COMPARISON AND DESIGN OF SIMPLIFIED GENERAL PERTURBATION MODELS (SGP4) AND CODE FOR NASA JOHNSON SPACE CENTER, ORBITAL DEBRIS PROGRAM OFFICE

A Graduate Project

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Nicholas Zwiep Miura

May 2009

COMMITTEE MEMBERSHIP

TITLE:	Comparison and Design of Simplified General Perturbation Models (SGP4) and Code for NASA Johnson Space Center, Orbital Debris Program Office.
AUTHOR:	Nicholas Zwiep Miura
DATE SUBMITTED:	May 29, 2009
COMMITTEE CHAIR:	Kira Abercromby, Ph.D
COMMITTEE MEMBER:	Eric Mehiel, Ph.D
COMMITTEE MEMBER:	Jordi Puig-Suari, Ph.D
COMMITTEE MEMBER:	Edwin Barker, Ph.D

ABSTRACT

Comparison and Design of Simplified General Perturbation Models (SGP4) and Code for NASA Johnson Space Center, Orbital Debris Program Office

Nicholas Zwiep Miura

This graduate project compares legacy simplified general perturbation model (SGP4) code developed by NASA Johnson Space Center, Orbital Debris Program Office, to a recent public release of SGP4 code by David Vallado. The legacy code is a subroutine in a larger program named PREDICT, which is used to predict the location of orbital debris in GEO. Direct comparison of the codes showed that the new code yields better results for GEO objects, which are more accurate by orders of magnitude (error in meters rather than kilometers). The public release of SGP4 also provides effective results for LEO and MEO objects on a short time scale. The public release code was debugged and modified to provide instant functionality to the Orbital Debris Program Office. Code is provided in an appendix to this paper along with an accompanying CD. A User's Guide is presented in Chapter 7.

ACKNOWLEDGMENTS

I would like to thank my wife, Jesselle, for all her support throughout the year. Also, a special thanks to Dr. Abercromby, my thesis advisor, who provided excellent guidance and resources during the accelerated process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

NASA	National Aeronautics and Space Administration
SGP(4)	Simplified General Perturbation
PREDICT	NASA code that predicts position of orbital debris
SPACETRACK	Government program initiated to track satellites
AIAA	American Institute of Aeronautics and Astronautics
JSC	Johnson Space Center
SATRAK	Satellite Tracking (Government owned propagator)
GSFC	Goddard Space Flight Center
JPL	Jet Propulsion Laboratory
FORTRAN	Formula Translating (IBM developed language)
COES	Classical Orbital Element Set
TLE	Two Line Element
WGS	World Geodetic System
LES	Lincoln Experimental Satellite
GEO	Geosynchronous Earth Orbit
LEO	Low Earth Orbit
MEO	Medium Earth Orbit
ISS	International Space Station
min	minutes
m	meters
cm	centimeters
mm	millimeters
sec	seconds
km	kilometers
Polysat	Cube satellites developed by Cal Poly
CP3	Polysat 3
CP4	Polysat 4
FOV	Field of view

1. Introduction

This project has been completed for NASA Johnson Space Center, Orbital Debris Program Office, with the purpose of comparing the effectiveness of legacy SGP4 propagation code and a recent public release of SGP4 propagation code. Furthermore, new SGP4 propagation code based on the public release was developed and delivered.

The Orbital Debris Program Offices uses a program called PREDICT to predict the motion of deep space debris. The program was developed in-house and uses a simplified version of Simplified General Perturbation Theory to propagate the position and velocity of a spacecraft forward in time based on classic orbital elements. In 2004, there was a public release of Simplified General Perturbation 4 (SGP4) code, based on the same theory. The Orbital Debris Program Office wanted to know how their legacy code compared with the public release.

This project compares the two codes to truth data provided by the Orbital Debris Program Office, and shows the accuracy of the delivered results. Additionally, the public release code was debugged and provided for use and the public release code was modified to run within the PREDICT program.

This paper describes the problems involved in this undertaking, as well as the procedure taken to solve these problems. Next, a detailed display of the results plus analysis and conclusions is given. Finally, a User's Guide is provided to help navigate using the public release code and to help update the legacy code.

2. Space Propagation Models

Space propagation models use current state information of a satellite to predict a future state of the satellite. As a simplistic example, imagine a car driving down a highway. If we know the location and the speed of the car now, we can make an accurate prediction of where the car will be in an hour. Similarly for satellites, if we know the position and the velocity now, we can make a reasonable guess where the satellite will be in the future. The satellite however, encounters disturbances, or perturbations, along its path that complicates its motion. These perturbations are caused by the Earth's shape (spherical harmonics), drag, radiation, and effects from other bodies (the sun and moon generally). In our car example, imagine driving up and down hills, over different terrain, with changing speed limits. It makes our simplified example much harder.

Space propagation models are used primarily by agencies that track orbiting objects. The ultimate goal is to know where everything in space is at all times. Propagation models are needed because there are simply too few telescopes to watch everything in the sky at all times. Thus, you find the position and velocity of an object once, and then use the propagator to tell you where it is in the future if you ever need to locate it again. The first attempt to use these models came in 1959, by the National Space Surveillance Control Center. This was a highly simplified model that failed when trying to propagate position more than a week. In 1960, the initial model was replaced by the 'Simplified General Perturbations (SGP)' model[ii]. Though this model provided accurate results, by 1969 there were too many satellite in orbit to maintain a catalog based on SGP. SGP4, published in SPACETRACK Report #3, was created based on a simplified version of

SGP. By 1979, it was the sole model for catalog maintenance. SGP4 failed when objects were in 'deep space' orbits, so in 1977, an extension to the program was developed to track object with an altitude over 255 km – the height where solar/lunar perturbations have a larger effect than atmospheric drag.

After SPACETRAK Report #3 was released, the 'official' code maintained by the U.S. government became an export controlled black-box. Changes to the code to reflect new programming techniques, better computing power, and more accurate constants were never made public; instead, were made available to select agencies through an executable program called SATRAK.

Agencies not privy to SATRAK, or who needed customized code for their project (including NASA) were forced to manipulate the original code found in SPACETRAK Report #3 themselves, leading to a wide variety of propagation codes based on the same model. David Vallado[iii], working through the Center for Space, released an AIAA paper in 2004, which attempted to reconcile the many codes into one standardized code. This new code was made available to the public through celestrak.com.

3. SGP4 for Johnson Space Center

Johnson Space Center (JSC) – Orbital Debris Program Office was one project that required the use of a Space Propagation Model. Though they had access to an executable version of SATRAK, customized code was needed to run within a larger program called PREDICT. A telescope observes a debris object and takes right ascension and declination data over a short period of time. The PREDICT code then takes this data and formulates a classical orbital element set (COES) that is fed into the space propagation model. The model can then estimate where the object will be in the near future so it can be found again and analyzed further.

Dr. Mark Matney of the Orbital Debris Program Office was the original architect of the PREDICT space propagation model in 1998. Dr. Matney based his work on the original SPACETRACK Report #3 as well as a publicly released SGP4 model from Goddard Space Flight Center (1997). Though Dr. Matney's program outputs workable results, the Orbital Debris Program Office was intrigued by Vallado's standardized code and wanted to explore the differences between codes. Furthermore, Dr. Matney's code did not accurately predict the location of the objects in some instances, though specific cases were not supplied for analysis.

This project, made possible through a research contract between JSC and Cal Poly San Luis Obispo, explores the differences in the two propagation codes. The scope of the project is as follows:

- Quantify the errors in the two codes based on 'truth' data.

- Deliver new PREDICT code integrated with updated SGP4 module.
- Deliver SGP4 propagator in MATLAB.

All analysis and deliverables are contained within this final report, as well as user guides for attached software code.

4. Procedure

This project was divided into three distinct stages.

1. Understanding the software.
2. Analyzing differences.
3. Programming deliverables.

4.1 Understanding the Software

The starting point of this phase was David Vallado's (et al) AIAA paper, Revisiting Spacetrack Report #3[iv]. In this paper, updates to the code are described in detail, and a link to the code in both FORTRAN and MATLAB is available at:

http://CelesTrak.com/software/vallado-sw.asp

In Vallado's analysis, results from the new code are compared against selected results from existing public code (i.e. GSFC, JPL, T.S. Kelso's FORTRAN). Though there are obvious differences, there is no proof that the new code is closer or further from the truth. What the paper does do is analyze each difference and explain why the results differ. These differences are commented as 'fixes' within the code and are cataloged in detail in Vallado's paper.

The code that Vallado created first prompts users for an operations mode. This selects how the code calculates epoch of the moon and sun for lunar/solar perturbations. Next, the code prompts users to input the type of run. Of the three choices, the first is catalog mode, where the code displays position, velocity, and COES for 12 hours forwards and backwards in time with a step of 20 minutes. Second is verification mode, designed to test the output of a given two line element (TLE) set and verify the results against a given file. Third is manual mode, where users define the propagation parameters. If manual mode is chosen, the user gets to determine if they want to use minutes from epoch, epoch, or day of year approach to define the window of time to analyze. The code itself converts epoch and day of year inputs to minutes from epoch when calling the propagator. Next, the user chooses which Earth constants to use, WGS-72 or WGS-84 referring to the World Geodetic System constants and when they became standard. Though the WGS-84 numbers are the current standard (through 2010), WGS-72 values are used in most SGP4 propagators because the 1984 values were not available at the time of creation. Though it is unknown what values SATRAK uses, analysis shows that WGS-72 numbers give more accurate results than WGS-84. Finally, the code calls for TLE inputs. The TLE file must be in the same directory as the main program, and the format of the TLE must be congruent with standards. Multiple TLEs may be in the same file with non TLE lines marked with a '#' symbol. The program then opens or creates an OUT file and prints the data. The OUT file has a header line containing the satellite number. Following lines are in the following form: a column for minutes from epoch, three columns of position data, three columns of velocity data, and four columns of date and time, delineated by a space.

4.1.1 MATLAB Code

The MATLAB code provided by Vallado contained some bugs. There were two major issues encountered. The first involved calling an undefined subroutine. The code tried to call a subroutine called ‘days2mdhms’, but the subroutine needed was named ‘days2mdh’. This subroutine had inputs of the year and day of year, and output the month, day, hour, minute, and second. After this bugged was corrected, another problem was encountered with time. The propagator uses minutes since epoch as an input, but instead was receiving days since epoch. This bug was corrected by dividing critical numbers by 1440 (minutes in a day), and reformatting the output to reflect changes. This code was tested in this debugged form and will be delivered to NASA in its current form to be used as an alternative to SATRAK.

4.1.2 FORTRAN Code

Before working on the FORTRAN code a compiler was needed. The chosen compiler was a free compiler available online called Silverfrost.

(http://download.cnet.com/Silverfrost-FTN95/3000-2069_4-10491439.html)

The code is written in fixed-format FORTRAN 90 language. Additionally, the complete code was spread amongst 25 small .FOR files and 4 common files upon initial download. Based on my understanding of the MATLAB code, I was able to collect all the relevant subroutines and create a new .FOR file that provided the same functionality as the MATLAB code. The code itself did not have any major bugs like the MATLAB code; however, significant time was spent formatting the combined code to fit the fixed-format FORTRAN standards.

4.1.3 PREDICT Code

JSC Orbital Debris Program Office provided their PREDICT code written in FORTRAN. The code is written as a series of subroutines. Though the focus was directed towards the SGP4 subroutine, the interactions with other parts of the program also needed to be taken into account. The SGP4 subroutine is called with a COES input and a time from epoch (when the COES was taken). The output is a single position and velocity vector based on the time from epoch. The code itself is relatively simplistic. It only takes into account J2 perturbations (the first mode of Earth's spherical harmonics), and then uses Kepler's solution with Newton's method to solve for position and velocity.

4.2 Analyzing Differences

JSC had three distinct test cases to analyze that were all debris in geosynchronous orbits. For these cases (8832, 25000, and 30000), JSC sent a TLE and SATRAK truth data. The truth data consists of a data set with 60 minute intervals between predictions and a data set with 360 minute intervals between predictions – but with a larger time range. These objects are of particular importance to the Orbital Debris. All three objects are pieces of debris from a Titan 3C transtage launch vehicle. The pieces themselves are not exactly in GEO, but are relatively close in a subGEO orbit. Below describes the break-up of objects 25000 and 30000.

> The breakup of the Titan 3C-4 transtage occurred on 21February 1992 at an altitude of ~35,600 km, inclination (INC) of 11.9 degrees and a right ascension (RA) of 21.8 hours. The operator of the GEODSS sensor on

Maui, Hawaii witnessed approximately 20 pieces in the breakup, but none were tracked at the time. Subsequent to the breakup, the U. S. Space Surveillance Network (SSN) identified three pieces of the debris and assigned them to the catalogue as SSN25000, SSN25001 and SSN30000.[v]

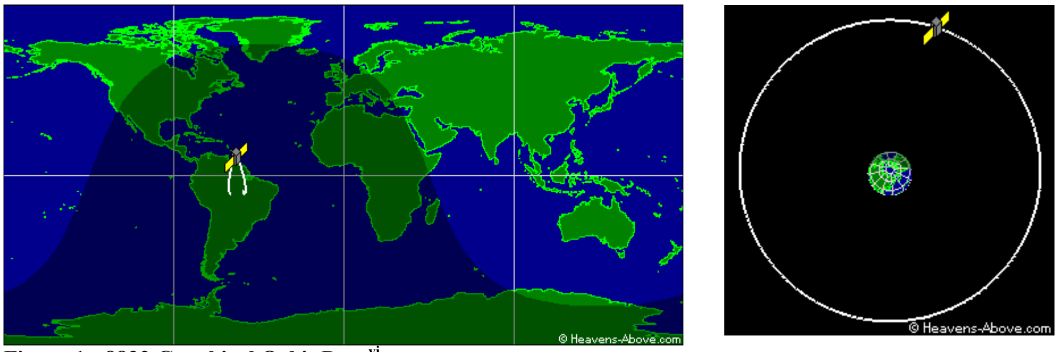


Figure 1 - 8832 Graphical Orbit Data[vi]

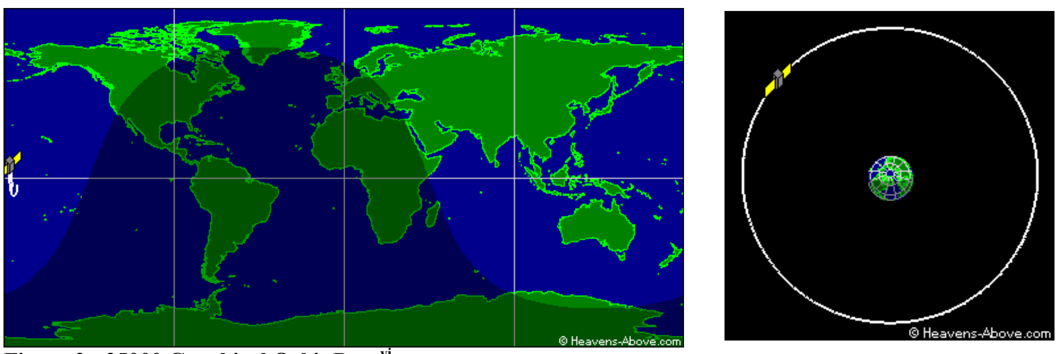


Figure 2 - 25000 Graphical Orbit Data[vi]

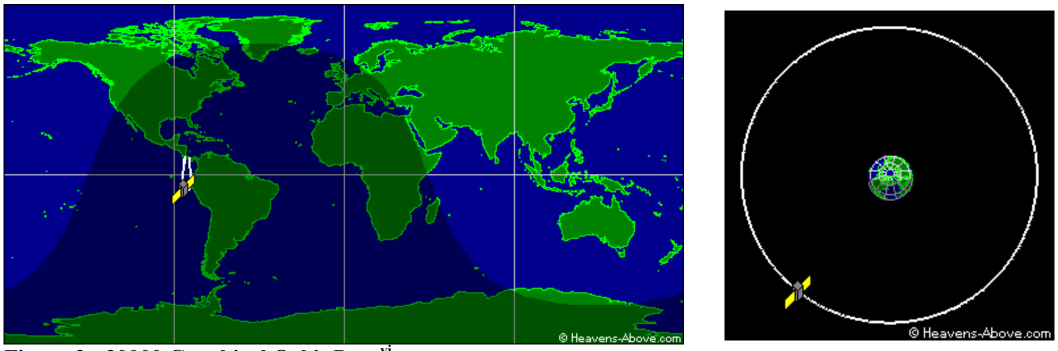


Figure 3 - 30000 Graphical Orbit Data[vi]

Next, to further the analysis, TLEs were gathered from Celestrak for non-geosynchronous orbits. Then, the same TLEs were again gathered on a later date. Analysis then compared where the object is based on the latter TLE, with where the object was predicted to be based on the former TLE. Analysis was completed on both the MATLAB and FORTRAN codes, as well as JSC's code.

4.2.1 Objects 8832, 25000, and 30000

First, each TLE was processed through the MATLAB and FORTRAN versions of Vallado's code and compared side by side with the SATRAK data. The MATLAB and FORTRAN codes can be programmed to have the same output as the SATRAK data. Results were first compared by hand, and then further analyzed using Excel. Error between SATRAK and Vallado's code is quantified as the absolute magnitudes of difference in the position and velocity vectors.

4.2.1.1 Code Modification

Since the ultimate goal of the project is to replace the current SGP4 propagator in JSC's code, the next step was to modify the FORTRAN code to act as a subroutine, instead of a stand-alone executable, with the same input and output variables as the original PREDICT model. Because the PREDICT model called for less functionality than the new code provided, much of the work done came in deleting unnecessary lines of code. New lines of code were added to remove any prompts given to the user; however, because the new code can accept LEO objects, a B* prompt was added to the process. There are comments within the new code with instructions on how to remove this if the user does not find it necessary.

Next, a simple program was created in which the user defines the COES from the TLE and gives a 'minutes from epoch' value. The program then calls the propagator and prints the expected position and velocity on the screen.

At this point, the inputs and outputs to the FORTRAN propagator and the PREDICT propagator were the same, and it was trivial to copy and paste the SGP4 subroutine from PREDICT into a new program called by the same simple program from above.

4.2.1.2 SATRAK Comparison

The objects given by JSC were also run through the modified program. However, since the new programs were restrictive in their input and output, only select data points were analyzed until trends were discovered.

4.2.2 Other Objects

To complete the analysis, truth data for LEO and MEO objects were found on SATRAK. These objects were:

- Polysat CP3 (31129)
- Polysat CP4 (31132)
- Iridium 33 Debris (33771)
- LES 9 (08747)
- ISS Tool bag (33442)

The analysis was completed using the modified code because of the single input and output. Error from truth data is again expressed as the absolute magnitude differences in position and velocity vectors.

4.3 Programming Deliverables

The final step of the procedure was installing the new code into the original PREDICT code. The majority of the leg work had already been completed in the analysis phase, though there were still some issues to be resolved. First, some of the variable and subroutine names needed to be altered to work with the PREDICT code. Specifically, both the PREDICT code and the new code have a subroutine labeled 'JDAY'.

Next, through analysis, it was found that the replacement code needs an extra bit of data to work properly. The PREDICT code only looks at J2 effects, which use a statistical average of spherical harmonic forces to determine perturbations. This makes epoch data irrelevant because it doesn't matter when you start your calculation, it only matters how long the J2 force has been acting on the object. Thus, when the propagator is called in the PREDICT model, it only inputs time from epoch and not the epoch itself. In contrast, the new code not only takes into account J2 effects, but also looks at forces on the object from lunar and solar influences. These effects are heavily dependent on epoch data because the location of the moon and sun need to be known to correctly calculate the direction and magnitude of the forces.

Epoch data for objects analyzed by the PREDICT code is determined in another subroutine, which is a predecessor for the SGP4 subroutine. To solve the problem, a global common file was created to pass the necessary data to the new SGP4 routine. Details of this can be found in comments in the code and in the User's Guide portion of this report.

Complete testing of the PREDICT code with the modified SGP4 code will be completed by the Orbital Debris Program Office.

5. Results

Sections 5.2 through 5.5 graphically display results that represent the magnitude of the difference in position and velocity vectors between the output of various SGP propagators and SATRAK output over the time span defined by given SATRAK truth data. The given time span ranges from ~15 days to ~27 days. Each section highlights results from the four main codes that were generated from this project. Furthermore, analysis of the results are presented after each section.

Sections 5.6 through 5.8 show the results of the four objects not in GEO orbit, with truth data taken from the TLE data provided by Celestrak.

Section 5.9 links the results to real-world application.

5.1 Data Points

All results were generated from TLEs for the following objects:

- 8832 (Titan debris)
- 25000 (Titan debris)
- 30000 (Titan debris)
- 31129 (Poly Sat 3)
- 31132 (Poly Sat 4)
- 33771 (Iridium 33 debris)
- 33442 (ISS tool bag debris)

5.2 MATLAB vs. SATRAK

Results are displayed for both scenarios created in SATRAK (60 and 360 minute intervals).

5.2.1 Object 8832

The maximum position error calculated between SATRAK and the MATLAB results is 32 cm and the maximum velocity error calculated is 0.023 mm/sec.

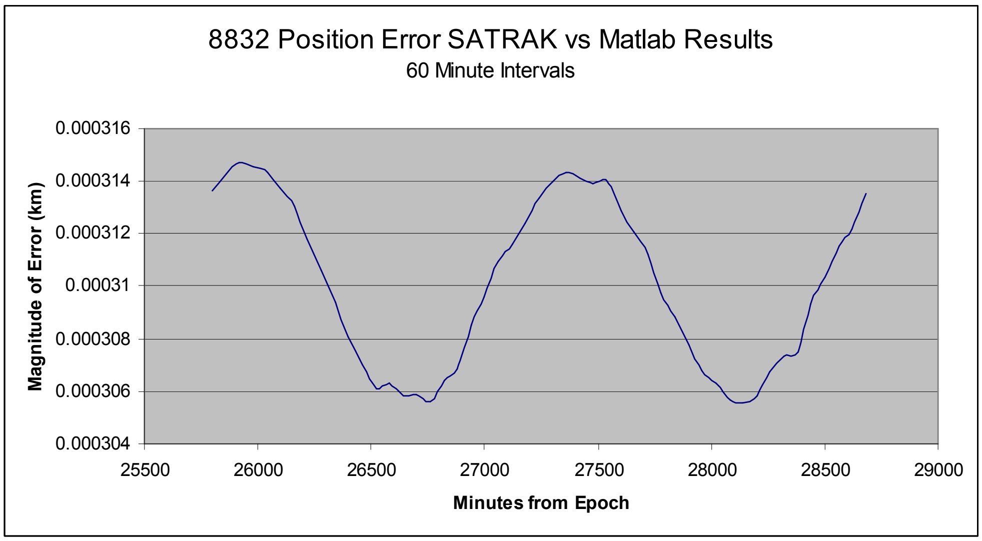


Figure 4 - 8832 Position Error MATLAB (60 min intervals)

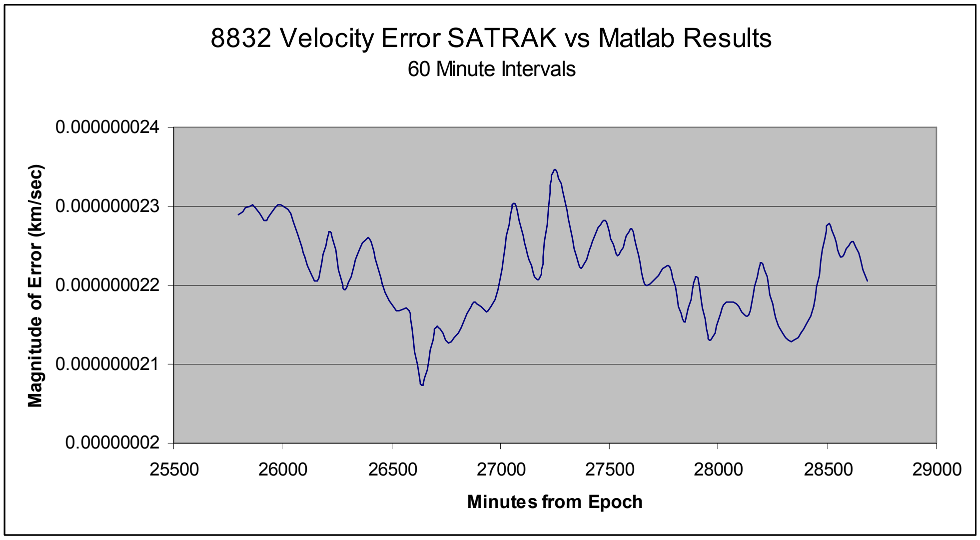


Figure 5 - 8832 Velocity Error MATLAB (60 min intervals)

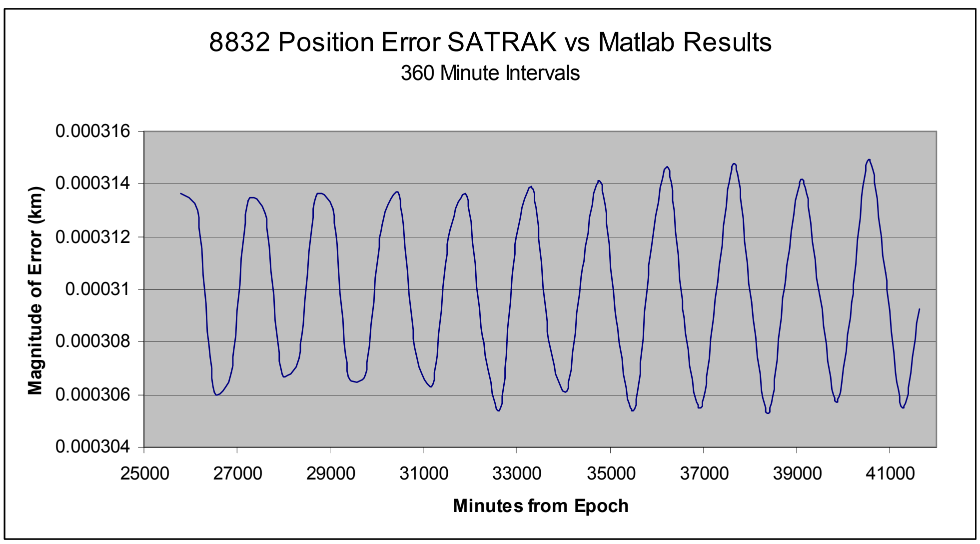


Figure 6 - 8832 Position Error MATLAB (360 min intervals)

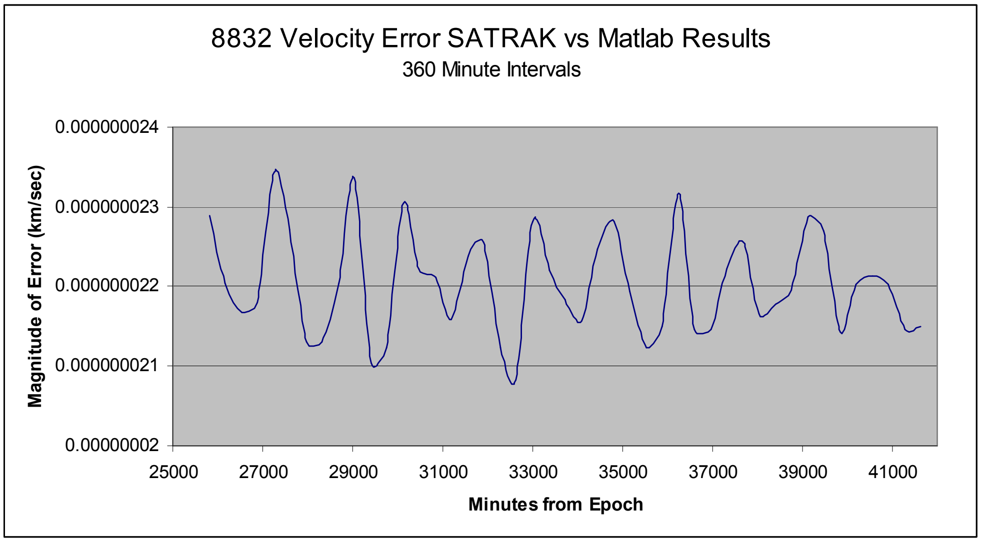


Figure 7 - 8832 Velocity Error MATLAB (360 min intervals)

5.2.2 Object 25000

The maximum position error calculated between the SATRAK and MATLAB results is 1.5m and the maximum velocity error calculated is .11 mm/sec.

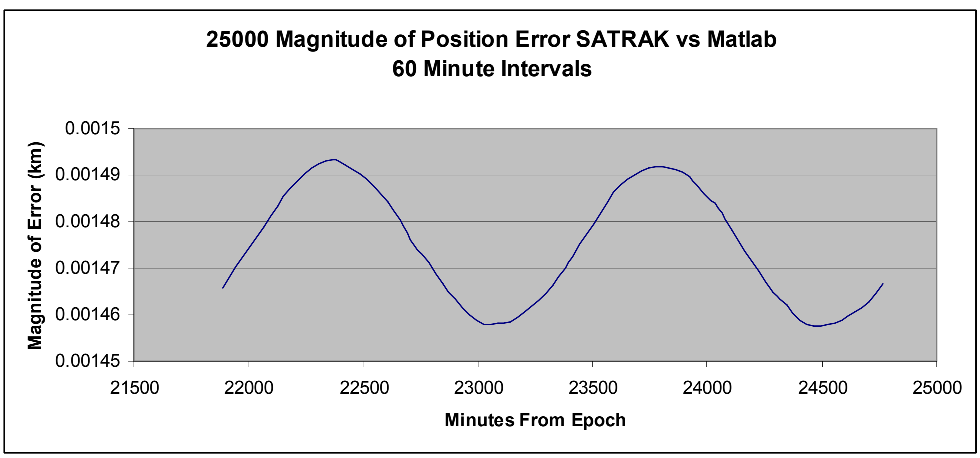


Figure 8 - 25000 Position Error MATLAB (60 min intervals)

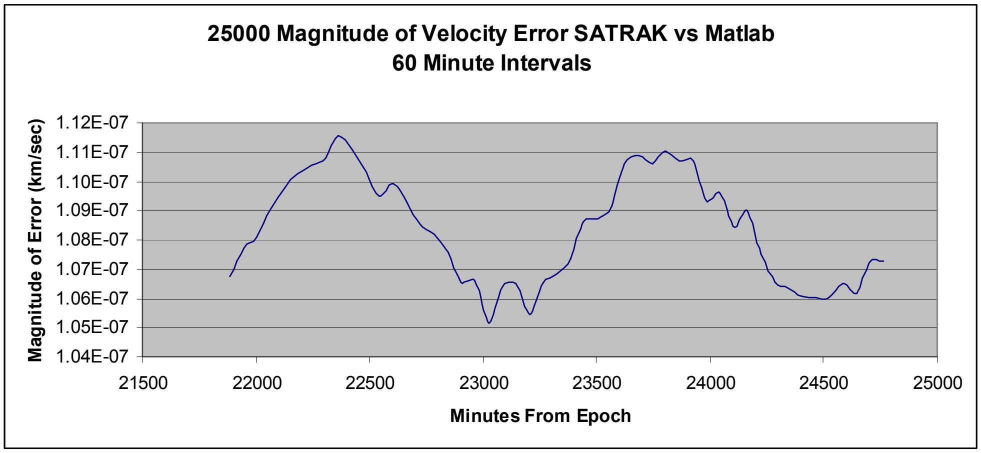


Figure 9 - 25000 Velocity Error MATLAB (60 min interval)

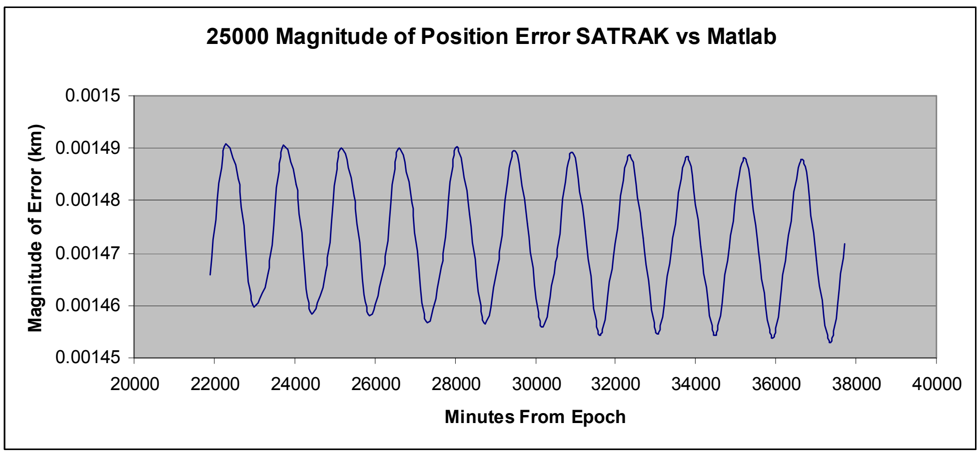


Figure 10 - 25000 Position Error MATLAB (360 min interval)

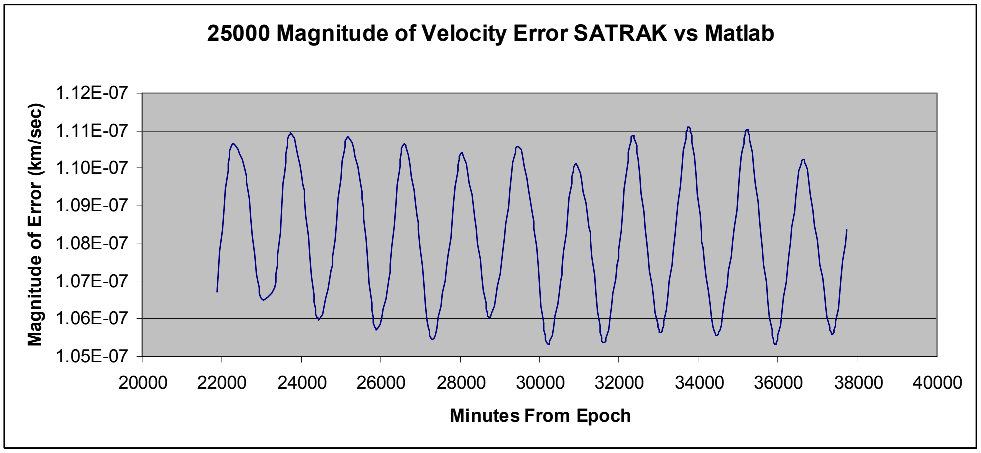


Figure 11 - 25000 Velocity Error MATLAB (360 min interval)

5.2.3 Object 30000

The maximum position error calculated is 51 cm and the maximum velocity error calculated is 0.038 mm/sec.

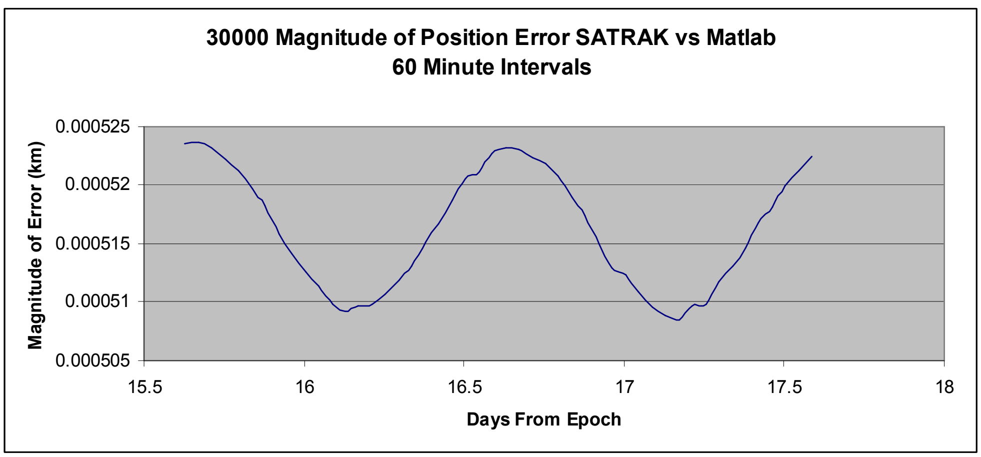


Figure 12 - 30000 Position Error MATLAB (60 min interval)

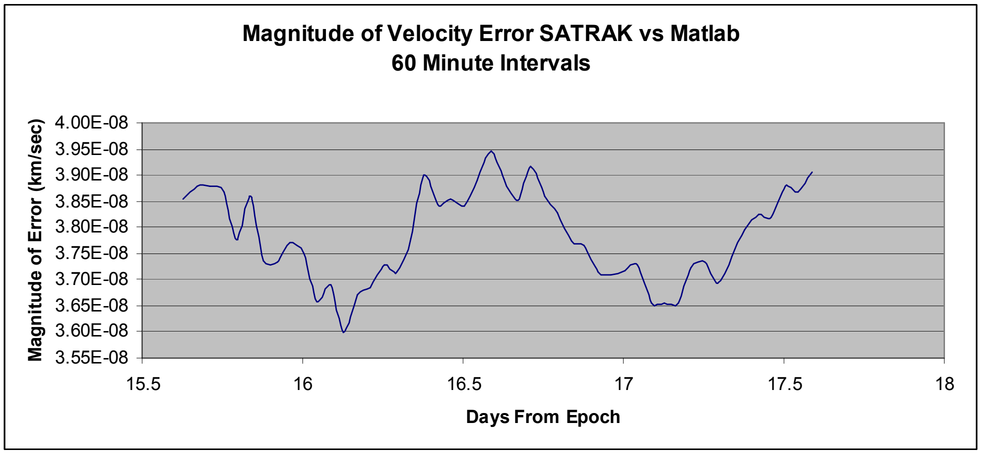


Figure 13 - 30000 Velocity Error MATLAB (60 min interval)

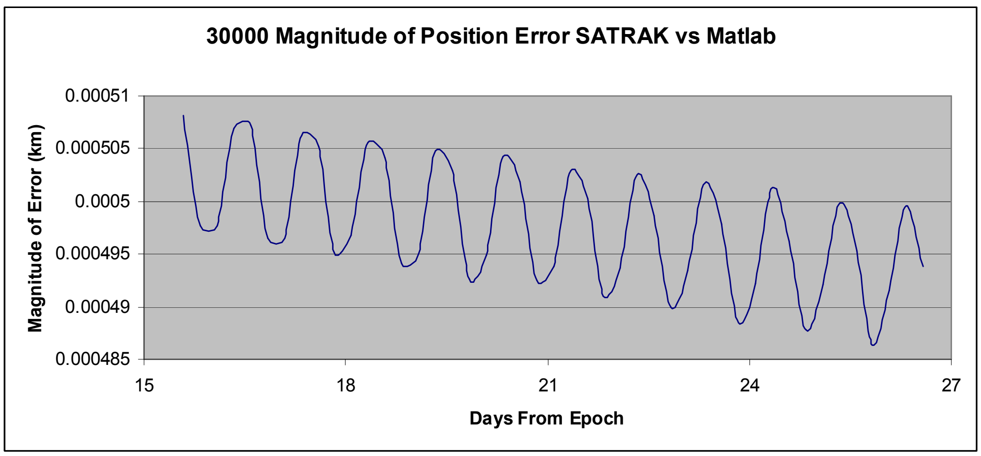


Figure 14 - 30000 Position Error MATLAB (360 min intervals)

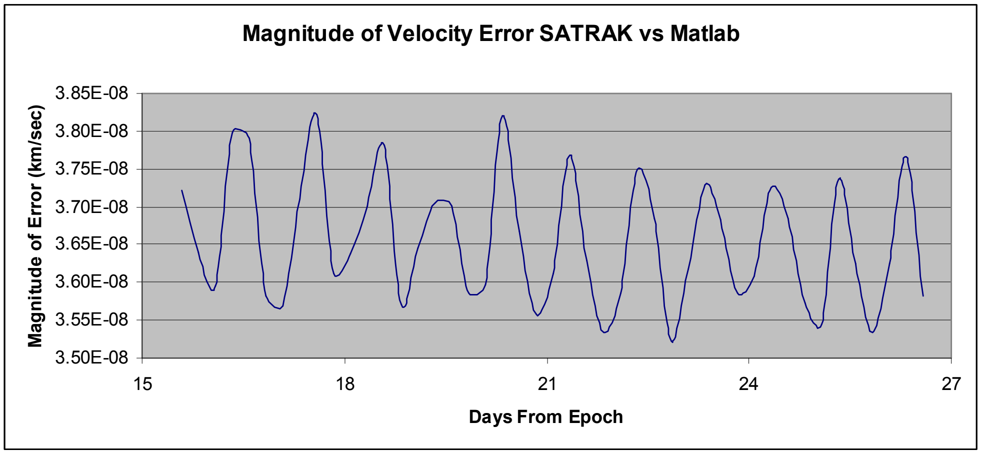


Figure 15 - 30000 Velocity Error MATLAB (360 min intervals)

5.2.4 Analysis - MATLAB

In a broad sense, the error between SATRAK and the MATLAB outputs are insignificant. For practical uses of this data, JSC is looking at units of kilometers, which is orders of magnitudes larger than the error given by MATLAB (1.5 m maximum). However, in each case, there is an obvious periodic variation which either corresponds to the orbital period of the satellite or rotation of the Earth. This can be explained by additional types of perturbations that are modeled in SATRAK that are ignored in Vallado's code. The hypothesis is that SATRAK encompasses some form of triaxilary effects; though there is no way to test this hypothesis.

Next, the difference between the 60 minute intervals and 360 minute intervals graphs was examined to check for contradictory data (see figure 16) by overlaying data for object 8832). As expected, the two graphs show the same long periodic trend; however, the 60 minute interval captures some short-term perturbations as well. The order of magnitude difference between the two intervals is less than the initial error and can safely be ignored. Thus, there is no contradictory data. Similar analysis was performed on all subsequent objects and codes with similar results and therefore, are not shown.

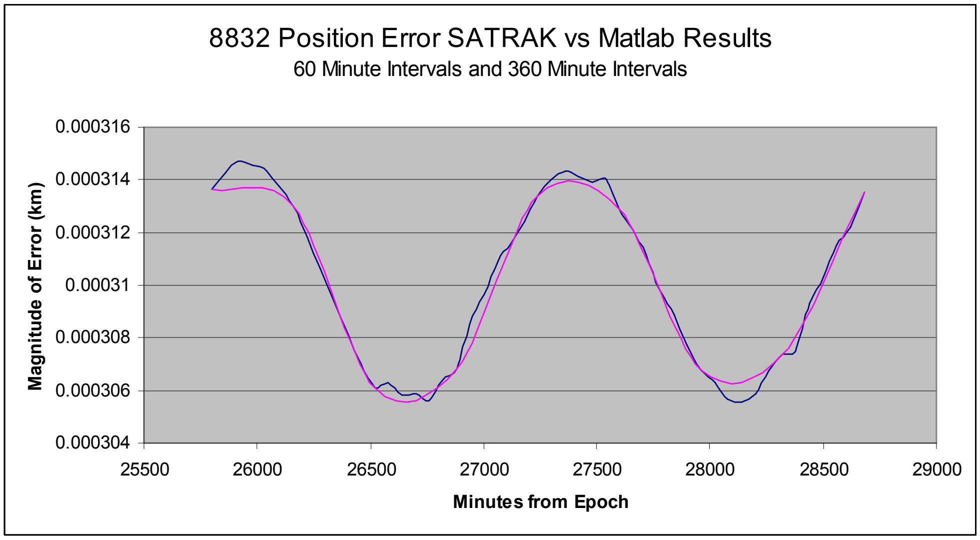


Figure 16 - Comparison of 60 and 360 minute intervals

5.3 FORTRAN vs. SATRAK

Results are displayed for both scenarios created in SATRAK (60 and 360 minute intervals).

5.3.1 Object 8832

The maximum position error calculated is 32 cm and the maximum velocity error calculated is 0.028 mm/sec.

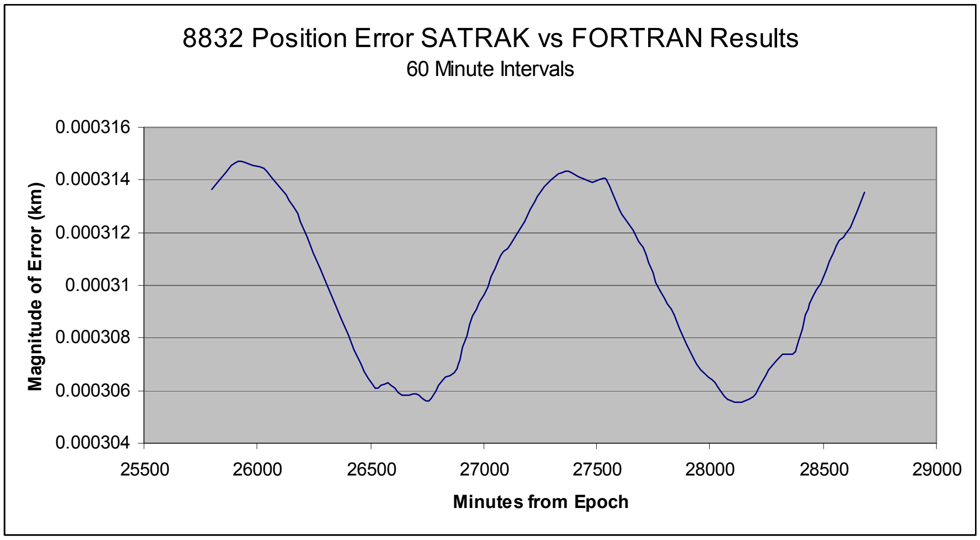


Figure 17 - 8832 Position Error FORTRAN (60 min intervals)

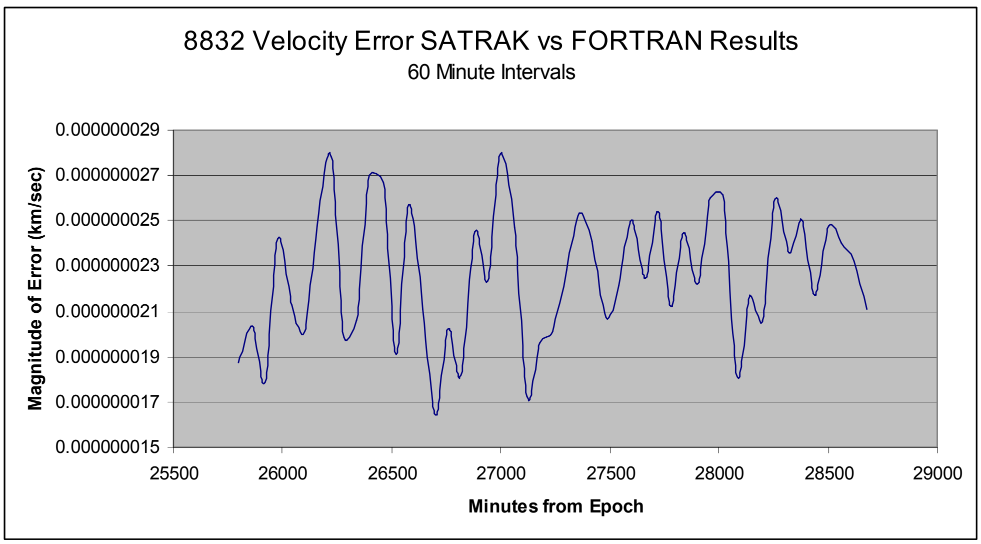


Figure 18 - 8832 Velocity Error FORTRAN (60 min intervals)

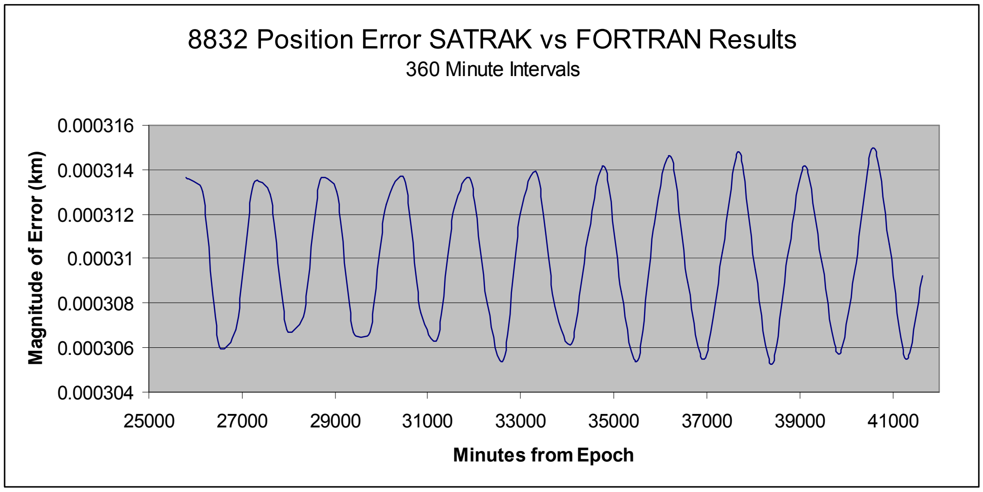


Figure 19 – 8832 Position Error FORTRAN (360 min intervals)

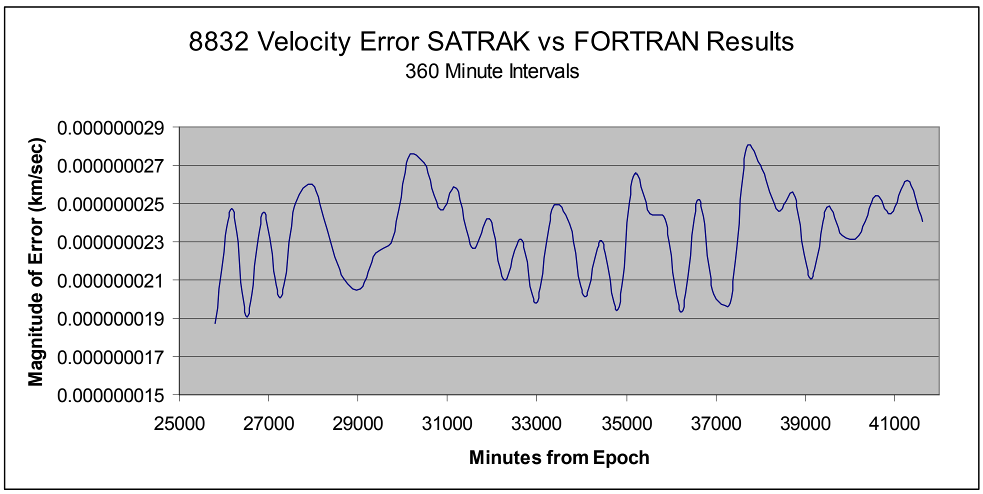


Figure 20 – 8832 Velocity Error FORTRAN (360 min intervals)

5.3.2 Object 25000

The maximum position error calculated is 1.5 m and the maximum velocity error calculated is 0.11 mm/sec

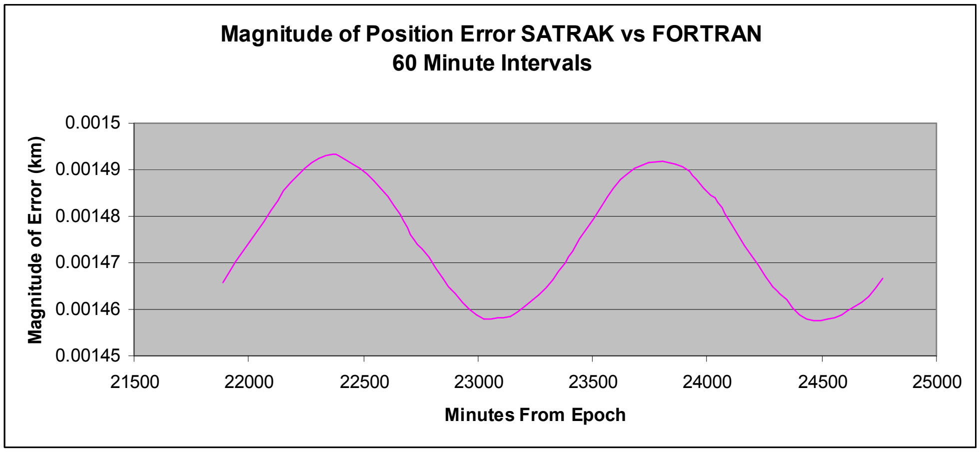


Figure 21 - 25000 Position Error FORTRAN (60 min intervals)

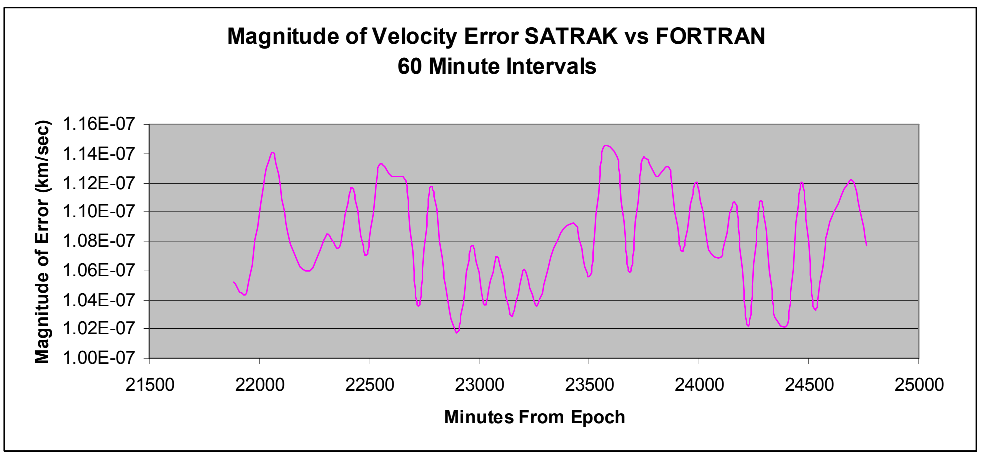


Figure 22 - 25000 Velocity Error FORTRAN (60 min intervals)

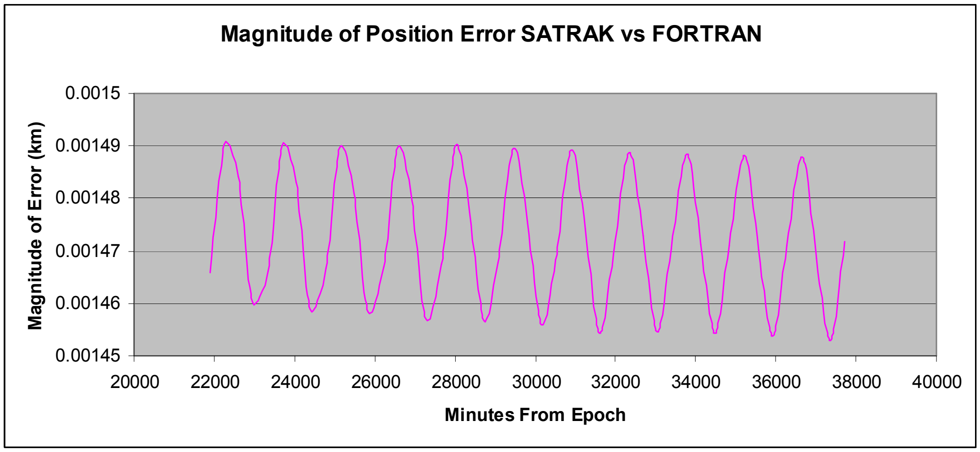


Figure 23 - 25000 Position Error FORTRAN (360 min intervals)

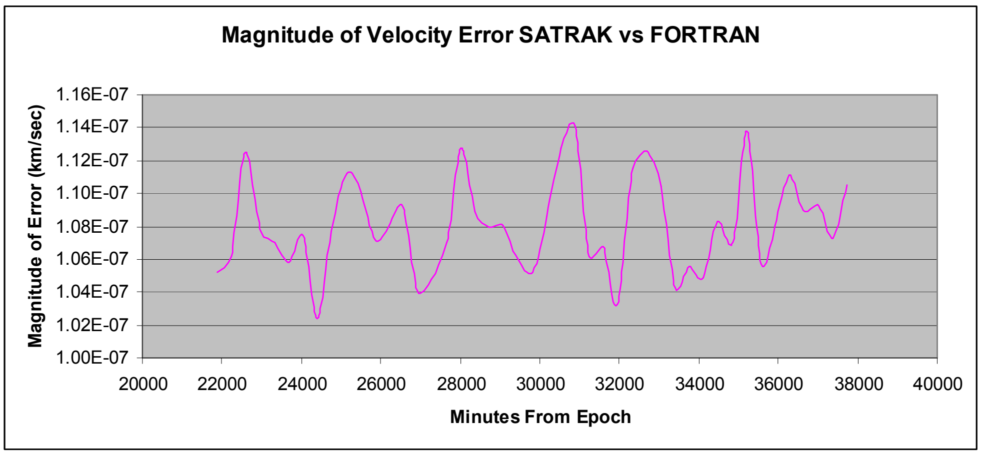


Figure 24 - 25000 Velocity Error FORTRAN (60 min intervals)

5.3.3 Object 30000

The maximum position error between the FORTRAN and SATRAK results is 53 cm and the maximum velocity error is 4.4 mm/sec.

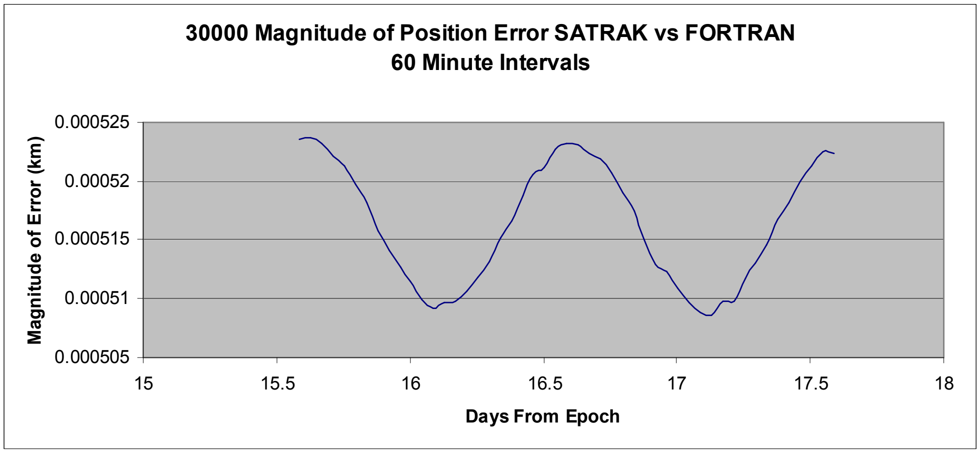


Figure 25 - 30000 Position Error FORTRAN (60 min intervals)

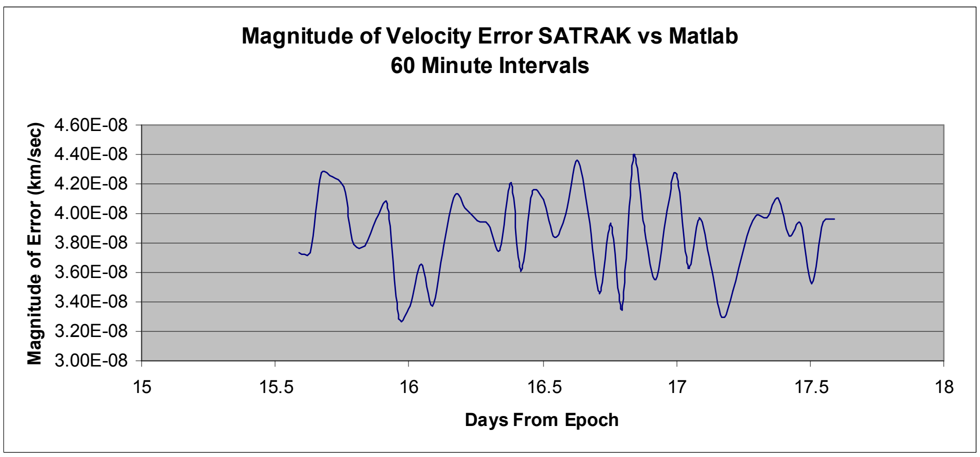


Figure 26 - 30000 Velocity Error FORTRAN (60 min intervals)

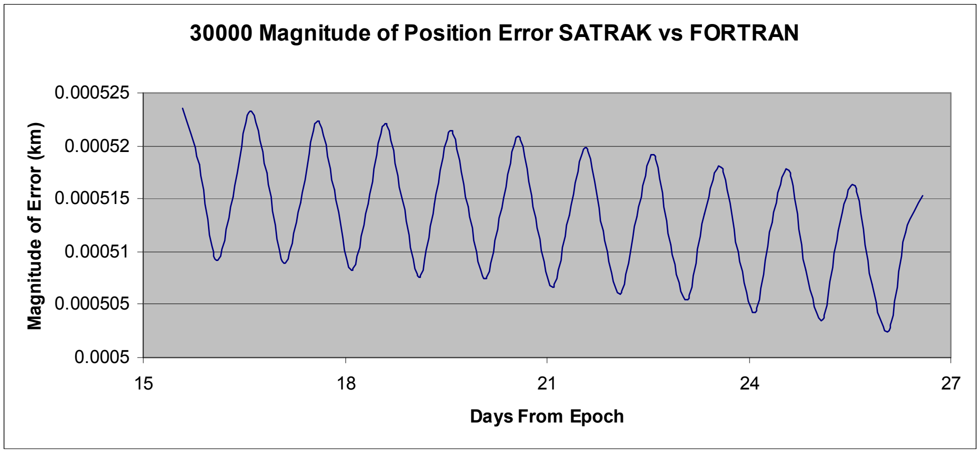


Figure 27 - 30000 Position Error FORTRAN (360 min intervals)

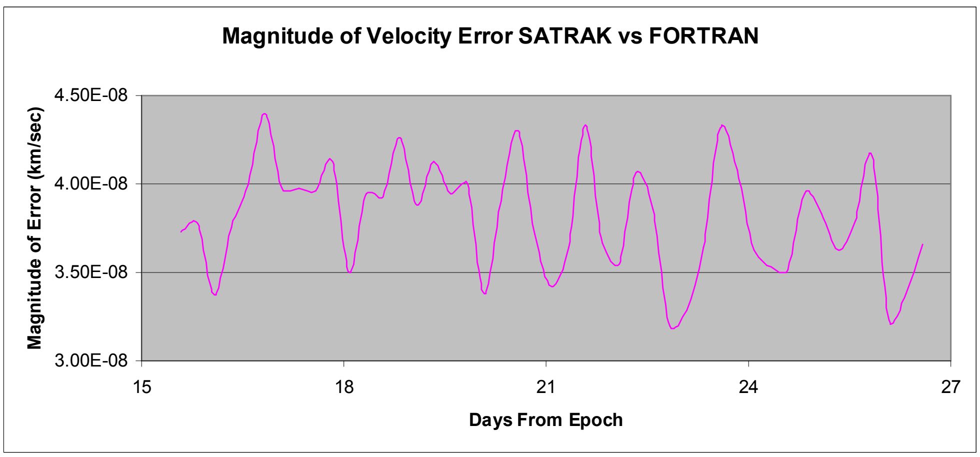


Figure 28 - 30000 Velocity Error FORTRAN (360 min intervals)

5.3.4 Analysis - FORTRAN

The FORTRAN code outputs very similar data to the MATLAB code in terms of position (see below). From visual inspection the two sets of data are indistinguishable.

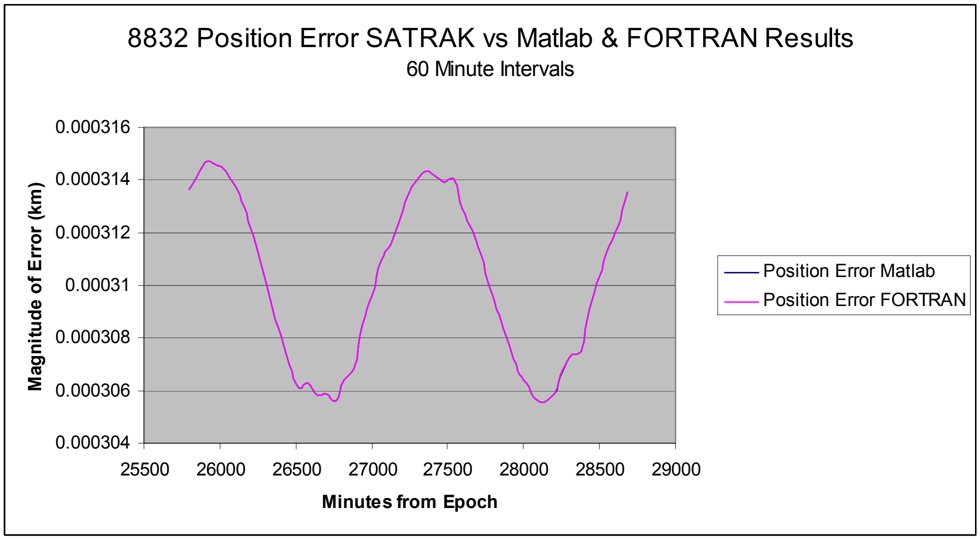


Figure 29 - Comparison of MATLAB and FORTRAN results (8832 position)

There is slightly more difference in the velocity output, see figure 26 (object 8832). The cause of the variation in the output of the FORTRAN code is unknown, but consistent with results found with object 25000. An initial hypothesis was a difference in the precision of the numbers carried through the FORTRAN calculations versus the MATLAB calculations; however, further results from the next section show otherwise. Though the fluctuations look significant, please remember that the graph is displaying differences of .004 mm/sec between the output of the data which is describing the motion of an object traveling roughly 3 km/sec.

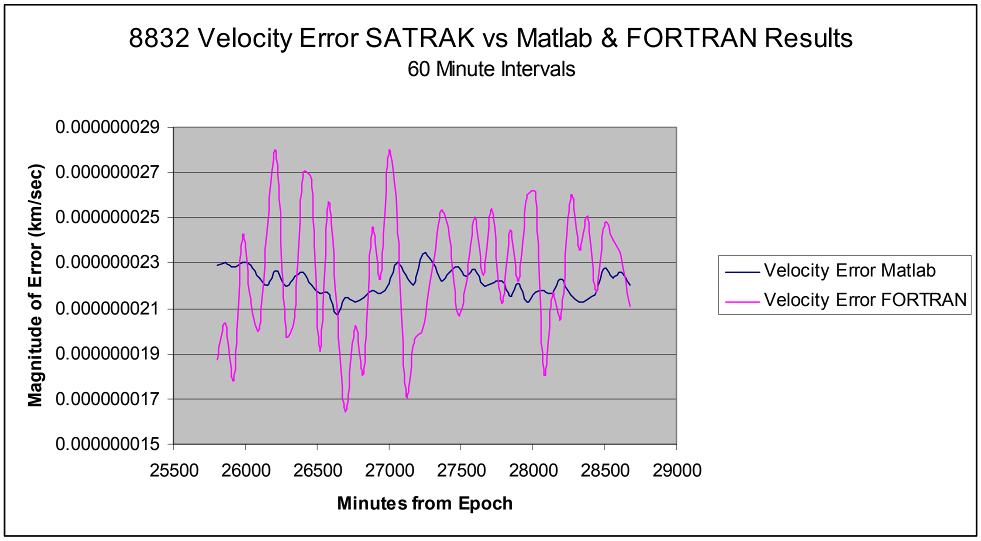


Figure 30 - Comparison of MATLAB and FORTRAN (8832 Velocity)

5.4 Modified Code vs. SATRAK

Results are displayed for both scenarios created in SATRAK (60 and 360 minute intervals). The modified code (labeled Vallado in the graphs) is designed to simulate the inputs and outputs of NASA JSC's existing code. Due to increased complexity, epoch data is also included as an input to the system. This leads to further complications when installing the upgrade into the PREDICT code, which is discussed in a later section.

5.4.1 Object 8832

The maximum position error calculated is 32 cm and the maximum velocity error calculated is 0.024 mm/sec.

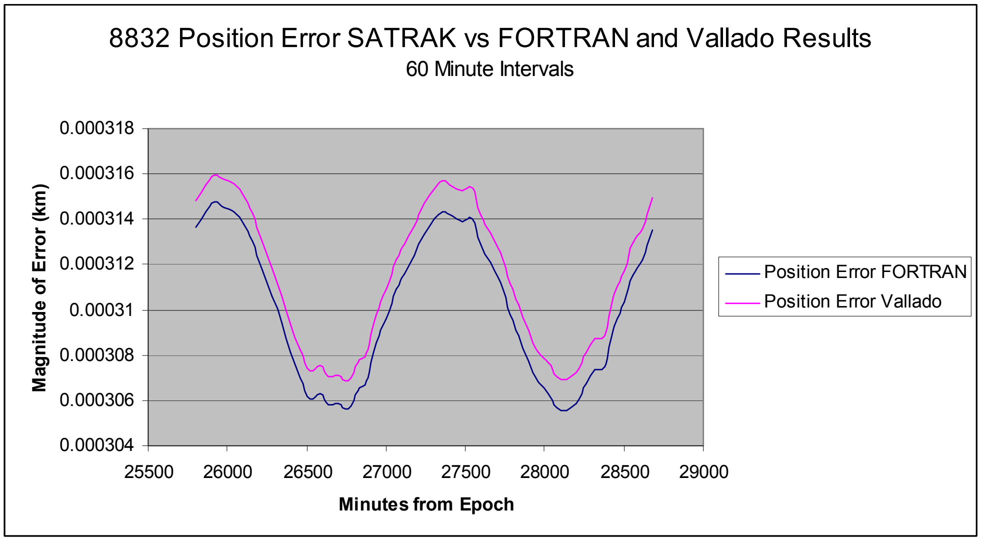


Figure 31 - 8832 Position Error Vallado (60 min intervals)

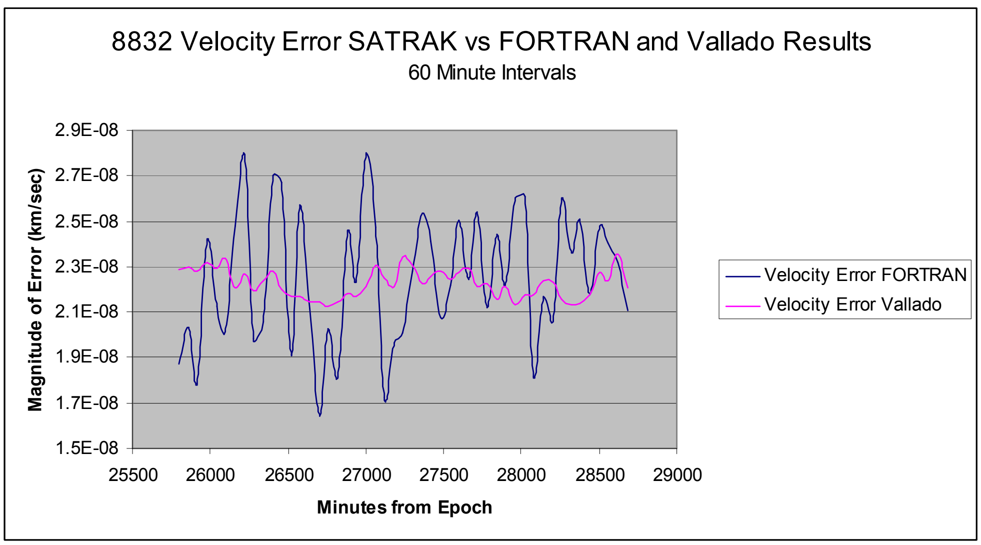


Figure 32 - 8832 Velocity Error Vallado (60 min intervals)

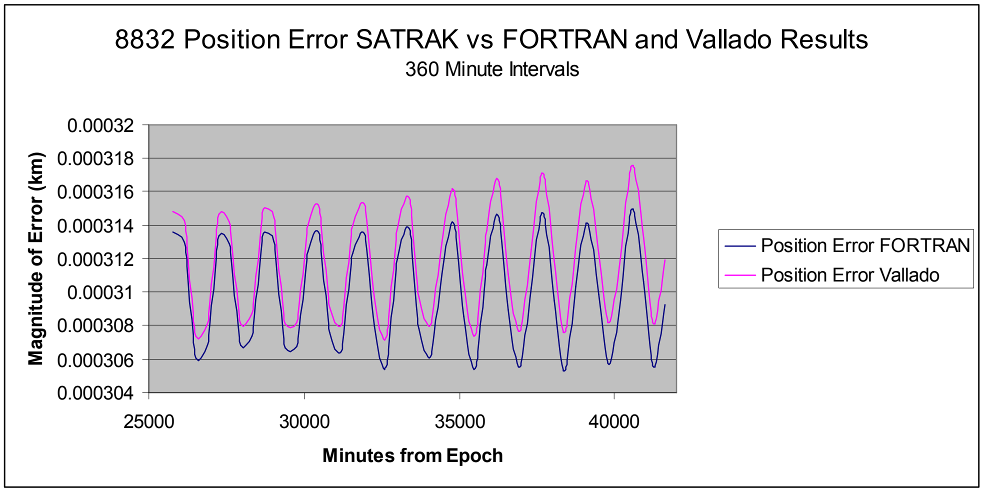


Figure 33 – 8832 Position Error Vallado (360 min intervals)

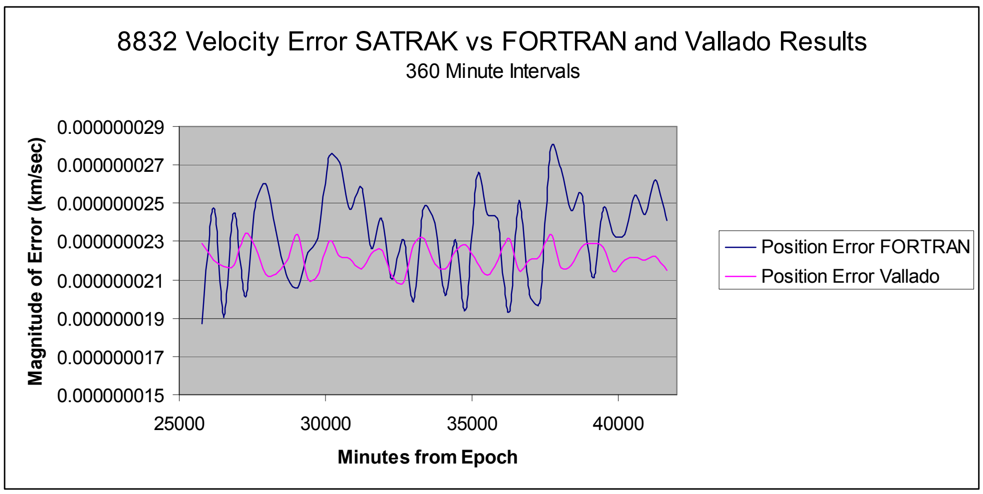


Figure 34 - 8832 Velocity Error Vallado (360 min intervals)

5.4.2 Object 25000

The maximum position error calculated is 1.5 m and the maximum velocity error calculated is .11 mm/sec.

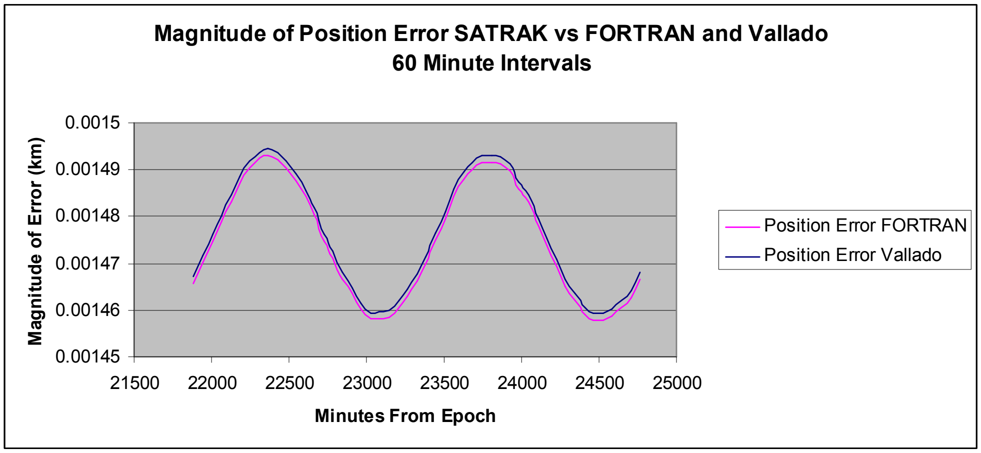


Figure 35 - 25000 Position Error Vallado (60 min intervals)

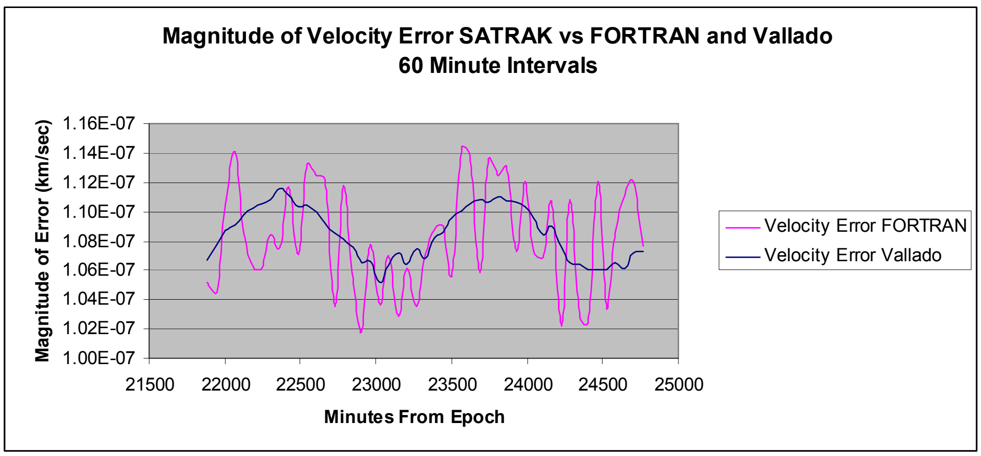


Figure 36 - 25000 Velocity Error Vallado (60 min intervals)

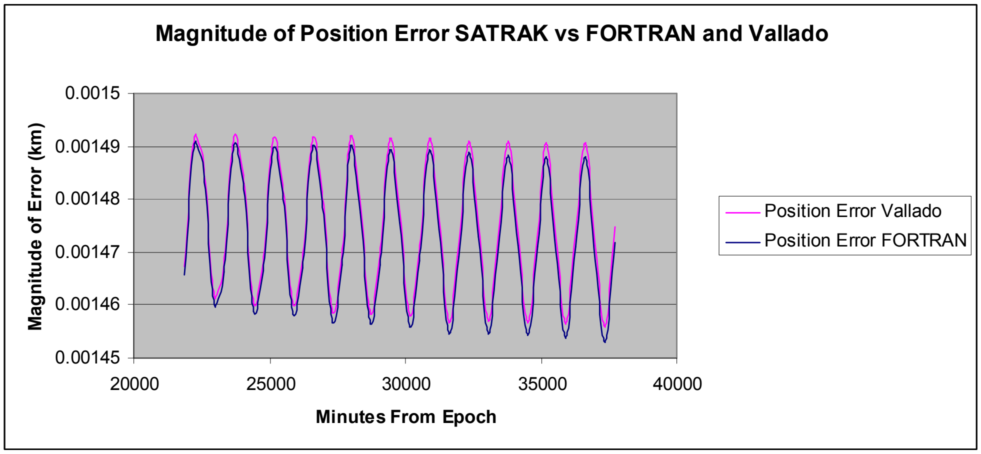


Figure 37 - 25000 Position Error Vallado (360 min intervals)

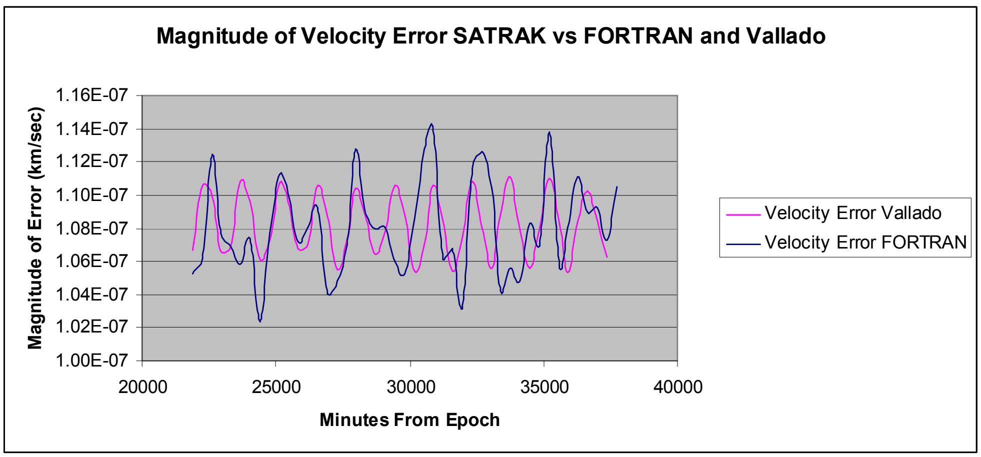


Figure 38 - 25000 Velocity Error Vallado (360 min intervals)

5.4.3 Object 30000

The maximum position error calculated is 6.3 m and the maximum velocity error calculated is 0.47 mm/sec.

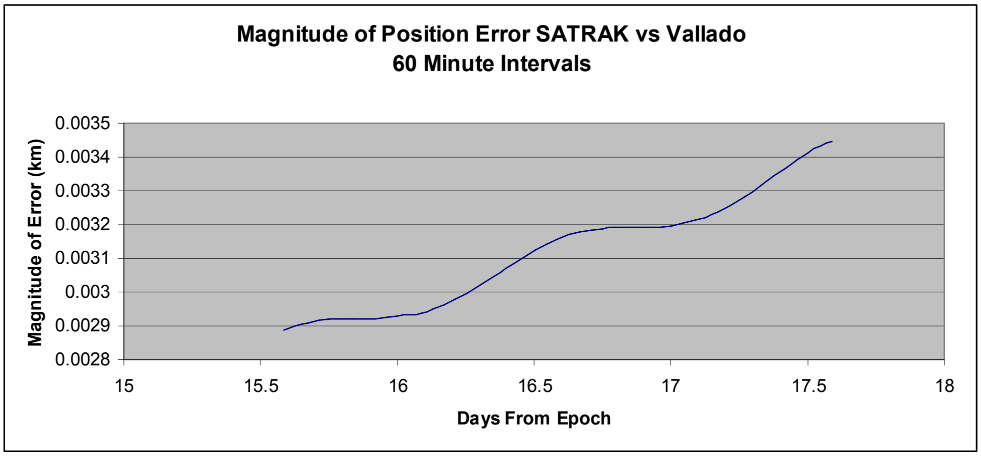


Figure 39 - 30000 Position Error Vallado (60 min intervals)

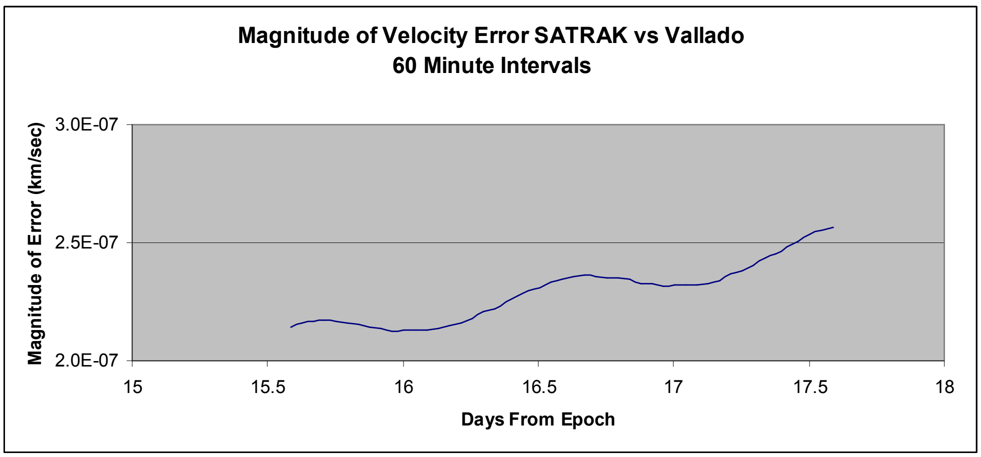


Figure 40 - 30000 Velocity Error Vallado (60 min intervals)

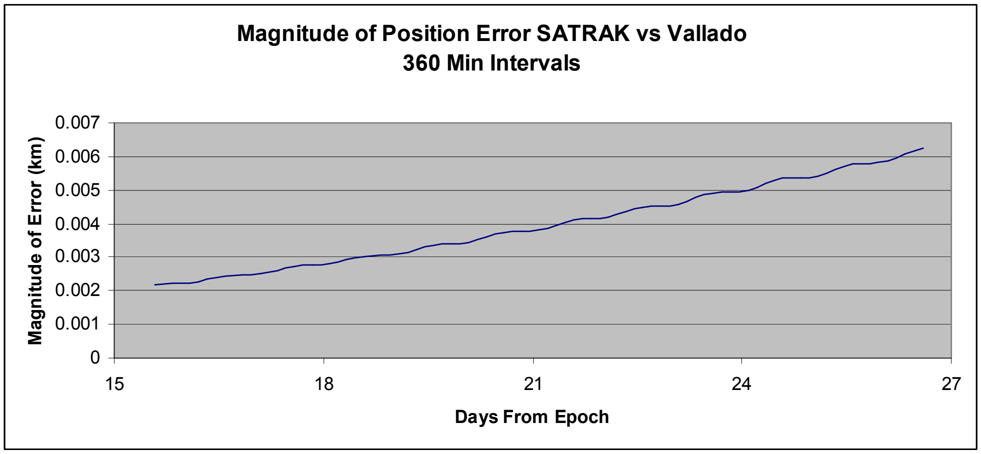


Figure 41 - 30000 Position Error Vallado (360 min intervals)

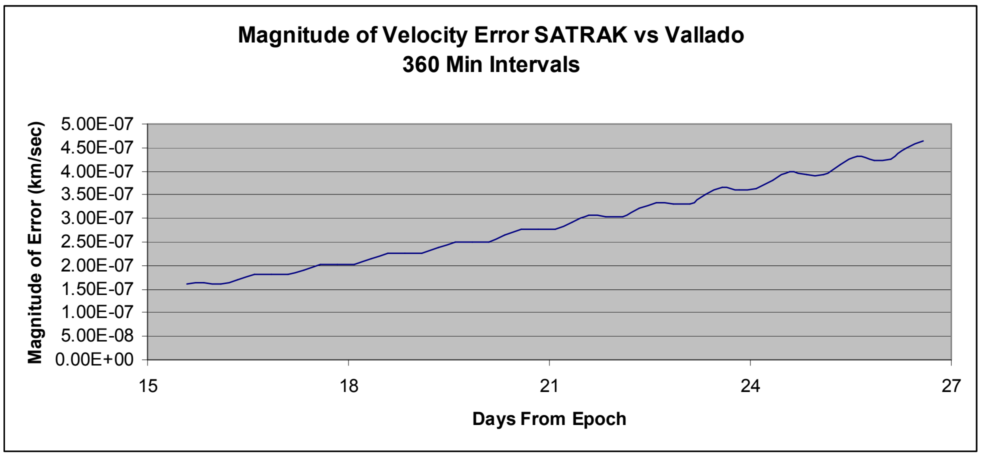


Figure 42 - 30000 Velocity Error Vallado (360 min intervals)

5.4.4 Analysis – Modified Code

Since the Vallado code is a stripped down version of the FORTRAN code, it is expected that the results come out equal or close to equal. Looking at the resultant graphs for object 8832 and object 25000 show some noteworthy trends. First, that the position error for the full FORTRAN code provides results slightly closer to truth. In the two cases, FORTRAN code is consistently fractions of a millimeter better. Why this happens is unknown. Second, the velocity error is more similar to the MATLAB error than the FORTRAN error. The reason for this is also unknown, but can be seen in figure 39.

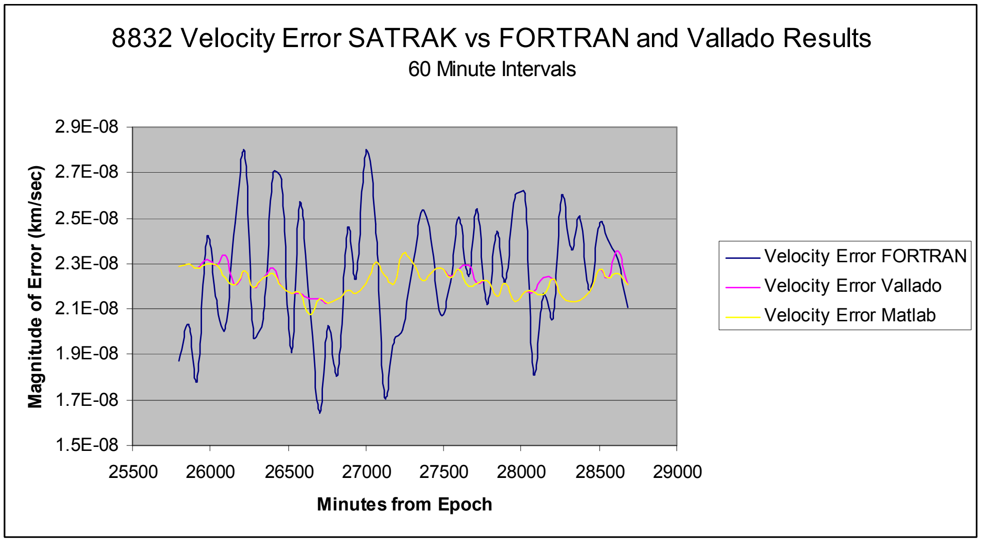


Figure 43 - Comparison of Vallado to MATLAB (8832 Velocity Error)

Finally, the results of object 30000 do not match the pattern initiated by the first two objects. The error involved is much greater than other runs and though has periodic tendencies, adheres more towards linear growth. Even though the error is larger than other runs, it is still orders of magnitude less than the operating threshold.

5.5 JSC Code vs. SATRAK

The JSC SGP4 subroutine code was taken directly from October 2008 version of the PREDICT code. In these tests, it is called by the same program that calls the modified code. In the resultant graphs, the JSC code is labeled as Matney, after Dr. Matney who wrote the original code.

5.5.1 Object 8832

The maximum position error calculated is 356.1 km and the maximum velocity error calculated is 25.4 m/sec.

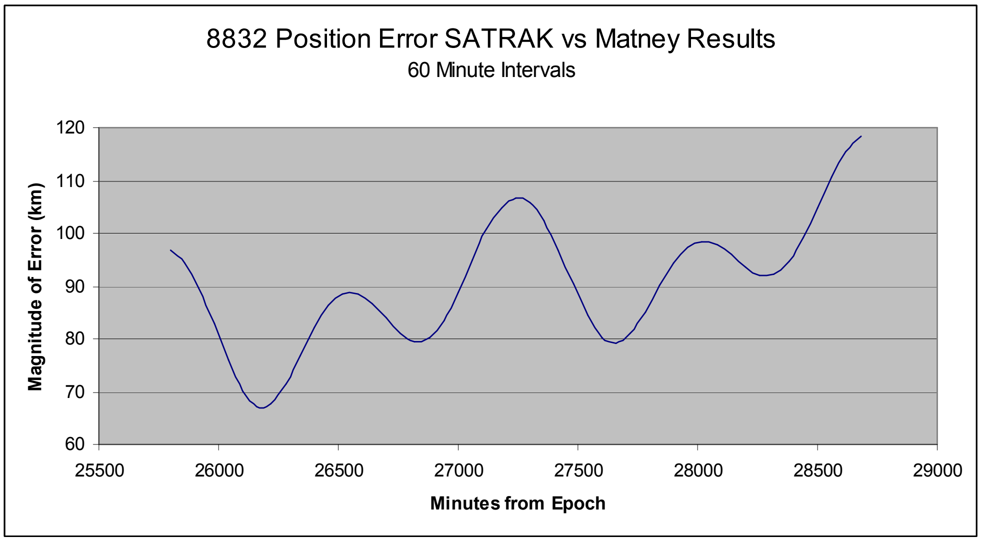


Figure 44 - 8832 Position Error Matney (60 min intervals)

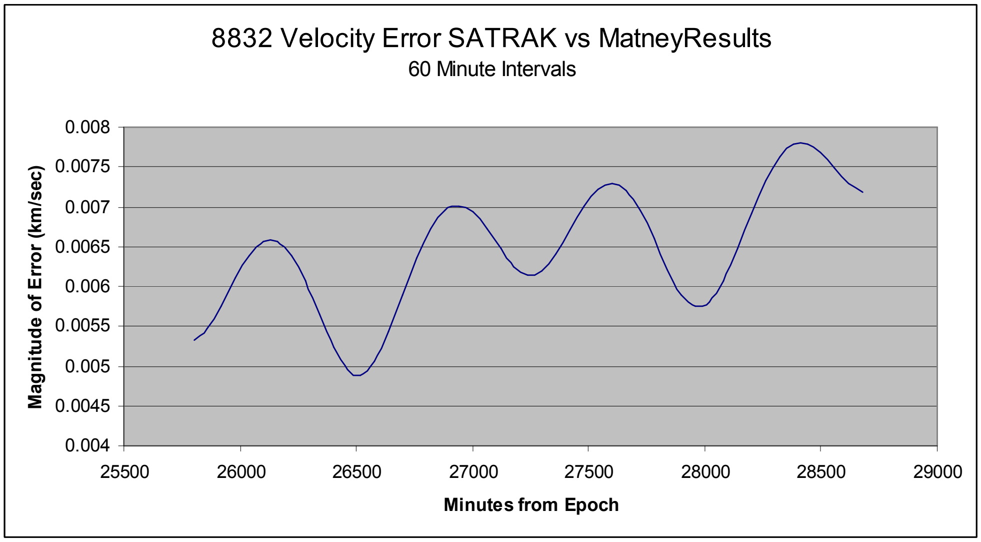


Figure 45 - 8832 Velocity Error Matney (60 min intervals)

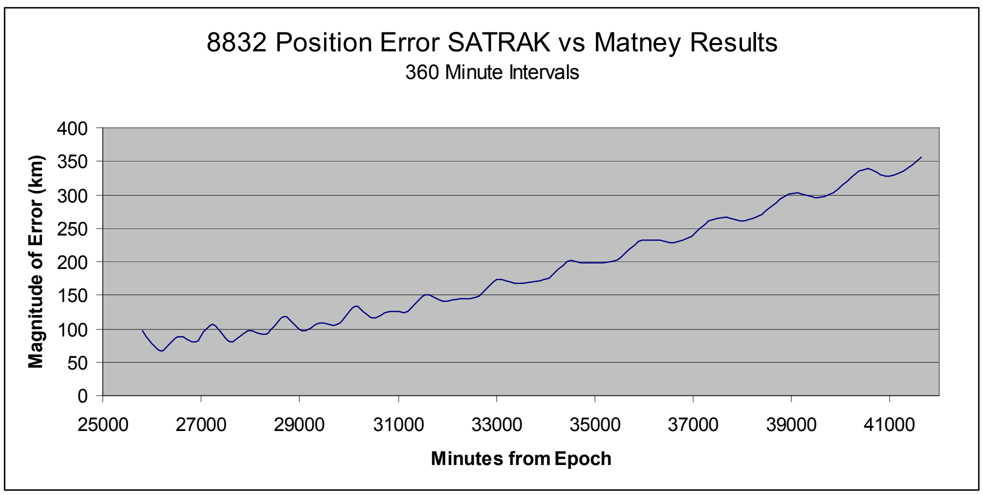


Figure 46 - 8832 Position Error Matney (360 min intervals)

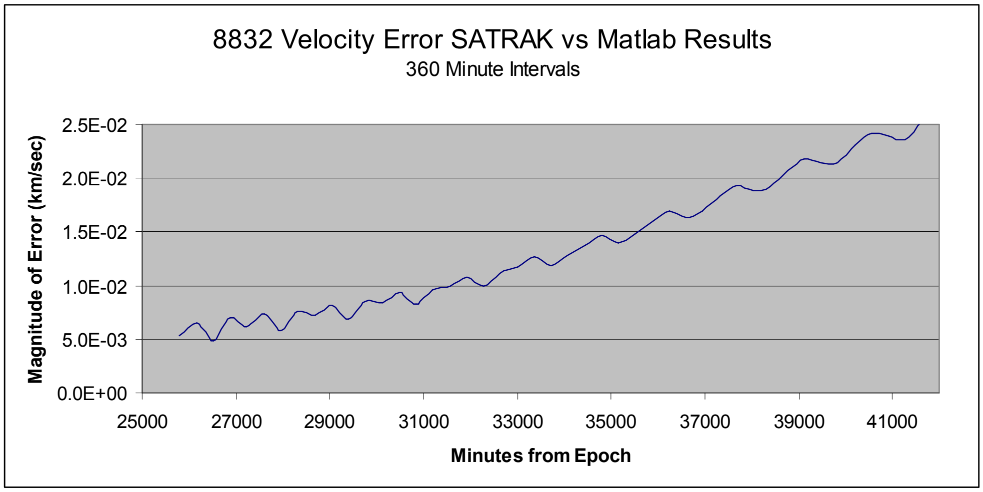


Figure 47 - 8832 Velocity Error Matney (360 min intervals)

5.5.2 Object 25000

The maximum position error calculated is 137.8 km and the maximum velocity error calculated is 10 m/sec.

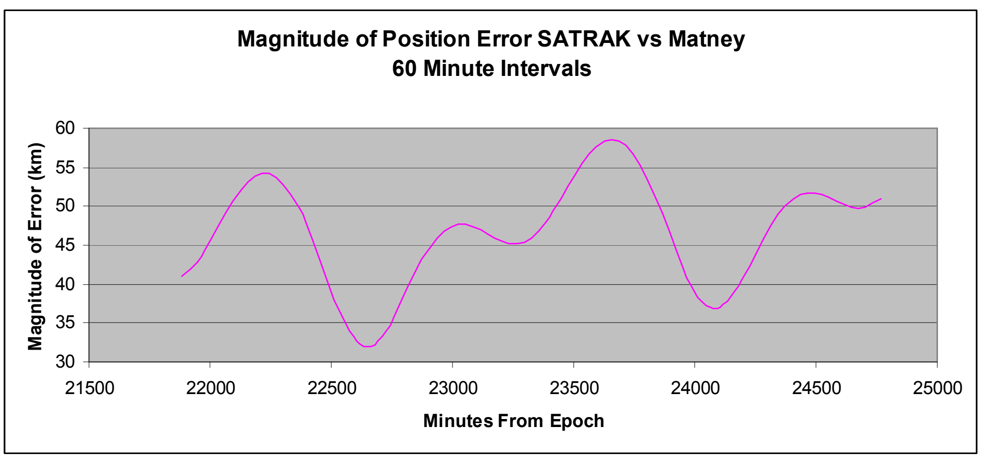


Figure 48 - 25000 Position Error Matney (60 min intervals)

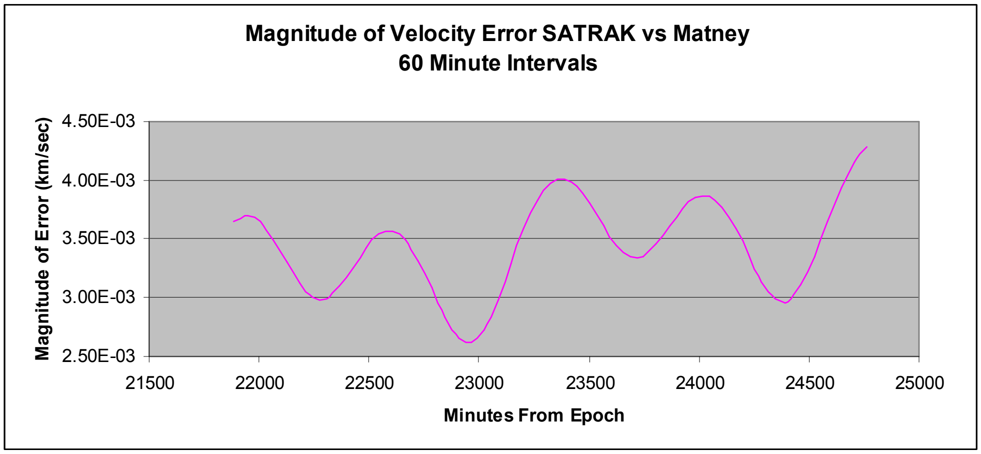


Figure 49 - 25000 Velocity Error Matney (60 min intervals)

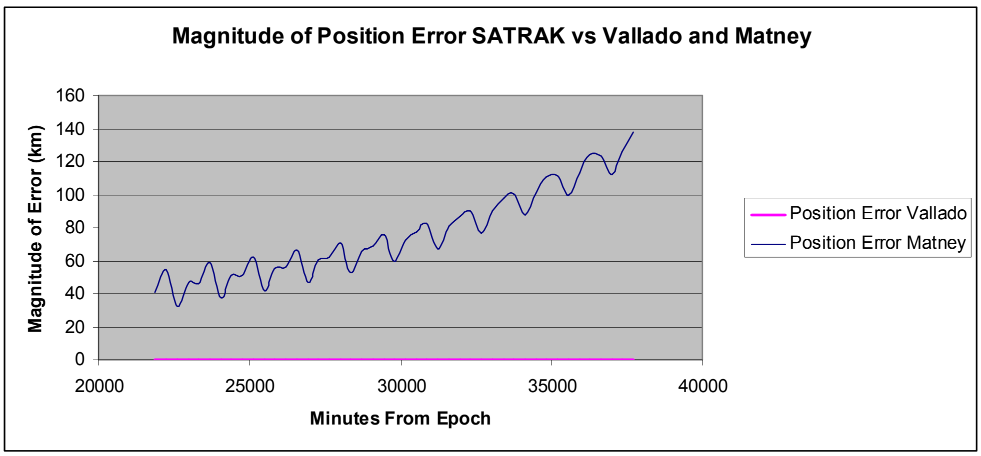


Figure 50 - 25000 Position Error Matney (360 min intervals)

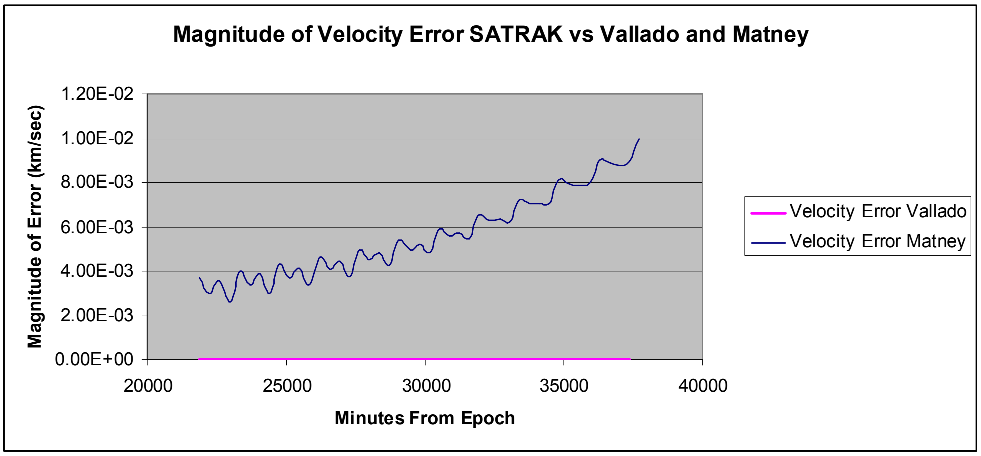


Figure 51 - 25000 Velocity Error Matney (360 min intervals)

5.5.3 Object 30000

The maximum position error calculated is 61.2 km and the maximum velocity error calculated is 4.4 m/sec.

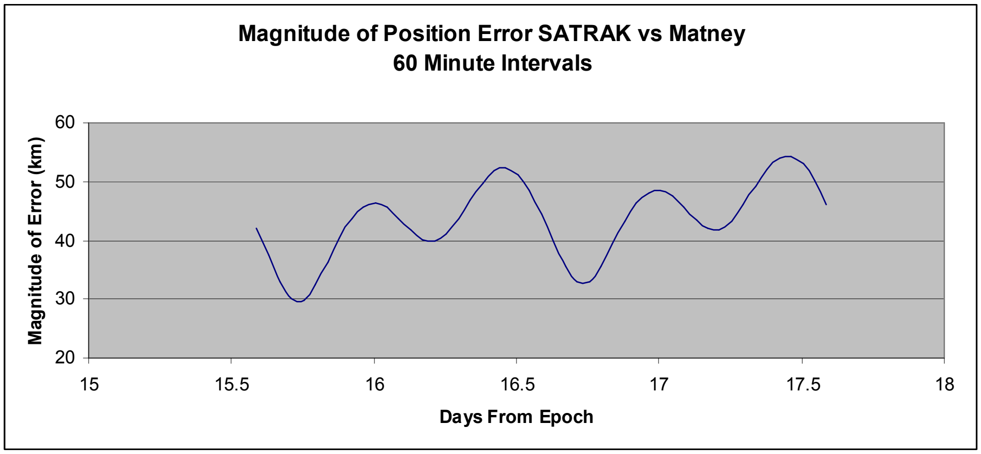


Figure 52 - 30000 Position Error Matney (60 min intervals)

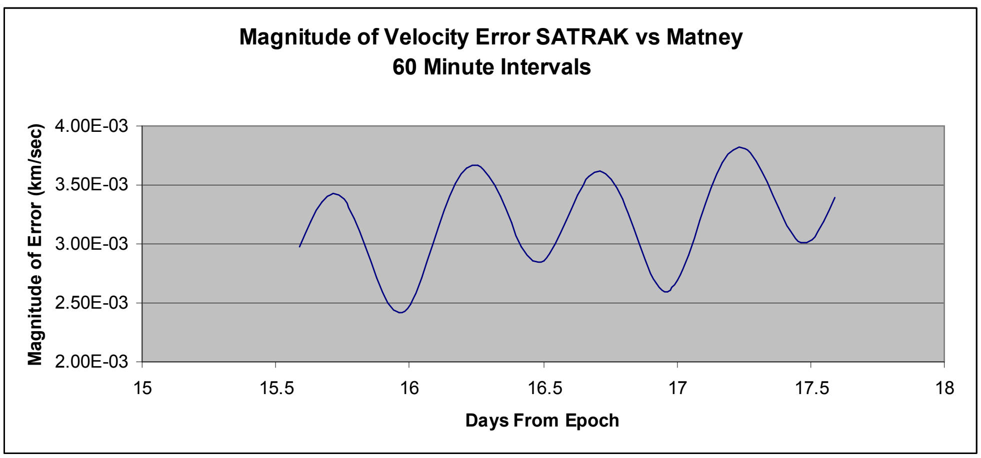


Figure 53 - 30000 Velocity Error Matney (60 min intervals)

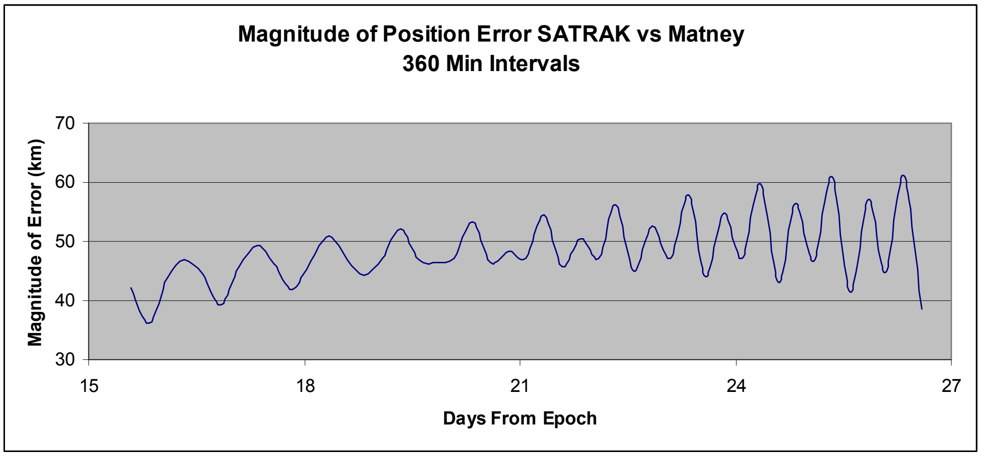


Figure 54 - 30000 Position Error Matney (360 min intervals)

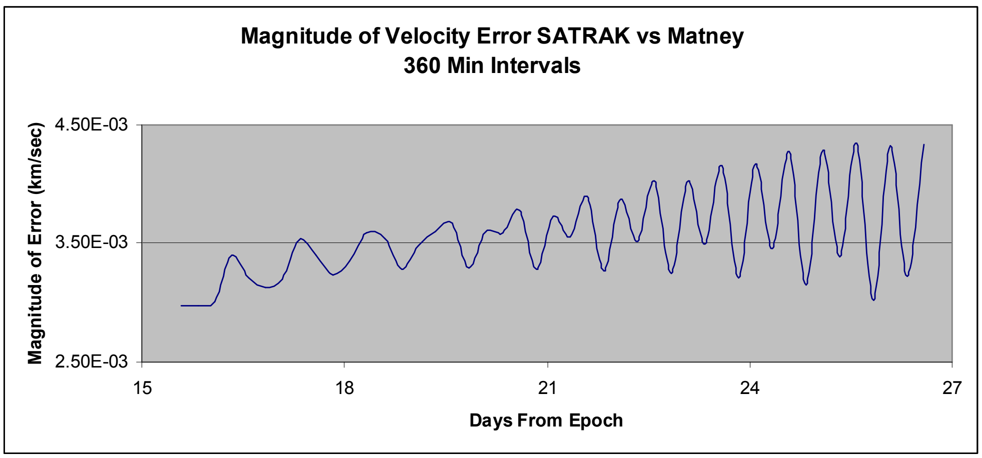


Figure 55 - 30000 Velocity Error Matney (360 min intervals)

5.5.4 Analysis – JSC Code

From visual inspection and direct comparison with the three previous codes, it is easy to see that the JSC code does not provide the same level of accuracy in the output. It is suspected that there are two main reasons for this. First, it is believed that there is a fundamental flaw in the programming of Kepler's solution. This is skewing output results even before any perturbations are taken into account. Second, the JSC code does not account for any third body interactions. This causes the periodic variations seen in the graphs, as sometimes the third body effects help the simulation while at other times it hurts the simulation.

Furthermore, the scope of the error is of similar magnitude to the threshold, and thus cannot be ignored like the previous three cases. This will be discussed further in Section 5.9 Operational Issues.

5.6 Polysat

Polysats are small satellites designed by Cal Poly under the Cube Sat project.. CP3 and CP4 were launched into 640 km sun-synchronous orbits on April 17, 2007. For the Polysats, there were two sets of truth data. First, there was SATRAK data of the CP3 available. Second, there was TLE data provided by Celestrak. Using archived and current TLE data, it is possible to see how far away the prediction is from official data.

For this analysis, only the modified code was compared to truth data. This is to check the operation potential of the new software beyond GEO objects. The original code was not tested because it was not designed for LEO or MEO objects.

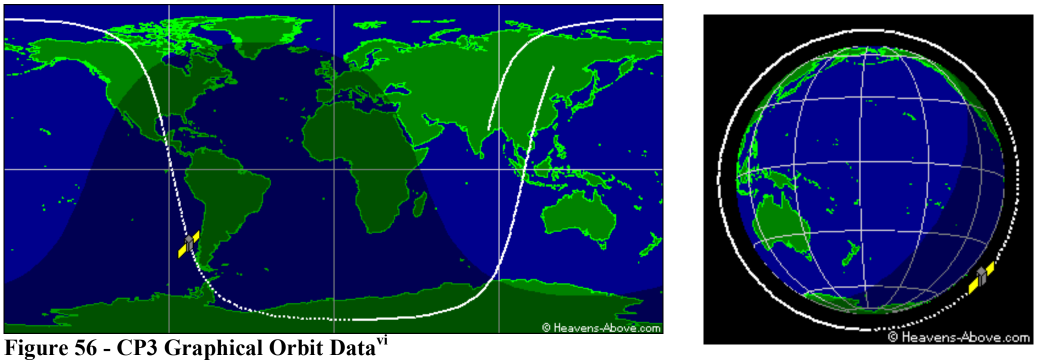


Figure 56 - CP3 Graphical Orbit Data[vi]

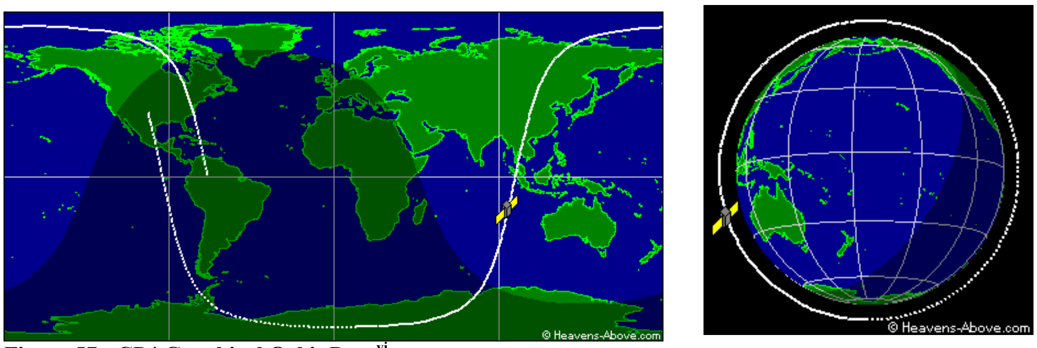


Figure 57 - CP4 Graphical Orbit Data[vi]

5.6.1 CP3

Below are graphs showing the magnitude of the vector error in both position and velocity of the modified code compared to SATRAK truth data.

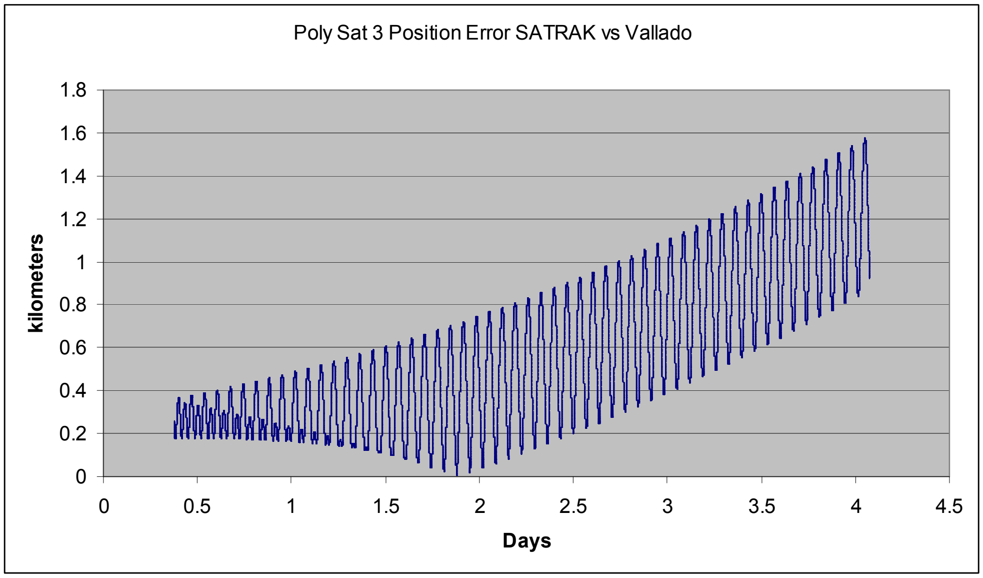


Figure 58 - CP3 Position Error Vallado (10 sec intervals)

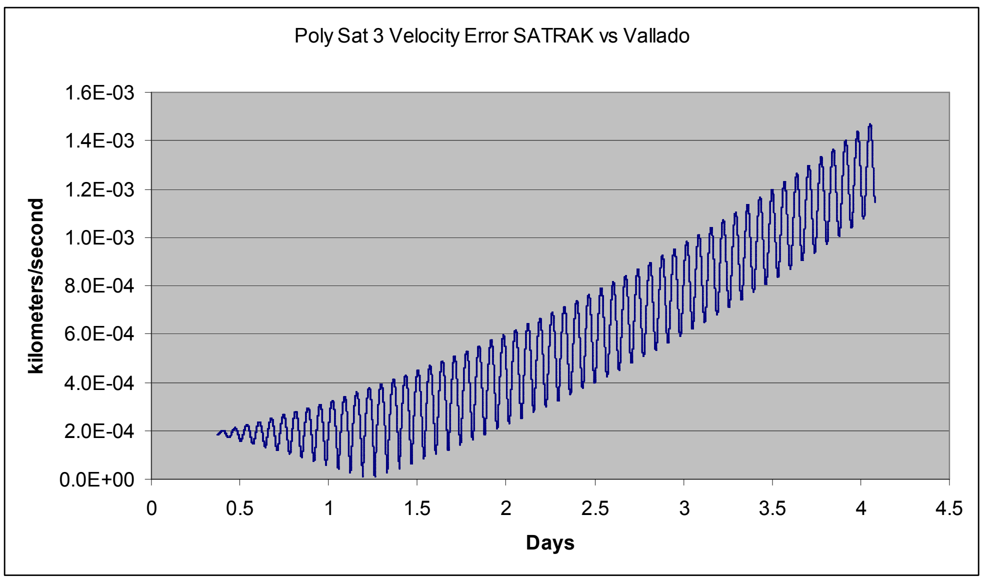


Figure 59 - CP3 Velocity Error Vallado (10 sec intervals)

These graphs show two trends. First, the variation in error is a function of the objects period and not of the Earth's rotation. The periodic motion of both the position and velocity corresponds to the mean motion of the spacecraft. Second, the slope of the error trends is also proportional to the mean motion of the object. In other words, the faster the object is moving, the faster error will be a significant factor.

Beyond SATRAK, there is also TLE data that can be used as truth data. TLE's were pulled from Celestrak at four different times and are shown in the table below.

Table 1 - TLE readings for CP3

Date of Reading	TLE
4/22/09	1 31129U 07012N 09112.08372896 .00000092 00000-0 31739-4 0 8719 2 31129 97.9954 167.8345 0103695 98.8601 262.4373 14.52123890106741
5/1/09	1 31129U 07012N 09121.11032672 .00000119 00000-0 37832-4 0 8897 2 31129 97.9938 176.4337 0103587 70.9779 290.2602 14.52126376108059
5/6/09	1 31129U 07012N 09126.14038987 .00000017 00000-0 14486-4 0 9001 2 31129 97.9934 181.2249 0103743 55.7944 305.3039 14.52126523108786
5/23/09	1 31129U 07012N 09143.15980035 .00000024 00000-0 16378-4 0 9438 2 31129 97.9925 197.4331 0102509 3.5446 356.6451 14.52130919111251

Table 2 - CP3 TLE Results

Days since 4/22 TLE	Error in Position (km)	Error in Velocity (km/sec)
9.03	0.638	0.008662
14.06	2.802	0.002583
31.08	9.197	0.000703

These results are significantly better than expected given the SATRAK analysis. This shows that SATRAK data and TLE data do not necessarily match. Based on this, the modified code may give more accurate results than SATRAK. If the new code contains more modeled perturbations than SATRAK, this would also explain the periodic variations seen in most of the preceding figures.

5.6.2 CP4

Table 3 - CP4 TLE Data

Date of Reading	TLE
4/22/09	1 31132U 07012Q 09112.17610138 .00000080 00000-0 27710-4 0 5883 2 31132 97.9989 171.5508 0086862 87.9115 273.2019 14.55199299106882
5/1/09	1 31132U 07012Q 09120.15222967 .00000158 00000-0 45009-4 0 5943 2 31132 97.9986 179.1896 0086768 63.3890 297.6162 14.55201137108040
5/23/09	1 31132U 07012Q 09143.18655556 -.00000231 00000-0 -40476-4 0 6101 2 31132 097.9973 201.2422 0085319 352.7051 007.2899 14.55204047111390

Table 4 - CP4 Results

Days since 4/22 TLE	Error in Position (km)	Error in Velocity (km/sec)
7.98	0.5924	0.000514
31.01	12.253	0.0115

The CP4 results are similar to the CP3 results. The error in both position and velocity are of similar magnitude after similar amounts of time. From this, a general rate of error propagation can be assessed.

5.7 ISS Tool bag

The ISS Tool bag was a piece of debris dropped by astronauts working on the outside of the International Space Station. The ISS is in orbit at approximately 350 km altitude.

The ISS Tool bag is currently in orbit at approximately 300 km altitude due to drag perturbations and lack of station keeping.

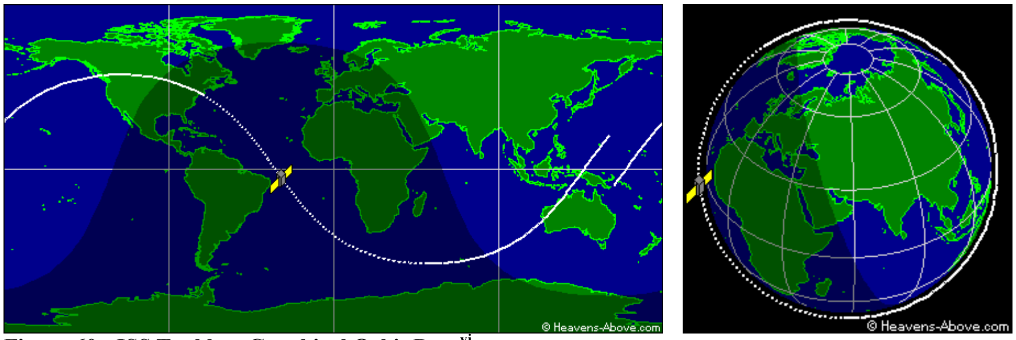


Figure 60 - ISS Tool bag Graphical Orbit Data[vi]

The following TLE's were downloaded from Celestrak.

Table 5 - ISS Tool bag TLE Data

Date of Reading	TLE
5/1/09	1 33442U 98067BL 09120.27106401 .00062601 00000-0 24248-3 0 2271 2 33442 51.6371 184.2223 0003583 339.3988 20.6682 15.86587612 25511
5/6/09	1 33442U 98067BL 09125.49770747 .00067504 00000-0 25238-3 0 2356 2 33442 51.6390 156.8544 0001434 341.7735 18.3083 15.87287894 26344
5/23/09	1 33442U 98067BL 09143.17121749 .00091180 14199-4 29936-3 0 2571 2 33442 51.6360 64.0961 0002463 89.7331 270.1497 15.90545869 29150

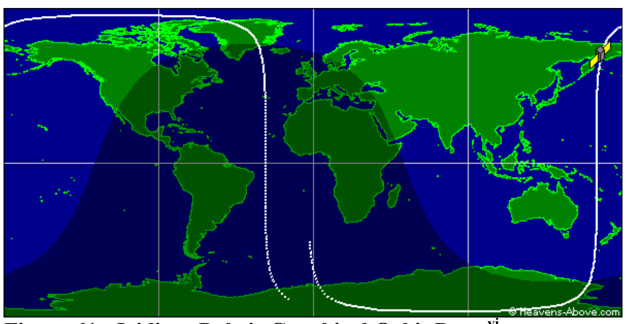
Table 6 - ISS Tool bag Results

Days since 4/22 TLE	Error in Position (km)	Error in Velocity (km/sec)
5.23	33.645	0.0388
22.90	3041.2	3.519

The SGP4 propagator breaks down quickly when dealing with extremely low orbits. After 5 days, the error in position has already become significant and after 23 days, the projection is useless. The error in the projection is caused by an ever changing B* term in the TLE, or the term used to approximate drag. The force of drag is changing constantly due to the object falling further and further into the atmosphere. Propagation for LEO objects should be limited to short periods of time using the new code.

5.8 Iridium Debris

Iridium Debris is a piece of the Iridium 33 satellite that broke up in LEO. The altitude of the debris is approximately 775 km, putting it in a similar orbit to the Polysats.



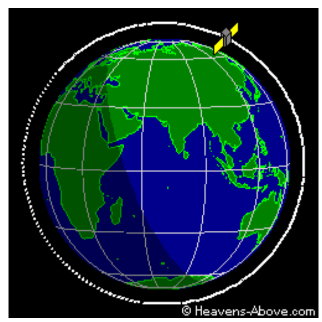


Figure 61 - Iridium Debris Graphical Orbit Data [vi]

Table 7 - Iridium Debris TLE Data

Date of Reading	TLE
4/22/09	1 33771U 97051J 09111.44856353 .00002894 00000-0 10166-2 0 511 2 33771 86.4092 92.3272 0016638 106.1355 254.1660 14.34588702 9984
5/1/09	1 33771U 97051J 09120.30665597 .00004031 00000-0 14163-2 0 587 2 33771 86.4081 88.6364 0017979 83.2643 277.0615 14.34660268 11251
5/23/09	1 33771U 97051J 09142.83354878 .00003334 00000-0 11665-2 0 776 2 33771 086.4045 079.2409 0018814 026.0516 334.1596 14.34826651 14481

Table 8 - Iridium Results

Days since 4/22 TLE	Error in Position (km)	Error in Velocity (km/sec)
8.59	41.315	0.0444
31.39	470.04	0.4969

5.9 Operational Issues

There are two main issues to consider when investigating whether or not to switch to the new system. First, how will the improvements in the code affect the program effectiveness? Second, how will the additions to the code affect the speed of operation?

5.9.1 Program Effectiveness

Working with the assumption that the original JSC code provides 'good enough' answers, the following estimates how long the JSC code could successfully track the satellite. If this range of time falls within the scope of program operations, then JSC would not be able to differentiate between the old and new SGP4 versions.

To calculate this estimation, there were some simplifying assumption made. First, NASA uses a telescope with a 2 degree square FOV to initially find the object. It then uses the PREDICT software to try to find the same object with a 0.2 degree square FOV telescope. For this estimation, a 0.2 degree round FOV was assumed. This means that the threshold for the angle away from the position truth vector is half of this, or 0.1 degrees. Second, the angle from the center of the Earth was calculated rather than from some point on the surface of the Earth. This assumption has both the ability to help or harm the prediction based on the vector error and where the telescope actually is, but should give a reasonable answer. Third, since SATRAK data was not available for all the points needed, the MATLAB data was used as truth. Earlier tests showed that MATLAB provided accurate outputs consistently.

Below are graphs of the three objects defined by NASA JSC showing the angle at which the object is from where the old SGP4 program tells the telescope to point. A threshold line of 0.1 degrees is also drawn for a point of comparison.

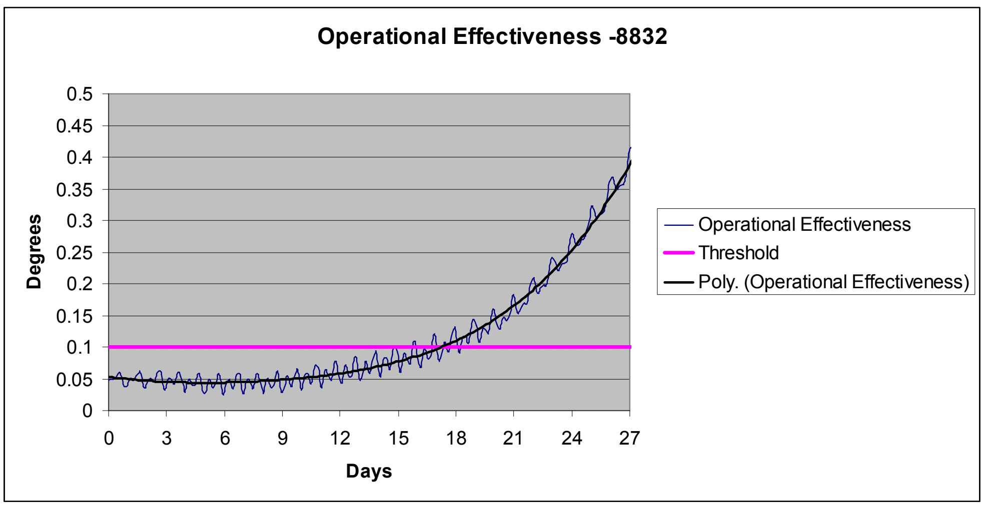


Figure 62 - 8832 Operational Effectiveness

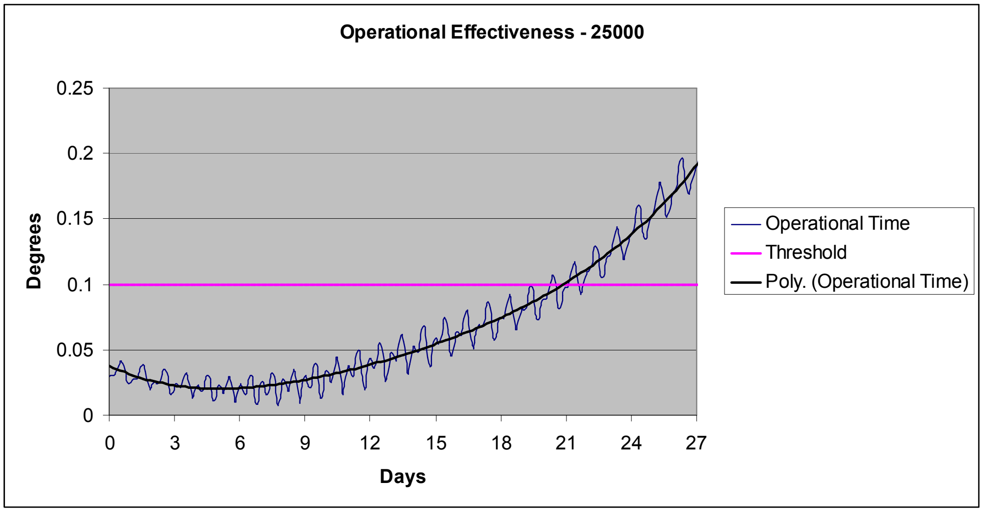


Figure 63 - 25000 Operational Effectiveness

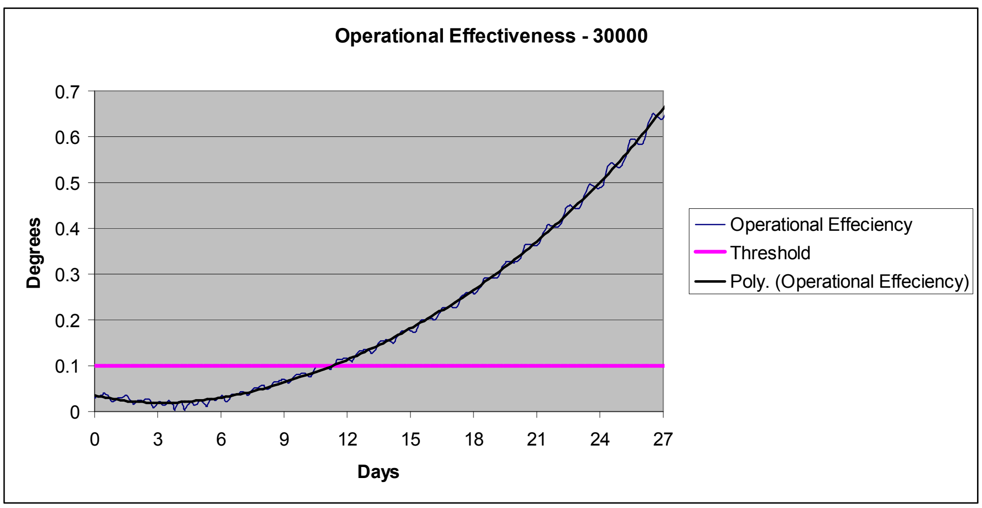


Figure 64 - 30000 Operational Effectiveness

All three cases show that the existing SGP4 propagator breaks down and is no longer effective after a matter of days. From these limited examples, it takes 12 days for the first object to leave the FOV. Furthermore, these examples show that the propagator is most effective between three and nine days. What is troubling about these graphs, mentioned briefly earlier, is the fact that at time zero, the telescope will not be centered on the target. Again, it is believed this to be caused by a flaw in programming Kepler's solution. The above graphs can be compared to the following:

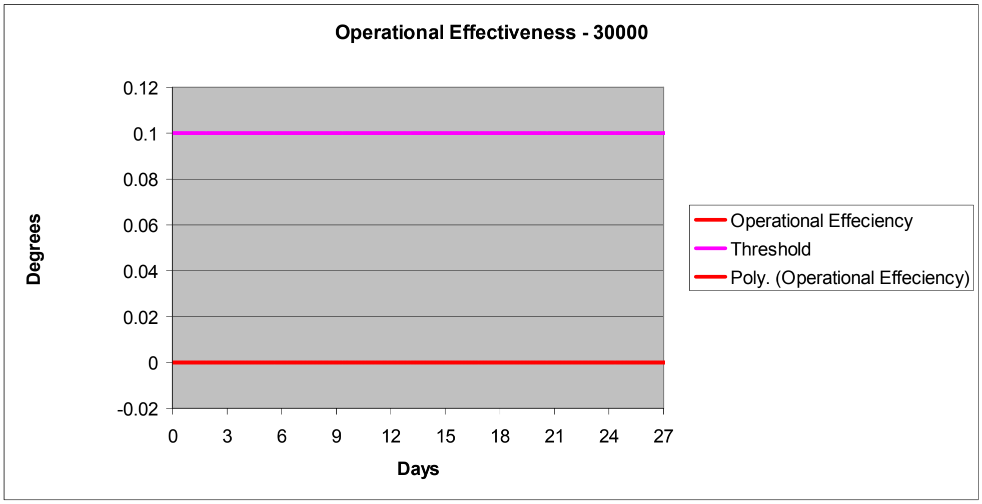


Figure 65 - 30000 Operational Effectiveness (new code)

The error with the new code is essentially zero throughout the entirety of the simulation and is difficult to read in the above chart. Object 30000 offered the poorest results and thus was chosen as a point of comparison. The new code offers significant improvement over the old code and will stay within threshold limits indefinitely. Additionally, even in short term situations, the new code offers superior accuracy.

Below is the operational effectiveness of the new code in relation to Polysat 3 (CP3):

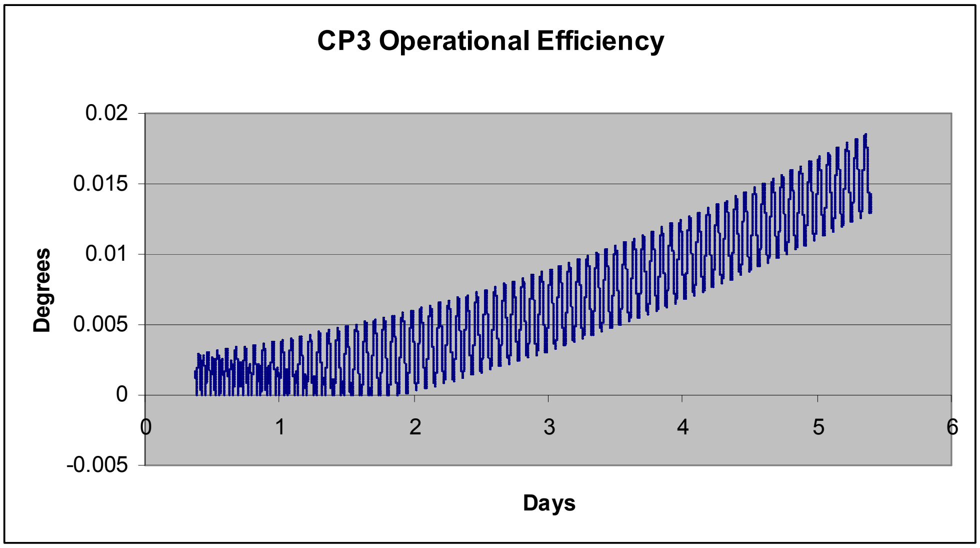


Figure 66 - CP3 Operational Effectiveness

Though the values only extend to roughly 5 days, extrapolating the data using a quadratic suggests that the modified code will be able to keep the object in the FOV for roughly 15 days. Using the TLE truth data points to a different model. Using data between 4/22/09 and 5/23/2009 which is roughly 31 days, shows that the object will be about .071 degrees away from the center of the FOV – still within threshold limits. This shows that the growth in error is linear instead of polynomial, suggesting that the modified code will be able to keep the object in the FOV for roughly 40 days.

5.9.2 Operational Speed

Comparing lines of code, the new program contains 4475 lines with comments and blank lines, while the existing code only contains 2510 lines. No noticeable difference was encountered when running the two programs.

6. Conclusion

The preceding results lead to the following conclusions:

In reference to the original question of whether or not NASA JSC should change their system to incorporate the new Vallado code, the answer is yes. This should be done for two distinct reasons. First, the modified code is more accurate by orders of magnitude over a much longer period of time. Second, the old code provides flawed answers off by kilometers even at time zero. Though the old code provides 'good enough' answers for periods up to about one week, this assumes that the input data is accurate. In reality, the nature of the input data may contain significant errors as well. This is because the code uses right ascension and declination data separated by brief periods of time (less than 5 minutes), thus the object being followed does not move far and the right ascension and declination change only slightly. Calculating COES from this data is limited to the accuracy and the significant figures provided by the telescope data. In such, the 'good enough' answers provided by the old code may not be 'good enough' when compounded with errors in telescope readings.

From the analysis, it is apparent that Vallado's code does not exactly match the output of SATRAK, though it is close. The periodic nature of the error between the two codes, which mimics the period of the orbiting object around the Earth, implies that there are perturbations or constants that differ between the two programs. For this analysis, SATRAK data is taken as truth; however, the error in the SATRAK program is unknown. It is possible that Vallado better predicts the position and velocity of orbiting objects. For the purposes of this project, the two outputs are statistically identical.

If using the Vallado code to replace SATRAK, the MATLAB version of the code is more reliable than the FORTRAN version. Even though the two programs yield indistinguishable results, the MATLAB version of the code is more intuitive and easier to follow. Furthermore, FORTRAN proved more difficult to use as the program balked at times for unknown reasons. Both programs have been provided for use on the attached CD with a User's Guide in section 7.

Finally, the modified code has limitations based on the altitude of the satellite. For objects in deep space orbits such as GEO, the code accurately predicts (to the same level as SATRAK) the position and the velocity of the object within threshold limits indefinitely. In MEO, the effectiveness drops to roughly two weeks of predictions within the threshold limits. In LEO, the effectiveness drops to about one day of predictions within the threshold limits. Thus, the new code will work for all orbiting objects; however, it is important to keep in mind the operational timeline of activities. Furthermore, for LEO and MEO objects, the B* term needs to be added to the system, or else the accuracy of the results may decrease. Improvements to the PREDICT code must be made to accommodate this before use.

7. User's Guide

This section provides instructions on how to install and run the three different codes discussed in this paper. Sections 7.1 and 7.2 describe the SATRAK replacement codes written in MATLAB and FORTRAN. Section 7.3 describes how to incorporated the modified code into the existing NASA JSC PREDICT code as well as the various options the user has within the code itself.

7.1 MATLAB

Note that the output file will be over-written each time the program is run unless the filename is changed.

The MATLAB code mimics the SATRAK code and outputs very similar results. Work was done using MATLAB 7.0.

7.1.1 Installation

On the attached CD, there is a folder named SGP4 MATLAB. Copy and paste this folder to MATLAB's current directory. The folder contains the following files:

- testmat.m — Driver script for testing and example usage
- sgp4.m — Main sgp4 routine
- sgp4init.m — Initialization routine for sgp4
- initl.m — Initialization for sgp4
- dsinit.m — Deep space initialization

- dspace.m Deep space perturbations
- dpper.m Deep Space periodics
- dscom.m Deep Space common variables
- twoline2rv.m TLE conversion routine
- angl.m Find the angle between two vectors
- constmath.m set mathematical constants
- days2mdh.m convert days to month day hour minute second
- getgravc.m Get the gravity constants
- gstime.m Find Greenwich sidereal time
- invjday.m Inverse Julian Date
- jday.m Find the Julian Date
- mag.m Magnitude of a vector
- newtonnnu.m Kepler's iteration given eccentricity and true anomaly
- rv2coe.m Convert position and velocity vectors to classical orbital elements
- SGP4-VER.TLE Verification file
- Sgp4_CodeReadme.pdf

7.1.2 Verification Run

1. Run testmat.m
2. Enter opsmode 'i' for improved method
3. Enter typerun 'v' for verification mode

4. Enter constant ‘72’ corresponding to WGS-72 standards
5. Enter file name ‘SGP4-VER.TLE’ – see appendix 1 for file information
6. The program should output a file called ‘tmatver.out’ can be compared to the results found in appendix 2 for verification that the program is running properly.

7.1.3 TLE inputs

The program does not discriminate against file types when opening TLEs, though it has only been tested with .TXT and .TLE file extensions. Make sure that the input TLE does have proper the proper format. A simple way to load a TLE is to open a new script in MATLAB and copy and paste the desired TLE, then save the script with a .TLE file extension.

The program is capable of handling multiple TLEs in a single run and will output the file with a header representing the satellite number before each set of data. When formatting a multiple TLE input, use a ‘#’ symbol to begin each line that does not contain TLE data. This tells the computer to skip the line and move to the next line.

7.1.4 Catalog Mode

Catalog mode takes a TLE input and outputs data for -24 hours to +24 hours with 20 minute intervals. This is useful for short term propagations of many objects (i.e. multiple TLEs).

1. Run ‘testmat.m’

2. Enter ‘a’ for afspc or ‘i’ for improved method
3. For type of run, enter ‘c’ for catalog mode
4. Enter TLE data – file must be completely written with the file extension name
5. A new file should be created with the name ‘tmatall.out’ with the catalog data.

7.1.5 Manual Mode

Manual mode gives the user the opportunity to define when to start and stop calculating and at what time intervals. To do this, the user is given three options to define time:

- Minutes from epoch
- Epoch
- Day of Year

The minutes from epoch approach assumes that the epoch is defined by the time stated within the TLE. Simply input (in minutes) how much time you want to elapse before the first data point, and how much time you want to elapse before the last data point. The epoch approach asks for a start date and time and a stop date and time. Finally, the day of year approach asks for a year and a day of the year. The day of the year can be in decimal form to indicate a fraction of a day. For all three approaches, the reference time frame is UTC. Furthermore, whatever mode chosen will prompt for the time step that will be used (in minutes).

Note, when working with multiple TLEs in manual mode, the user will be prompted for start and stop points and time step for each individual TLE.

1. run ‘testmat.m’
2. Enter ‘a’ for afspc or ‘i’ for improved mehod
3. For type of run, enter ‘m’ for manual operation
4. Enter how the start and stop points are determined
    a. m = minutes from epoch
    b. e = epoch (year, month, day, hour, min, sec)
    c. d = day of year (year, day)
5. Enter constant choice (72 gives answers closest to truth)
6. Enter TLE filename
7. Follow prompts on screen to input start and stop points and time step
8. Program sends output to file named ‘tmat.out’

7.1.6 Understanding the Output File

The output comes without column headers, so it is important to know what the values represent. The first line is a header line with information about the satellite number. When working with multiple TLEs, data from each TLE is separated by a similar header with the satellite number. For each satellite, there are 7-13 columns of data (columns 8-13 are date and time data which is not calculated for catalog mode):

Column 1 Minutes from Epoch

Column 2 X-Position

Column 3 Y-Position

Column 4	Z-Position
Column 5	X-Velocity
Column 6	Y-Velocity
Column 7	Z-Velocity
Column 8	Year
Column 9	Month
Column 10	Day
Column 11	Hour
Column 12	Minute
Column 13	Second

Note, the position vector is based in ECI coordinate frame.

7.2 FORTRAN (Vallado Code)

The main FORTRAN program has many similarities with the MATLAB program and may reference certain sections.

Note that the output file will be over-written each time the program is run unless the filename is changed.

7.2.1 Installation

On the attached CD, there is a folder labeled SGP4 FORTRAN. Copy and paste the folder to a working directory. The files contained in the folder are:

ASTMATH.CMN	Common file with mathematical constants
SGP4.CMN	Common file with constants applicable to SGP4 model
TESTFOR.FOR	Fixed Format FORTRAN file
TESTFOR.EXE	Executable file of TESTFOR.FOR
Validation.OUT	Data to validate correct installation
SGP4-VER.TLE	Validation TLE Data

Either use the executable (compiled and built on Silverfrost), or compile and build the FORTRAN file using another compiler. Correct installation should result in the program running in a command prompt.

7.2.2 Verification Run

** FORTRAN is case sensitive, please keep this in mind when entering text.**

1. Run TESTFOR.EXE
2. Select opsmode ‘i’
3. Select run type ‘v’
4. Input filename ‘SGP4-VER.TLE’
5. Program will output a file named ‘tfor.out’ to the working directory

6. Compare this output with data from the file labeled ‘Verification.OUT’

7.2.3 TLE Input

This function is identical to MATLAB. See section 7.1.3.

7.2.4 Catalog Mode

Catalog mode takes a TLE input and outputs data for -24 hours to +24 hours with 10 minute intervals. This is useful for short term propagations of many objects (i.e. multiple TLEs).

1. Run ‘TESTFOR.EXE – This should bring up a command prompt
2. Enter ‘a’ for afspc or ‘i’ for improved method
3. For type of run, enter ‘c’ for catalog mode
4. Enter TLE filename– file must be completely written with the file extension name
5. A new file should be created with the name ‘tfor.out’ with the catalog data.

7.2.5 Manual Mode

Manual mode gives the user the opportunity to define when to start and stop calculating and at what time intervals. To do this, the user is given three options to define time:

- Minutes from epoch
- Epoch
- Day of Year

The minutes from epoch approach assumes that the epoch is defined by the time stated within the TLE. Simply input (in minutes) how much time you want to elapse before the first data point, and how much time you want to elapse before the last data point. The epoch approach asks for a start date and time and a stop date and time. Finally, the day of year approach asks for a year and a day of the year. The day of the year can be in decimal form to indicate a fraction of a day. For all three approaches, the reference time frame is UTC. Furthermore, whatever mode chosen will prompt for the time step that will be used (in minutes).

Note, when working with multiple TLEs in manual mode, the user will be prompted for start and stop points and time step for each individual TLE.

9. run 'TESTFOR.EXE' – a new command prompt should appear
10. Enter 'a' for afspc or 'i' for improved method
11. For type of run, enter 'm' for manual operation
12. Enter how the start and stop points are determined (Case sensitive)
    a. M = minutes from epoch
    b. E = epoch (year, month, day, hour, min, sec)
    c. D = day of year (year, day)
13. Enter constant choice (72 gives answers closest to truth)
14. Enter TLE filename
15. Follow prompts on screen to input start and stop points and time step

16. Program sends output to file named ‘tfor.out’

7.2.6 Understanding the Output File

The output file created is in the same format at the MATLAB output. Please see section 7.1.6 for more details.

7.3 Modified FORTRAN Code – For JSC

The modified FORTRAN code is designed to be compatible with the current PREDICT program; however, epoch data needs to be passed to the new program. This section does not describe how to run the PREDICT code. Rather, it explains how to integrate the modified code with the existing code.

7.3.1 Installation

On the attached CD in the folder ‘MODIFIED CODE’, there is a file called:

SGP4_Classic Subroutine.txt

Open this file and copy all of the contents. Next in the PREDICT source code, locate the SGP4_Classic subroutine (line 853 – line 1116 in Feb. version). Delete the entire subroutine and replace it with the text copied above.

To solve the epoch data problem, a global file called /global2/ is created in the surveychasenew subroutine. In the line following the definition of /global1/ (or line 4314 after addition of the new code) add the following line (□ represents a space):

□□□□□□COMMON /global2/ DATE, UT

```
cccccccccccccccccccccccccccccccc
C   SUBROUTINE SURVEYCHASENEW
cccccccccccccccccccccccccccccccc
c
c      esb 3/21/09 changed the a58 formats back to a80 to match the original data formats
c        for xline, cline, testline
c         this will allow additional data to be in the original data files.
c
C      Last Change: 16 April 2008
C      KJA 16 April, made the propagation to start at 2200 UTday -1 and for 24 hours
c
C      KJA 13 Mar 2007
C      Changed the prop time to be rounded up to the minute, changed inputs

C      KJA 1 Mar 2007
C      New formats to RA,DEC, and MAG

C      This program will be used to build the prediction files for the
C      survey and chance program

C      Inputs need to be one set of observations with the
C      YYYYDOY.obj# line at the top and the observations following it

       SUBROUTINE SURVEYCHASENEW
       IMPLICIT NONE
       COMMON /global1/ PROPNAME, tele, FILEIN
       COMMON /global2/ DATE, UT

       INTEGER YYYY,JDOY,NDET,MONTE,i
       INTEGER TIMEhrs, TIMEmins, JDET,DAY,HOUR

       DOUBLE PRECISION TIME, MINS,HOURb4mid,newUTb4mid, newUT
       DOUBLE PRECISION HOURnew, MINsnew, UT
       CHARACTER*20 FILEOUT,FILETEMP,FILEOUT2
       CHARACTER*80 XLINE,CLINE,TESTLINE
       CHARACTER*12 XXX
       CHARACTER*34 SHORT
       CHARACTER*10 DATE
       CHARACTER*8  BEGDATE,CHECKHOUR
       CHARACTER*51 CHARSTRING2
       CHARACTER*52 CHARSTRING1
       CHARACTER*20 PROPNAME
       CHARACTER*2 TELE
```
Figure 67 - Code Addition to SURVEYCHASENEW

Next, in the same directory on the CD, there are two common files entitled:

- SGP4.CMN
- ASTMATH.CMN

Copy and paste these files into the directory with the PREDICT source code, as the new SGP4 propagator references these files.

Now, the updated PREDICT code will be ready to be compiled.

7.3.2 Code Options

The Vallado code contains various options that have been pre-programmed for the Orbital Debris Program Office use; however, these can changed if necessary.

7.3.2.1 Constant Values

The program is set to use World Geodetic System (WGS) 72 parameters for Earth defined constants. These parameters are legacy values from the first code release of SGP4, and best match SATRAK truth data. The Vallado code is programmed to allow an update to the WGS 84 parameters, which are considered up-to-date until 2010[vii]. To make this switch, find where 'whichconst' is defined in the SGP4_Classic subroutine (~line 925) and change the value from 72 to 84.

```
        Real*8 ro(3),vo(3)

* ----------------------------  Locals  -------------------------------
        REAL*8 J2,TwoPi,Rad,mu, RadiusEarthKm, xke,
     &         de2ra, xpdotp, T, sec, JD, pi, j3, j4, j3oj2, tumin
        INTEGER EYr, EDay, EMon, EHr, EMin

        INCLUDE 'SGP4.CMN'
        COMMON /global2/ Date, UT


* ------------------------  Implementation   --------------------------

        Opsmode = 'i'
        typerun = 'm'
        typeinput = 'M'

        whichconst = 72

        pi              =     4.0D0 * datan(1.0D0)   ! 3.14159265358979D0
        TwoPi           =     2.0D0 * pi     ! 6.28318530717959D0
        Rad             =   180.0D0 / pi     ! 57.29577951308230D0
        DE2RA           =     pi / 180.0D0   ! 0.01745329251994330D0
        xpdotp          =  1440.0 / (2.0 *pi)   ! 229.1831180523293D0
```

Figure 68 - Changing WGS values (1)

If manual changes are necessary to the constants, find the subroutine 'getgravconst' (~line 1012). The constants for specific called cases are listed in various IF statements.

The user can define radius of Earth, J2, J3, and J4 harmonics for either the WGS 72 or WGS 84 system.

```
          j4      =  -0.00000165597D0
          j3oj2   =   j3 / j2
        ENDIF
      if (whichconst.eq.72) THEN
          ! ------------ wgs-72 constants ------------
          mu      = 398600.8D0            ! in km3 / s2
          radiusearthkm = 6378.135D0      ! km
          xke     = 60.0D0 / dsqrt(radiusearthkm**3/mu)
          tumin   = 1.0D0 / xke
          j2      =    0.001082616D0
          j3      =  -0.00000253881D0
          j4      =  -0.00000165597D0
          j3oj2   =   j3 / j2
        ENDIF
      if (whichconst.eq.84) THEN
          ! ------------ wgs-84 constants ------------
          mu      = 398600.5D0            ! in km3 / s2
          radiusearthkm = 6378.137D0      ! km
          xke     = 60.0D0 / dsqrt(radiusearthkm**3/mu)
          tumin   = 1.0D0 / xke
          j2      =    0.00108262998905D0
          j3      =  -0.00000253215306D0
          j4      =  -0.00000161098761D0
          j3oj2   =   j3 / j2
        ENDIF
```

Figure 69 - Changing WGS values (2)

7.3.2.2 B*

The B* term is used to estimate the drag force felt by the object in orbit. For the Orbital Debris Program Office, this value has been set to zero since B* is not calculated by the PREDICT program and the objects in GEO do not experience significant drag forces. There are two ways to change the default value of B*. First, the user can define it as something other than zero inside the program. To do this, find where B* is defined in the TwoLine2RVSGP4 subroutine (~line 1157). Change the 0.0d0 value to desired B* value. Second, the user can comment the above line out completely and uncomment the two lines above. Doing this will prompt the user to input a B* term. This may cause some problems in running the PREDICT code since it will prompt the user during every iteration of the SGP4 program. To solve this, the user can comment out all three lines and define the B* term in the main program. The term can then be accessed by the subroutine either by a global variable or by having the B* variable be called by the subroutine itself.

```
        Code = 0

* ---------------------- CONVERT TO INTERNAL UNITS --------------------
* ---- RADIANS, DISTANCE IN EARTH RADII, AND VELOCITY IN ER/KEMIN) ----

        NDot    = MMDOTO2 / (XPDOTP*1440)
        NDDot   = MMDDOTO6 / (XPDOTP*1440*1440)

c added the possibility to put in a Bstar term.  If this is unnecessary
c comment out the next two lines and uncomment the third line.

        !Write(*,*) 'Input Bstar term: '
        !read(*,*) Bstar
        Bstar = 0.0d0

c Changed initial element set to match NASA input
        No      = MMO / XPDOTP
        a       = (No*TUMin)**(-2.0D0/3.0D0)
        Inclo   = INCDEGO  * Deg2Rad
        nodeo   = RAANDEGO * Deg2Rad
        Argpo   = APDEGO * Deg2Rad
        Mo      = MADEGO* Deg2Rad

        IF (DABS(ECCO-1.0D0) .gt. 0.000001D0) THEN
            Altp= (a*(1.0D0-ECCO))-1.0D0
            Alta= (a*(1.0D0+ECCO))-1.0D0
          ELSE
```

Figure 70 - Changing the B* term

BIBLIOGRAPHY

1. AIAA-2006-6753.zip 700 KB, Feb 16, 2009. , < http://www.centerforspace.com/downloads/>

2. Chobotov, Vladimir A. Orbital Mechanics, Second Edition. Reston, VA, AIAA, Inc., 1996.

3. Hoots, Felix R., et al. “History of Analytical Orbit Modeling in the U.S. Space Surveillance System.” Journal of Guidance Control, and Dynamics March-April 2004: 174-185.

4. Hoots, Felix R. and Roehrich, Ronal L. “Models for Propagation of NORAD Element Sets.” SPACETRACK REPORT NO.3 December 1980: Packaged compiled by Kelso, T.S.: December 31, 1988. < ftp://ftp.amsat.org/amsat/docs/spacetrk.pdf>

5. Kelso, T.S. “TLE Data”, 2008-2009. < http://celestrak.com/>

6. Meissner, Loren P. Fortran 90. Boston, MA, PWS Publishing Co., 1995

7. “Selected Orbital Data” 2009. <http://www.heavens-above.com>

8. Vallado, David A. Fundamentals of Astrodynamics and Applications, Third Edition. Hawthorne CA, Microcosm Press, 2007.

9. Vallado, David A., et al. "Revisiting Spacetrack Report #3." 2006. < http://celestrak.com/publications/AIAA/2006-6753/AIAA-2006-6753.pdf>

10. "World Geodetic System." Wikipedia May 11, 2009. < http://en.wikipedia.org/wiki/WGS84>

APPENDIX A. SGP4 Code

All working codes are included in the accompanying CD. The FORTRAN code designed to replace the legacy SGP4 propagator within PREDICT is displayed below for reference.

A.1 Modified FORTRAN Code (Vallado)

This code compiles and runs on LAHEY express 7.2. Output files are generated; however, when testing the code with PREDICT_Feb09, there were issues with rewriting a file at the end of the program. There is no evidence that this is linked with the new SGP4 propagator.

```
*********************************************************************
*
*           Subroutine SGP_Classic
*
* code modified by Nicholas Miura
* Cal Poly - San Luis Obispo
* nicholas.miura@gmail.com
*
* subroutine should interface with existing NASA predictor code
*
* Inputs
*  DTDAYS = Time since final observation in Days
*  MMO = Mean motion at final observation
*  ECCO = Eccentricity at final observation
*  INCDEGO = Inclination at final observation
*  APDEGO = Argument of Perigee (Degrees)at observation
*  RAANDEGO = Right Ascension of Ascending Node (Degrees)at observation
*  MADEGO = Mean Anomaly (Degrees) at observation
*  MMDOTO2 = First time derivative
*  MMDDOTO6 = Second time derivative
*
* Outputs
*  POS = Position of Spacecraft
*  VEL = Velocity of Spacecraft
*  UP = Is spacecraft still in orbit?
*
* Called Subroutines
```

```
*  getgravconst


     Subroutine SGP_Classic(DTDAYS,MM0,ECC0,
    &  INCDEG0,APDEG0,RAANDEG0,MADEG0,
    &  MMDOTO2,MMDDOTO6,POS,VEL,UP)

      IMPLICIT NONE



c these declarations are copied from NASA Predictor code
c and define the input and output of the subroutine

                REAL*8 DTDAYS,MM0,ECC0
      REAL*8 INCDEG0,APDEG0,RAANDEG0,MADEG0
      REAL*8 MMDOTO2,MMDDOTO6,EHour, ESec
      Real*8 JDSatEpoch2
      REAL*8 POS(1:3),VEL(1:3)
         LOGICAL UP
      DOUBLE PRECISION UT

c these declarations are used in the code

                Character typerun, typeinput
      Character*58 Longstring
      Character*10 Date

      Integer Code, NumSats, error, whichconst
      Real*8 ro(3),vo(3)

* ---------------------------- Locals -------------------------------
      REAL*8 J2,TwoPi,Rad,mu, RadiusEarthKm, xke,
    &        de2ra, xpdotp, T, sec, JD, pi, j3, j4, j3oj2, tumin
      INTEGER EYr, EDay, EMon, EHr, EMin

      INCLUDE 'SGP4.CMN'
      COMMON /global2/ Date, UT


* ----------------------- Implementation -------------------------

                Opsmode = 'i'
      typerun = 'm'
      typeinput = 'M'
```

```
                 whichconst = 72

      pi          =   4.0D0 * datan(1.0D0)  ! 3.14159265358979D0
      TwoPi       =   2.0D0 * pi   ! 6.28318530717959D0
      Rad         =   180.0D0 / pi  ! 57.29577951308230D0
      DE2RA       =   pi / 180.0D0  ! 0.01745329251994330D0
      xpdotp      =  1440.0 / (2.0 *pi)  ! 229.1831180523293D0

c sgp4fix identify constants and allow alternate values
      CALL getgravconst( whichconst, tumin, mu, radiusearthkm, xke,
     &      j2, j3, j4, j3oj2 )


c Read Date and Time

           READ (Date(1:4),FMT='(I4)') EYr
      READ (Date(6:7),FMT='(I2)') EMon
      READ (Date(9:10),FMT='(I2)') EDay

      CALL HMS (UT, EHr, EMin, ESec)
      CALL JDAY1 (EYr, EMon, EDay, EHr ,EMin, ESec, JDSatEpoch2)


c Open a temporary file containing initialized orbital data
c Called by other subroutines

c       OPEN(115,FILE = 'Sgp4Rec.bak', ACCESS = 'DIRECT',
c     &         FORM = 'UNFORMATTED', RECL = 1100, STATUS = 'REPLACE' )

      NumSats = 1

c  bring orbital elements to the new two line element converter - trick the system
      CALL TwoLine2RVSGP4 ( NumSats,typerun,typeinput,whichconst,
     &                Code, DTDAYS,MM0,ECC0,
     &                INCDEG0,APDEG0,RAANDEG0,MADEG0,
     &                MMDOTO2,MMDDOTO6, JDSatEpoch2, Error )



c  Call the SGP4 propagator based on Time in minutes

      T = DTDAYS*1440
      CALL SGP4 ( whichconst, T, Ro, Vo, Error )

      If (Error .eq. 1)Then
        Up = .FALSE.
```

```
      ENDIF


c the next WRITE line displays the result on the screen
c comment this out if you don't want to test the results

c       WRITE( *,800 ) T, ro(1),ro(2),ro(3),vo(1),vo(2),vo(3)

c  800  FORMAT(F17.8,3F17.8,3(1X,F14.9))


      Pos = Ro
      Vel = Vo

c      CLOSE(115)

c      STOP
      END

* ---------------------------------------------------------------------------
*
*                           function getgravconst
*
*  this function gets constants for the propagator. note that mu is identified to
*    facilitiate comparisons with newer models.
*
*  author        : david vallado                  719-573-2600   21 jul 2006
*
*  inputs        :
*    whichconst  - which set of constants to use  721, 72, 84
*
*  outputs       :
*    tumin       - minutes in one time unit
*    mu          - earth gravitational parameter
*    radiusearthkm - radius of the earth in km
*    xke         - reciprocal of tumin
*    j2, j3, j4  - un-normalized zonal harmonic values
*    j3oj2       - j3 divided by j2
*
*    norad spacetrack report #3
*    vallado, crawford, hujsak, kelso  2006
*  --------------------------------------------------------------------------

      SUBROUTINE getgravconst ( whichconst, tumin, mu,
     &             radiusearthkm, xke, j2, j3, j4, j3oj2 )
      IMPLICIT NONE
```

```
      REAL*8 radiusearthkm, xke, j2, j3, j4, j3oj2, mu, tumin
      INTEGER whichconst

      if (whichconst.eq.721) THEN
        ! -- wgs-72 low precision str#3 constants --
        radiusearthkm = 6378.135D0     ! km
        xke    = 0.0743669161D0
        mu     = 398600.79964D0            ! in km3 / s2
        tumin  = 1.0D0 / xke
        j2     =   0.001082616D0
        j3     =  -0.00000253881D0
        j4     =  -0.00000165597D0
        j3oj2  =  j3 / j2
       ENDIF
      if (whichconst.eq.72) THEN
        ! ------------ wgs-72 constants ------------
        mu     = 398600.8D0            ! in km3 / s2
        radiusearthkm = 6378.135D0     ! km
        xke    = 60.0D0 / dsqrt(radiusearthkm**3/mu)
        tumin  = 1.0D0 / xke
        j2     =   0.001082616D0
        j3     =  -0.00000253881D0
        j4     =  -0.00000165597D0
        j3oj2  =  j3 / j2
       ENDIF
      if (whichconst.eq.84) THEN
        ! ------------ wgs-84 constants ------------
        mu     = 398600.5D0            ! in km3 / s2
        radiusearthkm = 6378.137D0     ! km
        xke    = 60.0D0 / dsqrt(radiusearthkm**3/mu)
        tumin  = 1.0D0 / xke
        j2     =   0.00108262998905D0
        j3     =  -0.00000253215306D0
        j4     =  -0.00000161098761D0
        j3oj2  =  j3 / j2
       ENDIF

      RETURN
      END  !  SUBROUTINE getgravconst

* -----------------------------------------------------------------------------
*
*                           SUBROUTINE TWOLINE2RVSGP4
*
*  this function converts the two line element set character string data to
*    variables and initializes the sgp4 variables. several intermediate variables
```

```
*    and quantities are determined. note that the result is a "structure" so multiple
*    satellites can be processed simultaneously without having to reinitialize. the
*    verification mode is an important option that permits quick checks of any
*    changes to the underlying technical theory. this option works using a
*    modified TLE file in which the start, stop, and delta time values are
*    included at the end of the second line of data. this only works with the
*    verification mode. the catalog mode simply propagates from -1440 to 1440 min
*    from epoch and is useful when performing entire catalog runs.
*
*  author        : david vallado                  719-573-2600    1 mar 2001
*
*  inputs        :
*    Numsats     - Number of satellites processed. It also becomes the record
*                  number for each satellite
*    typerun     - type of run                    verification 'V', catalog 'C',
*                                                 manual 'M'
*    typeinput   - type of manual input           mfe 'M', epoch 'E', dayofyr 'D'
*    whichconst  - which set of constants to use  72, 84
*    opsmode   - type of manual input           afspc 'a', imporved 'i'
*
*  outputs       :
*    Code        - EOF indicator. Code = 999 when EOF reached
*    startmfe    - starttime of simulation,       min from epoch
*    stopmfe     - stoptime of simulation,        min from epoch
*    deltamin    - time step                      min
*
*  coupling      :
*    days2mdhms  - conversion of days to month, day, hour, minute, second
*    jday        - convert day month year hour minute second into Julian date
*    sgp4init    - initialize the sgp4 variables
*
*  Files         :
*    Unit 10     - test.elm        input 2-line element set file
*    Unit 11     - test.bak        output file
*    Unit 15     - sgp4rec.bak     temporary file of record for 2 line element sets
*
*  references    :
*    norad spacetrack report #3
*    vallado, crawford, hujsak, kelso  2006
*------------------------------------------------------------------------------

      SUBROUTINE TwoLine2RVSGP4 ( NumSats, Typerun, typeinput,
     &                         whichconst, Code, DTDAYS,MM0,ECC0,
     &                         INCDEG0,APDEG0,RAANDEG0,MADEG0,
     &                         MMDOTO2,MMDDOTO6, JDSatEpoch1, Error )
```

```
      IMPLICIT NONE
      Character Typerun, typeinput
      Integer Code, NumSats, whichconst
      REAL*8 startmfe, stopmfe, deltamin

* ----------------------------  Locals  -------------------------------
      REAL*8 J2, mu, RadiusEarthKm,VKmPerSec, xke, tumin
      REAL*8 BC,EPDay, sec, xpdotp, j3, j4, j3oj2
      REAL*8 startsec, stopsec, startdayofyr, stopdayofyr, jdstart,
     &        jdstop, JDSatEpoch1
      INTEGER startyear, stopyear, startmon, stopmon, startday,
     &         stopday, starthr, stophr, startmin, stopmin
      INTEGER Yr,Mon,Day,Hr,Minute,  ICrdno,nexp,bexp, error
      CHARACTER Show
      Character*130 LongStr1,LongStr2

      REAL*8 DTDAYS,MM0,ECC0
      REAL*8 INCDEG0,APDEG0,RAANDEG0,MADEG0
      REAL*8 MMDOTO2,MMDDOTO6

      COMMON /DebugHelp/ Help
      CHARACTER Help

      INCLUDE 'SGP4.CMN'
      INCLUDE 'ASTMATH.CMN'

      ! --------------------  Implementation   ----------------------
      Show = 'N'
      xpdotp        =  1440.0D0 / (2.0D0 * pi) ! 229.1831180523293

      CALL getgravconst( whichconst, tumin, mu, radiusearthkm, xke,
     &      j2, j3, j4, j3oj2 );
      VKmPerSec     =  RadiusEarthKm * xke / 60.0D0



* ----------------- READ THE FIRST LINE OF ELEMENT SET ----------------
      Code = 0

* ---------------------- CONVERT TO INTERNAL UNITS --------------------
* ---- RADIANS, DISTANCE IN EARTH RADII, AND VELOCITY IN ER/KEMIN) --
--

      NDot   = MMDOTO2 / (XPDOTP*1440)
      NDDot  = MMDDOTO6 / (XPDOTP*1440*1440)
```

```
c added the possibility to put in a Bstar term.  If this is unnecessary
c comment out the next two lines and uncomment the third line.

        !Write(*,*) 'Input Bstar term: '
        !read(*,*) Bstar
        Bstar = 0.0d0

c Changed initial element set to match NASA input
        No     = MM0 / XPDOTP
        a      = (No*TUMin)**(-2.0D0/3.0D0)
        Inclo  = INCDEG0  * Deg2Rad
        nodeo  = RAANDEG0 * Deg2Rad
        Argpo  = APDEG0 * Deg2Rad
        Mo     = MADEG0* Deg2Rad

        IF (DABS(ECC0-1.0D0) .gt. 0.000001D0) THEN
            Altp= (a*(1.0D0-ECC0))-1.0D0
            Alta= (a*(1.0D0+ECC0))-1.0D0
          ELSE
            Alta= 999999.9D0
            Altp= 2.0D0* (4.0D0/(No*No)**(1.0D0/3.0D0))
          ENDIF

        ! ---- Ballistic Coefficient ----
        IF (DABS(BStar) .gt. 0.00000001D0) THEN
            BC= 1.0D0/(12.741621D0*BStar)
          ELSE
            BC= 1.111111111111111D0
          ENDIF

        ! ---------------------------------------------------------------
        ! find sgp4epoch time of element set
        ! remember that sgp4 uses units of days from 0 jan 1950 (sgp4epoch)
        ! and minutes from the epoch (time)
        ! ---------------------------------------------------------------


* ------------------- MAKE INITIAL PREDICTION AT EPOCH ----------------
        ! 2433281.5 - 2400000.5 = 33281.0, thus time from 1950
        CALL SGP4Init( whichconst,
     &          SatNum,BStar, ECC0, JDSatEpoch1-2433281.5D0,
     &          Argpo,Inclo,Mo,No, nodeo, Error )

        IF(Error .GT. 0) THEN
            WRITE( *,*) '# *** SGP4 Model Error ***',Error
          ENDIF
```

Formatted: German (Germany)

```
c      write tle output details
c      INCLUDE 'debug8.for'

       ! ---- Fix to indicate end-of-file
       GOTO 1000
  999   Code = 999
 1000   CONTINUE

      RETURN
      END  !      SUBROUTINE TwoLine2RVSGP4

* -----------------------------------------------------------------------------
*
*                           SUBROUTINE SGP4INIT
*
*  This subroutine initializes variables for SGP4.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    satn        - satellite number
*    bstar       - sgp4 type drag coefficient              kg/m2er
*    ecco        - eccentricity
*    epoch       - epoch time in days from jan 0, 1950. 0 hr
*    argpo       - argument of perigee (output if ds)
*    inclo       - inclination
*    mo          - mean anomaly (output if ds)
*    no          - mean motion
*    nodeo       - right ascension of ascending node
*
*  outputs       :
*    satrec      - common block values for subsequent calls
*    return code - non-zero on error.
*                   1 - mean elements, ecc >= 1.0 or ecc < -0.001 or a < 0.95 er
*                   2 - mean motion less than 0.0
*                   3 - pert elements, ecc < 0.0  or  ecc > 1.0
*                   4 - semi-latus rectum < 0.0
*                   5 - epoch elements are sub-orbital
*                   6 - satellite has decayed
*
*  coupling      :
*    getgravconst-
*    initl       -
*    dscom       -
*    dpper       -
```

```
*    dsinit      -
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
* ------------------------------------------------------------------------ }

      SUBROUTINE SGP4Init ( whichconst,
     &                      Satn,   xBStar, xEcco,  Epoch, xArgpo,
     &                      xInclo, xMo, xNo, xnodeo, Error )
        IMPLICIT NONE
        INTEGER Satn, error, whichconst
        REAL*8  xBStar, xEcco, Epoch, xArgpo, xInclo, xMo, xNo, xnodeo
        REAL*8 T, r(3), v(3)

        INCLUDE 'SGP4.CMN'

        !COMMON /DebugHelp/ Help ! removed by NM 4/16
        CHARACTER Help

* -------------------------- Local Variables --------------------------

        REAL*8  Ao,ainv,con42,cosio,sinio,cosio2,Eccsq,omeosq,
     &          posq,rp,rteosq, CNODM , SNODM , COSIM , SINIM , COSOMM,
     &          SINOMM, Cc1sq ,
     &          Cc2   , Cc3   , Coef  , Coef1 , Cosio4, DAY   , Dndt  ,
     &          Eccm  , EMSQ  , Eeta  , Etasq , GAM   , Argpm , nodem,
     &          Inclm , Mm  , Xn    , Perige, Pinvsq, Psisq , Qzms24,
     &          RTEMSQ, S1    , S2    , S3    , S4    , S5    , S6    ,
     &          S7    , SFour , SS1   , SS2   , SS3   , SS4   , SS5   ,
     &          SS6   , SS7   , SZ1   , SZ2   , SZ3   , SZ11  , SZ12  ,
     &          SZ13  , SZ21  , SZ22  , SZ23  , SZ31  , SZ32  , SZ33  ,
     &          Tc    , Temp  , Temp1 , Temp2 , Temp3 , Tsi   , XPIDOT,
     &          Xhdot1, Z1    , Z2    , Z3    , Z11   , Z12   , Z13   ,
     &          Z21   , Z22   , Z23   , Z31   , Z32   , Z33
        REAL*8  qzms2t, SS, mu, RadiusEarthKm, J2, j3oJ2,J4,X2o3,
     &          temp4, j3, xke, tumin
        INCLUDE 'ASTMATH.CMN'

* ---------------------------- INITIALIZATION -------------------------
        method = 'n'

c       clear sgp4 flag
        Error = 0
```

```
c      sgp4fix - note the following variables are also passed directly via sgp4 common.
c      it is possible to streamline the sgp4init call by deleting the "x"
c      variables, but the user would need to set the common values first. we
c      include the additional assignment in case twoline2rv is not used.

       bstar  = xbstar
       ecco   = xecco
       argpo  = xargpo
       inclo  = xinclo
       mo     = xmo
       no     = xno
       nodeo  = xnodeo

       ! sgp4fix identify constants and allow alternate values
       CALL getgravconst( whichconst, tumin, mu, radiusearthkm, xke,
     &       j2, j3, j4, j3oj2 )

       SS     = 78.0D0/RadiusEarthKm + 1.0D0
       QZMS2T = ((120.0D0-78.0D0)/RadiusEarthKm) ** 4
       X2o3   =  2.0D0 / 3.0D0

c     sgp4fix divisor for divide by zero check on inclination
c     the old check used 1.0D0 + cos(pi-1.0D-9), but then compared it to
c     1.5D-12, so the threshold was changed to 1.5D-12 for consistency
       temp4    =   1.5D-12

       Init = 'y'
       T = 0.0D0

       CALL INITL( Satn , whichconst, Ecco  , EPOCH , Inclo , No,
     &     Method, AINV  , AO    , CON41 , CON42 , COSIO , COSIO2,
     &     Eccsq , OMEOSQ, POSQ  , rp    , RTEOSQ, SINIO ,
     &     GSTo, Opsmode )

       IF(rp .lt. 1.0D0) THEN
         Error = 5
        ENDIF

       IF(OMEOSQ .ge. 0.0D0 .OR. No .ge. 0.0D0) THEN
         ISIMP = 0
         IF (rp .lt. (220.0D0/RadiusEarthKm+1.0D0)) THEN
           ISIMP = 1
          ENDIF
         SFour  = SS
         QZMS24 = QZMS2T
```

Formatted: French (France)

Formatted: French (France)

```
         PERIGE = (rp-1.0D0)*RadiusEarthKm

* ----------- For perigees below 156 km, S and Qoms2t are altered -----
         IF(PERIGE .lt. 156.0D0) THEN
            SFour = PERIGE-78.0D0
            IF(PERIGE .le. 98.0D0) THEN
               SFour = 20.0D0
             ENDIF
            QZMS24 = ( (120.0D0-SFour)/RadiusEarthKm )**4
            SFour  = SFour/RadiusEarthKm + 1.0D0
          ENDIF
         PINVSQ = 1.0D0/POSQ

         TSI    = 1.0D0/(AO-SFour)
         ETA    = AO*Ecco*TSI
         ETASQ  = ETA*ETA
         EETA   = Ecco*ETA
         PSISQ  = DABS(1.0D0-ETASQ)
         COEF   = QZMS24*TSI**4
         COEF1  = COEF/PSISQ**3.5D0
         CC2    = COEF1*No* (AO* (1.0D0+1.5D0*ETASQ+EETA*
     &         (4.0D0+ETASQ) )+0.375D0*
     &      J2*TSI/PSISQ*CON41*(8.0D0+3.0D0*ETASQ*(8.0D0+ETASQ)))
         CC1    = BSTAR*CC2
         CC3    = 0.0D0
         IF(Ecco .GT. 1.0D-4) THEN
            CC3 = -2.0D0*COEF*TSI*J3OJ2*No*SINIO/Ecco
          ENDIF
         X1MTH2 = 1.0D0-COSIO2
         CC4    = 2.0D0*No*COEF1*AO*OMEOSQ*(ETA*(2.0D0+0.5D0*ETASQ)
     &         +Ecco*(0.5D0 + 2.0D0*ETASQ) - J2*TSI / (AO*PSISQ)*
     &         (-3.0D0*CON41*(1.0D0-2.0D0*
     &     EETA+ETASQ*(1.5D0-0.5D0*EETA))+0.75D0*X1MTH2*(2.0D0*ETASQ
     &     -EETA*(1.0D0+ETASQ))*DCOS(2.0D0*Argpo)))
         CC5    = 2.0D0*COEF1*AO*OMEOSQ* (1.0D0 + 2.75D0*
     &         (ETASQ + EETA) + EETA*ETASQ )
         COSIO4 = COSIO2*COSIO2
         TEMP1  = 1.5D0*J2*PINVSQ*No
         TEMP2  = 0.5D0*TEMP1*J2*PINVSQ
         TEMP3  = -0.46875D0*J4*PINVSQ*PINVSQ*No
         MDot   = No + 0.5D0*TEMP1*RTEOSQ*CON41 + 0.0625D0*TEMP2*
     &         RTEOSQ*(13.0D0 - 78.0D0*COSIO2 + 137.0D0*COSIO4)
         ArgpDot= -0.5D0*TEMP1*CON42 + 0.0625D0*TEMP2*
     &         (7.0D0 - 114.0D0*COSIO2 +
     &     395.0D0*COSIO4)+TEMP3*(3.0D0-36.0D0*COSIO2+49.0D0*COSIO4)
         XHDOT1 = -TEMP1*COSIO
```

Formatted: French (France)

Formatted: French (France)

```
      nodeDot = XHDOT1+(0.5D0*TEMP2*(4.0D0-19.0D0*COSIO2)+
     &         2.0D0*TEMP3*(3.0D0 - 7.0D0*COSIO2))*COSIO
      XPIDOT = ArgpDot+nodeDot
      OMGCOF = BSTAR*CC3*DCOS(Argpo)
      XMCOF  = 0.0D0
      IF(Ecco .GT. 1.0D-4) THEN
         XMCOF = -X2O3*COEF*BSTAR/EETA
       ENDIF
      XNODCF = 3.5D0*OMEOSQ*XHDOT1*CC1
      T2COF  = 1.5D0*CC1

c      sgp4fix for divide by zero with xinco = 180 deg
      if (dabs(cosio+1.0).gt. 1.5d-12) THEN
         XLCOF  = -0.25D0*J3OJ2*SINIO*
     &          (3.0D0+5.0D0*COSIO)/(1.0D0+COSIO)
       else
         XLCOF  = -0.25D0*J3OJ2*SINIO*
     &          (3.0D0+5.0D0*COSIO)/temp4
       ENDIF
      AYCOF  = -0.5D0*J3OJ2*SINIO
      DELMO  = (1.0D0+ETA*DCOS(Mo))**3
      SINMAO = DSIN(Mo)
      X7THM1 = 7.0D0*COSIO2-1.0D0

* ------------------------ Deep Space Initialization ------------------
      IF ((TWOPI/No) .ge. 225.0D0) THEN
         METHOD = 'd'
         ISIMP  = 1
         TC     = 0.0D0
         Inclm  = Inclo
         CALL DSCOM( EPOCH     , Ecco  , Argpo , Tc    , Inclo ,
     &         nodeo, No    ,
     &         SNODM , CNODM , SINIM , COSIM , SINOMM, COSOMM,
     &         DAY   , E3    , Ee2   , Eccm  , EMSQ  , GAM   ,
     &         Peo   , Pgho  , Pho   , PInco , Plo   ,
     &         RTemSq, Se2   , Se3   , Sgh2  , Sgh3  , Sgh4  ,
     &         Sh2   , Sh3   , Si2   , Si3   , Sl2   , Sl3   ,
     &         Sl4   , S1    , S2    , S3    , S4    , S5    ,
     &         S6    , S7    , SS1   , SS2   , SS3   , SS4   ,
     &         SS5   , SS6   , SS7   , SZ1   , SZ2   , SZ3   ,
     &         SZ11  , SZ12  , SZ13  , SZ21  , SZ22  , SZ23  ,
     &         SZ31  , SZ32  , SZ33  , Xgh2  , Xgh3  , Xgh4  ,
     &         Xh2   , Xh3   , Xi2   , Xi3   , Xl2   , Xl3   ,
     &         Xl4   , Xn    , Z1    , Z2    , Z3    , Z11   ,
     &         Z12   , Z13   , Z21   , Z22   , Z23   , Z31   ,
     &         Z32   , Z33   , Zmol  , Zmos )
```

Formatted: German (Germany)

Formatted: French (France)

Formatted: German (Germany)

```
          CALL DPPER( e3, ee2   , peo   , pgho  , pho   , pinco ,
     &         plo   , se2   , se3   , sgh2  , sgh3  , sgh4  ,
     &         sh2   , sh3   , si2   , si3   , sl2   , sl3   ,
     &         sl4   , T     , xgh2  , xgh3  , xgh4  , xh2   ,
     &         xh3   , xi2   , xi3   , xl2   , xl3   , xl4   ,
     &         zmol  , zmos  , Inclm , init  ,
     &         Ecco  , Inclo , nodeo, Argpo , Mo, Opsmode )

          Argpm  = 0.0D0 ! add for DS to work initial
          nodem  = 0.0D0
          Mm     = 0.0D0

          CALL DSINIT( whichconst,
     &          Cosim ,Emsq, Argpo, S1    , S2    , S3    ,
     &          S4    , S5    , Sinim , Ss1   , Ss2   , Ss3   ,
     &          Ss4   , Ss5   , Sz1   , Sz3   , Sz11  , Sz13  ,
     &          Sz21  , Sz23  , Sz31  , Sz33  , T     , Tc    ,
     &          GSTo  , Mo    , MDot  , No    ,nodeo,nodeDot,
     &          XPIDOT, Z1    , Z3    , Z11   , Z13   , Z21   ,
     &          Z23   , Z31   , Z33   , ecco  , eccsq,
     &          Eccm  , Argpm , Inclm , Mm    , Xn    , nodem,
     &          IREZ  , Atime , D2201 , D2211 , D3210 , D3222 ,
     &          D4410 , D4422 , D5220 , D5232 , D5421 , D5433 ,
     &          Dedt  , Didt  , DMDT  , DNDT  , DNODT , DOMDT ,
     &          Del1  , Del2  , Del3  , Xfact , Xlamo , Xli   ,
     &          Xni )
        ENDIF

* ------------ Set variables if not deep space or rp < 220 -------------
        IF (ISIMP .ne. 1) THEN
          CC1SQ = CC1*CC1
          D2    = 4.0D0*AO*TSI*CC1SQ
          TEMP  = D2*TSI*CC1 / 3.0D0
          D3    = (17.0D0*AO + SFour) * TEMP
          D4    = 0.5D0*TEMP*AO*TSI*
     &         (221.0D0*AO + 31.0D0*SFour)*CC1
          T3COF = D2 + 2.0D0*CC1SQ
          T4COF = 0.25D0* (3.0D0*D3+CC1*(12.0D0*D2+10.0D0*CC1SQ) )
          T5COF = 0.2D0* (3.0D0*D4 + 12.0D0*CC1*D3 + 6.0D0*D2*D2 +
     &         15.0D0*CC1SQ* (2.0D0*D2 + CC1SQ) )
         ENDIF

       ENDIF ! ------ if nodeo and No are gtr 0

     init = 'n'
```

Formatted: French (France)
Formatted: German (Germany)
Formatted: French (France)
Formatted: German (Germany)
Formatted: French (France)
Formatted: French (France)

```
      CALL SGP4(whichconst, 0.0D0, r, v, error)

      RETURN
      END  ! end sgp4init


* -----------------------------------------------------------------------------
*
*                           SUBROUTINE SGP4
*
*  this procedure is the sgp4 prediction model from space command. this is an
*    updated and combined version of sgp4 and sdp4, which were originally
*    published separately in spacetrack report #3. this version follows the
*    methodology from the aiaa paper (2006) describing the history and
*    development of the code.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    satrec         - initialised structure from sgp4init() call.
*    tsince         - time eince epoch (minutes)
*
*  outputs       :
*    r          - position vector                    km
*    v          - velocity                           km/sec
*
*  return code - non-zero on error.
*                   1 - mean elements, ecc >= 1.0 or ecc < -0.001 or a < 0.95 er
*                   2 - mean motion less than 0.0
*                   3 - pert elements, ecc < 0.0  or  ecc > 1.0
*                   4 - semi-latus rectum < 0.0
*                   5 - epoch elements are sub-orbital
*                   6 - satellite has decayed
*
*
*  coupling      :
*    getgravconst-
*    dpper
*    dpspace
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
*------------------------------------------------------------------------------
```

```
      SUBROUTINE SGP4 ( whichconst, T, r, v, Error )
        IMPLICIT NONE
        INTEGER  Error, whichconst
        REAL*8   T, r(3), v(3)

        INCLUDE 'SGP4.CMN'

* -------------------------- Local Variables --------------------------
        REAL*8 AM    , Axnl  , Aynl  , Betal , COSIM , Cnod  ,
     &        Cos2u , Coseo1, Cosi  , Cosip , Cosisq, Cossu , Cosu  ,
     &        Delm  , Delomg, Eccm  , EMSQ  , Ecose , El2   , Eo1   ,
     &        Eccp  , Esine , Argpm , Argpp , Omgadf, Pl    ,
     &        Rdotl , Rl    , Rvdot , Rvdotl, SINIM ,
     &        Sin2u , Sineo1, Sini  , Sinip , Sinsu , Sinu  ,
     &        Snod  , Su    , T2    , T3    , T4    , Tem5  , Temp  ,
     &        Temp1 , Temp2 , Tempa , Tempe , Templ , U     , Ux    ,
     &        Uy    , Uz    , Vx    , Vy    , Vz    , Inclm , Mm  ,
     &        XN    , nodem , Xinc  , Xincp , Xl    , Xlm   , Mp  ,
     &        Xmdf  , Xmx   , Xmy   , Xnoddf, Xnode , nodep,
     &        Tc    , Dndt

        REAL*8 X2O3, J2,J3,XKE,J3OJ2, mr,mv,
     &        mu, RadiusEarthkm, VKmPerSec, temp4, tumin, j4
            INTEGER iter

        CHARACTER Help
        INCLUDE 'ASTMATH.CMN'

* ------------------------ WGS-72 EARTH CONSTANTS ---------------------
* ---------------------- SET MATHEMATICAL CONSTANTS -------------------
      X2O3   = 2.0D0/3.0D0

c     Keep compiler ok for warnings on uninitialized variables
      mr = 0.0D0
      Coseo1 = 1.0D0
      Sineo1 = 0.0D0

      ! sgp4fix identify constants and allow alternate values
        CALL getgravconst( whichconst, tumin, mu, radiusearthkm, xke,
     &       j2, j3, j4, j3oj2 )

c     sgp4fix divisor for divide by zero check on inclination
c     the old check used 1.0D0 + cos(pi-1.0D-9), but then compared it to
c     1.5D-12, so the threshold was changed to 1.5D-12 for consistency
      temp4    =   1.5D-12
```

Formatted: French (France)

Formatted: German (Germany)

```
      VKmPerSec     =  RadiusEarthKm * xke/60.0D0

* ------------------------- CLEAR SGP4 ERROR FLAG ---------------------
      Error = 0

* ----------- UPDATE FOR SECULAR GRAVITY AND ATMOSPHERIC DRAG -------
--
      XMDF   = Mo + MDot*T
      OMGADF = Argpo + ArgpDot*T
      XNODDF = nodeo + nodeDot*T
      Argpm  = OMGADF
      Mm     = XMDF
      T2     = T*T
      nodem  = XNODDF + XNODCF*T2
      TEMPA  = 1.0D0 - CC1*T
      TEMPE  = BSTAR*CC4*T
      TEMPL  = T2COF*T2
      IF (ISIMP .ne. 1) THEN
         DELOMG = OMGCOF*T
         DELM   = XMCOF*(( 1.0D0+ETA*DCOS(XMDF) )**3-DELMO)
         TEMP   = DELOMG + DELM
         Mm     = XMDF + TEMP
         Argpm  = OMGADF - TEMP
         T3     = T2*T
         T4     = T3*T
         TEMPA  = TEMPA - D2*T2 - D3*T3 - D4*T4
         TEMPE  = TEMPE + BSTAR*CC5*(DSIN(Mm) - SINMAO)
         TEMPL  = TEMPL + T3COF*T3 + T4*(T4COF + T*T5COF)
        ENDIF
      XN     = No
      Eccm   = Ecco
      Inclm = Inclo
      IF(METHOD .EQ. 'd') THEN
         TC     = T
         CALL DSPACE( IRez  , D2201 , D2211 , D3210 , D3222 , D4410 ,
     &               D4422 , D5220 , D5232 , D5421 , D5433 , Dedt  ,
     &               Del1  , Del2  , Del3  , Didt  , Dmdt  , Dnodt ,
     &               Domdt , Argpo , ArgpDot, T    , TC    , GSTo ,
     &               Xfact , Xlamo , No    ,
     &               Atime , Eccm  , Argpm, Inclm , Xli   , Mm  ,
     &               XNi   , nodem, Dndt  , XN  )
        ENDIF

c     mean motion less than 0.0
      IF(XN .LE. 0.0D0) THEN
         Error = 2
```

Formatted: French (France)

Formatted: French (France)

Formatted: German (Germany)

```
        ENDIF
      AM = (XKE/XN)**X2O3*TEMPA**2
      XN = XKE/AM**1.5D0
      Eccm = Eccm-TEMPE

c   fix tolerance for error recognition
      IF (Eccm .GE. 1.0D0 .or. Eccm.lt.-0.001D0 .or. AM .lt. 0.95) THEN
c           write(6,*) '# Error 1, Eccm = ',  Eccm, ' AM = ', AM
          Error = 1
        ENDIF

c   sgp4fix change test condition for eccentricity
      IF (Eccm .lt. 1.0D-6) Eccm = 1.0D-6
      Mm     = Mm+No*TEMPL
      XLM    = Mm+Argpm+nodem
      EMSQ   = Eccm*Eccm
      TEMP   = 1.0D0 - EMSQ
      nodem  = DMOD(nodem,TwoPi)
      Argpm  = DMOD(Argpm,TwoPi)
      XLM    = DMOD(XLM,TwoPi)
      Mm     = DMOD(XLM - Argpm - nodem,TwoPi)

* --------------------- COMPUTE EXTRA MEAN QUANTITIES -----------------
      SINIM  = DSIN(Inclm)
      COSIM  = DCOS(Inclm)

* ------------------------ ADD LUNAR-SOLAR PERIODICS ------------------
      Eccp   = Eccm
      XINCP  = Inclm
      Argpp  = Argpm
      nodep = nodem
      Mp     = Mm
      SINIP  = SINIM
      COSIP  = COSIM
      IF(METHOD .EQ. 'd') THEN
          CALL DPPER( e3    , ee2   , peo   , pgho  , pho   , pinco ,
     &                plo   , se2   , se3   , sgh2  , sgh3  , sgh4  ,
     &                sh2   , sh3   , si2   , si3   , sl2   , sl3   ,
     &                sl4   , T     , xgh2  , xgh3  , xgh4  , xh2   ,
     &                xh3   , xi2   , xi3   , xl2   , xl3   , xl4   ,
     &                zmol  , zmos  , Inclo , 'n'   ,
     &                Eccp  , XIncp , nodep, Argpp, Mp, Opsmode )
          IF(XINCP .lt. 0.0D0) THEN
              XINCP  = -XINCP
              nodep  = nodep + PI
              Argpp  = Argpp - PI
```

Formatted: French (France)

```
          ENDIF
        IF(Eccp .lt. 0.0D0 .OR. Eccp .GT. 1.0D0) THEN
            Error = 3
          ENDIF
      ENDIF

* ------------------------ LONG PERIOD PERIODICS ----------------------
      IF(METHOD .EQ. 'd') THEN
        SINIP =  DSIN(XINCP)
        COSIP =  DCOS(XINCP)
        AYCOF = -0.5D0*J3OJ2*SINIP

c         sgp4fix for divide by zero with xincp = 180 deg
        if (dabs(cosip+1.0).gt. 1.5d-12) THEN
            XLCOF  = -0.25D0*J3OJ2*SINIP*
     &               (3.0D0+5.0D0*COSIP)/(1.0D0+COSIP)
          else
            XLCOF  = -0.25D0*J3OJ2*SINIP*
     &               (3.0D0+5.0D0*COSIP)/temp4
          ENDIF
      ENDIF
      AXNL = Eccp*DCOS(Argpp)
      TEMP = 1.0D0 / (AM*(1.0D0-Eccp*Eccp))
      AYNL = Eccp*DSIN(Argpp) + TEMP*AYCOF
      XL   = Mp + Argpp + nodep + TEMP*XLCOF*AXNL

* ------------------------- SOLVE KEPLER'S EQUATION -------------------
      U    = DMOD(XL-nodep,TwoPi)
      EO1  = U
      ITER=0

c   sgp4fix for kepler iteration
c   the following iteration needs better limits on corrections
      Temp = 9999.9D0
      DO WHILE ((Temp.ge.1.0D-12).and.(ITER.lt.10))
        ITER=ITER+1
        SINEO1= DSIN(EO1)
        COSEO1= DCOS(EO1)
        TEM5  = 1.0D0 - COSEO1*AXNL - SINEO1*AYNL
        TEM5  = (U - AYNL*COSEO1 + AXNL*SINEO1 - EO1) / TEM5
        Temp  = DABS(Tem5)
        IF(Temp.gt.1.0D0) Tem5=Tem5/Temp ! Stop excessive correction
        EO1   = EO1+TEM5
      ENDDO

* ----------------- SHORT PERIOD PRELIMINARY QUANTITIES ---------------
```

```
      ECOSE = AXNL*COSEO1+AYNL*SINEO1
      ESINE = AXNL*SINEO1-AYNL*COSEO1
      EL2   = AXNL*AXNL+AYNL*AYNL
      PL    = AM*(1.0D0-EL2)

c     semi-latus rectum < 0.0
      IF ( PL .lt. 0.0D0 ) THEN
        Error = 4
       ELSE
        RL    = AM*(1.0D0-ECOSE)
        RDOTL = DSQRT(AM)*ESINE/RL
        RVDOTL= DSQRT(PL)/RL
        BETAL = DSQRT(1.0D0-EL2)
        TEMP  = ESINE/(1.0D0+BETAL)
        SINU  = AM/RL*(SINEO1-AYNL-AXNL*TEMP)
        COSU  = AM/RL*(COSEO1-AXNL+AYNL*TEMP)
        SU    = DATAN2(SINU,COSU)
        SIN2U = (COSU+COSU)*SINU
        COS2U = 1.0D0-2.0D0*SINU*SINU
        TEMP  = 1.0D0/PL
        TEMP1 = 0.5D0*J2*TEMP
        TEMP2 = TEMP1*TEMP

* ------------------ UPDATE FOR SHORT PERIOD PERIODICS ----------------
        IF(METHOD .EQ. 'd') THEN
          COSISQ = COSIP*COSIP
          CON41  = 3.0D0*COSISQ - 1.0D0
          X1MTH2 = 1.0D0 - COSISQ
          X7THM1 = 7.0D0*COSISQ - 1.0D0
         ENDIF
        mr   = RL*(1.0D0 - 1.5D0*TEMP2*BETAL*CON41) +
     &          0.5D0*TEMP1*X1MTH2*COS2U
        SU   = SU - 0.25D0*TEMP2*X7THM1*SIN2U
        XNODE= nodep + 1.5D0*TEMP2*COSIP*SIN2U
        XINC = XINCP + 1.5D0*TEMP2*COSIP*SINIP*COS2U
        mv   = RDOTL - XN*TEMP1*X1MTH2*SIN2U / XKE
        RVDOT= RVDOTL + XN*TEMP1* (X1MTH2*COS2U+1.5D0*CON41) / XKE

* ------------------------- ORIENTATION VECTORS -----------------------
        SINSU=  DSIN(SU)
        COSSU=  DCOS(SU)
        SNOD =  DSIN(XNODE)
        CNOD =  DCOS(XNODE)
        SINI =  DSIN(XINC)
        COSI =  DCOS(XINC)
        XMX  = -SNOD*COSI
```

```
        XMY  =  CNOD*COSI
        UX   =  XMX*SINSU + CNOD*COSSU
        UY   =  XMY*SINSU + SNOD*COSSU
        UZ   =  SINI*SINSU
        VX   =  XMX*COSSU - CNOD*SINSU
        VY   =  XMY*COSSU - SNOD*SINSU
        VZ   =  SINI*COSSU

* ----------------------- POSITION AND VELOCITY -----------------------
        r(1) = mr*UX * RadiusEarthkm
        r(2) = mr*UY * RadiusEarthkm
        r(3) = mr*UZ * RadiusEarthkm
        v(1) = (mv*UX + RVDOT*VX) * VKmPerSec
        v(2) = (mv*UY + RVDOT*VY) * VKmPerSec
        v(3) = (mv*UZ + RVDOT*VZ) * VKmPerSec
       ENDIF

* --------------------------- ERROR PROCESSING ------------------------
c     sgp4fix for decaying satellites
      if (mr .lt. 1.0D0) THEN
        error = 6
       ENDIF

      RETURN
      END  ! end sgp4


* -----------------------------------------------------------------------------
*
*                           SUBROUTINE INITL
*
*  this subroutine initializes the spg4 propagator. all the initialization is
*    consolidated here instead of having multiple loops inside other routines.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    ecco        - eccentricity                           0.0 - 1.0
*    epoch       - epoch time in days from jan 0, 1950. 0 hr
*    inclo       - inclination of satellite
*    no          - mean motion of satellite
*    satn        - satellite number
*
*  outputs       :
*    ainv        - 1.0 / a
*    ao          - semi major axis
```

```
*    con41       -
*    con42       - 1.0 - 5.0 cos(i)
*    cosio       - cosine of inclination
*    cosio2      - cosio squared
*    eccsq       - eccentricity squared
*    method      - flag for deep space                    'd', 'n'
*    omeosq      - 1.0 - ecco * ecco
*    posq        - semi-parameter squared
*    rp          - radius of perigee
*    rteosq      - square root of (1.0 - ecco*ecco)
*    sinio       - sine of inclination
*    gsto        - gst at time of observation               rad
*    no          - mean motion of satellite
*
*
*  coupling      :
*    getgravconst-
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
*-----------------------------------------------------------------------------

      SUBROUTINE INITL( Satn , whichconst, Ecco  , EPOCH , Inclo , No,
     &         Method, AINV  , AO    , CON41 , CON42 , COSIO , COSIO2,
     &         Eccsq , OMEOSQ, POSQ  , rp    , RTEOSQ, SINIO ,
     &         GSTo, operationmode )
        IMPLICIT NONE
        CHARACTER Method, operationmode
        INTEGER Satn, whichconst
        REAL*8 Ecco  , EPOCH , Inclo , No   ,
     &         AINV  , AO    , CON41 , CON42 , COSIO , COSIO2,
     &         Eccsq , OMEOSQ, POSQ  , rp    , RTEOSQ, SINIO , GSTo

        COMMON /DebugHelp/ Help
        CHARACTER Help

* -------------------------- Local Variables --------------------------
c       sgp4fix use old way of finding gst
        Integer ids70
        REAL*8 ts70, ds70, tfrac, c1, thgr70, fk5r, c1p2p, thgr, thgro

        REAL*8  RadPerDay, Temp, TUT1
        REAL*8  ak, d1, del, adel, po
```

```
      REAL*8  X2o3, J2, XKE, tumin, mu, radiusearthkm, j3, j4, j3oj2
      INCLUDE 'ASTMATH.CMN'

* ------------------------ WGS-72 EARTH CONSTANTS ---------------------
      X2o3   = 2.0D0/3.0D0
      ! sgp4fix identify constants and allow alternate values
      CALL getgravconst( whichconst, tumin, mu, radiusearthkm, xke,
     &       j2, j3, j4, j3oj2 )

* ----------------- CALCULATE AUXILLARY EPOCH QUANTITIES --------------
      Eccsq  = Ecco*Ecco
      OMEOSQ = 1.0D0 - Eccsq
      RTEOSQ = DSQRT(OMEOSQ)
      COSIO  = DCOS(Inclo)
      COSIO2 = COSIO*COSIO

* ---------------------- UN-KOZAI THE MEAN MOTION ---------------------
      AK   =  (XKE/No)**X2O3
      D1   =  0.75D0*J2* (3.0D0*COSIO2-1.0D0) / (RTEOSQ*OMEOSQ)
      DEL  =  D1/(AK*AK)
      ADEL =  AK * ( 1.0D0 - DEL*DEL - DEL*
     &               (1.0D0/3.0D0 + 134.0D0*DEL*DEL / 81.0D0) )
      DEL  =  D1/(ADEL*ADEL)
      No   =  No/(1.0D0 + DEL)

      AO   =  (XKE/No)**X2O3
      SINIO=  DSIN(Inclo)
      PO   =  AO*OMEOSQ
      CON42=  1.0D0-5.0D0*COSIO2
      CON41=  -CON42-COSIO2-COSIO2
      AINV =  1.0D0/AO
      POSQ =  PO*PO
      rp   =  AO*(1.0D0-Ecco)
      METHOD = 'n'

* ----------------- CALCULATE GREENWICH LOCATION AT EPOCH -------------
c       sgp4fix modern approach to finding sidereal time
      IF (operationmode .ne. 'a') THEN
        RadPerDay  = twopi * 1.002737909350795D0  !6.30038809866574D0
        Temp = Epoch + 2433281.5D0
        TUT1= ( DINT(Temp-0.5D0) + 0.5D0 - 2451545.0D0 ) / 36525.0D0
        Gsto= 1.75336855923327D0 + 628.331970688841D0*TUT1
     &          + 6.77071394490334D-06*TUT1*TUT1
     &          - 4.50876723431868D-10*TUT1*TUT1*TUT1
     &          + RadPerDay*( Temp-0.5D0-DINT(Temp-0.5D0) )
        ELSE
```

Formatted: German (Germany)

Formatted: French (France)

Formatted: German (Germany)

Formatted: French (France)

```
         ! sgp4fix use old way of finding gst
         ! count integer number of days from 0 jan 1970
        TS70  = EPOCH-7305.0D0
        IDS70 = TS70 + 1.0D-8
        TFRAC = TS70-IDS70
         ! find greenwich location at epoch
        C1    = 1.72027916940703639D-2
        THGR70= 1.7321343856509374D0
         FK5R  = 5.07551419432269442D-15
        C1P2P = C1+TWOPI
        gsto  = THGR70+C1*IDS70+C1P2P*TFRAC+TS70*TS70*FK5R
       ENDIF

      ! ------------------------ Check quadrants ---------------------
      Gsto = DMOD( Gsto,TwoPi )
      IF ( Gsto .lt. 0.0D0 ) THEN
         Gsto= Gsto + TwoPi
       ENDIF

     RETURN
     END  ! end initl

* -----------------------------------------------------------------------------
*
*                           SUBROUTINE DPPER
*
*  This Subroutine provides deep space long period periodic contributions
*    to the mean elements.  by design, these periodics are zero at epoch.
*    this used to be dscom which included initialization, but it's really a
*    recurring function.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    e3          -
*    ee2         -
*    peo         -
*    pgho        -
*    pho         -
*    pinco       -
*    plo         -
*    se2 , se3 , Sgh2, Sgh3, Sgh4, Sh2, Sh3, Si2, Si3, Sl2, Sl3, Sl4 -
*    t           -
*    xh2, xh3, xi2, xi3, xl2, xl3, xl4 -
*    zmol        -
*    zmos        -
```

Formatted: French (France)

Formatted: German (Germany)

```
*    ep          - eccentricity                        0.0 - 1.0
*    inclo       - inclination - needed for lyddane modification
*    nodep       - right ascension of ascending node
*    argpp       - argument of perigee
*    mp          - mean anomaly
*
*  outputs       :
*    ep          - eccentricity                        0.0 - 1.0
*    inclp       - inclination
*    nodep       - right ascension of ascending node
*    argpp       - argument of perigee
*    mp          - mean anomaly
*
*  coupling      :
*    none.
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
*-----------------------------------------------------------------------------

      SUBROUTINE DPPER( e3    , ee2   , peo   , pgho  , pho   , pinco ,
     &                  plo   , se2   , se3   , sgh2  , sgh3  , sgh4  ,
     &                  sh2   , sh3   , si2   , si3   , sl2   , sl3   ,
     &                  sl4   , T     , xgh2  , xgh3  , xgh4  , xh2   ,
     &                  xh3   , xi2   , xi3   , xl2   , xl3   , xl4   ,
     &                  zmol  , zmos  , inclo , init  ,
     &                  Eccp  , Inclp , nodep, Argpp , Mp,
     &                  operationmode )
        IMPLICIT NONE
        CHARACTER Init, operationmode
        REAL*8  e3    , ee2   , peo   , pgho  , pho   , pinco , plo   ,
     &          se2   , se3   , sgh2  , sgh3  , sgh4  , sh2   , sh3   ,
     &          si2   , si3   , sl2   , sl3   , sl4   , T     , xgh2  ,
     &          xgh3  , xgh4  , xh2   , xh3   , xi2   , xi3   , xl2   ,
     &          xl3   , xl4   , zmol  , zmos  , inclo ,
     &          Eccp  , Inclp , nodep, Argpp , Mp

* -------------------------- Local Variables --------------------------
        REAL*8  alfdp , betdp , cosip , cosop , dalf  , dbet  , dls   ,
     &          f2    , f3    , pe    , pgh   , ph    , pinc  , pl    ,
     &          sel   , ses   , sghl  , sghs  , shl   , shs   , sil   ,
     &          sinip , sinop , sinzf , sis   , sll   , sls   , xls   ,
     &          xnoh  , zf    , zm
```

```
      REAL*8  Zel    , Zes    , Znl    , Zns
      !COMMON /DebugHelp/ Help
      CHARACTER Help
      INCLUDE 'ASTMATH.CMN'

*  ----------------------------- Constants -----------------------------
      ZES  = 0.01675D0
      ZEL  = 0.05490D0
      ZNS  = 1.19459D-5
      ZNL  = 1.5835218D-4

*  ------------------- CALCULATE TIME VARYING PERIODICS ----------------
      ZM   = ZMOS + ZNS*T

      IF (Init.eq.'y') ZM = ZMOS
      ZF   = ZM + 2.0D0*ZES*DSIN(ZM)
      SINZF= DSIN(ZF)
      F2   =  0.5D0*SINZF*SINZF - 0.25D0
      F3   = -0.5D0*SINZF*DCOS(ZF)
      SES  = SE2*F2 + SE3*F3
      SIS  = SI2*F2 + SI3*F3
      SLS  = SL2*F2 + SL3*F3 + SL4*SINZF
      SGHS = SGH2*F2 + SGH3*F3 + SGH4*SINZF
      SHS  = SH2*F2 + SH3*F3
      ZM   = ZMOL + ZNL*T

      IF (Init.eq.'y') ZM = ZMOL
      ZF   = ZM + 2.0D0*ZEL*DSIN(ZM)
      SINZF= DSIN(ZF)
      F2   =  0.5D0*SINZF*SINZF - 0.25D0
      F3   = -0.5D0*SINZF*DCOS(ZF)
      SEL  = EE2*F2 + E3*F3
      SIL  = XI2*F2 + XI3*F3
      SLL  = XL2*F2 + XL3*F3 + XL4*SINZF
      SGHL = XGH2*F2 + XGH3*F3 + XGH4*SINZF
      SHL  = XH2*F2 + XH3*F3
      PE   = SES + SEL
      PINC = SIS + SIL
      PL   = SLS + SLL
      PGH  = SGHS + SGHL
      PH   = SHS + SHL

      IF (Init.eq.'n') THEN
        PE    = PE   - PEO
        PINC  = PINC - PINCO
        PL    = PL   - PLO
```

Formatted: German (Germany)

Formatted: German (Germany)

Formatted: French (France)

Formatted: German (Germany)

Formatted: French (France)

```
        PGH   = PGH  - PGHO
        PH    = PH   - PHO
        Inclp = Inclp  + PINC
        Eccp  = Eccp   + PE
        SINIP = DSIN(Inclp)
        COSIP = DCOS(Inclp)

* ------------------------ APPLY PERIODICS DIRECTLY ------------------
c    sgp4fix for lyddane choice
c    strn3 used original inclination - this is technically feasible
c    gsfc used perturbed inclination - also technically feasible
c    probably best to readjust the 0.2 limit value and limit discontinuity
c    0.2 rad = 11.45916 deg
c    use next line for original strn3 approach and original inclination
c          IF (inclo.ge.0.2D0) THEN
c    use next line for gsfc version and perturbed inclination
        IF (Inclp.ge.0.2D0) THEN

            PH     = PH/SINIP
            PGH    = PGH - COSIP*PH
            Argpp  = Argpp + PGH
            nodep  = nodep + PH
            Mp     = Mp + PL
          ELSE

* ----------------- APPLY PERIODICS WITH LYDDANE MODIFICATION ---------
            SINOP  = DSIN(nodep)
            COSOP  = DCOS(nodep)
            ALFDP  = SINIP*SINOP
            BETDP  = SINIP*COSOP
            DALF   =  PH*COSOP + PINC*COSIP*SINOP
            DBET   = -PH*SINOP + PINC*COSIP*COSOP
            ALFDP  = ALFDP + DALF
            BETDP  = BETDP + DBET
            nodep = DMOD(nodep,TwoPi)

            ! sgp4fix for afspc written intrinsic functions
            ! nodep used without a trigonometric function ahead
            IF ((nodep .LT. 0.0D0) .and. (operationmode .eq. 'a'))
     &              THEN
              nodep = nodep + twopi
             ENDIF
            XLS    = Mp + Argpp + COSIP*nodep
            DLS    = PL + PGH - PINC*nodep*SINIP
            XLS    = XLS + DLS
            XNOH   = nodep
```

```
           nodep  = DATAN2(ALFDP,BETDP)

           ! sgp4fix for afspc written intrinsic functions
           ! nodep used without a trigonometric function ahead
           IF ((nodep .LT. 0.0D0) .and. (operationmode .eq. 'a'))
     &             THEN
             nodep = nodep + twopi
            ENDIF
           IF (DABS(XNOH-nodep) .GT. PI) THEN
             IF(nodep .lt. XNOH) THEN
               nodep = nodep+TWOPI
              ELSE
               nodep = nodep-TWOPI
              ENDIF
            ENDIF
           Mp   = Mp + PL
           Argpp=  XLS - Mp - COSIP*nodep
          ENDIF
        ENDIF



      RETURN
      END  !  end dpper


* ----------------------------------------------------------------------------
*
*                           SUBROUTINE DSCOM
*
*  This Subroutine provides deep space common items used by both the secular
*    and periodics subroutines.  input is provided as shown. this routine
*    used to be called dpper, but the functions inside weren't well organized.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    epoch       -
*    ep          - eccentricity
*    argpp       - argument of perigee
*    tc          -
*    inclp       - inclination
*    nodep       - right ascension of ascending node
*    np          - mean motion
*
*  outputs       :
```

```
*   sinim  , cosim  , sinomm , cosomm , snodm  , cnodm
*   day        -
*   e3         -
*   ee2        -
*   em         - eccentricity
*   emsq       - eccentricity squared
*   gam        -
*   peo        -
*   pgho       -
*   pho        -
*   pinco      -
*   plo        -
*   rtemsq     -
*   se2, se3         -
*   sgh2, sgh3, sgh4        -
*   sh2, sh3, si2, si3, sl2, sl3, sl4         -
*   s1, s2, s3, s4, s5, s6, s7          -
*   ss1, ss2, ss3, ss4, ss5, ss6, ss7, sz1, sz2, sz3         -
*   sz11, sz12, sz13, sz21, sz22, sz23, sz31, sz32, sz33        -
*   xgh2, xgh3, xgh4, xh2, xh3, xi2, xi3, xl2, xl3, xl4         -
*   nm         - mean motion
*   z1, z2, z3, z11, z12, z13, z21, z22, z23, z31, z32, z33         -
*   zmol       -
*   zmos       -
*
*
*  coupling      :
*   none.
*
*  references    :
*   hoots, roehrich, norad spacetrack report #3 1980
*   hoots, norad spacetrack report #6 1986
*   hoots, schumacher and glover 2004
*   vallado, crawford, hujsak, kelso  2006
*----------------------------------------------------------------------------

      SUBROUTINE DSCOM( EPOCH , Eccp  , Argpp , Tc    , Inclp , nodep,
     &                  Np    ,
     &                  SNODM , CNODM , SINIM , COSIM , SINOMM, COSOMM,
     &                  DAY   , E3    , Ee2   , Eccm  , EMSQ  , GAM   ,
     &                  Peo   , Pgho  , Pho   , PInco , Plo   ,
     &                  RTemSq, Se2   , Se3   , Sgh2  , Sgh3  , Sgh4  ,
     &                  Sh2   , Sh3   , Si2   , Si3   , Sl2   , Sl3   ,
     &                  Sl4   , S1    , S2    , S3    , S4    , S5    ,
     &                  S6    , S7    , SS1   , SS2   , SS3   , SS4   ,
     &                  SS5   , SS6   , SS7   , SZ1   , SZ2   , SZ3   ,
```

```
     &              SZ11  , SZ12  , SZ13  , SZ21  , SZ22  , SZ23  ,
     &              SZ31  , SZ32  , SZ33  , Xgh2  , Xgh3  , Xgh4  ,
     &              Xh2   , Xh3   , Xi2   , Xi3   , Xl2   , Xl3   ,
     &              Xl4   , Xn    , Z1    , Z2    , Z3    , Z11   ,
     &              Z12   , Z13   , Z21   , Z22   , Z23   , Z31   ,
     &              Z32   , Z33   , Zmol  , Zmos )
        IMPLICIT NONE
        REAL*8  EPOCH , Eccp  , Argpp , Tc    , Inclp , nodep, Np    ,
     &         SNODM , CNODM , SINIM , COSIM , SINOMM, COSOMM, DAY   ,
     &         E3    , Ee2   , Eccm  , EMSQ  , GAM   , RTemSq, Se2   ,
     &         Peo   , Pgho  , Pho   , PInco , Plo   ,
     &         Se3   , Sgh2  , Sgh3  , Sgh4  , Sh2   , Sh3   , Si2   ,
     &         Si3   , Sl2   , Sl3   , Sl4   , S1    , S2    , S3    ,
     &         S4    , S5    , S6    , S7    , SS1   , SS2   , SS3   ,
     &         SS4   , SS5   , SS6   , SS7   , SZ1   , SZ2   , SZ3   ,
     &         SZ11  , SZ12  , SZ13  , SZ21  , SZ22  , SZ23  , SZ31  ,
     &         SZ32  , SZ33  , Xgh2  , Xgh3  , Xgh4  , Xh2   , Xh3   ,
     &         Xi2   , Xi3   , Xl2   , Xl3   , Xl4   , Xn    , Z1    ,
     &         Z2    , Z3    , Z11   , Z12   , Z13   , Z21   , Z22   ,
     &         Z23   , Z31   , Z32   , Z33   , Zmol  , Zmos

* -------------------------- Local Variables --------------------------
        REAL*8  c1ss  , c1L   , zcosis, zsinis, zsings, zcosgs,
     &         Zes   , zel
        INTEGER LsFlg
        REAL*8  a1    , a2    , a3    , a4    , a5    , a6    , a7    ,
     &         a8    , a9    , a10   , betasq, cc    , ctem  , stem  ,
     &         x1    , x2    , x3    , x4    , x5    , x6    , x7    ,
     &         x8    , xnodce, xnoi  , zcosg , zcosgl, zcosh , zcoshl,
     &         zcosi , zcosil, zsing , zsingl, zsinh , zsinhl, zsini ,
     &         zsinil, zx    , zy

        CHARACTER Help
        INCLUDE 'ASTMATH.CMN'

* ------------------------------ Constants ----------------------------
        ZES    =  0.01675D0
        ZEL    =  0.05490D0
        C1SS   =  2.9864797D-6
        C1L    =  4.7968065D-7
        ZSINIS =  0.39785416D0
        ZCOSIS =  0.91744867D0
        ZCOSGS =  0.1945905D0
        ZSINGS = -0.98088458D0

* ----------------- DEEP SPACE PERIODICS INITIALIZATION ---------------
```

```
        XN     = Np
        Eccm   = Eccp
        SNODM  = DSIN(nodep)
        CNODM  = DCOS(nodep)
        SINOMM = DSIN(Argpp)
        COSOMM = DCOS(Argpp)
        SINIM  = DSIN(Inclp)
        COSIM  = DCOS(Inclp)
        EMSQ   = Eccm*Eccm
        BETASQ = 1.0D0-EMSQ
        RTEMSQ = DSQRT(BETASQ)

* --------------------- INITIALIZE LUNAR SOLAR TERMS ------------------
        PEO    = 0.0D0
        PINCO  = 0.0D0
        PLO    = 0.0D0
        PGHO   = 0.0D0
        PHO    = 0.0D0
        DAY    = EPOCH + 18261.5D0 + TC/1440.0D0
        XNODCE = DMOD(4.5236020D0 - 9.2422029D-4*DAY,TwoPi)
        STEM   = DSIN(XNODCE)
        CTEM   = DCOS(XNODCE)
        ZCOSIL = 0.91375164D0 - 0.03568096D0*CTEM
        ZSINIL = DSQRT(1.0D0 - ZCOSIL*ZCOSIL)
        ZSINHL = 0.089683511D0*STEM / ZSINIL
        ZCOSHL = DSQRT(1.0D0 - ZSINHL*ZSINHL)
        GAM    = 5.8351514D0 + 0.0019443680D0*DAY
        ZX     = 0.39785416D0*STEM/ZSINIL
        ZY     = ZCOSHL*CTEM + 0.91744867D0*ZSINHL*STEM
        ZX     = DATAN2(ZX,ZY)
        ZX     = GAM + ZX - XNODCE
        ZCOSGL = DCOS(ZX)
        ZSINGL = DSIN(ZX)

* ---------------------------- DO SOLAR TERMS -------------------------
        ZCOSG = ZCOSGS
        ZSING = ZSINGS
        ZCOSI = ZCOSIS
        ZSINI = ZSINIS
        ZCOSH = CNODM
        ZSINH = SNODM
        CC    = C1SS
        XNOI  = 1.0D0 / XN

        DO LSFlg = 1,2
          A1 =   ZCOSG*ZCOSH + ZSING*ZCOSI*ZSINH
```

Formatted: German (Germany)

```
          A3 =  -ZSING*ZCOSH + ZCOSG*ZCOSI*ZSINH
          A7 =  -ZCOSG*ZSINH + ZSING*ZCOSI*ZCOSH
          A8 =   ZSING*ZSINI
          A9 =   ZSING*ZSINH + ZCOSG*ZCOSI*ZCOSH
          A10=   ZCOSG*ZSINI
          A2 =   COSIM*A7 + SINIM*A8
          A4 =   COSIM*A9 + SINIM*A10
          A5 =  -SINIM*A7 + COSIM*A8
          A6 =  -SINIM*A9 + COSIM*A10

          X1 =  A1*COSOMM + A2*SINOMM
          X2 =  A3*COSOMM + A4*SINOMM
          X3 = -A1*SINOMM + A2*COSOMM
          X4 = -A3*SINOMM + A4*COSOMM
          X5 =  A5*SINOMM
          X6 =  A6*SINOMM
          X7 =  A5*COSOMM
          X8 =  A6*COSOMM

          Z31= 12.0D0*X1*X1 - 3.0D0*X3*X3
          Z32= 24.0D0*X1*X2 - 6.0D0*X3*X4
          Z33= 12.0D0*X2*X2 - 3.0D0*X4*X4
          Z1 =  3.0D0* (A1*A1 + A2*A2) + Z31*EMSQ
          Z2 =  6.0D0* (A1*A3 + A2*A4) + Z32*EMSQ
          Z3 =  3.0D0* (A3*A3 + A4*A4) + Z33*EMSQ
          Z11= -6.0D0*A1*A5 + EMSQ* (-24.0D0*X1*X7-6.0D0*X3*X5)
          Z12= -6.0D0* (A1*A6 + A3*A5) + EMSQ*
     &        ( -24.0D0*(X2*X7+X1*X8) - 6.0D0*(X3*X6+X4*X5) )
          Z13= -6.0D0*A3*A6 + EMSQ*(-24.0D0*X2*X8 - 6.0D0*X4*X6)
          Z21=  6.0D0*A2*A5 + EMSQ*(24.0D0*X1*X5-6.0D0*X3*X7)
          Z22=  6.0D0* (A4*A5 + A2*A6) + EMSQ*
     &        (  24.0D0*(X2*X5+X1*X6) - 6.0D0*(X4*X7+X3*X8) )
          Z23=  6.0D0*A4*A6 + EMSQ*(24.0D0*X2*X6 - 6.0D0*X4*X8)
          Z1 = Z1 + Z1 + BETASQ*Z31
          Z2 = Z2 + Z2 + BETASQ*Z32
          Z3 = Z3 + Z3 + BETASQ*Z33
          S3 = CC*XNOI
          S2 = -0.5D0*S3 / RTEMSQ
          S4 = S3*RTEMSQ
          S1 = -15.0D0*Eccm*S4
          S5 = X1*X3 + X2*X4
          S6 = X2*X3 + X1*X4
          S7 = X2*X4 - X1*X3

* ------------------------------ DO LUNAR TERMS -----------------------
          IF (LSFLG.eq.1) THEN
```

```
            SS1   = S1
            SS2   = S2
            SS3   = S3
            SS4   = S4
            SS5   = S5
            SS6   = S6
            SS7   = S7
            SZ1   = Z1
            SZ2   = Z2
            SZ3   = Z3
            SZ11  = Z11
            SZ12  = Z12
            SZ13  = Z13
            SZ21  = Z21
            SZ22  = Z22
            SZ23  = Z23
            SZ31  = Z31
            SZ32  = Z32
            SZ33  = Z33
            ZCOSG = ZCOSGL
            ZSING = ZSINGL
            ZCOSI = ZCOSIL
            ZSINI = ZSINIL
            ZCOSH = ZCOSHL*CNODM+ZSINHL*SNODM
            ZSINH = SNODM*ZCOSHL-CNODM*ZSINHL
            CC    = C1L
          ENDIF
        ENDDO

      ZMOL  = DMOD( 4.7199672D0 + 0.22997150D0*DAY-GAM,TwoPi )
      ZMOS  = DMOD( 6.2565837D0 + 0.017201977D0*DAY,TwoPi )

* ---------------------------- DO SOLAR TERMS -------------------------
      SE2 =   2.0D0*SS1*SS6
      SE3 =   2.0D0*SS1*SS7
      SI2 =   2.0D0*SS2*SZ12
      SI3 =   2.0D0*SS2*(SZ13-SZ11)
      SL2 =  -2.0D0*SS3*SZ2
      SL3 =  -2.0D0*SS3*(SZ3-SZ1)
      SL4 =  -2.0D0*SS3*(-21.0D0-9.0D0*EMSQ)*ZES
      SGH2=   2.0D0*SS4*SZ32
      SGH3=   2.0D0*SS4*(SZ33-SZ31)
      SGH4= -18.0D0*SS4*ZES
      SH2 =  -2.0D0*SS2*SZ22
      SH3 =  -2.0D0*SS2*(SZ23-SZ21)
```

Formatted: German (Germany)

Formatted: French (France)

```
* ---------------------------- DO LUNAR TERMS -------------------------
      EE2 =   2.0D0*S1*S6
      E3  =   2.0D0*S1*S7
      XI2 =   2.0D0*S2*Z12
      XI3 =   2.0D0*S2*(Z13-Z11)
      XL2 =  -2.0D0*S3*Z2
      XL3 =  -2.0D0*S3*(Z3-Z1)
      XL4 =  -2.0D0*S3*(-21.0D0-9.0D0*EMSQ)*ZEL
      XGH2=   2.0D0*S4*Z32
      XGH3=   2.0D0*S4*(Z33-Z31)
      XGH4= -18.0D0*S4*ZEL
      XH2 =  -2.0D0*S2*Z22
      XH3 =  -2.0D0*S2*(Z23-Z21)


    RETURN
    END  !  dscom


* -----------------------------------------------------------------------------
*
*                        SUBROUTINE DSINIT
*
*  This Subroutine provides Deep Space contributions to Mean Motion Dot due
*    to geopotential resonance with half day and one day orbits.
*
*  Inputs        :
*    Cosim, Sinim-
*    Emsq        - Eccentricity squared
*    Argpo       - Argument of Perigee
*    S1, S2, S3, S4, S5      -
*    Ss1, Ss2, Ss3, Ss4, Ss5 -
*    Sz1, Sz3, Sz11, Sz13, Sz21, Sz23, Sz31, Sz33 -
*    T           - Time
*    Tc          -
*    GSTo        - Greenwich sidereal time                   rad
*    Mo          - Mean Anomaly
*    MDot        - Mean Anomaly dot (rate)
*    No          - Mean Motion
*    nodeo       - right ascension of ascending node
*    nodeDot     - right ascension of ascending node dot (rate)
*    XPIDOT      -
*    Z1, Z3, Z11, Z13, Z21, Z23, Z31, Z33 -
*    Eccm        - Eccentricity
*    Argpm       - Argument of perigee
*    Inclm       - Inclination
```

```
*    Mm          - Mean Anomaly
*    Xn          - Mean Motion
*    nodem       - right ascension of ascending node
*
*  Outputs       :
*    Eccm        - Eccentricity
*    Argpm       - Argument of perigee
*    Inclm       - Inclination
*    Mm          - Mean Anomaly
*    Xn          - Mean motion
*    nodem       - right ascension of ascending node
*    IRez        - Resonance flags              0-none, 1-One day,  2-Half day
*    Atime       -
*    D2201, D2211, D3210, D3222, D4410, D4422, D5220, D5232, D5421, D5433       -
*    Dedt        -
*    Didt        -
*    DMDT        -
*    DNDT        -
*    DNODT       -
*    DOMDT       -
*    Del1, Del2, Del3 -
*    Ses  , Sghl , Sghs , Sgs  , Shl  , Shs  , Sis  , Sls
*    THETA       -
*    Xfact       -
*    Xlamo       -
*    Xli         -
*    Xni
*
*  Locals        :
*    ainv2       -
*    aonv        -
*    cosisq      -
*    eoc         -
*    f220, f221, f311, f321, f322, f330, f441, f442, f522, f523, f542, f543       -
*    g200, g201, g211, g300, g310, g322, g410, g422, g520, g521, g532, g533       -
*    sini2       -
*    temp, temp1 -
*    Theta       -
*    xno2        -
*
*  Coupling      :
*    getgravconst-
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
```

Formatted: German (Germany)

Formatted: French (France)

```
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
*-----------------------------------------------------------------------------

      SUBROUTINE DSINIT( whichconst,
     &                 Cosim , Emsq  , Argpo , S1    , S2    , S3    ,
     &                 S4    , S5    , Sinim , Ss1   , Ss2   , Ss3   ,
     &                 Ss4   , Ss5   , Sz1   , Sz3   , Sz11  , Sz13  ,
     &                 Sz21  , Sz23  , Sz31  , Sz33  , T     , Tc    ,
     &                 GSTo  , Mo    , MDot  , No    , nodeo ,nodeDot,
     &                 XPIDOT, Z1    , Z3    , Z11   , Z13   , Z21   ,
     &                 Z23   , Z31   , Z33   , Ecco  , EccSq ,
     &                 Eccm  , Argpm , Inclm , Mm    , Xn    , nodem,
     &                 IREZ  , Atime , D2201 , D2211 , D3210 , D3222 ,
     &                 D4410 , D4422 , D5220 , D5232 , D5421 , D5433 ,
     &                 Dedt  , Didt  , DMDT  , DNDT  , DNODT , DOMDT ,
     &                 Del1  , Del2  , Del3  , Xfact , Xlamo , Xli   ,
     &                 Xni )
        IMPLICIT NONE
        INTEGER  IRez, whichconst
        REAL*8   Cosim , Emsq  , Argpo , S1    , S2    , S3    , S4    ,
     &          S5    , Sinim , Ss1   , Ss2   , Ss3   , Ss4   , Ss5   ,
     &          Sz1   , Sz3   , Sz11  , Sz13  , Sz21  , Sz23  , Sz31  ,
     &          Sz33  , T     , Tc    , GSTo  , Mo    , MDot  , No    ,
     &          nodeo ,nodeDot,XPIDOT , Z1    , Z3    , Z11   , Z13   ,
     &          Z21   , Z23   , Z31   , Z33   , Eccm  , Argpm , Inclm ,
     &          Mm    , Xn    , nodem , Atime , D2201 , D2211 , D3210 ,
     &          D3222 , D4410 , D4422 , D5220 , D5232 , D5421 , D5433 ,
     &          Dedt  , Didt  , DMDT  , DNDT  , DNODT , DOMDT , Del1  ,
     &          Del2  , Del3  , Xfact , Xlamo , Xli   , Xni   , Ecco  ,
     &          Eccsq

* -------------------------- Local Variables --------------------------
        REAL*8  ainv2 , aonv  , cosisq, eoc   , f220  , f221  , f311  ,
     &          f321  , f322  , f330  , f441  , f442  , f522  , f523  ,
     &          f542  , f543  , g200  , g201  , g211  , g300  , g310  ,
     &          g322  , g410  , g422  , g520  , g521  , g532  , g533  ,
     &          ses   , sgs   , sghl  , sghs  , shs   , shl   , sis   ,
     &          sini2 , sls   , temp  , temp1 , Theta , xno2
        REAL*8  Q22   , Q31   , Q33   , ROOT22, ROOT44, ROOT54,
     &          RPTim , Root32, Root52, X2o3  , XKe   , Znl   ,
     &          Zns,  Emo, emsqo , tumin, mu, radiusearthkm, j2, j3, j4,
     &          j3oj2

        CHARACTER Help
        INCLUDE 'ASTMATH.CMN'
```

Formatted: German (Germany)
Formatted: French
Formatted: German (Germany)
Formatted: German (Germany)

```
      Q22    = 1.7891679D-6
      Q31    = 2.1460748D-6
      Q33    = 2.2123015D-7
      ROOT22 = 1.7891679D-6
      ROOT44 = 7.3636953D-9
      ROOT54 = 2.1765803D-9
      RPTim  = 4.37526908801129966D-3 ! this equates to 7.29211514668855e-5 rad/sec
      Root32 = 3.7393792D-7
      Root52 = 1.1428639D-7
      X2o3   = 2.0D0 / 3.0D0
      ZNL    = 1.5835218D-4
      ZNS    = 1.19459D-5

      ! sgp4fix identify constants and allow alternate values
      CALL getgravconst( whichconst, tumin, mu, radiusearthkm, xke,
     &       j2, j3, j4, j3oj2 )

* ------------------------ DEEP SPACE INITIALIZATION ------------------
      IREZ = 0
      IF ((XN.lt.0.0052359877D0).AND.(XN.GT.0.0034906585D0)) THEN
          IREZ = 1
        ENDIF
      IF ((XN.ge.8.26D-3).AND.(XN.LE.9.24D-3).AND.(Eccm.GE.0.5D0))THEN
          IREZ = 2
        ENDIF

* ---------------------------- DO SOLAR TERMS -------------------------
      SES  =  SS1*ZNS*SS5
      SIS  =  SS2*ZNS*(SZ11 + SZ13)
      SLS  = -ZNS*SS3*(SZ1 + SZ3 - 14.0D0 - 6.0D0*EMSQ)
      SGHS =  SS4*ZNS*(SZ31 + SZ33 - 6.0D0)
      SHS  = -ZNS*SS2*(SZ21 + SZ23)

c       sgp4fix for 180 deg incl
      IF ((Inclm.lt.5.2359877D-2).or.(Inclm.gt.pi-5.2359877D-2)) THEN
          SHS = 0.0D0
        ENDIF
      IF (SINIM.ne.0.0D0) THEN
          SHS = SHS/SINIM
        ENDIF
      SGS  = SGHS - COSIM*SHS

* ----------------------------- DO LUNAR TERMS ------------------------
      DEDT = SES + S1*ZNL*S5
      DIDT = SIS + S2*ZNL*(Z11 + Z13)
```

Formatted: French (France)

```
      DMDT = SLS - ZNL*S3*(Z1 + Z3 - 14.0D0 - 6.0D0*EMSQ)
      SGHL = S4*ZNL*(Z31 + Z33 - 6.0D0)
      SHL  = -ZNL*S2*(Z21 + Z23)

c      sgp4fix for 180 deg incl
      IF ((Inclm.lt.5.2359877D-2).or.(Inclm.gt.pi-5.2359877D-2)) THEN
         SHL = 0.0D0
       ENDIF
      DOMDT= SGS+SGHL
      DNODT= SHS
      IF (SINIM .ne. 0.0D0) THEN
         DOMDT = DOMDT-COSIM/SINIM*SHL
         DNODT = DNODT+SHL/SINIM
      ENDIF

* --------------- CALCULATE DEEP SPACE RESONANCE EFFECTS --------------
      DNDT  = 0.0D0
      THETA = DMOD(GSTo + TC*RPTIM,TwoPi)
      Eccm  = Eccm + DEDT*T
      emsq  = eccm**2
      Inclm = Inclm + DIDT*T
      Argpm = Argpm + DOMDT*T
      nodem = nodem + DNODT*T
      Mm    = Mm + DMDT*T


* ------------------ Initialize the resonance terms -------------------
      IF (IREZ .ne. 0) THEN
         AONV = (XN/XKE)**X2O3

* -------------- GEOPOTENTIAL RESONANCE FOR 12 HOUR ORBITS ------------
      IF (IREZ .eq. 2) THEN
         COSISQ = COSIM*COSIM
         emo    = Eccm
         emsqo  = emsq
         Eccm   = ecco
         emsq   = eccsq
         EOC    = Eccm*EMSQ
         G201   = -0.306D0-(Eccm-0.64D0)*0.440D0
         IF (Eccm.le.0.65D0) THEN
            G211 =   3.616D0 -  13.2470D0*Eccm +  16.2900D0*EMSQ
            G310 = -19.302D0 + 117.3900D0*Eccm - 228.4190D0*EMSQ +
     &             156.591D0*EOC
            G322 = -18.9068D0+ 109.7927D0*Eccm - 214.6334D0*EMSQ +
     &             146.5816D0*EOC
            G410 = -41.122D0 + 242.6940D0*Eccm - 471.0940D0*EMSQ +
```

Formatted: German (Germany)

```
&           313.953D0*EOC
       G422 =-146.407D0 + 841.8800D0*Eccm - 1629.014D0*EMSQ +
&          1083.435D0*EOC
       G520 =-532.114D0 + 3017.977D0*Eccm - 5740.032D0*EMSQ +
&          3708.276D0*EOC
      ELSE
       G211 =  -72.099D0 +  331.819D0*Eccm -  508.738D0*EMSQ +
&           266.724D0*EOC
       G310 = -346.844D0 + 1582.851D0*Eccm - 2415.925D0*EMSQ +
&           1246.113D0*EOC
       G322 = -342.585D0 + 1554.908D0*Eccm - 2366.899D0*EMSQ +
&           1215.972D0*EOC
       G410 =-1052.797D0 + 4758.686D0*Eccm - 7193.992D0*EMSQ +
&           3651.957D0*EOC
       G422 =-3581.690D0 + 16178.11D0*Eccm - 24462.77D0*EMSQ +
&          12422.52D0*EOC
       IF (Eccm.gt.0.715D0) THEN
         G520 =-5149.66D0 + 29936.92D0*Eccm -54087.36D0*EMSQ
&            + 31324.56D0*EOC
        ELSE
         G520 = 1464.74D0 -  4664.75D0*Eccm + 3763.64D0*EMSQ
        ENDIF
      ENDIF
     IF (Eccm.lt.0.7D0) THEN
       G533 = -919.22770D0 + 4988.6100D0*Eccm-9064.7700D0*EMSQ
&         + 5542.21D0*EOC
       G521 = -822.71072D0 + 4568.6173D0*Eccm-8491.4146D0*EMSQ
&         + 5337.524D0*EOC
       G532 = -853.66600D0 + 4690.2500D0*Eccm-8624.7700D0*EMSQ
&         + 5341.4D0*EOC
      ELSE
       G533 =-37995.780D0 + 161616.52D0*Eccm-229838.20D0*EMSQ+
&         109377.94D0*EOC
       G521 =-51752.104D0 + 218913.95D0*Eccm-309468.16D0*EMSQ+
&         146349.42D0*EOC
       G532 =-40023.880D0 + 170470.89D0*Eccm-242699.48D0*EMSQ+
&         115605.82D0*EOC
      ENDIF
     SINI2 =  SINIM*SINIM
     F220  =  0.75D0* (1.0D0+2.0D0*COSIM+COSISQ)
     F221  =  1.5D0*SINI2
     F321  =  1.875D0*SINIM * (1.0D0-2.0D0*COSIM-3.0D0*COSISQ)
     F322  = -1.875D0*SINIM * (1.0D0+2.0D0*COSIM-3.0D0*COSISQ)
     F441  = 35.0D0*SINI2*F220
     F442  = 39.3750D0*SINI2*SINI2
     F522  =  9.84375D0*SINIM * (SINI2* (1.0D0-2.0D0*COSIM-
```

Formatted: French (France)

```
     &            5.0D0*COSISQ)+0.33333333D0 * (-2.0D0+4.0D0*COSIM+
     &            6.0D0*COSISQ) )
        F523  =  SINIM * (4.92187512D0*SINI2 * (-2.0D0-4.0D0*COSIM+
     &            10.0D0*COSISQ) + 6.56250012D0*
     &            (1.0D0+2.0D0*COSIM-3.0D0*COSISQ))
        F542  =  29.53125D0*SINIM * (2.0D0-8.0D0*COSIM+COSISQ*
     &            (-12.0D0+8.0D0*COSIM+10.0D0*COSISQ) )
        F543  = 29.53125D0*SINIM * (-2.0D0-8.0D0*COSIM+COSISQ*
     &            (12.0D0+8.0D0*COSIM-10.0D0*COSISQ) )

        XNO2   =  XN * XN
        AINV2  =  AONV * AONV
        TEMP1  =  3.0D0*XNO2*AINV2
        TEMP   =  TEMP1*ROOT22
        D2201  =  TEMP*F220*G201
        D2211  =  TEMP*F221*G211
        TEMP1  =  TEMP1*AONV
        TEMP   =  TEMP1*ROOT32
        D3210  =  TEMP*F321*G310
        D3222  =  TEMP*F322*G322
        TEMP1  =  TEMP1*AONV
        TEMP   =  2.0D0*TEMP1*ROOT44
        D4410  =  TEMP*F441*G410
        D4422  =  TEMP*F442*G422
        TEMP1  =  TEMP1*AONV
        TEMP   =  TEMP1*ROOT52
        D5220  =  TEMP*F522*G520
        D5232  =  TEMP*F523*G532
        TEMP   =  2.0D0*TEMP1*ROOT54
        D5421  =  TEMP*F542*G521
        D5433  =  TEMP*F543*G533
        XLAMO  =  DMOD(Mo+nodeo+nodeo-THETA-THETA,TwoPi)
        XFACT  = MDot + DMDT + 2.0D0 * (nodeDot+DNODT-RPTIM) - No

        Eccm = emo
        emsq = emsqo
       ENDIF

      IF (Irez .eq. 1) THEN
* -------------------- SYNCHRONOUS RESONANCE TERMS --------------------
        G200  = 1.0D0 + EMSQ * (-2.5D0+0.8125D0*EMSQ)
        G310  = 1.0D0 + 2.0D0*EMSQ
        G300  = 1.0D0 + EMSQ * (-6.0D0+6.60937D0*EMSQ)
        F220  = 0.75D0 * (1.0D0+COSIM) * (1.0D0+COSIM)
        F311  = 0.9375D0*SINIM*SINIM*
     &            (1.0D0+3.0D0*COSIM) - 0.75D0*(1.0D0+COSIM)
```

```
          F330  = 1.0D0+COSIM
          F330  = 1.875D0*F330*F330*F330
          DEL1  = 3.0D0*XN*XN*AONV*AONV
          DEL2  = 2.0D0*DEL1*F220*G200*Q22
          DEL3  = 3.0D0*DEL1*F330*G300*Q33*AONV
          DEL1  = DEL1*F311*G310*Q31*AONV
          XLAMO = DMOD(Mo+nodeo+Argpo-THETA,TwoPi)
          XFACT = MDot + XPIDOT - RPTIM + DMDT + DOMDT + DNODT - No
        ENDIF

* ---------------- FOR SGP4, INITIALIZE THE INTEGRATOR ----------------
        XLI   = XLAMO
        XNI   = No
        ATIME = 0.0D0
        XN    = No + DNDT
      ENDIF ! Ires non-zero

      RETURN
      END  ! end dsinit


* -----------------------------------------------------------------------------
*
*                           SUBROUTINE DSPACE
*
*  This Subroutine provides deep space contributions to mean elements for
*    perturbing third body.  these effects have been averaged over one
*    revolution of the sun and moon.  for earth resonance effects, the
*    effects have been averaged over no revolutions of the satellite.
*    (mean motion)
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  inputs        :
*    d2201, d2211, d3210, d3222, d4410, d4422, d5220, d5232, d5421, d5433       -
*    dedt        -
*    del1, del2, del3  -
*    didt        -
*    dmdt        -
*    dnodt       -
*    domdt       -
*    irez        - flag for resonance           0-none, 1-one day, 2-half day
*    argpo       - argument of perigee
*    argpdot     - argument of perigee dot (rate)
*    t           - time
*    tc          -
```

Formatted: German (Germany)

Formatted: German (Germany)

```
*     gsto        - gst
*     xfact       -
*     xlamo       -
*     no          - mean motion
*     atime       -
*     em          - eccentricity
*     ft          -
*     argpm       - argument of perigee
*     inclm       - inclination
*     xli         -
*     mm          - mean anomaly
*     xni         - mean motion
*     nodem       - right ascension of ascending node
*
*   outputs       :
*     atime       -
*     em          - eccentricity
*     argpm       - argument of perigee
*     inclm       - inclination
*     xli         -
*     mm          - mean anomaly
*     xni         -
*     nodem       - right ascension of ascending node
*     dndt        -
*     nm          - mean motion
*
*
*   coupling      :
*     none        -
*
*   references    :
*     hoots, roehrich, norad spacetrack report #3 1980
*     hoots, norad spacetrack report #6 1986
*     hoots, schumacher and glover 2004
*     vallado, crawford, hujsak, kelso  2006
*------------------------------------------------------------------------------

      SUBROUTINE DSPACE( IRez  , D2201 , D2211 , D3210 , D3222 , D4410 ,
     &                   D4422 , D5220 , D5232 , D5421 , D5433 , Dedt  ,
     &                   Del1  , Del2  , Del3  , Didt  , Dmdt  , Dnodt ,
     &                   Domdt , Argpo , ArgpDot, T    , TC    , GSTo  ,
     &                   Xfact , Xlamo , No    ,
     &                   Atime , Eccm  , Argpm , Inclm , Xli   , Mm  ,
     &                   XNi   , nodem, Dndt  , XN  )
        IMPLICIT NONE
        INTEGER  IRez
```

```
      Real*8   D2201 , D2211 , D3210 , D3222 , D4410 , D4422 , D5220 ,
     &         D5232 , D5421 , D5433 , Dedt  , Del1  , Del2  , Del3  ,
     &         Didt  , Dmdt  , Dnodt , Domdt , Argpo , ArgpDot,T     ,
     &         TC    , GSTo  , Xfact , Xlamo , No    , Atime , Eccm  ,
     &         Argpm , Inclm , Xli   , Mm    , Xni   , nodem, Dndt  ,
     &         XN

* -------------------------- Local Variables --------------------------
      INTEGER  iretn , iret
      REAL*8   Delt  , Ft    , theta , x2li  , x2omi , xl    , xldot ,
     &         xnddt , xndt  , xomi
      REAL*8   G22   , G32   , G44   , G52   , G54   , Fasx2 ,
     &         Fasx4 , Fasx6 , RPtim , Step2 , Stepn , Stepp

      COMMON /DebugHelp/ Help
      CHARACTER Help
      INCLUDE 'ASTMATH.CMN'

* ----------------------------- Constants -----------------------------
      FASX2 = 0.13130908D0
      FASX4 = 2.8843198D0
      FASX6 = 0.37448087D0
      G22   = 5.7686396D0
      G32   = 0.95240898D0
      G44   = 1.8014998D0
      G52   = 1.0508330D0
      G54   = 4.4108898D0
      RPTIM = 4.37526908801129966D-3
      STEPP =    720.0D0
      STEPN =   -720.0D0
      STEP2 = 259200.0D0

* --------------- CALCULATE DEEP SPACE RESONANCE EFFECTS --------------
      DNDT   = 0.0D0
      THETA = DMOD(GSTo + TC*RPTIM,TwoPi)
      Eccm  = Eccm + DEDT*T

      Inclm = Inclm + DIDT*T
      Argpm = Argpm + DOMDT*T
      nodem = nodem + DNODT*T
      Mm    = Mm + DMDT*T

c   sgp4fix for negative inclinations
c   the following if statement should be commented out
c       IF(Inclm .lt. 0.0D0) THEN
c           Inclm  = -Inclm
```

Formatted: French (France)

Formatted: German (Germany)

Formatted: German (Germany)

```
c          Argpm  = Argpm-PI
c          nodem = nodem+PI
c        ENDIF
```

Formatted: German (Germany)

```
c   sgp4fix for propagator problems
c   the following integration works for negative time steps and periods
c   the specific changes are unknown because the original code was so convoluted
c      sgp4fix take out atime = 0.0 and fix for faster operation
       Ft    = 0.0D0      ! Just in case - should be set in loops if used.

       IF (IREZ .ne. 0) THEN
* ----- UPDATE RESONANCES : NUMERICAL (EULER-MACLAURIN)
INTEGRATION ---
* ---------------------------- EPOCH RESTART --------------------------
        ! sgp4fix streamline check
        IF ((atime .eq. 0.0D0) .or. (t * atime .le. 0.0D0) .or.
     &      (dabs(t) .lt. dabs(atime)) ) THEN
            atime  = 0.0D0
            xni    = no
            xli    = xlamo
          ENDIF
         ! sgp4fix move check outside loop
         IF (t .gt. 0.0D0) THEN
            delt = stepp
          else
            delt = stepn
          ENDIF

          iretn = 381 ! added for do loop
          iret  =   0 ! added for loop
          DO WHILE (IRetn.eq.381)

* --------------------------- DOT TERMS CALCULATED --------------------
* ------------------- NEAR - SYNCHRONOUS RESONANCE TERMS --------------
          IF (IREZ .ne. 2) THEN
              XNDT  = DEL1*DSIN(XLI-FASX2) +
     &              DEL2*DSIN(2.0D0*(XLI-FASX4)) +
     &              DEL3*DSIN(3.0D0*(XLI-FASX6))
              XLDOT = XNI + XFACT
              XNDDT = DEL1*DCOS(XLI-FASX2) +
     &         2.0D0*DEL2*DCOS(2.0D0*(XLI-FASX4)) +
     &         3.0D0*DEL3*DCOS(3.0D0*(XLI-FASX6))
              XNDDT = XNDDT*XLDOT
            ELSE
```

Formatted: German (Germany)

```
* -------------------- NEAR - HALF-DAY RESONANCE TERMS ---------------
```

```
            XOMI = Argpo + ArgpDot*ATIME
            X2OMI= XOMI + XOMI
            X2LI = XLI + XLI
            XNDT = D2201*DSIN(X2OMI+XLI-G22) + D2211*DSIN(XLI-G22) +
     &           D3210*DSIN( XOMI+XLI-G32) +
     &           D3222*DSIN(-XOMI+XLI-G32) +
     &           D4410*DSIN(X2OMI+X2LI-G44)+ D4422*DSIN(X2LI-G44)+
     &           D5220*DSIN( XOMI+XLI-G52) +
     &           D5232*DSIN(-XOMI+XLI-G52) +
     &           D5421*DSIN( XOMI+X2LI-G54)+
     &           D5433*DSIN(-XOMI+X2LI-G54)
            XLDOT = XNI+XFACT
            XNDDT = D2201*DCOS(X2OMI+XLI-G22) + D2211*DCOS(XLI-G22)+
     &            D3210*DCOS( XOMI+XLI-G32) +
     &            D3222*DCOS(-XOMI+XLI-G32) +
     &            D5220*DCOS( XOMI+XLI-G52) +
     &            D5232*DCOS(-XOMI+XLI-G52) +
     &            2.0D0*(D4410*DCOS(X2OMI+X2LI-G44) +
     &            D4422*DCOS(X2LI-G44) +
     &            D5421*DCOS( XOMI+X2LI-G54) +
     &            D5433*DCOS(-XOMI+X2LI-G54))
            XNDDT = XNDDT*XLDOT
          ENDIF

* ------------------------------- INTEGRATOR --------------------------
          !  sgp4fix move end checks to end of routine
          IF (DABS(T-ATIME).ge.STEPP) THEN
              IRET  = 0
              IRETN = 381
            ELSE
              FT    = T-ATIME
              IRETN = 0
            ENDIF

          IF (IRETN.EQ.381) THEN
              XLI   = XLI + XLDOT*DELT + XNDT*STEP2
              XNI   = XNI + XNDT*DELT + XNDDT*STEP2
              ATIME = ATIME + DELT
            ENDIF

          ENDDO

        XN = XNI + XNDT*FT  + XNDDT*FT*FT*0.5D0
        XL = XLI + XLDOT*FT + XNDT*FT*FT*0.5D0
        IF(IREZ .ne. 1) THEN
            Mm   = XL-2.0D0*nodem+2.0D0*THETA
```

Formatted: German (Germany)

Formatted: French (France)

Formatted: German (Germany)

Formatted: French (France)

```
          DNDT = XN-No
        ELSE
          Mm   = XL-nodem-Argpm+THETA
          DNDT = XN-No
        ENDIF

      XN = No + DNDT
    ENDIF

   RETURN
   END  ! end dspace

* ---------------------------------------------------------------------------
*
*                        SUBROUTINE JDay1
*
*  This subroutine finds the Julian date given the Year, Month, Day, and Time.
*
*  Author       : David Vallado                719-573-2600    1 Mar 2001
*
*  Inputs         Description                    Range / Units
*    Year        - Year                        1900 .. 2100
*    Mon         - Month                         1 .. 12
*    Day         - Day                         1 .. 28,29,30,31
*    Hr          - Universal Time Hour           0 .. 23
*    Min         - Universal Time Min            0 .. 59
*    Sec         - Universal Time Sec          0.0D0 .. 59.999D0
*    WhichType   - Julian .or. Gregorian calender   'J' .or. 'G'
*
*  Outputs      :
*    JD          - Julian Date                  days from 4713 BC
*
*  Locals       :
*    B           - Var to aid Gregorian dates
*
*  Coupling     :
*    None.
*
*  References   :
*    Vallado       2007, 189, Alg 14, Ex 3-14
* ---------------------------------------------------------------------------

   SUBROUTINE JDAY1        ( Year,Mon,Day,Hr,Min, Sec, JD )
     IMPLICIT NONE
     INTEGER Year, Mon, Day, Hr, Min
     REAL*8  Sec, JD
```

```
    ! --------------------  Implementation   ----------------------
    JD= 367.0D0 * Year
   &       - INT( (7* (Year+INT ( (Mon+9)/12.0) ) ) * 0.25D0 )
   &       + INT( 275*Mon / 9.0 )
   &       + Day + 1721013.5D0
   &       + ( (Sec/60.0D0 + Min ) / 60.0D0 + Hr ) / 24.0D0
*   &      - 0.5D0*DSIGN(1.0D0, 100.0D0*Year + Mon - 190002.5D0) + 0.5D0
   RETURN
   END  ! end jday1


* ---------------------------------------------------------------------------
*
*                        SUBROUTINE DAYS2MDHMS
*
*  This subroutine converts the day of the year, days, to the equivalent month
*    day, hour, Minute and second.
*
*  Algorithm     : Set up array for the Number of days per month
*                  Find Leap Year - be sure to account for the 400 years
*                  Loop through a Temp value for WHILE the value is .lt. the days
*                  Perform INTEGER conversions to the correct day and month
*                  Convert remainder into H M S using type conversions
*
*  Author        : David Vallado                  719-573-2600    1 Mar 2001
*
*  Inputs          Description                    Range / Units
*    Year        - Year                           +1900 .. 2100+
*    Days        - Julian Day of the year         0.0D0  .. 366.0D0
*
*  OutPuts       :
*    Mon         - Month                          1 .. 12
*    Day         - Day                            1 .. 28,29,30,31
*    Hr          - Hour                           0 .. 23
*    Min         - Minute                         0 .. 59
*    Sec         - Second                         0.0D0 .. 59.999D0
*
*  Locals        :
*    DayofYr     - Day of year
*    Temp        - Temporary REAL*8 values
*    IntTemp     - Temporary INTEGER value
*    i           - Index
*    LMonth[12]  - INTEGER Array containing the Number of days per month
*
*  Coupling      :
```

Formatted: French (France)

```
*    None.
* ---------------------------------------------------------------------------

     SUBROUTINE DAYS2MDHMS  ( Year,Days,  Mon,Day,Hr,Min,Sec )
       IMPLICIT NONE
       REAL*8 Days,Sec
       INTEGER Year, Mon, Day, Hr, Min
* ---------------------------  Locals  -------------------------------
       INTEGER IntTemp,i,DayofYr, LMonth(12)
       REAL*8 Temp

       ! --------------------  Implementation   ----------------------
       ! -------------- Set up array of days in month  ---------------
       DO i = 1,12
           LMonth(i) = 31
         ENDDO
       LMonth( 2) = 28
       LMonth( 4) = 30
       LMonth( 6) = 30
       LMonth( 9) = 30
       LMonth(11) = 30

       DayofYr= IDINT(Days )

       ! ---------------- Find month and Day of month ----------------
       IF (MOD(Year,4).eq.0) THEN
           LMonth(2)= 29
         ENDIF
       i= 1
       IntTemp= 0
       DO WHILE ( (DayofYr.gt.IntTemp + LMonth(i) ) .and. ( i.lt.12 ))
           IntTemp= IntTemp + LMonth(i)
           i= i+1
         ENDDO
       Mon= i
       Day= DayofYr - IntTemp

       ! ---------------- Find hours Minutes and seconds -------------
       Temp= (Days - DayofYr )*24.0D0
       Hr  = IDINT( Temp )
       Temp= (Temp-Hr) * 60.0D0
       Min = IDINT( Temp )
       Sec = (Temp-Min) * 60.0D0

     RETURN
     END  ! end days2mdhms
```

Formatted: French (France)

```
****************************************************8
    Subroutine HMS (EHour, EHr, EMin, ESec)

    Implicit None

    Real*8 EHour, TempMin1, TempMin2, TempSec, ESec
    Integer EHr, EMin

    EHr = Int(EHour)
    TempMin1=EHour-EHr
    TempMin2=Tempmin1*60.0d0
    EMin= Int(TempMin2)
    TempSec = TempMin2-EMin
    ESec = TempSec*60.0D0

    End
```

Formatted: French (France)

A.2 Other Codes

Attached on the accompanying CD, there are other codes in their entirety. These include:

- PREDICT (Feb. Version) with updated SGP4 propagator.
- MATLAB version of Vallado's SGP4 propagator.
- FORTRAN version of Vallado's SGP4 propagator.

Additionally, the analysis files are also provided on the CD.

APPENDIX B. Verification TLE File

```
#   ------------------ Verification test cases ----------------------
#                       # TEME example
1 00005U 58002B   00179.78495062  .00000023  00000-0  28098-4 0  4753
2 00005  34.2682 348.7242 1859667 331.7664  19.3264 10.82419157413667     0.00      4320.0        360.00
#                       ## fig show lyddane fix error with gsfc ver
1 04632U 70093B   04031.91070959 -.00000084  00000-0  10000-3 0  9955
2 04632  11.4628 273.1101 1450506 207.6000 143.9350  1.20231981 44145  -5184.0     -4896.0        120.00
#   DELTA 1 DEB         # near earth normal drag equation
#                       # perigee = 377.26km, so moderate drag case
1 06251U 62025E   06176.82412014  .00008885  00000-0  12808-3 0  3985
2 06251  58.0579  54.0425 0030035 139.1568 221.1854 15.56387291  6774      0.0      2880.0        120.00
#   MOLNIYA 2-14              # 12h resonant ecc in 0.65 to 0.7 range
1 08195U 75081A   06176.33215444  .00000099  00000-0  11873-3 0   813
2 08195  64.1586 279.0717 6877146 264.7651  20.2257  2.00491383225656      0.0      2880.0        120.00
#   MOLNIYA 1-36              ## fig 12h resonant ecc in 0.7 to 0.715 range
1 09880U 77021A   06176.56157475  .00000421  00000-0  10000-3 0  9814
2 09880  64.5968 349.3786 7069051 270.0229  16.3320  2.00813614112380      0.0      2880.0        120.00
#   SMS 1 AKM           # show the integrator problem with gsfc ver
1 09998U 74033F   05148.79417928 -.00000112  00000-0  00000+0 0  4480
2 09998   9.4958 313.1750 0270971 327.5225  30.8097  1.16186785 45878  -1440.0      -720.00        60.0
#                       # Original STR#3 SDP4 test
1 11801U          80230.29629788  .01431103  00000-0  14311-1      13
2 11801  46.7916 230.4354 7318036  47.4722  10.4117  2.28537848    13      0.0      1440.0        360.00
#   EUTELSAT 1-F1 (ECS1)## fig lyddane choice in GSFC at 2080 min
1 14128U 83058A   06176.02844893 -.00000158  00000-0  10000-3 0  9627
2 14128  11.4384  35.2134 0011562  26.4582 333.5652  0.98870114 46093      0.0      2880.0        120.00
#   SL-6 R/B(2)         # Deep space, perigee = 82.48 (<98) for
#                       # s4 > 20 mod
1 16925U 86065D   06151.67415771  .02550794 -30915-6  18784-3 0  4486
2 16925  62.0906 295.0239 5596327 245.1593  47.9690  4.88511875148616      0.0      1440.0        120.00
#   SL-12 R/B           # Shows Lyddane choice at 1860 and 4700 min
1 20413U 83020D   05363.79166667  .00000000  00000-0  00000+0 0  7041
2 20413  12.3514 187.4253 7864447 196.3027 356.5478  0.24690082  7978   1440.0      4320.0        120.00
#   MOLNIYA 1-83              # 12h resonant, ecc > 0.715 (negative BSTAR)
1 21897U 92011A   06176.02341244 -.00001273  00000-0 -13525-3 0  3044
2 21897  62.1749 198.0096 7421690 253.0462  20.1561  2.01269994104880      0.0      2880.0        120.00
#   SL-6 R/B(2)         # last tle given, decayed 2006-04-04, day 94
1 22312U 93002D   06094.46235912  .99999999  81888-5  49949-3 0  3953
2 22312  62.1486  77.4698 0308723 267.9229  88.7392 15.95744531 98783  54.2028672   1440.0        20.00
#   SL-6 R/B(2)         # 12h resonant ecc in the > 0.715 range
1 22674U 93035D   06176.55909107  .00002121  00000-0  29868-3 0  6569
2 22674  63.5035 354.4452 7541712 253.3264  18.7754  1.96679808 93877      0.0      2880.0        120.00
#   ARIANE 44L+ R/B     # Lyddane bug at <= 70 min for atan2(),
#                       # no quadrant fix
1 23177U 94040C   06175.45752052  .00000386  00000-0  76590-3 0    95
2 23177   7.0496 179.8238 7258491 296.0482   8.3061  2.25906668 97438      0.0      1440.0        120.00
#   WIND                      # STR#3 Kepler failes past about 200 min
1 23333U 94071A   94305.49999999 -.00172956  26967-3  10000-3 0    15
2 23333  28.7490   2.3720 9728298  30.4360   1.3500  0.07309491    70      0.0      1600.0        120.00
#   ARIANE 42P+3 R/B    ## fig Lyddane bug at > 280.5 min for AcTan()
1 23599U 95029B   06171.76535463  .00085586  12891-6  12956-2 0  2905
2 23599   6.9327   0.2849 5782022 274.4436  25.2425  4.47796565123555      0.0       720.0         20.00
#   ITALSAT 2           # 24h resonant GEO, inclination > 3 deg
1 24208U 96044A   06177.04061740 -.00000094  00000-0  10000-3 0  1600
2 24208   3.8536  80.0121 0026640 311.0977  48.3000  1.00778054 36119      0.0      1440.0        120.00
#   AMC-4               ## fig low incl, show incl shift with
#                       ## gsfc version from 240 to 1440 min
1 25954U 99060A   04039.68057285 -.00000108  00000-0  00000-0 0  6847
```

```
2 25954   0.0004 243.8136 0001765  15.5294  22.7134  1.00271289 15615  -1440.0      1440.0        120.00
#   INTELSAT 902                # negative incl at 9313 min then
#                     # 270 deg Lyddane bug at 37606 min
1 26900U 01039A   06106.74503247  .00000045  00000-0  10000-3 0  8290
2 26900   0.0164 266.5378 0003319  86.1794 182.2590  1.00273847 16981   9300.00     9400.00        60.00
#   COSMOS 1024 DEB     # 12h resonant ecc in 0.5 to 0.65 range
1 26975U 78066F   06174.85818871  .00000620  00000-0  10000-3 0  6809
2 26975  68.4714 236.1303 5602877 123.7484 302.5767  2.05657553 67521       0.0      2880.0        120.00
#   CBERS 2                     # Near Earth, ecc = 8.84E-5 (< 1.0e-4)
#                     # drop certain normal drag terms
1 28057U 03049A   06177.78615833  .00000060  00000-0  35940-4 0  1836
2 28057  98.4283 247.6961 0000884  88.1964 271.9322 14.35478080140550      0.0      2880.0        120.00
#   NAVSTAR 53 (USA 175)# 12h non-resonant GPS (ecc < 0.5 ecc)
1 28129U 03058A   06175.57071136 -.00000104  00000-0  10000-3 0   459
2 28129  54.7298 324.8098 0048506 266.2640  93.1663  2.00562768 18443      0.0      1440.0        120.00
#   COSMOS 2405         # Near Earth, perigee = 127.20 (< 156) s4 mod
1 28350U 04020A   06167.21788666  .16154492  76267-5  18678-3 0  8894
2 28350  64.9977 345.6130 0024870 260.7578  99.9590 16.47856722116490      0.0      2880.0        120.00
#   H-2 R/B             # Deep space, perigee = 135.75 (<156) s4 mod
1 28623U 05006B   06177.81079184  .00637644  69054-6  96390-3 0  6000
2 28623  28.5200 114.9834 6249053 170.2550 212.8965  3.79477162 12753      0.0      1440.0        120.00
#   XM-3                        # 24h resonant geo, incl < 3 deg goes
#                     # negative around 1130 min
1 28626U 05008A   06176.46683397 -.00000205  00000-0  10000-3 0  2190
2 28626   0.0019 286.9433 0000335  13.7918  55.6504  1.00270176  4891       0.0      1440.0        120.00
#   MINOTAUR R/B        # Sub-orbital case - Decayed 2005-11-29
#                     #(perigee = -51km), lost in 50 minutes
1 28872U 05037B   05333.02012661  .25992681  00000-0  24476-3 0  1534
2 28872  96.4736 157.9986 0303955 244.0492 110.6523 16.46015938 10708      0.0        50.0          5.00
#   SL-14 DEB           # Last stage of decay - lost in under 420 min
1 29141U 85108AA  06170.26783845  .99999999  00000-0  13519-0 0   718
2 29141  82.4288 273.4882 0015848 277.2124  83.9133 15.93343074  6828      0.0       440.0         20.00
#   SL-12 DEB           # Near Earth, perigee = 212.24 < 220
#                     # simplified drag eq
1 29238U 06022G   06177.28732010  .00766286  10823-4  13334-2 0   101
2 29238  51.5595 213.7903 0202579  95.2503 267.9010 15.73823839  1061      0.0      1440.0        120.00
#                     # Original STR#3 SGP4 test
1 88888U          80275.98708465  .00073094  13844-3  66816-4 0    87
2 88888  72.8435 115.9689 0086731  52.6988 110.5714 16.05824518  1058      0.0      1440.0        120.00
#
```

APPENDIX C – Verification Results Data

5 xx

0.00000000	7022.46529266	-1400.08296755	0.03995155	1.893841015	6.405893759	4.534807250
360.00000000	-7154.03120202	-3783.17682504	-3536.19412294	4.741887409	-4.151817765	-2.093935425
720.00000000	-7134.59340119	6531.68641334	3260.27186483	-4.113793027	-2.911922039	-2.557327851
1080.00000000	5568.53901181	4492.06992591	3863.87641983	-4.209106476	5.159719888	2.744852980
1440.00000000	-938.55923943	-6268.18748831	-4294.02924751	7.536105209	-0.427127707	0.989878080
1800.00000000	-9680.56121728	2802.47771354	124.10688038	-0.905874102	-4.659467970	-3.227347517
2160.00000000	190.19796988	7746.96653614	5110.00675412	-6.112325142	1.527008184	-0.139152358
2520.00000000	5579.55640116	-3995.61396789	-1518.82108966	4.767927483	5.123185301	4.276837355
2880.00000000	-8650.73082219	-1914.93811525	-3007.03603443	3.067165127	-4.828384068	-2.515322836
3240.00000000	-5429.79204164	7574.36493792	3747.39305236	-4.999442110	-1.800561422	-2.229392830
3600.00000000	6759.04583722	2001.58198220	2783.55192533	-2.180993947	6.402085603	3.644723952
3960.00000000	-3791.44531559	-5712.95617894	-4533.48630714	6.668817493	-2.516382327	-0.082384354
4320.00000000	-9060.47373569	4658.70952502	813.68673153	-2.232832783	-4.110453490	-3.157345433

4632 xx

0.00000000	2334.11450085	-41920.44035349	-0.03867437	2.826321032	-0.065091664	0.570936053
-5184.00000000	-29020.02587128	13819.84419063	-5713.33679183	-1.768068390	-3.235371192	-0.395206135
-5064.00000000	-32982.56870101	-11125.54996609	-6803.28472771	0.617446996	-3.379240041	0.085954707
-4944.00000000	-22097.68730513	-31583.13829284	-4836.34329328	2.230597499	-2.166594667	0.426443070
-4896.00000000	-15129.94694545	-36907.74526221	-3487.56256701	2.581167187	-1.524204737	0.504805763

6251 xx

0.00000000	3988.31022699	5498.96657235	0.90055879	-3.290032738	2.357652820	6.496623475
120.00000000	-3935.69800083	409.10980837	5471.33577327	-3.374784183	-6.635211043	-1.942056221
240.00000000	-1675.12766915	-5683.30432352	-3286.21510937	5.282496925	1.508674259	-5.354872978
360.00000000	4993.62642836	2890.54969900	-3600.40145627	0.347333429	5.707031557	5.070699638
480.00000000	-1115.07959514	4015.11691491	5326.99727718	-5.524279443	-4.765738774	2.402255961
600.00000000	-4329.10008198	-5176.70287935	409.65313857	2.858408303	-2.933091792	-6.509690397
720.00000000	3692.60030028	-976.24265255	-5623.36447493	3.897257243	6.415554948	1.429112190
840.00000000	2301.83510037	5723.92394553	2814.61514580	-5.110924966	-0.764510559	5.662120145
960.00000000	-4990.91637950	-2303.42547880	3920.86335598	-0.993439372	-5.967458360	-4.759110856
1080.00000000	642.27769977	-4332.89821901	-5183.31523910	5.720542579	4.216573838	-2.846576139
1200.00000000	4719.78335752	4798.06938996	-943.58851062	-2.294860662	3.492499389	6.408334723
1320.00000000	-3299.16993602	1576.83168320	5678.67840638	-4.460347074	-6.202025196	-0.885874586
1440.00000000	-2777.14682335	-5663.16031708	-2462.54889123	4.915493146	0.123328992	-5.896495091
1560.00000000	4992.31573893	1716.62356770	-4287.86065581	1.640717189	6.071570434	4.338797931
1680.00000000	-8.22384755	4662.21521668	4905.66411857	-5.891011274	-3.593173872	3.365100460
1800.00000000	-4966.20137963	-4379.59155037	1349.33347502	1.763172581	-3.981456387	-6.343279443
1920.00000000	2954.49390331	-2080.65984650	-5754.75038057	4.895893306	5.858184322	0.375474825
2040.00000000	3363.28794321	5559.55841180	1956.05542266	-4.587378863	0.591943403	6.107838605
2160.00000000	-4856.66780070	-1107.03450192	4557.21258241	-2.304158557	-6.186437070	-3.956549542
2280.00000000	-497.84480071	-4863.46005312	-4700.81211217	5.960065407	2.996683369	-3.767123329
2400.00000000	5241.61936096	3910.75960683	-1857.93473952	-1.124834806	4.406213160	6.148161299
2520.00000000	-2451.38045953	2610.60463261	5729.79022069	-5.366560525	-5.500855666	0.187958716
2640.00000000	-3791.87520638	-5378.82851382	-1575.82737930	4.266273592	-1.199162551	-6.276154080
2760.00000000	4730.53958356	524.05006433	-4857.29369725	2.918056288	6.135412849	3.495115636
2880.00000000	1159.27802897	5056.60175495	4353.49418579	-5.968060341	-2.314790406	4.230722669

8195 xx

0.00000000	2349.89483350	-14785.93811562	0.02119378	2.721488096	-3.256811655	4.498416672
120.00000000	15223.91713658	-17852.95881713	25280.39558224	1.079041732	0.875187372	2.485682813
240.00000000	19752.78050009	-8600.07130962	37522.72921090	0.238105279	1.546110924	0.986410447

360.00000000	19089.29762968	3107.89495018	39958.14661370	-0.410308034	1.640332277	-0.306873818
480.00000000	13829.66070574	13977.39999817	32736.32082508	-1.065096849	1.279983299	-1.760166075
600.00000000	3333.05838525	18395.31728674	12738.25031238	-1.882432221	-0.611623333	-4.039586549
720.00000000	2622.13222207	-15125.15464924	474.51048398	2.688287199	-3.078426664	4.494979530
840.00000000	15320.56770017	-17777.32564586	25539.53198382	1.064346229	0.892184771	2.459822414
960.00000000	19769.70267785	-8458.65104454	37624.20130236	0.229304396	1.550363884	0.966993056
1080.00000000	19048.56201523	3260.43223119	39923.39143967	-0.418015536	1.639346953	-0.326094840
1200.00000000	13729.19205837	14097.70014810	32547.52799890	-1.074511043	1.270505211	-1.785099927
1320.00000000	3148.86165643	18323.19841703	12305.75195578	-1.895271701	-0.678343847	-4.086577951
1440.00000000	2890.80638268	-15446.43952300	948.77010176	2.654407490	-2.909344895	4.486437362
1560.00000000	15415.98410712	-17699.90714437	25796.19644689	1.049818334	0.908822332	2.434107329
1680.00000000	19786.00618538	-8316.74570581	37723.74539119	0.220539813	1.554518900	0.947601047
1800.00000000	19007.28688729	3412.85948715	39886.66579255	-0.425733568	1.638276809	-0.345353807
1920.00000000	13627.93015254	14216.95401307	32356.13706868	-1.083991976	1.260802347	-1.810193903
2040.00000000	2963.26486560	18243.85063641	11868.25797486	-1.908015447	-0.747870342	-4.134004492
2160.00000000	3155.85126036	-15750.70393364	1422.32496953	2.620085624	-2.748990396	4.473527039
2280.00000000	15510.15191770	-17620.71002219	26050.43525345	1.035454678	0.925111006	2.408534465
2400.00000000	19801.67198812	-8174.33337167	37821.38577439	0.211812700	1.558576937	0.928231880
2520.00000000	18965.46529379	3565.19666242	39847.97510998	-0.433459945	1.637120585	-0.364653213
2640.00000000	13525.88227400	14335.15978787	32162.13236536	-1.093537945	1.250868256	-1.835451681
2760.00000000	2776.30574260	18156.98538451	11425.73046481	-1.920632199	-0.820370733	-4.181839232
2880.00000000	3417.20931586	-16038.79510665	1894.74934058	2.585515864	-2.596818146	4.456882556

9880 xx

0.00000000	13020.06750784	-2449.07193500	1.15896030	4.247363935	1.597178501	4.956708611
120.00000000	19190.32482476	9249.01266902	26596.71345328	-0.624960193	1.324550562	2.495697637
240.00000000	11332.67806218	16517.99124008	38569.78482991	-1.400974747	0.710947006	0.923935636
360.00000000	328.74217398	19554.92047380	40558.26246145	-1.593281066	0.126772913	-0.359627307
480.00000000	-10684.90590680	18057.15728839	33158.75253886	-1.383205997	-0.582328999	-1.744412556
600.00000000	-17069.78000550	9944.86797897	13885.91649059	0.044133354	-1.853448464	-3.815303117
720.00000000	13725.09398980	-2180.70877090	863.29684523	3.878478111	1.656846496	4.944867241
840.00000000	19089.63879226	9456.29670247	27026.79562883	-0.656614299	1.309112636	2.449371941
960.00000000	11106.41248373	16627.60874079	38727.35140296	-1.409722680	0.698582526	0.891383535
1080.00000000	72.40958621	19575.08054144	40492.12544001	-1.593394604	0.113655142	-0.390556063
1200.00000000	-10905.89252576	17965.41205111	32850.07298244	-1.371396120	-0.601706604	-1.782817058
1320.00000000	-17044.61207568	9635.48491849	13212.59462953	0.129244030	-1.903551430	-3.884569098
1440.00000000	14369.90303735	-1903.85601062	1722.15319852	3.543393116	1.701687176	4.913881358
1560.00000000	18983.96210441	9661.12233804	27448.99557732	-0.687189304	1.293808870	2.403630759
1680.00000000	10878.79336704	16735.31433954	38879.23434264	-1.418239666	0.686235750	0.858951848
1800.00000000	-184.03743100	19593.09371709	40420.40606889	-1.593348925	0.100448697	-0.421571993
1920.00000000	-11125.12138631	17870.19488928	32534.21521208	-1.359116236	-0.621413776	-1.821629856
2040.00000000	-17004.43272827	9316.53926351	12526.11883812	0.220330736	-1.955594322	-3.955058575
2160.00000000	14960.06492693	-1620.68430805	2574.96359381	3.238634028	1.734723385	4.868880331
2280.00000000	18873.46347257	9863.57004586	27863.46574735	-0.716736981	1.278632817	2.358448535
2400.00000000	10649.86857581	16841.14172669	39025.48035006	-1.426527152	0.673901057	0.826632332
2520.00000000	-440.53459323	19608.95524423	40343.10675451	-1.593138597	0.087147884	-0.452680559
2640.00000000	-11342.45028909	17771.44223942	32211.12535721	-1.346344015	-0.641464291	-1.860864234
2760.00000000	-16948.06005711	8987.64254880	11826.28284367	0.318007297	-2.009693492	-4.026726648
2880.00000000	15500.53445068	-1332.90981042	3419.72315308	2.960917974	1.758331634	4.813698638

9998 xx

0.00000000	25532.98947267	-27244.26327953	-1.11572421	2.410283885	2.194175683	0.545888526
-1440.00000000	-11362.18265118	-35117.55867813	-5413.62537994	3.137861261	-1.011678260	0.267510059
-1380.00000000	309.25349929	-36960.43090143	-4198.48007670	3.292429375	-0.002166046	0.402111628
-1320.00000000	11949.04009077	-35127.37816804	-2565.89806468	3.119942784	1.012096444	0.497284100
-1260.00000000	22400.45329336	-29798.63236321	-677.91515122	2.638533344	1.922477736	0.542792913
-1200.00000000	30640.84752458	-21525.02340201	1277.34808722	1.903464941	2.634294312	0.534540934
-1140.00000000	35899.56788035	-11152.71158138	3108.72535238	0.997393045	3.079858548	0.474873291
-1080.00000000	37732.45438600	288.18821054	4643.87587495	0.016652226	3.225184410	0.371669746
-1020.00000000	36045.92961699	11706.61816230	5746.32646574	-0.942409065	3.069888941	0.236662980

-960.00000000	31076.77273609	22063.44379776	6325.93403705	-1.794027976	2.642072476	0.083556127
-900.00000000	23341.26015320	30460.88002531	6342.91707895	-2.469409743	1.990861658	-0.073612096
-840.00000000	13568.39733054	36204.45930900	5806.79548733	-2.919354203	1.178920217	-0.221646814
-780.00000000	2628.58762420	38840.10855897	4771.91979854	-3.114400514	0.276239109	-0.348926401
-720.00000000	-8535.81598158	38171.79073851	3331.00311285	-3.043839958	-0.644462527	-0.445808894

11801 xx

0.00000000	7473.37102491	428.94748312	5828.74846783	5.107155391	6.444680305	-0.186133297
360.00000000	-3305.22148694	32410.84323331	-24697.16974954	-1.301137319	-1.151315600	-0.283335823
720.00000000	14271.29083858	24110.44309009	-4725.76320143	-0.320504528	2.679841539	-2.084054355
1080.00000000	-9990.05800009	22717.34212448	-23616.88515553	-1.016674392	-2.290267981	0.728923337
1440.00000000	9787.87836256	33753.32249667	-15030.79874625	-1.094251553	0.923589906	-1.522311008

14128 xx

0.00000000	34747.57932696	24502.37114079	-1.32832986	-1.731642662	2.452772615	0.608510081
120.00000000	18263.33439094	38159.96004751	4186.18304085	-2.744396611	1.255583260	0.528558932
240.00000000	-3023.38840703	41783.13186459	7273.03412906	-3.035574793	-0.271656544	0.309645251
360.00000000	-23516.34391907	34424.42065671	8448.49867693	-2.529120477	-1.726186020	0.009582303
480.00000000	-37837.46699511	18028.39727170	7406.25540271	-1.360069525	-2.725794686	-0.292555349
600.00000000	-42243.58460661	-3093.72887774	4422.91711801	0.163110919	-3.009980598	-0.517584362
720.00000000	-35597.57919549	-23407.91145393	282.09554383	1.641405246	-2.506773678	-0.606963478
840.00000000	-19649.19834455	-37606.11623860	-3932.71525948	2.689647056	-1.349150016	-0.537710698
960.00000000	1431.30912160	-41982.04949668	-7120.45467057	3.035263353	0.160882945	-0.327993994
1080.00000000	22136.97605384	-35388.19823762	-8447.62393401	2.587624889	1.630097136	-0.032349004
1200.00000000	37050.15790219	-19537.23321425	-7564.83463543	1.461844494	2.674654256	0.272202191
1320.00000000	42253.81760945	1431.81867593	-4699.87621174	-0.049247334	3.019518960	0.505890058
1440.00000000	36366.59147396	22023.54245720	-601.47121821	-1.549681546	2.571788981	0.607057418
1560.00000000	20922.12287985	36826.33975981	3654.91125886	-2.644070068	1.447521216	0.548722983
1680.00000000	-23.77224182	41945.51688402	6950.29891751	-3.043358385	-0.057417440	0.346112094
1800.00000000	-20964.17821076	36039.06206172	8418.91984963	-2.642795221	-1.546099886	0.052725852
1920.00000000	-36401.63863057	20669.75286162	7677.19769359	-1.549488154	-2.627052310	-0.254079652
2040.00000000	-42298.30327543	-119.03351118	4922.96388841	-0.052232768	-3.018152669	-0.493827331
2160.00000000	-37125.62383511	-20879.63058368	879.86971348	1.456499841	-2.619358421	-0.604081694
2280.00000000	-22250.12320553	-36182.74736487	-3393.15365183	2.583161226	-1.536647628	-0.556404555
2400.00000000	-1563.06258654	-42035.43179159	-6780.02161760	3.034917506	-0.052702046	-0.363395654
2520.00000000	19531.64069587	-36905.65470956	-8395.46892032	2.693682199	1.446079999	-0.075256054
2640.00000000	35516.53506142	-22123.71916638	-7815.04516935	1.646882125	2.568416058	0.232985912
2760.00000000	42196.03535976	-1547.32646751	-5187.39401981	0.166491841	3.019211549	0.480665780
2880.00000000	37802.25393045	19433.57330019	-1198.66634226	-1.359930580	2.677830903	0.602507466

16925 xx

0.00000000	5559.11686836	-11941.04090781	-19.41235206	3.392116762	-1.946985124	4.250755852
120.00000000	12339.83273749	-2771.14447871	18904.57603433	-0.871247614	2.600917693	0.581560002
240.00000000	-3385.00215658	7538.13955729	200.59008616	-2.023512865	-4.261808344	-6.856385787
360.00000000	12805.22442200	-10258.94667177	13780.16486738	0.619279224	1.821510542	2.507365975
480.00000000	5682.46556318	7199.30270473	15437.67134070	-2.474365406	2.087897336	-2.583767460
600.00000000	7628.94243982	-12852.72097492	2902.87208981	2.748131081	-0.740084579	4.125307943
720.00000000	11531.64866625	-858.27542736	19086.85993771	-1.170071901	2.660311986	0.096005705
840.00000000	-3866.98069515	2603.73442786	-4577.36484577	1.157257298	-8.453281164	-4.683959407
960.00000000	13054.77732721	-8707.92757730	15537.63259903	0.229846748	2.119467054	2.063396852
1080.00000000	3496.91064652	8712.83919778	12845.81838327	-2.782184997	1.552950644	-3.554436131
1200.00000000	9593.07424729	-13023.75963608	6250.46484931	2.072666376	0.278735334	3.778111073
1320.00000000	10284.79205084	1487.89914169	18824.37381327	-1.530335053	2.663107730	-0.542205966
1440.00000000	-984.62035146	-5187.03480813	-5745.59594144	4.340271916	-7.266811354	1.777668888

20413 xx

0.00000000	25123.29290741	-13225.49966287	3249.40351869	0.488683419	4.797897593	-0.961119693
1440.00000000	-151669.05280515	-5645.20454550	-2198.51592118	-0.869182889	-0.870759872	0.156508219

1560.00000000	-157497.71657495	-11884.99595074	-1061.44439402	-0.749657961	-0.864016715	0.157766101
1680.00000000	-162498.32255577	-18062.99733167	81.00915253	-0.638980378	-0.853687105	0.158098992
1800.00000000	-166728.76010920	-24155.99648299	1222.84128677	-0.535600687	-0.840455444	0.157680857
1920.00000000	-169935.81924592	-31767.29787964	2749.01540345	-0.430050431	-0.828904183	0.157812340
2040.00000000	-172703.07831815	-37662.95639336	3883.60052579	-0.338004891	-0.810277487	0.156020035
2160.00000000	-174823.19337404	-43417.55605219	5003.26312809	-0.250258622	-0.789828672	0.153764903
2280.00000000	-176324.63925775	-49018.51958648	6104.85025002	-0.166136613	-0.767706262	0.151092242
2400.00000000	-177231.42142458	-54454.12699497	7185.48661607	-0.085067854	-0.744001567	0.148033403
2520.00000000	-177563.73583232	-59713.14859144	8242.48472591	-0.006561730	-0.718760309	0.144608676
2640.00000000	-177338.48026483	-64784.54644698	9273.27220003	0.069809946	-0.691990238	0.140829236
2760.00000000	-176569.65151461	-69657.21976255	10275.33063459	0.144426878	-0.663665876	0.136698419
2880.00000000	-175268.65299073	-74319.77625463	11246.14177160	0.217631370	-0.633731091	0.132212491
3000.00000000	-173444.53039609	-78760.31560396	12183.13775212	0.289737325	-0.602099929	0.127361017
3120.00000000	-171104.14813653	-82966.21323591	13083.65278381	0.361037779	-0.568655903	0.122126889
3240.00000000	-168252.31543803	-86923.89363433	13944.87382716	0.431811396	-0.533249797	0.116486022
3360.00000000	-164891.86832887	-90618.58225954	14763.78794247	0.502328269	-0.495695896	0.110406725
3480.00000000	-161023.71139825	-94034.02398835	15537.12375729	0.572855321	-0.455766412	0.103848688
3600.00000000	-156646.82136725	-97152.15370791	16261.28409305	0.643661538	-0.413183688	0.096761524
3720.00000000	-151758.21285737	-99952.70098346	16932.26607548	0.715023254	-0.367609561	0.089082727
3840.00000000	-146352.86521283	-102412.70506284	17545.56394158	0.787229695	-0.318630913	0.080734873
3960.00000000	-140423.60777444	-104505.90799734	18096.04807097	0.860588979	-0.265739987	0.071621768
4080.00000000	-133960.95961851	-106201.98091318	18577.81121953	0.935434758	-0.208307307	0.061623110
4200.00000000	-126952.91860010	-107465.51906186	18983.96903112	1.012133628	-0.145543878	0.050587007
4320.00000000	-119384.69396454	-108254.71115372	19306.39581892	1.091093313	-0.076447479	0.038319282

21897 xx

0.00000000	-14464.72135182	-4699.19517587	0.06681686	-3.249312013	-3.281032707	4.007046940
120.00000000	-19410.46286123	-19143.03318969	23114.05522619	0.508602237	-1.156882269	2.379923455
240.00000000	-12686.06129708	-23853.75335645	35529.81733588	1.231633829	-0.221718202	1.118440291
360.00000000	-2775.46649359	-22839.64574119	39494.64689967	1.468963405	0.489481769	-0.023972788
480.00000000	7679.87883570	-16780.50760106	34686.21815555	1.364171080	1.211183897	-1.385151371
600.00000000	14552.40023028	-4819.50121461	17154.70672449	0.109201591	2.176124494	-3.854856805
720.00000000	-15302.38845375	-5556.43440300	1095.95088753	-2.838224312	-3.134231137	3.992596326
840.00000000	-19289.20066748	-19427.04851118	23759.45685636	0.552495087	-1.112499437	2.325112654
960.00000000	-12376.21976437	-23893.38020018	35831.33691892	1.246701529	-0.194294048	1.074867282
1080.00000000	-2400.55677665	-22698.62264640	39482.75964390	1.472582922	0.513555654	-0.069306561
1200.00000000	8031.66819252	-16455.77592085	34298.94391742	1.351357426	1.239633234	-1.448195324
1320.00000000	14559.48780372	-4238.43773813	16079.23154704	-0.026409655	2.218938770	-4.012628896
1440.00000000	-16036.04980660	-6372.51406468	2183.44834232	-2.485113443	-2.994994355	3.955891272
1560.00000000	-19156.71583814	-19698.89059957	24389.29473934	0.594278133	-1.069418599	2.271152044
1680.00000000	-12062.72925552	-23925.82362911	36120.66680667	1.261238798	-0.167201856	1.031478939
1800.00000000	-2024.96136966	-22551.56626703	39458.50085787	1.475816889	0.537615764	-0.114887472
1920.00000000	8379.80916204	-16123.95878459	33894.75123231	1.337468254	1.268432783	-1.512473301
2040.00000000	14527.86748873	-3646.33817120	14960.74306518	-0.180035839	2.261273515	-4.179355590
2160.00000000	-16680.12147335	-7149.80800425	3257.64227208	-2.178897351	-2.863927095	3.904876943
2280.00000000	-19013.58793448	-19958.93766022	25003.81778666	0.634100431	-1.027559823	2.218002685
2400.00000000	-11745.76155818	-23951.19438627	36397.87676581	1.275261813	-0.140425132	0.988259441
2520.00000000	-1648.81945070	-22398.50594576	39421.83273890	1.478660174	0.561671519	-0.160733093
2640.00000000	8723.97652795	-15784.99406275	33473.35215527	1.322433593	1.297602497	-1.578055493
2760.00000000	14452.25571587	-3043.42332645	13796.84870805	-0.355190169	2.302485443	-4.355767077
2880.00000000	-17246.31075678	-7890.72601508	4315.39410307	-1.910968458	-2.740945672	3.844722726

22312 xx

0.00000000	1442.10132912	6510.23625449	8.83145885	-3.475714837	0.997262768	6.835860345
54.20286720	306.10478453	-5816.45655525	-2979.55846068	3.950663855	3.415332543	-5.879974329
74.20286720	3282.82085464	2077.46972905	-5189.17988770	0.097342701	7.375135692	2.900196702
94.20286720	530.82729176	6426.20790003	1712.37076793	-3.837120395	-1.252430637	6.561602577
114.20286720	-3191.69170212	170.27219912	5956.29807775	-1.394956872	-7.438073471	-0.557553115
134.20286720	-1818.99222465	-6322.45146616	681.95247154	3.349795173	-1.530140265	-6.831522765
154.20286720	2515.66448634	-2158.83091224	-5552.13320544	2.571979660	7.311930509	-1.639865620

174.20286720	2414.52833210	5749.10150922	-1998.59693165	-2.681032960	3.527589301	6.452951429
194.20286720	-1877.98944331	3862.27848302	5112.48435863	-3.261489804	-6.026859137	3.433254768
214.20286720	-3117.36584395	-4419.74773864	3840.85960912	1.545479182	-5.475416581	-5.207913748
234.20286720	815.32034678	-5231.67692249	-3760.04690354	3.870864200	4.455588552	-5.211082191
254.20286720	3269.54341810	3029.00081083	-4704.67969713	-0.526711345	6.812157950	3.929825087
274.20286720	-10.18099756	6026.23341453	2643.50518407	-3.953623254	-2.616070012	6.145637500
294.20286720	-3320.58819584	-1248.42679945	5563.06017927	-0.637046974	-7.417786044	-2.076120187
314.20286720	-1025.48974616	-6366.98945782	-911.23559153	3.811771909	0.438071490	-6.829260617
334.20286720	3003.75996128	-413.85708003	-5706.15591435	1.674350083	7.694169068	0.316915204
354.20286720	1731.42816980	6258.27676925	-409.32527982	-3.400497806	1.447945424	6.904010052
374.20286720	-2582.52111460	2024.19020680	5647.55650268	-2.530348121	-7.221719393	1.438141553
394.20286720	-2440.56848578	-5702.77311877	1934.81094689	2.731792947	-3.350576075	-6.527773339
414.20286720	1951.22934391	-3423.59443045	-5121.67808201	3.249039133	6.465974362	-3.069806659
434.20286720	2886.50939356	4888.68626216	-3096.29885989	-1.973162139	4.877039020	5.832414910
454.20286720	-1276.55532182	4553.26898463	4406.19787375	-3.715146421	-5.320176914	4.418210777
474.20286720	-3181.54698042	-3831.29976506	4096.80242787	1.114159970	-6.104773578	-4.829967400

22674 xx

0.00000000	14712.22023280	-1443.81061850	0.83497888	4.418965470	1.629592098	4.115531802
120.00000000	25418.88807860	9342.60307989	23611.46690798	0.051284086	1.213127306	2.429004159
240.00000000	21619.59550749	16125.24978864	36396.79365831	-0.963604380	0.685454965	1.177181937
360.00000000	12721.50543331	19258.96193362	40898.47648359	-1.457448565	0.179955469	0.071502601
480.00000000	1272.80760054	18458.41971897	37044.74742696	-1.674863386	-0.436454983	-1.201040990
600.00000000	-10058.43188619	11906.60764454	21739.62097733	-1.245829683	-1.543789125	-3.324449221
720.00000000	10924.40116466	-2571.92414170	-2956.34856294	6.071727751	1.349579102	3.898430260
840.00000000	25332.14851525	8398.91099924	21783.90654357	0.222320754	1.272214306	2.580527192
960.00000000	22317.71926039	15574.82086129	35495.77144092	-0.892750056	0.737383381	1.291738834
1080.00000000	13795.68675885	19088.83051008	40803.69584385	-1.420277669	0.235599456	0.185517056
1200.00000000	2515.17145049	18746.63776282	37864.58088636	-1.668016053	-0.360431458	-1.052854596
1320.00000000	-9084.48602106	12982.62608646	24045.63900249	-1.378032363	-1.373184736	-3.013963835
1440.00000000	5647.00909495	-3293.90518693	-5425.85235063	8.507977176	0.414560797	2.543322806
1560.00000000	25111.63372210	7412.55109488	19844.25781729	0.416496290	1.332106006	2.739301737
1680.00000000	22961.47461641	14985.74459578	34511.09257381	-0.816711048	0.789391108	1.407901804
1800.00000000	14841.15301459	18876.91439870	40626.25901619	-1.380403341	0.290228810	0.298258120
1920.00000000	3750.70174081	18978.57939698	38578.11783220	-1.656939412	-0.287930881	-0.910825599
2040.00000000	-8027.30219489	13939.54436955	26136.49045637	-1.474476061	-1.222693624	-2.737178731
2160.00000000	-1296.95657092	-2813.69369768	-5871.09587258	9.881929371	-1.978467207	-1.922261005
2280.00000000	24738.60364819	6383.41644019	17787.27631900	0.639556952	1.392554379	2.906206324
2400.00000000	23546.85388669	14358.15602832	33441.67679479	-0.734895006	0.841564851	1.526009909
2520.00000000	15855.87696303	18624.05633582	40367.13420574	-1.337753546	0.343969522	0.410018472
2640.00000000	4976.44933591	19156.75504042	39189.68603184	-1.642084365	-0.218525096	-0.774148204
2760.00000000	-6909.20746210	14790.44707042	28034.46732222	-1.545152610	-1.088119523	-2.487447214
2880.00000000	-7331.65006707	-604.17323419	-2723.51014575	6.168997265	-3.634011554	-5.963531682

23177 xx

0.00000000	-8801.60046706	-0.03357557	-0.44522743	-3.835279101	-7.662552175	0.944561323
120.00000000	-1684.34352858	-31555.95196340	3888.99944319	2.023055719	-2.151306405	0.265065778
240.00000000	12325.51410155	-38982.15046244	4802.88832275	1.763224157	-0.102514446	0.012397139
360.00000000	22773.66831936	-34348.02176606	4228.77407391	1.067616787	1.352427865	-0.166956367
480.00000000	26194.40441089	-19482.94203672	2393.84774063	-0.313732186	2.808771328	-0.346204118
600.00000000	8893.50573448	5763.38890561	-713.69884164	-7.037399220	3.022613131	-0.370272416
720.00000000	-6028.75686537	-25648.99913786	3164.37107274	1.883159288	-3.177051976	0.390793162
840.00000000	8313.57299056	-38146.45710922	4697.80777535	1.905002133	-0.625883074	0.076098187
960.00000000	20181.29108622	-36842.60674073	4529.12568218	1.326244476	0.921916487	-0.114527455
1080.00000000	26302.61794569	-25173.39539436	3084.65309986	0.245398835	2.329974347	-0.287495880
1200.00000000	19365.07045602	-2700.00490122	317.42727417	-3.009733018	3.902496058	-0.478928582
1320.00000000	-9667.81878780	-16930.19112642	2095.87469034	1.279288285	-4.736005905	0.582878255
1440.00000000	4021.31438583	-36066.09209609	4442.91587411	2.007322354	-1.227461376	0.149383897

23333 xx

0.00000000	-9301.24542292	3326.10200382	2318.36441127	-8.729303005	-0.828225037	-0.122314827
120.00000000	-44672.91239679	-6213.11996581	-1738.80131727	-3.719475070	-1.336673022	-0.621888261
240.00000000	-67053.08885387	-14994.69685946	-5897.99072793	-2.860576613	-1.183771565	-0.568473909
360.00000000	-85227.84253168	-22897.08484471	-9722.59184564	-2.426469823	-1.078592475	-0.525341431
480.00000000	-100986.00419136	-30171.19698695	-13283.77044765	-2.147108978	-1.000530827	-0.491587582
600.00000000	-115093.00686386	-36962.56316477	-16634.15682929	-1.945446188	-0.938947736	-0.464199202
720.00000000	-127965.80064891	-43363.32967165	-19809.90480432	-1.789652016	-0.888278463	-0.441254468
840.00000000	-139863.28332206	-49436.45704153	-22836.80438139	-1.663762568	-0.845315913	-0.421548627
960.00000000	-150960.22978258	-55227.45413896	-25734.01408879	-1.558730986	-0.808061065	-0.404293846
1080.00000000	-161381.71414630	-60770.64040903	-28516.26290017	-1.468977174	-0.775190459	-0.388951810
1200.00000000	-171221.18736947	-66092.76474442	-31195.19847387	-1.390837596	-0.745785633	-0.375140398
1320.00000000	-180550.82888745	-71215.23290630	-33780.24938270	-1.321788672	-0.719184752	-0.362579495
1440.00000000	-189427.87533074	-76155.54943344	-36279.19882816	-1.260024473	-0.694896053	-0.351058133
1560.00000000	-197898.69401408	-80928.29015181	-38698.57972447	-1.204211888	-0.672544709	-0.340413731
1600.00000000	-200638.82986236	-82484.14969882	-39488.34331447	-1.186748462	-0.665472422	-0.337037582

23599 xx

0.00000000	9892.63794341	35.76144969	-1.08228838	3.556643237	6.456009375	0.783610890
20.00000000	11931.95642997	7340.74973750	886.46365987	0.308329116	5.532328972	0.672887281
40.00000000	11321.71039205	13222.84749156	1602.40119049	-1.151973982	4.285810871	0.521919425
60.00000000	9438.29395675	17688.05450261	2146.59293402	-1.907904054	3.179955046	0.387692479
80.00000000	6872.08634639	20910.11016811	2539.79945034	-2.323995367	2.207398462	0.269506121
100.00000000	3933.37509798	23024.07662542	2798.25966746	-2.542860616	1.327134966	0.162450076
120.00000000	816.64091546	24118.98675475	2932.69459428	-2.626838010	0.504502763	0.062344306
140.00000000	-2334.41705804	24246.86096326	2949.36448841	-2.602259646	-0.288058266	-0.034145135
160.00000000	-5394.31798039	23429.42716149	2850.86832586	-2.474434068	-1.074055982	-0.129868366
180.00000000	-8233.35130237	21661.24480883	2636.51456118	-2.230845533	-1.875742344	-0.227528603
200.00000000	-10693.96497348	18909.88168891	2302.33707548	-1.835912433	-2.716169865	-0.329931880
220.00000000	-12553.89669904	15114.63990716	1840.93573231	-1.212478879	-3.619036996	-0.439970633
240.00000000	-13450.20591864	10190.57904289	1241.95958736	-0.189082511	-4.596701971	-0.559173899
260.00000000	-12686.60437121	4079.31106161	498.27078614	1.664498211	-5.559889865	-0.676747779
280.00000000	-8672.55867753	-2827.56823315	-342.59644716	5.515079852	-5.551222962	-0.676360044
300.00000000	1153.31498060	-6411.98692060	-779.87288941	9.689818102	1.388598425	0.167868798
320.00000000	9542.79201056	-533.71253081	-65.73165428	3.926947087	6.459583539	0.785686755
340.00000000	11868.80960100	6861.59590848	833.72780602	0.452957852	5.632811328	0.685262323
360.00000000	11376.23941678	12858.97121366	1563.40660172	-1.087665695	4.374693347	0.532207051
380.00000000	9547.70300782	17421.48570758	2118.56907515	-1.876540262	3.253891728	0.395810243
400.00000000	7008.51470263	20725.47471227	2520.56064289	-2.308703599	2.270724438	0.276138613
420.00000000	4082.28135104	22911.04184601	2786.37568309	-2.536665546	1.383670232	0.168153407
440.00000000	969.17978149	24071.23673676	2927.31326579	-2.626695115	0.557172428	0.067536854
460.00000000	-2184.71515444	24261.21671601	2950.08142825	-2.607072866	-0.236887607	-0.029125215
480.00000000	-5253.42223370	23505.37595671	2857.66120738	-2.484424544	-1.022255436	-0.124714444
500.00000000	-8108.27961017	21800.81688388	2649.72981961	-2.247597251	-1.821159176	-0.221925624
520.00000000	-10594.77795556	19117.80779221	2322.72136979	-1.863118484	-2.656426668	-0.323521502
540.00000000	-12497.32045995	15398.64085906	1869.69983897	-1.258130763	-3.551583368	-0.432338888
560.00000000	-13467.92475245	10560.90147785	1280.78399181	-0.271870523	-4.520514224	-0.550016092
580.00000000	-12848.18843590	4541.21901842	548.53826427	1.494157156	-5.489585384	-0.667472039
600.00000000	-9152.70552728	-2344.24950144	-287.98121970	5.127921095	-5.650383025	-0.685989008
620.00000000	280.38490909	-6500.10264018	-790.36092984	9.779619614	0.581815811	0.074171345
640.00000000	9166.25784315	-1093.12552651	-129.49428887	4.316668714	6.438636494	0.785116609
660.00000000	11794.48942915	6382.21138354	780.88439015	0.604412453	5.731729369	0.697574333
680.00000000	11424.30138324	12494.26088864	1524.33165488	-1.021328075	4.463448968	0.542532698
700.00000000	9652.09867350	17153.84762075	2090.48038336	-1.844516637	3.327522235	0.403915232
720.00000000	7140.41945884	20539.25485336	2501.21469368	-2.293173684	2.333507912	0.282716311

24208 xx

0.00000000	7534.10987189	41266.39266843	-0.10801028	-3.027168008	0.558848996	0.207982755
120.00000000	-14289.19940414	39469.05530051	1428.62838591	-2.893205245	-1.045447840	0.179634249
240.00000000	-32222.92014955	26916.25425799	2468.59996594	-1.973007929	-2.359335071	0.102539376

360.00000000	-41413.95109398	7055.51656639	2838.90906671	-0.521665080	-3.029172207	-0.002066843
480.00000000	-39402.72251896	-14716.42475223	2441.32678358	1.066928187	-2.878714619	-0.105865729
600.00000000	-26751.08889828	-32515.13982431	1384.38865570	2.366228869	-1.951032799	-0.181018498
720.00000000	-6874.77975542	-41530.38329422	-46.60245459	3.027415087	-0.494671177	-0.207337260
840.00000000	14859.52039042	-39302.58907247	-1465.02482524	2.869609883	1.100123969	-0.177514425
960.00000000	32553.14863770	-26398.88401807	-2485.45866002	1.930064459	2.401574539	-0.099250520
1080.00000000	41365.67576837	-6298.09965811	-2828.05254033	0.459741276	3.051680214	0.006431872
1200.00000000	38858.83295070	15523.39314924	-2396.86850752	-1.140211488	2.867567143	0.110637217
1320.00000000	25701.46068162	33089.42617648	-1308.68556638	-2.428713821	1.897381431	0.184605907
1440.00000000	5501.08137100	41590.27784405	138.32522930	-3.050691874	0.409203052	0.207958133

25954 xx

0.00000000	8827.15660472	-41223.00971237	3.63482963	3.007087319	0.643701323	0.000941663
-1440.00000000	8118.18519221	-41368.40537378	4.11046687	3.017696741	0.591994297	0.000933016
-1320.00000000	27766.34015328	-31724.97000557	9.93297846	2.314236153	2.024903193	0.000660861
-1200.00000000	39932.57237973	-13532.60040454	13.12958252	0.987382819	2.911942843	0.000213298
-1080.00000000	41341.01365441	8305.71681955	12.84988501	-0.605098224	3.014378268	-0.000291034
-960.00000000	31614.99210558	27907.29155353	9.16618797	-2.034243523	2.305014102	-0.000718418
-840.00000000	13375.75227587	39994.27017651	3.05416854	-2.915424366	0.975119874	-0.000955576
-720.00000000	-8464.89963309	41312.93549892	-3.86622919	-3.011600615	-0.617275050	-0.000939664
-600.00000000	-28026.23406158	31507.89995661	-9.76047869	-2.296840160	-2.043607595	-0.000674889
-480.00000000	-40040.01314363	13218.00579413	-13.06594832	-0.963328772	-2.919827983	-0.000231414
-360.00000000	-41268.43291976	-8632.06859693	-12.90661266	0.630042315	-3.009677376	0.000273163
-240.00000000	-31377.85317015	-28156.13970334	-9.32605530	2.054021717	-2.288554158	0.000704959
-120.00000000	-13031.41552688	-40092.33381029	-3.27636660	2.924657466	-0.950541167	0.000949381
0.00000000	8827.15660472	-41223.00971237	3.63482963	3.007087319	0.643701323	0.000941663
120.00000000	28306.85426674	-31243.80147394	9.57216891	2.279137743	2.064316875	0.000684127
240.00000000	40159.05128805	-12845.39151157	12.96086316	0.937265422	2.928448287	0.000245505
360.00000000	41192.55903455	9013.79606759	12.90495666	-0.656727442	3.003543458	-0.000257479
480.00000000	31131.69755798	28445.55681731	9.42419238	-2.073484842	2.269770851	-0.000691233
600.00000000	12687.81846530	40217.83324639	3.44726249	-2.931721827	0.924962230	-0.000940766
720.00000000	-9172.23500245	41161.63475527	-3.43575757	-3.000571486	-0.668847508	-0.000940101
840.00000000	-28562.51093192	31022.45987587	-9.39562161	-2.261449202	-2.082713897	-0.000689669
960.00000000	-40260.77504549	12529.11484344	-12.84915105	-0.913097031	-2.935933528	-0.000256181
1080.00000000	-41114.14376538	-9338.87194483	-12.87952404	0.681588815	-2.998432565	0.000245006
1200.00000000	-30890.01512240	-28690.40750792	-9.48037212	2.092989805	-2.252978152	0.000680459
1320.00000000	-12341.46194020	-40310.06316386	-3.55833201	2.940537098	-0.900219523	0.000934170
1440.00000000	9533.27750818	-41065.52390214	3.30756482	2.995596171	0.695200236	0.000938525

26900 xx

0.00000000	-42014.83795787	3702.34357772	-26.67500257	-0.269775247	-3.061854393	0.000336726
9300.00000000	40968.68133298	-9905.99156086	11.84946837	0.722756848	2.989645389	-0.000161261
9360.00000000	42135.66858481	1072.99195618	10.83481752	-0.078150602	3.074772455	-0.000380063
9400.00000000	41304.75156132	8398.27742944	9.74006214	-0.612515135	3.014117469	-0.000511575

26975 xx

0.00000000	-14506.92313768	-21613.56043281	10.05018894	2.212943308	1.159970892	3.020600202
120.00000000	7309.62197950	6076.00713664	6800.08705263	1.300543383	5.322579615	-4.788746312
240.00000000	-3882.62933791	11960.00543452	-25088.14383845	-2.146773699	-1.372461491	-2.579382089
360.00000000	-16785.45507465	-734.79159704	-34300.57085853	-1.386528125	-1.907762641	-0.220949641
480.00000000	-23524.16689356	-13629.45124622	-30246.27899200	-0.462846784	-1.586139830	1.269293624
600.00000000	-22890.23597092	-22209.35900155	-16769.91946116	0.704351342	-0.671112594	2.432433851
720.00000000	-11646.39698980	-19855.44222106	3574.00109607	2.626712727	1.815887329	2.960883901
840.00000000	7665.76124241	11159.78946577	345.93813117	-0.584818007	3.193514161	-5.750338922
960.00000000	-6369.35388112	10204.80073022	-27844.52150384	-2.050573276	-1.582940542	-2.076075232
1080.00000000	-18345.64763145	-2977.76684430	-34394.90760612	-1.243589864	-1.892050757	0.060372061
1200.00000000	-23979.74839255	-15436.44139571	-28616.50540218	-0.294973425	-1.482987916	1.478255628
1320.00000000	-21921.97167880	-22852.45147658	-13784.85308485	0.945455629	-0.428940995	2.596964378
1440.00000000	-8266.43821031	-17210.74590112	6967.95546070	3.082244069	2.665881872	2.712555075

1560.00000000	6286.85464535	13809.56328971	-6321.60663781	-1.615964016	1.383135377	-5.358719132
1680.00000000	-8730.87526788	8244.63344365	-30039.92372791	-1.935622871	-1.724162072	-1.631224738
1800.00000000	-19735.81883249	-5191.76593007	-34166.14974143	-1.097835530	-1.860148418	0.324401050
1920.00000000	-24232.73847703	-17112.08243255	-26742.88893252	-0.119786184	-1.364365317	1.680220468
2040.00000000	-20654.45640708	-23184.54386047	-10611.55144716	1.209238113	-0.144169639	2.748054938
2160.00000000	-4337.15988957	-13410.46817244	9870.45949215	3.532753095	3.772236461	2.088424247
2280.00000000	4074.62263523	14698.07548285	-12248.65327973	-2.053824693	0.203325817	-4.607867718
2400.00000000	-10950.23438984	6148.66879447	-31736.65532865	-1.809875605	-1.816179062	-1.233364913
2520.00000000	-20952.40702045	-7358.71507895	-33633.06643074	-0.948973031	-1.813594137	0.573893078
2640.00000000	-24273.48944134	-18637.15546906	-24633.27702390	0.064161440	-1.228537560	1.875728935
2760.00000000	-19057.55468077	-23148.29322082	-7269.38614178	1.500802809	0.195383037	2.879031237
2880.00000000	43.69305308	-8145.90299207	11634.57079913	3.780661682	5.105315423	0.714401345

28057 xx

0.00000000	-2715.28237486	-6619.26436889	-0.01341443	-1.008587273	0.422782003	7.385272942
120.00000000	-1816.87920942	-1835.78762132	6661.07926465	2.325140071	6.655669329	2.463394512
240.00000000	1483.17364291	5395.21248786	4448.65907172	2.560540387	4.039025766	-5.736648561
360.00000000	2801.25607157	5455.03931333	-3692.12865694	-0.595095864	-3.951923117	-6.298799125
480.00000000	411.09332812	-1728.99769152	-6935.45548810	-2.935970964	-6.684085058	1.492800886
600.00000000	-2506.52558454	-6628.98655094	-988.07784497	-1.390577189	-0.556164143	7.312736468
720.00000000	-2090.79884266	-2723.22832193	6266.13356576	1.992640665	6.337529519	3.411803080
840.00000000	1091.80560222	4809.88229503	5172.42897894	2.717483546	4.805518977	-5.030019896
960.00000000	2811.14062300	5950.65707171	-2813.23705389	-0.159662742	-3.121215491	-6.775341949
1080.00000000	805.72698304	-812.16627907	-7067.58483968	-2.798936020	-6.889265977	0.472770873
1200.00000000	-2249.59837532	-6505.84890714	-1956.72365062	-1.731234729	-1.528750230	7.096660885
1320.00000000	-2311.57375797	-3560.99112891	5748.16749600	1.626569751	5.890482233	4.293545048
1440.00000000	688.16056594	4124.87618964	5794.55994449	2.810973665	5.479585563	-4.224866316
1560.00000000	2759.94088230	6329.87271798	-1879.19518331	0.266930672	-2.222670878	-7.119390567
1680.00000000	1171.50677137	125.82053748	-7061.96626202	-2.605687852	-6.958489749	-0.556333225
1800.00000000	-1951.43708472	-6251.71945820	-2886.95472355	-2.024131483	-2.475214272	6.741537478
1920.00000000	-2475.70722288	-4331.90569958	5117.31234924	1.235823539	5.322743371	5.091281211
2040.00000000	281.46097847	3353.51057102	6302.87900650	2.840647273	6.047222485	-3.337085992
2160.00000000	2650.33118860	6584.33434851	-908.29027134	0.675457235	-1.274044972	-7.323921567
2280.00000000	1501.17226597	1066.31132756	-6918.71472952	-2.361891904	-6.889669974	-1.574718619
2400.00000000	-1619.73468334	-5871.14051991	-3760.56587071	-2.264093975	-3.376316601	6.254622256
2520.00000000	-2581.04202505	-5020.05572531	4385.92329047	0.829668458	4.645048038	5.789262667
2640.00000000	-119.22080628	2510.90620488	6687.45615459	2.807575712	6.496549689	-2.384136661
2760.00000000	2486.23806726	6708.18210028	80.43349581	1.057274905	-0.294294027	-7.384689123
2880.00000000	1788.42334580	1990.50530957	-6640.59337725	-2.074169091	-6.683381288	-2.562777776

28129 xx

0.00000000	21707.46412351	-15318.61752390	0.13551152	1.304029214	1.816904974	3.161919976
120.00000000	18616.75971861	3166.15177043	18833.41523210	-2.076122016	2.838457575	1.586210535
240.00000000	-3006.50596328	18522.20742011	18941.84078154	-3.375452789	1.032680773	-1.559324534
360.00000000	-21607.02086957	15432.59962630	206.62470309	-1.306049851	-1.817011568	-3.163725018
480.00000000	-18453.06134549	-3150.83256134	-18685.83030936	2.106017925	-2.860236337	-1.586151870
600.00000000	3425.11742384	-18514.73232706	-18588.67200557	3.394666340	-1.003072030	1.610061295
720.00000000	21858.23838149	-15101.51661554	387.34517048	1.247973967	1.856017403	3.161439948
840.00000000	18360.69935796	3506.55256762	19024.81678979	-2.122684184	2.830618605	1.537510677
960.00000000	-3412.84765409	18646.85269710	18748.00359987	-3.366815728	0.986039922	-1.607874972
1080.00000000	-21758.08331586	15215.44829478	-180.82181406	-1.250144680	-1.856490448	-3.163774870
1200.00000000	-18193.41290284	-3493.85876912	-18877.14757717	2.153326942	-2.852221264	-1.536617760
1320.00000000	3833.57386848	-18635.77026711	-18388.68722886	3.384748179	-0.955363841	1.658785020
1440.00000000	22002.20074562	-14879.72595593	774.32827099	1.191573619	1.894561165	3.159953047

28350 xx

0.00000000	6333.08123128	-1580.82852326	90.69355720	0.714634423	3.224246550	7.083128132
120.00000000	-3990.93845855	3052.98341907	4155.32700629	-5.909006188	-0.876307966	-5.039131404
240.00000000	-603.55232010	-2685.13474569	-5891.70274282	7.572519907	-1.975656726	0.121722605

360.00000000	4788.22345627	782.56169214	4335.14284621	-4.954509026	3.683346464	4.804645839
480.00000000	-6291.84601644	1547.82790772	-453.67116498	-0.308625588	-3.341538574	-7.082659115
600.00000000	4480.74573428	-3028.55200374	-3586.94343641	5.320920857	1.199736275	5.626350481
720.00000000	-446.42460916	2932.28872588	5759.19389757	-7.561000245	1.550975493	-1.374970885
840.00000000	-3713.79581831	-1382.66125130	-5122.45131136	6.090931626	-3.512629733	-3.467571746
960.00000000	6058.32017522	-827.47406722	2104.04678651	-1.798403024	3.787067272	6.641439744
1080.00000000	-5631.73659006	2623.70953644	1766.49125084	-3.216401578	-2.309140959	-6.788609120
1200.00000000	2776.84991560	-3255.36941953	-4837.19667790	6.748135564	-0.193044825	4.005718698
1320.00000000	1148.04430837	2486.07343386	5826.34075913	-7.420162295	2.589456382	0.356350006
1440.00000000	-4527.90871828	-723.29199041	-4527.44608319	5.121674217	-3.909895427	-4.500218556

28623 xx

0.00000000	-11665.70902324	24943.61433357	25.80543633	-1.596228621	-1.476127961	1.126059754
120.00000000	-11645.35454950	979.37668356	5517.89500058	3.407743502	-5.183094988	-0.492983277
240.00000000	5619.19252274	19651.44862280	-7261.38496765	-2.013634213	3.106842861	0.284235517
360.00000000	-9708.68629714	26306.14553149	-1204.29478856	-1.824164290	-0.931909596	1.113419052
480.00000000	-14394.03162892	6659.30765074	5593.38345858	1.556522911	-4.681657614	0.296912248
600.00000000	7712.09476270	15565.72627434	-7342.40465571	-1.646800364	4.070313571	-0.109483081
720.00000000	-7558.36739603	27035.11367962	-2385.12054184	-1.999583791	-0.393409283	1.078093515
840.00000000	-15495.61862220	11550.15897828	5053.83178121	0.469277336	-4.029761073	0.679054742
960.00000000	9167.02568222	10363.65204210	-6871.52576042	-0.881621027	5.223361510	-0.740696297
1080.00000000	-5275.80272094	27151.78486008	-3494.50687216	-2.129609388	0.150196480	1.021038089
1200.00000000	-15601.37656145	15641.29379850	4217.03266850	-0.249183123	-3.405238557	0.888214503
1320.00000000	9301.05872300	3883.15265574	-5477.86477017	0.871447821	6.493677331	-1.885545282
1440.00000000	-2914.31065828	26665.20392758	-4511.09814335	-2.216261909	0.710067769	0.940691824

28626 xx

0.00000000	42080.71852213	-2646.86387436	0.81851294	0.193105177	3.068688251	0.000438449
120.00000000	37740.00085593	18802.76872802	3.45512584	-1.371035206	2.752105932	0.000336883
240.00000000	23232.82515008	35187.33981802	4.98927428	-2.565776620	1.694193132	0.000163365
360.00000000	2467.44290178	42093.60909959	5.15062987	-3.069341800	0.179976276	-0.000031739
480.00000000	-18962.59052991	37661.66243819	4.04433258	-2.746151982	-1.382675777	-0.000197633
600.00000000	-35285.00095313	23085.44402778	2.08711880	-1.683277908	-2.572893625	-0.000296282
720.00000000	-42103.20138132	2291.06228893	-0.13274964	-0.166974816	-3.070104560	-0.000311007
840.00000000	-37580.31858370	-19120.40485693	-2.02755702	1.394367848	-2.740341612	-0.000248591
960.00000000	-22934.20761876	-35381.23870806	-3.16495932	2.580167539	-1.672360951	-0.000134907
1080.00000000	-2109.90332389	-42110.71508198	-3.36507889	3.070935369	-0.153808390	-0.000005855
1200.00000000	19282.77774728	-37495.59250598	-2.71861462	2.734400524	1.406220933	0.000103486
1320.00000000	35480.60990600	-22779.03375285	-1.52841859	1.661210676	2.587414593	0.000168300
1440.00000000	42119.96263499	-1925.77567263	-0.19827433	0.140521206	3.071541613	0.000179561

28872 xx

0.00000000	-6131.82730456	2446.52815528	-253.64211033	-0.144920228	0.995100963	7.658645067
5.00000000	-5799.24256134	2589.14811119	2011.54515100	2.325207364	-0.047125672	7.296234071
10.00000000	-4769.05061967	2420.46580562	4035.30855837	4.464585796	-1.060923209	6.070907874
15.00000000	-3175.45157340	1965.98738086	5582.12569607	6.049639376	-1.935777558	4.148607019
20.00000000	-1210.19024802	1281.54541294	6474.68172772	6.920746273	-2.580517337	1.748783868
25.00000000	896.73799533	447.12357305	6607.22400507	6.983396282	-2.925846168	-0.872655207
30.00000000	2896.99663534	-440.04738594	5954.92675486	6.211488246	-2.926949815	-3.433959806
35.00000000	4545.78970167	-1273.55952872	4580.16512984	4.656984233	-2.568711513	-5.638510954
40.00000000	5627.43299371	-1947.94282469	2634.16714930	2.464141047	-1.873985161	-7.195743032
45.00000000	5984.72318534	-2371.37691609	349.87996209	-0.121276950	-0.911981546	-7.859613894
50.00000000	5548.43325922	-2480.16469245	-1979.24314527	-2.763269534	0.199691915	-7.482796996

29141 xx

0.00000000	423.99295524	-6658.12256149	136.13040356	1.006373613	0.217309983	7.662587892
20.00000000	931.80883587	-1017.17852239	6529.19244527	-0.298847918	7.613891977	1.226399480
40.00000000	-83.44906141	6286.20208453	2223.49837161	-1.113515974	2.530970283	-7.219445568

60.00000000	-958.57681221	3259.26005348	-5722.63732467	-0.101225813	-6.735338321	-3.804851872
80.00000000	-255.25619985	-5132.59762974	-4221.27233118	1.077709303	-4.905938824	5.892521264
100.00000000	867.44295097	-5038.40402933	4256.73810533	0.479447535	5.032326446	5.857126248
120.00000000	559.16882013	3376.30587937	5699.22017391	-0.906749328	6.646149867	-3.852331832
140.00000000	-669.85184205	6196.00229484	-2281.95741770	-0.795804092	-2.752114827	-7.202478520
160.00000000	-784.20708019	-1278.53125553	-6449.19892596	0.636702380	-7.595425203	1.431090802
180.00000000	406.15811659	-6607.03115799	148.33021477	1.009818575	0.231843765	7.692047844
200.00000000	916.34911813	-884.08649248	6491.09810362	-0.302163049	7.669887109	1.084336909
220.00000000	-104.02490970	6304.31821405	1960.08739882	-1.108873823	2.259522809	-7.351147710
240.00000000	-944.61642849	2872.17248379	-5846.94103362	-0.051117686	-6.989747076	-3.413102600
260.00000000	-187.16569888	-5404.86163467	-3731.97057618	1.094696706	-4.412110995	6.326060952
280.00000000	884.59720467	-4465.74516163	4725.83632696	0.380656028	5.691554046	5.303910983
300.00000000	446.40767236	4086.66839620	5093.05596650	-0.982424447	6.072965199	-4.791630682
320.00000000	-752.24467495	5588.35473301	-3275.04092573	-0.661161370	-4.016290740	-6.676898026
340.00000000	-643.72872525	-2585.02528560	-5923.01306608	0.807922142	-7.171597814	3.041115058
360.00000000	584.40295819	-6202.35605817	1781.00536019	0.869250450	2.226927514	7.471676765
380.00000000	779.59211765	1100.73728301	6311.59529480	-0.599552305	7.721032522	-1.275153027
400.00000000	-403.03155588	6399.18000837	-364.12735875	-1.008861924	-0.516636615	-7.799812287
420.00000000	-852.93910071	192.65232023	-6322.47054784	0.396006194	-7.882964919	-0.289331517

29238 xx

0.00000000	-5566.59512819	-3789.75991159	67.60382245	2.873759367	-3.825340523	6.023253926
120.00000000	4474.27915495	-1447.72286142	4619.83927235	4.712595822	5.668306153	-2.701606741
240.00000000	1922.17712474	5113.01138342	-4087.08470203	-6.490769651	-0.522350158	-3.896001154
360.00000000	-6157.93546882	-2094.70798790	-1941.63730960	0.149900661	-5.175192523	5.604262034
480.00000000	2482.64052411	-3268.45944555	5146.38006190	6.501814698	4.402848754	-0.350943511
600.00000000	4036.26455287	4827.43347201	-2507.99063955	-5.184409515	1.772280695	-5.331390168
720.00000000	-5776.81371622	-118.64155319	-3641.22052418	-2.539917207	-5.622701582	4.403125405
840.00000000	67.98699487	-4456.49213473	4863.71794283	7.183809420	2.418917791	2.015642495
960.00000000	5520.62207038	3782.38203554	-596.73193161	-3.027966069	3.754152525	-6.013506363
1080.00000000	-4528.05104455	1808.46273329	-4816.99727762	-4.808419763	-5.185789345	2.642104494
1200.00000000	-2356.61468078	-4852.51202272	3856.53816184	6.688446735	0.118520958	4.021854210
1320.00000000	6149.65800134	2173.59423261	1369.29488732	-0.345832777	5.109857861	-5.842951828
1440.00000000	-2629.55011449	3400.98040158	-5344.38217129	-6.368548448	-3.998963509	0.577253064

88888 xx

0.00000000	2328.96975262	-5995.22051338	1719.97297192	2.912073281	-0.983417956	-7.090816210
120.00000000	1020.69234558	2286.56260634	-6191.55565927	-3.746543902	6.467532721	1.827985678
240.00000000	-3226.54349155	3503.70977525	4532.80979343	1.000992116	-5.788042888	5.162585826
360.00000000	2456.10706533	-6071.93855503	1222.89768554	2.679390040	-0.448290811	-7.228792155
480.00000000	787.16457349	2719.91800946	-6043.86662024	-3.759883839	6.277439314	2.397897864
600.00000000	-3110.97648029	3121.73026235	4878.15217035	1.244916056	-6.124880425	4.700576353
720.00000000	2567.56229695	-6112.50383922	713.96374435	2.440245751	0.098109002	-7.319959258
840.00000000	556.05661780	3144.52288201	-5855.34636178	-3.754660143	6.044752775	2.957941672
960.00000000	-2982.47940539	2712.61663711	5192.32330472	1.475566773	-6.427737014	4.202420227
1080.00000000	2663.08964352	-6115.48290885	196.40072866	2.196121564	0.652415093	-7.362824152
1200.00000000	328.54999674	3557.09490552	-5626.21427211	-3.731193288	5.769341172	3.504058731
1320.00000000	-2842.06876757	2278.42343492	5472.33437150	1.691852635	-6.693216335	3.671022712
1440.00000000	2742.55398832	-6079.67009123	-326.39012649	1.948497651	1.211072678	-7.356193131

APPENDIX D. REFERENCES

[ii] Hoots, Felix R., et al. "History of Analytical Orbit Modeling in the U.S. Space Surveillance System." Journal of Guidance Control, and Dynamics March-April 2004: 174-185.

[iii] Vallado, David, et al. 'Revisiting Spacetrack Report #3', AIAA 2006-6753

[iv] Vallado, David, et al. 'Revisiting Spacetrack Report #3', AIAA 2006-6753

[v] Abercromby, Kira. "Object 8832" E-Mail to N. Miura, 26 May 2009.

[vi] "Selected Orbital Data" 2009. <http://www.heavens-above.com>

[vii] http://en.wikipedia.org/wiki/WGS84

Deleted: Endnotes

Formatted: Font: 12 pt

Deleted: ¶

Formatted: Font: 12 pt

Formatted: Line spacing: Double

Formatted: Normal, Line spacing: Double

Formatted: Font: 12 pt