An Algorithm for Reducing Atmospheric Density Model Errors Using Satellite Observation Data in Real-Time

by

Sarah Elizabeth Bergstrom

Bachelor of Science, Swarthmore College, 2000

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics and Astronautics
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2002

Author ....

Department of Aeronautics and Astronautics
May 10, 2002

Certified by ....

Dr. Paul J. Cefola
Technical Staff, the MIT Lincoln Laboratory
Lecturer, Department of Aeronautics and Astronautics
Thesis Supervisor

Certified by ....

Dr. Ron J. Proulx
Principal Member of the Technical Staff,
the Charles Stark Draper Laboratory
Thesis Supervisor

Accepted by ....

Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

An Algorithm for Reducing Atmospheric Density Model Errors Using Satellite Observation Data in Real-Time

by

Sarah Elizabeth Bergstrom

Submitted to the Department of Aeronautics and Astronautics
on May 24, 2002, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Atmospheric density mismodeling is a large source of errors in satellite orbit determination and prediction in the 200–600 kilometer range. Algorithms for correcting or "calibrating" an existing atmospheric density model to improve accuracy have been seen as a major way to reduce these errors. This thesis examines one particular algorithm, which does not require launching special "calibration satellites" or new sensor platforms. It relies solely on the large quantity of observations of existing satellites, which are already being made for space catalog maintenance. By processing these satellite observations in near real-time, a linear correction factor can be determined and forecasted into the near future. As a side benefit, improved estimates of the ballistic coefficients of some satellites are also produced. Also, statistics concerning the accuracy of the underlying density model can also be extracted from the correction. This algorithm had previously been implemented and the implementation had been partially validated using simulated data. This thesis describes the completion of the validation process using simulated data and the beginning of the real data validation process. It is also intended to serve as a manual for using and modifying the implementation of the algorithm.

Thesis Supervisor: Dr. Paul J. Cefola
Title: Technical Staff, the MIT Lincoln Laboratory and Lecturer, Department of Aeronautics and Astronautics

Thesis Supervisor: Dr. Ron J. Proulx
Title: Principal Member of the Technical Staff, the Charles Stark Draper Laboratory

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

First, and foremost, I want to thank my thesis advisor at the MIT Lincoln Laboratory (LL), Dr. Paul Cefola.

I'd also like to thank Dr. Ronald Proulx at the Charles Stark Draper Laboratory (CSDL), who has been a part of this research project since its commencement. He has offered continuing encouragement and technical assistance throughout this project, including enormous help with the seemingly-neverending adventures in file transfer between CSDL and LL.

Second, I want to thank George Granholm, Jack Fischer, Prof. Andrey Nazarenko, and Dr. Vasiliy Yurasov for their fine work, upon which all of my endeavours here have depended. Their papers and theses have been a continual source of inspiration and insight into the intricacies of atmospheric density modelling and satellite orbit determination. Thanks especially to George for finding time in his busy schedule with the U.S. Air Force to provide some initial guidance on working with the software.

Thanks go to Lt. Col. David Vallado (USAF) for providing real observation data.

At LL, I'd like to extend thanks to Group 98 Leader Dr. Sid Sridharan for supporting this project. I'd also like to thank Jim Apicella and Sherry Robarge for computer support, Zach Folcik for assistance with the gtds_granholm makefiles and various other software, and Gladys Chaput, Kathy Fellows, Bonnie Tuohy, and Nancy Alusow for administrative and travel assistance.

At CSDL, I'd like to thank Darryl Sargent for his help in coordinating the joint work between the Labs. I'd also like to thank Linda Leonard for computer support in the EDCF computing facility.

I'd like to thank Dr. Richard Battin at MIT for cultivating my interest in orbital mechanics and pointing me at this research project. Thanks also go to my academic advisor, Dr. J. P. Clarke, for giving me the freedom to pursue what interested me most, and for assisting with travel arrangements, along with assistance from Jennie Leith and Marie Stuppard.

On a personal level, I'd like to thank my family for their encouragement of all

my academic endeavours. To my best friend Jen – you've always been ahead of me, inspiring me to keep going. To my beloved Dave, thanks for being there for me. To my roommate (and LaTeX consultant) Chaos – thanks for putting up with the dishes I didn't wash during the last few weeks of every semester, and for being a great friend.

Contents

1 Introduction **17**

1.1 Atmospheric Density Modeling 17

1.1.1 A Brief History 17

1.1.2 An Overview of the Entire Atmosphere 18

1.1.3 Thermosphere Modeling Details 21

1.1.4 Model Errors 23

1.1.5 A New Empiricism 23

1.2 Prior Work on this Algorithm 24

1.2.1 Nazarenko and Yurasov's Original Development 24

1.2.2 George Granholm's Work 24

1.3 Outline of this Thesis 25

2 Mathematical Details **29**

2.1 Basic Concepts 29

2.2 Linear Correction Factors 30

2.3 Ballistic Factors to Correction Coefficients 31

2.3.1 Fitting Ballistic Factors to Data 31

2.3.2 Deriving Corrections from Ballistic Factors 32

2.3.3 Weighted Least Squares 33

2.3.4 Solution Boundaries 36

2.4 Forecasting Linear Correction Factors 37

2.5 Ballistic Factor Estimation 38

2.5.1 BFE basic process 38

2.5.2 Derivation for One Standard Satellite 39
2.5.3 Multiple Standard Satellites 43

3 Implementation Overview 45
3.1 Computer Code 45
3.1.1 GTDS 46
3.1.2 AtmoCal 47
3.2 Data Flow 48

4 Simulated Data Validation 51
4.1 Simulated Data Generation 51
4.1.1 Preparation 52
4.1.2 Truth Orbit Generation 53
4.2 Reproduction of Prior Results using Simulated Data 55
4.2.1 Data Flow Verification 55
4.2.2 Differences between Truth and Fit Models 57
4.2.3 Simulating "Noisy" Observations 61

5 New Validation Results with Simulated Data 65
5.1 Ballistic Factor Distortion 66
5.2 Results 70
5.2.1 Data Flow 70
5.2.2 Effects on Atmospheric Density Correction 70
5.2.3 Convergence 73
5.2.4 Update Cycle Length 73
5.2.5 Height Dependence of Errors 73
5.2.6 Global Distortion/Correction Effects 78
5.2.7 Omitting Recalculation of $\hat{k}_{ij}$ Values 80

6 Real Data Validation **83**
6.1 Data Preparation 83
6.1.1 Conversion of B3 Observations 83
6.1.2 Station Data 85
6.1.3 Observation Scheduling 85
6.1.4 Current Status 85

7 Conclusions and Future Work **91**
7.1 Conclusions 91
7.2 Future Work 91
7.2.1 Further Tests 92
7.2.2 GTDS Bugs 93
7.2.3 New GTDS Features 93
7.2.4 GTDS Integration 94
7.2.5 AtmoCal Refinements 94
7.2.6 Major AtmoCal Additions/Changes 95

A Key to Symbols, Abbreviations, Etc. **99**
A.1 Text Conventions 99
A.2 Expressions and Abbreviations 100

B Implementation Miscellanea **105**
B.1 CVS and Revision Control 105
B.2 File Names and Locations 106
B.3 Environment Variables 109

C GTDS **111**
C.1 GTDS Changes 111
C.2 GTDS Data Files 113
C.3 Additions to the Metzinger Test Cases 116

C.4 List of Known Bugs/Issues 129

D Annotated Code 131

D.1 TLE2osc.pl 132

D.2 genobs.pl 141

D.3 distort_bfs.m 151

D.4 estbfs.pl 154

D.5 calcvars.pl 174

D.6 Drivers for *estbfs.pl* and *calcvars.pl* 183

D.7 calc_b.m 191

D.8 Dates.pm 198

D.9 b3conv.pl 202

D.10 Other Utilities 208

D.11 Graphing Utilities 212

E File Utilities and Formats 219

E.1 B3 to OBSCARD Conversion Utility 219

E.2 Building New GTDS Binary Files 222

E.3 Detailed File Formats 222

E.4 GTDS Input Decks 225

F LaTeX Notes 231

Bibliography 233

About the Author 241

List of Figures

1-1 Atmospheric Regions 19

1-2 Atmospheric Composition at Low Exospheric Temperature 20

1-3 Atmospheric Composition at High Exospheric Temperature 21

1-4 Summary of George Granholm's Work 25

2-1 Flowchart for Overall AtmoCal Operation 31

2-2 Visual Representation of the Fit Window for Satellite j 32

2-3 Ratio of True Density to Jacchia 1971 Model Density 34

2-4 Block Diagram for Correction Factor Forecasting 38

2-5 Flowchart for Ballistic Factor Updating Cycle 44

3-1 AtmoCal Operation Flowchart Including File Names 48

4-1 Flowchart for Simulated Observation Creation 54

4-2 B-values with No Noise, No Mismodeling 56

4-3 Linear Atmospheric Density Correction Factors with No Noise, No Mismodelling 56

4-4 Observed Daily Mean and Schatten Ap Values for 12/15/99–2/15/00 58

4-5 Daily and Schatten $F_{10.7}$ Values for 12/15/99–2/15/00 58

4-6 Observed $F_{10.7}$ and a_P Values During Fit Window 59

4-7 B-values with No Noise, Schatten Mismodeling 60

4-8 Linear Atmospheric Density Correction Factors with No Noise, Schatten Mismodeling . . . 60
4-9 B-values with Observation Noise, No Mismodelling . . . 63
4-10 Linear Atmospheric Density Correction Factors with Observation Noise, No Mismodeling . . . 63
4-11 B-values with Observation Noise, Schatten Mismodelling . . . 64
4-12 Linear Atmospheric Density Correction Factors with Observation Noise, Schatten Mismodeling . . . 64

5-1 Distribution of (undistorted) Ballistic Factors for All Satellites . . . 67
5-2 Distribution of Ballistic Factors for the Standard Satellites used in BFE validation . . . 67
5-3 Distribution of Perigee Heights for All Satellites . . . 68
5-4 Distribution of Perigee Heights for the Standard Satellites used in BFE validation . . . 68
5-5 Ballistic Factor Distortion Ratios . . . 69
5-6 BF Percent Errors after Iteration, with No Noise, No Mismodeling, No Initial Distortion . . . 71
5-7 BF Errors Sorted by Initial BF, with No Noise, No Mismodelling, No Initial Distortion . . . 72
5-8 Comparison of B-values with and without Initial Ballistic Factor Distortion . . . 72
5-9 Convergence for five-day case . . . 74
5-10 Convergence for ten-day case . . . 74
5-11 Percent Errors for all Satellites before BFE iteration . . . 75
5-12 Percent Errors for all Satellites after 5 BFE iterations . . . 75
5-13 Average Absolute Percent Deviation as a Function of Iteration Period (example 1) . . . 76

5-14 Average Absolute Percent Deviation as a Function of Iteration Period (example 2) 76
5-15 Remaining Errors After Iteration, Sorted by Perigee Height 77
5-16 BFE iteration with No Global Distortion 78
5-17 Percent Errors for all Satellites before BFE iteration, with no Global Distortion 79
5-18 Percent Errors for all Satellites after 5 BFE iterations, with no Global Distortion 79

6-1 Flowchart for Real Observation Preparation 84

B-1 File Structure for Large Data Files 107
B-2 File Structure for AtmoCal and Small Data Files 107
B-3 File Structure for gtds_granholm 108

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

4.1 Statistics for B-values with No Noise, No Mismodeling 55

4.2 Statistics for B-values with No Noise, Schatten Mismodeling 57

4.3 Statistics for B-values with Observation Noise, No Mismodeling . . . 62

4.4 Statistics for B-values with Observation Noise, Schatten Mismodeling 62

5.1 Statistics for Ballistic Factor "Improvements" with No Noise, No Mismodeling, No Initial Distortion 71

B.1 Common CVS commands 106

B.2 Environment Variable List 109

C.1 GTDS Code Alteration List 112

C.2 List of GTDS Data Files 113

D.1 TLE2osc.pl Fact Sheet 132

D.2 genobs.pl Fact Sheet 141

D.3 distort_bfs.m Fact Sheet 151

D.4 estbfs.pl Fact Sheet 154

D.5 calcvars.pl Fact Sheet 174

D.6 runestbfs.pl Fact Sheet 183

D.7 runcalcvars.pl Fact Sheet 183

D.8 bfe_iter.pl Fact Sheet 184

D.9 calc_b.m Fact Sheet 191

D.10 Dates.pm Fact Sheet 198
D.11 b3conv.pl Fact Sheet 202
D.12 dateconvert.pl Fact Sheet 208
D.13 get_peri.pl Fact Sheet 208
D.14 read_b.m Fact Sheet 212
D.15 analyze_atmcal.m Fact Sheet 212
D.16 readinitinfo.m Fact Sheet 212

E.1 List of NORADPP Files Modified 221
E.2 OBSCARD Format 223
E.3 Format of *initinfo.txt* 224
E.4 Format of *jac_densvars.txt* 224
E.5 Format of *ballfcts.txt* 224
E.6 Format of Station Card 0 227
E.7 Format of Station Card 1 228
E.8 Format of ATMCAL card 229

Chapter 1

Introduction

1.1 Atmospheric Density Modeling

1.1.1 A Brief History

When Sputnik was launched in 1957 [7], very little was known about the nature of the atmosphere above 100 kilometers. Data from the first high-altitude sounding rockets and satellites in the late 1950's and early 1960's provided enough information for researchers to create elementary models, based mainly on the ideal gas equation and the hydrostatic equation [42]. Most notable among these early models was that of Luigi Jacchia, based in part on earlier models by Marcel Nicolet [23]. In 1977, Alan Hedin published the first of a series of models based on (and named after) Mass Spectrometer and Incoherent Scatter (MSIS) data[19]. The MSIS models are still under active development, with the Naval Research Laboratory's NRLMSISE-2000 being the most recent version[44].

A multitude of other models have also been created since the 1970's, but none, as of yet[1], has demonstrated any significant improvement over the Jacchia-Roberts 1971

---

[1] The cited comparison was performed before MSISE-90 and NRLMSISE-2000 were available. These and other recent models may offer some improvements, although the same modeling difficulties listed in Section 1.1.4 apply.

(JR-71) model [34]. All models seem to show a 10-15% error in quiet and normal conditions, with errors potentially reaching 30% in highly perturbed conditions.

Increasing the accuracy of atmospheric density models would allow satellite orbits to be determined and predicted into the future with higher precision and for longer time periods. This in turn allows for more efficient planning of maneuvers, including routine stationkeeping as well as collision-avoidance, de-orbiting, or maneuvers to transition between two orbits. Collision avoidance is especially important now that the International Space Station (ISS) orbits in the 300-400 kilometer region[58].

1.1.2 An Overview of the Entire Atmosphere

Most people are only familiar with the lowest region of the atmosphere, called the *troposphere*, which extends for the first 11 kilometers above the Earth's surface. All weather takes place in this region, and it behaves according to simple, intuitive principles. As one ascends through the troposphere, the air gets colder, since the main source of heat in this region is the surface of the Earth, and thinner, due to decreased gravitational forces, but remains relatively similar in composition. (All of the regions of the Earth's atmosphere are summarized in Figures 1-1, 1-2 and 1-3.)

Beyond the troposphere, the temperature begins to rise again, due to the effects of solar radiation on atmospheric oxygen. Some components of ultraviolet solar radiation split molecular oxygen (O_2) into atomic oxygen (O) and ozone (O_3), while others are absorbed by the ozone and heat both the ozone molecules and the surrounding air. This region, formally called the *stratosphere*, familiar to most people as "the ozone layer" extends upwards to approximately 50 kilometers, where the ozone heating effect no longer dominates, and the temperature begins to drop once more. This region of decreasing temperature, known as the mesosphere, extends to approximately 90 kilometers, whereupon solar radiation heating again begins to dominate. Everything above this final temperature inflection point, known as the *mesopause*, is referred to as the *thermosphere*, because of the extremely high temperatures[2] reached in the region. The various regions of the atmosphere and average temperature are shown in Figure 1-1[3].

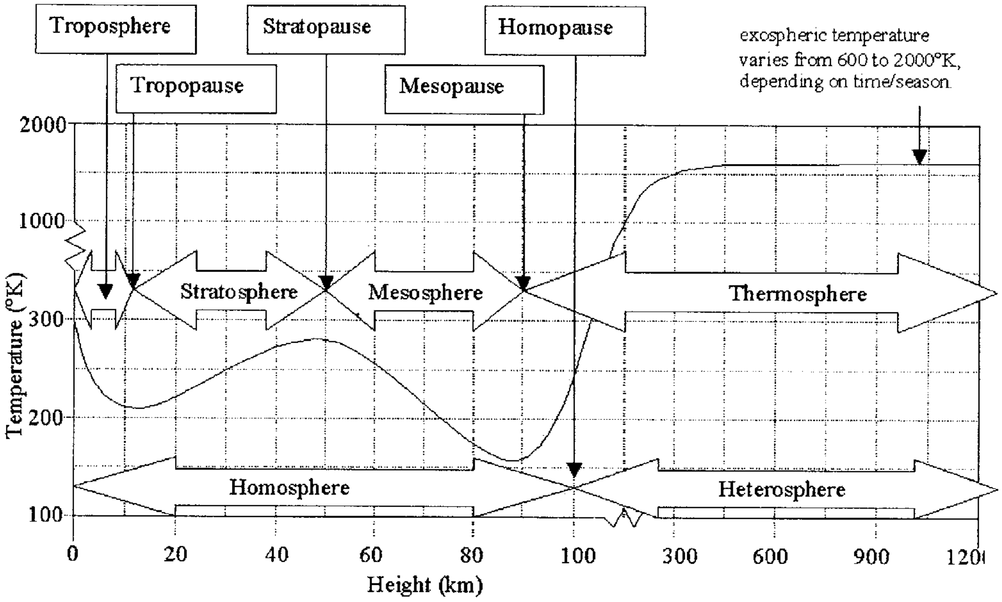


Figure 1-1: Atmospheric Regions

Two other regional divisions are often found in atmospheric modeling literature: the *ionosphere*, which refers to the region of the atmosphere containing ionized particles (roughly equivalent to the thermosphere), and the *exosphere*, which is the entire atmosphere above the *exobase*, which is the point at which individual gas atoms may be thought of as being in individual orbits around the earth. The *exospheric temperature* is the temperature that is asymptotically approached in the exosphere as the height increases to infinity, as seen in Figure 1-1.

Another important division of the atmosphere occurs around 100 kilometers, where the composition of the atmosphere begins to change. Below this point, known

---

[2] Note that a strict, scientific definition of "temperature", based on the kinetic energy of individual gas molecules, must be used in this region, since gas densities are so low that a thermometer would be useless.

[3] Figure 1-1 is based closely on figures in [26] and [42].

as the *homopause*, the atmosphere contains the familiar mix of 78 percent nitrogen (N_2), 21 percent oxygen (O_2), 1 percent argon (Ar), with trace amounts of water vapor and other compounds. Around the homopause, the air becomes thin enough that particle collisions become rare. This has two effects: first, atomic oxygen becomes a major component, since the atoms rarely collide to reform O_2, and mixing no longer keeps the proportions of various components steady. Instead, the particles of each component gas react individually to the Earth's gravitational field, and the components stratify by molecular weight. Approximate individual concentrations[4] in the 200-600 kilometer range are shown below for lower and upper extreme exospheric temperatures (500 and 1900 $^\circ K$)[5].

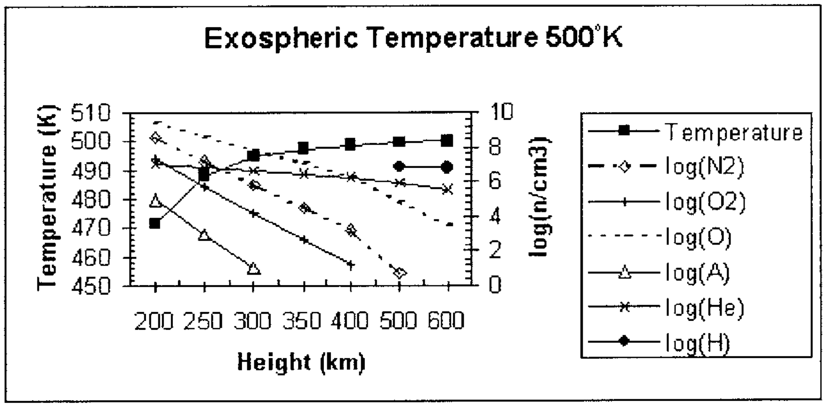


Figure 1-2: Atmospheric Composition at Low Exospheric Temperature

Normal daytime temperatures are in the 1500–2000 $^\circ K$range, and nighttime temperatures during quiet periods fall in the 500-700 $^\circ K$range. Thus, values close to or at the extremes shown in Figures 1-2 and 1-3 tend to be seen on a daily basis, with the density at any particular altitude in the thermosphere fluctuating by several hundred percent.

---

[4] Hydrogen is not included in the JR-71 model below 500 kilometers.

[5] Figure 1-2 uses data from pages 78–79 and Figure 1-3 uses data from pages 106–107 of Jacchia's 1971 model [25].

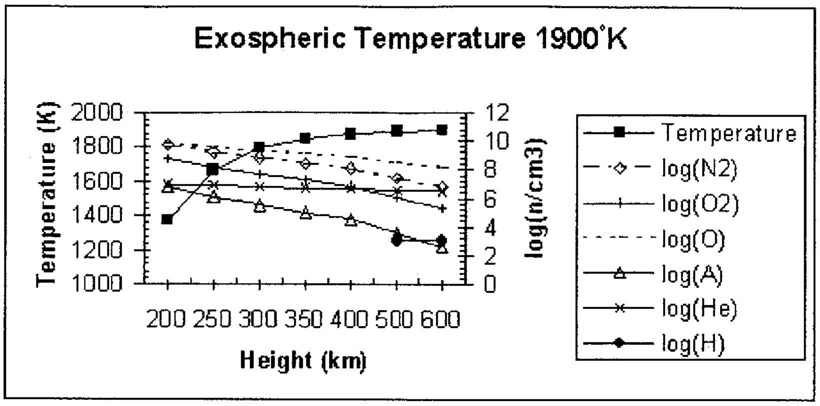


Figure 1-3: Atmospheric Composition at High Exospheric Temperature

1.1.3 Thermosphere Modeling Details

Most modern thermospheric density models include several major factors:

Lower Boundary Conditions: Thermospheric models must have a starting point, and most start at altitudes between 90 and 120 kilometers, setting either constant or seasonally-dependent boundary conditions[25, 17]. (The E (for Extended) in the MSISE-series models denotes that a model for the lower atmosphere has been linked to these boundary conditions from the other side, but we are only concerned here with the thermospheric model.)

Diurnal Variation: This is simply the exospheric temperature difference between day and night. The maximum density increase due to the sun's heating effect occurs around 2 pm local solar time, at a latitude known as the sub-solar point, and the minimum around 3 am. The strength of this effect and the location of the sub-solar point varies seasonally, and is well-understood[25].

Annual and Semi-annual Variations: There are several seasonal atmospheric composition changes, including the winter helium bulge and some low-altitude hydrogen variations. The hydrogen variations are sometimes modeled as temperature variations for simplicity and compatibility with boundary conditions.

These phenomena are well measured, although the accuracy to which they are modeled varies, especially at lower altitudes[25].

Solar Activity Variations: Extreme ultraviolet (EUV) radiation from the sun is the primary source of heat in the thermosphere, and the amount of radiation produced by the sun varies greatly over the 11-year solar cycle and with sunspot activity. Since no appreciable amount of the EUV wavelengths which cause heating reach the surface of the earth, we rely on measurements of the solar radio flux at a wavelength of 10.7 cm (which is a frequency of 2800 MHz). This radio flux is known as the $\mathrm{F}_{10.7}$ index, and is usually tabulated on a daily basis, along with the average flux ($\overline{\mathrm{F}}_{10.7}$) seen over the preceding 90 or 180 days. The $\mathrm{F}_{10.7}$ index is used to determine short-term variations due to sunspots and other temporary solar phenomenon, while $\overline{\mathrm{F}}_{10.7}$ gives a measure of the average flux seen during that portion of the 11-year solar cycle. Past values of $\overline{\mathrm{F}}_{10.7}$ from the appropriate time in the solar cycle can be used to create lists of predicted $\overline{\mathrm{F}}_{10.7}$ values. Ken Schatten designed one such prediction method, details of which can be found on his web site[49]. Measurements of the actual EUV radiation taken from various upper-atmospheric experiments in the 1960's and 1970's were used to determine that the $\mathrm{F}_{10.7}$ and $\overline{\mathrm{F}}_{10.7}$ indices are more accurate than the CaII K plage index or visible sunspot observations.[33]

Geomagnetic Activity Variations: Geomagnetic storms, caused by coronal mass ejections and other solar eruptions create strong short-term density fluctuations[57]. The planetary geomagnetic index a_P (or the closesly related index K_p) is used as the indicator for these effects. The a_P index is usually tabulated as a smoothed daily average, and the K_p index is not smoothed (and is tablulated every 3 hours), and both are useful in density calculation[36].

1.1.4 Model Errors

The solar and geomagnetic activity variations discussed in the preceding list are the effects that give rise to the greatest errors in atmospheric density determination and prediction. First, $F_{10.7}$, $\overline{F}_{10.7}$, a_P, and K_p are not perfect indicators of the underlying effects. Attempts to replace both of them are underway, but no replacements have yet been widely adopted[44, 52]. Second, none of the methods for predicting future values of these indices are able to capture the random nature of unexpected sunspots or coronal mass ejections.

1.1.5 A New Empiricism

Observational data has always been at the core of atmospheric density models, but it was not until the past decade, when sufficient computer speed and storage capabilities became available, that the idea of improving models by incorporating real-time data from large numbers of satellites became popular. The hope is that the so-called 15% (one-sigma) barrier can be broken consistently by using this algorithm or another"calibration method"[36]. This project is one of several in this field – the High Accuracy Satellite Drag Model (HASDM) is another, and Frank Marcos also has a project in this area[51, 35]. One major alternative to the "calibration" method is the use of satellites with direct atmospheric drag and/or composition observation capabilities, instead of relying solely on ground-based data. Current projects include the CHAMP and GRACE satellites, which are both near-spherical and carry high-accuracy accelerometers, and the DMSP satellite, which measures atmospheric density and composition, and the TIMED satellite, which will measure EUV radiation directly[36]. These projects, however, are costly, while the "calibration" methods require only a small amount of processor time, using data that is already being collected for space catalog maintenance.

1.2 Prior Work on this Algorithm

1.2.1 Nazarenko and Yurasov's Original Development

This algorithm was originally developed and tested by Andrey Nazarenko and Vasiliy Yurasov in the early and mid-1990's. In 1997, the Charles Stark Draper Laboratory (CDSL) commissioned a report detailing the latest implementation of Nazarenko and Yurasov's work[41]. This report for CSDL provided both theoretical and empirical support for the algorithm, which appeared both promising and portable.

1.2.2 George Granholm's Work

The algorithm was re-implemented from scratch beginning in 1999 by George Granholm at CSDL[13]. This new implementation used the Goddard Trajectory Determination System (GTDS)[14] to calculate satellite trajectories (and atmospheric densities), on a SGI-UNIX platform. JR-71 was chosen as the underlying atmospheric density model since it was already fully implemented in GTDS, is considered to be one of the most accurate models available, and is in common usage. George implemented the atmospheric density correction algorithm by creating a series of Perl scripts that automatically run GTDS and several MATLAB routines (also written by Granholm). To verify that his implementation was functioning properly, he created simulated testing data, and proved that the main components of the algorithm were operating properly. The flowchart in Figure 1-4 shows the sections that Granholm wrote, completed and/or validated. The dotted lines denote sections that Granholm began, which were not completed due to time constraints.

In March 2001, Dr. Paul Cefola, who had been one of the major investigators of the atmospheric density correction project at CSDL, retired from CSDL and assumed a position at the MIT Lincoln Laboratory (LL). Subsequently, in May 2001, LL technical staff met with the CSDL technical and project office staff, and an agreement was made that the project should become a joint CSDL-LL venture.

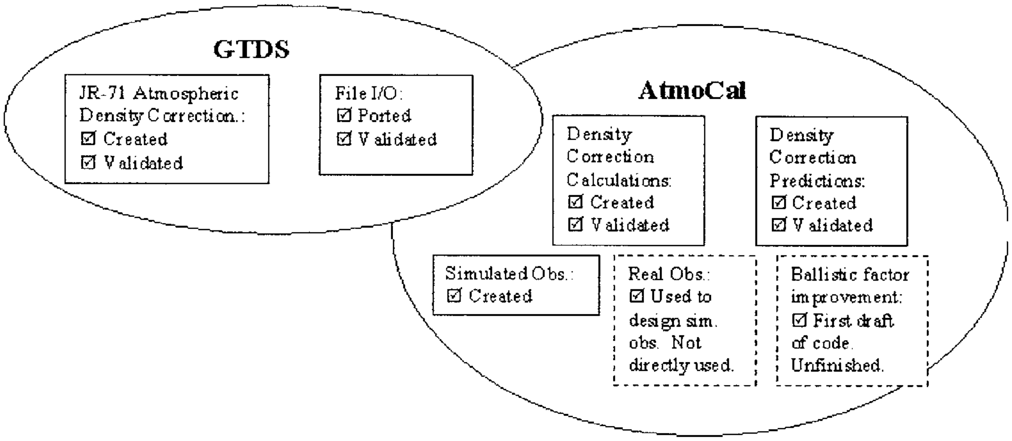


Figure 1-4: Summary of George Granholm's Work

During the summer of 2001, the code was moved by the author and Ron Proulx to the Pisces SGI-UNIX machine at LL, and was given the name AtmoCal.

1.3 Outline of this Thesis

The following outline is intended to serve as an index for finding particular information in the remainder of this thesis.

Chapter 1: Introduction details the motivation and the history of this project.

Chapter 2: Mathematical Details includes the derivation of all of the equations used in the atmospheric density correction process. The first part (Sections 2.2–2.3.4) derives the main atmospheric density correction algorithm, the second (Section 2.4) describes the current techniques used to predict the correction factors into the future, and the third (Sections 2.5.1–2.5.3) derives the ballistic factor improvement algorithm.

Chapter 3: Implementation Overview gives a brief description of the current software implementation of the algorithm detailed in Chapter 2. Details of the computer code are left to the appendices.

Chapter 4: Simulated Data Validation describes and gives results from the sections of George Granholm's simulated data validation process which were recreated on the Pisces machine at LL. It also includes an overview of how the simulated data was generated.

Chapter 5: New Validation Results with Simulated Data shows the results of validating the ballistic factor updating algorithm with simulated data.

Chapter 6: Real Data Validation gives an overview of the process of running AtmoCal on real data.

Chapter 7: Conclusions and Future Work summarizes the current state and the future goals of this project.

Appendix A: Key to Symbols, Abbreviations, Etc. lists all of the mathematical symbols, abbreviations, acronyms, and text conventions used in preparing this thesis.

Appendix B: Implementation Miscellanea describes the use of the Concurrent Version System (CVS) for configuration management and gives information on file locations and shortcuts needed for running AtmoCal.

Appendix C: GTDS describes Granholm's alterations to GTDS and the validation process, which was repeated at LL. This appendix also includes a list of the GTDS binary and text data files used while running AtmoCal.

Appendix D: Annotated Code contains the full text of each of the AtmoCal routines. It also includes tables of user options for AtmoCal routines.

Appendix E: File Utilities and Formats describes the utility for converting NORAD B3 observations to OBSCARD format and lists the formats of all of the AtmoCal I/O files.

Appendix F: LaTeX Notes includes information on the creation of this thesis.

Bibliography lists all of the works consulted in preparing this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Mathematical Details

2.1 Basic Concepts

A wealth of data is constantly being collected on every object in orbit around the Earth. This data is used to maintain the U.S. space catalog, determine desired orbit corrections, and predict collision risks. The goal of this and other atmospheric density correction methods is to provide a "correction factor" of some sort, which improves an existing atmospheric density model. The correction factor could then be used by anyone using the same density model, in order to improve satellite orbit determination and prediction.

Thus, we need to find a simple and robust way to extract information about the errors of the atmospheric density model from observations of multiple satellites. Once the errors can be quantified, a correction factor that removes or reduces them can then be determined. The algorithm detailed in this thesis, as it is currently operational, provides a linear correction factor for the JR-71 model, using data from over 300 satellites in low earth orbit (LEO).

2.2 Linear Correction Factors

The algorithm operates by determining a linear density correction for every three-hour span where sufficient data[1] is present, and then predicting those correction factors, in three-hour spans, into the near future. The time period of three hours was chosen because it was long enough to accumulate sufficent data under normal conditions. If more data becomes available, this period could be shortened. A linear model was chosen by Nazarenko and Yurasov because it would not try to extract too much information from the data, but would model the observed errors reasonably well. Thus, we want to determine some linear coefficients b_{1j} and b_{2j} that describe the best correction factor in a given three-hour interval. Designating satellite height by h, the fundamental linear correction equation for the three-hour span t_j is:

$$correction\,(h, t_j) = b_{1j} + b_{2j}\left(\frac{h - 400}{200}\right) \tag{2.1}$$

Aside from the observational data (in range/azimuth/elevation format) , the algorithm requires only tabulated values of the ballistic factor of each satellite in the catalog. The *a priori* values for the ballistic factors should be the best ones available when correction begins. (The improvement of ballistic factor estimates is described in Section 2.5. Note that the definition of ballistic factor[2] used in this paper is:

$$k = \frac{C_D A_x}{2m} \tag{2.2}$$

For each three-hour period, then, we want to calculate the correction factor that best approximates the actual difference $\delta\rho$ between the model density ρ_m and the true density ρ.

---

[1] To be precise, 35 data points, in the form of observed ballistic factors, are required for each 3-hour span. The description of how those ballistic factors are created and processed is detailed later in this chapter.

[2] The AtmoCal software performs conversions between the tabulated values of k and A_x in meters and A_x in kilometers and mass m when required for the use of GTDS, using a standard value of 2.2 for C_D.

$$\rho = \rho_m + \delta\rho = \rho_m \left(1 + \frac{\delta\rho}{\rho_m}\right) \tag{2.3}$$

The entire operation of the algorithm can be summarized in the flowchart in Figure 2-1:

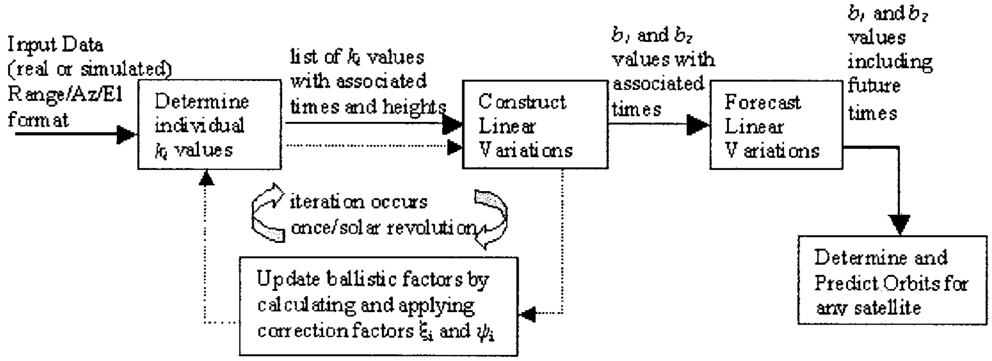


Figure 2-1: Flowchart for Overall AtmoCal Operation

2.3 Ballistic Factors to Correction Coefficients

2.3.1 Fitting Ballistic Factors to Data

Ballistic factors are determined by fitting orbits to three-day blocks[3] of the observational data. GTDS uses the tabulated ballistic factor as an initial guess, and iterates to find the state vector and the observed ballistic factor. This observed ballistic factor $\hat{k}_{ij}$ is attributed to time j, at the middle of the three-day span, and the fit window is moved forward three hours[4]. The process is repeated until the end of the fit period is reached. This does, however, mean that there is a 1.5 day gap between the start of

---

[3] In highly perturbed conditions, or when data is sparse, the three-day window can easily be lengthened to five days or more.

[4] This is a batch-fit method, and was chosen both for consistency with Nazarenko and Yurasov's implementation, and for software simplicity. Granholm discusses the possibility of using recursive methods in Chapter 2 of his thesis, but this capability has not yet been incorporated into AtmoCal.

data collection and the first linear correction factor, as well as a 1.5 day gap between the last observation-based (as opposed to prediction-based) correction factor and the end of the data. Any fit runs that do not converge or have a high convergence error at the end of the GTDS run are thrown out, since the remaining observations should be sufficient.

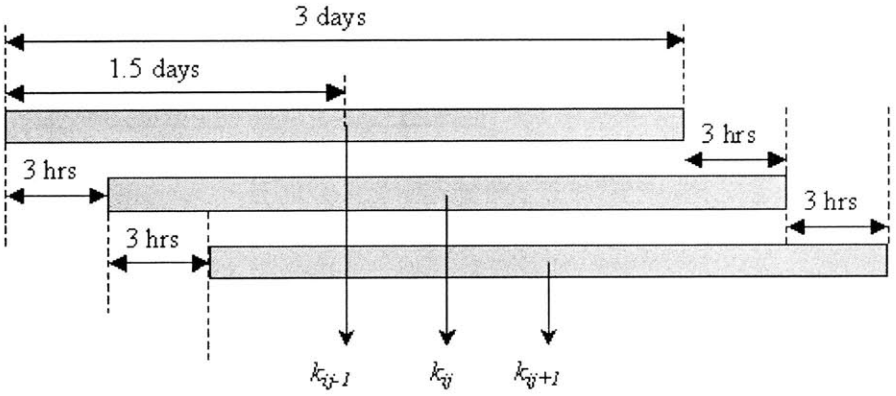


Figure 2-2: Visual Representation of the Fit Window for Satellite j

2.3.2 Deriving Corrections from Ballistic Factors

The derivation of an expression for $\frac{\delta\rho}{\rho_m}$ begins with the equation for the period rate[5] of a satellite's orbit. Note that $f(\underline{x})$ is some unspecified function (connected to the equations of motion) of the state vector $\underline{x}$ of orbital elements[6].

$$\dot{T}_i = k_i \cdot \rho(h_{ij}, t_j) \cdot f(\underline{x}) \tag{2.4}$$

This equation can then be rewritten in terms of the observed ballistic factor and the model density, with the assumption that the observed orbital elements closely match the actual ones.

---

[5] Any orbital element that is directly related to the energy of the orbit may be substituted for period rate, yielding a similar derivation.

[6] Any set of orbital elements which fully describe the motion of the satellite is acceptable.

$$\hat{\dot{T}}_i = \hat{k}_{ij} \cdot \rho_m(h_{ij}, t_j) \cdot f(\underline{x}) \tag{2.5}$$

By dividing Equation 2.4 by Equation 2.5, and assuming that the observed period rate is a good approximation of the actual period rate (i.e. $\dot{T}_i \approx \hat{\dot{T}}_i$) an expression for $\frac{\delta\rho}{\rho_m}$ in terms of the observed and actual ballistic factors is obtained.

$$\begin{aligned}
\frac{\dot{T}_i}{\hat{\dot{T}}_i} &= \frac{k_i \cdot \rho(h_{ij}, t_j)}{\hat{k}_{ij} \cdot \rho_m(h_{ij}, t_j)} \approx 1 \\
\frac{\hat{k}_{ij}}{k_i} &\approx \frac{\rho(h_{ij}, t_j)}{\rho_m(h_{ij}, t_j)} \\
\frac{\hat{k}_{ij}}{k_i} - 1 &\approx \frac{\delta\rho(h_{ij}, t_j)}{\rho_m(h_{ij}, t_j)} \qquad (2.6)
\end{aligned}$$

2.3.3 Weighted Least Squares

Now, we have a long list of density corrections expressed as ballistic factor ratios, each associated with a time and height. To convert these into single three-hour linear corrections requires some sort of fitting algorithm. Jaeck-Berger and Barlier showed that the errors in Jacchia's 1971 (J71) model are approximately zero-mean and Gaussian. Figure 2-3, reprinted from their work[27], demonstrates this adequately. The dotted line is the normal J71 model, while the solid line is a modified version of J71 described in Jaeck-Berger and Barlier's paper. Both have an approximately normal distribution, with a mean of one, implying that errors of the form $\delta\rho = \frac{\rho_m}{\rho}$ are also normally-distributed, with a mean of zero. Since the JR-71 model differs from J71 only in the mathematical methods used to calculate several quantities (the JR-71 model was designed to reduce computation time and the size of the required data tables), the results for J71 also apply to JR-71[25, 48].

Since the average error in the density model is zero, a weighted least-squares method provides an appropriate fit. For this method, each error term is defined as:

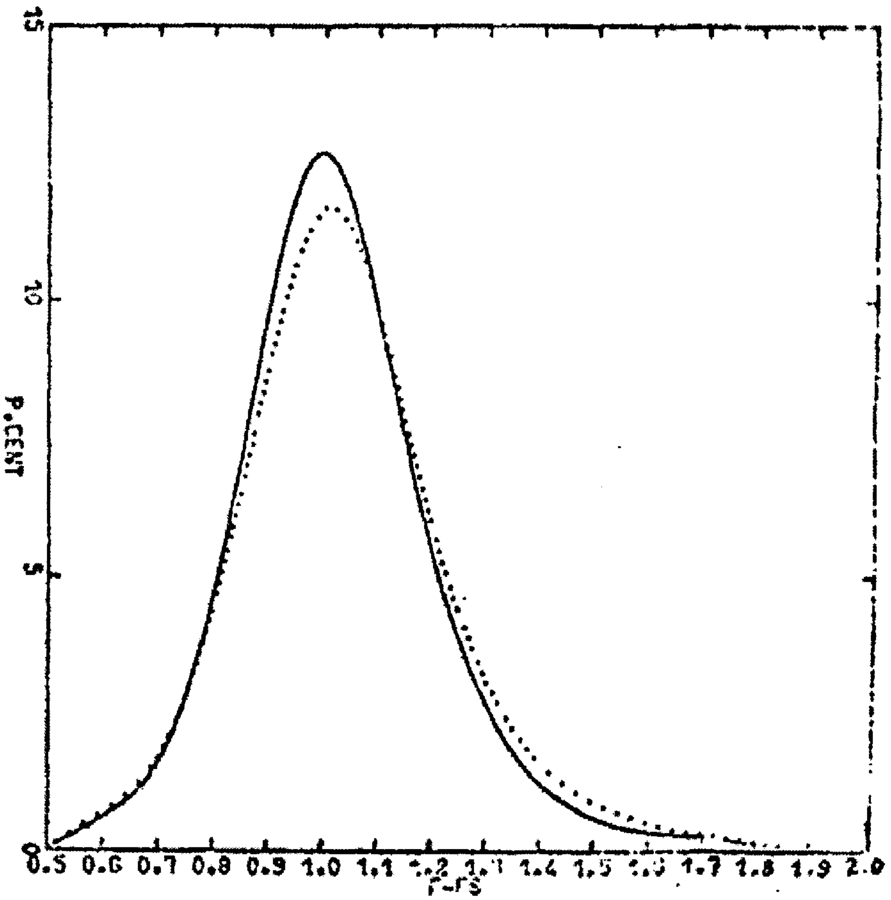


Figure 2-3: Ratio of True Density to Jacchia 1971 Model Density

$$\Delta_{ij} = \frac{\hat{k}_{ij}}{\overline{k}_i} - 1 - \left(b_{1j} + b_{2j} \left(\frac{h_{ij} - 400}{200} \right) \right) \tag{2.7}$$

The Δ-terms are grouped into a matrix:

$$\boldsymbol{\Delta}_{\mathrm{j}} = \begin{bmatrix} \Delta_{1j} \\ \vdots \\ \Delta_{mj} \end{bmatrix} \tag{2.8}$$

Some satellites have more well-known tabulated ballistic factors than others, and the weighting matrix reflects this[7].

$$\mathbf{W} = \begin{bmatrix} \frac{1}{\sigma_1^2} & & \\ & \ddots & \\ & & \frac{1}{\sigma_m^2} \end{bmatrix} \tag{2.9}$$

Next, we define two matrices, **F** and **B**, which together give the linear correction equation detailed in Equation 2.1.

$$\mathbf{F_j} = \begin{bmatrix} 1 & (h_{1j} - 400)/200 \\ \vdots & \vdots \\ 1 & (h_{mj} - 400)/200 \end{bmatrix} \tag{2.10}$$

$$\mathbf{b_j} = \begin{bmatrix} b_{1j} \\ b_{2j} \end{bmatrix} \tag{2.11}$$

Lastly, we define a matrix $\mathbf{a_j}$ of the ballistic factor ratio terms.

$$\mathbf{a_j} = \begin{bmatrix} (\hat{k}_{1j}/\bar{k}_1) - 1 \\ \vdots \\ (\hat{k}_{mj}/\bar{k}_m) - 1 \end{bmatrix} \tag{2.12}$$

We can then write a cost function using the matrices defined in the previous equations:

$$I(\mathbf{b_j}) = \mathbf{\Delta_j^T W \Delta_j} = (\mathbf{a_j} - \mathbf{F_j b_j})^{\mathbf{T}} \mathbf{W} (\mathbf{a_j} - \mathbf{F_j b_j}) \tag{2.13}$$

That cost function has the standard least-squares solution:

---

[7] Standard and non-standard satellites are treated identically in this step, since the tabulated ballistic factor variances already reflect that standard satellites have better-known characteristics. See Section 2.5 for the definitions of standard and non-standars satellites, and for details on reducing the variances for non-standard satellites.

$$\hat{\mathbf{b}}_\mathrm{j} = \left(\mathbf{F}_\mathrm{j}^\mathrm{T}\mathbf{W}\mathbf{F}_\mathrm{j}\right)^{-1}\mathbf{F}_\mathrm{j}^\mathrm{T}\mathbf{W}\mathbf{a}_\mathrm{j} \tag{2.14}$$

These linear correction factors are constant throughout their respective three-hour spans, and change only with height. Latitude and longitude are not included. Like the decision to use only a linear, rather than a second or higher-order model, this choice was made by Nazarenko in order to avoid attempting to extract too much information from limited data. If location-dependent phenomena dominate the remaining errors when such a correction is applied, and sufficient data is available, this limitation should be re-examined.

2.3.4 Solution Boundaries

Any values in $\mathbf{a}_\mathrm{j}$ that exceed a certain tolerance (usually 3–sigma) are discarded before the least-squares solution is carried out. This should discard any outlying values, possibly due to flawed data. Another test is performed by placing a tolerance on the ρ value GTDS gives as a measure of convergence. Since a large amount of data is available, it seems preferable to simply throw out any questionable points.

Boundaries have also been set on how large these linear correction factors can be [13, 41]. These boundaries are based on the fact that a maximum of 30% error at low altitudes, and a factor of two error at high altitudes seem appropriate based on observations of errors[34]. These rules yield the following boundary equations (at time t_j):

$$\left.\frac{\delta\rho}{\rho_m}\right|_{h=200} = b_{1j} - b_{2j} \in (-0.3, 0.3) \tag{2.15}$$

$$\left.\frac{\delta\rho}{\rho_m}\right|_{h=600} = b_{1j} + b_{2j} \in (-.5, 2.0) \tag{2.16}$$

2.4 Forecasting Linear Correction Factors

The next section of AtmoCal is that which predicts these correction factors into the near future. Since the prediction equations are identical for b_{1j} and b_{2j} , the generic variable $x(t)$ will represent either of them in this section. The linear correction factors b_{1j} and b_{2j} are both modeled as measurements of independent stochastic processes. First, each "process" is split into a random and a deterministic component:

$$x(t) = x_d(t) + x_r(t) \tag{2.17}$$

The deterministic component is then modeled as a sum of sinusoids[8], with $\lambda = \frac{2\pi}{T}$ and $T \approx 27$ days (one solar rotation):

$$x_d(t) = \bar{x} + (x_d(t_o) - \bar{x} \cdot \cos(\lambda(t - t_o)) + \frac{\dot{x}_d(t_o)}{\lambda} \cdot \sin(\lambda(t - t_o)) \tag{2.18}$$

An unweighted least-squares curve fit is used to determine the various coefficients ($\bar{x}$, $x_d(t_o)$, and $\dot{x}_d(t_o)$) in the above equation. The random component is modeled as a stationary Gaussian random process, with the correlation function $K_{x_r}(\tau)$ and power spectral density $Sx_rx_r(s)$ as follows:

$$K_{x_r}(\tau) = \sigma^2_{x_r} \cdot e^{-\alpha|\tau|} \tag{2.19}$$

$$S_{x_rx_r}(s) = \frac{2\sigma^2_{x_r}\alpha}{\alpha^2 - s^2} \tag{2.20}$$

Nazarenko and Yurasov empirically determined that $\sigma^2_{x_r}$ should be in the range 0.1–0.6, and α should be .241/day[40]. A scalar Kalman filter can then be used to project the random component into the future, as a function of t_0, the last recorded time.

---

[8] This equation can easily be modified if any major, non-sinusoidal, patterns begin to appear in the corrections, but for now appears suitable.

$$\hat{x}_r(t) = e^{-\alpha(t-t_o)} \cdot \hat{x}_r(t_o) \tag{2.21}$$

This entire operation can also be represented as a block diagram, shown in Figure 2-4.

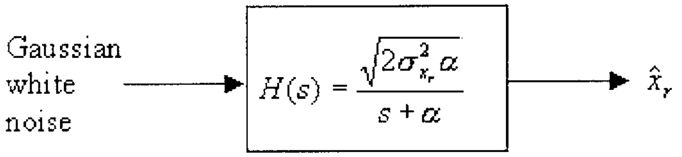


Figure 2-4: Block Diagram for Correction Factor Forecasting

2.5 Ballistic Factor Estimation

2.5.1 BFE basic process

The ballistic factor updating cycle, which is delimited in Figure 2-1 with dotted lines, is the only section of AtmoCal that does not need to run in near real-time. It requires a larger amount of computer space and time, since it must process a span of observations totalling much more than three days. Some ways to reduce the amount of computer resources required will be discussed in Section 7.1. Since this process only needs to be run occasionally (normally once per 27-day solar rotation), it is usually not a problem to allocate the resources required. After the updated ballistic factors are available, they are incorporated into the real-time orbit determination and prediction section.

The first step in improving ballistic factor estimations is the separation of the satellites in the catalog into two groups: "standard", and "non-standard". Standard satellites have well-known, invariant ballistic factors and masses, and should make up 5-10% of the satellites used. Non-standard satellites may have less well-known and/or slowly varying characteristics. (Observations of objects with highly erratic or completely unknown ballistic factors, including debris and satellites undergoing reconfiguration, should be omitted entirely from those used in the atmospheric density correction process. Satellites with abnormally high eccentricity values should also be omitted. These satellites can still benefit from using the corrected atmospheric density model, but should not be included in its creation.) The tabulated ballistic factors for standard satellites will not be changed by the ballistic factor updating cycle.

2.5.2 Derivation for One Standard Satellite

The ballistic factor updating equations are presented first for the case where there is only one standard satellite. This simplifies the derivation, and the equations can then be easily adapted for multiple standard satellites. The heart of the ballistic factor updating algorithm is the use of "quality factors", or Q-factors, which are used to determine how much an individual ballistic factor should be modified to more closely match the results from other satellites. Nazarenko and Yurasov tested five different Q-factors, and the one used in AtmoCal was the one empirically proven to be most effective. The Δ error values below are the same ones defined in Equation 2.7.

First, we define a Q-factor in terms of the error terms for the standard satellite.

$$\begin{aligned} Q_s &= \frac{\sum\limits_{j \in N_s} \Delta_{sj}}{|N_s|}, \text{ where} \qquad (2.22) \\ N_s &= \text{the set of time spans that contain observations of standard satellite } s. \\ |N_s| &= \text{the number of such spans.} \end{aligned}$$

Then, using the same format, we define a Q-factor for each non-standard satellite.

$$\begin{aligned} Q_n &= \frac{\sum\limits_{j \in N_n} \Delta_{nj}}{|N_n|} && (2.23) \\ N_n &= \text{the set of time spans that contain observations of non-standard satellite } n. \\ |N_n| &= \text{the number of such spans.} \end{aligned}$$

First, we want to use these Q-factors to find a global correction factor ξ, which will remove any overall bias in the tabulated ballistic factors of all of the non-standard satellites. Such biases are included in the simulated data validation, although it seems unlikely that a clear-cut division between standard and non-standard satellites would appear in real data. (If no such bias exists, the following formulas can still be applied without changing the tabulated ballistic factors, since ξ will equal 1. If the inclusion of the global correction factor appears to be slowing convergence, it can be turned off with a software option.) In the following formulae, k_n is the actual, unknown ballistic factor of non-standard satellite n, and $\overline{k_n}$ is the *a priori* tabulated value.

$$\overline{k_n} = \xi \cdot k_n \qquad (2.24)$$

To obtain this global correction factor, we begin with the equations for the residual sum of the errors for all non-standard satellites and standard satellites. Since atmospheric density errors are assumed to be zero-mean (see Section 2.3.3), these sums must equal zero, whether we use the tabulated or observed ballistic factors to calculate them. Note that b_{1j} and b_{2j} denote the ideal correction factors, while $\hat{b}_{1j}$ and $\hat{b}_{2j}$ denote the actual values obtained from Equation 2.14.)

For non-standard satellites and empirical measurements: (2.25)

$$\sum_{j \in N_n} \frac{\hat{k}_{nj}}{\overline{k_n}} - \sum_{j \in N_n} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{nj} - 400}{200} \right) \right) = \sum \Delta_{nj} \quad = \quad 0$$

For non-standard satellites and ideal corrections: (2.26)

$$\sum_{j\in N_n} \frac{k_{rj}}{k_n} - \sum_{j\in N_n} \left(b_{1j} + b_{2j}\left(\frac{h_{rj} - 400}{200}\right)\right) = \sum \Delta_{rj} \quad = \quad 0$$

For the single standard satellite and ideal corrections: (2.27)

$$\sum_{j\in N_s} \frac{\hat{k}_{sj}}{k_s} - \sum_{j\in N_s} \left(b_{1j} + b_{2j}\left(\frac{h_{sj} - 400}{200}\right)\right) = \sum \Delta_{sj} \quad = \quad 0$$

By substituting Equations 2.25 and 2.26 into Equation 2.24, we obtain the following relationship:

$$\xi \cdot \sum_{j\in N_n} \left(\hat{b}_{1j} + \hat{b}_{2j}\left(\frac{h_{rj} - 400}{200}\right)\right) = \sum_{j\in N_n} \left(b_{1j} + b_{2j}\left(\frac{h_{rj} - 400}{200}\right)\right) \tag{2.28}$$

The global correction factor ξ approximately represents the bias of the variation model caused by the bias of the tabulated ballistic factors[9]. Each biased non-standard ballistic factor moves the calculated atmospheric density correction factor away from the ideal variation. Thus, the ideal correction factors are related to those observed by the standard satellite by the following equation:

$$\xi \cdot \sum_{j\in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j}\left(\frac{h_{sj} - 400}{200}\right)\right) \approx \sum_{j\in N_s} \left(b_{1j} + b_{2j}\left(\frac{h_{sj} - 400}{200}\right)\right) \tag{2.29}$$

Combining Equations 2.27 and 2.29, we get:

---

[9] The extension of this approximation to multiple standard satellites is based partly on the fact that standard satellites make up only a small fraction of the list of satellites being used for atmospheric density correction. If this is not the case, this equation should be re-examined. The addition scaling factor based on the percentage of standard satellites in the catalog may be required.

$$\xi \cdot \sum_{j \in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{sj} - 400}{200} \right) \right) \approx \sum_{j \in N_s} \frac{\hat{k}_{sj}}{k_s} \tag{2.30}$$

Subtracting $\sum_{j \in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{sj}-400}{200} \right) \right)$ from both sides yields:

$$\sum_{j \in N_s} \frac{\hat{k}_{sj}}{k_s} - \sum_{j \in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{sj} - 400}{200} \right) \right) \approx \sum_{j \in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{sj} - 400}{200} \right) \right) \cdot (\xi - 1) \tag{2.31}$$

Substituting the expression for Q_s and $|N_s|$ as defined in Equation 2.22 into the left-hand side of the prior equation, and rearranging, we obtain the final definition for ξ:

$$\xi \approx 1 + \frac{Q_s \cdot |N_s|}{\sum_{j \in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{sj}-400}{200} \right) \right)} \tag{2.32}$$

We then turn to the individual satellite correction factor, which is determined from the average bias between an individual satellite's k-value and those of all of the non-standard satellites. The derivation is derived in an identical fashion to the derivation of ξ above, and is not repeated here. The resulting individual correction factor ψ_n is:

$$\overline{k_n} = \psi_n \cdot k_n \tag{2.33}$$

$$\psi_n \approx 1 + \frac{Q_n \cdot |N_n|}{\sum_{j \in N_s} \left(\hat{b}_{1j} + \hat{b}_{2j} \left(\frac{h_{nj}-400}{200} \right) \right)} \tag{2.34}$$

This entire operation is summarized in Figure 2-5.

Nazarenko determined that 3–4 iterations (with a cycle length of 20 or more days) were normally enough to ensure convergence, and that convergence is improved if the global correction factor is only applied on the first iteration. This makes logical sense, since there should be sufficient data in the initial cycle to remove a simple bias in the table. AtmoCal is normally set to operate in this fashion, although that behavior can be easily changed.

2.5.3 Multiple Standard Satellites

Expanding the ballistic factor algorithm to include multiple standard satellites is very straightforward. The single Q_s value is replaced by a height-dependent linear function $F(h)$:

$$F(h) = a_1 + a_2 \cdot \frac{h - 200}{200} \tag{2.35}$$

Each individual Q_s value is viewed as a noisy (white, zero-mean Gaussian noise) measurement of $F(h)$, taken at the average perigee height $\overline{h}_s$ for that satellite over the entire update cycle. An unweighted linear least-squares fit is used to fit $F(h)$ to the list of Q_s values. Then, $F(\overline{h}_n)$ is calculated for each individual non-standard satellite, and substituted for Q_s in Equation 2.32.

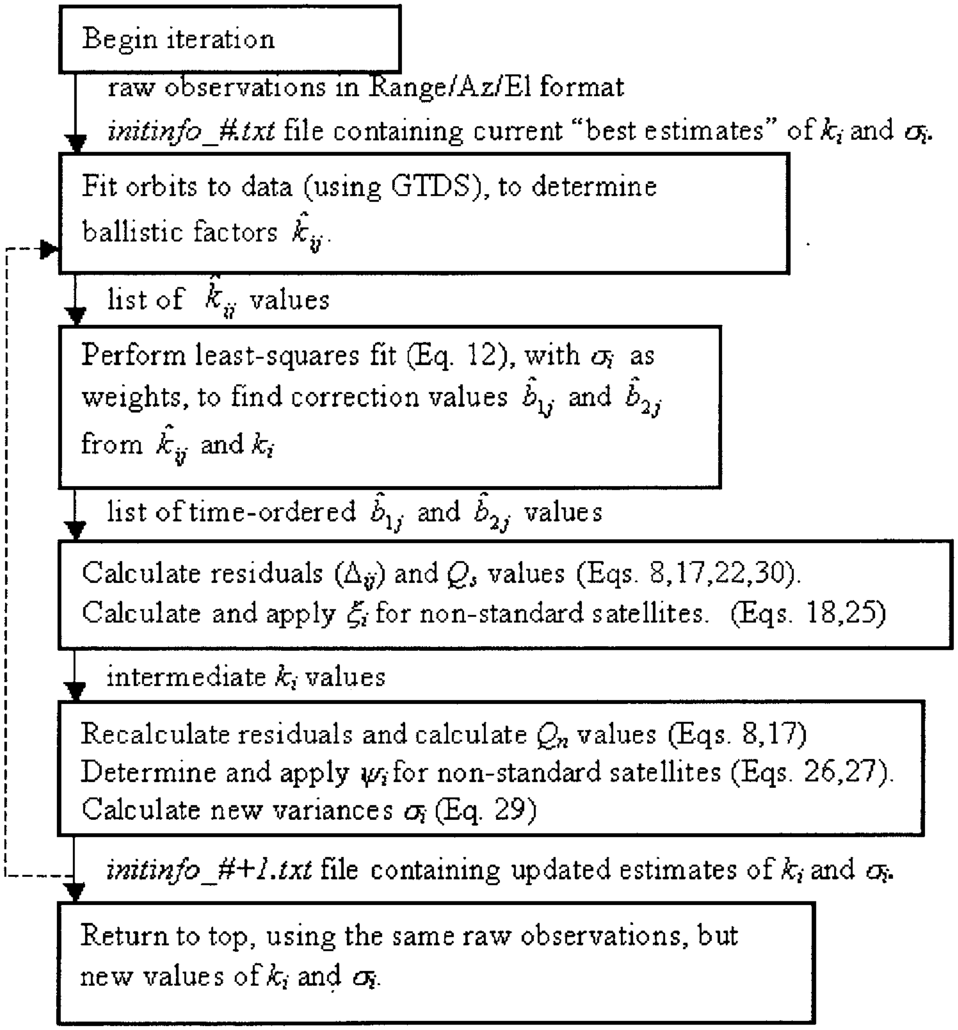


Figure 2-5: Flowchart for Ballistic Factor Updating Cycle

Chapter 3

Implementation Overview

3.1 Computer Code

George Granholm chose to use Perl as the main language for AtmoCal, since it is especially good at handling file input/output and UNIX process control. The ability to spawn multiple subprocesses allows AtmoCal to run more quickly on a multiprocessor computer, since multiple copies of GTDS can run at once, on different processors. Perl is more user-friendly and tolerant of slight differences in input file format than languages like FORTRAN and C, and is far more flexible and portable than using UNIX shell scripts. Perl also does not need to be manually recompiled when changes are made, which means that small changes can be made in one script without needing to recompile and relink the entire set of AtmoCal routines. For these reasons, Perl was chosen for AtmoCal, and is generally the language used to create "wrappers" for older FORTRAN programs[59].

Matrix algebra in Perl is facilitated by using the MatrixReal module, but detailed analyses and statistics are still clumsy in Perl. Thus, MATLAB was used for the in-depth mathematics involved in calculating and predicting the b_{1j} and b_{2j} atmospheric density correction coefficients. Several MATLAB scripts were also created for analyzing and graphing results, and have been included in AtmoCal.

3.1.1 GTDS

The first step in creating AtmoCal was to modify GTDS to include atmospheric density corrections. Granholm chose, after some examination, to start with Jack Fischer's NT-GTDS PR-5 version. This version was ported to the SGI-UNIX platform and validated using the standard "Metzinger" test cases[38]. The JR-71 model, which is fully supported in PR-5 GTDS, was altered to include the option of reading b_{1j} and b_{2j} values from a file and applying them after all of the other model effects are calculated. This altered version of GTDS is henceforth referred to as gtds_granholm. Several main GTDS routines were changed, a routine called CALCCALJAC was added to calculate the appropriate density correction from the b-values, and a new optional GTDS control card called ATMCAL was created. This card includes an option for specifying the underlying density model to be corrected, although only JR-71 is currently supported. The file containing b_{1j} and b_{2j} values has been given a reference number (106), and the three routines that calculate JR-71 density in various regions have been altered. Note that this means that, while corrections are calculated in the 200-600 kilometer range, they are applied throughout the JR-71 model, starting at 90 kilometers. A detailed list of the changes made to each file are listed in Appendix C, as well as a listing of the precise versions of each binary data file containing GTDS physical model information that are required to reproduce the validation cases and the results in this thesis.

When the project was moved from the Charles Stark Draper Laboratory (CSDL), to the MIT Lincoln Laboratory (LL), both the unaltered GTDS and gtds_granholm were compiled on the new machine (also an SGI-UNIX platform) and re-validated using the Metzinger test cases. Both versions were placed under version management using CVS (Concurrent Version Management System)[46, 47]. No modifications except for the addition of new coordinate system transformations[1] not used by AtmoCal or the Metzinger test cases have been made since the validation. Shell scripts to run

---

[1] These routines were added by Paul Cefola and Zach Folcik, and are not described in this thesis.

each of the Metzinger test cases, along with two added test cases for the NAVSPASUR PPT2 routines and one new test case for the atmospheric density correction routines are also included in the CVS tree for gtds_granholm. (See Appendix C.3 for more details on running these new test cases.)

The current version of gtds_granholm inherited several limitations and bugs from the NT-GTDS version. Three of these were fixed by George Granholm, and should be re-incorporated into any new versions of UNIX-GTDS, even if those versions do not contain the atmospheric density modifications. These were: the ability to produce ascii, rather than binary, output files was added (by porting the appropriate sections of VAX-GTDS, which already had this capability), a bug that crashed DC runs that spanned a year boundary, and a bug that would crash DATASIM runs if no observations were created for a specific satellite and station. More details are included in Granholm's thesis, on page 59[13].

3.1.2 AtmoCal

AtmoCal is written mainly in Perl, since that language handles large data files elegantly, and also is capable of sending the many GTDS runs required to different processors on a multi-processor machine, if available. Some of the large matrix calculations, including the main weighted least-squares solution, are implemented in MATLAB[37]. The MATLAB sections were modified from Granholm's versions so that no extra packages beyond basic MATLAB were required, and were tested to verify that no changes were required after MATLAB was upgraded from version $5.x$ to $6.x$. The entire AtmoCal source code was put under version management using CVS. The version number for all files at the publication of this thesis was set to 2.0 to facilitate easy retrieval of the version used to produce the results contained herein[2]

The minor changes made to AtmoCal are too numerous to describe in detail, but the main categories of changes were: many corrections of typographical errors,

---

[2] See Appendix B for information on retrieving a particular version by number.

replacements of hard-coded directory paths to ones involving environment variables, and more user-defined options to increase flexibility. The user-defined options are now all located in a block at the beginning of each program. Driver programs (*runestbfs.pl*, *runcalcvars.pl*, and *bfe_iter.pl*) for the *estbfs.pl* and *calcvars.pl* subroutines were created to manage running both the normal near real-time correction-finding process and the longer ballistic factor iteration.

3.2 Data Flow

The various scripts and data files used to run the main portion of AtmoCal can be summarized in Figure 3-1, which is a modified version of Figure 2-1.

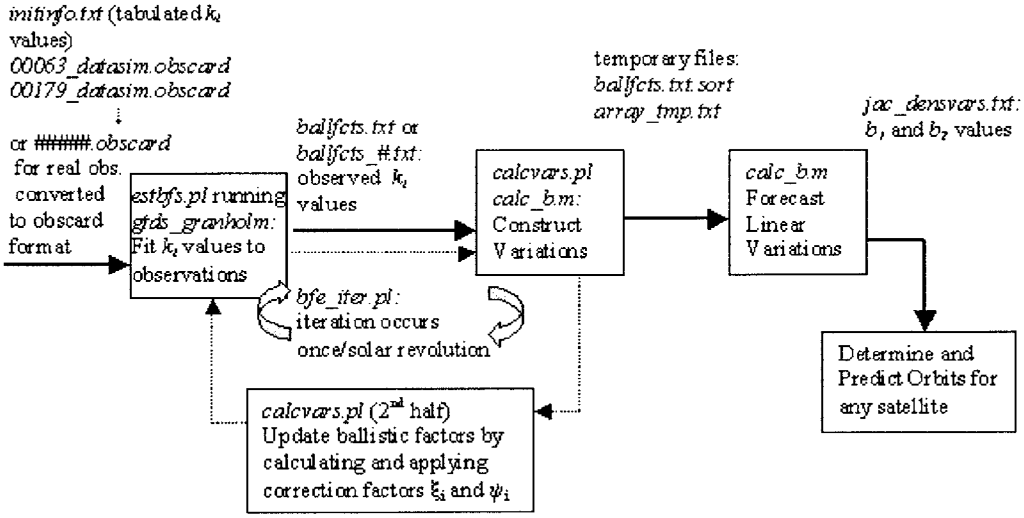


Figure 3-1: AtmoCal Operation Flowchart Including File Names

Descriptions of the layout of all of the AtmoCal file types (and some other formats) are listed in Appendix E. The preparation of the *initinfo.txt* and OBSCARD data files for real and simulated data sets are detailed in Sections 4.1.1 (simulated) and

6.1 (real).

One important thing to note when working with AtmoCal is that there are up to three separate areas where files are stored. Small input and output files, including the tables of satellite characteristics, the output b-values, and the various logs created during operation, are located in the same directory structure as the AtmoCal code itself. The large number of long ascii data files created by individual GTDS runs are stored in another directory structure, allowing the large files to be kept on a different disk, if desired. (This was the case on the machine at CSDL.) Options were added to automatically delete some or all of these large files after the data relevant for AtmoCal (usually the ballistic factor and the convergence measure) have been extracted. Finally, the gtds_granholm code may be in an entirely separate location, if desired. The locations of all three file structures are specified by environment variables (instructions on setting these up can be found in Appendix B.3). An overview of the file structures can be found in Appendix B.2.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Simulated Data Validation

In order to prove that the AtmoCal code is operating properly, and to determine that the underlying algorithm is providing useful corrections, a validation process was designed and begun by George Granholm. This validation process used simulated satellite orbits, since this approach produces a "truth" orbit, which can then be used to evaluate the success of the algorithm. Granholm completed all sections of the validation except for testing the effects of errors in the initial ballistic factors, and the efficacy of the ballistic factor iteration.

4.1 Simulated Data Generation

Two months of real tracking data were graciously provided to this project early in 2000, by Lt. Col. Dave Vallado (USAF). The simulated data was constructed to closely match the real data, both in time and satellite distribution. This was done to ensure that the simulated data was representative of what would normally be available in real-time operation[4], as well as to facilitate the transition to real data validation.

One noteworthy facet of both the simulated and real data sets is that, since they cover a period beginning on December 15, 1999 and ending on February 12, 2000, Y2K problems with GTDS and helper utilities are very obvious, and had to be addressed.

NORAD assumes that all dates fall in the range 1956–2055, and this assumption has been used throughout AtmoCal when two-digit dates were required. In some cases, the year 2000 is denoted by year "100", and, while clumsy, this notation is compatible with the GTDS OBSCARD data format.

This validation method differs slightly from the one used by Nazarenko and Yurasov: they directly simulated the observed ballistic factors, and added error and noise at that point. Simulating the actual satellite observations adds another layer of complexity to the process, and made working with simulated data closely resemble processing real data.

4.1.1 Preparation

To operate AtmoCal with either real or simulated data, we must compile a list containing an initial orbit estimation and an *a-priori* ballistic factor estimate for every satellite being used for density correction. The first step in determining these is to obtain the real two-line element sets (TLEs) for each satellite at the beginning[1] of the fit interval. These TLEs were obtained via the Jet Propulsion Laboratory's anonymous FTP site[54]. This site is, unfortunately, no longer available (as of February 5, 2002).

George Granholm sorted these TLEs to find the 454 with perigee heights in the 200–600 kilometer range, and then eliminated any satellites with apogee heights above 800 kilometers. Objects known to be debris or in a rapidly decaying orbit were discarded, leaving 335 objects. These 335 satellites were then the only ones used both in simulated and real data processing.

The remainder of the preparation is automated by the *TLE2osc.pl* Perl script. The script begins by processing the TLE file, formatting the TLE for each of these objects to be compatible with GTDS, which requires two conversions. First, the

---

[1] To be more precise, the TLEs must be at least one minute before the beginning of the fit interval, but should be as close to this time as availability permits.

NORAD day-of-year, which is given in the form YYDDD, must be converted to a Julian date. The `Dates.pm` Perl module was created, and contains formulae for converting between Julian and Gregorian calendar dates[10]. Second, the ballistic coefficient must be converted from BSTAR (which is in measured units of inverse Earth radii) to the format given in Equation 2.2 (where k is measured in m^2/kg). This is done using the following formula, adapted from one defined by Vallado[55]:

$$k = 6.3708105 * BSTAR \tag{4.1}$$

This k value is then separated into drag coefficient $\mathrm{C_D}$, cross-sectional area $\mathrm{A_x}$, and mass m, by assuming that $\mathrm{C_D} = 2.2$ and using the radar cross-section (RCS) for $\mathrm{A_x}$, we can solve for the mass[43]. The RCS values are taken from a file (also provided by Dave Vallado)[56]. These k values are also used to create the table of *a priori* ballistic factors contained in the *initinfo.txt* file, details of which can be found in Appendix E.3.

Once these conversions are complete, GTDS EPHEM can be used to convert the TLEs into osculating elements, to propagate the truth orbits (creating the *.output* files required by GTDS DATASIM for simulating observations) and the *.output* files containing *a priori* state vectors, required by *estbfs.pl*[21].

4.1.2 Truth Orbit Generation

The *genobs.pl* script then uses GTDS DATASIM routine to propagate these initial ephemerides forward. Only four ground stations were used to produce all observations: Eglin AFB, Florida (EGLQ); Kaena Point, Hawaii (KAEQ); Fylingdales, England (FLYQ); and Grand Forks, North Dakota (PARQ). These locations were chosen by Granholm to produce observations similar in quantity and geometry to those found in real NORAD data. Since Kaena Point is the only one of the four not equipped with a phased array radar system, its observation rate was modeled as half that of the other stations.

In order to speed up the process, Granholm chose to use a truncated version (4x4) of the JGM2 gravity model. Since the same gravity model is used throughout the validation process with simulated data, it should not affect the results. (This is the reason for the "lowgrav" designations seen in the directory structures for the simulated data files.) It is, however, recommended that a more accurate model be used when working with real data. (To alter this, look in the GTDS keyword list[15] for information on the POTFIELD, MAXDEGEQ, and MAXORDEG cards in the GTDS input deck.)

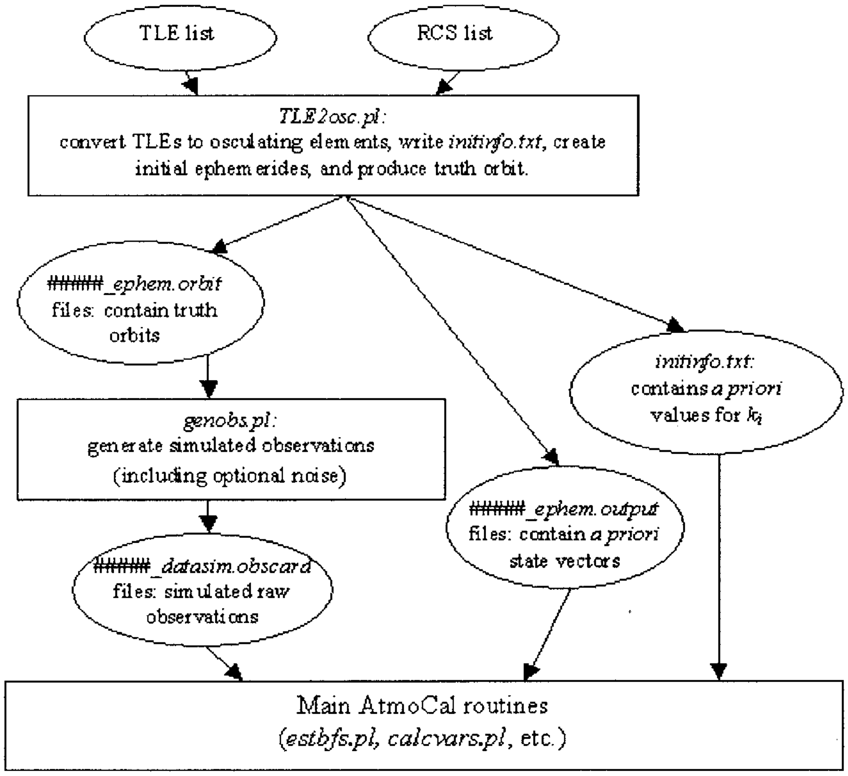


Figure 4-1: Flowchart for Simulated Observation Creation

Table 4.1: Statistics for B-values with No Noise, No Mismodeling

Mean Value of b_{1j}	1.4066e-08
Mean Value of b_{2j}	-2.0349e-08
Largest Value of b_{1j}	7.7292e-08
Largest Value of b_{2j}	-1.71749e-07
Largest Correction Factor	2.1312e-07
(taken at 200 km during span #55/56)	

4.2 Reproduction of Prior Results using Simulated Data

To verify that all sections of AtmoCal were working properly on Pisces, George Granholm's validation process was repeated and compared to the original results. First, the *TLE2osc.pl* and *genobs.pl* scripts were re-run to produce simulated data, and compared to the older versions using the `xdiff` command. The results were identical to those obtained by Granholm. (The newly created noisy data had different noise values, obviously, but the underlying truth orbits and the characteristics of the noise were identical to Granholm's results.)

4.2.1 Data Flow Verification

The first test was designed to catch any major problems in data input/output. (Y2K errors, mismatched coordinates, etc. On Pisces, the main concern was finding any inconsistencies in file management left over from the transfer.) Observations were generated without any noise, the same values of $F_{10.7}$ and a_P used for orbit generation were used by the ballistic factor estimation process. Since the models are identical, any deviation of the b-values from zero should only be the result of round-off and GTDS fit-convergence error. The resulting b-values Granholm obtained were extremely small, proving that this was, in fact, the case. These results were reproduced on Pisces. (Compare Figure 4-2 to Figures 5.1 and 5.2 of Granholm's thesis[13], noting that the scales differ substantially.)

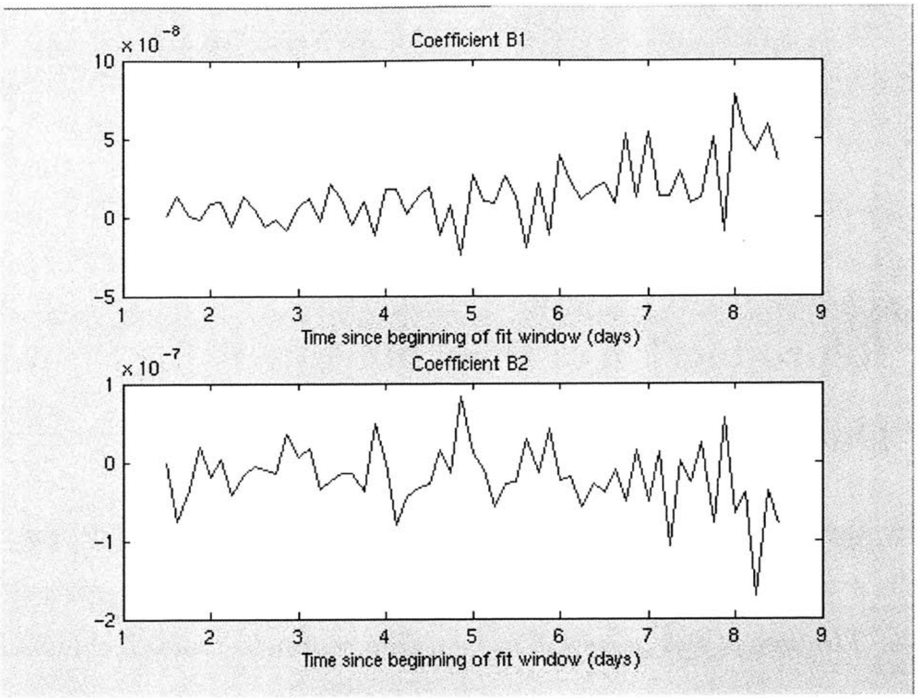


Figure 4-2: B-values with No Noise, No Mismodeling

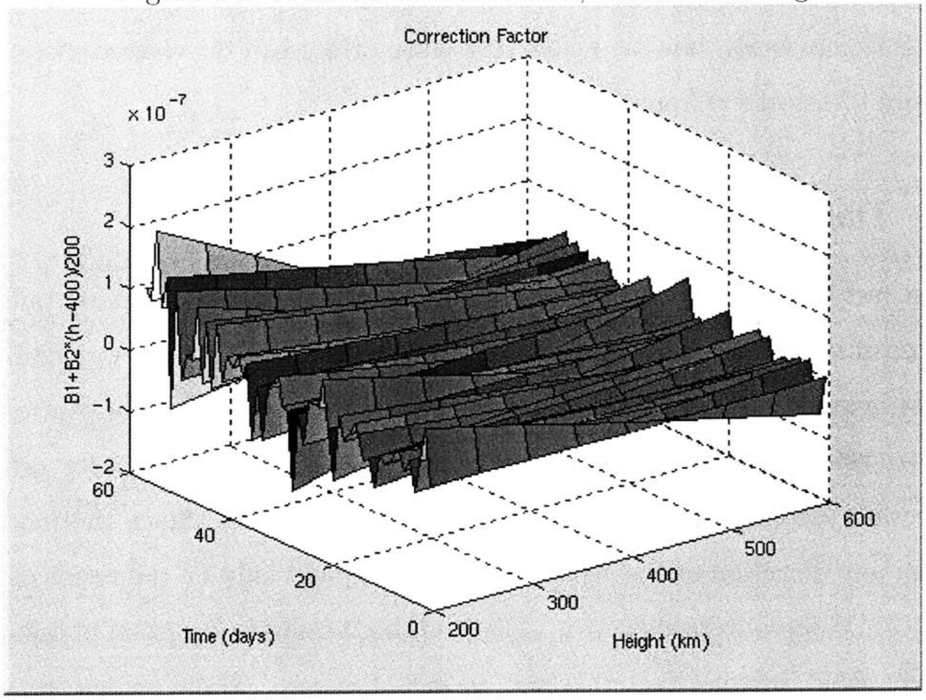


Figure 4-3: Linear Atmospheric Density Correction Factors with No Noise, No Mismodelling

4.2.2 Differences between Truth and Fit Models

Once the basic data flow had been verified, model error was introduced. As mentioned in Section 1.1.4, the major sources of error in JR-71 are due to the inability of the a_P/K_p and $F_{10.7}$ indicies to properly reflect atmospheric effects, and to the difficulties inherent in predicting these indices. To simulate these errors, the observed a_P and $F_{10.7}$ values (which were used in data creation) are replaced by those found[2] using Ken Schatten's prediction method[49]. Schatten's predictions consist of a single number per month, and intermediate values are interpolated. Using the Schatten predictions simply requires replacing the GTDS binary file *jrdat_nomn_new.dat* with *jrdat_nomn.dat*, which only contained real observations through mid-1997.

The differences between the observed[3] and Schatten values for a_P and $F_{10.7}$ during the entire simulated data period are shown in Figures 4-4 and 4-5[4]. Values of $F_{10.7}$ and a_P for just the fit period used in the simulated data fit windows are given in 4-6, using the same scale as the graphs of b_{1j} and b_{2j} values, to facilitate comparison.

With this mismodelling, using the same simulated observations as in Section 4.2.1, new atmospheric density corrections were generated. These are summarized in Table 4.2 and Figures 4-7 and 4-8.

---

[2] It appears that Granholm was using specifically the "late" series of prediction values according to the lists on Schatten's web site[49].

[3] Geomagnetic activity is shown as daily mean a_P in order to make the graph more legible, but GTDS actually uses the more accurate 3-hour values, and uses K_p in place of a_P.

[4] These figures are similar to Figures 4.3 and 5.9 in Granholm's thesis.

Table 4.2: Statistics for B-values with No Noise, Schatten Mismodeling

Mean Value of b_{1j}	-0.010652
Mean Value of b_{2j}	0.0047594
Largest Value of b_{1j}	0.22353
Largest Value of b_{2j}	0.15967
Largest Correction Factor	0.38028
(taken at 600 km during span #29/225)	

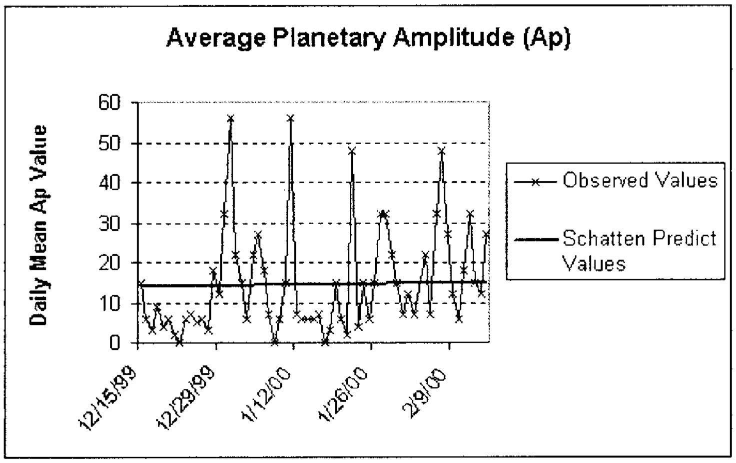


Figure 4-4: Observed Daily Mean and Schatten Ap Values for 12/15/99–2/15/00

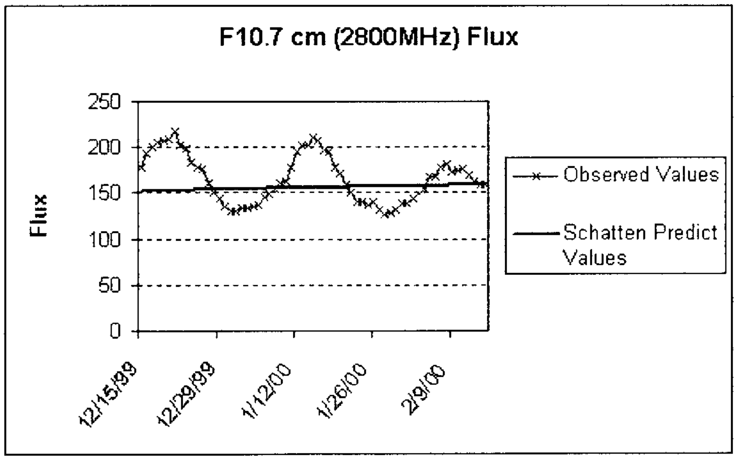


Figure 4-5: Daily and Schatten $F_{10.7}$ Values for 12/15/99–2/15/00

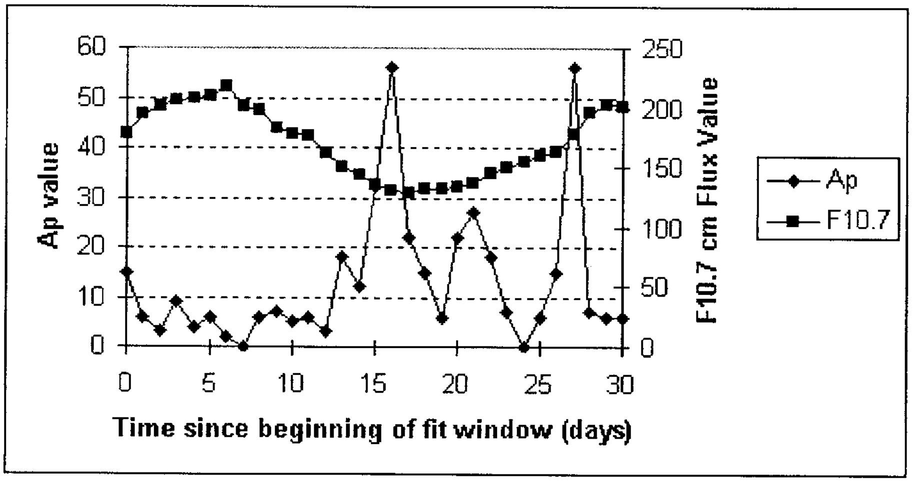


Figure 4-6: Observed $F_{10.7}$ and a_P Values During Fit Window

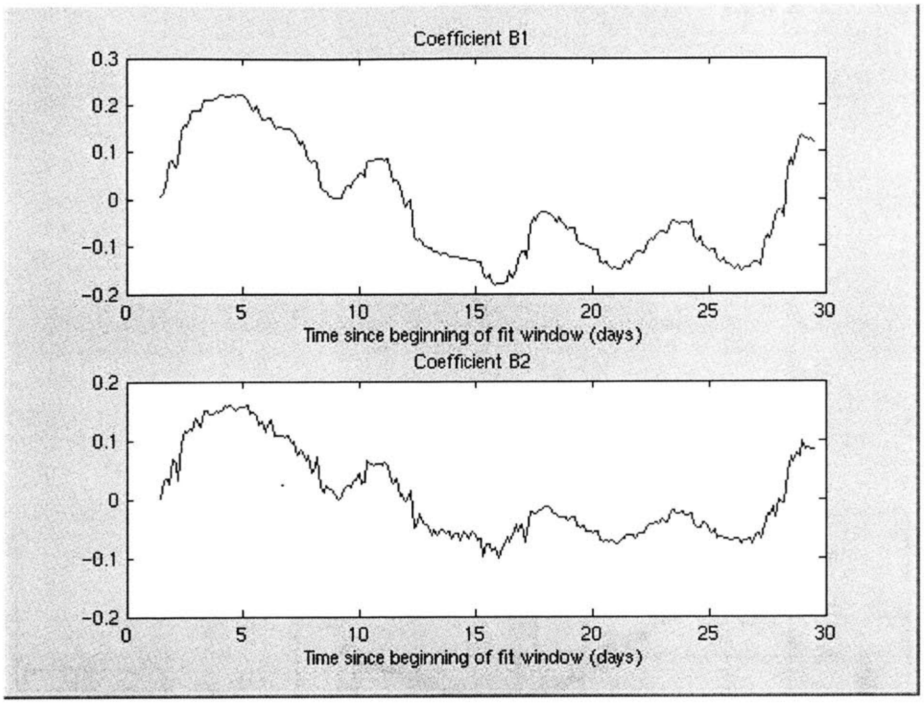


Figure 4-7: B-values with No Noise, Schatten Mismodeling

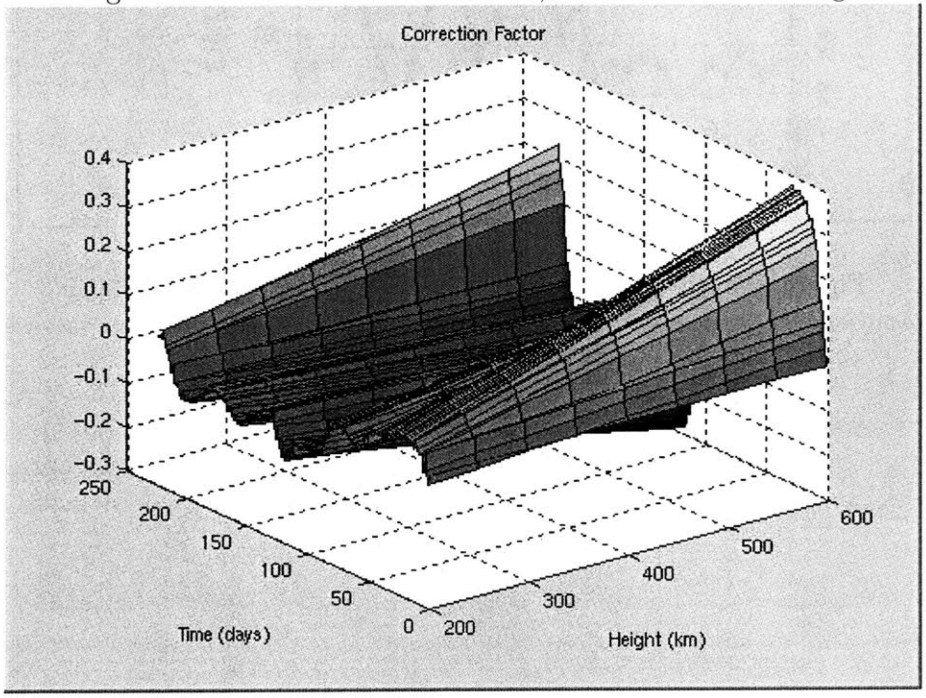


Figure 4-8: Linear Atmospheric Density Correction Factors with No Noise, Schatten Mismodeling

4.2.3 Simulating "Noisy" Observations

Noise was added to the observations using parameters determined by Capt. Jack Fischer (USAF) in his Station Location and Accuracy Database (SLAD), described on page 317 of his thesis[9]. In the SLAD, he lists an accuracy figure[5] for each type of observation available from over 70 stations. These 70 stations include EGLQ, FLYQ, KAEQ, and PARQ, the four used for creating the simulated data.

When the gtds_granholm code is compiled with optimization on, Granholm discovered that the random number generation used to add noise to observations is no longer random. This problem was not present when the "debug" version of gtds_granholm, which is not optimized[6], is used. Until the origins of this bug are traced and fixed, *genobs.pl* should always be set to call "gtds_dbg.exe" instead of "gtds.exe". The optimized version can and should be used in all other sections, since it runs faster.

Finally, noisy observations were created, using the same orbit ephemerides as the non-noisy observations. A somewhat perplexing facet of the GTDS *randu.for* routine was discovered while creating new noisy observations. The new observations were, in fact, identical to the old noisy observations. While this did verify that the noisy-observations process had not been disturbed by changes made to other sections of *genobs.pl*, this was not the desired result. The *randu.for* GTDS subroutine already was noted to show problems when optimized – this may be another symptom of a larger problem in the GTDS random number generation.

Using the noisy observations and no mismodeling, the b_{1j} and b_{2j} values were larger than without noise, but still clustered around zero and showed no particular pattern. (See Table 4.3, Figure 4-9, and 4-10.) The largest values came during the first span, and may have been partly due to some sort of edge phenomenon. Once the

---

[5] Please note that, due to the sensitive nature of the SLAD contents, as well as the data used to create the SLAD, the actual accuracy figures are not publicly available. Please contact Dr. Paul Cefola at the MIT Lincoln Laboratory or Dr. Ron Proulx at the Charles Stark Draper Laboratory for information about obtaining a copy of the full SLAD results.

[6] A separate makefile for compiling this debug version is included in the gtds_granholm CVS distribution.

Schatten data was substituted, the b-values closely resembled those seen found using noiseless observations and Schatten data. (See Table 4.4, Figure 4-11, and 4-12.) Therefore, we can conclude that the amount of noise seen in these observations does not appear to introduce significant errors in determining b_{1j} and b_{2j} values when other errors are present.

Table 4.3: Statistics for B-values with Observation Noise, No Mismodeling

Mean Value of b_{1j}	2.8290e-05
Mean Value of b_{2j}	-2.2410e-04
Max Value of b_{1j}	-1.0398e-03
Max Value of b_{2j}	8.3397e-03
Max Correction Factor	8.2578e-03
(taken at 600 km during span #1/56)	

Table 4.4: Statistics for B-values with Observation Noise, Schatten Mismodeling

Mean Value of b_{1j}	-0.010364
Mean Value of b_{2j}	4.7286e-03
Max Value of b_{1j}	.22318
Max Value of b_{2j}	.16058
Max Correction Factor	.37960
(taken at 600 km during span #25/225)	

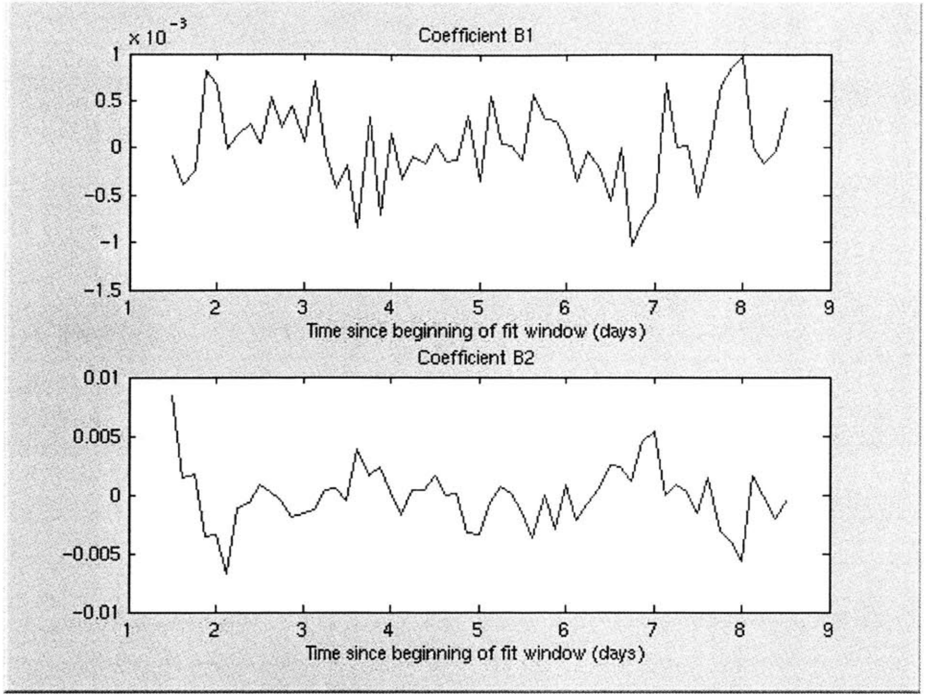


Figure 4-9: B-values with Observation Noise, No Mismodelling

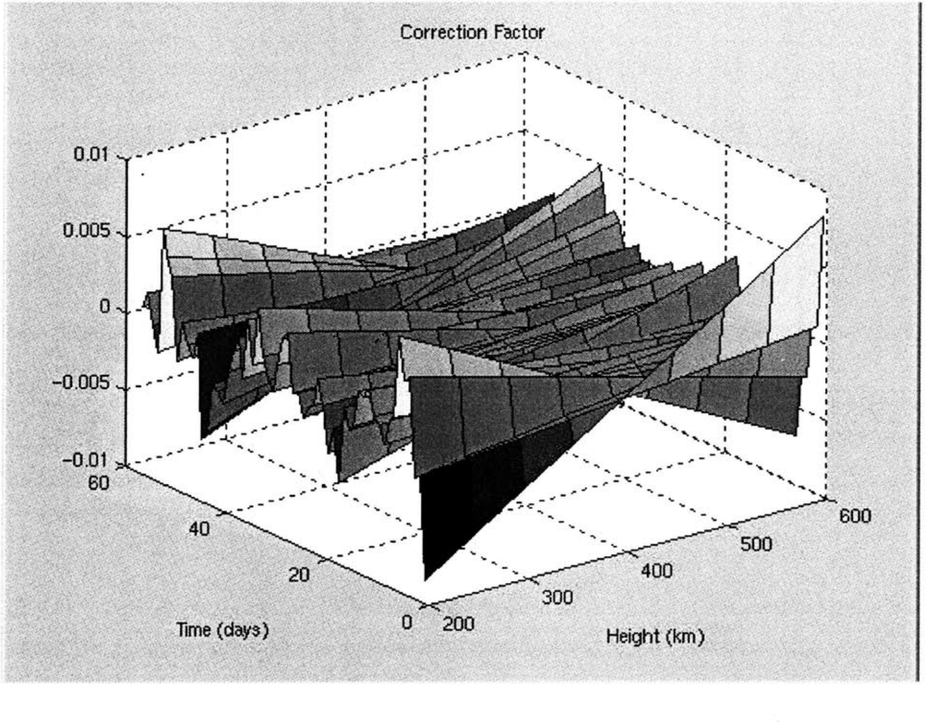


Figure 4-10: Linear Atmospheric Density Correction Factors with Observation Noise, No Mismodeling

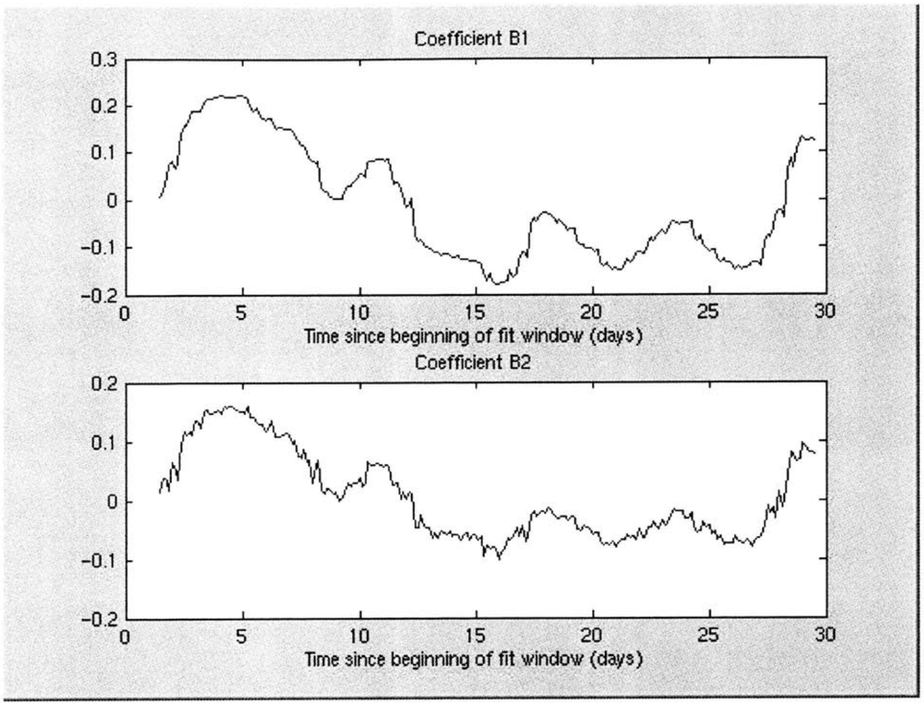


Figure 4-11: B-values with Observation Noise, Schatten Mismodelling

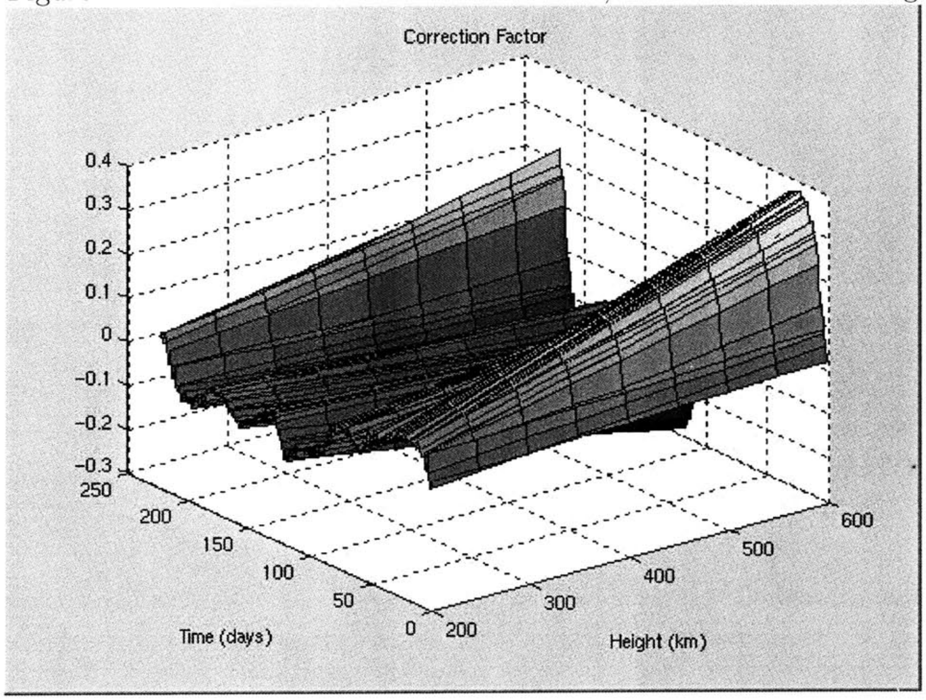


Figure 4-12: Linear Atmospheric Density Correction Factors with Observation Noise, Schatten Mismodeling

Chapter 5

New Validation Results with Simulated Data

The only section of AtmoCal that Granholm did not validate was the update cycle for ballistic factor estimation (BFE). This section is marked with dotted lines in Figure 2-1. Section 2.5 contains the mathematical details of the BFE updating cycle. In order to validate this section, the updating cycle must be shown to provide improvements to inaccurate *a priori* ballistic factors.

The same simulated data (with and without observation noise), as described in Chapter 4 were used to test the BFE updating cycle. First, the entire process was run without noise, mismodeling, or inaccuracy in the initial ballistic factors. Then, the original set of tabulated ballistic factors were separated into "standard" and "non-standard" satellites, and the *a-priori* ballistic factors for non-standard satellites were distorted. The atmospheric density correction process was run, with both noise and density mismodeling included, and updated ballistic factors were calculated for non-standard satellites.

5.1 Ballistic Factor Distortion

Every tenth satellite (when sorted by NSSC#) was chosen to be a standard satellite. As is shown in Figures 5-1, 5-3, 5-2 and 5-4, this yielded a representative group of satellites. All of the other satellites were designated non-standard, and distorted ballistic factors (k_n) were generated for these satellites. Non-standard satellites were all initially assigned the same σ_i value, which was twice the one assigned to standard satellites.

First, all of the original k_n values were multiplied by a global distortion factor (ξ_1, uniformly distributed between zero and one), and then each individual k_n was multiplied by a different, individual distortion factor (ξ_2, also uniformly distributed between zero and one). The distortion equation, with weights m_k and a_k is:

$$\bar{k}_i = k_i(1 + m_k(\xi_1 - 0.5) + a_k(\xi_2 - 0.5)) \tag{5.1}$$

This is almost identical to the distortion method used by Nazarenko and Yurasov, except that Granholm used a larger individual bias[1]. Granholm chose $a_k = 1.6$, rather than Nazarenko's choice of $a_k = 0.4$, but both used $m_k = 1$. The MATLAB program *distort_bfs.m* was created by Granholm to perform this distortion, and was modified to eliminate use of the optional MATLAB statistics package. The global distortion factor was also made optional, set by a flag in the code. Figure 5-5 shows examples of the results of the distortion, with and without a global bias. These results were typical, and were used in the all of the cases shown in the results section.

---

[1] It seems unrealistic to assume that all non-standard satellites have a bias that is not present in the standard satellites. Since the primary goal of this investigation, however, was to reproduce Nazarenko and Yurasov's results, the global bias was included. It seems likely that when working with real data, a global bias will not be present, and the global correction factor need not be calculated.

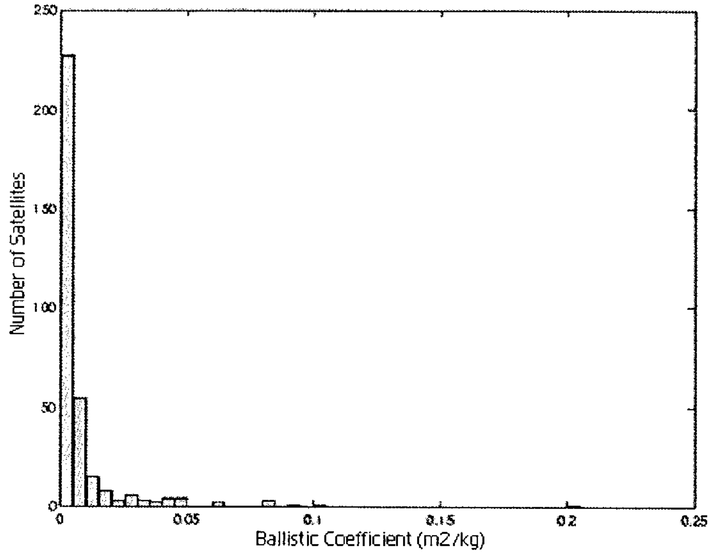


Figure 5-1: Distribution of (undistorted) Ballistic Factors for All Satellites

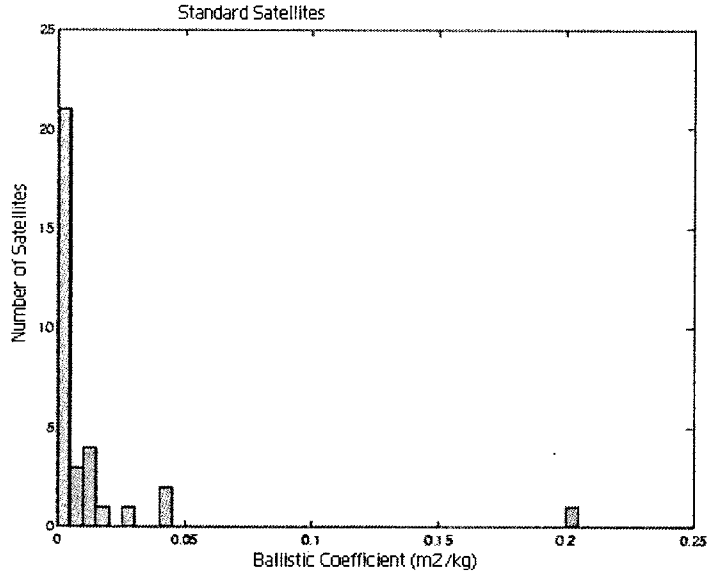


Figure 5-2: Distribution of Ballistic Factors for the Standard Satellites used in BFE validation

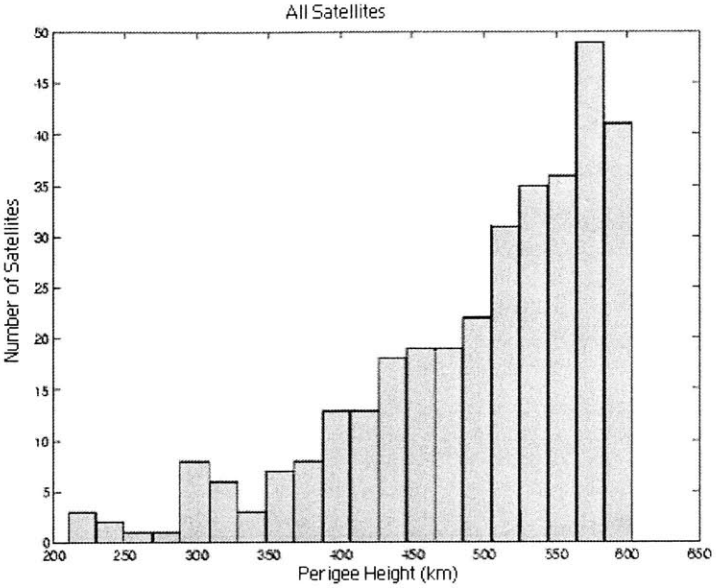


Figure 5-3: Distribution of Perigee Heights for All Satellites

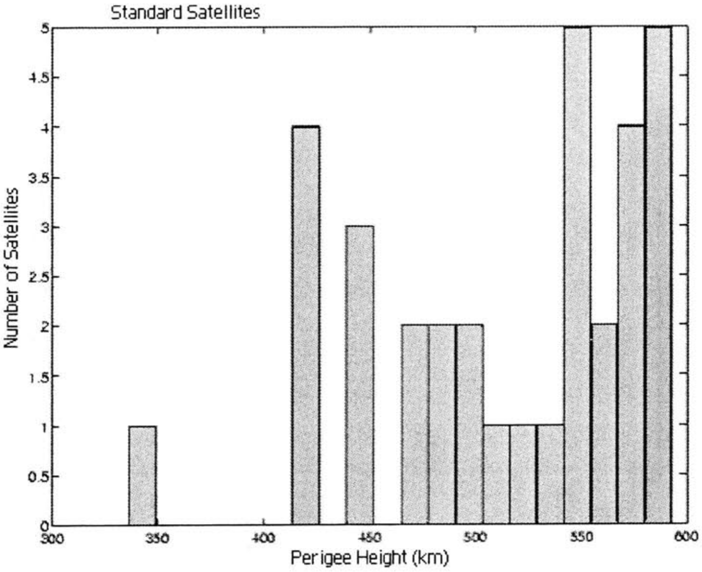


Figure 5-4: Distribution of Perigee Heights for the Standard Satellites used in BFE validation

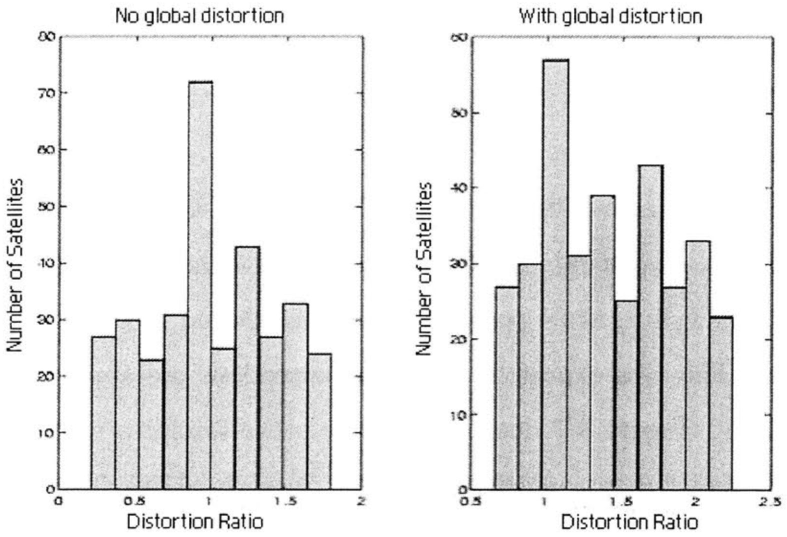


Figure 5-5: Ballistic Factor Distortion Ratios

5.2 Results

5.2.1 Data Flow

To ensure that the algorithm was operating properly, the ballistic factor updating routines were run on the ten-day set of results from Section 4.2.1, where no noise or mismodelling were included. Undistorted ballistic factors were used, and every tenth satellite was designated "standard". As expected, the "updated" ballistic factors were nearly identical to the original ones. Satellite #19764 was the only one that showed a large change in ballistic factor[2]. Satellite #19764 appeared to be simply a random outlier, since it had a sufficient number of observations (35), and the GTDS ρ_1 values, used as a convergence test, were similar to those for other satellites. It does have one of the smallest ballistic factors in the data set, which may have played a part in the errors. In general, absolute corrections to other satellites increased with increasing ballistic factor, while percent corrections showed no dependence on initial ballistic factor. This was expected, since the corrections are applied as ratios, not absolute amounts. Figure 5-7 does show a few other satellites with small ballistic factors yielding larger-than average corrections. Satellite #19764 was removed from the data set used for Figure 5-6 in order to make it easier to read the rest of the data[3].

5.2.2 Effects on Atmospheric Density Correction

The distorted ballistic factors did affect the resulting atmospheric density corrections, since the global bias in the ballistic factors translated into a bias (in the opposite direction) in density corrections. This can be seen in Figures 5-8, which compares the b_{1j} and b_{2j} values obtained with and without ballistic factor distortion. Both sets of b-values were calculated with observation noise and atmospheric mismodeling.

---

[2] Five satellites did not have enough observations to update their ballistic factors.

[3] Note that the figures and averages still include the standard satellites, adding some extra zeros.

Table 5.1: Statistics for Ballistic Factor "Improvements" with No Noise, No Mismodeling, No Initial Distortion

Q-factor coefficient a_1	9.7520e-08
Q-factor coefficient a_2	-7.0650e-08
Mean correction to a single satellite	-1.0263e-08
Mean correction omitting #19764	-7.2929e-11
Maximum correction to a single satellite	-3.4139e-06
This correction was applied to satellite 19764	
Second largest correction	9.4470e-09
This correction was applied to satellite 00179	

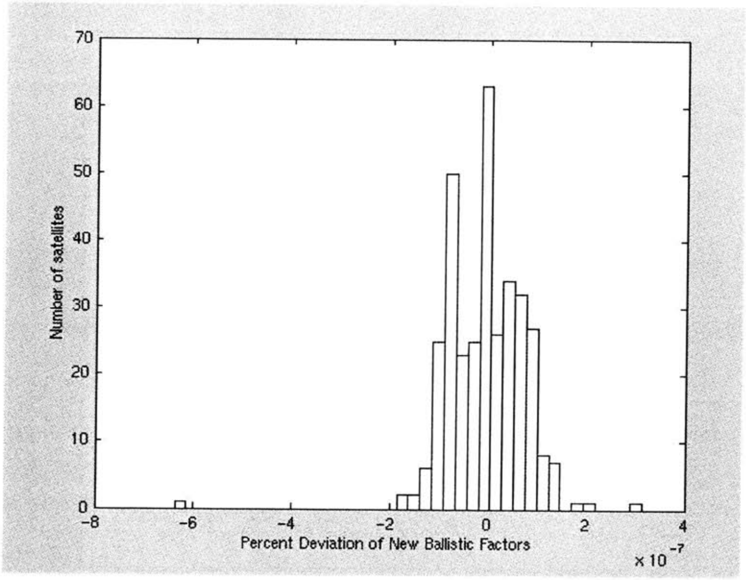


Figure 5-6: BF Percent Errors after Iteration, with No Noise, No Mismodeling, No Initial Distortion

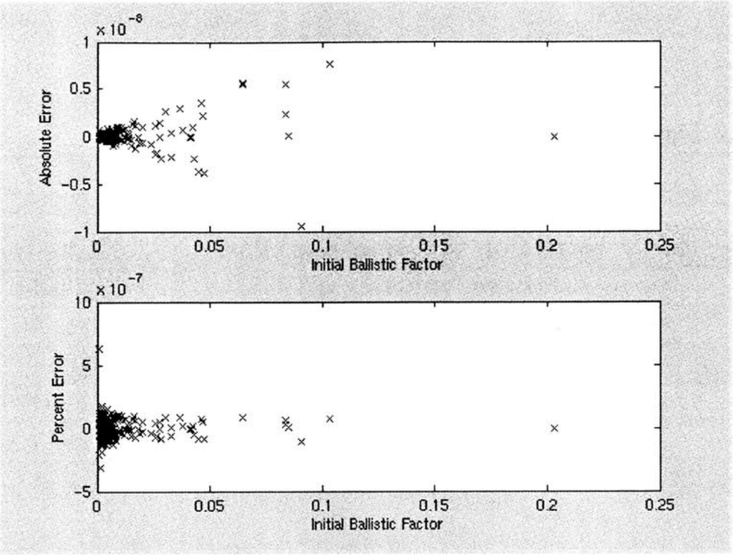


Figure 5-7: BF Errors Sorted by Initial BF, with No Noise, No Mismodelling, No Initial Distortion

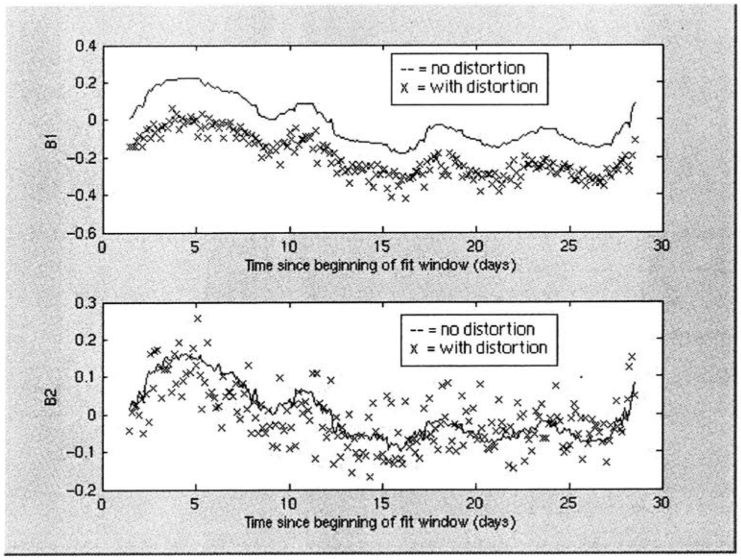


Figure 5-8: Comparison of B-values with and without Initial Ballistic Factor Distortion

5.2.3 Convergence

The ballistic factor updating cycle converged in all test cases. Some individual satellites, especially those with large initial distortions, did not have any converging GTDS DC runs, and thus no new ballistic factors were computed for them. These few outlying cases did not impede the convergence of the overall iteration[4] Figures 5-9 and 5-10 show the average absolute percent deviation (taken across all satellites[5]) for a five-day and a ten-day run using the distorted ballistic factors shown on the left side of Figure 5-5, which include global distortion. Similar results were obtained with different initial distortions. .The individual percent deviations for all of the satellites[6] in the ten-day case after 5 iterations are shown in Figures 5-11 and 5-12.

5.2.4 Update Cycle Length

Nazarenko and Yurasov found that a cycle length of 20+ days (about one solar rotation) was optimal[41], and this was confirmed by the new results, although any length over 10 days performed well. Figures 5-13 and 5-14 show the results for two cases[7]. Also, the longer test cases not only had lower final errors, but generally took fewer iterations to converge, which is an added benefit.

5.2.5 Height Dependence of Errors

Nazarenko and Yurasov showed results that indicated that the remaining ballistic factor error after iteration increased with altitude. This effect was not seen in these

---

[4] In a real-data processing case, satellites that repeatedly fail to yield converging results should probably be removed from the list used for atmospheric density correction.

[5] Standard satellites were included in these and all other statistics.

[6] Again, standard satellites are included, and can clearly be seen as the zeros in the graph. Satellites that did not have enough observed ballistic factors to update their ballistic factor are likewise still included, and account for the handful of large errors remaining after ten days.

[7] Example 1 in Figure 5-13 is the same example as used in Figure 5-9. Example 2 was included mostly to show a case where increasing update cycle length from 10 to 20 days provides further improvement, which was the case in several tests.

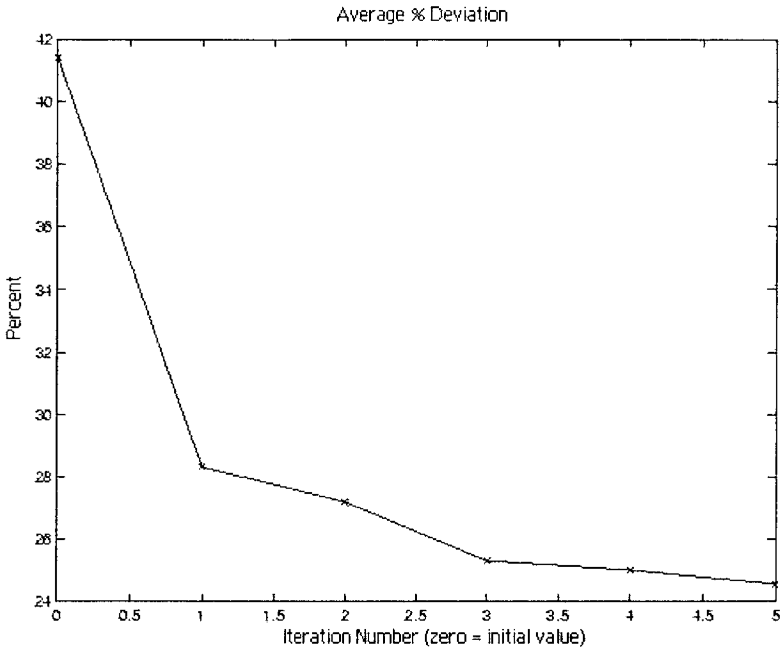


Figure 5-9: Convergence for five-day case

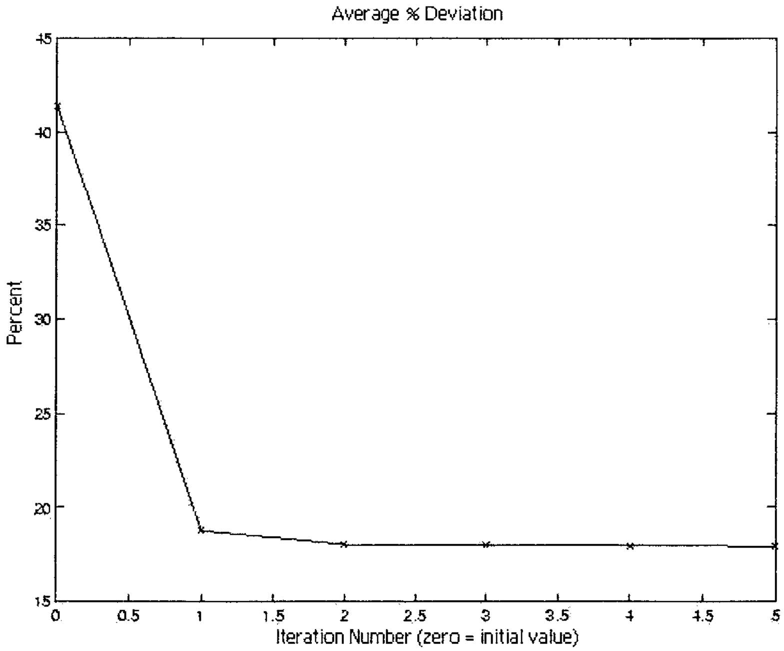


Figure 5-10: Convergence for ten-day case

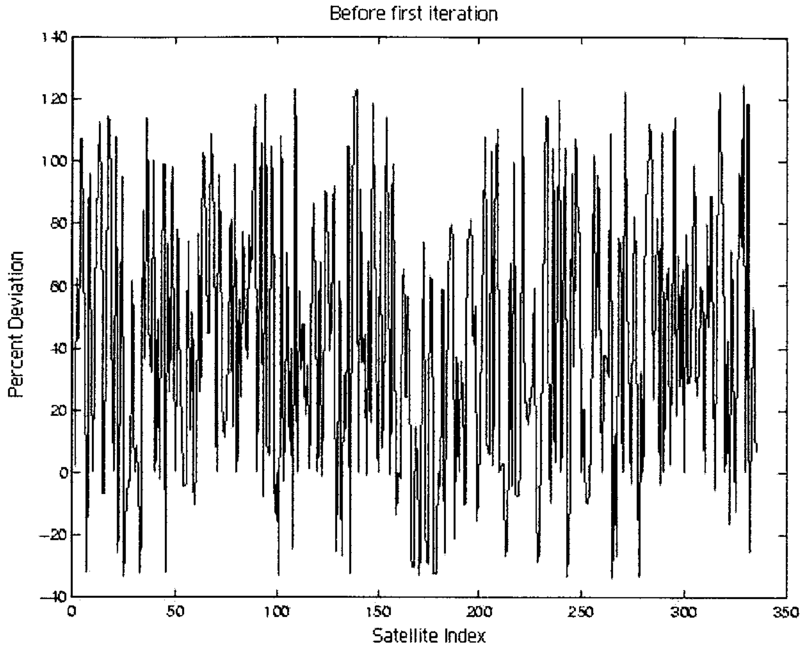


Figure 5-11: Percent Errors for all Satellites before BFE iteration

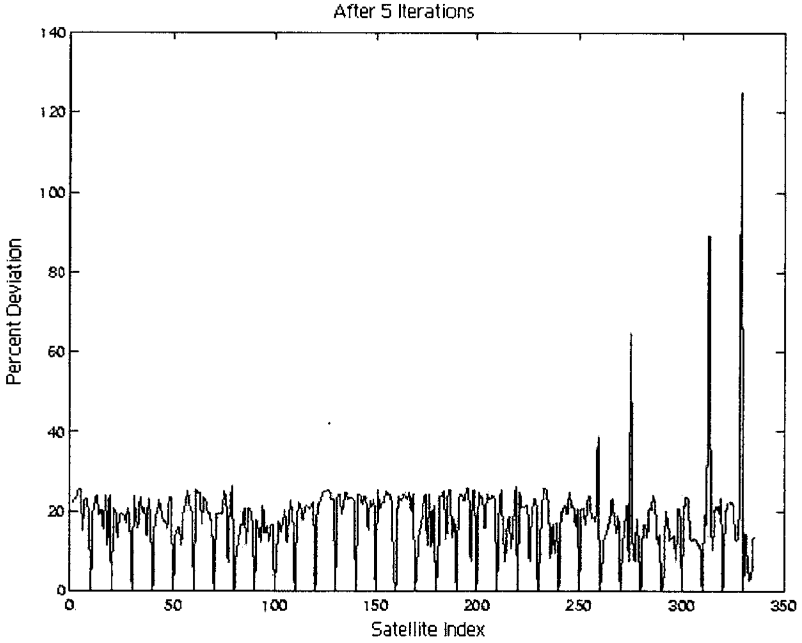


Figure 5-12: Percent Errors for all Satellites after 5 BFE iterations

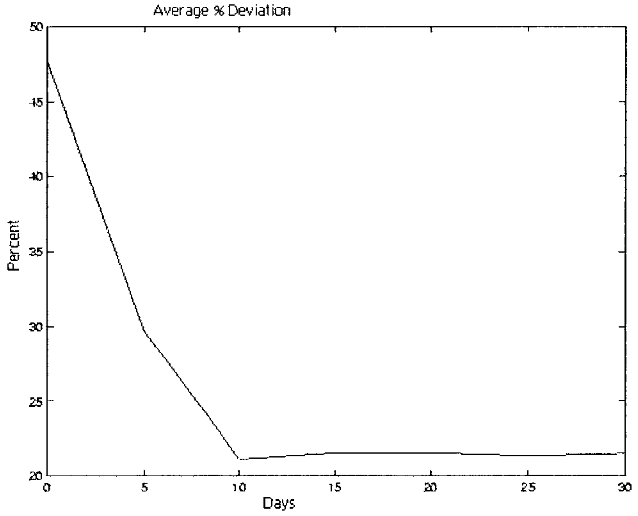


Figure 5-13: Average Absolute Percent Deviation as a Function of Iteration Period (example 1)

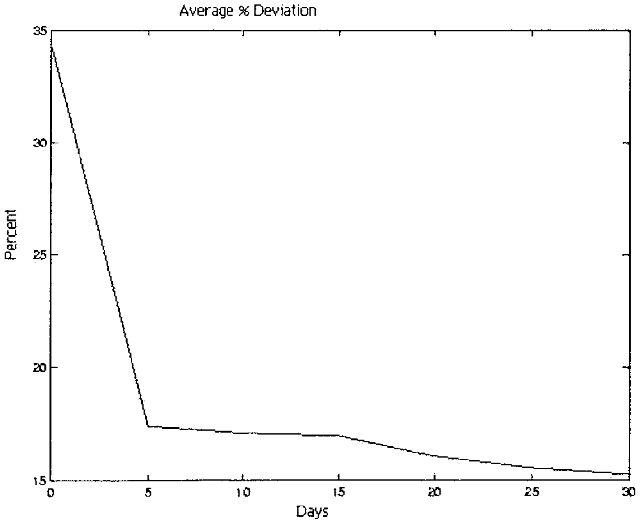


Figure 5-14: Average Absolute Percent Deviation as a Function of Iteration Period (example 2)

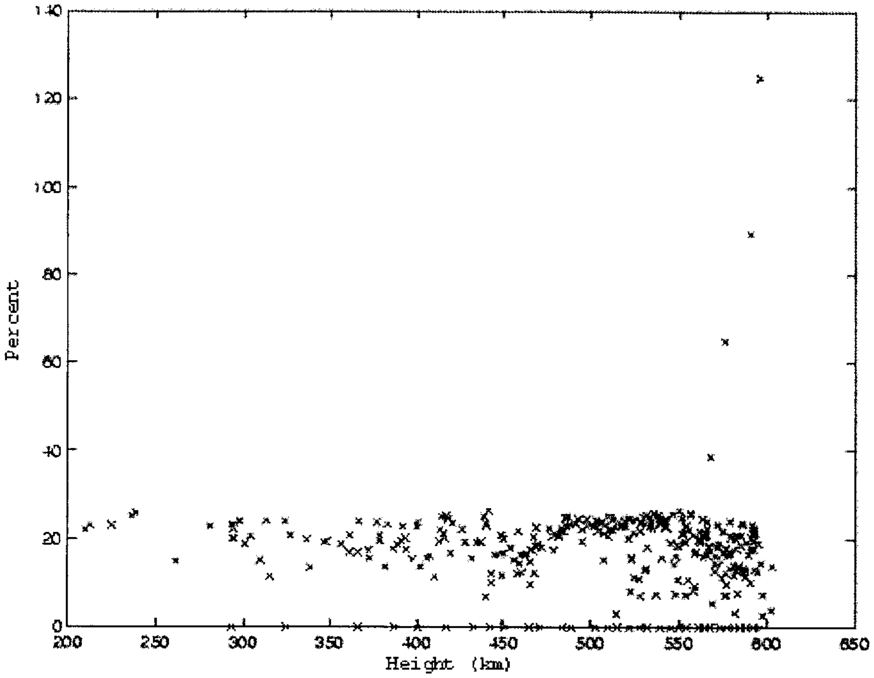


Figure 5-15: Remaining Errors After Iteration, Sorted by Perigee Height

test cases. Figure 5-15 shows the remaining ballistic factor errors from Figure 5-12 sorted by height[8]. Nazarenko and Yurasov's group of 214 test satellites included fewer high-altitude objects than the current set of 335 used here. The apparent removal of the height-dependence may be due to using more high-altitude objects, including more high-altitude standard satellites, or it may have been an artifact of the methods used to create their simulated observations.

---

[8] The standard satellites and the satellites that did not have enough observations are, again, included in this figure.

5.2.6 Global Distortion/Correction Effects

Substantially smaller final errors were seen when starting with ballistic factors that did not include a global distortion. When a global distortion was included, the ballistic factors converged, as a group, to a level of global distortion smaller than the initial distortion, but not non-zero. Figure 5-16 shows some results from a five-day run using the initial ballistic factors shown on the right side of Figure 5-5, which include only an individual distortion factor.

Unexpectedly, the same was true to a lesser extent of results including no initial global distortion (see Figures 5-17 and 5-18, which were created using the same five-day run as 5-16. Some form of global bias, which varies between data sets, is affecting the final results, and is not fixed by a global correction factor.

Since a global distortion factor seems unlikely to occur in real data, it stands to reason that BFE iteration results using real data will show better convergence than those using globally-distorted, simulated data, and that using the global correction factor does not substantially affect results with no global distortion. It remains to be determined why all runs show a final bias.

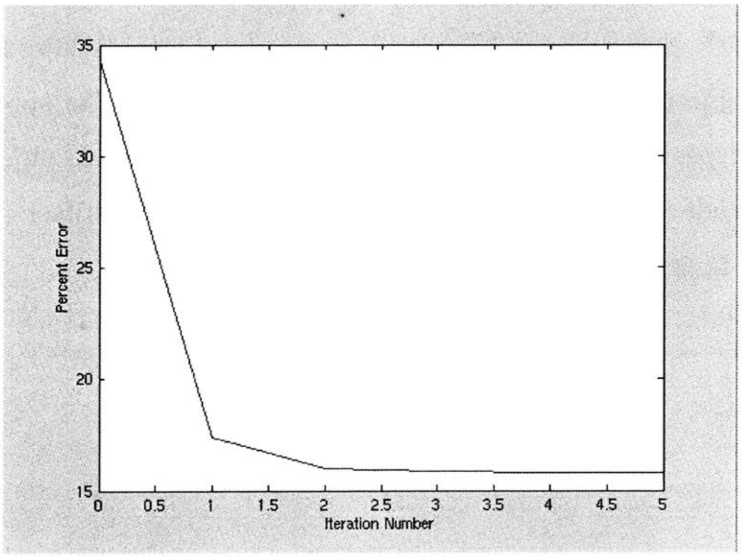


Figure 5-16: BFE iteration with No Global Distortion

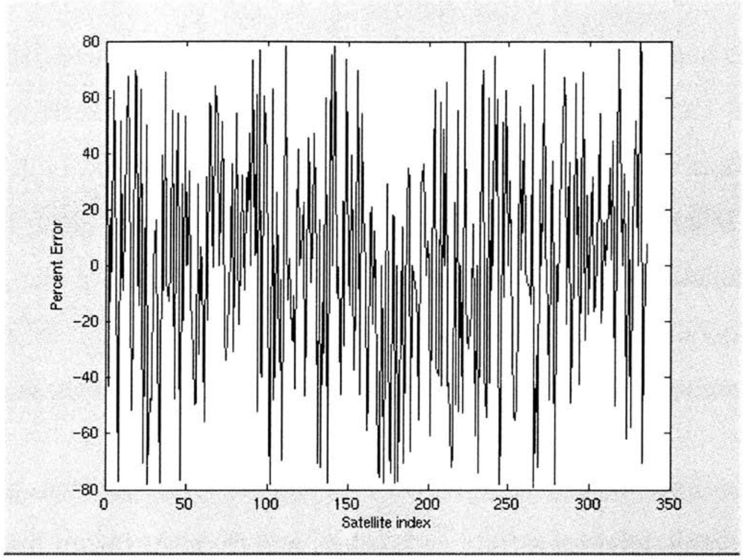


Figure 5-17: Percent Errors for all Satellites before BFE iteration, with no Global Distortion

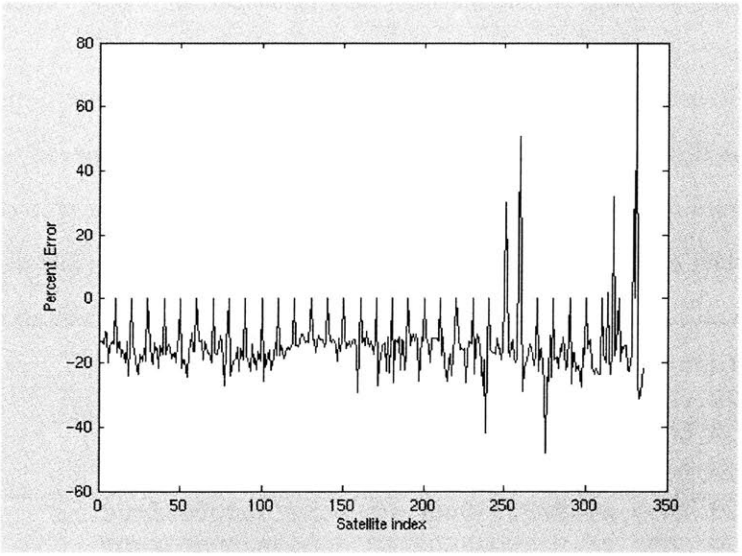


Figure 5-18: Percent Errors for all Satellites after 5 BFE iterations, with no Global Distortion

5.2.7 Omitting Recalculation of $\hat{k}_{ij}$ Values

The only difference when calculating $\hat{k}_{ij}$ values on the second or later iteration is that the initial guess (k_{ij}) has been altered, and for standard satellites, there is no difference. Unless the changes in k_{ij} are large, the new observed ballistic factor will be close to or identical to the one from the previous iteration. Since the calculation of observed ballistic factors is the most time-consuming part of the BFE iteration (consuming days on Pisces, while the calculation of b-values and improved ballistic factors takes minutes), it may be desirable to omit some or all of these recalculations.

The first ten lines of *ballfcts_1.txt.sort* and *ballfcts_2.txt.sort* from a five-day BFE run show only small differences from each other, and from the beginning of a run using an undistorted *initinfo.txt*, and this is true throughout the files. Satellite #01377 is the only standard satellite in the sample, and therefore shows no difference at all.

Thus, it follows that running just the section of the atmospheric density correction process contained in *calcvars.pl* on the first set of calculated observed ballistic factors should yield similar results as running both *estbfs.pl* and *calcvars.pl* in sequence. This appears to be the case, but no detailed testing was done on this subject.

from ballfcts_1.txt.sort...

```
00063 2451529.0000 2.1419918442E-03 5.4550800000E+02
00179 2451529.0000 9.1365130324E-02 5.6590800000E+02
00229 2451529.0000 5.0787569299E-03 5.7169200000E+02
00369 2451529.0000 2.3303299086E-03 5.5214400000E+02
00399 2451529.0000 2.1316206550E-03 6.0128400000E+02
00603 2451529.0000 1.0232360197E-02 4.3948200000E+02
00647 2451529.0000 9.3286234496E-03 5.5297800000E+02
00840 2451529.0000 5.3827785326E-03 5.3713000000E+02
00841 2451529.0000 5.8017187661E-03 5.3224500000E+02
01377 2451529.0000 3.8215995676E-03 4.8779800000E+02
⋮
```

from ballfcts_2.txt.sort...

```
00063 2451529.0000 2.1423138046E-03 5.4550800000E+02
00179 2451529.0000 9.3842153463E-02 5.6571900000E+02
00229 2451529.0000 5.0783218336E-03 5.7169200000E+02
00369 2451529.0000 2.3210805693E-03 5.5214500000E+02
00399 2451529.0000 2.1320345989E-03 6.0128300000E+02
00603 2451529.0000 9.8788911606E-03 4.3942200000E+02
00647 2451529.0000 9.3258156189E-03 5.5297900000E+02
00840 2451529.0000 5.3916855047E-03 5.3713000000E+02
00841 2451529.0000 5.8017115098E-03 5.3224500000E+02
01377 2451529.0000 3.8215995676E-03 4.8779800000E+02
⋮
```

from ballfcts.txt.sort using perfect initinfo.txt...

```
00063 2451529.0000 2.1419665216E-03 5.4550800000E+02
00179 2451529.0000 8.5986001561E-02 5.6637300000E+02
00229 2451529.0000 5.0779275783E-03 5.7169200000E+02
00369 2451529.0000 2.3176282674E-03 5.5214600000E+02
00399 2451529.0000 2.1304389253E-03 6.0128300000E+02
00603 2451529.0000 1.0223934090E-02 4.3951300000E+02
00647 2451529.0000 9.3276782062E-03 5.5297800000E+02
00840 2451529.0000 5.3952466937E-03 5.3713100000E+02
00841 2451529.0000 5.8016916534E-03 5.3224400000E+02
01377 2451529.0000 3.8215995676E-03 4.8779800000E+02
⋮
```

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 6

Real Data Validation

Validation with real data does not allow for the detailed comparisons seen in the prior two chapters, but is required in order to begin applying the algorithm and analyzing its performance on a large scale. This thesis includes only the first steps in the real data validation process. This limited effort simply to proves that AtmoCal will operate on real data (provided in NORAD B3 format) and will yield reasonable results for the interval examined. This allows for future

6.1 Data Preparation

The *initinfo.txt* files and the *a priori* state vectors are created using *TLE2osc.pl*, in the same manner as for simulated observations. At this point, standard and non-standard satellites should be chosen, and *initinfo.txt* modified accordingly. (Currently, this requires changing an N to an S for each standard satellite by hand, and selecting appropriate guesses for initial error variances.)

6.1.1 Conversion of B3 Observations

NORAD provides observations in B3 format[9], which is not compatible with GTDS or AtmoCal. Thus, the first step in real data processing is to convert the data to

OBSCARD format, identical to that produced by the *genobs.pl* script for simulated data. Joe Lombardo at CSDL wrote a conversion utility called "runadcob", which was later ported to FORTRAN 77 by Leo Early and modified by Jack Fischer and renamed "NORADPP". The details of NORADPP and some new modifications are described in Appendix E.1. The *b3conv.pl* Perl script uses this utility to convert a file containing B3 observations for many satellites into separate OBSCARD files for each satellite. (All of the required files for compiling and running the NORADPP, as well as *b3conv.pl* are included in the `utils/b3conv` directory of the AtmoCal CVS library.

The AtmoCal code automatically disregards observations for any satellite not included in *initinfo.txt*, so the files created by *b3conv.pl* do not need any further sorting. The following flowchart is the real data analogue of Figure 4-1:

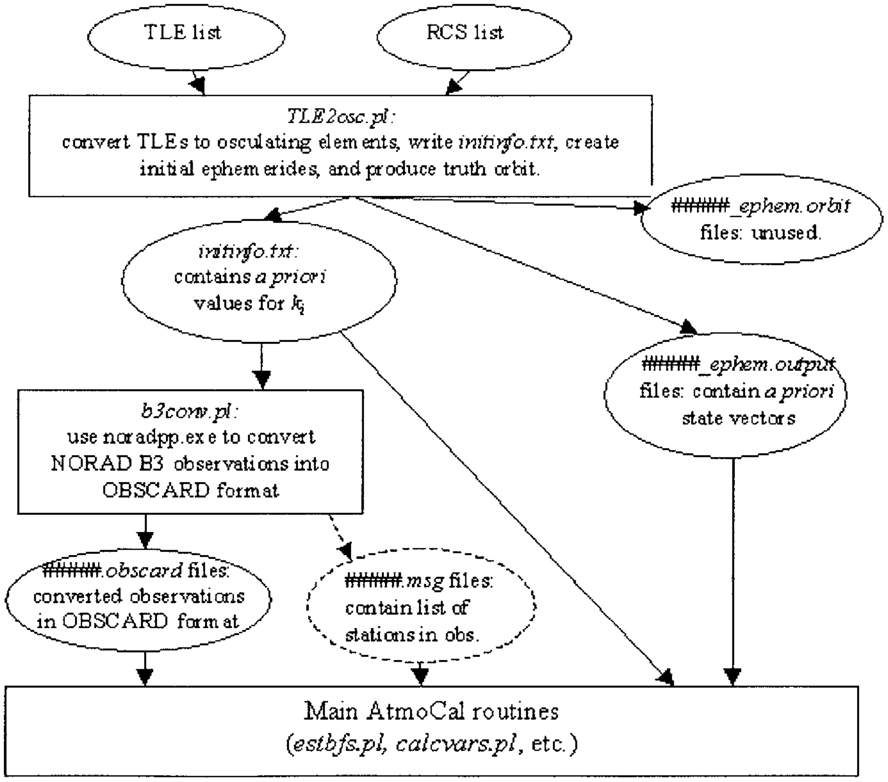


Figure 6-1: Flowchart for Real Observation Preparation

6.1.2 Station Data

Since the real data included more stations than those used in generating the simulated data, GTDS station cards (0 and 1) had to be added for several other stations. Those were generated[1] using data from Fischer's SLAD[9]. The *.msg* files created during the NORAD B3 to OBSCARD conversion list the stations used, and the individual station cards were created by hand. Automating this procedure would be helpful, but is not currently included in AtmoCal. Adding additional stations is the only change that should be made to AtmoCal (version 2.0, as reproduced in this thesis) before processing real data.

6.1.3 Observation Scheduling

Granholm included a section in *estbfs.pl* to determine if new observations were available during a given time interval for a particular satellite. This worked by looking at the GTDS DATASIM output files created when the simulated observations were generated. Since no such files were available for real data, this section is skipped when processing real data. This could somewhat increase the number of GTDS DC runs required, but decreases preparation time, since the times of the input observations do not need to be converted into a observation schedule.

6.1.4 Current Status

At this point, the GTDS DC runs required to calculate $\hat{k}_{ij}$ values run properly when called by the AtmoCal *estbfs.pl* routine. These DC runs had convergence measures (using the ρ_1 value) similar to those using noisy simulated data and a mismodeled atmosphere. Due to time constraints, more detailed results are not yet available. They can, however, now be produced, and the real data validation process can begin in earnest.

---

[1] For a copy of the SLAD, contact Dr. Paul Cefola at the MIT Lincoln Laboratory.

A sample GTDS input deck (automatically created by AtmoCal) and the output of *runestbfs.pl* are reproduced on the following few pages, demonstrating AtmoCal running on real observations. The results[2] shown are for satellite NSSC #00063, since it happened to be the first on the list of 335.

---

[2] User options were set to choose the appropriate directories on Pisces, $simulated = 0, and the particular satellite was selected by deleting the others from *initinfo.txt*.

Input Deck for the first GTDS run on #00063 using real data

```
CONTROL   DC                                                60016A       00063
EPOCH             991215               000
ELEMENT1  1  1  1 3512.468417          5274.365230         2843.015660
ELEMENT2          -3.437224880         4.810807515         -4.736111427
ORBTYPE   2  1  1 60.
OBSINPUT  5        991215000000.0000    991218000000.0000
DMOPT
/FLYF   1 0346  3   388.900             541242.8299        3591947.6900
/PPWQ   1 0388  3    82.780             390809.3764        2383857.3529
/PPWF   1 0389  3    82.780             390810.1652        2383856.8705
/EGLQ   1 0399  3     0.380             303420.7790        2734706.5526
/NAVQ   1 0745  3   305.300             333314.3388        2611413.5272
END
DCOPT
/FLYF   0 1  4  5    35.0                54.0               54.0
/PPWQ   0 1  4  5    40.0                36.0               36.0
/PPWF   0 1  4  5    40.0                36.0               36.0
/EGLQ   0 1  4  5    30.0                45.0               45.0
/NAVQ   0 1  4  5  1979.0                64.8              122.4
ELLMODEL  1        6378.135            298.26
/FLYF   200001
/PPWQ   200001
/PPWF   200001
/EGLQ   200001
/NAVQ   200001
TRACKELV  3        5.0
EDIT               3.0
PRINTOUT  1     1
CONVERG  25  6     1.0D-4
END
OGOPT
DRAG      1        1
ATMOSDEN        1
DRAGPAR   3  0    2.2
DRAGPAR   1
SCPARAM           1.1300000000E-06     630.745847292649
MAXDEGEQ  1        4
MAXORDEQ  1        4
MAXDEGVE  1        4
MAXORDVE  1        4
POTFIELD  1  4
SOLRAD    1        1.0
END
FIN
CONTROL   EPHEM                         OUTPUT              60016A       00063
OUTPUT    1  2  1 991216               000000.0             10800.0
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN        1
```

```
DRAG        1         1
DRAGPAR     3  0     2.2
SCPARAM              1.1300000000E-06     630.745847292649
POTFIELD    1  4
MAXDEGEQ    1         4.0
MAXORDEQ    1         4.0
SOLRAD      1         1.0
END
FIN
CONTROL     EPHEM                          OUTPUT             60016A        00063
OUTPUT      1  2   1 991226               000000.0            86400.0
ORBTYPE     2  1   1  60.0
OGOPT
ATMOSDEN           1
DRAG        1         1
DRAGPAR     3  0     2.2
SCPARAM              1.1300000000E-06     630.745847292649
POTFIELD    1  4
MAXDEGEQ    1         4.0
MAXORDEQ    1         4.0
SOLRAD      1         1.0
END
FIN
```

Output of *runestbfs.pl* for the first GTDS run on #00063 using real data

```
pisces 291% runestbfs.pl
--------------------------------------------------
--------------------------------------------------
        estbfs.pl: Processing /AtmoDenTrk/atm_cal/realdata/initinfo.txt
        Process # 1
--------------------------------------------------
        Job started at 17:37:28 EST 5/22/102
--------------------------------------------------
----------------------------------------
  Processing NORAD Catalog #00063
  Process # 1
  Run number 1
  Epoch 991215 000000.0000
----------------------------------------

UNIX-GTDS
Charles Stark Draper Laboratory

Run started at: 17:37:28 EST 5/22/102
Run ended at: 17:37:33 EST 5/22/102
Run converged with rho1 = 0.35230740e+01
.
----------------------------------------
  Processing NORAD Catalog #00063
  Process # 1
  Run number 2
  Epoch 991215 030000.0000
----------------------------------------

UNIX-GTDS
Charles Stark Draper Laboratory

Run started at: 17:37:33 EST 5/22/102
Run ended at: 17:37:37 EST 5/22/102
Run converged with rho1 = 0.35230738e+01
.

⋮
```

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The AtmoCal code now contains all the functionality of Nazarenko and Yurasov's initial implementation. All sections of both the code and the algorithm itself have now been validated using simulated data. All of the files employed in AtmoCal and gtds_granholm have been placed under configuration management, and are ready for further development. The ballistic factor estimation appears to converge even in extremely perturbed circumstances, although large speed and accuracy improvements using only small refinements may be possible. The main atmospheric density correction routines have also been proven to operate properly on a small piece of real data, although time constraints prevented a full-scale test on real data.

7.2 Future Work

Now that AtmoCal is operational, there are three major areas for future work. First, the performance of the current version of the AtmoCal algorithm should be further investigated, with both real and simulated data. Second, there are a number of minor improvements to gtds_granholm and AtmoCal that would improve speed and useability. Finally, there are a variety of major additions that are desireable, ranging from adding corrections for other atmospheric density models, including different correction models (quadratic, etc. in place of linear), to integrating the results of direct atmospheric observation.

7.2.1 Further Tests

Comprehensive Performance Analysis Granholm chose a representative group of satellites and examined the improvements gained in orbit determination and prediction for that group. These tests, or some like them, should be repeated on all 335 satellites used in the simulated data test, as well as examining the effects on orbit determination and prediction for satellites not included in the initial list. (This could be done simply by re-running the correction process for 300 of the satellites, and then examining those results as applied to the other 35.) Statistical measures for considering the effects on all 335 satellites should be developed to give an "average" performance index for this or any other atmospheric density correction algorithm[1, 45].

The Satellite List George Granholm showed that the atmospheric density corrections found using only 214 satellites were nearly identical to those found using 335 satellites (see page 101 of his thesis[13]). This was done to show consistency with Nazarenko's results, since Nazarenko worked with a list of 214 satellites[41]. A comprehensive study of the effects of varying the number of satellites and the number of standard satellites would help future users choose which satellites to include and which to list as standard.

Data Density Nazarenko chose 3-hour atmospheric density correction spans based on the number of raw observations that were available. If that number were increased, shorter spans might provide more accurate corrections. And, if observations were scarce for a particular time period, it would be useful to know how long of a correction span is feasible.

Real Data Performance Analysis Working with real data does not lend itself as easily to performance analysis, since there is no "truth" model for comparison. Nevertheless, real data analysis is important for showing that an algorithm is not just a theoretical fancy. Any performance analyses (including the ones performed in this thesis and in Granholm's thesis) conducted using simulated data should be repeated with real data. The BFE iteration should be run on real data, possibly using initial ballistic factors distorted in the same manner as those used with simulated data to facilitate comparison.

7.2.2 GTDS Bugs

Granholm found several bugs in the GTDS implementation (gtds_granholm) used with AtmoCal. These bugs have not yet been addressed, since workarounds were included in AtmoCal. The most important one, which should be addressed first, is the random-number generation bug. Details are included in C.4. The other major problem is that the PR-5 version of GTDS is not Y2K compliant. There have been efforts to make other versions of GTDS Y2K compliant, most notably Chris Sabol's work on the CSDL PC version of GTDS, but those changes have not been merged into the gtds_granholm source tree. This correction would also require changes in all of the AtmoCal routines, since AtmoCal compensates for this deficiency in GTDS. (Since the available real data stretched from December 1999 to February 2000, Y2K issues were prevalent.)

7.2.3 New GTDS Features

There has been interest in applying this type of atmospheric density correction to other density models, especially MSISE-90[17] or the new NRLMSISE-2000 [44]. Also, to facilitate further comparison with Nazarenko and Yurasov's results[41], the GOST model would be a desirable addition[12]. Jacchia-70 may also be a useful addition, since this model is still in use by the USAF. Jacchia-70 has been partially implemented in gtds_granholm, but some features required for AtmoCal have not been added. NRLMSISE-2000, GOST, and other desired models would first need to be added to gtds_granholm, and then any model would need to be fully tested before they could be used with AtmoCal. (Jack Fischer's MSISE-90 implementation is already available in the current gtds_granholm version, and thus would be easier to include in future AtmoCal releases.)

7.2.4 GTDS Integration

Currently, several versions of Research and Development (R&D) GTDS exist, the SGI version of gtds_granholm among them, as well as the IBM-PC version and the VAX version. Various improvements have been included in some, but not others. Features not included currently in gtds_granholm include Scott Carter's work to include the 50x50 Geopotential, J2000 coordinate system, and solid Earth tide models, as well as Chris Sabol's aforementioned Y2K fixes. Both these, and other new features, should be incorporated into all R&D GTDS versions. A re-integration of the variant GTDS development trees would be beneficial to users of all versions.

7.2.5 AtmoCal Refinements

Several minor refinements to the AtmoCal code would make it more portable and accessible. There is no user interface to speak of -- options are set by directly modifying the Perl and MATLAB code. (All of the options have been moved to a marked block at the beginning of each program to make them more visible.) Eventually, in order to provide a useful atmospheric density correction service, this code should be put into a format where it can be run with one or two commands, a single set of options, and an easy-to-use, standardized output format.

More analysis tools would also be useful, as would more conversion utilities for other types of data. Currently, AtmoCal requires input in the form of an initial set of TLEs and RCS values, and observations in NORAD B3 or GTDS OBSCARD format.

7.2.6 Major AtmoCal Additions/Changes

The following list of possible AtmoCal feature additions includes some taken from suggestions and questions made by numerous participants in the Quebec City (August 2001) and San Antonio (January 2002) AAS/AIAA conferences, as well as fellow LL group 98 personnel.

Many of these additions are intended to address any possible statistical biases created by the data processing methods currently employed. Before any of these methods are chosen, an in-depth study of the possible statistical problems in the mathematics behind AtmoCal should be performed. The method has been shown, in its present form, to provide improvements, but greater accuracy may be possible with simple changes based on a deeper statistical understanding.

Data Types Currently, AtmoCal processes only raw satellite observations, and does not distinguish between observations from various types of observation platforms. Several experiments have been planned, including CHAMP and GRACE, which will take direct measurements of atmospheric density[36]. AtmoCal was designed not to need direct atmospheric measurements or “calibration satellites”, but it would be foolish not to use all available data. Any “calibration satellite” projects can already be included, since they could be listed as standard satellites with extremely well-known ballistic factors. Direct accelerometer measurements could not be included in quite this fashion, but it seems reasonable to assume that they could be converted into a form compatible with the current weighted least-squares fit. A relationship akin to Equation 2.6 should be derived, and then an expression for δ_{ij} could be found. The appropriate weights for accelerometer data in the least-squares fit would likely be much higher than those for normal satellite observational data, but this could be determined empirically.

Data Density and Quality As described in Section 2.2, all of the $\hat{k}_{ij}$ values are obtained by using equal-length, overlapping spans (nominally three day fit spans, offset by three hours). This results in a similar data density for each satellite in the catalog, since a particular satellite only lacks an observation for any time span where the GTDS DC run did not converge. However, individual spans for specific satellites with higher data densities may have more accurate $\hat{k}_{ij}$ estimates, and perhaps should be weighted accordingly. Also, some stations provide much more accurate measurements than others, and this may also affect the accuracy of individual $\hat{k}_{ij}$ values. (Adding the capability to weight individual spans independently of satellite could be done hand-in-hand with adding support for weighting results by the accuracy of the measurement type.)

Ballistic Factor Update Cycle The ballistic factor update cycle is currently the slowest part of the algorithm, and much of this may be caused by unnecessary re-processing of data. See Sections 5.2.7 for more details. A systematic study of the effects of using the global correction factor also should be made, as well as a study of the effects, if any, of changing the percentage of standard satellites.

Recursive Fit Methods Granholm supported adding a recursive fit method as an alternative the three-hour fit windows. This method involves first obtaining valid fits for each satellite at the beginning of the observation interval, but this only needs to be done once. These initial fits need not be three days, but are simply as long as is necessary to get a good fit for all of the satellites. (If a particular satellite takes much longer to converge, or does not converge at all, it should be removed from the list used.) Then, each new observation is processed and yields a ballistic factor estimate. This method does mean that satellites with higher data densities will figure more prominently in the final corrections data, this may add a statistical bias that would affect both the correction process and the ballistic factor update process, especially if the satellite in question had an inaccurate ballistic factor. In general, the entire operation of AtmoCal could be altered to run recursively, processing each new observation and the resultant $\hat{k}_{ij}$, b_{1j} , and b_{2j} values as they occur.

New Correction Models The linear correction model was chosen in order to avoid extracting too much information from the data. It is not yet certain how close a linear fit comes to an "optimal extraction" which provides the most accurate corrections possible, while still being robust.

Automatic Adjustments All of the fit options, like correction span and fit span are currently chosen once by the user. Automatic fit lengthening for satellites with little data or ones that previously did not converge would decrease the number of divergent GTDS DC runs, speeding up the process and providing more accurate corrections. There is also the possibility of automatically choosing the standard satellites (as well as flagging satellites that are so non-standard that they should be removed from the list used).

Converting between Density Models Currently, only one atmospheric density model (JR-71) is supported, and corrections calculated for that model are not applicable to any other model. There is the possibility of devising a method to convert corrections between models. An initial step might be by analogy to the methods used to convert orbits from SGP4 to special perturbations[22], where orbits are propagated forward and backwards.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Key to Symbols, Abbreviations, Etc.

A.1 Text Conventions

Entire programs, programming languages, and CVS distributions are referred to using regular roman text. Examples include AtmoCal, GTDS, MATLAB, CVS, gtds_granholm.

Specific filenames and directory structures are usually written in italics, like this: *filename.ext.*

A typewriter font is used for variable names, brief code quotes, and computer instructions. Variables will usually appear as `$variablename` (for a normal perl variable), `%variablename` (for an array in perl), or `VARIABLENAME` (for a FORTRAN variable). Computer instructions will include a quoted prompt (`prompt%`), which, obviously, should not be typed when entering the instructions.

To agree with conventions used elsewhere, GTDS card decks are referred to using capital letters. (For example, DCOPT is the differential corrections option subdeck.

Mathematical scalar variables are normally denoted by italics, and matrices by capital, bold roman letters.

A.2 Expressions and Abbreviations

This list is intended to be exhaustive. All expressions and abbreviations are either defined where first used or should be obvious in meaning, but this list is included for reference. They are listed below in alphabetical order by english letter, followed by alphabetical order by greek letter.

b_{1j} is the ideal value of the bias coefficient of the linear correction factor at time j. See Equation 2.1.

b_{2j} is the ideal value of the slope coefficient of the linear correction factor at time j. See Equation 2.1.

$\hat{b}_{1j}$ is the calculated value of the bias coefficient of the linear correction factor at time j. See Equation 2.14.

$\hat{b}_{2j}$ is the calculated value of the bias coefficient of the linear correction factor at time j. See Equation 2.14.

$\mathrm{F}_{10.7}$ is the 10.7 cm flux, used as a proxy for EUV radiation.

$\overline{\mathrm{F}}_{10.7}$ is the average value of $\mathrm{F}_{10.7}$.

FRN is the FORTRAN reference number used to refer to a particular data file by GTDS.

Gregorian Date The Gregorian calendar is the one used by most people in the U.S.A. GTDS uses the Gregorian date in some places, although usually the year field is replaced by $year - 1900$. This results in references to "year 100" when working with data from 2000 CE.

i is an index denoting a particular satellite in the catalog. (AtmoCal normally sorts satellites by NSSC number, but that is an arbitrary choice.)

j is an index denoting a particular time.

JR-71 is the atmospheric density model created in 1971 by Charles Roberts [48], based on Luigi Jacchia's 1970 atmospheric density model [24].

Julian Date The Julian dating system gives a single floating point number for any particular date and time after noon (Universal Time) on January 1, 4713 BCE. This dating system is convenient, since it does not have months with irregular days, leap years, etc., and hours, minutes, and seconds are converted into fractions of days.

k_i is the true ballistic factor for satellite i.

$\bar{k}_i$ is the approximate, tabulated value of the ballistic factor for satellite i. (Normally, this is obtained from some sort of time-average.)

k_n is the true ballistic factor for non-standard satellite n.

$\bar{k}_n$ is the approximate, tabulated value of the ballistic factor for non-standard satellite n. (Initial values are normally obtained from some sort of time-average, and are updated as detailed in Section 2.5.)

k_s is the true ballistic factor for standard satellite s.

$\bar{k}_s$ is the approximate, tabulated value of the ballistic factor for satellite s. (Normally, this is found by some sort of time-average, and is expected to be quite accurate for all standard satellites.)

$\hat{k}_{ij}$ is the observed ballistic factor for satellite i at time t_j.

Modified Julian Date GTDS uses a modified version of the Julian date system. The $MJD = JD - 2430000$, which corresponds to a reference date "0.0" at noon on January 5, 1941. These MJDs are used in creating the GTDS$075 binary file. Other modified Julian dates are used by various people and programs, in order to shorten the number of digits required for storing recent dates.

MSIS is one of a series of atmospheric density models based on work by A. Hedin. The initials stand for "Mass Spectrometer and Incoherent Scatter", which were the types of data used to create the model. [19]. The later MSISE and NRLMSISE models are also often referred to simply as MSIS models.

MSISE The MSISE models are "Extended" MSIS models, which include the lower and middle thermosphere. [17]

n is an index denoting a particular non-standard satellite in the catalog.

N_n is the set of timespans that contain observations of non-standard satellite n.

$|N_n|$ is the number of timespans that contain observations of non-standard satellite n.

NRLMSISE is the name of the Naval Research Laboratory's new version of A. Hedin's Extended MSIS model. [44]

N_s is the set of timespans that contain observations of standard satellite s.

$|N_s|$ is the number of timespans that contain observations of standard satellite s.

range/az/el stands for "range, azimuth, and elevation", which is a common format for satellite observations.

s is an index denoting a particular standard satellite in the catalog.

t_j is time interval j.

$\delta\rho$ is the difference between the model atmospheric density and the true atmospheric density.

Δ_{ij} is the difference between the actual and observed ballistic factor for satellite i at time j. See Equation 2.7.

ρ is the true atmospheric density.

ρ_m is the model atmospheric density.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Implementation Miscellanea

B.1 CVS and Revision Control

The Concurrent Version System (CVS) was used for configuration management and revision control on both gtds_granholm and AtmoCal[46]. CVS is the most widely used package for revision control on a UNIX/LINUX platform, and is freely available. CVS operates by maintaining a repository of all of the code for a given project, and allowing users to “check out” a copy for individual use and alteration. When a user feels that their changes should be included in the repository, they can ”check in” the altered code, which is then available to everyone on the project. Copies of older versions are always kept, and each new ”check in” increases the version number for an individual file, so that versions are easily identifiable. A log is also kept of all changes. This enables multiple people working on the same code to ensure that one person’s changes do not interfere with someone else’s work. Version management is also important just for tracking changes (and recovering from accidental mistakes).

To retrieve the versions of gtds_granholm and AtmoCal used to produce the results in this thesis, on the Pisces machine, execute the following commands[1]:

```
prompt% cvs checkout gtds_granholm
prompt% cvs checkout AtmoCal
```

Some common CVS commands are listed below, although a manual (like the CVS Pocket Reference [47]) is recommended for doing anything more complicated than checking out and running a copy of gtds_granholm and AtmoCal[46].

Table B.1: Common CVS commands

Command Syntax	Function.	List of Important Flags
`cvs add` *file*	Adds a new file or directory to a repository	-m "message describing new file/directory"
`cvs commit` *file*	Commits changes to a repository	-m "message describing changes" -r *[revision number]*
`cvs history` *file(s)*	Shows revision history for file	
`cvs remove` *file*	Remove a file from the repository	-f (deletes file before removing)
`cvs update` *files(s)*	Update files from repository that were changed by other users	-j *[revision number]* will merge files. Use with caution.

B.2 File Names and Locations

A graphical representation of the three file structures required for running AtmoCal and gtds_granholm are shown in Figures B.2, B.2, and B.2.

---

[1] AtmoCal must be checked out in the directory specified by the `$ATM_CAL` environment variable, but gtds_granholm may be checked out elsewhere, provided that the `$GTDS_DIR` environment variable is set properly.

Figure B-1: File Structure for Large Data Files

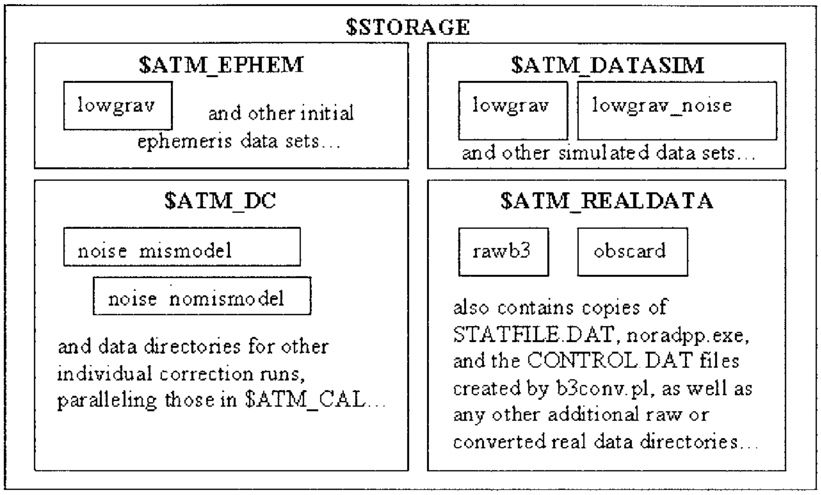


Figure B-2: File Structure for AtmoCal and Small Data Files

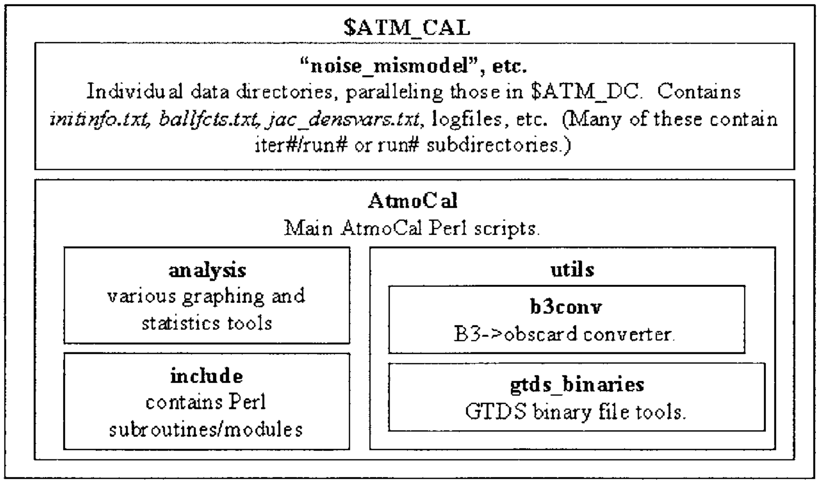
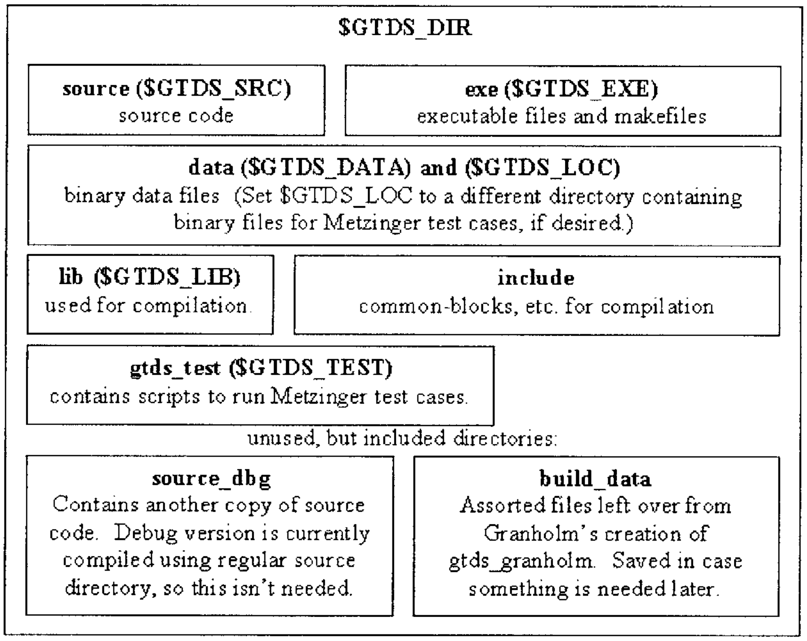


Figure B-3: File Structure for gtds_granholm



$GTDS_DIR

source ($GTDS_SRC)
source code

exe ($GTDS_EXE)
executable files and makefiles

data ($GTDS_DATA) and ($GTDS_LOC)
binary data files (Set $GTDS_LOC to a different directory containing binary files for Metzinger test cases, if desired.)

lib ($GTDS_LIB)
used for compilation.

include
common-blocks, etc. for compilation

gtds_test ($GTDS_TEST)
contains scripts to run Metzinger test cases.

unused, but included directories:

source_dbg
Contains another copy of source code. Debug version is currently compiled using regular source directory, so this isn't needed.

build_data
Assorted files left over from Granholm's creation of gtds_granholm. Saved in case something is needed later.

B.3 Environment Variables

The following list of environment variables are required for using AtmoCal. These environment variables inform AtmoCal about the locations of various sections of the directory structures shown in the previous section. The directory specified by `$ATM_CAL` is the one in which a CVS checkout of AtmoCal should be performed, and gtds_granholm can be checked out there as well, or elsewhere. The environment variable `$GTDS_DIR` must be set to the directory created by the check-out process. The directories for large data storage (`$STORAGE` and its subdirectories), should be created by hand before running AtmoCal, since some are not automatically created. While some of the following variables are redundant, reducing the number of environment variables was not a priority, and both copies of a redundant variable are required by different scripts.

If a user is using *csh* or *tcsh* for their login shell, the following list is formatted so that it may be cut-and-pasted directly into their *.cshrc* file. Users of *bash* or another shell should consult their system administrator for help on setting environment variables. The `$GTDS_DIR, $STORAGE,` and `$ATM_CAL` directories should obviously be changed, and if the implementation is not on Pisces, the two CVS variables should be changed to the appropriate values.

Table B.2: Environment Variable List

```
setenv CVSROOT  /lccroot/repository # location of CVS repository******PISCES-SPECIFIC
setenv CVSEDITOR 'emacs -nw' # choose an editor for CVS          ******PISCES-SPECIFIC
setenv GTDS_DIR /AtmoDenTrk/gtds_granholm # for a variety of things
setenv GTDS_SRC $GTDS_DIR/source # used by gtds_granholm makefile
setenv GTDS_LIB $GTDS_DIR/lib  # also used by gtds_granholm makefile
setenv GTDS_DATA $GTDS_DIR/data # GTDS binary data files, for atm_cal
setenv GTDS_LOC $GTDS_DIR/data # GTDS binary data files, for running Metzinger tests
setenv GTDS_EXE $GTDS_DIR/exe # location of GTDS executable
```

continued on next page...

...continued from previous page

```
setenv GTDS_TEST $GTDS_DIR/gtds_test # location of scripts for running Metzinger tests
setenv STORAGE /AtmoDenTrk/bergstrom  # large file storage
setenv ATM_EPHEM $STORAGE/ephem_runs # storage for GTDS EPHEM runs
setenv ATM_DATASIM $STORAGE/datasim_runs # storage for GTDS DATASIM runs
setenv ATM_DATA $STORAGE/realdata # location of real data
setenv ATM_DC $STORAGE/dc_runs # storage for GTDS DC runs (these can be quite large)
setenv ATM_CAL /AtmoDenTrk/atm_cal # AtmoCal should be in $ATM_CAL/AtmoCal
```

Appendix C

GTDS

C.1 GTDS Changes

No changes to the GTDS subroutines that apply the atmospheric density correction factors have been made to gtds_granholm since George Granholm created it. The code has been put under revision control, and the included *makefile*[1] has been revised because of differences between the Pisces machine at LL and the DC1 machine at CSDL. Shell scripts to run each of the Metzinger test cases were added. Finally, Paul Cefola and Zach Folcik have added options that allow SGI GTDS to input and output quasi-inertial (USM compatible) and J2000 coordinates.

All of the changes made by Granholm, as well as those made by Cefola and Folcik, to the original PR-5 version of GTDS are listed in the following table, which is based on a nearly-identical section on pages 56–59 of his thesis, and the information is reproduced here for completeness. The routines can each be found in the source code file *[routine].for*.

---

[1] A makefile is a text file that contains instructions on how the source code should be compiled into an executable. If a properly-written makefile exists, a user need only type `make all` at the prompt in order to compile the code, or `make all -f` *[filename]* if the name of the makefile is not *makefile* or *Makefile*. This is relevant to gtds_granholm, since makefiles for both the optimized (*Makefile*) and debug (*Makefile_dbg*) versions are included in the CVS distribution.

Table C.1: GTDS Code Alteration List

<table>
<tr><th colspan="3">Routines directly used in atmospheric correction</th></tr>
<tr><th>Routine</th><th>Description</th><th>Change</th></tr>
<tr><td>ATMCALJACBD</td><td>Initializes variables in common block /ATMCALJAC/</td><td>added routine</td></tr>
<tr><td>BARODE</td><td>Calculates JR-71 density in 90-100 kilometer range</td><td>added call to CALCCALJAC</td></tr>
<tr><td>CALCCALJAC</td><td>Main atmospheric density correction routine</td><td>added routine</td></tr>
<tr><td>DIFFDE</td><td>Calculates JR-71 density in 100-125 kilometer range</td><td>added call to CALCCALJAC</td></tr>
<tr><td>FILESBD</td><td>Defines FRN for I/O files</td><td>added FRN 106 for JR-71 corrections file</td></tr>
<tr><td>HIALT</td><td>Calculates JR-71 density above 125 kilometers</td><td>added call to CALCCALJAC</td></tr>
<tr><td>INITCALJAC</td><td>Reads in JR-71 corrections file (FRN 106)</td><td>added routine</td></tr>
<tr><td>JACROB</td><td>Driver routine for JR-71 model</td><td>added call to INITCALJAC</td></tr>
<tr><td>SETDAF</td><td>Opens files</td><td>added opening FRN 106</td></tr>
<tr><td>SETOG1</td><td>Handles orbit generator cards after DRAG in SETORB</td><td>added ATMCAL card</td></tr>
<tr><td>SETORB</td><td>Interprets optional orbit generator cards</td><td>added interpretation of ATMCAL card</td></tr>
<tr><td>SHUTDAF</td><td>Closes I/O files</td><td>added closing FRN 106</td></tr>
<tr><th colspan="3">Variables added in ATMCALJAC common block</th></tr>
<tr><th>Variable</th><th>Description</th><th>Type</th></tr>
<tr><td>CALB1JAC</td><td>Array of b_{1j} values</td><td>Output, REAL*8</td></tr>
<tr><td>CALB2JAC</td><td>Array of b_{2j} values</td><td>Output, REAL*8</td></tr>
<tr><td>CALINITJAC</td><td>Switch to initialize ATMCALJAC common block</td><td>Input/Output, Logical</td></tr>
<tr><td>CALSWITJAC</td><td>Switch to turn on JR-71 correction</td><td>Input, Logical</td></tr>
<tr><td>DATEBEGJAC</td><td>Beginning date of correction file</td><td>Output, REAL*8</td></tr>
<tr><td>DATEENDJAC</td><td>Ending date of correction file</td><td>Output, REAL*8</td></tr>
</table>

continued on next page...

...continued from previous page

Routine	Description	Change
SPANEPCHJAC	Array of span length for each span j	Output, REAL*8

Routines changed to fix bugs or include features

Routine	Description	Change
ASCII_ORB1_DATA	Writes .ASCII text files along with .ORB1 binary files	ported routine from VAX-GTDS
ELEME	Converts element sets among Cartesian, Keplerian, and spherical formats	Removed debugging print statement
FILESBD	Defines FRN for I/O files	Added FRNs 101–105 for .ASCII files
OBSWF	Writes observation working file for DATASIM	fixed year-rollover bug
ORB1	Writes .ORB1 binary files	added call to ASCII_ORB1_DATA
SETDAF	Opens files	added opening FRN 101–105, and made data file opens read-only to permit multi-user access
STARPT	generates printer summary report of passes in DATASIM run	Added test to ensure there was a non-zero number of records to fix the no-observations bug.

C.2 GTDS Data Files

The following table lists the particular GTDS binary files required when executing the Metzinger test cases [38] and when running AtmoCal. Links to these files must be created in the current directory in the format *GTDS$###*, where *###* stands for the three-digit FORTRAN reference number (FRN) of the data file. These links are automatically created by Atmocal and the _*.com_ files that run the Metzinger test cases. The binary or ascii GTDS input and output files are also linked in the same fashion, and the ones commonly used with AtmoCal and the Metzinger test cases are also included in the table. This includes the ascii data file (described in more detail in Section E.3) containing the atmospheric density correction coefficients.

Table C.2: List of GTDS Data Files

FRN	Description	File Name(s)	Notes
001	stub for small files directory	*sfdir.dat*	Universally applicable.
002	Harris-Priester atmosphere density tables	*atmosden.dat*	Universally applicable.
008	Earth Geopotential Field (21x21 models)	*radarsat_earthfld.dat*	Updated for Radarsat FD Program. Use for gtds_granholm.
		old_earthfld.dat	Baseline version. Use for Metzinger test cases.
013	Error Messages	*errormsg.dat*	Universally applicable.
014	SLP Mean of 1950 (from GSFC)	*june94.msgen.slp.mn1950.dat*	Updated in 1994. Use for gtds_granholm.
		gtds.de96.slp1950.bin.data	Older version. Use for Metzinger test cases.
015	list of observations in OBSCARD format	*#####_datasim.obscard*	used by GTDS DC subroutine, created by *genobs.pl*
		#####_datasim.obscard	created by converting B3 observations
023	Modified Newcomb Operator File	*newcomb.dat*	Universally applicable.
038	Timing Coefficient File (from GSFC)	*june94.msgen.slp.timcof.dat*	Updated in 1994. Use for gtds_granholm.
		gtds.de96.timecoef.bin.data	Older version. Use for Metzinger test cases.
075	Jacchia-Roberts Atmospheric Density Model data	jacchia.data	Covers 03/02/1966 to 02/15/1986. Use for Metzinger test cases.

continued on next page...

...continued from previous page

FRN	Description	File Name(s)	Notes
		jrdat_nomn.dat	Covers 01/10/1980 to 09/30/2008. Uses real data through 1997. Use for gtds_granholm to introduce mismodeling.
		jrdat_nomn_new.dat	Covers 01/10/1980 to 09/30/2008. Uses real data through 2000. Use for gtds_granholm when simulating data and when no mismodeling is desired.
076	MSISE-90 Atmospheric Density Model data	it ms90_nomn	Covers 01/10/1980 to 09/30/2008. Uses real data through 1997. (Analogous to *jrdat_nomn.*)
078	SLP True of Date (from GSFC)	*june94.msgen.slp.tod1950.dat*	Updated in 1994. Use for gtds_granholm.
		gtds.de96.slptod.bin.data	Older version. Use for Metzinger test cases.
106	Jacchia-Roberts Correction File	*jac_densvars.txt*	Created by AtmoCal. The filename may include a number (e.g. *jac_densvars1.txt* if ballistic factor iteration is running.

Many of the files used for the Metzinger test cases are older and may be considered obsolete when compared with new versions, but are still required for reproducing the validation results, and may be appropriate if working with older data. All of the data files listed in the following table are included in the gtds_granholm CVS source tree, in the *data* subdirectory, with the exception of the input/output files and versions of the atmospheric density correction file (FRN 106) created by AtmoCal.

If AtmoCal is to be run for time periods later than the end of 2000, a new version of the JR-71 Atmospheric Density Model data file (FRN 075) should be built. This file contains values for K_p and exospheric temperature (which is dependent on $F_{10.7}$) used by the JR-71 model, listed by modified Julian date ($MJD = JD - 2430000$). Utilities for converting the data files between binary and ascii versions (which may be modified or appended to) are available for various platforms. A UNIX utility for doing so is included in the *utils/gtds_binaries/jacchia* subdirectory of AtmoCal.

C.3 Additions to the Metzinger Test Cases

The Metzinger test cases are included in the gtds_granholm CVS tree. The *gtds_test* directory contains all of the *.com* files that run individual tests, along with *run_all.com* and *clear_output.com*, which are used to run all of the test cases and to delete all of the output files created by the test cases, respectively.

The current version of gtds_granholm includes several additions to the main GTDS core. Most notable are the NORAD PPT2 theory and the atmospheric density correction for the JR-71 model. Both of these were added after the Metzinger test cases were created, and so are not included. Paul Cefola supplied two test cases for the PPT2 theory (listed as cases #22 and #23), and an atmospheric density correction case was created (listed as case #25, leaving case #24 for an additional PPT2 test case). Scripts to run all of the test cases were updated or created and added to the gtds_granholm CVS source tree, in the *gtds_test* directory. The following pages give the details of these new test cases, in the same format as the Metzinger test cases.

GTDS Implementation Comparisons

RUN # 22

Run Description: PPT2

This test case is the first of two designed to test the PPT2 routines.

Parameter (start of ephem)	IBM value	SGI value	Δ IBM - SGI
X-position	-4362.799993995842	-4362.799993995843	1.0000e-12
Y-position	-4996.725762887517	-4996.725762887517	0
Z-position	58.34324887128177	58.34324887128177	0
X-velocity	2.460574958340194	2.460574958340194	0
Y-velocity	-2.171362444742125	-2.171362444742125	0
Z-velocity	7.023659023412546	7.023659023412548	-2.0000e-15

Parameter (end of ephem)	IBM-PC value	SGI value	Δ IBM-PC − SGI
X-position	-3698.959898403356	-3698.959898403598	2.4200e-10
Y-position	-5108.586932191689	-5108.586932191692	3.0000e-12
Z-position	2138.612735536394	2138.612735536028	3.6600e-10
X-velocity	4.058805916691082	4.058805960119506	-4.3428e-8
Y-velocity	-0.2793133613614866	-0.2793133643505460	2.9891e-09
Z-velocity	6.567329961251271	6.567330031521115	-7.0270e-08

Input Deck for Run 22: PPT2

```
CONTROL   EPHEM                                                   NSSC      9494
EPOCH                820223.0            0.0
ELEMENT1  8 19  1    6635.0814           0.010201164         64.9567
ELEMENT2             228.6393            271.2229            88.164558
ELEMENT7             5.37D-8             0.0
OUTPUT    8  2  1    820224.0            0.0                 3600.0
ORBTYPE  19  1  8    1.0                                     3.0
OGOPT
POTFIELD  1  7
STATEPAR  3
STATETAB  1  2  3    4.0                 5.0                 6.0
DRAGPAR   7
END
FIN
```

Output from IBM-PC version

```
                                        GTDS FINAL REPORT                                                     PAGE 19

                    SATELLITE NAME                    NSSC
                    SATELLITE NUMBER                    9494
                    RUN REFERENCE DATE            FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                    RUN EPOCH DATE                FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                    RUN FINAL TIME                FEB  24, 1982     0 HRS     0 MINS      0.00000 SECONDS
                    TOTAL TIME OF FLIGHT                  1 DAYS    0 HRS     0 MINS      0.00000 SECONDS
                    CAUSE OF TERMINATION          SPECIFIED TIME OF FLIGHT REACHED


                                           ***END  CONDITIONS***
CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                             "NORAD" TRUE OF REF. -- EARTH EQUATOR
X    -0.3698959898403356E+04            Y    -0.5108586932191689E+04            Z     0.2138612735536394E+04
VX    0.4058805916691082E+01            VY   -0.2793133613614866E+00            VZ    0.6567329961251271E+01

SMA   0.6641090937280772E+04            ECC   0.9348801209777984E-02            INC   0.6496916825830493E+02
LAN   0.2249825559215082E+03            AP    0.2726586686448444E+03            MA    0.1070778675769625E+03
EA    0.1075884737303912E+03            P     0.1496129390682183E+01            SLR   0.6640510505374604E+04
PR    0.6579004698292076E+04            APR   0.6703177176269468E+04            PH    0.2008696982920756E+03
APH   0.3250421762694677E+03            C3   -0.3000995089212615E+02            TA    0.1080983674296496E+03

RA    0.2340929380179728E+03            DEC   0.1873068078222401E+02            VPA   0.8948938505007797E+02
AZ    0.2653647127553109E+02            RMAG  0.6659852040875556E+04            VMAG  0.7725396057364970E+01


                                       ***INITIAL  CONDITIONS***
CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                             "NORAD" TRUE OF REF. -- EARTH EQUATOR
X    -0.4362799993995842E+04            Y    -0.4996725762887517E+04            Z     0.5834324887128177E+02
VX    0.2460574958340194E+01            VY   -0.2171362444742125E+01            VZ    0.7023659023412546E+01

SMA   0.6635081399999997E+04            ECC   0.1020116400000378E-01            INC   0.6495669999999998E+02
LAN   0.2286392999999999E+03            AP    0.2712228999999976E+03            MA    0.8816455800000188E+02
EA    0.8874890230765868E+02            P     0.1494099074174506E+01            SLR   0.6634390928568162E+04
PR    0.6567395846485223E+04            APR   0.6702766953514772E+04            PH    0.1892608464852228E+03
APH   0.3246319535147713E+03            C3   -0.3003713155620669E+02            TA    0.8933332183825829E+02

RA    0.2288747564790733E+03            DEC   0.5039289725295981E+00            VPA   0.8941564554736215E+02
AZ    0.2504433545387155E+02            RMAG  0.6633603550997212E+04            VMAG  0.7752485412383039E+01
```

Output from gtds_granholm on SGI-UNIX

```
1
                                                 GTDS FINAL REPORT                                                     PAGE 19

                             SATELLITE NAME                    NSSC
                             SATELLITE NUMBER                    9494
                             RUN REFERENCE DATE            FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             RUN EPOCH DATE                FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             RUN FINAL TIME                FEB  24, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             TOTAL TIME OF FLIGHT                1 DAYS    0 HRS     0 MINS      0.00000 SECONDS
                             CAUSE OF TERMINATION          SPECIFIED TIME OF FLIGHT REACHED


                                                    ***END  CONDITIONS***
         CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                             "NORAD" TRUE OF REF. -- EARTH EQUATOR
         X    -0.3698959898403598D+04             Y    -0.5108586932191692D+04             Z     0.2138612735536028D+04
         VX    0.4058805960119506D+01             VY   -0.2793133643505460D+00             VZ    0.6567330031521115D+01

         SMA   0.6641091078597903D+04             ECC   0.9348794762049925D-02             INC   0.6496916825830496D+02
         LAN   0.2249825559215084D+03             AP    0.2726587933083938D+03             MA    0.1070777429079298D+03
         EA    0.1075883490613897D+03             P     0.1496129438436848D+01             SLR   0.6640510647480014D+04
         PR    0.6579004881108011D+04             APR   0.6703177276087796D+04             PH    0.2008698811080103D+03
         APH   0.3250422760877955D+03             C3   -0.3000995025353829D+02             TA    0.1080982427660965D+03

         RA    0.2340929380179712D+03             DEC   0.1873068078222063D+02             VPA   0.8948938505007749D+02
         AZ    0.2653647127553051D+02             RMAG  0.6659852040875576D+04             VMAG  0.7725396140025803D+01


                                                ***INITIAL  CONDITIONS***
         CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                             "NORAD" TRUE OF REF. -- EARTH EQUATOR
         X    -0.4362799993995843D+04             Y    -0.4996725762887517D+04             Z     0.5834324887128177D+02
         VX    0.2460574958340194D+01             VY   -0.2171362444742125D+01             VZ    0.7023659023412548D+01

         SMA   0.6635081400000000D+04             ECC   0.1020116400001255D-01             INC   0.6495670000000001D+02
         LAN   0.2286393000000000D+03             AP    0.2712228999999993D+03             MA    0.8816455799999986D+02
         EA    0.8874890230765665D+02             P     0.1494099074174507D+01             SLR   0.6634390928568164D+04
         PR    0.6567395846485167D+04             APR   0.6702766953514833D+04             PH    0.1892608464851664D+03
         APH   0.3246319535148332D+03             C3   -0.3003713155620668D+02             TA    0.8933332183825677D+02

         RA    0.2288747564790735D+03             DEC   0.5039289725295982D+00             VPA   0.8941564554736216D+02
         AZ    0.2504433545387155D+02             RMAG  0.6633603550997212D+04             VMAG  0.7752485412383041D+01
```

GTDS Implementation Comparisons

RUN # 23

Run Description: PPT2_DSST

This test case is the second of two designed to test the PPT2 routines, using the Draper Semianalytical Satellite Theory (DSST).

Parameter (start of ephem)	IBM-PC value	SGI value	Δ IBM-PC – SGI
X-position	-4395.322718525774	-4395.322718525774	0
Y-position	-4969.867719377015	-4969.867719377015	0
Z-position	-55.43442842496479	-55.43442842496481	3.0000e-14
X-velocity	1.460354979847724	1.460354979847724	0
Y-velocity	-1.385569234061044	-1.385569234061044	0
Z-velocity	7.485026733917411	7.485026733917411	0

Parameter (end of ephem)	IBM value	SGI value	Δ IBM - SGI
X-position	-4066.935924059110	-4066.935924059165	5.5000e-11
Y-position	-4950.373019647758	-4950.373019647797	3.9000e-11
Z-position	1745.839972520243	1745.839972520178	6.5000e-11
X-velocity	2.857256375140294	2.857256375140107	1.8700e-13
Y-velocity	0.1955615298471337	0.1955615298469322	2.015e-13
Z-velocity	7.200507756981554	7.200507756981573	-1.9000e-14

Input Deck for Run 23: PPT2_DSST

```
CONTROL   EPHEM                                                  NSSC      9494
EPOCH              820223.0            0.0
ELEMENT1  8 19  1  6635.0814           0.0010201164        74.9567
ELEMENT2           228.6393            271.2229             88.164558
ELEMENT7           0.0                 0.0
OUTPUT    8  2  1  820224.0            0.0                  3600.0
ORBTYPE  19  1  8  1.0                                      2.0
OGOPT
POTFIELD  1  7
MAXDEGEQ  1        5.0
MAXORDEQ  1        5.0
PPT2_POS  2  3     180.0
PPT2_COF  1  3     43200.0
PPT2_MDY  5  5  3
PPT2_TLC  5  5  4  2.                  -4.0                 +4.0
PPT2_OUT     1  1  1.0
END
FIN
```

Output from IBM-PC version

```
1
                                                GTDS FINAL REPORT                                                     PAGE 19

                             SATELLITE NAME                   NSSC
                             SATELLITE NUMBER                   9494
                             RUN REFERENCE DATE          FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             RUN EPOCH DATE              FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             RUN FINAL TIME              FEB  24, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             TOTAL TIME OF FLIGHT                1 DAYS    0 HRS     0 MINS      0.00000 SECONDS
                             CAUSE OF TERMINATION        SPECIFIED TIME OF FLIGHT REACHED


                                                   ***END  CONDITIONS***
         CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                            "NORAD" TRUE OF REF. -- EARTH EQUATOR
         X    -0.4066935924059110E+04           Y    -0.4950373019647758E+04            Z    0.1745839972520243E+04
         VX    0.2857256375140294E+01           VY    0.1955615298471337E+00            VZ    0.7200507756981554E+01

         SMA   0.6642861474678574E+04           ECC   0.5089343483117673E-03            INC   0.7496416526150213E+02
         LAN   0.2263977269652457E+03           AP    0.5759183759472086E+02            MA    0.3182442963879283E+03
         EA    0.3182248698783509E+03           P     0.1496727740323145E+01            SLR   0.6642859754083317E+04
         PR    0.6639480694303033E+04           APR   0.6646242255054115E+04            PH    0.2613456943030333E+03
         APH   0.2681072550541148E+03           C3   -0.3000195226976088E+02            TA    0.3182054396803783E+03

         RA    0.2305954826571028E+03           DEC   0.1524305312449083E+02            VPA   0.9001942651134878E+02
         AZ    0.1559778711201418E+02           RMAG  0.6640340206172306E+04            VMAG  0.7749159326248721E+01


                                               ***INITIAL  CONDITIONS***
         CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                            "NORAD" TRUE OF REF. -- EARTH EQUATOR
         X    -0.4395322718525774E+04           Y    -0.4969867719377015E+04            Z   -0.5543442842496479E+02
         VX    0.1460354979847724E+01           VY   -0.1385569234061044E+01            VZ    0.7485026733917411E+01

         SMA   0.6635081399999998E+04           ECC   0.1020116399966777E-02            INC   0.7495670000000000E+02
         LAN   0.2286392999999999E+03           AP    0.2712228999999877E+03            MA    0.8816455800001597E+02
         EA    0.8822297825519422E+02           P     0.1494099074174506E+01            SLR   0.6635074495285680E+04
         PR    0.6628312844648744E+04           APR   0.6641849955351253E+04            PH    0.2501778446487433E+03
         APH   0.2637149553512527E+03           C3   -0.3003713155620669E+02            TA    0.8828139944454566E+02

         RA    0.2285106384484784E+03           DEC  -0.4787124722071585E+00            VPA   0.8994157973466987E+02
         AZ    0.1504383749398396E+02           RMAG  0.6634871507719595E+04            VMAG  0.7751004062507056E+01
```

Output from gtds_granholm on SGI-UNIX

```
1
                                                 GTDS FINAL REPORT                                                    PAGE 19

                             SATELLITE NAME                    NSSC
                             SATELLITE NUMBER                    9494
                             RUN REFERENCE DATE            FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             RUN EPOCH DATE                FEB  23, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             RUN FINAL TIME                FEB  24, 1982     0 HRS     0 MINS      0.00000 SECONDS
                             TOTAL TIME OF FLIGHT                  1 DAYS    0 HRS     0 MINS      0.00000 SECONDS
                             CAUSE OF TERMINATION          SPECIFIED TIME OF FLIGHT REACHED


                                                    ***END  CONDITIONS***
         CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                             "NORAD" TRUE OF REF. -- EARTH EQUATOR
         X    -0.4066935924059165D+04            Y    -0.4950373019647797D+04            Z     0.1745839972520178D+04
         VX    0.2857256375140107D+01            VY    0.1955615298469322D+00            VZ    0.7200507756981573D+01

         SMA   0.6642861474678571D+04            ECC   0.5089343483094239D-03            INC   0.7496416526150215D+02
         LAN   0.2263977269652458D+03            AP    0.5759183759339012D+02            MA    0.3182442963892545D+03
         EA    0.3182248698796784D+03            P     0.1496727740323144D+01            SLR   0.6642859754083313D+04
         PR    0.6639480694303045D+04            APR   0.6646242255054095D+04            PH    0.2613456943030451D+03
         APH   0.2681072550540948D+03            C3   -0.3000195226976090D+02            TA    0.3182054396817085D+03

         RA    0.2305954826571028D+03            DEC   0.1524305312449015D+02            VPA   0.9001942651134750D+02
         AZ    0.1559778711201413D+02            RMAG  0.6640340206172353D+04            VMAG  0.7749159326248664D+01


                                                ***INITIAL  CONDITIONS***
         CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                             "NORAD" TRUE OF REF. -- EARTH EQUATOR
         X    -0.4395322718525774D+04            Y    -0.4969867719377015D+04            Z    -0.5543442842496481D+02
         VX    0.1460354979847724D+01            VY   -0.1385569234061044D+01            VZ    0.7485026733917411D+01

         SMA   0.6635081399999998D+04            ECC   0.1020116399990371D-02            INC   0.7495670000000001D+02
         LAN   0.2286393000000000D+03            AP    0.2712228999999878D+03            MA    0.8816455800001398D+02
         EA    0.8822297825519223D+02            P     0.1494099074174506D+01            SLR   0.6635074495285680D+04
         PR    0.6628312844648587D+04            APR   0.6641849955351409D+04            PH    0.2501778446485869D+03
         APH   0.2637149553514091D+03            C3   -0.3003713155620669D+02            TA    0.8828139944454574D+02

         RA    0.2285106384484785D+03            DEC  -0.4787124722071587D+00            VPA   0.8994157973466987D+02
         AZ    0.1504383749398397D+02            RMAG  0.6634871507719595D+04            VMAG  0.7751004062507056D+01
```

GTDS Implementation Comparisons

RUN #25

Run Description: EPHEM_M50_COWELL_JACCHIA_CORR

This run is a modified version of run #12 designed to test the atmospheric density correction routines. A set of correction factors is applied to the JR-71 model, found in the file *testcase_jac_densvars.txt*. This run generates a 3 day ephemeris from an initial osculating keplerian state vector using the Cowell orbit generator. The GEM-10B gravity model is used.

Parameter (start of ephem)	SGI value
X-position	150.5086950768926
Y-position	-1146.217167965407
Z-position	-6990.621444318102
X-velocity	-6.964869483515692
Y-velocity	2.707531390482961
Z-velocity	-0.5944769832228497

Parameter (end of ephem)	SGI value
X-position	3885.285299122795
Y-position	-436.0351407180455
Z-position	5916.839606713098
X-velocity	5.784088135415042
Y-velocity	-2.596727010799154
Z-velocity	-3.991908863790896

Input Deck for Run 25: EPHEM_M50_COWELL_JACCHIA_CORR

```
CONTROL   EPHEM                                              LNDSAT-4   8207201  00000100
EPOCH               820224.0            0.0                                      00000200
ELEMENT1  1  2  1   7077.8              0.0011              98.2                 00000300
ELEMENT2            158.1               89.4                176.0                00000400
OUTPUT    1  2  1   820227.0            0.0                 43200.               00000500
ORBTYPE   2  1  1   60.0                                                             00000600
OGOPT                                                                            00000700
DRAG      1         1
ATMOSDEN        1
ATMCAL    1  1
SCPARAM             1.D-6               100.D0
POTFIELD  1  6                                                                   00000710
OUTOPT    1         820224000000.       820227000000.       60.
END                                                                              00000800
FIN                                                                              00000900
```

Contents of *jacchia_corr.txt* for Run 25: EPHEM_M50_COWELL_JACCHIA_CORR

```
2445024.5000  2.0000000000E-01  1.0000000000E-01
2445024.6250  2.0000000000E-01  1.0000000000E-01
2445024.7500  2.0000000000E-01  1.0000000000E-01
2445024.8750  2.0000000000E-01  1.0000000000E-01
2445025.0000  2.0000000000E-01  1.0000000000E-01
2445025.1250  2.0000000000E-01  1.0000000000E-01
2445025.2500  2.0000000000E-01  1.0000000000E-01
2445025.3750  2.0000000000E-01  1.0000000000E-01
2445025.5000  2.0000000000E-01  1.0000000000E-01
2445025.6250  2.0000000000E-01  1.0000000000E-01
2445025.7500  2.0000000000E-01  1.0000000000E-01
2445025.8750  2.0000000000E-01  1.0000000000E-01
2445026.0000  2.0000000000E-01  1.0000000000E-01
2445026.1250  2.0000000000E-01  1.0000000000E-01
2445026.2500  2.0000000000E-01  1.0000000000E-01
2445026.3750  2.0000000000E-01  1.0000000000E-01
2445026.5000  2.0000000000E-01  1.0000000000E-01
2445026.6250  2.0000000000E-01  1.0000000000E-01
2445026.7500  2.0000000000E-01  1.0000000000E-01
2445026.8750  2.0000000000E-01  1.0000000000E-01
2445027.0000  2.0000000000E-01  1.0000000000E-01
```

Output from gtds_granholm on SGI-UNIX

```
                              GTDS FINAL REPORT                                              PAGE 12

                  SATELLITE NAME              LNDSAT-4
                  SATELLITE NUMBER               8207201
                  RUN REFERENCE DATE          FEB  24, 1982     0 HRS      0 MINS      0.00000 SECONDS
                  RUN EPOCH DATE              FEB  24, 1982     0 HRS      0 MINS      0.00000 SECONDS
                  RUN FINAL TIME              FEB  27, 1982     0 HRS      0 MINS      0.00000 SECONDS
                  TOTAL TIME OF FLIGHT               3 DAYS     0 HRS      0 MINS      0.00000 SECONDS
                  CAUSE OF TERMINATION        SPECIFIED TIME OF FLIGHT REACHED


                                    ***END  CONDITIONS***
CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                          MEAN OF 1950.0 -- EARTH EQUATOR
X      0.3885285299122795D+04           Y    -0.4360351407180455D+03            Z      0.5916839606713098D+04
VX     0.5784088135415042D+01           VY   -0.2596727010799154D+01            VZ    -0.3991908863790896D+01

SMA    0.7082881037729563D+04           ECC   0.1297085922290635D-02            INC    0.9818746774924638D+02
LAN    0.1610200796951653D+03           AP    0.2905176602292393D+03            MA     0.1920652024849152D+03
EA     0.1920496879622824D+03           P     0.1647871725313703D+01            SLR    0.7082869121264634D+04
PR     0.7073693932446265D+04           APR   0.7092068143012862D+04            PH     0.6955559324462647D+03
APH    0.7139301430128617D+03           C3   -0.2813829837580983D+02            TA     0.1920341832671292D+03

RA     0.3535966421180496D+03           DEC   0.5654442695725306D+02            VPA    0.9001551453530473D+02
AZ     0.1949707386649364D+03           RMAG  0.7091865722860912D+04            VMAG   0.7492262882712758D+01


                                 ***INITIAL  CONDITIONS***
CENTRAL BODY IS EARTH   (INERTIAL SYSTEM)                          MEAN OF 1950.0 -- EARTH EQUATOR
X      0.1505086950768926D+03           Y    -0.1146217167965407D+04            Z     -0.6990621444318102D+04
VX    -0.6964869483515692D+01           VY    0.2707531390482961D+01            VZ    -0.5944769832228497D+00

SMA    0.7077800000000003D+04           ECC   0.1099999999988060D-02            INC    0.9820000000000002D+02
LAN    0.1581000000000001D+03           AP    0.8940000000000003D+02            MA     0.1760000000000000D+03
EA     0.1760043916076793D+03           P     0.1646098845766884D+01            SLR    0.7077791435862003D+04
PR     0.7070014420000087D+04           APR   0.7085585579999919D+04            PH     0.6918764200000869D+03
APH    0.7074475799999191D+03           C3   -0.2815849840345869D+02            TA     0.1760087808084915D+03

RA     0.2774806566775013D+03           DEC  -0.8060983937027528D+02            VPA    0.8999560838967234D+02
AZ     0.2409486213137905D+03           RMAG  0.7085566656322731D+04            VMAG   0.7496234790642598D+01
```

C.4 List of Known Bugs/Issues

As mentioned in Section 7.2.2, several bugs still exist in gtds_granholm. The following list is partially quoted from Appendix B.4 of Granholm's thesis [13], and includes both a description of the bug and its effect on the AtmoCal code:

1) **Hang-up Error:** This error sometimes occurs when low-altitude objects are calculated to impact the Earth. GTDS appears to hang up and mus be manually interrupted. A possible culprit is the *SECHECK.FOR* routine. This error may be reducing the number of runs that converge in the density correction process.

   All AtmoCal routines that call GTDS include the `$time_limit` variable, which contains the number of seconds for which GTDS may run. After that much time has elapsed, AtmoCal assumes that GTDS has hit a hang-up error and terminates the run.

2) **DC Epoch Limitation:** When using an input *.OBS* file (GTDS$ 029) for a DC run, GTDS halts execution unless the start of the OBSINPUT card matches the solve-for epoch.

3) **Random Number Generation Bug:** There appears to be a bias in random noise added to observations using DATASIM only when the optimized compilation of GTDS is executed. If the non-optimized (debug) version of the code is used, the bias disappears. The source of the error appears to be the *RANDU.FOR* routine. Every time random noise is generated for the same list of satellites, using the same initial conditions, identical results are produced. This is most likely the result of using the same seed for the pseudorandom number generation.

4) **Residual Plot Error:** DC Residual plots are not functional.

5) **Y2K Bug in Station Pass Report:** The full date field does not appear for dates after Jan 1, 2000 in the DATASIM Station Pass Report.

There are other Y2K errors as well, which should be fixed in gtds_granholm whenever a patch from the another development tree of GTDS is merged into gtds_granholm. Chris Sabol at CSDL has developed a Y2K patch for the PC version of GTDS. Since gtds_granholm only alters a few minor sections of GTDS, merging the two versions should not only be possible, but quite simple. (This would probably be done using context diffs. A "context diff" is a line-by-line list of differences between two version of a piece of software. CVS stores information about earlier versions by using context diffs, and numerous UNIX utilities exist for manipulating diffs. Type `man diff` at a UNIX prompt for more details.) .

Appendix D

Annotated Code

This chapter contains the code of all of the files included in the AtmoCal CVS distribution at the time of printing this thesis[1]. This version can be retrieved by checking out AtmoCal Version 2.0, even if newer versions have become available. (Version 1.0 contains the code as it was at the start of this research, which is similar but not identical to the code printed in Granholm's thesis[13], and most portions will not run properly or at all without modification on Pisces, since there are direct references to the directory structure on the DC1 machine at CSDL, as well as some unfinished and undocumented changes Granholm made after publication of his thesis.)

Please note: the line numbers in this documentation will not always exactly match those seen in the code, since a handful of extremely long lines have been line-wrapped.

---

[1] The only differences between the code printed here and the CVS distribution are possible changes to the user-defined options, since those are set by the user before each individual run.

D.1 TLE2osc.pl

Table D.1: TLE2osc.pl Fact Sheet

Function	Performs initial setup for both simulated and real data processing.
Language	Perl
Type	Main Program
Location	Main AtmoCal directory
Input Files	TLE file (contains two-line element sets) RCS file (contains radar cross-sectional areas of satellites)
Other Code Req'd	*Dates.pm, Filehandle.pm, Localmath.pm*
Environment Variables	`$ATM_CAL` and `$ATM_EPHEM` must be set.
Data Structure	Requires: The `$ATM_CAL` and `$ATM_EPHEM` directories must exist. Creates: The *ATM_CALmodel_opt* and *$ATM_EPHEM/$model_opt* directories are created, if not already present.
Output Files	*initinfo.txt* in *ATM_CALmodel_opt* *.output, .orbit, .orb1, .ascii* files for each satellite in *$ATM_EPHEM/$model_opt* (optional – if $initinfo_only is set to 1, these are not created)
User-Defined Variables	
`$start_epoch`	Starting time (¿1min after last TLE)
`$end_epoch`	Ending time for orbit generation (If working with real data, this can be fairly arbitrary, as long as it is after the $start_epoch.)
`$model_opt`	Directory in `$ATM_EPHEM` for output files
`$tle_file`	Path to TLE file
`$rcsfile`	Path to RCS file
`$time_limit`	Allowed time (in seconds) for individual GTDS EPHEM runs. Normally 180.
`$initinfo_only`	0 normally, 1 to only create initinfo.txt and not run GTDS EPHEM

Commented Code (TLE2osc.pl)

```
#!/usr/bin/perl -w
#
# TLE2osc.pl - TLE Conversion Program
#
# Author:
#
# George R. Granholm
# 22 Mar 00
#
#
################################################################
#                                                              #
# Header section: All user-definable variables are here.       #
#                                                              #
################################################################

# Add environment-specific path and import modules.

BEGIN {
   push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

use Dates;       # Necessary to use cal2jul, jul2cal, & get_time subroutines
use FileHandle; # For autoflush

# Set options and variables

$start_epoch = "991215 000000.0"; # Must be at least 1 minute after last TLE epoch
$end_epoch   = "1000211 000000.0";
$modeℓ_opt   = "lowgrav_new";
$tℓe_file    = "$ENV{ATM_CAL}/200_600_tles.txt";

$rcsfiℓe     = "$ENV{ATM_CAL}/rcs.txt";
$time_limit  = 180;
$initinfo_only = 0; # if this is 1, only create initinfo.txt

($start_ymd, $start_hms) = split(" ", $start_epoch);
($end_ymd, $end_hms) = split(" ", $end_epoch);

# Open necessary files

mkdir "$ENV{ATM_CAL}/${model_opt}", 0777;
mkdir "$ENV{ATM_EPHEM}/${model_opt}", 0777;
chdir "$ENV{ATM_CAL}/${model_opt}"; # This keeps the GTDS$FRN symlinks out of
                                    # the code directory.
$ℓogfiℓe     = "$ENV{ATM_CAL}/${model_opt}/TLE2osc.log";
$initfiℓe    = "$ENV{ATM_CAL}/${model_opt}/initinfo.txt";
open LOGINFO, ">>$logfile" or die "Unable to create $logfile, died";
open STDERR, ">>&LOGINFO" or die "Unable to redirect stderr to $logfile, died";
```

```
open TLES, $tℓe_file or die "Invalid TLE filename: $!\n";
open INITINFO, ">>$initfile" or die "Unable to create $initfile, died";

foreach $fh ("STDOUT", "LOGINFO", "STDERR", "INITINFO") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\tTLE2osc.pl: Processing $tle_file\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Read in RCS into $rcs{$catnum} hash

open RCSFILE, "<$rcsfile" or die "Unable to open $rcsfile, died";
while (defined($rcsℓine = <RCSFILE>)) {
    chop $rcsℓine;
    ($catnum, $area) = split(" ", $rcsℓine);
    $catnum = sprintf "%5.5d", $catnum;
    if ($area) { $rcs{$catnum} = $area;}
    else { $rcs{$catnum} = 2.2;}    # Set default to 2.2 m^2 if no data
}
close RCSFILE;

##################################################################
#                                                                #
# Main Loop through TLE file                                     #
#                                                                #
##################################################################

# Read from TLE file

LINE: while (defined($ℓine = <TLES>)) {    # Main loop through TLE file

    next LINE if ($ℓine =~ /^[^12][^\s][^\d]/);
    if ($ℓine =~ s/^1\s(\d{5})\w\s(\d{5})\s*(\w{1,3})//) {    #  Match first TLE line
        $catnum = $1;
        $intℓ_des = $2 . $3;
        chop $ℓine;
        @line = split(" ",$ℓine);
        $norad_date = $ℓine[0];

        # Convert NORAD epoch to calender date

        ($yr, $day) = ($norad_date =~ /(^\d{2})(\d{3}\.\d{8})/);
```

```
        $yr_days = cal2jul($yr,1,1,0,0,0);          # First convert year to Julian date
        $nor_juldat = ($yr_days + $day - 1);         # Add day number to Julian date
        @nor_caldat = jul2cal($nor_juldat);          # Convert back to calender date
        ($nor_caldat[0]) = ($nor_caldat[0] =~ /\d{2}(\d{2})/);    # Two-digit year
        if ($nor_caldat[0] == 0) {$nor_caldat[0] = "100";}        # GTDS Y2K fix
        $ymd = join("",@nor_caldat[0 .. 2]);
        $hms = join("",@nor_caldat[3 .. 5]);

        # Calculate end time of GP4 propagation (one minute after NORAD epoch)

        $gp4end_jul = $nor_juldat + 1/1440;      # Next minute after NORAD epoch
        @gp4end_cal = jul2cal($gp4end_jul);
        ($gp4end_cal[0]) = ($gp4end_cal[0] =~ /\d{2}(\d{2})/);    # Two-digit year
        if ($gp4end_cal[0] == 0) {$gp4end_cal[0] = "100";}        # GTDS Y2K fix
        $gp4end_ymd = join("",@gp4end_cal[0 .. 2]);
        $gp4end_hms = join("",@gp4end_cal[3 .. 5]);

        # Read remaining elements

        $dndt = $ℓine[1];
        $d2ndt2 = $ℓine[2];
        $bstar = $ℓine[3];

        # Convert d2n/dt2 and B* to standard numerical formats

        if ($d2ndt2 =~ /(-*)(\d{5})([+-]\d)/) {
            $d2ndt2 = $1 . "0." . $2 . "E" . $3;
        }

        if ($bstar =~ /(-*)(\d{5})([+-]\d)/) {
            $bstar = $1 . "0." . $2 . "E" . $3;
        }

        # Apply Dave Vallado's multiplier to obtain B from B*, and
        # compute drag coefficient using RCS area and default C_d

        $baℓℓ_fact = 6.3708105*$bstar;       # where B = 1/2 (Ax/m) C_d
        $Ax = $rcs{$catnum};                  # in m^2
        $C_d = 2.2;                           # Default LEO C_d
        $mass = ($Ax*$C_d)/(2*$baℓℓ_fact);  # in kg
        $Ax_km = sprintf("%7.10E",($Ax/1000000));         # Convert to km^2

    }

    elsif ($ℓine =~ /^2\s(\d{5})/) {   #  Match second TLE line
        if ($bstar == 0 ) {next LINE};
        chop $ℓine;
        @line = split(" ",$ℓine);
        $incℓ = $ℓine[2];
        $raan = $ℓine[3];
        $ecc = $ℓine[4];
```

```
        $aop = $ℓine[5];
        $ma = $ℓine[6];
        $mm = $ℓine[7];

        # Convert eccentricity to standard numerical format

        if ($ecc =~ /(\d{7})/) {
            $ecc = "0." . $1;
        }

        # Separate mean motion from rev number if necessary

        if ((length($mm) > 11) && ($mm =~ /(\d{1,2}\.\d{8})/)) {
            $mm = $1;
        }

##########################################################
#                                                  #
# Run GTDS EPHEM to create .OUTPUT and .ORBIT/.ORB1 file      #
#                                                  #
##########################################################
        unless ($initinfo_only==1) {

          # Write GTDS card file

          $ephem_card  = "${catnum}_ephem.gtds";
          $output_file = "${catnum}_ephem.output";
          $orbit_file  = "${catnum}_ephem.orbit";
          $orb1_file   = "${catnum}_ephem.orb1";
          $ascii_file  = "${catnum}_ephem.ascii";

          open(EPHEM_CARD, ">$ENV{ATM_EPHEM}/${model_opt}/$ephem_card")
or die          "Unable to open $ENV{ATM_EPHEM}/${model_opt}/$ephem_card, died";
          write EPHEM_CARD;
          close EPHEM_CARD;

          # Make standard data file links

          system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };
# Remove any GTDS$* links
          symlink("$ENV{GTDS_DATA}/sfdir.dat",              "GTDS\$001");
          symlink("$ENV{GTDS_DATA}/atmosden.dat",           "GTDS\$002");
          symlink("$ENV{GTDS_DATA}/radarsat_earthfld.dat",    "GTDS\$008");
          symlink("$ENV{GTDS_DATA}/errormsg.dat",     "GTDS\$013");
          symlink("$ENV{GTDS_DATA}/june94.msgen.slp.mn1950.dat","GTDS\$014");
          symlink("$ENV{GTDS_DATA}/newcomb.dat",            "GTDS\$023");
          symlink("$ENV{GTDS_DATA}/june94.msgen.slp.timcof.dat","GTDS\$038");
          symlink("$ENV{GTDS_DATA}/jrdat_nomn_new.dat",       "GTDS\$075");
          symlink("$ENV{GTDS_DATA}/june94.msgen.slp.tod1950.dat","GTDS\$078");

          # Make satellite-specific data links
```

```
symlink("$ENV{ATM_EPHEM}/${model_opt}/$ephem_card",  "GTDS\$005");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$output_file", "GTDS\$006");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$orbit_file",  "GTDS\$020");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$orb1_file",   "GTDS\$024");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$ascii_file",  "GTDS\$101");

# Run GTDS!

foreach $fh ("STDOUT","LOGINFO") {
    print $fh "-" x 40,"\n";
    print $fh "  Processing NORAD Catalog \#$catnum\n";
    print $fh "-" x 40,"\n";
    print $fh "UNIX-GTDS\n";
#   print $fh "Charles Stark Draper Laboratory\n\n";
    print $fh "MIT Lincoln Laboratory\n\n";
    print $fh ("Run started at: ", get_time(), "\n");
}

undef $child_id;

if ($child_id = fork) {          # Parent process here

   local $SIG{USR1} = sub {    # Define anonymous sub to kill GTDS

      (my $gtds_id) = split (" ", `ps | grep gtds`);

      foreach $fh ("STDOUT","LOGINFO") {
         print $fh "GTDS run has exceeded time limit;\n";
         print $fh "Killing process $gtds_id\n";
      }

      kill 'QUIT', $gtds_id;
   };
   waitpid $child_id, 0;        # Wait for child process to finish
}

elsif (defined $child_id) {     # Child process here

   $par_id = getppid;

   local $SIG{ALRM} = sub {    # Define local ALRM signal handler
      kill 'USR1', $par_id;  # Send USR1 signal to parent if local alarm goes off
      foreach $fh ("STDOUT","LOGINFO") {
         print $fh "Sending USR1 to $par_id. .\n";
      }
   };

   alarm $time_limit;            # Initialize alarm to go off in $time_limit sec

#  system("dbx $ENV{GTDS_EXE_DBG}/gtds_dbg.exe");
```

```
#           system(“$ENV{GTDS_EXE}/gtds_dbg.exe”);
            system("$ENV{GTDS_EXE}/gtds.exe");
            alarm 0;              # Turn off alarm if GTDS finishes before $time_limit
            die "Exiting child process. . ."
          }

          foreach $fh ("STDOUT","LOGINFO") {
             print $fh ("Run ended at: ", get_time(), "\n");
          }

          # Compress output files using gzip
          # These files are used by genobs.pl and estbfs.pl.

          foreach $fh ("STDOUT","LOGINFO") {
             print $fh ("Compressing .orbit file. . .\n");
          }

          system "gzip -v -f $ENV{ATM_EPHEM}/${model_opt}/$orbit_file";
          system "gzip -v -f $ENV{ATM_EPHEM}/${model_opt}/$output_file";

          system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };
# Remove any GTDS$* links
          system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };
# Remove any temp files

        } # $initinfo_only flag skips to here.

        # Write line in INITINFO array

        $stan_flag = 'S';   # All satellites are standard
        $var = 1E-6;        # Default variance for standard satellites
        $obs_type = 29;     # Obs type for simulated observations
        printf INITINFO "%5s %8s %7.10E %7.10E %7.10E %1s %2d\n", $catnum,
           $intℓ_des, $baℓℓ_fact, $Ax, $var, $stan_flag, $obs_type;

     }

  }

close TLES;
close INITINFO;
close LOGINFO;

#==============   EPHEM card deck formatting ============
======================

format EPHEM_CARD =
CONTROL   EPHEM                                  @<<<<<<<  @>>>>>>>
                                             $intℓ_des, $catnum
EPOCH          @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
             $ymd,               $hms
```

```
ELEMENT1   8 18   1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                    $mm,                  $ecc,                 $incℓ
ELEMENT2            @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                    $raan,                $aop,                 $ma
ELEMENT3            @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                    $dndt,                $d2ndt2,              $bstar
OUTPUT     1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 60.0
                    $gp4end_ymd,          $gp4end_hms
ORBTYPE   14  1  8  1
OGOPT
POTFIELD  1  7
END
FIN
CONTROL   EPHEM                      OUTPUT          @<<<<<<<  @>>>>>>>
                                                     $intℓ_des, $catnum
OUTPUT     1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 86400.0
                    $start_ymd,           $start_hms
ORBTYPE    2  1  1  60.0
OGOPT
ATMOSDEN         1
DRAG       1     1
DRAGPAR    3  0    @<<<<<<<<<<<<<<<<<<<
                    $C_d
SCPARAM             @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $Ax_km,               $mass
POTFIELD  1  4
MAXDEGEQ  1       4.0
MAXORDEQ  1       4.0
SOLRAD     1      1.0
END
FIN
CONTROL   EPHEM                      OUTPUT          @<<<<<<<  @>>>>>>>
                                                     $intℓ_des, $catnum
OUTPUT     1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 86400.0
                    $end_ymd,             $end_hms
ORBTYPE    2  1  1  60.0
OGOPT
ATMOSDEN         1
DRAG       1     1
DRAGPAR    3  0    @<<<<<<<<<<<<<<<<<<<
                    $C_d
SCPARAM             @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $Ax_km,               $mass
POTFIELD  1  4
MAXDEGEQ  1       4.0
MAXORDEQ  1       4.0
SOLRAD     1      1.0
OUTOPT     2  2  1 @>>>>>>@<<<<<<<<<<<< @>>>>>>@<<<<<<<<<<<< 600
```

```
                $start_ymd, $start_hms, $end_ymd, $end_hms
END
FIN
```

D.2 genobs.pl

Table D.2: genobs.pl Fact Sheet

Function	Generates simulated observations
Language	Perl
Type	Main Program
Location	Main AtmoCal directory
Input Files	*initinfo.txt* (created by TLE2osc.pl)
	initial ephemerides in .output, .orbit files (also created by TLE2osc.pl) in *$ATM_EPHEM/$ephem_opt*
Environment Variables	`$ATM_DATASIM` must be set.
Data Structure	Requires: The `$ATM_CAL`, `$ATM_EPHEM`, and `$ATM_DATASIM` directories must exist. Creates: The *$ATM_CAL/$datasim_opt* and *$ATM_DATASIM/$datasim_opt* directories are created, if not already present.
Output Files	*.output, .obscard* files containing simulated observations
User-Defined Variables	
`$start_epoch`	Starting time (normally identical to that used in TLE2osc.pl)
`$end_epoch`	Ending time
`$ephem_opt`	Location of *TLE2osc.pl* output files (should equal value $model_opt in *TLE2osc.pl* run
`$datasim_opt`	Directory in `$ATM_DATASIM` for output files
`$time_limit`	Allowed time (in seconds) for individual GTDS DATASIM runs. Normally 180.
`$noise`	Flag for including noise in observations. 1 for noise, 0 for no noise. Other values are potentially available to use for different noise models.

Commented Code (genobs.pl)

```
#!/usr/bin/perl -w
#
# genobs.pl - Observation Generator Program
#
# Author:
#
# George R. Granholm
# 06 Apr 00
#
# Revision History:
#
# Removed explicit directories, added more comments.
```

```
# Sarah E. Bergstrom
# 01 May 2002
#
################################################################
#                                                              #
# Header section: All user-definable variables are here.        #
#                                                              #
################################################################

# Import modules

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

use Dates;
use Localmath;
use FileHandle;

# Set variables and options

$start_epoch = "991215 000000.0";
$end_epoch   = "1000211 000000.0";
$ephem_opt   = "lowgrav";
$datasim_opt = "lowgrav";
$time_limit  = 180;
$noise       = 0;        # 1 = noise.  0 = no noise.

################################################################
#                                                              #
# Preparation Section.                                          #
#                                                              #
################################################################
# set up directory stuff.

mkdir "$ENV{ATM_DATASIM}/${datasim_opt}", 0777;
chdir "$ENV{ATM_CAL}/${datasim_opt}"; #this keeps all of the GTDS$0## files
                                      # out of the CVS directory.
$ℓogfiℓe    = "$ENV{ATM_CAL}/${datasim_opt}/genobs.log";
$initfiℓe   = "$ENV{ATM_CAL}/${datasim_opt}/initinfo.txt";
*PI         = \3.14159265358979;


# Define hash which contains obs types

%obstype = (
            RANG => 1,
            AZ   => 4,
            EL   => 5,
            );
```

```
# Format start epoch

($start_ymd, $start_hms) = split(" ", $start_epoch);
$start_ymd2 = $start_ymd;
if (length($start_ymd2) == 7) {
    ($start_ymd2) = ($start_ymd2 =~ /(\d{6})$/);          # Take off GTDS Y2K fix
}                                                      # for Julian date conversion
($y,$m,$d) = ($start_ymd2 =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($start_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$start_jul = cal2jul($y,$m,$d,$h,$mn,$s);

# Calculate interval times for tracking schedule

$end_interval1 = $start_jul + 1/4;    # Six hours after start
$end_interval2 = $start_jul + 2/4;    # Twelve hours after start
$end_interval3 = $start_jul + 3/4;    # Eighteen hours after start
$end_interval4 = $start_jul + 1;   # Twenty-four hours after start
@interval1 = jul2cal($end_interval1);
@interval2 = jul2cal($end_interval2);
@interval3 = jul2cal($end_interval3);
@interval4 = jul2cal($end_interval4);
($intervaℓ1[0]) = ($intervaℓ1[0] =~ /\d{2}(\d{2})/);    # Two-digit year
if ($intervaℓ1[0] == 0) {$intervaℓ1[0] = "100";}        # GTDS Y2K fix
($intervaℓ2[0]) = ($intervaℓ2[0] =~ /\d{2}(\d{2})/);    # Two-digit year
if ($intervaℓ2[0] == 0) {$intervaℓ2[0] = "100";}        # GTDS Y2K fix
($intervaℓ3[0]) = ($intervaℓ3[0] =~ /\d{2}(\d{2})/);    # Two-digit year
if ($intervaℓ3[0] == 0) {$intervaℓ3[0] = "100";}        # GTDS Y2K fix
($intervaℓ4[0]) = ($intervaℓ4[0] =~ /\d{2}(\d{2})/);    # Two-digit year
if ($intervaℓ4[0] == 0) {$intervaℓ4[0] = "100";}        # GTDS Y2K fix

$intervaℓ1_ymdhms = join("",@interval1);
$intervaℓ2_ymdhms = join("",@interval2);
$intervaℓ3_ymdhms = join("",@interval3);
$intervaℓ4_ymdhms = join("",@interval4);

# Format end epoch

($end_ymd, $end_hms) = split(" ", $end_epoch);
$end_ymd2 = $end_ymd;
if (length($end_ymd2) == 7) {
    ($end_ymd2) = ($end_ymd2 =~ /(\d{6})$/);  # Take off GTDS Y2K fix
}
($y,$m,$d) = ($end_ymd2 =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($end_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$end_jul = cal2jul($y,$m,$d,$h,$mn,$s);

$span_len = round($end_jul - $start_jul);

# Open log file

open LOGINFO, ">>$logfile" or die "Unable to open $logfile, died";
```

```
open STDERR, ">>&LOGINFO" or die "Unable to redirect stderr, died";

foreach $fh ("STDOUT", "LOGINFO", "STDERR") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\tgenobs.pl: Processing $initfile\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Open and read $initfile

open INITINFO, "<$initfile" or die "Unable to open $initfile, died";

 INITLINE: while (defined($ℓine = <INITINFO>)) {

    $ℓine =~ s/^(\d{5})\s//;
    $initinfo{$1} = [ split(" ",$ℓine) ];
 }

close INITINFO;

################################################################
#                                                              #
# Main program loop (by $catnum)                               #
#                                                              #
################################################################

foreach $catnum (sort keys %initinfo) {

    foreach $fh ("STDOUT","LOGINFO") {
            print $fh "-" x 40,"\n";
            print $fh "  Processing NORAD Catalog \#$catnum\n";
            print $fh "-" x 40,"\n";
        }

    $intℓ_des = $initinfo{$catnum}[0];

    # Write GTDS card file (with or w/o noise parameters)

    $datasim_card = "${catnum}_datasim.gtds";
    $output_file  = "${catnum}_datasim.output";
    $orbit_file   = "${catnum}_ephem.orbit";
    $obs_file     = "${catnum}_datasim.obscard";
```

```
    if ($noise) {
        open(DATASIM_CARD_NOISE, ">$ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card")
or die "Unable to open $ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card, died";
        write DATASIM_CARD_NOISE;
        close DATASIM_CARD_NOISE;
    } else {

        open(DATASIM_CARD, ">$ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card")
or die "Unable to open $ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card, died";
        write DATASIM_CARD;
        close DATASIM_CARD;
    }

    # Make standard data file links

    system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' }; # Remove any GTDS$* links
    system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

##############################################################
#                                                            #
# GTDS BINARY FILE LINKS (change as needed)                  #
#                                                            #
    symlink("$ENV{GTDS_DATA}/sfdir.dat",                      "GTDS\$001");
    symlink("$ENV{GTDS_DATA}/atmosden.dat",                   "GTDS\$002");
    symlink("$ENV{GTDS_DATA}/radarsat_earthfld.dat",          "GTDS\$008");
    symlink("$ENV{GTDS_DATA}/errormsg.dat",        "GTDS\$013");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.mn1950.dat",  "GTDS\$014");
    symlink("$ENV{GTDS_DATA}/newcomb.dat",                    "GTDS\$023");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.timcof.dat",  "GTDS\$038");
    symlink("$ENV{GTDS_DATA}/jrdat_nomn_new.dat",             "GTDS\$075");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.tod1950.dat", "GTDS\$078");
#                                                            #
#                                                            #
##############################################################
    # Inflate .orbit file

    foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Inflating .orbit file. . .\n");
    }
    system "gunzip -v $ENV{ATM_EPHEM}/${ephem_opt}/${orbit_file}.gz";

    # Make job-specific data links

    symlink("$ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card","GTDS\$005");
    symlink("$ENV{ATM_DATASIM}/${datasim_opt}/$output_file", "GTDS\$006");
    symlink("$ENV{ATM_EPHEM}/${ephem_opt}/$orbit_file",     "GTDS\$020");

    # Run GTDS!

    foreach $fh ("STDOUT","LOGINFO") {
```

```
        print $fh "\nUNIX-GTDS\n";
        print $fh "Charles Stark Draper Laboratory\n\n";
        print $fh ("Run started at: ", get_time(), "\n");
    }

    undef $chiℓd_id;

    if ($chiℓd_id = fork) {          # Parent process here

        local $SIG{USR1} = sub {   # Define anonymous sub to kill GTDS

            (my $gtds_id) = split (" ", `ps | grep gtds`);

            foreach $fh ("STDOUT","LOGINFO") {
                print $fh "GTDS run has exceeded $time_limit seconds;\n";
                print $fh "Killing process $gtds_id\n";
            }

            kill 'QUIT', $gtds_id;
        };
        waitpid $chiℓd_id, 0;        # Wait for child process to finish
    }

    elsif (defined $chiℓd_id) {     # Child process here

        $par_id = getppid;

        local $SIG{ALRM} = sub {    # Define local ALRM signal handler
            kill 'USR1', $par_id;  # Send USR1 signal to parent if local alarm goes off
            foreach $fh ("STDOUT","LOGINFO") {
                print $fh "Sending USR1 to $par_id. .\n";
            }
        };

        alarm $time_limit;            # Initialize alarm to go off in
                                     # $time_limit sec
#### RUN GTDS_GRANHOLM.  You must use _DBG version if noise is included,
#     and it won't hurt to do so if noise isn't included.

#          system("$ENV{GTDS_EXE}/gtds.exe");
        system("$ENV{GTDS_EXE}/gtds_dbg.exe");
        alarm 0;                     # Turn off alarm if GTDS finishes before $time_limit
        die "Exiting child process. . .";
    }

    foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Run ended at: ", get_time(), "\n");
    }

    system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };
 # Remove any GTDS$* links
```

```
    system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

    foreach $fh ("STDOUT","LOGINFO") {                        # Recompress .orbit file
        print $fh ("Compressing .orbit file. . .\n");
    }
    system "gzip -v $ENV{ATM_EPHEM}/${ephem_opt}/$orbit_file";

    # Read .output file and create OBSCARD file (FRN 15)

    foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Writing OBSCARD\n");
    }

    open OUTFILE, "<$ENV{ATM_DATASIM}/${datasim_opt}/$output_file" or die
"Unable to open $ENV{ATM_DATASIM}/${datasim_opt}/$output_file, died";
    open OBSCARD, ">$ENV{ATM_DATASIM}/${datasim_opt}/$obs_file" or die
"Unable to open $ENV{ATM_DATASIM}/${datasim_opt}/$obs_file, died";
    printf OBSCARD "OBSCARD \n";

  OBSLINE: while (defined($outℓine = <OUTFILE>)) {
      if ($outℓine =~ m/
          ^\s{0,1}(\d{6,7})\s+
          (\d{5,6}\.\d{3})\s+
          (\w{4})\s+
          (\w+)\s+
          (0\.\d{16})D([-+]\d{2})
          /x) {
          $ymd = $1;
          $hms = sprintf "%010.3f", $2;
          $statid = $3;
          $type = $obstype{$4};
          $observtn = sprintf("%16.14fE%3s", $5, $6);
          if (($type == 4) || ($type == 5)) {
              $observtn = ($observtn*$PI)/180;  # Convert to radians
          }
          write OBSCARD;
      }
      elsif ($outℓine =~ /^\s+RETURN 1/) {
          printf OBSCARD "END     \n";
          last OBSLINE;
      }
  }

    close OUTFILE;
    close OBSCARD;

}

foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Observation Generation Complete.  Exiting. . .\n");
}
```

```
close LOGINFO;

################################################################
#                                                              #
# OBSCARD File Format                                          #
#                                                              #
################################################################
#234567890#234567890#234567890#234567890#234567890#234567890#234567890#234567890

format OBSCARD =
@<<<    @>>      @>>>>>>>@<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
$statid, $type,  $ymd,  $hms,        $observtn,           $observtn
.

################################################################
#                                                              #
#  DATASIM Card Deck Format (without noise)                    #
#                                                              #
################################################################
#234567890#234567890#234567890#234567890#234567890#234567890#234567890#234567890

format DATASIM_CARD =
CONTROL  DATAMGT                                          @<<<<<<<  @>>>>>>>
                                                     $intℓ_des, $catnum
OGOPT
POTFIELD  1 4
END
FIN
CONTROL   DATASIM                                         @<<<<<<<  @>>>>>>>
                                                     $intℓ_des, $catnum
DMOPT
/FLYQ    1 0346  3   338.900          541242.8299       3591947.6900
/PARQ    1 0396  3   347.300          484329.1839       2620600.8719
/EGLQ    1 0399  3     0.380          303420.7790       2734706.5526
/KAEQ    1 0932  3   300.459648       213419.4537       2014359.7376002
END
DCOPT
DSPEA1    1  0    @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 60.0
               $start_ymd,          $start_hms
DSPEA2   20  1  @ @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
         $noise, $end_ymd,            $end_hms
DSPEA3    2  1  0
ELLMODEL  1        6378.135           298.26
/FLYQ   200001
/PARQ   200001
/EGLQ   200001
/KAEQ   200001
/FLYQ   7 1   @>>  60.0             24.0              5.0
              $span_len
```

```
/PARQ   7 1   @>>  60.0             24.0             5.0
          $span_len
/EGLQ   7 1   @>>  60.0             24.0             5.0
          $span_len
/KAEQ   7 1   @>>  120.0            24.0             5.0
          $span_len
/FLYQ   9 1       @>>>>>>@<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
              $start_ymd, $start_hms, $intervaℓ1_ymdhms
/PARQ   9 1       @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
              $intervaℓ1_ymdhms,   $intervaℓ2_ymdhms
/EGLQ   9 1       @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
              $intervaℓ2_ymdhms,   $intervaℓ3_ymdhms
/KAEQ   9 1       @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
              $intervaℓ3_ymdhms,   $intervaℓ4_ymdhms
TRACKELV  3         5.0
END
FIN
.

################################################################
#                                                              #
#  DATASIM Card Deck Format (with noise)                      #
#                                                              #
################################################################
#234567890#234567890#234567890#234567890#234567890#234567890#234567890#234567890

format DATASIM_CARD_NOISE =
CONTROL   DATAMGT                                   @<<<<<<<  @>>>>>>>
                                               $intℓ_des, $catnum
OGOPT
POTFIELD  1 4
END
FIN
CONTROL   DATASIM                                   @<<<<<<<  @>>>>>>>
                                               $intℓ_des, $catnum
DMOPT
/FLYQ    1 0346  3   338.900           541242.8299        3591947.6900
/PARQ    1 0396  3   347.300           484329.1839        2620600.8719
/EGLQ    1 0399  3     0.380           303420.7790        2734706.5526
/KAEQ    1 0932  3   300.459648        213419.4537        2014359.7376002
END
DCOPT
DSPEA1     1  0    @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 60.0
              $start_ymd,          $start_hms
DSPEA2   20  1  @ @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
         $noise, $end_ymd,          $end_hms
DSPEA3     2  1  0
/FLYQ   0 1  4  5    35.0             54.0             54.0
/PARQ   0 1  4  5    48.0             54.0             46.8
/EGLQ   0 1  4  5    30.0             45.0             45.0
/KAEQ   0 1  4  5    4.631            29.5             30.42
```

```
ELLMODEL  1          6378.135            298.26
/FLYQ   200001
/PARQ   200001
/EGLQ   200001
/KAEQ   200001
/FLYQ   7 1   @>>  60.0               24.0               5.0
            $span_len
/PARQ   7 1   @>>  60.0               24.0               5.0
            $span_len
/EGLQ   7 1   @>>  60.0               24.0               5.0
            $span_len
/KAEQ   7 1   @>>  120.0              24.0               5.0
            $span_len
/FLYQ   9 1       @>>>>>>@<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                $start_ymd, $start_hms, $intervaℓ1_ymdhms
/PARQ   9 1       @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                $intervaℓ1_ymdhms,   $intervaℓ2_ymdhms
/EGLQ   9 1       @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                $intervaℓ2_ymdhms,   $intervaℓ3_ymdhms
/KAEQ   9 1       @<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                $intervaℓ3_ymdhms,   $intervaℓ4_ymdhms
TRACKELV  3       5.0
END
FIN
.
```

D.3 distort_bfs.m

Table D.3: distort_bfs.m Fact Sheet

Function	Distorts *a-priori* ballistic factors for testing BFE iteration
Language	MATLAB
Type	MATLAB program
Location	Main AtmoCal directory
Input Files	*initinfo.txt* (in the current directory)
Environment Variables	None.
Data Structure	Requires: None
	Creates: *initinfo_0.txt*
Output Files	*initinfo_0.txt* (in the current directory)
User-Defined Variables	
`m_k`	weight for global distortion factor: nominally 1, set to zero for no global distortion.
`a_k`	weight for individual distortion factor: nominally 1.6.

Commented Code (distort_bfs.m)

```
% distort_bfs.m - Distorts A-priori Ballistic Factors
%
% Author:
%
% George R. Granholm
% 23 May 00
%
% Modified by:
% Sarah E. Bergstrom
% 13 Aug 01
%
% Changed statements to not require the MATLAB stat package (rnd instead of unifrnd).
%
% This program processes the file 'initinfo.txt' and produces the file
% 'initinfo_0.txt'. The non-standard ballistic factors are distorted
% using Eq. 1.15 in DFY 98 Stage 2 of Nazarenko's report.

clear all;
warning off;
more off;

% Set options
% To remove the global weight entirely, set m_k to zero.
% To remove both weights (just mark every tenth satellite as standard), set m_k=a_k=0.

m_k = 1;    % The weight of the bias in all ballistic factors
a_k = 1.6;  % The weight of the random error for each ballistic factor

% Calculate bias for all ballistic factors

xi_1 = rnd(0,1);    % Uniform between 0 and 1

% Open input and output files

initid = fopen('initinfo.txt','r');
outid  = fopen('initinfo_0.txt','w');

line = fgetl(initid);
index = 1;

% Begin main loop

while line ~= -1

  catnum  = line(1:5);
  intl_id = line(7:14);
  bf_orig = str2num(line(16:31));
  area    = line(33:48);
  sigma   = str2num(line(50:65));
```

```
    flag     = line(67:67);
    type     = line(69:70);

    % Skip every tenth object

    if index == 10
        b = 1;
        index = 0;
    else

        % Calculate distortion factor for non-standard objects

        xi_2 = rnd(0,1);      % Uniform between 0 and 1
        b = (1 + m_k*(xi_1 - 0.5) + a_k*(xi_2 - 0.5));

        % Set a-priori sigma, flag

        sigma = sigma*2;
        flag = 'N';

    end

    % Apply distortion factor

    bf_new = bf_orig*b;

    % Write new line to 'initinfo_0.txt'

    fprintf(outid,'%5s %8s %7.10E %16s %7.10E %1s %2s\n', catnum, intl_id, ...
          bf_new, area, sigma, flag, type);

    line = fgetl(initid);
    index = index + 1;

end

fclose(initid);
fclose(outid);
```

D.4 estbfs.pl

Table D.4: estbfs.pl Fact Sheet

Function	Runs GTDS DC many times to create list of observed ballistic factors.
Language	Perl
Type	Subroutine for driver programs *runestbfs.pl* and *bfe_iter.pl*
Location	AtmoCal *include* subdirectory
Input Files	*$initfile* containing table of *a-priori* ballistic coefficient values .output ephemerides files in *$ATM_EPHEM/$ephem_opt* created by *TLE2osc.pl* *.obscard* files in *$ATM_DATASIM/$data_opt* or *$ATM_REALDATA/$data_opt*
Data Structure	Requires: *$ATM_CAL*, *$ATM_EPHEM*, and one of *$ATM_DATASIM* or *$ATM_REALDATA* as appropriate Creates:*$ATM_CAL/$iter_opt*, *$ATM_DC/$iter_opt*
Output Files	*$blfcfile* file containing ballistic coefficients optional (depending on `$keep_data`) tarred and gzipped (see note below) *NSSC#_dc_all.output.tar.gz* GTDS output files in *$ATM_DC/$iter_opt*
Syntax	`&estbfs($start_epoch, $end_epoch, $ephem_opt, $data_opt, $iter_opt, $initfile, $blfcfile, $num_procs, $keep_data, $simulated);`
User-Defined Variables also see driver program listings	
$print_sched	Flag to print schedule of passes
$increment	Amount to shift each DC span by, in days. Equals the length of the correction span, normally 0.125 (3 hrs).
$fit_len	Length of fit span, in days. Normally 3.
$diverge_tol	Allowed length of "sparse" data period and/or period of consecutive divergent runs, in days.
$time_limit	Allowed time (in seconds) for individual GTDS DC runs. Normally 300.

continued on next page...

...continued from previous page	
$rho1_tol	Tolerance for accepting the results of an individual GTDS DC run, based on the GTDS-generated convergence measure ρ.

Note: Tar and gzip are two indispensible Unix utilities for working with large blocks of data. Tar combines several files into one (originally for tape archives, hence the name). Gzip compresses the files, and is extremely effective when working with ascii text files like those produced by GTDS. AtmoCal automatically tars/gzips and untars/gunzips files when needed. To uncompress and split an archive called *filename.tar.gz* by hand, type:

```
prompt% gunzip filename.tar.gz
prompt% tar xvf filename.tar
```

The built-in Unix manuals (type `man tar` and `man gzip`) have much more information on using these utilities.

Commented Code (estbfs.pl)

```
# estbfs.pl - Ballistic Factor Estimator Subroutine
#
# Author:
#
# George R. Granholm
# 09 Apr 00
#
# Edited by Sarah E. Bergstrom to make it a subroutine.
# 19 Oct 01
#

sub estbfs {

#################################################################
#                                                               #
# Header section.  All user-definable variables are in here.   #
#                                                               #
#################################################################

# Add environment-specific path and import modules

BEGIN {
   push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

use Dates;
use Localmath;
use FileHandle;

sub numerically { $a <=> $b }; # To sort in ascending order

$print_sched = 0;          # Flag to print pass schedule; 1 = yes, 0 = no
$increment   = 0.125;      # Shift span for each DC by this much (days)
$fit_len     = 3;          # Length of each fit span (days)
$diverge_tol = 11;         # Allowed length of "sparse" area in data (days)
                           # (or num. of days of consecutive divergent runs)
$time_limit  = 300;        # Allowed duration of GTDS run (secs)
$rho1_tol    = 10;         # Max absolute value of $rho1

my ($start_epoch,$end_epoch,$ephem_opt,$data_opt,$iter_opt,
          $initfiℓe,$bℓfcfiℓe,$num_procs,$keep_data,$simuℓated) = @_;

mkdir "$ENV{ATM_CAL}/${iter_opt}",0777;
mkdir "$ENV{ATM_DC}/${iter_opt}",0777;

#################################################################
#                                                               #
# Preparation Section                                           #
#                                                               #
```

```
#################################################################

# Read and format start epoch

($start_ymd, $start_hms) = split(" ", $start_epoch);
if (length($start_ymd) == 7) {
    ($start_ymd) = ($start_ymd =~ /(\d{6})$/);          # Take off GTDS Y2K fix
}                                                       # for Julian date conversion
($y,$m,$d) = ($start_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($start_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$start_jul = cal2jul($y,$m,$d,$h,$mn,$s);

# Read and format end epoch

($end_ymd, $end_hms) = split(" ", $end_epoch);
if (length($end_ymd) == 7) {
    ($end_ymd) = ($end_ymd =~ /(\d{6})$/);
}
($y,$m,$d) = ($end_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($end_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$end_jul = cal2jul($y,$m,$d,$h,$mn,$s);

# Open and read $initfile

open INITINFO, "$initfile" or die "Unable to open $initfile, died";

 INITLINE: while (defined($ℓine = <INITINFO>)) {

    $ℓine =~ s/^(\d{5})\s//;
    $initinfo{$1} = [ split(" ",$ℓine) ];
 }

close INITINFO;
@initinfo = sort keys %initinfo;

# Open output file so that all processes can access it

open BALLFCTS, ">>$blfcfile" or die "Unable to open $blfcfile, died";

#################################################################
#                                                               #
# Process-Specific Preparation Section                          #
#                                                               #
#################################################################

# Spawn appropriate number of processes

$chiℓd_id[1] = $$;  # Parent process number

 SPAWN: for ($proc_num = 2; $proc_num <= $num_procs; $proc_num++) {
    $chiℓd_id[$proc_num] = fork;        # The parent knows all process nums
```

```
    if ($child_id[$proc_num] == 0) {  # The child only knows the parent's
        $child_id[$proc_num] = $$;    # And its own process num
        last SPAWN; }
 }

# Create subdirectories for each process

if ($$ == $child_id[1]) { $proc_num = 1; }
$iter_opt .= "/run${proc_num}";
mkdir "$ENV{ATM_CAL}/${iter_opt}",0777;
mkdir "$ENV{ATM_DC}/${iter_opt}",0777;
chdir "$ENV{ATM_CAL}/${iter_opt}";
$logfile = "$ENV{ATM_CAL}/${iter_opt}/estbfs.log";

# Open or redirect files

open LOGINFO, ">>$logfile" or die "Unable to open $logfile, died";
open STDERR, ">>&LOGINFO" or die "Unable to redirect stderr, died";
                                          # Redirect STDERR to LOGINFO

foreach $fh ("STDOUT", "LOGINFO", "STDERR", "BALLFCTS") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\testbfs.pl: Processing ${initfile}\n";
    print $fh "\tProcess \# ${proc_num}\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Assign chunk of

for ($i = 1; $i <= $num_procs; $i++) {
    $cutoff[$i] = int(($#initinfo/$num_procs)*$i);
}
$cutoff[0] = -1;

@objects = @initinfo[($cutoff[$proc_num-1]+1)..$cutoff[$proc_num]];

##################################################
#                                          #
# Main Loop by $catnum: initialization for specific sat.   #
#                                          #
##################################################
```

```
# Begin main loop by $catnum

 OBJLOOP: foreach $catnum (@objects) {

    # Initialize variables

    $intℓ_des = $initinfo{$catnum}[0];

    $ephem_output = "$ENV{ATM_EPHEM}/${ephem_opt}/${catnum}_ephem.output";

    if ($simuℓated) {
        $obs_file = "$ENV{ATM_DATASIM}/${data_opt}/${catnum}_datasim.obscard";
        $datasim_output = "$ENV{ATM_DATASIM}/${data_opt}/${catnum}_datasim.output";
    # Read in array of observation times.  If an .obscard file doesn't
    # exist for a particular satellite, skip it.  This allows for there
    # to be satellites in the initinfo.txt (or initinfo_#.txt) list that
    # don't show up in a particular run.

    open SIMOUT, $datasim_output or next OBJLOOP;

    $index = 0;
    while (defined($simℓine = <SIMOUT>)) {
        if ($simℓine =~ /^\s+INTERVAL\s{1,2}\d{1,2}/) {
            foreach $i (1..4) {
                $simℓine = <SIMOUT>;
                if ($simℓine =~ m{          # Match start of pass
                    ^\s+TIME\s1ST\sOB\s\=\s*
                    (\d+)\s+
                    (\d+)\s+
                    (\d+\.\d+)
                    }x) {
                    $obstart_ymd = $1;
                    $obstart_hms = sprintf("%04d", $2) . sprintf("%06.3f",$3);
                }
                elsif ($simℓine =~ m{       # Match end of pass
                    ^\s+TIME\sLAST\sOB\=\s*
                    (\d+)\s+
                    (\d+)\s+
                    (\d+\.\d+)
                    }x) {
                    $obend_ymd = $1;
                    $obend_hms = sprintf("%04d", $2) . sprintf("%06.3f",$3);

                    if (length($obstart_ymd) == 7) {
                        ($obstart_ymd) = ($obstart_ymd =~ /(\d{6})$/);
                    }
                    ($y,$m,$d) = ($obstart_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
                    ($h,$mn,$s) = ($obstart_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
                    $obstart_jul = cal2jul($y,$m,$d,$h,$mn,$s);

                    if (length($obend_ymd) == 7) {
```

```
                        ($obend_ymd) = ($obend_ymd =~ /(\d{6})$/);
                    }
                    ($y,$m,$d) = ($obend_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
                    ($h,$mn,$s) = ($obend_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
                    $obend_jul = cal2jul($y,$m,$d,$h,$mn,$s);

                    if ($obstart_jul != $obend_jul) {      # If pass contains any obs
                        $obstart[$index] = $obstart_jul;   # Store in respective arrays
                        $obend[$index] = $obend_jul;
                        $index++;
                    }
                }
            }
        }
    }
    close SIMOUT;

    # Sort and parse array of observation times

    @obstart = sort numerically @obstart;
    @obend = sort numerically @obend;

    for ($i = 1; $i <= $#obstart; $i++) {     # Eliminate overlap
        if ($obstart[$i] <= $obend[$i-1]) {
            splice(@obstart,$i,1);
            splice(@obend,$i-1,1);
            $i -= 1;
        }
    }

    if ($print_sched) {
        open SIMSCHED, ">$ENV{ATM_DC}/${iter_opt}/${catnum}_sched.txt";
        for ($i = 0; $i <= $#obend; $i++) {
            print SIMSCHED "Span ${i}: $obstart[$i] - $obend[$i]\n";
            @obstart_greg = jul2cal($obstart[$i]);
            $start_greg = join("",@obstart_greg);
            @obend_greg = jul2cal($obend[$i]);
            $end_greg = join("",@obend_greg);
            print SIMSCHED "Gregorian: $start_greg - $end_greg\n";
        }
        close SIMSCHED;
    }

# Since real data doesn't have an .output file, skip that.
# Eventually, reading the .obscard file could be used here (and for
# simulated data as well). For now, it doesn't slow things down much
# (if at all) to assume that every time increment has new data.

    } else {
        $obs_file = "$ENV{ATM_REALDATA}/${data_opt}/${catnum}.obscard";
    }
```

```
    # Calculate mass and area for DC

    $baℓℓ_fact = $initinfo{$catnum}[1];   # Can be "perfect" B value or with error
    $Ax = $initinfo{$catnum}[2];
    $C_d = 2.2;                           # Default for LEO
    $mass = ($Ax*$C_d)/(2*$baℓℓ_fact);    # in kg
    $Ax_km = sprintf("%7.10E",($Ax/1000000));          # Convert to km^2

#   $obs_type = $initinfo{$catnum}[5];    # Stub for real data ...?

    # Initialize DC start and end epochs

    $dc_start_jul = $start_jul;
    $dc_end_jul = $start_jul + $fit_len;
    $div_cnt = 0;                         # Identifies last run that converged
    $run_num = 1;
    $first_run = 1;
    $in_span = 1;
    $have_obs = 1;
    undef %conv_epoch;                    # Hash of converged epochs

#############################################################
#                                                           #
# Main Loop by $catnum continued: Set up for GTDS DC run    #
#                                                           #
#############################################################

    # Begin loop for DC spans

  DCLOOP: while ($in_span) {

      $converged = 0;
      $i = 0;
      if ($first_run) {$have_obs = 1;}
      else {$have_obs = 0};

      # Test if there are any new observations for this object
      if ($simuℓated) {
        TESTOBS: while (!$first_run && ($i <= $#obstart)) {

            if ((($obstart[$i] >= ($dc_end_jul - $increment)) &&
                 ($obstart[$i] <= $dc_end_jul)) or
                (($obend[$i] >= ($dc_end_jul - $increment)) &&
                 ($obend[$i] <= $dc_end_jul))) {
                $have_obs = 1;
                last TESTOBS;
            }
        } continue {$i++;}
      } else {
          $have_obs = 1;
```

```
        }
        next DCLOOP unless ($have_obs);

        # Continue with run

        foreach $fh ("STDOUT","LOGINFO") {
            print $fh "-" x 40,"\n";
            print $fh "  Processing NORAD Catalog \#$catnum\n";
            print $fh "  Process \# $proc_num\n";
            print $fh "  Run number $run_num\n";
        }

        @dc_start_cal = jul2cal($dc_start_jul);

        # Check if $diverge_tol has been exceeded; assign Julian date
        # to look for in .output file and assign epoch & epoch advance date

        if (!$first_run && (($dc_start_jul - (cal2jul(@{ $conv_epoch{$div_cnt} })))
                             > $diverge_tol)) {
            foreach $fh ("STDOUT","LOGINFO") {
                print $fh "Object $catnum not converged for $diverge_tol
consecutive days;\n";
                print $fh "Going to next object\n";
            }
            next OBJLOOP;
        }
        elsif (!$first_run && (($conv_epoch{$div_cnt}[1] ==
                                 $dc_start_cal[1]) &&         # If last epoch that converged
                                ($conv_epoch{$div_cnt}[2] ==
                                 $dc_start_cal[2]))) {        # is on same day as current epoch
            $read_jul = sprintf("%12.4f",$dc_start_jul);
            @epoch = @dc_start_cal;
            $epoch_adv = 0;
        }

        else {  # Either first run or epoch is on different day as last conv. epoch
            $read_jul = sprintf("%12.4f",cal2jul(@dc_start_cal[0..2],0,0,0));
            @epoch = (@dc_start_cal[0..2],0,0,0);
            if (($dc_start_cal[3] == 0) && ($dc_start_cal[4] == 0) &&
                ($dc_start_cal[5] == 0)) {$epoch_adv = 0;}
            else {
                $epoch_adv = 1;
                @epoch_adv = @dc_start_cal;
            }
        }

        # Format $epoch_adv for GTDS

        if ($epoch_adv) {
            ($epoch_adv[0]) = ($epoch_adv[0] =~ /\d{2}(\d{2})$/);
            if ($epoch_adv[0] == 0) {$epoch_adv[0] = 100};
```

```
        $epoch_adv_ymd = join("",@epoch_adv[0..2]);
        $epoch_adv_hms = join("",@epoch_adv[3..5]);
    }
    else {
        $epoch_adv_ymd = "";
        $epoch_adv_hms = "";
    }

    # Format rest of dates for GTDS

    $dc_strt_eph_jul = cal2jul(@dc_start_cal[0..2],0,0,0) + 1; # Beg of nxt day aftr epoch

    @dc_end_cal      = jul2cal($dc_end_jul);
    @dc_strt_eph_cal = jul2cal($dc_strt_eph_jul);
    @dc_end_eph_cal  = jul2cal($dc_start_jul + $diverge_tol);
# Allowd num days w/o convrg

    ($epoch[0])           = ($epoch[0] =~ /\d{2}(\d{2})$/);
    ($dc_start_cal[0])    = ($dc_start_cal[0] =~ /\d{2}(\d{2})$/);
    ($dc_end_cal[0])      = ($dc_end_cal[0] =~ /\d{2}(\d{2})$/);
    ($dc_strt_eph_cal[0]) = ($dc_strt_eph_cal[0] =~ /\d{2}(\d{2})$/);
    ($dc_end_eph_cal[0])  = ($dc_end_eph_cal[0] =~ /\d{2}(\d{2})$/);

    if ($epoch[0] == 0)           {$epoch[0] = "100";}
    if ($dc_start_cal[0] == 0)    {$dc_start_cal[0] = "100";}
    if ($dc_end_cal[0] == 0)      {$dc_end_cal[0] = "100";}
    if ($dc_strt_eph_cal[0] == 0) {$dc_strt_eph_cal[0] = "100";}
    if ($dc_end_eph_cal[0] == 0)  {$dc_end_eph_cal[0] = "100";}

    $epoch_ymd       = join("",@epoch[0..2]);
    $epoch_hms       = join("",@epoch[3..5]);
    $dc_start_ymd    = join("",@dc_start_cal[0..2]);
    $dc_start_hms    = join("",@dc_start_cal[3..5]);
    $dc_end_ymd      = join("",@dc_end_cal[0..2]);
    $dc_end_hms      = join("",@dc_end_cal[3..5]);
    $dc_strt_eph_ymd = join("",@dc_strt_eph_cal[0..2]);
    $dc_strt_eph_hms = "000000.0";
    $dc_end_eph_ymd  = join("",@dc_end_eph_cal[0..2]);
    $dc_end_eph_hms  = "000000.0";

    # Assign input and output file names

    if ($first_run) {$dc_input_file = $ephem_output;}
    else {$dc_input_file =
             "$ENV{ATM_DC}/${iter_opt}/${catnum}_dc_${div_cnt}.output";}

    # Get a-priori elements from appropriate .output file

    open INFILE, $dc_input_file or die "Unable to open $dc_input_file, died";
    $endflag = 0;
```

```
if ($first_run) {   # Then read from EPHEM .output file

  EPHEMLINE: while (defined($inline = <INFILE>)) {

      if ($inline =~ /^ ENTERED ORBINT/) {
          $endflag = 1;
      }
      elsif ($endflag && ($inline =~ /^ DATE.*JULIAN DATE = $read_jul/ )) {
          while (defined($inline = <INFILE>)) {
              if ($inline =~ m{
                  ^\sX\s*(-*\d+\.\d+)
                  \s*Y\s*(-*\d+\.\d+)
                  \s*Z\s*(-*\d+\.\d+)
                  \s*DX\s*(-*\d+\.\d+)
                  \s*DY\s*(-*\d+\.\d+)
                  \s*DZ\s*(-*\d+\.\d+)
                  }x ) {
                  @aprioris = ($1,$2,$3,$4,$5,$6);
                  last EPHEMLINE;
              }
          }
      }
  }
}

else {              # Read from appropriate DC .output file

  DCLINE: while (defined($inline = <INFILE>)) {

      if ($inline =~ /^ DATE.*JULIAN DATE = $read_jul/ ) {
          while (defined($inline = <INFILE>)) {
              if ($inline =~ m{
                  ^\sX\s*(-*\d+\.\d+)
                  \s*Y\s*(-*\d+\.\d+)
                  \s*Z\s*(-*\d+\.\d+)
                  \s*DX\s*(-*\d+\.\d+)
                  \s*DY\s*(-*\d+\.\d+)
                  \s*DZ\s*(-*\d+\.\d+)
                  }x ) {
                  @aprioris = ($1,$2,$3,$4,$5,$6);
                  last DCLINE;
              }
          }
      }
  }
}

close INFILE;

# Write GTDS DC card file
```

```
    $dc_card = "${catnum}_dc_${run_num}.gtds";
    $dc_output_file = "${catnum}_dc_${run_num}.output";

    my $file="$ENV{ATM_DC}/${iter_opt}/$dc_card";
    if ($simulated) {
        open (DC_CARD_SIM, ">$file") || die("Unable to open $file, $!");
        write DC_CARD_SIM;
        close DC_CARD_SIM;
    } else {
        open (DC_CARD_REAL, ">$file") || die("Unable to open $file, $!");
        write DC_CARD_REAL;
        close DC_CARD_REAL;
    }

    # Make standard data file links.  (Multiple options for some
    # files are included - comment out all but one.)

    system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };
# Remove any GTDS$* links
    system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

    symlink("$ENV{GTDS_DATA}/sfdir.dat",                      "GTDS\$001");
    symlink("$ENV{GTDS_DATA}/atmosden.dat",                   "GTDS\$002");
    symlink("$ENV{GTDS_DATA}/radarsat_earthfld.dat",          "GTDS\$008");
    symlink("$ENV{GTDS_DATA}/errormsg.dat",         "GTDS\$013");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.mn1950.dat",  "GTDS\$014");
    symlink("$ENV{GTDS_DATA}/newcomb.dat",                    "GTDS\$023");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.timcof.dat",  "GTDS\$038");
    symlink("$ENV{GTDS_DATA}/jrdat_nomn.dat",                 "GTDS\$075");
#     symlink("$ENV{GTDS_DATA}/jrdat_nomn_new.dat",              "GTDS\$075");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.tod1950.dat", "GTDS\$078");

    # Make job-specific data links

    symlink("$ENV{ATM_DC}/${iter_opt}/$dc_card",              "GTDS\$005");
    symlink("$ENV{ATM_DC}/${iter_opt}/$dc_output_file",        "GTDS\$006");
    symlink("$obs_file",   "GTDS\$015");
# $obs_file is set up appropriately in the scheduling section.

#############################################################
#                                                           #
# Main Loop by $catnum continued: Run GTDS DC               #
#                                                           #
#############################################################

    foreach $fh ("STDOUT","LOGINFO") {
        print $fh "  Epoch ${dc_start_ymd} ${dc_start_hms}\n";
        print $fh "-" x 40,"\n";
        print $fh "\nUNIX-GTDS\n";
        print $fh "Charles Stark Draper Laboratory\n\n";
        print $fh ("Run started at: ", get_time(), "\n");
```

```
        }

        undef $grandchiℓd_id;

        if ($grandchiℓd_id = fork) {         # Parent or first-generation child process

            local $SIG{USR1} = sub {         # Define anonymous sub to kill GTDS

# The next several lines contain debug stuff. Uncomment if having problems
# with process control.
#
#
#                print “Process ${proc_num}\n”;
#                foreach $line (
#                    print $line;
#                }
#                print“${proc_num} Grandchild $grandchild_id\n”;
                $ps = `ps -f | grep -v grep | grep $grandchiℓd_id | grep gtds`;
#                print “${proc_num} $ps\n”; #another debug lines
                ($uid,$gtds_id) = split (" ",$ps);
                foreach $fh ("STDOUT","LOGINFO") {
                    print $fh "GTDS run has exceeded time limit;\n";
                    print $fh "Killing process $gtds_id\n";
                }
                kill 'QUIT', $gtds_id;

            };
            waitpid $grandchiℓd_id, 0;        # Wait for child process to finish
        }

        elsif (defined $grandchiℓd_id) {      # Grandchild process

            local $SIG{ALRM} = sub {    # Define local ALRM signal handler
                kill 'USR1', $chiℓd_id[$proc_num];  # Send USR1 signal to parent
                                                    # if local alarm goes off
                foreach $fh ("STDOUT","LOGINFO") {
# DEBUG code, again
                    print $fh "${proc_num} Sending USR1 to $child_id[$proc_num]. .\n";
                }
            };

            alarm $time_limit;          # Initialize alarm to go off in $time_limit sec
            system("$ENV{GTDS_EXE}/gtds.exe");
            alarm 0;                    # Turn off alarm if GTDS finishes before $time_limit
            die "Exiting grandchild process. . .\n";
        }

        foreach $fh ("STDOUT","LOGINFO") {
            print $fh ("Run ended at: ", get_time(), "\n");
        }
################################################################
```

```
#                                                          #
# Main Loop by $catnum continued: Process GTDS DC output       #
#                                                          #
############################################################

        # Read GTDS outfile.
        # Test if run converged; set flags and read $rho1, $ht_per

        open OUTFILE, "GTDS\$006" or die "Couldn't open GTDS\$006, $!\n";
        $ht_per = 0;
        $rho1 = 0;

      OUTLINE: while (defined($outline = <OUTFILE>)) {
          if (!$converged && ($outline =~ /^\s+\*{5} DC CONVERGED/)) {
              $converged = 1;
              $div_cnt = $run_num;
              $conv_epoch{$run_num} = [ @dc_start_cal ];
              if ($conv_epoch{$run_num}[0] > 99) {  # Remove GTDS formatting if necessary
                  $conv_epoch{$run_num}[0] -= 100;
                  $conv_epoch{$run_num}[0] = sprintf("%02d", $conv_epoch{$run_num}[0]);
              }
              if ($first_run) {$first_run = 0;}
          }
          if ($converged) {
              if (!$ht_per && ($outline =~ /^\s+HT\. OF PERIFOCUS\s+(\d+\.\d+)\s/)) {
                  $ht_per = $1;
              }
              if ($ht_per && ($outline =~ s/^\s+AERO VARIATION
\(RHO1\)\s+=\s*(-*\d\.\d{8})D([+-]\d{2})/$1e$2/)) {
                  $rho1 = $outline;

                  # Throw out $rho1 values that are obviously not valid

                  if (abs($rho1) > $rho1_tol) {$converged = 0;}
                  last OUTLINE;
              }
          }
      }

        close OUTFILE;

        # If converged, write to log, write line to ballfcts.txt

        if ($converged) {
            foreach $fh ("STDOUT","LOGINFO") {
                print $fh ("Run converged with rho1 = $rho1.\n");
            }
            $attrib_time = ($dc_start_jul + $dc_end_jul)/2;
            $C_d_est = $C_d*(1+$rho1);
            $B_est = ($C_d_est*$Ax)/(2*$mass);
            printf BALLFCTS "%5s %12.4f %7.10E %7.10E\n", $catnum, $attrib_time,
```

```
$B_est, $ht_per;
      }

      else {
          foreach $fh ("STDOUT","LOGINFO") {
              print $fh ("Run diverged or bad rho1: $rho1\n");
          }
      }

###############################################################
#                                                             #
# Main Loop by $catnum continued: finish loop and clean up   #
#                                                             #
###############################################################

      system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };
# Remove any GTDS$* links
      system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };     # Remove any temp files
      system q { /usr/bin/tcsh -c 'rm core >& /dev/null' };      # Remove core

    # Increment counters and check for end-of-run.

   } continue {

      $dc_start_jul += $increment;
      $dc_end_jul += $increment;
      $run_num += 1;
      if ($dc_end_jul > $end_jul) {$in_span = 0;}
   }

 } continue {

    # Deal with large data files generated by DC runs by zipping or
    # deleting, depending on the value of the $keep_data input variable.

    if ($keep_data == 1) {
     foreach $fh ("STDOUT","LOGINFO") {
         print $fh ("Compacting .output and .gtds files. . .\n");
     }
     system qq! tar cf $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_all.output.tar \\
                 $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_\[0-9\]\*.output;
               gzip -v $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_all.output.tar;
               rm $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_\[0-9\]\*.output; !;
     system qq! tar cf $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_all.gtds.tar \\
                 $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_\[0-9\]\*.gtds;
               gzip -v $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_all.gtds.tar;
               rm $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_\[0-9\]\*.gtds; !;
    }
    else {
         foreach $fh ("STDOUT","LOGINFO") {
           print $fh ("Deleting .output and .gtds files. . .\n");
```

```
          }
          system qq(rm \\
                  $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_\[0-9\]\*.output;
                  rm $ENV{ATM_DC}/${iter_opt}/${catnum}_dc_\[0-9\]\*.gtds;);
     }
 }

################################################################
#                                                              #
# Other cleanup                                                 #
#                                                              #
################################################################

# If parent, wait for slow-finishing children processes

if ($proc_num == 1) {
   for ($i = 2; $i <= $num_procs; $i++) {
      waitpid $chiℓd_id[$i],0;
      foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Ended process $i at or before: ", get_time(), ".\n");
      }
   foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Ballistic Factor Estimation Complete.\n");
   }
   }
}
else {
   print LOGINFO ("Process $proc_num finished at: ", get_time(),".\n");
   close LOGINFO;
   close BALLFCTS;
   die ("Ending process $proc_num");
      # This is probably not the cleanest way to end child processes,
      # but it works.

}

close LOGINFO;
close BALLFCTS;
return;

################################################################
#                                                              #
# DC Card Deck Format: Sim Data                                      #
#                                                              #
################################################################
#234567890#234567890#234567890#234567890#234567890#234567890#234567890#234567890

format DC_CARD_SIM =
CONTROL   DC                                             @<<<<<<<<  @>>>>>>>>
                                                    $intℓ_des, $catnum
EPOCH             @<<<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<<
```

```
@>>>>>>@<<<<<<<<<<<
                 $epoch_ymd,        $epoch_hms, $epoch_adv_ymd, $epoch_adv_hms
ELEMENT1  1  1  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                 $apriori s[0],        $aprioris[1],       $aprioris[2]
ELEMENT2          @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                 $aprioris[3],         $aprioris[4],       $aprioris[5]
ORBTYPE   2  1  1 60.
OBSINPUT  5       @>>>>>>@<<<<<<<<<<<< @>>>>>>@<<<<<<<<<<<
            $dc_start_ymd, $dc_start_hms, $dc_end_ymd, $dc_end_hms
DMOPT
/FLYQ    1 0346  3   338.900            541242.8299        3591947.6900
/PARQ    1 0396  3   347.300            484329.1839        2620600.8719
/EGLQ    1 0399  3     0.380            303420.7790        2734706.5526
/KAEQ    1 0932  3   300.459648          213419.4537        2014359.7376002
END
DCOPT
/FLYQ    0 1   4   5    35.0            54.0               54.0
/PARQ    0 1   4   5    48.0            54.0               46.8
/EGLQ    0 1   4   5    30.0            45.0               45.0
/KAEQ    0 1   4   5    4.631           29.5               30.42
ELLMODEL  1          6378.135           298.26
/FLYQ   200001
/PARQ   200001
/EGLQ   200001
/KAEQ   200001
TRACKELV  3          5.0
EDIT              3.0
PRINTOUT  1     1
CONVERG  25  6     1.0D-4
END
OGOPT
DRAG       1         1
ATMOSDEN          1
DRAGPAR    3  0     @<<<<<<<<<<<<<<<<<<<
                 $C_d
DRAGPAR    1
SCPARAM             @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
                 $Ax_km,               $mass
MAXDEGEQ  1         4
MAXORDEQ  1         4
MAXDEGVE  1         4
MAXORDVE  1         4
POTFIELD  1  4
SOLRAD     1         1.0
END
FIN
CONTROL   EPHEM                          OUTPUT            @<<<<<<<  @>>>>>>>
                                                        $intℓ_des, $catnum
OUTPUT     1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<  10800.0
```

```
                 $dc_strt_eph_ymd, $dc_strt_eph_hms
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN         1
DRAG      1      1
DRAGPAR   3  0    2.2
SCPARAM          @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                 $Ax_km,           $mass
POTFIELD  1  4
MAXDEGEQ  1       4.0
MAXORDEQ  1       4.0
SOLRAD    1       1.0
END
FIN
CONTROL   EPHEM                      OUTPUT          @<<<<<<<  @>>>>>>>
                                                 $intℓ_des, $catnum
OUTPUT    1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<  86400.0
                 $dc_end_eph_ymd, $dc_end_eph_hms
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN         1
DRAG      1      1
DRAGPAR   3  0    2.2
SCPARAM          @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                 $Ax_km,           $mass
POTFIELD  1  4
MAXDEGEQ  1       4.0
MAXORDEQ  1       4.0
SOLRAD    1       1.0
END
FIN
.
################################################################
#                                                 #
# DC Card Deck Format: Real Data                        #
#                                                 #
################################################################
#234567890#234567890#234567890#234567890#234567890#234567890#234567890#234567890

format DC_CARD_REAL =
CONTROL   DC                                     @<<<<<<<  @>>>>>>>
                                                 $intℓ_des, $catnum
EPOCH           @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@>>>>>>@<<<<<<<<<<<
                 $epoch_ymd,      $epoch_hms, $epoch_adv_ymd, $epoch_adv_hms
ELEMENT1  1  1  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                 $aprioris[0],      $aprioris[1],      $aprioris[2]
ELEMENT2         @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
@<<<<<<<<<<<<<<<<<<<
                 $aprioris[3],      $aprioris[4],      $aprioris[5]
```

```
ORBTYPE   2  1  1 60.
OBSINPUT  5       @>>>>>>>@<<<<<<<<<<<<< @>>>>>>>@<<<<<<<<<<<<
            $dc_start_ymd, $dc_start_hms, $dc_end_ymd, $dc_end_hms
DMOPT
/FLYF   1 0346  3   388.900            541242.8299        3591947.6900
/PPWQ    1 0388  3    82.780        390809.3764       2383857.3529
/PPWF    1 0389  3    82.780        390810.1652       2383856.8705
/EGLQ    1 0399  3     0.380           303420.7790        2734706.5526
/NAVQ    1 0745  3   305.300        333314.3388       2611413.5272
END
DCOPT
/FLYF   0 1  4  5    35.0                54.0                54.0
/PPWQ    0 1  4  5    40.0            36.0               36.0
/PPWF    0 1  4 5    40.0            36.0               36.0
/EGLQ    0 1  4  5    30.0               45.0                45.0
/NAVQ 0 1  4 5  1979.0               64.8               122.4
ELLMODEL  1        6378.135            298.26
/FLYF   200001
/PPWQ    200001
/PPWF    200001
/EGLQ    200001
/NAVQ    200001
TRACKELV  3        5.0
EDIT                3.0
PRINTOUT  1     1
CONVERG  25  6     1.0D-4
END
OGOPT
DRAG       1         1
ATMOSDEN         1
DRAGPAR    3  0    @<<<<<<<<<<<<<<<<<<<
                $C_d
DRAGPAR    1
SCPARAM           @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
                $Ax_km,                $mass
MAXDEGEQ  1        4
MAXORDEQ  1        4
MAXDEGVE  1        4
MAXORDVE  1        4
POTFIELD  1  4
SOLRAD     1        1.0
END
FIN
CONTROL   EPHEM                        OUTPUT            @<<<<<<<  @>>>>>>>
                                               $intℓ_des, $catnum
OUTPUT     1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<  10800.0
                $dc_strt_eph_ymd, $dc_strt_eph_hms
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN         1
DRAG       1         1
```

```
DRAGPAR   3  0    2.2
SCPARAM          @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
             $Ax_km,              $mass
POTFIELD  1  4
MAXDEGEQ  1       4.0
MAXORDEQ  1       4.0
SOLRAD    1       1.0
END
FIN
CONTROL   EPHEM                   OUTPUT           @<<<<<<<<  @>>>>>>>>
                                         $intℓ_des, $catnum
OUTPUT    1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 86400.0
             $dc_end_eph_ymd, $dc_end_eph_hms
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN        1
DRAG      1     1
DRAGPAR   3  0    2.2
SCPARAM          @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
             $Ax_km,              $mass
POTFIELD  1  4
MAXDEGEQ  1       4.0
MAXORDEQ  1       4.0
SOLRAD    1       1.0
END
FIN
.
} # closes sub estbfs

1; # returns true for require statement in dr_atmcal.pl
```

D.5 calcvars.pl

Table D.5: calcvars.pl Fact Sheet

Function	Calculates and predicts atmospheric density corrections. Optionally calculates improved ballistic factors.
Language	Perl
Type	Subroutine
Location	AtmoCal *include* subdirectory
Input Files	*initinfo.txt* and *ballfcts.txt* or *initinfo_(#-1).txt* and *ballfcts_#.txt*
Environment Variables	`$ATM_CAL` must be set.
Data Structure	Requires: *$ATM_CAL/$iter_opt* must exist. Creates: None.
Output Files	*calcvars.log* in *$ATM_CAL/$iter_opt*, *ballfcts.txt.sort*, *array_tmp.txt*, and *jac_densvars.txt*, or *ballfcts_#.txt.sort*, *array_tmp_#.txt*, *jac_densvars_#.txt* and *initinfo_#.txt* in *$ATM_CAL/$dc_opt*
Syntax	`&calcvars($dc_opt, $iter_opt, $initfile, $outfile, $blfcfile, $tmpfile, $dnsvarfile, $iterate, $do_global);`
User-Defined Variables	
`$tau_min`	minimum length of correction span (nominally .125 days, which is 3 hours).
`$min_num_k`	minimum number of ballistic factor estimations required per span.
`$increment`	amount to lengthen `$tau_min` by if necessary.

Commented Code

```
# calcvars.pl - Density Variation Calculator Program
#
# Author:
#
# George R. Granholm
# 30 Apr 00
#
#

sub calcvars {

##################################################################
#                                                                #
# Header section                                                 #
#                                                                #
##################################################################

# Add environment-specific path and import modules

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

use Dates;
use Localmath;
use FileHandle;
use MatrixReal;

# Read input options

my ($dc_opt,$iter_opt,$initfiℓe,$initnew,$bℓfcfiℓe,$tmpfiℓe,$dnsvarfiℓe,$bfe_iterate,$do_global) = @_;

# Set options & variables

$ℓogfiℓe    = "$ENV{ATM_CAL}/${iter_opt}/calcvars.log";
$sortdfiℓe  = $bℓfcfiℓe . ".sort";

$tau_min    = .125;     # Minimum length of each span j (days)
$min_num_k  = 35;
            # Minimum number of ballistic factor estimation per span j
$increment  = .125;     # Increment to add to $tau_min

# Define f_1 and f_2 (linear density variation functions)

sub f_1 {
    return "1";
}

sub f_2 {
```

```
    my $h = shift(@_);
    my $vaℓue = ($h - 400)/200;
    return $vaℓue;
}

# Open or redirect files

open LOGINFO, ">>$logfile" or die "Unable to open $logfile, died";
open STDERR, ">>&LOGINFO" or die "Unable to redirect stderr, died";
                        # Redirect STDERR to LOGINFO
open TMPFILE, ">$tmpfile" or die "Unable to open $tmpfile, died";

foreach $fh ("STDOUT", "LOGINFO", "STDERR", "TMPFILE") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\tcalcvars.pl: Processing ${initfile}\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Open and read $initfile

$std_count = 0;
$nonstd_count = 0;

open INITINFO, "$initfile" or die "Unable to open $initfile, died";

 INITLINE: while (defined($ℓine = <INITINFO>)) {

    $ℓine =~ s/^(\d{5})\s//;
    $initinfo{$1} = [ split(" ",$ℓine) ];
    if ($initinfo{$1}[4] eq 'S') {
        $initinfo_std{$1} = $initinfo{$1};
        $std_count++;
    }
    else {
        $initinfo_nonstd{$1} = $initinfo{$1};
        $nonstd_count++;
    }

 }

close INITINFO;

```

```
# Sort $blfcfile by attribution time

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "Sorting ballistic factors by attribution time. . .\n";
}

# A test should be put to make sure $blfcfile exists.  The following
# doesn't work properly, though.  (it always dies)
# unless (-e $blfcfile) {die "$blfcfile does not exist, died";}

system "sort -nk2,2 $blfcfile > $sortdfile";

# Read $sortdfile into

undef $ℓine;
undef @blfcs;
open BLFCFILE, "$sortdfile" or die "Unable to open $sortdfile, died";
$index = 0;

while (defined($ℓine = <BLFCFILE>)) {
    $bℓfcs[$index] = [ split(" ",$ℓine) ];
    $index++;
}

close BLFCFILE;

$j = 0;
$span_time[0] = $bℓfcs[0][1];
$end_time = $bℓfcs[$#blfcs][1];
$i_save = 0;

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "Building $tmpfile. . .\n";
}

# Begin main loop

while ($span_time[$j] <= $end_time) {

    $tau[$j] = $tau_min;

  COUNTBLFCS: for ($i = $i_save; $i <= $#blfcs; $i++) {
      if ($bℓfcs[$i][1] < ($span_time[$j] + $tau[$j])) {
          # Put in test for negative ball. factors here??
          $temp_array[$i−$i_save] = $bℓfcs[$i];
      }
      else {
          $i_save = $i;
          last COUNTBLFCS;
      }
  }
```

```
    # Test for enough estimations in span

    if ($#temp_array < $min_num_k) {
        $tau[$j] += $increment;
        $i_save  -= ($#temp_array+1);
        goto COUNTBLFCS;
    }

    else {

        # Define arrays for MATLAB input

        foreach $fh ("STDOUT", "LOGINFO") {
            print $fh "Span $j: [$span_time[$j],",
            $span_time[$j] + $tau[$j],")\n";
        }

        undef @F;
        undef @a;
        undef @P;

        print TMPFILE "$span_time[$j] ", ($#temp_array+1), "\n";

        for ($n = 0; $n <= $#temp_array; $n++) {
            $F[$n] = [ f_1($temp_array[$n][3]), f_2($temp_array[$n][3]) ];
            $a[$n] = ($temp_array[$n][2]/$initinfo{$temp_array[$n][0]}[1]) - 1;
            $P[$n] = 1/$initinfo{$temp_array[$n][0]}[3];
            print TMPFILE "$F[$n][0] $F[$n][1] $a[$n] $P[$n]\n";
        }
    }

} continue {
    undef @temp_array;
    $j++;
    $span_time[$j] = $span_time[$j-1] + $tau[$j-1];
}

close TMPFILE;

# Run MATLAB to calculate density variations

system q { /usr/bin/tcsh -c 'rm startup.m >& /dev/null' }; # Remove startup.m
symlink("$ENV{ATM_CAL}/calc_b.m","startup.m");             # Make link
system 'matlab';                                           # Run calc_b
system q { /usr/bin/tcsh -c 'rm startup.m >& /dev/null' }; # Remove startup.m

# Load in densvars.txt

$index = 0;
```

```
open DENSVARS, "<$dnsvarfile";

while (defined($densℓine = <DENSVARS>)) {
    $densvars[$index] = [ split(" ",$densℓine) ];
    $index++;
}

close DENSVARS;

if ($bfe_iterate) {
# Calculate residuals for estimation of "true" ballistic factors

    foreach $fh ("STDOUT", "LOGINFO") {
        print $fh ("Calculating \"true\" ballistic factors. . .\n");
    }

    $densvars[$index] = $densvars[$index−1];
    $densvars[$index][0] += $tau_min*2;
    $j = 0;
    undef %resid_sum;
    undef %height_avg;
    undef %N;
    undef %Delta_part;
    undef %Delta_sqrd;
    undef %Q;

    for ($i = 0; $i <= $#blfcs; $i++) {

        if ($bℓfcs[$i][1] >= $densvars[$j+1][0]) {
            $j++;
        }

        $resid_sum{$bℓfcs[$i][0]} += (1 + $densvars[$j][1]*f_1($bℓfcs[$i][3]) +
                                     $densvars[$j][2]*f_2($bℓfcs[$i][3]));
        $height_avg{$bℓfcs[$i][0]} += $bℓfcs[$i][3];
        $N{$bℓfcs[$i][0]} += 1;
        $Deℓta_part{$bℓfcs[$i][0]} += ($bℓfcs[$i][2]/$initinfo{$bℓfcs[$i][0]}[1]) − 1;
        $Deℓta_sqrd{$bℓfcs[$i][0]} += (($bℓfcs[$i][2]/$initinfo{$bℓfcs[$i][0]}[1]) − 1
                                      − $densvars[$j][1]*f_1($bℓfcs[$i][3])
                                      − $densvars[$j][2]*f_2($bℓfcs[$i][3]))**2;
    }

    if ($do_global == 1) {

        # Organize data for first-stage corrections

        undef $F;
        undef $F_trans;
        undef $x;
        undef $a;
        $F = new Math::MatrixReal($std_count,2);
```

```
  $F_trans = new Math::MatrixReal(2,$std_count);
  $x = new Math::MatrixReal($std_count,1);
  $a = new Math::MatrixReal(2,1);
  $row = 1;

STDSAT: foreach $catnum (sort keys %initinfo_std) {

    next STDSAT unless ($N{$catnum});

    # Calculate Q factor and average height for each standard sat.

    $height_avg{$catnum} *= 1/$N{$catnum};
    $Q{$catnum} = ($Deℓta_part{$catnum} - $resid_sum{$catnum} + $N{$catnum})/$N{$catnum};

    # Store Eq.(2.36) calculations and Q values in $F and $a matrices

    $f1 = 1;
    $f2 = ($height_avg{$catnum}-200)/200;
    $F->assign($row,1,$f1);
    $F->assign($row,2,$f2);
    $x->assign($row,1,$Q{$catnum});

    $row++;

}

  # Calculate a[1] and a[2] in Eq.(2.36) using least squares

  $F_trans->transpose($F);
  $prod = $F_trans*$F;
  $LR_prod = $prod->decompose_LR();
  $prod_inv = $LR_prod->invert_LR();
  $prod2 = $F_trans*$x;

  $a = $prod_inv*$prod2;

  $a1 = $a->element(1,1);
  $a2 = $a->element(2,1);
  foreach $fh ("STDOUT", "LOGINFO") {
      print $fh "A1 = ${a1} and A2 = ${a2}\n";
  }

  # Apply 1st stage correction factor in Eq.(2.37)

  foreach $fh ("STDOUT", "LOGINFO") {
      print $fh "Calculating Global Correction Factors. . .\n";
  }

NONSTDSAT: foreach $catnum (sort keys %initinfo_nonstd) {
    if ($N{$catnum}) {
```

```
            print LOGINFO "${N{$catnum}} observations for satellite ${catnum}\n";
        }
        else {
            print LOGINFO "No observations for satellite ${catnum}\n";
        }
        next NONSTDSAT unless ($N{$catnum});
        $height_avg{$catnum} *= 1/$N{$catnum};
        $Q{$catnum} = ($Deℓta_part{$catnum} - $resid_sum{$catnum} +
$N{$catnum})/$N{$catnum};
        $xi = 1 + ($a1 + $a2*(($height_avg{$catnum} - 200)/200))*
$N{$catnum}/$resid_sum{$catnum};
        $initinfo{$catnum}[1] *= $xi;
        print LOGINFO "${xi}, ${catnum}\n";

    }

      # Apply second-stage correction factor in Eq.(2.32)

      # First recalculate residuals with corrected “true” ballistic factors
      # Note: $resid_sum and $N do not change

      undef %Delta_part;
      undef %Q;

      for ($i = 0; $i <= $#blfcs; $i++) {
          $Deℓta_part{$bℓfcs[$i][0]} += ($bℓfcs[$i][2]/$initinfo{$bℓfcs[$i][0]}[1]) - 1;
      }

      # Now calculate correction factor
      foreach $fh ("STDOUT","LOGINFO") {
          print $fh "Calculating individual correction factors. . .\n";
      }
    NONSTDSAT2: foreach $catnum (sort keys %initinfo_nonstd) {

        next NONSTDSAT2 unless ($N{$catnum});

        $Q{$catnum} = ($Deℓta_part{$catnum} - $resid_sum{$catnum} +
$N{$catnum})/$N{$catnum};
        $psi = 1 + $Q{$catnum}*$N{$catnum}/$resid_sum{$catnum};
        $initinfo{$catnum}[1] *= $psi;
        print LOGINFO "${psi}, ${catnum}\n";
    }
  }

  else { # If not first iteration

  # Calculate second-stage correction factor only

    NONSTDSAT3: foreach $catnum (sort keys %initinfo_nonstd) {
        if ($N{$catnum}) {
            print LOGINFO "${N{$catnum}} observations for satellite ${catnum}\n";
```

```
          }

          next NONSTDSAT3 unless ($N{$catnum});
          $Q{$catnum} = ($Deℓta_part{$catnum} - $resid_sum{$catnum} +
$N{$catnum})/$N{$catnum};
          $psi = 1 + $Q{$catnum}*$N{$catnum}/$resid_sum{$catnum};
          print LOGINFO "Psi = ${psi} for satellite ${catnum}\n";
          $initinfo{$catnum}[1] *= $psi;
      }
    }

# Calculate variance using ML estimate:

  SAT: foreach $catnum (sort keys %initinfo) {

      next SAT unless ($N{$catnum});
      $initinfo{$catnum}[3] = $Deℓta_sqrd{$catnum}/$N{$catnum};
  }

# Write new initinfo.txt

    open INITNEW, ">$initnew";

    foreach $catnum (sort keys %initinfo) {
        printf INITNEW "%5s %8s %7.10E %7.10E %7.10E %1s %2d\n", $catnum,
$initinfo{$catnum}[0],$initinfo{$catnum}[1],$initinfo{$catnum}[2],$initinfo{$catnum}[3],
$initinfo{$catnum}[4],
        $initinfo{$catnum}[5]
        }

    close INITNEW;
} # End BFE iteration

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh ("\tJob ended at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

close LOGINFO;

} # End subroutine

1;  # Returns true for require statement in dr_atmcal.pl
```

D.6 Drivers for *estbfs.pl* and *calcvars.pl*

Table D.6: runestbfs.pl Fact Sheet

Function	Runs the *estbfs.pl* subroutine once.
Language	Perl
Type	Main Program
Location	Main AtmoCal Directory
Input Files	None (see *estbfs.pl* fact sheet)
Environment Variables	`$ATM_CAL` and `$ATM_DC` must be properly set.
Data Structure	Requires: *$ATM_DC* must exist.
	Creates: *$ATM_DC/$dc_opt*
Output Files	None (see *estbfs.pl* fact sheet)
User-Defined Variables Also see variables for *estbfs.pl*	
`$start_epoch`	Beginning of fit window
`$end_epoch`	End of fit window
`$ephem_opt`	subdirectory in *$ATM_EPHEM* for initial ephemerides created by *TLE2osc.pl*
`$data_opt`	subdirectory containing OBSCARD data (in *$ATM_DATASIM* for simulated observations, and in *$ATM_REALDATA* for real observations)
`$dc_opt`	subdirectory in *$ATM_DC* and *$ATM_CAL* for data and log files
`$num_procs`	Number of processes to spawn in *estbfs.pl* - must be at least 1
`$keep_data`	Save large data files? 1=yes, 0=no.
`$iterate`	is this part of a BFE iteration? 1=yes, 0=no.
`$iter`	iteration number (is irrelevant if `$iterate`=0)
`$simulated`	Is this simulated data? 1=yes, 0=no.

Table D.7: runcalcvars.pl Fact Sheet

Function	Runs the *calcvars.pl* subroutine once.
Language	Perl
Type	Main Program
Location	Main AtmoCal directory
Input Files	None (see *calcvars.pl* fact sheet)
Environment Variables	Requires: `$ATM_CAL` properly set.

continued on next page...

...continued from previous page	
	Creates: `$ATM_ITER`, `$ATM_TMP`, `$ATM_DNSVAR`, `$ATM_ITER_OPT` for *calc_b.m* to use.
Data Structure	Requires: None (see *calcvars.pl* fact sheet)
	Creates: None (see *calcvars.pl* fact sheet)
Output Files	None (see *calcvars.pl* fact sheet)
User-Defined Variables	
`$dc_opt`	subdirectory in *$ATM_DC* and *$ATM_CAL* for data and log files (normally the same as that used in the preceding *runestbfs.pl* run.
`$iterate`	is this part of a BFE iteration? 1=yes, 0=no.
`$iter`	iteration number (is irrelevant if `$iterate`=0)
`$do_global`	apply global ballistic factor correction? 1=yes, 0=no. (irrelevant if `$iterate`=0.)

Table D.8: bfe_iter.pl Fact Sheet

Function	Runs *estbfs.pl* and *calcvars.pl* repeatedly for improving BFEs.
Language	Perl
Type	Main Program
Location	Main AtmoCal directory
Input Files	None (see *estbfs.pl* and *calcvars.pl* fact sheets)
Environment Variables	Requires: `$ATM_CAL`, `$ATM_DC` must be properly set.
	Creates: `$ATM_ITER`, `$ATM_TMP`, `$ATM_DNSVAR`, `$ATM_ITER_OPT` for *calc_b.m* to use.
Data Structure	
	Creates:
Output Files	None (see *estbfs.pl* and *calcvars.pl* fact sheets)
User-Defined Variables	
`$start_epoch`	Beginning of fit window
`$end_epoch`	End of fit window
`$ephem_opt`	subdirectory in *$ATM_EPHEM* for initial ephemerides created by *TLE2osc.pl*
`$data_opt`	subdirectory containing OBSCARD data (in *$ATM_DATASIM* for simulated observations, and in *$ATM_REALDATA* for real observations)
	continued on next page...

...continued from previous page	
`$dc_opt`	subdirectory in *$ATM_DC* and *$ATM_CAL* for data and log files
`$num_procs`	Number of processes to spawn in *estbfs.pl* – must be at least 1
`$keep_data`	Save large data files? 1=yes, 0=no.
`$iterate`	is this part of a BFE iteration? 1=yes, 0=no.
`$iter`	iteration number (is irrelevant if `$iterate`=0)
`$simulated`	Is this simulated data? 1=yes, 0=no.
`$do_global`	apply global ballistic factor correction? 1=yes, 0=no. (irrelevant if `$iterate`=0.)
`$num_iters`	Number of times to iterate – must be at least 1
`$start_iter`	Starting iteration number. Normally 1, can be set higher to run iterations one at a time or to restart after a crash.
`$datafiles_to_save`	Sets which large data files to save: "all", "none", or "last", which saves only those from the final iteration.

Commented Code (runestbfs.pl)

```
#!/usr/bin/perl -w
#
# runestbfs.pl Drives just the estbfs.pl subroutine.
#
# Author:
#
# Sarah E. Bergstrom
# May 02, 2002
#

#############################################################
#                                                            #
# Header section                                              #
#                                                            #
#############################################################

# Include subroutines

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

require "estbfs.pl";

# Set file options and variables

$start_epoch = "991215 000000.0";
$end_epoch    = "1000115 000000.0";
$ephem_opt    = "lowgrav"; # location of initial ephemerides
$data_opt     = "lowgrav_noise"; # location of simulated (or real) data.
$dc_opt       = "noise_mismodel"; # playground for GTDS large files
$num_procs    = 1;   # Number of processes to spawn in estbfs.pl (including parent)
$keep_data=1; # Save large data files?
$iterate=0; # Is this part of BFE iteration?
$iter=1; # Iteration number, only used if iterate=1.
$simuℓated=1; # 1 = sim data, 0 = real data.

# Call estbfs.pl with appropriate options

mkdir "$ENV{ATM_DC}/${dc_opt}",0777;

if ($iterate) {
    $iter_opt = $dc_opt . "/iter${iter}";
    $initfiℓe = "$ENV{ATM_CAL}/${dc_opt}/initinfo.txt_" . ($iter-1) . ".txt";
    $bℓfcfiℓe = "$ENV{ATM_CAL}/${dc_opt}/ballfcts_" . $iter . ".txt";
    mkdir "$ENV{ATM_CAL}/${iter_opt}",0777;
    mkdir "$ENV{ATM_DC}/${iter_opt}",0777;
} else {
    $iter_opt     = $dc_opt;
```

```
    $initfiℓe    = "$ENV{ATM_CAL}/${dc_opt}/initinfo.txt";
    $bℓfcfiℓe    = "$ENV{ATM_CAL}/${dc_opt}/ballfcts.txt";
}

&estbfs($start_epoch,$end_epoch,$ephem_opt,$data_opt,$iter_opt,
        $initfiℓe,$bℓfcfiℓe,$num_procs,$keep_data,$simuℓated);
```

Commented Code (runcalcvars.pl)

```
#!/usr/bin/perl -w
#
# runcalcvars.pl  – Run just calcvars.pl.
#
####################################################################
#                                                                  #
# Header section                                                   #
#                                                                  #
####################################################################

# Include subroutines

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

require 'calcvars.pl';

# Set file options and variables

$dc_opt      = "nonoise_mismodel";
$iterate     = 0; # Is this part of a BFE iteration? 1=yes, 0=no.
$iter        = 1;  # set this by hand, only used if $iterate=1.
$do_global   = 1; # only looked at if $iterate=1


if ($iterate) {

    $iter_opt = $dc_opt . "/iter${iter}";
    $initfiℓe    = "$ENV{ATM_CAL}/${dc_opt}/initinfo_" . ($iter−1) . ".txt";
    $bℓfcfiℓe    = "$ENV{ATM_CAL}/${dc_opt}/ballfcts_" . $iter . ".txt";
```

```
    $outfiℓe      = "$ENV{ATM_CAL}/${dc_opt}/initinfo_" . $iter . ".txt";
    $tmpfiℓe      = "$ENV{ATM_CAL}/${dc_opt}/array_tmp_" . $iter . ".txt";
    $dnsvarfiℓe  = "$ENV{ATM_CAL}/${dc_opt}/jac_densvars_" . $iter . ".txt";

} else {

    $iter_opt = $dc_opt;
    $initfiℓe     = "$ENV{ATM_CAL}/${dc_opt}/initinfo.txt";
    $bℓfcfiℓe     = "$ENV{ATM_CAL}/${dc_opt}/ballfcts.txt";
    $outfiℓe      = "$ENV{ATM_CAL}/${dc_opt}/junk.txt";
# $outfile = "junk.txt" because it won't actually be used, but it was
# easier just to put a placeholder in.
    $tmpfiℓe      = "$ENV{ATM_CAL}/${dc_opt}/array_tmp.txt";
    $dnsvarfiℓe  = "$ENV{ATM_CAL}/${dc_opt}/jac_densvars.txt";
    $iter=1;

}

    # Call calcvars.pl with appropriate options

    $ENV{ATM_ITER} = $iter;           # So that MATLAB can access the variables
    $ENV{ATM_TMP} = $tmpfiℓe;
    $ENV{ATM_DNSVAR} = $dnsvarfiℓe;
    $ENV{ATM_ITER_OPT} = $iter_opt;

    &calcvars($dc_opt,$iter_opt,$initfiℓe,$outfiℓe,$bℓfcfiℓe,$tmpfiℓe,$dnsvarfiℓe,
$iterate,$do_global);
```

Commented Code (bfe_iter.pl)

```
#!/usr/bin/perl -w
#
# bfe_iter.pl - Ballistic Factor Estimation Iteration Program
#
# Author:
#
# George R. Granholm
# 23 May 00
#
# Edited to make estbfs.pl a real subroutine
# 19 Oct 01
#
```

```
# Various and sundry minor changes, including formatting
# 01 Mar 02
#

##################################################################
#                                                                #
# Header section                                                  #
#                                                                #
##################################################################

# Include subroutines

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

require "estbfs.pl";
require "calcvars.pl";

# Set file options and variables:
# Options affecting both estbfs.pl and calcvars.pl:

$start_epoch = "991220 000000.0";
$end_epoch   = "991223 000000.0";
$ephem_opt   = "lowgrav";
$data_opt = "lowgrav_noise"; # in $ATM_REALDATA for real, $ATM_DATASIM for sim
$dc_opt      = "test2";
$num_iters   = 1;   # Number of iterations in est. of “true” ball. factors
$start_iter  = 1;  # 1 normally.
$simuℓated = 1;

mkdir "$ENV{ATM_DC}/${dc_opt}",0777;

# Options affecting estbfs.pl only:
$num_procs   = 1;
      # Number of processes to spawn in estbfs.pl (including parent)

$datafiℓes_to_save   = "none";
      # May be set to “all”, “last”, or “none” (anything else also  = “none”)
if ($datafiℓes_to_save eq "all") { $keep_data = 1; }
else { $keep_data = 0; }

# Options affecting calcvars.pl only:
$when_global = "first";   # Set to “first”, “all”, “none”.  (anything else also = “none”.)

##################################################################
#                                                                #
# Main Section                                                    #
#                                                                #
##################################################################
```

```
for ($iter = $start_iter; $iter <= $num_iters; $iter++) {

    # Generate iteration-specific options for estbfs.pl

    $iter_opt = $dc_opt . "/iter${iter}";
    mkdir "$ENV{ATM_CAL}/${iter_opt}",0777;
    mkdir "$ENV{ATM_DC}/${iter_opt}",0777;

    $initfiℓe    = "$ENV{ATM_CAL}/${dc_opt}/initinfo_" . ($iter−1) . ".txt";
    $bℓfcfiℓe    = "$ENV{ATM_CAL}/${dc_opt}/ballfcts_" . $iter . ".txt";

    if (($datafiℓes_to_save eq "last") && ($iter == $num_iters)) {
        $keep_data = 1;
    }

    # Call estbfs.pl

    &estbfs($start_epoch,$end_epoch,$ephem_opt,$data_opt,$iter_opt,
            $initfiℓe,$bℓfcfiℓe,$num_procs,$keep_data, $simuℓated);

    # Generate additional iteration-specific options for calcvars.pl

    if ($when_global == "all")  ||  (($iter == 1) && ($when_global == "first") {
                $do_global=1;
    } else  { $do_global=0; }

    $tmpfile     = "$ENV{ATM_CAL}/${dc_opt}/array_tmp_" . $iter . ".txt";
    $dnsvarfile  = "$ENV{ATM_CAL}/${dc_opt}/jac_densvars_" . $iter . ".txt";
    $initnew     = "$ENV{ATM_CAL}/${dc_opt}/initinfo_" .  $iter . ".txt";

    $ENV{ATM_ITER} = $iter;          # So that MATLAB can access the variables
    $ENV{ATM_TMP} = $tmpfile;
    $ENV{ATM_DNSVAR} = $dnsvarfile;
    $ENV{ATM_ITER_OPT} = $iter_opt;

    &calcvars($dc_opt,$iter_opt,$initfile,$initnew,$blfcfile,$tmpfile,$dnsvarfile,
$do_global, $simulated);

}
```

D.7 calc_b.m

Table D.9: calc_b.m Fact Sheet

Function	Perform weighted linear least-squares calculation to determine b_{1j} and b_{2j} .
Language	MATLAB
Type	Subroutine of *calcvars.pl*
Location	Main AtmoCal directory
Input Files	*array_tmp.txt* or *array_tmp_#.txt*
Environment Variables	`$ATM_ITER, $ATM_TMP, $ATM_DNSVAR, $ATM_ITER_OPT` must be set by *runcalcvars.pl* or *bfe_iter.pl*
Data Structure	Required: None
	Creates: None
Output Files	*jac_densvars.txt* or *jac_densvars_#.txt* and *calc_b.log*
User-Defined Variables	
NONE	

Commented Code (calc˙b.m)

```
% calc˙b.m - Density Variation Coefficient Calculator
%
% Author:
%
% George R. Granholm
% 1 May 00
%
%

clear all;
warning off;
more off;

% Set environment variables and filenames


[status,ATM˙CAL]      = unix('echo $ATM_CAL');
[status,iter]         = unix('echo $ATM_ITER');
[status,ATM_TMP]      = unix('echo $ATM_TMP');
[status,ATM_DNSVAR]   = unix('echo $ATM_DNSVAR');
[status,ATM_ITER_OPT] = unix('echo $ATM_ITER_OPT');

ATM_CAL      = ATM_CAL(1:length(ATM_CAL)-1);  % Remove newline
iter         = iter(1:length(iter)-1);
ATM_TMP      = ATM_TMP(1:length(ATM_TMP)-1);
ATM_DNSVAR   = ATM_DNSVAR(1:length(ATM_DNSVAR)-1);
ATM_ITER_OPT = ATM_ITER_OPT(1:length(ATM_ITER_OPT)-1);

model_opt = ATM_ITER_OPT;
tmpfile   = ATM_TMP;
outfile   = ATM_DNSVAR;
logfile   = 'calc_b.log';

% Open files and initialize variables

logid   = fopen(strcat(ATM_CAL,'/',model_opt,'/',logfile),'a');
toler        = 3;              % Num. of sigma tolerance for measurements
frcst_days   = 0;              % Number of days to forecast
T            = 27;             % Assumed period of density variations
lambda       = 2*pi/T;
time_grid    = .125;           % Time grid for forecasting (days)
sigma_b1     = 0.07;           % Std dev of WGN in b1
sigma_b2     = 0.07;           % Std dev of WGN in b2
sigma_b1_r   = 0.4;            % Std dev of Gauss-Markov RP for b1
sigma_b2_r   = 0.3;            % Std dev of Gauss-Markov RP for b2
alpha        = 0.241;          % Rate of decay of correlation
calc_flag    = 1;              % Input flag
                               %    1 = calculate dens vars
                               %    2 = read from infile
```

```
if (calc_flag==1)

% Begin loop to calculate density variations in data span

tempid = fopen(tmpfile,'r');
outid  = fopen(outfile,'w');
j = 1;
line  = fgetl(tempid);        % Get first line

while line ~= -1

   clear F a P Pvec;

   values = str2num(line);
   if length(values) ~= 2
      fprintf(logid,'Error - improper formatting of array_tmp.txt\n');
      disp('Error - improper formatting of array_tmp.txt\n');
      return
   end
   start_time(j) = values(1);
   array_len  = values(2);

% Read in data for span j

   for i = 1:array_len,
      values = str2num(fgetl(tempid));
      F(i,1) = values(1);
      F(i,2) = values(2);
      a(i) = values(3);
      Pvec(i) = values(4);
   end

   P = diag(Pvec);

% Test for erroneous measurements

   a_avg = mean(a);
   a_sigma = std(a);

   delete_count = 0;
   i = 1;

   while i<=array_len,
     if (abs(a(i)-a_avg)/a_sigma) > toler

         F(i,:) = [];        % Delete offending row
         a(i)   = [];        % from matrices or
         P(i,:) = [];        % vectors
         P(:,i) = [];
         array_len = array_len - 1;
         delete_count = delete_count + 1;
```

```
      end
      i = i+1;

   end

   disp(sprintf('%3d meas. > %2d-sigma tol.',...
                delete_count, toler));
   fprintf(logid,'%3d meas. > %2d-sigma tol. ',...
                delete_count, toler);

% Calculate b1 and b2 for span j

   b = (inv(F'*P*F))*(F'*P*a');
   densvars(j,1:2) = b';

% Print line to output file

   fprintf(outid,'%12.4f % 10.10E % 10.10E \n',start_time(j),b);
   fprintf(logid,'%12.4f % 10.10E % 10.10E \n',start_time(j),b);
   disp(sprintf('%12.4f % 10.10E % 10.10E',start_time(j),b));

   line = fgetl(tempid);
   j = j + 1;

end

% End loop to calculate density variations in data span

else

% Read dens vars in data span from file

   outid  = fopen(outfile,'a+');
   line   = fgetl(outid);          % Get first line
   j=1;

   while line ~= -1

      values = str2num(line);
      start_time(j) = values(1);
      densvars(j,1:2) = [values(2) values(3)];
      line   = fgetl(outid);
      j = j + 1;

   end

end

% Do forecasting if desired
```

```
if (frcst_days)

  fprintf(logid,'Calculating deterministic component...\n');
  disp(sprintf('Calculating deterministic component...'));

  % First solve for deterministic component

  j_max = j - 1;
  t_0 = start_time(j_max);
  for j = 1:j_max,
    G(j,1:3) = [ (1-cos(lambda*(start_time(j) - t_0))) ...
                 cos(lambda*(start_time(j) - t_0))     ...
                 sin(lambda*(start_time(j) - t_0)) ];

  end

  Z_b1 = densvars(:,1);
  Z_b2 = densvars(:,2);

  S_b1 = (inv(G'*G))*(G'*Z_b1);
  S_b2 = (inv(G'*G))*(G'*Z_b2);

  x_bar_b1  = S_b1(1);
  x_bar_b2  = S_b2(1);
  x_0_b1    = S_b1(2);
  x_0_b2    = S_b2(2);
  xdot_0_b1 = S_b1(3);
  xdot_0_b2 = S_b2(3);

  % Calculate estimate of deterministic component over entire time interval

  j_frcst_max = j_max + frcst_days/time_grid;

  for j=1:j_frcst_max,
    if (j>j_max)
      start_time(j) = start_time(j-1) + time_grid;
    end

    determ(j,1) = x_bar_b1 + (x_0_b1-x_bar_b1)*cos(lambda*(start_time(j) - t_0)) ...
                  +(xdot_0_b1/lambda)*sin(lambda*(start_time(j) - t_0));
    determ(j,2) = x_bar_b2 + (x_0_b2-x_bar_b2)*cos(lambda*(start_time(j) - t_0)) ...
                  +(xdot_0_b2/lambda)*sin(lambda*(start_time(j) - t_0));

  end

  % Calculate estimate of random component using scalar Kalman filter

  fprintf(logid,'Calculating random component...\n');
  disp(sprintf('Calculating random component...'));

  p_pred_b1(1) = sigma_b1^2;   % The b1 filter variance at j=1
```

```
p_pred_b2(1) = sigma_b2^2;  % The b2 filter variance at j=1
x_pred_b1(1) = 0;           % The prediction of b1 at j=1
x_pred_b2(1) = 0;           % The prediction of b2 at j=1

for j=1:j_max,

% Calculate residuals (which function as measurements of y(j))

   y_b1(j) = densvars(j,1) - determ(j,1);
   y_b2(j) = densvars(j,2) - determ(j,2);

   % Compute Kalman gain

   g_b1(j) = p_pred_b1(j)/(p_pred_b1(j) + sigma_b1^2);
   g_b2(j) = p_pred_b2(j)/(p_pred_b2(j) + sigma_b2^2);

   % Update states and errors based on actual measurement

   x_curr_b1(j) = x_pred_b1(j) + g_b1(j)*(y_b1(j)-x_pred_b1(j));
   x_curr_b2(j) = x_pred_b2(j) + g_b2(j)*(y_b2(j)-x_pred_b2(j));
   p_curr_b1(j) = (p_pred_b1(j)*sigma_b1^2)/(p_pred_b1(j)+sigma_b1^2);
   p_curr_b2(j) = (p_pred_b2(j)*sigma_b2^2)/(p_pred_b2(j)+sigma_b2^2);

   % Prediction ahead to next time step

   tau   = start_time(j+1) - start_time(j);
   x_pred_b1(j+1) = exp(-alpha*tau)*x_curr_b1(j);
   x_pred_b2(j+1) = exp(-alpha*tau)*x_curr_b2(j);
   p_pred_b1(j+1) = exp(-2*alpha*tau)*p_curr_b1(j) + ...
               (1-exp(-2*alpha*tau))*sigma_b1_r^2;
   p_pred_b2(j+1) = exp(-2*alpha*tau)*p_curr_b2(j) + ...
               (1-exp(-2*alpha*tau))*sigma_b2_r^2;

end

% Save estimates of random component at beginning of forecast span

x_r_0_b1 = x_curr_b1(j);
x_r_0_b2 = x_curr_b2(j);

% Write predicted density variations with deterministic + random components

for j=j_max+1:j_frcst_max,

   densvars(j,1) = determ(j,1) + exp(-alpha*(start_time(j)-t_0))*x_r_0_b1;
   densvars(j,2) = determ(j,2) + exp(-alpha*(start_time(j)-t_0))*x_r_0_b2;

   fprintf(outid,'%12.4f % 10.10E % 10.10E \n',start_time(j),densvars(j,1:2));
   fprintf(logid,'%12.4f % 10.10E % 10.10E \n',start_time(j),densvars(j,1:2));
   disp(sprintf('%12.4f % 10.10E % 10.10E',start_time(j),densvars(j,1:2)));
```

```
   end

end

warning on;

if (calc_flag==1)
   fclose(tempid);
end

fclose(outid);
fclose(logid);

exit;
```

D.8 Dates.pm

Table D.10: Dates.pm Fact Sheet

Function	Converts dates with the jul2cal and cal2jul subroutines.
Language	Perl
Type	Perl Module
Location	AtmoCal *include* subdirectory
Syntax	jdate = cal2jul(year, month, day, hour, minute, second.sss) (year, month, day, hour, minute, second.sss) = jul2cal(jdate)
User-Defined Variables	
NONE	

Commented Code (Dates.pm)

```
#!/usr/bin/perl
#
# Dates Package
#
# This package contains the following subroutines:
#
#————————————————————————————————-
# cal2jul($y,$m,$d,$h,$mn,$s) converts conventional calender dates into Julian
# dates. The returned date is in units of days and fractions of days.  The input
# date is entered in the following units:
#
#       '$d'  -   day              '$h'  -   hour
#       '$m'  -   month            '$mn' -   minute
#       '$y'  -   four-digit year  '$s'  -   second
#
#              Note - this function is only valid for dates after JD 0, i.e.
#                     dates after -4713 Nov 23.  Conversions are accurate to
#                     1/10000 of a second or better.
#
#————————————————————————————————-
# jul2cal($jdate) converts Julian dates into Gregorian calender dates.
# The returned date is in the following units:
#
#       '$d'  -   day              '$h'  -   hour
#       '$m'  -   month            '$mn' -   minute
#       '$y'  -   four-digit year  '$s'  -   second
#
#              Note - this function is only valid for dates after JD 0, i.e.
#                     dates after -4713 Nov. 23.  Conversions are accurate to
#                     1/10000 of a second or better.
#
#————————————————————————————————-
# get_time() invokes the Perl localtime function and converts to a string with
# the following format:
#                        "hh:mm EST MM/DD/YY"
#
#————————————————————————————————-

package     Dates;
require     Exporter;
@ISA        = qw(Exporter);
@EXPORT     = qw(cal2jul jul2cal get_time);

# Add environment-specific path and import modules

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}
use Localmath;
```

```
sub cal2jul {

    my $jdate;
    my ($y, $m, $d, $h, $mn, $s) = @_;

    if (length($y) == 2) {
        if ($y > 56) { $y = "19" . $y; }
        else { $y = "20" . $y; }
    }

    if ($m == 0) {$m = 1;}
    if ($d == 0) {$d = 1;}

    $jdate = int((1461*($y+4800+int(($m-14)/12)))/4) + int(367*($m-2-12*
int(($m-14)/12))/12) - int(3*int(($y+4900+int(($m-14)/12))/100)/4) + $d -
32075.5 + $h/24 + $mn/1440 + $s/86400;

    return $jdate;

}

sub jul2cal {

    my ($jdate) = @_;
    my ($ℓ, $n, $i, $j, $d, $m, $y
        , $s, $h, $mn, $rndjdate, $var, $ints, $fracts, $fr);

    #=== Calculate year, month, day

    $rndjdate = round($jdate);

    $ℓ = $rndjdate + 68569;
    $n = int((4*$ℓ)/146097);
    $ℓ = $ℓ - int((146097*$n+3)/4);
    $i = int((4000*($ℓ+1))/1461001);
    $ℓ = $ℓ - int((1461*$i)/4) + 31;
    $j = int((80*$ℓ)/2447);
    $d = $ℓ - int((2447*$j)/80);
    $ℓ = int($j/11);
    $m = $j + 2 - 12*$ℓ;
    $y = 100*($n-49) + $i + $ℓ;

    #=== Calculate hour, minute, second

    $jdate = $jdate - cal2jul($y,$m,$d,0,0,0);
    $h = int($jdate*24);
    $jdate = $jdate*24 - $h;
    $mn = int($jdate*60);
    $jdate = $jdate*60 - $mn;
    $s = $jdate*60;
```

```
    #=== Force month, day, hour, min, second to two-digit format with padded zeroes

    $ints = int($s);
    $fr = sprintf("%1.4f", ($s - $ints));  # force 0.1 millisecond accuracy
    if ($fr == 1) {$fr = "0.9999";}
    ($fracts) = ($fr =~ /^0(\..*)/);
    foreach $var ($m, $d, $h, $mn, $ints) {
        $var = (sprintf "%2.2d",$var);
    }
    $s = $ints . $fracts;

    return ($y, $m, $d, $h, $mn, $s);
}

sub get_time {

    my (@time, $time, $date, $tot);

    @time = (localtime);
    $time = sprintf("%2.2d",$time[2]) . ":" . sprintf("%2.2d",$time[1]) . ":"
        . sprintf("%2.2d",$time[0]) . " EST ";
    $date = ($time[4] +1) . "/" . $time[3] . "/" . $time[5];
    $tot = $time . $date;

    return $tot;

}
```

D.9 b3conv.pl

Table D.11: b3conv.pl Fact Sheet

Function	Convert multiple files of NORAD B3 observations to OBSCARD format
Language	Perl
Type	Main Program
Location	AtmoCal *utils/b3conv* subdirectory
Input Files	*initinfo.txt*, *STATFILE.DAT* and one or more observation files specified by the `$obs_match` variable, in the *\$ATM_REALDATA/\$obspath* directory
Environment Variables	`$ATM_REALDATA` must be set.
Data Structure	Requires a properly-compiled copy of noradpp.exe in the same directory.
	Creates: None.
Output Files	OBSCARD files for each satellite, as specified by `$output_suffix` in *\$ATM_REALDATA/\$output_path* directory.
User-Defined Variables	
`$obspath`	directory containing B3 observations (in *\$ATM_REALDATA*)
`$rawmatch`	Pattern[2] (like ".obs") that all B3 observation files match.
`$logfile`	Log File name (can include a relative path from *\$ATM_REALDATA*)
`$outpath`	directory for OBSCARD observations (in *\$ATM_REALDATA*)
`$outsuffix`	Suffix for OBSCARD files. Normally ".obscard".
`$msgsuffix`	Suffix for individual satellite log files. Normally ".msg" or ".log".

Commented Code (b3conv.pl)

```
#!/usr/bin/perl
use strict;
no strict "refs";

my ($rawmatch, $rawpath, $rawdir, @rawlist, $infile, $fh);
my ($tempfile, $tempoutfile, $tempclean, $tempsort, $sorthead);
my ($header, $tailer, $satstring, $rawname);
my ($rawfile, $outsuffix, $outpath, $msgsuffix);
my ($logfile, $line,$outfile,$msgfile);
my ($entry, $sat, $station, $year, $day, $jyear, $start_ymd, $end_ymd);
my ($new_start_ymd, $new_end_ymd, $firstloop);
my (%satlist,%stations, @sd, @ed);

# Add environment-specific path and import modules.

BEGIN {
   push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

use Dates;      # Necessary to use cal2jul, jul2cal, & get_time subroutines
                # Note that Dates uses Localmath

# This script takes a file in B3 format containing many satellites
# and stations, and automatically creates the CONTROL.DAT files and
# runs the conversion to GTDS obscard format.

chdir "$ENV{ATM_REALDATA}" or die "Can't chdir to $ENV{ATM_REALDATA}";
$rawpath = "rawb3";
$rawmatch = ".obs";
$logfile = "b3conv.log";
$outpath = "test";
$outsuffix = ".obscard";
$msgsuffix = ".msg";
%satlist = ();

$rawdir = $rawpath;
opendir RAWDIR, $rawdir or die "Cannot open $rawdir, died";
@rawlist = grep /$rawmatch/,  readdir(RAWDIR);
closedir RAWDIR;

# print STDOUT "

open LOGFILE, ">$logfile" or die "Unable to open $logfile, died";

# Open and read $initfile

foreach $fh ("LOGFILE", "STDOUT") {
   print $fh "Processing initinfo.txt. . . \n";
}
```

```

open INITINFO, "<initinfo.txt" or die "Unable to open initinfo.txt, died";

 INITLINE: while (defined($ℓine = <INITINFO>)) {

     $ℓine =~ s/^(\d{5})\s//;
     $satℓist{$1}=1;

 }

close INITINFO;

# Open and read STATIONS.DAT file

foreach $fh ("LOGFILE", "STDOUT") {
    print $fh "Processing station list. . . \n";
}

# Currently, not much is done with the station list.

open STATFILE, "<STATFILE.DAT" or die "Unable to open STATFILE.DAT, died";
 STATLINE: while (defined($ℓine= <STATFILE>)) {
     $ℓine =~ /\s+(\d+)\s{5}(\w{4})/;
     $stations{$1}=$2;
 }

close STATFILE;

$header = $outpath . "/header.txt";
$taiℓer = $outpath . "/tailer.txt";

open HEADFILE, ">$header";
print HEADFILE "OBSCARD\n";
close HEADFILE;
open TAILFILE, ">$tailer";
print TAILFILE "END\n";
close TAILFILE;

foreach $sat (sort keys %satlist) {

    foreach $fh ("LOGFILE", "STDOUT") {
        print $fh "Processing satellite $sat. . . \n";
    }

    $tempfiℓe = $outpath . "/" . $sat . ".tempfile";
    if (-e $tempfiℓe) { system("rm $tempfile"); }
    foreach $rawname (@rawlist) {
        $rawfiℓe = $rawpath . "/" . $rawname;
        $satstring = "U" . $sat;
        system("grep $satstring $rawfile >>$tempfile");
    }
```

```
open INFILE,"$tempfile" or die "Can't open $tempfile: $!\n";

foreach $fh ("LOGFILE", "STDOUT") {
    print $fh "Determining Obs. Start/End times. . . \n";
}

$firstℓoop = 1;

while ($entry = <INFILE>) {  # Process the first few fields

    $entry =~ /U(\w{5})(\w{3})(\w{2})(\w{3}).*$/;
    if ($1 == $sat) {
        $station=$2;
        $year=$3;
        $day=$4;

        $jyear = cal2jul($year,1,1,0,0,0); # gets the Julian date of the
                                           # beginning of that year.
                                       # Note: assumes dates in 1956-2055

        @sd = jul2cal($jyear+$day-1);
        @ed = jul2cal($jyear+$day+1);

        if ($firstℓoop == 1) {
            $start_ymd = ($sd[0]-1900) . $sd[1] . $sd[2];
            $end_ymd = ($ed[0]-1900) . $ed[1] . $ed[2];
            $firstℓoop = 0;
        } else {
            $new_start_ymd = ($sd[0]-1900) . $sd[1] . $sd[2];
            $new_end_ymd = ($ed[0]-1900) . $ed[1] . $ed[2];
        }

        if ($new_start_ymd < $start_ymd) {
            $start_ymd = $new_start_ymd;
        }
        elsif ($new_end_ymd > $end_ymd) {
            $end_ymd = $new_end_ymd;
        }
    }
    close INFILE;

    if (length($start_ymd) == 6) {
        $start_ymd = "0" . $start_ymd;
    }
    if (length($end_ymd) == 6) {
        $end_ymd = "0" . $end_ymd;
    }

    foreach $fh ("LOGFILE", "STDOUT") {
        print $fh "Converting observations. . . \n";
```

```
        }

        $outfiℓe =  $outpath . "/" . $sat . $outsuffix;
        $tempoutfiℓe = $outpath . "/" . $sat . $outsuffix . ".temp";
        $tempcℓean = $tempoutfiℓe . ".clean";
        $tempsort = $tempoutfiℓe . ".sort";
        $sorthead = $tempsort . ".head";

        $msgfiℓe = $outpath . "/" . $sat . $msgsuffix;

        open(CARD, ">CONTROL.DAT") or die "Can't create card $!\n";
        write CARD;
        close CARD;
        system("$ENV{ATM_REALDATA}/noradpp.exe");

        foreach $fh ("LOGFILE", "STDOUT") {
            print $fh "Sorting observations. . .\n";
        }

        system("rm $tempfile");
        system("grep -F '.' $tempoutfile >$tempclean");
        system("rm $tempoutfile");
        system("sort -n -k3 $tempclean >$tempsort");
        system("rm $tempclean");
        system("cat $header $tempsort >$sorthead");
        system("rm $tempsort");
        system("cat $sorthead $tailer >$outfile");
        system("rm $sorthead");
    }

    foreach $fh ("LOGFILE", "STDOUT") {
        print $fh "Finished processing satellite $sat.\n";
    }
}

system("rm $header");
system("rm $tailer");
close LOGFILE;

# CONTROL.DAT formatting
#123456789#123456789#123456789#123456789#123456789#123456789#
format CARD =
SATELLITE DESIGNATOR        : NSSC@<<<<
                          ${sat}
START DATE (YYYMMDD.)       :   @>>>>>>.
                          $start_ymd
STOP DATE (YYYMMDD.)        :   @>>>>>>.
                          $end_ymd
NORAD OBS DATA INPUT FILE  :   @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          $tempfiℓe
GTDS OBS DATA OUTPUT FILE  :   @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
                        $tempoutfile
OUTPUT PRINT FILE          :   @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                        $msgfile
.
```

D.10 Other Utilities

Several other utilities are included in the AtmoCal package. The *dateconvert.pl* script is an interactive interface to the *Dates.pm* perl module, the *get_peri.pl* script was written by Granholm to parse GTDS EPHEM output files for perigee height.

Table D.12: dateconvert.pl Fact Sheet

Function	Converts dates interactively
Language	Perl
Type	Main Program
Location	AtmoCal *utils* directory
Syntax	Run at UNIX prompt. Program will ask for all relevant information.

Table D.13: get_peri.pl Fact Sheet

Function	Parses GTDS EPHEM *.output* files for perigee height
Function	Converts dates interactively
Language	Perl
Type	Main Program
Location	AtmoCal *utils* directory
Environment Variables	`$ATM_EPHEM` must be set.
Input files	*initinfo.txt* in same directory as *get_peri.pl* *#####_ephem.output* files in *$ATM_EPHEM/$model_opt*
User-Defined Variables	
`$model_opt`	Directory in *$ATM_EPHEM* containing GTDS EPHEM .output files.

Commented Code (dateconvert.pl)

```
#!/usr/bin/perl -w

# This is just a little front-end for the routines in Dates.pm.  Mostly
# intended so that I could quickly convert a date from Julian to Gregorian
# to verify that I was looking at the right place in a dataset.
# Sarah Bergstrom

BEGIN {
    push @INC, "$ENV{ATM_CAL}/AtmoCal/include";
}

use Dates;
use Localmath;

print STDOUT " Choose an option:\n [1] Julian --> Gregorian.\n [2] Gregorian -->
Julian\n [3] Quit.\n";

$choice = <STDIN>;
chomp $choice;

while ($choice !=3 ) {

    if ($choice == 1) {
        print STDOUT "Type Julian date:\n";
        $juℓ= <STDIN>;
        chomp $juℓ;
        ($y, $m, $d, $h, $n, $s) = jul2cal($juℓ);
        print STDOUT "$h:$n:$s on $m/$d/$y.\n";
    } else {
        print STDOUT "Year.\n";
        $y = <STDIN>;
        chomp $y;
        print STDOUT "Month.\n";
        $m = <STDIN>;
        chomp $m;
        print STDOUT "Day.\n";
        $d = <STDIN>;
        chomp $d;
        print STDOUT "Hours, Min, Sec = 0.\n";
        $juℓ = cal2jul($y,$m,$d,0,0,0);
        print STDOUT "Julian Date $jul.\n";
    }
    print STDOUT " Choose an option:\n [1] Julian --> Gregorian.\n [2] Gregorian -->
Julian\n [3] Quit.\n";
    $choice = <STDIN>;
    chomp $choice;

}
```

Commented Code (get_peri.pl)

```
#!/usr/bin/perl
#
# get_peri.pl
#
# Author:
#
# George R. Granholm
# 5 Apr 00
#
# This file is used to parse GTDS .output files for perigee height.
# It uses the following files:
#
#        initinfo.txt -        contains desired catalog numbers in
#                               format output by TLE2osc.pl
#
#        #####_ephem.output - the name of the output file, where
#                               ##### is the NORAD catalog number.
#                               get_peri assumes that the .output
#                               files are in the $GTDS_STOR global var.
#

$modeℓ_opt = "highgrav";

open INITINFO, "initinfo.txt" or die "Can't find initinfo.txt";

 INITLINE: while ($ℓine = <INITINFO>) {

    $ℓine =~ s/^(\d{5})\s//;
    $initinfo{$1} = [ split(" ",$ℓine) ];

 }

close INITINFO;

 READOUTPUT: foreach $catnum (keys %initinfo) {

    open OUTFILE, "$ENV{ATM_EPHEM}/${model_opt}/${catnum}_ephem.output" or die
      "Unable to find file $ENV{ATM_EPHEM}/${model_opt}/${catnum}_ephem.output, died" ;
    $endfℓag = 0;

    while ($outℓine = <OUTFILE>) {

       if ($outℓine =~ /^ ENDED ORBIT FILE/) {
          $endfℓag = 1;
       }
       elsif ($endfℓag && ($outℓine =~ /.* PH     (0\.\d{16}D[+-]\d{2})\s*$/)) {
          $ph = $1;
          last;
```

```
        }
    }

    $ph{$catnum} = $ph;
    close OUTFILE;
}

open PERIHTS, ">perigees.txt" or die "Unable to open perigees.txt, died";
foreach $catnum (keys %ph) {
    print PERIHTS "$catnum: $ph{$catnum} km\n";
}

```

D.11 Graphing Utilities

Table D.14: read_b.m Fact Sheet

Function	Reads a *jac_densvars.txt*-format file, calculates some statistics, and plots b_{1j} and b_{2j} values (example graphs: Figures 4-2 and 4-3).
Language	MATLAB
Type	main MATLAB routine
Location	AtmoCal *analysis* subdirectory
Input Files	Prompts for a filename (relative path to current directory)
Output Files	None – graphs must be saved by hand.
User-Defined Variables	
None, but prompts for input file.	

Table D.15: analyze_atmcal.m Fact Sheet

Function	Reads a series of *initinfo_#.txt* files, calculates some statistics, and creates a series of plots. (An example of the fourth plot type can be seen in Figure 5-10.)
Language	MATLAB
Type	main MATLAB routine
Location	AtmoCal *analysis* subdirectory
Input Files	None, but see *readinitinfo.m* fact sheet.
Data Structure	Requires: *readinitinfo.m* must be in the same directory.
Output Files	None – graphs must be saved by hand.
User-Defined Variables	
Prompts for number of iterations and whether or not an *initinfo.txt* truth file exists.	

Table D.16: readinitinfo.m Fact Sheet

Function	Reads an *initinfo.txt*-formatted file into a series of MATLAB arrays.
Language	MATLAB
Type	main MATLAB routine, called by *analyze_atmcal.m* but can stand alone as well.

continued on next page...

...continued from previous page	
Location	AtmoCal *analysis* directory
Input Files	*initinfo.txt*-formatted file specified by `filename`.
Output Files	None.
User-Defined Variables these must be defined before running *readinitinfo.m*[3].	
`filename`	Name of file to read
`n`	numerical index for file (maximum 20). When called by *analyze_atmcal.m*, this is set to iteration number + 1, with the truth file in index 1).

Other utilities George Granholm's "rca_plots.m" and "rca_plots_novel.m" have been included in the AtmoCal *analysis* directory without modification. These utilities appear to have been used to create the plots showing distance and velocity error for individual satellites in Granholm's thesis. These utilities have not been run on Pisces, and may or may not work as included in AtmoCal.

Commented Code (read_b.m)

```
% This MATLAB script reads a file of b-values (created by calc_b.m)
% plots them, and does some simple statistics.

clear
clf

filename='jac_densvars.txt';
%filename='jac_densvars_1.txt';

[time, b1, b2] = textread(filename,'%f%f%f');

% The following is a bit of a kludge - rather than convert julian to
% to calendar, just plot days-since-start. The first b-value is
% 1.5 days into the original fit window. Obviously, if the fit span
% is changed to something other than 3 days, fix this.

time = time - time(1) + 1.5;

subplot(2,1,1);
title('Coefficient B1');
xlabel('Time since beginning of fit window (days)');
plot(time,b1);
subplot(2,1,2);
title('Coefficient B2');
xlabel('Time since beginning of fit window (days)');
plot(time,b2);

avgb1 = mean(b1)
avgb2 = mean(b2)
stdb1 = std(b1)
stdb2 = std(b2)
maxb1 = max(b1)
maxb2 = max(b2)

x=200:50:600;
[a,b]=size(b1);
y=1:1:a;
z=b1*ones(size(x))+b2*(x-400)/200;

pause;
subplot(1,1,1);
title('Correction Factor');
```

```
xlabel('Height (km)');
ylabel('Time since beginning of fit window (days)';
surf(x,y,z)
```

Commented Code (analyze_atmcal.m)

```
% This program reads in the initinfo files from an atmcal run and plots
% the convergence (or lack thereof) of the ballistic coefficients.


% Ask the user if there is a "truth" file. If there isn't, then it will
% assume that the truth is the results of the last iteration. It'll still
% be obvious if it didn't converge...

clear
clf
type=input('Type 1 for analysis with a truth file, 2 for analysis from real data:\n');

% Ask the user how many iterations were run so it knows how many data files
% to look for.
iters=input('How many iterations were run?\n');

if type == 1
filename='initinfo.txt'; n=1;
elseif type == 2
filename=strcat('initinfo_',int2str(iters),'.txt'); n=1;
end

readinitinfo

devper=zeros(max(size(nssc_num)),iters+1);
avgdevper=zeros(1,iters+1);
stddevper=zeros(1,iters+1);
maxvar=zeros(1,iters+1);
avgvar=zeros(1,iters+1);

for n=2:iters+2

  m=n-1;
  filename=strcat('initinfo_',int2str(m-1),'.txt');
  readinitinfo
  devper(:,m)=100*((Ki(:,n)-Ki(:,1))./Ki(:,1));
  maxdevper(m)=max(devper(:,m));
  avgdevper(m)=mean(devper(:,m));
  stddevper(m)=std(devper(:,m));
  maxvar(m)=max(sigma_i_squared(:,m));
  avgvar(m)=mean(sigma_i_squared(:,m));
end

% The relevant information are the Kis and the sigma_i_squareds. The
% other fields should match up, since they're from the same run.


% Plot the actual Ki values.
subplot(2,2,1)
```

```
plot(nssc_num,Ki)
xlabel('NSSC Satellite Number')
ylabel('Ballistic Coefficient Ki')
title('Ki Data')
% Plot the deviations in %.
subplot(2,2,2)
plot(nssc_num(:,2:iters+2),devper)
xlabel('NSSC Satellite Number')
ylabel('Deviation from "true" Ki, in %')
title('Percent Deviations of Ki values from True/Final values')
% Plot the maximum deviation in %.
subplot(2,2,3)
iter_indices=0:iters;
plot(iter_indices,maxdevper)
xlabel('Iteration Number (zero=initial value)')
ylabel('Percent')
title('Maximum % Deviation')
% Plot the average deviation in %.
subplot(2,2,4)
plot(iter_indices,avgdevper)
xlabel('Iteration Number (zero=initial value)')
ylabel('Percent')
title('Average % Deviation')
```

Commented Code (readinitinfo.m)

```
% This m-file takes an initinfo-layout file and reads it into matlab.
% SEB 1/26/01
% The following lines are commented out so that it can be called from
% analyze_atmcal with standardized filenames and indicies.
% It's not a function because that would be a pain with variable-size
% inputs.
%filename=input('Input a filename here:\n','s');
%n=input('Numerical index for this file?\n');

if exist('filelist')
 oldfilelist=filelist;
else
oldfilelist=strvcat(' ',' ',' ',' ',' ',' ',' ',' ',' ',' ');
end
filelist='';
for a=1:10  % maximum twenty files
    if n>10
       disp ('N too large...')
    else
     if a~=n
        filelist=strvcat(filelist,oldfilelist(a,:));
     else
        filelist=strvcat(filelist,filename);
     end
    end
end
[nssc_num(:,n),intl_num(:,n),Ki(:,n),RCS(:,n),sigma_i_squared(:,n),std_s(:,n),obstype(:,n)]=
textread(filename,'%u %s %f %f %f %s %u');
```

Appendix E

File Utilities and Formats

E.1 B3 to OBSCARD Conversion Utility

The real data available to this project consisted of several groups of time-sorted observations, all in NORAD B3 format. These files each contained observations for hundreds of different objects. This format is substantially different from that of OBSCARD files, which only contain observations for one satellite/object. Thus, a conversion utility must loop through each B3 file, looking for an individual satellite. Jack Fischer modified Joe Lombardo's *runadcob* utility to allow the use of a control card to specify the desired satellite, so that recompilation was not required for each satellite of interest. His modifications are described in his thesis, and are well documented in the code[9]. Jack Fischer called the utility "NORADPP", for NORAD Pre-Processor, and that name has been retained. The B3 files available did not precisely fit the B3 format as outlined by Fischer and the "NORADPP" code – this is due to varying standards, and the code was modified to read the alternate format. The original format statements were commented out, and should be restored and/or altered if necessary[1]

---

[1] The statements in question fall at the end of the *runadcob.for* file, which includes a brief description of the differences.

To facilitate the conversion of multiple files containing hundreds of different satellites, a Perl script were created to run *noradpp.exe* on every satellite in a particular file or group of files, and to sort and merge multiple OBSCARD files for the same satellite. The NORADPP source code and the *b3conv.pl* script are included in the AtmoCal CVS package, in the *utils/b3conv* subdirectory, and the user options for *b3conv.pl* are listed in Appendix D.9. The *b3conv.pl* script requires a copy of *initinfo.txt*[2] containing all of the satellites which should be processed, since it saves a significant amount of time and resources if only the desired observations are converted.

Several minor alterations were also made to the NORADPP code, as described in the following table and documented in the code itself. (Use `grep` or another search utility to look for the "`C_SEB`" delimiters.) The OBSCARD format uses "100" to denote the year 2000 (101 = 2001, etc.), , while B3 format uses "00". All years were converted to fall between 1956 and 2055, and the resulting three-digit dates are now handled appropriately by NORADPP[10].

One final caution: the OBSCARD files created may be substantially larger than the original NORAD B3 files, since OBSCARD files require three lines, not one, for a range/az/el observation triple.

The NORADPP code can be compiled and linked on a 64-bit SGI-UNIX platform (using the MIPSpro Fortran 77 compiler, which is included in the SGI IRIX operating system) by issuing the following commands:

```
prompt% f77 -c -64 -O2 *.for
prompt% f77 -o noradpp *.o
```

The OBSCARD format is described in the following section, for the use of anyone who wishes to build a converter similar to NORADPP for another observation format. A detailed description of the B3 format can be found in Fischer's thesis[9], on pages

---

[2] Actually, only the first field of *initinfo.txt*, containing the NSSC# of the satellite, is read. So this script could be used with an *initinfo.txt* file containing only an NSSC# per line.

Table E.1: List of NORADPP Files Modified

Filename	Modification
obsdat.cmn	changed name from obsdat#.cmn to obsdat.cmn, since # is often used as a comment character
astron.for	changed `include 'obsdat#.cmn'` to `include 'obsdat.cmn'`
	changed format to allow three-digit years.
	changed subroutine call from DATE to NEWDATE
azimut.for	changed format to allow three-digit years.
elevat.for	changed format to allow three-digit years.
newdate.for	renamed *date.for* to avoid conflict with built-in unix date utility
ranger.for	changed format to allow three-digit years.
ranges.for	changed format to allow three-digit years.
runadcob.for	(Main File) lengthened allowable filenames from A12 to A30 finished support for named log files
	added support for three-digit years in CONTROL.DAT
	added two extra spaces in format for reading B3 files to agree with AtmoCal real data
	changed subroutine call from DATE to NEWDATE

315–316, and is not repeated here. The only differences between that format and the one used here were that the `ITYPE` field was moved one space right (to column 75) and the `EQNYR` field was moved two spaces to the right (to column 77).

E.2 Building New GTDS Binary Files

Since a near-real time implementation of AtmoCal should be supplied with up-to-date lists of $\mathrm{F}_{10.7}$ and K_p the appropriate GTDS binary files must be constantly rewritten to include new data. Thus, a pair of programs have been included in the AtmoCal CVS distribution which can convert a properly formatted text file to the GTDS$075 binary file format, and vice versa. These utilities can be found in the *utils/gtds_binaries/jacchia* subdirectory. (The *utils/gtds_binaries/other* subdirectory contains some untested utilities for working with the other GTDS binary files.)

E.3 Detailed File Formats

The main types of files used and/or created by AtmoCal and gtds_granholm are:

OBSCARD format OBSCARD files are ascii text files created by GTDS DATASIM and NORADPP, and contain a list of observations for a single satellite. As used in AtmoCal, they conform to the format called "LAYOUT 3" in the GTDS data set reference[30], with the year field lengthened. Observations created by NORADPP contain identical data in the "corrected" and "uncorrected" data fields. Table E.2 contains the updated version of "Layout 3" with three-digit dates. The first line must contain `OBSCARD␣` and the last line must contain `END␣␣␣␣␣`, where "␣" denotes a space, in place of the station name.

Table E.2: OBSCARD Format

Spaces	Variable Name in OBSCRD	Contents
1–8	SNAME	Station Name (usually only the first 4 characters are used, e.g. FLYQ, KAEQ)
9–11	MTYPE	Observation Type: 1 = Range 4 = Azimuth 5 = Elevation 9 = Range Rate
12–14	IGATE	Range-gating indicator. (Not used in any of the data in this project)
15–16		Empty space
17–38	R1	Observation Time (YYYMMDDHH-MMSS.SSSS)
39–59	OM1	Uncorrected Observation
60–80	OM2	Corrected Observation

Example for one Range/Az/El observation (from the simulated observations of satellite 00063):

```
OBSCARD
KAEQ      1       991215214600.000    0.20433717982159E+04 0.20433717982159E+04
KAEQ      4       991215214600.000    2.37436203800539     2.37436203800539
KAEQ      5       991215214600.000    0.135035811832215    0.135035811832215
END
```

TLE format Two-line element sets contain data that, when used with the NORAD SGP4/SDP4 model, provides the position and velocity of a satellite. AtmoCal uses TLEs to obtain initial orbit estimates, which are then used to create simulated observations and as initial guesses for GTDS DC runs. For more information on reading TLEs, see reference [28].

initinfo.txt This file, created by *TLE2osc.pl* and used by *genobs.pl*, *estbfs.pl*, and *calcvars.pl*, lists the ballistic factors and other pertinent characteristics of each satellite. Granholm chose 29 as the observation type for simulated data, and 15 for real observations, but this data is currently unused, and all copies of *initinfo.txt* currently just contain 29 in this field.

Table E.3: Format of *initinfo.txt*

Spaces	1–5	8–14	16–31	32–47	48–63	65	67–68
Type	Int 5	Char 7	Exp. 11	Exp. 11	Exp. 11	Char 1	Int 2
Info.	NSSC#	Int'l Des.	k_i	RCS	σ_i^2	(S)tandard or (N)on-St.	Obs.Type real = 15 sim = 29

Example:

```
00063   60016A 1.9706828120E-03 1.1300000000E+00 1.0000000000E-06 S 29
00179  61015BD 9.0312609648E-02 1.6670000000E-01 1.0000000000E-06 S 29
⋮
```

jac_densvars.txt The density variations file (FRN 106) contains time-sorted b_{1j} and b_{2j} values. The format is:

Table E.4: Format of *jac_densvars.txt*

Spaces	1–12	15–30	33–48
Type	F12.4	Exp. 15	Exp. 15
Info.	Julian Date	B1	B2

Example:

```
2451529.0000  8.0705485337E-03  1.3900485352E-02
2451529.1250  1.8248157892E-02  3.7869159153E-02
2451529.2500  4.8232116750E-02  3.3850276952E-02
⋮
```

ballfcts.txt The observed ballistic coefficient list *ballfcts.txt* contains all of the ballistic factor estimations from the GTDS DC runs. This file is later sorted by date/time into *ballfcts.txt.sort*, since the initial observed $\hat{k}_{ij}$ values may not be in order if multiple processes are used. The format is:

Table E.5: Format of *ballfcts.txt*

Spaces	1–5	7–18	20–35	37–52
Type	Int. 5	F10.2	Exp. 15	Exp. 15
Info.	NSSC#	Julian Date t_i	$\hat{k}_{ij}$	Height h_i

Example:

```
11849 2451529.0000 1.4818508931E-03 5.3087600000E+02 00063 2451529.0000
2.1419918442E-03 5.4550800000E+02 ⋮
```

GTDS binary data files The GTDS binary data file formats are not detailed here. Please refer to a copy of the report "Data Set Layouts for the Goddard Trajectory Determination System"[30]. Copies of the binary-to-ascii and ascii-to-binary converters for these files have been included in AtmoCal, in the *utils* directory, but no rigorous tests have been made on these utilities.

RCS file The radar cross-section (RCS) file contains a list of NSSC numbers and radar-cross sections, obtained from Dave Vallado.[56]. The NSSC numbers can either include the leading zeros (e.g. 00063) or be truncated (and flushed left or right).

Example:

```
1        20.4200
2
⋮
10096    11.1735
⋮
```

E.4 GTDS Input Decks

A copy of the list of GTDS keywords[15] for creating GTDS input decks is a must-have for anyone who wants to make major changes to the input decks automatically created by AtmoCal. The details of each of these decks are too long to enumerate here, since they can all be looked up in the GTDS keyword guide. A few subdecks, however, are especially noteworthy, and are listed below.

POTFIELD, MAXDEGEQ, MAXORDEG are the cards where the truncated (4x4) JGM2 gravitational model was chosen. This should be changed to a more accurate gravitational model when processing real data.

Station Card 0 lists station noise characteristics. If new simulated data is being generated with different stations, or if real data with more stations is being processed, station card 0 must be added for each new station. The parameters for this card should come from the SLAD[9] or some similar source. Note that *genobs.pl* includes two versions of the GTDS DATASIM input deck, one with noise, and one without. Make sure to add any new stations to the noisy deck, since otherwise the noise characteristics will be ignored. The format for Station Card 0 is shown in Table E.6. For a complete list of observation types, see Table A-2 in the GTDS keyword guide[15]. If a station has more than three observation types, more than one Station Card 0 can be used.

Station Card 1 lists station locations. When working with real data, more stations will appear than just the original four stations used for simulated data generation. The locations of these stations must be included in the GTDS DC input deck, using station card 1. Table E.7 gives a partial description of the Station Card 1 format sufficient for creating new cards to be used with AtmoCal. (Station Card 1 can also be used for landmarks, which is not detailed here.)

Station Card 2 This card contains information about the ellipsoid model used and other station dependent data. Currently, this card looks like the example given below for all stations used. (Replace the **** with the station name.) See also the entry for the ELLMODEL card in the GTDS keyword list[15].

Example:

```
/****    200001
```

Table E.6: Format of Station Card 0

Columns	Format	Description
1–8	A8	Station Name
9	I1	=0 (card number) – defines Station Card number. Station Card 0 defines station-dependent noise.
10–11	I2	First observation type Types seen in available data: =1 Range =4 Azimuth =5 Elevation =9 Range-Rate
12–14	I3	Second observation type
15–17	I3	Third observation type
18–38	G21.14	Error associated with first observation type
39–59	G21.14	Error associated with second observation type
39–59	G21.14	Error associated with third observation type

Table E.7: Format of Station Card 1

Columns	Format	Description
1–8	A8	Station Name
9	I1	=1 (card number) – defines Station Card number. Station Card 1 defines tracking station type..
10	1X	Blank
11–14	I4	Station catalog number.
15–17	I3	Station type (the last letter of the station name matches the letter(s) in parentheses listed below): =1 VHF (V) =2 Minitrack (M) =3 C-band (T,Q,F) =4 S-band (G) =5 USB–30 foot (S,A,W) =6 USB–85 foot (S,A,W) =7 SRE-VHF (X,Y,Z) =8 ATS (R) =9 ATS–ground transponder(B) =10 DSN (D) =11 SRE (S) =12 Laser (L) =13 Optical (C) =14 X-Y Parabolic (E,4)
18–38	G21.14	Height above/below sea level
39–59	G21.14	Geodetic latitude (ddmmss.ssss)
39–59	G21.14	Geodetic longitude (ddmmss.ssss)

ATMCAL Control Card To turn the atmospheric density correction on, make sure FRN 106 is linked to the appropriate file, and use the ATMCAL control card. Most of the fields are currently unused, giving options for further customizing the atmospheric density correction process. It must be placed in the OGOPT subdeck for a GTDS DC, EPHEM, or FILTER run. The format for this card is shown in Table E.8.

Table E.8: Format of ATMCAL card

Columns	Format	Description
1–8	A8	ATMCAL – Input card for atmospheric corrections.
9–11	I3	Turn on/off atmospheric correction =0 Off (default) =1 On
12–14	I3	Number of atmospheric density model to apply corrections to (only JR71 is currently operational): **=1 Jacchia-Roberts 1971** *=2 Harris-Priester* *=3 Jacchia-64* *=4 Jacchia-70* *=5 MSIS-77* *=6-8 Reserved for RADARSAT* *=9 MSISE-90* *=10 Reserved for GOST*
15–17	I3	Unused
18–38	G21.14	Unused
39–59	G21.14	Unused
60–80	G21.14	Unused

Appendix F

LaTeX Notes

This thesis was created using LaTeX 2_ε and BibTeX on Windows, Unix, and Macintosh platforms. The MikTex distribution[50] and the TeXShell editing package[53] were used under Windows 2000, the TeXShop editing package[29] was used under Mac OS X, the built-in Emacs LaTeX and BibTeX editors[6] were used under Solaris 8, and the teTeX distribution[8] was used under both Solaris 8 and Mac OS X. A modified version of the MIT thesis class was used, and a personalized bibliography style file (*sebplain.bst*), as well as the graphicx, hhline, lgrind, lscape, and supertabular packages. All of the files required to print this thesis can be obtained from Dr. Paul Cefola, and are on the CD containing this research project, in the *thesis/source* directory. To compile from a command prompt[1], type the following commands:

```
prompt% pdflatex sebthesis.tex
prompt% bibtex sebthesis
prompt% pdflatex sebthesis.tex
prompt% pdflatex sebthesis.tex
```

This will create a file called *sebthesis.pdf* in the same directory, which should be

---

[1] If using an editing package, the compilation can probably be run by pressing a series of buttons which issue commands corresponding to those listed. See the documentation of your particular LaTeX distribution for details.

identical to this paper copy. Please note that the repeated `pdflatex` commands are not superfluous — the first is required to set up the list of citations for the BibTeX bibliography creation, and the second and third are both required to properly create internal references (e.g. Appendix F).

Copies of the graphics created using Word or Excel in their original Microsoft Office format can be found in the *thesis/worddocs* directory.

Bibliography

[1] W. N. Barker and R. N. Wallner. The Accuracy of General Perturbations and Semianalytic Satellite Ephemeris Theories. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Halifax, Nova Scotia, Canada, 1995. AAS/AIAA. AAS 95-432.

[2] Sarah E. Bergstrom, Paul J. Cefola, Ronald J. Proulx, Andrey I. Nazarenko, and Vasiliy S. Yurasov. Validation Results of an Algorithm for Real-Time Atmospheric Density and Correction. In *Proceedings of the AAS/AIAA Spaceflight Mechanics Meeting*, San Antonio, TX, January 2002. AAS/AIAA.

[3] Sarah E. Bergstrom, Paul J. Cefola, Ronald J. Proulx, Andrey I. Nazarenko, and Vasiliy S. Yurasov. Atmospheric Density Correction Using Space Catalog Data. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Quebec City, Canada, August 2001. AAS/AIAA.

[4] Paul Cefola, Wayne McClain, and Dave Carter. Single Station Orbit Determination for Landsat 6. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Portland, OR, August 1990. AIAA/AAS. AAS 90-2923.

[5] National Geophysical Data Center. Solar-Terrestrial Physics Division FTP Site. Public FTP site. a_P and K_p data are in the `ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/KP_AP/` di-

rectory, and $F_{10.7}$ data (described as 2800 MHz, rather than 10.7 cm) are in the `ftp://ftp.ngdc.noaa.gov/STP/SOLAR_DATA/SOLAR_RADIO/FLUX/` directory.

[6] Gnu EMACS, version 20.7.1. Computer Software, May 2001. Available at http://www.gnu.org/software/emacs/emacs.html.

[7] Eugene M. Emme. *Aeronautics and Astronautics: An American Chronology of Science and Technology in the Exploration of Space, 1915-1960.* National Aeronautics and Space Administration, Washington, DC, 1961. Pages 77–88 were cited by http://www.hq.nasa.gov/office/pao/History/Timeline/1955-57.html.

[8] Thomas Esser. TeTeX, version 1.0. Computer software, February 2000. Available at http://www.tug.org/teTeX/.

[9] J.D. Fischer. The Evolution of Highly Eccentric Orbits. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1998. CSDL-T-1310.

[10] Henry F. Fliegel and Thomas C. Van Flandern. A Machine Algorithm for Processing Calendar Dates. *Communications of the ACM*, 11(10):657, October 1968.

[11] Michel Goosens, Frank Mittlebach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Longman, Inc., Reading, Massachusetts, 1994.

[12] Upper atmosphere of the Earth: Density model for ballistic maintenance of Earth artificial satellite flights. Technical Report GOST 25645.115-84, Moscow, 1984.

[13] George R. Granholm. Near-Real Time Atmospheric Density Model Correction Using Space Catalog Data. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2000. CSDL-T-1380.

[14] Goddard Trajectory Determination Software. Computer Software. SGI-UNIX version, based on PR-5.

[15] Goddard Trajectory Determination System (GTDS) User's Guide. Technical report, Charles Stark Draper Laboratory, Updated by Rick Metzinger: 1995. Copies available through Dr. Paul Cefola, (718)-981-5723.

[16] A. E. Hedin. A Revised Thermospheric Model Based on Mass Spectrometer and Incoherent Scatter Data: MSIS-83. *Journal of Geophysical Research*, 88(A12):10170–10188, December 1983.

[17] A. E. Hedin. Extension of the MSIS Thermosphere Model into the Middle and Lower Atmosphere. *Journal of Geophysical Research*, 96(A2):1159–1172, February 1991.

[18] A. E. Hedin, C. A. Reber, G. P. Newton, N. W. Spencer, H. C. Brinton, H. G. Mayr, and W. E. Potter. A Global Thermospheric Model Based on Mass Spectrometer and I ncoherent Scatter Data. MSIS 2. Composition. *Journal of Geophysical Research*, 82(16):2148–2156, jun 1977.

[19] A. E. Hedin, J. E. Salah, J. V. Evans, C. A. Reber, G. P Newton, N. W. Spencer, D. C. Kayser, D. Alcaydé, P. Bauer, L. Cogger, and J. P. McClure. A Global Thermospheric Model Based on Mass Spectrometer and Incoherent Scatter Data. MSIS 1. N2 Density and Temperature. *Journal of Geophysical Research*, 82(16):2139–2147, June 1977.

[20] Alan E. Hedin. MSIS-86 Thermospheric Model. *Journal of Geophysical Research*, 92(A5):4649–4662, May 1987.

[21] Herriges. NORAD General Perturbation Satellite Theories: An Independent Analysis. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1987 (revised in 1988). Copies available through Paul Cefola at the MIT Lincoln Laboratory.

[22] Felix R. Hoots and Richard G. France. Transformations Between Element Sets. Technical report, Space Command, United States Air Force, June 1986.

[23] Luigi G. Jacchia. Static Diffusion Models of the Upper Atmosphere with Empirical Temperature Profiles. *Smithsonian Contributions to Astrophysics*, 8(9):215–257, 1965.

[24] Luigi G. Jacchia. New Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles. SAO Special Report 313, Smithsonian Astrophysical Observatory, Cambridge, MA, May 1970.

[25] Luigi G. Jacchia. New Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles. SAO Special Report 332, Smithsonian Astrophysical Observatory, Cambridge, MA, May 1971.

[26] Luigi G. Jacchia. The Earth's Upper Atmosphere. *Sky and Telescope*, pages 155–159, 229–232, 294–299, March and April and May 1975. (This article was published as a three–part serial).

[27] C. Jaeck-Berger and F. Barlier. Review of Drag Effects on Satellite Orbits for Geodynamic Studies. In *The Use of Artificial Satellites for Geodesy and Geodynamics*, Proceedings of the International Symposium on Geodesy and Geodynamics, pages 275–311, Athens, Greece, May 1973. National Technical University of Athens.

[28] T. S. Kelso. CELESTRAK: Norad Two-Line Element Set Format. Web Page, December 2000. Accessed May 2002 at http://www.celestrak.com/NORAD/documentation/tle-fmt.shtml.

[29] Richard Koch and Dirk Olmes. TeXShop, version 1.19. Computer software, March 2002. Available at http://www.uoregon.edu/ koch/texshop/texshop.html.

[30] J.R. Kuhn. Data Set Layouts for the Goddard Trajectory Determination System (GTDS). Technical Report Contract NAS 5-24300, National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, MD, December 1979. Task Assignments 717 and 740.

[31] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1986.

[32] G. Laneve. Small Satellites for Aeronomic Missions in the Lower Theromosphere. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, pages 1999–2014, Sun Valley, ID, August 1997. Vol. 97, pt.2, Advances in the Astronautical Sciences.

[33] J. Lean. Solar EUV Irradiances and Indices. In D. Rees, editor, *COSPAR International Reference Atmosphere: 1986*, volume 8, number 5–6 of *Advances in Space Research*, chapter 7, pages 263–292. Pergamon Press, Inc., Elmsford, New York, 1988.

[34] F. A. Marcos. Accuracy of Atmosphere Drag Models at Low Satellite Altitudes. *Adv. Space Research*, 10(3):417–422, 1990.

[35] F. A. Marcos, M. Kendra, J. Griffin, J. Bass, J. Liu, and D. Larson. Precision Low Earth Orbit Determination Using Atmospheric Density Calibration. *Journal of the Astronautical Sciences*, 46:395, 1998.

[36] Frank A. Marcos and Felix R. Hoots. A Perpective on Neutral Density Progress. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Quebec City, Canada, August 2001. AAS/AIAA.

[37] Mathworks, Inc. MATLAB Computer Software, September 2000. Version 6.0.0.88 Release 12.

[38] R.W. Metzinger. Validation of the Workstation Version of R&D GTDS. Technical report, Charles Stark Draper Laboratory, Cambridge, MA, February 1993. Copies available through Dr. Paul Cefola, (781)-981-5723.

[39] James G. Miller. Atmospheric Density Model Errors and Variations in the Ballistic Coefficient. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Quebec City, Canada, August 2001. AAS 01-395.

[40] Andrey I. Nazarenko. A-priori and A-posteriori Orbit Prediction Errors Evaluation of Low Height Artificial Earth Satellites. *Cosmic Research*, 29(4), 1991.

[41] Andrey I. Nazarenko. Atmospheric Density Tracking Studies. Technical Report CSDL-C-6505, Scientific-Industrial Firm "NUCLON" for the Charles Stark Draper Laboratory, August 1999.

[42] Marcel Nicolet. La Constitution et la Composition de l'atmosphère supérieure. In C. Dewitt, J. Hieblot, and A. Lebean, editors, *Geophysics, The Earth's Environment*, pages 201–277. Gorden and Breach, Science Publishers, New York, NY, 1963. Article is in french.

[43] C. Pardini and L. Anselmo. Calibration of Semi-Empirical Atmosphere Models Through the Orbital Decay of Spherical Satellites. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*. AIAA/AAS, August 1999. AAS 99-384.

[44] J. Picone, A. Hedin, D. Drob, and J. Lean. NRLMSISE-00 Empirical Atmospheric Model: Comparisons to Data and Standard Models. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Quebec City, Canada, August 2001. AAS/AIAA. AAS 01-394.

[45] T.D. Platt and L. E. Herder. Ranking Satellite Propagators: A Statistical Approach. In *Proceedings of the 1995 Space Surveillance Workshop*, Lexington, MA, 1995. MIT Lincoln Laboratory. ESC-TR-95-022.

[46] Derek Price. The Concurrent Version System. Computer software, 2001. Available at http://www.cvshome.org.

[47] Gregor N. Purdy. *CVS Pocket Reference.* O'Reilly & Associates, Inc., Sebastol, CA, 2000.

[48] Charles E. Roberts. An Analytic Model for Upper Atmosphere Densities Based Upon Jacchia's 1970 Models. *Celestial Mechanics*, 4:368 377, December 1971.

[49] Ken Schatten. Solar Activity Prediction Methods. Web Page, September 1997. http://denali.gsfc.nasa.gov/926/schatten/sunpred.htm.

[50] Christian Schenk. MikTeX, version 2.1. Computer software, June 2001. Available at http://www.miktex.org.

[51] M. F. Storz. Satellite drag accuracy improvements estimated from orbital energy dissipation rates. *Advances in the Astronautical Sciences*, 103(2):1307–1327, 2000. AAS 99-385, presented at the *AAS/AIAA Astrodynamics Specialist Conference* in Girdwood, AK, August 1999.

[52] M. F. Storz. Modeling and Simulation Tool for the High Accuracy Satellite Drag Model. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Quebec City, Canada, August 2001. AAS/AIAA.

[53] Dick Struve and Peter Nagel. TeXShell, version 0.63. Computer software, October 2000. Available at http://www.projectory.de/texshell/index.html.

[54] Allen Thompson. FTP site for NORAD Two-Line Elements. Web Page. ftp://kilroy.jpl.nasa.gov/pub/space/elements/satelem, Accessed February, 2000. This site is no longer available, as of February 5, 2002.

[55] D. A. Vallado. *Fundamentals of Astrodynamics and Applications.* Space Technology series. McGraw-Hill, New York, NY, 1997.

[56] David Vallado. Private e-mail to Paul Cefola, May 2002.

[57] Rodney Viereck and J. Joselyn. Solar Cycle Effects on Thermospheric Density and Satellite Drag. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*. AAS/AIAA, August 1999. AAS 99-379.

[58] Jesco von Puttkamer. ISS Visibility Data. Web Page, May 2002. http://www.hq.nasa.gov/osf/station/viewing/issvis.html.

[59] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl*. O'Reilly & Associates, Inc., Sebastopol, CA, second edition, 1996.

[60] J. R. Wright. Sequential Orbit Determination with Auto-Correlated Gravity Modelling Errors. *Journal of Guidance and Control*, 4(3):304–309, May 1981.

About the Author

In 1999, while an intern at NASA Glenn Research Center, Sarah Bergstrom decided to concentrate her studies in Aerospace Engineering. She graduated in May of 2000 from Swarthmore College with a B.S. in Engineering and a minor in Classical Greek. She then enrolled at MIT in pursuit of a Master of Science in Aeronautical and Astronautical Engineering, focusing on orbital mechanics and control systems. After her expected graduation in June 2002, she intends to work on aerospace control systems at the Scientific Systems Company in Woburn, MA. She can be reached by email at *sbergst1@alum.swarthmore.edu* or *sarah_bergstrom@alum.mit.edu*.

THIS PAGE INTENTIONALLY LEFT BLANK

3231-9