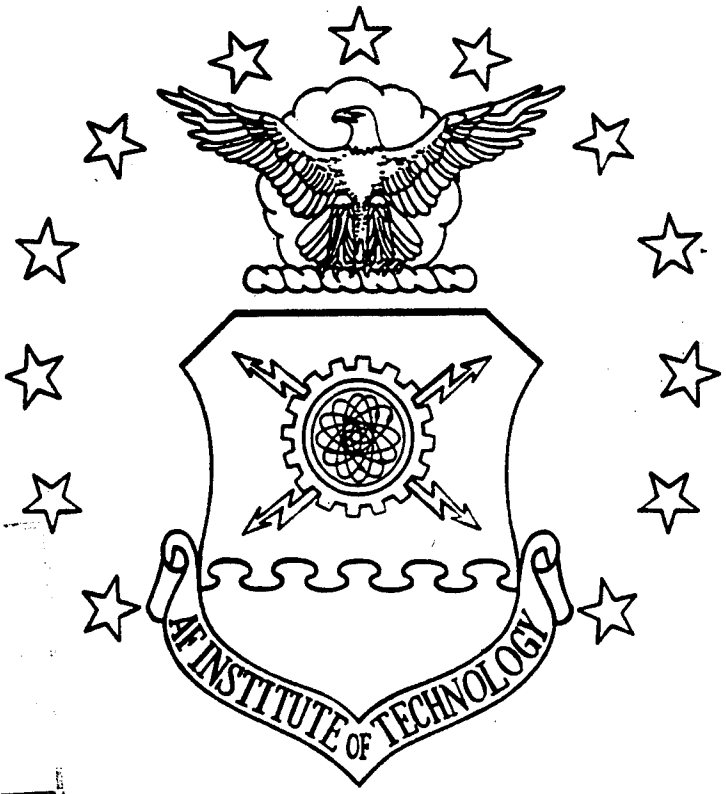

COMPUTING NORAD MEAN ORBITAL ELEMENTS FROM A STATE VECTOR

THESIS

Dwight E. Andersen, B.S.
Captain, USAF

AFIT/GSO/ENS/94D-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GSO/ENS/94D-01

COMPUTING NORAD MEAN ORBITAL ELEMENTS FROM A STATE VECTOR

THESIS

Dwight E. Andersen, B.S.
Captain, USAF

AFIT/GSO/ENS/94D-01

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Master's Thesis

4. TITLE AND SUBTITLE

Computing NORAD Mean Orbital Elements From A State Vector

5. FUNDING NUMBERS

6. AUTHOR(S)

Dwight E. Andersen, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GSO/ENS/94D-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NAIC/TANB
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

NORAD maintains and disseminates mean orbital elements on Earth-orbiting satellites in the form of Two-Line Element Sets (TLE). Five mathematical propagator models were developed for NORAD's use to predict the position and velocity using TLEs. This study investigated two approaches, Newton's method and direct iteration, to inverting this process by iterating to obtain NORAD-compatible mean orbital elements from a position and velocity state vector and the drag term. The Newton's iteration method was developed but not tested. The less computationally intensive direct iteration method was developed, coded in FORTRAN, and tested. The initial guess and subsequent corrections in the iterative process used the osculating elements computed from the state. The results showed the computation of the osculating elements to be unstable for low-Earth orbits with low eccentricity and moderate inclination, never converging to a solution for those cases. The scope of this research was limited to using two of the five propagators: Simplified General Perturbation Version 4 (SGP4) for low Earth orbits and SDP4 for deep space orbits.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Mean orbital elements, earth orbits, artificial satellites, iterations celestial mechanics	57
	16. PRICE CODE

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

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

*U.S.GPO:1993-0-350-770

AFIT/GSO/ENS/94D-01

COMPUTING NORAD MEAN ORBITAL ELEMENTS FROM A STATE VECTOR

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of Master of Science in Space Operations

Dwight E. Andersen, B.S.
Captain, USAF

December 1994

THESIS APPROVAL

STUDENT: Captain Dwight E. Andersen, USAF *CLASS:* GSO-94D

THESIS TITLE: Computing NORAD Mean Orbital Elements from a State Vector

DEFENSE DATE: November 22, 1994

COMMITTEE:	*NAME/DEPARTMENT*	*SIGNATURE*
Advisor ENS Representative	ROGER C. BURK, Maj, USAF Assistant Professor of Operations Research Department of Operational Sciences	Roger C Burk
Reader	WILLIAM E. WIESEL, JR., Ph.D. Professor of Astronautical Engineering Department of Aeronautics and Astronautics	William Wiesel

Acknowledgements

First I would like to thank my thesis committee members, Major Roger C. Burk (advisor) and Dr. William E. Wiesel (reader), for their efforts toward this project. Major Burk provided the necessary direction and sanity checks throughout the project evolution, keeping my endeavor focused and helping me to maintain the proper perspective. He suffered through the numerous report drafts along the way and I know it was "crushingly boring" at times. I want him to know that I will always be grateful for his patience and understanding. Dr. Wiesel provided the technical expertise throughout the project and it was he who proposed the Newton's method of iteration, which I will someday get to work. His knowledge in orbital mechanics appears to have no bound. Dr. Wiesel can teach anyone...even a Space Operations Officer. My respect for him is absolute. In addition to my committee members, I thank Lieutenant Colonel Thomas S. Kelso. Lt Col Kelso's pure love for the Space Operations field was, is, and always will be an inspiration. He was omitted from my official thesis committee due to a then-pending move. Nonetheless, I will always remember him as an unofficial co-advisor. He is solely responsible (or should I say to blame) for my involvement in this work. For his sincere interest in this research and the enormous amount of help he provided to me even after his move, I will forever be indebted to him. Lastly, but most important of all, I want to thank "my girls": Marcia, my wife, and my daughters, Tiffany and Samantha (Sam). They struggled through the 18 months at AFIT more than I. I will always feel I cheated them out of their rightfully deserved time as husband and "Dad". To Marcia: You're the best thing that has ever happened to me. You're smarter, more organized, and much better looking than I. As the years continue to roll by I realize more and more that I'd be nothing without you. All my successes are because of you. I'll love you forever. Thanx!

Dwight E. Andersen

Table of Contents

Page

Page

List of Tables

Abstract

NORAD maintains and disseminates mean orbital elements on Earth-orbiting satellites in the form of Two-Line Element Sets (TLE). Five mathematical propagator models were developed for NORAD's use to predict the position and velocity using TLEs. This study investigated two approaches, Newton's method and direct iteration, to inverting this process by iterating to obtain NORAD-compatible mean orbital elements from a position and velocity state vector and the drag term. The Newton's iteration method was developed but not tested. The less computationally intensive direct iteration method was developed, coded in FORTRAN, and tested. The initial guess and subsequent corrections in the iterative process used the osculating elements computed from the state. The results showed the computation of the osculating elements to be unstable for low-Earth orbits with low eccentricity and moderate inclination, never converging to a solution for those cases. The scope of this research was limited to using two of the five propagators: Simplified General Perturbation Version 4 (SGP4) for low Earth orbits and SDP4 for deep space orbits.

COMPUTING NORAD MEAN ORBITAL ELEMENTS FROM A STATE VECTOR

I. INTRODUCTION

1.1 Background.

Under sponsorship of the Joint National Intelligence Defense Staff (JNIDS), the Air Force Institute of Technology (AFIT) developed Satellite Modeler (SM), a virtual near-Earth space environment computer application. In individual theses, Captain David L. Pond and Captain Andrea A. Kunz, in 1992 and 1993 respectively, developed the application to run on a Silicon Graphics workstation [Pon92] [Kun93]. In its present form, SM allows interaction with a three-dimensional virtual environment to graphically visualize orbital motion of Earth-orbiting satellites. The interaction allows the user to chose individual satellites, constellations, viewpoint, and other graphical depictions, e.g. satellite orbit trails.

The North American Aerospace Defense Command (NORAD) maintains orbital data for Earth-orbiting objects. These objects are classified as near-Earth (orbital periods less than 225 minutes) or deep-space (periods greater than or equal to 225). NORAD distributes these orbital data to various users in the form of a two-line mean orbital element set (TLE) for each object. Five mathematical models were developed for NORAD's use to propagate TLEs forward in time to obtain a position and velocity of the space object. Together, the three-dimensional position and velocity are referred to as the state of the object. The five propagation models are based on general perturbation theory but implement different models for gravitational and drag effects, and also differ in the manner in which the differential equations of motion are integrated. The Simplified General Perturbations (SGP), SGP Version 4 (SGP4), and SGP Version 8 (SGP8)

prediction models are used for near-Earth objects. The deep-space models are SDP4 and SDP8 [Hoo80]. SM takes as input a NORAD TLE. The state of the satellite is then propagated forward in time through either SGP4 or SDP4 general perturbation propagation models.

The National Air Intelligence Center (NAIC) plans to use SM as an analytical tool, but to do so requires that SM be enhanced to include many additional interactive modules. The primary additions involve interactive manipulation of the satellite orbit, modifying the current dynamical state of the satellite through a thrusting maneuver.

NAIC wishes to add the capability of changing the satellite orbit and "flying" this new orbit for analysis. They propose two general inputs to perform this orbit modification. The first is to impart a specified maneuvering thrust, either impulsive or non-impulsive. The second is to specify a subpoint for the satellite to pass over.

The orbit modification can be broken down into two parts, powered and non-powered flight. The state of the satellite must be propagated through the powered flight regime by a means other than SGP4 or SDP4, since these models are for satellites under natural acceleration. For the impulsive maneuver, modification of the state is a matter of adding the velocity change to the velocity at the maneuver time. The non-impulsive maneuver is more complex, requiring the inertial position, velocity, and attitude to be modified throughout the duration of the burn. Propagation through the powered flight regime requires integrating the equations of motion throughout the duration. After the maneuver is complete the satellite returns to non-powered natural flight.

SM, using SGP4 and SDP4, takes as input NORAD mean elements along with the current simulation time and computes the position and velocity of the satellite. Once the orbit is modified through a thrusting maneuver, the original NORAD elements no longer describe the satellite's orbit. At this point the new orbit must be propagated forward in time by means other than the SGP4 or SDP4 model. A numerical integrator could be

employed to propagate those satellites which have had their original orbits modified by thrusting maneuvers, but this approach would yield the computational inefficiency of two different methods of satellite propagation within SM. However, if the NORAD-compatible mean elements for the modified orbit could be obtained, these could then be substituted for the original two-line element set and SGP4 or SDP4 could be then employed for propagation.

In addition to NAIC, many others in the space community use the NORAD element sets and associated propagation models to predict satellite states forward in time. Therefore, devising a means of obtaining compatible mean element sets from position and velocity would have a much broader application to the space community in general.

1.2 Problem Statement.

NAIC, and the space community at large, need a method of obtaining NORAD-compatible mean orbital elements from a given satellite state vector. The problem is to take a given set of mean elements, propagate the state forward in time, add an impulsive maneuver, and compute a NORAD-compatible TLE for the new orbit.

1.3 Scope. The effort of this thesis was concentrated in devising a method for obtaining NORAD-compatible mean orbital elements for a satellite, given the known position, velocity, and drag term. Thrusting maneuvers were limited to impulsive. The drag term was assumed to remain unchanged, which implied small orbit changes. While the non-impulsive thrust was not addressed, nor was obtaining an orbit which overflies a specified subpoint, NAIC's future needs for enhancements to SM were kept in mind. A modular approach was employed which left the prediction routines unmodified. Although five prediction models exist for NORAD elements, this research dealt specifically with SGP4 and SDP4, the only prediction models currently used by SM.

II. DETAILED PROBLEM ANALYSIS AND PREVIOUS WORK

2.1 Overview.

NORAD generates orbital element sets on all Earth-orbiting space objects. These element sets are general perturbation mean elements constructed by a least squares estimation from observations of the Space Surveillance Network (SSN). These element sets are periodically updated and provided to users. A discussion of the NORAD elements and their published format follows in section 2.2 .

In *Spacetrack Report No. 3: Models for Propagation of NORAD Element Sets* by Hoots and Roehrich (1980), the introduction emphasizes:

> *The most important point to be noted is that not just any prediction model will suffice. The NORAD element sets are "mean" values obtained by removing periodic variations in a particular way. In order to obtain good predictions, these periodic variations must be reconstructed (by the prediction model) in exactly the same way they were removed by NORAD.* [Hoo80: 1]

This implies the reverse, that if prediction is via NORAD propagators, not just any mean element set will suffice. Therefore, the problem is to obtain mean elements obtained for the modified orbit ensuring they are compatible with the NORAD prediction routines.

Treating the propagation models as generic vector functions which compute, for a given time, the satellite state from the mean elements, the routines could each be represented as

$$\mathbf{f(TLE, t) = X} \qquad (2.1)$$

where **f** is the prediction model represented as a function, **TLE** is the vector of orbital elements in the NORAD Two-Line Element set, **t** is the time of prediction, and **X** is the position-velocity state vector in Earth Centered Inertial (ECI) coordinates.

To obtain the TLE, this formula needs to be inverted. The inverse of **f** is needed which calculates the mean elements from the state. Inverting the complex prediction routines analytically is difficult and perhaps impossible. However, an iteration technique, e.g. Newton's multi-dimensional method, could be employed using an initial guess at the solution then iteratively correcting the guess until a specified convergence criterion is reached.

2.2 NORAD Mean Orbital Elements.

The six classical Keplerian orbital elements are eccentricity (e), inclination (i), right ascension of the ascending node (Ω), argument of perigee (ω), semi-major axis (a), and time of perigee passage (T_0). These six elements describe the orbit and its orientation in space. With a time specified, the phasing of the object within the orbit can be determined. In the perfect two-body case, the orbit and its orientation remain constant. In actuality, the orbit is perturbed by small variations in the orbital elements. The major perturbations arise from the oblateness of the Earth, atmospheric drag, and lunar and solar gravitational effects. These perturbations cause periodic and secular variations in the orbit. NORAD TLEs are mean element sets which have been generated by averaging out these variations in a specific manner. The TLEs include the first four of the classical elements, subscripting each with an "o" to denote mean values, but replace the last two with the mean anomaly (M_o) and the mean motion (n_o). A drag term (B*) and two pseudo drag terms, the first and second time derivatives of mean motion with scale factors ($\dot{n}/2$ and $\ddot{n}/6$), are also included. These drag terms are used by the prediction models to account for atmospheric drag effects. The reader is referred to any introductory orbital mechanics text, e.g. *Fundamentals of Astrodynamics* by Bate Mueller and White [Bat71], if unfamiliar with the above orbital elements. The NORAD two-line element sets are published in ASCII form to outside users in actually three lines

each. The first line contains a satellite identifier and is not used by the prediction models. The TLEs also contain the epoch time for the data, as well as additional information that is not used by the propagators. The exact line by line and column by column format for the TLE is included in Appendix A.

The elements in the published TLEs are not in the units required by the propagators and therefore must be converted prior to calling the prediction routines. Table 2.1 provides the TLE elements and their published units as well as the units used by the prediction routines.

Table 2.1. Units of TLE

Element	Published Units	Propagator Units
e_o	unitless	unitless
i_o	degrees	radians
Ω_o	degrees	radians
ω_o	degrees	radians
M_o	degrees	radians
n_o	revolutions/day	radians/minute
B^*	Earth-radii^{-1}	Earth-radii^{-1}
$\dot{n}/2$	revolutions/day^2	radians/minute^2
$\ddot{n}/6$	revolutions/day^3	radians/minute^3

2.3 Propagation Models.

Five different mathematical satellite prediction models exist for propagating Earth-orbiting satellites with NORAD mean elements. SGP, SGP4, and SGP8 are models for near-Earth satellites, while SDP4 and SDP8 are for deep-space satellites.

SM uses SGP4 and SDP4 prediction models. Atmospheric drag is handled via B*. The pseudo-drag terms $\dot{n}/2$ and $\ddot{n}/6$ are used in SGP only and are of no consequence when predicting orbits with SGP4 or SDP4. SGP4 was developed by Crawford in 1970. It is a simplification of the analytical theory of Lane and Crawford (1969). SGP4 uses a power density function for modeling the atmosphere and the solution of Brouwer (1959) for its gravitational model [Hoo80: 1]. SDP4 was developed by Hujsak (1979) by extending SGP4 to deep-space orbits. It included gravitational effects of the sun and moon along with certain sectoral and tesseral Earth harmonics. These harmonics are significant in accurately predicting the state for half-day and one-day period orbits [Hoo80: 1]. The reader is referred to *Spacetrack Report No. 3: Models for Propagation of NORAD Element Sets* by Hoots and Roehrich (1980) [Hoo80] for a detailed discussion of each of these two models as well as the remaining three models and their differences. In addition, the FORTRAN computer code for each of the five prediction models and the required subprograms can also be found in *Spacetrack Report No. 3* .

2.4 Previous Related Work.

A literature search revealed two previous works which involved obtaining NORAD-compatible mean orbital elements from a known orbiting satellite state. Both treat the prediction models as "black-box" functions and iterate to obtain a solution, but differ in their approach.

2.4.1 Vector to Two-Line Elements (VEC2TLE) Software. In May of 1994, Kenneth J. Ernandes copyrighted his *Vector to Two-Line Elements (VEC2TLE) Version 9425* shareware software for International Business Machines (IBM) compatible personal computers using Microsoft Disk Operating System (MSDOS) [Enr94]. It provides the user with the ability to convert near-Earth position/velocity/time state vectors in various formats to NORAD-compatible TLEs. It uses SGP or SGP4 prediction models.

The discovery of Mr. Ernandes' software came late in this research project. Mr. Ernandes, in conversations with this author, understandably displayed concern over revealing details of his solution technique, wishing to protect his substantial investment in developing his software. However, when this author discussed his proposed method of employing Newton's iteration method using numerically calculated partial derivatives with the osculating orbital elements as an initial guess (discussed later in Chapter III) Mr. Ernandes replied, "Fundamentally, our techniques are the same."

Mr. Ernandes was willing to make available the computer code provided certain non-disclosure conditions were met. Attempts to meet his requirements within the time and funding constraints of this research were not successful.

2.4.2 Conversion of Osculating Elements to Mean Elements. In section 3.4.8 of *NORAD Technical Publication: SPADOC Computation Center Computer Program Product Specification Mathematical Foundation for SSC Astrodynamic Theory*, a general algorithm is presented for converting osculating elements to mean elements via a direct iteration scheme using the osculating elements as an initial guess [NOR82: 147]. The algorithm is implemented in program GP2SP. However, attempts to obtain this computer code from AFSPC/CNY failed. CNY stated that policy prevented them from releasing this or any other computer code, other than the five prediction models.

III DEVELOPMENT OF PROBLEM SOLUTION

3.1 Overview.

A top-level analysis of the problem produced a general algorithm to solve for the mean elements from a known satellite state and drag term at some given time. The algorithm called for an iterative approach. Two methods were investigated. Initially, Newton's multi-dimensional iteration technique was applied analytically as an approach to converge on a solution. Implementing this approach numerically on a computer posed complexities which are discussed later in this report. Additional research, discussed in Section 2.4.2, revealed a simpler method of directly iterating on a solution. Since Mr. Kenneth J. Ernandes' VEC2TLE computer program, discussed in Section 2.4.1, is based on Newton's method, it was decided to attack the problem from another perspective and the direct iteration technique was developed and tested. However, it did not provide convergence in each case. Therefore both approaches are presented as background for future research.

3.2 General Algorithm.

The task was to take a given TLE, propagate forward in time with SGP4 or SDP4, add an impulsive thrust, and calculate a NORAD-compatible TLE for the new orbit. The general algorithm developed for computer implementation was as follows:

1) Read in the original TLE and time of thrust application.
2) Convert the elements of the TLE to the internal units used in the propagators.
3) Propagate the state forward to the time of thrust application.
4) Add the impulsive thrusting maneuver (ΔV) to the velocity of the state.
5) Calculate a first guess at the solution.

6) Iterate until the desired convergence accuracy is reached.

This research was in the areas of steps 5 and 6 above. With or without the maneuver in step 4, one must still solve for the mean elements which, when input to the appropriate prediction model, yield the desired state. The computer algorithm, however, does allow for an impulsive maneuver, since the problem statement called for it. As will be seen in later results, it also served as an additional validation of the code to show that one could modify the state with a thrust, obtain a new TLE, then reverse the process by negating the thrust and obtain the original TLE.

3.3 Initial Guess.

Both approaches required an initial guess at the solution. At any given time the actual state vector associated with the true perturbed orbit is identical to the state vector of the fictitious two-body orbit. Under the theory of General Perturbation, the underlying theory of SGP, SGP4, SDP4, SGP8, and SDP8, the assumption is that the perturbations remain small [Wie89-2]. At any particular time, the real orbit and the instantaneous, or osculating, orbit meet and share the same state vector. Therefore the osculating elements should provide a close approximation for the first guess at the mean elements in the iteration scheme.

3.4 Approach 1: Newton's Method [Bur85: 496-509]

A multi-dimensional Newton's iteration method allows one to solve for the root of a function by expanding the function via a Taylor series. This method is generally expected to give a quadratic rate of convergence as long as the guess is close to the actual solution [Bur85: 498].

3.4.1 Analytic Development. Denoting the prediction routines as a function **f** as in equation 2.1, subtracting the actual state from both sides of the equation, and denoting the left hand side as a new function **F**, yields:

$$\mathbf{f(TLE, t) - X = 0} \tag{3.1}$$

$$\mathbf{F \equiv f(TLE, t) - X = 0} \tag{3.2}$$

If instead of **TLE** some approximation $\underline{\mathbf{TLE}}$ is used, the right hand side of equation 3.2 will not yield exactly zero but some error, $\Delta\mathbf{X}$, until the correct solution for **TLE** is obtained, and even then there will be a precision limit when implementing it on a digital computer. As in all mathematics performed on a finite-word-length computer, the iteration should be continued until the $\Delta\mathbf{X}$ becomes less than some predefined tolerance.

Since the function **f** is evaluated at a given time **t** one can think of the time as being a constant throughout the iteration. Likewise the drag elements of **TLE** are either of no consequence when using SGP4 or SDP4, as in the case of $\dot{n}/2$ and $\ddot{n}/6$, or, as with B*, can be assumed to be constant for small thrusting modifications to the orbit, and so can also be considered as constants within the function **f**. This reduces the **TLE** vector to six elements allowed to vary during the iteration: e_o, i_o, Ω_o, ω_o, M_o, and n_o.

Newton's method in solving for the true **TLE** includes expanding equation 3.2 in a Taylor series:

$$\mathbf{F(TLE) = F(\underline{TLE}) + (TLE - \underline{TLE})\, J(\underline{TLE}) + H.O.T.} \tag{3.3}$$

where **TLE** is the unknown vector of mean elements, $\underline{\mathbf{TLE}}$ is the guess, **J** is the Jacobian matrix of **F**, and **H.O.T** refers to "higher order terms." Truncating equation 3.3 to first order and substituting **F(TLE) = 0** by definition , equation 3.3 can now be written as:

$$\mathbf{F}(\underline{\mathbf{TLE}}) + (\mathbf{TLE} - \underline{\mathbf{TLE}})\, \mathbf{J}(\underline{\mathbf{TLE}}) \approx \mathbf{0} \tag{3.4}$$

The approximation sign is, for the rest of the derivation, replaced with an equality sign. Rearranging equation 3.4 produces:

$$\Delta\mathbf{TLE} = -\, \mathbf{J}^{-1}(\underline{\mathbf{TLE}})\, \mathbf{F}(\underline{\mathbf{TLE}}) \tag{3.5}$$

where:

$$\Delta\mathbf{TLE} \equiv (\mathbf{TLE} - \underline{\mathbf{TLE}})$$

Then in iteration notation, the TLE for the k^{th} iteration is calculated from:

$$\Delta\mathbf{TLE}^{(k-1)} = -\, \mathbf{J}^{-1}(\mathbf{TLE}^{(k-1)})\, \mathbf{F}(\mathbf{TLE}^{(k-1)}) \tag{3.6}$$

$$\mathbf{TLE}^{(k)} = \mathbf{TLE}^{(k-1)} + \Delta\mathbf{TLE}^{(k-1)} \tag{3.7}$$

where the superscripts in parentheses denote the iteration number.

3.4.2 Implementation Complexities. The computer code development encountered some complexities .

A weakness of Newton's method for solving systems of n nonlinear equations is that the Jacobian matrix must be computed for each iteration and an n x n linear system solved that involves this matrix. The Jacobian for this problem is the matrix whose n^2 elements are the partial derivatives of the function **F** with respect to the elements of **TLE**. Analytically, the individual elements of **J** are:

$$\mathbf{J}(\underline{\mathbf{TLE}})_{i,j} = \frac{\partial \mathbf{F}_i(\underline{\mathbf{TLE}})}{\partial \underline{\mathbf{TLE}}_j} \tag{3.8}$$

where **i** and **j** denote the row and column respectively with respect to the Jacobian and the individual elements of the **F** and **TLE** vectors. Since the prediction routines are complicated computer subroutines, taking the partial derivatives analytically is not feasible. Therefore they need to be obtained numerically by finite differences. This finite difference form of the Jacobian for the k^{th} iteration is:

$$\frac{\partial \mathbf{F}_i(\underline{\mathbf{TLE}}^{(k)})}{\partial \underline{\mathbf{TLE}}_j} = \frac{\mathbf{F}_i(\underline{\mathbf{TLE}}^{(k)} + \xi_j \mathbf{h}) - \mathbf{F}_i(\underline{\mathbf{TLE}}^{(k)})}{\mathbf{h}} \tag{3.9}$$

where **h** is small in absolute value and ξ_j is the vector whose only nonzero entry is a one in the j^{th} coordinate. For this case (n=6), seven calls to the propagator are required for each iteration: one to evaluate **F** and six to approximate the Jacobian matrix. There are quasi-Newton techniques one can employ to reduce the number of computations per iteration by replacing the Jacobian matrix with an approximation that is updated at each iteration. These methods, however, have disadvantages. Newton's method is self correcting where quasi-Newton methods are not. Newton's method will generally correct for round-off error with successive iterations, quasi-Newton methods will not. Another disadvantage is that the quadratic convergence of Newton's method is replaced with superlinear convergence when modified by the techniques of the quasi-Newton methods [Bur85: 504, 505].

In addition to the number of computations required for Newton's method, implementing equation 3.9 required determining a suitable **h** to capture the slope

information for all orbits. The large dynamic range in values of the different orbital elements meant different values of h were required.

Due to the complexity in implementing this approach, coupled with the loss of convergence speed, a linearly converging direct iteration approach was developed and tested as an alternative approach. It was also felt that since Mr. Kenneth J. Ernandes' VEC2TLE computer program, discussed in Section 2.4.1, was based on Newton's method, a different approach might yield additional insight to the problem.

3.5 Approach 2: Direct Iteration.

As stated earlier in this report, the AFSPC/CNY-controlled program GP2SP implements a direct iteration algorithm for solving the mean elements, using the osculating elements as a first guess. The only convergence criterion used is the mean motion since: *"This is the most important element when predicting into the future"* [NOR82]. Rather than converging on one element, the decision was made to iterate until all elements of the state vector were within the desired accuracy tolerance, ε. The direct iteration algorithm implemented for this research was as follows:

1) Start with the osculating elements corresponding to the true state as the initial guess for the mean elements.
2) Determine the prediction model based on the orbital period.
3) Calculate the state vector associated with these elements with the selected prediction model.
4) Correct the mean elements by approximating the correction with the change in the osculating elements between the current iteration step and the last. The new guess for the mean elements becomes the last guess plus this correction.

3) Return to step 2 and repeat until the change in the state of two successive steps is less than or equal to ε.

3.6 Computer Code Implementation.

The general algorithm was implemented in FORTRAN and run on a SUN workstation. SM is written in C and C++ but NAIC, and the space community at large, have several other potential target applications for this research based on FORTRAN so the decision was made to code in FORTRAN and translate it into C++ and implement it into SM at a later date. The scope of this research was to develop the procedure only.

The procedure for obtaining the mean elements for an orbit which undergoes a ΔV at some time equal to or greater than the TLE epoch follows:

1) Read in data: TLE, maneuver time t, ΔV, and ε.
2) Convert units to those used internally by the prediction models.
3) Determine the state vector X at time t via the appropriate propagator.
4) Add the ΔV to the state: $X = X + \Delta V$.
5) Set the new epoch time to t.
6) Calculate the osculating elements, Y, associated with the modified state.
7) Let $TLE^{(0)} = Y$. The value of B* remains as its original value in TLE.
8) Determine the appropriate prediction model based on the orbital period.
9) Calculate the state from the appropriate prediction model: $X^{(i)} = f(TLE^{(i)})$.
10) $\Delta X^{(i)} = X - X^{(i)}$.
11) Calculate the associated $Y^{(i)}$ from $X^{(i)}$.
12) $\Delta TLE^{(i)} = Y - Y^{(i)}$.
13) If $\Delta X^{(i)} \leq \varepsilon$ then stop the iteration and assign $TLE = TLE^{(i)}$.
14) Else $TLE^{(i+1)} = TLE^{(i)} + \Delta TLE^{(i)}$, increment i and return to step 8.

In addition to the above, provisions in the computer code were made to propagate forward in time the orbit once the mean element set had converged, if the user so desires. This feature is contained within the main DRIVER program of *Space Track Report No. 3* and was included in the DRIVER program of this research project for testing purposes and validation of test cases.

A subroutine for computing the osculating elements from a state vector was written combining methods from *NORAD TP SSC 008*; the works of Wiesel (1989); and Bate, Mueller, and White (1971) [NOR82; Wie89; Bat71]. The FORTRAN code can be found in Appendix B.

3.7 Computer Code Validation.

The FORTRAN code for the prediction models, SGP4 and SDP4, was obtained from *Space Track Report No. 3* and used without modification. However, a compiler option was set to declare all real variables as double precision variables. Initial validation of the prediction models was made to ensure that the results of the test cases within *Space Track Report No. 3* could be reproduced. Those results were reproduced allowing for the difference in precision.

The subprogram RV2OSC, used to compute the osculating elements from the state, was validated with several test cases from textbook examples and previous coursework.

Two methods were employed for validating the algorithm developed for this research. The first was to input a TLE and propagate the state without adding a ΔV, then iterate to see if convergence to the original TLE was obtained. The second method was to modify the state vector with a ΔV, iterate to get a new TLE, then using this newly obtained TLE, repeat the procedure negating the ΔV and iterate to see if the original TLE was produced. Both validation methods were successful.

3.8 Data Runs.

Thirty data sets were run using a variety of orbits. Both near-Earth and deep-space orbits were utilized, as were orbits with low eccentricity, high eccentricity, low inclination, moderate inclination, and near-polar inclination. The data sets and the associated results of these runs are detailed in the following chapter.

IV. RESULTS

4.1 General Overview of Results.

Fifteen initial data sets with varying orbital parameters were run. The convergence criteria used were one centimeter in position and one centimeter per second in velocity. The number of iterations required to reach convergence ranged from four to 175. Out of the 15 runs, four failed to converge. Initially a larger portion of the cases failed to converge and run time errors occurred. Further detailed software testing revealed that in certain cases, the initial guess calculated for the eccentricity was far enough from the actual value that the correction produced a negative value that increased, in absolute value, with each iteration. Once the divergence of the eccentricity produced an absolute value greater than unity the prediction models yielded errors trying to perform operations which are undefined with such values. Within the correction step of the iteration portion of the DRIVER program, the eccentricity was forced to be non-negative. This corrected all previously non-converging cases except for cases 6,7,8, and 10.

Table 4.1 is a summary of the results. The values in the table have been rounded off to aid in readability.

Table 4.1. Initial Results Overview For Runs 1 - 11

Run	Satellite	Inclination	Node	Eccentricity	Argument of Perigee	Mean Anamoly	Σ (see note 1)	Mean Motion	Prediction Model	Number of Iterations
1		73	116	0.00867	53	111	163	16	SGP4	8
1a		73	116	0.00867	53	111	163	16	SGP4	4
1b		73	116	0.20658	161	2	163	11	SGP4	8
2		47	230	0.73180	47	10	58	2	SDP4	4
2a		47	230	0.73180	47	10	58	2	SDP4	4
2b		47	230	0.57356	29	30	59	5	SDP4	4
3	TDRS4	0	252	0.00008	191	311	502	1	SDP4	114
4	CRRES	18	246	0.71975	83	347	429	2	SDP4	4
5	SPOT 1	99	104	0.00006	84	276	360	14	SGP4	175
6	Mir	52	171	0.00044	243	117	360	16	SGP4	Did NOT Converge
7	Kvant-1	52	176	0.00043	248	112	360	16	SGP4	Did NOT Converge
8	EUVE	28	276	0.00105	240	120	360	15	SGP4	Did NOT Converge
9	ProgressM17	52	153	0.00630	192	168	360	16	SGP4	8
10	HST Array	28	77	0.00050	183	177	360	15	SGP4	Did NOT Converge
11	1994004A	67	82	0.00341	164	196	360	16	SGP4	15

All angles in degrees. Mean motion in revolutions per day.
note 1: Summation of two preceding column entries.

4.2 Detailed Results.

The TLEs used in the first 6 cases were taken from two test cases in *Space Track Report No. 3* [Hoo80]. The TLEs of runs 3 through 11 were taken from actual published NORAD element sets. Runs 1, 1a, 1b, 2, 2a, 2b, 3, 4, 5, 9, and 11 converged while 6, 7, 8, and 10 did not.

4.2.1 Validation Runs. Runs 1, 1a, 1b, 2, 2a, and 2b were taken from the test cases of *Space Track Report No. 3* and used to show that a ΔV could be added to the state to obtain a new TLE then, by negating the maneuver, one could return to the original TLE. The report provided one test case for each of the five prediction models. The data used in Run 1 were from the test case for SGP4 and Run 2 was from the test case for SDP4. Run 1 converged on the original TLE using the iteration scheme starting with the initial guess of the osculating elements computed from the state vector. Run 1a took the state from Run 1, added a 10% ΔV in all three directions and computed a new

TLE. Run 1b reversed the process by using the new TLE of 1a: by negating the previous ΔV, the original TLE was converged upon. The deep-space runs of 2, 2a, and 2b were analogous to 1, 1a, and 1b except a negative 10% ΔV was applied.

4.2.2 Converging Cases. Runs 3, 4, and 5 converged in 114, 4, and 175 iterations respectively. Runs 3 and 5 were orbits of low eccentricity (on the order of 10^{-5}) and therefore computing the osculating elements at each step produced poor guesses close to the zero eccentricity singularity. Run 4 had an eccentricity on the order of 10^{-1} and converged much faster due to more stable osculating element computations.

Runs 9 and 11 were for near-Earth satellite orbits of moderate inclination, 52 and 67 degrees respectively, and eccentricity on the order of 10^{-3}. Convergence was in 8 and 15 iterations respectively.

Most of the runs converged in four to eight iterations. Interactively running the computer program on these cases, results were returned in less than one second actual elapsed time. Case 5 was at the extreme, taking 175 iterations to converge to the specified tolerance. Even this case returned results within four seconds total elapsed time.

4.2.3 Non-converging Cases. Cases 6, 7, 8, and 10 never converged. The errors for these runs came within five kilometers for position and three meters per second for velocity, not the one centimeter for position and one centimeter per second for velocity convergence criteria. Runs 6 and 7 were for the Mir and Kvant-1 respectively. These had moderate 52 degree inclinations and low eccentricities of approximately 4×10^{-4}. Runs 8 and 10 were of lower inclinations of approximately 28 degrees. The eccentricities of these cases were 1×10^{-4} and 5×10^{-3} respectively. In all the cases that failed to converge the sums of ω_0 and M_0 were approximately 360 degrees. Initially this appeared

to be a point of singularity possibly explaining the failure to converge. However, runs 5, 9, and 11 also had these same angle summations and yet those cases converged. An analysis of the output values for the individual iterations of the non-converging cases showed either ω_0 and M_0 would approach the extremes of one going toward 360 degrees and the other going to zero or wildly swinging throughout the orbit plane.

4.2.4 Additional Testing for the Non-converging Cases. Further testing was performed on cases 6, 7, 8, and 10 to try to get them to converge. The hypothesis was that if one propagated the orbit away from the node, i.e. where $\omega_0 + M_0 = 360$ degrees, and set the new epoch at this future time then one might escape the instability in the iterations. Additional runs were made for each of the four non-converging cases pushing the epoch forward in time and away from node crossing. The instability remained and still convergence was not achieved.

Various ΔVs were added to each case to see if the orbits could be modified to where they would converge. In each case, adding a ΔV which modified the eccentricity to a value on the order of 10^{-3} produced an orbit which did converge. The ΔV required to do this was approximately 10 m/s. However, reversing the process and negating the ΔV failed to converge.

The input data file format and sample data runs for both converging and non-converging cases can be found in Appendix C.

V. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions.

The problem was to take a given set of NORAD mean elements, propagate the state forward in time with either SGP4 or SDP4 prediction model, add a thrusting maneuver, and compute NORAD elements for the new orbit. The results in Chapter IV show that it was possible to obtain the mean elements from the position and velocity state, using the original B* drag term, with the direct iteration method described in chapter III, but not for all orbits when accuracy of one centimeter in position and one centimeter per second in velocity was desired. Convergence to this accuracy was never achieved for four of the 15 initial data runs. The failed runs all had in common that the sum of the argument of perigee and mean anomaly was approximately 360 degrees, i.e. at node crossing. It is a standard practice to place the epoch at the node crossing. However, for some orbits with such an angle sum the approach did yield the correct solution. Additional testing on the failed runs showed that even when the orbits were propagated forward away from the node crossing, where the sum of the argument of perigee and mean anomaly was not 360 degrees, the iterations were still unstable and failed to converge. Low eccentricity of approximately 10^{-4} appears to be a factor causing the instability in the method used to calculate the mean elements, as does the inclination. But low eccentricity by itself does not mean the iteration will not converge. The TLE for SPOT 1 (Run 3) converged, though taking 175 iterations, with an eccentricity of 0.00006 but the inclination was 99 degrees. The TLE for TDRS4 converged, again taking a large number of iterations (114), with an eccentricity of 0.00008 and an inclination of nearly zero degrees. However, runs for orbits of moderate inclination and low eccentricities failed to converge. When the non-converging cases were modified with ΔVs to push the eccentricity to a value on the order of 10-3, they all converged even with the moderate

inclinations. Therefore it appeared that eccentricity drove the instability at moderate inclinations and its affect was lessened as the inclinations moved toward zero or 90 degrees, equatorial or polar orbits.

The correction to the mean elements for each iteration was made from the osculating elements computed from the state provided by the prediction models. The results showed that the osculating elements were not always good enough approximations to yield convergence.

The computer program developed for this research was not completely optimized for execution speed. Nonetheless, most runs executed in less than one second actual elapsed time running on the Sun workstation, while the case requiring the most iterations still executed in less than four seconds. Optimizing the code and running it on the much faster Silicon Graphics workstation should provide near instantaneous execution.

5.2 Recommendations for Future Research.

As the work progressed during this research project it became apparent that the problem solution is one which has great potential in space operations/analysis and yet is a substantial problem to solve for all cases. In order to solve this problem entirely, extensions to this research are needed. The following are recommendations for six areas of future research related to this topic.

5.2.1 Additional Testing. Additional testing needs to be performed to more clearly define for which types of orbits the algorithm does not converge. The code developed for this project should be modified to batch process files containing many TLEs. The results, for both converging and non-converging cases, should be output in a format which could be graphically analyzed. For the non-converging cases, the orbital elements of each iteration could be plotted to see if any patterns emerge which might help to force convergence.

5.2.2 Expanding to Include SGP, SGP8, and SDP8. This research used only SGP4 and SDP4 prediction models. Future work should expand on this research to include the other three propagators, SGP, SGP8, and SDP8.

5.2.3 Obtaining Osculating Elements. The approach was based on the General Perturbation theory assumption that the osculating elements would provide a close approximation to the mean elements. This was found to not always be true. Low eccentricity and/or inclinations close to zero or 180 degrees produce unstable results when numerically computing Keplerian elements. These are points of singularity for computing the classical elements from position and velocity [Wie89-2].

One possible approach to avoid these singularities is to use an alternate form of the classical Keplerian elements, e.g. equinoctal elements, in the transformation from position and velocity. Equinoctal elements are free from the singularities of zero eccentricity or inclinations of zero or 180 degrees [Wie89-2: 22]. This may provide a more accurate transformation of position and velocity to orbital elements. However, the equinoctal elements must still be converted to the classical elements used by the prediction models.

Another candidate approach is to try to extract the osculating elements directly from the prediction models themselves rather than computing them with a separate routine. The scope of this research was to treat the models as "black box" functions. It was required to peer inside the models themselves only to gain a general understanding and also to determine the units of input variables and reassignment of variables within common blocks. The models were used unmodified. Internally, the propagators calculate partially osculating elements, adding back in periodic variations, in order to compute the state. Detailed analysis may reveal that, with modifications to the

propagators, better calculations of the osculating elements themselves could be obtained than by RV2OSC. These modifications, however, would negate the modularity that was sought after in this research project.

Lastly, research into developing another method of obtaining an initial guess and correcting the mean elements other than using the osculating elements might prove useful.

5.2.4 Newton's Method. In theory, implementing Newton's method as discussed in Chapter III would provide faster convergence. This comes from utilizing the rates of change rather than just the changes in the function for which the root is sought. However, this method is more complex and computationally intensive It necessitates seven propagator calls and solving a 6 x 6 linear system of equations for each iteration. Admittedly, a 6 x 6 system can be solved very rapidly on a computer and should not add any significant time to execution.

The application of Newton's method to this problem also eliminates the need to compute the osculating elements at each iteration step. The corrections to the mean elements can be expressed in terms of the Jacobian of the function and changes in position and velocity.

If future work were to implement Newton's method and include SGP, another problem would present itself. Since SGP uses two pseudo-drag terms rather than the one term used by the other four propagators, the input to the function is a set of seven elements allowed to change where the output, the state, is only six. This creates a non-square Jacobian matrix which needs to be inverted. A method would need to be devised to solve this situation.

5.2.5 Optimization and Extensions of the Computer Program. The computer program developed for this research was intended as a research tool only. If, in the future, a method is developed which converges even for the cases which, in this project, did not, a computer program should be developed and made available to interested users within the space community. The code should be optimized for speed and an interface provided which checks input for data which may cause runtime errors.

The FORTRAN source code for the propagators used in this research was written to be compatible with FORTRAN IV. This source code could be updated to FORTRAN 77 and further optimized. Ports could be made to C and C++ taking advantage of the advanced data structures offered by these languages. Likewise, NAIC still needs a solution implemented into SM in C++.

5.2.6 Drag Estimation. The method implemented in this project showed a way to take the position and velocity of an Earth-orbiting satellite and compute NORAD-compatible mean elements, as long as the drag term, B*, was known. An extension of this would be to compute, from just the state, the mean elements and the drag terms: $\dot{n}/2$ and $\ddot{n}/6$ in SGP, and B* in SGP4, SDP4, SGP8, and SDP8. Future research on satellite drag and methods of estimating the drag would be valuable. Since the drag term is a function of the physical geometry as well as atmospheric conditions, some aspects of the satellites' physical characteristics would be needed. However, a parametric analysis of how the drag terms affect the propagators' computations of the state might reveal information that could be utilized to approximate and converge on a solution.

Appendix A: *NORAD TWO-LINE ELEMENTS*

NORAD maintains orbital data for each Earth-orbiting satellite in a Two-Line Orbital Element Set (TLE). These TLEs are made available to users within the space community in an ASCII file format. In this file, a line is added in front of each TLE with an eleven character name. The three lines of the published NORAD data are in the following format:

```
AAAAAAAAAAA
1 NNNNNU NNNNNAAA NNNNN.NNNNNNNN +.NNNNNNNN +NNNNN-N +NNNNN-N N NNNNN
2 NNNNN NNN.NNNN NNN.NNNN NNNNNNN NNN.NNNN NNN.NNNN NN.NNNNNNNNNNNNNN
```

Line 0 is an eleven-character name.

Lines 1 and 2 are the standard Two-Line Orbital Element Set Format identical to that used by NORAD and NASA. The format description is:

Line 1

Column	Description
01-01	Line Number of Element Data
03-07	Satellite Number
10-11	International Designator (Last two digits of launch year)
12-14	International Designator (Launch number of the year)
15-17	International Designator (Piece of launch)
19-20	Epoch Year (Last two digits of year)
21-32	Epoch (Julian Day and fractional portion of the day)

34-43	One Half the First Time Derivative of the Mean Motion or Ballistic Coefficient (Depending on ephemeris type)
45-52	One Sixth the Second Time Derivative of Mean Motion (decimal point assumed; blank if N/A)
54-61	BSTAR drag term if GP4 general perturbation theory was used. Otherwise, radiation pressure coefficient. (Decimal point assumed)
63-63	Ephemeris type
65-68	Element number
69-69	Check Sum (Modulo 10) (Letters, blanks, periods, plus signs = 0; minus signs = 1)

Line 2

Column	Description
01-01	Line Number of Element Data
03-07	Satellite Number
09-16	Inclination [Degrees]
18-25	Right Ascension of the Ascending Node [Degrees]
27-33	Eccentricity (decimal point assumed)
35-42	Argument of Perigee [Degrees]
44-51	Mean Anomaly [Degrees]
53-63	Mean Motion [Revs per day]
64-68	Revolution number at epoch [Revs]
69-69	Check Sum (Modulo 10)

All other columns are blank or fixed.

Example:

```
NOAA 6
1 11416U          86 50.28438588 0.00000140           67960-4 0  5293
2 11416  98.5105  69.3305 0012788  63.2828 296.9658 14.24899292346978
```

Appendix B: *COMPUTER CODE AND SUBPROGRAMS*

```
*       DRIVER                                                          OCT 94
*
        PROGRAM DRIVER

*   WGS-72 PHYSICAL AND GEOPOTENTIAL CONSTANTS
*          CK2= .5*J2*AE**2      CK4=-.375*J4*AE**4

       DOUBLE PRECISION EPYR,EPDAY,DS,YS,EPOCH,DS50,MXDEL1,MXDEL2
       DIMENSION DELTA(6)
       COMMON/E1/XMO,XNODEO,OMEGAO,EO,XINCL,XNO,XNDT2O,XNDD6O,BSTAR,
      1            X,Y,Z,XDOT,YDOT,ZDOT,EPOCH,DS50
       COMMON/C1/CK2,CK4,E6A,QOMS2T,S,TOTHRD,
      1           XJ3,XKE,XKMPER,XMNPDA,AE
       COMMON/C2/DE2RA,PI,PIO2,TWOPI,X3PIO2

       DATA DE2RA,E6A,PI,PIO2,QO,SO,TOTHRD,TWOPI,X3PIO2,XJ2,XJ3,
      1            XJ4,XKE,XKMPER,XMNPDA,AE/.174532925E-1,1.E-6,
      2            3.14159265,1.57079633,120.0,78.0,.66666667,
      4            6.2831853,4.71238898,1.082616E-3,-.253881E-5,
      5            -1.65597E-6,.743669161E-1,6378.135,1440.,1./

       CK2=.5*XJ2*AE**2
       CK4=-.375*XJ4*AE**4
       QOMS2T=((QO-SO)*AE/XKMPER)**4
       S=AE*(1.+SO/XKMPER)

*      READ IN START TIME & STOP TIME (MIN SINCE EPOCH),
*      DELTA V (KM/S),
*      AND MEAN ELEMENTS PER NORAD T-TYPE TWO LINE ELEMENT SET

       READ(*,*) TS,TF,DELT
       READ(*,*) DXDOT,DYDOT,DZDOT
       READ(*,*) EPYR,EPDAY,XNDT2O,XNDD6O,BSTAR
       READ(*,*) XINCL,XNODEO,EO,OMEGAO,XMO,XNO
       READ(*,*) MAXITR,TOL1,TOL2

*      ECHO INPUT
*
       WRITE(*,*) 'INPUT ECHO'
       WRITE(*,*) '-----------------'
       WRITE(*,*) 'TS: ',TS
       WRITE(*,*) 'TF: ',TF
       WRITE(*,*) 'DELT: ',DELT
       WRITE(*,*) 'DXDOT: ',DXDOT
       WRITE(*,*) 'DYDOT: ',DYDOT
       WRITE(*,*) 'DZDOT: ',DZDOT
       WRITE(*,*) 'EPYR: ',EPYR
       WRITE(*,*) 'EPDAY: ',EPDAY
       WRITE(*,*) 'XNDT2O: ',XNDT2O
       WRITE(*,*) 'XNDD6O: ',XNDD6O
       WRITE(*,*) 'BSTAR: ',BSTAR
       WRITE(*,*) 'XINCL: ',XINCL
       WRITE(*,*) 'XNODEO: ',XNODEO
       WRITE(*,*) 'EO: ',EO
```

```
      WRITE(*,*) 'OMEGAO: ',OMEGAO
      WRITE(*,*) 'XMO: ',XMO
      WRITE(*,*) 'XNO: ',XNO
      WRITE(*,*) 'MAXITR: ',MAXITR
      WRITE(*,*) 'TOL1: ',TOL1
      WRITE(*,*) 'TOL2: ',TOL2
      WRITE(*,*)


*     CONVERT TO INTERNAL UNITS:
*
*     EPOCH: YYDDD.DDDDDDDD
*       YY:            LAST 2 DIGITS OF YEAR (ASSUMES 20TH CENTURY)
*       DDD.DDDDDDDD:  JULIAN DAY
*
*     TIME IN MINUTES
*     DISTANCE IN EARTH RADII

      EPOCH=(EPYR)*1000.+EPDAY
      XNODEO=XNODEO*DE2RA
      OMEGAO=OMEGAO*DE2RA
      XMO=XMO*DE2RA
      XINCL=XINCL*DE2RA
      TEMP=TWOPI/XMNPDA/XMNPDA
      XNO=XNO*TEMP*XMNPDA
      XNDT2O=XNDT2O*TEMP
      XNDD6O=XNDD6O*TEMP/XMNPDA
      BSTAR=BSTAR/AE
      TEMP=XKMPER/AE*XMNPDA/86400.
      DXDOT=DXDOT/TEMP
      DYDOT=DYDOT/TEMP
      DZDOT=DZDOT/TEMP

*     INITIALIZE IDEEP TO 0, IE PERIOD < 225 MINUTES
*     IF PERIOD >= 225 MINUTES THEN SET IDEEP TO 1

      IDEEP=0
      A1=(XKE/XNO)**TOTHRD
      TEMP=1.5*CK2*(3.*COS(XINCL)**2-1.)/(1.-EO*EO)**1.5
      DEL1=TEMP/(A1*A1)
      AO=A1*(1.-DEL1*(.5*TOTHRD+DEL1*(1.+134./81.*DEL1)))
      DELO=TEMP/(AO*AO)
      XNODP=XNO/(1.+DELO)
      IF((TWOPI/XNODP/XMNPDA) .GE. .15625) IDEEP=1

      TSINCE=TS
      IFLAG=1

*     PROPAGATE STATE TO TS, I.E., DELTA V TIME

      IF (IDEEP.EQ.0) THEN
            CALL SGP4(IFLAG,TSINCE)
      ELSE
            CALL SDP4(IFLAG,TSINCE)
      ENDIF

*     OUTPUT THE STATE BEFORE ADDING DELTA V
*     IN UNITS OF KM AND KM/S LEAVING INTERNAL
*     STATE VALUES IN INTERNAL UNITS

      WRITE(*,*) 'STATE W/O DELTA V @ T=EPOCH+',TS,' MINUTES:'
      WRITE(*,*) '-------------------------------------------'
      WRITE(*,*) 'EPOCH:  ',EPOCH
```

```
      WRITE(*,*) 'X (KM): ',X*XKMPER/AE
      WRITE(*,*) 'Y (KM): ',Y*XKMPER/AE
      WRITE(*,*) 'Z (KM): ',Z*XKMPER/AE
      WRITE(*,*) 'XDOT (KM/S): ',XDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'YDOT (KM/S): ',YDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'ZDOT (KM/S): ',ZDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*)

*     ADD DELTA V

      XDOT=XDOT+DXDOT
      YDOT=YDOT+DYDOT
      ZDOT=ZDOT+DZDOT

*     ASSIGN TRUE STATE IN INTERNAL UNITS

      TX=X
      TY=Y
      TZ=Z
      TXDOT=XDOT
      TYDOT=YDOT
      TZDOT=ZDOT

*     OBTAIN TRUE OSCULATING ELEMENTS FROM TRUE STATE
*     TRUE ELEMENTS ARE: TINCL,TNODE,TECC,TOMEGA,TM,TN
*
*     ASSIGN CONVERTED STATE TO EXTERNAL UNITS, KM & KM/S

      TEMP=XKMPER/AE
      CX=TX*TEMP
      CY=TY*TEMP
      CZ=TZ*TEMP
      TEMP=XKMPER/AE*XMNPDA/86400.
      CXDOT=TXDOT*TEMP
      CYDOT=TYDOT*TEMP
      CZDOT=TZDOT*TEMP

      CALL RV2OSC(CX,CY,CZ,CXDOT,CYDOT,CZDOT,TINCL,TNODE,TECC,TOMEGA,
     1      TM,TN)

*     UPDATE NEW EPOCH TO TIME OF MANEUVER

      DS=TS/XMNPDA
      EPDAY=EPDAY+DS
      IF(EPDAY.GE.365.) THEN
        YS=AINT(EPDAY/365.)
        EPYR=EPYR+YS
        EPDAY=MOD(EPDAY,365.)
      ENDIF
      EPOCH=(EPYR*1000.)+EPDAY
      TSINCE=0.

*     OUTPUT THE TRUE STATE AFTER ADDING DELTA V
*     IN UNITS OF KM AND KM/S LEAVING INTERNAL
*     STATE VALUES IN INTERNAL UNITS
*

      WRITE(*,*) 'TRUE STATE W/ DELTA V @ T = NEW EPOCH'
      WRITE(*,*) '-----------------------------------------'
      IF(IDEEP.EQ.0) WRITE(*,*) 'MODEL:  SGP4'
      IF(IDEEP.EQ.1) WRITE(*,*) 'MODEL:  SDP4'
      WRITE(*,*) 'EPOCH:  ',EPOCH
      WRITE(*,*) 'X (KM): ',CX
```

```
      WRITE(*,*) 'Y (KM): ',CY
      WRITE(*,*) 'Z (KM): ',CZ
      WRITE(*,*) 'XDOT (KM/S): ',CXDOT
      WRITE(*,*) 'YDOT (KM/S): ',CYDOT
      WRITE(*,*) 'ZDOT (KM/S): ',CZDOT
      WRITE(*,*) 'INCL (DEGREES): ',TINCL
      WRITE(*,*) 'NODE (DEGREES): ',TNODE
      WRITE(*,*) 'ECC : ',TECC
      WRITE(*,*) 'OMEGA (DEGREES): ',TOMEGA
      WRITE(*,*) 'M (DEGREES): ',TM
      WRITE(*,*) 'N (DEGREES): ',TN
      WRITE(*,*)


**************************
*     ITERATION SCHEME   *
**************************

      ITER=0
      ICONV=0

*     INITIALIZE FIRST GUESS @ TLE AS TRUE OSCULATING ELEMENTS

      XINCL=TINCL*DE2RA
      XNODEO=TNODE*DE2RA
      EO=TECC
      OMEGAO=TOMEGA*DE2RA
      XMO=TM*DE2RA
      TEMP=TWOPI/XMNPDA/XMNPDA
      XNO=TN*TEMP*XMNPDA

      DO 100,I=1,MAXITR

      ITER=ITER+1

*     INITIALIZE IDEEP TO 0, IE PERIOD < 225 MINUTES
*     IF PERIOD >= 225 MINUTES THEN SET IDEEP TO 1

      IDEEP=0
      A1=(XKE/XNO)**TOTHRD
      TEMP=1.5*CK2*(3.*COS(XINCL)**2-1.)/(1.-EO*EO)**1.5
      DEL1=TEMP/(A1*A1)
      AO=A1*(1.-DEL1*(.5*TOTHRD+DEL1*(1.+134./81.*DEL1)))
      DELO=TEMP/(AO*AO)
      XNODP=XNO/(1.+DELO)
      IF((TWOPI/XNODP/XMNPDA) .GE. .15625) IDEEP=1

      IFLAG=1

      IF (IDEEP.EQ.0) THEN
            CALL SGP4(IFLAG,TSINCE)
      ELSE
            CALL SDP4(IFLAG,TSINCE)
      ENDIF

*     OBTAIN OSCULATING ELEMENTS FROM STATE
*     CURRENT ITERATION OSCULATING ELEMENTS ARE PREFIXED WITH "Y"
*
*     ASSIGN CONVERTED STATE TO EXTERNAL UNITS, KM & KM/S
*     TO INPUT TO RV2OSC SUBROUTINE

      TEMP=XKMPER/AE
```

```
      CX=X*TEMP
      CY=Y*TEMP
      CZ=Z*TEMP
      TEMP=XKMPER/AE*XMNPDA/86400.
      CXDOT=XDOT*TEMP
      CYDOT=YDOT*TEMP
      CZDOT=ZDOT*TEMP

      CALL RV2OSC(CX,CY,CZ,CXDOT,CYDOT,CZDOT,YINCL,YNODE,YECC,YOMEGA,
     1      YM,YN)

*     CHECK FOR CONVERGENCE
*
*     TEST FOR CONVERGENCE BY CHECKING MAX ABSOLUTE VALUES OF THE
*     DELTA OF BOTH POSITION AND VELOCITY

      TEMP=XKMPER/AE
      DELTA(1)=(TX-X)*TEMP
      DELTA(2)=(TY-Y)*TEMP
      DELTA(3)=(TZ-Z)*TEMP
      TEMP=XKMPER/AE*XMNPDA/86400.
      DELTA(4)=(TXDOT-XDOT)*TEMP
      DELTA(5)=(TYDOT-YDOT)*TEMP
      DELTA(6)=(TZDOT-ZDOT)*TEMP

      MXDEL1=ABS(DELTA(1))
      MXDEL2=ABS(DELTA(4))
      DO 90,J=2,3
            IF(ABS(DELTA(J)).GT.MXDEL1) MXDEL1=ABS(DELTA(J))
            IF(ABS(DELTA(J+3)).GT.MXDEL1) MXDEL2=ABS(DELTA(J+3))
   90 CONTINUE

      IF ((MXDEL1.LE.TOL1).AND.(MXDEL2.LE.TOL2)) ICONV=1

      IPRNT=0
      ISTEP=10
      DO 95, K=1,ISTEP
        IF (ITER.EQ.K*MAXITR/ISTEP) IPRNT=1
   95 CONTINUE

      IF (IPRNT.EQ.1) THEN
*     OUTPUT ITERATION VALUES
*
*
      WRITE(*,*) 'ITERATION: ',ITER
      WRITE(*,*) '-----------------------------------------'
      IF(IDEEP.EQ.0) WRITE(*,*) 'MODEL:  SGP4'
      IF(IDEEP.EQ.1) WRITE(*,*) 'MODEL:  SDP4'
      WRITE(*,*) 'BSTAR: ',BSTAR*AE
      WRITE(*,*) 'XINCL: ',XINCL/DE2RA, ' DEGREES'
      WRITE(*,*) 'XNODEO: ',XNODEO/DE2RA, ' DEGREES'
      WRITE(*,*) 'EO: ',EO
      WRITE(*,*) 'OMEGAO: ',OMEGAO/DE2RA, ' DEGREES'
      WRITE(*,*) 'XMO: ',XMO/DE2RA, ' DEGREES'
      TEMP=TWOPI/XMNPDA/XMNPDA
      WRITE(*,*) 'XNO: ',XNO/TEMP/XMNPDA, ' REVS/DAY'
      TEMP=XKMPER/AE
      WRITE(*,*) 'X (KM): ',X*TEMP
      WRITE(*,*) 'Y (KM): ',Y*TEMP
      WRITE(*,*) 'Z (KM): ',Z*TEMP
      TEMP=XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'XDOT (KM/S): ',XDOT*TEMP
      WRITE(*,*) 'YDOT (KM/S): ',YDOT*TEMP
```

```
      WRITE(*,*) 'ZDOT (KM/S): ',ZDOT*TEMP
      WRITE(*,*) 'DELTA X: ',DELTA(1)
      WRITE(*,*) 'DELTA Y: ',DELTA(2)
      WRITE(*,*) 'DELTA Z: ',DELTA(3)
      WRITE(*,*) 'MXDEL1: ',MXDEL1
      WRITE(*,*) 'DELTA XDOT: ',DELTA(4)
      WRITE(*,*) 'DELTA YDOT: ',DELTA(5)
      WRITE(*,*) 'DELTA ZDOT: ',DELTA(6)
      WRITE(*,*) 'MXDEL2: ',MXDEL2
      WRITE(*,*)

      ENDIF

*     IF CONVERGED THEN END ITERATION

      IF(ICONV.EQ.1) GOTO 110

*     IF NOT CONVERGED THEN MAKE THE NEXT GUESS AT ELEMENTS

      XINCL=XINCL+(TINCL-YINCL)*DE2RA
      XNODEO=FMOD2P(XNODEO+(TNODE-YNODE)*DE2RA)

      EO=ABS(EO+(TECC-YECC))

      OMEGAO=FMOD2P(OMEGAO+(TOMEGA-YOMEGA)*DE2RA)
      XMO=FMOD2P(XMO+(TM-YM)*DE2RA)
      TEMP=TWOPI/XMNPDA/XMNPDA
      XNO=XNO+(TN-YN)*TEMP*XMNPDA

  100 CONTINUE
  110 CONTINUE
      IF(ICONV.EQ.1) THEN
            WRITE(*,*) 'CONVERGED AFTER ',ITER,' ITERATIONS'
            WRITE(*,*)
      ELSE
            WRITE(*,*) 'DID NOT CONVERGE AFTER',ITER,' ITERATIONS'
            WRITE(*,*)
      ENDIF


*     OUTPUT THE STATE AFTER ADDING DELTA V
*     IN UNITS OF KM AND KM/S LEAVING INTERNAL
*     STATE VALUES IN INTERNAL UNITS

      WRITE(*,*) 'STATE W/ DELTA V @ NEW EPOCH'
      WRITE(*,*) '--------------------------------------------'
      IF(IDEEP.EQ.0) WRITE(*,*) 'MODEL:  SGP4'
      IF(IDEEP.EQ.1) WRITE(*,*) 'MODEL:  SDP4'
      WRITE(*,*) 'EPOCH:  ',EPOCH
      WRITE(*,*) 'X (KM): ',X*XKMPER/AE
      WRITE(*,*) 'Y (KM): ',Y*XKMPER/AE
      WRITE(*,*) 'Z (KM): ',Z*XKMPER/AE
      WRITE(*,*) 'XDOT (KM/S): ',XDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'YDOT (KM/S): ',YDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'ZDOT (KM/S): ',ZDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*)

*     OUTPUT THE NEW MEAN ELEMENTS

      WRITE(*,*) 'NEW MEAN ELEMENTS'
      WRITE(*,*) '-----------------'
      IF(IDEEP.EQ.0) WRITE(*,*) 'MODEL:  SGP4'
      IF(IDEEP.EQ.1) WRITE(*,*) 'MODEL:  SDP4'
```

```
      WRITE(*,*) 'BSTAR: ',BSTAR*AE
      WRITE(*,*) 'XINCL: ',XINCL/DE2RA, ' DEGREES'
      WRITE(*,*) 'XNODEO: ',XNODEO/DE2RA, ' DEGREES'
      WRITE(*,*) 'EO: ',EO
      WRITE(*,*) 'OMEGAO: ',OMEGAO/DE2RA, ' DEGREES'
      WRITE(*,*) 'XMO: ',XMO/DE2RA, ' DEGREES'
      TEMP=TWOPI/XMNPDA/XMNPDA
      WRITE(*,*) 'XNO: ',XNO/TEMP/XMNPDA, ' REVS/DAY'
      WRITE(*,*)

*     IF THE ITERATION CONVERGED THEN
*     PROPAGATE THE STATE FROM NEW EPOCH (MANEUVER TIME)
*     TO NEW EPOCH + TF (MIN)
*     IN STEPS OF DELT (MIN)

      IF ((ICONV.NE.1).OR.(TS.EQ.TF)) STOP

      TSINCE=0.

      IFLAG=1

  200 IF (IDEEP.EQ.0) THEN
            CALL SGP4(IFLAG,TSINCE)
      ELSE
            CALL SDP4(IFLAG,TSINCE)
      ENDIF

*     OUTPUT THE STATE
*     IN UNITS OF KM AND KM/S LEAVING INTERNAL
*     STATE VALUES IN INTERNAL UNITS

      WRITE(*,*) 'STATE @ ',TSINCE,' MINUTES MINUTES FROM EPOCH'
      WRITE(*,*) '-------------------------------------------'
      WRITE(*,*) 'EPOCH:  ',EPOCH
      WRITE(*,*) 'X (KM): ',X*XKMPER/AE
      WRITE(*,*) 'Y (KM): ',Y*XKMPER/AE
      WRITE(*,*) 'Z (KM): ',Z*XKMPER/AE
      WRITE(*,*) 'XDOT (KM/S): ',XDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'YDOT (KM/S): ',YDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*) 'ZDOT (KM/S): ',ZDOT*XKMPER/AE*XMNPDA/86400.
      WRITE(*,*)

      TSINCE=TSINCE+DELT

      IF((TSINCE .LE. TF).AND.(DELT.GT.0.)) GO TO 200

      STOP
      END

      include 'sgp4.f'
      include 'sdp4.f'
      include 'deep.f'
      include 'rv2osc.f'
      include 'thetag.f'
      include 'fmod2p.f'
      include 'actan.f'
```

```
      SUBROUTINE  RV2OSC(RI,RJ,RK,VI,VJ,VK,XINCL,XNODE,ECC,OMEGA,XM,XN)
*
*     THIS SUBROUTINE TAKES ECI STATE VECTOR ELEMENTS X,Y,Z,VX,VY,VZ
*     IN UNITS OF KM AND KM/S AND RETURNS OCSULATING ORBITAL ELEMENTS:
*
*     XINCL:      INCLINATION IN DEGREES
*     XNODE:      RIGHT ASCENSION OF ASCENDING NODE IN DEGREES
*     ECC:        ECCENTRICITY
*     OMEGA:      AGRUMENT OF PERIGEE IN DEGREES
*     XM:         MEAN ANOMOLY IN DEGREES
*     XN:         MEAN MOTION IN REVS/DAY
*

      DATA XMU/398601.2/

      PI=4.*ATAN(1.)

*     CALCULATE THE MAGNITUDE OF POSITION (R) AND VELOCITY (V)
*     AND THE MAGNITUDE OF THE POSITION VECTOR DOTTED WITH THE
*     VELOCITY VECTOR (RDOTV)

      R=VMAG(RI,RJ,RK)
      V=VMAG(VI,VJ,VK)
      RDOTV=DOT(RI,RJ,RK,VI,VJ,VK)


*     CALCULATE THE ANGULAR MOMENTUM VECTOR
*     AND ITS MAGNITUDE

      CALL CROSS(RI,RJ,RK,VI,VJ,VK,HI,HJ,HK)
      H=VMAG(HI,HJ,HK)


*     CALCULATE THE NODAL VECTOR AND ITS MAGNITUDE

      XNVECI=-HJ
      XNVECJ=HI
      XNVECK=0.
      XNMAG=VMAG(XNVECI,XNVECJ,XNVECK)


*     CALCULATE THE ECCENTRICITY VECTOR AND ITS MAGNITUDE (ECC)

      ECCI=1./XMU*((V*V-XMU/R)*RI-RDOTV*VI)
      ECCJ=1./XMU*((V*V-XMU/R)*RJ-RDOTV*VJ)
      ECCK=1./XMU*((V*V-XMU/R)*RK-RDOTV*VK)
      ECC=VMAG(ECCI,ECCJ,ECCK)


*     CALCULATE THE INCLINATION (XINCL)

      COSX=HK/H
      XINCL=DEGREE(ACOS(COSX))


*     CALCULATE THE RIGHT ASCENSION OF ASCENDING NODE (XNODE)

      COSX=XNVECI/XNMAG
      XNODE=DEGREE(ACOS(COSX))
      IF(XNVECJ.LE.0.) XNODE=360.-XNODE
```

```
*       CALCULATE THE ARGUMENT OF PERIGEE (OMEGA)

        COSX=DOT(XNVECI,XNVECJ,XNVECK,ECCI,ECCJ,ECCK)/(XNMAG*ECC)
        OMEGA=DEGREE(ACOS(COSX))
        IF(ECCK.LE.0.) OMEGA =360.-OMEGA


*       CALCULATE THE MEAN ANOMOLY (XM)

        P=H*H/XMU
        A=P/(1-ECC*ECC)
        ECOSE=1-R/A
        ESINE=RDOTV/SQRT(XMU*A)
        E=ACTAN(ESINE,ECOSE)
        XM=DEGREE(E-ESINE)
        IF(XM.LT.0.) XM=360+XM


*       CALCULATE THE MEAN MOTION (XN)

        PERIOD=2.*PI*SQRT(A*A*A/XMU)
        XN=86400./PERIOD

        RETURN
        END
```

```
*
*
*
*       SUBPROGRAMS USED BY RV2OSC ARE CONTAINED BELOW
*
*       IN ADDITION TO THESE, RV2OSC USES THE ACTAN FUNCTION
*
*
      FUNCTION DOT(X1,Y1,Z1,X2,Y2,Z2)
      DOT=X1*X2+Y1*Y2+Z1*Z2
      RETURN
      END


      FUNCTION VMAG(X,Y,Z)
      VMAG=SQRT(X*X+Y*Y+Z*Z)
      RETURN
      END


      FUNCTION DEGREE(X)
      PI=4.*ATAN(1.)
      DEGREE=X*180./PI
      RETURN
      END


      SUBROUTINE CROSS(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3)
      X3=Y1*Z2-Z1*Y2
      Y3=-(X1*Z2-Z1*X2)
      Z3=X1*Y2-Y1*X2
      RETURN
      END
```

Appendix C: *INPUT AND OUTPUT DATA*

C.1 Input File Description.

The input file was a five-line data set with format as given in Table C.1.

Table C.1 Input Data File Format

Line	Variable Name					
1	TS					
2	DXDOT	DYDOT	DZDOT			
3	EPYR	EPDAY	XNDT2O	XNDD6O	BSTAR	
4	XINCL	XNODE	EO	OMEGAO	XMO	XNO
5	MAXITR	TOL1	TOL2			

The variable names are defined and the associated units are given in Table C.2.

Table C.2 Input Values

Variable Name	Definition	Units
TS	Maneuver time	minutes past epoch
DXDOT	Velocity change in X direction	kilometers/second
DYDOT	Velocity change in Y direction	kilometers/second
DZDOT	Velocity change in Z direction	kilometers/second
EPYR	Last two digits of epoch year	
EPDAY	Epoch Day	
XNDT2O	One half of the first time derivative of mean motion	revolutions/day/day
XNDD6O	One sixth of the second time derivative of mean motion	revolutions/day/day/day
BSTAR	Drag coefficient	1/earth radii
XINCL	Mean orbital inclination	degrees
XNODE	Mean right ascension of the ascending node	degrees
EO	Mean eccentricity	
OMEGAO	Mean argument of perigee	degrees
XMO	Mean mean anamoly	degrees
XNO	Mean mean motion	revolutions/day
MAXITR	Maximum number of iterations	
TOL1	Positional convergence criterion	kilometers
TOL2	Velocity convergence criterion	kilometers/second

C.2 Sample Input and Output.

C.2.1 Converging (Run 1).

Input:

```
0.  0.  0.
0.  0.  0.
80. 275.98708465    .00073094   .13844E-3   .66816E-4
72.8435 115.9689    .0086731    52.6988 110.5714    16.05824518
1000  1.D-5   1.D-5
```

Output

```
INPUT ECHO
----------------
TS:   0.
DXDOT:   0.
DYDOT:   0.
DZDOT:   0.
EPYR:      80.000000000000000000000000000000
EPDAY:      275.98708464999999999999999999999999
BSTAR:      6.6816000000000D-05
XINCL:      72.843500000000
XNODEO:      115.96890000000
EO:     8.6731000000000D-03
OMEGAO:      52.698800000000
XMO:     110.57140000000
XNO:     16.058245180000
MAXITR:   1000
TOL1:     1.0000000000000D-05
TOL2:     1.0000000000000D-05

STATE W/O DELTA V @  T = EPOCH +  0. MINUTES:
---------------------------------------------
EPOCH :      80275.98708465000000000000000000000
TSINCE (MIN):   0.
X (KM):     2328.9706719836
Y (KM):    -5995.2208387137
Z (KM):     1719.9707009129
XDOT (KM/S):     2.9120722772560
YDOT (KM/S):   -0.98341535999325
ZDOT (KM/S):    -7.0908169483094

STATE W/ DELTA V @ T = NEW EPOCH
------------------------------------
MODEL:  SGP4
X (KM):     2328.9706719836
Y (KM):    -5995.2208387137
Z (KM):     1719.9707009129
XDOT (KM/S):     2.9120722772560
```

```
YDOT (KM/S):    -0.98341535999325
ZDOT (KM/S):     -7.0908169483094
OSCULATING ELEMENTS:
EPOCH:      80275.987084650000000000000000000000
TSINCE (MIN):    0.
INCL (DEGREES):      72.853850793758
NODE (DEGREES):      115.96229565319
ECC:      9.6688686502438D-03
OMEGA (DEGREES):      59.407389985514
M (DEGREES):     103.834335347262
N (REV/DAY):      16.039008453174

CONVERGED AFTER    8 ITERATIONS

STATE @ T =      80275.987084650000000000000000000000
-------------------------------------------
MODEL:  SGP4
X (KM):      2328.9706707997
Y (KM):     -5995.2208359032
Z (KM):      1719.9707003254
XDOT (KM/S):      2.9120722786609
YDOT (KM/S):    -0.98341536059274
ZDOT (KM/S):     -7.0908169515364

COMPUTED MEAN ELEMENTS
----------------------
MODEL:  SGP4
EPOCH:      80275.987084650000000000000000000000
BSTAR:      6.6816000000000D-05
XINCL:      72.843499999930 DEGREES
XNODEO:      115.96889999989 DEGREES
EO:      8.6730998226599D-03
OMEGAO:      52.698802777752 DEGREES
XMO:      110.57139722179 DEGREES
XNO:      16.058245180245 REVS/DAY
```

C.2.2 Non-converging (Run 6).

Input:

```
0.  0.  0.
0.  0.  0.
94  027.71283080     .00010322    .00000E-0    .13245E-3
51.6150 171.3210     .0004383     242.7692 117.2855     15.59769565
10000  1.E-5   1E-5
```

Output:

```
INPUT ECHO
-----------------
TS:   0.
```

```
DXDOT:    0.
DYDOT:    0.
DZDOT:    0.
EPYR:      94.000000000000000000000000000000000
EPDAY:      27.712830800000000000000000000000000
BSTAR:      1.3245000000000D-04
XINCL:      51.615000000000
XNODEO:      171.32100000000
EO:      4.3830000000000D-04
OMEGAO:      242.76920000000
XMO:      117.28550000000
XNO:      15.597695650000
MAXITR:    10000
TOL1:      1.0000000000000D-05
TOL2:      1.0000000000000D-05

STATE W/O DELTA V @  T = EPOCH +  0. MINUTES:
-------------------------------------------------
EPOCH :      94027.712830799999999999999999999998
TSINCE (MIN):    0.
X (KM):  . -6691.0583844454
Y (KM):      1021.3614767624
Z (KM):      5.8265559592820D-03
XDOT (KM/S):    -0.71533823038558
YDOT (KM/S):     -4.7105374070153
ZDOT (KM/S):      6.0189300496620

STATE W/ DELTA V @ T = NEW EPOCH
-------------------------------------
MODEL:  SGP4
X (KM):     -6691.0583844454
Y (KM):      1021.3614767624
Z (KM):      5.8265559592820D-03
XDOT (KM/S):    -0.71533823038558
YDOT (KM/S):     -4.7105374070153
ZDOT (KM/S):      6.0189300496620
OSCULATING ELEMENTS:
EPOCH:      94027.712830799999999999999999999998
TSINCE (MIN):    0.
INCL (DEGREES):      51.635116093688
NODE (DEGREES):      171.32099986061
ECC:      8.0636571495733D-04
OMEGA (DEGREES):      36.246775600803
M (DEGREES):      323.80789458967
N (REV/DAY):      15.575233183472

ITERATION:    1000
-----------------------------------------
MODEL:  SGP4
BSTAR:      1.3245000000000D-04
XINCL:      51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529344D-04
OMEGAO:      359.70499516529 DEGREES
XMO:     0.34969896155076 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):     -6687.0312717373
Y (KM):      1024.0383009368
Z (KM):     -4.1046931858184
XDOT (KM/S):    -0.71865926437066
YDOT (KM/S):     -4.7125757148652
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081396
```

```
DELTA Y:     -2.6768241743931
DELTA Z:      4.1105197417777
MXDEL1:      4.11051974177768553175837951130233 71
DELTA XDOT:      3.3210339850807D-03
DELTA YDOT:      2.0383078498944D-03
DELTA ZDOT:     -3.1785367071631D-03
MXDEL2:      3.321033985080671692424747476479751 6Q-003

ITERATION:    2000
-----------------------------------------
MODEL:   SGP4
BSTAR:       1.3245000000000D-04
XINCL:       51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529289D-04
OMEGAO:      359.70499516532 DEGREES
XMO:     0.34969896151364 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):      -6687.0312717373
Y (KM):       1024.0383009369
Z (KM):      -4.1046931860445
XDOT (KM/S):    -0.71865926437098
YDOT (KM/S):     -4.7125757148652
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081594
DELTA Y:     -2.6768241745711
DELTA Z:      4.1105197420038
MXDEL1:      4.11051974200382375101980869658291 34
DELTA XDOT:      3.3210339854026D-03
DELTA YDOT:      2.0383078498435D-03
DELTA ZDOT:     -3.1785367071594D-03
MXDEL2:      3.321033985402550500753982021251431 4Q-003

ITERATION:    3000
-----------------------------------------
MODEL:   SGP4
BSTAR:       1.3245000000000D-04
XINCL:       51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529288D-04
OMEGAO:      359.70499516529 DEGREES
XMO:     0.34969896153871 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):      -6687.0312717372
Y (KM):       1024.0383009374
Z (KM):      -4.1046931865628
XDOT (KM/S):    -0.71865926437172
YDOT (KM/S):     -4.7125757148651
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127082288
DELTA Y:     -2.6768241749757
DELTA Z:      4.1105197425220
MXDEL1:      4.11051974252204210813488316489383 58
DELTA XDOT:      3.3210339861461D-03
DELTA YDOT:      2.0383078497314D-03
DELTA ZDOT:     -3.1785367071609D-03
MXDEL2:      3.321033986146071630835008292592647 2Q-003

ITERATION:    4000
-----------------------------------------
MODEL:   SGP4
BSTAR:       1.3245000000000D-04
XINCL:       51.614999975422 DEGREES
```

```
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529298D-04
OMEGAO:      359.70499516530 DEGREES
XMO:     0.34969896151944 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):     -6687.0312717371
Y (KM):      1024.0383009378
Z (KM):     -4.1046931870904
XDOT (KM/S):    -0.71865926437248
YDOT (KM/S):     -4.7125757148649
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127082911
DELTA Y:     -2.6768241753886
DELTA Z:      4.1105197430497
MXDEL1:      4.11051974304968847917507446254603 56
DELTA XDOT:      3.3210339869009D-03
DELTA YDOT:      2.0383078496163D-03
DELTA ZDOT:     -3.1785367071624D-03
MXDEL2:      3.321033986900933515640232940313580 9Q-003

ITERATION:    5000
-----------------------------------------
MODEL:   SGP4
BSTAR:      1.3245000000000D-04
XINCL:      51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529314D-04
OMEGAO:      359.70499516537 DEGREES
XMO:     0.34969896147373 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):     -6687.0312717374
Y (KM):      1024.0383009362
Z (KM):     -4.1046931850976
XDOT (KM/S):    -0.71865926436962
YDOT (KM/S):     -4.7125757148654
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127080476
DELTA Y:     -2.6768241738270
DELTA Z:      4.1105197410569
MXDEL1:      4.11051974105688788796442167949862 78
DELTA XDOT:      3.3210339840426D-03
DELTA YDOT:      2.0383078500515D-03
DELTA ZDOT:     -3.1785367071631D-03
MXDEL2:      3.321033984042563724781160772181465 3Q-003

ITERATION:    6000
-----------------------------------------
MODEL:   SGP4
BSTAR:      1.3245000000000D-04
XINCL:      51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529336D-04
OMEGAO:      359.70499516532 DEGREES
XMO:     0.34969896151489 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):     -6687.0312717372
Y (KM):      1024.0383009369
Z (KM):     -4.1046931860163
XDOT (KM/S):    -0.71865926437094
YDOT (KM/S):     -4.7125757148652
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081651
DELTA Y:     -2.6768241745477
```

```
DELTA Z:      4.1105197419756
MXDEL1:      4.11051974197555125556391431018710l4
DELTA XDOT:      3.3210339853629D-03
DELTA YDOT:      2.0383078498538D-03
DELTA ZDOT:     -3.1785367071653D-03
MXDEL2:      3.321033985362903395710532095108646 9Q-003

ITERATION:    7000
-----------------------------------------
MODEL:   SGP4
BSTAR:       1.3245000000000D-04
XINCL:       51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529316D-04
OMEGAO:      359.70499516529 DEGREES
XMO:     0.34969896154274 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):      -6687.0312717372
Y (KM):      1024.0383009369
Z (KM):     -4.1046931860163
XDOT (KM/S):    -0.71865926437094
YDOT (KM/S):     -4.7125757148652
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081651
DELTA Y:     -2.6768241745477
DELTA Z:      4.1105197419756
MXDEL1:      4.11051974197555125556391431018710l4
DELTA XDOT:      3.3210339853639D-03
DELTA YDOT:      2.0383078498531D-03
DELTA ZDOT:     -3.1785367071646D-03
MXDEL2:      3.321033985363917775263109533057104 3Q-003

ITERATION:    8000
-----------------------------------------
MODEL:   SGP4
BSTAR:       1.3245000000000D-04
XINCL:       51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529279D-04
OMEGAO:      359.70499516529 DEGREES
XMO:     0.34969896154848 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):      -6687.0312717373
Y (KM):      1024.0383009368
Z (KM):     -4.1046931858938
XDOT (KM/S):    -0.71865926437076
YDOT (KM/S):     -4.7125757148652
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081410
DELTA Y:     -2.6768241744532
DELTA Z:      4.1105197418531
MXDEL1:      4.11051974185306789877358824014663 70
DELTA XDOT:      3.3210339851879D-03
DELTA YDOT:      2.0383078498738D-03
DELTA ZDOT:     -3.1785367071609D-03
MXDEL2:      3.321033985187903190411384812819051 1Q-003

ITERATION:    9000
-----------------------------------------
MODEL:   SGP4
BSTAR:       1.3245000000000D-04
XINCL:       51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
```

```
EO:      3.2734297529317D-04
OMEGAO:     359.70499516535 DEGREES
XMO:     0.34969896148073 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):     -6687.0312717372
Y (KM):      1024.0383009371
Z (KM):     -4.1046931862848
XDOT (KM/S):    -0.71865926437132
YDOT (KM/S):     -4.7125757148651
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081934
DELTA Y:     -2.6768241747582
DELTA Z:      4.1105197422441
MXDEL1:      4.11051974224408755986814867355860 77
DELTA XDOT:      3.3210339857455D-03
DELTA YDOT:      2.0383078497926D-03
DELTA ZDOT:     -3.1785367071631D-03
MXDEL2:      3.321033985745543929551937267774519 6Q-003

ITERATION:   10000
-----------------------------------------
MODEL:  SGP4
BSTAR:      1.3245000000000D-04
XINCL:      51.614999975422 DEGREES
XNODEO:      171.32103979104 DEGREES
EO:      3.2734297529345D-04
OMEGAO:      359.70499516539 DEGREES
XMO:     0.34969896145111 DEGREES
XNO:      15.597710222807 REVS/DAY
X (KM):     -6687.0312717373
Y (KM):      1024.0383009369
Z (KM):     -4.1046931859644
XDOT (KM/S):    -0.71865926437086
YDOT (KM/S):     -4.7125757148652
ZDOT (KM/S):      6.0221085863692
DELTA X:     -4.0271127081566
DELTA Y:     -2.6768241745073
DELTA Z:      4.1105197419237
MXDEL1:      4.11051974192373048566651050350628 79
DELTA XDOT:      3.3210339852855D-03
DELTA YDOT:      2.0383078498642D-03
DELTA ZDOT:     -3.1785367071631D-03
MXDEL2:      3.321033985285545570703691353742215 1Q-003

DID NOT CONVERGE AFTER  10000 ITERATIONS
```

Bibliography

[Bat71] Bate, R., Mueller, D., White, J., *Fundamentals of Astrodynamics*, New York, Dover Publications, 1971

[Bro61] Brouwer, D., Clemence, G. M., *Methods of Celestial Mechanics*, New York, Academic Press, 1961

[Bur85] Burden, R.L., Faires, J.D., *Numerical Analysis,* third edition, Boston, PWS-KENT Publishing Co., 1985.

[Che67] Chebotarev, G. A., *Analytical and Numerical Methods of Celestial Mechanics*, New York, American Elsevier Publishing Company, 1970

[Dan62] Danby, J. M. A., *Fundamentals of Celestial Mechanics*, New York, The Macmillan Company, 1962

[Ern94] Ernandes, Kenneth J., *Vector to Two-Line Elements (VEC2TLE) Software User Manual Version 9425*, 16 Freshman Lane, Stony Brook, NY, Kenneth J. Ernandes, May 1994

[Fit70] Fitzpatrick, P. M., *Principles of Celestial Mechanics*, New York, Academic Press, 1970

[Hoo80] Hoots, F., Roehrich, R. L. "Models for Propogation of NORAD Element Sets" *Space Track Report No. 3,* Peterson AFB, Colorado, 1980

[Kun93] Kunz, A.A., *Virtual Environment for Satellite Modeling and Orbital Analysis in a Distributed Interactive Simulation*, MS Thesis, AFIT/GCS/ENG/93D-14, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1993

[NOR82] North American Aerospace Defense Command, *TP SCC 008 Spadoc Computation Center Computer Program Product Specification Mathematical Foundation for SSC Astrodynamic Theory* , Technical Publication 008, Colorado Springs, CO, HQ NORAD, 6 April 1982.

[Pon92] Pond, D., *A Synthetic Environment for Satellite Modeling and Satellite Orbital Motion,* MS Thesis, AFIT/GCS/ENG/92D-12, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1992

[Roy82] Roy, A. E., *Orbital Motion*, Bristol, Adam Hilger Ltd, 1982

[Wie89] Wiesel, William E., *Spaceflight Dynamics*, New York, NY, McGraw-Hill Publishing Company, 1989

[Wie89-2] Wiesel, William E., *Advanced Astrodynamics*, Class notes for MECH 636, Advanced Astrodynamics, School of Engineering, Air Force Institute of Technology (AETC), Wright -Patterson Air Force Base, OH, 1989

[Wie93] Wiesel, William E., *Modern Methods of Orbit Determination*, Class notes for MECH 731, Modern Methods of Orbit Determination, School of Engineering, Air Force Institute of Technology (AETC), Wright -Patterson Air Force Base, OH, 1993

Vita

Captain Dwight E. Andersen was born on 15 April 1960 in Atlantic, Iowa. He graduated from Atlantic High School in 1978. In 1980 Captain Andersen attended Emery School of Aviation in Greeley, Colorado and earned his private, commercial, and instrument airplane pilot ratings. Afterwards he became Director of Operations for a non-profit organization based in Tulsa, Oklahoma. In 1984 he enlisted in the Air Force as an Airman Basic determined to earn his commission through the Airmen Education and Commission Program (AECP). After initial training, he was assigned to the Air Force Global Weather Center at Offutt AFB, Nebraska as a computer operator for their flagship Cray supercomputer. He was selected for AECP and promoted directly to Staff Sergeant. In 1989 he graduated with a Bachelor of Science degree in Aerospace Engineering from University of Missouri-Rolla. He received his commission in April of 1990 following Officers Training School. After attending Undergraduate Space Training, he was assigned to United States Space Command's Missile Warning Center (MWC) at Cheyenne Mountain AFB, Colorado, as a Deputy Crew Commander. There he became the first Second Lieutenant ever to be qualified as a MWC Evaluator. Within a year he was selected for a staff position where he was responsible for the worldwide missile warning upgrade project. In 1992 he was selected as a Missile Warning Instructor for Air Force Space Command's 21st Crew Training Squadron, where he maintained his qualification as a MWC Deputy Crew Commander. As a First Lieutenant he was selected to pursue a Masters Degree in Space Operations at the Air Force Institute of Technology and entered the School of Engineering in May of 1993. Captain Andersen is married to Marcia, his high school sweetheart and wife of 14 years. They have two daughters: Tiffany (age 11) and Samantha (age 7).

Permanent address: 4453 Cross Bow Drive
Beavercreek, OH 45432