DOI: http://dx.doi.org/10.18524/1810-4215.2018.31.145962

FEATURES OF KOTLIN ORBIT ESTIMATION LIBRARY

L.S.Shakun

Astronomical Observatory of Odessa I.I.Mechnikov National University,
Odessa, Ukraine, *leomspace@gmail.com*

ABSTRACT. Since the launch of Sputnik 1, the number of objects in near-earth orbit has been constantly increasing. The growth of number of these objects increases the risk of their collisions with existing satellites and ones which will be launched, that can be accompanied by their failure or even destruction. Most space agencies and many countries have their own space surveillance networks. These networks continually measure the positions of known objects, find new ones and predict their future positions. It is necessary to know the position of the objects with an accuracy of up to the characteristic size of the operating satellites (about 10 cm or more) to prevent collisions. Today, this task can be solved only for a small number of reference satellites and only for the past.

The calculation of the objects positions in near-Earth space requires the implementation of complex models of the Earth motion and a space object using many factors affecting the final result. Some of these factors, for example, the atmospheric density and the attitude of the satellite in the space, are not well predicted by modern models and require constant refinement from observations. Ukrainian Optical Facilities for Near-Earth Space Surveillance Network (UMOS) is used to surveillance and study near-Earth space in Ukraine.

There are many practical tasks that require knowledge of the positions of the space objects. The different software solutions are being applied to solve them. All of them must implement motion models of near-Earth space objects, the Earth and the main bodies of the Solar System for their needs. Space dynamics libraries are developed to implement these models. Orekit is one of these libraries. The Kotlin Orbit Estimation Library (KOrbEstLib) is built on the features of Orekit and extends them. KOrbEstLib expands the set of input and output data types, implementing the support of a number of Ukrainian and international data formats, in particular, the formats used in the UMOS network. In addition, KOrbEstLib offers an alternative implementation for estimating the parameters of the motion model of space objects in comparison with Orekit. This paper discusses a number of implementation features of the KOrbEstLib.

Keywords: artificial satellite, optical observation, space dynamic software, orbit estimation, Orekit.

АБСТРАКТ. З моменту запуску першого космічного супутника кількість об'єктів на навколоземній орбіті неухильно зростає. Збільшення кількості цих об'єктів підвищує ризик їх зіткнень з діючими супутниками і тими, що тільки виводяться, це може супроводжуватися виходом їх з ладу або навіть руйнуванням. Для запобігання зіткнень необхідно знати положення всіх великих космічних об'єктів. Більшість космічних агентств і багато країн мають свої мережі спостереження за космічним простором. Ці мережі постійно вимірюють положення відомих об'єктів, займаються пошуком нових і прогнозують їхнє майбутнє положення. Для запобігання зіткнень необхідно знати розташування об'єктів з точністю до характерних розмірів діючих супутників (від 10 см до декількох метрів). На сьогоднішній день така задача може бути вирішена тільки для невеликої кількості еталонних супутників і тільки для минулого часу.

Обчислення положень об'єктів в навколоземному просторі вимагає реалізації складних моделей руху Землі та космічного об'єкта з урахуванням великої кількості факторів, що впливають на кінцевий результат. Ряд цих факторів, наприклад густина атмосфери і орієнтація супутника в просторі, недостатньо добре передбачаються сучасними моделями і вимагають постійного уточнення з спостережень. Українська мережа оптичних станцій для дослідження навколоземного простору (УМОС) використовується для цілей контролю та його вивчення.

Існує безліч практичних завдань, що вимагають знання положення космічних об'єктів. Для їх вирішення застосовуються різноманітні програмні рішення. Всі вони повинні для своїх потреб реалізовувати моделі руху навколоземних космічних об'єктів, Землі і основних тіл Сонячної системи. Для реалізації цих моделей розробляються бібліотеки для космічної динаміки. Однією з таких бібліотек є Orekit. Kotlin Orbit Estimation Library (KOrbEstLib) спирається на можливості Orekit та розширює їх. KOrbEstLib впроваджує додатковий набір вхідних і вихідних типів даних, реалізуючи підтримку ряду українських і міжнародних форматів даних, зокрема форматів прийнятих в мережі УМОС. Крім цього, KOrbEstLib пропонує альтернативну в порівнянні з Orekit реалізацію для оцінки параметрів моделі руху космічних об'єктів. У роботі розглядаються ряд особливостей реалізації бібліотеки Kotlin Orbit Estimation Library.

Ключові слова: штучні спутники, оптичні спостереження, програмне забезпечення, визначення орбіт, Orekit.

1. Introduction

Since the launch of Sputnik 1 on October 4, 1957, the number of objects in near-earth orbit has been constantly increasing. The launch of artificial satellites of Earth and results of the destruction of objects in orbit is the main sources of these objects. Only 8% of cataloged objects are active satellites and the rest are space debris (ESA, 2018). According the NASA forecast the number of the objects will grow on in the future (Liou et al., 2004; 2018; NASA, 2006; 2014; Liou, 2010). Retrospective comparison of past forecast with the current object number shows that now the object number is growing faster than in optimistic forecasts (Radtke & Stoll, 2016). Thus the risk of collisions of the operating objects with the space debris that can lead their breakdown will increase. The most famous collision occurred on February 10, 2009, when the non-operational satellite Kosmos-2251 and the active satellite Iridium-33 collided. Today, an active satellite can avoid the collision only by performing a debris avoidance manoeuvre. It is necessary to foresee possible collisions for this. Many space agencies and countries create and maintain surveillance space networks for this purpose. The responsibilities of these networks include cataloging of objects and predicting the object positions in orbits.

The typical size of most of the currently operating satellites are in the range from 10 cm to 10 m for low orbits, and from 1 m to 10 m for high orbits (ESA, 2018). To predict a collision, it is necessary to be able to estimate the position of space objects (including space debris) in near-Earth space with an accuracy up to the characteristic sizes of operating satellites over an interval of several days. We need to perform observations and obtain measurements of the position of the satellite for this. There are many different methods to measure the position of a satellite. We usually need our own coordinate system in which measurements are made for each method. Many effects that cause a distortion in measurement results are conveniently described in their own coordinate systems. Simulation of satellite motion usually performed in one of the pseudo-inertial geocentric coordinate systems. In addition to the explicitly required coordinate systems, it usually makes sense to introduce a number of intermediate coordinate systems. As one can see, the process of measuring and modeling of satellite motion requires the determination of many different coordinate systems. Thus, we need

1. To provide the possibility of free determination of various coordinate systems and to provide transformations between them in accordance with modern recommendations for using Earth rotation information (for example IERS2010);
2. To develop the satellite motion model that propagates the satellite position over a period of several days with an accuracy 1 m or better.
3. To obtain a sufficient set of observations for estimation of the parameters of the satellite's motion model with an accuracy better than 1 m.
4. To develop algorithms for estimation of the model parameters based on the obtained observations with an accuracy better than 1 m.

At the present time, there are no open observations and software solutions that satisfy all of the listed requirements. Items from the first to the second related to the forward problem of the calculating the satellite position in any moment of time when the parameters of the motion model are known. Items 3 to 4 are related to the inverse problem. It is well known that the forward problems are easier to solve than inverse ones. Nevertheless, finding free software that solves only items 1-2 without any additional restrictions is a difficult task. It is even more difficult to find well-documented software with a free commercial use license. Many scientists develop their own software for this purpose or buy one.

We chose the Orbit Extrapolation Kit (Orekit) (Maisonobe et al., 2008), a low level space dynamics library written in Java. Orekit fully satisfies the requirements of item 1 and satisfies item 2 with a number of limitations. Orekit is open source software. It is distributed under the Apache License version 2.0, a well-known business-friendly license. This means anybody can use it to build any application, free or not.

Ukrainian Optical Facilities for Near-Earth Space Surveillance Network (UMOS – Ukrainian acronym) is used for purposes of space surveillance in Ukraine (Shulga et al., 2015). It is a voluntary association of academic observatories mainly. This network has been operating since 2011. It has obtained 14,401 tracks for 1,832 objects for all type of near-earth orbits from 2011 to 2017. This number of observations is certainly not enough to fulfill the requirements of paragraph 4. However, they are sufficient for the development and debugging of algorithms for estimating the parameters of the orbit of objects in near-earth space using optical observations.

2. Comparison of the implementation features of the measurements representation and optimization of orbit parameters in Orekit and KOrbEstLib

The Kotlin Orbit Estimation Library (KOrbEstLib) aims to provide access to all observations of the UMOS network and provide algorithms for estimating the parameters of the Orekit satellites motion models based on the optical observations. Figure 1 illustrates the KOrbEstLib architecture.

Orekit includes the ability to estimate the motion parameters of a space object based on observations beginning from version 8.0. KOrbEstLib introduces an alternative way to estimate the parameters of the satellite motion model.

In Orekit, any type of measurement is represented as an array of real numbers with an appropriate weight for each array element of measurement (tab. 1). A model estimation is a real array with size equal to the size of measurement array. The physical sense of array elements of estimation and measurement is the same. The code of the residuals is hidden from a user and is implemented as the weighted differences of measurements and their estimates. Optimization of the motion model parameters is minimizing the squares of the residuals. In fact, Orekit implements the least squares method, where the realization form of measurements defines the optimization form of the parameters of the satellite motion model. This approach provides the right implementation of the least squares method in the case of non-correlated measurements. However, it will not be the optimal solution in the case of correlated measurements or if we have some a priori information about the motion model of a space object.

KOrbEstLib offers a different measurement model. Each measurement type can have own form that describes measurement. The user can define this form by his prefere-

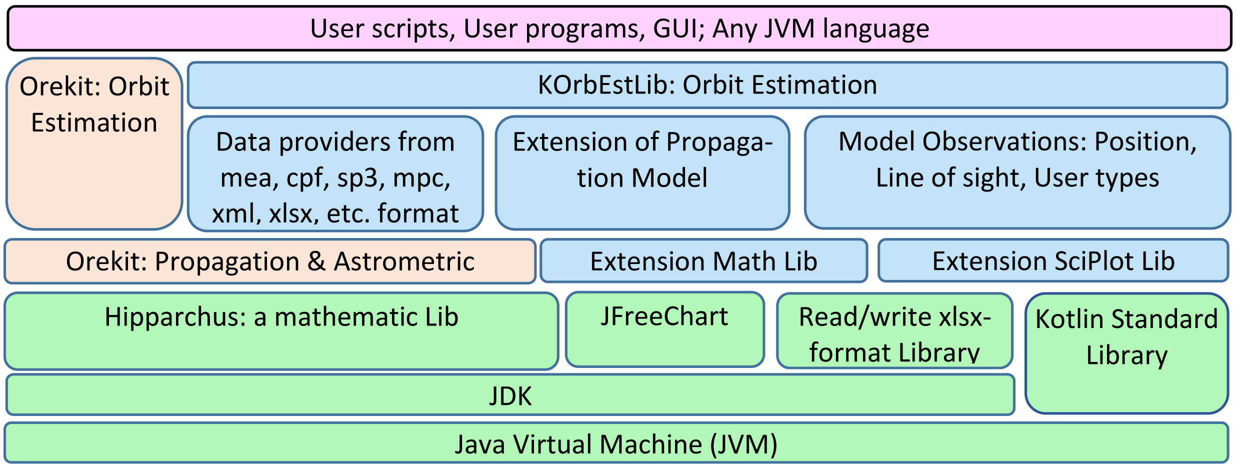
User scripts, User programs, GUI; Any JVM language
Orekit: Orbit Estimation
KOrbEstLib: Orbit Estimation
Data providers from mea, cpf, sp3, mpc, xml, xlsx, etc. format
Extension of Propagation Model
Model Observations: Position, Line of sight, User types
Orekit: Propagation & Astrometric
Extension Math Lib
Extension SciPlot Lib
Hipparchus: a mathematic Lib
JFreeChart
Read/write xlsx-format Library
Kotlin Standard Library
JDK
Java Virtual Machine (JVM)

Figure 1: the architecture of the Kotlin Orbit Estimation Library. Blue is the KOrbEstLib components; brown is Orekit components; green is free libraries and JVM; pink is user capabilities to develop their own software using KOrbEstLib.

Table 1: the comparison of the most important differences in the approaches for estimating the parameters of the motion model.

Feature	Orekit	KOrbEstLib
measurement	public $x_1, x_2, \dots, x_n$	user defined
weights	public $w_1, w_2, \dots, w_n$ the same size that in the measurement	user defined
estimation	$x'_1, x'_2, \dots, x'_n$ the same size that in the measurement	user defined
residuals	private $w_1 \cdot (x_1 - x'_1)$, $w_2 \cdot (x_2 - x'_2)$, …, $w_n \cdot (x_n - x'_n)$ the same size that in the measurement	required interface; user defined
Optimized function	private $f = \sum_{i=1,n} r_i^2$ r_i – residuals	parameter user defined

nce. The estimation type that corresponds to measurement type can have the form that describes estimation, not same with measurement form. The only one requirement that KOrbEstLib imposes is the ability to calculate the residuals between the measurement and its estimation. Moreover, this requirement is important only when we optimizing the model for an observations set consisted of several measurement types (for example, optical and radio observations). The free form of the optimize function allows defining the problems wider than least squares problem.

In order to illustrate the difficulties that arise when the measurements implemented in the Orekit form, let us consider ordinary angular measurements that obtained in optical observations. Usually, such measurements are represented

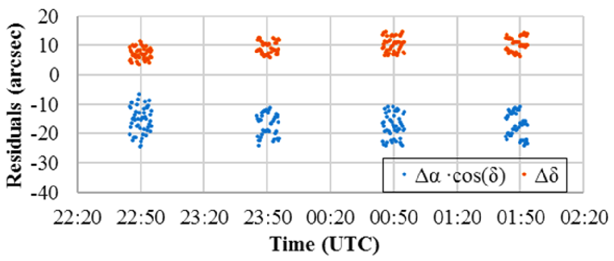

Figure 2: the residuals between observations and estimations in right ascension and declination.

in the form of angle coordinate pair (α – right ascension and δ – declination) of an object in the star catalog frame. The residuals between the model and the measurements are represented as $\Delta\alpha \cdot \cos\delta$ and $\Delta\delta$ respectively. Already at this stage, we are enforced to correct the nonlinearity of right ascension via using the weighting factor cosδ. This approach has application limits. $\Delta\alpha \cdot \cos\delta$ incorrectly describes the residuals in right ascension since nonlinear terms in Δα cannot be neglected in the polar regions. Since that the magnitude of residuals is usually very small, of the order of the arcseconds, such the polar region is small and the researchers simply avoid use measurements in it. Figure 2 shows an example of the residuals in terms of $\Delta\alpha \cdot \cos\delta$ и Δδ. Considering Figure 2, it is easy to make the following conclusion. Random errors for each of the coordinates measurements are approximately equal. But the cause of the observed systematic bias is difficult to determine immediately.

One of the main features of satellites optical observations is related to the high mobility of satellites compared with stars. Important, this property is characteristic of satellites at all altitudes. If this is obvious for objects on low orbits, then for high objects it is necessary to take note the exposure time for the image. It reaches several seconds and the satellite displacement during this time is much larger than satellite size on the image. Hence such objects are highly mobile. Thus, regardless of the orbit type of a near-Earth space object in optical measurements, there is a special direction along the visible direction of satellite motion.

Figure 3 shows the same differences between measurements and their estimates as in Figure 2, but already along and across the visible satellite track. Such representation of residuals becomes possible only after calculating the model satellite positions and velocities, and, therefore, does not agree with the representation form of measurements in Orekit. It should be noted that Orekit allows defining the similar measurement type if the measurement itself encapsulates the a priori information about the satellite orbit. But this is not always achievable. KOrbEstLib does not impose any restrictions on the software implementation of the measurements and does not require the correspondences of the measurements and estimations implementation. Hence, there are no obstacles to implement measurements in the form of right ascension and declination and the minimize objective function in the least square problem in the form of the differences along and across the visible satellite track. Thus, the form of the measurement implementation in KOrbEstLib can be easily matched with the representation of the measurement residuals along and across the trajectory. The residuals along and across the visible trajectory have another useful property, the nonlinear terms are small at any point of the celestial sphere, and all the residuals do not require renormalization from measurement to measurement. We do not need to avoid near-polar regions as in the case of the residuals in right ascension and declination.

In Figure 3, we clearly see that the systematic differences between measurements and estimates are related with the direction along visible satellite track. Moreover, the random error of measurement is significantly larger in the direction along track than across it. In the overwhelming number of cases and in the example we are considering, this is due to the errors in the moment registration of measurement. Such errors are usually hardly revealed since they cannot be detected within the scheme of the measurements themselves. It is necessary to compare observations with a priori high-precision orbits that to reveal them, which is not always easy to perform at the site of observation. Physically correctly choice of directions of decomposition of residuals allow to introduce weights for measurements easily, and often more correctly.

The values that describe the measurement can be implemented in KOrbEstLib "measurement" class as private. The consequence of this is the possibility of implementation of immutable objects of measurement, that under certain conditions allows you to write more high-performance code.

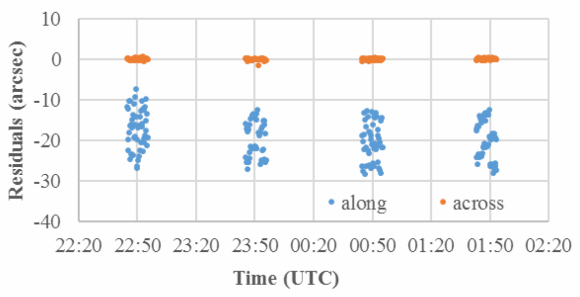

Figure 3: the residuals between observations and estimations in the along and across directions relatively the visible satellite track.

3. Additional features for support input and output data formats in KOrbEstLib

KOrbEstLib extends the set of ephemeris formats that supported Orekit by adding consolidated prediction format (CPF). KOrbEstLib allows you to combine several ephemeris with time gaps between them, which allows you just to make comparisons of observations and ephemeris for the large time intervals when there are difficulties with obtaining continuous ephemeris.

KOrbEstLib supports reading observations from the minor planet center (MPC) format and reading and writing in the MEA format. The MEA format is used to exchange observations by members of the UMOS network. The MPC format was designed to represent astronomical observations, where the duration of exposure is usually in seconds, and the measurement accuracy of the observation time moment rarely exceeds 0.1 seconds. The MPC format does not allow to specify moments in time with an accuracy of more than 0.001 seconds. The measurement accuracy of the time moments should be not worse than 0.0001 sec when measuring of the coordinates of low-orbit objects with an accuracy of the order of 0.1 arcseconds. The time moments of the measurements have to be aligned to the moments of the seconds beginning, to overcome complications in representing the time in the MPC format. The MEA-format avoids these complexities by allowing you to transfer moments with an accuracy of 0.0001 sec and can be extended in a compatible way to support greater accuracy in the time presentation.

KOrbEstLib supports the ability to describe models of satellite motion in the XML-format, which simplifies the writing of scripts for processing observations. The results of the calculation of the orbits or residuals can be saved in Excel files, which simplifies their further analysis.

4. Conclusion

KOrbEstLib extends the Orekit. All the features of the source library are available for use.

KOrbEstLib is written in Kotlin language, which is very well compatible with the Java language. This makes it easy to use KOrbEstLib and Orekit everywhere and at the same time, where JVM compatible languages can be used.

KOrbEstLib offers a more flexible model for the implementing measurements and the objective functions, which makes it possible to expand the orbit optimization scenarios. KOrbEstLib was used to processing of real observations in paper Shakun L., 2017, where the general considerations from this article were applied in practice.

KOrbEstLib supports international data exchange formats and formats used in the UMOS network. It provides data conversion tools. These features make it quite simple to use KOrbEstLib and Orekit to analyze the observations of the UMOS network.

References

ESA's Annual Space Environment Report, [online] Available at: (https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf) [Accessed 10 October 2018].

Liou J.-C., Hall D., Krisko P. et al.: 2004, *Adv. Sp. Res.*, **34(5)**, 981.

Liou J.-C.: 2010, *Orbital Debris Quarterly News*, **14(1)**, 7 [online] Available at: (https://orbitaldebris.jsc.nasa.gov/quarterly-news/pdfs/odqnv14i1.pdf)

Liou J.-C., Matney M., a. Vavrin A. et al.: 2018, *Orbital Debris Quarterly News*, **22(3)**, [online] Available at: (https://orbitaldebris.jsc.nasa.gov/quarterly-news/pdfs/odqnv22i3.pdf.)

Maisonobe, L., Pommier V., Parraud P.: 2008, Orekit, An accurate and efficient core layer for space flight dynamics applications, Available at: (https://orekit.org/).

NASA:, 2006, 10(2), 1 Orbital Debris Quarterly News, [online] Available at: (https://orbitaldebris.jsc.nasa.gov/quarterly-news/pdfs/odqnv10i2.pdf).

NASA: 2010, Orbital Debris Quarterly News, **14(1)**, 1, [online] Available at: (https://orbitaldebris.jsc.nasa.gov/quarterly-news/pdfs/odqnv14i1.pdf)

Radtke J., Stoll E.: 2016, *Acta Astroautica1*, **127**, 482.

Shakun, L., Koshkin, N., Korobeynikova E. et al.: 2017, *Odessa Astron. Publ.*, **30**, 242.

Shulga A., Kravchuk, S., Sybiryakova Ye. et al.: 2015, *KNIT*, **21(3)**, 74.