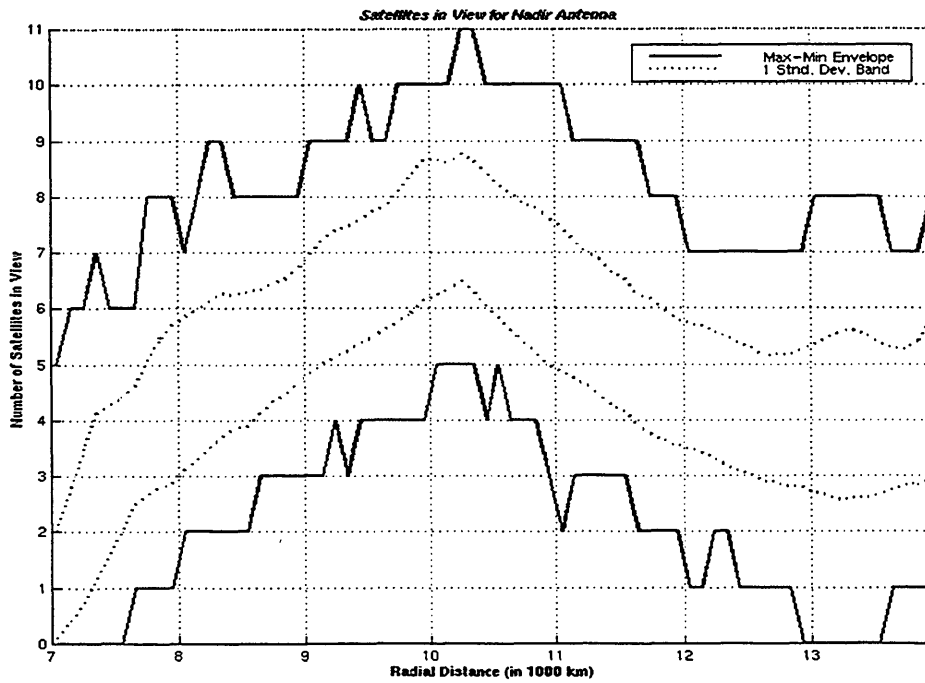


Figure 5.4 Number of Satellites in View for the Borealis, Node at Noon (nadir antenna)

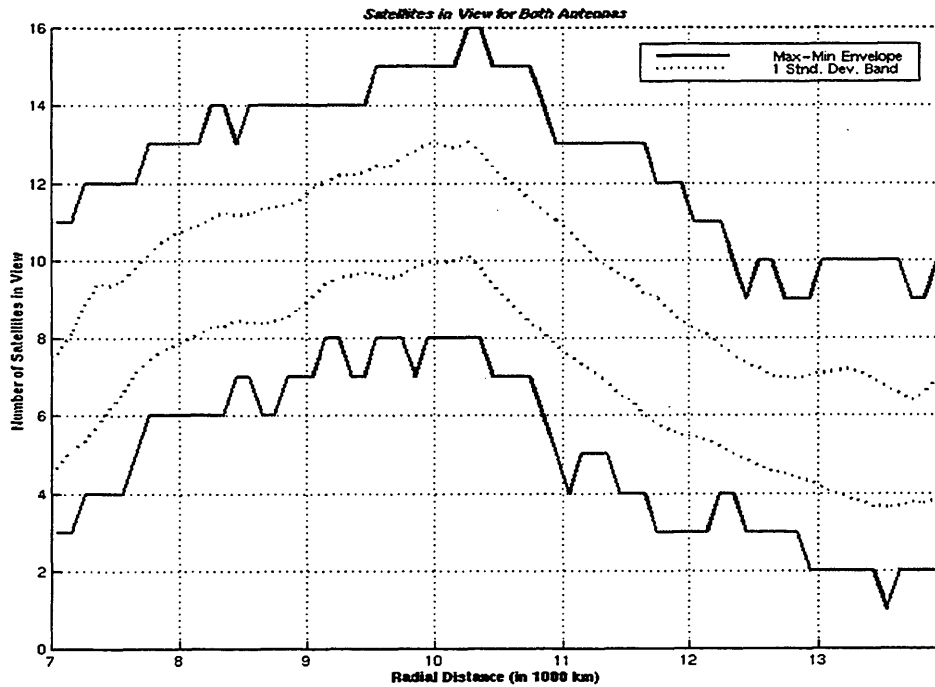


Figure 5.5 Number of Satellites in View for the Borealis, Node at Noon (both antennas)

Another feature included in the satellite in view algorithm is the ability to identify an altitude below which no signals may pass. This ability was used to reduce the effects of ionospheric delay on the navigation solutions. The function is activated by identifying the altitude to be blocked. The simulation treats this altitude as a soft boundary. This means that if the target satellite is below the blocked altitude, it may look up through the boundary; the target can never look down on a path that passes below that altitude. To block the ionosphere, the altitude is set to 1000 km, the maximum altitude of the ionosphere in the simulation. By eliminating a downward view through the ionosphere, the worst ionosphere delays are the same as those experienced by a ground based receiver. The blockage reduces the number of navigation solutions that can be determined, but the quality of the ones that can be computed is far superior. This function could be designed into an algorithm for use in space qualified GPS receivers that are implemented with a nadir facing antenna.

5.1.2 The Satellite Clock Algorithm

The satellite clock algorithm is presented in Section 3.2.3.1. This algorithm is an attempt to produce an error source that is statistically similar to the actual selective availability errors planted in the satellite clocks onboard the GPS satellites. Figure 5.6 shows a representative sample of the ranging errors associated with the satellite clock dither.

This algorithm was tested using a separate process driver to operate the clocks and collect their ranging errors. Using the recommended values for the damping coefficient, the natural frequency, and c^2 [54], the satellite clock algorithm produced the ranging errors displayed in Figure 5.7. This is only a sample from one of the clocks used in the simulation, but it demonstrates the similarity to the representative sample of the actual errors. The simulated errors are noisier than the actual SA errors, but that behavior was expected from the Gauss-Markov model [54].

5.1.3 The Navigation Solution Algorithm

The navigation solution algorithm was tested to determine the level of error associated with the least squares fit process. To isolate these errors, the computed error sources were turned off. The satellite clock errors and the ionospheric delays were overridden and set to zero. The navigation solutions from the simulation were compared with the target's "truth" positions.

In order to perform the comparison between the "truth" and the least squares fit, the "truth" file had to be modified. The "truth" file presented all of the "truth" positions on the simulation time grid, but the simulation rejected time steps based on the number of GPS satellites in view and the geometry of those satellites. A utility, MATCHUP, was constructed that removes the rejected times from the "truth" file to facilitate the comparison. Since the data in both the "truth" file and the navigation solutions file are in chronological

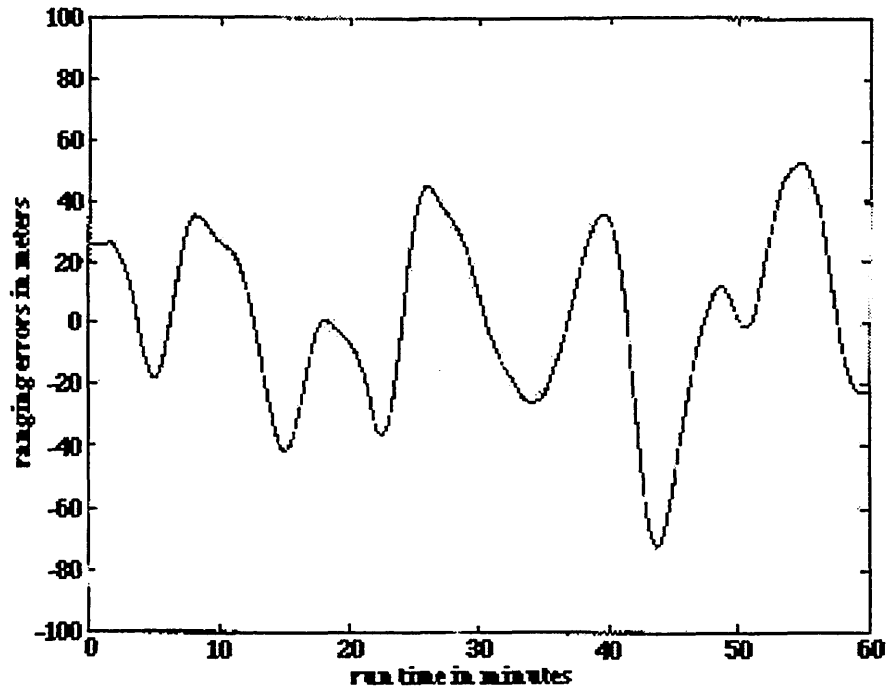


Figure 5.6 Ranging Errors from Satellite Clock Dither [54]

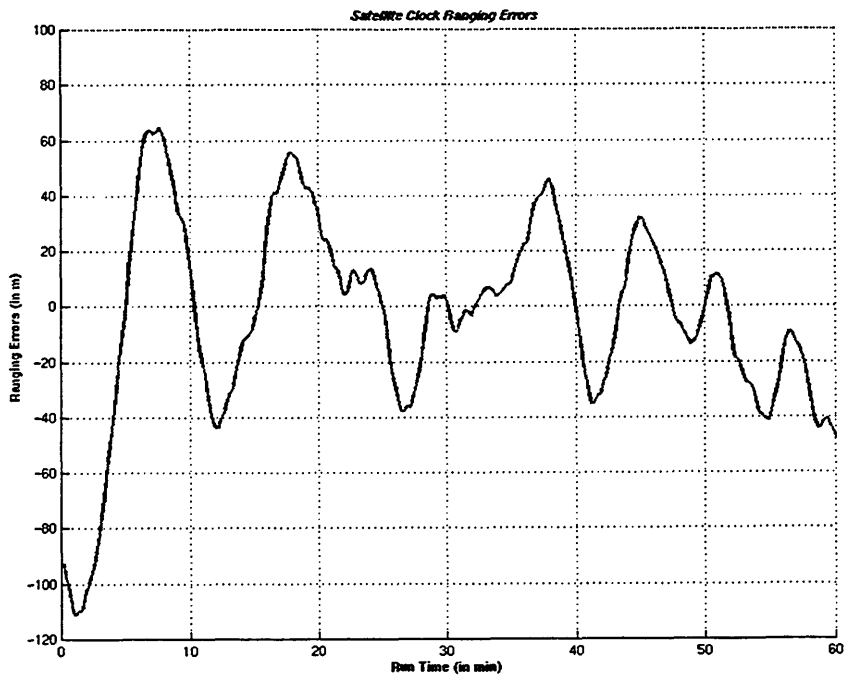


Figure 5.7 Simulated Ranging Errors

order, the utility reads the time from the navigation solution file and begins to read the times of the “truth” file until it finds a match. Once the match has been found, the utility writes that time and the corresponding position to a new file. This new “truth” file is used in the comparison with the navigation solutions.

The total position error associated with the least squares fit has a mean value of 1.5 cm and a standard deviation of 3.1 cm (Figure 5.8). However, the maximum error value in this comparison is on the order of 1.5 m. This maximum value and the other relatively high error values are a result of poor GPS satellite geometry. This conjecture is supported by a comparison of the time plots of the total position error and the GDOP factor (Figure 5.9). Each high error value is accompanied by a GDOP value that is well above the mean GDOP. These higher GDOP values also correspond with low numbers of GPS satellites in view. This observation confirms the belief that the use of more GPS satellites in the navigation solution improves its geometry characteristics.

5.1.4 The Entire Simulation

The simulation was tested by comparing its output to actual navigation solutions produced by the GPS receiver onboard TOPEX. The comparison of the simulated navigation solutions to the actual TOPEX navigation solutions was used to benchmark this first stage of the simulation’s development. It also provides continuity with the past efforts in orbit determination using GPS navigation solutions [8].

To perform this comparison, the simulation had to use the same epoch as the actual TOPEX data, which was on December 25, 1992 at midnight UTC time. The GPS constellation used in the simulation was not an exact representation of the GPS constellation at this epoch. It was an approximation of the constellation that had the same relative distribution and longitudinally independent coverage characteristics. Since TOPEX is a LEO satellite, this representative distribution should be sufficient for the purposes of

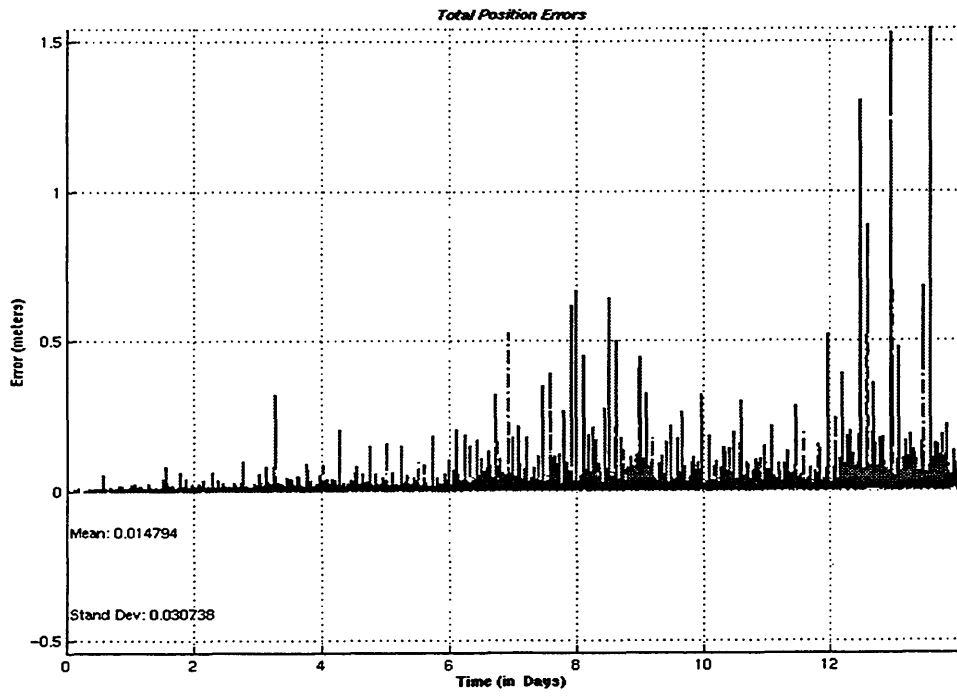


Figure 5.8 Total Errors in Navigation Solutions with No Simulated Errors

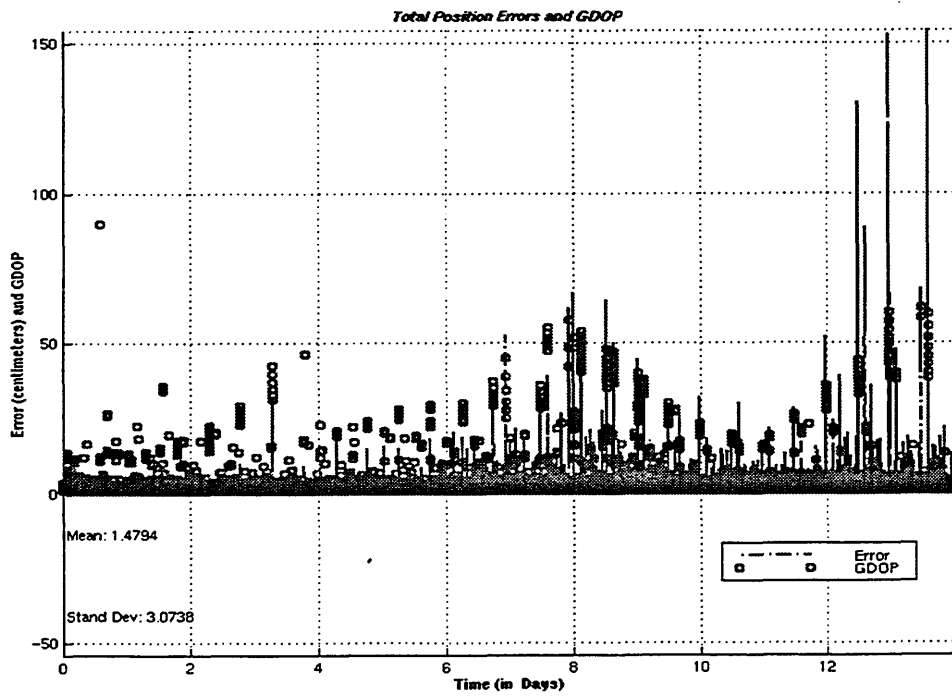


Figure 5.9 Comparison of Total Errors with GDOP

the simulation. The TOPEX “truth” orbit was produced by generating an ephemeris from the osculating elements that were determined in Carter’s processing of the precise orbit ephemerides (POE) [8]. The ephemeris was generated using the same force models that he used, so it should have been accurate to 1 meter.

Once the simulated navigation solutions were created, the MATCHUP utility was used on the TOPEX “truth” file. The comparison revealed that the simulated solutions had a higher mean and a smaller standard deviation, as well as a much lower maximum error value (Figures 5.10 and 5.11). There are four factors which may be contributors to these differences. First of all, the simulation does not contain all of the error models; only the satellite clock errors and ionospheric delays are included at this time. Second, the quality of the modeled errors could be improved. For instance, the ionosphere has been modeled with a uniform electron density rather than a density model that varies with altitude. Third, the GPS receiver used on TOPEX was only capable of handling six simultaneous GPS signals while the simulation produced solutions using all of the available signals; on several occasions, the simulation acquired seven signals, and eight signals twice. Finally, the simulation threw away potential solutions if the GPS satellite geometry was not satisfactory. All of these elements could have an effect on the statistics of the navigation solutions.

Even with these differences, the simulation produces navigation solutions that have a similar “look and feel” to the actual ones. Since this project is only evaluating the feasibility of using GPS for determination of higher, elliptical orbits, these results indicate that the simulation is adequate to perform this task. More stringent testing could be conducted using the actual GPS positions as the GPS “truth” orbits during this epoch; testing with the actual GPS positions would provide a more accurate assessment of the simulation’s performance.

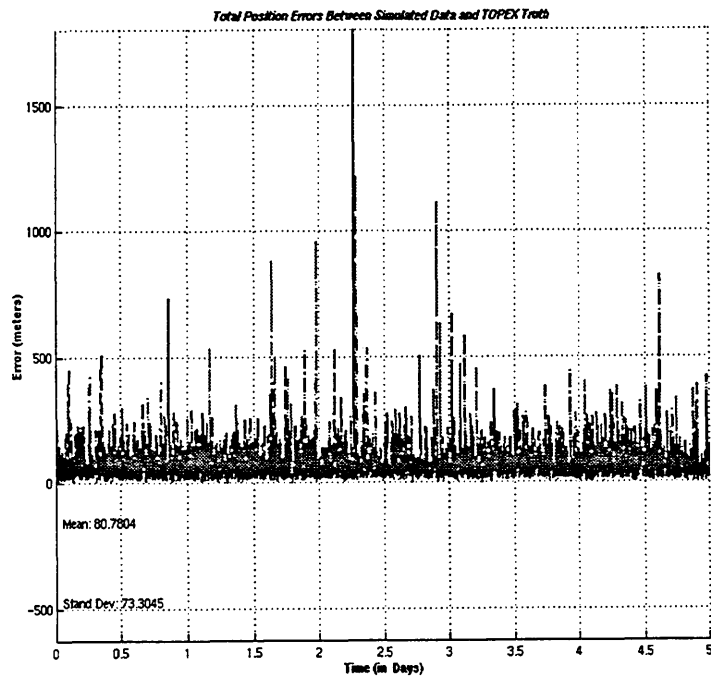


Figure 5.10 Comparison between Simulated Navigation Solutions and TOPEX “Truth”

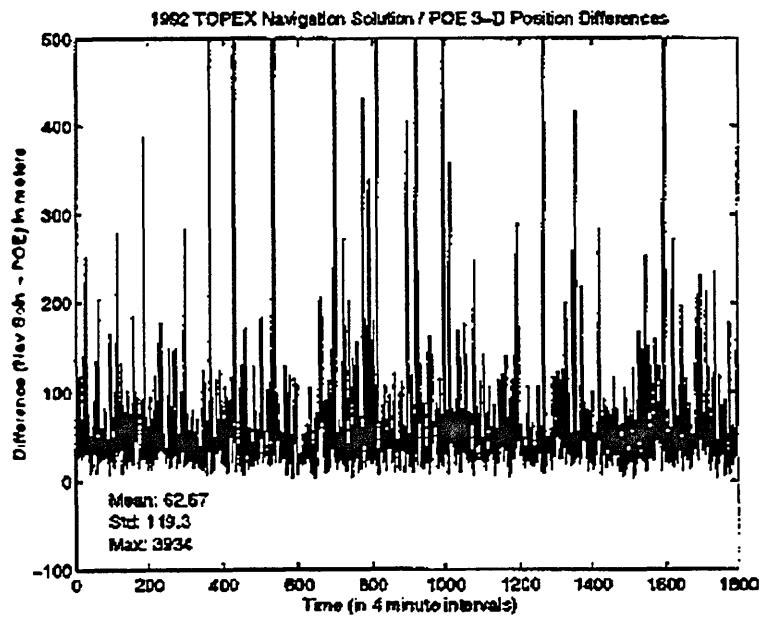


Figure 5.11 Comparison between Actual Navigation Solutions and POE

5.2 Orbit Determination (OD)

With the simulation's validation completed, the orbit determination process is ready to begin. But how well can the orbits be determined from GPS navigation solutions? Carter's work demonstrated that the navigation solutions could be used for orbit determination of LEO satellites, but these higher, elliptical orbits may be too different for the process to work.

The navigation solutions are used as the input for the orbit determination process. This process is performed using the Draper version of GTDS (Appendix B). To facilitate the use of GTDS, the navigation solutions are stored in an observation file format [75]. These solutions are position vectors only in the ECEF coordinate system.

There are two steps to the orbit determination process (Figure 5.12), the differential corrections operation and the ephemeris generation. The differential corrections (DC) operation performs the orbit determination. The DC requires an a priori state, which was chosen to be the initial state of the generated input. Using the a priori state as the first approximation, the DC fits the observation data using the weighted least-squares method. The output from the DC is the converged initial state of the determined orbit, which is used as the initial state of the ephemeris generation. The ephemeris produced consists of two parts, the fit span and the predict span. The fit span is the portion of the ephemeris corresponding to the time of the observations used to determine the orbit. The predict span is everything that follows the fit. The ephemeris generation stores both spans into a single ASCII file, like the one created in the generation of the target orbit. This file is ported to Draper's Data Center 1 (DC1) for comparison with the "truth" target orbit.

The steps in this process may use any of the orbit propagators available on GTDS. During this project, the DSST, the Cowell, and time-regularized Cowell propagators were

used at different times in the processing; in most instances, the propagator of choice was Cowell. But in each case, the same propagator was used throughout the process.

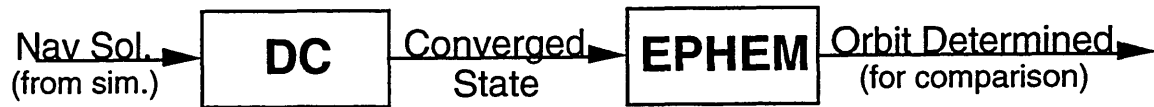


Figure 5.12 Data Flow of the Orbit Determination Process

In an attempt to reproduce the target orbits, the same force models were used in the orbit determination process as the ones used in the generation of the target orbit. Unless otherwise stated within the section describing a specific orbit, the previous statement should be assumed. In Section 5.3, the use of different force models is discussed.

To test the sensitivity of the orbit determination to different time considerations, three different observation files were generated for each orbit and antenna configuration. These observation files present the navigation solutions at three different measurement densities. These densities are defined by the amount of time between navigation solution opportunities: one minute, four minutes, and fifteen minutes. Also the portion of the observations that was actually read by the DC was varied from one to two weeks.

Before the orbit determination process was performed on any of the target satellites, there should be knowledge of the errors that can be attributed to the deterministic analysis process. For this reason, a validation was performed on the analysis process. In this validation, the analysis errors were isolated from the simulated errors and the errors due to mismodeling.

5.2.1 Validation of the Deterministic Analysis Process

To perform the validation of the deterministic analysis process, the “truth” orbit of a target satellite was run through the entire process without introducing any errors, from the simulation or from force mismodeling. The simulation was operated with the satellite clock

and ionospheric delay algorithms turned off. The orbit determination used the exact force models that were used in the production of the target's "truth" orbit. By not including these error sources, the remaining errors can be attributed to the normal performance of the deterministic analysis process.

To expedite the validation, the navigation solutions from the validation of the navigation solution algorithm were used as the input to the orbit determination process. The navigation solutions were processed from the Borealis node at noon "truth" orbit with the errors turned off for purposes of isolating the errors in the navigation solutions that were associated with the algorithm. These solutions had total position errors with a mean value of 1.5 centimeters and a standard deviation of 3.1 centimeters.

The orbit determination process was performed using the same set of force models used in the production of the Borealis NAN "truth" orbit. These force models are:

- Atmospheric drag using the Jacchia-Roberts atmospheric density model
- The JGM2 gravitational potential field with degree and order of 50
- Solar and lunar point masses
- Solar radiation pressure
- Solar and lunar solid Earth tidal effects

The epoch for this process was April 21, 1999 at 1:48AM UTC time. There are no data on the actual atmospheric density for this future time, so the input to the Jacchia-Roberts density model used a predicted and smoothed model of solar activity and geomagnetic indices.

The observations used in the orbit determination had a measurement density of one navigation solution for every four minutes, and a total duration of two weeks. This observation set was selected because the DC process would not be able to use all of the observations presented in the one minute density set for the entire two week period. The

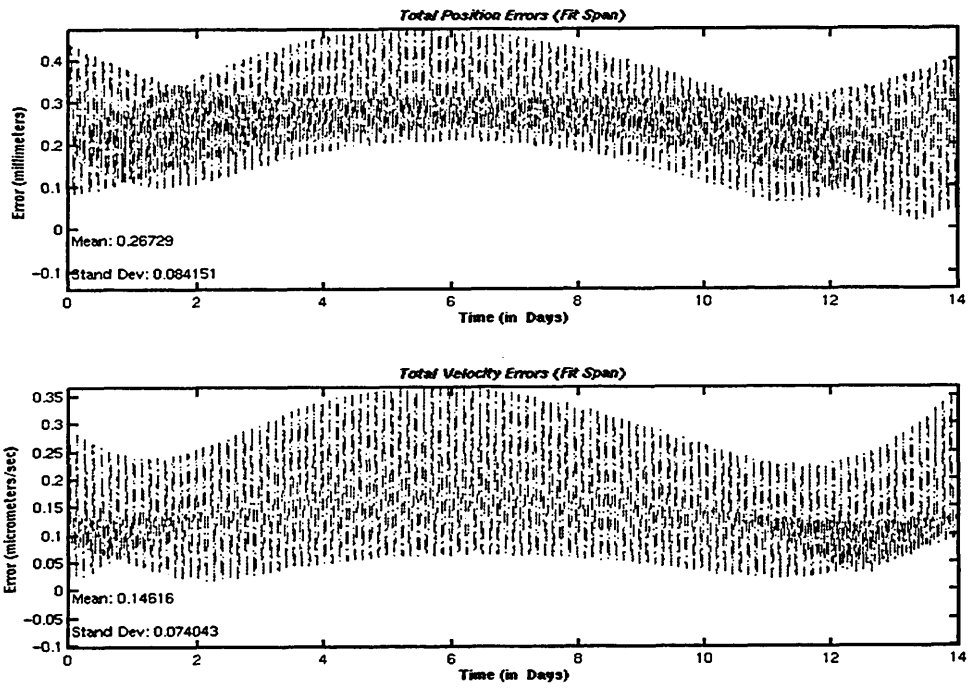


Figure 5.13 Errors in the Deterministic Analysis Process (Fit Span)

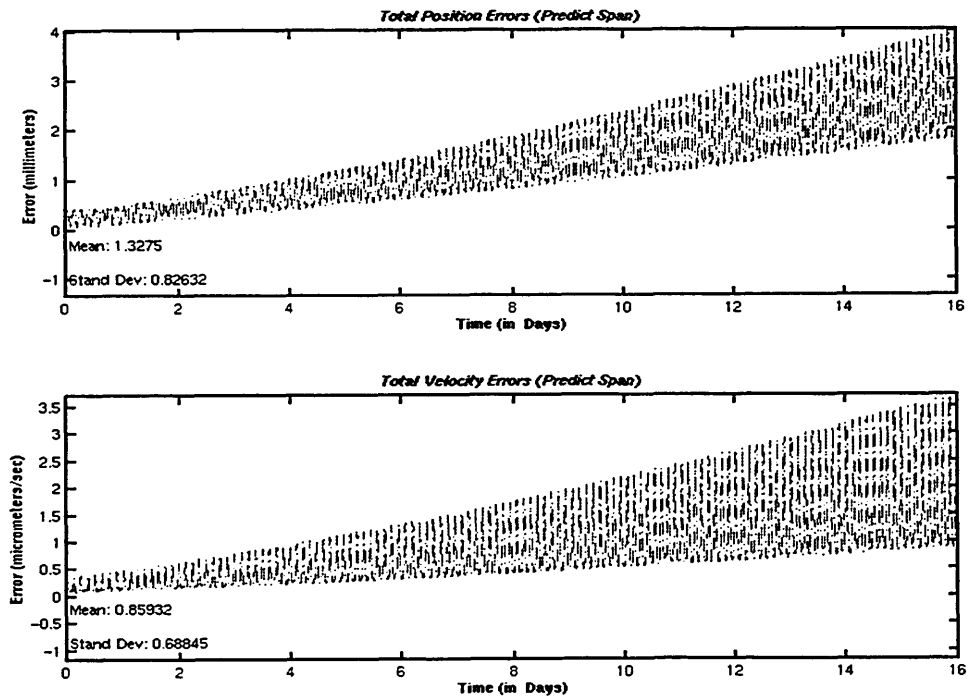


Figure 5.14 Errors in the Deterministic Analysis Process (Predict Span)

initial conditions for the DC were the values for position and velocity for the epoch time of the “truth” orbit.

The results of the comparison between the determined orbit and the “truth” orbit were remarkable. In the fit span of the determined orbit (Figure 5.13), the errors from the analysis process were on the order of half a millimeter in total position error and 0.375 micrometers per second in total velocity error. After 16 days of prediction (Figure 5.14), the errors only reached 4 millimeters in total position error and 3.75 micrometers per second in total velocity error. This extremely low level of error indicates that the analysis process contributes very little to the total errors in the output.

With the recognition of its impressive performance, this analysis process was performed on four different orbit configurations: Ellipso Borealis with the ascending node at noon, Ellipso Borealis with the ascending node at midnight, Ellipso Concordia, and Molniya. The result of the orbit determination process for each orbit type is presented in the next sections.

5.2.2 Ellipso

The Ellipso System is one of the new satellite telecommunication systems that is currently in development [14,15]. This system is designed to provide nearly global cellular communication with a direct link from the cellular phones to the satellites. The name Ellipso was selected from the elliptical orbits used in two of the three planes of the satellite constellation. The three orbital planes are broken down into two classes: the Borealis planes, which use the elliptical orbits; and the Concordia plane, which uses a near-circular, near-equatorial orbit. These classes of orbits were each examined for the possibility of orbit determination using GPS. The standard epoch for the orbit determination was April of 1999, at precisely 1:48 AM UTC time.

5.2.2.1 Borealis

The Borealis portion of the Ellipso constellation consists of two sun-synchronous elliptical orbits, each containing five satellites. The term sun-synchronous refers to a critically inclined orbit with a precession rate that matches the Earth revolution rate about the Sun. As a result, the ascending node of the orbit is consistently at the same local time. The local time of the ascending node is used in the identification of the Borealis orbits: node at noon (NAN) and node at midnight (NAM). As the local times indicate, the planes of these orbits are evenly spaced. The elliptical orbits have a perigee height of approximately 630 km in the southern hemisphere and an apogee height of about 7600 km in the northern hemisphere. This configuration results in an extended high-altitude pass over the northern hemisphere, where a majority of the cellular phone users reside. These orbits in combination with Concordia provide 24-hour coverage of the Earth, with the exception of extreme Southern latitudes.

These NAN and NAM Borealis orbits have slightly different behavior characteristics. This is due to the facts that the shadowing conditions are different and that the drag environment at perigee is different. To evaluate the orbit determination possibilities for the Borealis subconstellation, each orbit is examined separately.

As described earlier, the Borealis node at noon orbit was used in the early testing of the simulation. Because the pieces were already in place when the simulation testing was complete, this orbit was the first to undergo the orbit determination process. But before performing the orbit determination process, the error characteristics of the different antenna configurations were examined (Table 5.1). The error characteristics of the zenith antenna configuration are similar to those of the TOPEX LEO orbit, only slightly worse in the mean and standard deviation. For the nadir configuration, the navigation solutions are drastically affected by the increase in ionospheric delay. The two-antenna configuration has error characteristics that fall between the other cases, but it leans toward the zenith

characteristics. By employing the blocking function, the errors in the nadir and two-antenna cases are reduced significantly; the nadir improves beyond the unblocked two-antenna, and the blocked two-antenna produces the best results of all (Figure 5.15). From these statistics, the blocked two-antenna case should produce the best estimate of the true orbit, followed in order by zenith, blocked nadir, standard two-antenna, and standard nadir cases.

Table 5.1 Error Characteristics for Borealis NAN (in meters)

Antenna Configuration	Mean Error	Standard Deviation
Zenith	90.1879	76.3855
Nadir	1705.1599	1280.9419
Two Antenna	246.1895	374.7431
Nadir (blocked ion.)	238.1807	249.9126
Two Antenna (blocked ion.)	50.6567	59.5771

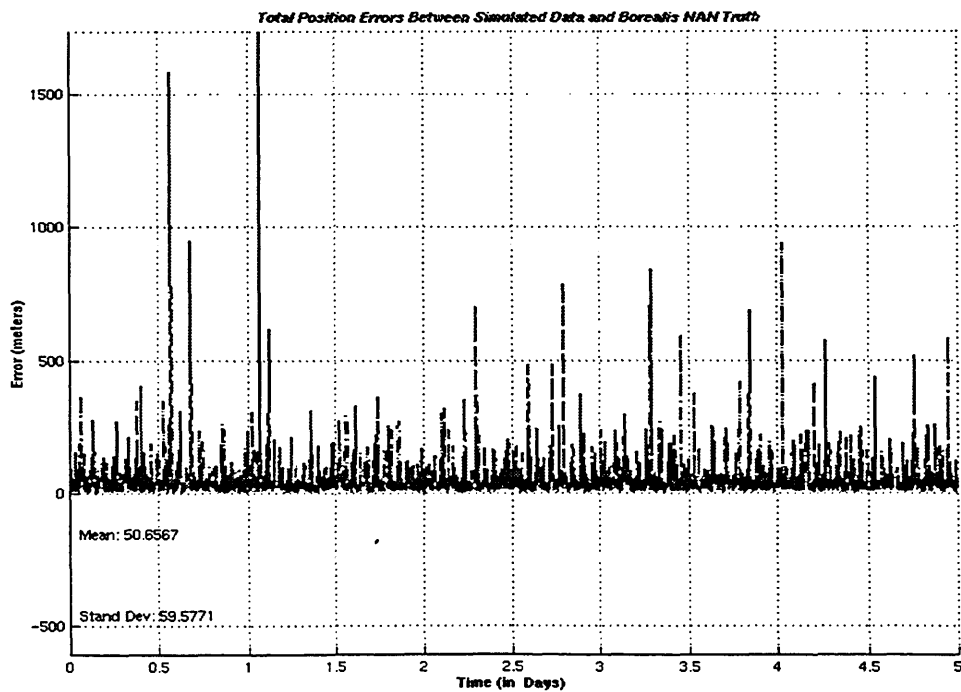


Figure 5.15 Borealis NAN Navigation Solution Errors - Two Antennas, Blocked Ionosphere

The first attempt at orbit determination for the Borealis node at noon orbit was performed using the densest and longest of the observation files. This file contained the observations from the two antenna unblocked configuration with the one minute density for a two-week time period. The simulation only lost five percent of the total times due to too few GPS signals and rejections for unacceptable geometry. The file was too long for the DC to accept all the observations; it could only accept 10 days, 13 hours, and 16 minutes worth of data. The DC quickly converged on a set of initial conditions and the ephemeris was compared the true orbit. The total position errors in the fit span (Figure 5.16) reached a maximum value of about 45 meters with a mean of 30 meters, and velocity errors up to 25 millimeters per second with a mean of 13.3 millimeters per second. The error growth in the predict span (Figure 5.17) was approximately linear for the first 8 days, rising from about 45 meters to about 75 meters in position and from about 25 to about 60 millimeters per second in velocity; by the end of 19 1/2 day predict span, the total errors rose to 225 meters and 0.25 meters per second.

The second set of observations had the one minute data density for two weeks using two antennas, but the ionosphere had been blocked from the observations. This blocking reduced the number of observations by another nine percent. The DC still could not accept all the observations, but it did take 11 days, 14 hours, and 21 minutes. Again, the DC converged quickly, but the comparison with the “truth” showed a significant improvement in both the fit and predict spans. The total position errors in the fit span (Figure 5.18) only reached 3 meters with a mean of 1.9 meters, and the total velocity errors up to 1.65 millimeters per second with a mean of 0.889 millimeters per second. The growth rate in the predict span (Figure 5.19) is not linear, but the errors only grow from 3 meters to about 45 meters in position and from 1.65 to approximately 45 millimeters per second in velocity, both over a period of 18 1/2 days. From the examination of the error characteristics, this result was expected.

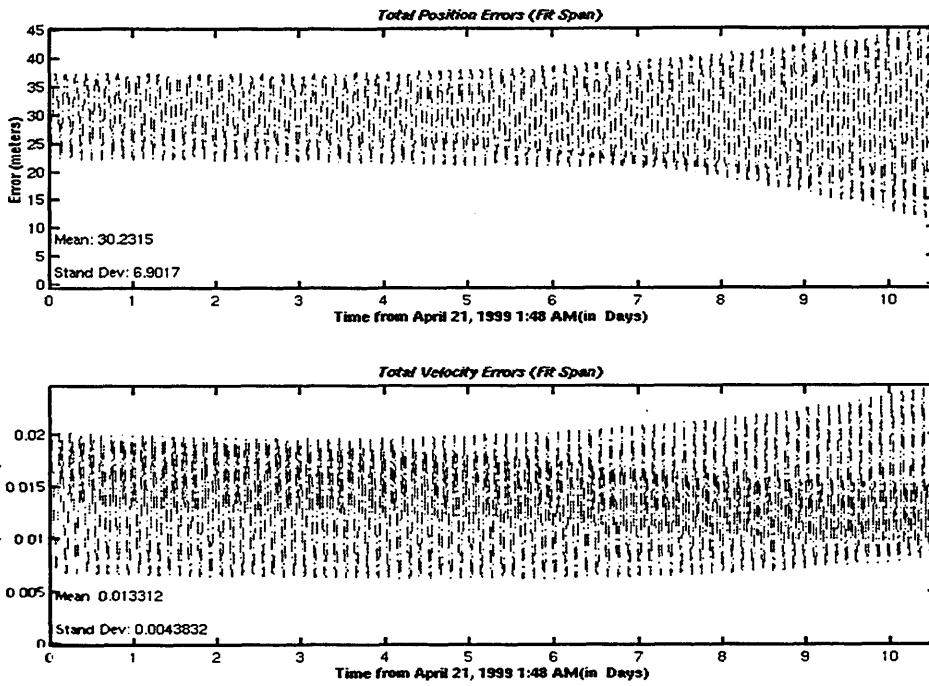


Figure 5.16 OD Errors in Borealis, Node at Noon - Two Antennas, 1 Minute Density, Maximum Length (Fit Span)

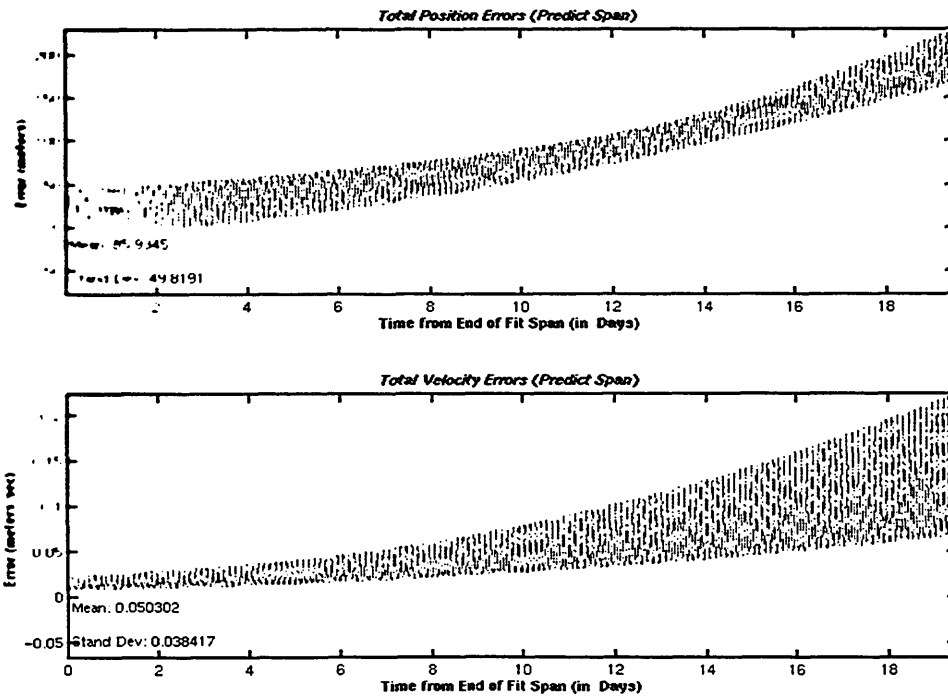


Figure 5.17 OD Errors in Borealis, Node at Noon - Two Antennas, 1 Minute Density, Maximum Length (Predict Span)

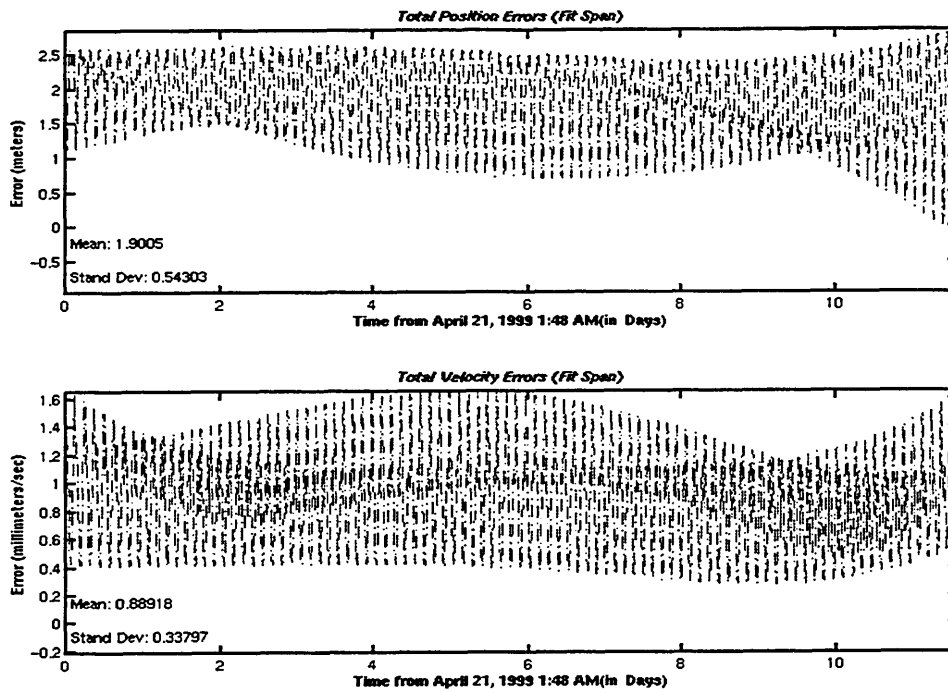


Figure 5.18 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 1 Minute Density, Maximum Length (Fit Span)

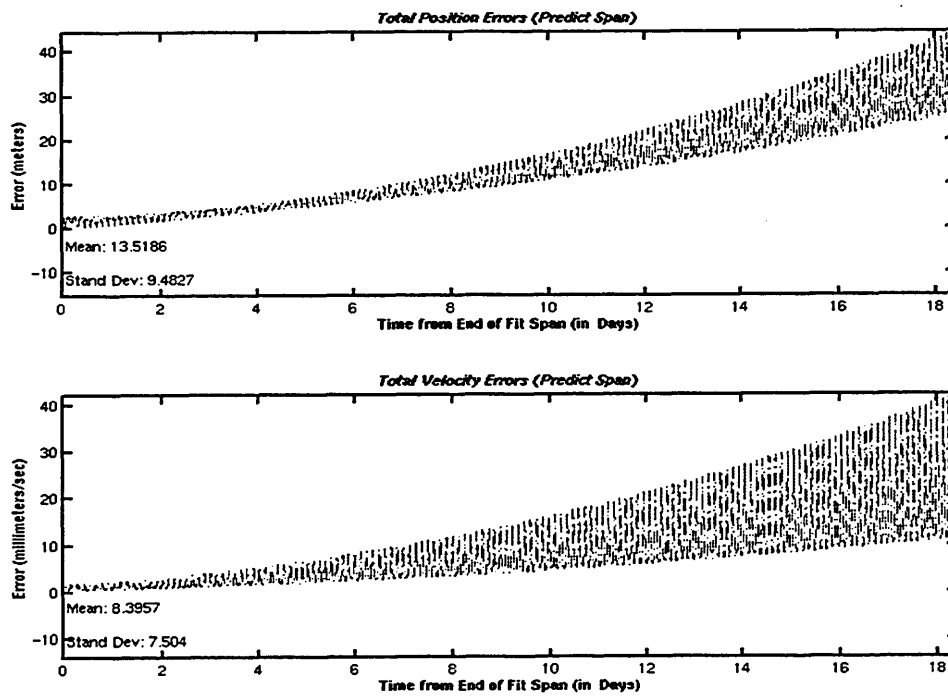


Figure 5.19 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 1 Minute Density, Maximum Length (Predict Span)

At this stage in the testing, the point was made that the one minute measurement density may not be realistic. The amount of data accumulated from attempting to generate navigation solutions every minute may be too unwieldy for the target satellite to transfer to its ground stations. From this point on, the maximum observation density for the orbit determination process was once every four minutes.

All antenna configurations and blocking conditions were examined for the four minute density with two weeks of observations (Tables 5.2 and 5.3). The four minute density cases produced results that were as good as or better than the better of the one minute density cases (Figures 5.20 and 5.21); neither of the nadir facing antenna cases are included in that statement. The poor performance of the blocked nadir case is somewhat puzzling. Its error statistics are comparable with those of the unblocked two antenna case, but the errors in the predict span are much worse. As a result of the blocked ionosphere,

Table 5.2 Borealis NAN OD Position Errors (in meters) - 4 Minute Density

Antenna Configuration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
Zenith	6.9195	2.9415	12.7	47.8
Nadir	340.1995	61.8215	459.0	596.6
Two Antenna	28.3662	5.5568	36.9	46.1
Nadir (blocked ion.)	12.8306	4.7638	27.4	184.4
Two Antenna (blocked ion.)	2.1947	0.5194	3.7	8

Table 5.3 Borealis NAN OD Velocity Errors (in millimeters/sec) - 4 Minute Density

Antenna Configuration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
Zenith	2.8628	1.0060	5.9	40.4
Nadir	138.0285	44.2012	238.1	291.8
Two Antenna	12.1570	3.7581	21.2	18.7
Nadir (blocked ion.)	6.7860	4.6807	21.6	171.6
Two Antenna (blocked ion.)	0.96382	0.34037	2.0	5.5

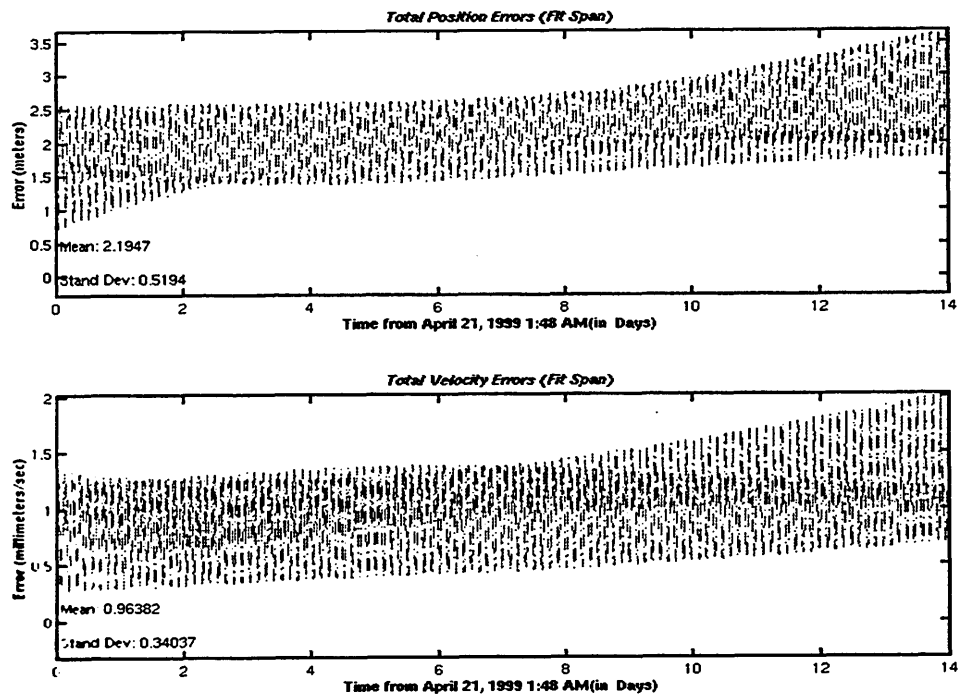


Figure 5.20 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Fit Span)

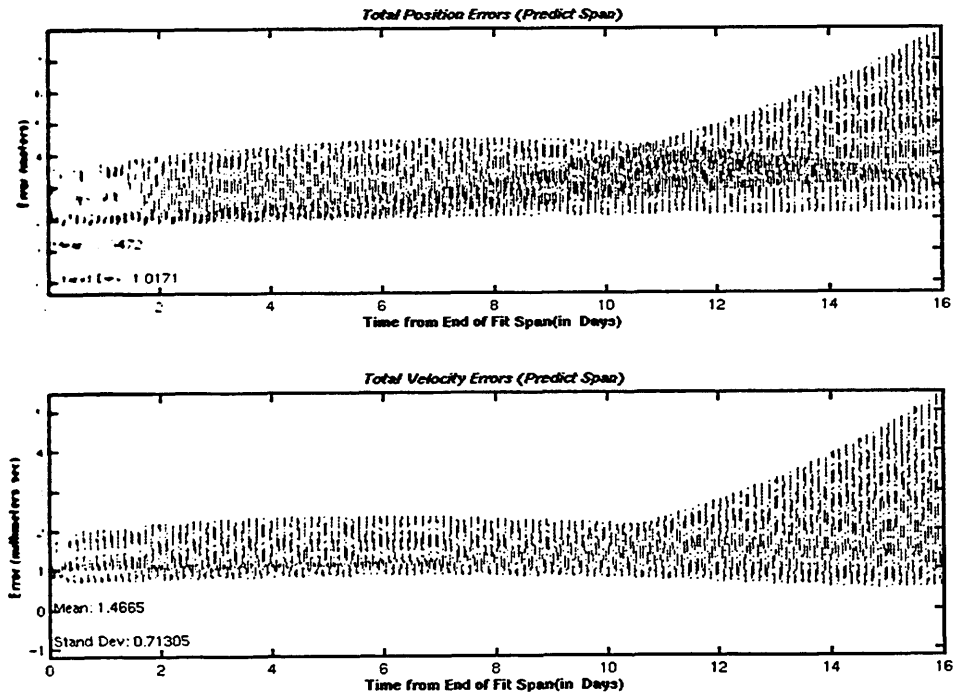


Figure 5.21 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span)

the nadir blocked configuration has the lowest number of observations, 26.7% of the total time steps. This low number of observations may be the reason for this poor performance.

The remaining tests on the Borealis NAN orbit determination focused on reducing the amount of observations without severely impacting the results of the process. The density of the observations was dropped to once every fifteen minutes (Tables 5.4 and 5.5). Since the two-antenna cases were producing the best results, their configuration was chosen to be the baseline of the tests. In another test, the duration of the fit span was also reduced by half while maintaining the fifteen minute observation density. In each case, the error characteristics of the fit span were barely affected by the reducing in density or duration (Figures 5.22 and 5.23). Reducing the density of the navigation solutions had little effect on the predict span of the unblocked case, but the blocked case's predict changed notably (Figure 5.24). Reducing the length of the fit span resulted in a more rapid growth in the predict span (Figures 5.24 and 5.25).

Table 5.4 Borealis NAN OD Position Errors (in meters) - 15 Minute Density, Two Antennas

Duration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
2 Weeks (unblocked)	32.3277	6.3456	41.9	62.4
2 Weeks (blocked)	3.1340	1.3868	6.2	66.0
1 Week (unblocked)	34.1481	6.5670	48.4	1021.6
1 Week (blocked)	3.0046	0.7776	5.6	180.8

Table 5.5 Borealis NAN OD Velocity Errors (in millimeters/sec) - 15 Minute Density, Two Antennas

Duration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
2 Weeks (unblocked)	14.0618	4.416	24.0	28.8
2 Weeks (blocked)	1.5293	0.68994	3.6	58.9
1 Week (unblocked)	14.9208	4.8668	27.8	963.7
1 Week (blocked)	1.4505	0.59502	3.2	169.5

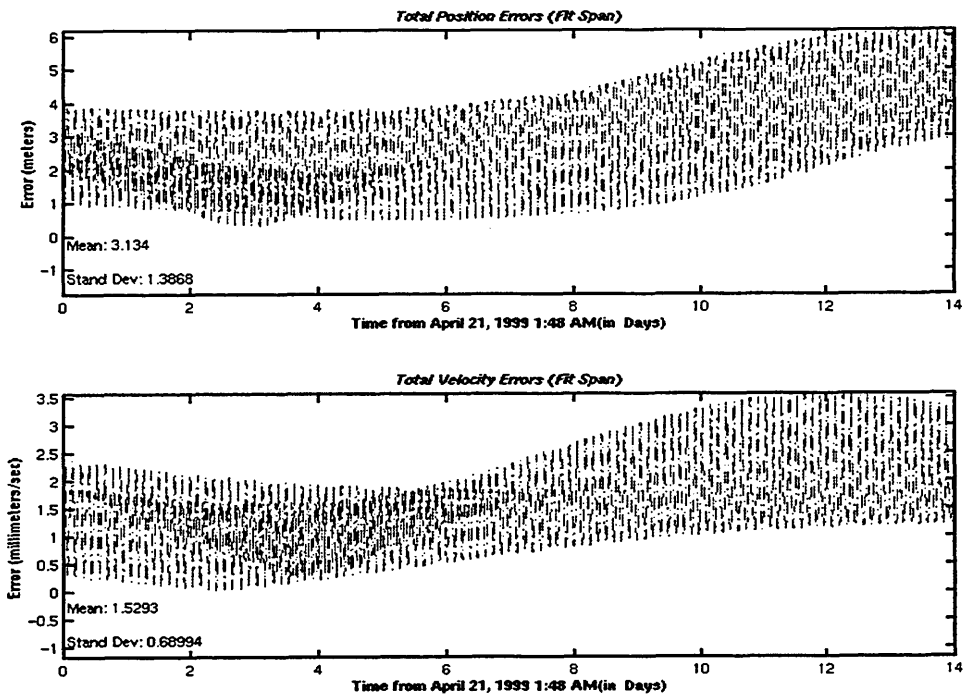


Figure 5.22 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Fit Span)

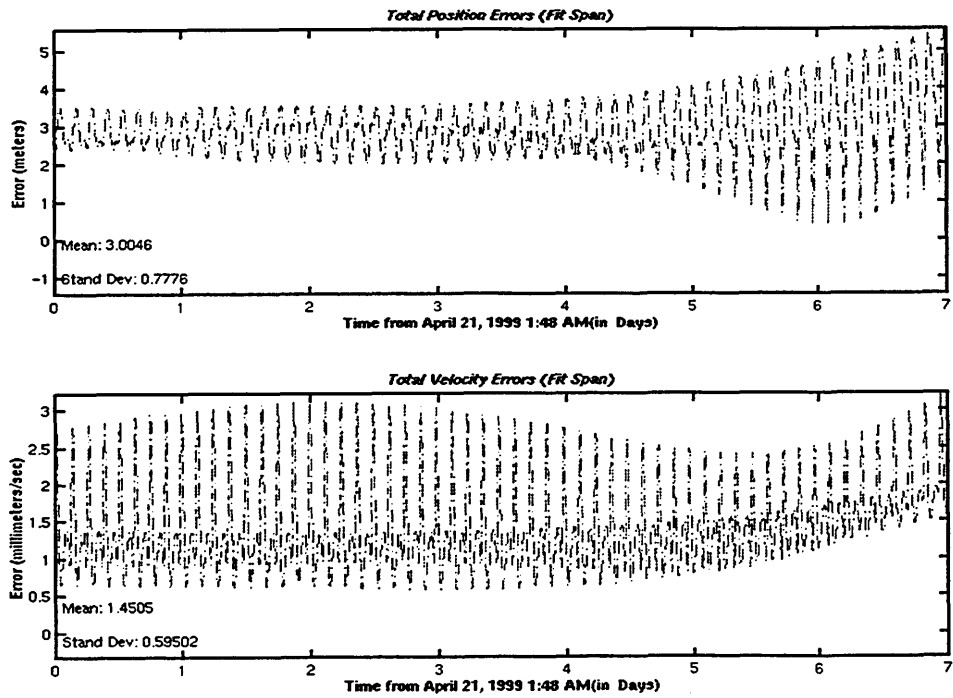


Figure 5.23 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 1 Week of Obs. (Fit Span)

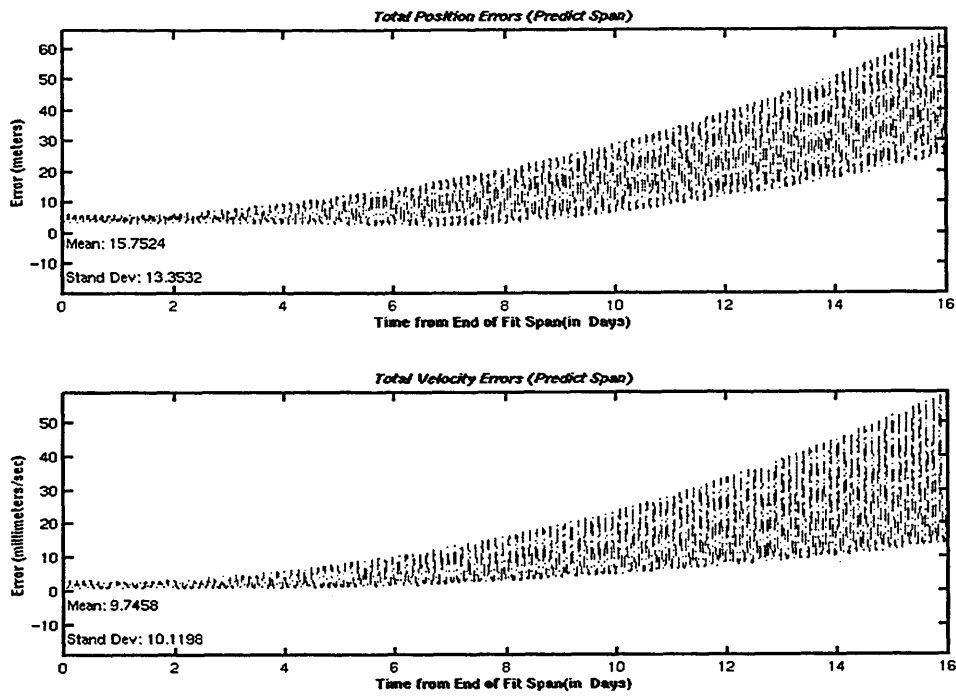


Figure 5.24 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Predict Span)

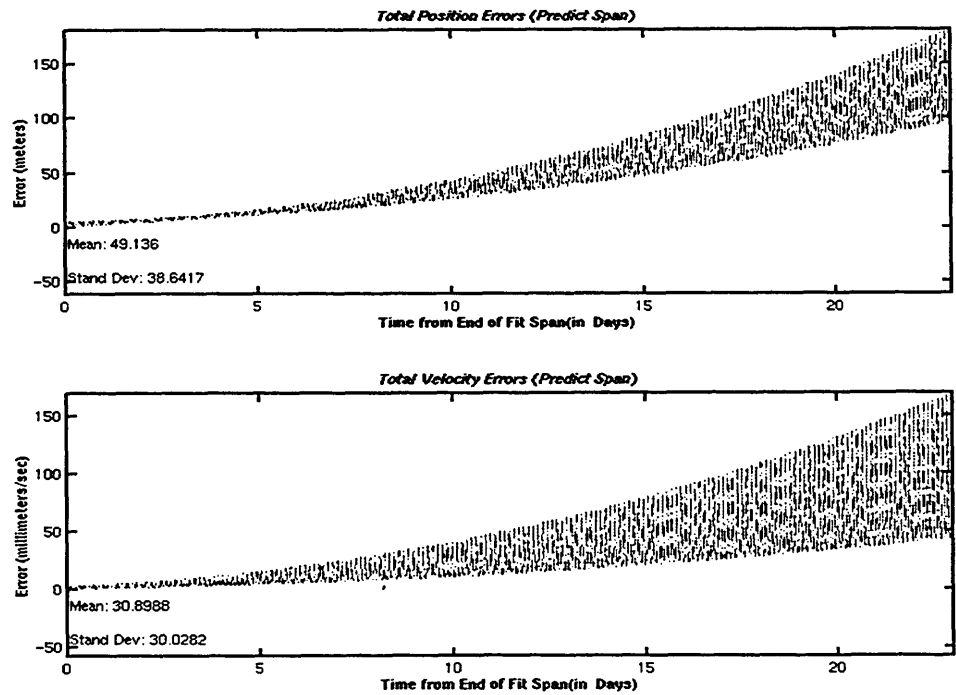


Figure 5.25 OD Errors in Borealis, Node at Noon - Two Antennas, Blocked Ionosphere, 15 Minute Density, 1 Week of Obs. (Predict Span)

Although they have some behavioral differences, node at noon and node at midnight are similar orbits. Their error characteristics are not identical, but they are relatively close to each other (Table 5.6). For this reason, the results of the node at midnight were not expected to deviate terribly from those of the node at noon.

Table 5.6 Error Characteristics for Borealis NAM (in meters)

Antenna Configuration	Mean Error	Standard Deviation
Zenith	88.1328	69.367
Nadir	1789.8352	1594.9881
Two Antenna	221.1131	286.6109
Nadir (blocked ionosphere)	227.45	244.3582
Two Antenna (blocked ion.)	48.2728	51.0939

The Borealis node at midnight was examined using the same antenna configurations and blocking conditions for its two weeks of observations at four minute density. The results of these orbit determination processes were reasonably comparable with those for the node at noon; the zenith configuration produced a more accurate sixteen day prediction than the two antenna case with the ionosphere blocked (Tables 5.7 and 5.8). The two antenna case provided a more accurate prediction for the first week, but its error growth rate was higher than that of the zenith case (Figures 5.26 and 5.27). Since most orbit prediction applications are interested in the shorter term predictions, the two antenna blocked configuration was still considered the better case.

5.2.2.2 Concordia

Although Concordia [14] is not an elliptical orbit, it does fall into the other category mentioned in the thesis title: medium altitude. Concordia is a near-circular equatorial orbit with a radius of approximately 14428 km. Also, all Ellipso satellites will be identical,

regardless which plane they are occupying. So, the Concordia orbit must also be evaluated for orbit determination with GPS.

Table 5.7 Borealis NAM OD Position Errors (in meters) - 4 Minute Density

Antenna Configuration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
Zenith	6.1099	2.0821	11.9	14.6
Nadir	356.0985	46.4418	442.9	680.9
Two Antenna	23.6153	5.4592	39.0	232.7
Nadir (blocked ion.)	12.0079	4.0728	22.9	91.1
Two Antenna (blocked ion.)	1.6343	0.66936	3.4	45.1

Table 5.8 Borealis NAM OD Velocity Errors (in millimeters/sec) - 4 Minute Density

Antenna Configuration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
Zenith	2.6041	0.99305	5.9	9.4
Nadir	148.2219	47.1231	267.9	523.3
Two Antenna	10.0798	3.2317	20.3	204.6
Nadir (blocked ion.)	6.0267	3.6439	15.2	68.9
Two Antenna (blocked ion.)	0.84285	0.44096	2.9	41.3

Because of the high altitude, the zenith configuration never receives enough GPS signals to produce a navigation solution. The nadir configuration acquires enough signals for solutions about half of the time, but the solutions exhibit the same levels of error as usual. The two antenna configuration is more likely to acquire GPS satellites with a good geometric distribution. Besides these considerations, the previous paragraph states that the Ellipso satellites are identical and the Borealis planes achieve their best orbit determination performances using the two antenna arrangement. So all Concordia processes used the two antenna configuration.

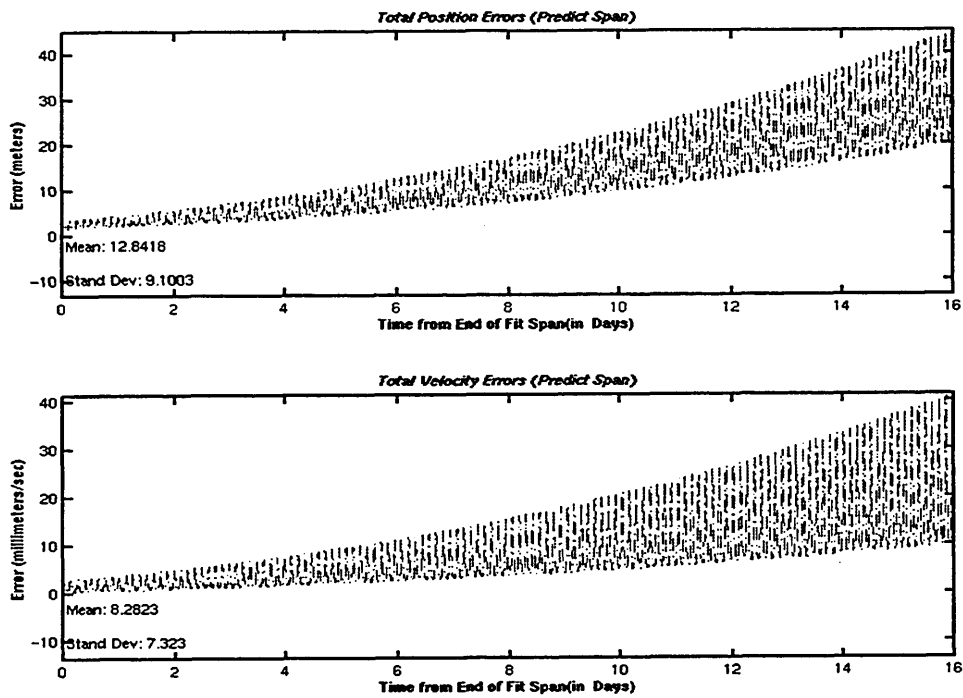


Figure 5.26 OD Errors in Borealis, Node at Midnight - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span)

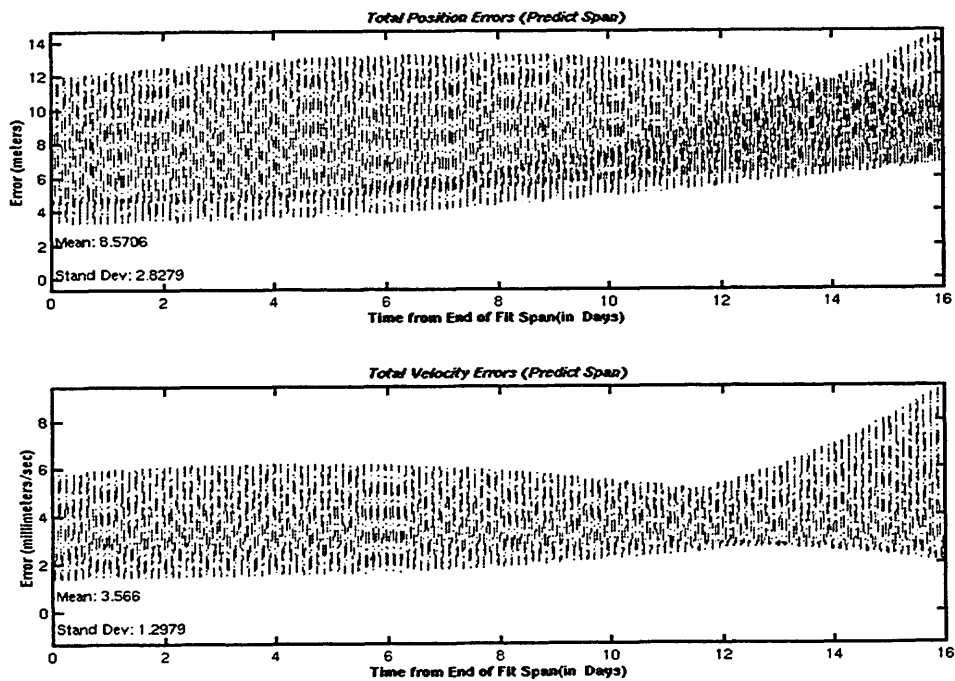


Figure 5.27 OD Errors in Borealis, Node at Midnight - Zenith Antenna, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span)

Because the Concordia orbit does not have the large changes in velocity and altitude, its initial orbit determination tests were performed with an observation density of fifteen minutes. Using one or two weeks worth of observations, the orbits produced from this fifteen minute data provided outstanding predictions for periods of three or two weeks, respectively (Tables 5.9 and 5.10). Blocking the ionosphere produces even better result (Figures 5.28 and 5.29). The error growth in the predict span of each case is nearly linear throughout the span (Figure 5.29). The density was increased to one measurement every four minutes to observe Concordia's reaction, but there was no appreciable difference in the performance.

Table 5.9 Concordia OD Position Errors (in meters) - Two Antennas, 15 Minute Density

Duration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
2 Weeks	31.0344	13.3512	49.0	54.8
2 Weeks (blocked ion.)	2.7979	1.2307	5.7	11.4
1 Week	33.0769	13.6223	50.8	105.5
1 Week (blocked ion.)	3.0825	1.6249	7.8	35.8

Table 5.10 Concordia OD Velocity Errors (in millimeters/sec) - Two Antennas, 15 Minute Density

Duration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
2 Weeks	11.1336	4.9501	17.3	17.9
2 Weeks (blocked ion.)	0.90264	0.34468	1.8	2.8
1 Week	11.8743	4.9582	18.5	33.1
1 Week (blocked ion.)	1.059	0.54869	2.5	11.4

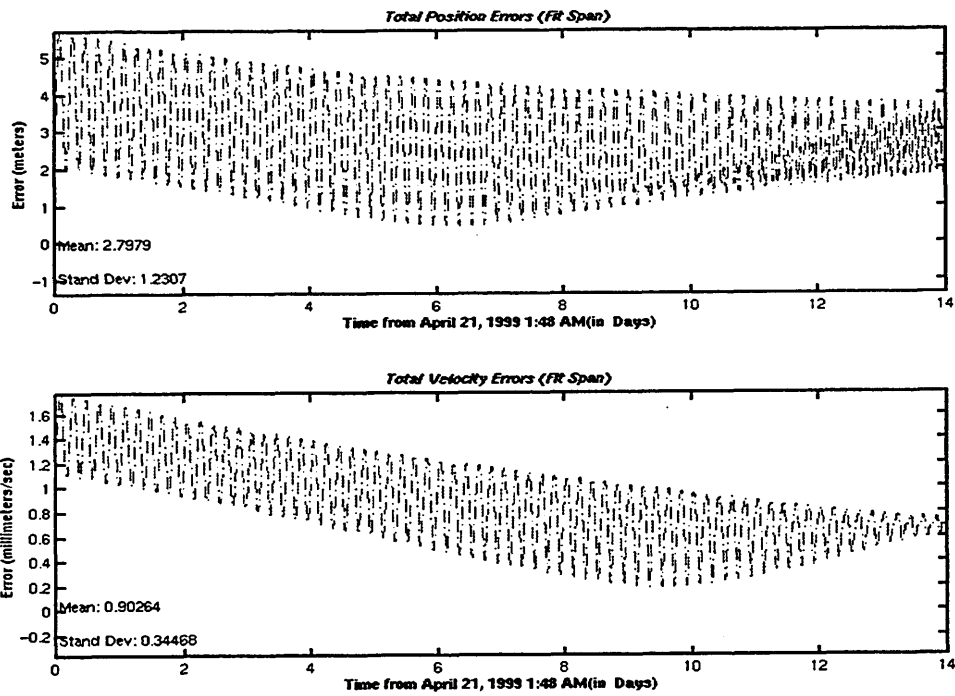


Figure 5.28 OD Errors in Concordia - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Fit Span)

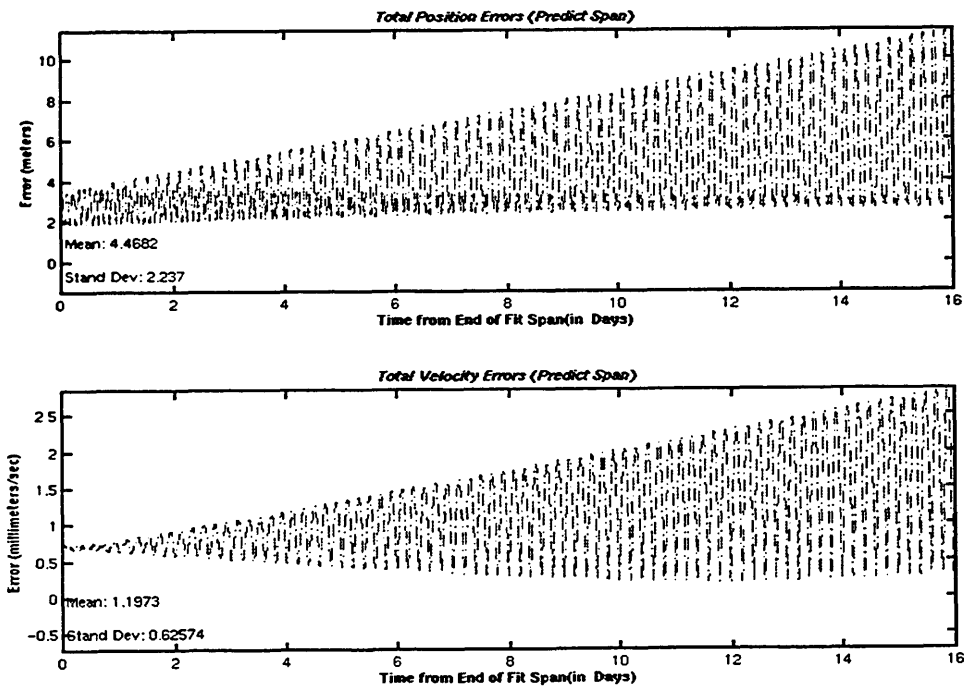


Figure 5.29 OD Errors in Concordia - Two Antennas, Blocked Ionosphere, 15 Minute Density, 2 Weeks of Obs. (Predict Span)

5.2.3 Molniya

The Molniya class of orbit is a favorite of the former Soviet Union. The Russians have placed many communication and surveillance satellites in this type of orbit [34]. A Molniya is a highly eccentric orbit with a period of twelve hours. The name Molniya comes from the Russian word for lightning, and it was given to these orbits for the high velocity at which they passed through perigee (as low as 400 km altitude). The main feature of this type of orbit is its nearly critical inclination. This orbit reaches apogee at approximately 40,000 km altitude, which is well above the altitude of the GPS satellite's orbit (20,200 km). Because of this extremely high altitude, orbit determination based on the use of GPS measurements was expected to be more difficult.

For the Molniya target orbit, the perigee and apogee altitudes are 450 km and 39730 km, respectively. The error characteristics from the simulation were not extraordinarily high (Table 5.11), but the measurements were sparse; this scarcity of observations was due to the extended duration that the satellite spends at very high altitude. Using the two antenna configuration, the simulation only produced solutions about ten percent of the time during this orbit's processing; the single antenna cases only produced half of that amount. With two antennas, these solutions can be calculated from altitudes as high as 17,000 km or even 33,000 km (from the far side of the GPS constellation) (Figure 5.30).

Table 5.11 Error Characteristics for Molniya (in meters)

Antenna Configuration	Mean Error	Standard Deviation
Zenith	88.6468	78.6841
Nadir	2689.4688	3909.29
Two Antenna	1034.898	3343.3516
Nadir (blocked ionosphere)	304.6109	346.0166
Two Antenna (blocked ion.)	72.1693	133.7917

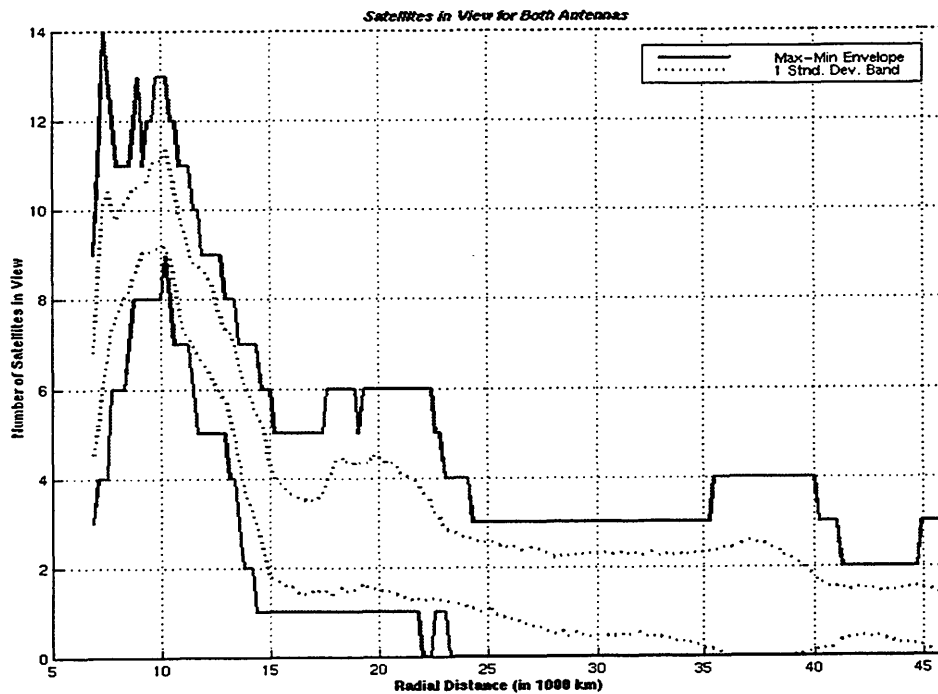


Figure 5.30 Number of Satellites in View for the Molniya (both antennas)

The full set of antenna configurations and blocking conditions were tested for their performance. The observations from the full two weeks were input with a four minute density, and the propagator for the orbit determination process was time-regularized Cowell. The epoch used for this orbit was January 1, 1995 at 11:03 AM and 28.656576 seconds. The atmospheric density used for this epoch was smoothed in a similar way as the density for the April 1999 epoch. The results were similar to those of the other orbits, but the cross-track error appeared to dominate the total; the total errors for each case appeared as a band with the approximate size of the cross-track errors (Figure 5.31) and very little variation throughout the fit and predict spans (Figures 5.32 and 5.33). Because of the shortage of input data, this performance seemed impressive. To determine if the results would improve with more data, the two-antenna cases were reprocessed using one minute observation density for the full two weeks. There was no apparent improvement with the higher data density.

Table 5.12 Molniya OD Position Errors (in meters) - 4 Minute Density, 2 Weeks of Obs.

Antenna Configuration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
Zenith	37.3140	16.1638	63.5	64.7
Nadir	961.6131	414.1428	1474.7	1421.9
Two Antenna	119.6393	55.3957	189.1	261.0
Nadir (blocked ion.)	132.7123	64.1570	204.1	263.2
Two Antenna (blocked ion.)	16.2870	7.5542	25.3	23.7

Table 5.13 Molniya OD Velocity Errors (in millimeters/sec) - 4 Minute Density, 2 Weeks of Obs.

Antenna Configuration	Mean	Fit Span Std. Dev.	Max	Predict Span Max
Zenith	3.2251	1.9364	14.0	53.6
Nadir	87.4423	64.3121	370.0	568.4
Two Antenna	11.6337	9.2528	51.8	217.9
Nadir (blocked ion.)	11.0520	7.5251	29.8	217.1
Two Antenna (blocked ion.)	1.3552	0.85256	3.3	11.2

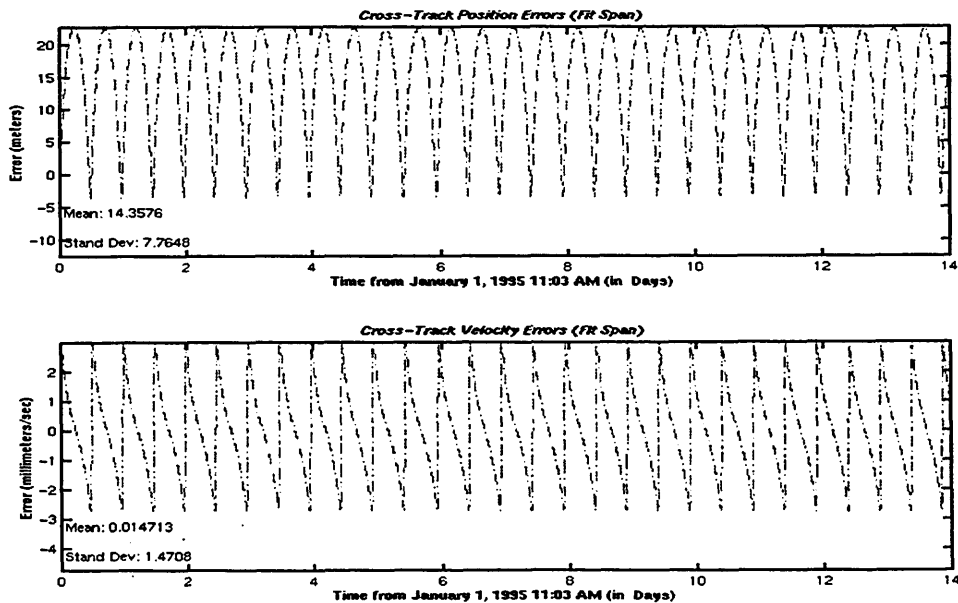


Figure 5.31 Crosstrack Errors in Molniya - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Fit Span)

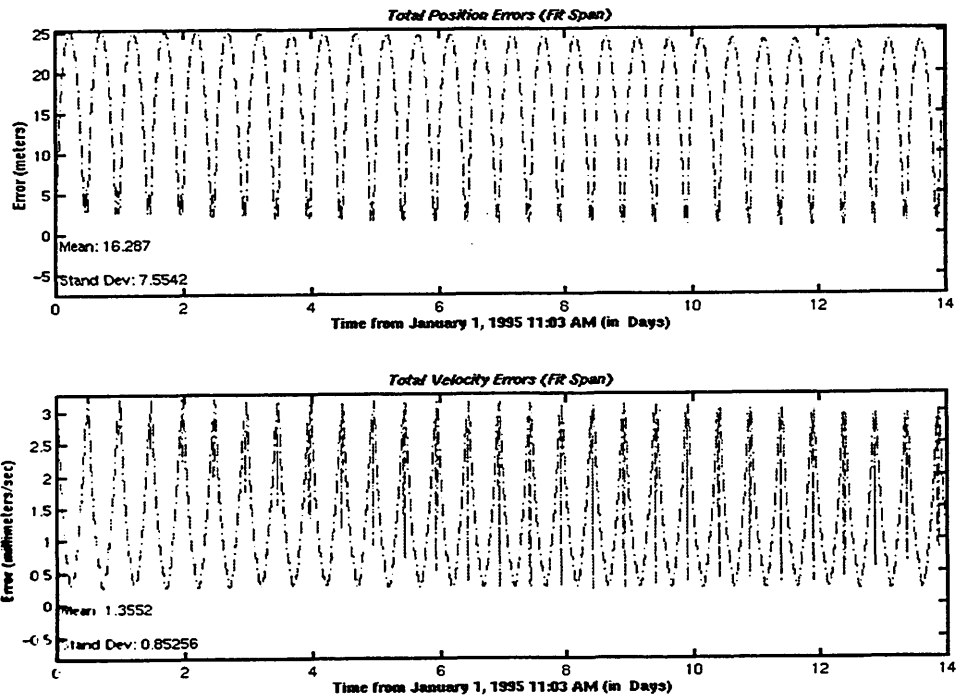


Figure 5.32 OD Errors in Molniya - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Fit Span)

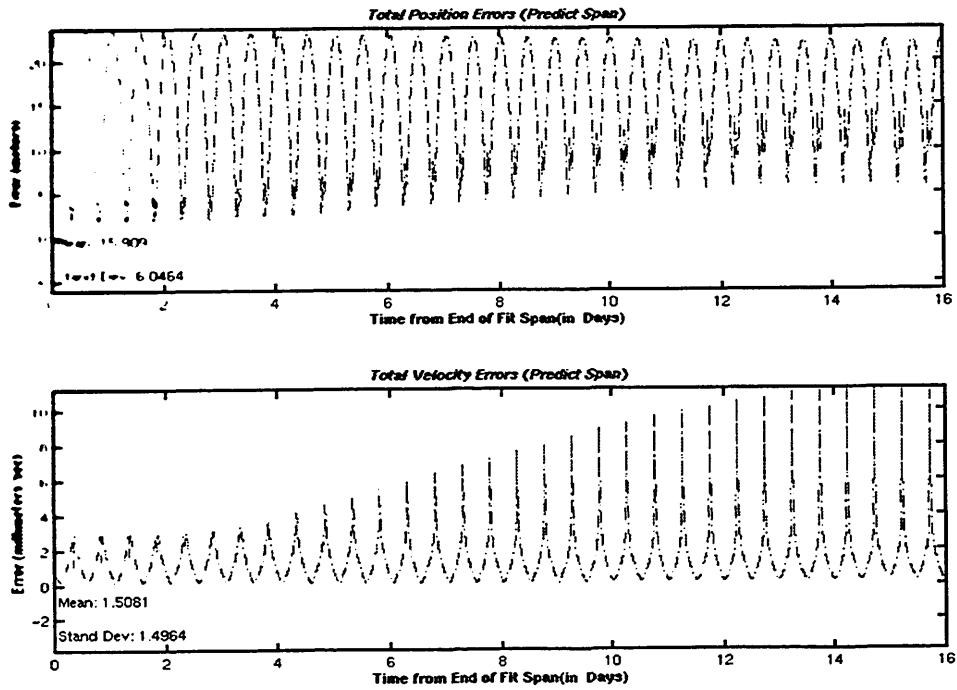


Figure 5.33 OD Errors in Molniya - Two Antennas, Blocked Ionosphere, 4 Minute Density, 2 Weeks of Obs. (Predict Span)

5.3 Effects of Mismodeling

In the initial orbit determination processes, the GTDS operations are set up to duplicate the models used in the generation of the target orbit. This type of test is conducted to evaluate the accuracy of an orbit that was determined using the exact force models. But in reality, the models used in the orbit determination will not match the actual forces experienced by the spacecraft. So, in a selected number of cases, the orbit determination process was repeated using a different variant of some of the force models. Some of these variations were:

- employing the GEMT3 geopotential field instead of JGM2
- reducing the degree and order of the current geopotential model (either JGM2 or GEMT3), in one case to as low as 12
- using a different atmospheric density model
- varying the coefficients of drag or solar radiation pressure between the fit and predict spans of the ephemeris

In the cases when the coefficients were changed between the fit and predict spans, the ephemeris generation had to be initiated for each of the spans and each span was stored in a separate file.

Each of these model changes tests the sensitivity of the orbit to that type of force model error. For instance, a LEO orbit or an eccentric orbit with a low perigee height is extremely susceptible to changes in atmospheric density and drag characteristics.

5.3.1 Borealis

The Borealis orbits were the subject of the most mismodeling experiments. The size and model of the geopotential field were changed, as were the effects of drag and solar radiation pressure. The effects of each change were examined individually.

The effects of solar radiation pressure were changed by altering the coefficient of solar reflectivity by twenty-five percent between the fit and predict spans of the ephemeris. This alteration causes an increase in position error on the order of 12-14 km after a two week prediction (Figures 5.34 and 5.35). The same technique was also used on the coefficient of drag, but this change was only by twenty percent. The change in drag caused errors of 25 to 35 km, depending on whether the drag was decreased or increased (Figures 5.36 and 5.37).

Drag was mismodeled in more than one way. In addition to the change in the drag coefficient, the atmospheric model was also changed from Jacchia-Roberts to Harris-Priester. This model change introduced an error of 200 meters in the fit span and 17 kilometers in the predict span (Figures 5.38 and 5.39). The errors also exhibit an oscillating behavior through the fit span, ranging from 0 to about 80 meters.

The next drag mismodeling case included a change in the simulation epoch. The entire analysis process (“truth” generation, simulation and OD) was performed in February of 1992, during a period of high solar activity and high geomagnetic index instability. The high solar activity increased the atmospheric density; the geomagnetic index instability produced significant short-term variations in the atmospheric density. Initially, the orbit was determined using observations from the unblocked two antenna configuration. For this epoch, the fits produced with these observations were consistent with each other, regardless of the density and duration. But this noisy drag environment produced different levels of error in the predictions depending on the observation density, the length of the fit span, and which week of the epoch was used in the one-week fits (Tables 5.14 and 5.15).

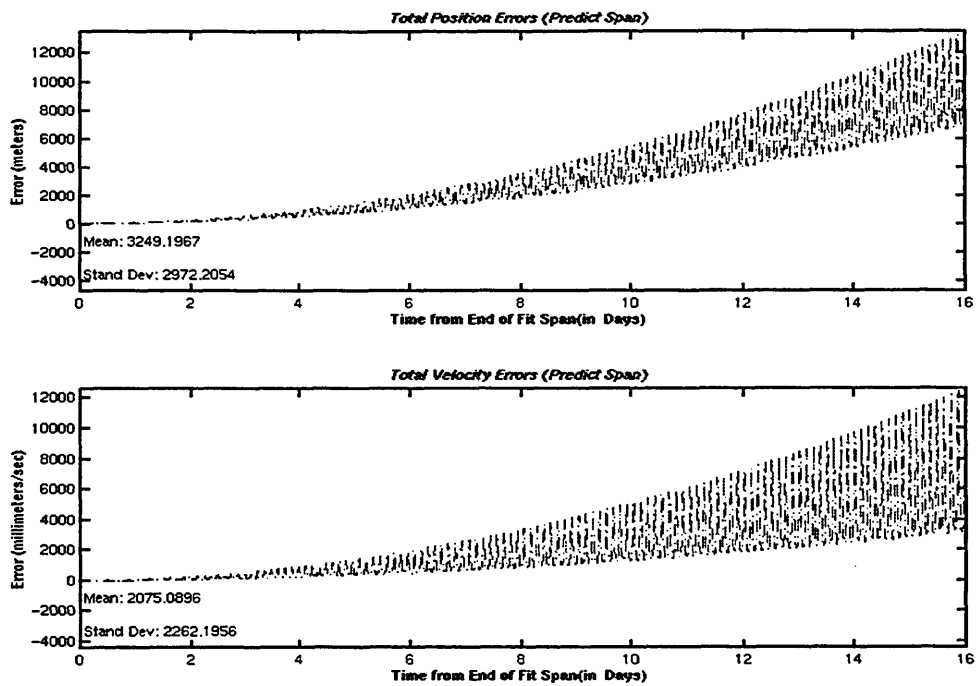


Figure 5.34 Prediction Errors in Borealis NAN - Mismodeling Solar Radiation Pressure, Solar Reflectivity (-25%)

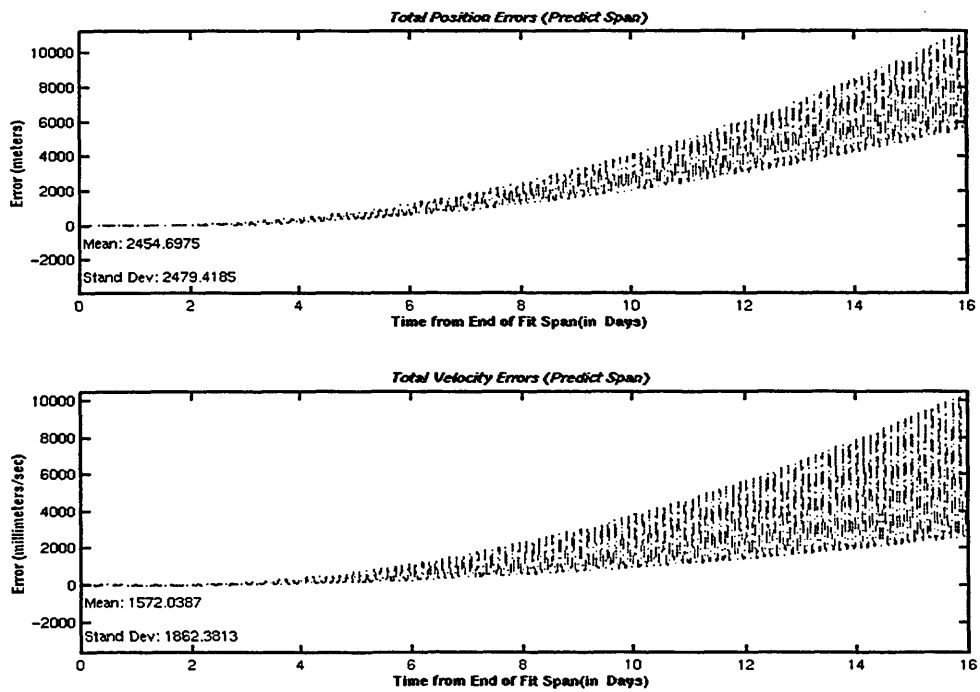


Figure 5.35 Prediction Errors in Borealis NAN - Mismodeling Solar Radiation Pressure, Solar Reflectivity (+25%)

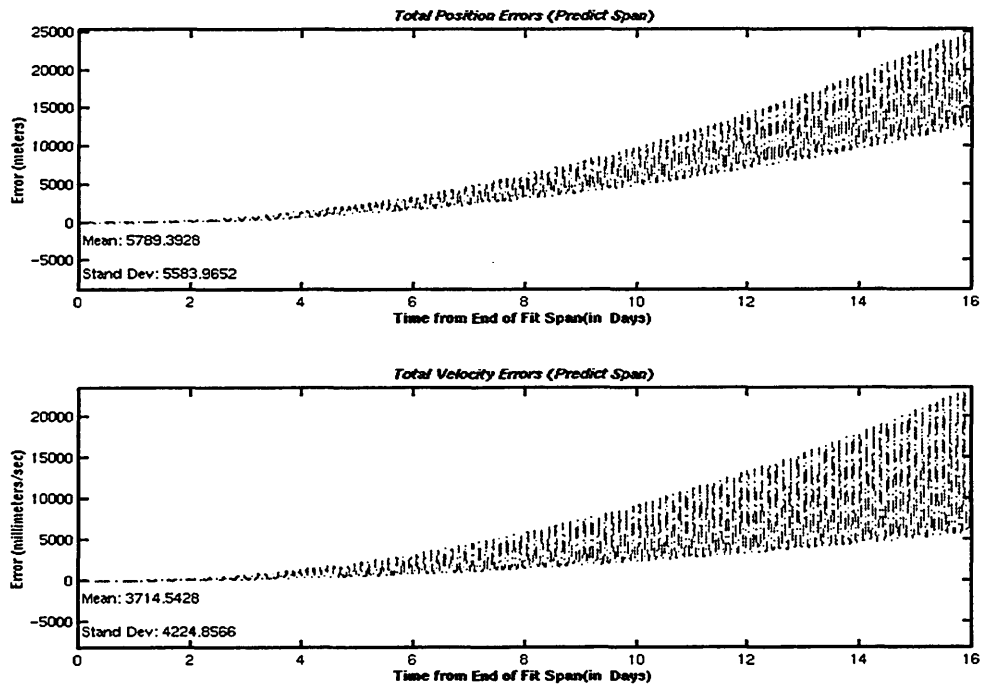


Figure 5.36 Prediction Errors in Borealis NAN - Mismodeling Drag, Drag Coefficient (-25%)

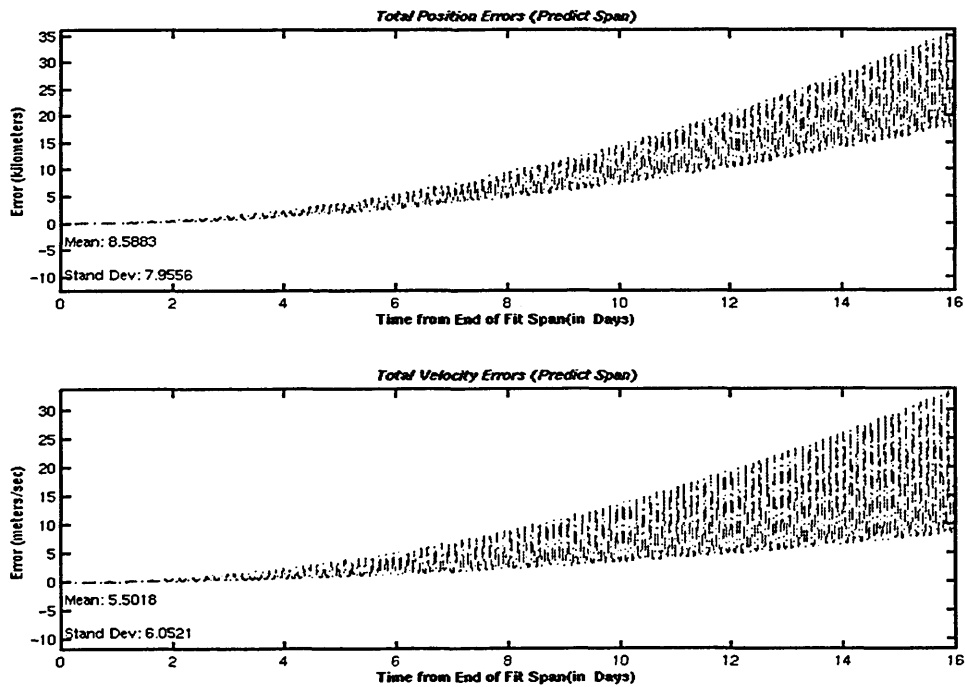


Figure 5.37 Prediction Errors in Borealis NAN - Mismodeling Drag, Drag Coefficient (+25%)

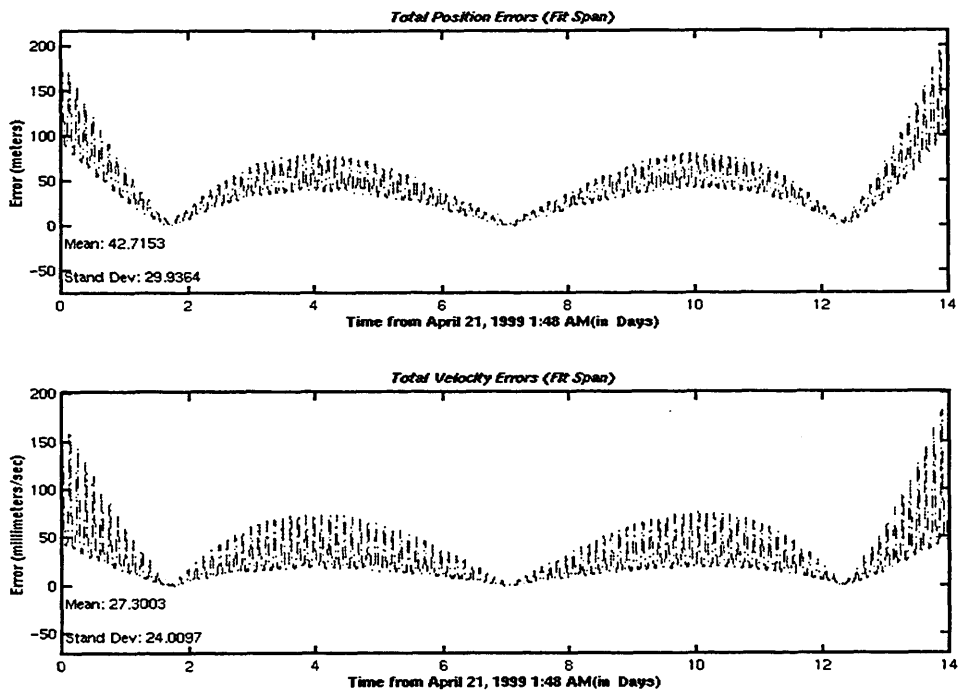


Figure 5.38 OD Errors in Borealis NAN - Mismodeling Drag, Harris-Priester (Fit Span)

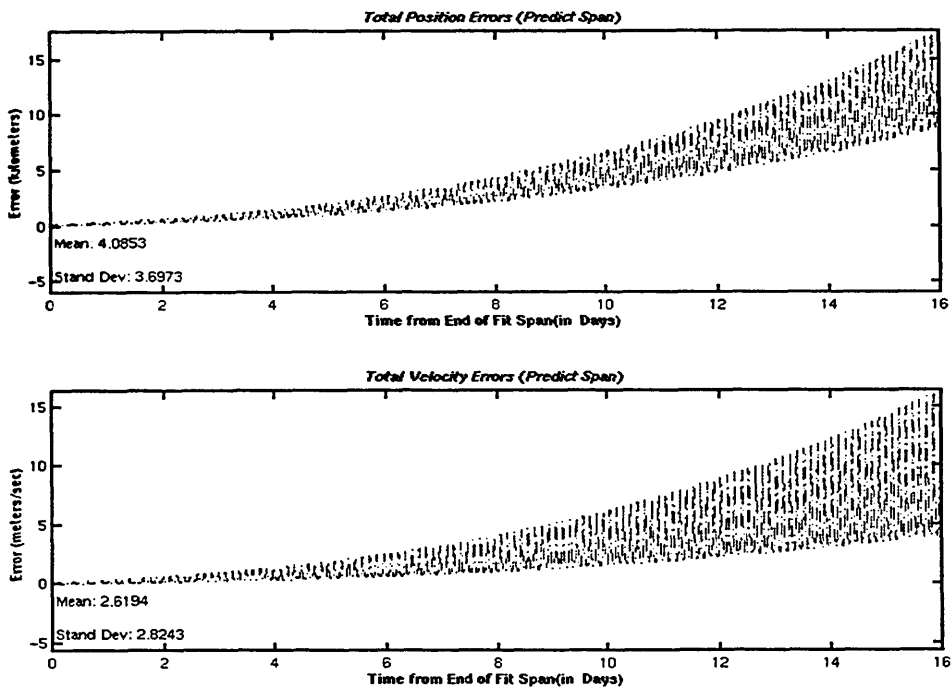


Figure 5.39 OD Errors in Borealis NAN - Mismodeling Drag, Harris-Priester (Predict Span)

The predict spans varied with the length and timing of the fit spans. A two week fit was followed by a 16 day predict. For the one week fits, the predict span length depended on which week was used; the predict span was 23 days for the first week fit, and 16 days for the second.

Table 5.14 Borealis NAN OD Position Errors (in meters) - '92 Epoch

Density/Duration of Obs.	Mean	Fit Span Std. Dev.	Max	Predict Span Max
4 min., 1 week (first)	30.6683	8.9722	46.9	1593.1
4 min., 1 week (second)	27.2374	9.9017	44.0	353.7
4 min., 2 weeks	29.0184	9.5709	48.4	308.8
15 min., 1 week (first)	29.1351	8.151	40.6	282.7
15 min., 1 week (second)	28.686	8.6566	43.3	369.4
15 min., 2 weeks	28.7066	8.1904	41.8	501.3

Table 5.15 Borealis NAN OD Velocity Errors (in millimeters/sec) - '92 Epoch

Density/Duration of Obs.	Mean	Fit Span Std. Dev.	Max	Predict Span Max
4 min., 1 week (first)	14.4609	5.7591	25.2	1470.7
4 min., 1 week (second)	13.035	5.5536	23.6	332.7
4 min., 2 weeks	13.649	5.68	24.4	294.7
15 min., 1 week (first)	13.824	5.6777	29.7	254.6
15 min., 1 week (second)	13.3681	5.2681	24.2	349.3
15 min., 2 weeks	13.3248	5.4992	29.5	473.9

The last of the drag mismodeling cases was a combination of the previous two cases. This combination of noisy atmospheric density and a less accurate density model used in the orbit determination process should produce the most severe effects for drag mismodeling. The case lived up to these expectations. The fit span started with an extremely high error value of 12 kilometers, but dropped to a more reasonable value within

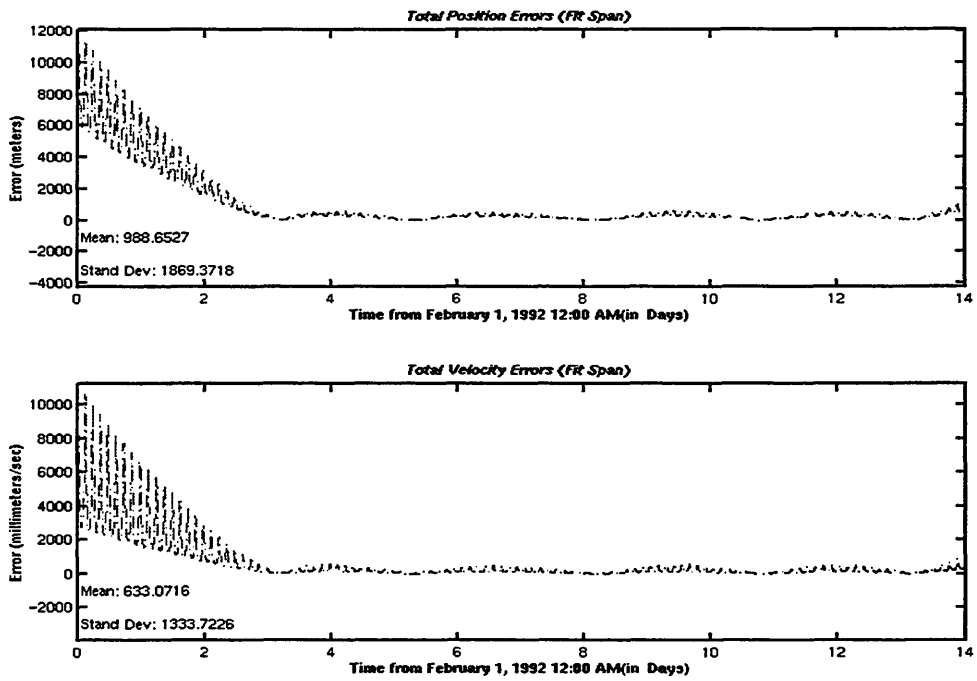


Figure 5.40 OD Errors in Borealis NAN - Mismodeling Drag, '92 Epoch and Harris-Priester (Fit Span)

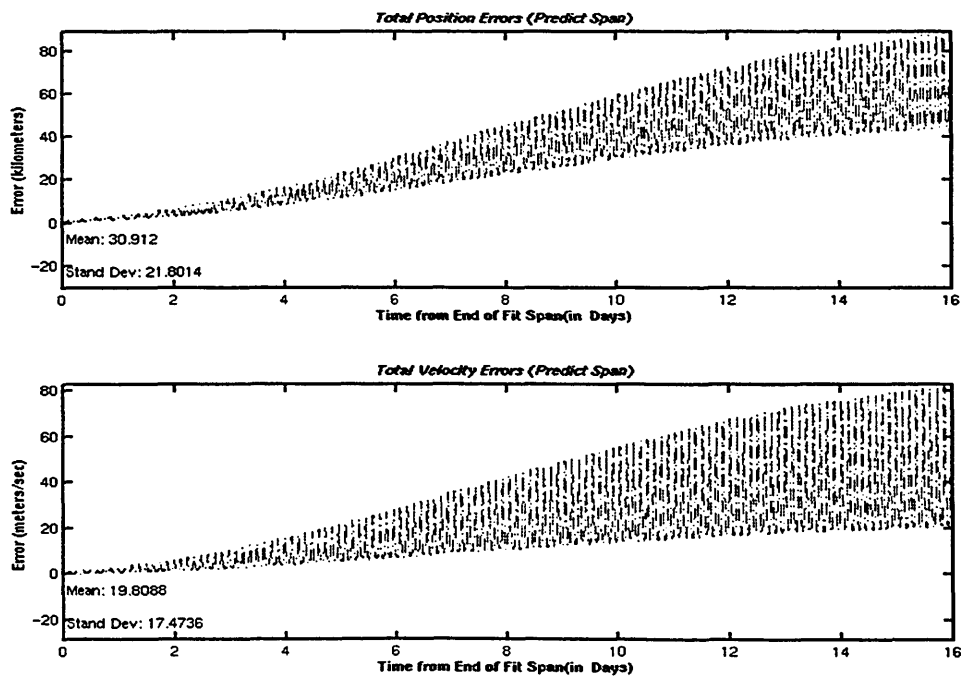


Figure 5.41 OD Errors in Borealis NAN - Mismodeling Drag, '92 Epoch and Harris-Priester (Predict Span)

the first 3 days; it also exhibited the same oscillating behavior seen in the mismodeling with Harris-Priester alone, but with an upper bound of approximately 600 meters. The error in the predict span displayed a rapid growth rate and climbed to a value of 89 kilometers.

The geopotential field was not left out of the mismodeling. Both the field size and the field model were changed in the course of the tests. The field size had a more pronounced effect on the orbit than the model type. A change in field size from 50 X 50 to 21 X 21 resulted in a total position error of approximately 3 km over sixteen days of prediction (Figure 5.42). A change in the field model from JGM2 to GEMT3 only produced differences of less than 8 meters in the fit span and 153 meters in the predict span (Figure 5.43 and 5.44).

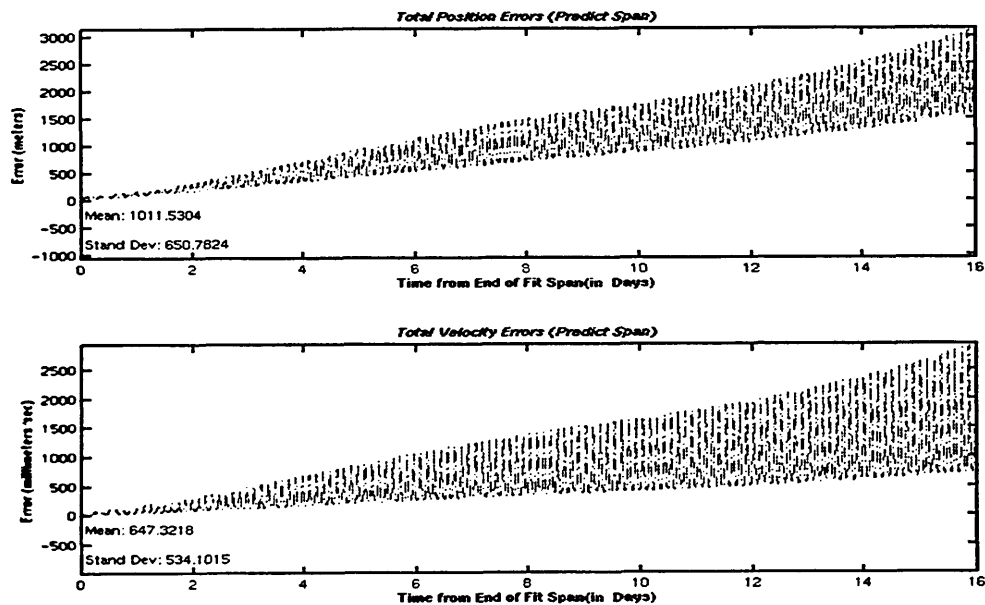


Figure 5.42 OD Errors in Borealis NAN - Mismodeled Geopotential Size, 21 x 21 (Predict Span)

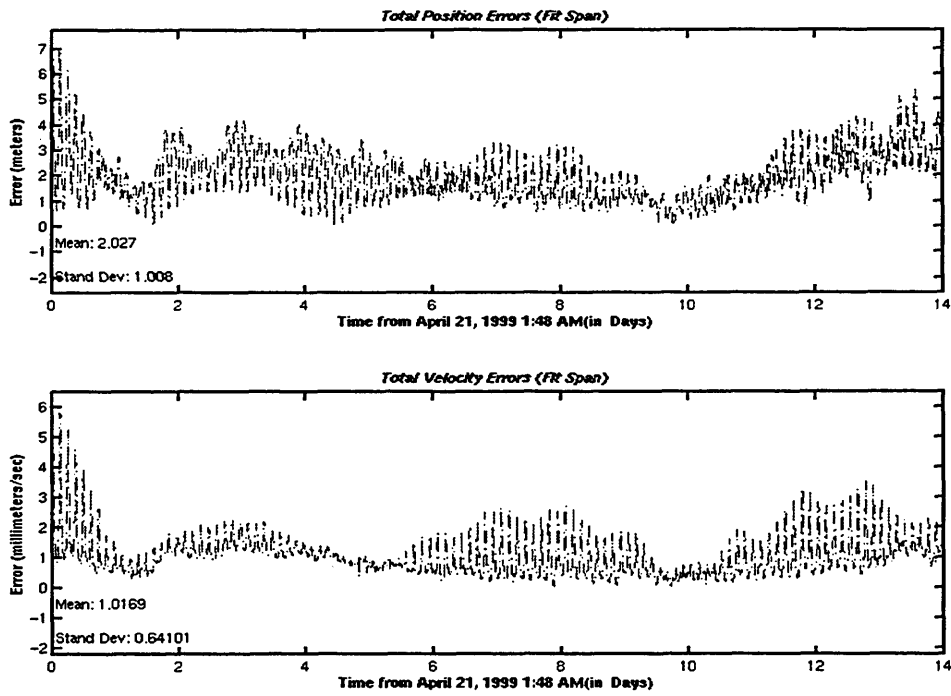


Figure 5.43 Difference between JGM2 21 x 21 and GEMT3 21 x 21 (Fit Span)

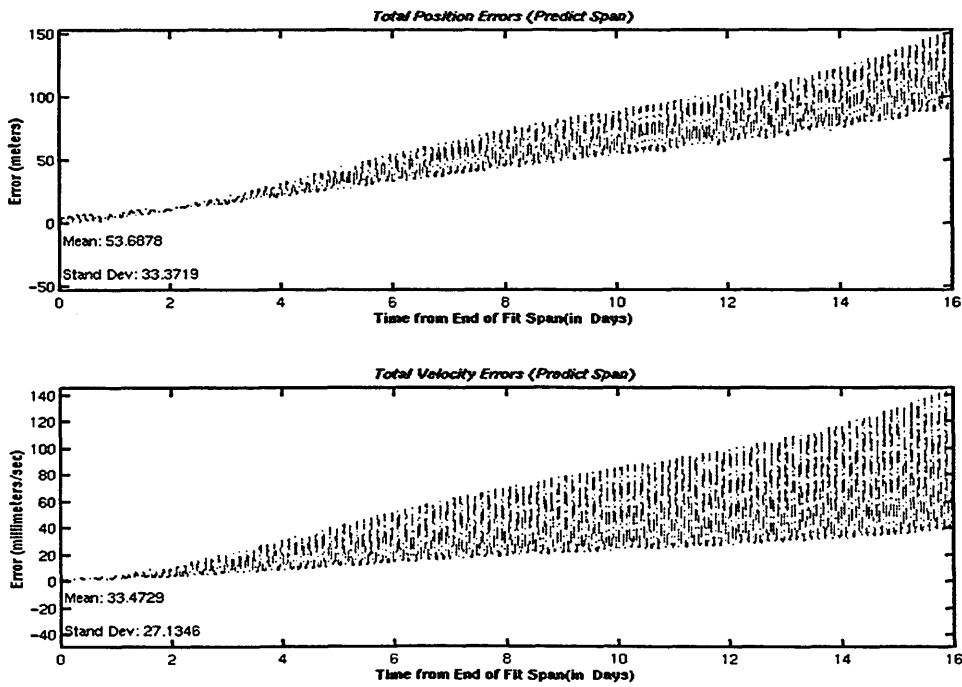


Figure 5.44 Difference between JGM2 21 x 21 and GEMT3 21 x 21 (Predict Span)

5.3.2 Concordia

Some attempts were made to mismodel the forces acting on Concordia, but these attempts were made in vain. Due to the altitude of the orbit, the effects of the geopotential field and drag are negligible. The only mismodeled force that affected this orbit is the force due to solar radiation pressure. However, the effects of this mismodeling were much less dramatic than those experienced by Borealis. The induced errors were an order of magnitude smaller (Figures 5.45 and 5.46).

5.3.3 Molniya

The main focus of mismodeling for the Molniya orbit was the geopotential field model. In the first test, the size of the geopotential field was reduced to a degree and order of 12. This reduction in the field induced total position errors on the order of 17 km after sixteen days of prediction (Figure 5.44). These errors swamped all of the other error contributions. While at this reduced field size, the geopotential model was changed from JGM2 to GEMT3. But the effects of the reduced field size overwhelmed any errors associated with this change.

The size for the geopotential field was increased in each subsequent test, from the initial 12 to 16, to 21, and finally to 36 (Tables 5.16 and 5.17). For each of the second and third tests, the size of the induced errors was reduced by half. By the time the geopotential size was brought up to 36, the errors were reduced to the same level as those experienced using the 50 X 50 geopotential model.

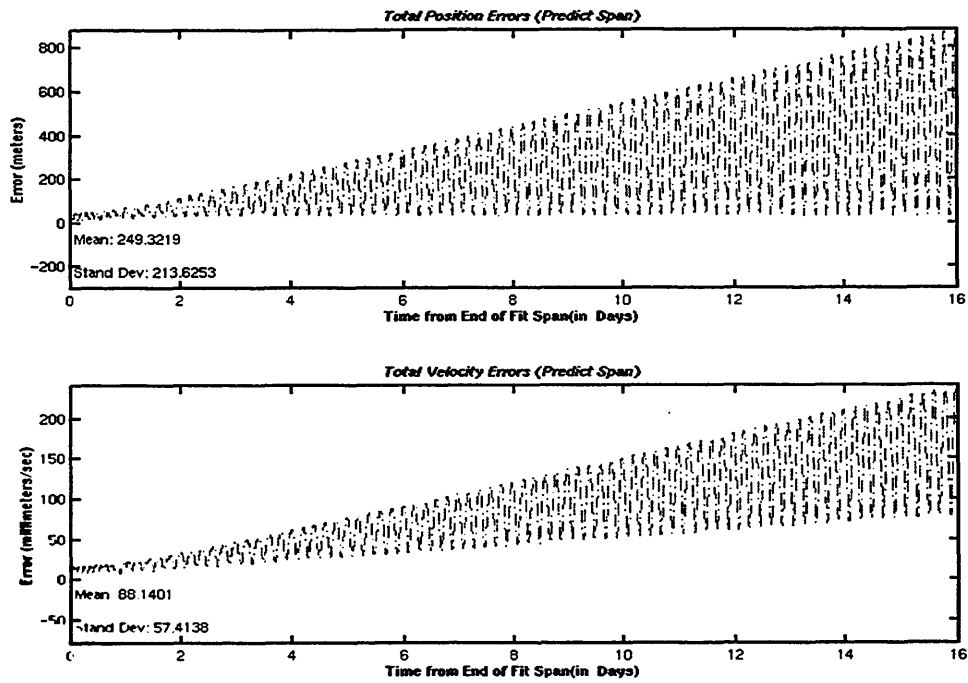


Figure 5.45 Prediction Errors in Concordia -Mismodeled Solar Radiation Pressure, Solar Reflectivity (-25%)

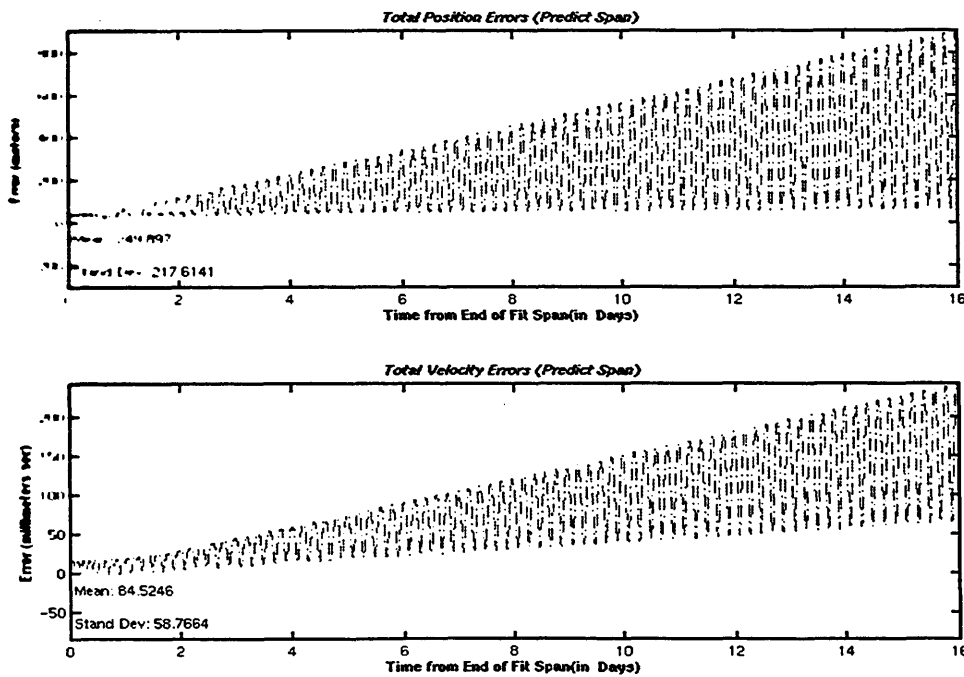


Figure 5.46 Prediction Errors in Concordia -Mismodeled Solar Radiation Pressure, Solar Reflectivity (+25%)

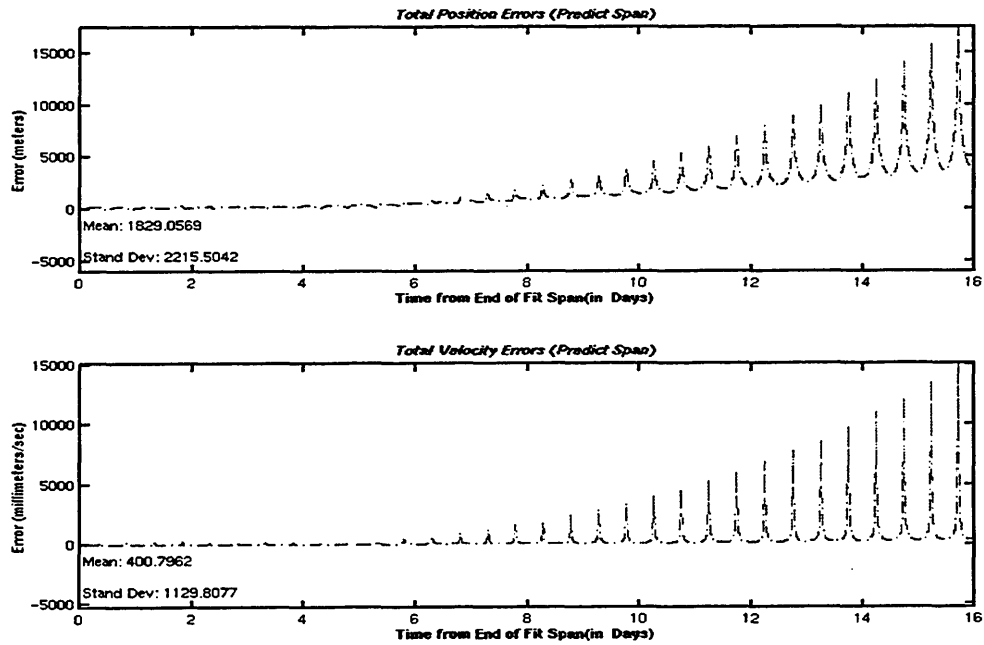


Figure 5.47 OD Errors in Molniya - Mismodeled Geopotential Size 12 x 12 (Predict Span)

Table 5.16 Molniya OD Position Errors (in meters) - Mismodeling Geopotential Field

Field and Size	Mean	Fit Span Std. Dev.	Max	Predict Span Max
JGM2, 12 x 12	140.1483	59.3275	316.7	17442
GEMT3, 12 x 12	140.1766	59.3834	319.4	16197
JGM2, 16 x 16	124.3468	55.2783	236.6	7882.1
JGM2, 21 x 21	121.2930	57.4639	210.6	3739.5
JGM2, 36 x 36	113.9074	54.5253	184.4	226.7

Table 5.17 Molniya OD Velocity Errors (in millimeters/sec) - Mismodeling Geopotential Field

Field and Size	Mean	Fit Span Std. Dev.	Max	Predict Span Max
JGM2, 12 x 12	16.6463	19.5807	231.0	15074
GEMT3, 12 x 12	16.7517	19.9572	234.4	14006
JGM2, 16 x 16	12.3146	9.9974	67.5	6744.4
JGM2, 21 x 21	12.8354	11.9002	101.1	3256.0
JGM2, 36 x 36	11.2981	9.2264	54.9	202.0

5.4 Disposition of OD Related Data

There are three groups of data that were used in the operation and analysis of the orbit determination process. They are the observation data, the orbits produced from the OD, and the files required for the operation of the OD. These data are stored in different locations in the Draper Laboratory computer environment.

The observation data is stored on DC1. It is placed under the directory `/usr/people/jgn4786/thesis/mpi/data`. Included with this data is all of the statistical data that was collected from the simulation during its operations.

The output from the orbit determination process is also stored on DC1. The directory for this data is `/usr/people/jgn4786/fit`. The individual orbits are stored in subdirectories of this main directory. These subdirectories are named after the orbits stored within. For example, the determined orbits for Molniya are in the `/molniya` subdirectory. Both Borealis orbits are in the `/borealis` subdirectory. But within that subdirectory, each have a lower subdirectory named for its node (`/nanoon` and `/namid`). The files in these directories include the output files from the OD process, the ASCII ORB1 files, and the stripped files for graphic processing.

The operational files are not on DC1, but on the VAX. These files were constructed to operate the VAX GTDS, so they were stored where they were most useful. The directory setup is similar to that described for the OD output files. Each orbit's operational files are stored in subdirectories of the `ELROND$DKA0:[JGN4786.THESIS]` directory. The subdirectories are also named after the orbits, with the same division under Borealis. To use these files, they must be copied into the thesis directory or the `J2000_SWAT_GTDS.COM` file must be copied into the appropriate subdirectory. During this project, the former was chosen to conserve space on the VAX. Once the operation of an individual process was complete, the copies of the operational files were removed from

the thesis directory. After porting the output over to DC1, the output files from the process were also erased from this directory. If these steps had not been taken, the VAX would not have been available for the OD processes.

Chapter 6

Conclusions and Future Work

This work examined the use of GPS navigation solutions as measurements in the orbit determination process for higher altitude elliptical orbits. Some concerns associated with this question are the visibility between such orbits and the GPS constellation, and the increase of ionospheric delay experienced when using a nadir facing GPS antenna. The current work is an extension of an earlier investigation [8] in which the use of actual navigation solutions as a measurement source was demonstrated for several LEO orbits (TOPEX, TAOS, and EUVE).

A prerequisite to the performance of this examination was the production of simulated GPS navigation solution data for the selected orbit determination process. The previous application of this method had used actual navigation solution data for several LEO satellites, but no such data was available for the higher altitude elliptical orbits. A simulation was constructed to produce an approximation of the GPS navigation solutions for use in orbit determination.

The simulation requires the true orbit of the satellite as input for determining its GPS navigation solutions. This orbit was generated using the Cowell special perturbations propagator in GTDS to compute the precise osculating orbit. The simulation uses this orbit to identify the true position and velocity of the satellite at any time during the simulation's run.

The simulation also requires a representation of the GPS constellation. This representation does not need to be as precise as the orbit of the receiver's satellite, but the satellite distribution characteristics are required to be similar to those of the actual GPS constellation. An analytical approximation of the GPS orbit was used with a generic epoch

for its initial conditions. The positions presented by this GPS approximation are used by the simulation as the true positions of the GPS satellites.

The simulation uses the positions of the receiving satellite and the GPS satellites at a given signal reception time to determine the ranges between them. The range between the receiver and each GPS satellite is converted to a time of signal transit, which includes the delays associated with passage through the ionosphere. Each GPS satellite's position is reevaluated based on its time of signal transmission. The geometry between the receiving satellite and each GPS satellite is evaluated to identify the GPS signals that could reach the receiving satellite. For each receivable signal, the times of transit are modified by the addition of realistic errors related to the performance of the GPS satellites' clocks while SA is in operation. The new transit times are subtracted from the reception time to determine the times that the receiver believes the GPS satellites began their transmissions. The positions of the GPS satellites at their perceived transmission times and the pseudoranges determined from the modified transit times are used as inputs to a navigation solution algorithm. This algorithm selects the four received GPS signals that have the best geometry and computes an initial estimate of the position. This estimate and all of the GPS signals are used as input to an iterative least squares approximation. The result of this least squares approximation is the GPS navigation solution for the given reception time.

This simulation could have been implemented in a sequential program, but the number of repetitive computations gave an indication that the simulation was a candidate for parallel processing. The simulation was implemented using the MPI Standard as the basis of its parallelization. The architecture selected for this program was a modified version of the master-slave architecture. The master's primary responsibility is the assignment of tasks, and all of the computations are left to the slaves. The slaves take their assigned time of signal reception, determine the target satellite's position, calculate the pseudoranges and ephemeris positions of the GPS satellites in view, and request their next assignments from the master. The slaves are not capable of depositing their results in a common file, and

passing any additional responsibilities to the master would slow its response to a request for a new assignment. So, a special slave was designed to provide a single point for output processing. Also included in the special slave's responsibilities is the performance of the navigation solution algorithm.

The simulation was tested to evaluate the performance of its algorithms and the quality of the navigation solutions it produces. Then, the navigation solutions from the simulation were used in the orbit determination process. In addition, an examination of the effects of force mismodeling was conducted.

6.1 Conclusions

The simulation has performed as expected. The patterns of GPS satellite acceptance match those predicted in the geometric analysis of the effects of receiver satellite altitude on two-way visibility between the GPS satellites and the receiver. The test of the navigation solution algorithm with no GPS related errors included in the simulation demonstrated accuracy to within 1.6 meters; each instance of elevated error was also accompanied with a high level of GDOP. A comparison of the simulated GPSR navigation solutions with actual GPSR navigation solutions for the TOPEX case revealed that the simulation is producing solutions with a similar "look and feel," but the precise statistics of these errors are somewhat different; contributions to this difference may be incomplete modeling, mismodeling of the ionosphere, using more simultaneous GPS signals than the actual receiver could handle, and rejecting potential navigation solutions due to poor geometry.

The results of orbit determination processes employing the simulated data and ideal force modeling are very promising. For all orbit configurations studied (Ellipso Borealis, Ellipso Concordia, and Molniya), the differential corrections process converged to a solution. The best results came from the navigation solutions using two antennas and

blocking the signals from below that passed through the ionosphere. The four minute observation density is the favored choice for performing the orbit determinations, and the longer fit spans usually produced better fits and predictions. While some of the tested satellites had exceptionally low levels of error in their predictions, all of them had errors of less than 50 meters after two weeks of predictions when using the favored antenna configuration with four minute measurement density and a two week fit span. However, all of the “truth” orbits and their ODs were performed using a smoothed model of the atmospheric density. Additional testing should be performed using a noisier representation of the atmosphere.

The effects of force mismodeling varied greatly with the orbital configuration. The OD for the Borealis orbits is susceptible to the mismodeling of solar radiation pressure, drag, and the geopotential field model and size. Of these forces, the mismodeling of drag had the greatest impact on the accuracy of the Borealis ODs. The Concordia OD is only affected by the mismodeling of solar radiation pressure; because of its high altitude and low eccentricity, Concordia experiences very small accelerations due to drag and the geopotential field, so mismodeling them has little effects. The Molniya OD was only tested for mismodeling of the geopotential field; the errors from the truncated geopotential decreased by a factor of 2 for every increase of 4-5 in the geopotential field size from a minimum size of 12 x 12. Additional mismodeling tests should be performed on all of the target orbits to determine any functional relations between these forces and the errors they produce. Because of the low altitude of the selected Molniya orbit, tests should be conducted on its response to drag mismodeling.

6.2 Recommendations

Orbits determined from GPS navigation solutions can be extremely accurate. But the accuracy depends on the quality, duration, and density of the navigation solutions.

The best quality navigation solutions for these higher orbits are always produced using the two-antenna configuration with an algorithm to ignore signals received by the nadir antenna that have passed through the ionosphere. In addition to the quality, the two-antenna configuration also provides more navigation solutions at high altitudes, and it has a higher altitude ceiling to its measurement capability than either antenna alone.

An observation density of one navigation solution every four minutes appears to be the best for orbit determination purposes. Higher densities seem to provide little improvement on the OD performance. Lower densities can be selected for applications that have a lower accuracy requirement.

An adequate duration for the observations depends on the length of time desired for the predict span, the size of errors allowable in the predict span, and any unmodeled effects in the OD. An example of the dependence on unmodeled effects is drag variability; if the OD treats drag as a constant in a noisy drag environment, longer durations can produce larger errors. Before selecting the duration, the characteristics of the desired predict span and the unmodeled effects should be identified, and the impact of these unmodeled effects should be evaluated.

Potterveld suggested the use of arcs of navigation solutions as opposed to a continuous stream [77]. This approach was suggested as a way to reduce the power consumption of the GPS receiver. This method sounds promising. In fact, a similar type of segmenting occurred in the simulation naturally from the lack of visibility at higher altitudes (zenith antenna configuration for Borealis, two antenna configuration for Molniya), and the OD still produced orbits with reasonable levels of error. To verify that

this method is appropriate for an individual orbital configuration, the OD performance with this segmented data should be determined.

Unless the GPS receiver antennas being used on a satellite have a small reception cone, the only reason to employ more than two antennas for position purposes is for redundancy. By the time a satellite reaches 4000 km altitude, its nadir and zenith antenna combination using antennas with a reception half-beam angle of 70° can receive signals from all GPS satellites that are capable transmitting to its position. Additional antennas will be required if the satellite is also using GPS for its attitude determination.

The Ellipso spacecraft (Borealis and Concordia) and the two antenna nadir-zenith configuration are an excellent match for mutual performance. To achieve the best communication performance, these spacecraft maintain a nadir pointing direction at all times. By mounting the antennas with their central axes on or parallel to this pointing axis, the antenna directions will be maintained by the altitude control system of the spacecraft without any special considerations. This feature in combination with the increased navigation solution capability makes the antenna configuration an exceptional choice for Ellipso.

6.3 Future Work

This simulation is a first attempt at modeling GPS navigation solutions. There are several ways in which the accuracy of the modeling can be improved. Some of the error sources that were ignored could be included, as could the GPS ephemeris errors that were intended to be in this simulation. Also, the ionospheric delay model could be upgraded. Another potential improvement to the simulation is the inclusion of relativistic effects. Perhaps the carrier phase of the GPS signals could be computed and used to enhance the accuracy of the navigation solutions.

But the use of this work in future projects is not limited to improving the error models. It could also include examinations of even higher orbits, like GEO or geosynchronous transfer orbits (GTO). The existing navigation solution algorithm could be removed and replaced with a Kalman filter algorithm for determining the target's position. Or the program architecture may be isolated from the algorithms and used in a different parallel processing application.

The next sections describe some of the model improvements that may be implemented in the simulation at some future time.

6.3.1 GPS Ephemeris Errors

As stated in Section 3.2.3.3, there are a number of ways that GPS ephemeris errors could be added to the simulation. But the candidate that is most likely for initial implementation is the addition of errors to the GPS satellite position as perceived by the target satellite.

One possible method for determining the amount of error to be added to the perceived position is to generate random Gaussian errors with zero mean and a selected standard deviation. This could be accomplished by generating the magnitude of the error and determining a random direction for its application. For the random direction, a Gaussian random value, with zero mean and standard deviation of one, is computed for each of the three components. These components are combined into a vector that is increased or reduced in size relative to a unit vector. The unit vector is multiplied by the magnitude of the error, and the result is added to the GPS position.

6.3.2 Improving the Ionospheric Model

The ionosphere is currently modeled as a thick shell around the Earth with a uniform electron density distribution. This implementation was selected to provide a starting point

and to present a worst case scenario. But the electron density of the ionosphere is not uniform. It varies with altitude, location above the Earth, local time of day, and wind currents. Each of these variation types could be modeled, with the possible exception of the effects of the wind.

The least complex of the improvement concepts relates to the variation with altitude. The electron density can be determined for each point along the path based on the altitude, and the instantaneous delay resulting from that density is accumulated in the total ionospheric delay. This accumulation of instantaneous delay may be achievable through analytical means.

To model the variations with location and local time, the precise location of each point of the path through the ionosphere must be known to determine the electron density. These models would add complexity to the simulation.

6.3.3 Relativistic Effects

Relativistic effects were mentioned in Section 1.1.2 in the description of corrections to the time of transmission, but these effects were not described. These effects are the result of observing a body in a moving reference frame while both the observing and observed reference frames are being accelerated. On Earth, these effects are related to the velocities of the GPS satellites, the change in gravitational potential between the GPS satellites and the users, and the rotation of the Earth. The result of these effects is a change in the time and time rate measured by the user. For instance, when observed from the Earth, the GPS satellite clocks experience a constant frequency offset. There are also effects observable from Earth that are related to the small eccentricity of the GPS orbits, and to the rotation of the Earth (similar to the idea of the Coriolis effect). The changes associated with these relativistic effects are significant because of the scale of the values being measured in the GPS navigation solutions.

For GPS receivers in orbit, these effects can be quite different. Their effects are related to the velocities and eccentricities of both the GPS constellation and the receiver satellite itself. It may also experience a completely different frequency offset from that experienced by users on the Earth's surface.

To implement the changes from relativistic effects into the simulation, an algorithm must be developed that can compute the timing changes based on the GPS constellation and the target orbit. This algorithm may also have to model the prelaunch frequency adjustment that is performed on each GPS satellite to compensate for the constant frequency offset described earlier in this section.

(This page intentionally left blank.)

Appendix A

Software Creation and Modification

This appendix presents the software that was created or modified in the creation of the simulation described in this thesis. The new software is divided into two cases, new programs and stubs. The stubs are the hooks for subroutine that have not been created yet, but are planned for the future.

A description of each new program is provided here, along with a list of its connections to other subroutines. Only the names, purposes and the next higher subroutines are provided for the stubs. For the single modified subroutine, `gmatinv.for`, a description of the change is presented.

Following the descriptions of the new and modified software, the source code of the main program, the master, the slave, and the navslave is presented. The code for these programs and all programs described in this appendix is stored in Draper's Data Center 1 (DC1), a Silicon Graphics Origin 2000 computer using the IRIX 6.4 operating system; the programs are in directory `/usr/people/jgn4786/thesis/mpi`.

A.1 New Programs

This section is a listing of the new programs created for the simulation. The main program and the top level subroutines are presented as the main divisions of this section. The lower subroutines are listed alphabetically under their top level subroutines (master, slave and navslave). Each entry consists of the subroutine's name, a brief description of its purpose, its calling sequence, a list of the subroutines that call it, and a list of the subroutines it calls.

A.1.1 `gps_pred.for`

This is the main program of the simulation. It is responsible for setting up and closing the MPI environment. It also performs a test to verify that at least three processes have been requested for the programs operation. The underscore in the program name was placed intentionally to make it easier to distinguish from the executable.

calling sequence: `mpirun -np # gpspred`
 where # is the number of requested processes

called by: N/A

calls to: `master.for`
 `MPI_INIT.for`
 `MPI_FINALIZE.for`
 `MPI_COMM_RANK.for`
 `MPI_COMM_SIZE.for`
 `navslave.for`
 `slave.for`

A.1.2 master.for

This subroutine performs the operations of the master in the self-scheduling process. It acquires the initialization data and distributes it to the appropriate process. It assigns the initial tasks and waits for the slaves to request additional tasks. Requests are processed on a first come, first served basis. If no more tasks remain, the reply to the slave is a shutdown command. When all slaves have received the shutdown command, the master returns to the main program.

calling sequence: call master(navid,numprocs)
called by: gps_pred.for
calls to: MPI_RECV.for
 MPI_SEND.for
 startup.for

startup.for

This subroutine establishes values to be used in the simulation. These values are the epoch time, the initial time since epoch, the total number of time steps, the size of the time steps, and the input type identifier. All of these values are returned to the master.

calling sequence: call startup(itime, stepsize, nsteps, epochday, epochsec, inputype)
called by: master.for
calls to: julian.for

A.1.3 navslave.for

This subroutine is the gatekeeper for the navigation solution. It receives the navigation data from the slaves. If this data contains at least 4 GPS satellites, the subroutine decides how the a priori estimate should be made and passes the data into the navigation solution algorithm. Once the algorithm is complete or if it was bypassed, navslave resets its variables and starts again. It returns to the main program after the last data set is processed.

calling sequence: call navslave(mastid,ecase,tcase)
called by: gps_pred.for
calls to: GETENV.for
 init_headers.for
 init_navinfo.for
 MPI_RECV.for
 navfix.for

init_navinfo.for

This subroutine sets the initial values for the arrays and variable used in the calculation of navigation solutions. The amount of the navdat array to be initialized is dependent on the value of ecase. These values are stored in navinfo.h and can be called upon by any subroutine in the navslave's process.

calling sequence: call init_navinfo(ecase)
called by: navslave.for
calls to: N/A

geocheck.for

This subroutine converts the vectors from the a priori calculation matrix into unit vectors, computes the normal to the first two vectors, and determines the angle between the third vector and the computed normal. This angle is compared to 90° and the compared value is returned.

calling sequence: call geocheck(priomat,geoval)
called by: navfix.for
calls to: cross.for
 dot33.for
 mag.for

navfix.for

This subroutine computes the navigation solution from the GPS data and stores it in an external file. It initializes its conditional test and stores the GPS data in local variables. Unless they are included in the GPS data, it determines the ephemeris positions of the received GPS satellites. It selects the four satellites to be used for the a priori estimate, tests the geometry of this selected set and computes the a priori position estimate. Or, it extrapolates the a priori estimate from previous navigation solutions and elapsed time. If the geometry passes the test, the iterative least squares solution is computed and stored, and the extrapolation data is updated for the next time. Otherwise, the subroutine ends without finding a solution.

calling sequence: call navfix(tc case)
called by: navslave.for
calls to: geocheck.for
 gps_ephem.for
 gmatinv.for
 mag.for
 matset.for
 matvec.for
 mave33.for
 outnav.for
 selectsat.for
 trucoef.for

outnav.for

This subroutine deposits the navigation solution and the statistical data from the navigation solution into two separate files. It time-codes the data based on the epoch and the elapsed time in the simulation.

calling sequence: call outnav(time,priori,gdop,cndmin,cndmax,geoval)
called by: navfix.for
calls to: aldiff.for
 calndr.for

selectsat.for

This subroutine selects the four GPS satellites to be used in the a priori position estimate for the navigation solution. It constructs a matrix for each combination of four GPS satellites and computes the determinant. All combinations with a non-zero determinant are compared to find the one with the largest magnitude of determinant. That combination is returned to the navfix subroutine.

calling sequence: call selectsat(gpseph, selsat)
called by: navfix.for
calls to: N/A

A.1.4 `slave.for`

This subroutine performs the slave operations for the self-scheduling process. It initializes its variables and accepts start time and input type data from the master. It enters its main loop and receives its first assignment. It finds the position of the target satellite and the times of transmission for the acceptable GPS satellites. It computes the clock errors and finds the perceived time of transmission for these GPS satellites. From these times, it finds the GPS ephemeris positions. When all 24 satellites have been evaluated, the slave transmits the number of acceptable satellites and the navigation data to the navslave. Then, it resets its variables and requests a new assignment. When the assignment is the shutdown command, the slave returns to the main program.

calling sequence: `call slave(mastid, myid, navid, ecase, tcase)`

called by: `gps_pred.for`

calls to: `ephcoef.for`
 `gpstime.for`
 `init_clock.for`
 `init_slaveinfo.for`
 `MPI_RECV.for`
 `MPI_SEND.for`
 `recvdgps.for`
 `satclock.for`
 `tarpos.for`

antenna.for

This subroutine calculates a unit vector in the direction of the receiving antenna's central axis. To accomplish this task, it computes the normal vector to the orbital plane by taking the cross product of the position and velocity of the target satellite. This normal is once again crossed with the target's position to produce a vector that is within the orbital plane, normal to the target's position vector, and facing the general direction of forward motion. This vector is then rotated about the normal to the orbital plane by an angle that is passed into the subroutine in its call. The resulting vector is reduced in magnitude to one and the unit vector is returned.

calling sequence: call antenna(rotang,uant)
called by: verify.for
calls to: cross.for
 mag.for
 rotvec.for

delays.for

This subroutine sums the signal delays related to ionospheric and tropospheric effects, multipath phenomena, and receiver measurements. It returns the delay value in units of seconds.

calling sequence: call delays(dtime)
called by: gpstime.for
calls to: ioneq.for
 mpather.for
 recmeas.for
 tropef.for

difference.for

This subroutine takes the difference of two three-component vectors.

calling sequence: call difference(vec1,vec2,diff)
called by: gpstime.for
 ionef.for
 verify.for
calls to: N/A

gpstime.for

This subroutine determines the time of signal transmission of a GPS satellite. It is given the time of reception by the target satellite and the identification of the GPS satellite. First, it determines the position of the GPS satellite at the time of signal reception. From this position and the position of the target at the same time, it calculates the range between them. This range is converted to transit time. The transit time is subtracted from the reception time to acquire the first estimate of the time of transmission. The process is repeated to refine the time of transmission, but all subsequent iterations of the process include the signal delays in the transit time. After determining the time of transmission, a test is performed to determine if the target (at the reception time) can receive the signal that the GPS satellite sent at the indicated time and location of transmission. If the satellite is accepted, the time of transmission is returned.

calling sequence: call gpstime(time,satnum,tsent,tcase)
called by: slave.for
calls to: delays.for
 difference.for
 get_sat.for
 gpstru.for
 mag.for
 verify.for

init_clock.for

This subroutine sets the initial values for the matrices and variable used in the calculation of the satellite clock error. These values are stored in clock.h and can be called upon by any subroutine in the slave's process.

calling sequence: call init_clock
called by: slave.for
calls to: gauss.for

init_slaveinfo.for

This subroutine sets the initial values for the arrays and variable used in the slave subroutine. These values are stored in slaveinfo.h and can be called upon by any subroutine in the slave's process.

calling sequence: call init_slaveinfo(ecase)
called by: slave.for
calls to: N/A

ionef.for

This subroutine computes the signal delay from ionospheric effects. This computation treats the ionosphere as if the electron density were constant at all depths and locations. The path length through the ionosphere is calculated and multiplied by the ionospheric delay per unit of path length. The ionospheric delay is returned in units of seconds.

calling sequence: call ionef(delion)
called by: delays.for
calls to: difference.for
 dot33.for
 mag.for

recvdgps.for

This subroutine computes the GPS ephemeris position at the time that the target believes the signal was transmitted. The ephemeris position is presented in Earth centered, Earth Fixed (ECEF) coordinates and adds ephemeris errors. The subroutine returns the ephemeris position with errors included.

calling sequence: call recvdgps(satnum,time, tcase, pos)
called by: slave.for
calls to: adderr.for
 get_sat.for
 gpstru.for
 mave33.for

rotvec.for

This subroutine rotates a vector about a given axis normal to the vector through a specified angle. A positive rotation moves the vector toward the direction of the cross product of the vector and the axis of rotation. The rotated vector is returned to antenna.

calling sequence: call rotvec(angle, axis, ivec, fvec)
called by: antenna.for
calls to: cross.for
 mag.for

satclock.for

This subroutine computes the total time error of the satellite clock using the second-order Gauss-Markov model. It integrates the state-space equations using two Gaussian white noise generators as input and computes range errors. If the desired time of output is not on the update grid, the range value is interpolated. The range errors are converted to time errors by dividing it by the speed of light, and returned.

calling sequence: call satclock(satnum, time, delclock)
called by: slave.for
calls to: gauss.for
 matvec.for

tarpos.for

This subroutine finds the position and velocity of the target satellite, given the desired time and the identification of the target satellite. It performs an initialization on its first call. Identification codes indicate the constellation and number of the target and the method for calculating the target's state. The position and velocity are not returned; they are stored in the structured data file slaveinfo.h.

calling sequence: call tarpos(time, satnum, constnum)
called by: slave.for
calls to: init_bsaf.for
 get_bsaf.for
 get_ext_sat.for

verify.for

This subroutine verifies that the target satellite can receive the signal from a particular GPS satellite. It computes the values necessary to determine the angles between the vectors. With these angles determined, they are tested against the maximum allowed values. If any test fails, the GPS satellite is rejected.

calling sequence: call verify
called by: gpstime.for
calls to: antenna.for
 difference.for
 dot33.for
 mag.for

A.2 Stubs

Stubs are the hook-up points for subroutines that have not been written yet. Listed below are the stubs installed in this simulation and the subroutines that call them. These stubs are really functionality that has remained in the preliminary design stage.

<u>Stub</u>	<u>Purpose</u>	<u>Parent Subroutines</u>
adderr.for	Adding ephemeris errors	recvdgps.for
ephcoef.for	Compute coefficients for ephemeris calculation	slave.for
gps_ephem.for	Compute ephemeris position from calculated coefficients	navfix.for
gpstru.for	Compute GPS position from actual data	gpstime.for, recvdgps.for
mpather.for	Compute multipath delays	delays.for
recmeas.for	Compute receiver measurement delays	delays.for
tropef.for	Compute tropospheric delays	delays.for
trucoef.for	Compute ephemeris position from stored coefficients	navfix.for

A.3 Modified Subroutines

There is only one modified subroutine, `gmatinv.for`. This subroutine is a modified version of the `matinv.for` subroutine. The modification has nothing to do with the calculations performed, only the output that is passed back to the calling subroutine.

`Matinv.for` is a subroutine that performs matrix inversion using the Moore-Penrose Generalized Inverse. During the inversion process, the norm of the inverse is calculated. The value of this norm for the geometry matrix G is the value of GDOP. The variable corresponding to this value was added to the calling sequence, and the value of GDOP is returned from this subroutine.

A.4 Source Code

A.4.1 Main Program - gps_pred.for

```
program gps_pred
C
C
C
C
C   Revision History *****
C   Rev  Date   Who/Where   Comments
C   1.0  07/22/98  JGN/CSDL   Initial design and coding
C
C   Description *****
C
C       This program performs the self-scheduling process for
C       eccentric orbit prediction using GPS.
C
C   Subroutines *****
C
C       master(navid,numprocs)
C       MPI_INIT(ierr)
C       MPI_FINALIZE(ierr)
C       MPI_COMM_RANK(communicator,myid,ierr)
C       MPI_COMM_SIZE(communicator,numprocs,ierr)
C       navslave(mastid,ephemcase,truthcase)
C       slave(mastid,myid,navid,ephemcase,truthcase)
C
C
C
C
C   Local Variable Descriptions *****
C
C   Integers
C       ephemcase   Identifier of the GPS ephemeris case
C                   1 = Errors added to truth
C                   2 = Coefficients calculated
C                   3 = Coefficients in file
C       ierr        MPI error flag
C       mastid      Identification tag of the master process
C       myid        Identification tag of any process
C       navid       Identification tag of the navigation fix
C                   process
C       numprocs    Number of processes
C       truthcase   Identifier of the GPS "truth" case
C                   1 = Simulated
C                   2 = Actual
C
C
C   Declarations *****
C
C   implicit none
C
C   Header Files *****
```

```

C
include "mpif.h"
C
C
C
Local Variables *****
C
integer*4 ephemcase
integer*4 ierr
integer*4 mastid
integer*4 myid
integer*4 navid
integer*4 numprocs
integer*4 truthcase
C
C
C
Initialize MPI
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)
C
C
C
Verify that there are enough processors
C
if (numprocs .ge. 3) then
C
C
C
Initialize variables
    mastid = 0
    navid = numprocs-1
    ephemcase = 1
    truthcase = 1
C
    if ( myid .eq. mastid ) then
        write(*,*) 'numprocs = ',numprocs
        call master(navid,numprocs)
    else
        if (myid .eq. navid) then
            call navslave(mastid,ephemcase,truthcase)
        else
            call slave(mastid,myid,navid,ephemcase,truthcase)
        endif
    endif
else
C
C
C
Notify user of processor shortage
    write(*,*) 'gpspred requires at least 3 processors.'
    write(*,*) 'Please try again with acceptable number.'
endif
C
C
C
Shut down MPI
call MPI_FINALIZE(ierr)
stop
end

```



```

C
C Header Files *****
C
include "mpif.h"
C
C Local Variables *****
C
integer*4 ans
integer*4 i
integer*4 inputype
integer*4 ierr
integer*4 k
integer*4 navid
integer*4 nsteps
integer*4 numprocs
integer*4 sender
integer*4 stat(MPI_STATUS_SIZE)
integer*4 step

C
real*8 epochday
real*8 epochsec
real*8 initinfo(3)
real*8 itime
real*8 stepsize
real*8 time

C
C Initialize variable
C
step = 1

C
C Acquire initial time, stepsize, and number of steps
C
call startup(itime,stepsize,nsteps,epochday,epochsec,inputype)
C
C Send initial information to nav slave
C
initinfo(1) = dble(nsteps)
initinfo(2) = epochday
initinfo(3) = epochsec

call MPI_SEND(initinfo, 3, MPI_DOUBLE_PRECISION, navid, 1,
$   MPI_COMM_WORLD, ierr)
C
C Send slaves first tasks
C
do i = 1,numprocs-2
write(*,*) 'assigning task ',i
C
C Calculate time of signal reception
C
time = itime + stepsize * (dble(i) - 1.d0)
call MPI_SEND(inputype, 1, MPI_INTEGER, i, step,
$   MPI_COMM_WORLD,ierr)
call MPI_SEND(itime, 1, MPI_DOUBLE_PRECISION, i, step,

```

```

$      MPI_COMM_WORLD,ierr)
      call MPI_SEND(time, 1, MPI_DOUBLE_PRECISION, i, step,
$      _COMM_WORLD,ierr)
      step = step + 1
    enddo
    step = numprocs-1
C
C      Receive response from slave
C
    do k = 1,nsteps
      call MPI_RECV(ans, 1, MPI_INTEGER, MPI_ANY_SOURCE,
$      MPI_ANY_TAG, MPI_COMM_WORLD, stat, ierr)
      sender = ans
C
C      Assign new task to same slave
C
      if (step .le. nsteps) then
        time = itime + stepsize * (dble(step) - 1.d0)
        call MPI_SEND(time, 1, MPI_DOUBLE_PRECISION, sender,
S      step, MPI_COMM_WORLD, ierr)
      else
        call MPI_SEND(0.d0, 1, MPI_DOUBLE_PRECISION, sender, 0,
S      MPI_COMM_WORLD, ierr)
      endif
      step = step + 1
    enddo
  return
end

```

A.4.3 navslave.for

```
subroutine navslave(mastid,ecase,tcase)
C
C
C
C
Revision History *****
C
Rev   Date   Who/Where   Comments
C
1.0   07/22/98   JGN/CSDL   Initial design and coding
C
Description *****
C
      This program performs the navigational fix operations for
C
      the GPS orbit prediction.
C
Subroutines *****
C
      GETENV (ENV_INPUT,INDATA_PATH)
C
      init_headers
C
      init_navinfo(ecase)
C
      MPI_RECV(buffer,size,type,source,tag,communicator,
C
               stat,ierr)
C
      navfix(tcase)
C
Local Variable Descriptions *****
C
Integers
C
      ecase      Identifier of the GPS ephemeris case
C
      i          Incrementer for initializing navdat array
C
      ierr       MPI error flag
C
      mastid     Identification tag of the master process
C
      nsteps     Total number of time steps
C
      source     Source of nav data array that has
C
               just been sized
C
      stat       Status array of received message
C
      step       Time step being evaluated
C
      tcase      Identifier of the GPS "truth" case
C
Parameters
C
      toler      Tolerance for reinitialization
C
Reals
C
      initinfo   Initial information for navslave, including
C
               total number of time steps and run epoch
C
      reinit     Condition for reinitializing nav fix
C
Declarations *****
C
implicit none
C
Header Files *****
C
include "mpif.h"
INCLUDE 'anfil.h'
include 'navinfo.h'
```



```

C
C Constants *****
C
real*8 toler
parameter (toler = 3.0d1)
C
C Local Variables *****
C
integer*4 ecase
integer*4 i,idum
integer*4 IENDIN
integer*4 ierr
integer*4 mastid
integer*4 nsteps
integer*4 source
integer*4 stat(MPI_STATUS_SIZE)
integer*4 step
integer*4 tcase
C
character*(ANLFIL_MAX_PATH_LEN) INPATH
character*(ANLFIL_MAX_PATH_LEN) INDATA_PATH
character*(ANLFIL_MAX_PATH_LEN + 20) input
C
real*8 initinfo(3)
real*8 reinit
C
C Initialize variables
C
call init_headers
call init_navinfo(ecase)
step = 1
CALL GETENV (ANLFIL.ENV_INPUT,INDATA_PATH)
IENDIN = INDEX(INDATA_PATH,' ')
INPATH = INDATA_PATH(1:IENDIN-1)
input = INPATH(1:IENDIN-1)//'timecoef'
OPEN(UNIT=ANLFIL.NTCORR,FORM='UNFORMATTED',
1 ACCESS='DIRECT',RECL=248,FILE=input,STATUS='OLD',
2 READONLY,SHARED)
C
C Receive initial information from master
C
call MPI_RECV(initinfo, 3, MPI_DOUBLE_PRECISION, mastid, 1,
$ MPI_COMM_WORLD, stat, ierr)
nsteps = int(initinfo(1))
navinfo.epochday = initinfo(2)
navinfo.epochsec = initinfo(3)
C
C Receive data for navigation fix from slave
C
do while (step .le. nsteps)
call MPI_RECV(navinfo.oksat, 1, MPI_INTEGER,
$ MPI_ANY_SOURCE, step, MPI_COMM_WORLD, stat, ierr)
source = stat(MPI_SOURCE)

```

```

    call MPI_RECV(navinfo.navdat, navinfo.oksat*(2
$      + navinfo.ndat)+1, MPI_DOUBLE_PRECISION,
$      source, step, MPI_COMM_WORLD, stat, ierr)
    if (navinfo.oksat .ge. 4) then
        reinit = navinfo.navdat(1) - toler
        if (navinfo.time .le. reinit) then
            navinfo.init = .true.
        endif
    endif
C
C      Calculate navigation fix
C
        call navfix(tcase)
    endif
C
C      Increment step, and reinitialize variables
C
        do i = 1,navinfo.oksat*(2+navinfo.ndat)+1
            navinfo.navdat(i) = 0.d0
        enddo
        step = step + 1
    enddo
C
    return
end

```



```

C          ephdat      GPS ephemeris data (for ecase of 1 or 2)
C          navdat      Array of data required for navigational fix
C          start       Reception time of first execution
C          time        Reception time of current execution
C          tsent       Time of signal transmission from GPS sat
C
C  Declarations *****
C
C  implicit none
C
C  Header Files *****
C
C  include "mpif.h"
C  include 'clock.h'
C  include 'slaveinfo.h'
C  include 'matrix.h'
C
C  Local Variables *****
C
C  integer*4 addon
C  integer*4 ans
C  integer*4 ecase
C  integer*4 i
C  integer*4 idummy(9)
C  integer*4 ierr
C  integer*4 j
C  integer*4 mastid
C  integer*4 myid
C  integer*4 navid
C  integer*4 ndat
C  integer*4 oksat
C  integer*4 stat(MPI_STATUS_SIZE)
C  integer*4 step
C  integer*4 tcase
C  integer*4 idum
C
C  real*8 delclock
C  real*8 ephdat(SLAVE_DATA2)
C  real*8 navdat(SLAVE_NGPS*(2+SLAVE_DATA2)+1)
C  real*8 start
C  real*8 time
C  real*8 tsent
C  real*8 dummy(6,8)
C  real*8 b(3,3)
C  real*8 tpos(3)
C
C  Initiatize variable
C
C  step = 1
C  oksat = 0
C  ndat = 0
C  call init_clock
C  call init_slaveinfo
C  do i = 1,SLAVE_NGPS*(2+SLAVE_DATA2)+1

```

```

        navdat(i) = 0.d0
    enddo
    if (ecase .eq. 1) then
        ndat = SLAVE_DATA1
    elseif (ecase .eq. 2) then
        ndat = SLAVE_DATA2
    endif
    call MPI_RECV(slaveinfo.inputype, 1, MPI_INTEGER, mastid,
$       MPI_ANY_TAG, MPI_COMM_WORLD, stat, ierr)
    call MPI_RECV(start, 1, MPI_DOUBLE_PRECISION, mastid,
$       MPI_ANY_TAG, MPI_COMM_WORLD, stat, ierr)
C
C   Slave receives message
C
    do while (step .ne. 0)
        call MPI_RECV(time, 1, MPI_DOUBLE_PRECISION, mastid,
$       MPI_ANY_TAG, MPI_COMM_WORLD, stat, ierr)
        step = stat(MPI_TAG)
        if (step .ne. 0) then
            ans = myid
            navdat(1) = time
C
C   Find positions of target sat and GPS sats
C
            call tarpos(time,SLAVE_TNUM,SLAVE_TCON)
            call eval(time,2,2,2,2,1,idummy,dummy)
            b(1,1) = cos(matrix.gha)
            b(2,1) = -sin(matrix.gha)
            b(3,1) = 0.d0
            b(1,2) = sin(matrix.gha)
            b(2,2) = cos(matrix.gha)
            b(3,2) = 0.d0
            b(1,3) = 0.d0
            b(2,3) = 0.d0
            b(3,3) = 1.d0
            call mave33(tpos,b,slaveinfo.tpos)
            do i = 1,SLAVE_NGPS
                call gpstime(time,i, tsent,tcase)
                if (slaveinfo.flag) then
                    oksat = oksat + 1
                    addon = (oksat-1)*(2+ndat)
                    navdat(addon+2) = dble(i)
C
C   Calculate satellite clock errors and modify transmission time
C
                    if (step .eq. 1) then
                        delclock = clock(i).xmat(1)/(SLAVE_C*1.0d3)
                    else
                        call satclock(i,tsent-start,delclock)
                    endif

                    navdat(addon+3) = tsent - delclock
C
C   If the ephemeris case requires it, compute ephemeris data

```

```

C
        if (ndat .gt. 0) then
            if (ndat .eq. SLAVE_DATA1) then
                call recvdgps(i,navdat(addon+3),tcase,ephdat)
            elseif (ndat .eq. SLAVE_DATA2) then
                call ephcoef(i,navdat(addon+3),tcase,ephdat)
            endif
            do j = 1,ndat
                navdat(addon+j+3) = ephdat(j)
            enddo
        endif
    endif
enddo

C
C
C    Send data to navslave for processing
C
C        call MPI_SEND(oksat, 1, MPI_INTEGER, navid, step,
C            $          MPI_COMM_WORLD, ierr)
C        call MPI_SEND(navdat, oksat*(2+ndat)+1,
C            $          MPI_DOUBLE_PRECISION, navid, step,
C            $          MPI_COMM_WORLD,ierr)
C
C
C    Reset navdat array and oksat
C
C        do i = 1,oksat*(2+ndat)+1
C            navdat(i) = 0.d0
C        enddo
C        oksat = 0
C
C
C    Notify master about task completion
C
C        call MPI_SEND(ans, 1, MPI_INTEGER, mastid, ans,
C            $          MPI_COMM_WORLD, ierr)
C        endif
C    enddo
C
C    1000 format (4f21.12)
C    return
C    end

```

Appendix B

The Goddard Trajectory Determination System (GTDS)

B.1 The History of GTDS/DSST

Development of the GTDS program began in the early 1970's at Goddard Space Flight Center (GSFC). The engineers at the Computer Science Corporation developed simple models for orbit propagation using non-singular orbital elements. This effort resulted in equations of motion for the mean elements based on analytical averaging for limited force models and based on numerical averaging for a broader set of force models. In 1974, A. Fuchs from NASA Goddard raised the question of developing a nonsingular, semianalytical theory that combined the best characteristics of existing numerical and semianalytical satellite theories; this question was answered with a combination of the generalized method of averaging, analytical averaging, and recursion that was implemented in GTDS. In 1976, GTDS's development split into two versions: the operational version, and the research and development (R&D) version.

In late 1978, Draper Laboratory received a R&D version of GTDS from NASA GSFC to support a project for the U.S. Air Force Space Division. This version of GTDS contained components of a semianalytic model for the mean element equations of motion, including zonal harmonic terms, solar and lunar point mass models, and a truncated tesseral harmonic resonance theory. GTDS development moved toward a semianalytic model that separates the orbital motion into its mean behavior and higher frequency (short) periodic motion. Draper technical staff enhanced the mean element theory to gain high precision calculations and implemented the short periodics as functions of the mean elements. The

short periodics included: zonal harmonic terms, tesseral m-dailies, linear combinations of tesseral motion, atmospheric density models, and perturbations from solar radiation pressure. These modeled behaviors and the creation of routines to calculate the partial derivatives completed the initial development of the Semianalytic Satellite Theory (DSST) [13]. DSST provides high precision values for the mean orbital elements and short periodic motion. The first generation of the DSST included the following physical models:

Mean Element Equations of Motion

- Central-body gravitational spherical harmonics of arbitrary degree and order (zonals and tesseral resonance) based on the 21 x 21 geopotential
- J_2 -squared second order effect (model based on eccentricity truncation)
- Third-body point mass effects (both single and double averaging theories)
- Atmospheric drag with J_2 /drag coupling
- Solar radiation pressure with eclipsing
- Integration Coordinate System based on the FK4 coordinate frame

Short-Periodic Motion

- Central-body gravitational zonal harmonics of arbitrary degree based on the 21 x 21 geopotential
- Central-body gravitational m-daily sectoral and tesseral harmonics of arbitrary degree and order based on the 21 x 21 geopotential
- Central-body gravitational high-frequency sectoral and tesseral harmonics of arbitrary degree and order based on the 21 x 21 geopotential
- J_2 -squared 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

State Transition Matrix

- General concept based on the Generalized Method of Averaging

General (applicable to both the mean element and short periodic motion)

- Numerical interpolation techniques
- Modified representations of the Hansen coefficients
- Jacchia-Roberts atmospheric density models

In 1982, Aerospace Corporation had requirements for a high precision mean element propagator to facilitate long term mission analysis, including maneuver planning. In a cooperative effort, B. Baxter from Aerospace and P. Cefola from Draper organized a project to separate the DSST models from GTDS to produce a "Standalone" DSST program that was easily portable to non-IBM operating systems [16]. A major portion of this technical effort was completed by L. Early. The resulting Standalone code was comprised of 200 subroutines. Their functions were pulled from GTDS subroutines and, when necessary, were rewritten to improve the code. The program resulting from these new routines was portable, but could not operate without an additional interface layer. These interfacing programs [16] were also constructed by Mr. Early. The program became operational in 1984 [9].

Draper Lab's technical staff have continued the development of GTDS. Some of their improvements are:

- Addition of sources and coordinate systems for Precise Conversion of Element observation data.
- Maintenance and upgrade of the data bases used in physical models (geopotential models, solar activity/geomagnetic index files for the Jacchia-Roberts atmospheric density model, solar/lunar/planetary (SLP) ephemeris and timing coefficient files)
- Conversion of the source code to FORTRAN 77 and improving the structure [22]
- Inclusion of external satellite theories, including NORAD General Perturbations theories (GP's) and the Naval Space Command second Position, Partials, and Time theory (PPT2)
- Development of recursive estimation techniques (Kalman filters) matched to the DSST

Several MIT graduate students, supported by the Draper/MIT educational program, have assisted in development process. Among these students were D. Fonte and S. Carter.

Fonte performed research on the gravity field model [22]. His research revealed that the DSST recursions designed for the gravity field with a degree and order of 21 were sufficiently stable to support a gravity field of larger size. He implemented an increase in the potential size of the gravity field from 21 x 21 to 50 x 50. This implementation was

performed in a much more flexible manner than the original 21 x 21 gravity field, and left the opportunity open for further size increases. The results of his verification showed that the implementation of the new gravity field was accomplished correctly. This modification affected the Cowell orbit generator and DSST.

S. Carter made two contributions to GTDS in the course of his research [8]. First, he expanded the program's reference frame capabilities to include the J2000 system as well as the previously existing mean of 1950 system; these reference frames reflect motion based on the mean equator and equinox on given dates. Also, he refined the program by including a model of the perturbations due to solid Earth tides, the redistribution of the Earth's mass caused by the solar and lunar pull of gravity. These modifications built on all previous work and resulted in orbit accuracy on the order of one meter; this one meter accuracy was based on a comparison with externally generated TOPEX orbits and was obtained for both Cowell and DSST.

In the time between Fonte's departure and S. Carter's arrival, improvements began on the Radarsat baseline GTDS program [48]. These improvements were made in three areas: the incorporation of a new function that predicts motion during and after a maneuver, the presentation of output in NORAD format, and the debugging of the existing code. These enhancements were accomplished over a two year period, and were made independently of Fonte's changes. The work of S. Carter built on these changes.

The most recent project for Draper GTDS development was the investigation of highly elliptical orbits. During their decay phase, orbits of this type pass lower in the Earth's atmosphere than standard low Earth orbits (LEO) and require higher order drag terms and a new atmospheric model for orbit prediction. These elliptical orbits also require the inclusion of a J_2 -squared term [57] in its zonal harmonics. This investigation and the introduction of the MSIS-90 atmospheric model into the PC version of GTDS was accomplished by 2nd Lt. J. Fischer during the course of his Master's program [20].

Draper Laboratory is not the only facility that does research and development on GTDS. Currently, GTDS exists in many different versions. Improvements made at different locations by many people cause each version to be unique in overall function. A single uniform version of GTDS with all of the functionality for each of the current versions is a desirable commodity. A consolidation of GTDS modifications is in progress at Draper; the result will be this uniform version that everyone can use.

Another issue of concern to GTDS is the need to improve the maintainability of the source code. This concern was first expressed in 1994 [7] and a coding standard was established [11], but the process has not begun for GTDS.

B.2 Orbit Propagators

The perturbed equations of motion cannot be solved directly by analytical means. Several methods have been devised to incorporate the deviations from classical two-body motion resulting from drag, solar radiation pressure, effects of a non-spherical gravity field, solar and lunar tidal effects, and other perturbations. These methods have been divided into three categories: special perturbations, general perturbations, and the semianalytic method. Draper R&D GTDS has the benefit of a selection of orbit propagators based on each of these categories.

B.2.1 Special perturbations

Special perturbation methods are based on the numerical integration of the perturbed equations of motion, incorporating all of the accelerations associated with the force models. Through the use of small step sizes, these integrations provide extremely accurate results. However, the calculations consume large amounts of computational time. As computers

continue to improve in both speed and memory, the computational time is becoming less of an issue. Based on their level of usage, people will decide if the improved accuracy is worth the extra time for each run. Draper R&D GTDS has the following options from this category:

Cowell
 Time-regulated Cowell
 Variation of Parameters (VOP)

B.2.1.1 Cowell

The first two methods listed are named for P. H. Cowell, who originated the idea of developing the equations of motion in rectangular (Cartesian) coordinates and integrating them with an algorithm comprised of multiple steps. The equations of classical two-body motion are derived from Newton's Laws of Gravity and Motion, and can be expressed as:

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3} \mathbf{r} = 0 \quad (\text{B.1})$$

where \mathbf{r} is the three dimensional radius vector from the primary body to the secondary body, and μ is the gravitational constant. Applying a disturbing acceleration to the equation produces:

$$\ddot{\mathbf{r}} = \mathbf{a}_p - \frac{\mu}{r^3} \mathbf{r} \quad (\text{B.2})$$

This equation can be divided into its vector component equations in Cartesian coordinates as follows:

$$\begin{aligned} \ddot{r}_x &= a_x + \frac{\mu}{r^3} x \\ \ddot{r}_y &= a_y + \frac{\mu}{r^3} y \\ \ddot{r}_z &= a_z + \frac{\mu}{r^3} z \end{aligned} \quad (\text{B.3})$$

where

$$r = \sqrt{x^2 + y^2 + z^2} \quad (\text{B.4})$$

Equations B.3 are second order differential equations. They could be reduced to first order by realizing that the velocity vector is the first derivative of the position vector ($\mathbf{v} = \dot{\mathbf{r}}$). By reducing the perturbing acceleration to its component level, these equations can be integrated through time to determine the position and velocity of a satellite.

B.2.1.2 VOP

While Cowell focused on numerical integration of the equations of motion, the VOP method presents satellite motion as a set of orbital elements. These elements are physical characteristics of the orbit which can be affected by perturbations. This method is most effective at representing perturbed orbital motion when the perturbing accelerations are small relative to unperturbed motion [63]. It is an efficient means of determining these accelerations when integration must be frequently restarted.

VOP presents the rates of the element set in the differential equation:

$$\frac{d\mathbf{c}}{dt} = \mathbf{f}(\mathbf{c}, t) \quad (\text{B.5})$$

where \mathbf{c} is a vector of orbital elements. Again, these elements are affected by the perturbations and are not constants as in two-body motion. At any given time, these elements correspond to an instance of unperturbed two-body motion [40]. However, the elements and this unperturbed equivalent vary with time as the elements are changed by the perturbing accelerations.

The perturbing accelerations can be divided into two types, conservative and non-conservative accelerations [2]. Conservative accelerations are a result of conservative forces and can be expressed as the gradient of a disturbing potential. Non-conservative accelerations are caused by phenomena that transfer energy away from the system, such as

drag, and are represented as a common vector. These accelerations are combined in the VOP equations, which as written as:

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3} \mathbf{r} = \mathbf{a}_d + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{r}} \right)^T \quad (\text{B.6})$$

where

\mathbf{a}_d is the non-conservative portion of of the perturbing acceleration
 $\left(\frac{\partial \mathbf{R}}{\partial \mathbf{r}} \right)^T$ is the conservative portion of the perturbing acceleration.

Using the facts that velocity is the time-based derivative of position and that position and velocity are both functions of time and the orbital elements, equation B.6 can be converted to the following form:

$$\frac{d\mathbf{v}(\mathbf{c}, t)}{dt} + \frac{\mu}{r^3} \mathbf{r}(\mathbf{c}, t) = \mathbf{a}_d(\mathbf{c}, t) + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{r}(\mathbf{c}, t)} \right)^T \quad (\text{B.7})$$

where

$$\frac{d\mathbf{r}(\mathbf{c}, t)}{dt} = \mathbf{v}(\mathbf{c}, t)$$

Since the elements are not constant the time derivatives should be expressed in terms of the partial derivatives of these elements. Using the chain rule, the derivatives of position and velocity become:

$$\begin{aligned} \frac{d\mathbf{r}}{dt} &= \frac{\partial \mathbf{r}}{\partial t} + \frac{\partial \mathbf{r}}{\partial \mathbf{c}} \frac{d\mathbf{c}}{dt} \\ \frac{d\mathbf{v}}{dt} &= \frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{v}}{\partial \mathbf{c}} \frac{d\mathbf{c}}{dt} \end{aligned} \quad (\text{B.8})$$

The only difference between perturbed and two-body motion is the inclusion of the element rates. By substituting equations B.8 into the equations of B.7 and eliminating the terms related to two-body motions, the following equations are derived:

$$\frac{\partial \mathbf{r}}{\partial \mathbf{c}} \frac{d\mathbf{c}}{dt} = 0 \quad (\text{B.9})$$

$$\frac{\partial \mathbf{v}}{\partial \mathbf{c}} \frac{d\mathbf{c}}{dt} = \mathbf{a}_d(\mathbf{c}, t) + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{r}(\mathbf{c}, t)} \right)^T$$

Note that the velocity is the same for perturbed and two-body motion.

The equations in B.9 can be combined to produce a single equation for the element rates. For simplification of further derivation, this merging is accomplished by multiplying each equation by the transpose of the partial derivative factor in the other equation and produces:

$$\mathbf{L} \frac{d\mathbf{c}}{dt} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}} \right)^T \left[\mathbf{a}_d + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{r}} \right)^T \right] \quad (\text{B.10})$$

where

$$\mathbf{L} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}} \right)^T \left(\frac{\partial \mathbf{v}}{\partial \mathbf{c}} \right) - \left(\frac{\partial \mathbf{v}}{\partial \mathbf{c}} \right)^T \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}} \right)$$

From this point, the equation is split into the conservative and non-conservative portions. Each segment is treated differently to determine its contribution to the element rates. The conservative accelerations are evaluated using the Lagrange equations of planetary motion and the non-conservative accelerations make use of Gauss's equations.

Conservative accelerations cause no net work during the course of an orbit. Lagrange's equations are used to evaluate the effects of forces that produce these accelerations. For the purposes of this evaluation, the accelerations from non-conservative forces are set to zero. The new equation is:

$$\mathbf{L} \frac{d\mathbf{c}}{dt} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}} \right)^T \left(\frac{\partial \mathbf{R}}{\partial \mathbf{r}} \right)^T$$

simplified to

$$\mathbf{L} \frac{d\mathbf{c}}{dt} = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{c}} \right)^T \quad (\text{B.11})$$

This equation can be solved for the element rates by taking the inverse of the Lagrange matrix \mathbf{L} .

$$\frac{d\mathbf{c}}{dt} = \mathbf{L}^{-1} \left(\frac{\partial \mathbf{R}}{\partial \mathbf{c}} \right)^T \quad (\text{B.12})$$

An alternate method to solving this equation is the use of the Poisson matrix. This matrix has been shown to be the negative inverse of the Lagrange matrix and produces:

$$\frac{d\mathbf{c}}{dt} = -\mathbf{P} \left(\frac{\partial \mathbf{R}}{\partial \mathbf{c}} \right)^T = \mathbf{P}^T \left(\frac{\partial \mathbf{R}}{\partial \mathbf{c}} \right)^T \quad (\text{B.13})$$

Gauss's equations are not limited in the functions they can evaluate. They can handle either conservative or non-conservative accelerations. But since the conservative accelerations have already been covered, only the non-conservative accelerations will be addressed with the Gauss equations. Using the same Poisson substitution that was used in the Lagrange equations and ignoring the disturbing potential, equation B.10 becomes:

$$\frac{d\mathbf{c}}{dt} = \mathbf{P}^T \left(\frac{\partial \mathbf{r}}{\partial \mathbf{c}} \right)^T \mathbf{a}_d \quad (\text{B.14})$$

The definition of the Poisson matrix states [2]:

$$\mathbf{P} = \frac{\partial \mathbf{c}}{\partial \mathbf{r}} \left(\frac{\partial \mathbf{c}}{\partial \mathbf{v}} \right)^T - \frac{\partial \mathbf{c}}{\partial \mathbf{v}} \left(\frac{\partial \mathbf{c}}{\partial \mathbf{r}} \right)^T \quad (\text{B.15})$$

When substituted into equation B.14, it becomes:

$$\frac{d\mathbf{c}}{dt} = \frac{\partial \mathbf{c}}{\partial \mathbf{v}} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{r}} \right)^T \mathbf{a}_d - \frac{\partial \mathbf{c}}{\partial \mathbf{r}} \left(\frac{\partial \mathbf{r}}{\partial \mathbf{v}} \right)^T \mathbf{a}_d \quad (\text{B.16})$$

The position and velocity are considered to be independent of each other, so the partial derivatives become:

$$\frac{\partial \mathbf{r}}{\partial \mathbf{r}} = \mathbf{I} \qquad \frac{\partial \mathbf{r}}{\partial \mathbf{v}} = \mathbf{0} \quad (\text{B.17})$$

and the resulting solution is:

$$\frac{d\mathbf{c}}{dt} = \frac{\partial \mathbf{c}}{\partial \mathbf{v}} \mathbf{a}_d \quad (\text{B.18})$$

The elemental equations equivalent to Equation B.18 can be expressed in a number of different reference frames. Battin presents them in a radial-normal-tangent frame, which produces the following equations [2]:

$$\begin{aligned}
\frac{da}{dt} &= \frac{2a^2 v}{\mu} a_{dt} \\
\frac{de}{dt} &= \frac{1}{v} \left[2(e + \cos f) a_{dt} - \frac{r \sin f}{a} a_{dn} \right] \\
\frac{di}{dt} &= \frac{r \cos \theta}{h} a_{dh} \\
\frac{d\Omega}{dt} &= \frac{r \sin \theta}{h \sin i} a_{dh} \\
\frac{d\omega}{dt} &= \frac{1}{ev} \left[(2 \sin f) a_{dt} + \left(2e + \frac{r}{a} \cos f \right) a_{dn} \right] - \frac{r \sin \theta \cos i}{h \sin i} a_{dh} \\
\frac{dM}{dt} &= n - \frac{b}{eav} \left[2 \sin f \left(1 + \frac{e^2 r}{p} \right) a_{dt} + \left(\frac{r}{a} \cos f \right) a_{dn} \right]
\end{aligned} \tag{B.19}$$

The GTDS Mathematical Specification presents these equations in the equinoctial orbit frame [25]:

$$\begin{aligned}
\frac{da}{dt} &= \frac{2\mathbf{v}}{n^2 a} \mathbf{a}_d \\
\frac{dh}{dt} &= \left[\frac{1}{\mu} \left[(2\dot{X}_1 Y_1 - X_1 \dot{Y}_1) \hat{f} - X_1 \dot{X}_1 \hat{g} \right] + \frac{k}{G} (qIY_1 - pX_1) \hat{w} \right] \mathbf{a}_d \\
\frac{dk}{dt} &= \left[-\frac{1}{\mu} \left[Y_1 \dot{Y}_1 \hat{f} - (2X_1 \dot{Y}_1 - \dot{X}_1 Y_1) \hat{g} \right] - \frac{h}{G} (qIY_1 - pX_1) \hat{w} \right] \mathbf{a}_d \\
\frac{dp}{dt} &= \left[\frac{1 + p^2 + q^2}{2G} Y_1 \hat{w} \right] \mathbf{a}_d \\
\frac{dq}{dt} &= \left[\frac{I(1 + p^2 + q^2)}{2G} X_1 \hat{w} \right] \mathbf{a}_d \\
\frac{d\lambda}{dt} &= \left[n - \frac{2}{na^3} \mathbf{r} + \beta \left(k \frac{\partial h}{\partial \mathbf{v}} - h \frac{\partial k}{\partial \mathbf{v}} \right) + \frac{1}{na^2} (qIY_1 - pX_1) \hat{w} \right] \mathbf{a}_d
\end{aligned} \tag{B.20}$$

where

- I = the retrograde factor
- n = the mean motion
- \mathbf{r} = Cartesian position in the inertial reference frame
- \mathbf{v} = Cartesian velocity in the inertial reference frame
- $\hat{f}, \hat{g}, \hat{w}$ = unit vectors in the equinoctial frame

The position and velocity in the equinoctial frame are given by the equations:

$$\begin{aligned}
 X_1 &= a[(1 - h^2\beta)\cos F + hk\beta\sin F - k] \\
 Y_1 &= a[(1 - k^2\beta)\sin F + hk\beta\cos F - h] \\
 \dot{X}_1 &= \frac{na^2}{r}[hk\beta\cos F - (1 - h^2\beta)\sin F] \\
 \dot{Y}_1 &= \frac{na^2}{r}[(1 - k^2\beta)\cos F - hk\beta\sin F]
 \end{aligned}
 \tag{B.21}$$

and the variables G and β are defined to be:

$$\begin{aligned}
 G &= na^2\sqrt{1 - h^2 - k^2} \\
 \beta &= \frac{1}{1 + \sqrt{1 - h^2 - k^2}}
 \end{aligned}
 \tag{B.22}$$

B.2.2 General perturbations

General perturbation methods use an analytical approach with series approximations of the deviation from two-body motion. The perturbations included in these methods are limited to those that can be modeled as analytical functions. This tactic reduces the accuracy of the solution, but it gains the benefits of simpler and quicker operation. These methods are unencumbered by time steps, and are not severely affected by changes in the initial conditions. For the special perturbation methods, a change in the initial conditions means the entire integration must be reprocessed because each subsequent integration step is dependent on the results of the previous one. The general perturbation methods that exist in Draper R&D GTDS are:

- Brouwer
- Brouwer-Lyddane
- Brouwer-Lyddane-Gordon
- Vinti
- NORAD Simplified General Perturbations (SGP)
- NORAD GP4
- NORAD DP4
- NORAD HANDE
- NAVSPASUR PPT2

Because these methods are not used for this project, no detailed descriptions will be given.

B.2.3 Semianalytic methods

The semianalytic methods for orbit propagation were devised to take advantage of the best traits of the special and general perturbation techniques. The high accuracy of numerical integration and the low computation requirements of the analytical formulations are combined by separating the orbital motion into its mean motion and a series of high-frequency periodic components (Figure B.1). Using the Generalized Method of Averaging (GMA), the short periodics are removed from the orbital motion, leaving a mean representation of the motion. The perturbations are divided into conservative and non-conservative accelerations. The conservative accelerations are derived using Lagrange's VOP equations and are averaged analytically. The non-conservative accelerations use Gauss's VOP equations and are averaged numerically. Since it is no longer constrained by the high frequency components, this mean motion can be numerically integrated using a much larger time step. The mean motion is propagated to the desired time, and the short periodics are recombined to maintain the high accuracy. Draper R&D GTDS currently has two semianalytic propagators available:

Draper Semianalytic Satellite Theory (DSST)
NORAD Semianalytic Theory (SALT)

B.2.3.1 DSST

The Semianalytic Satellite Theory uses a set of equations similar to the VOP equations derived earlier, but these equations represent the motion in terms of the mean elements as opposed to the osculating elements. These equations are derived from an altered form of the Lagrange and Gauss equations. The perturbations are still considered to be very small relative to the two-body motion, so the osculating equations of motion can be written in the following manner:

$$\frac{dc}{dt} = \varepsilon F_i(\mathbf{c}, l) \quad i = 1, 2, \dots, 5 \quad (\text{B.23})$$

$$\frac{dl}{dt} = n(\mathbf{c}_1) + \varepsilon F_6(\mathbf{c}, l)$$

where

- \mathbf{c} = the vector of the slowly changing elements
- l = the rapidly changing variable
- n = the mean motion of the spacecraft
- ε = the parameter associated with the perturbation F

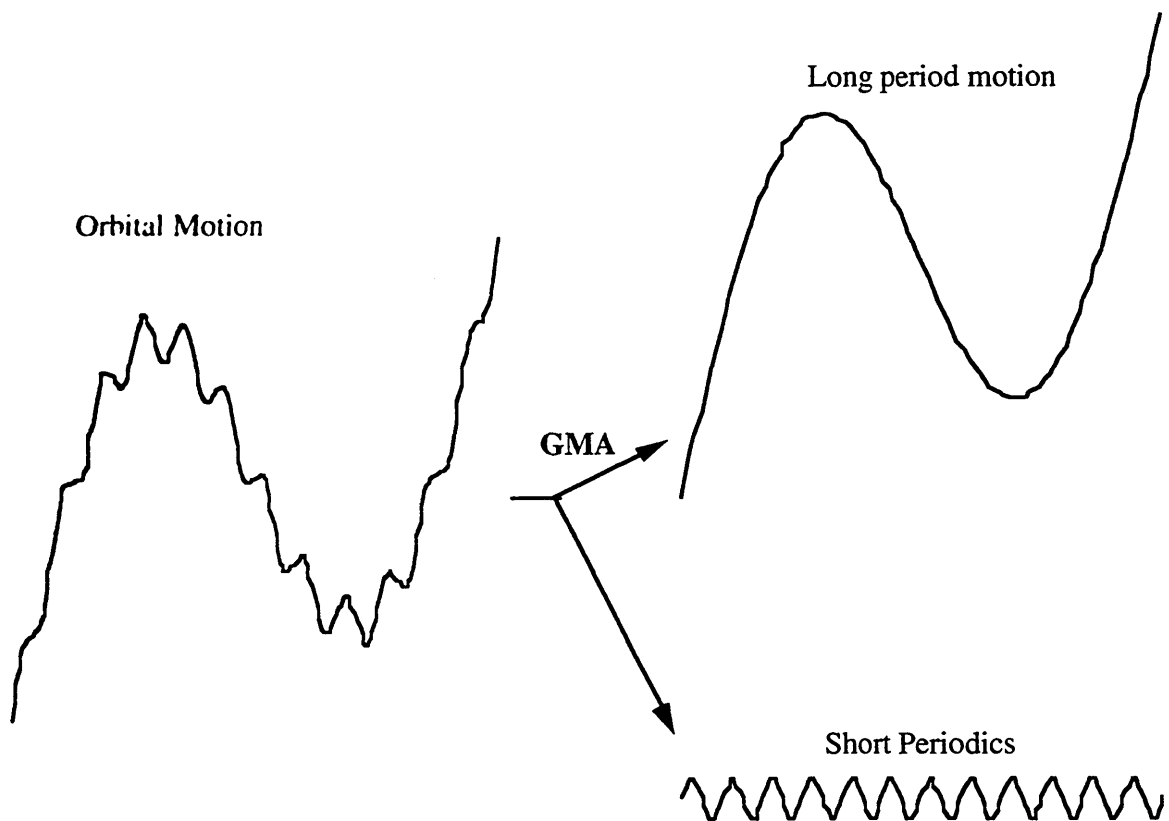


Figure B.1 Orbital Motion Decomposition

In order to get a mean elements representation of the equations of motion, the relationship between the mean and osculation elements must be identified. By using a near identity transformation, the osculation elements can be represented in the following equations:

$$\begin{aligned}
c &= \bar{c} + \varepsilon \eta_{i,1}(\bar{c}, \bar{l}) + \varepsilon^2 \eta_{i,2}(\bar{c}, \bar{l}) + \dots & i = 1, 2, \dots, 5 \\
l &= \bar{l} + \varepsilon \eta_{6,1}(\bar{c}, \bar{l}) + \varepsilon^2 \eta_{6,2}(\bar{c}, \bar{l}) + \dots
\end{aligned} \tag{B.24}$$

The overbar indicates the mean representation of the elements, and the remainder of each equation is a power series representation of the short periodics as a function of the mean elements. A short periodic has a period shorter than the orbital period of the spacecraft. Subscripts indicate the element and order associated to the series terms. All of the short periodic terms are assumed to be 2π periodic with respect to the rapidly changing variable:

$$\varepsilon \eta_{i,j}(\bar{c}, \bar{l}) = \varepsilon \eta_{i,j}(\bar{c}, \bar{l} + 2\pi) \tag{B.25}$$

Differentiating the equations of the osculating elements with respect to time produces the relationship between the mean and osculating element rates, which are expressed in the following equations:

$$\begin{aligned}
\frac{dc}{dt} &= \frac{d\bar{c}}{dt} + \varepsilon \frac{\partial \eta_{i,1}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \varepsilon^2 \frac{\partial \eta_{i,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \dots \\
\frac{dl}{dt} &= \frac{d\bar{l}}{dt} + \varepsilon \frac{\partial \eta_{6,1}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \varepsilon^2 \frac{\partial \eta_{6,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \dots
\end{aligned} \tag{B.26}$$

where \bar{X} is the six component vector of all the mean elements.

$$\bar{X} = \begin{bmatrix} \bar{c} \\ \bar{l} \end{bmatrix} \tag{B.27}$$

The rates of the mean elements are assumed to be functions of the slowly changing elements and the perturbation parameter ε , but not the rapidly changing variable. These rates are written as:

$$\begin{aligned}
\frac{d\bar{c}}{dt} &= \varepsilon A_{i,1}(\bar{c}) + \varepsilon^2 A_{i,2}(\bar{c}) + \dots & i = 1, 2, \dots, 5 \\
\frac{d\bar{l}}{dt} &= n(\bar{c}_1) + \varepsilon A_{6,1}(\bar{c}) + \varepsilon^2 A_{6,2}(\bar{c}) + \dots
\end{aligned} \tag{B.28}$$

where $A_{i,j}$ is the function corresponding to the i th element's j th order contribution to the rates of the mean elements.

Substituting the mean element rates into the equations for the osculating element produces the following equations:

$$\begin{aligned}\frac{dc}{dt} &= (\varepsilon A_{i,1}(\bar{c}) + \varepsilon^2 A_{i,2}(\bar{c}) + \dots) + \left(\varepsilon \frac{\partial \eta_{i,1}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \varepsilon^2 \frac{\partial \eta_{i,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \dots \right) \quad i = 1, 2, \dots, 5 \\ \frac{dl}{dt} &= (n(\bar{c}_1) + \varepsilon A_{6,1}(\bar{c}) + \varepsilon^2 A_{6,2}(\bar{c}) + \dots) + \left(\varepsilon \frac{\partial \eta_{6,1}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \varepsilon^2 \frac{\partial \eta_{6,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} + \dots \right) \quad (\text{B.29})\end{aligned}$$

and can be regrouped by powers of ε to produce:

$$\begin{aligned}\frac{dc}{dt} &= \varepsilon \left(A_{i,1}(\bar{c}) + \frac{\partial \eta_{i,1}}{\partial \bar{X}} \frac{d\bar{X}}{dt} \right) + \varepsilon^2 \left(A_{i,2}(\bar{c}) + \frac{\partial \eta_{i,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} \right) + \dots \quad i = 1, 2, \dots, 5 \\ \frac{dl}{dt} &= n(\bar{c}_1) + \varepsilon \left(A_{6,1}(\bar{c}) + \frac{\partial \eta_{6,1}}{\partial \bar{X}} \frac{d\bar{X}}{dt} \right) + \varepsilon^2 \left(A_{6,2}(\bar{c}) + \frac{\partial \eta_{6,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} \right) + \dots \quad (\text{B.30})\end{aligned}$$

These equations represent the left-hand side of the equations for osculating element rates presented in (B.23). These rates are equal to the perturbing functions displayed on the right-hand side of those equations. The perturbing functions can be expressed as a power series expansion in ε . Performing a Taylor series expansion on the perturbing functions about the mean elements produces:

$$F_i(\mathbf{c}, l) = F_i(\bar{c}, \bar{l}) + \frac{\partial F_i}{\partial \bar{X}} (\mathbf{X} - \bar{\mathbf{X}}) + \dots \quad i = 1, 2, \dots, 5 \quad (\text{B.31})$$

The coefficients of the partial derivatives are merely the difference between the osculating and mean elements, which can easily be extracted from (B.24).

$$\mathbf{X} - \bar{\mathbf{X}} = \varepsilon \eta_{i,1} + \varepsilon^2 \eta_{i,2} + \dots \quad i = 1, 2, \dots, 6 \quad (\text{B.32})$$

transforming (B.31) into:

$$F_i(\mathbf{c}, l) = F_i(\bar{c}, \bar{l}) + \frac{\partial F_i}{\partial \bar{X}} (\varepsilon \eta_{i,1} + \varepsilon^2 \eta_{i,2}) + \dots \quad i = 1, 2, \dots, 6 \quad (\text{B.33})$$

Since the mean motion in the equation for the rate of the rapidly changing is a function of the first osculating element, it is reasonable to express the mean motion as a Taylor expansion about that mean element and arrive at the equation:

$$n(\mathbf{c}_1) = n(\bar{\mathbf{c}}_1) + \frac{\partial n(\bar{\mathbf{c}}_1)}{\partial \bar{\mathbf{c}}_1} (\mathbf{c}_1 - \bar{\mathbf{c}}_1) + \frac{1}{2!} \frac{\partial^2 n(\bar{\mathbf{c}}_1)}{\partial \bar{\mathbf{c}}_1^2} (\mathbf{c}_1 - \bar{\mathbf{c}}_1)^2 + \dots \quad (\text{B.34})$$

By using the relationship between the mean motion and the semimajor axis, which is the first of the osculating elements, and the difference between the mean and osculating elements from (B.32), the Taylor series expansion can be expressed as the following power series in ε :

$$n(\mathbf{c}_1) = n(\bar{\mathbf{c}}_1) + \varepsilon \left(-\frac{3n(\bar{\mathbf{c}}_1)}{2\bar{\mathbf{c}}_1} \eta_{1,1} \right) + \varepsilon^2 \left(\frac{15n(\bar{\mathbf{c}}_1)}{8\bar{\mathbf{c}}_1^2} \eta_{1,1}^2 - \frac{3n(\bar{\mathbf{c}}_1)}{2\bar{\mathbf{c}}_1} \eta_{1,2} \right) + \dots \quad (\text{B.35})$$

Substituting (B.33) and (B.35) into the right-hand side of the equations in (B.23) results in the following equations:

$$\begin{aligned} \varepsilon F_i(\mathbf{c}, l) &= \varepsilon F_i(\bar{\mathbf{c}}, \bar{l}) + \frac{\partial F_i}{\partial \bar{\mathbf{X}}} (\varepsilon^2 \eta_{i,1} + \varepsilon^3 \eta_{i,2}) + \dots \quad i = 1, 2, \dots, 5 \\ n(\mathbf{c}_1) + \varepsilon F_6(\mathbf{c}, l) &= n(\bar{\mathbf{c}}_1) + \varepsilon \left(F_6(\bar{\mathbf{c}}, \bar{l}) - \frac{3n(\bar{\mathbf{c}}_1)}{2\bar{\mathbf{c}}_1} \eta_{1,1} \right) \\ &+ \varepsilon^2 \left(\frac{\partial F_6}{\partial \bar{\mathbf{X}}} \eta_{6,1} + \frac{15n(\bar{\mathbf{c}}_1)}{8\bar{\mathbf{c}}_1^2} \eta_{1,1}^2 - \frac{3n(\bar{\mathbf{c}}_1)}{2\bar{\mathbf{c}}_1} \eta_{1,2} \right) + \dots \end{aligned} \quad (\text{B.36})$$

Now that both sides of (B.23) have been expressed in terms of the mean elements, the equivalent expressions can be compared to each other. Equating the coefficients of like powers of ε produces the following equations:

First order

$$\begin{aligned} F_i(\bar{\mathbf{c}}, \bar{l}) &= A_{i,1}(\bar{\mathbf{c}}) + \frac{\partial \eta_{i,1}}{\partial \bar{\mathbf{X}}} \frac{d\bar{\mathbf{X}}}{dt} \quad i = 1, 2, \dots, 5 \\ F_6(\bar{\mathbf{c}}, \bar{l}) - \frac{3n(\bar{\mathbf{c}}_1)}{2\bar{\mathbf{c}}_1} \eta_{1,1} &= A_{6,1}(\bar{\mathbf{c}}) + \frac{\partial \eta_{6,1}}{\partial \bar{\mathbf{X}}} \frac{d\bar{\mathbf{X}}}{dt} \end{aligned} \quad (\text{B.37})$$

Second order

$$\frac{\partial F_i}{\partial \bar{\mathbf{X}}} \eta_{i,1} = A_{i,2}(\bar{\mathbf{c}}) + \frac{\partial \eta_{i,2}}{\partial \bar{\mathbf{X}}} \frac{d\bar{\mathbf{X}}}{dt} \quad i = 1, 2, \dots, 5$$

$$\frac{\partial F_6}{\partial \bar{X}} \eta_{6,1} + \frac{15n(\bar{c}_1)}{8\bar{c}_1^2} \eta_{1,1}^2 - \frac{3n(\bar{c}_1)}{2\bar{c}_1} \eta_{1,2} = A_{6,2}(\bar{c}) + \frac{\partial \eta_{6,2}}{\partial \bar{X}} \frac{d\bar{X}}{dt} \quad (\text{B.38})$$

But the time derivative of the mean elements, $\frac{d\bar{X}}{dt}$, has only one term that does not contain any powers of ε . Therefore, the first order terms produce the equations:

$$\begin{aligned} F_i(\bar{c}, \bar{l}) &= A_{i,1}(\bar{c}) + \frac{\partial \eta_{i,1}}{\partial \bar{X}} n(\bar{c}_1) & i = 1, 2, \dots, 5 \\ F_6(\bar{c}, \bar{l}) - \frac{3n(\bar{c}_1)}{2\bar{c}_1} \eta_{1,1} &= A_{6,1}(\bar{c}) + \frac{\partial \eta_{6,1}}{\partial \bar{X}} n(\bar{c}_1) \end{aligned} \quad (\text{B.39})$$

The mean equations of motion cannot be expressed while the short periodics and the rapidly changing variable still have an effect on the perturbing function, F. These high frequency effects must be stripped away from the mean motion. This task is accomplished by the use of the averaging operation, which is defined as:

$$\langle f(\bar{c}, \bar{l}) \rangle_i = \frac{1}{2\pi} \int_0^{2\pi} f(\bar{c}, \bar{l}) d\bar{l} \quad (\text{B.40})$$

The three properties of this operation that are most useful for the stripping process are:

$$\begin{aligned} \langle g \cdot f(\bar{c}, \bar{l}) \rangle_i &= g \cdot \langle f(\bar{c}, \bar{l}) \rangle_i \\ \left\langle \frac{\partial f(\bar{c}, \bar{l})}{\partial \bar{X}} \right\rangle_i &= \frac{\partial}{\partial \bar{X}} \langle f(\bar{c}, \bar{l}) \rangle_i \\ \langle g + f(\bar{c}, \bar{l}) \rangle_i &= \langle g \rangle_i + \langle f(\bar{c}, \bar{l}) \rangle_i \end{aligned} \quad (\text{B.41})$$

where g is not related to f. The averaging operation is applied to (B.39), resulting in the equations:

$$\begin{aligned} \langle F_i(\bar{c}, \bar{l}) \rangle_i &= \langle A_{i,1}(\bar{c}) \rangle_i + \left\langle \frac{\partial \eta_{i,1}}{\partial \bar{l}} n(\bar{c}_1) \right\rangle_i & i = 1, 2, \dots, 5 \\ \langle F_6(\bar{c}, \bar{l}) \rangle_i - \left\langle \frac{3n(\bar{c}_1)}{2\bar{c}_1} \eta_{1,1} \right\rangle_i &= \langle A_{6,1}(\bar{c}) \rangle_i + \left\langle \frac{\partial \eta_{6,1}}{\partial \bar{l}} n(\bar{c}_1) \right\rangle_i \end{aligned} \quad (\text{B.42})$$

Earlier in this section, the assumption was made that the short periodic functions had a period of 2π in the rapidly changing variable. So, applying the averaging operation to these short periodics nulls them out.

$$\begin{aligned} \left\langle \frac{\partial \eta_{i,1}}{\partial \bar{l}} n(\bar{c}_1) \right\rangle_i &= 0 \\ \left\langle \frac{3n(\bar{c}_1)}{2\bar{c}_1} \eta_{i,1} \right\rangle_i &= 0 \end{aligned} \quad i = 1, 2, \dots, 6 \quad (\text{B.43})$$

reducing the equations in (B.42) to:

$$\left\langle F_i(\bar{c}, \bar{l}) \right\rangle_i = \left\langle A_{i,1}(\bar{c}) \right\rangle_i \quad i = 1, 2, \dots, 6 \quad (\text{B.44})$$

Once the perturbing function has been averaged over the mean of the rapidly changing variable, the averaged perturbing function should only be dependent on the slowly changing mean elements

$$\bar{F}_i(\bar{c}) = A_{i,1}(\bar{c}) \quad i = 1, 2, \dots, 6 \quad (\text{B.45})$$

When substituted into (B.28), these equations provide the mean equations of motion. These equations present the mean elements rates as functions of the first five, slowly changing mean elements. By removing the fast variable, it is now possible to integrate these equations much more quickly by employing larger time steps. The mean equations of motion are:

$$\begin{aligned} \frac{d\bar{c}}{dt} &= \varepsilon \bar{F}_i(\bar{c}) \\ \frac{d\bar{l}}{dt} &= n(\bar{c}_1) + \varepsilon \bar{F}_6(\bar{c}) \end{aligned} \quad i = 1, 2, \dots, 5 \quad (\text{B.46})$$

The mean elements are propagated to the point of the desired output. To restore accuracy to the calculations, the short periodics are reintroduced to the solution at the requested time. The short periodic perturbing functions are the difference between the osculating and mean perturbing functions. They can be recovered from (B.39) by

subtracting the perturbations associated with mean motion from the total perturbing function.

$$\begin{aligned}
 F_i^{sp}(\bar{\mathbf{c}}, \bar{l}) &= \frac{\partial \eta_{i,1}}{\partial \bar{l}} n(\bar{\mathbf{c}}_1) & i = 1, 2, \dots, 5 \\
 F_6^{sp}(\bar{\mathbf{c}}, \bar{l}) &= \frac{\partial \eta_{6,1}}{\partial \bar{l}} n(\bar{\mathbf{c}}_1) + \frac{3n(\bar{\mathbf{c}}_1)}{2\bar{\mathbf{c}}_1} \eta_{1,1} & (B.47)
 \end{aligned}$$

Solving these equations for the short periodic functions results in:

$$\begin{aligned}
 \eta_{i,1} &= \frac{1}{n(\bar{\mathbf{c}}_1)} \int_0^{\bar{l}} F_i^{sp}(\bar{\mathbf{c}}, \bar{l}) d\bar{l} & i = 1, 2, \dots, 5 \\
 \eta_{6,1} &= \frac{1}{n(\bar{\mathbf{c}}_1)} \int_0^{\bar{l}} \left(F_6^{sp}(\bar{\mathbf{c}}, \bar{l}) - \frac{3\eta_{1,1}}{2\bar{\mathbf{c}}_1} \right) d\bar{l} & (B.48)
 \end{aligned}$$

Using larger time steps allows the DSST to operate more quickly, but it must also be able to provide the mean elements and short periodics at any desired time. For the mean elements, this is accomplished by interpolation between points on the integration grid. The short periodics can also make use of interpolation techniques, but if the interpolation is invalid at the output time, a new interval is calculated that contains the time of interest. The short periodics are computed as a Fourier series expansion and added to the mean motion to produce the osculating orbit.

B.3 Operations

The Draper R&D GTDS uses the propagation techniques described above to perform a variety of different tasks. These tasks can be divided into the nine main functions in GTDS:

- Ephemeris Generation
- Differential Corrections
- Filtering
- Ephemeris Comparison
- Data Management
- Data Simulation

File Report Generation
Error Analysis
Early Orbit Determination

The ephemeris generation function produces a history of the orbit for a desired time frame. It propagates the orbit forward from a set of initial conditions using any of the available propagation methods. A selection of perturbations is provided for inclusion in the ephemeris generation. The output of this function consists of a report of the ephemeris production and files containing the ephemeris data that can be used as future input for GTDS or other orbit determination programs. These ephemeris files can be in the form of ORB1, ORBIT or EPHEM files.

The differential corrections (DC) function uses estimation theory to determine the orbital elements and variations in perturbation parameters. The function takes a set of initial conditions and a series of observations as its inputs, computes an orbit from the initial state using the indicated perturbation models, and compares the observations with its computed orbit. In an iterative process, the computed orbit is modified to minimize the sum of the squares of the residuals in a weighted least-squares fit. Some of the other parameters that can be solved for are the drag coefficient, solar reflectivity, and observation biases. The statistics of the iterations are presented in an output file, along with the results of the fit.

The filter function takes a different approach than the DC for estimation of the orbit. Instead of the DC's batch process, the filter uses sequential estimators that employ dynamic models of the motion. The linear and extended Kalman filters (LKF and EKF) update the orbital estimate for each observation, and they maintain information related to previous observation in the covariance matrix to enhance the next estimate. This technique produces the best approximation for the orbital motion from a set of observations. This function is not available in the GTDS version at GSFC, but Draper has incorporated this function with the capability to handle calculations in both Cowell and DSST.

The ephemeris comparison function determines the difference between ephemeris information contained in two ephemeris files produced by GTDS. The comparison can be

performed over any portion of the ephemeris files. The output can be represented in terms of radial, cross-track, and along-track vector components or in terms of latitude, longitude, and altitude. In addition, the output includes the element histories and differences associated to the compared ephemeris files in both Keplerian and equinoctial elements.

The data management function opens temporary files that will be used in other GTDS functions. The information in these working files is drawn from the GTDS databases. These files can be generated during the execution of the program that requires the file, or as an initialization procedure long before the file is needed.

The data simulation function generates simulated observations for use as input to support and test mission operations. The location and frequency of the observations can be specified, and can be modified to include errors relevant to the mission performance.

The file report function produces a report of the data stored in the GTDS databases. This function is normally used to identify the appropriate database for execution of a specific task.

The error analysis function examines the effects of variations in epoch state, observation accuracy, and other quantities relating to the tracking of a specific satellite. It accepts observation input that conforms to the observation types produced by the DC function.

The early orbit determination function is designed to produce an estimate for the initial conditions of an orbit when the a priori state, required for DC operation, is unavailable. GTDS performs this function using any one of the three following methods: the Gauss method, the Double R-Iteration method, and the Range and Azimuth method. The initial conditions can be estimated with as few as six observations using this function.

Three files are required to initiate the performance of the desired functions: an input card file, a command file, and an overrides file. The input card file identifies the functions to be executed and the options that will be use during the execution. The command file specifies the databases that will provide the data for the working files, and commands the

execution of the functions. To compensate for the lack of a short periodics input processor, the overrides file replaces the predefined values in VAX GTDS with the appropriate values for proper calculation of the short periodics.

The input card file contains a series of keyword cards. Each keyword card corresponds with one line of the input file, and consists of the identifying keyword and a series of input fields. Each keyword maps to an option in GTDS operation. Within the keyword cards, the input fields provide the ability to select options for that keyword. The construction of a keyword card's input fields allows for the inclusion of three integers and three reals. An example of a keyword card is provided in Figure B.2.

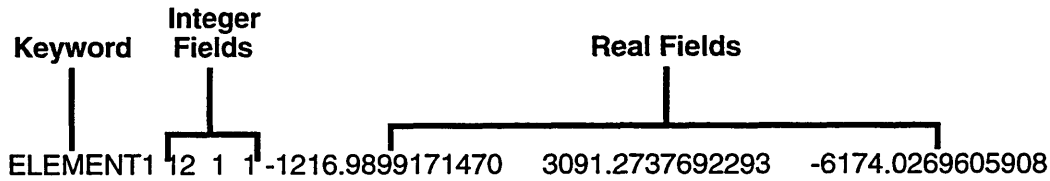


Figure B.2 Construction of a Keyword Card

In this example, ELEMENT1 is the keyword. The three integer fields indicate the input coordinate system, the type of elements being input, and the central body of the initial state. The three real fields contain the first three elements of the initial state. A listing of all keywords and their descriptions is presented in [76].

If the file only contains keyword cards, the input card file is not complete. It also requires a control card at the beginning of the file and a finishing card at the end. The control card identifies the function to be executed, and the finishing card marks the end of the input card file. The control card has a different construction than the keyword cards, which is explained in the Goddard Trajectory Determination System (GTDS) User's Guide [26].

For a particular control card, there is a series of required keyword cards necessary for the execution of the stated function. The number of required keyword cards varies from zero to five, depending on the function. For example, the ephemeris function

requires five control cards: 1) the epoch time for the start of the ephemeris, 2) and 3) the elements of the initial state, 4) the output type and end of ephemeris time, and 5) the type of propagation being performed. The keyword card displayed in Figure B.2 is an example of the first of the cards used to input the elements of the initial state.

The remaining cards are considered to be optional. But the desired option may not be directly accessible. Some of the options are grouped into subdeck based on the function being executed and the purpose of the desired option. Not all subdecks are constrained to a single function, but some are. For example, the COMPOPT subdeck is only useable by the compare function while DCOPT can be used in DC, data simulation, error analysis, early orbit determination, and filter functions. Within the input card file, the subdecks must be separated from the other optional cards by the subdeck identifier at the beginning and an END card at the end.

GTDS also has the capability of performing multiple functions in a single execution. Multiple input card files may be stored in a sequence in a single file, and GTDS will start the subsequent functions in order. If no changes have been indicated in the next function and the current function shares a working file with it, the new function will use the file from the previous one. This feature is convenient for instances when the output from one function is intended to be used as input to the next, such as a differential corrections run producing the initial conditions for an ephemeris generation.

The major functions of GTDS and its operation are described in more detail in the GTDS User's Guide [26].

Appendix C

Space Qualified GPS Receivers

As more study is conducted on the performance of GPS in space, companies are producing more space qualified GPS receivers. To compete with other manufacturers, these companies are also building more functions into their receivers. Some of these functions are: the use of Kalman filters for navigation solutions, attitude determination using GPS, radiation hardening, and algorithms for more precise navigation solutions. Another way these manufacturers compete is by producing receivers that perform the same functions, but are smaller and lighter and consume less power.

Table C.1 presents a listing of GPS receivers that have been used in space applications [8,64]. Each receiver is identified by its manufacturer and model. Along with the receiver identification is a listing of the receiver's past or projected missions. The reception characteristics of each model are provided and include the number of reception channels, the frequency capability, and the number of antennas. The table also identifies the functions that each receiver is capable of performing. The table does not present every receiver, but it does include the most recent receivers produced and those that are currently being developed [64].

Table C.1 Spaceborne GPS Receivers

Manufact.	Model	Mission	Chan.	Freq.	Ant.	Functions
AOA	Turbostar	GPS-MET, GFO, Wakeshield 3	8	Dual P	1	Prec. nav., timing, P&P codeless
APL	GNS	TIMED	12-72	Single	2	Filt. nav., timing, rad. hardened
Ashtech and E-systems	3DF upgrade	none	24	Single		Navigation, timing
Ashtech and E-systems	P-12 upgrade	none	12	Dual P		Navigation, timing
Ashtech and E-systems	OEM upgrade	none	12	Single		Navigation, timing
GSFC	PiVoT	Spartan 251	24	Single	2/4	Filt. nav., timing (attitude option)
Honeywell/ Trimble/ GSFC	SIGI	STS-96,97	12	Single	4	Attitude, navigation, timing
JPL	microGPS	SNOE	No limit	Single	1	"Bit Grabber," post-processed navigation
JPL/GSFC/ Stanford	GPS on a Chip	SAC-C, Champ, GRACE	12	Dual	4	Prec. nav., timing, attitude
Magnavox	GPSPAC	Landsat 4,5 DOD	2	Dual P	1	Navigation
Motorola	GPSDR	Explorer	12	Single	2	Navigation, timing
Motorola	GPSDR	TOPEX	6	Dual P	1	Navigation, timing
Motorola	Viceroy	MSTI-3, Seastar	12	Single	2	Navigation, timing
Motorola	Monarch		12	Dual P(Y)	2	Filt. nav., timing, rad. hardened
Rockwell	AST V	TAOS	6	Dual P	1	Navigation, timing
Rockwell	3M upgrade	NASA-JSC Shuttle	5	Dual P	1	Navigation, timing
Rockwell	Spaceborne 5-channel	P91-1 ARGOS	5	Dual P	1	Navigation, timing
Rockwell	SPINSAT	SPINSAT (canceled)	2	Dual P	1	Navigation, timing
Rockwell/ Collins	MAGR/S	Space Shuttle	5	Dual P(Y)	1	Navigation, timing
SS/L	Tensor	SSTI-Lewis, EO- 1	9	Single	4	Attitude, filt. nav., timing, rad. hardened

SSTL	SGR	TMSAT	12/24	Single	4/5	Attitude, filt. nav., timing
Trimble	TANS II upgrade	APEX, Pegasus, LEAP	6	Single	1	Navigation, timing
Trimble	TANS Quadrex	RADCAL	6	Single	1	Navigation, timing
Trimble	TANS Vector	Orbcomm, Crista, SPAS, REX-II, etc.	6	Single	4	Attitude, navigation, timing
Texas Instruments	SNR	DOD	1	Dual P(Y)	1	Navigation, timing
Texas Instruments	ASNR	none	6	Dual P(Y)	1	Navigation, timing

(This page intentionally left blank.)

REFERENCES

- [1] Axelrad, P., Brown, R., "GPS Navigation Algorithms," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [2] Battin, R.H., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, AIAA, Inc., New York, NY, 1987.
- [3] Blume, W., Jet Propulsion Laboratory, Personal Correspondence, October-December 1998.
- [4] Braasch, M.S., "Multipath Effects," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [5] Carter, D.W., *The Standalone - Notes on Porting to VAX and PC*, Draper Laboratory working paper, July 26, 1989.
- [6] Carter, D.W., *Port of the Semianalytic Satellite Theory to the Sun*, Draper Laboratory IOC EGC-90-149, May 25, 1990.
- [7] Carter, D.W., *GTDS Code Upgrade Proposal*, Draper Laboratory working paper, June 29, 1994.
- [8] Carter, S.S., *Precision Orbit Determination from GPS Receiver Navigation Solutions*, Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, CSDL-T-1260, June 1996.
- [9] Cefola, P.J., "Standalone Semianalytic Propagator Delivery," CSDL Letter SA-008-15Z-PJC to B.E. Baxter, The Aerospace Corporation, April 1984.
- [10] Cefola, P., Proulx, R., Metzinger, R., Cohen, M., Carter, D., "The RADARSAT Flight Dynamics System: An Extensible, Portable, Workstation-based Mission Support System," AIAA Paper 94-3726, presented at the AIAA/AAS Astrodynamics Conference, Scottsdale, Arizona, August 1-3, 1994.
- [11] Cefola, P.J., "Preliminary FORTRAN 90 Coding Standards", Draper Laboratory IOC, August 23, 1994.
- [12] Coffey, S., Jenkins, E., Neal, H., Reynolds, H., "Parallel Processing of Uncorrelated Observations into Satellite Orbits," AAS Paper 96-146, presented at the AAS/AIAA Spaceflight Mechanics Conference, Austin TX, February 12-15, 1996.
- [13] Danielson, D., Neta, B., Early, L., *Semianalytic Satellite Theory (SST): Mathematical Algorithms*, Naval Postgraduate School. Report Number NPS-MA-94-001, January 1994. [This overview document contains an extensive list of references to the original developer documentation.]
- [14] Draim, J., Castiel, D., "Optimization of the Borealis and Concordia Sub-Constellations of the Ellipso Mobile Communications System", 47th International Astronautical Congress, Beijing, China, October 1996.

- [15] Draim, J., Cefola, P., Proulx, R., Larsen, D., "Designing the ELLIPSO™ Satellites into the Elliptical Orbit Environment", 49th International Astronautical Congress, Melbourne, Australia, Sept.-Oct. 1998.
- [16] Early, L.W., "Portable Orbit Generator Using Semianalytical Satellite Theory," AIAA Paper 86-2164-CP, presented at the AIAA/AAS Astrodynamics Conference, Williamsburg, VA, August 1986.
- [17] Fennessey, R., Roberts, P., Knight, R., Van Volkinburg, B., "GPS as an Orbit Determination Subsystem," presented at the Flight Mechanics/Estimation Theory Symposium, Greenbelt, MD, May 16-18, 1995.
- [18] *Final Software Design Document for the Mission Support (MSN) Task of the Landsat Satellite Operation Center (LSOC)*, Draper Laboratory Report CSDL-R-2401, April 1992.
- [19] Fischer, J. D., *The Evolution of Highly Eccentric Orbits*, Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, CSDL-T-1310, June 1998.
- [20] Fischer, J., Cefola, P., Proulx, R., "The Evolution of Highly Eccentric Orbits: Decay Analysis Phase," AAS Paper 97-667, presented at the AIAA/AAS Astrodynamics Specialist Conference, Sun Valley, ID, August 4-7 1997.
- [21] Fonte, D.J., *Implementing the Jacchia-Roberts Atmosphere Density in the Standalone*, Draper Laboratory working paper, October 1991.
- [22] Fonte, D.J., *Implementing a 50X50 Gravity Field Model in an Orbit Determination System*, Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, CSDL-T-1169, June 1993.
- [23] Francisco, S.G., "GPS Operational Control Segment," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [24] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V., *PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, MA, 1994.
- [25] *Goddard Trajectory Determination System (GTDS) Mathematical Theory*, NASA's Operational GTDS Mathematical Specification, Revision 1, Edited by Computer Sciences Corporation and NASA Goddard Space Flight Center, Contract NAS 5-31500, Task 213, July 1989.
- [26] *Goddard Trajectory Determination System (GTDS) User's Guide*, NASA's Operational GTDS User's Guide, Revision 2, Update 8, Edited by Computer Sciences Corporation and NASA Goddard Space Flight Center, Contract NAS 5-31500, Task 52 617, September 1994.
- [27] Green, G.B., *Semi-Analytic Propagator Alterations*, Aerospace Corporation Interoffice Correspondence, August 1985.

- [28] Gropp, W., Lusk, E., Skjellum, A., *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, Cambridge, MA, 1995.
- [29] Herklotz, R.L., *Incorporation of Cross-Link Range Measurements in the Orbit Determination Process to Increase Satellite Constellation Autonomy*, Ph.D. Thesis submitted to the Department of Aeronautics and Astronautics, MIT, December 1987.
- [30] Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J., *Global Positioning System: Theory and Practice*, Springer-Verlag Wien, New York, NY, 1994.
- [31] Hopkins, R.G., *Description of the Astrodynamics Department MEANELT Stationkeeping and Orbit Propagation Capabilities*, Aerospace Corporation, Aerospace Technical Memorandum #86(9975)-23, January 1986.
- [32] Hopkins, R.G., *A User's Guide to Program MEANELT*, Aerospace Corporation, Aerospace Technical Memorandum #87(9975)-62, September 1987.
- [33] *Implementation of a Semianalytic Orbit Theory on the SPACECOM/D06 SEL Computer*, CSDL Proposal No. 5-503, August 1984. [Copy available from P. Cefola]
- [34] Johnson, N.L., *The Soviet Year in Space 1986*, Teledyne Brown Engineering, Colorado Springs, CO, 1987.
- [35] Kantsiper, B.L., *A Systematic Approach to Station-Keeping of Constellations of Satellites*, Doctor of Philosophy Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, CSDL-T-1298, March 1998.
- [36] Klobuchar, J.A., "Ionospheric Effects on GPS," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [37] Kuijper, D.C., "Precise Orbit Determination for the MOMS/Priroda Camera Experiment based on DGPS Carrier-Phase and Pseudo-Range Measurements," AAS Paper 98-197, presented at the AAS/AIAA Space Flight Mechanics Meeting, Monterey, CA, February 9-11, 1998.
- [38] Lichten, S., Estefan, J., "High-Precision Orbit Determination for High-Earth Elliptical Orbiters Using the Global Positioning System," AIAA Paper 90-2954, presented at the AIAA/AAS Astrodynamics Conference, Portland, OR, August 20-22, 1990.
- [39] Lichten, S., Wu, S., Young, L., Srinivasan, J., Haines, B., Coulson, P., "New Techniques for Orbit Determination of Geosynchronous, Geosynchronous-Transfer, and other High-Altitude Earth Orbiters," AAS Paper 97-676, presented at the AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, ID, August 4-7, 1997.
- [40] McClain, W., *A Recursively Formulated First-Order Semianalytic Artificial Satellite Theory based on the Generalized Method of Averaging*, Volume 1, Contract NAS 5-24300, Task Assignment 880, November 1977.

- [41] McLucas, J.L., *Space Commerce*, Harvard University Press, Cambridge, MA, 1991.
- [42] Melvin, P., "LEO Orbit Determination with Differential GPS," AAS Paper 98-195, presented at the AAS/AIAA Space Flight Mechanics Meeting, Monterey, CA, February 9-11, 1998.
- [43] Neal, H., Coffey, S., Knowles, S., "Maintaining the Space Object Catalog with Special Perturbations," AAS Paper 97-687, presented at the AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, ID, August 4-7, 1997.
- [44] Neelon, J.G. Jr., *The Transfer of Functionality from the Goddard Trajectory Determination System to the 'Standalone' Draper Semianalytic Satellite Theory Orbit Propagator*, Draper Laboratory working paper, February 28, 1997. [Copy available from P. Cefola]
- [45] Neelon, J., Cefola, P., Proulx, R., "Current Development of the Draper Semianalytic Satellite Theory Standalone Orbit Propagator Package," AAS Paper 97-731, presented at the AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, ID, August 4-7, 1997.
- [46] Parkinson, B.W., "Introduction and Heritage of NAVSTAR, the Global Positioning System," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [47] Parkinson, B.W., "GPS Error Analysis," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [48] Radarsat's CMS Library, VAX/VMS [radarsat:\[gtds.gtds lib\]](#), The Charles Stark Draper Laboratory, Inc. January 1997.
- [49] Shah, N.H., *Automated Station-Keeping for Satellite Constellations*, Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, CSDL-T-1288, June 1997.
- [50] Spilker, J.J., Jr., "GPS Signal Structure and Theoretical Performance," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [51] Spilker, J.J., Jr., "GPS Navigation Data," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [52] Spilker, J.J., Jr., "Tropospheric Effects on GPS," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [53] Spilker, J., Parkinson, B., "Overview of GPS Operation and Design," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [54] van Graas, F., Braasch, M., "Selective Availability", *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.

- [55] Wallace, S.T., *Parallel Orbit Propagation and the Analysis of Satellite Constellations*, Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, CSDL-T-1245, June 1995.
- [56] Wallace, S., Cefola, P., Proulx, R., "Parallel Orbit Propagation and the Analysis of Satellite Constellations," AAS Paper 95-428, presented at the AAS/AIAA Astrodynamics Specialist Conference, Halifax, Nova Scotia, Canada, August 14-17, 1995.
- [57] Zeis, E.G, *A Computerized Algebraic Utility for the Construction of Nonsingular Satellite Theories*, Master of Science Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, September 1978.
- [58] Zumberge, J., Bertiger, W., "Ephemeris and Clock Navigation Message Accuracy," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [59] Garrison, T., Ince, M., Pizzicaroli, J., Swan, P., "Iridium Constellation Dynamics - The Systems Engineering Trades," IAF Paper 95-U.2.04, presented at the 46th International Astronautical Congress, Oslo, Norway, October 2-6, 1995.
- [60] Matossian, M., "A Teledesic Space Infrastructure Overview," presented at the International Workshop on Mission Design & Implementation of Satellite Constellations, Toulouse, France, November 17-19, 1997.
- [61] Smith, D., "Operations Innovations for the 48-Satellite Globalstar Constellation," AIAA Paper 96-1051, presented at the AIAA/AAS Astrodynamics Specialist Conference, San Diego, CA, July 29-31, 1996.
- [62] Spilker, J.J., Jr., "Satellite Constellation and Geometric Dilution of Precision," *Global Positioning System: Theory and Applications Volume I*, AIAA, Inc., Washington, DC, 1996.
- [63] Bate, R., Mueller, D., White, J., *Fundamentals of Astrodynamics*, Dover Publications, Inc., New York, NY, 1971.
- [64] Bauer, F., Hartman, K., Lightsey, E., "Spaceborne GPS: Current Status and Future Visions," presented at the ION GPS 98 Conference, Nashville, TN, September 1998.
- [65] Payne, T.P., "First 'Confirmed' Natural Collision between Two Cataloged Satellites," presented at the Second European Conference on Space Debris, Esoc, Darmstadt, Germany, March 17-19, 1997.
- [66] Walker, R., Crowther, R., Swinerd, G., Marsh, V., Stokes, H., "Satellite Constellations and Their Long Term Impact on the Debris Environment in Low Earth Orbit," presented at the Second European Conference on Space Debris, Esoc, Darmstadt, Germany, March 17-19, 1997.
- [67] Prislun, R., Walker, D., Mercer, R., *TRACE66 Trajectory Analysis and Orbit Determination Program. Volume I. General Program Objectives, Description, and*

- Summary*, NTIS # AD-735 301/XAB, Aerospace Corporation, El Segundo, CA, 1971. [Copy available from NTIS, 5285 Port Royal Rd., Springfield, VA 22161]
- [68] Simpson, B., Rade, T., "Trends in Position Determination for Air Force Satellite Operations," AAS Paper 97-675, presented at the AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, ID, August 4-7, 1997.
- [69] *Supercomputing '92*(proceedings from the Supercomputing '92 conference, Minneapolis, MN, November 16-20, 1992), IEEE Computer Society Press, Inc., December 1992.
- [70] Shah, N., Proulx, R., Kantsiper, B., Cefola, P., Draim, J., "Automated Station-Keeping for Satellite Constellations," presented at the International Workshop on Mission Design & Implementation of Satellite Constellations, Toulouse, France, November 17-19, 1997.
- [71] Kantsiper, B., Weiss, S., "An Analytic Approach to Calculating Earth Coverage," AAS Paper 97-620, presented at the AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, ID, August 4-7, 1997.
- [72] Levine, D., *User's Guide to the PGAPack Parallel Genetic Algorithm Library*, Argonne National Laboratory, ANL-95/18, 1996.
- [73] Proulx, R., Smith, J., Draim, J., Cefola, P., "Ellipso Gear Array Coordinated Elliptical / Circular Constellations," presented at the AIAA/AAS Astrodynamics Specialist Conference, Boston, MA, August 10-12, 1998.
- [74] Smith, J., Proulx, R., Cefola, P., Draim, J., "Optimal Station Keeping Strategies via Parallel Genetic Algorithms," to be presented at the AAS/AIAA Spaceflight Mechanics Meeting, Breckenridge, CO, February 7-10, 1999.
- [75] Kuhn, J., Soskey, D., White, G., *Data Set Layouts for the Goddard Trajectory Determination System (GTDS)*, prepared by Computer Sciences Corporation for NASA Goddard Space Flight Center, Contract NAS 5-24300, Tasks 717 and 740, December 31, 1979.
- [76] *Research and Development Goddard Trajectory Determination System (R&D GTDS) User's Guide*, edited jointly by Computer Sciences Corporation and Systems Development and Analysis Branch, Goddard Space Flight Center, July 1978.
- [77] Potterveld, C., Puig-Suari, J., "Optimization of GPS Usage for Low-Power Satellites," AIAA Paper 96-3653, presented at the AIAA/AAS Astrodynamics Conference, San Diego, CA, July 29-31, 1996.