AN IMPLEMENTATION OF SGP4 IN NON-SINGULAR VARIABLES USING A FUNCTIONAL PARADIGM

*Martín Lara**

GRUCACI – Scientific Computation Group
University of La Rioja
C/ Luis de Ulloa, s/n. Edificio Vives,
26004 Logroño, Spain

Pablo Pita Leira

pablo.pita@gmail.com

ABSTRACT

A new implementation of the SGP4 algorithm is presented, which allows for choosing different sets of variables in the computation of the periodic corrections. The project is implemented in Scala, a hybrid functional/object oriented programming language running in the Java Virtual Machine. Validation of the new implementations is made by carrying out different tests based on Vallado's results. Finally, applicability for massive data processing tasks like prediction of orbital collision events and performance are discussed.

Index Terms— SGP4, Orbital Propagation, Scala, Perturbation theory, non-singular variables

1. INTRODUCTION TO SGP4EXTENSIONS

The SGP4 (Simplified General Perturbations 4) orbit propagator is a widely used tool for the fast, short term propagation of earth satellite orbits. The algorithms in which it is based are thoroughly described in the SPACETRACK report #3 [1], as well as in Vallado et al. update [2]. Current implementations of SGP4 are based on Brouwer's gravity solution [3] and Lane atmospheric model [4], but using Lyddane's modifications for avoiding loss of precision in the evaluation of the periodic corrections [5], which are, besides, notably simplified for improving evaluation efficiency. Different alternatives in the literature discuss other variable sets, either canonical or not, that can be used in the computation of periodic corrections (see, for instance, [6, 7, 8, 9]).

Due to its popularity, there are numerous versions of SGP4 in several programming languages. The most important implementation comes from David Vallado [2] who has done a comprehensive analysis of the algorithm. Many other versions are derived from his work. The work presented in this paper, SGP4Extensions, also derives from his software. It is written in Scala and can be compiled to run in the Java Virtual Machine or in the browser with the scala.js compiler backend. The SGP4Extensions software however is designed differently to those versions from Vallado:

1. It is heavily influenced by the functional software paradigm.
2. Implementations using other variables and/or extra terms can be easily introduced to create new propagation algorithms
3. Equations in the code have almost always the same notation as expressed in the papers to allow easy review

In particular, the usage of Lara's non-singular variables [9] is optional for all periodic corrections, in this way avoiding the mixture of Lyddane's and polar variables used for the long- and short-period corrections, respectively, in the original implementation.

In the current version of SGP4Extensions, only extensions around SGP4 are implemented. The deep space algorithm SDP4 is not implemented.

The project is hosted in
`https://github.com/pleira/SGP4Extensions`
and it is licensed with an open source Apache Version 2 license.

2. SCALA FEATURES

Scala has been created by Martin Odersky as an statically typed language fusing the object oriented and functional paradigms [10]. It is designed with Java compatibility in mind. Users normally compile scala programs to Java byte code that will run in the Java Virtual Machine.

Scala fully supports the functional paradigm where functions are implemented in a referential transparent way, called pure functions with no side effects [11, p. 3].

*Funded by the Ministry of Economic Affairs and Competitiveness of Spain: Projects ESP2013-41634-P and ESP2014-57071-R.

There is a separation of the run time part, like for example reading/writing from/to files, called generally the interpreter, from a pure functional part, the description, which is just responsible to describe the algorithms that will be applied. The algorithms in the description just return new immutable structures with the calculations carried out. In this implementation of SGP4, mutable state and side effects have been completely avoided in the algorithm allowing easily to support software concerns like testability and parallelism.

Scala's support for unicode to represent algebraic equations has been explored in the literature [12]. Unicode is used intensively in SGP4Extensions with some simplifications to alleviate the notation: no primes nor overdots for derivatives have been done in the code. As example, equations giving Lara's corrections in Sec.3.2 are so stated in the code in the scala trait LaraFirstOrderCorrections.

Regarding optimizations, SGP4Extensions uses the @specialized annotation for the Double numeric type. The usage of primitive types like Double comes with the cost of boxing/unboxing operations when working with unspecialized methods as the compiler creates wrapper objects for that, an operation called boxing. With @specialized the compiler will support creating specialized methods or classes for the declared primitive types that avoids creating the boxing objects.

2.1. Scala libraries used

The current version of the SGP4Extensions code uses the Scala libraries Spire [13], Scalactic [14] and Scalatest [15].

Spire is a library that provides efficient numeric types and mathematical functions. The implementation of most of the functions and methods in SGP4Extensions are parameterized through Spire's type classes. As example, the algorithm to calculate Lane's coefficients for atmospheric drag just needs Spire's Field type class as its operations are expressed via products and sums:

```
def calcLaneCoefs[@sp(Double) F : Field]
 (gcoefs : GeoPotentialCoefs[F]) : LaneCoefs[F] = {
 import gcoefs._
 val `C1²` = C1*C1
 LaneCoefs(
   3*C1/2,
   D2 + 2*`C1²`,
   (3*D3 + C1*(12*D2 + 10 * `C1²`))/4,
   (3*D4 + 12*C1*D3 + 6*D2*D2
                    + 15*`C1²`*(2*D2+`C1²`))/5)
}
```

For most of the methods, the following type classes are needed:

- Field: provides for the sum, product and division operation following associative laws.
- Trig: provides trigonometric functions.
- NRoot: provides fractional exponents, square root and nroots.
- Order: type-safe equivalence and comparison functions.

As example, the trigonometric series for the geopotential coefficients use these four type classes.

The user has then a choice of a particular numeric class to define a SGP4 propagator. The Double type has been used for testing SGP4Extensions as it is fast and sufficient for almost all cases. The @sp annotation is an alias for @specialized, and therefore, a version of this method is created by the compiler using the primitive representation of double in the Java Virtual Machine avoiding the creation of objects. But the user has a choice for other numeric types. As example, an exact numeric type like Real could be used. A numeric type like Interval[Double] would allow for automatic calculation of errors when those are defined as intervals. These numeric types can have uses in certain areas but their execution times are several orders of magnitude slower than Double.

Scalactic provides support for types expressing either the good result or an error message and for comparing Doubles with a certain tolerance.

```
// final results in cartesians
type SGPPropCtx[F] = (CartesianElems[F],
          CartesianElems[F], SpecialPolarNodal[F])
type SGPPropResult[F] = SGPPropCtx[F] Or ErrorMessage
```

Scalatest is used to automatically check the results of the different algorithms.

2.2. Description of the types used

Several groups of types are defined to make clear the processing steps involved in this new SGP4 implementation. First, the coordinate types in the TEME (True Equator Mean Equinox) reference system and supporting classes for coordinate transformations:

- SGPElems: classical keplerian elements plus bStar drag coefficient and julian day
- CartesianElems: orbital position and velocity reference frame
- SpecialPolarNodal: polar nodal coordinates with the inclination instead of the polar component of the angular momentum
- LaraNonSingular: non singular variables

- LyddaneElems

Then, models for the corrections like BrouwerLaneSecularCorrections, LyddaneLongPeriodCorrections, Lyddane2ndOrderLongPeriodCorrections, LaraFirstOrderCorrections. For the resolution of the different Kepler equations because of the different coordinate systems, there are SimpleKeplerEq and TwoTermsKeplerEq.

All those types are mixed in the SGP4 algorithm subclasses, like SGP4Vallado in Fig. (1) and SGP4Lara in Fig. (2).

3. SGP4 ALGORITHM DESCRIPTIONS

The SGP4 propagator is thoroughly described in the literature [1, 2]. For completeness, we summarize it here, yet without mentioning anything related to the deep space or atmospheric drag corrections.

First, we recall that Brouwer's gravitational theory relies on the canonical set of Delaunay variables

- $\ell = M$: mean anomaly
- $g = \omega$: argument of the periapsis
- $h = \Omega$: RAAN (right ascension of the ascending node)
- $L = \sqrt{\mu\, a}$: Delaunay action, where μ is the earth gravitational constant and a is the orbit semimajor axis
- $G = \sqrt{\mu\, p}$: total angular momentum, where $p = a(1-e^2)$ is the orbit parameter (*semilatus rectum*), and e is the orbit eccentricity
- $H = G\cos I$: projection of the angular momentum on the earth's rotation axis, where I is the orbit inclination

However, Brouwer's theory finds trouble when evaluating the short-period corrections for the lower eccentricity orbits. The trouble is artificial, and is related to the singularity of Delaunay variables for circular orbits. Hence, SGP4 implements a different set of elements based on Lyddane's approach which completely avoids that trouble. In particular, the elements used in the computation of short-period corrections are[1]

$$F = \ell + g + h, \quad C = e\cos\omega, \quad S = e\sin\omega, \quad a, \quad I, \quad h.$$

3.1. Standard implementation of SGP4

3.1.1. Initialization and secular corrections

The code, starts from the initial mean elements at epoch $(I_0, \Omega_0, e_0, \omega_0, M_0, n_0)$, where n stands for mean motion, which should be obtained from the so called two line elements, or TLEs.[2] From them, Delaunay elements at epoch $(\ell''_0, g''_0, h''_0, L''_0, G''_0, H''_0)$ are obtained from their definition. The double prime notation is used for "secular" elements. In fact, the Delaunay actions, viz. L, G, and H, are never computed in SGP4, which uses $n = \mu^2/L^3$, $e = \sqrt{1 - G^2/L^2}$, and $I = \arccos(H/G)$, instead.

From these initial elements, Brouwer's theory provides the secular frequencies $\dot\ell''$, $\dot g''$, and $\dot h''$, due to the earth's gravitational potential, where the overdot means time derivative. Each of these frequencies is a function of (n_0, e_0, I_0), and remain constant for the evaluation of the theory at different dates. Besides, the initialization process provides a series of coefficients needed to apply drag secular corrections as computed from Lane's theory [4].

The code works with internal units of length LU (units of earth's radius $R_\oplus$ in km) and time TU (units of the orbit's period in min) such that $\mu = 1\,\mathrm{LU}^3/\mathrm{TU}^2$ in internal units. Besides, the expansion of the gravitational potential in SGP4 is limited to the (unnormalized) zonal harmonics J_2, J_3, and J_4. This selection of units helps in optimizing the code, but it is irrelevant for the description of the algorithms. Therefore, following descriptions include explicitly both the earth's gravitational parameter as well as the earth's equatorial radius.

The next step is to update the secular elements at epoch to the desired date given by the time t.

Brouwer's gravitational corrections are applied first

$$\ell''_j = M_0 + \dot\ell''\, t, \quad g''_j = \omega_0 + \dot g''\, t, \quad h''_j = \Omega_0 + \dot h''\, t,$$

where the subindex j stands for gravitational effects only. Besides, corrections due to the atmospheric drag, which are not discussed in the paper, are incorporated at this stage.

After that, we obtain the set of secular elements

$$(\ell'', g'', h'', a'', e'', I'').$$

The secular mean motion $n'' = (\mu/a''^3)^{1/2}$ is also computed.

3.1.2. Geopotential periodic corrections

Brouwer long-period gravitational corrections are reformulated in Lyddane's variables. At the precision of SGP4, that is, neglecting terms $\mathcal{O}(e^2)$, there are only corrections for F and S. Then, $C' = C''$, $a' = a''$,

[1] The original variables proposed by Lyddane were slightly different, namely a, F, $e\cos\ell$, $e\sin\ell$, $\sin\frac{1}{2}I\cos h$, and $\sin\frac{1}{2}I\sin h$.

[2] Keep in mind that the TLE mean motion needs to be converted first from Kozai's definition [16] to Brouwer's secular one.

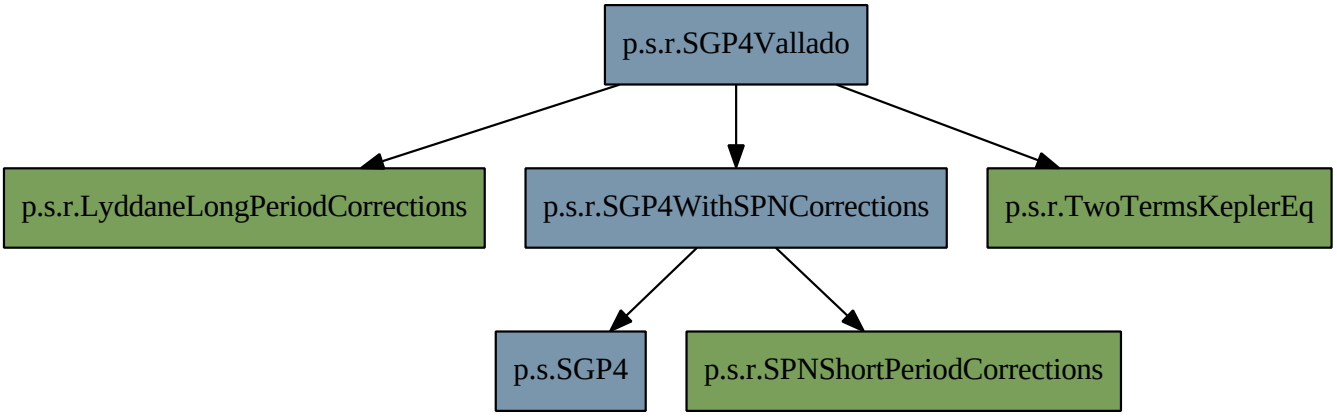


Fig. 1. Structure of SGP4Vallado algorithm

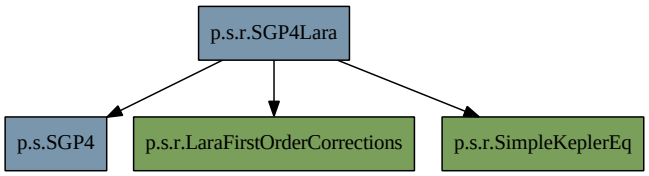


Fig. 2. Structure of SGP4Lara algorithm

$h' = h''$, $I' = I''$, and

$$\begin{aligned} F' &= (\ell'' + g'' + h'') - \frac{J_3}{4J_2}\frac{R_\oplus}{p''}C''\frac{3+5\cos I''}{1+\cos I''}\sin I'' \\ S' &= e''\sin g'' - \frac{J_3}{2J_2}\frac{R_\oplus}{p''}\sin I'' \end{aligned}$$

where $C'' = e''\cos g''$, $p'' = a''(1-e''^2)$. The single prime notation means that Lyddane elements include long-period effects. Note that $n' = n''$. Since there is no risk of ambiguity, the notation can be unified and simplified by removing primes, to read

$$\begin{aligned} F &= M + \omega + \Omega - \epsilon_3 se\cos\omega\frac{3+5c}{2(1+c)} \\ S &= e\sin\omega - \epsilon_3 s \end{aligned}$$

where $c = \cos I$, $s = \sin I$, and $\epsilon_3 = \frac{1}{2}(J_3/J_2)(R_\oplus/p)$.

It follows a change from Lyddane's to polar variables

$$(r, \theta, R = \dot{r}, \Theta = r^2\dot{\theta}) \longrightarrow (F, C, S, a).$$

To do that, first the Kepler equation

$$U \equiv F' - h' = \Psi + S'\cos\Psi - C'\sin\Psi,$$

is solved to compute $\Psi = E' + g'$, where E is the eccentric anomaly. Newton-Raphson iterations start from $\Psi_0 = U$.

Next g' and e' are solved from $C' = e'\cos g'$, $S' = e'\sin g'$. Then, compute $E' = \Psi - g'$, $L' = \sqrt{\mu a'}$, and $\beta' = \sqrt{1-e'^2}$. It follows the usual transformation

$$\begin{aligned} r' &= a'(1 - e'\cos E') && (1) \\ R' &= (L'/r')e'\sin E' && (2) \\ \Theta' &= L'\beta' && (3) \\ \theta' &= f' + g' && (4) \end{aligned}$$

where the true anomaly is obtained without ambiguity from

$$\sin f' = (a'/r')\beta'\sin E', \quad \cos f' = (a'/r')(\cos E' - e'). \tag{5}$$

Note that, SGP4 computes the correction to $r\dot{\theta} = \Theta/r$ instead of the correction to the polar variable Θ.

Then, compute $p' = a'\beta'^2$, $c' = \cos I'$, $s' = \sin I'$, and $\epsilon_2 = -\frac{1}{4}J_2(R_\oplus/p')^2$, to obtain the SGP4 short-period corrections

$$\begin{aligned} \delta r &= \epsilon_2\left[3\beta' r'(3c'^2 - 1) - p's'^2\cos 2\theta'\right] \\ \delta\theta &= \frac{1}{2}\epsilon_2(7c'^2 - 1)\sin 2\theta' \\ \delta\Omega &= -3\epsilon_2 c'\sin 2\theta' \\ \delta R &= 2\epsilon_2 p' n' s'^2\sin 2\theta' \\ \delta I &= -3\epsilon_2 c'\, s'\cos 2\theta' \\ \delta\frac{\Theta}{r} &= -\epsilon_2\frac{\Theta'}{p'}\left[2s'^2\cos 2\theta' + 3\left(3c'^2 - 1\right)\right] \end{aligned}$$

which neglect terms of $\mathcal{O}(e)$.

Finally, the state vector is computed from the standard transformation from Cartesian to polar-nodal variables

$$\begin{pmatrix} x & \dot{x} \\ y & \dot{y} \\ z & \dot{z} \end{pmatrix} = R_3(-\Omega)\,R_1(-I)\,R_3(-\theta)\begin{pmatrix} r & R \\ 0 & \Theta/r \\ 0 & 0 \end{pmatrix}$$

where R_1 and R_3 are the usual rotation matrices about the x and z axes, respectively

3.2. Lara's algorithm in non-singular variables

A straightforward alternative to the SGP4 implementation is to proceed in Lara's non-singular variables [9] both for the long- and short-period gravitational corrections. The modifications to the SGP4 algorithm start from Section 3.1.2 as follows.

Now, Kepler equation $\ell'' = E'' - e'' \sin E''$ must be solved first to compute the (secular) eccentric anomaly, and then the (secular) true anomaly

$$f'' = 2 \arctan \left(\sqrt{\frac{1+e''}{1-e''}} \tan \frac{E''}{2} \right).$$

Next, Delaunay variables are transformed to polar variables using the same equations as Eqs. (1)–(5) but now with double prime variables, which in turn are transformed to Lara's non-singular variables

$$\psi = \theta + \Omega, \quad \xi = \sin I \sin \theta, \quad \chi = \sin I \cos \theta, \quad r,\ R,\ \Theta.$$

Then, apply the long-period corrections

$$\begin{aligned}
\delta\psi &= \epsilon_3 \left[2\chi + (\kappa\chi - c\xi\sigma)/(1+c) \right], \\
\delta\xi &= \epsilon_3 \left[2\chi^2 + \kappa \left(1 - \xi^2\right) \right], \\
\delta\chi &= -\epsilon_3 \left[c^2\sigma + (2+\kappa)\xi\chi \right], \\
\delta r &= \epsilon_3 \, \xi \, p, \\
\delta R &= \epsilon_3 \left(1+\kappa\right) \chi \, \Theta / r, \\
\delta\Theta &= \epsilon_3 (\kappa\xi - \sigma\chi) \, \Theta,
\end{aligned}$$

where $c = \sqrt{1 - \xi^2 - \chi^2}$, $\kappa = -1 + p/r$, $\sigma = pR/\Theta$, and $p = \Theta^2/\mu$. Note that this corrections have no contributions of $\mathcal{O}(e^2)$ and, therefore, do not accept any simplification at the precision of SGP4. The notation has been simplified avoiding primes because there is no risk of confusion.

It follows the computation of the short-period corrections

$$\begin{aligned}
\Delta\psi &= -\epsilon_2 \frac{1+7c}{1+c} \, \xi \, \chi \\
\Delta\xi &= -\epsilon_2 \left(\chi^2 - 3c^2\right) \xi \\
\Delta\chi &= \epsilon_2 \left(\xi^2 - 3c^2\right) \chi \\
\Delta r &= 2\epsilon_2 r \left(\xi^2 - 2 + 5c^2\right) \\
\Delta R &= 4\epsilon_2 \left(\Theta/r\right) \xi \, \chi \\
\Delta\Theta &= 3\epsilon_2 \, \Theta (\xi^2 - \chi^2)
\end{aligned}$$

where $c = \cos I$, $s = \sin I$, and $\epsilon_2 = -\frac{1}{4}(R_\oplus/p)^2 J_2$, which have been simplifyed neglecting terms of $\mathcal{O}(e)$ according to SGP4 criteria.

Finally, the state vector is computed from

$$\begin{aligned}
x &= r \left(b \cos\psi + q \sin\psi\right) \\
y &= r \left(b \sin\psi - q \cos\psi\right) \\
z &= r \, \xi \\
X &= x \left(R/r\right) - \left(\Theta/r\right) \left(q \cos\psi + \tau \sin\psi\right) \\
Y &= y \left(R/r\right) - \left(\Theta/r\right) \left(q \sin\psi - \tau \cos\psi\right) \\
Z &= z \left(R/r\right) + \left(\Theta/r\right) \chi
\end{aligned}$$

where

$$b = 1 - \frac{\xi^2}{1+c}, \qquad \tau = 1 - \frac{\chi^2}{1+c}, \qquad q = \frac{\xi \, \chi}{1+c},$$

and $c = H/\Theta$. Remark that H is an integral of the zonal problem and, therefore, its value is always known from given initial conditions $H = H''_0$.

4. VALIDATION

The validation of the software has been done by comparing the results with those of Vallado's original C++ source with propagations up to 3 weeks. The C++ source has been slightly modified in order to save the cartesian coordinates, the unit vector, the polar nodals coordinates and the long period corrections into a file. The modified sources and output file results are available as well online at

`https://github.com/pleira/sgpvallado`

The outputs were collected for the Near Earth test cases detailed in Table 1.

The results for the SGP4Vallado algorithm in Scala matches the C++ results with neglible differences for all test propagations (maximum differences of the order of 3E-8).

When propagating satellites 22312 and 28350, the SGP4Vallado implementation went into error at an earlier time than the C++ version because of the negative value for the eccentricity, being the times 454.2028672 versus 474.2028672 for 22312 and 960 versus 1440 for 28350. For the satellite 28872, propagation with SGP4Vallado ends by epoch minute 50 as in Vallado's C++.

4.1. Test for eccentric anomaly

Lara's algorithm is done in non singular variables which are related to polar nodal variables. To transform from classical secular elements to those non-singular variables, the eccentric anomaly has to be calculated first, as described in Section 3.2. Vallado's algorithm does the calculation of the eccentric anomaly after the long period corrections have been applied using Lyddane's variables. We have checked that, as expected, the order in which

Table 1. Near Earth TLEs used for testing

Satellite	Comment
00005	TEME example satellite.
06251	Normal drag case, perigee 377.26 km is low, but above the threshold of 220 km for simplified equations.
22312	Very low perigee (86.98 km) that uses the branch perigee <98 km for the atmospheric fitting parameter.
28057	Normal drag case, low eccentricity (0.000 088 4), so certain drag terms are set to zero to avoid math errors.
28350	Low perigee (127.20 km) that uses the branch perigee <156 km for the atmospheric fitting parameter.
28872	Perigee 51 km, lost about 50 minutes from epoch, used to test error handling.
29141	Last stages of decay.
29238	Perigee 212.24 km, simplified drag branch (perigee <220 km) test.

the eccentric anomaly is calculated has negligible effect in the long period corrections.

Two algorithms have been developed to test this fact, SGP4PN and SGP4ValladoLong. SGP4PN does the calculation of long period corrections in polar nodal variables (SpecialPolarNodal in the code) and therefore requires to calculate Kepler equation before the long period corrections as in SGP4Lara. SGP4PN includes the same terms for long period corrections as in SGP4Lara, which include some more effects than those in SGP4Vallado. For the comparison, the SGP4ValladoLong algorithm includes the same long period effects expressed in Lyddane's variables, that is, without neglecting long-period corrections of $\mathcal{O}(e^2)$ and higher, and afterwards, solves the Kepler equation as in SGP4Vallado. The short period corrections for all SGP4Vallado, SGP4PN and SGP4ValladoLong share the same code expressed in polar nodals variables. The results of comparing the output in polar nodals variables in internal units after the periodic corrections were calculated between SGP4ValladoLong and SGP4PN for propagations up to 3 weeks gave maximum differences of the order 1.6E-6, which shows no significant effect introduced by the order of the calculation of the eccentric anomaly.

Figures (3) and (4) show the comparison of propagation results for the test TLE 00005 for 1 day with SGP4Vallado (which correspond to Vallado's C++ results) to the other three different algorithms. The inclusion of more terms in the long period corrections for the other algorithms produce differences. SGP4PN and SGP4ValladoLong come displayed one a top the other, which shows no effect by the order in the calculation of the eccentric anomaly as mentioned previously. Also, the differences between in the short period corrections between Lara and the rest of the algorithms account for the different behaviour of Lara's algorithm.

5. APPLICATIONS

SGP4Extensions produces the same results as other SGP4 implementations. Its interest resides in the flexibility to introduce and test new ideas related to SGP4 itself. As example when doing conjunction analysis for space debris, the user has the possibility to design filter algorithms based on results obtained in polar nodal variables without the need to obtain cartesian values. One possibility when comparing two objects is to find out the crossing points of the orbital planes and then estimate the passing times at those points with the true anomaly [17, sec. 9.6].

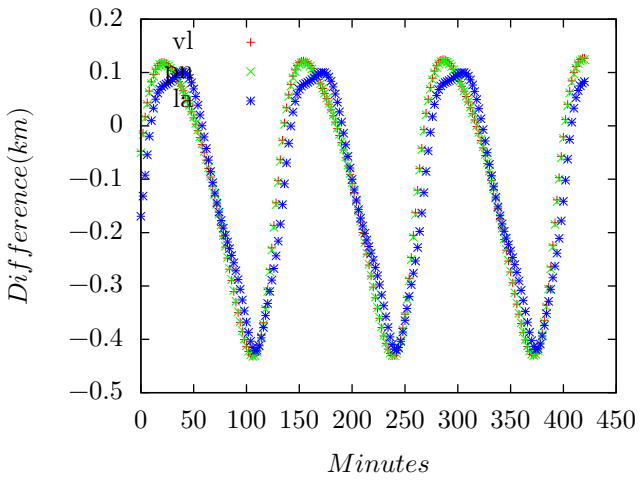
As concerns like errors and uncertainties have also to be taken into account when designing these algorithms, more special numeric types related to Spire's Intervals can be introduced to automatically calculate such uncertainties with the normal conjunction algorithm.

For other filter possibilities, the cartesian values can be easily obtained on demand, saving computational time and memory requirements.

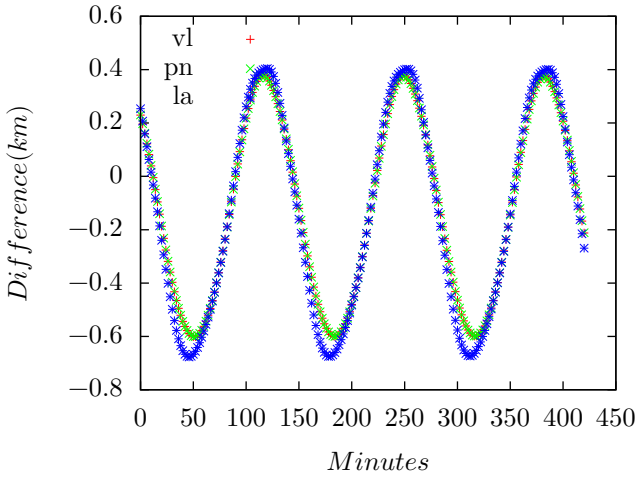
Regarding speed optimizations in the current SGP4 code, when doing conjunction analysis most of the computational time goes into finding the points of close approach. Therefore, speed optimizations at SGP4 level might not have a significant impact compared with the actual design of the conjunction analysis strategy.

6. CONCLUSIONS

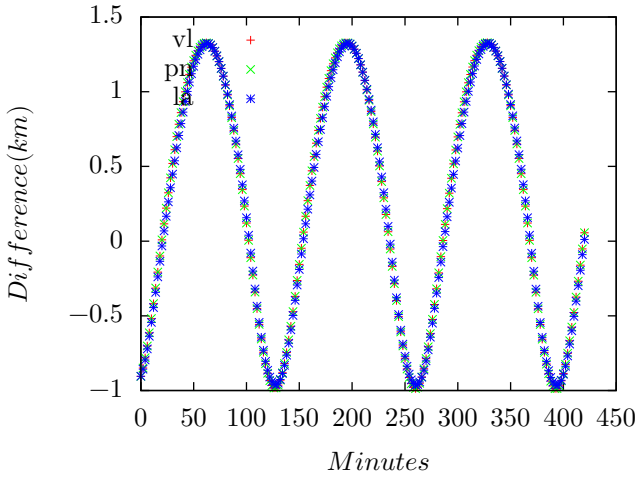
A new implementation of the SGP4 algorithm in scala is available for the community. It provides implementations of the models using a unicode notation resembling the actual descriptions of the models in the literature. It also allows to easily introduce alternative models and variables related to the theory. The software is designed for parallelization and for fine grain access to the data generated by the propagation steps. This provides an interesting base for massive conjunction analysis tasks.



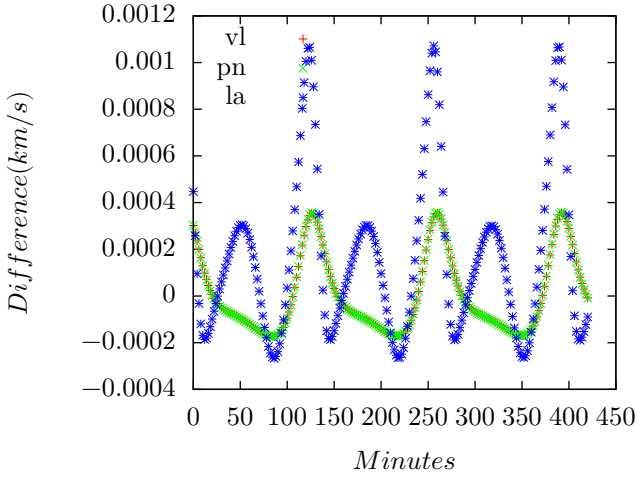
x coordinate differences to Vallado's SGP4



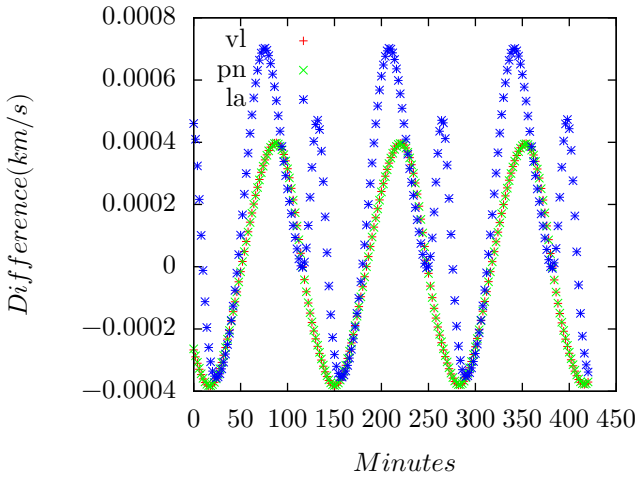
y coordinate differences to Vallado's SGP4



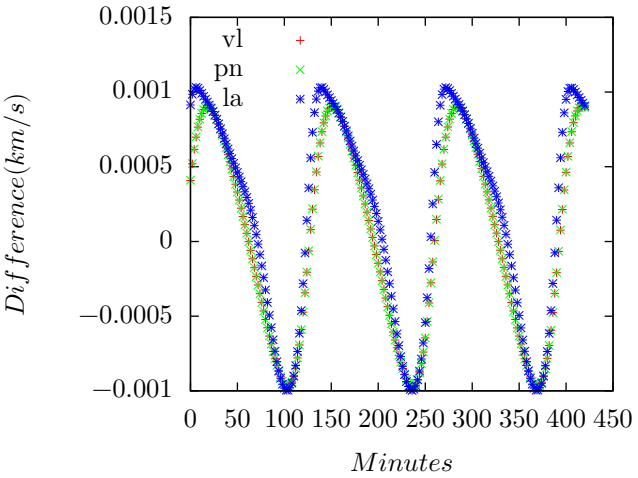
z coordinate differences to Vallado's SGP4



v_x coordinate differences to Vallado's SGP4

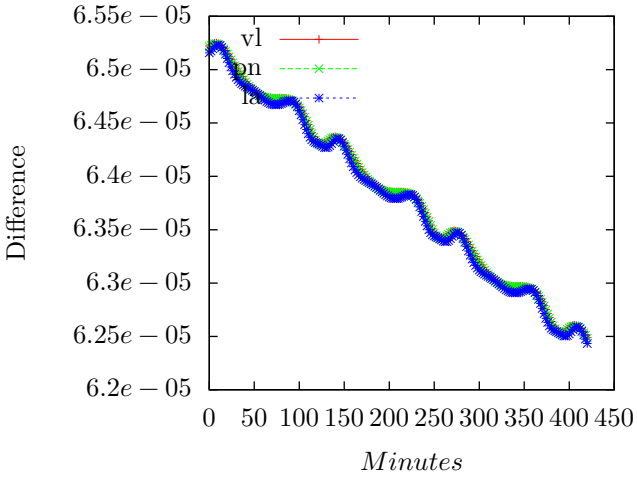


v_y coordinate differences to Vallado's SGP4

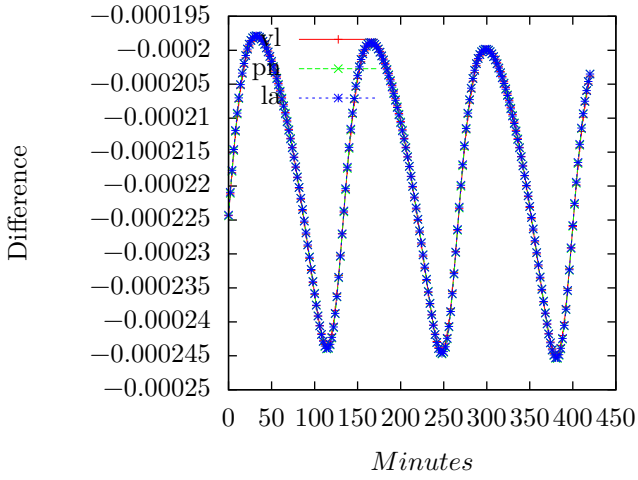


v_z coordinate differences to Vallado's SGP4

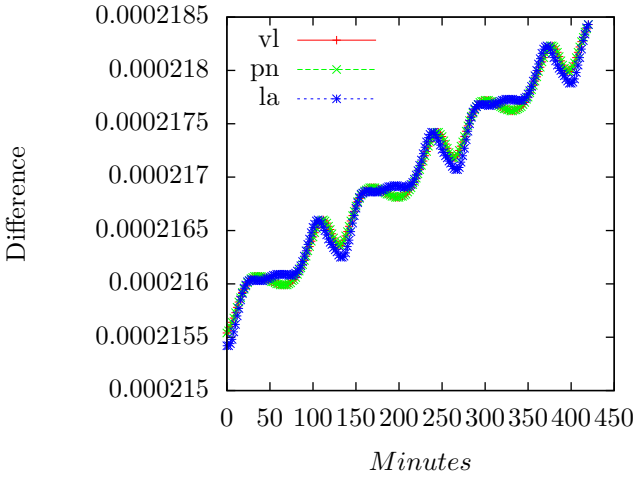
Fig. 3. Differences for the cartesian coordinates position (km) and velocity (km/s) of SGP4Extension algorithms Vallado Long, Polar Nodals and Lara with SGP4Vallado/Vallado's C++ results for TLE 00005 test case



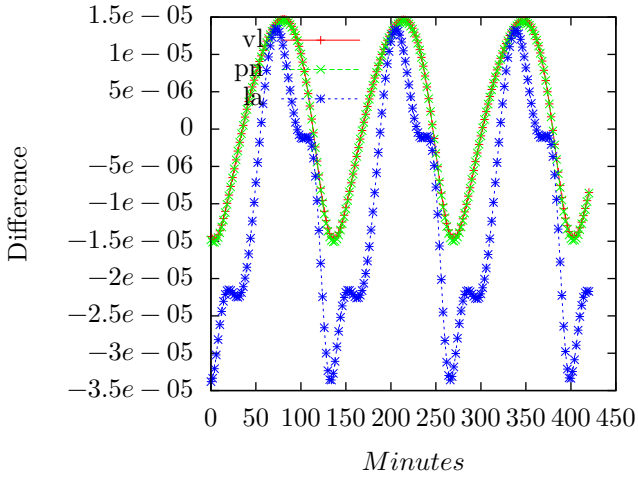
Inclination differences (radians) to Vallado's SGP4



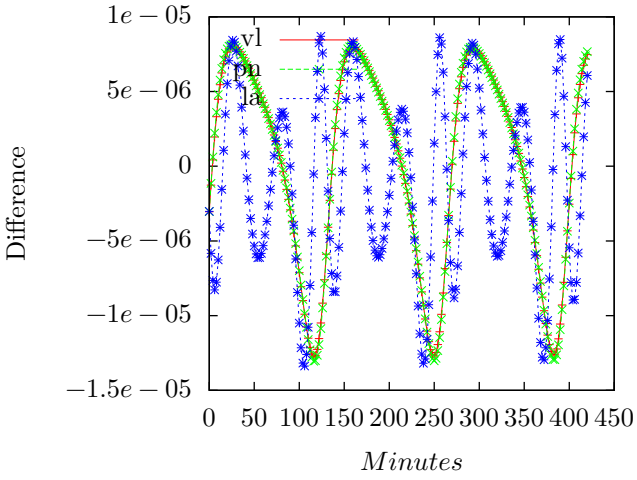
Argument of latitude diffs (radians) to Vallado's SGP4



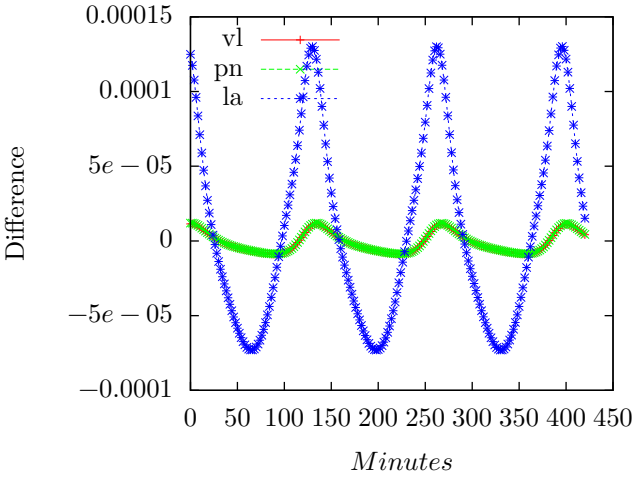
Ascending node differences (radians) to Vallado's SGP4



radial distance diffs (km) to Vallado's SGP4



radial velocity diffs (internal units) to Vallado's SGP4



angular momentum diffs (internal units) to Vallado's SGP4

Fig. 4. Differences for the Polar Nodal coordinates in internal units of the SGP4Extension algorithms Vallado Long, Polar Nodals and Lara with SGP4Vallado/Vallado's C++ results for TLE 00005 test case

7. REFERENCES

[1] F. R. Hoots and R. L. Roehrich, "Models for Propagation of the NORAD Element Sets," Project SPACETRACK, Rept. 3, U.S. Air Force Aerospace Defense Command, Colorado Springs, CO, December 1980.

[2] D. A. Vallado, P. Crawford, R. Hujsak, and T. S. Kelso, "Revisiting Spacetrack Report #3 (AIAA 2006-6753)," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, USA, August 2006, Guidance, Navigation, and Control and Co-located Conferences, pp. 1–88, American Institute of Aeronautics and Astronautics.

[3] D. Brouwer, "Solution of the problem of artificial satellite theory without drag," *The Astronomical Journal*, vol. 64, pp. 378–397, Nov. 1959.

[4] M. Lane, "The development of an artificial satellite theory using a power-law atmospheric density representation," in *Proceedings of the 2nd Aerospace Sciences Meeting*. Jan. 1965, pp. 1–16, American Institute of Aeronautics and Astronautics.

[5] R. H. Lyddane, "Small eccentricities or inclinations in the Brouwer theory of the artificial satellite," *Astronomical Journal*, vol. 68, no. 8, pp. 555–558, Oct. 1963.

[6] I. G. Izsak, "A note on perturbation theory," *The Astronomical Journal*, vol. 68, no. 8, pp. 559–561, Oct. 1963.

[7] K. Aksnes, "On the Use of the Hill Variables in Artificial Satellite Theory," *Astronomy and Astrophysics*, vol. 17, no. 1, pp. 70–75, Feb. 1972.

[8] F. R. Hoots, "Reformulation of the Brouwer geopotential theory for improved computational efficiency," *Celestial Mechanics*, vol. 24, no. 2, pp. 367–375, Aug. 1981.

[9] M. Lara, "Efficient Formulation of the Periodic Corrections in Brouwer's Gravity Solution," *Mathematical Problems in Engineering*, vol. 2015, no. Article ID 980652, pp. 1–9, 2015.

[10] Martin Odersky and al., "An overview of the scala programming language," Tech. Rep. IC/2004/64, EPFL Lausanne, Switzerland, 2004.

[11] Paul Chiusano and Rúnar Bjarnason, *Functional Programming in Scala*, Manning Publications, 1st edition, September 2014.

[12] Michael E. Cotterell, John A. Miller, and Tom Horton, "Unicode in domain-specific programming languages for modeling & simulation: Scalation as a case study," *CoRR*, vol. abs/1112.1751, 2011.

[13] Tom Switzer Erik Osheim, "Spire," `https://github.com/non/spire`, 2011.

[14] Bill Venners, "Scalactic," `https://github.com/scalatest/scalatest`, 2015.

[15] Bill Venners, "Scalatest," `https://github.com/scalatest/scalatest`, 2009.

[16] Y. Kozai, "The Motion of a Close Earth Satellite," *The Astronomical Journal*, vol. 64, no. 11, pp. 367–377, November 1959.

[17] Alberto Abad, *Astrodinámica*, Bubok Publishing S.L., 2012, Freely available at `http://www.bubok.es/libros/219952/Astrodinamica`.