AIAA 2006-6753

Revisiting Spacetrack Report #3

David A. Vallado*
Center for Space Standards and Innovation, Colorado Springs, Colorado, 80920

Paul Crawford†
Crawford Communications Ltd., Dundee, DD2 1EW, UK

Richard Hujsak‡
Analytical Graphics, Inc., Exton, PA, 19341

and

T. S. Kelso§
Center for Space Standards and Innovation, Colorado Springs, Colorado, 80920

Over a quarter century ago, the United States Department of Defense (DoD) released the equations and source code used to predict satellite positions through *SpaceTrack Report Number 3* (STR#3). Because the DoD's two-line element sets (TLEs) were the only source of orbital data, widely available through NASA, this code became commonplace among users needing accurate results. However, end users made code changes to correct the implementation of the equations and to handle rare cases encountered in operations. These changes migrated into numerous new versions and compiled programs outside the DoD. Changes made to the original STR#3 code have not been released in a comprehensive form to the public, so the code available to the public no longer matches the code used by DoD to produce the TLEs. Fortunately, independent efforts, technical papers, and source code enabled us to synthesize a non-proprietary version which we believe is up-to-date and accurate. This paper provides source code, test cases, results, and analysis of a version of SGP4 theory designed to be highly compatible with recent DoD versions.

I. INTRODUCTION AND HISTORY

The Simplified General Perturbations (SGP) model series began development in the 1960s (Lane 1965), and became operational in the early 1970s (Lane and Cranford, 1969). The original release of the refined Simplified General Perturbations-4 (SGP4) propagator source code was *Spacetrack Report Number 3* (Hoots and Roehrich, 1980). That release resulted from a user compatibility survey of space surveillance operational sites and official users. The magnitude of the resulting variations spurred an effort to promote better compatibility for users. The intent was to get the operational community, as well as ordinary users, synchronized with respect to the implementation. The best vehicle for this was a technical report, including the computer source code. It was designed for the widest possible dissemination. Although most of the equations were given, the use of the source code became common practice for using Two-line Element (TLE) sets.**

* Technical Program Manager, Center for Space Standards and Innovation, 7150 Campus Dr, Suite 260, dvallado@centerforspace.com, AIAA Associate Fellow.
† Principal Engineer, 25 Blackness Avenue, pcrawford@dundee0.demon.co.uk.
‡ Orbit Determination Lead Engineer, 220 Valley Creek Blvd, rhujsak@agi.com.
§ Technical Program Manager, Center for Space Standards and Innovation, 7150 Campus Dr, Suite 260, tskelso@centerforspace.com, AIAA Associate Fellow.
** Note that the code is not vetted as a consensus standard. The well-confirmed and long-established industry consensus standards process requires consensus on all elements of a technique and its implementation throughout a wide community of experts. There is no formal consensus standard for orbit determination or propagation.

Spacetrack Report Number 3 officially introduced five orbital propagation models to the user community—SGP, SGP4, SDP4, SGP8 and SDP8—all "generally" compatible with the TLE data. At the time, SGP had just been replaced by SGP4/SDP4 (the latter having included deep-space perturbations). The SGP8/SDP8 model was developed to alleviate deficiencies of SGP4/SDP4 for the special cases of orbital decay and reentry. The approach provided a closed-form solution based on the general trends of orbital elements as they neared reentry, and was quite successful. However, there is no evidence to suggest that SGP8/SDP8 was implemented for operational TLE formation.

After STR#3, *Spacetrack Report Number 6* (Hoots, 1986) was publicly released by North American Aerospace Defense Command (NORAD). Some researchers initially assumed this release was intended to update portions of the SDP4 deep-space routines, but the actual intention was to document HANDE[*] and had little to do with SGP4/SDP4. Nevertheless, it provided amateur satellite trackers and researchers with a confirmation of identified deficiencies in the original validation and verification efforts. This report has not been as widely circulated as STR#3, which benefited from its early electronic availability (Kelso, 1988).

In the early 1990s, the NASA Goddard Space Flight Center (GSFC) obtained a copy of the 1990 standalone SGP4 code[†] from project SpaceTrack as part of a study on orbit propagation models for the SeaWiFS Mission (Patt et al., 1993). In 1996–7 they released the unrestricted code on the Internet and to numerous organizations around the world involved in the SeaWiFS Mission. It confirmed changes already discovered by many independent researchers, and we refer to it simply as the "GSFC version."

In 1998, Hoots published a history of the equations, background, and technical information on SGP4. In 2004, Hoots et al. published a complete documentation of all the equations (including the deep-space portion). These publications cover the incorporation of resonances, third-body forces, atmospheric drag, and other perturbations into the mathematical technique. We note that all published reports on SGP4 have suggested only improvements in the code used to implement it, and not any changes to the underlying theory. Thus, the equations in Hoots (2004) should be representative of the current mathematical theory. This is a fundamental and essential assumption we use in this paper.

Outside the DoD, perhaps the most comprehensive external version of the software resided with Paul Crawford. His "Dundee code" kept track of the many changes inferred by real-world observations by independent researchers, and those confirmed by DoD releases. Many of the results contained in the code pre-date matters that were later confirmed in the DoD standalone releases. We use the change history from the Dundee in this analysis.

A. Motivation

Spacetrack Report Number 3 noted the importance of using the specific equations and data input to ensure proper operation and we repeat it here. "*The most important point to be noted is that not just any prediction model will suffice… The NORAD element sets must be used with one of the models described in this report in order to retain maximum prediction accuracy.*" The numerous releases and modifications to the original SGP4 standalone code have made it virtually impossible to satisfy that direction today. For instance, using element sets generated with the operational SGP4 code will not reproduce the same ephemeris as the original STR#3 code (without modifications) would. Similarly, using this TLE data in another general perturbations propagator will result in completely erroneous results. Simply converting the orbital elements to an osculating state vector and propagating with a numerical propagator is equally invalid. These are consequences of the model-based parameter estimation technique of orbit determination, and are most noticeable when using general perturbation techniques.

In fact, one may infer that none of the public releases meet this criterion because Kaya, et al. (2004) says *"Air Force Space Command (AFSPC) developed Astrodynamic Standard Software to emulate the operational astrodynamic algorithms used by the Space Control Center (SCC) in the Cheyenne Mountain Operations Center (CMOC)"* by "*extracting desired algorithms from the larger programs in the Space Defense Operations Center (SPADOC) within the SCC."* Thus, there are multiple versions of the SGP4 code even within the DoD. We must recognize that the true official code is inextricably linked and embedded within the operational computer system at CMOC (we designate it as the "operational" version). CMOC uses this operational version to produce all the TLE data that are distributed daily to worldwide users. A similar "standalone" version of the official code is maintained

[*] The HANDE model was intended to replace the analytical SGP4/SDP4 model. It incorporated the effects of the Jacchia dynamic atmosphere models for the average solar flux during the propagation interval, while retaining the speed and character of an analytic general perturbations model. It also included the full Brouwer gravity solution, much of which had been dropped for the SGP4 simplification. The code was implemented in the operational system, but its use is unknown.

[†] It appears that the merged SGP4/SDP4 models were now referred to simply as 'SGP4' from this 1990 code onwards.

by technical offices within AFSPC, which, under various organizational names,[*] published the Spacetrack series of reports. The mention of emulating the operational codes leads us to think that AFSPC routinely tests and aligns these two versions for compatibility. *Spacetrack Report Number 3* report contained a snapshot of this standalone code in 1980 and is the basis for our discussion.

Kaya et al. (2001) note the lack of enforcement for early AFSPC instructions (publicly available administrative documents) concerning the use of their standalone code, and discusses changes in AFSPC policy about releasing code. We see this in the evolution of Air Force Space Command Instructions. These documents imply that models and computer codes have been extracted from larger programs, modified frequently, and that those modifications are not promulgated or available to the broader user community.[†]

Perhaps the best motivation for the paper came from a 1998 version of AFSPCI 33-105, which stated,

> The need for this instruction was identified by the lack of any HQ AFSPC procedures for releasing a certain set of software, commonly called the "astrodynamics algorithms," used in the Space Defense Operations Center system (SPADOC 4C) for the space control mission. With no configuration control in place, various versions of executable and source code of the "astrodynamics algorithms" have been used for certain contracts and research projects.
>
> 1.1. Over the past 15 years or so, various commercial companies have produced and marketed products that these companies claim contain some of AFSPC's astrodynamics algorithms. Not only are these claims very difficult to confirm, very few of these claims, if any, have ever been confirmed. Also, in many cases, AFSPC has no documentation that states why, when, and from whom the contractor obtained the command's code. Consequently, AFSPC and other DoD units may have purchased their own software, often unknowingly.
>
> 1.2. Frequently, the algorithms and code contained in these products were outdated versions or had even been modified without consultation and certification from AFSPC. Additionally, the contractor rarely provides source code of their proprietary system to AFSPC so AFSPC cannot confirm whether the system's software actually contains the AFSPC "astrodynamics algorithms." Consequently, AFSPC cannot perform verification and validation that the astrodynamics algorithms have been utilized correctly in decision support systems, potentially critical to the space support provided to other combat units. Because of the severity of the problem with AFSPC's astrodynamics algorithms, an overall instruction for all of the command's software is required.

Thus today, there are perhaps more versions in use than at the time of original publication and compatibility and interoperability for users has been impacted. Many organizations routinely use a "version" of SGP4 that they received from "someone" at "sometime". Precise documentation is often scarce. Thus, a primary motivation for this paper is to bring the community up to speed with respect to the current implementation of SGP4 and the TLE data released by NORAD.

B. Purpose

The technical community has increasingly sought more information about SGP4 because its TLE data set continues to be widely disseminated even today and represents the only 'public' source of data covering the majority of orbiting objects. Although many of today's most important operations have switched to numerical processing methods, the analytical approach still has value, especially when dealing with large numbers of satellites. Examples of these include:

- Rapid searches for satellite visibility for ground stations, and generation of communication schedules.
- Programmed tracking of medium beamwidth antennas (or initial acquisition for narrow beamwidth auto-track systems) using limited CPU power embedded devices.
- Investigations into initial orbit design based on low-precision requirements, such as general sensor and/or ground station visibility statistics.

---

[*] We provide background information on some of the organizational acronyms used within this paper in the Appendix as they may be confusing.

[†] In the late 1990's AFSPC formalized the STR#3 advice and implemented regulations mandating procedures pertaining to the use and distribution of the standalone code stating in a 1998 version of AFSPCI 33-105 that, *"AFSPCI 60-102, Space Surveillance Astrodynamics Standards, requires that legacy government astrodynamics software be used in new systems to ensure interoperability with Space Defense Operations Center system (SPADOC 4C) orbital data and to reduce acquisition costs by using verified and validated standard astrodynamics algorithms that are Government Off-The Shelf (GOTS) software".* The 2004 version of AFSPCI 33-105, says, "*AFSPCI 60-102, Space Surveillance Astrodynamic Standards, mandates that only standard constants, physical models and astrodynamic algorithms will be used in all AFSPC systems requiring space vehicle trajectory data from or providing space vehicle trajectory data to the Space Control Center (SCC),*" implying that standards at that time were not "legacy … software."

- Rapid assessment of close conjunctions (http://CelesTrak.com/SOCRATES) (Kelso and Alfano, 2005) can be made computationally efficient by pre-processing with analytical techniques, and then applying numerical techniques only to those cases that appear to warrant additional consideration.

This paper provides source code, test cases, results, and analysis of a version of SGP4 designed to be similar to the standalone code. Because the complete equations for SGP4/SDP4 are given in Hoots et al. (2004), they are not repeated here. Instead, the focus is more on the actual code development, testing methodology, and results. The references at the end of the paper attempt to list the various papers that document the SGP4 theory and practice. This will establish a consistent new baseline and permit improved accuracy of operations for worldwide users that routinely process TLE data. The TLE are routinely available from CelesTrak (http://CelesTrak.com) and AFSPC (www.space-track.org). The basic format for the data has not changed much over the years and is described in many places and we have included a discussion in the Appendix.

II. PROGRAM INTERFACE ISSUES

A few technical questions and comments are necessary to effectively integrate these analytical solutions into today's environment.

A. Theoretical Issues

TLE data support a mix of coordinate systems and analytical theories. The SGP theory was largely based on Kozai (1959), while the SGP4 theory was primarily based on Brouwer (1959). The two theories are rather different, but both are still in use today. Neither the Kozai nor Brouwer theory originally included drag effects, so different treatments of atmospheric drag are in use. SGP approximates drag via rate changes of mean motion (Hilton and Kuhlman, 1966), while SGP4 uses power density functions (Lane and Cranford, 1969; Lane and Hoots, 1979) that require a term that encapsulates the ballistic coefficient, Bstar (see Vallado, 2004: 113–116). Simplified force modeling and the batch-least-squares processing of observational data often yield a Bstar that has "soaked up" force model errors. Occasionally, one finds negative Bstar values, indicating erroneously that energy is being added to the system, but this is simply a consequence of the limited SGP4 force modeling with respect to the actual dynamical environment.

B. Configuration Control

TLE data do not reveal which version of SGP4 was employed to estimate the orbital parameters. Different definitions of the so-called True Equator Mean Equinox (TEME) coordinate system and time systems may also have been used at different times. Without a list of dates to synchronize these changes with historical TLE data, the user must decide which version of the SGP4 propagator might be consistent. Because the accuracy of the propagator is generally in the kilometer-level range (Hartman, 1993), this may not be a problem for most cases, but as we'll see shortly, some of the technical modifications can cause results to differ by hundreds of kilometers. This topic is perhaps the least likely to have a simple solution, but could potentially account for significant differences in ephemeris generation.

C. Data Formats

The TLE format appears to have changed slightly over the years, and numerous TLE data were disseminated with missing or erroneous values. Some of these cases simply test the error handling of the code and its ability to handle premature ending of the propagation.

The TLE Element Type is always set to zero for distributed data, although STR#3 suggests the following assignments: 1 = SGP, 2 = SGP4, 3 = SDP4, 4 = SGP8, 5 = SDP8. The TLE sets also use differing formats (e.g., use of leading zeros, or not). Sometimes, parameters are omitted within the TLE data (*e.g.*, a second time derivative of mean motion or Bstar drag term equal to zero). These variations can confound fixed-read implementations in a computer program. The parsing of the TLE files is a bigger problem in languages such as C where the fixed-position approach (common in FORTRAN) is unusual, and where the 'NUL' (zeroth in the ASCII collating sequence) has a special end-of-string significance. Additionally, there are possible differences between DOS-formatted text files (CR/LF for end of line) and UNIX format (LF only). Attention is paid in the conversion utility to account for these discrepancies and the parsing routine is kept separate from the SGP4 routines to permit users the option of tailoring their parsing needs for a particular operation.

The TLE format has a simplistic form of error checking by having a checksum character for each line; however, it is prudent to check for other 'fixed' aspects (such as the "1" and "2" for each line, matching satellite numbers on the two lines, variable ranges, etc.) since the modulo-10 checksum only provides a 90% detection rate for uniformly

random errors. Even with the checksum, there has been some ambiguity over the value assigned to the characters (the + sign in particular, which we believe should be zero), some additional explanation can be found on the web.*

D. Coordinate System

The actual SGP4 model has little need for any specific coordinate or time system (*e.g.*, the near-Earth part is rotationally symmetric about the pole), but when used for propagating TLE generated by DoD it becomes important to use the same coordinate system as the DoD orbit determination routines use. The commonly accepted output coordinate system is that of the "true equator, mean equinox" (TEME) (Herrick, 1971:325, 338, 341). An exact operational definition of TEME is very difficult to find in the literature, but conceptually its primary direction is related to the "uniform equinox" (Seidelmann, 1992:116, and Atkinson and Sadler, 1951). The intent was to provide an efficient, if approximate, coordinate system for use with the AFSPC analytical theories. Technically, the direction of the uniform equinox resides along the true equator "between" the origin of the intermediate Pseudo Earth Fixed (PEF) and True of Date (TOD) frames (Vallado, 2004:211, 221). It is found by observing that θ_{GAST82} may be separated into its components. Thus,

$$\begin{aligned} \vec{r}_{TOD} &= ROT3(-\theta_{GAST82})\vec{r}_{PEF} \quad and \quad \theta_{GAST82} = \theta_{GMST82} + Eqe_{82} \\ \vec{r}_{TEME} &= ROT3(-\theta_{GMST82})\vec{r}_{PEF} \\ \vec{r}_{PEF} &= ROT3(\theta_{GMST82})\vec{r}_{TEME} \end{aligned} \tag{1}$$

We recommend converting TEME to a truly standard coordinate frame before interfacing with other external programs. The preferred approach is to rotate to PEF using Greenwich Mean Sidereal Time (GMST), and then rotate to other standard coordinate frames. Conversions are well documented from this point. To implement, you simply apply a sidereal rotation about the Z-axis by GMST (using UT1 as we discuss later). Because polar motion has been historically neglected for General Perturbation (GP) applications, we assume that the pseudo Earth-fixed frame is the closest conventional frame.†

If a rotation is made to TOD using the equation of the equinoxes, several approximations are introduced with the calculation of the nutation of the longitude ($\Delta\Psi$) and the obliquity of the ecliptic (ϵ). There are at least three possible sources of uncertainty with this method: the number of terms to include in the nutation series, the inclusion of the post-1996 "kinematic correction" terms to the equation of the equinoxes, and small angle approximations. After choosing the length of the IAU 1980 nutation series (4, 10, and 106 terms are popular choices with 4 being most common), the transformation is sometimes further reduced by assuming that $\Delta\Psi \approx 0$, $\epsilon \approx \bar{\varepsilon}$, and $\Delta\epsilon \approx 0$. This results in a nutation matrix that is significantly simpler than the complete nutation matrix, although the complete form is more common today. The equation of the equinox may be approximated by ignoring the "kinematic correction" terms starting in 1997 [such that $EQ_{eqe1980} \approx \Delta\Psi\mathrm{COS}(\epsilon)$]. Finally, because some of the multiplicative quantities are small, second-order terms may be neglected.

However, you should be aware of an additional nuance, specifically the 'of date' and 'of epoch' formulations.

- TEME of Date—With this option, the epoch of the TEME frame is always the same as the epoch of the associated ephemeris generation time. The transformation to ECEF is done by first finding the conversion from TEME to TOD (third equation in Eq. (1)). Next the standard transformation from TOD to ECF is computed. We could have gone directly to PEF without the TOD frame (second equation in Eq. (1)), but this implementation enables comparison with the TEME of Epoch approach. All transformations are found using the complete IAU-76/FK5 formulae, including nutation.
- TEME of Epoch—In this approach, the epoch of the TEME frame is held constant. Subsequent rotation matrices must therefore account for the change in precession and nutation from the epoch of the TEME frame to the epoch of the transformation. This is accomplished by finding a static transformation from TEME to J2000—this includes the equation of the equinoxes, the nutation, and the precession which are all calculated at the epoch of the TLE. This static transformation is applied at each time requested in an

* The data available on CelesTrak undergoes extensive testing prior to publication. This includes checksum, individual column checking (e.g., a number field can only have 0 – 9, a decimal field only a period), and range checking, where appropriate (e.g., inclination between 0 and 180). Not all archive sources of TLE have had such checks performed, and end users are advised to consider this aspect before using those TLE. Additional information can be found at the CelesTrak website. http://celestrak.com/NORAD/documentation/checksum.asp

† We assume that CMOC orbit detemination approximates the reference frames of radar and optical differently, and that numerical and analytical orbit determination methods use different techniques due to the differences in TEME, ECI, and the uncertain use of polar motion in coordinate systems.

ephemeris generation. Once the J2000 vector is found, standard techniques can convert this to other coordinate systems, at the appropriate time. This is computationally intensive, and introduces error into the subsequent solutions. All transformations, after the initial static calculations, are computed using the complete IAU-76/FK5 formulae, including all terms of the nutation theory.

Researchers generally believe the 'of date' option is correct, but confirmation from official sources is uncertain, and others infer that the 'of epoch' is correct. To be complete, we provide the equations and an example problem of both in the Appendix.

E. Time System Issues

Time accounting within SGP4 is referenced to the epoch of the TLE data. This practice makes individual satellite ephemeris generation and use relatively easy, although it can complicate multiple satellite analyses. The time system is assumed here to be UTC, but no formal documentation exists and UTC, as currently defined, was only introduced in 1972. UT1 is needed to calculate GMST for the coordinate transformations discussed in the appendix, but it is unknown whether UT1 or UTC is what is required by the software, although we assume UT1 for this paper. The error associated with approximating UT1 with UTC is within the theoretical uncertainty of the SGP4 theory itself. Except for the GMST calculation, this paper and code assumes time to be realized as UTC.

Time accounting also affects how the year of epoch values are handled within a system. This feature is only peripherally related to SGP4, and not part of the mathematical definition. It appears in the epoch calculations and affects how the two-digit year of the TLE is treated. Several possibilities exist. If the year is less than 50, 57, or some other value, one can add 2000, otherwise, 1900 is added. Of course, these are only temporary fixes with the correct option to be the use of a 4-digit year, Julian Date, Modified Julian Date, etc. During the so-called "Y2K" millennial rollover, some attention was focused here although nothing apparently changed.

It is doubtful that a leap-second capability was implemented into the peripheral software for SGP4 since the historical source code uses relative "time since epoch". Any such addition is clearly outside the mathematical formulation of SGP4, but necessary for programs to interface with other agencies. As some software libraries have no support for the '61 second' minute that is needed to properly represent or convert UTC time at the point of leap-second insertion, we suspect this is simply ignored for the majority of non-critical users, and a 1-second timing difference will occur sometime during the period between TLE updates where the leap second is added or subtracted. Although this is outside the direct scope of SGP4, it is part of many System Acceptance Tests for large programs, and is included here as a reminder of those operations.

F. GHA Calculation

The Greenwich Hour Angle is usually calculated using the Julian Date. However, you can also find expressions using the elapsed time from some epoch. Among the versions of SGP4 that are available today, several epochs arise: 1950 Jan 1 0^h, 1970 Jan 0 0^h, and 1970 Jan 1 12^h, UT1. The various combined constants illustrate the potential for error when using this approach. As new timing systems are developed, the associated timing parameters change slightly. The precision of these parameters also change slightly. Consider the following examples from various versions:

```
Jan 1, 1950 0 hr (original STR#3)
   THETA = 1.72944494D0 + 6.3003880987D0*DS50
Jan 0, 1970 0 hr
      C1    = 1.72027916940703639D-2
      THGR70 = 1.7321343856509374D0
      FK5R  = 5.07551419432269442D-15
      C1P2P = C1+TWOPI
      THGR  = DMOD(THGR70+C1*DS70+C1P2P*TFRAC+TS70*TS70*FK5R,twopi)
```

These approaches yield "essentially" the same values. A series of calculations were constructed to test these against the IAU convention (Vallado, 2004:191) using the Julian centuries of UT1 (T_{UT1}).

$$\theta_{GMST1982} = 67{,}310.548\,41^s + (876{,}600^h + 8{,}640{,}184.812\,866^s)T_{UT1} + 0.093104T_{UT1}^2 - 6.2x10^{-6}T_{UT1}^3 \qquad (2)$$

The results showed comparisons of about 10^{-9} degrees difference. This is well below the level that the answers would be affected. We have chosen to implement the conventional approach of Eq (2).

III. COMPUTER CODE DEVELOPMENT

The revised computer code developed in this paper is provided in C++, FORTRAN, MATLAB, and Pascal to permit reasonable flexibility for applications (C++ is given in the appendix as this language is becoming

commonplace). Conversion to other languages should be aided by the re-structuring effort that has been performed on the code.

There can be large variations between the numerous implementations of SGP4—hence the need to establish a newer baseline that is compatible with CMOC as closely as possible to provide enhanced compatibility. Where obvious updates and corrections have been made and verified, we account for each in our revised code. For other improvements that appeared "obvious" to us, we tried to determine if these changes might be present in today's standalone version.

Our starting point was the 1980 version from STR#3. From this point, STR#6, the Dundee modifications, and the GSFC code release verified several suspected code changes. There were too many changes in this update to describe them all—we list a few of the major ones below. Note that all satellite numbers refer to examples in the test case file (sgp4-all.tle). Also, all element plots are osculating values.

- A primary change from STR#3 was the merging of SGP4 and SDP4 code. A large number of researchers had noticed the commonality of the two models and simplified the code in this manner, however, not all had recognized and simplified the initialization code. Due to this simplification, most now refer to the merged SGP4/SDP4 models simply as 'SGP4'.
- Although ultimately STR#6 has little relevance for TLE use, one notable change was the move to double-precision code throughout (rather than the mix of single and double in the original) and corresponding increase in accuracy for certain astrodynamic constants, all made practical by the improvement in computing power since STR#3. Such changes do not improve the "accuracy" of the model as such, but they lead to "smoother" behavior which helps with some tasks (such as differential correction), and to greater consistency of results on differing computer systems and/or compilers.
- Solving Kepler's equation was updated, but not completely fixed. The solution of Kepler's equation continues to present challenges in astrodynamics hundreds of years after its introduction. The original 1980 version of SGP4 had a fixed limit of 10 iterations and a tolerance of 10^{-6}, but contained no code to prevent certain high-eccentricity orbits from failing to converge. *Spacetrack Report Number 6* changed the tolerance to the tolerance to 10^{-12} (commensurate with double-precision work) and the GSFC version tried to solve the convergence problem by removing the iteration limit. However, these are incomplete approaches and can still result in infinite loops. The revised version code includes an updated SGP4 routine following the Dundee version that allows realistic controls on the iterations. Figure 1 shows the impact of this practice.
- The practice of only computing the lunar-solar terms if propagation time changes by more than 30 minutes to save CPU effort was dropped, thus resulting in smoother behavior for deep-space orbits with small time steps. This was the only function of the SAVTSN variable in the original DPPER subroutine. This resulted in 'choppy' behavior in some ephemerides from the STR#3 version.
- The application of periodic lunar-solar perturbations was updated. There are actually three problems relating to the application of the periodic lunar-solar perturbations. The first of these, sometimes known as the "Lyddane bug" (because it was first noted in independent investigations of the Lyddane modifications in DPPER), is due to the jump in the actan/atan2 output where the perturbed value due to the discontinuity of this function at either 90°/270° or ±180° (respectively). In the STR#3 code, the actan discontinuity occurred at 270°. *Spacetrack Report Number 6* tried using the atan2 function, but that simply moved the discontinuity to 180°. The GSFC code (the IF statements at the end of the 'apply periodics' section in DPPER) confirmed the suspicions of several researchers about the need to evaluate the relative quadrant of the resulting angle and to correct accordingly. A similar problem exists with the modulo 2π reduction of the XNODE variable. The effect of not correcting the quadrant is illustrated in Fig. 2. This problem also occurs when intrinsic functions (mod, atan, etc.) are used instead of the STR#3 versions. We feel intrinsic functions are better suited for the program, but that full envelope testing of the Lyddane implementation is probably in order.
- The second difficulty with the lunar-solar perturbations was the initialization of deep-space terms based on perturbed values. This was corrected in the DPPER and SGP4 routines of the Dundee and GSFC versions. In STR#3, the terms computed during initialization assumed fixed epoch values for inclination, etc., but of course they are perturbed by the deep-space terms. The approach used by the Dundee and GSFC versions includes any terms based on the Keplerian orbit being re-computed based on the new perturbed values.

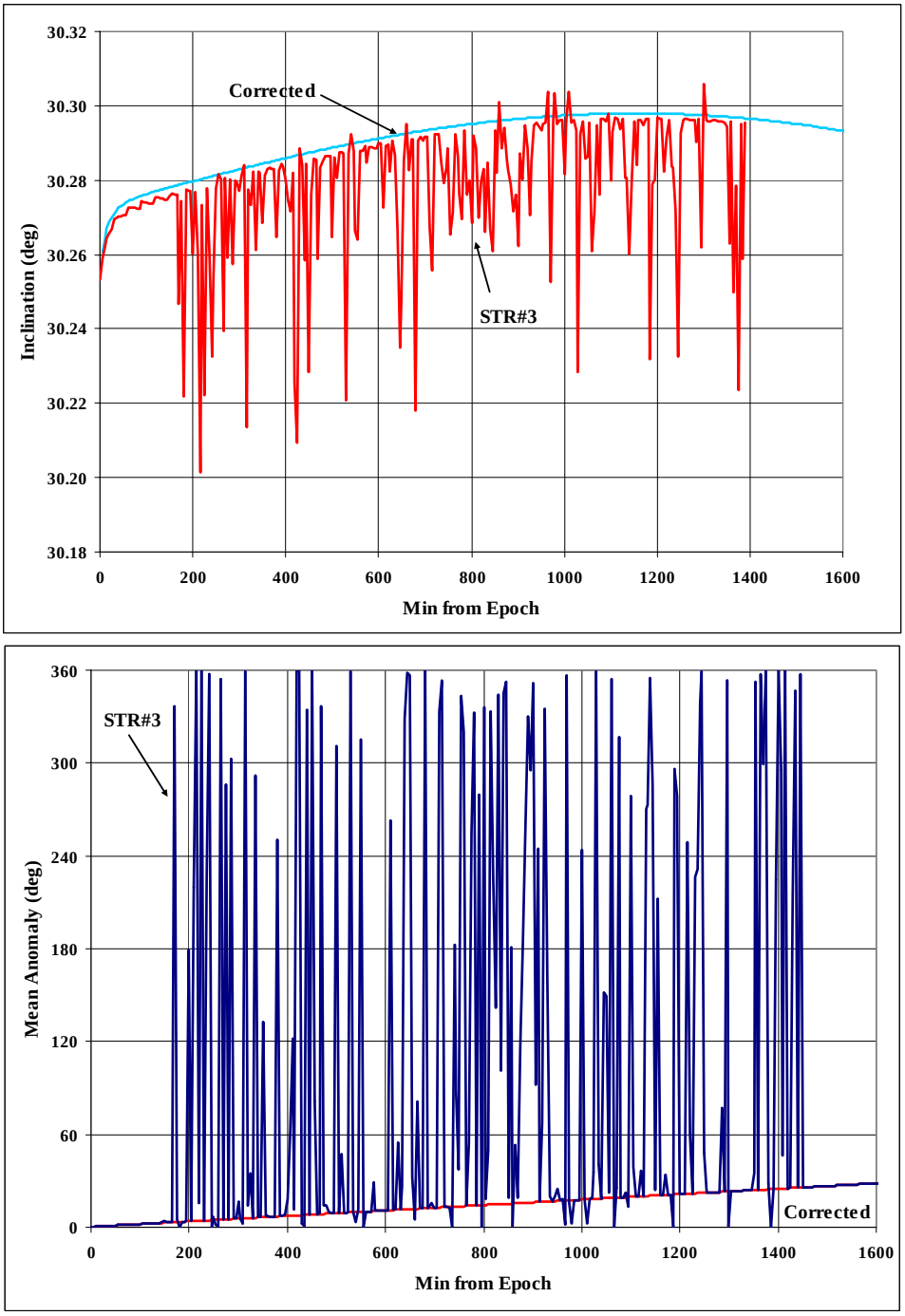


Figure 1. Solving Kepler's Equation for Satellite 23333. The mean anomaly (bottom) illustrates the severe discrepancy in incorrectly solving Kepler's equation after about 200 minutes. The effect also shows up in the inclination (top). The problem existed in the STR#3 version, shown here, but corrections were attempted in STR#6 and the GSFC version, with a better approach in the Dundee version. The inclination plot also shows the choppy (but smaller) behavior of the 30 minute updating of the lunar-solar terms in the STR#3 version before about 200 minutes. Notice that the effect goes away after about 1400 minutes.

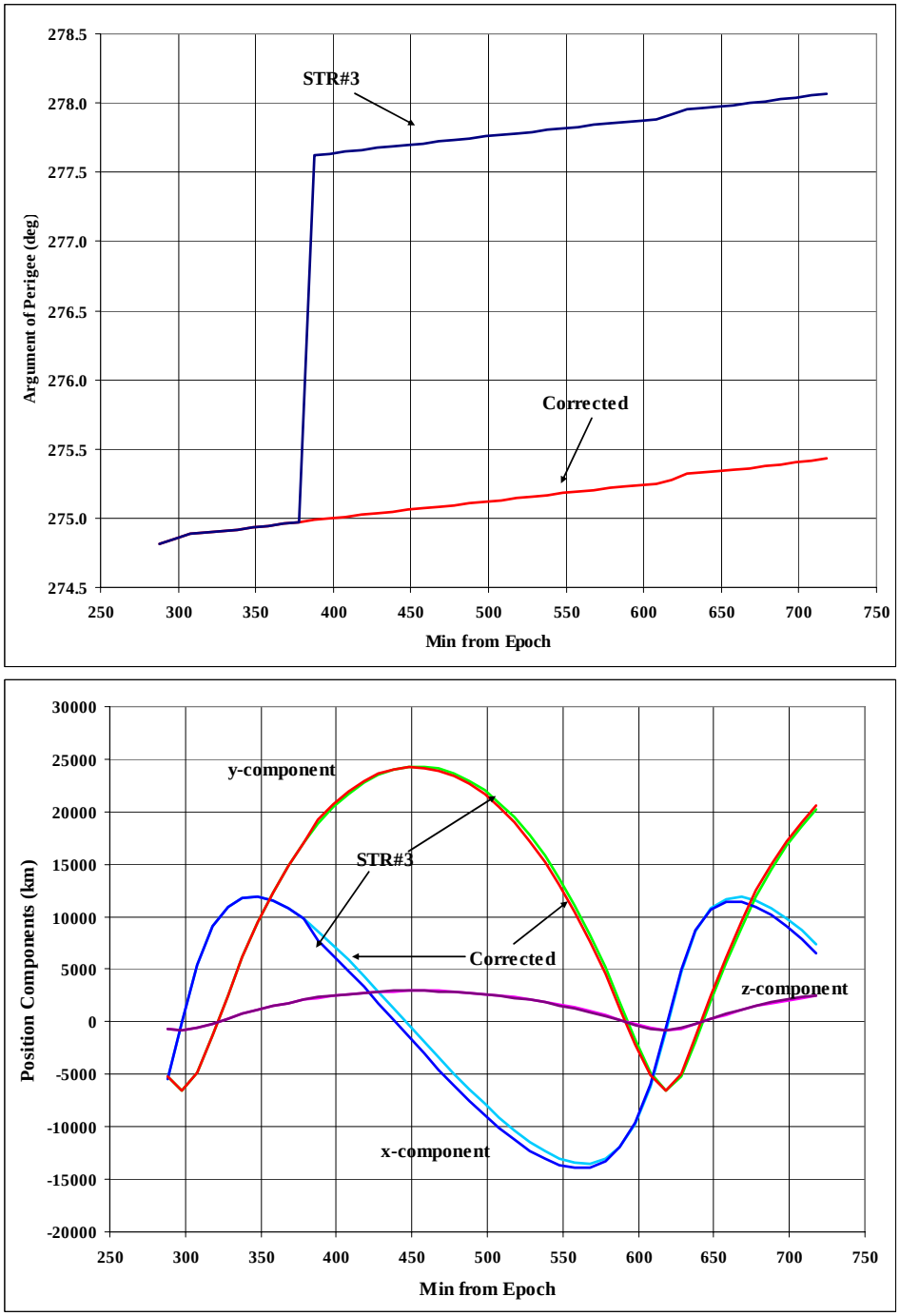


Figure 2. Lunar-solar modifications for Satellite 23599. The argument of perigee and positional components illustrate the discrepancy in incorrectly updating the lunar-solar perturbations, and not accounting for the proper quadrant in the periodic calculations. The problem existed in the STR#3 version, shown here, and an attempted correction was made in STR#6, but it was mostly corrected in the GSFC version.

- The third area of confusion with the lunar-solar perturbations is the decision for when to use the Lyddane modification, and we refer to it as the "Lyddane choice" (Satellites 14128, 20413). Lyddane (1963) reformulated the Brouwer expansions (done in Delaunay variables) in Poincare variables. Both are canonical, and the Poincare variables were intended to be non-singular for small eccentricity and inclination values. Because this was a reformulation, its use was intended for all computations, and

remains that way today in the Navy Position Partials and Time (PPT3) (Hoots et al., 2004). During the development of SDP4 equations, some SIN(inclination) divisor problems were noted and the Lyddane formulation was examined. Because it also exhibited singularities, an alternate formulation was sought, ultimately resulting in new parameter choices. The decision was made to use these variables when the inclination was less than 11.4592º (0.2 rad).

Thus, the code implementation introduced two methods of applying the lunar-solar perturbations in the deep-space code, with the Lyddane modification used with smaller inclinations to avoid a divide-by-zero type of computation problem. In the STR#3 version, the choice was based on the unperturbed epoch inclination proximity to 11.4592º. In STR#6 (and the GSFC code subroutine DPPER), the test used the perturbed inclination (XIP, with the secular term applied, unlike the XQNCL common term used in STR#3). This approach leads to a potential for the model switching lunar-solar methods as a function of propagation time, which is clearly undesirable. Note that the difference is usually small and relies on positional differences rather than the actual positional values. However, the results can be greatly magnified in some orbits (satellite 20413 which is a multi-day orbit). The basic effect can be demonstrated by satellite 14128 after about 2000 minutes from epoch when the perturbed inclination emerges above 11.4592º as shown in Fig. 3.

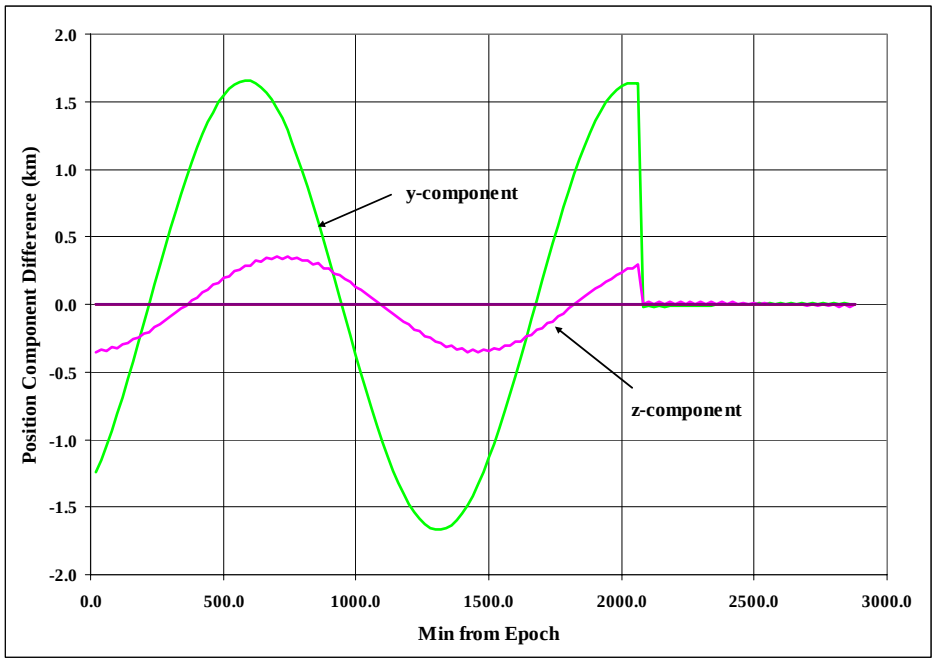


Figure 3. Lyddane Choice Modification for Satellite 14128. The difference in positional components illustrates the discrepancy in applying the Lyddane modification using the perturbed inclination rather than the original inclination. The effect exists when comparing the GSFC and STR#3 versions. Note the differences diminish once the inclination crosses the 11.4592º threshold (after 2000 min).

A few comments are necessary on the Lyddane choice. This case occurs exceedingly infrequently as the satellite inclination must be very close to the 11.4592º limit, and it must be a deep-space satellite. Without carefully crafted test cases, this (like several of the problems we discuss) is not easy to detect in normal operation. At the time of coding the STR#3 version, this form of software testing was not a commonly taught practice.* We can consider several methods of resolving the situation, such as (a) going back to the STR#3 practice of testing the unperturbed epoch inclination at $t = 0$ and using that as a fixed decision for the TLE, (b) testing the perturbed inclination at each propagation time (as in the GSFC version), or (c) making more significant code changes to find a smoother way of

* In 1980, the limitations of computer memory and storage space often dictated stringent code length requirements. Code that would only be exercised once or twice, if a small effect, could be safely omitted in deference to more critical techniques, applicable to numerous satellites. The testing philosophy of the time also influenced the outcome. Using a single set of test cases for all analyses was quite common. The notion of targeted test cases for individual loops and constructs in the code itself didn't arise until many years later. Thus, this small nuance could easily have been missed by the testing of the time, or by code limitations themselves.

blending the two lunar-solar perturbation methods. Although a rare case, we think some fix should be included and hope that AFSPC will confirm the current state of the code so users can be compatible in all cases. For the present paper, we have included Option (b) in the code with qualifiers to assist in location and potential future resolution. However, we note that there is probably a better "crossover" point to apply the Lyddane modification (Option c) that will not result in such large discrepancies in the two ephemerides, but time did not permit a thorough investigation and recommendation for such a change.

Next, the following changes were made to comply with modern programming standards, and to facilitate any changes in the future. With the exception of the variable precision and the integrator issues (discussed later), none of these changes affected the technical performance of the program and could be considered "cosmetic".

- Implicit typing in FORTRAN was replaced by comprehensive variable declarations. This was a critical step before conversion to C++ and others. Modern compilers can generally sort out the variable names, but the possibility of mistaken variables, variables being set to zero and used in calculations, etc., was too great. In addition, knowing which variables were calculated and set assisted the process of forming structures. Finally, this also eliminated much of the need for the FORTRAN SAVE command to hold values between function calls with certain compilers.
- Structures were created to pass the large amounts of data between functions. Numerous variables were passed between functions in the original code. With no typing in the original code, this approach proved relatively easy, but it was difficult to gain an understanding of the underlying structure. The structures were set up to support integrated near-earth and deep-space functionality provided in the code. This change also supported processing multiple satellites at one time. While processing a single satellite is illustrative for simple scenarios, it is unrealistic for many modern applications. For instance, the SOCRATES effort uses TLE data to generate potential conjunction information. During these runs, one must have two or more satellites in memory at one time.
- GOTO statements in FORTRAN were eliminated, using more modern constructs. This old programming construct is often seen in legacy programs, but completely unnecessary with modern programming techniques and tools. Looping and decision constructs were inserted, as appropriate.
- Intrinsic functions replace user-written routines. Trigonometric and exponential routines should use intrinsic calls within the programming language. The only exception should be in cases where a specific quadrant, ordering, etc. is required. None of these were deemed necessary within the SGP4 routines.
- Initialization functions were separated for better organization. The code was modularized, keeping initialization functions separate from routine function use. Although modern compilers can generally sort these differences out, the code is easier to maintain if the functions are isolated for a particular operation. The reorganization of the computer code simplified the processing flow. In addition, simple timing studies performed during the original development demonstrated increased processing speed of about 10%. The basic program structure is illustrated in Fig. 4.
- Variable names were changed to better conform to the variables they represent. Many variable names were limited to conform to the former FORTRAN limitation of six (6) characters. This is no longer necessary and has been dropped. Variable names were changed to match "standard" nomenclature, such as that used throughout Vallado (2004). Constants were kept as constants in the code, and not assigned as variables with limited precision.

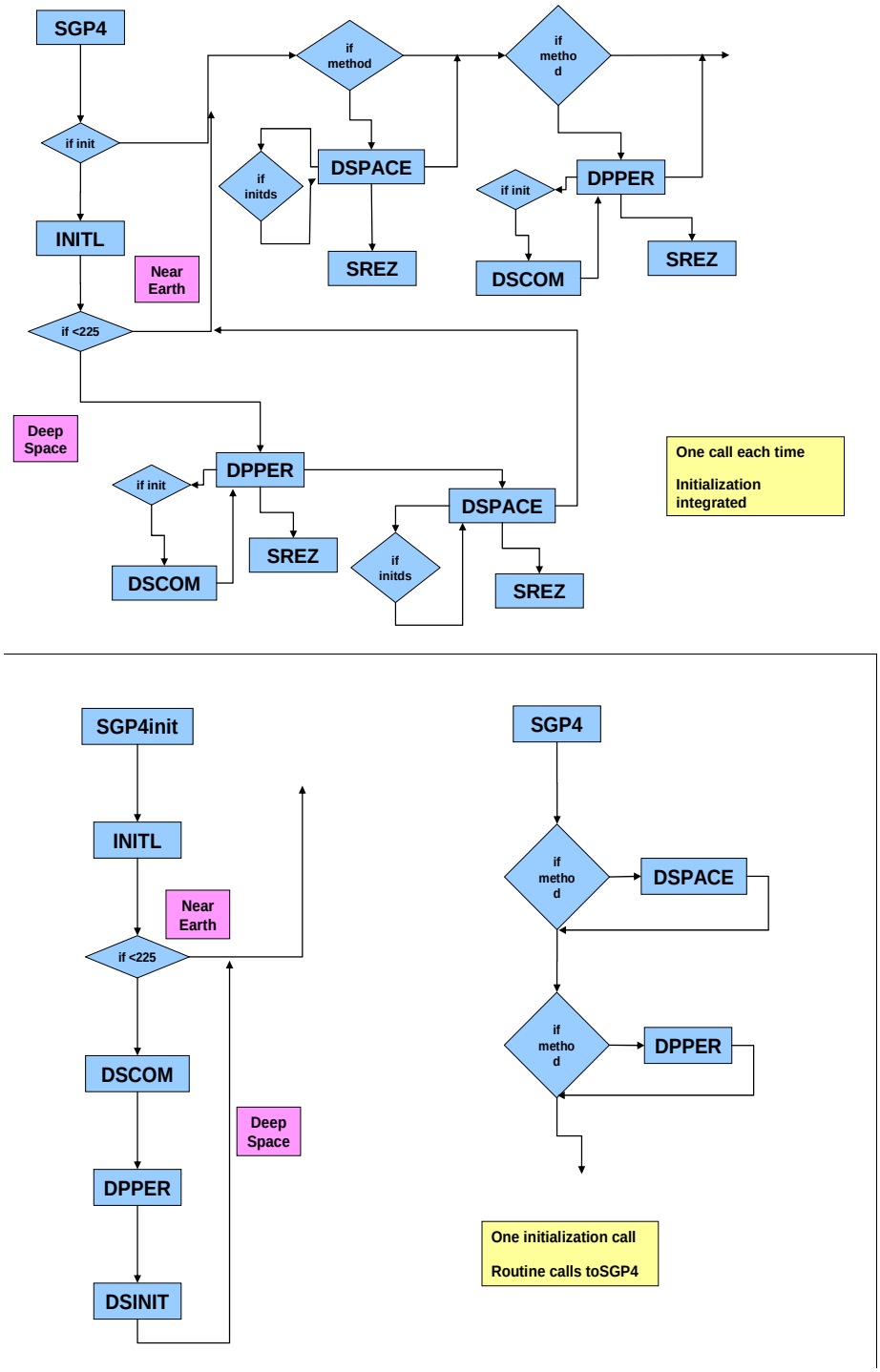


Figure 4. SGP4 Structural Organization. The computer program structure is shown for the original and derivative programs (top) and the revised version for this paper (bottom). Note that the initialization was interspersed throughout the original program, while it is better isolated in the revised code.

IV. SAMPLE TEST CASES

The original STR#3 included several test cases and sample outputs, but only for two sample satellites. Given the number of branches possible in the deep-space case, many more tests are needed to fully test the code. The original cases have been extended over the years as users have encountered real-world situations. The only other official test cases are referenced in AFSPCI 60-102. No publishing information is given in the AFSPCI.

Because the theory is based on analytical expressions, comparisons are relatively simple because the output should be the same from each program. Different programming languages (C++, FORTRAN, MATLAB, or Pascal) and compilers produced very small differences, but these were well below the accuracy of two-line element sets that are commonly used, and below the comparison between differing implementations.

For analysis, the computer code was set up with three primary execution paths. First, there is a "verification" path in which the program accepts an input TLE file that includes start and stop dates and time steps. Mechanizing this step was important to quickly review any changes against "known" test results. The second mode processed the entire space object catalog from one day before to one day after the epoch time. The negative time propagation was chosen to highlight any difficulties in the secular integrator part of the deep-space code—a most convoluted example of programming in the official versions. Several space object catalogs (having about 9000 satellites in them) were tested from the historical database. This provided a quick-look at performance for each of the programs against a wide range of satellite orbits. The third mode of operation is the standard mode whereby input element sets are read, and some operation takes place with the data. We separated the driver and TLE-conversion function from the SGP4 code, to permit a user to modify the driver as needed, without having to change the underlying SGP4 code.

The test cases were divided into two categories. First, there were verification runs that tested the basic algorithm implementation. The second set of tests demonstrates cases that we believe indicate additional technical considerations that AFSPC may have incorporated in their models, or should consider in the future.

V. VERIFICATION TEST CASES

Essentially, these cases allowed several features of SGP4 to be tested, but the answers were generally agreed upon during the testing phase of research for this paper. Cases for which there were technical questions about how the code was implemented are discussed in a subsequent section. The element sets were sorted numerically in the computer file to aid location of specific test cases, but are grouped here by effect. Comments were added to indicate what each test was accomplishing. The original SGP4 model had two types defined in the code, normal (near Earth) and 'simplified drag', while the original SDP4 had three types, normal (deep space), resonant (12^h Molniya style) and synchronous (24^h GEO). Table 1 shows a sample. The file (sgp4-ver.tle) is on the Internet at the web site listed at the end of the paper, and in the Appendix.

Table 1. SGP4 Verification Test Cases. These satellites highlight the primary test cases used for analysis and verification of the SGP4 code. A few other satellites are included in the full test set. The satellites used for the figures are also included, but at a reduced ephemeris density. The file gives the applicable time range in minutes from epoch (MFE). The original STR#3 tests are kept for continuity.[*]

Satellite	Category	Comments
00005	Near Earth	TEME example satellite.
28129	Deep Space	A GPS navigation satellite in a near circular 12^h orbit.
26975	Resonant	Molniya style debris launch. Exercises the 0.5 to 0.65 eccentricity branches in deep space.
08195	Resonant	Molniya launch. Exercises the 0.65 to 0.7 eccentricity branches of the deep-space code.
09880	Resonant	Molniya launch. Exercises the 0.7 to 0.715 eccentricity branches of the deep-space code.
21897	Resonant	Molniya launch. Exercises the eccentricity branches above 0.715, with a negative Bstar value.
22674	Resonant	Rocket body, similar to 21897 ($e > 0.715$) but positive Bstar
28626	Synchronous	Low-inclination (< 3 deg) geostationary orbit that shows the problems in premature correction of negative inclination at around 1130 minutes from epoch.
25954	Synchronous	Low-inclination GEO case like 28626, shows negative inclination problem at around 274

[*] All TLE data given in this paper is representative of actual satellites and can be obtained from www.CelesTrak.com except for the original Report #3 test cases which do not appear in the archives.

Satellite	Category	Comments
		minutes from epoch.
24208	Synchronous	Geostationary orbit above 3 deg.
09998	Synchronous	Relatively high eccentricity for GEO (e = 0.027) shows secular integrator problem clearly.
14128, 04632	Synchronous	Geostationary orbit close to 0.2 radian inclination. Shows Lyddane choice problem at about 2080 minutes and about −5000 minutes from epoch.
20413	Deep Space	Long period orbit (~4 days) shows Lyddane choice at 1860 minutes from epoch.
23333	Deep Space	Very high eccentricity, shows Kepler solution problems in Report #3 code.
28623	Deep Space	Deep-space object with low perigee (135.75 km) that uses the branch (perigee < 156 km) for modifying the 's4' drag coefficient.
16925	Deep Space	Deep-space object with very low perigee (82.48 km) that uses the second branch (perigee < 98 km) for limiting the 's4' drag coefficient to 20
06251	Near Earth	Near Earth normal drag case. The perigee of 377.26 km is low, but above the threshold of 220 km for simplified equations, so moderate drag case.
28057	Near Earth	Near Earth normal drag case but with low eccentricity (0.000 088 4) so certain drag terms are set to zero to avoid math errors / loss of precision.
29238	NE/S	Near Earth with perigee 212.24 km, thus uses simplified drag branch (perigee < 220 km) test.
28350	NE/S	Near Earth low perigee (127.20 km) that uses the branch (perigee < 156 km) for modifying the 's4' drag coefficient. Propagation beyond approximately 1460 minutes should result in error trap (modified eccentricity too low).
22312	NE/S	Near Earth with very low perigee (86.98 km) that uses the second branch (perigee < 98 km) for limiting the 's4' drag coefficient to 20. Propagation beyond approximately 2840 min should result in error trap (modified eccentricity too low).
28872	NE/S	Sub-orbital case (perigee −51 km, lost about 50 minutes from epoch) used to test error handling.
23177, 23599	Deep Space	Lyddane bug at less than 70 min and 380 min respectively, with atan2(), but no quadrant fix
26900	Deep Space	Lyddane bug at 37,606 min, negative inclination at 9313 min
29141	Near Earth	Last stages of decay. Crashes before 440 min
11801/ 88888	Deep Space, Near Earth	Original STR#3 report test cases

VI. EXPECTED CODE UPDATES

Although we searched many locations to obtain the latest openly available documentation on official AFSPC practice, a few topics remain unknown. The primary areas of discussion are those giving the largest differences in results—specifically negative inclinations, integrator problems, and solution of Kepler's equation. If the reader is aware of other corrections, we would appreciate learning about them. The intention is to produce a new baseline that is as close as possible to the current operational version to enhance compatibility for the external user. While we could not verify these, we felt the changes were so obvious that AFSPC has already made them, thus we have included the options in the code. We used a comment (keyword "sgp4fix" in the codes) by each change to make any future retraction or addition easier. For official users who are constrained by the AFSPCI 33-105 restrictions and have only an executable version of the current code, it should be a simple matter to confirm these fixes.[*]

[*] The AFSPC instructions have applied to different entities over time. By August 2004, AFSPCI 33-105 states the instruction *"applies to Headquarters Air Force Space Command (HQ AFSPC), subordinate units, supporting activities and contractors who develop, acquire, maintain or deliver computer software, including all systems that require astrodynamic algorithms. It also applies to Air Force Reserve Command (AFRC) and Air National Guard (ANG) units gained by HQ AFSPC."* Previous versions incrementally added each of these groups, so it would appear that the scope of the intended audience is increasing with time.

A. Error Checking

We increased the amount of error checking in our code to handle cases such as decayed satellites, or satellites having inconsistent values. CelesTrak employs a significant amount of error checking on the TLE data, but programs allowing the user to enter data could result in values that would cause errors. Inclination values near 180.0 degrees can cause divide-by-zero problems in the initialization and the routine operation. This is fixed by setting a tolerance in both routines. The decay condition simply checks the position magnitude on each step.

B. Constants

Kaya et al. (2001, 2004) focuses on the difficulties encountered when mixing WGS-72 and WGS-84 constants. Because the SGP4 codes contain references to WGS-72, AFSPC may have updated the constants to WGS-84, but there is no other documentation supporting this so we present the development in case new official documentation is released. However, because many operational sites may still have embedded software containing a version of SGP4 using WGS-72, and the fact that the accuracy of the theory would not really be impacted, AFSPC may well have chosen to retain the older set of constants to better maintain interoperability with its internal resources. We use WGS-72 as the default value. As with other changes we discuss, this is only necessary to interface with external programs, but it will cause a difference in ephemeris results. The proper sequence to form the constants for WGS-72 is shown below. Note that we determined μ from the SGP4 code value of XKE because it is not specified directly in the code, and this makes future revisions easier. We also provide TUMin because XKE is simply the reciprocal of this quantity. TUMin is possibly more familiar as it is the number of minutes in one time unit—a necessary conversion when using canonical constants.

Table 2. WGS-72 Constants. The fundamental and derived constants are shown below. Notice that XKE and TUMin are reciprocal values. The original STR#3 listed XKE as 0.074 366 916.

Symbol	Calculation	Value
μ		398,600.8 km^3/s^2
$R_\oplus$		6378.135 km
J_2		0.001 082 616
J_3		–0.000 002 538 81
J_4		–0.000 001 655 97
XKE	$60/\sqrt{(R_\oplus^3/\mu)}$	0.074 366 916 133 17 /min
TUMin	$\sqrt{(R_\oplus^3/\mu)}/60$	13.446 839 696 959 31 min

If we use WGS-84 values, we find the following values.

Table 3. WGS-84 Constants. The fundamental and derived constants are shown below. The zonal harmonic values are converted from the normalized values.

Symbol	Calculation	Value
μ		398,600.5 km^3/s^2
$R_\oplus$		6378.137 km
J_2	$C_{2,0}$ = –0.000 484 166 850 00	0.001 082 629 989 05
J_3	$C_{3,0}$ = 0.000 000 957 063 90	–0.000 002 532 153 06
J_4	$C_{4,0}$ = 0.000 000 536 995 87	–0.000 001 610 987 61
XKE	$60/\sqrt{(R_\oplus^3/\mu)}$	0.074 366 853 168 71 /min
TUMin	$\sqrt{(R_\oplus^3/\mu)}/60$	13.446 851 082 044 98 min

Other constants may not be familiar at first. For example, XPDOTP is a conversion from rev/day to rad/min.

$$XPDOTP = 1440.0/2\pi = 229.183\ 118\ 052\ 329\ 3.$$

RPTIM is simply the rotational velocity of the earth in rad/min. Note this does not use the GRS-80 defining parameter for the rotation of the Earth, $2*\pi$ / (86,400/1.002 737 909 350 795) * 60.0 = 7.292 115 855 3×10^{-5}, but rather the GRS-67 value that Aoki et al. (1982) used in the definition of time.

$$RPTIM = 7.292\ 115\ 146\ 7\times10^{-5} * 60.0 = 0.004\ 375\ 269\ 088\ 02 \text{ rad/min}.$$

Other constants are combined with other values, or use the values mentioned previously in their formulation. We do not believe any update has occurred to any of the embedded constants in the deep-space portions as no documentation has ever suggested this.

C. Negative Inclination Orbits (Satellite 25954, 28626)

Deep-space orbits with low-inclination values (typically geosynchronous orbits) can, due to the effects of lunar and solar gravity, result in a negative inclination with time. This can create a step-function discontinuity in the positional components. Normally this is resolved by shifting the ascending node longitude by 180°. In the computer code, we corrected this by removing the quadrant check from DSINIT before the 'initialize resonance terms' section, but kept the check in SGP4 before the 'long period periodics' section.

Satellites 25954 (at times beyond 274 minutes) and 28626 (at times beyond 1130 minutes) illustrate the effect of correcting negative inclination prematurely. The bottom graph in Fig. 5 of *z*-position reveals a discontinuity around this time in incorrect implementations of the code.

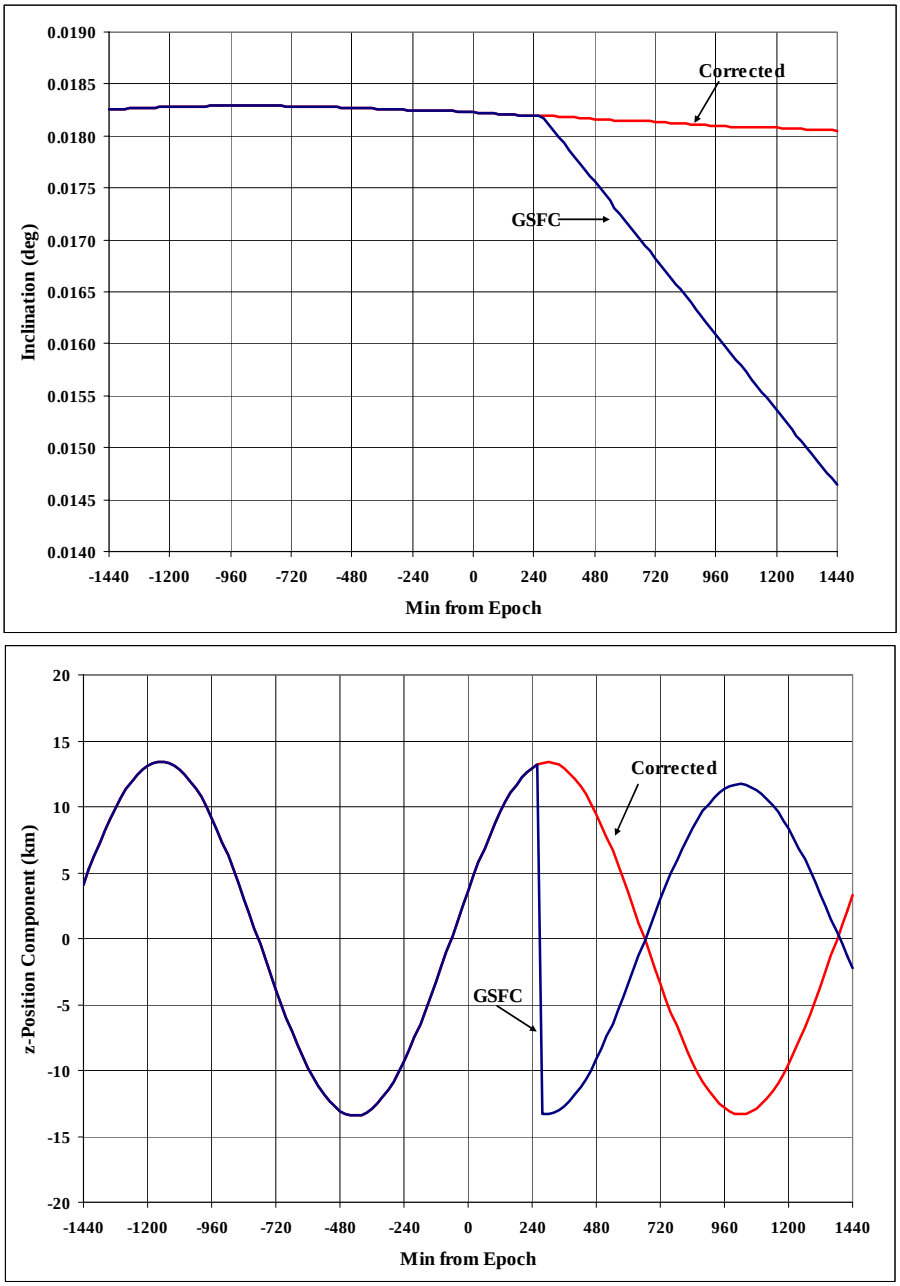


Figure 5. Negative Inclination Performance for Satellite 25954. The inclination and *z*-component of the position vector show the step function discontinuity of the previous SGP4 versions. The STR#3 and GSFC versions exhibits the problem while the Dundee version does not.

D. Integrator Problems (Satellite 09880)

The original FORTRAN codes contained a generous mix of GOTOs and other structures that made accurate debugging nearly impossible. One area that appears to have suffered from this practice was the secular integrator used for 12^h and 24^h resonant cases. In particular, several satellites show difficulties when propagated 'backwards,' that is going to some time away from epoch (either positive or negative time) and then taking time steps towards the epoch again. The problem seems to be with the setup of the positive and negative steps (stepp and stepn) with values of 720 minutes in the DSPACE routine (SREZ in the older programs). It appears that 'cleaning up' the code has fixed the problem. The original STR#3 style of logic would integrate from epoch to the required time using a Taylor series approximation:

$$F(x+h) = F(x) + \frac{h}{1!}F'(x) + \frac{h^2}{2!}F''(x) + \cdots \tag{3}$$

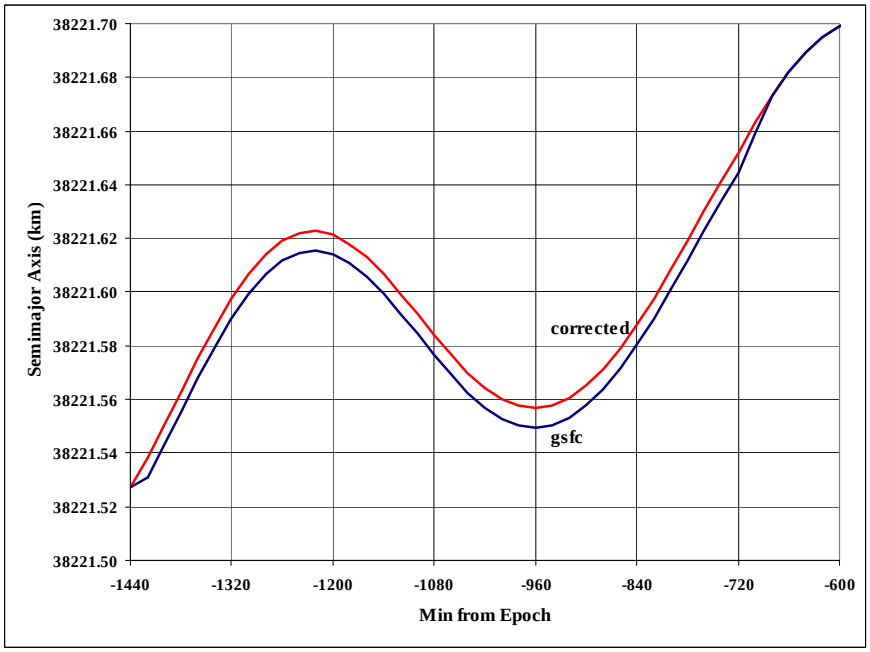
where the integral at epoch $F(0)$ is defined as zero. If the time (h) from epoch was greater than 720 minutes, it would step in 720 minute intervals (x = 720, 1440, …), recalculating the 1st and 2nd derivatives each time and saving the current (multiple of 720 minute) values for future use. Integrator resets occurred only when crossing the epoch. This was correct and efficient provided the model was only called with increasing time steps (either positive or negative), but it gives inconsistent results if you go to a time far from the epoch and return 'backwards' towards the epoch. The code from this paper always integrates from the epoch to the required time, and restarts each time the model is called with a 'backwards' step. This is slightly less CPU efficient but leads to repeatable results.* Satellite 09998 demonstrates this symptom quite clearly as it is propagated from one day before the epoch until one day after the epoch, though all of the resonant and synchronous cases show it to some extent. Consider Fig. 6.

E. Solving Kepler's Equation (Satellite 23333)

The partial fix discussed earlier handles a majority of the problem cases one would encounter in operations. However, additional robustness could be handled via several alternative methods. A simple but very effective fix for this is covered in Crawford (1995) where it is noted that the difference between mean and eccentric anomaly is never more than $\pm e$ radians, so if you limit the first Newton-Raphson correction to somewhere around this, it converges reliably for all cases. As this problem only applies to very high eccentricity orbits, an even simpler option fixes a limit of 0.9 – 1.0 for the maximum correction. Another option is that of Nijenhuis (1991), who examines the problem for eccentricities of 0.999 and 0.9999 and also examines the overall CPU load as well as the iteration count. Note that this iteration is not the 'traditional' iteration to find eccentric anomaly discussed in the literature (e.g., Vallado, 2004: 72–85). Results for this change were shown in Fig. 2.

A series of tests were run to determine the number of iterations for a complete satellite catalog, and the satellite tests we have included with this paper. For an example case of e = 0.9 the STR#3 version took an average of 5.685 iterations with a maximum of 8. The corrected Dundee version had an average of 3.984 iterations, with a maximum of 5. As with the Lyddane choice mentioned earlier, this change affects only a very small number of satellites.

* The ProjectPluto code had a variation on this method. It always integrates in the "shortest path" (improving CPU use slightly over both the STR#3 logic and our 'repeatable' logic) but did not keep to the 720 minute step size for re-computing terms, leading to discrepancies.



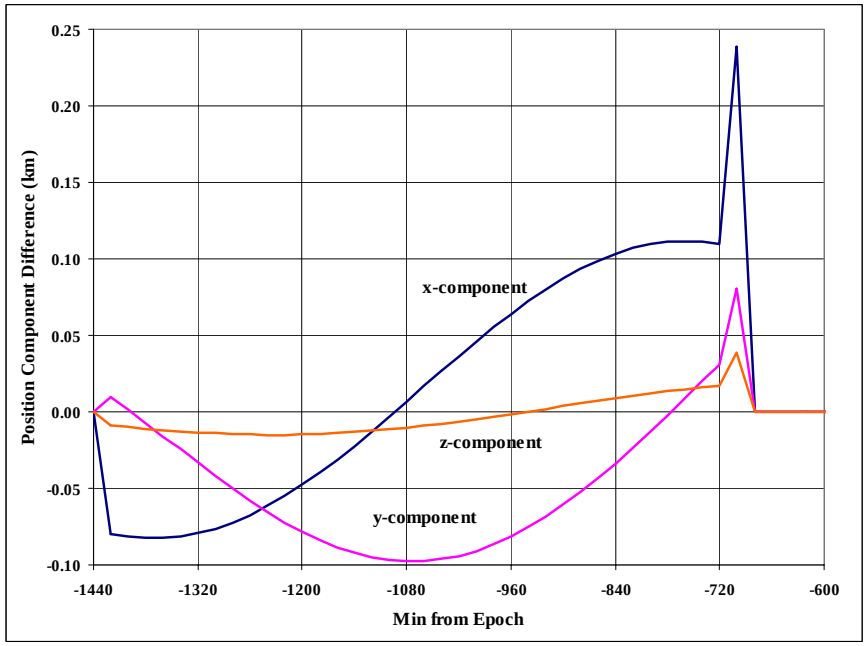


Figure 6. Propagation Problems for Satellite 09998. The integrator problem is shown by looking at the semimajor axis (top) and the positional component differences (bottom). The scale is small, but the semimajor axis clearly shows the jump caused by incorrect integrator performance near 720 minutes prior to the epoch. The problem appears in all older versions.

VII. COMPARISON ANALYSES

Many versions of SGP4 are available in code today, although most are initially from STR#3. Virtually none have been re-worked to restructure the code or to provide multiple computer programming languages and test results. Our aim is to correct that situation. Note that the basic structure of the computer code given in this paper has been available for several years in FORTRAN, Pascal, Ada, and C++ on the following web site (http://CelesTrak.com/software/vallado-sw.asp), although there has been extensive analysis to update the code for this paper. There are only three known "official" versions with which we could make comparisons. These include:

STR#3 (FORTRAN)
Both the original single/double mix, and a double-precision version (just by adding the IMPLICIT DOUBLE statement) of STR#3 code were used. The electronic code was released to all users who asked for it. T. S. Kelso released an electronic package of the 1980 report in December 1988.

GSFC (FORTRAN)
http://seawifs.gsfc.nasa.gov/SEAWIFS/SOFTWARE/src/bobdays/sgp4sub.f (original)
Note this version is no longer available at this website although numerous downloads are known by organizations and countries. In addition, the code is still easily found on archive pages throughout the internet. A current site is similar, but potentially confusing as the subroutine name is the same, but the module is clearly labeled as a Brouwer–Lyddane model.
http://www.icess.ucsb.edu/seawifs/seadas/src/utils/bobdays/sgp4sub.f (new, but different file)

JPL (FORTRAN)
ftp://naif.jpl.nasa.gov/pub/naif/toolkit/FORTRAN/PC_Linux/packages/toolkit.tar.Z
An additional source of SGP4 implementations is the JPL NAIF 'spicelib' toolkit, with source files ev2lin.f (basically SGP4.FOR equivalent), dpspce.f (basically SDP4.FOR) and zznrddp.f (basically the DEEP.FOR).

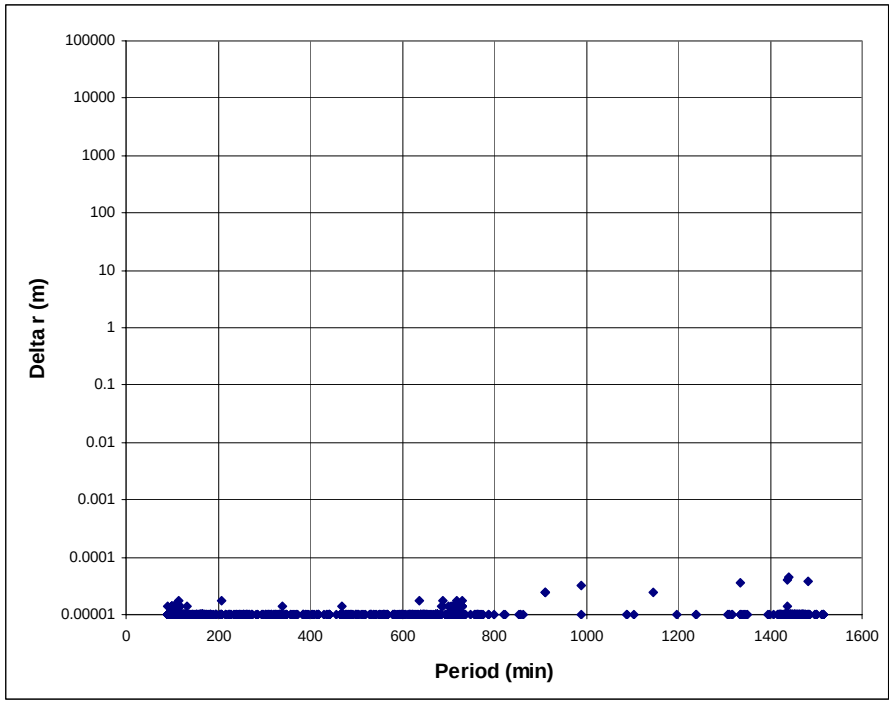
A few other codes were examined to determine what other researchers had done with the code. Examples that were tested but not included in the results presented here were:

ProjectPluto (C++)
http://www.projectpluto.com/sat_code.htm
Not an official version, but it is one of the more interesting and intelligent conversions to C++ available.

TrakStar (Pascal)
http://CelesTrak.com/software/tskelso-sw.asp
This is a very well known example, but is essentially a direct conversion of STR#3 and so it can be expected to behave in a manner similar to the STR#3 double-precision case.

Dundee (C)
http://www.sat.dundee.ac.uk/~psc/sgp4.html
The original translation into C by Paul Crawford and Andrew Brooks was virtually identical in behavior to STR#3, but with much better code structuring. Then many of the other fixes included such as the Kepler's equation solution and secular integrator were added. The update over the last year for this paper had all of the corrections discussed and agreed with the authors, resulting in virtually identical results to the paper's versions.

Because our version of SGP4 does not claim to be the official version, it was important to compare the results over a wide range of test conditions, and to compare with the released official versions. Specifically, the verification and stressing test cases provided a technical look at the performance, but these comparison tests were intended to show the robustness of the calculations under full-catalog simulations. Tests were run on several complete catalogs for varying dates. Each satellite was propagated from –1440 minutes to 1440 minutes at 20-minute time steps. The results were then compared between programs. The C++, FORTRAN, MATLAB, and Pascal versions gave virtually the same results, as shown in Fig. 7.



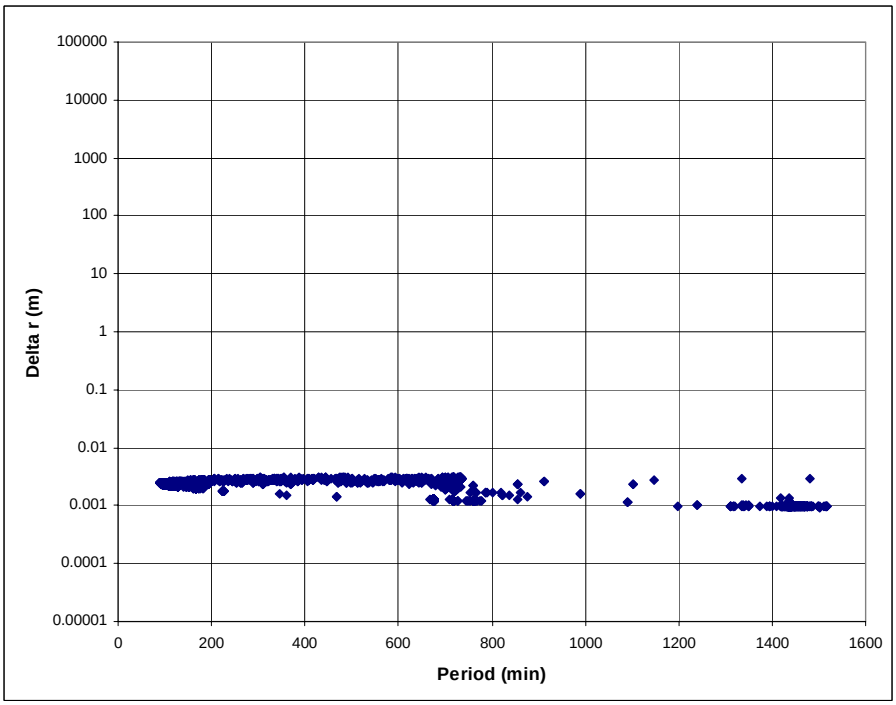


Figure 7. SGP4 Full-Catalog Comparisons. Maximum differences between ephemerides generated for two days are shown. Note the small scale for the C++ and FORTRAN comparison (top). The Pascal comparison (bottom) shows very small additional variations and these are from the 8-byte versus 10-byte precision in the language.

Comparisons were then run between the versions. Each figure shows the largest difference between the simulations, and each satellite is plotted against the orbital period. The scales are kept constant within each figure to permit rapid assessment of the differences. Figure 8 shows the results compared to the GSFC version. This was important to illustrate the similarity with the last known release.

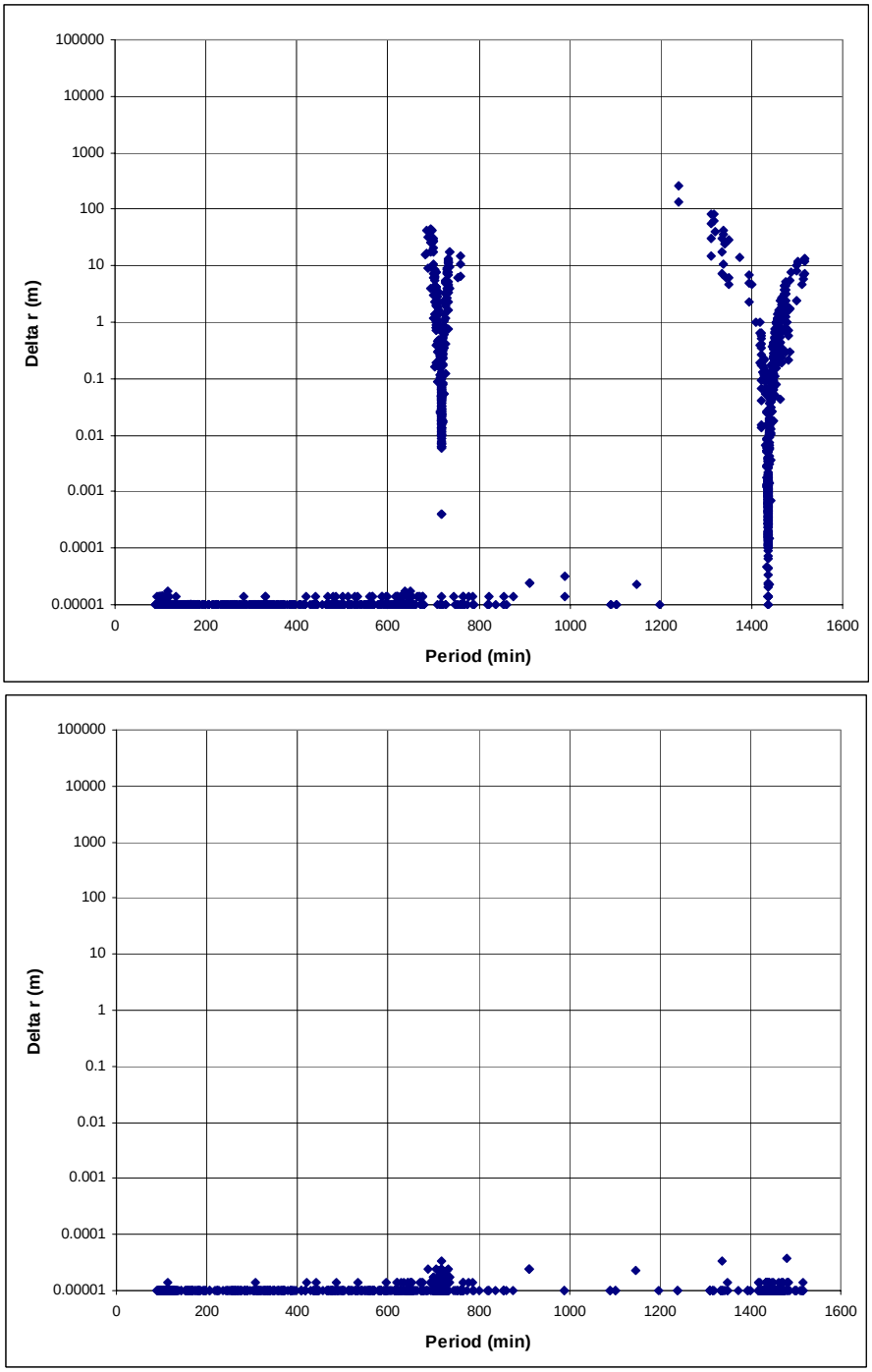


Figure 8. SGP4 Full-Catalog Comparisons. Maximum differences between ephemerides generated for 2 days are shown. The top plot shows the paper C++ version against the GSFC code, while the bottom plot shows the comparison to the GSFC code, but only for propagations positive from the epoch. The differences are all related to the integrator problems before 720 minutes prior to epoch with geosynchronous and semi-synchronous orbits.

As Fig. 8 shows, the GSFC version is very close to our revised version, and nearly identical to the performance between languages for the revised versions. The minor differences (usually a few meters) in the resonant cases (718-minute Molnyia and 1436-minute geostationary orbits) only show up with time steps that 'go backwards' in time (a problem in the secular integrator). In these tests, we begin at –1440 minutes and then step towards zero, before going 'forwards' towards +1440 minutes. In our revised version, the direction of propagation is not important. The

GSFC version also has larger errors with the direct/Lyddane choice, and the inclination going negative during propagation, but these are not shown in Fig. 8 (it requires rare or 'difficult' TLEs to show up). Those differences were discussed earlier.

The comparisons with results from STR#3 show significantly larger differences for almost all satellites. Note that both (mixed) single and double-precision results are given in Fig. 9.

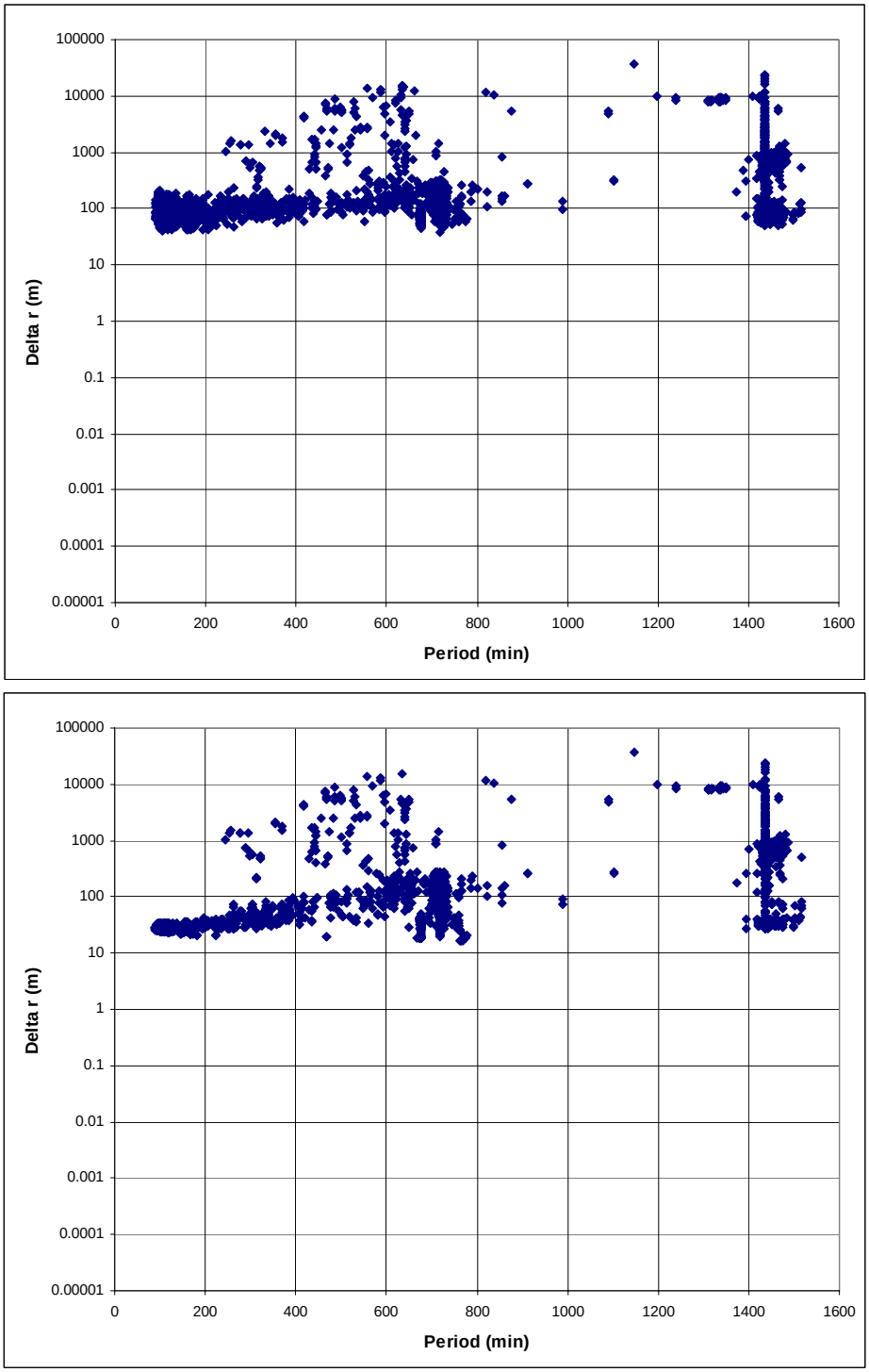


Figure 9. SGP4 Full-Catalog Comparisons. Maximum differences between ephemerides generated for 2 days are shown. The top plot shows the single-precision STR#3 version against the paper C++ version, while the bottom plot shows the STR#3 double-precision version comparison.

Both versions of STR#3 (single/double and double only) show similar results, with agreement to reasonable accuracy (sub-km) for near-Earth orbits (limited by the precision of specifying the astrodynamic constants). This is less in deep space, where some of the limited precision (e.g., Kepler's equation tolerance), the re-computing of perturbed terms, and the possibly the 'Lyddane bug' behavior show up more strongly. The differences between the GSFC version and the STR#3 versions are nearly identical to those in Fig. 9 for this catalog snapshot. This is important because many "correct" implementations of SGP4 in use are based on the STR#3 version (e.g., TrakStar), but this comparison shows the typical additional errors that users can expect as compared to the standalone AFSPC code.

Despite a rigorous attempt to review the fundamental constructs of the code, the JPL case is not particularly good in its original form. For near-Earth satellites, there is clearly some problem with the implementation (drag equations perhaps?) as it is much worse than STR#3 code in mixed precision. The deep-space cases have other problems, one of which is the choice to zero the LS offsets at epoch[*] (which does not appear to be correctly implemented in any case). It also shares the negative inclination problem. These are unfortunate, as it is an interesting attempt to order and modernize the FORTRAN code, showing some insight into improving things, but missing others (such as the commonality of the SGP4/SDP4 codes) completely. The Project Pluto code (not shown) compared favorably to the revised code version but its lack of 'official heritage' made it difficult to accept changes based solely on its presence in this version. Results for the JPL code are shown in Fig. 10.

[*] It appears that JPL made the same change as several other authors who assumed that the zeroing of the Lunar-Solar perturbations at epoch mentioned in STR#6 and the GSFC code also applied to the code when used in the deep space part of the merged SGP4 model. This is not the case, and the reader is reminded that what matters most for accuracy, if the theory is not completely documented, is to use the same code for propagating the elements as was used in generating them.

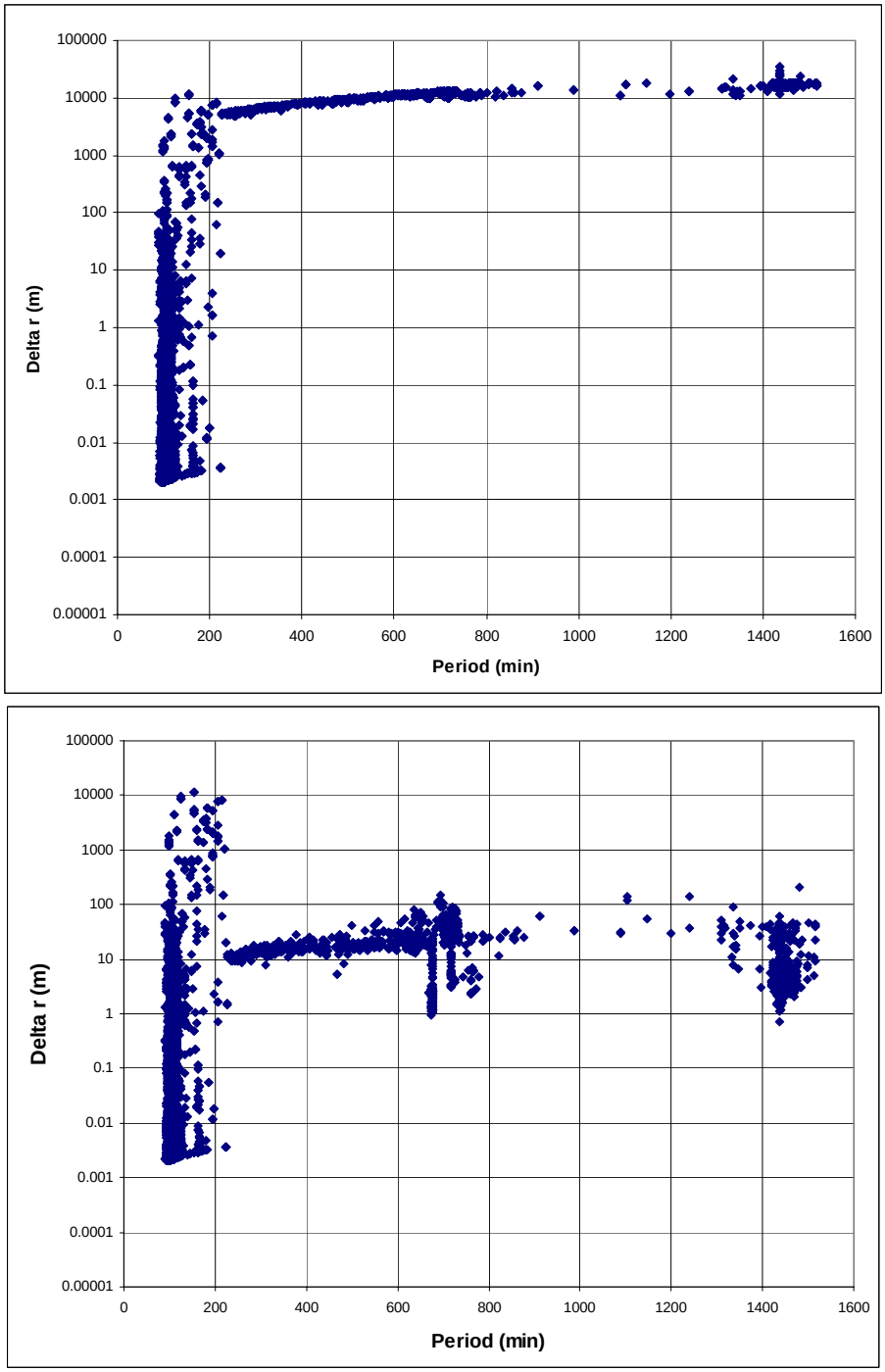


Figure 10. SGP4 Full-Catalog Comparisons. Maximum differences between ephemerides generated for 2 days are shown. The plot shows the paper C++ version against the original JPL code (plot on the top). Note that by changing the DOPERT variable in the JPL code, the results can be improved by about two orders of magnitude (plot on the bottom).

VIII. AVAILABILITY

The primary computer source code discussed in this paper has been available on the Internet for nearly six years, but was re-worked for this paper with inputs from many people and organizations. The current code is available in C++, FORTRAN, MATLAB, and Pascal as these appear to be the most common languages for operations today.

The appendices contain definitions and examples of TLE data and TEME conversions, the C++ code, along with the input TLE data, and results. The code for debugging the software is not included as it was not pertinent to the discussion. The files have 'include' statements which are commented out where these lines of code would be inserted. All the necessary files are located on the Internet for convenience. They are available from the Center for Space website:*

http://www.centerforspace.com/downloads/

IX. CONCLUSIONS

This paper has re-examined the *Spacetrack Report Number 3* formulation of analytical propagation. By incorporating changes posted over the last quarter century, a unified and improved version is presented for general use. Structural changes to the code have been completed permitting the ability to process multiple satellites at one time. We chose to omit the Lyddane choice change for certain inclinations to maintain as close a performance to what we believe AFSPC is doing today. However, we also included comments in the source code to facilitate location of any updates now, or at a future time. Test cases are included to demonstrate verification of operation with the branches in the code, for difficult orbits, as well as cases encountered throughout the years. The results show that continued use of the STR#3 version, and to a lesser extent some of the more recent versions, can result in potentially large errors when producing ephemerides. We also noted the difficulty with aligning a particular version of SGP4 with a particular TLE as the data formats and processing have changed throughout the years. Finally, we hope this form of documentation will motivate similar efforts for additional analytical theories in a similar fashion, along with satellite data to use with each theory. Any questions, comments, additions, etc. may be addressed to David Vallado at dvallado@centerforspace.com.

Acknowledgements

Many people were involved with this project in addition to the co-authors listed. I am very grateful for all the help, and support I received during this long project. Felix Hoots provided significant insight and details of the original development and John Seago provided suggestions for a more concise description of the time and coordinate systems. A special thank you is due to Jeff Beck who provided the MATLAB version based on the C++ code.

References

Note that some of these references may be difficult to find. AFSPC should be able to provide all the necessary information.

Air Force Space Command Instruction (AFSPCI) 33-105. 2004. "Distribution of AFSPC Software to Outside Organizations." Colorado Springs, CO. (See http://www.e-publishing.af.mil/pubs/majcom.asp?org=AFSPC)

Air Force Space Command Instruction (AFSPCI) 60-102. 1996. "Space Astrodynamic Standards Software." Colorado Springs, CO.

Aoki, S. et al. 1982. The New Definition of Universal Time. *Astronomy and Astrophysics*. Vol. 105: 359–361.

Arsenault, J. L., L. Chaffee, and J. R. Kuhlman. 1964. "General Ephemeris Routine Formulation Document." Report ESD-TDR-64-522, Aeronutronic Publ. U-2731.

Atkinson R d'E, and D. H. Sadler. 1951. On the use of Mean Sidereal Time. *Monthly Newsletters of the Royal Astronomical Society*. 111:619–623.

Brouwer, D. 1959. Solution of the Problem of Artificial Satellite Theory without Drag. *Astronomical Journal*, Vol. 64, No. 1274, pp. 378–397.

Brouwer, D., and G. Hori. 1961a. Theoretical Evaluation of Atmospheric Drag Effects in the Motion of an Artificial Satellite. *Astronomical Journal*, Vol. 66, No. 5, pp. 193–225.

_____ 1961b. Appendix to Theoretical Evaluation of Atmospheric Drag Effects in the Motion of an Artificial Satellite. *Astronomical Journal.* Vol. 66, No. 6, pp. 264–265.

David A. Cappellucci. 2005. "Special Perturbations to General Perturbations Extrapolation Differential Correction in Satellite Catalog Maintenance." Paper AAS 05-402 presented at the AIAA/AAS Astrodynamics Specialist Conference. Lake Tahoe, California.

Cefola, Paul J., and Wayne McClain. 1987. Accuracy of the NORAD DP4 Satellite Theory for Synchronous Equatorial Orbits. Interoffice Memorandum NOR/PL-002-15Z-PJC. Draper Laboratory, MA.

* Users of Analytical Graphics Inc. Satellite Toolkit (STK) will find the source code integrated within the latest release of the program.

Cefola, Paul J., and D. J. Fonte. 1996. Extension of the Naval Space Command Satellite Theory to include a General Tesseral m-daily Model. Paper AIAA-96-3606 presented at the AIAA/ AAS Astrodynamics Conference. San Diego, CA.

Coffey, S. L., and H. L. Neal. 1998. "An Operational Special-Perturbations-Based Catalog." Paper AAS 98-113 presented at the AAS/AIAA Space Flight Mechanics Conference. Monterey, CA.

Crawford P. S. 1995. Kepler's Equations in C. *International Journal of Remote Sensing.* Vol. 16, No. 3 pp 549–557.

Glover, R. A. 1996. "The Naval Space Command (NAVSPACECOM) PPT3 Orbit Model." NAVSPACECOM Technical Report.

Hartman, Paul G. 1993. "Long-term SGP4 Performance." Space Control Operations Technical Note J3SOM-TN-93-01. US Space Command, USSPACECOM/J3SO. Colorado Springs, CO.

Hilton, C. G. 1963. "The SPADATS Mathematical Model." Report ESD-TDR- 63-427, Aeronutronic Publ. U-2202.

Hilton, C. G., and J. R. Kuhlman. 1966. "Mathematical Models for the Space Defense Center." Philco-Ford Publication No. U-3871, 17–28.

Hoots, Felix R. 1980. "A Short, Efficient Analytical Satellite Theory." AIAA Paper No. 80-1659.

_____. 1981. Theory of the Motion of an Artificial Earth Satellite. *Celestial Mechanics.* Vol. 23, pp. 307–363.

_____. 1986. "Spacetrack Report #6: Models for Propagation of Space Command Element Sets." Space Command, United States Air Force, CO.

_____. 1998. "A History of Analytical Orbit Modeling in the United States Space Surveillance System." Third US-Russian Space Surveillance Workshop. Washington, D.C.

Hoots, Felix R. et al. 1986. "Improved General Perturbations Prediction Capability." Air Force Space Command, Astrodynamics Analysis Memorandum 86-3.

Hoots, Felix R., and R. G. France. 1983. "Performance of an Analytic Satellite Theory in a Real World Environment." AAS/AIAA Paper No. 83-395.

_____. 1987. An Analytical Satellite Theory using Gravity and a Dynamic Atmosphere. *Celestial Mechanics.* Vol. 40, pp. 1–18.

Hoots, Felix R., and R. L. Roehrich. 1980. "Spacetrack Report #3: Models for Propagation of the NORAD Element Sets." U.S. Air Force Aerospace Defense Command, Colorado Springs, CO.

Hoots, Felix R., P. W. Schumacher, and R. A. Glover. 2004. History of Analytical Orbit Modeling in the U. S. Space Surveillance System. *Journal of Guidance, Control, and Dynamics*. 27(2):174–185.

Hujsak, R. S. 1979. "Spacetrack Report #1: A Restricted Four Body Solution for Resonating Satellites Without Drag." U.S. Air Force Aerospace Defense Command, Colorado Springs, CO.

_____. 1979. "A Restricted Four Body Solution for Resonating Satellites with an Oblate Earth." AIAA Paper No. 79-136.

Hujsak, R. S., and F. R. Hoots. 1977. "Deep Space Perturbations Ephemeris Generation. Aerospace Defense Command Space Computational Center Program Documentation, DCD 8, Section 3, 82:104."

_____. 1982. "Deep Space Perturbations Ephemeris Generation." NORAD Technical Publication TP-SCC 008. 129–145.

Jacchia, L. G. 1970. "New Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles." SAO Report 313. Cambridge, MA: Smithsonian Institution Astrophysical Observatory.

Kaya, Denise, et al. 2004. "AFSPC Astrodynamic Standard Software." Paper AAS 04-124 presented at the AAS/AIAA Space Flight Mechanics Conference. Maui, HI.

Kaya, Denise, et al. 2001. "AFSPC Astrodynamic Standards – The Way of The Future." Paper presented at the MIT/LL Conference. Lexington, MA.

Kelso, T.S. 2004. "Frequently Asked Questions: Two-Line Element Set Format." (See http://CelesTrak.com/columns/v04n03/)

Kelso, T. S., and S. Alfano. 2005. "Satellite Orbital Conjunction Reports Assessing Threatening Encounters in Space (SOCRATES)." Paper AAS 05-124 presented at the AAS/AIAA Space Flight Mechanics Conference. Copper Mountain, CO.

Kozai, Y. 1959. The Motion of a Close Earth Satellite. *Astronomical Journal.* Vol. 64, No. 1274, pp. 367–377.

Lane, M. H. 1965. "The Development of an Artificial Satellite Theory Using Power-Law Atmospheric Density Representation." AIAA Paper 65-35.

Lane, M. H., and K. H. Cranford. 1969. "An Improved Analytical Drag Theory for the Artificial Satellite Problem." *AIAA* Paper No. 69-925.

Lane, M. H., P. M. Fitzpatrick, and J. J. Murphy. 1962. "Spacetrack Report #APGC-TDR-62-15: On the Representation of Air Density in Satellite Deceleration Equations by Power Functions with Integral Exponents." Air Force Systems Command, Eglin AFB, FL.

Lane, M. H., and F. R. Hoots. 1979. "Spacetrack Report #2: General Perturbations Theories Derived from the 1965 Lane Drag Theory." Aerospace Defense Command, Peterson AFB, CO.

Lyddane, R. H. 1963. Small Eccentricities or Inclinations in the Brouwer Theory of the Artificial Satellite. *Astronomical Journal*. Vol. 68, No. 8, 1963, pp. 555–558.

Morris, Robert F., and Timothy P. Payne. 1993. "SGP4 Version 3.01 Validation Test Cases." Publishing data unknown. Referenced in AFSPC I 60-102.

Nijenhuis, Albert. 1991. Solving Kepler's equation with high efficiency and accuracy. *Celestial Mechanics and Dynamical Astronomy*. Vol. 51, No. 4, pp 319–330.

Patt, Frederick S., Hoisington, Charles M., Gregg, Watson W., and Coronado, Patrick L. 1993. NASA Technical Memorandum 104566, Vol. 11 "Volume 11, Analysis of Selected Orbit Propagation Models for the SeaWiFS Mission" available at http://library.gsfc.nasa.gov/Databases/Gtrs/Data/TM-1993-104566v11.pdf.

Schumacher, P. W., and R. A. Glover. 1995. "Analytical Orbit Model for U.S. Naval Space Surveillance: An Overview." Paper AAS 95-427 presented at the AIAA/AAS Astrodynamics Specialist Conference. Halifax, Canada.

Seago, John, and David Vallado. 2000. "Coordinate Frames of the U.S. Space Object Catalogs." Paper AIAA 2000-4025 presented at the AIAA/AAS Astrodynamics Specialist Conference. Denver, CO.

Tanygin, Sergei, and James R. Wright. 2004. "Removal of Arbitrary Discontinuities in Atmospheric Density Modeling." Paper AAS 04-176 presented at the AAS/AIAA Space Flight Mechanics Conference. Maui, HI.

Vallado, David A. 1999. "Joint Astrodynamic Working Group Meeting Minutes." September 20, 1999. USSPACECOM/AN. Colorado Springs, CO.

_____. 2001. "A Summary of the AIAA Astrodynamic Standards Effort." Paper AAS 01-429 presented at the AIAA/AAS Astrodynamics Specialist Conference. Quebec City, Canada.

_____. 2004. *Fundamentals of Astrodynamics and Applications*. Second Edition, second printing. Microcosm, El Segundo, CA.

_____. 2005. "An Analysis of State Vector Propagation using Differing Flight Dynamics Programs." Paper AAS 05-199 presented at the AAS/AIAA Space Flight Mechanics Conference. Copper Mountain, CO.

Vallado, David, and Salvatore Alfano. 1999. "A Future Look at Space Surveillance and Operations." Paper AAS 99-113 presented at the AAS/AIAA Space Flight Mechanics Conference. Breckenridge, CO.

Appendices

Appendix A – Organizational Nomenclature

Tracing the reports, documents, and files back to the original release may present some confusion to users that are not familiar with the various organizational structures that have been in place over time. We have tried to use the appropriate organizational names when referencing information, but the following information may help associating those references. We use DoD, AFSPC, NORAD, CMOC, etc. The following quotes were assembled from the Cheyenne Mountain website: https://www.cheyennemountain.af.mil.

> "The original North American Air Defense Command (NORAD) Combat Operations Center … has evolved into the Cheyenne Mountain Operations Center (CMOC). The original requirement for an operations center in Cheyenne Mountain was to provide command and control in support of the air defense mission against the Soviet manned bomber threat ... In the early 1960s, the advent of an Intercontinental Ballistic Missile (ICBM) attack against North America became a top priority. Missile warning and air sovereignty were the primary missions in the Mountain throughout the 1960s and 70s. During a brief period in the mid 1970s, the Ballistic Missile Defense Center was installed within the Mountain. … In 1979, the Air Force established a Space Defense Operations Center [SPADOC] to counter the emerging Soviet's anti-satellite threat … The evolution continued into the 1980s when Air Force Space Command [AFSPC] was created and tasked with the Air Force Space mission … In April 1981, Space Defense Operations Center crews and their worldwide sensors, under the direction of Air Defense Command [ADC], supported the first flight of the space shuttle … Oct. 1, 2002 marked the welcoming of two new commands, U.S. Northern Command and U.S. Strategic Command, to Cheyenne Mountain. CMOC is responsible for providing support to USNORTHCOM's mission of homeland defense and USSTRATCOM's mission of space and missile warning, formerly associated with U.S. Space Command.
>
> Today, the Cheyenne Mountain Complex is known as Cheyenne Mountain Air Force Station (CMAFS). CMAFS is host to four commands: North American Aerospace Defense Command (NORAD), United States Northern Command (USNORTHCOM), United States Strategic Command (USSTRATCOM), and Air Force Space Command (AFSPC). CMOC serves as the command center for both NORAD and USNORTHCOM. It is the central collection and coordination center for a worldwide system of satellites, radars, and sensors that provide early warning of any missile, air, or space threat to North America. Supporting the NORAD mission, CMOC provides warning of ballistic missile or air attacks against North America, assists the air sovereignty mission for the U.S. and Canada, and if necessary, serves as the focal point for air defense operations to counter enemy bombers or cruise missiles. In addition, CMOC also provides theater ballistic missile warning for U.S. and allied forces. In support of USSTRATCOM, CMOC provides a day-to-day picture of precisely what is in space and where it is located. Space control operations include protection, prevention, and negation functions supported by the surveillance of space. … Operations are conducted in seven centers manned 24 hours a day, 365 days a year. The centers are the Air Warning Center, Missile Correlation Center, Space Control Center [JSPOC], Operational Intelligence Watch, Systems Center, Weather Center, and the Command Center ... The Joint Space Operations Center (JSPOC) supports United States Strategic Command (USSTRATCOM) missions of surveillance and protection of U.S. assets in space. The JSPOC's primary objective in performing the surveillance mission is to detect, track, identify, and catalog all man-made objects orbiting earth. … The JSPOC maintains a current computerized catalog of all orbiting man-made objects, charts preset positions, plots future orbital paths, and forecasts times and general location for significant man-made objects reentering the Earth's atmosphere."

The organizational names identify whether they are joint (international), or not. For generic applications, we use DoD because this is the broadest identification for all the organizations. We generally use NORAD when referring to the TLE data because their formation was begun under NORAD. Today, the JPSOC [a.k.a. Space Control Center] within the CMOC produces the TLE data, but we retain the historical name due to its familiarity in the community. Regulations and documentation have generally come from AFSPC and are identified as such.

Appendix B – Two Line Element Set Format

The format for the TLE is shown in Fig. 11 with sample data.

Card #	Satellite Number	Class	International Designator			Yr	Epoch Day of Year (plus fraction)	Mean motion derivative (rev/day /2)	Mean motion second derivative (rev/day2 /6)	Bstar (/ER)	Eph	Elem num	Chk Sum
			Year	Lch#	Piece			S	S .	S . S E			
1	1 6 6 0 9	U	8 6 0 1 7	A		9 3	3 5 2 . 5 3 5 0 2 9 3 4	. 0 0 0 0 0 7 8 8 9	0 0 0 0 0 - 0	1 0 5 2 9 - 3	0	3 4	2

		Inclination (deg)	Right Ascension of the Node (deg)	Eccentricity	Arg of Perigee (deg)	Mean Anomaly (deg)	Mean Motion (rev/day)	Epoch Rev	Chk
2	1 6 6 0 9	5 1 . 6 1 9 0	1 3 . 3 3 4 0	0 0 0 5 7 7 0	1 0 2 . 5 6 8 0	2 5 7 . 5 9 5 0	1 5 . 5 9 1 1 4 0 7 0	4 4 7	8 6 9

Figure 11. Two-line Element Set Format. An example TLE is shown, with descriptions and units of each field. Note that the eccentricity, mean motion second derivative, and Bstar have implied decimal points before the first numerical value. The mean motion derivative is already divided by 2, and the second derivative is already divided by 6. Shaded cells do not contain data. The signs may be blank, "+" or "–". A classification field is sometimes included after the satellite number.

There are several notes.

1. The maximum accuracy for a TLE is limited by the number of decimal places in each field (Vallado, 2004:116). In general, TLE data is accurate to about a kilometer or so at epoch and it quickly degrades (Hartman, 1993). We note that the SGP4 theory is capable of much better accuracy through additional modeling and sufficient observational data. Cefola and McClain (1987) noted that certain low-inclination geosynchronous orbits exhibited large discrepancies from numerical simulations due to oversimplifications in the node rate calculations. Cefola and Fonte (1996) showed that the addition of additional terms to the theory could improve the overall accuracy by almost an order of magnitude. Cappellucci (2005) showed that using numerically generated state vectors and performing an SGP4 orbit determination on the resulting ephemerides produced errors representative of a numerical technique. This is not unexpected as additional (continuous) observations provide the needed observability over a simple "3-obs per pass" approach (Vallado and Alfano, 1999). However, the results diverge very rapidly once outside the fit span of the orbit determination. We do not address orbit determination here.

It is also worth noting that there are numerous other analytical orbital theories that have been developed for a wide range of applications, but unfortunately, no comprehensive source of data for those theories exists. SGP4 is interesting because it tries to satisfy many orbital regimes. The Russians developed a series of analytical propagators, each tuned for a specific satellite regime. The narrower focus permits additional attention to detail, and higher resulting accuracy. It is hoped that some of these analytical routines can eventually be documented for general use in the same manner as this paper.

2. The satellite number consists of any numeric value 0 – 99999. Discussions have hinted at a lengthening of the field size to 7 or 9 characters to accommodate future satellites.

3. Sometimes additional assignments are made – signs = 0; minus signs = 1.

4. The International designator is broken up into the last two digits of the lunch year, the launch number for that year (3 digits), and the piece of the launch (3 digits). Kelso (2004) also indicates:

> [The] International Designator of the object [ ] is an additional unique designation assigned by the World Data Center-A for Rockets and Satellites (WDC-A-R&S) in accordance with international treaty (1975 Convention on Registration of Objects Launched into Outer Space). The WDC-A-R&S works together with NORAD and NASA's National Space Science Data Center (NSSDC) in maintaining this registry. Although there have been some changes in format since it was first used back in the late 1950s (see "Space Surveillance" in Satellite Times Volume 4 Number 1), the International Designator indicates the year of the launch (field 1.4 only gives the last two digits), the launch of that year (field 1.5), and the piece of that launch (field 1.6) for each object. These three fields can be left blank, but all must be present if any is. Finally, field 1.6 can be either right or left justified—the latter is preferred.
>
> As an aside, there are some significant differences between NORAD's Catalog Number and the International Designator. For example, NORAD assigns a catalog number based upon when the object was first observed, whereas the

International Designator is always tied to the original launch. For example, the 81st launch of 1968 carried four payloads into orbit: OV2-5, ERS 21 and 28, and LES 6. Together with the Titan 3C transtage rocket body, these objects were assigned International Designators 1968-081A through E and Catalog Numbers 03428 through 03431. Just this past October, however, NORAD cataloged two additional pieces associated with this launch as Catalog Numbers 25000 and 25001—they have the International Designators 1968-081F and G.

5. The mean motion rates are not used by SGP4 and are only valid for the older SGP model.

6. Bstar is an SGP4 drag-like coefficient. Usually, ballistic coefficients (*BC*) are used in aerodynamic theory. The *BC* is $m/c_D A$, or the reciprocal (*A* is cross-sectional area, c_D is the coefficient of drag, and *m* is mass). Bstar is an adjusted value of *BC* using the reference value of atmospheric density, $\rho_o = 2.461 \times 10^{-5}$ kg/m^2, at one Earth radius.

$$BC = R_e \rho_o / (2Bstar) \tag{B-1}$$

7. The Ephemeris type is not used external to CMOC. All TLE data is generated by SGP4.

8. From Kelso (2004):

The element set number. Normally, this number is incremented each time a new element set is generated. In practice, however, this doesn't always happen. When operations switch between the primary and backup Space Control Centers, sometimes the element set numbers get out of sync, with some numbers being reused and others skipped. Unfortunately, this makes it difficult to tell if you have all the element sets for a particular object."

The last column on each line represents a modulo-10 checksum of the data on that line. To calculate the checksum, simply add the values of all the numbers on each line—ignoring all letters, spaces, periods, and plus signs—and assigning a value of 1 to all minus signs. The checksum is the last digit of that sum. Although this is a very simple error-checking procedure, it should catch 90 percent of all errors. However, many errors can still sneak through. To eliminate these, all data posted on the CelesTrak WWW site not only pass the checksum test, but must also pass both format and range-checking tests (as described in this article).

The final field on line 2, prior to the checksum, is the rev number. Since there are several conventions for determining rev numbers, this field also bears some clarification. In NORAD's convention, a revolution begins when the satellite is at the ascending node of its orbit and a revolution is the period between successive ascending nodes. The period from launch to the first ascending node is considered to be Rev 0 and Rev 1 begins when the first ascending node is reached. Since many element sets are generated with epochs that place the satellite near its ascending node, it is important to note whether the satellite has reached the ascending node when calculating subsequent rev numbers.

Appendix C – TEME Coordinate System

This section describes the equations necessary to implement the nutation equations for the TEME approach. There are two approaches – using the GMST, and using the equation of the equinoxes.

For sidereal time, GMST is needed. GMST is found using UT1. From McCarthy (1992:30)

$$\theta_{GMST1982} = 67{,}310.548\ 41^s + (876{,}600^h + 8{,}640{,}184.812\ 866^s)T_{UT1} + 0.093\ 104T_{UT1}^2 - 6.2x10^{-6}T_{UT1}^3 \quad \text{(C-1)}$$

The transformation to ITRF is found using the polar motion (x_p, y_p) values and the GMST. Note that "PEF" implies the *pseudo-Earth-fixed* frame, where polar motion has not yet been applied (Vallado, 2004: 217).

$$\begin{aligned} [\mathbf{W}]_{ITRF-PEF} &= ROT1(y_p)ROT2(x_p) \\ \vec{r}_{ITRF} &= [\mathbf{W}]^T [ROT3(\theta_{GMST1982})]^T \vec{r}_{TEME} \\ \vec{v}_{ITRF} &= [\mathbf{W}]^T \left\{ [ROT3(\theta_{GMST1982})]^T \vec{v}_{TEME} + \vec{\omega}_{\oplus} \times \vec{r}_{PEF} \right\} \end{aligned} \quad \text{(C-2)}$$

If the equation of the equinox approach is taken, you must find the nutation parameters. The IAU-80 nutation uses so-called Delaunay variables and coefficients to calculate nutation in longitude ($\Delta\psi_{1980}$) and nutation in the obliquity of the ecliptic ($\Delta\varepsilon_{1980}$). (McCarthy, 1992:32)

$$\begin{aligned} M_{(} &= 134.962\ 981\ 39^\circ + 1{,}717{,}915{,}922.6330'' T_{TT} + 31.31T_{TT}^2 + 0.064T_{TT}^3 \\ M_{O} &= 357.527\ 723\ 33^\circ + 129{,}596{,}581.2240'' T_{TT} - 0.577T_{TT}^2 + 0.012T_{TT}^3 \\ \mu_{(} &= 93.271\ 910\ 28^\circ + 1{,}739{,}527{,}263.1370'' T_{TT} - 13.257T_{TT}^2 - 0.011T_{TT}^3 \\ D_{O} &= 297.850\ 363\ 06^\circ + 1{,}602{,}961{,}601.3280'' T_{TT} - 6.891T_{TT}^2 + 0.019T_{TT}^3 \\ \Omega_{(} &= 125.044\ 522\ 22^\circ - 6{,}962{,}890.5390'' T_{TT} + 7.455T_{TT}^2 + 0.008T_{TT}^3 \end{aligned} \quad \text{(C-3)}$$

The nutation parameters are then found using (McCarthy, 1992:33)

$$\begin{aligned} a_{p_i} &= a_{n1_i} M_{(} + a_{n2_i} M_{O} + a_{n3_i} \mu_{(} + a_{n4_i} D_{O} + a_{n5_i} \Omega_{(} \\ \Delta\psi &= \sum_{i=1}^{106} (A_{p_i} + A_{pl_i} T_{TDB}) \sin(a_{p_i}) \quad \Delta\varepsilon = \sum_{i=1}^{106} (A_{e_i} + A_{el_i} T_{TDB}) \cos(a_{p_i}) \end{aligned} \quad \text{(C-4)}$$

Corrections to the nutation parameters ($\delta\Delta\psi_{1980}$ and $\delta\Delta\varepsilon_{1980}$) supplied as Earth Orientation Parameters (EOP) from the IERS are simply added to the resulting values in Eq. 4 to provide compatibility with the newer IAU 2000 Resolutions (Kaplan, 2005). These corrections also include effects from Free Core Nutation (FCN) that correct errors in the IAU-76 precession and IAU-80 nutation. However for TEME, these corrections do not appear to be used. The nutation parameters let us find the true obliquity of the ecliptic, ε. (McCarthy, 1992:29–31)

$$\begin{aligned} \Delta\psi_{1980} &= \Delta\psi + \delta\Delta\psi_{1980} \qquad \Delta\varepsilon_{1980} = \Delta\varepsilon + \delta\Delta\varepsilon_{1980} \\ \bar{\varepsilon} &= 84{,}381.448'' - 46.8150T_{TT} - 0.000\ 59T_{TT}^2 + 0.001\ 813T_{TT}^3 \\ \varepsilon &= \bar{\varepsilon} + \Delta\varepsilon_{1980} \end{aligned} \quad \text{(C-5)}$$

The equation of the equinoxes ($EQ_{eqe1980}$) can then be found. The last two terms in the $EQ_{eqe1980}$ are probably not included in AFSPC formulations. From McCarthy (1992:30)

$$\begin{aligned} EQ_{eqe1980} &= \Delta\psi_{1980} \cos(\bar{\varepsilon}) + 0.002\ 64'' \sin(\Omega_{(}) + 0.000\ 063 \sin(2\Omega_{(}) \\ \theta_{GMST1982} &= 67{,}310.54841^s + (876{,}600^h + 8{,}640{,}184.812\ 866^s)T_{UT1} + 0.093\ 104T_{UT1}^2 - 6.2x10^{-6}T_{UT1}^3 \\ \theta_{GAST1982} &= \theta_{GMST1982} + EQ_{eqe1980} \end{aligned} \quad \text{(C-6)}$$

These relations let us form the transformation equations.

$$
\begin{aligned}
[\mathbf{P}]_{MOD-J2000} &= ROT3(\zeta)ROT2(-\Theta)ROT3(z) \\
[\mathbf{N}]_{TOD-MOD} &= ROT1(-\bar{\varepsilon})ROT3(\Delta\Psi)ROT1(\varepsilon) \\
\vec{r}_{J2000} &= [\mathbf{P}][\mathbf{N}]\left[ROT3(-EQ_{eqe1980})\right]\vec{r}_{\text{TEME}} \\
\vec{v}_{J2000} &= [\mathbf{P}][\mathbf{N}]\ [ROT3(-EQ_{eqe1980})]\ \vec{v}_{\text{TEME}}
\end{aligned}
\tag{C-7}
$$

An example is useful to show the various options and their effect on the resulting vectors. Consider an initial ECI (J2000.0, IAU76/FK5) state vector.

June 28, 2000, 15: 8:51.655 000 UTC

$\Delta UT1 = 0.162\ 360^{s}$, $\Delta AT\ 21^{s}$, $x_p = 0.098\ 700"$, $y_p = 0.286\ 000"$

$\boldsymbol{r}_{J2000} =$ 3961.744 260 3 6010.215 610 9 4619.362 575 8 km

$\boldsymbol{v}_{J2000} =$ –5.314 643 386 3.964 357 585 1.752 939 153 km/s

Converting to standard TOD, PEF via IAU 76/FK5, without the nutation corrections ($\delta\Delta\psi_{1980}$ and $\delta\Delta\varepsilon_{1980}$), but using the two additional terms with $EQ_{eqe1980}$,

$JD_{UT1} = 2{,}451{,}724.131\ 155\ 293\ 40$, $T_{TT} =$ 0.004 904 360 547

$\boldsymbol{r}_{TOD} =$ 3961.421 498 5 6010.475 268 8 4619.301 531 0 km

$\boldsymbol{v}_{TOD} =$ –5.314 833 569 3.964 181 915 1.752 759 802 km/s

$\boldsymbol{r}_{PEF} =$ 298.803 632 8 –7192.314 622 9 4619.301 531 0 km

$\boldsymbol{v}_{PEF} =$ 6.105 014 271 –0.131 824 177 1.752 759 802 km/s

For the TEME transformation, use equations (3) to (6) to find the approximate parameters (with 4 nutation terms in Eq (4), no additional two terms in Eq (6), and no small angle approximations). Then, transform TOD or PEF to TEME using Eq (7) or Eq (2) respectively.

$\boldsymbol{r}_{TEME} =$ 3961.003 549 8 6010.751 174 0 4619.300 930 1 km

$\boldsymbol{v}_{TEME} =$ –5.315 109 069 3.963 813 071 1.752 758 562 km/s

Notice this vector is "in between TOD and PEF, but much closer to the TOD value – a reason it is sometimes [imprecisely] considered "inertial". We consider these numbers are within a few mm of CMOC results.

The related issue for TEME 'of date' and TEME 'of epoch' can also be demonstrated with the following TLE data at epoch and at 3 days into the future.

```
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
```

Some users have assumed comments in CMOC-produced computer outputs suggested TEME of epoch (precession is used), but most users appear to assume "of date." These headers often state: *"Ephemeris generated by SGP4 using the WGS-72 earth model. Coordinate frame = true equator and mean equinox of epoch using the FK5 mean of J2000 time and reference frame."* We believe this statement still exists today. There is no mention of TEME in the FK5 theory and applicable documents. Analytical Graphic Inc.'s STK provides options for each with the 'of date' option as the default, and we concur with the 'of date' position. Although the change between the two over a week or so is small, it is something measurable if agreement to a centimeter or less is desired. Note that this ignores the general accuracy of the TLE data being no better than a few kilometers. Using the TLE example data, we find the TEME vector at a time 3 days in the future (day = 182.784 950 62) from the TLE epoch.

$\boldsymbol{r}_{TEME} =$ –9060.473 735 69 4658.709 525 02 813.686 731 53 km

$\boldsymbol{v}_{TEME} =$ –2.232 832 783 –4.110 453 490 –3.157 345 433 km/s

Next, find the nutation transformation matrix at this time.

$$
[\mathbf{R}]_{TEME} = \begin{bmatrix}
0.999\ 999\ 999\ 56 & 0.000\ 000\ 000\ 00 & 0.000\ 029\ 505\ 95 \\
-0.000\ 000\ 000\ 65 & 0.999\ 999\ 999\ 76 & 0.000\ 022\ 007\ 05 \\
-0.000\ 029\ 505\ 95 & 0.000\ 022\ 007\ 05 & 0.999\ 999\ 999\ 32
\end{bmatrix}
\tag{C-8}
$$

Using Eq (7), we find the inertial ("J2000") vector at the future time.

$\boldsymbol{r}_{J2000} =$ –9059.941 378 6 4659.697 200 0 813.958 887 5 km

$\boldsymbol{v}_{J2000} =$ –2.233 348 094 –4.110 136 162 –3.157 394 074 km/s

For the future time using the 'of epoch' option, the nutation matrix is found at the epoch time as

$$[\mathbf{R}]_{TEME} = \begin{bmatrix} 0.999\ 999\ 999\ 55 & 0.000\ 000\ 000\ 00 & 0.000\ 030\ 111\ 90 \\ -0.000\ 000\ 000\ 66 & 0.999\ 999\ 999\ 76 & 0.000\ 021\ 766\ 37 \\ -0.000\ 030\ 111\ 90 & 0.000\ 021\ 766\ 37 & 0.999\ 999\ 999\ 31 \end{bmatrix} \tag{C-9}$$

and the resulting vector is

$$\boldsymbol{r}_{J2000} = -9059.951\ 079\ 9 \quad 4659.680\ 755\ 6 \quad 813.945\ 045\ 1 \text{ km}$$
$$\boldsymbol{v}_{J2000} = -2.233\ 336\ 111 \quad -4.110\ 141\ 024 \quad -3.157\ 396\ 220 \text{ km/s}$$

This is a difference of 23.6 m in just 3 days.

Appendix D – Test Case Listing

SGP4-VER.TLE –

Test cases include those used for the figures in the paper (with a keyword "## fig"), and those used for verification to exercise various aspects of the code. The additional values on the second line were added to simplify automatic processing of each test – the ephemeris starting minutes from epoch (MFE) to the ending MFE, and the delta time step in minutes.

```
#    ------------------ 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 E – Test Case Results Listing

TCPPVER.OUT –

The results are given below for the verification TLE data in Appendix D. Note that this version includes the results of the Lyddane choice using the perturbed inclination as indicated in the GSFC code, and also uses the WGS-72 constants. There is also a "true" time reference at the end of each line to facilitate the conversion with minutes from epoch and UTC.

Min from epoch	position x km	position y km	position z km	vel km/s	vel km/s	vel km/s	year	mon	day	hr	min	sec
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	2000	6	28	0:	50:	19.733571
720.00000000	-7134.59340119	6531.68641334	3260.27186483	-4.113793027	-2.911922039	-2.557327851	2000	6	28	6:	50:	19.733571
1080.00000000	5568.53901181	4492.06992591	3863.87641983	-4.209106476	5.159719888	2.744852980	2000	6	28	12:	50:	19.733571
1440.00000000	-938.55923943	-6268.18748831	-4294.02924751	7.536105209	-0.427127707	0.989878080	2000	6	28	18:	50:	19.733571
1800.00000000	-9680.56121728	2802.47771354	124.10688038	-0.905874102	-4.659467970	-3.227347517	2000	6	29	0:	50:	19.733571
2160.00000000	190.19796988	7746.96653614	5110.00675412	-6.112325142	1.527008184	-0.139152358	2000	6	29	6:	50:	19.733571
2520.00000000	5579.55640116	-3995.61396789	-1518.82108966	4.767927483	5.123185301	4.276837355	2000	6	29	12:	50:	19.733571
2880.00000000	-8650.73082219	-1914.93811525	-3007.03603443	3.067165127	-4.828384068	-2.515322836	2000	6	29	18:	50:	19.733571
3240.00000000	-5429.79204164	7574.36493792	3747.39305236	-4.999442110	-1.800561422	-2.229392830	2000	6	30	0:	50:	19.733571
3600.00000000	6759.04583722	2001.58198220	2783.55192533	-2.180993947	6.402085603	3.644723952	2000	6	30	6:	50:	19.733571
3960.00000000	-3791.44531559	-5712.95617894	-4533.48630714	6.668817493	-2.516382327	-0.082384354	2000	6	30	12:	50:	19.733571
4320.00000000	-9060.47373569	4658.70952502	813.68673153	-2.232832783	-4.110453490	-3.157345433	2000	6	30	18:	50:	19.733571
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	2004	1	28	7:	27:	25.308584
-5064.00000000	-32982.56870101	-11125.54996609	-6803.28472771	0.617446996	-3.379240041	0.085954707	2004	1	28	9:	27:	25.308597
-4944.00000000	-22097.68730513	-31583.13829284	-4836.34329328	2.230597499	-2.166594667	0.426443070	2004	1	28	11:	27:	25.308611
-4896.00000000	-15129.94694545	-36907.74526221	-3487.56256701	2.581167187	-1.524204737	0.504805763	2004	1	28	12:	15:	25.308600
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	2006	6	25	21:	46:	43.980124
240.00000000	-1675.12766915	-5683.30432352	-3286.21510937	5.282496925	1.508674259	-5.354872978	2006	6	25	23:	46:	43.980097
360.00000000	4993.62642836	2890.54969900	-3600.40145627	0.347333429	5.707031557	5.070699638	2006	6	26	1:	46:	43.980111
480.00000000	-1115.07959514	4015.11691491	5326.99727718	-5.524279443	-4.765738774	2.402255961	2006	6	26	3:	46:	43.980124
600.00000000	-4329.10008198	-5176.70287935	409.65313857	2.858408303	-2.933091792	-6.509690397	2006	6	26	5:	46:	43.980097
720.00000000	3692.60030028	-976.24265255	-5623.36447493	3.897257243	6.415554948	1.429112190	2006	6	26	7:	46:	43.980111
840.00000000	2301.83510037	5723.92394553	2814.61514580	-5.110924966	-0.764510559	5.662120145	2006	6	26	9:	46:	43.980124
960.00000000	-4990.91637950	-2303.42547880	3920.86335598	-0.993439372	-5.967458360	-4.759110856	2006	6	26	11:	46:	43.980097
1080.00000000	642.27769977	-4332.89821901	-5183.31523910	5.720542579	4.216573838	-2.846576139	2006	6	26	13:	46:	43.980111
1200.00000000	4719.78335752	4798.06938996	-943.58851062	-2.294860662	3.492499389	6.408334723	2006	6	26	15:	46:	43.980124
1320.00000000	-3299.16993602	1576.83168320	5678.67840638	-4.460347074	-6.202025196	-0.885874586	2006	6	26	17:	46:	43.980097
1440.00000000	-2777.14682335	-5663.16031708	-2462.54889123	4.915493146	0.123328992	-5.896495091	2006	6	26	19:	46:	43.980111
1560.00000000	4992.31573893	1716.62356770	-4287.86065581	1.640717189	6.071570434	4.338797931	2006	6	26	21:	46:	43.980124
1680.00000000	-8.22384755	4662.21521668	4905.66411857	-5.891011274	-3.593173872	3.365100460	2006	6	26	23:	46:	43.980097
1800.00000000	-4966.20137963	-4379.59155037	1349.33347502	1.763172581	-3.981456387	-6.343279443	2006	6	27	1:	46:	43.980111
1920.00000000	2954.49390331	-2080.65984650	-5754.75038057	4.895893306	5.858184322	0.375474825	2006	6	27	3:	46:	43.980124
2040.00000000	3363.28794321	5559.55841180	1956.05542266	-4.587378863	0.591943403	6.107838605	2006	6	27	5:	46:	43.980097
2160.00000000	-4856.66780070	-1107.03450192	4557.21258241	-2.304158557	-6.186437070	-3.956549542	2006	6	27	7:	46:	43.980111
2280.00000000	-497.84480071	-4863.46005312	-4700.81211217	5.960065407	2.996683369	-3.767123329	2006	6	27	9:	46:	43.980124
2400.00000000	5241.61936096	3910.75960683	-1857.93473952	-1.124834806	4.406213160	6.148161299	2006	6	27	11:	46:	43.980097
2520.00000000	-2451.38045953	2610.60463261	5729.79022069	-5.366560525	-5.500855666	0.187958716	2006	6	27	13:	46:	43.980111
2640.00000000	-3791.87520638	-5378.82851382	-1575.82737930	4.266273592	-1.199162551	-6.276154080	2006	6	27	15:	46:	43.980124
2760.00000000	4730.53958356	524.05006433	-4857.29369725	2.918056288	6.135412849	3.495115636	2006	6	27	17:	46:	43.980097
2880.00000000	1159.27802897	5056.60175495	4353.49418579	-5.968060341	-2.314790406	4.230722669	2006	6	27	19:	46:	43.980111

```
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    2006  6 25  9:58:18.143649
     240.00000000   19752.78050009   -8600.07130962   37522.72921090  0.238105279  1.546110924  0.986410447    2006  6 25 11:58:18.143622
     360.00000000   19089.29762968    3107.89495018   39958.14661370 -0.410308034  1.640332277 -0.306873818    2006  6 25 13:58:18.143636
     480.00000000   13829.66070574   13977.39999817   32736.32082508 -1.065096849  1.279983299 -1.760166075    2006  6 25 15:58:18.143649
     600.00000000    3333.05838525   18395.31728674   12738.25031238 -1.882432221 -0.611623333 -4.039586549    2006  6 25 17:58:18.143622
     720.00000000    2622.13222207  -15125.15464924     474.51048398  2.688287199 -3.078426664  4.494979530    2006  6 25 19:58:18.143636
     840.00000000   15320.56770017  -17777.32564586   25539.53198382  1.064346229  0.892184771  2.459822414    2006  6 25 21:58:18.143649
     960.00000000   19769.70267785   -8458.65104454   37624.20130236  0.229304396  1.550363884  0.966993056    2006  6 25 23:58:18.143622
    1080.00000000   19048.56201523    3260.43223119   39923.39143967 -0.418015536  1.639346953 -0.326094840    2006  6 26  1:58:18.143636
    1200.00000000   13729.19205837   14097.70014810   32547.52799890 -1.074511043  1.270505211 -1.785099927    2006  6 26  3:58:18.143649
    1320.00000000    3148.86165643   18323.19841703   12305.75195578 -1.895271701 -0.678343847 -4.086577951    2006  6 26  5:58:18.143622
    1440.00000000    2890.80638268  -15446.43952300     948.77010176  2.654407490 -2.909344895  4.486437362    2006  6 26  7:58:18.143636
    1560.00000000   15415.98410712  -17699.90714437   25796.19644689  1.049818334  0.908822332  2.434107329    2006  6 26  9:58:18.143649
    1680.00000000   19786.00618538   -8316.74570581   37723.74539119  0.220539813  1.554518900  0.947601047    2006  6 26 11:58:18.143622
    1800.00000000   19007.28688729    3412.85948715   39886.66579255 -0.425733568  1.638276809 -0.345353807    2006  6 26 13:58:18.143636
    1920.00000000   13627.93015254   14216.95401307   32356.13706868 -1.083991976  1.260802347 -1.810193903    2006  6 26 15:58:18.143649
    2040.00000000    2963.26486560   18243.85063641   11868.25797486 -1.908015447 -0.747870342 -4.134004492    2006  6 26 17:58:18.143622
    2160.00000000    3155.85126036  -15750.70393364    1422.32496953  2.620085624 -2.748990396  4.473527039    2006  6 26 19:58:18.143636
    2280.00000000   15510.15191770  -17620.71002219   26050.43525345  1.035454678  0.925111006  2.408534465    2006  6 26 21:58:18.143649
    2400.00000000   19801.67198812   -8174.33337167   37821.38577439  0.211812700  1.558576937  0.928231880    2006  6 26 23:58:18.143622
    2520.00000000   18965.46529379    3565.19666242   39847.97510998 -0.433459945  1.637120585 -0.364653213    2006  6 27  1:58:18.143636
    2640.00000000   13525.88227400   14335.15978787   32162.13236536 -1.093537945  1.250868256 -1.835451681    2006  6 27  3:58:18.143649
    2760.00000000    2776.30574260   18156.98538451   11425.73046481 -1.920632199 -0.820370733 -4.181839232    2006  6 27  5:58:18.143622
    2880.00000000    3417.20931586  -16038.79510665    1894.74934058  2.585515864 -2.596818146  4.456882556    2006  6 27  7:58:18.143636
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    2006  6 25 15:28:40.058423
     240.00000000   11332.67806218   16517.99124008   38569.78482991 -1.400974747  0.710947006  0.923935636    2006  6 25 17:28:40.058396
     360.00000000     328.74217398   19554.92047380   40558.26246145 -1.593281066  0.126772913 -0.359627307    2006  6 25 19:28:40.058410
     480.00000000  -10684.90590680   18057.15728839   33158.75253886 -1.383205997 -0.582328999 -1.744412556    2006  6 25 21:28:40.058423
     600.00000000  -17069.78000550    9944.86797897   13885.91649059  0.044133354 -1.853448464 -3.815303117    2006  6 25 23:28:40.058396
     720.00000000   13725.09398980   -2180.70877090     863.29684523  3.878478111  1.656846496  4.944867241    2006  6 26  1:28:40.058410
     840.00000000   19089.63879226    9456.29670247   27026.79562883 -0.656614299  1.309112636  2.449371941    2006  6 26  3:28:40.058423
     960.00000000   11106.41248373   16627.60874079   38727.35140296 -1.409722680  0.698582526  0.891383535    2006  6 26  5:28:40.058396
    1080.00000000      72.40958621   19575.08054144   40492.12544001 -1.593394604  0.113655142 -0.390556063    2006  6 26  7:28:40.058410
    1200.00000000  -10905.89252576   17965.41205111   32850.07298244 -1.371396120 -0.601706604 -1.782817058    2006  6 26  9:28:40.058423
    1320.00000000  -17044.61207568    9635.48491849   13212.59462953  0.129244030 -1.903551430 -3.884569098    2006  6 26 11:28:40.058396
    1440.00000000   14369.90303735   -1903.85601062    1722.15319852  3.543393116  1.701687176  4.913881358    2006  6 26 13:28:40.058410
    1560.00000000   18983.96210441    9661.12233804   27448.99557732 -0.687189304  1.293808870  2.403630759    2006  6 26 15:28:40.058423
    1680.00000000   10878.79336704   16735.31433954   38879.23434264 -1.418239666  0.686235750  0.858951848    2006  6 26 17:28:40.058396
    1800.00000000    -184.03743100   19593.09371709   40420.40606889 -1.593348925  0.100448697 -0.421571993    2006  6 26 19:28:40.058410
    1920.00000000  -11125.12138631   17870.19488928   32534.21521208 -1.359116236 -0.621413776 -1.821629856    2006  6 26 21:28:40.058423
    2040.00000000  -17004.43272827    9316.53926351   12526.11883812  0.220330736 -1.955594322 -3.955058575    2006  6 26 23:28:40.058396
    2160.00000000   14960.06492693   -1620.68430805    2574.96359381  3.238634028  1.734723385  4.868880331    2006  6 27  1:28:40.058410
    2280.00000000   18873.46347257    9863.57004586   27863.46574735 -0.716736981  1.278632817  2.358448535    2006  6 27  3:28:40.058423
    2400.00000000   10649.86857581   16841.14172669   39025.48035006 -1.426527152  0.673901057  0.826632332    2006  6 27  5:28:40.058396
    2520.00000000    -440.53459323   19608.95524423   40343.10675451 -1.593138597  0.087147884 -0.452680559    2006  6 27  7:28:40.058410
    2640.00000000  -11342.45028909   17771.44223942   32211.12535721 -1.346344015 -0.641464291 -1.860864234    2006  6 27  9:28:40.058423
    2760.00000000  -16948.06005711    8987.64254880   11826.28284367  0.318007297 -2.009693492 -4.026726648    2006  6 27 11:28:40.058396
    2880.00000000   15500.53445068   -1332.90981042    3419.72315308  2.960917974  1.758331634  4.813698638    2006  6 27 13:28:40.058410
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    2005  5 27 19: 3:37.089777
   -1380.00000000     309.25349929  -36960.43090143   -4198.48007670  3.292429375 -0.002166046  0.402111628    2005  5 27 20: 3:37.089763
   -1320.00000000   11949.04009077  -35127.37816804   -2565.89806468  3.119942784  1.012096444  0.497284100    2005  5 27 21: 3:37.089790
   -1260.00000000   22400.45329336  -29798.63236321    -677.91515122  2.638533344  1.922477736  0.542792913    2005  5 27 22: 3:37.089777
   -1200.00000000   30640.84752458  -21525.02340201    1277.34808722  1.903464941  2.634294312  0.534540934    2005  5 27 23: 3:37.089763
```

```
  -1140.00000000   35899.56788035  -11152.71158138    3108.72535238  0.997393045  3.079858548  0.474873291    2005  5 28  0: 3:37.089790
  -1080.00000000   37732.45438600     288.18821054    4643.87587495  0.016652226  3.225184410  0.371669746    2005  5 28  1: 3:37.089777
  -1020.00000000   36045.92961699   11706.61816230    5746.32646574 -0.942409065  3.069888941  0.236662980    2005  5 28  2: 3:37.089763
   -960.00000000   31076.77273609   22063.44379776    6325.93403705 -1.794027976  2.642072476  0.083556127    2005  5 28  3: 3:37.089790
   -900.00000000   23341.26015320   30460.88002531    6342.91707895 -2.469409743  1.990861658 -0.073612096    2005  5 28  4: 3:37.089777
   -840.00000000   13568.39733054   36204.45930900    5806.79548733 -2.919354203  1.178920217 -0.221646814    2005  5 28  5: 3:37.089763
   -780.00000000    2628.58762420   38840.10855897    4771.91979854 -3.114400514  0.276239109 -0.348926401    2005  5 28  6: 3:37.089790
   -720.00000000   -8535.81598158   38171.79073851    3331.00311285 -3.043839958 -0.644462527 -0.445808894    2005  5 28  7: 3:37.089777
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    1980  8 17 13: 6:40.136822
     720.00000000   14271.29083858   24110.44309009   -4725.76320143 -0.320504528  2.679841539 -2.084054355    1980  8 17 19: 6:40.136822
    1080.00000000   -9990.05800009   22717.34212448  -23616.88515553 -1.016674392 -2.290267981  0.728923337    1980  8 18  1: 6:40.136822
    1440.00000000    9787.87836256   33753.32249667  -15030.79874625 -1.094251553  0.923589906 -1.522311008    1980  8 18  7: 6:40.136822
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    2006  6 25  2:40:57.987566
     240.00000000   -3023.38840703   41783.13186459    7273.03412906 -3.035574793 -0.271656544  0.309645251    2006  6 25  4:40:57.987539
     360.00000000  -23516.34391907   34424.42065671    8448.49867693 -2.529120477 -1.726186020  0.009582303    2006  6 25  6:40:57.987553
     480.00000000  -37837.46699511   18028.39727170    7406.25540271 -1.360069525 -2.725794686 -0.292555349    2006  6 25  8:40:57.987566
     600.00000000  -42243.58460661   -3093.72887774    4422.91711801  0.163110919 -3.009980598 -0.517584362    2006  6 25 10:40:57.987539
     720.00000000  -35597.57919549  -23407.91145393     282.09554383  1.641405246 -2.506773678 -0.606963478    2006  6 25 12:40:57.987553
     840.00000000  -19649.19834455  -37606.11623860   -3932.71525948  2.689647056 -1.349150016 -0.537710698    2006  6 25 14:40:57.987566
     960.00000000    1431.30912160  -41982.04949668   -7120.45467057  3.035263353  0.160882945 -0.327993994    2006  6 25 16:40:57.987539
    1080.00000000   22136.97605384  -35388.19823762   -8447.62393401  2.587624889  1.630097136 -0.032349004    2006  6 25 18:40:57.987553
    1200.00000000   37050.15790219  -19537.23321425   -7564.83463543  1.461844494  2.674654256  0.272202191    2006  6 25 20:40:57.987566
    1320.00000000   42253.81760945    1431.81867593   -4699.87621174 -0.049247334  3.019518960  0.505890058    2006  6 25 22:40:57.987539
    1440.00000000   36366.59147396   22023.54245720    -601.47121821 -1.549681546  2.571788981  0.607057418    2006  6 26  0:40:57.987553
    1560.00000000   20922.12287985   36826.33975981    3654.91125886 -2.644070068  1.447521216  0.548722983    2006  6 26  2:40:57.987566
    1680.00000000     -23.77224182   41945.51688402    6950.29891751 -3.043358385 -0.057417440  0.346112094    2006  6 26  4:40:57.987539
    1800.00000000  -20964.17821076   36039.06206172    8418.91984963 -2.642795221 -1.546099886  0.052725852    2006  6 26  6:40:57.987553
    1920.00000000  -36401.63863057   20669.75286162    7677.19769359 -1.549488154 -2.627052310 -0.254079652    2006  6 26  8:40:57.987566
    2040.00000000  -42298.30327543    -119.03351118    4922.96388841 -0.052232768 -3.018152669 -0.493827331    2006  6 26 10:40:57.987539
    2160.00000000  -37125.62383511  -20879.63058368     879.86971348  1.456499841 -2.619358421 -0.604081694    2006  6 26 12:40:57.987553
    2280.00000000  -22250.12320553  -36182.74736487   -3393.15365183  2.583161226 -1.536647628 -0.556404555    2006  6 26 14:40:57.987566
    2400.00000000   -1563.06258654  -42035.43179159   -6780.02161760  3.034917506 -0.052702046 -0.363395654    2006  6 26 16:40:57.987539
    2520.00000000   19531.64069587  -36905.65470956   -8395.46892032  2.693682199  1.446079999 -0.075256054    2006  6 26 18:40:57.987553
    2640.00000000   35516.53506142  -22123.71916638   -7815.04516935  1.646882125  2.568416058  0.232985912    2006  6 26 20:40:57.987566
    2760.00000000   42196.03535976   -1547.32646751   -5187.39401981  0.166491841  3.019211549  0.480665780    2006  6 26 22:40:57.987539
    2880.00000000   37802.25393045   19433.57330019   -1198.66634226 -1.359930580  2.677830903  0.602507466    2006  6 27  0:40:57.987553
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    2006  5 31 18:10:47.226141
     240.00000000   -3385.00215658    7538.13955729     200.59008616 -2.023512865 -4.261808344 -6.856385787    2006  5 31 20:10:47.226115
     360.00000000   12805.22442200  -10258.94667177   13780.16486738  0.619279224  1.821510542  2.507365975    2006  5 31 22:10:47.226128
     480.00000000    5682.46556318    7199.30270473   15437.67134070 -2.474365406  2.087897336 -2.583767460    2006  6  1  0:10:47.226141
     600.00000000    7628.94243982  -12852.72097492    2902.87208981  2.748131081 -0.740084579  4.125307943    2006  6  1  2:10:47.226115
     720.00000000   11531.64866625    -858.27542736   19086.85993771 -1.170071901  2.660311986  0.096005705    2006  6  1  4:10:47.226128
     840.00000000   -3866.98069515    2603.73442786   -4577.36484577  1.157257298 -8.453281164 -4.683959407    2006  6  1  6:10:47.226141
     960.00000000   13054.77732721   -8707.92757730   15537.63259903  0.229846748  2.119467054  2.063396852    2006  6  1  8:10:47.226115
    1080.00000000    3496.91064652    8712.83919778   12845.81838327 -2.782184997  1.552950644 -3.554436131    2006  6  1 10:10:47.226128
    1200.00000000    9593.07424729  -13023.75963608    6250.46484931  2.072666376  0.278735334  3.778111073    2006  6  1 12:10:47.226141
    1320.00000000   10284.79205084    1487.89914169   18824.37381327 -1.530335053  2.663107730 -0.542205966    2006  6  1 14:10:47.226115
    1440.00000000    -984.62035146   -5187.03480813   -5745.59594144  4.340271916 -7.266811354  1.777668888    2006  6  1 16:10:47.226128
20413 xx
       0.00000000   25123.29290741  -13225.49966286    3249.40351869  0.488683419  4.797897593 -0.961119693
    1440.00000000 -151669.05280515   -5645.20454550   -2198.51592118 -0.869182889 -0.870759872  0.156508219    2005 12 30 19: 0: 0.000268
    1560.00000000 -157497.71657495  -11884.99595074   -1061.44439402 -0.749657961 -0.864016715  0.157766101    2005 12 30 21: 0: 0.000282
    1680.00000000 -162498.32255577  -18062.99733167      81.00915253 -0.638980378 -0.853687105  0.158098992    2005 12 30 23: 0: 0.000255
```

```
    1800.00000000 -166728.76010920  -24155.99648299    1222.84128677 -0.535600687 -0.840455444  0.157680857    2005 12 31  1: 0: 0.000268
    1920.00000000 -169935.81924592  -31767.29787964    2749.01540345 -0.430050431 -0.828904183  0.157812340    2005 12 31  3: 0: 0.000282
    2040.00000000 -172703.07831815  -37662.95639336    3883.60052579 -0.338004891 -0.810277487  0.156020035    2005 12 31  5: 0: 0.000255
    2160.00000000 -174823.19337404  -43417.55605219    5003.26312809 -0.250258622 -0.789828672  0.153764903    2005 12 31  7: 0: 0.000268
    2280.00000000 -176324.63925775  -49018.51958648    6104.85025002 -0.166136613 -0.767706262  0.151092242    2005 12 31  9: 0: 0.000282
    2400.00000000 -177231.42142458  -54454.12699497    7185.48661607 -0.085067854 -0.744001567  0.148033403    2005 12 31 11: 0: 0.000255
    2520.00000000 -177563.73583232  -59713.14859144    8242.48472591 -0.006561730 -0.718760309  0.144608676    2005 12 31 13: 0: 0.000267
    2640.00000000 -177338.48026483  -64784.54644698    9273.27220003  0.069809946 -0.691990238  0.140829236    2005 12 31 15: 0: 0.000281
    2760.00000000 -176569.65151461  -69657.21976255   10275.33063459  0.144426878 -0.663665876  0.136698419    2005 12 31 17: 0: 0.000254
    2880.00000000 -175268.65299073  -74319.77625463   11246.14177160  0.217631370 -0.633731091  0.132212491    2005 12 31 19: 0: 0.000267
    3000.00000000 -173444.53039609  -78760.31560396   12183.13775212  0.289737325 -0.602099929  0.127361017    2005 12 31 21: 0: 0.000281
    3120.00000000 -171104.14813653  -82966.21323591   13083.65278381  0.361037779 -0.568655903  0.122126889    2005 12 31 23: 0: 0.000254
    3240.00000000 -168252.31543803  -86923.89363433   13944.87382716  0.431811396 -0.533249797  0.116486022    2006  1  1  1: 0: 0.000268
    3360.00000000 -164891.86832887  -90618.58225954   14763.78794247  0.502328269 -0.495695896  0.110406725    2006  1  1  3: 0: 0.000282
    3480.00000000 -161023.71139825  -94034.02398835   15537.12375729  0.572855321 -0.455766412  0.103848688    2006  1  1  5: 0: 0.000255
    3600.00000000 -156646.82136726  -97152.15370791   16261.28409305  0.643661538 -0.413183688  0.096761524    2006  1  1  7: 0: 0.000268
    3720.00000000 -151758.21285737  -99952.70098346   16932.26607548  0.715023254 -0.367609561  0.089082727    2006  1  1  9: 0: 0.000282
    3840.00000000 -146352.86521283 -102412.70506284   17545.56394158  0.787229695 -0.318630913  0.080734873    2006  1  1 11: 0: 0.000255
    3960.00000000 -140423.60777444 -104505.90799734   18096.04807097  0.860588979 -0.265739987  0.071621768    2006  1  1 13: 0: 0.000268
    4080.00000000 -133960.95961851 -106201.98091318   18577.81121953  0.935434758 -0.208307307  0.061623110    2006  1  1 15: 0: 0.000282
    4200.00000000 -126952.91860010 -107465.51906186   18983.96903112  1.012133628 -0.145543878  0.050587007    2006  1  1 17: 0: 0.000255
    4320.00000000 -119384.69396454 -108254.71115372   19306.39581892  1.091093313 -0.076447479  0.038319282    2006  1  1 19: 0: 0.000268
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    2006  6 25  2:33:42.834827
     240.00000000  -12686.06129708  -23853.75335645   35529.81733588  1.231633829 -0.221718202  1.118440291    2006  6 25  4:33:42.834800
     360.00000000   -2775.46649359  -22839.64574119   39494.64689967  1.468963405  0.489481769 -0.023972788    2006  6 25  6:33:42.834814
     480.00000000    7679.87883570  -16780.50760106   34686.21815555  1.364171080  1.211183897 -1.385151371    2006  6 25  8:33:42.834827
     600.00000000   14552.40023028   -4819.50121461   17154.70672449  0.109201591  2.176124494 -3.854856805    2006  6 25 10:33:42.834800
     720.00000000  -15302.38845375   -5556.43440300    1095.95088753 -2.838224312 -3.134231137  3.992596326    2006  6 25 12:33:42.834814
     840.00000000  -19289.20066748  -19427.04851118   23759.45685636  0.552495087 -1.112499437  2.325112654    2006  6 25 14:33:42.834827
     960.00000000  -12376.21976437  -23893.38020018   35831.33691892  1.246701529 -0.194294048  1.074867282    2006  6 25 16:33:42.834800
    1080.00000000   -2400.55677665  -22698.62264640   39482.75964390  1.472582922  0.513555654 -0.069306561    2006  6 25 18:33:42.834814
    1200.00000000    8031.66819252  -16455.77592085   34298.94391742  1.351357426  1.239633234 -1.448195324    2006  6 25 20:33:42.834827
    1320.00000000   14559.48780372   -4238.43773813   16079.23154704 -0.026409655  2.218938770 -4.012628896    2006  6 25 22:33:42.834800
    1440.00000000  -16036.04980660   -6372.51406468    2183.44834232 -2.485113443 -2.994994355  3.955891272    2006  6 26  0:33:42.834814
    1560.00000000  -19156.71583814  -19698.89059957   24389.29473934  0.594278133 -1.069418599  2.271152044    2006  6 26  2:33:42.834827
    1680.00000000  -12062.72925552  -23925.82362911   36120.66680667  1.261238798 -0.167201856  1.031478939    2006  6 26  4:33:42.834800
    1800.00000000   -2024.96136966  -22551.56626703   39458.50085787  1.475816889  0.537615764 -0.114887472    2006  6 26  6:33:42.834814
    1920.00000000    8379.80916204  -16123.95878459   33894.75123231  1.337468254  1.268432783 -1.512473301    2006  6 26  8:33:42.834827
    2040.00000000   14527.86748873   -3646.33817120   14960.74306518 -0.180035839  2.261273515 -4.179355590    2006  6 26 10:33:42.834800
    2160.00000000  -16680.12147335   -7149.80800425    3257.64227208 -2.178897351 -2.863927095  3.904876943    2006  6 26 12:33:42.834814
    2280.00000000  -19013.58793448  -19958.93766022   25003.81778666  0.634100431 -1.027559823  2.218002685    2006  6 26 14:33:42.834827
    2400.00000000  -11745.76155818  -23951.19438627   36397.87676581  1.275261813 -0.140425132  0.988259441    2006  6 26 16:33:42.834800
    2520.00000000   -1648.81945070  -22398.50594576   39421.83273890  1.478660174  0.561671519 -0.160733093    2006  6 26 18:33:42.834814
    2640.00000000    8723.97652795  -15784.99406275   33473.35215527  1.322433593  1.297602497 -1.578055493    2006  6 26 20:33:42.834827
    2760.00000000   14452.25571587   -3043.42332645   13796.84870805 -0.355190169  2.302485443 -4.355767077    2006  6 26 22:33:42.834800
    2880.00000000  -17246.31075678   -7890.72601508    4315.39410307 -1.910968458 -2.740945672  3.844722726    2006  6 27  0:33:42.834814
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    2006  4  4 12: 0: 0.000000
      74.20286720    3282.82085464    2077.46972905   -5189.17988770  0.097342701  7.375135692  2.900196702    2006  4  4 12:20: 0.000009
      94.20286720     530.82729176    6426.20790003    1712.37076793 -3.837120395 -1.252430637  6.561602577    2006  4  4 12:40: 0.000018
     114.20286720   -3191.69170212     170.27219912    5956.29807775 -1.394956872 -7.438073471 -0.557553115    2006  4  4 13: 0: 0.000027
     134.20286720   -1818.99222465   -6322.45146616     681.95247154  3.349795173 -1.530140265 -6.831522765    2006  4  4 13:19:59.999996
     154.20286720    2515.66448634   -2158.83091224   -5552.13320544  2.571979660  7.311930509 -1.639865620    2006  4  4 13:40: 0.000004
     174.20286720    2414.52833210    5749.10150922   -1998.59693165 -2.681032960  3.527589301  6.452951429    2006  4  4 14: 0: 0.000013
     194.20286720   -1877.98944331    3862.27848302    5112.48435863 -3.261489804 -6.026859137  3.433254768    2006  4  4 14:20: 0.000022
     214.20286720   -3117.36584395   -4419.74773864    3840.85960912  1.545479182 -5.475416581 -5.207913748    2006  4  4 14:40: 0.000031
```

```
     234.20286720     815.32034678   -5231.67692249   -3760.04690354  3.870864200  4.455588552 -5.211082191    2006  4  4 15: 0: 0.000000
     254.20286720    3269.54341810    3029.00081083   -4704.67969713 -0.526711345  6.812157950  3.929825087    2006  4  4 15:20: 0.000009
     274.20286720     -10.18099756    6026.23341453    2643.50518407 -3.953623254 -2.616070012  6.145637500    2006  4  4 15:40: 0.000018
     294.20286720   -3320.58819584   -1248.42679945    5563.06017927 -0.637046974 -7.417786044 -2.076120187    2006  4  4 16: 0: 0.000027
     314.20286720   -1025.48974616   -6366.98945782    -911.23559153  3.811771909  0.438071490 -6.829260617    2006  4  4 16:19:59.999996
     334.20286720    3003.75996128    -413.85708003   -5706.15591435  1.674350083  7.694169068  0.316915204    2006  4  4 16:40: 0.000004
     354.20286720    1731.42816980    6258.27676925    -409.32527982 -3.400497806  1.447945424  6.904010052    2006  4  4 17: 0: 0.000013
     374.20286720   -2582.52111460    2024.19020680    5647.55650268 -2.530348121 -7.221719393  1.438141553    2006  4  4 17:20: 0.000022
     394.20286720   -2440.56848578   -5702.77311877    1934.81094689  2.731792947 -3.350576075 -6.527773339    2006  4  4 17:40: 0.000031
     414.20286720    1951.22934391   -3423.59443045   -5121.67808201  3.249039133  6.465974362 -3.069806659    2006  4  4 18: 0: 0.000000
     434.20286720    2886.50939356    4888.68626216   -3096.29885989 -1.973162139  4.877039020  5.832414910    2006  4  4 18:20: 0.000009
     454.20286720   -1276.55532182    4553.26898463    4406.19787375 -3.715146421 -5.320176914  4.418210777    2006  4  4 18:40: 0.000018
     474.20286720   -3181.54698042   -3831.29976506    4096.80242787  1.114159970 -6.104773578 -4.829967400    2006  4  4 19: 0: 0.000027
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    2006  6 25 15:25: 5.468479
     240.00000000   21619.59550749   16125.24978864   36396.79365831 -0.963604380  0.685454965  1.177181937    2006  6 25 17:25: 5.468452
     360.00000000   12721.50543331   19258.96193362   40898.47648359 -1.457448565  0.179955469  0.071502601    2006  6 25 19:25: 5.468465
     480.00000000    1272.80760054   18458.41971897   37044.74742696 -1.674863386 -0.436454983 -1.201040990    2006  6 25 21:25: 5.468479
     600.00000000  -10058.43188619   11906.60764454   21739.62097733 -1.245829683 -1.543789125 -3.324449221    2006  6 25 23:25: 5.468452
     720.00000000   10924.40116466   -2571.92414170   -2956.34856294  6.071727751  1.349579102  3.898430260    2006  6 26  1:25: 5.468465
     840.00000000   25332.14851525    8398.91099924   21783.90654357  0.222320754  1.272214306  2.580527192    2006  6 26  3:25: 5.468479
     960.00000000   22317.71926039   15574.82086129   35495.77144092 -0.892750056  0.737383381  1.291738834    2006  6 26  5:25: 5.468452
    1080.00000000   13795.68675885   19088.83051008   40803.69584385 -1.420277669  0.235599456  0.185517056    2006  6 26  7:25: 5.468465
    1200.00000000    2515.17145049   18746.63776282   37864.58088636 -1.668016053 -0.360431458 -1.052854596    2006  6 26  9:25: 5.468479
    1320.00000000   -9084.48602106   12982.62608646   24045.63900249 -1.378032363 -1.373184736 -3.013963835    2006  6 26 11:25: 5.468452
    1440.00000000    5647.00909495   -3293.90518693   -5425.85235063  8.507977176  0.414560797  2.543322806    2006  6 26 13:25: 5.468465
    1560.00000000   25111.63372210    7412.55109488   19844.25781729  0.416496290  1.332106006  2.739301737    2006  6 26 15:25: 5.468479
    1680.00000000   22961.47461641   14985.74459578   34511.09257381 -0.816711048  0.789391108  1.407901804    2006  6 26 17:25: 5.468452
    1800.00000000   14841.15301459   18876.91439870   40626.25901619 -1.380403341  0.290228810  0.298258120    2006  6 26 19:25: 5.468465
    1920.00000000    3750.70174081   18978.57939698   38578.11783220 -1.656939412 -0.287930881 -0.910825599    2006  6 26 21:25: 5.468479
    2040.00000000   -8027.30219489   13939.54436955   26136.49045637 -1.474476061 -1.222693624 -2.737178731    2006  6 26 23:25: 5.468452
    2160.00000000   -1296.95657092   -2813.69369768   -5871.09587258  9.881929371 -1.978467207 -1.922261005    2006  6 27  1:25: 5.468465
    2280.00000000   24738.60364819    6383.41644019   17787.27631900  0.639556952  1.392554379  2.906206324    2006  6 27  3:25: 5.468479
    2400.00000000   23546.85388669   14358.15602832   33441.67679479 -0.734895006  0.841564851  1.526009909    2006  6 27  5:25: 5.468452
    2520.00000000   15855.87696303   18624.05633582   40367.13420574 -1.337753546  0.343969522  0.410018472    2006  6 27  7:25: 5.468465
    2640.00000000    4976.44933591   19156.75504042   39189.68603184 -1.642084365 -0.218525096 -0.774148204    2006  6 27  9:25: 5.468479
    2760.00000000   -6909.20746210   14790.44707042   28034.46732222 -1.545152610 -1.088119523 -2.487447214    2006  6 27 11:25: 5.468452
    2880.00000000   -7331.65006707    -604.17323419   -2723.51014575  6.168997265 -3.634011554 -5.963531682    2006  6 27 13:25: 5.468465
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    2006  6 24 12:58:49.772928
     240.00000000   12325.51410155  -38982.15046244    4802.88832275  1.763224157 -0.102514446  0.012397139    2006  6 24 14:58:49.772902
     360.00000000   22773.66831936  -34348.02176606    4228.77407391  1.067616787  1.352427865 -0.166956367    2006  6 24 16:58:49.772915
     480.00000000   26194.40441089  -19482.94203672    2393.84774063 -0.313732186  2.808771328 -0.346204118    2006  6 24 18:58:49.772928
     600.00000000    8893.50573448    5763.38890561    -713.69884164 -7.037399220  3.022613131 -0.370272416    2006  6 24 20:58:49.772902
     720.00000000   -6028.75686537  -25648.99913786    3164.37107274  1.883159288 -3.177051976  0.390793162    2006  6 24 22:58:49.772915
     840.00000000    8313.57299056  -38146.45710922    4697.80777535  1.905002133 -0.625883074  0.076098187    2006  6 25  0:58:49.772928
     960.00000000   20181.29108622  -36842.60674073    4529.12568218  1.326244476  0.921916487 -0.114527455    2006  6 25  2:58:49.772902
    1080.00000000   26302.61794569  -25173.39539436    3084.65309986  0.245398835  2.329974347 -0.287495880    2006  6 25  4:58:49.772915
    1200.00000000   19365.07045602   -2700.00490122     317.42727417 -3.009733018  3.902496058 -0.478928582    2006  6 25  6:58:49.772928
    1320.00000000   -9667.81878780  -16930.19112642    2095.87469034  1.279288285 -4.736005905  0.582878255    2006  6 25  8:58:49.772902
    1440.00000000    4021.31438583  -36066.09209609    4442.91587411  2.007322354 -1.227461376  0.149383897    2006  6 25 10:58:49.772915
23333 xx
       0.00000000   -9301.24542292    3326.10200382    2318.36441127 -8.729303005 -0.828225037 -0.122314827
     120.00000000  -44672.91239680   -6213.11996581   -1738.80131727 -3.719475070 -1.336673022 -0.621888261    1994 11  1 13:59:59.999169
     240.00000000  -67053.08885388  -14994.69685946   -5897.99072793 -2.860576613 -1.183771565 -0.568473909    1994 11  1 15:59:59.999142
     360.00000000  -85227.84253168  -22897.08484471   -9722.59184564 -2.426469823 -1.078592475 -0.525341431    1994 11  1 17:59:59.999155
     480.00000000 -100986.00419136  -30171.19698695  -13283.77044765 -2.147108978 -1.000530827 -0.491587582    1994 11  1 19:59:59.999169
```

```
     600.00000000 -115093.00686387  -36962.56316477  -16634.15682929 -1.945446188 -0.938947736 -0.464199202    1994 11  1 21:59:59.999142
     720.00000000 -127965.80064891  -43363.32967165  -19809.90480432 -1.789652016 -0.888278463 -0.441254468    1994 11  1 23:59:59.999155
     840.00000000 -139863.28332207  -49436.45704153  -22836.80438139 -1.663762568 -0.845315913 -0.421548627    1994 11  2  1:59:59.999169
     960.00000000 -150960.22978259  -55227.45413896  -25734.01408879 -1.558730986 -0.808061065 -0.404293846    1994 11  2  3:59:59.999142
    1080.00000000 -161381.71414630  -60770.64040903  -28516.26290017 -1.468977174 -0.775190459 -0.388951810    1994 11  2  5:59:59.999155
    1200.00000000 -171221.18736947  -66092.76474442  -31195.19847387 -1.390837596 -0.745785633 -0.375140398    1994 11  2  7:59:59.999169
    1320.00000000 -180550.82888746  -71215.23290630  -33780.24938270 -1.321788672 -0.719184752 -0.362579495    1994 11  2  9:59:59.999142
    1440.00000000 -189427.87533074  -76155.54943344  -36279.19882816 -1.260024473 -0.694896053 -0.351058133    1994 11  2 11:59:59.999155
    1560.00000000 -197898.69401409  -80928.29015181  -38698.57972447 -1.204211888 -0.672544709 -0.340413731    1994 11  2 13:59:59.999169
    1600.00000000 -200638.82986236  -82484.14969882  -39488.34331447 -1.186748462 -0.665472422 -0.337037582    1994 11  2 14:39:59.999146
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    2006  6 20 18:42: 6.640047
      40.00000000   11321.71039205  13222.84749156   1602.40119049 -1.151973982  4.285810871  0.521919425    2006  6 20 19: 2: 6.640056
      60.00000000    9438.29395675  17688.05450261   2146.59293402 -1.907904054  3.179955046  0.387692479    2006  6 20 19:22: 6.640025
      80.00000000    6872.08634639  20910.11016811   2539.79945034 -2.323995367  2.207398462  0.269506121    2006  6 20 19:42: 6.640034
     100.00000000    3933.37509798  23024.07662542   2798.25966746 -2.542860616  1.327134966  0.162450076    2006  6 20 20: 2: 6.640043
     120.00000000     816.64091546  24118.98675475   2932.69459428 -2.626838010  0.504502763  0.062344306    2006  6 20 20:22: 6.640052
     140.00000000   -2334.41705804  24246.86096326   2949.36448841 -2.602259646 -0.288058266 -0.034145135    2006  6 20 20:42: 6.640020
     160.00000000   -5394.31798039  23429.42716149   2850.86832586 -2.474434068 -1.074055982 -0.129868366    2006  6 20 21: 2: 6.640029
     180.00000000   -8233.35130237  21661.24480883   2636.51456118 -2.230845533 -1.875742344 -0.227528603    2006  6 20 21:22: 6.640038
     200.00000000  -10693.96497348  18909.88168891   2302.33707548 -1.835912433 -2.716169865 -0.329931880    2006  6 20 21:42: 6.640047
     220.00000000  -12553.89669904  15114.63990716   1840.93573231 -1.212478879 -3.619036996 -0.439970633    2006  6 20 22: 2: 6.640056
     240.00000000  -13450.20591864  10190.57904289   1241.95958736 -0.189082511 -4.596701971 -0.559173899    2006  6 20 22:22: 6.640025
     260.00000000  -12686.60437121   4079.31106161    498.27078614  1.664498211 -5.559889865 -0.676747779    2006  6 20 22:42: 6.640034
     280.00000000   -8672.55867753  -2827.56823315   -342.59644716  5.515079852 -5.551222962 -0.676360044    2006  6 20 23: 2: 6.640043
     300.00000000    1153.31498060  -6411.98692060   -779.87288941  9.689818102  1.388598425  0.167868798    2006  6 20 23:22: 6.640052
     320.00000000    9542.79201056   -533.71253081    -65.73165428  3.926947087  6.459583539  0.785686755    2006  6 20 23:42: 6.640020
     340.00000000   11868.80960100   6861.59590848    833.72780602  0.452957852  5.632811328  0.685262323    2006  6 21  0: 2: 6.640029
     360.00000000   11376.23941678  12858.97121366   1563.40660172 -1.087665695  4.374693347  0.532207051    2006  6 21  0:22: 6.640038
     380.00000000    9547.70300782  17421.48570758   2118.56907515 -1.876540262  3.253891728  0.395810243    2006  6 21  0:42: 6.640047
     400.00000000    7008.51470263  20725.47471227   2520.56064289 -2.308703599  2.270724438  0.276138613    2006  6 21  1: 2: 6.640056
     420.00000000    4082.28135104  22911.04184601   2786.37568309 -2.536665546  1.383670232  0.168153407    2006  6 21  1:22: 6.640025
     440.00000000     969.17978149  24071.23673676   2927.31326579 -2.626695115  0.557172428  0.067536854    2006  6 21  1:42: 6.640034
     460.00000000   -2184.71515444  24261.21671601   2950.08142825 -2.607072866 -0.236887607 -0.029125215    2006  6 21  2: 2: 6.640043
     480.00000000   -5253.42223370  23505.37595671   2857.66120738 -2.484424544 -1.022255436 -0.124714444    2006  6 21  2:22: 6.640052
     500.00000000   -8108.27961017  21800.81688388   2649.72981961 -2.247597251 -1.821159176 -0.221925624    2006  6 21  2:42: 6.640020
     520.00000000  -10594.77795556  19117.80779221   2322.72136979 -1.863118484 -2.656426668 -0.323521502    2006  6 21  3: 2: 6.640029
     540.00000000  -12497.32045995  15398.64085906   1869.69983897 -1.258130763 -3.551583368 -0.432338888    2006  6 21  3:22: 6.640038
     560.00000000  -13467.92475245  10560.90147785   1280.78399181 -0.271870523 -4.520514224 -0.550016092    2006  6 21  3:42: 6.640047
     580.00000000  -12848.18843590   4541.21901842    548.53826427  1.494157156 -5.489585384 -0.667472039    2006  6 21  4: 2: 6.640056
     600.00000000   -9152.70552728  -2344.24950144   -287.98121970  5.127921095 -5.650383025 -0.685989008    2006  6 21  4:22: 6.640025
     620.00000000     280.38490909  -6500.10264018   -790.36092984  9.779619614  0.581815811  0.074171345    2006  6 21  4:42: 6.640034
     640.00000000    9166.25784315  -1093.12552651   -129.49428887  4.316668714  6.438636494  0.785116609    2006  6 21  5: 2: 6.640043
     660.00000000   11794.48942915   6382.21138354    780.88439015  0.604412453  5.731729369  0.697574333    2006  6 21  5:22: 6.640052
     680.00000000   11424.30138324  12494.26088864   1524.33165488 -1.021328075  4.463448968  0.542532698    2006  6 21  5:42: 6.640020
     700.00000000    9652.09867350  17153.84762075   2090.48038336 -1.844516637  3.327522235  0.403915232    2006  6 21  6: 2: 6.640029
     720.00000000    7140.41945884  20539.25485336   2501.21469368 -2.293173684  2.333507912  0.282716311    2006  6 21  6:22: 6.640038
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    2006  6 26  2:58:29.343360
     240.00000000  -32222.92014955  26916.25425799   2468.59996594 -1.973007929 -2.359335071  0.102539376    2006  6 26  4:58:29.343334
     360.00000000  -41413.95109398   7055.51656639   2838.90906671 -0.521665080 -3.029172207 -0.002066843    2006  6 26  6:58:29.343347
     480.00000000  -39402.72251896 -14716.42475223   2441.32678358  1.066928187 -2.878714619 -0.105865729    2006  6 26  8:58:29.343360
     600.00000000  -26751.08889828 -32515.13982431   1384.38865570  2.366228869 -1.951032799 -0.181018498    2006  6 26 10:58:29.343334
     720.00000000   -6874.77975542 -41530.38329422    -46.60245459  3.027415087 -0.494671177 -0.207337260    2006  6 26 12:58:29.343347
     840.00000000   14859.52039042 -39302.58907247  -1465.02482524  2.869609883  1.100123969 -0.177514425    2006  6 26 14:58:29.343360
     960.00000000   32553.14863770 -26398.88401807  -2485.45866002  1.930064459  2.401574539 -0.099250520    2006  6 26 16:58:29.343334
    1080.00000000   41365.67576837  -6298.09965811  -2828.05254033  0.459741276  3.051680214  0.006431872    2006  6 26 18:58:29.343347
```

```
    1200.00000000   38858.83295070   15523.39314924   -2396.86850752 -1.140211488  2.867567143  0.110637217    2006  6 26 20:58:29.343360
    1320.00000000   25701.46068162   33089.42617648   -1308.68556638 -2.428713821  1.897381431  0.184605907    2006  6 26 22:58:29.343334
    1440.00000000    5501.08137100   41590.27784405     138.32522930 -3.050691874  0.409203052  0.207958133    2006  6 27  0:58:29.343347
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    2004  2  7 16:20: 1.494254
   -1320.00000000   27766.34015328  -31724.97000557       9.93297846  2.314236153  2.024903193  0.000660861    2004  2  7 18:20: 1.494268
   -1200.00000000   39932.57237973  -13532.60040454      13.12958252  0.987382819  2.911942843  0.000213298    2004  2  7 20:20: 1.494241
   -1080.00000000   41341.01365441    8305.71681955      12.84988501 -0.605098224  3.014378268 -0.000291034    2004  2  7 22:20: 1.494254
    -960.00000000   31614.99210558   27907.29155353       9.16618797 -2.034243523  2.305014102 -0.000718418    2004  2  8  0:20: 1.494268
    -840.00000000   13375.75227587   39994.27017651       3.05416854 -2.915424366  0.975119874 -0.000955576    2004  2  8  2:20: 1.494241
    -720.00000000   -8464.89963309   41312.93549892      -3.86622919 -3.011600615 -0.617275050 -0.000939664    2004  2  8  4:20: 1.494254
    -600.00000000  -28026.23406158   31507.89995661      -9.76047869 -2.296840160 -2.043607595 -0.000674889    2004  2  8  6:20: 1.494268
    -480.00000000  -40040.01314363   13218.00579413     -13.06594832 -0.963328772 -2.919827983 -0.000231414    2004  2  8  8:20: 1.494241
    -360.00000000  -41268.43291976   -8632.06859693     -12.90661266  0.630042315 -3.009677376  0.000273163    2004  2  8 10:20: 1.494254
    -240.00000000  -31377.85317015  -28156.13970334      -9.32605530  2.054021717 -2.288554158  0.000704959    2004  2  8 12:20: 1.494268
    -120.00000000  -13031.41552688  -40092.33381029      -3.27636660  2.924657466 -0.950541167  0.000949381    2004  2  8 14:20: 1.494241
       0.00000000    8827.15660472  -41223.00971237       3.63482963  3.007087319  0.643701323  0.000941663    2004  2  8 16:20: 1.494254
     120.00000000   28306.85426674  -31243.80147394       9.57216891  2.279137743  2.064316875  0.000684127    2004  2  8 18:20: 1.494268
     240.00000000   40159.05128805  -12845.39151157      12.96086316  0.937265422  2.928448287  0.000245505    2004  2  8 20:20: 1.494241
     360.00000000   41192.55903455    9013.79606759      12.90495666 -0.656727442  3.003543458 -0.000257479    2004  2  8 22:20: 1.494254
     480.00000000   31131.69755798   28445.55681731       9.42419238 -2.073484842  2.269770851 -0.000691233    2004  2  9  0:20: 1.494268
     600.00000000   12687.81846530   40217.83324639       3.44726249 -2.931721827  0.924962230 -0.000940766    2004  2  9  2:20: 1.494241
     720.00000000   -9172.23500245   41161.63475527      -3.43575757 -3.000571486 -0.668847508 -0.000940101    2004  2  9  4:20: 1.494254
     840.00000000  -28562.51093192   31022.45987587      -9.39562161 -2.261449202 -2.082713897 -0.000689669    2004  2  9  6:20: 1.494268
     960.00000000  -40260.77504549   12529.11484344     -12.84915105 -0.913097031 -2.935933528 -0.000256181    2004  2  9  8:20: 1.494241
    1080.00000000  -41114.14376538   -9338.87194483     -12.87952404  0.681588815 -2.998432565  0.000245006    2004  2  9 10:20: 1.494254
    1200.00000000  -30890.01512240  -28690.40750792      -9.48037212  2.092989805 -2.252978152  0.000680459    2004  2  9 12:20: 1.494268
    1320.00000000  -12341.46194020  -40310.06316386      -3.55833201  2.940537098 -0.900219523  0.000934170    2004  2  9 14:20: 1.494241
    1440.00000000    9533.27750818  -41065.52390214       3.30756482  2.995596171  0.695200236  0.000938525    2004  2  9 16:20: 1.494254
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    2006  4 23  4:52:50.805439
    9360.00000000   42135.66858481    1072.99195618      10.83481752 -0.078150602  3.074772455 -0.000380063    2006  4 23  5:52:50.805426
    9400.00000000   41304.75156132    8398.27742944       9.74006214 -0.612515135  3.014117469 -0.000511575    2006  4 23  6:32:50.805444
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    2006  6 23 22:35:47.504573
     240.00000000   -3882.62933791   11960.00543452  -25088.14383845 -2.146773699 -1.372461491 -2.579382089    2006  6 24  0:35:47.504546
     360.00000000  -16785.45507465    -734.79159704  -34300.57085853 -1.386528125 -1.907762641 -0.220949641    2006  6 24  2:35:47.504559
     480.00000000  -23524.16689356  -13629.45124622  -30246.27899200 -0.462846784 -1.586139830  1.269293624    2006  6 24  4:35:47.504573
     600.00000000  -22890.23597092  -22209.35900155  -16769.91946116  0.704351342 -0.671112594  2.432433851    2006  6 24  6:35:47.504546
     720.00000000  -11646.39698980  -19855.44222106    3574.00109607  2.626712727  1.815887329  2.960883901    2006  6 24  8:35:47.504559
     840.00000000    7665.76124241   11159.78946577     345.93813117 -0.584818007  3.193514161 -5.750338922    2006  6 24 10:35:47.504573
     960.00000000   -6369.35388112   10204.80073022  -27844.52150384 -2.050573276 -1.582940542 -2.076075232    2006  6 24 12:35:47.504546
    1080.00000000  -18345.64763145   -2977.76684430  -34394.90760612 -1.243589864 -1.892050757  0.060372061    2006  6 24 14:35:47.504559
    1200.00000000  -23979.74839255  -15436.44139571  -28616.50540218 -0.294973425 -1.482987916  1.478255628    2006  6 24 16:35:47.504573
    1320.00000000  -21921.97167880  -22852.45147658  -13784.85308485  0.945455629 -0.428940995  2.596964378    2006  6 24 18:35:47.504546
    1440.00000000   -8266.43821031  -17210.74590112    6967.95546070  3.082244069  2.665881872  2.712555075    2006  6 24 20:35:47.504559
    1560.00000000    6286.85464535   13809.56328971   -6321.60663781 -1.615964016  1.383135377 -5.358719132    2006  6 24 22:35:47.504573
    1680.00000000   -8730.87526788    8244.63344365  -30039.92372791 -1.935622871 -1.724162072 -1.631224738    2006  6 25  0:35:47.504546
    1800.00000000  -19735.81883249   -5191.76593007  -34166.14974143 -1.097835530 -1.860148418  0.324401050    2006  6 25  2:35:47.504559
    1920.00000000  -24232.73847703  -17112.08243255  -26742.88893252 -0.119786184 -1.364365317  1.680220468    2006  6 25  4:35:47.504573
    2040.00000000  -20654.45640708  -23184.54386047  -10611.55144716  1.209238113 -0.144169639  2.748054938    2006  6 25  6:35:47.504546
    2160.00000000   -4337.15988957  -13410.46817244    9870.45949215  3.532753095  3.772236461  2.088424247    2006  6 25  8:35:47.504559
    2280.00000000    4074.62263523   14698.07548285  -12248.65327973 -2.053824693  0.203325817 -4.607867718    2006  6 25 10:35:47.504573
    2400.00000000  -10950.23438984    6148.66879447  -31736.65532865 -1.809875605 -1.816179062 -1.233364913    2006  6 25 12:35:47.504546
    2520.00000000  -20952.40702045   -7358.71507895  -33633.06643074 -0.948973031 -1.813594137  0.573893078    2006  6 25 14:35:47.504559
    2640.00000000  -24273.48944134  -18637.15546906  -24633.27702390  0.064161440 -1.228537560  1.875728935    2006  6 25 16:35:47.504573
```

```
   2760.00000000  -19057.55468077  -23148.29322082   -7269.38614178  1.500802809  0.195383037  2.879031237    2006  6 25 18:35:47.504546
   2880.00000000      43.69305308   -8145.90299207   11634.57079913  3.780661682  5.105315423  0.714401345    2006  6 25 20:35:47.504559
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    2006  6 26 20:52: 4.079709
    240.00000000    1483.17364291    5395.21248786    4448.65907172  2.560540387  4.039025766 -5.736648561    2006  6 26 22:52: 4.079682
    360.00000000    2801.25607157    5455.03931333   -3692.12865695 -0.595095864 -3.951923117 -6.298799125    2006  6 27  0:52: 4.079695
    480.00000000     411.09332812   -1728.99769152   -6935.45548810 -2.935970964 -6.684085058  1.492800886    2006  6 27  2:52: 4.079709
    600.00000000   -2506.52558454   -6628.98655094    -988.07784497 -1.390577189 -0.556164143  7.312736468    2006  6 27  4:52: 4.079682
    720.00000000   -2090.79884266   -2723.22832193    6266.13356576  1.992640665  6.337529519  3.411803080    2006  6 27  6:52: 4.079695
    840.00000000    1091.80560222    4809.88229503    5172.42897894  2.717483546  4.805518977 -5.030019896    2006  6 27  8:52: 4.079709
    960.00000000    2811.14062300    5950.65707171   -2813.23705389 -0.159662742 -3.121215491 -6.775341949    2006  6 27 10:52: 4.079682
   1080.00000000     805.72698304    -812.16627907   -7067.58483968 -2.798936020 -6.889265977  0.472770873    2006  6 27 12:52: 4.079695
   1200.00000000   -2249.59837532   -6505.84890714   -1956.72365062 -1.731234729 -1.528750230  7.096660885    2006  6 27 14:52: 4.079709
   1320.00000000   -2311.57375797   -3560.99112891    5748.16749600  1.626569751  5.890482233  4.293545048    2006  6 27 16:52: 4.079682
   1440.00000000     688.16056594    4124.87618964    5794.55994449  2.810973665  5.479585563 -4.224866316    2006  6 27 18:52: 4.079695
   1560.00000000    2759.94088230    6329.87271798   -1879.19518331  0.266930672 -2.222670878 -7.119390567    2006  6 27 20:52: 4.079709
   1680.00000000    1171.50677137     125.82053748   -7061.96626202 -2.605687852 -6.958489749 -0.556333225    2006  6 27 22:52: 4.079682
   1800.00000000   -1951.43708472   -6251.71945820   -2886.95472355 -2.024131483 -2.475214272  6.741537478    2006  6 28  0:52: 4.079695
   1920.00000000   -2475.70722288   -4331.90569958    5117.31234924  1.235823539  5.322743371  5.091281211    2006  6 28  2:52: 4.079709
   2040.00000000     281.46097847    3353.51057102    6302.87900650  2.840647273  6.047222485 -3.337085992    2006  6 28  4:52: 4.079682
   2160.00000000    2650.33118860    6584.33434851    -908.29027134  0.675457235 -1.274044972 -7.323921567    2006  6 28  6:52: 4.079695
   2280.00000000    1501.17226597    1066.31132756   -6918.71472952 -2.361891904 -6.889669974 -1.574718619    2006  6 28  8:52: 4.079709
   2400.00000000   -1619.73468334   -5871.14051991   -3760.56587071 -2.264093975 -3.376316601  6.254622256    2006  6 28 10:52: 4.079682
   2520.00000000   -2581.04202505   -5020.05572531    4385.92329047  0.829668458  4.645048038  5.789262667    2006  6 28 12:52: 4.079695
   2640.00000000    -119.22080628    2510.90620488    6687.45615459  2.807575712  6.496549689 -2.384136661    2006  6 28 14:52: 4.079709
   2760.00000000    2486.23806726    6708.18210028      80.43349581  1.057274905 -0.294294027 -7.384689123    2006  6 28 16:52: 4.079682
   2880.00000000    1788.42334580    1990.50530957   -6640.59337725 -2.074169091 -6.683381288 -2.562777776    2006  6 28 18:52: 4.079695
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    2006  6 24 15:41:49.461504
    240.00000000   -3006.50596328   18522.20742011   18941.84078154 -3.375452789  1.032680773 -1.559324534    2006  6 24 17:41:49.461477
    360.00000000  -21607.02086957   15432.59962630     206.62470309 -1.306049851 -1.817011568 -3.163725018    2006  6 24 19:41:49.461491
    480.00000000  -18453.06134549   -3150.83256134  -18685.83030936  2.106017925 -2.860236337 -1.586151870    2006  6 24 21:41:49.461504
    600.00000000    3425.11742384  -18514.73232706  -18588.67200557  3.394666340 -1.003072030  1.610061295    2006  6 24 23:41:49.461477
    720.00000000   21858.23838148  -15101.51661554     387.34517048  1.247973967  1.856017403  3.161439948    2006  6 25  1:41:49.461491
    840.00000000   18360.69935796    3506.55256762   19024.81678979 -2.122684184  2.830618605  1.537510677    2006  6 25  3:41:49.461504
    960.00000000   -3412.84765409   18646.85269710   18748.00359987 -3.366815728  0.986039922 -1.607874972    2006  6 25  5:41:49.461477
   1080.00000000  -21758.08331586   15215.44829478    -180.82181406 -1.250144680 -1.856490448 -3.163774870    2006  6 25  7:41:49.461491
   1200.00000000  -18193.41290284   -3493.85876912  -18877.14757717  2.153326942 -2.852221264 -1.536617760    2006  6 25  9:41:49.461504
   1320.00000000    3833.57386848  -18635.77026711  -18388.68722885  3.384748179 -0.955363841  1.658785020    2006  6 25 11:41:49.461477
   1440.00000000   22002.20074562  -14879.72595593     774.32827099  1.191573619  1.894561165  3.159953047    2006  6 25 13:41:49.461491
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    2006  6 16  7:13:45.407419
    240.00000000    -603.55232010   -2685.13474569   -5891.70274282  7.572519907 -1.975656726  0.121722605    2006  6 16  9:13:45.407392
    360.00000000    4788.22345627     782.56169214    4335.14284621 -4.954509026  3.683346464  4.804645839    2006  6 16 11:13:45.407405
    480.00000000   -6291.84601644    1547.82790772    -453.67116498 -0.308625588 -3.341538574 -7.082659115    2006  6 16 13:13:45.407419
    600.00000000    4480.74573428   -3028.55200374   -3586.94343641  5.320920857  1.199736275  5.626350481    2006  6 16 15:13:45.407392
    720.00000000    -446.42460916    2932.28872588    5759.19389757 -7.561000245  1.550975493 -1.374970885    2006  6 16 17:13:45.407405
    840.00000000   -3713.79581831   -1382.66125130   -5122.45131136  6.090931626 -3.512629733 -3.467571746    2006  6 16 19:13:45.407419
    960.00000000    6058.32017522    -827.47406722    2104.04678651 -1.798403024  3.787067272  6.641439744    2006  6 16 21:13:45.407392
   1080.00000000   -5631.73659006    2623.70953644    1766.49125084 -3.216401578 -2.309140959 -6.788609120    2006  6 16 23:13:45.407405
   1200.00000000    2776.84991560   -3255.36941953   -4837.19667790  6.748135564 -0.193044825  4.005718698    2006  6 17  1:13:45.407419
   1320.00000000    1148.04430837    2486.07343386    5826.34075913 -7.420162295  2.589456382  0.356350006    2006  6 17  3:13:45.407392
   1440.00000000   -4527.90871828    -723.29199041   -4527.44608319  5.121674217 -3.909895427 -4.500218556    2006  6 17  5:13:45.407405
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    2006  6 26 21:27:32.414976
```

```
    240.00000000    5619.19252274   19651.44862280   -7261.38496765 -2.013634213  3.106842861  0.284235517    2006  6 26 23:27:32.414949
    360.00000000   -9708.68629714   26306.14553149   -1204.29478856 -1.824164290 -0.931909596  1.113419052    2006  6 27  1:27:32.414963
    480.00000000  -14394.03162892    6659.30765074    5593.38345858  1.556522911 -4.681657614  0.296912248    2006  6 27  3:27:32.414976
    600.00000000    7712.09476270   15565.72627434   -7342.40465571 -1.646800364  4.070313571 -0.109483081    2006  6 27  5:27:32.414949
    720.00000000   -7558.36739603   27035.11367962   -2385.12054184 -1.999583791 -0.393409283  1.078093515    2006  6 27  7:27:32.414963
    840.00000000  -15495.61862220   11550.15897828    5053.83178121  0.469277336 -4.029761073  0.679054742    2006  6 27  9:27:32.414976
    960.00000000    9167.02568222   10363.65204210   -6871.52576042 -0.881621027  5.223361510 -0.740696297    2006  6 27 11:27:32.414949
   1080.00000000   -5275.80272094   27151.78486008   -3494.50687216 -2.129609388  0.150196480  1.021038089    2006  6 27 13:27:32.414963
   1200.00000000  -15601.37656145   15641.29379850    4217.03266850 -0.249183123 -3.405238557  0.888214503    2006  6 27 15:27:32.414976
   1320.00000000    9301.05872300    3883.15265574   -5477.86477017  0.871447821  6.493677331 -1.885545282    2006  6 27 17:27:32.414949
   1440.00000000   -2914.31065828   26665.20392758   -4511.09814335 -2.216261909  0.710067769  0.940691824    2006  6 27 19:27:32.414963
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    2006  6 25 13:12:14.455025
    240.00000000   23232.82515008   35187.33981802       4.98927428 -2.565776620  1.694193132  0.000163365    2006  6 25 15:12:14.454998
    360.00000000    2467.44290178   42093.60909959       5.15062987 -3.069341800  0.179976276 -0.000031739    2006  6 25 17:12:14.455012
    480.00000000  -18962.59052991   37661.66243819       4.04433258 -2.746151982 -1.382675777 -0.000197633    2006  6 25 19:12:14.455025
    600.00000000  -35285.00095313   23085.44402778       2.08711880 -1.683277908 -2.572893625 -0.000296282    2006  6 25 21:12:14.454998
    720.00000000  -42103.20138132    2291.06228893      -0.13274964 -0.166974816 -3.070104560 -0.000311007    2006  6 25 23:12:14.455012
    840.00000000  -37580.31858370  -19120.40485693      -2.02755702  1.394367848 -2.740341612 -0.000248591    2006  6 26  1:12:14.455025
    960.00000000  -22934.20761876  -35381.23870806      -3.16495932  2.580167539 -1.672360951 -0.000134907    2006  6 26  3:12:14.454998
   1080.00000000   -2109.90332389  -42110.71508198      -3.36507889  3.070935369 -0.153808390 -0.000005855    2006  6 26  5:12:14.455012
   1200.00000000   19282.77774728  -37495.59250598      -2.71861462  2.734400524  1.406220933  0.000103486    2006  6 26  7:12:14.455025
   1320.00000000   35480.60990600  -22779.03375285      -1.52841859  1.661210676  2.587414593  0.000168300    2006  6 26  9:12:14.454998
   1440.00000000   42119.96263499   -1925.77567263      -0.19827433  0.140521206  3.071541613  0.000179561    2006  6 26 11:12:14.455012
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    2005 11 29  0:33:58.939092
     10.00000000   -4769.05061967    2420.46580562    4035.30855837  4.464585796 -1.060923209  6.070907874    2005 11 29  0:38:58.939114
     15.00000000   -3175.45157340    1965.98738086    5582.12569607  6.049639376 -1.935777558  4.148607019    2005 11 29  0:43:58.939096
     20.00000000   -1210.19024802    1281.54541294    6474.68172772  6.920746273 -2.580517337  1.748783868    2005 11 29  0:48:58.939118
     25.00000000     896.73799533     447.12357305    6607.22400507  6.983396282 -2.925846168 -0.872655207    2005 11 29  0:53:58.939101
     30.00000000    2896.99663534    -440.04738594    5954.92675486  6.211488246 -2.926949815 -3.433959806    2005 11 29  0:58:58.939123
     35.00000000    4545.78970167   -1273.55952872    4580.16512984  4.656984233 -2.568711513 -5.638510954    2005 11 29  1: 3:58.939105
     40.00000000    5627.43299371   -1947.94282469    2634.16714930  2.464141047 -1.873985161 -7.195743032    2005 11 29  1: 8:58.939127
     45.00000000    5984.72318534   -2371.37691609     349.87996209 -0.121276950 -0.911981546 -7.859613894    2005 11 29  1:13:58.939109
     50.00000000    5548.43325922   -2480.16469245   -1979.24314527 -2.763269534  0.199691915 -7.482796996    2005 11 29  1:18:58.939092
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    2006  6 19  6:45:41.242102
     40.00000000     -83.44906141    6286.20208453    2223.49837161 -1.113515974  2.530970283 -7.219445568    2006  6 19  7: 5:41.242111
     60.00000000    -958.57681221    3259.26005348   -5722.63732467 -0.101225813 -6.735338321 -3.804851872    2006  6 19  7:25:41.242079
     80.00000000    -255.25619985   -5132.59762974   -4221.27233118  1.077709303 -4.905938824  5.892521264    2006  6 19  7:45:41.242088
    100.00000000     867.44295097   -5038.40402933    4256.73810533  0.479447535  5.032326446  5.857126248    2006  6 19  8: 5:41.242097
    120.00000000     559.16882013    3376.30587937    5699.22017391 -0.906749328  6.646149867 -3.852331832    2006  6 19  8:25:41.242106
    140.00000000    -669.85184205    6196.00229484   -2281.95741770 -0.795804092 -2.752114827 -7.202478520    2006  6 19  8:45:41.242075
    160.00000000    -784.20708019   -1278.53125553   -6449.19892596  0.636702380 -7.595425203  1.431090802    2006  6 19  9: 5:41.242084
    180.00000000     406.15811659   -6607.03115799     148.33021477  1.009818575  0.231843765  7.692047844    2006  6 19  9:25:41.242093
    200.00000000     916.34911813    -884.08649248    6491.09810362 -0.302163049  7.669887109  1.084336909    2006  6 19  9:45:41.242102
    220.00000000    -104.02490970    6304.31821405    1960.08739882 -1.108873823  2.259522809 -7.351147710    2006  6 19 10: 5:41.242111
    240.00000000    -944.61642849    2872.17248379   -5846.94103362 -0.051117686 -6.989747076 -3.413102600    2006  6 19 10:25:41.242079
    260.00000000    -187.16569888   -5404.86163467   -3731.97057618  1.094696706 -4.412110995  6.326060952    2006  6 19 10:45:41.242088
    280.00000000     884.59720467   -4465.74516163    4725.83632696  0.380656028  5.691554046  5.303910983    2006  6 19 11: 5:41.242097
    300.00000000     446.40767236    4086.66839620    5093.05596650 -0.982424447  6.072965199 -4.791630682    2006  6 19 11:25:41.242106
    320.00000000    -752.24467495    5588.35473301   -3275.04092573 -0.661161370 -4.016290740 -6.676898026    2006  6 19 11:45:41.242075
    340.00000000    -643.72872525   -2585.02528560   -5923.01306608  0.807922142 -7.171597814  3.041115058    2006  6 19 12: 5:41.242084
    360.00000000     584.40295819   -6202.35605817    1781.00536019  0.869250450  2.226927514  7.471676765    2006  6 19 12:25:41.242093
    380.00000000     779.59211765    1100.73728301    6311.59529480 -0.599552305  7.721032522 -1.275153027    2006  6 19 12:45:41.242102
    400.00000000    -403.03155588    6399.18000837    -364.12735875 -1.008861924 -0.516636615 -7.799812287    2006  6 19 13: 5:41.242111
```

```
     420.00000000    -852.93910071     192.65232023   -6322.47054784  0.396006194 -7.882964919 -0.289331517    2006  6 19 13:25:41.242079
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    2006  6 26  8:53:44.456634
     240.00000000    1922.17712474    5113.01138342   -4087.08470203 -6.490769651 -0.522350158 -3.896001154    2006  6 26 10:53:44.456607
     360.00000000   -6157.93546882   -2094.70798790   -1941.63730960  0.149900661 -5.175192523  5.604262034    2006  6 26 12:53:44.456620
     480.00000000    2482.64052411   -3268.45944555    5146.38006190  6.501814698  4.402848754 -0.350943511    2006  6 26 14:53:44.456634
     600.00000000    4036.26455287    4827.43347201   -2507.99063955 -5.184409515  1.772280695 -5.331390168    2006  6 26 16:53:44.456607
     720.00000000   -5776.81371622    -118.64155319   -3641.22052418 -2.539917207 -5.622701582  4.403125405    2006  6 26 18:53:44.456620
     840.00000000      67.98699487   -4456.49213473    4863.71794283  7.183809420  2.418917791  2.015642495    2006  6 26 20:53:44.456634
     960.00000000    5520.62207038    3782.38203554    -596.73193161 -3.027966069  3.754152525 -6.013506363    2006  6 26 22:53:44.456607
    1080.00000000   -4528.05104455    1808.46273329   -4816.99727762 -4.808419763 -5.185789345  2.642104494    2006  6 27  0:53:44.456620
    1200.00000000   -2356.61468078   -4852.51202272    3856.53816184  6.688446735  0.118520958  4.021854210    2006  6 27  2:53:44.456634
    1320.00000000    6149.65800134    2173.59423261    1369.29488732 -0.345832777  5.109857861 -5.842951828    2006  6 27  4:53:44.456607
    1440.00000000   -2629.55011449    3400.98040158   -5344.38217129 -6.368548448 -3.998963509  0.577253064    2006  6 27  6:53:44.456620
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    1980 10  2  1:41:24.113771
     240.00000000   -3226.54349155    3503.70977525    4532.80979343  1.000992116 -5.788042888  5.162585826    1980 10  2  3:41:24.113744
     360.00000000    2456.10706533   -6071.93855503    1222.89768554  2.679390040 -0.448290811 -7.228792155    1980 10  2  5:41:24.113757
     480.00000000     787.16457349    2719.91800946   -6043.86662024 -3.759883839  6.277439314  2.397897864    1980 10  2  7:41:24.113771
     600.00000000   -3110.97648029    3121.73026235    4878.15217035  1.244916056 -6.124880425  4.700576353    1980 10  2  9:41:24.113744
     720.00000000    2567.56229695   -6112.50383922     713.96374435  2.440245751  0.098109002 -7.319959258    1980 10  2 11:41:24.113757
     840.00000000     556.05661780    3144.52288201   -5855.34636178 -3.754660143  6.044752775  2.957941672    1980 10  2 13:41:24.113771
     960.00000000   -2982.47940539    2712.61663711    5192.32330472  1.475566773 -6.427737014  4.202420227    1980 10  2 15:41:24.113744
    1080.00000000    2663.08964352   -6115.48290885     196.40072866  2.196121564  0.652415093 -7.362824152    1980 10  2 17:41:24.113757
    1200.00000000     328.54999674    3557.09490552   -5626.21427211 -3.731193288  5.769341172  3.504058731    1980 10  2 19:41:24.113771
    1320.00000000   -2842.06876757    2278.42343492    5472.33437150  1.691852635 -6.693216335  3.671022712    1980 10  2 21:41:24.11374
    1440.00000000    2742.55398832   -6079.67009123    -326.39012649  1.948497651  1.211072678 -7.356193131    1980 10  2 23:41:24.113757
```

Appendix F – Computer Code Listing

Producing computer code in multiple languages is advantageous for testing as many smaller issues were corrected in this process. Although some features do not exist in each language, an attempt was made to separate the mathematical theory, the Input/Output, the debugging, and the extra routines for the main program. The debugging portion is not listed here to reduce the size of the paper, but the full files are available on the website. In addition, the extra routines (sgp4ext) are not included in this listing as they are primarily intended for use with a main program, and they vary widely by language. At a future time, it may be advisable to standardize debugging and warning output routines to handle these cases for integrated programs.

The dependence of each routine is shown below, with parentheses for the name of the file where the routine is found. This was discussed graphically with Fig. 4 in the paper, and is shown in Fig 12.

SGP4EXT- Misc routines for the main program, math, time, etc. (varies by language for intrinsic math routines)
- MAG
- CROSS
- DOT
- ANGLE
- NEWTONNU
- RV2COE
- JDAY
- DAYS2MDHMS
- INVJDAY

SGP4UNIT- SGP4 mathematical routines including GST and getting the constants.
- DPPER, DSCOM, DSPACE, GSTIME, and GETGRAVCONST have no coupling.
- DSINIT
  - – GETGRAVCONST,
- INITL
  - – GETGRAVCONST
  - – GSTIME (sgp4ext)
- SGP4INIT
  - – GETGRAVCONST
  - – INITL
  - – DSCOM
  - – DPPER
  - – DSINIT
  - – SGP4
- SGP4
  - – GETGRAVCONST
  - – DSPACE
  - – DPPER

SGP4IO- TLE data parser
- TWOLINE2RV
  - – SGP4INIT (sgp4unit)
  - – DAYS2MDHMS (sgp4ext)
  - – JDAY (sgp4ext)

TESTCPP- Main driver for test program (last three letters indicate the language)
- MAIN
  - – TWOLINE2RV (sgp4io)
  - – SGP4 (sgp4unit)
  - – INVJDAY (sgp4ext)
  - – RV2COE (sgp4ext)

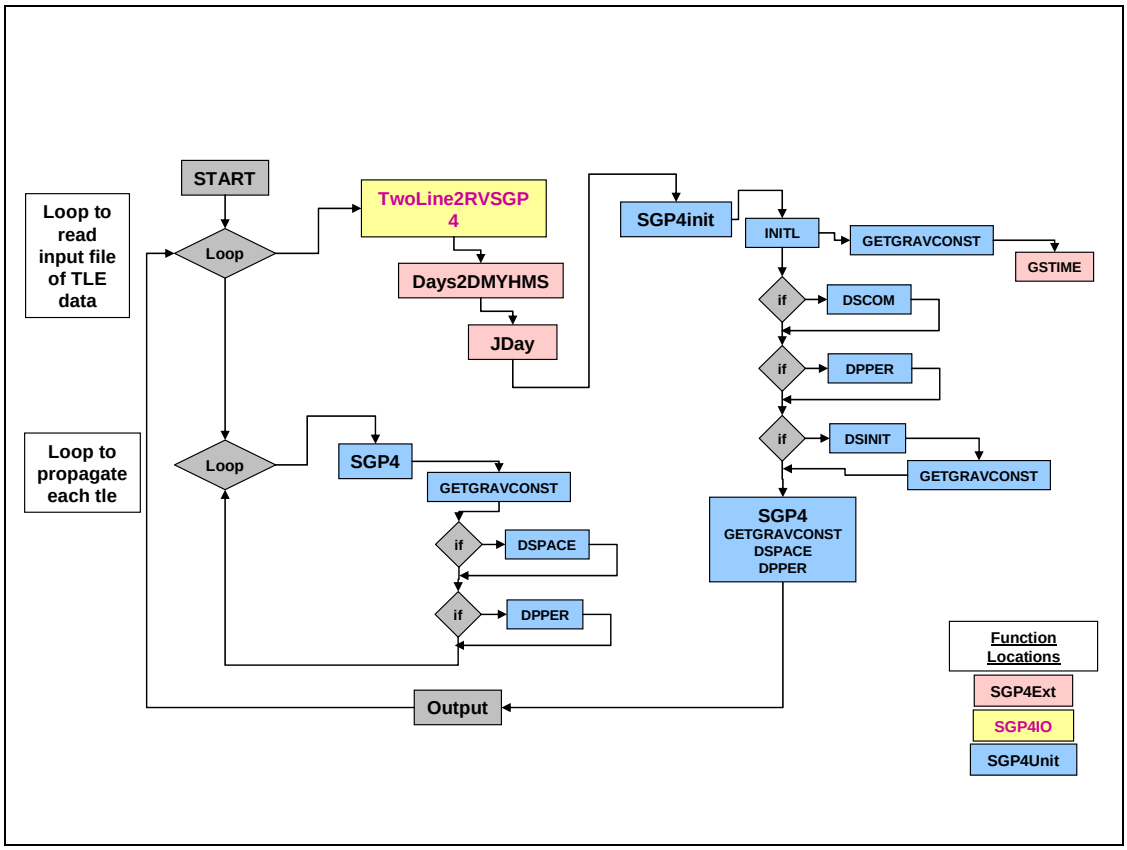


Figure 12. Program Code Structure. An example flowchart shows the relations between the various routines in the revised code. Note that the initialization is required a single time after each new TLE is processed.

```
/* -----------------------------------------------------------------------
*
*                               testcpp.cpp
*
*  this program tests the sgp4 propagator.
*
*                           companion code for
*             fundamentals of astrodynamics and applications
*                                    2004
*                             by david vallado
*
*     (w) 719-573-2600, email dvallado@agi.com
*     *************************************************************
*  current :
*            14 aug 06  david vallado
*                        update mfe for verification time steps, constants
*  changes :
*            20 jul 05  david vallado
*                         fixes for paper, corrections from paul crawford
*             7 jul 04  david vallado
*                         fix record file and get working
*            14 may 01  david vallado
*                         2nd edition baseline
*                   97  nasa
*                         internet version
*                   80  norad
*                         original baseline
*       ----------------------------------------------------------------      */

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <io.h>

#include "sgp4ext.h"
#include "sgp4unit.h"
#include "sgp4io.h"

#define pi 3.14159265358979323846

int main()
{
    char str[1];
    char infilename[12];
    double ro[3];
    double vo[3];
         char typerun;
         gravconsttype  whichconst;
         int whichcon;
    FILE *infile, *outfile, *outfilee;

// ----------------------------  locals  ------------------------------
         double p, a, ecc, incl, node, argp, nu, m, arglat, truelon, lonper;
    double sec,  jd, rad, tsince, startmfe, stopmfe, deltamin;
    int i; int  year; int mon; int day; int hr; int min;
    char longstr1[130];
         typedef char str3[4];
         str3 monstr[13];
         char outname[64];
    char longstr2[130];
    elsetrec satrec;

         rad = 180.0 / pi;
// -------------------------  implementation   --------------------------
strcpy(monstr[1], "Jan");
strcpy(monstr[2], "Feb");
strcpy(monstr[3], "Mar");
strcpy(monstr[4], "Apr");
strcpy(monstr[5], "May");
strcpy(monstr[6], "Jun");
strcpy(monstr[7], "Jul");
strcpy(monstr[8], "Aug");
strcpy(monstr[9], "Sep");
strcpy(monstr[10], "Oct");
strcpy(monstr[11], "Nov");
strcpy(monstr[12], "Dec");

         //typerun = 'c' compare 1 year of full satcat data
         //typerun = 'v' verification run, requires modified elm file with
         //              start stop and delta times
```

```
         printf("input type of run c, v \n");
         scanf( "%c",&typerun );

         printf("input which constants 72 84 \n");
         scanf( "%i",&whichcon );
         if (whichcon == 721) whichconst = wgs72old;
         if (whichcon == 72) whichconst = wgs72;
         if (whichcon == 84) whichconst = wgs84;

         // ---------------- setup files for operation ------------------
         // input 2-line element set file
         printf("input elset filename: \n");
         scanf( "%s",&infilename );
         infile = fopen(infilename, "r");
         if (infile == NULL)
      {
        printf("Failed to open file: %s\n", infilename);
        return 1;
      }

         if (typerun == 'c')
             outfile = fopen("tcppall.out", "w");
           else
             {
             if (typerun == 'v')
                 outfile = fopen("tcppver.out", "w");
               else
                 outfile = fopen("tcpp.out", "w");
             }

//         dbgfile = fopen("sgp4test.dbg", "w");
//         fprintf(dbgfile,"this is the debug output\n\n" );

         // ----------------- test simple propagation -------------------
         while (feof(infile) == 0)
           {
             do
               {
                 fgets( longstr1,130,infile);
                 strncpy(str, &longstr1[0], 1);
                 str[1] = '\0';
               } while ((strcmp(str, "#")==0)&&(feof(infile) == 0));

             if (feof(infile) == 0)
               {
                 fgets( longstr2,130,infile);
                 // convert the char string to sgp4 elements
                 // includes initialization of sgp4
                 twoline2rv( longstr1, longstr2, typerun, whichconst,
                             startmfe, stopmfe, deltamin, satrec );
                 fprintf(outfile, "%ld xx\n", satrec.satnum);
                 printf(" %ld\n", satrec.satnum);
                 // call the propagator to get the initial state vector value
                 sgp4 (whichconst, satrec,  0.0, ro,  vo);

// generate .e files
jd = satrec.jdsatepoch;
strncpy(outname,&longstr1[2],5);
outname[5]= '.';
outname[6]= 'e';
outname[7]= '\0';
invjday( jd, year,mon,day,hr,min, sec );
outfilee = fopen(outname, "w");
fprintf(outfilee,"stk.v.4.3 \n"); // must use 4.3...
fprintf(outfilee,"\n");
fprintf(outfilee,"BEGIN Ephemeris \n");
fprintf(outfilee," \n");
fprintf(outfilee,"NumberOfEphemerisPoints    146 \n");
fprintf(outfilee,"ScenarioEpoch     %3i %3s%5i%3i:%2i:%12.9f \n",day,monstr[mon],
                  year,hr,min,sec );
fprintf(outfilee,"InterpolationMethod    Lagrange \n");
fprintf(outfilee,"InterpolationOrder     5 \n");
fprintf(outfilee,"CentralBody            Earth \n");
fprintf(outfilee,"CoordinateSystem       J2000 \n");
fprintf(outfilee,"CoordinateSystemEpoch  %3i %3s%5i%3i:%2i:%12.9f \n",day,
                  monstr[mon],year,hr,min,sec );
fprintf(outfilee,"DistanceUnit        Kilometers \n");
fprintf(outfilee," \n");
fprintf(outfilee,"EphemerisTimePosVel \n");
fprintf(outfilee," \n");
fprintf(outfilee, " %16.8f %16.8f %16.8f %16.8f %12.9f %12.9f %12.9f\n",
```

```
                 satrec.t,ro[0],ro[1],ro[2],vo[0],vo[1],vo[2]);

                fprintf(outfile, " %16.8f %16.8f %16.8f %16.8f %12.9f %12.9f %12.9f\n",
                        satrec.t,ro[0],ro[1],ro[2],vo[0],vo[1],vo[2]);

                tsince = startmfe;
                // check so the first value isn't written twice
                if ( fabs(tsince) > 1.0e-8 )
                    tsince = tsince - deltamin;

                // loop to perform the propagation
                while ((tsince < stopmfe) && (satrec.error == 0))
                  {
                   tsince = tsince + deltamin;

                   if(tsince > stopmfe)
                       tsince = stopmfe;

                   sgp4 (whichconst, satrec,  tsince, ro,  vo);

                   if (satrec.error > 0)
                       printf("# *** error: t:= %f *** code = %3d\n",
                               satrec.t, satrec.error);

                   if (satrec.error == 0)
                     {
                       if ((typerun != 'v') && (typerun != 'c'))
                         {
                            jd = satrec.jdsatepoch + tsince/1440.0;
                            invjday( jd, year,mon,day,hr,min, sec );

                            fprintf(outfile,
                                    "%5i%3i%3i %2i:%2i:%9.6f %16.8f%16.8f%16.8%12.9f%12.9f%12.9f\n",
                                    year,mon,day,hr,min,sec );
//                            fprintf(outfile, " %16.8f %16.8f %16.8f %16.8f %12.9f %12.9f %12.9f\n",
//                                          tsince,ro[0],ro[1],ro[2],vo[0],vo[1],vo[2]);
                         }
                       else
                         {
                            jd = satrec.jdsatepoch + tsince/1440.0;
                            invjday( jd, year,mon,day,hr,min, sec );

fprintf(outfilee, " %16.6f %16.8f %16.8f %16.8f %12.9f %12.9f %12.9f \n",
                 tsince*60.0,ro[0],ro[1],ro[2],vo[0],vo[1],vo[2]);

                            fprintf(outfile, " %16.8f %16.8f %16.8f %16.8f %12.9f %12.9f %12.9f",
                                        tsince,ro[0],ro[1],ro[2],vo[0],vo[1],vo[2]);
                            rv2coe(ro, vo, p, a, ecc, incl, node, argp, nu, m, arglat, truelon, lonper );

                            fprintf(outfile, " %14.6f %8.6f %10.5f %10.5f %10.5f %10.5f %10.5f %5i%3i%3i
%2i:%2i:%9.6f\n",
                                    a, ecc, incl*rad, node*rad, argp*rad, nu*rad,
                                    m*rad,year,mon,day,hr,min,sec);
                         }
                     } // if satrec.error == 0

                  } // while propagating the orbit

              } // if not eof

fprintf(outfilee," END Ephemeris \n");
fclose (outfilee);

          } // while through the input file

  return 0;
}  // end testcpp
```

```
#ifndef _sgp4ext_
#define _sgp4ext_
/*     ------------------------------------------------------------------
*
*                                sgp4ext.h
*
*    this file contains extra routines needed for the main test program for sgp4.
*   q test these routines are derived from the astro libraries.
*
*                            companion code for
*               fundamentals of astrodynamics and applications
*                                    2004
*                              by david vallado
*
*       (w) 719-573-2600, email dvallado@agi.com
*
*    current :
*              14 aug 06  david vallado
*                           separate from ast libraries
*    changes :
*              14 aug 06  david vallado
*                           original baseline
*       ------------------------------------------------------------------      */

#include <string.h>
#include <math.h>

// ------------------------- function decarations -------------------------

double  sgn
        (
          double
        );
double  mag
        (
          double[]
        );
void    cross
        (
          double[], double[], double[]
        );
double  dot
        (
          double[] , double[]
        );
double  angle
        (
          double[],
          double[]
        );
void    newtonnu
        (
          double ecc, double nu,
          double& e0, double& m
        );
void    rv2coe
        (
          double[], double[],
          double&, double&, double&, double&, double&, double&,
          double&, double&, double&, double&, double&
        );
void    jday
        (
          int, int, int, int, int, double, double&
        );
void    days2mdhms
        (
          int, double, int&, int&, int&, int&, double&
        );
void    invjday
        (
          double, int&, int&, int&, int&, int&, double&
        );

#endif
```

```
#ifndef _sgp4io_
#define _sgp4io_
/*     -----------------------------------------------------------------
*
*                                sgp4io.h;
*
*    this file contains miscallaneous functions to read two line element
*    sets. while not formerly part of the sgp4 mathematical theory, they are
*    required for practical implemenation.
*
*              14 aug 06  david vallado
*                           separate functions, misc doc
*    changes :
*              15 dec 05  david vallado
*                           original baseline
*       ----------------------------------------------------------------      */

#include <math.h>

#include "sgp4ext.h"    // for several misc routines
#include "sgp4unit.h"   // for sgp4init and getgravconst

// ------------------------- function decarations -------------------------

void twoline2rv
     (
       char[130], char[130],
       char,
       gravconsttype,
       double&,
       double&,
       double&,
       elsetrec&
     );

#endif
```

```
/*     ------------------------------------------------------------
*
*                               sgp4io.cpp
*
*    this file contains miscallaneous functions to read two line element
*    sets. while not formerly part of the sgp4 mathematical theory, they are
*    required for practical implemenation.
*
*                            companion code for
*               fundamentals of astrodynamics and applications
*                                    2004
*                              by david vallado
*
*       (w) 719-573-2600, email dvallado@agi.com
*
*    current :
*              14 aug 06  david vallado
*                           separate functions, misc doc
*    changes :
*              15 dec 05  david vallado
*                           original baseline
*       ----------------------------------------------------------------      */

#include "sgp4ext.h"    // for several misc routines
#include "sgp4unit.h"   // for sgp4init and getgravconst
#include "sgp4io.h"

#define pi 3.14159265358979323846

/* -----------------------------------------------------------------------------
*
*                           function twoline2rv
*
*  this function converts the two line element set character string data to
*    variables and initializes the sgp4 variables. several intermediate varaibles
*    and quantities are determined. note that the result is a structure so multiple
*    satellites can be processed simaltaneously 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        :
*    longstr1    - first line of the tle
*    longstr2    - second line of the tle
*    typerun     - type of run                    verification 'v', catalog 'c', 'n'
*    whichconst  - which set of constants to use  72, 84
*
*  outputs       :
*    satrec      - structure containing all the sgp4 satellite information
*
*  coupling      :
*    getgravconst-
*    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
*
*  references    :
*    norad spacetrack report #3
*    vallado, crawford, hujsak, kelso  2006
  ---------------------------------------------------------------------------- */

void twoline2rv
     (
      char      longstr1[130], char longstr2[130],
      char      typerun,
      gravconsttype       whichconst,
      double& startmfe, double& stopmfe, double& deltamin,
      elsetrec& satrec
     )
     {
       const double rad    =   180.0 / pi;       //  57.29577951308230
       const double xpdotp =  1440.0 / (2.0 *pi);  // 229.1831180523293

       double sec, radiusearthkm, tumin, xke, j2, j3, j4, j3oj2;
       int cardnumb, numb, j;
       long revnum = 0, elnum = 0;
```

```
char classification, intldesg[11];
int year = 0;
int mon, day, hr, minute, nexp, ibexp;

getgravconst( whichconst, tumin, radiusearthkm, xke, j2, j3, j4, j3oj2 );

satrec.error = 0;

// set the implied decimal points since doing a formated read
// fixes for bad input data values (missing, ...)
for (j = 10; j <= 15; j++)
    if (longstr1[j] == ' ')
        longstr1[j] = '_';

if (longstr1[44] != ' ')
    longstr1[43] = longstr1[44];
longstr1[44] = '.';
if (longstr1[7] == ' ')
    longstr1[7] = 'U';
if (longstr1[9] == ' ')
    longstr1[9] = '.';
for (j = 45; j <= 49; j++)
    if (longstr1[j] == ' ')
        longstr1[j] = '0';
if (longstr1[51] == ' ')
    longstr1[51] = '0';
if (longstr1[53] != ' ')
    longstr1[52] = longstr1[53];
longstr1[53] = '.';
longstr2[25] = '.';
for (j = 26; j <= 32; j++)
    if (longstr2[j] == ' ')
        longstr2[j] = '0';
if (longstr1[62] == ' ')
    longstr1[62] = '0';
if (longstr1[68] == ' ')
    longstr1[68] = '0';

sscanf(longstr1,"%2d %5ld %1c %10s %2d %12lf %11lf %7lf %2d %7lf %2d %2d %6ld ",
               &cardnumb,&satrec.satnum,&classification, intldesg, &satrec.epochyr,
               &satrec.epochdays,&satrec.ndot, &satrec.nddot, &nexp, &satrec.bstar,
               &ibexp, &numb, &elnum );

if (typerun == 'v')  // run for specified times from the file
  {
    if (longstr2[52] == ' ')
        sscanf(longstr2,"%2d %5ld %9lf %9lf %8lf %9lf %9lf %10lf %6ld %lf %lf %lf \n",
               &cardnumb,&satrec.satnum, &satrec.inclo,
               &satrec.nodeo,&satrec.ecco, &satrec.argpo, &satrec.mo, &satrec.no,
               &revnum, &startmfe, &stopmfe, &deltamin );
      else
        sscanf(longstr2,"%2d %5ld %9lf %9lf %8lf %9lf %9lf %11lf %6ld %lf %lf %lf \n",
               &cardnumb,&satrec.satnum, &satrec.inclo,
               &satrec.nodeo,&satrec.ecco, &satrec.argpo, &satrec.mo, &satrec.no,
               & revnum, &startmfe, &stopmfe, &deltamin );
  }
  else  // simply run -1 day to +1 day or user input times
  {
    if (longstr2[52] == ' ')
        sscanf(longstr2,"%2d %5ld %9lf %9lf %8lf %9lf %9lf %10lf %6ld \n",
               &cardnumb,&satrec.satnum, &satrec.inclo,
               &satrec.nodeo,&satrec.ecco, &satrec.argpo, &satrec.mo, &satrec.no,
               &revnum );
      else
        sscanf(longstr2,"%2d %5ld %9lf %9lf %8lf %9lf %9lf %11lf %6ld \n",
               &cardnumb,&satrec.satnum, &satrec.inclo,
               &satrec.nodeo,&satrec.ecco, &satrec.argpo, &satrec.mo, &satrec.no,
               &revnum );
  }

// ---- find no, ndot, nddot ----
satrec.no   = satrec.no / xpdotp; //* rad/min
satrec.nddot= satrec.nddot * pow(10.0, nexp);
satrec.bstar= satrec.bstar * pow(10.0, ibexp);

// ---- convert to sgp4 units ----
satrec.a    = pow( satrec.no*tumin , (-2.0/3.0) );
satrec.ndot = satrec.ndot  / (xpdotp*1440.0);  //* ? * minperday
satrec.nddot= satrec.nddot / (xpdotp*1440.0*1440);

// ---- find standard orbital elements ----
```

```
    satrec.inclo = satrec.inclo  / rad;
    satrec.nodeo = satrec.nodeo / rad;
    satrec.argpo = satrec.argpo  / rad;
    satrec.mo    = satrec.mo     / rad;

    satrec.alta = satrec.a*(1.0 + satrec.ecco*satrec.ecco) - 1.0;
    satrec.altp = satrec.a*(1.0 - satrec.ecco*satrec.ecco) - 1.0;

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

    // ---- input start stop times manually
    if ((typerun != 'v') && (typerun != 'c'))
      {
        printf("input start min from epoch \n");
        scanf( "%lf",&startmfe );
        printf("input stop min from epoch \n");
        scanf( "%lf",&stopmfe );
        printf("input time step in minutes \n");
        scanf( "%lf",&deltamin );
      }

    // ---- perform complete catalog evaluation
    if (typerun == 'c')
      {
        startmfe = -1440.0;
        stopmfe  =  1440.0;
        deltamin =    20.0;
      }

    // ---------------- temp fix for years from 1950-2049 -------------------
    // --------- correct fix will occur when year is 4-digit in tle ---------
    if (satrec.epochyr < 50)
        year= satrec.epochyr + 2000;
      else
        year= satrec.epochyr + 1900;

    days2mdhms ( year,satrec.epochdays, mon,day,hr,minute,sec );
    jday( year,mon,day,hr,minute,sec, satrec.jdsatepoch );

    // ---------------- initialize the orbit at sgp4epoch -------------------
    sgp4init( whichconst, satrec.satnum, satrec.jdsatepoch-2433281.5, satrec.bstar,
              satrec.ecco, satrec.argpo, satrec.inclo, satrec.mo, satrec.no,
              satrec.nodeo, satrec);
} // end twoline2rv
```

```
#ifndef _sgp4unit_
#define _sgp4unit_
/*     ----------------------------------------------------------------
*
*                               sgp4unit.h
*
*    this file contains the sgp4 procedures. the code was originally
*    released in the 1980 and 1986 papers. in 1997 and 1998, the updated and
*    copmbined code (sgp4 and sdp4) was released by nasa on the internet.
*                seawifs.gsfc.nasa.gov/~seawifsp/src/bobdays/
*
*    current :
*              14 aug 06  david vallado
*                           chg lyddane choice back to strn3, constants, fmod,
*                           separate debug and writes, misc doc
*    changes :
*              26 jul 05  david vallado
*                           fixes for paper
*                           note that each fix is preceded by a
*                           comment with "sgp4fix" and an explanation of
*                           what was changed
*              14 may 01  david vallado
*                           2nd edition baseline
*                     97  nasa
*                           internet version
*                     80  norad
*                           original baseline
*       ----------------------------------------------------------------      */

#include <math.h>
#include <stdio.h>

// ------------------------- structure decarations ----------------------------
typedef enum
{
  wgs72old,
  wgs72,
  wgs84
} gravconsttype;

typedef struct elsetrec
{
  long int  satnum;
  int       epochyr, epochtynumrev;
  int       error;
  char      init, method;

  /* Near Earth */
  int    isimp;
  double aycof  , con41  , cc1    , cc4      , cc5    , d2      , d3   , d4    ,
         delmo  , eta    , argpdot, omgcof   , sinmao , t       , t2cof, t3cof ,
         t4cof  , t5cof  , x1mth2 , x7thm1   , mdot   , nodedot, xlcof , xmcof ,
         nodecf;

  /* Deep Space */
  int    irez;
  double d2201  , d2211  , d3210  , d3222    , d4410  , d4422   , d5220 , d5232 ,
         d5421  , d5433  , dedt   , del1     , del2   , del3    , didt  , dmdt  ,
         dnodt  , domdt  , e3     , ee2      , peo    , pgho    , pho   , pinco ,
         plo    , se2    , se3    , sgh2     , sgh3   , sgh4    , sh2   , sh3   ,
         si2    , si3    , sl2    , sl3      , sl4    , gsto    , xfact , xgh2  ,
         xgh3   , xgh4   , xh2    , xh3      , xi2    , xi3     , xl2   , xl3   ,
         xl4    , xlamo  , zmol   , zmos     , atime  , xli     , xni;

  double a      , altp   , alta   , epochdays, jdsatepoch       , nddot , ndot  ,
         bstar  , rcse   , inclo  , nodeo    , ecco             , argpo , mo    ,
         no;
} elsetrec;

// ------------------------- function decarations -----------------------------
int sgp4init
       (
         gravconsttype whichconst,      const int satn,     const double epoch,
         const double xbstar, const double xecco, const double xargpo,
         const double xinclo, const double xmo,   const double xno,
         const double xnodeo,
         elsetrec& satrec
       );

int sgp4
       (
```

```
          gravconsttype whichconst,
          elsetrec& satrec, double tsince,
          double r[], double v[]
        );

double  gstime
        (
          double
        );

void getgravconst
      (
       gravconsttype,
       double&,
       double&,
       double&,
       double&,
       double&,
       double&,
       double&
      );

#endif
```

```
/*     -----------------------------------------------------------------
*
*                               sgp4unit.cpp
*
*    this file contains the sgp4 procedures. the code was originally
*    released in the 1980 and 1986 papers. in 1997 and 1998, the updated and
*    copmbined code (sgp4 and sdp4) was released by nasa on the internet.
*                seawifs.gsfc.nasa.gov/~seawifsp/src/bobdays/
*
*                            companion code for
*               fundamentals of astrodynamics and applications
*                                    2004
*                              by david vallado
*
*       (w) 719-573-2600, email dvallado@agi.com
*
*    current :
*               14 aug 06  david vallado
*                            chg lyddane choice back to strn3, constants, fmod,
*                            separate debug and writes, misc doc
*    changes :
*               26 jul 05  david vallado
*                            fixes for paper
*                            note that each fix is preceded by a
*                            comment with "sgp4fix" and an explanation of
*                            what was changed
*               14 may 01  david vallado
*                            2nd edition baseline
*                      97  nasa
*                            internet version
*                      80  norad
*                            original baseline
*       ----------------------------------------------------------------      */

#include "sgp4unit.h"

const char help = 'n';
FILE *dbgfile;

#define pi 3.14159265358979323846


/* ----------- local functions - only ever used internally by sgp4 ---------- */
static void dpper
     (
       double e3,     double ee2,    double peo,     double pgho,   double pho,
       double pinco,  double plo,    double se2,     double se3,    double sgh2,
       double sgh3,   double sgh4,   double sh2,     double sh3,    double si2,
       double si3,    double sl2,    double sl3,     double sl4,    double t,
       double xgh2,   double xgh3,   double xgh4,    double xh2,    double xh3,
       double xi2,    double xi3,    double xl2,     double xl3,    double xl4,
       double zmol,   double zmos,   double inclo,
       char init,
       double& ep,    double& inclp, double& nodep,  double& argpp, double& mp
     );

static void dscom
     (
       double epoch,  double ep,     double argpp,   double tc,     double inclp,
       double nodep,  double np,
       double& snodm, double& cnodm, double& sinim,  double& cosim, double& sinomm,
       double& cosomm,double& day,   double& e3,     double& ee2,   double& em,
       double& emsq,  double& gam,   double& peo,    double& pgho,  double& pho,
       double& pinco, double& plo,   double& rtemsq, double& se2,   double& se3,
       double& sgh2,  double& sgh3,  double& sgh4,   double& sh2,   double& sh3,
       double& si2,   double& si3,   double& sl2,    double& sl3,   double& sl4,
       double& s1,    double& s2,    double& s3,     double& s4,    double& s5,
       double& s6,    double& s7,    double& ss1,    double& ss2,   double& ss3,
       double& ss4,   double& ss5,   double& ss6,    double& ss7,   double& sz1,
       double& sz2,   double& sz3,   double& sz11,   double& sz12,  double& sz13,
       double& sz21,  double& sz22,  double& sz23,   double& sz31,  double& sz32,
       double& sz33,  double& xgh2,  double& xgh3,   double& xgh4,  double& xh2,
       double& xh3,   double& xi2,   double& xi3,    double& xl2,   double& xl3,
       double& xl4,   double& nm,    double& z1,     double& z2,    double& z3,
       double& z11,   double& z12,   double& z13,    double& z21,   double& z22,
       double& z23,   double& z31,   double& z32,    double& z33,   double& zmol,
       double& zmos
     );

static void dsinit
     (
```

```
      gravconsttype whichconst,
      double cosim,  double emsq,   double argpo,   double s1,     double s2,
      double s3,     double s4,     double s5,      double sinim,  double ss1,
      double ss2,    double ss3,    double ss4,     double ss5,    double sz1,
      double sz3,    double sz11,   double sz13,    double sz21,   double sz23,
      double sz31,   double sz33,   double t,       double tc,     double gsto,
      double mo,     double mdot,   double no,      double nodeo,  double nodedot,
      double xpidot, double z1,     double z3,      double z11,    double z13,
      double z21,    double z23,    double z31,     double z33,    double ecco,
      double eccsq,  double& em,    double& argpm,  double& inclm, double& mm,
      double& nm,    double& nodem,
      int& irez,
      double& atime, double& d2201, double& d2211,  double& d3210, double& d3222,
      double& d4410, double& d4422, double& d5220,  double& d5232, double& d5421,
      double& d5433, double& dedt,  double& didt,   double& dmdt,  double& dndt,
      double& dnodt, double& domdt, double& del1,   double& del2,  double& del3,
      double& xfact, double& xlamo, double& xli,    double& xni
    );

static void dspace
     (
      int irez,
      double d2201,  double d2211,  double d3210,   double d3222,  double d4410,
      double d4422,  double d5220,  double d5232,   double d5421,  double d5433,
      double dedt,   double del1,   double del2,    double del3,   double didt,
      double dmdt,   double dnodt,  double domdt,   double argpo,  double argpdot,
      double t,      double tc,     double gsto,    double xfact,  double xlamo,
      double no,
      double& atime, double& em,    double& argpm,  double& inclm, double& xli,
      double& mm,    double& xni,   double& nodem,  double& dndt,  double& nm
    );

static void initl
     (
      int satn,      gravconsttype whichconst,
      double ecco,   double epoch,  double inclo,   double& no,
      char& method,
      double& ainv,  double& ao,    double& con41,  double& con42, double& cosio,
      double& cosio2,double& eccsq, double& omeosq, double& posq,
      double& rp,    double& rteosq,double& sinio , double& gsto
    );
```

```
/* -----------------------------------------------------------------------------
*
*                           procedure dpper
*
*  this procedure 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        -
*    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
*
*  locals        :
*    alfdp       -
*    betdp       -
*    cosip  , sinip  , cosop  , sinop  ,
*    dalf        -
*    dbet        -
*    dls         -
*    f2, f3      -
*    pe          -
*    pgh         -
*    ph          -
*    pinc        -
*    pl          -
*    sel   , ses   , sghl  , sghs  , shl   , shs   , sil   , sinzf , sis   ,
*    sll   , sls
*    xls         -
*    xnoh        -
*    zf          -
*    zm          -
*
*  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
  ----------------------------------------------------------------------------*/

static void dpper
     (
       double e3,     double ee2,    double peo,     double pgho,   double pho,
       double pinco,  double plo,    double se2,     double se3,    double sgh2,
       double sgh3,   double sgh4,   double sh2,     double sh3,    double si2,
       double si3,    double sl2,    double sl3,     double sl4,    double t,
       double xgh2,   double xgh3,   double xgh4,    double xh2,    double xh3,
       double xi2,    double xi3,    double xl2,     double xl3,    double xl4,
       double zmol,   double zmos,   double inclo,
       char init,
       double& ep,    double& inclp, double& nodep,  double& argpp, double& mp
     )
{
     /* --------------------- local variables ------------------------ */
```

```
const double twopi = 2.0 * pi;
char ildm;
double alfdp, betdp, cosip, cosop, dalf, dbet, dls,
     f2,    f3,    pe,    pgh,   ph,   pinc, pl ,
     sel,   ses,   sghl,  sghs,  shll, shs,  sil,
     sinip, sinop, sinzf, sis,   sll,  sls,  xls,
     xnoh,  zf,    zm,    zel,   zes,  znl,  zns;

/* ---------------------- constants ----------------------------- */
zns   = 1.19459e-5;
zes   = 0.01675;
znl   = 1.5835218e-4;
zel   = 0.05490;

/* --------------- calculate time varying periodics ----------- */
zm    = zmos + zns * t;
// be sure that the initial call has time set to zero
if (init == 'y')
    zm = zmos;
zf    = zm + 2.0 * zes * sin(zm);
sinzf = sin(zf);
f2    =  0.5 * sinzf * sinzf - 0.25;
f3    = -0.5 * sinzf * cos(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 == 'y')
    zm = zmol;
zf    = zm + 2.0 * zel * sin(zm);
sinzf = sin(zf);
f2    =  0.5 * sinzf * sinzf - 0.25;
f3    = -0.5 * sinzf * cos(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;
shll  = xh2 * f2 + xh3 * f3;
pe    = ses + sel;
pinc  = sis + sil;
pl    = sls + sll;
pgh   = sghs + sghl;
ph    = shs + shll;

if (init == 'n')
  {
  //  0.2 rad = 11.45916 deg
  //  sgp4fix for lyddane choice
  //  add next three lines to set up use of original inclination per strn3 ver
  ildm = 'y';
  if (inclo >= 0.2)
      ildm = 'n';

  pe    = pe - peo;
  pinc  = pinc - pinco;
  pl    = pl - plo;
  pgh   = pgh - pgho;
  ph    = ph - pho;
  inclp = inclp + pinc;
  ep    = ep + pe;
  sinip = sin(inclp);
  cosip = cos(inclp);

  /* ----------------- apply periodics directly ------------ */
  //  sgp4fix for lyddane choice
  //  strn3 used original inclination - this is technically feasible
  //  gsfc used perturbed inclination - also technically feasible
  //  probably best to readjust the 0.2 limit value and limit discontinuity
  //  use next line for original strn3 approach and original inclination
  //  if (inclo >= 0.2)
  //  use next line for gsfc version and perturbed inclination
  if (inclp >= 0.2)
    {
      ph     = ph / sinip;
      pgh    = pgh - cosip * ph;
      argpp  = argpp + pgh;
      nodep  = nodep + ph;
      mp     = mp + pl;
    }
```

```
          else
          {
            /* ---- apply periodics with lyddane modification ---- */
            sinop  = sin(nodep);
            cosop  = cos(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  = fmod(nodep, twopi);
            xls    = mp + argpp + cosip * nodep;
            dls    = pl + pgh - pinc * nodep * sinip;
            xls    = xls + dls;
            xnoh   = nodep;
            nodep  = atan2(alfdp, betdp);
            if (fabs(xnoh - nodep) > pi)
              if (nodep < xnoh)
                 nodep = nodep + twopi;
               else
                 nodep = nodep - twopi;
            mp     = mp + pl;
            argpp = xls - mp - cosip * nodep;
          }
        }   // if init == 'n'

//#include "debug1.cpp"
}  // end dpper
```

```
/*-----------------------------------------------------------------------------
*
*                           procedure dscom
*
*  this procedure 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        -
*
*  locals        :
*    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  , zsing  , zcosgl , zsingl , zcosh  , zsinh  , zcoshl , zsinhl ,
*    zcosi  , zsini  , zcosil , zsinil ,
*    zx          -
*    zy          -
*
*  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
  ----------------------------------------------------------------------------*/

static void dscom
     (
       double epoch,  double ep,     double argpp,   double tc,     double inclp,
       double nodep,  double np,
       double& snodm, double& cnodm, double& sinim,  double& cosim, double& sinomm,
       double& cosomm,double& day,   double& e3,     double& ee2,   double& em,
       double& emsq,  double& gam,   double& peo,    double& pgho,  double& pho,
       double& pinco, double& plo,   double& rtemsq, double& se2,   double& se3,
       double& sgh2,  double& sgh3,  double& sgh4,   double& sh2,   double& sh3,
       double& si2,   double& si3,   double& sl2,    double& sl3,   double& sl4,
       double& s1,    double& s2,    double& s3,     double& s4,    double& s5,
       double& s6,    double& s7,    double& ss1,    double& ss2,   double& ss3,
       double& ss4,   double& ss5,   double& ss6,    double& ss7,   double& sz1,
```

```
  double& sz2,   double& sz3,   double& sz11,   double& sz12,  double& sz13,
  double& sz21,  double& sz22,  double& sz23,   double& sz31,  double& sz32,
  double& sz33,  double& xgh2,  double& xgh3,   double& xgh4,  double& xh2,
  double& xh3,   double& xi2,   double& xi3,    double& xl2,   double& xl3,
  double& xl4,   double& nm,    double& z1,     double& z2,    double& z3,
  double& z11,   double& z12,   double& z13,    double& z21,   double& z22,
  double& z23,   double& z31,   double& z32,    double& z33,   double& zmol,
  double& zmos
)
{
    /* ------------------------- constants ------------------------- */
    const double zes     =  0.01675;
    const double zel     =  0.05490;
    const double c1ss    =  2.9864797e-6;
    const double c1l     =  4.7968065e-7;
    const double zsinis  =  0.39785416;
    const double zcosis  =  0.91744867;
    const double zcosgs  =  0.1945905;
    const double zsings  = -0.98088458;
    const double twopi   =  2.0 * pi;

    /* --------------------- local variables ------------------------ */
    int lsflg;
    double 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;

    nm     = np;
    em     = ep;
    snodm  = sin(nodep);
    cnodm  = cos(nodep);
    sinomm = sin(argpp);
    cosomm = cos(argpp);
    sinim  = sin(inclp);
    cosim  = cos(inclp);
    emsq   = em * em;
    betasq = 1.0 - emsq;
    rtemsq = sqrt(betasq);

    /* ----------------- initialize lunar solar terms --------------- */
    peo    = 0.0;
    pinco  = 0.0;
    plo    = 0.0;
    pgho   = 0.0;
    pho    = 0.0;
    day    = epoch + 18261.5 + tc / 1440.0;
    xnodce = fmod(4.5236020 - 9.2422029e-4 * day, twopi);
    stem   = sin(xnodce);
    ctem   = cos(xnodce);
    zcosil = 0.91375164 - 0.03568096 * ctem;
    zsinil = sqrt(1.0 - zcosil * zcosil);
    zsinhl = 0.089683511 * stem / zsinil;
    zcoshl = sqrt(1.0 - zsinhl * zsinhl);
    gam    = 5.8351514 + 0.0019443680 * day;
    zx     = 0.39785416 * stem / zsinil;
    zy     = zcoshl * ctem + 0.91744867 * zsinhl * stem;
    zx     = atan2(zx, zy);
    zx     = gam + zx - xnodce;
    zcosgl = cos(zx);
    zsingl = sin(zx);

    /* ------------------------- do solar terms ------------------- */
    zcosg = zcosgs;
    zsing = zsings;
    zcosi = zcosis;
    zsini = zsinis;
    zcosh = cnodm;
    zsinh = snodm;
    cc    = c1ss;
    xnoi  = 1.0 / nm;

    for (lsflg = 1; lsflg <= 2; lsflg++)
      {
        a1  =   zcosg * zcosh + zsing * zcosi * zsinh;
        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 = 12.0 * x1 * x1 - 3.0 * x3 * x3;
    z32 = 24.0 * x1 * x2 - 6.0 * x3 * x4;
    z33 = 12.0 * x2 * x2 - 3.0 * x4 * x4;
    z1  =  3.0 *  (a1 * a1 + a2 * a2) + z31 * emsq;
    z2  =  6.0 *  (a1 * a3 + a2 * a4) + z32 * emsq;
    z3  =  3.0 *  (a3 * a3 + a4 * a4) + z33 * emsq;
    z11 = -6.0 * a1 * a5 + emsq *  (-24.0 * x1 * x7-6.0 * x3 * x5);
    z12 = -6.0 *  (a1 * a6 + a3 * a5) + emsq *
           (-24.0 * (x2 * x7 + x1 * x8) - 6.0 * (x3 * x6 + x4 * x5));
    z13 = -6.0 * a3 * a6 + emsq * (-24.0 * x2 * x8 - 6.0 * x4 * x6);
    z21 =  6.0 * a2 * a5 + emsq * (24.0 * x1 * x5 - 6.0 * x3 * x7);
    z22 =  6.0 *  (a4 * a5 + a2 * a6) + emsq *
           (24.0 * (x2 * x5 + x1 * x6) - 6.0 * (x4 * x7 + x3 * x8));
    z23 =  6.0 * a4 * a6 + emsq * (24.0 * x2 * x6 - 6.0 * x4 * x8);
    z1  = z1 + z1 + betasq * z31;
    z2  = z2 + z2 + betasq * z32;
    z3  = z3 + z3 + betasq * z33;
    s3  = cc * xnoi;
    s2  = -0.5 * s3 / rtemsq;
    s4  = s3 * rtemsq;
    s1  = -15.0 * em * s4;
    s5  = x1 * x3 + x2 * x4;
    s6  = x2 * x3 + x1 * x4;
    s7  = x2 * x4 - x1 * x3;

    /* ----------------------- do lunar terms ------------------- */
    if (lsflg == 1)
      {
        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;
      }
  }

zmol = fmod(4.7199672 + 0.22997150  * day - gam, twopi);
zmos = fmod(6.2565837 + 0.017201977 * day, twopi);

/* ------------------------ do solar terms ---------------------- */
se2  =   2.0 * ss1 * ss6;
se3  =   2.0 * ss1 * ss7;
si2  =   2.0 * ss2 * sz12;
si3  =   2.0 * ss2 * (sz13 - sz11);
sl2  =  -2.0 * ss3 * sz2;
```

```
      sl3  =  -2.0 * ss3 * (sz3 - sz1);
      sl4  =  -2.0 * ss3 * (-21.0 - 9.0 * emsq) * zes;
      sgh2 =   2.0 * ss4 * sz32;
      sgh3 =   2.0 * ss4 * (sz33 - sz31);
      sgh4 = -18.0 * ss4 * zes;
      sh2  =  -2.0 * ss2 * sz22;
      sh3  =  -2.0 * ss2 * (sz23 - sz21);

      /* ------------------------ do lunar terms ---------------------- */
      ee2  =   2.0 * s1 * s6;
      e3   =   2.0 * s1 * s7;
      xi2  =   2.0 * s2 * z12;
      xi3  =   2.0 * s2 * (z13 - z11);
      xl2  =  -2.0 * s3 * z2;
      xl3  =  -2.0 * s3 * (z3 - z1);
      xl4  =  -2.0 * s3 * (-21.0 - 9.0 * emsq) * zel;
      xgh2 =   2.0 * s4 * z32;
      xgh3 =   2.0 * s4 * (z33 - z31);
      xgh4 = -18.0 * s4 * zel;
      xh2  =  -2.0 * s2 * z22;
      xh3  =  -2.0 * s2 * (z23 - z21);

//#include "debug2.cpp"
}  // end dscom
```

```
/*-----------------------------------------------------------------------------
*
*                           procedure dsinit
*
*  this procedure provides deep space contributions to mean motion dot due
*    to geopotential resonance with half day and one day orbits.
*
*  author        : david vallado                  719-573-2600   28 jun 2005
*
*  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       :
*    em          - eccentricity
*    argpm       - argument of perigee
*    inclm       - inclination
*    mm          - mean anomaly
*    nm          - mean motion
*    nodem       - right ascension of ascending node
*    irez        - flag for resonance           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
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------------*/

static void dsinit
```

```
    (
      gravconsttype whichconst,
      double cosim,  double emsq,   double argpo,   double s1,     double s2,
      double s3,     double s4,     double s5,      double sinim,  double ss1,
      double ss2,    double ss3,    double ss4,     double ss5,    double sz1,
      double sz3,    double sz11,   double sz13,    double sz21,   double sz23,
      double sz31,   double sz33,   double t,       double tc,     double gsto,
      double mo,     double mdot,   double no,      double nodeo,  double nodedot,
      double xpidot, double z1,     double z3,      double z11,    double z13,
      double z21,    double z23,    double z31,     double z33,    double ecco,
      double eccsq,  double& em,    double& argpm,  double& inclm, double& mm,
      double& nm,    double& nodem,
      int& irez,
      double& atime, double& d2201, double& d2211,  double& d3210, double& d3222,
      double& d4410, double& d4422, double& d5220,  double& d5232, double& d5421,
      double& d5433, double& dedt,  double& didt,   double& dmdt,  double& dndt,
      double& dnodt, double& domdt, double& del1,   double& del2,  double& del3,
      double& xfact, double& xlamo, double& xli,    double& xni
    )
{
    /* ---------------------- local variables ------------------------ */
    const double twopi = 2.0 * pi;

    double ainv2 , aonv=0.0, 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   , shll  , sis   ,
         sini2 , sls   , temp  , temp1 , theta , xno2  , q22   ,
         q31   , q33   , root22, root44, root54, rptim , root32,
         root52, x2o3  , xke   , znl   , emo   , zns   , emsqo,
         tumin, radiusearthkm, j2, j3, j4, j3oj2;

    q22    = 1.7891679e-6;
    q31    = 2.1460748e-6;
    q33    = 2.2123015e-7;
    root22 = 1.7891679e-6;
    root44 = 7.3636953e-9;
    root54 = 2.1765803e-9;
    rptim  = 4.37526908801129966e-3; // this equates to 7.29211514668855e-5 rad/sec
    root32 = 3.7393792e-7;
    root52 = 1.1428639e-7;
    x2o3   = 2.0 / 3.0;
    znl    = 1.5835218e-4;
    zns    = 1.19459e-5;

    // sgp4fix identify constants and allow alternate values
    getgravconst( whichconst, tumin, radiusearthkm, xke, j2, j3, j4, j3oj2 );

    /* -------------------- deep space initialization ------------ */
    irez = 0;
    if ((nm < 0.0052359877) && (nm > 0.0034906585))
        irez = 1;
    if ((nm >= 8.26e-3) && (nm <= 9.24e-3) && (em >= 0.5))
        irez = 2;

    /* ------------------------ do solar terms ------------------- */
    ses  =  ss1 * zns * ss5;
    sis  =  ss2 * zns * (sz11 + sz13);
    sls  = -zns * ss3 * (sz1 + sz3 - 14.0 - 6.0 * emsq);
    sghs =  ss4 * zns * (sz31 + sz33 - 6.0);
    shs  = -zns * ss2 * (sz21 + sz23);
    // sgp4fix for 180 deg incl
    if ((inclm < 5.2359877e-2) || (inclm > pi - 5.2359877e-2))
      shs = 0.0;
    if (sinim != 0.0)
      shs = shs / sinim;
    sgs  = sghs - cosim * shs;

    /* ------------------------- do lunar terms ------------------ */
    dedt = ses + s1 * znl * s5;
    didt = sis + s2 * znl * (z11 + z13);
    dmdt = sls - znl * s3 * (z1 + z3 - 14.0 - 6.0 * emsq);
    sghl = s4 * znl * (z31 + z33 - 6.0);
    shll = -znl * s2 * (z21 + z23);
    // sgp4fix for 180 deg incl
    if ((inclm < 5.2359877e-2) || (inclm > pi - 5.2359877e-2))
        shll = 0.0;
    domdt = sgs + sghl;
    dnodt = shs;
    if (sinim != 0.0)
```

```
  {
    domdt = domdt - cosim / sinim * shll;
    dnodt = dnodt + shll / sinim;
  }

/* ----------- calculate deep space resonance effects -------- */
dndt   = 0.0;
theta  = fmod(gsto + tc * rptim, twopi);
em     = em + dedt * t;
inclm  = inclm + didt * t;
argpm  = argpm + domdt * t;
nodem  = nodem + dnodt * t;
mm     = mm + dmdt * t;
//   sgp4fix for negative inclinations
//   the following if statement should be commented out
//if (inclm < 0.0)
//  {
//    inclm  = -inclm;
//    argpm  = argpm - pi;
//    nodem = nodem + pi;
//  }

/* -------------- initialize the resonance terms ------------- */
if (irez != 0)
  {
    aonv = pow(nm / xke, x2o3);

    /* ---------- geopotential resonance for 12 hour orbits ------ */
    if (irez == 2)
      {
        cosisq = cosim * cosim;
        emo    = em;
        em     = ecco;
        emsqo  = emsq;
        emsq   = eccsq;
        eoc    = em * emsq;
        g201   = -0.306 - (em - 0.64) * 0.440;

        if (em <= 0.65)
          {
            g211 =    3.616  -  13.2470 * em +  16.2900 * emsq;
            g310 =  -19.302  + 117.3900 * em - 228.4190 * emsq +  156.5910 * eoc;
            g322 =  -18.9068 + 109.7927 * em - 214.6334 * emsq +  146.5816 * eoc;
            g410 =  -41.122  + 242.6940 * em - 471.0940 * emsq +  313.9530 * eoc;
            g422 = -146.407  + 841.8800 * em - 1629.014 * emsq + 1083.4350 * eoc;
            g520 = -532.114  + 3017.977 * em - 5740.032 * emsq + 3708.2760 * eoc;
          }
          else
          {
            g211 =   -72.099 +   331.819 * em -   508.738 * emsq +   266.724 * eoc;
            g310 =  -346.844 +  1582.851 * em -  2415.925 * emsq +  1246.113 * eoc;
            g322 =  -342.585 +  1554.908 * em -  2366.899 * emsq +  1215.972 * eoc;
            g410 = -1052.797 +  4758.686 * em -  7193.992 * emsq +  3651.957 * eoc;
            g422 = -3581.690 + 16178.110 * em - 24462.770 * emsq + 12422.520 * eoc;
            if (em > 0.715)
                g520 =-5149.66 + 29936.92 * em - 54087.36 * emsq + 31324.56 * eoc;
              else
                g520 = 1464.74 -  4664.75 * em +  3763.64 * emsq;
          }
        if (em < 0.7)
          {
            g533 = -919.22770 + 4988.6100 * em - 9064.7700 * emsq + 5542.21  * eoc;
            g521 = -822.71072 + 4568.6173 * em - 8491.4146 * emsq + 5337.524 * eoc;
            g532 = -853.66600 + 4690.2500 * em - 8624.7700 * emsq + 5341.4  * eoc;
          }
          else
          {
            g533 =-37995.780 + 161616.52 * em - 229838.20 * emsq + 109377.94 * eoc;
            g521 =-51752.104 + 218913.95 * em - 309468.16 * emsq + 146349.42 * eoc;
            g532 =-40023.880 + 170470.89 * em - 242699.48 * emsq + 115605.82 * eoc;
          }

        sini2=  sinim * sinim;
        f220 =  0.75 * (1.0 + 2.0 * cosim+cosisq);
        f221 =  1.5 * sini2;
        f321 =  1.875 * sinim  *  (1.0 - 2.0 * cosim - 3.0 * cosisq);
        f322 = -1.875 * sinim  *  (1.0 + 2.0 * cosim - 3.0 * cosisq);
        f441 = 35.0 * sini2 * f220;
        f442 = 39.3750 * sini2 * sini2;
        f522 =  9.84375 * sinim * (sini2 * (1.0 - 2.0 * cosim- 5.0 * cosisq) +
                0.33333333 * (-2.0 + 4.0 * cosim + 6.0 * cosisq) );
```

```
            f523 = sinim * (4.92187512 * sini2 * (-2.0 - 4.0 * cosim +
                   10.0 * cosisq) + 6.56250012 * (1.0+2.0 * cosim - 3.0 * cosisq));
            f542 = 29.53125 * sinim * (2.0 - 8.0 * cosim+cosisq *
                   (-12.0 + 8.0 * cosim + 10.0 * cosisq));
            f543 = 29.53125 * sinim * (-2.0 - 8.0 * cosim+cosisq *
                   (12.0 + 8.0 * cosim - 10.0 * cosisq));
            xno2  =  nm * nm;
            ainv2 =  aonv * aonv;
            temp1 =  3.0 * 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  =  2.0 * 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  =  2.0 * temp1 * root54;
            d5421 =  temp * f542 * g521;
            d5433 =  temp * f543 * g533;
            xlamo =  fmod(mo + nodeo + nodeo-theta - theta, twopi);
            xfact =  mdot + dmdt + 2.0 * (nodedot + dnodt - rptim) - no;
            em    = emo;
            emsq  = emsqo;
          }

        /* ---------------- synchronous resonance terms -------------- */
        if (irez == 1)
          {
            g200  = 1.0 + emsq * (-2.5 + 0.8125 * emsq);
            g310  = 1.0 + 2.0 * emsq;
            g300  = 1.0 + emsq * (-6.0 + 6.60937 * emsq);
            f220  = 0.75 * (1.0 + cosim) * (1.0 + cosim);
            f311  = 0.9375 * sinim * sinim * (1.0 + 3.0 * cosim) - 0.75 * (1.0 + cosim);
            f330  = 1.0 + cosim;
            f330  = 1.875 * f330 * f330 * f330;
            del1  = 3.0 * nm * nm * aonv * aonv;
            del2  = 2.0 * del1 * f220 * g200 * q22;
            del3  = 3.0 * del1 * f330 * g300 * q33 * aonv;
            del1  = del1 * f311 * g310 * q31 * aonv;
            xlamo = fmod(mo + nodeo + argpo - theta, twopi);
            xfact = mdot + xpidot - rptim + dmdt + domdt + dnodt - no;
          }

        /* ------------ for sgp4, initialize the integrator ---------- */
        xli   = xlamo;
        xni   = no;
        atime = 0.0;
        nm    = no + dndt;
      }

//#include "debug3.cpp"
}  // end dsinit
```

```
/*-----------------------------------------------------------------------------
*
*                           procedure dspace
*
*  this procedure 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          -
*    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
*
*  locals        :
*    delt        -
*    ft          -
*    theta       -
*    x2li        -
*    x2omi       -
*    xl          -
*    xldot       -
*    xnddt       -
*    xndt        -
*    xomi        -
*
*  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
  ----------------------------------------------------------------------------*/

static void dspace
     (
       int irez,
       double d2201,  double d2211,  double d3210,   double d3222,  double d4410,
       double d4422,  double d5220,  double d5232,   double d5421,  double d5433,
       double dedt,   double del1,   double del2,    double del3,   double didt,
       double dmdt,   double dnodt,  double domdt,   double argpo,  double argpdot,
       double t,      double tc,     double gsto,    double xfact,  double xlamo,
```

```
  double no,
  double& atime, double& em,    double& argpm,  double& inclm, double& xli,
  double& mm,    double& xni,   double& nodem,  double& dndt,  double& nm
)
{
    const double twopi = 2.0 * pi;
    int iretn , iret;
    double delt, ft, theta, x2li, x2omi, xl, xldot , xnddt, xndt, xomi, g22, g32,
         g44, g52, g54, fasx2, fasx4, fasx6, rptim , step2, stepn , stepp;

    ft    = 0.0;
    fasx2 = 0.13130908;
    fasx4 = 2.8843198;
    fasx6 = 0.37448087;
    g22   = 5.7686396;
    g32   = 0.95240898;
    g44   = 1.8014998;
    g52   = 1.0508330;
    g54   = 4.4108898;
    rptim = 4.37526908801129966e-3; // this equates to 7.29211514668855e-5 rad/sec
    stepp =    720.0;
    stepn =   -720.0;
    step2 = 259200.0;

    /* ----------- calculate deep space resonance effects ----------- */
    dndt   = 0.0;
    theta  = fmod(gsto + tc * rptim, twopi);
    em     = em + dedt * t;

    inclm  = inclm + didt * t;
    argpm  = argpm + domdt * t;
    nodem  = nodem + dnodt * t;
    mm     = mm + dmdt * t;

    //   sgp4fix for negative inclinations
    //   the following if statement should be commented out
    //  if (inclm < 0.0)
    // {
    //    inclm = -inclm;
    //    argpm = argpm - pi;
    //    nodem = nodem + pi;
    //  }

    /* - update resonances : numerical (euler-maclaurin) integration - */
    /* ------------------------ epoch restart ----------------------  */
    //   sgp4fix for propagator problems
    //   the following integration works for negative time steps and periods
    //   the specific changes are unknown because the original code was so convoluted

    ft    = 0.0;
    atime = 0.0;
    if (irez != 0)
      {
        if ((atime == 0.0) || ((t >= 0.0) && (atime < 0.0)) ||
            ((t < 0.0) && (atime >= 0.0)))
          {
            if (t >= 0.0)
                delt = stepp;
              else
                delt = stepn;
            atime  = 0.0;
            xni    = no;
            xli    = xlamo;
          }
        iretn = 381; // added for do loop
        iret  =   0; // added for loop
        while (iretn == 381)
          {
            if ((fabs(t) < fabs(atime)) || (iret == 351))
              {
                if (t >= 0.0)
                    delt = stepn;
                  else
                    delt = stepp;
                iret  = 351;
                iretn = 381;
              }
              else
              {
                if (t > 0.0)  // error if prev if has atime:=0.0 and t:=0.0 (ge)
                    delt = stepp;
```

```
              else
                delt = stepn;
              if (fabs(t - atime) >= stepp)
                {
                  iret  = 0;
                  iretn = 381;
                }
                else
                {
                  ft    = t - atime;
                  iretn = 0;
                }
            }

          /* ------------------- dot terms calculated ------------- */
          /* ----------- near - synchronous resonance terms ------- */
          if (irez != 2)
            {
              xndt  = del1 * sin(xli - fasx2) + del2 * sin(2.0 * (xli - fasx4)) +
                      del3 * sin(3.0 * (xli - fasx6));
              xldot = xni + xfact;
              xnddt = del1 * cos(xli - fasx2) +
                      2.0 * del2 * cos(2.0 * (xli - fasx4)) +
                      3.0 * del3 * cos(3.0 * (xli - fasx6));
              xnddt = xnddt * xldot;
            }
            else
            {
              /* --------- near - half-day resonance terms -------- */
              xomi  = argpo + argpdot * atime;
              x2omi = xomi + xomi;
              x2li  = xli + xli;
              xndt  = d2201 * sin(x2omi + xli - g22) + d2211 * sin(xli - g22) +
                    d3210 * sin(xomi + xli - g32)  + d3222 * sin(-xomi + xli - g32)+
                    d4410 * sin(x2omi + x2li - g44)+ d4422 * sin(x2li - g44) +
                    d5220 * sin(xomi + xli - g52)  + d5232 * sin(-xomi + xli - g52)+
                    d5421 * sin(xomi + x2li - g54) + d5433 * sin(-xomi + x2li - g54);
              xldot = xni + xfact;
              xnddt = d2201 * cos(x2omi + xli - g22) + d2211 * cos(xli - g22) +
                    d3210 * cos(xomi + xli - g32) + d3222 * cos(-xomi + xli - g32) +
                    d5220 * cos(xomi + xli - g52) + d5232 * cos(-xomi + xli - g52) +
                    2.0 * (d4410 * cos(x2omi + x2li - g44) +
                    d4422 * cos(x2li - g44) + d5421 * cos(xomi + x2li - g54) +
                    d5433 * cos(-xomi + x2li - g54));
              xnddt = xnddt * xldot;
            }

          /* ----------------------- integrator ------------------- */
          if (iretn == 381)
            {
              xli   = xli + xldot * delt + xndt * step2;
              xni   = xni + xndt * delt + xnddt * step2;
              atime = atime + delt;
            }
        }  // while iretn = 381

      nm = xni + xndt * ft + xnddt * ft * ft * 0.5;
      xl = xli + xldot * ft + xndt * ft * ft * 0.5;
      if (irez != 1)
        {
          mm   = xl - 2.0 * nodem + 2.0 * theta;
          dndt = nm - no;
        }
        else
        {
          mm   = xl - nodem - argpm + theta;
          dndt = nm - no;
        }
      nm = no + dndt;
    }

//#include "debug4.cpp"
}  // end dsspace
```

```
/*-----------------------------------------------------------------------------
*
*                           procedure initl
*
*  this procedure 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
*
*  locals        :
*    ak          -
*    d1          -
*    del         -
*    adel        -
*    po          -
*
*  coupling      :
*    getgravconst
*    gstime      - find greenwich sidereal time from the julian date
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------------*/

static void initl
     (
       int satn,      gravconsttype whichconst,
       double ecco,   double epoch,  double inclo,   double& no,
       char& method,
       double& ainv,  double& ao,    double& con41,  double& con42, double& cosio,
       double& cosio2,double& eccsq, double& omeosq, double& posq,
       double& rp,    double& rteosq,double& sinio , double& gsto
     )
{
     /* --------------------- local variables ------------------------ */
     double ak, d1, del, adel, po, x2o3, j2, xke,
            tumin, radiusearthkm, j3, j4, j3oj2;

     /* ----------------------- earth constants ---------------------- */
     // sgp4fix identify constants and allow alternate values
     getgravconst( whichconst, tumin, radiusearthkm, xke, j2, j3, j4, j3oj2 );
     x2o3   = 2.0 / 3.0;

     /* ------------- calculate auxillary epoch quantities ---------- */
     eccsq  = ecco * ecco;
     omeosq = 1.0 - eccsq;
     rteosq = sqrt(omeosq);
     cosio  = cos(inclo);
     cosio2 = cosio * cosio;

     /* ------------------ un-kozai the mean motion ----------------- */
     ak    = pow(xke / no, x2o3);
     d1    = 0.75 * j2 * (3.0 * cosio2 - 1.0) / (rteosq * omeosq);
     del   = d1 / (ak * ak);
```

```
      adel  = ak * (1.0 - del * del - del *
              (1.0 / 3.0 + 134.0 * del * del / 81.0));
      del   = d1/(adel * adel);
      no    = no / (1.0 + del);

      ao    = pow(xke / no, x2o3);
      sinio = sin(inclo);
      po    = ao * omeosq;
      con42 = 1.0 - 5.0 * cosio2;
      con41 = -con42-cosio2-cosio2;
      ainv  = 1.0 / ao;
      posq  = po * po;
      rp    = ao * (1.0 - ecco);
      method = 'n';

      gsto = gstime(epoch + 2433281.5);

//#include "debug5.cpp"
}  // end initl
```

```
/*-----------------------------------------------------------------------------
*
*                           procedure sgp4init
*
*  this procedure 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 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
*
*  locals        :
*    cnodm  , snodm  , cosim  , sinim  , cosomm , sinomm
*    cc1sq  , cc2    , cc3
*    coef   , coef1
*    cosio4      -
*    day         -
*    dndt        -
*    em          - eccentricity
*    emsq        - eccentricity squared
*    eeta        -
*    etasq       -
*    gam         -
*    argpm       - argument of perigee
*    nodem       -
*    inclm       - inclination
*    mm          - mean anomaly
*    nm          - mean motion
*    perige      - perigee
*    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         -
*
*  coupling      :
*    getgravconst-
*    initl       -
*    dscom       -
*    dpper       -
*    dsinit      -
*    sgp4        -
*
*  references    :
*    hoots, roehrich, norad spacetrack report #3 1980
*    hoots, norad spacetrack report #6 1986
*    hoots, schumacher and glover 2004
*    vallado, crawford, hujsak, kelso  2006
  ----------------------------------------------------------------------------*/

int sgp4init
```

```
    (
      gravconsttype whichconst,       const int satn,     const double epoch,
      const double xbstar,  const double xecco, const double xargpo,
      const double xinclo,  const double xmo,   const double xno,
      const double xnodeo,  elsetrec& satrec
    )
{
    /* --------------------- local variables ------------------------ */
    double 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  ,
         em    , emsq  , eeta  , etasq , gam   , argpm , nodem ,
         inclm , mm    , nm    , 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,
         qzms2t, ss, j2, j3oj2, j4, x2o3, r[3], v[3],
         tumin, radiusearthkm, xke, j3;

    /* ------------------------ initialization --------------------- */
    // sgp4fix divisor for divide by zero check on inclination
    const double temp4    =   1.0 + cos(pi-1.0e-9);

    /* ----------- set all near earth variables to zero ------------ */
    satrec.isimp   = 0;   satrec.method = 'n'; satrec.aycof    = 0.0;
    satrec.con41   = 0.0; satrec.cc1    = 0.0; satrec.cc4      = 0.0;
    satrec.cc5     = 0.0; satrec.d2     = 0.0; satrec.d3       = 0.0;
    satrec.d4      = 0.0; satrec.delmo  = 0.0; satrec.eta      = 0.0;
    satrec.argpdot = 0.0; satrec.omgcof = 0.0; satrec.sinmao   = 0.0;
    satrec.t       = 0.0; satrec.t2cof  = 0.0; satrec.t3cof    = 0.0;
    satrec.t4cof   = 0.0; satrec.t5cof  = 0.0; satrec.x1mth2   = 0.0;
    satrec.x7thm1  = 0.0; satrec.mdot   = 0.0; satrec.nodedot  = 0.0;
    satrec.xlcof   = 0.0; satrec.xmcof  = 0.0; satrec.nodecf   = 0.0;

    /* ----------- set all deep space variables to zero ------------ */
    satrec.irez  = 0;   satrec.d2201 = 0.0; satrec.d2211 = 0.0;
    satrec.d3210 = 0.0; satrec.d3222 = 0.0; satrec.d4410 = 0.0;
    satrec.d4422 = 0.0; satrec.d5220 = 0.0; satrec.d5232 = 0.0;
    satrec.d5421 = 0.0; satrec.d5433 = 0.0; satrec.dedt  = 0.0;
    satrec.del1  = 0.0; satrec.del2  = 0.0; satrec.del3  = 0.0;
    satrec.didt  = 0.0; satrec.dmdt  = 0.0; satrec.dnodt = 0.0;
    satrec.domdt = 0.0; satrec.e3    = 0.0; satrec.ee2   = 0.0;
    satrec.peo   = 0.0; satrec.pgho  = 0.0; satrec.pho   = 0.0;
    satrec.pinco = 0.0; satrec.plo   = 0.0; satrec.se2   = 0.0;
    satrec.se3   = 0.0; satrec.sgh2  = 0.0; satrec.sgh3  = 0.0;
    satrec.sgh4  = 0.0; satrec.sh2   = 0.0; satrec.sh3   = 0.0;
    satrec.si2   = 0.0; satrec.si3   = 0.0; satrec.sl2   = 0.0;
    satrec.sl3   = 0.0; satrec.sl4   = 0.0; satrec.gsto  = 0.0;
    satrec.xfact = 0.0; satrec.xgh2  = 0.0; satrec.xgh3  = 0.0;
    satrec.xgh4  = 0.0; satrec.xh2   = 0.0; satrec.xh3   = 0.0;
    satrec.xi2   = 0.0; satrec.xi3   = 0.0; satrec.xl2   = 0.0;
    satrec.xl3   = 0.0; satrec.xl4   = 0.0; satrec.xlamo = 0.0;
    satrec.zmol  = 0.0; satrec.zmos  = 0.0; satrec.atime = 0.0;
    satrec.xli   = 0.0; satrec.xni   = 0.0;

    // sgp4fix - note the following variables are also passed directly via satrec.
    // it is possible to streamline the sgp4init call by deleting the "x"
    // variables, but the user would need to set the satrec.* values first. we
    // include the additional assignments in case twoline2rv is not used.
    satrec.bstar   = xbstar;
    satrec.ecco    = xecco;
    satrec.argpo   = xargpo;
    satrec.inclo   = xinclo;
    satrec.mo       = xmo;
    satrec.no       = xno;
    satrec.nodeo   = xnodeo;

    /* ------------------------ earth constants ----------------------- */
    // sgp4fix identify constants and allow alternate values
    getgravconst( whichconst, tumin, radiusearthkm, xke, j2, j3, j4, j3oj2 );
    ss     = 78.0 / radiusearthkm + 1.0;
    qzms2t = pow(((120.0 - 78.0) / radiusearthkm), 4);
    x2o3   =  2.0 / 3.0;

    satrec.init = 'y';
    satrec.t  = 0.0;
```

```
      initl
          (
            satn, whichconst, satrec.ecco, epoch, satrec.inclo, satrec.no, satrec.method,
            ainv, ao, satrec.con41, con42, cosio, cosio2, eccsq, omeosq,
            posq, rp, rteosq, sinio, satrec.gsto
          );
      satrec.error = 0;

      if (rp < 1.0)
        {
//          printf("# *** satn%d epoch elts sub-orbital ***\n", satn);
          satrec.error = 5;
        }

      if ((omeosq >= 0.0 ) || ( satrec.no >= 0.0))
        {
          satrec.isimp = 0;
          if (rp < (220.0 / radiusearthkm + 1.0))
              satrec.isimp = 1;
          sfour  = ss;
          qzms24 = qzms2t;
          perige = (rp - 1.0) * radiusearthkm;

          /* - for perigees below 156 km, s and qoms2t are altered - */
          if (perige < 156.0)
            {
              sfour = perige - 78.0;
              if (perige < 98.0)
                  sfour = 20.0;
              qzms24 = pow(((120.0 - sfour) / radiusearthkm), 4.0);
              sfour  = sfour / radiusearthkm + 1.0;
            }
          pinvsq = 1.0 / posq;

          tsi  = 1.0 / (ao - sfour);
          satrec.eta  = ao * satrec.ecco * tsi;
          etasq = satrec.eta * satrec.eta;
          eeta  = satrec.ecco * satrec.eta;
          psisq = fabs(1.0 - etasq);
          coef  = qzms24 * pow(tsi, 4.0);
          coef1 = coef / pow(psisq, 3.5);
          cc2   = coef1 * satrec.no * (ao * (1.0 + 1.5 * etasq + eeta *
                        (4.0 + etasq)) + 0.375 * j2 * tsi / psisq * satrec.con41 *
                        (8.0 + 3.0 * etasq * (8.0 + etasq)));
          satrec.cc1   = satrec.bstar * cc2;
          cc3   = 0.0;
          if (satrec.ecco > 1.0e-4)
              cc3 = -2.0 * coef * tsi * j3oj2 * satrec.no * sinio / satrec.ecco;
          satrec.x1mth2 = 1.0 - cosio2;
          satrec.cc4    = 2.0* satrec.no * coef1 * ao * omeosq *
                             (satrec.eta * (2.0 + 0.5 * etasq) + satrec.ecco *
                             (0.5 + 2.0 * etasq) - j2 * tsi / (ao * psisq) *
                             (-3.0 * satrec.con41 * (1.0 - 2.0 * eeta + etasq *
                             (1.5 - 0.5 * eeta)) + 0.75 * satrec.x1mth2 *
                             (2.0 * etasq - eeta * (1.0 + etasq)) * cos(2.0 * satrec.argpo)));
          satrec.cc5 = 2.0 * coef1 * ao * omeosq * (1.0 + 2.75 *
                        (etasq + eeta) + eeta * etasq);
          cosio4 = cosio2 * cosio2;
          temp1  = 1.5 * j2 * pinvsq * satrec.no;
          temp2  = 0.5 * temp1 * j2 * pinvsq;
          temp3  = -0.46875 * j4 * pinvsq * pinvsq * satrec.no;
          satrec.mdot     = satrec.no + 0.5 * temp1 * rteosq * satrec.con41 + 0.0625 *
                             temp2 * rteosq * (13.0 - 78.0 * cosio2 + 137.0 * cosio4);
          satrec.argpdot  = -0.5 * temp1 * con42 + 0.0625 * temp2 *
                              (7.0 - 114.0 * cosio2 + 395.0 * cosio4) +
                              temp3 * (3.0 - 36.0 * cosio2 + 49.0 * cosio4);
          xhdot1            = -temp1 * cosio;
          satrec.nodedot = xhdot1 + (0.5 * temp2 * (4.0 - 19.0 * cosio2) +
                              2.0 * temp3 * (3.0 - 7.0 * cosio2)) * cosio;
          xpidot            =  satrec.argpdot+ satrec.nodedot;
          satrec.omgcof   = satrec.bstar * cc3 * cos(satrec.argpo);
          satrec.xmcof    = 0.0;
          if (satrec.ecco > 1.0e-4)
              satrec.xmcof = -x2o3 * coef * satrec.bstar / eeta;
          satrec.nodecf = 3.5 * omeosq * xhdot1 * satrec.cc1;
          satrec.t2cof   = 1.5 * satrec.cc1;
          // sgp4fix for divide by zero with xinco = 180 deg
          if (fabs(cosio+1.0) > 1.5e-12)
              satrec.xlcof = -0.25 * j3oj2 * sinio * (3.0 + 5.0 * cosio) / (1.0 + cosio);
            else
```

```
      satrec.xlcof = -0.25 * j3oj2 * sinio * (3.0 + 5.0 * cosio) / temp4;
    satrec.aycof   = -0.5 * j3oj2 * sinio;
    satrec.delmo   = pow((1.0 + satrec.eta * cos(satrec.mo)), 3);
    satrec.sinmao  = sin(satrec.mo);
    satrec.x7thm1  = 7.0 * cosio2 - 1.0;

    /* --------------- deep space initialization ------------ */
    if ((2*pi / satrec.no) >= 225.0)
      {
        satrec.method = 'd';
        satrec.isimp  = 1;
        tc    =  0.0;
        inclm = satrec.inclo;

        dscom
            (
              epoch, satrec.ecco, satrec.argpo, tc, satrec.inclo, satrec.nodeo,
              satrec.no, snodm, cnodm,  sinim, cosim,sinomm,     cosomm,
              day, satrec.e3, satrec.ee2, em,         emsq, gam,
              satrec.peo,  satrec.pgho,   satrec.pho, satrec.pinco,
              satrec.plo,  rtemsq,        satrec.se2, satrec.se3,
              satrec.sgh2, satrec.sgh3,   satrec.sgh4,
              satrec.sh2,  satrec.sh3,    satrec.si2, satrec.si3,
              satrec.sl2,  satrec.sl3,    satrec.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,
              satrec.xgh2, satrec.xgh3,   satrec.xgh4, satrec.xh2,
              satrec.xh3,  satrec.xi2,    satrec.xi3,  satrec.xl2,
              satrec.xl3,  satrec.xl4,    nm, z1, z2, z3, z11,
              z12, z13, z21, z22, z23, z31, z32, z33,
              satrec.zmol, satrec.zmos
            );
        dpper
            (
              satrec.e3, satrec.ee2, satrec.peo, satrec.pgho,
              satrec.pho, satrec.pinco, satrec.plo, satrec.se2,
              satrec.se3, satrec.sgh2, satrec.sgh3, satrec.sgh4,
              satrec.sh2, satrec.sh3, satrec.si2,  satrec.si3,
              satrec.sl2, satrec.sl3, satrec.sl4,  satrec.t,
              satrec.xgh2,satrec.xgh3,satrec.xgh4, satrec.xh2,
              satrec.xh3, satrec.xi2, satrec.xi3,  satrec.xl2,
              satrec.xl3, satrec.xl4, satrec.zmol, satrec.zmos, inclm, satrec.init,
              satrec.ecco, satrec.inclo, satrec.nodeo, satrec.argpo, satrec.mo
            );

        argpm  = 0.0;
        nodem  = 0.0;
        mm     = 0.0;

        dsinit
            (
              whichconst,
              cosim, emsq, satrec.argpo, s1, s2, s3, s4, s5, sinim, ss1, ss2, ss3, ss4,
              ss5, sz1, sz3, sz11, sz13, sz21, sz23, sz31, sz33, satrec.t, tc,
              satrec.gsto, satrec.mo, satrec.mdot, satrec.no, satrec.nodeo,
              satrec.nodedot, xpidot, z1, z3, z11, z13, z21, z23, z31, z33,
              satrec.ecco, eccsq, em, argpm, inclm, mm, nm, nodem,
              satrec.irez,  satrec.atime,
              satrec.d2201, satrec.d2211, satrec.d3210, satrec.d3222 ,
              satrec.d4410, satrec.d4422, satrec.d5220, satrec.d5232,
              satrec.d5421, satrec.d5433, satrec.dedt,  satrec.didt,
              satrec.dmdt,  dndt,         satrec.dnodt, satrec.domdt ,
              satrec.del1,  satrec.del2,  satrec.del3,  satrec.xfact,
              satrec.xlamo, satrec.xli,   satrec.xni
            );
      }

  /* ----------- set variables if not deep space ----------- */
  if (satrec.isimp != 1)
    {
      cc1sq          = satrec.cc1 * satrec.cc1;
      satrec.d2    = 4.0 * ao * tsi * cc1sq;
      temp           = satrec.d2 * tsi * satrec.cc1 / 3.0;
      satrec.d3    = (17.0 * ao + sfour) * temp;
      satrec.d4    = 0.5 * temp * ao * tsi * (221.0 * ao + 31.0 * sfour) *
                       satrec.cc1;
      satrec.t3cof = satrec.d2 + 2.0 * cc1sq;
      satrec.t4cof = 0.25 * (3.0 * satrec.d3 + satrec.cc1 *
                       (12.0 * satrec.d2 + 10.0 * cc1sq));
      satrec.t5cof = 0.2 * (3.0 * satrec.d4 +
                       12.0 * satrec.cc1 * satrec.d3 +
```

```
                            6.0 * satrec.d2 * satrec.d2 +
                            15.0 * cc1sq * (2.0 * satrec.d2 + cc1sq));
              }
           } // if omeosq = 0 ...

           /* finally propogate to zero epoch to initialise all others. */
           if(satrec.error == 0)
               sgp4(whichconst, satrec, 0.0, r, v);

           satrec.init = 'n';

//#include "debug6.cpp"
           return satrec.error;
}  // end sgp4init
```

```
/*-----------------------------------------------------------------------------
*
*                           procedure 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 nasa
*    release on the internet. there are a few fixes that are added to correct
*    known errors in the existing implementations.
*
*  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
*
*  locals        :
*    am          -
*    axnl, aynl        -
*    betal       -
*    cosim   , sinim   , cosomm  , sinomm  , cnod    , snod    , cos2u   ,
*    sin2u   , coseo1  , sineo1  , cosi    , sini    , cosip   , sinip   ,
*    cosisq  , cossu   , sinsu   , cosu    , sinu
*    delm        -
*    delomg      -
*    dndt        -
*    eccm        -
*    emsq        -
*    ecose       -
*    el2         -
*    eo1         -
*    eccp        -
*    esine       -
*    argpm       -
*    argpp       -
*    omgadf      -
*    pl          -
*    r           -
*    rtemsq      -
*    rdotl       -
*    rl          -
*    rvdot       -
*    rvdotl      -
*    su          -
*    t2  , t3   , t4    , tc
*    tem5, temp , temp1 , temp2  , tempa  , tempe  , templ
*    u   , ux   , uy    , uz     , vx     , vy     , vz
*    inclm       - inclination
*    mm          - mean anomaly
*    nm          - mean motion
*    nodem       - right asc of ascending node
*    xinc        -
*    xincp       -
*    xl          -
*    xlm         -
*    mp          -
*    xmdf        -
*    xmx         -
*    xmy         -
*    nodedf      -
*    xnode       -
*    nodep       -
*    np          -
*
*  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
  ----------------------------------------------------------------------------*/

int sgp4
     (
       gravconsttype whichconst, elsetrec& satrec,  double tsince,
       double r[3],  double v[3]
     )
{
     double am   , axnl  , aynl , betal ,  cosim , cnod  ,
         cos2u, coseo1, cosi , cosip ,  cosisq, cossu , cosu,
         delm , delomg, em   , emsq  ,  ecose , el2   , eo1 ,
         ep   , esine , argpm, argpp ,  argpdf, pl,     mrt = 0.0,
         mvt  , 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  ,
         nm   , nodem, xinc , xincp ,  xl    , xlm   , mp  ,
         xmdf , xmx   , xmy  , nodedf, xnode , nodep, tc  , dndt,
         twopi, x2o3  , j2   , j3    , tumin, j4 , xke   , j3oj2, radiusearthkm,
         vkmpersec;
     int ktr;

     /* ------------------ set mathematical constants --------------- */
     // sgp4fix divisor for divide by zero check on inclination
     const double temp4    =   1.0 + cos(pi-1.0e-9);
     twopi = 2.0 * pi;
     x2o3  = 2.0 / 3.0;
     // sgp4fix identify constants and allow alternate values
     getgravconst( whichconst, tumin, radiusearthkm, xke, j2, j3, j4, j3oj2 );
     vkmpersec     = radiusearthkm * xke/60.0;

     /* --------------------- clear sgp4 error flag ----------------- */
     satrec.t     = tsince;
     satrec.error = 0;

     /* ------- update for secular gravity and atmospheric drag ----- */
     xmdf    = satrec.mo + satrec.mdot * satrec.t;
     argpdf  = satrec.argpo + satrec.argpdot * satrec.t;
     nodedf  = satrec.nodeo + satrec.nodedot * satrec.t;
     argpm   = argpdf;
     mm      = xmdf;
     t2      = satrec.t * satrec.t;
     nodem   = nodedf + satrec.nodecf * t2;
     tempa   = 1.0 - satrec.cc1 * satrec.t;
     tempe   = satrec.bstar * satrec.cc4 * satrec.t;
     templ   = satrec.t2cof * t2;

     if (satrec.isimp != 1)
       {
         delomg = satrec.omgcof * satrec.t;
         delm   = satrec.xmcof *
                  (pow((1.0 + satrec.eta * cos(xmdf)), 3) -
                  satrec.delmo);
         temp   = delomg + delm;
         mm     = xmdf + temp;
         argpm  = argpdf - temp;
         t3     = t2 * satrec.t;
         t4     = t3 * satrec.t;
         tempa  = tempa - satrec.d2 * t2 - satrec.d3 * t3 -
                          satrec.d4 * t4;
         tempe  = tempe + satrec.bstar * satrec.cc5 * (sin(mm) -
                          satrec.sinmao);
         templ  = templ + satrec.t3cof * t3 + t4 * (satrec.t4cof +
                          satrec.t * satrec.t5cof);
       }

     nm    = satrec.no;
     em    = satrec.ecco;
     inclm = satrec.inclo;
     if (satrec.method == 'd')
       {
         tc = satrec.t;
         dspace
             (
               satrec.irez,
               satrec.d2201, satrec.d2211, satrec.d3210,
```

```
                satrec.d3222, satrec.d4410, satrec.d4422,
                satrec.d5220, satrec.d5232, satrec.d5421,
                satrec.d5433, satrec.dedt,  satrec.del1,
                satrec.del2,  satrec.del3,  satrec.didt,
                satrec.dmdt,  satrec.dnodt, satrec.domdt,
                satrec.argpo, satrec.argpdot, satrec.t, tc,
                satrec.gsto, satrec.xfact, satrec.xlamo,
                satrec.no, satrec.atime,
                em, argpm, inclm, satrec.xli, mm, satrec.xni,
                nodem, dndt, nm
              );
        } // if method = d

      if (nm <= 0.0)
        {
//          printf("# error nm %f\n", nm);
          satrec.error = 2;
        }
      am = pow((xke / nm),x2o3) * tempa * tempa;
      nm = xke / pow(am, 1.5);
      em = em - tempe;

      // fix tolerance for error recognition
      if ((em >= 1.0) || (em < -0.001) || (am < 0.95))
        {
//          printf("# error em %f\n", em);
          satrec.error = 1;
        }
      if (em < 0.0)
          em  = 1.0e-6;
      mm     = mm + satrec.no * templ;
      xlm    = mm + argpm + nodem;
      emsq   = em * em;
      temp   = 1.0 - emsq;

      nodem  = fmod(nodem, twopi);
      argpm  = fmod(argpm, twopi);
      xlm    = fmod(xlm, twopi);
      mm     = fmod(xlm - argpm - nodem, twopi);

      /* ------------------ compute extra mean quantities ------------- */
      sinim = sin(inclm);
      cosim = cos(inclm);

      /* -------------------- add lunar-solar periodics -------------- */
      ep     = em;
      xincp  = inclm;
      argpp  = argpm;
      nodep  = nodem;
      mp     = mm;
      sinip  = sinim;
      cosip  = cosim;
      if (satrec.method == 'd')
        {
          dpper
              (
                satrec.e3,   satrec.ee2,  satrec.peo,
                satrec.pgho, satrec.pho,  satrec.pinco,
                satrec.plo,  satrec.se2,  satrec.se3,
                satrec.sgh2, satrec.sgh3, satrec.sgh4,
                satrec.sh2,  satrec.sh3,  satrec.si2,
                satrec.si3,  satrec.sl2,  satrec.sl3,
                satrec.sl4,  satrec.t,    satrec.xgh2,
                satrec.xgh3, satrec.xgh4, satrec.xh2,
                satrec.xh3,  satrec.xi2,  satrec.xi3,
                satrec.xl2,  satrec.xl3,  satrec.xl4,
                satrec.zmol, satrec.zmos, satrec.inclo,
                'n', ep, xincp, nodep, argpp, mp
              );
          if (xincp < 0.0)
            {
              xincp  = -xincp;
              nodep = nodep + pi;
              argpp  = argpp - pi;
            }
          if ((ep < 0.0 ) || ( ep > 1.0))
            {
 //             printf("# error ep %f\n", ep);
              satrec.error = 3;
            }
        } // if method = d
```

```
      /* -------------------- long period periodics ------------------ */
      if (satrec.method == 'd')
        {
          sinip =  sin(xincp);
          cosip =  cos(xincp);
          satrec.aycof = -0.5*j3oj2*sinip;
          // sgp4fix for divide by zero for xincp = 180 deg
          if (fabs(cosip+1.0) > 1.5e-12)
              satrec.xlcof = -0.25 * j3oj2 * sinip * (3.0 + 5.0 * cosip) / (1.0 + cosip);
            else
              satrec.xlcof = -0.25 * j3oj2 * sinip * (3.0 + 5.0 * cosip) / temp4;
        }
      axnl = ep * cos(argpp);
      temp = 1.0 / (am * (1.0 - ep * ep));
      aynl = ep* sin(argpp) + temp * satrec.aycof;
      xl   = mp + argpp + nodep + temp * satrec.xlcof * axnl;

      /* --------------------- solve kepler's equation --------------- */
      u    = fmod(xl - nodep, twopi);
      eo1  = u;
      tem5 = 9999.9;
      ktr = 1;
      //   sgp4fix for kepler iteration
      //   the following iteration needs better limits on corrections
      while (( fabs(tem5) >= 1.0e-12) && (ktr <= 10) )
        {
          sineo1 = sin(eo1);
          coseo1 = cos(eo1);
          tem5   = 1.0 - coseo1 * axnl - sineo1 * aynl;
          tem5   = (u - aynl * coseo1 + axnl * sineo1 - eo1) / tem5;
          if(fabs(tem5) >= 0.95)
              tem5 = tem5 > 0.0 ? 0.95 : -0.95;
          eo1    = eo1 + tem5;
          ktr = ktr + 1;
        }

      /* ------------- short period preliminary quantities ----------- */
      ecose = axnl*coseo1 + aynl*sineo1;
      esine = axnl*sineo1 - aynl*coseo1;
      el2   = axnl*axnl + aynl*aynl;
      pl    = am*(1.0-el2);
      if (pl < 0.0)
        {
//          printf("# error pl %f\n", pl);
          satrec.error = 4;
        }
        else
        {
          rl     = am * (1.0 - ecose);
          rdotl  = sqrt(am) * esine/rl;
          rvdotl = sqrt(pl) / rl;
          betal  = sqrt(1.0 - el2);
          temp   = esine / (1.0 + betal);
          sinu   = am / rl * (sineo1 - aynl - axnl * temp);
          cosu   = am / rl * (coseo1 - axnl + aynl * temp);
          su     = atan2(sinu, cosu);
          sin2u  = (cosu + cosu) * sinu;
          cos2u  = 1.0 - 2.0 * sinu * sinu;
          temp   = 1.0 / pl;
          temp1  = 0.5 * j2 * temp;
          temp2  = temp1 * temp;

          /* -------------- update for short period periodics ------------ */
          if (satrec.method == 'd')
            {
              cosisq                 = cosip * cosip;
              satrec.con41  = 3.0*cosisq - 1.0;
              satrec.x1mth2 = 1.0 - cosisq;
              satrec.x7thm1 = 7.0*cosisq - 1.0;
            }
          mrt   = rl * (1.0 - 1.5 * temp2 * betal * satrec.con41) +
                  0.5 * temp1 * satrec.x1mth2 * cos2u;
          su    = su - 0.25 * temp2 * satrec.x7thm1 * sin2u;
          xnode = nodep + 1.5 * temp2 * cosip * sin2u;
          xinc  = xincp + 1.5 * temp2 * cosip * sinip * cos2u;
          mvt   = rdotl - nm * temp1 * satrec.x1mth2 * sin2u / xke;
          rvdot = rvdotl + nm * temp1 * (satrec.x1mth2 * cos2u +
                  1.5 * satrec.con41) / xke;

          /* --------------------- orientation vectors ------------------- */
```

```
          sinsu =  sin(su);
          cossu =  cos(su);
          snod  =  sin(xnode);
          cnod  =  cos(xnode);
          sini  =  sin(xinc);
          cosi  =  cos(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 (in km and km/sec) ---------- */
          r[0] = (mrt * ux)* radiusearthkm;
          r[1] = (mrt * uy)* radiusearthkm;
          r[2] = (mrt * uz)* radiusearthkm;
          v[0] = (mvt * ux + rvdot * vx) * vkmpersec;
          v[1] = (mvt * uy + rvdot * vy) * vkmpersec;
          v[2] = (mvt * uz + rvdot * vz) * vkmpersec;
        }  // if pl > 0

      // sgp4fix for decaying satellites
      if (mrt < 1.0)
        {
//         printf("# decay condition %11.6f \n",mrt);
        satrec.error = 6;
        }

//#include "debug7.cpp"
      return satrec.error;
}  // end sgp4
```

```
/* -----------------------------------------------------------------------------
*
*                           function gstime
*
*  this function finds the greenwich sidereal time.
*
*  author        : david vallado                  719-573-2600    1 mar 2001
*
*  inputs          description                    range / units
*    jdut1       - julian date in ut1             days from 4713 bc
*
*  outputs       :
*    gstime      - greenwich sidereal time        0 to 2pi rad
*
*  locals        :
*    temp        - temporary variable for doubles   rad
*    tut1        - julian centuries from the
*                  jan 1, 2000 12 h epoch (ut1)
*
*  coupling      :
*    none
*
*  references    :
*    vallado       2004, 191, eq 3-45
* --------------------------------------------------------------------------- */

double  gstime
        (
          double jdut1
        )
   {
     const double twopi = 2.0 * pi;
     const double deg2rad = pi / 180.0;
     double       temp, tut1;

     tut1 = (jdut1 - 2451545.0) / 36525.0;
     temp = -6.2e-6* tut1 * tut1 * tut1 + 0.093104 * tut1 * tut1 +
             (876600.0*3600 + 8640184.812866) * tut1 + 67310.54841;  // sec
     temp = fmod(temp * deg2rad / 240.0, twopi); //360/86400 = 1/240, to deg, to rad

     // ------------------------ check quadrants ---------------------
     if (temp < 0.0)
         temp += twopi;

     return temp;
   }  // end gstime
```

```
/* -----------------------------------------------------------------------------
*
*                           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  72, 84
*
*  outputs       :
*    tumin       - minutes in one time unit
*    radiusearthkm - radius of the earth in km
*    xke         - reciprocal of tumin
*    j2, j3, j4  - un-normalized zonal harmonic values
*    j3oj2       - j3 divided by j2
*
*  locals        :
*    mu          - earth gravitational parameter
*
*  coupling      :
*    none
*
*  references    :
*    norad spacetrack report #3
*    vallado, crawford, hujsak, kelso  2006
  ---------------------------------------------------------------------------- */
void getgravconst
     (
      gravconsttype whichconst,
      double& tumin,
      double& radiusearthkm,
      double& xke,
      double& j2,
      double& j3,
      double& j4,
      double& j3oj2
     )
     {
       double mu;
       switch (whichconst)
         {
           // -- wgs-72 low precision str#3 constants --
           case wgs72old:
           radiusearthkm = 6378.135;     // km
           xke    = 0.0743669161;
           tumin  = 1.0 / xke;
           j2     =   0.001082616;
           j3     =  -0.00000253881;
           j4     =  -0.00000165597;
           j3oj2  =  j3 / j2;
         break;
           // ------------ wgs-72 constants ------------
           case wgs72:
           mu     = 398600.8;            // in km3 / s2
           radiusearthkm = 6378.135;     // km
           xke    = 60.0 / sqrt(radiusearthkm*radiusearthkm*radiusearthkm/mu);
           tumin  = 1.0 / xke;
           j2     =   0.001082616;
           j3     =  -0.00000253881;
           j4     =  -0.00000165597;
           j3oj2  =  j3 / j2;
         break;
           case wgs84:
           // ------------ wgs-84 constants ------------
           mu     = 398600.5;            // in km3 / s2
           radiusearthkm = 6378.137;     // km
           xke    = 60.0 / sqrt(radiusearthkm*radiusearthkm*radiusearthkm/mu);
           tumin  = 1.0 / xke;
           j2     =   0.00108262998905;
           j3     =  -0.00000253215306;
           j4     =  -0.00000161098761;
           j3oj2  =  j3 / j2;
         break;
         default:
           fprintf(stderr,"unknown gravity option (%d)\n",whichconst);
         break;
         }
     }   // end getgravconst
```