ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 53–65*

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК: 004.428

БИБЛИОТЕКА KIAM ASTRODYNAMICS TOOLBOX ДЛЯ ПРОЕКТИРОВАНИЯ ОРБИТАЛЬНОГО ДВИЖЕНИЯ КОСМИЧЕСКИХ АППАРАТОВ

М. Г. Широбоков[a, *] (ORCID: 0000-0002-1747-6430),
С. П. Трофимов[a, **] (ORCID: 0000-0002-2850-5292)

[a]*Институт прикладной математики им. М.В. Келдыша РАН,*
125047 Москва, Миусская пл., д. 4, Россия

**E-mail: shirobokov@keldysh.ru*
***E-mail: trofimov@keldysh.ru*

Поступила в редакцию 16.02.2023
После доработки: 15.04.2023
Принята к публикации: 05.05.2023

Представлен обзор новой программной библиотеки для проектирования и моделирования орбитального движения космического аппарата KIAM Astrodynamics Toolbox. Библиотека создана в Институте прикладной математики им. М.В. Келдыша РАН на языках Fortran и Python и, таким образом, сочетает в себе скорость и гибкость. Библиотека будет полезна специалистам в области механики космического полета, а также обучающимся по соответствующим программам подготовки.

Ключевые слова: библиотека программного обеспечения, траектория, космический аппарат, межпланетный полет, Python, Fortran

DOI: 10.31857/S0132347424010058 **EDN:** HKPXDM

1. ВВЕДЕНИЕ

Проектирование и оптимизация космических траекторий – краеугольный камень в процессе разработки космических миссий. Соответствующие задачи, решаемые специалистами по механике космического полета (астродинамике), разнообразны: поиск оптимальных по затратам топлива или времени полета траекторий с учетом ограничений, исследование траекторий на чувствительность к начальным условиям и параметрам аппарата, исследование динамики вблизи небесных тел и условий радиовидимости аппарата с наземных станций наблюдения, расчет накопленной дозы ионизирующего излучения и продолжительности движения в тени небесных тел. Решение этих задач выполняется, как правило, численными методами и требует быстрых и удобных способов моделирования управляемого и неуправляемого движения космических аппаратов.

Существует множество программных комплексов, помогающих решать задачи астродинамики. Среди тех, что пользуются большим интересом, можно назвать продукт Systems Tool Kit (STK) компании Analytical Graphics [23]. Эта программа первоначально была создана для анализа спутниковых орбит и называлась Satellite Tool Kit; сейчас эта программа предоставляет интегрированную среду для анализа сложных космических, воздушных и морских систем для аэрокосмических и оборонных приложений. STK имеет графический интерфейс и позволяет производить анализ и выбор орбит, оптимизировать траектории перелета, моделировать стыковку аппаратов и учитывать внешние возмущения. Сходными возможностями для решения астродинамических задач обладает комплекс FreeFlyer компании a.i. solutions [24]. Кроме стандартных возможностей этот комплекс позволяет рассчитывать оптимальные маневры перехода между орбитами, причем как для импульсной, так и для непрерывной модели управления. Еще один популярный программный комплекс, General Mission Analysis Tool (GMAT), разрабатываемый специалистами космического агентства NASA, служит для проектирования реальных космических миссий в околоземном пространстве и дальнем космосе [25]. Это свободное программное обеспечение (ПО) с открытым исходным кодом представляет собой отдельную программу с интерфейсами для популярных языков программирования, таких как Python и MATLAB. Программа позволяет учесть многочисленные технические особенности реализации миссий, например, моделировать навигационные системы и связь. Одной из проблем при использовании данной программы является отсутствие возможностей автоматизации моделирования (кодирование происходит на собственном ограниченном языке сценариев). Другие примеры включают High-Precision Orbit

Propagator (HPOP) компании Microcosm, SatTrack Suite компании Bester, Aero/Astro Vehicle Control компании Princeton Satellite Systems, но у них меньше возможностей, чем у вышеназванных крупных программных комплексов. В Институте прикладной математики им. М.В. Келдыша РАН тоже был создан программный комплекс для моделирования вращательного и орбитального движения космических аппаратов [26]. Комплекс позволяет решать большое разнообразие задач, содержит модели внешней среды, модели датчиков и актюаторов, модель работы бортового компьютера и разнообразные алгоритмы управления движением. Комплекс заточен на задачи вращательного и околоземного орбитального движения и, как и продукты выше, представляет собой отдельное ПО со своим интерфейсом.

К известным библиотекам методов проектирования движения космического относится библиотека Fortran Astrodynamics Toolkit [27], разработанная Джейкобом Уильямсом (Jacob Williams) на современной версии языка Fortran. Библиотека содержит средства для решения задачи Ламберта (двухточечной краевой задачи в рамках динамики двух тел), методы интегрирования уравнений движения, функции для моделирования гравитационного поля Земли, обращения к эфемеридам небесных тел, преобразования орбитальных элементов и построения гало-орбит (периодических орбит вокруг точек либрации в модели круговой ограниченной задачи трех тел). Библиотека является открытой и свободной для использования. Отметим, что библиотека имеет существенно ограниченные функциональные возможности для того, чтобы можно было ее применять на стадиях предварительного анализа миссии. Так, она не содержит систем координат, связанных с Землей и Луной, функций для расчета сложного поля Луны и некоторых часто встречающихся в анализе миссий возмущений (например, давления солнечного излучения, силы сопротивления атмосферы). К сожалению, проект сейчас не развивается.

Библиотека pykep – это открытая программная библиотека, разработанная Европейским космическим агентством для исследований в области астродинамики и решения в первую очередь задач орбитального маневрирования [28]. Основа библиотеки написана на C++ и доступна для Python. В основе библиотеки лежат реализация эффективного решателя многооборотной задачи Ламберта, инструменты для решения задач оптимизации с малой тягой, эффективные кеплеровские пропагаторы, интеграторы Тейлора и многое другое. Выпуски новых версий pykep происходят один или два раза в год. Из недостатков можно отметить отсутствие преобразований между полезными для прикладных исследований систем координат, отсутствие возможности интегрирования траекторий с возмущениями, а также технические проблемы – необходимость сборки пакета на своей машине, отсутствие документированной установки сторонних пакетов и самостоятельную настройку Python–C++ интерфейса. Распространены и нерешенные проблемы с установкой пакетов из репозитория conda-forge [29].

Poliastro – еще одна открытая библиотека методов на языке Python для моделирования движения в орбитальной механике [30]. Библиотека сфокусирована на быстром и удобном рисовании орбит и позволяет аналитически и численно интегрировать дифференциальные уравнения движения космических аппаратов, выполнять преобразования между переменными, системами координат, рассчитывать маневры между орбитами, работать с эфемеридами. Этот проект может быть полезен для обучающихся астродинамике, содержит интересные объектно-ориентированные инструменты для моделирования орбит и решения задач маневрирования, однако библиотека недостаточно развита для ее использования в реальных приложениях: весь код написан на высокоуровневом языке Python, что делает его исполнение медленным в высоконагруженных расчетах, а интегрирование орбит даже с часто встречающимися возмущениями подразумевает их ручную реализацию.

SPICE Toolkit – крупная программная библиотека астродинамических методов, созданная в NASA для помощи в планировании и интерпретации научных наблюдений с использованием космических приборов, а также для моделирования, планирования и выполнения мероприятий в рамках миссий по исследованию планет [31, 32]. Методы в рамках этой библиотеки позволяют рассчитывать положения, скорости, размеры, форму и ориентацию планет, спутников, комет и астероидов, а также ориентацию аппарата и его составных частей. Библиотека также позволяет осуществлять временную привязку различных событий, например, определять интервалы времени пребывания спутника в тени планеты или на заданных высотах над небесным телом. Методы SPICE в своей работе обращаются к заранее подготовленным осведомленными источниками первичным наборам данных (“kernels”), которые содержат информацию о небесных телах, системах координат, научных приборах, космических аппаратах, их физические и геометрические параметры. Библиотека была изначально создана на языке

FORTRAN 77, но теперь доступна также на языках C, MATLAB, Java, Python, Julia. К недостаткам SPICE Toolkit можно отнести отсутствие моделей углового и орбитального движения аппарата, средств интегрирования уравнений как пассивного, так и управляемого движения. Библиотека представляет собой набор низкоуровневых подпрограмм; их использование в реальных проектах так или иначе подразумевает создание пользователем вспомогательных высокоуровневых интерфейсов.

Для многих исследователей и специалистов по астродинамике было бы удобно иметь единую поддерживаемую программную библиотеку методов моделирования и проектирования движения аппарата, которая, с одной стороны, была бы написана на распространенном высокоуровневом языке программирования, а с другой стороны, работала быстро. Такая библиотека не только должна иметь астрофизические и геометрические постоянные небесных тел, позволять выполнять преобразования между системами координат, переменными и времени, но также содержать разнообразные настраиваемые модели движения аппарата, возможности проектирования и оптимизации траекторий в выбранных моделях. Указанные выше программные библиотеки полностью таким требованиям не удовлетворяют.

Сейчас стали актуальны и процессы импортозамещения ПО отечественными разработками. Особенную важность появление таких свободно распространяемых библиотек имеет и в процессе обучения новых специалистов. Студенты, обучающиеся по соответствующим программам подготовки, зачастую получают от своих научных руководителей научно-исследовательские задачи, решение которых предполагает пользование средствами моделирования движения космического аппарата. В результате они вынуждены затрачивать время на написание вспомогательных подпрограмм (преобразование переменных, переход между системами координат, решение уравнений в вариациях и пр.) вместо того, чтобы сосредотачиваться на исходной астродинамической задаче. Студенты могут сравнивать собственные реализации вспомогательных программ с теми, что есть в библиотеке.

Данная статья посвящена описанию разработанной в Институте прикладной математики им. М.В. Келдыша РАН программной библиотеки численных методов KIAM Astrodynamics Toolbox для моделирования орбитального движения космического аппарата. Библиотека изначально была написана в среде MATLAB [33, 34, 35] и на данный момент полностью переписана на языках Fortran и Python. На Fortran реализованы часто используемые в механике космического полета преобразования и численные методы (преобразования систем координат, функции правых частей дифференциальных уравнений, численные методы интегрирования и др.). На языке Python реализованы интерфейсы для запуска скомпилированных на Fortran функций и высокоуровневый класс Trajectory для работы с объектами, моделирующими траектории космических аппаратов.

Предлагаемая библиотека методов отличается от существующих аналогов тем, что содержит как низкоуровневые часто используемые операции перевода между системами координат, переменных и времени, астрофизические и геометрические параметры небесных тел, так и широко применяемые модели движения, средства распространения траекторий как пассивных, так и с управлением, уравнения в вариациях, матрицы Якоби преобразований, а также высокоуровневые средства проектирования траекторий с автоматическим преобразованием между системами координат, переменными и системами единиц. Библиотека не является надстройкой над сторонними библиотеками, часть из которых не поддерживается и/или имеет собственные интерфейсы, а является единой системой, содержащей наиболее часто используемые функции для проведения исследований орбитальной динамики. Таким образом, библиотека собирает в себе преимущества других отдельно взятых библиотек. Уникальным в библиотеке является автоматический расчет последовательности преобразований систем координат, переменных и систем единиц на основе графов, а также возможность задавать произвольную функцию управления на языке Python, в то время как траектория будет проинтегрирована на Fortran.

В главе 2 приводится общее видение проекта разработки библиотеки и его особенности. В главе 3 приводится Fortran/Python-структура библиотеки, ее основные компоненты и способы компиляции для работы. В главе 4 перечислены основные функциональные возможности библиотеки. Глава 5 резюмирует все сказанное.

2. ВИДЕНИЕ И ОСОБЕННОСТИ ПРОЕКТА

Предполагаемыми пользователями библиотеки являются специалисты в области механики космического полета, проводящие исследования динамики и схем перелетов, решающие свои астродинамические задачи, а также студенты, обучающиеся астродинамике. Во время решения астродинамических задач библиотека может играть вспомогательную

роль, например, через реализацию множества регулярно используемых действий с массивами фазовых состояний космического аппарата и помощь в ответах на часто встречающиеся вопросы о динамике аппарата. Таким образом, пользователи программируют решения своих астродинамических задач, но для удобства могут использовать готовые реализованные средства библиотеки.

На наш взгляд, такое ПО должно быть открытым, так как открытое ПО обладает полезными для разрабатываемой библиотеки преимуществами:

1. Проверяемость. Открытое ПО, исходный код которого открыт, может быть проверено любым человеком, в том числе специалистами из сторонних заинтересованных организаций со своими методами исследования и решения задач, вследствие чего ошибки могут быть обнаружены на ранней стадии проектирования.

2. Привлечение разработчиков для развития проекта. Участие в открытых проектах проще для всех желающих. Современные веб-сервисы помогают создавать ветви разработки и предоставляют удобные инструменты для контроля версий проекта. Это особенно важно для развития предлагаемой библиотеки, так как множество используемых специалистами методов велико и требуются разработчики, в том числе из сторонних организаций, которые способны сделать свой вклад в проект. Кроме того, в разработке ПО сопровождение зачастую является длительным процессом и занимает намного больше времени, чем программирование перед первым выпуском, и у авторов, как ученых в своей предметной области, может не хватать времени на сопровождение. Вместе с тем у них появляются ученики, решающие задачи в своих направлениях, которым было бы полезно добавлять в код новые функциональные возможности и совершенствовать старые.

3. Распространение. Открытое ПО значительно проще распространять, что удобно, так как эта библиотека может быть использована при исполнении контрактов с индустриальными партнерами. Обратная связь благоприятно сказывается на развитии проекта.

4. Обучение. При обучении механике космического полета студенты иногда сталкиваются с проблемами входа в специальность и начала участия в крупных проектах, подразумевающих проектирование траекторий. Наибольшие трудности при этом приходятся на технические (смена систем координат, преобразование переменных, расчет матрицы монодромии и т.п.), а не содержательные аспекты работы. Существование библиотеки численных методов проектирования траекторий облегчает освоение специальности.

Отметим и некоторые недостатки открытого ПО и возможности им противостоять:

1. Опасность для бизнеса. Размещение в открытом доступе исходных кодов проекта может повлечь утечку идей в сторонние организации, имеющие конфликт интересов с организацией, в которой разрабатывается проект. Для того чтобы этого избежать, исходные коды для решения конкретных астродинамических задач не размещаются в открытом доступе и не добавляются в библиотеку. Некоторые исходные коды для решения общих задач и общематематических функций могут содержаться в библиотеке, но быть при этом закрытыми. Наконец, некоторые коды могут быть временно закрыты и открыты спустя некоторое время, когда острая актуальность в них пропадет.

2. Разветвление плохих проектов. На основе изначально размещенного проекта могут быть созданы ветви, которые станут популярными, но которые обладают неочевидными недостатками и ненадежностью. Эту проблему можно частично решить размещением в открытом доступе информации о тех ветвях проекта, которые, с точки зрения авторов начального проекта, являются надежными.

3. Нерегулярность выпусков. Версии открытого ПО, как правило, выходят нерегулярно, но эту проблему можно контролировать, во всяком случае для главной ветви проекта.

В связи с этим было принято решение сделать проект открытым и для удобства контроля версий и ветвей изменений разместить его на веб-сервисе GitHub [36]. На указанном сайте можно найти короткую справку о проекте, скачать библиотеку или присоединиться к ее разработке. Библиотека распространяется в рамках лицензии MIT License [37].

Еще одной особенностью проекта является сочетание скорости и гибкости, что достигается за счет использования компилируемого (Fortran) и популярного сейчас интерпретируемого (Python) языков программирования. На Fortran реализован почти весь функционал, связанный с проведением расчетов: интегрирование уравнений движения, расчет возмущающих ускорений, преобразование массивов переменных, систем координат. На Python реализованы интерфейсы для обращения к скомпилированным на Fortran функциям, а также высокоуровневые классы, например класс Trajectory для проектирования траекторий. Запуск расчетов на компилируемом языке делает их гораздо более быст-

рыми, чем если бы они были реализованы на интерпретируемом языке. Вместе с тем код на Python является гибким, немногословным и простым для освоения.

3. СТРУКТУРА БИБЛИОТЕКИ

Библиотека состоит из модулей на языке Fortran с реализованными астродинамическими функциями и модулей на языке Python, которые представляют собой интерфейсы для обращения к скомпилированным на Fortran функциям и высокоуровневые классы для проектирования траекторий.

Перечислим модули на языке Fortran:

1. **ConstantsAndUnits.f90** содержит астрономические постоянные согласно стандарту Международного астрономического союза (International Astronomical Union, IAU) 2009/2012. Сюда входят гравитационные параметры и размеры небесных тел Солнечной системы и подпрограммы, которые по названию небесного тела (или пары небесных тел, если речь идет о задаче трех тел) выдают системы согласованных единиц расстояния, скорости, времени и ускорения. Перейти к таким единицам бывает полезно, так как тогда уравнения движения записываются в более простом безразмерном виде.

2. **LinearAlgebraInterfaces.f90** содержит интерфейсы к подпрограммам LAPACK, осуществляющим операции линейной алгебры (решение системы линейных уравнений, матричные и векторные умножения). Подпрограммы LAPACK в виде кодов на языке Fortran находятся в открытом доступе [38].

3. **BaseMeansToolbox.f90** содержит набор подпрограмм для удобного создания матриц и векторов (матриц из нулей, единичной матрицы), расчета нормы вектора, скалярного и векторного произведения векторов.

4. **Ephemeris.f90** содержит интерфейсы к программам NASA›s Jet Propulsion Laboratory для расчета положений и скоростей небесных тел. Эти подпрограммы на языке Fortran также находятся в открытом доступе [39]. К этому модулю поставляется файл JPLEPH с эфемеридами модели DE430, который можно скачать на указанном выше сайте. Вместо файла JPLEPH можно использовать любой другой файл с эфемеридами той же структуры, например, эфемериды лаборатории эфемеридной астрономии Института прикладной астрономии РАН [40].

5. **EquationsModule.f90** содержит подпрограммы для расчета правых частей уравнений движения космического аппарата с возмущениями (сложное гравитационное поле Луны, давление солнечного излучения, возмущение за счет гармоники J_2 геопотенциала, атмосферное сопротивление). Реализованы как сами уравнения движения, так и уравнения в вариациях, которые используются для расчета матриц чувствительности фазовых векторов к начальным условиям. Во вспомогательном модуле GravityMoonCoefficients50.f90 содержатся коэффициенты сложного поля Луны до степени и порядка 50 включительно.

6. **Transformations.f90** содержит подпрограммы для преобразования переменных, описывающих движение (например, из декартовых координат в орбитальные элементы и обратно), и систем координат (например, между селеноцентрической небесной системой координат и системой Mean Earth/Mean-Rotation, связанной с селенографическими координатами на поверхности Луны).

7. **OdeToolbox.f90** содержит реализации численных методов интегрирования систем обыкновенных дифференциальных уравнений. Среди имеющихся на данный момент методов – классический метод Рунге–Кутты 4-го порядка, метод Рунге–Кутты 8-го порядка с постоянным шагом, метод Дормана–Принса 5(4), а также многошаговый метод Адамса переменного порядка (до 12-го включительно) с адаптивным шагом.

8. **PropagationModule.f90** содержит подпрограммы для интегрирования уравнений движения с какими-либо из упомянутых выше возмущений указанными методами интегрирования и указанными возмущениями.

9. **VisibilityModule.f90** содержит подпрограммы для расчета радиовидимости космического аппарата из точек на поверхности небесного тела.

Эти модули были скомпилированы в файл **FKIAMToolbox.cp39-win_amd64.pyd** с использованием программы Fortran to Python interface generator (F2PY) [41], поставляемой в пакете numpy с использованием компилятора Intel(r) Visual Fortran Compiler, который свободно распространяется в рамках системы oneAPI [42] компании Intel. В результате в программах на Python можно импортировать скомпилированную библиотеку как **import FKIAMToolbox** и обращаться к функциям внутри через точку. Например, команда

```
FKIAMToolbox.ephemeris.juliandate(year,
month, day, hour, minute, second)
```

обратится к подпрограмме juliandate модуля ephemeris и возвратит юлианскую дату для заданного года, месяца, дня, часа, минуты и секунды. Если

на вход какой-то Fortran-функции нужно передать числовой массив, он должен иметь тип numpy.ndarray в Python.

Из-за особенностей языка программирования Fortran и создаваемого программой F2PY Fortran-Python интерфейса, обращение в прикладных программах к скомпилированному модулю FKIAMToolbox неудобно и даже небезопасно (глобальные параметры внутри FKIAMToolbox легко изменить извне), поэтому для удобства пользователей был написан Python-модуль kiam.py, содержащий интерфейсы для обращения к FKIAMToolbox. Именно эти интерфейсы проверяют формат и правильность данных, которые затем будут переданы в процедуры FKIAMToolbox: эти процедуры направлены на максимизацию скорости вычислений и проверку входных данных, кроме типа данных, не осуществляют.

Кроме модуля kiam.py в библиотеку также входит модуль Trajectory.py, реализующий класс Trajectory для проектирования траекторий космических аппаратов. Траектории могут быть произвольными, они определяются начальными условиями и длиной интервала времени. Класс Trajectory позволяет создавать объекты, содержащие поля states (массивы фазовых состояний) и times (моменты времени), а также другие вспомогательные атрибуты. Класс содержит метод change_system(target_system), переводящий одновременно все фазовые состояния в целевую систему координат, метод change_vars(target_vars), преобразующий одновременно все фазовые состояния из states к новым переменным, и другие методы.

Библиотека включает модуль engine.py, содержащий классы двигательных установок, которые отличаются силой тяги, удельным импульсом, массой, размерами и прочими параметрами. Например, реализованы классы SPT50, SPT50M, SPT70 двигателей малой тяги [43].

Подробную справку о модулях, функциях и классах можно получить по ссылкам на GitHub-странице проекта [44].

Опишем теперь более подробно возможности класса Trajectory для проектирования траекторий.

4. КЛАСС TRAJECTORY ДЛЯ ПРОЕКТИРОВАНИЯ ТРАЕКТОРИЙ

Класс Trajectory определен в модуле Trajectory.py. Его назначение – быстрая и простая генерация траекторий в выбранной модели с заданными начальными условиями и удобное преобразование фазовых векторов траектории между различными системами координат, переменными и единицами измерения.

Метод __init__ принимает на вход начальный вектор состояния, начальный момент времени, начальную юлианскую дату, название используемых для задания состояния переменных, название системы координат и системы единиц измерения. Например, команда

```
tr = Trajectory(numpy.array([1, 0, 0, 0, 1, 0]), 0.0,
2459905.5, 'rv', 'gcrs', 'earth')
```

создает объект tr класса Trajectory с начальным состоянием numpy.array([1,0,0,0,1,0] (радиус-вектор [1,0,0], скорость [0,1,0]), начальным моментом времени 0.0, начальной юлианской датой 2459905.5 (отвечает дате 21.11.2022), в переменных 'rv' (положение–скорость), в системе координат GCRS (геоцентрическая небесная система координат), в системе единиц 'earth' (единица расстояния – средний радиус Земли, единица скорости – первая космическая скорость).

Возможно создание объекта в классических или равноденственных орбитальных элементах, других инерциальных или неинерциальных системах координат, связанных с Солнцем, Луной и планетами Солнечной системы, в других безразмерных или размерных системах единиц. Подробную информацию о всех возможностях можно найти на GitHub-странице проекта [44].

Метод set_model устанавливает модель движения. Например, строка

```
tr.set_model('rv', 'nbp', 'earth', ['moon', 'sun'])
```

задаст модель задачи n-тел ('nbp') в переменных положение–скорость ('rv'), с центральным телом Земля ('earth') и возмущениями от гравитационного притяжения Луной и Солнцем (['moon', 'sun']). Модель движения в случае задачи n-тел может быть задана и в других переменных, например, равноденственных элементах ('ee' вместо 'rv'). При этом не важно, в каких переменных была инициализирована траектория 'tr'. Каждая определенная в классе Trajectory модель имеет собственные переменные, систему координат и систему единиц, которые могут не совпадать с заданными пользователем переменными, системами координат и системами единиц во время создания объекта траектории.

Некоторые данные, используемые для последующего интегрирования траектории, необходимо установить вручную. Например,

```
tr.model['data']['jd_zero'] = jd0
```

установит юлианскую дату jd0, соответствующую $t = 0$ в уравнениях движения,

```
tr.model['data']['mass'] = 100.0
```

задаст массу космического аппарата в кг,

```
tr.model['data']['area'] = 2.0
```

задаст площадь миделева сечения в м^2,

```
tr.model['data']['order'] = 5
```

задаст степень и порядок сложного гравитационного поля Луны.

Распространение по времени (интегрирование) траекторий осуществляется с помощью метода propagate. Например,

```
tr.propagate(100, 1000)
```

запустит интегрирование траектории в рамках выбранной модели на 100 единиц времени вперед и сохранит 1000 равномерно отстоящих по времени узлов, при этом в tr.times запишет узлы времени, а в tr.states запишет фазовые векторы в этих узлах.

Особенностью класса Trajectory является автоматическое преобразование переменных интегрирования, систем координат и систем единиц из тех, что задал пользователь, в те, что используются моделью движения для ее интегрирования. Опишем подробно механизм этого автоматического преобразования.

Класс Trajectory содержит графы преобразований между переменными, между системами координат и между системами единиц. Граф преобразований переменных показан на рис. 1. Здесь rv, oe, ee обозначают соответственно переменные положение–скорость, классические орбитальные элементы и равноденственные орбитальные элементы. Суффикс _stm означает те же векторы переменных, расширенные переходной матрицей (state transition matrix, STM). Суффикс m означает те же векторы переменных, расширенные добавлением массы аппарата. Стрелка означает преобразование, которое реализовано явно в классе Trajectory. Надпись над стрелкой указывает на ограничения, при которых это преобразование может быть выполнено. В данном случае некоторые преобразования ограничены используемыми системами единиц: либо 'earth', либо 'moon'.

Некоторые преобразования действуют только в одну сторону. Например, легко может быть преобразован вектор, содержащий положение, скорость и массу аппарата, в вектор только из положения

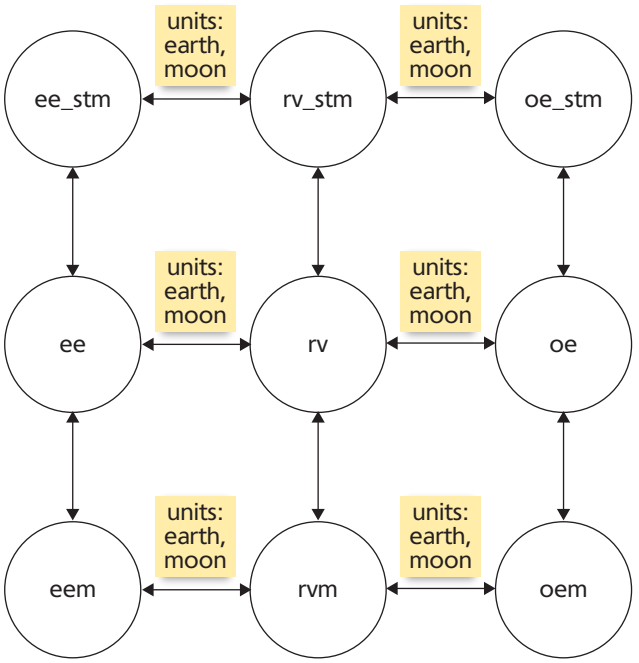

Рис. 1. Граф преобразований переменных в классе Trajectory для моделей движения, использующих эфемериды небесных тел.

и скорости, а обратного преобразования не существует.

Метод change_vars позволяет переключаться между переменными. Например, пусть траектория описывается переменными oe и записана команда tr.change_vars('ee').

В этом случае алгоритм найдет кратчайший пусть от вершины 'oe' до вершины 'ee' и выполнит цепочку преобразований (по порядку обратится к функциям, осуществляющим соответствующие преобразования). Метод change_vars автоматически запускается в методе propagate перед интегрированием уравнений движения, осуществляя переход к переменным выбранной пользователем модели.

Часть графа преобразований систем координат для моделей движения, использующих эфемериды небесных тел, показана на рис. 2. Здесь есть несколько групп систем координат. Системы координат sors, scrs, mer, ssrf_em связаны с Луной, системы itrs, gcrs, gsrf_se, gsrf_em связаны с Землей, а системы hcrs, hsrf_se связаны с Солнцем. Описания аббревиатур можно найти в справке на GitHub [44]. Аналогично в библиотеке реализованы преобразования, отвечающие стрелкам. Метод change_system, реализующий преобразования между любыми системами координат, находит кратчайший путь между вершинами и выполняет всю цепочку необходимых преобразований. Реализованы также преобразования между системами координат в других моделях

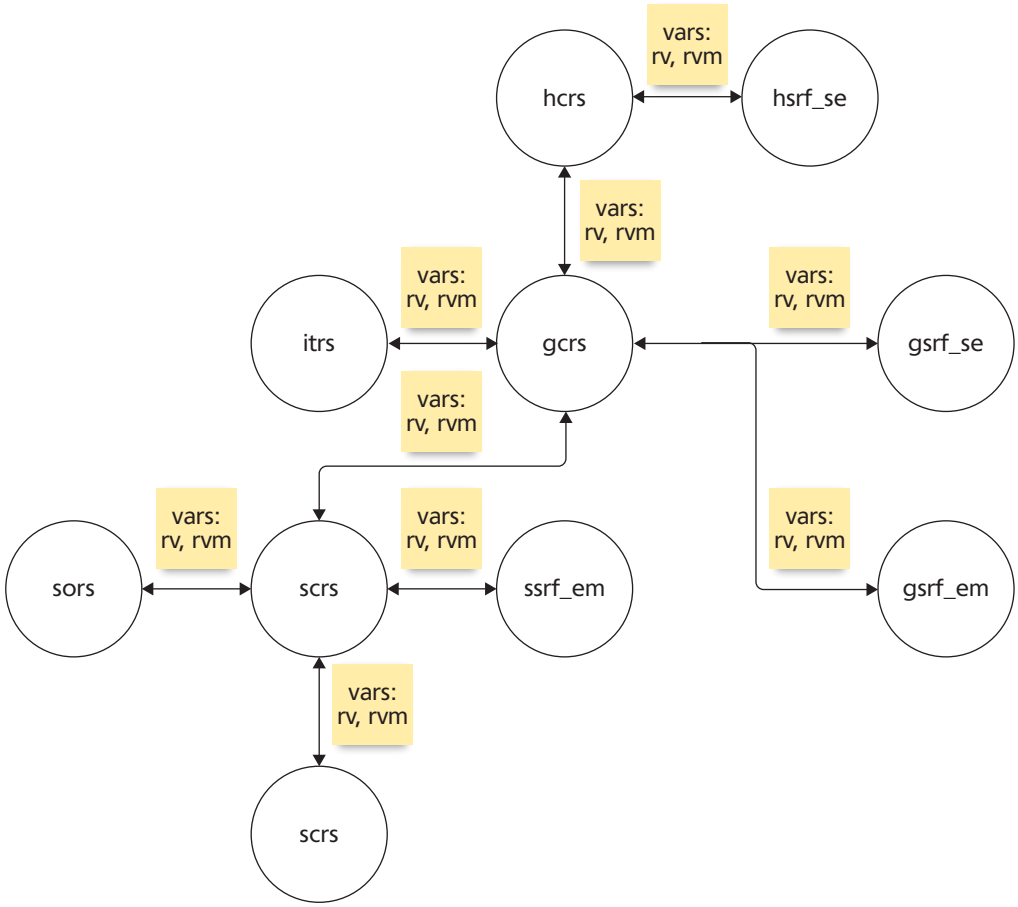

Рис. 2. Граф преобразований систем координат, связанных с Землей, Луной и Солнцем в классе Trajectory.

движения – круговой ограниченной задаче трех тел и бикруговой ограниченной задаче четырех тел, а также в задаче Хилла.

Похожий граф преобразований есть и для преобразования систем единиц; метод, отвечающий за это преобразование, называется change_units.

Итак, в любой момент пользователь может вызвать методы change_vars, change_system, change_units, и тогда преобразования затронут все фазовые состояния в массиве фазовых состояний tr.states. Преобразования переменных и систем координат реализованы в модуле Translations.f90.

Рассмотрим несколько конкретных примеров использования модуля Trajectory. Начнем с простейшего примера. В модели задачи двух тел круговую орбиту можно получить следующим образом (с точностью до настроек осей графика и шрифтов):

```
t0 = 0.0
s0 = numpy.array([1, 0, 0, 0, 1, 0])
jd0 = kiam.juliandate(2022,4,30,0,0,0)
tr = Trajectory.Trajectory(s0, t0, jd0, 'rv', 'gcrs',
'earth')
tr.set_model('rv', 'r2bp', 'earth', [])
tr.propagate(2*numpy.pi, 10000)
fig = tr.show('xy', language='rus')
```

В результате будет рассчитана траектория с начальным временем $t_0 = 0$, начальным состоянием (положение и скорость) x_0=[4.0,0,0,0,0.5,0] в системе отсчета GCRS (геоцентрическая небесная система координат) в безразмерной системе единиц 'earth' (гравитационный параметр равен единице, единица расстояния – радиус Земли, единица скорости – первая космическая скорость) на одном витке (период орбиты равен одной безразмерной единице времени). Последняя строчка запускает команду отображения орбиты в проекции на плоскость земного экватора на рисунке (рис. 3).

Приведем пример интегрирования траектории с управлением (с точностью до настроек осей графика и шрифтов):

```
units = kiam.units('sun')
AU = units['AccUnit']
TU = units['TimeUnit']
engine = SPT50()
```

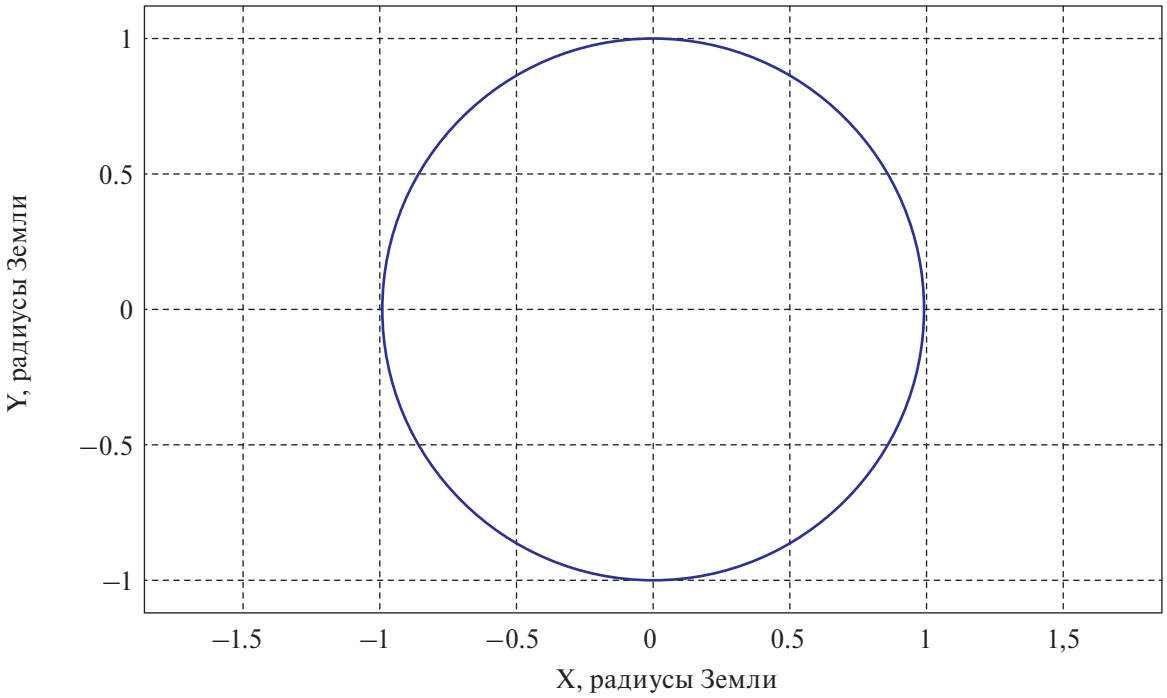

Рис. 3. Круговая экваториальная орбита, проинтегрированная на одном витке в модели задачи двух тел.

```
engine.force /= AU
engine.specific_impulse /= TU*24*3600

def control(t, x):
force_vector = x[3:6]/norm(x[3:6]) * engine.force
specific_impulse = engine.specific_impulse
return force_vector, specific_impulse

t0 = 0.0
jd0 = kiam.juliandate(2023,1,1,0,0,0)
s0 = numpy.array([1,0,0,0,1,0,100])
tr = Trajectory(s0, t0, jd0, 'rvm', 'hcrs', 'sun')
tr.set_model('rvm', 'nbp', 'sun', ['jupiter'])
tr.model['data']['jd_zero'] = jd0
tr.model['data']['area'] = 2.0
tr.model['data']['mass'] = s0[6]
tr.model['control'] = control
tr.propagate(500/TU, 100)
tr.show('xy')
```

В первых трех строках загружается система единиц, связанная с Солнцем: единицей расстояния является 1 а.е., единицей скорости – круговая скорость вокруг Солнца на расстоянии 1 а.е., гравитационный параметр Солнца равен единице. Далее создается экземпляр класса SPT50, содержащий информацию о двигателе малой тяги СПД-50; его сила и удельный импульс приводятся к безразмерной системе единиц. Далее определяется функция control, которая сопоставляет моменту времени и фазовому вектору аппарата силу тяги и удельный импульс. Затем задаются начальные условия, объект траектории, модель. Траектория распространяется на интервале 500 дней. В процессе интегрирования используются уравнения движения и интегратор, реализованные на Fortran, но при каждом обращении к функции правой части производится вызов функции control, написанной на Python. Функция control кроме вектора силы тяги выдает также удельный импульс: это может быть полезно в тех случаях, когда величина удельного импульса может быть переменной. В результате выводится график траектории на плоскости международной небесной системы координат (рис. 4).

Рассмотрим пример с более сложной динамикой: движение вокруг Луны с учетом гравитационного притяжения Земли и Солнца, а также сложного поля Луны до 10-й степени и порядка включительно (с точностью до настроек осей графика и шрифтов):

```
units = kiam.units('moon')
t0 = 0.0
s0 = numpy.array([3, 0.01,
kiam.deg2rad(80), 0, 0, 0])
jd0 = kiam.juliandate(2022,4,30,0,0,0)
tr = Trajectory.Trajectory(s0, t0, jd0, 'oe', 'mer',
'moon')
tr.set_model('rv', 'nbp', 'moon', ['earth', 'sun',
'cmplxmoon'])
tr.model['data']['jd_zero'] = jd0
```

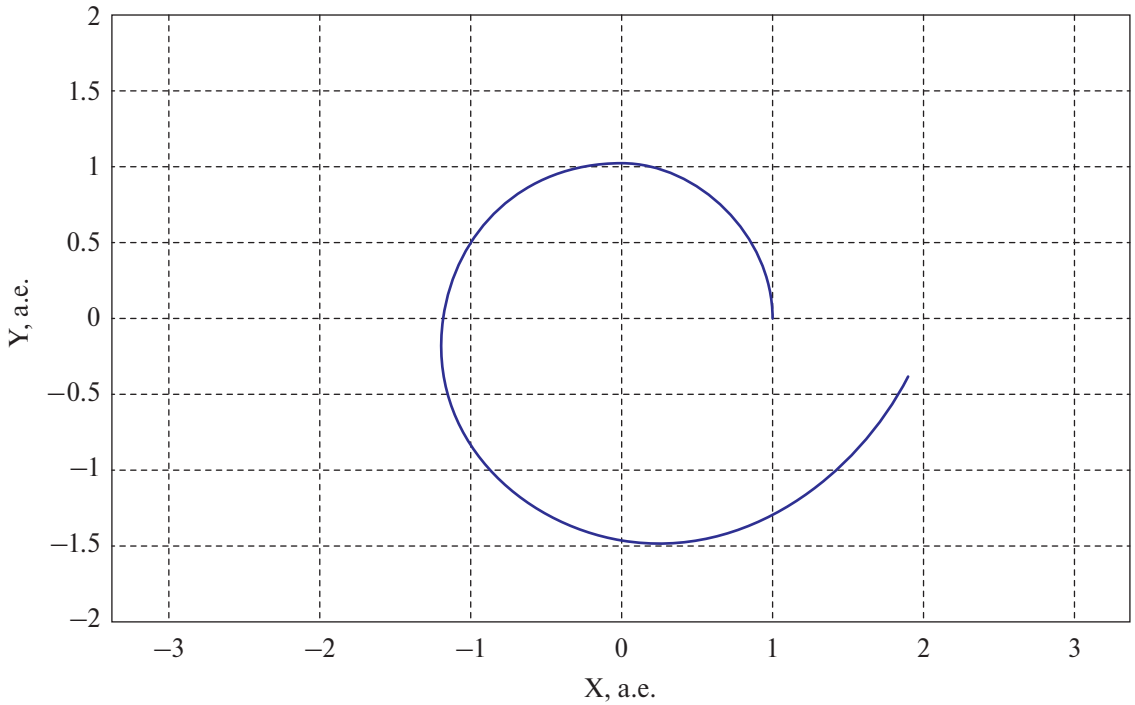

Рис. 4. Пример плоской траектории с малой тягой вокруг Солнца.

```
tr.model['data']['mass'] = 100.0
tr.model['data']['area'] = 2.0
tr.model['data']['order'] = 10
tr.propagate(10/units['TimeUnit'], 5000)
tr.change_system('mer')
tr.change_vars('oe')
tr.change_units('dim')
tr.show('e', language='rus')
```

В результате будет получена траектория с начальным временем $t_0 = 0$, начальными классическими орбитальными элементами $a_0 = 3$ (большая полуось, безразмерные единицы), $e_0 = 0.01$ (эксцентриситет орбиты), $i_0 = 80°$ (наклонение орбиты), $\Omega_0 = 0°$ (долгота восходящего узла), $\omega_0 = 0°$ (аргумент перицентра), $\vartheta_0 = 0°$ (истинная аномалия) на интервале времени 10 дней. Орбитальные элементы даны в связанной с Луной системе координат MER.

При исполнении команды tr.propagate переменные будут трансформированы в декартовы положение и скорость, а система координат – в систему SCRS, так как именно в этих терминах определяется модель и правые части уравнений, соответствующие параметрам 'rv' и 'nbp' в строке с tr.set_model. После интегрирования уравнений движения переменные останутся 'rv' (положение и скорость), а система координат останется SCRS. Так как орбитальными элементами обычно интересуются в связанной с небесным телом системе координат, в примере выше после интегрирования идут команды, которые возвращают все фазовые состояния рассчитанной траектории в систему MER, классическим орбитальным элементам и размерной системе единиц. Последняя строка отображает график эволюции эксцентриситета с течением времени (рис. 5).

Для анализа видимости поверхности небесного тела полезно визуализировать подспутниковую трассу орбиты – точки поверхности тела, в которых ее пересекает радиус-вектор аппарата в ходе орбитального движения. На следующем примере показано, как отобразить подспутниковые точки на низкой окололунной орбите. Сначала рассчитывается окололунная орбита:

```
    units = kiam.units('moon')
    t0 = 0.0
    s0 = numpy.array([1.2, 0.01,
kiam.deg2rad(80), 0, 0, 0])
    jd0 = kiam.juliandate(2022,4,30,0,0,0)
    tr = Trajectory.Trajectory(s0, t0, jd0, 'oe', 'mer',
'moon')
    tr.set_model('rv', 'nbp', 'moon', ['earth', 'sun',
'cmplxmoon'])
    tr.model['data']['jd_zero'] = jd0
    tr.model['data']['mass'] = 100.0
    tr.model['data']['area'] = 2.0
    tr.model['data']['order'] = 40
    tr.propagate(1.0/units['TimeUnit'], 10000)
    tr.change_system('mer')
```

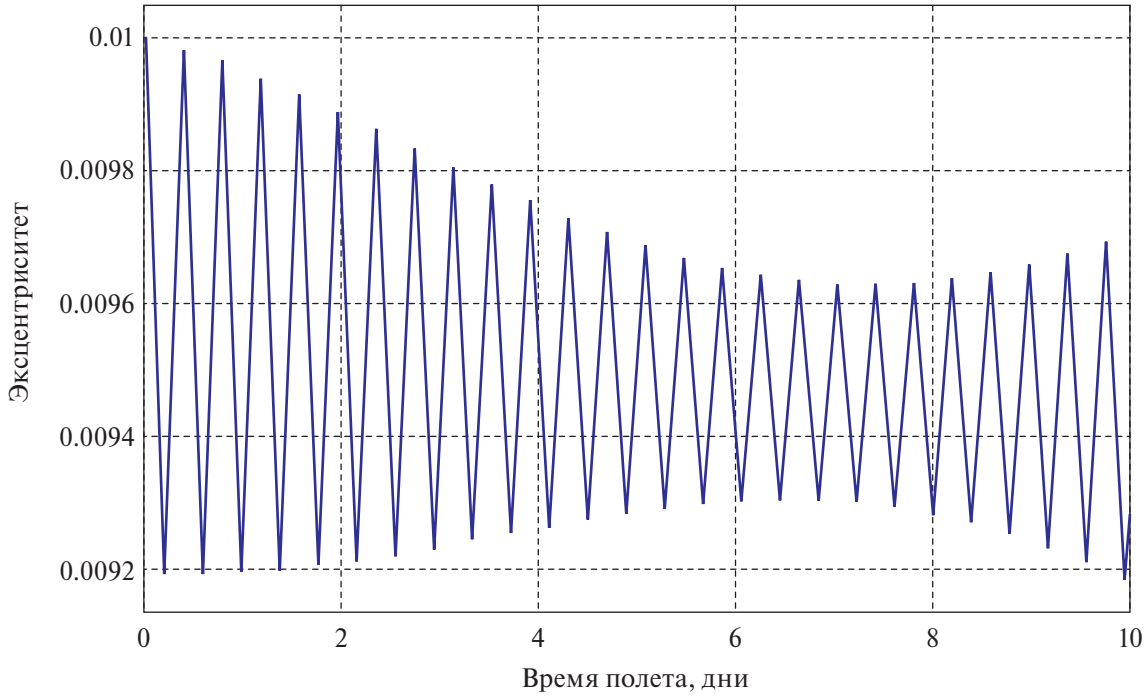

Рис. 5. Эволюция эксцентриситета окололунной орбиты в сложной модели движения.

Далее средствами рисования графиков (например, пакета plotly) формируется графический объект траектории

```
traj_go = go.Scatter3d(
x=tr.states[0, :],
y=tr.states[1, :],
z=tr.states[2, :],
mode='lines',
line='color': 'rgb(178,34,34)', 'width': 3),
```

рассчитываются расстояния до притягивающего центра

```
r = numpy.sqrt(tr.states[0, :] ** 2 +
tr.states[1, :] ** 2 +
tr.states[2, :] ** 2)
```

координаты подспутниковых точек (радиус Луны принят равным единице)

```
traj_surf_go = go.Scatter3d(
x=tr.states[0, :]/r,
y=tr.states[1, :]/r,
z=tr.states[2, :]/r,
mode='lines',
line='color': 'rgb(178,34,34)', 'width': 3)
```

В модуле kiam.py есть возможность отображения поверхности Луны:

```
fig_moon = kiam.body_surface('moon')
moon_go = fig_moon['data'][0]
```

Наконец, на экран выводятся все графические объекты:

```
fig_traj = go.Figure()
fig_traj.add_trace(traj_go)
fig_traj.add_trace(moon_go)
fig_traj.add_trace(traj_surf_go)
```

Результат приводится на рис. 6.

5. ЗАКЛЮЧЕНИЕ

Разработана новая программная библиотека для проектирования и моделирования орбитального движения космического аппарата KIAM Astrodynamics Toolbox. Библиотека написана на языках Fortran (основные вычислительные модули) и Python (интерфейсы к ним, а также высокоуровневые классы). Библиотека позволяет выполнять преобразования переменных, систем координат и систем единиц, получать эфемеридную информацию о движении небесных тел, интегрировать и визуализировать траектории космических аппаратов в выбранной модели с заданными начальными условиями. Библиотека находится в открытом доступе, свободна для использования и рассчитана на специалистов в области механики космического полета и обучающихся по соответствующим программам подготовки. Она является альтернативой существующим проприетарным решениям с пользовательским графическим интерфейсом, а также аналогичным прог-

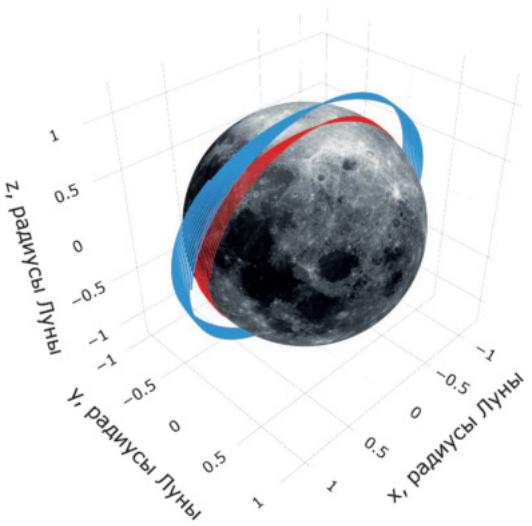

Рис. 6. Низкая окололунная орбита с подспутниковой трассой.

раммным библиотекам, заметно ограниченным по функционалу.

СПИСОК ЛИТЕРАТУРЫ

1. AGI Products // Сайт компании «AGI» (https://www.agi.com/products). Просмотрено: 14.04.2023.
2. FreeFlyer // Сайт компании «a.i. solutions» (https://ai-solutions.com/freeflyer/). Просмотрено: 14.04.2023.
3. GMAT // Хостинг проектов «SourceForge.net» (https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2020a/). Просмотрено: 14.04.2023.
4. *Иванов Д.С., Овчинников М.Ю., Ролдугин Д.С., Ткачев С.С., Трофимов С.П., Шестаков С.А., Широбоков М.Г.* Программный комплекс для моделирования орбитального и углового движения спутников // Математическое моделирование. 2019. Т. 31. № 12. С. 44–56.
5. Fortran Astrodynamics Toolkit // Страница на веб-сервисе GitHub (https://github.com/jacobwilliams/Fortran-Astrodynamics-Toolkit). Просмотрено: 14.04.2023.
6. Pykep // Страница на веб-сервисе GitHub (https://esa.github.io/pykep/). Просмотрено: 14.04.2023.
7. Can't install anything using conda, it hangs in solving environment // Страница на веб-сервисе GitHub (https://github.com/conda/conda/issues/8051). Просмотрено: 14.04.2023.
8. Poliastro // Страница команды разработчиков (https://docs.poliastro.space/en/stable/index.html). Просмотрено: 14.04.2023.
9. SPICE, An Observation Geometry System for Space Science Missions // Главная страница сайта про SPICE Toolkit (https://naif.jpl.nasa.gov/naif/index.html). Просмотрено: 14.04.2023.
10. An Overview of SPICE // Презентация библиотеки SPICE Toolkit (https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/03_spice_overview.pdf). Просмотрено: 14.04.2023.
11. *Широбоков М.Г., Трофимов С.П.* Высокоуровневые средства моделирования межпланетного орбитального движения // XLV Академические чтения по космонавтике, посвященные памяти академика С.П. Королёва и других выдающихся отечественных ученых – пионеров освоения космического пространства. Сб. тез. в 4 т. 2021. Т. 1. С. 415–418.
12. *Широбоков М.Г.* KIAM Astrodynamics Toolbox 2.0 для проектирования космических миссий // Тр. 64-й Всерос. науч. конф. МФТИ. 29 ноября – 03 декабря 2021 г. Прикладная математика и информатика. 2021.
13. *Широбоков М.Г.* Высокоуровневое моделирование управляемого орбитального движения в KIAM Astrodynamics Toolbox // Тр. 63-й Всерос. науч. конф. МФТИ. 23–29 ноября 2020 года. Прикладная математика и информатика. 2020. С. 52–54.
14. KIAM Astrodynamics Toolbox // Страница на веб-сервисе GitHub (https://github.com/shmaxg/KIAMToolbox). Просмотрено: 14.04.2023.
15. MIT License // Страница на веб-сервисе GitHub (https://choosealicense.com/licenses/mit/). Просмотрено: 14.04.2023.
16. Netlib Repository at UTK and ORNL // Сайт репозитория программного обеспечения Netlib (www.netlib.org). Просмотрено: 14.04.2023.
17. Solar System Dynamics // Сайт лаборатории реактивного движения НАСА (https://ssd.jpl.nasa.gov/). Просмотрено: 14.04.2023.
18. Эфемериды EPM // Сайт лаборатории эфемеридной астрономии Института прикладной астрономии Российской академии наук (https://iaaras.ru/dept/ephemeris/epm/). Просмотрено: 14.04.2023.
19. F2PY user guide and reference manual // Сайт программы F2PY пакета numpy (https://numpy.org/doc/stable/f2py/). Просмотрено: 14.04.2023.
20. Intel oneAPI Toolkits // Сайт системы Intel oneAPI (https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html). Просмотрено: 14.04.2023.
21. Продукция ОКБ Факел // Страница сайта ОКБ Факел (https://fakel-russia.com/produkciya). Просмотрено: 14.04.2023.
22. KIAM Astrodynamics Toolbox Wiki // Страница на веб-сервисе GitHub (https://github.com/shmaxg/KIAMToolbox/wiki). Просмотрено: 14.04.2023.

KIAM ASTRODYNAMICS TOOLBOX FOR SPACECRAFT ORBITAL MOTION DESIGN

M. G. Shirobokov[a], S. P. Trifimov[a]

[a] *Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, Miusskaya sq. 4, Moscow, 125047, Russia*

The KIAM Astrodynamics Toolbox, a new software library for designing spacecraft orbital motion, is introduced. The toolbox is developed at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences using Fortran and Python languages, thus combining the computational speed and the flexibility of program design. The library can be useful for space flight mechanics specialists, as well as for students in relevant educational programs.

Keywords: Software library, trajectory, spacecraft, interplanetary flight, Python, Fortran

REFERENCES

1. AGI Products // AGI company website (https://www.agi.com/products). Viewed: 14.04.2023.
2. FreeFlyer // Сайт компании «a.i. solutions» (https://ai-solutions.com/freeflyer/). Viewed: 14.04.2023.
3. GMAT // Хостинг проектов «SourceForge.net» (https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2020a/). Viewed: 14.04.2023.
4. *Ivanov D.S., Ovchinnikov M.Yu., Roldugin D.S., Tkachev S.S., Trofimov S.P., Shestakov S.A., Shirobokov M.G.* Software package for modeling orbital and angular motion satellites // Mathematical modeling. 2019. Vol. 31. No. 12. P. 44–56.
5. Fortran Astrodynamics Toolkit // Page on the GitHub web service (https://github.com/jacobwilliams/Fortran-Astrodynamics-Toolkit). Просмотрено: 14.04.2023.
6. Pykep // Page on the GitHub web service (https://esa.github.io/pykep/). Viewed: 14.04.2023.
7. Can›t install anything using conda, it hangs in solving environment // Page on the GitHub web service (https://github.com/conda/conda/issues/8051). Viewed: 14.04.2023.
8. Poliastro // Development team page (https://docs.poliastro.space/en/stable/index.html). Viewed: 14.04.2023.
9. SPICE, An Observation Geometry System for Space Science Missions // Home page of the site about SPICE Toolkit (https://naif.jpl.nasa.gov/naif/index.html). Viewed: 14.04.2023.
10. An Overview of SPICE // Presentation of the SPICE Toolkit library (https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/03_spice_overview.pdf). Viewed: 14.04.2023.
11. *Shirobokov M.G., Trofimov S.P.* High-level tools for modeling interplanetary orbital motion // XLV Academic readings on astronautics, dedicated to the memory of Academician S.P. Korolev and other outstanding domestic scientists – pioneers of space exploration: collection of abstracts: in 4 volumes, 2021. Vol. 1. P. 415–418.
12. *Shirobokov M.G.* KIAM Astrodynamics Toolbox 2.0 for designing space missions // Proceedings of the 64th All-Russian Scientific Conference of MIPT. November 29 – December 03, 2021 Applied mathematics and computer science. 2021.
13. *Shirobokov M.G.* High-level modeling of controlled orbital motion in KIAM Astrodynamics Toolbox // Proceedings of the 63rd All-Russian Scientific Conference of MIPT. November 23–29, 2020. Applied mathematics and computer science. 2020. P. 52–54.
14. KIAM Astrodynamics Toolbox // Page on the GitHub web service (https://github.com/shmaxg/KIAM-Toolbox) Viewed: 14.04.2023.
15. MIT License // Страница на веб-сервисе GitHub (https://choosealicense.com/licenses/mit/) Viewed: 14.04.2023.
16. Netlib Repository at UTK and ORNL // Netlib software repository website (www.netlib.org) Viewed: 14.04.2023.
17. Solar System Dynamics // NASA Jet Propulsion Laboratory website (https://ssd.jpl.nasa.gov/) Viewed: 14.04.2023.
18. Эфемериды EPM // Website of the Laboratory of Ephemeris Astronomy of the Institute of Applied Astronomy of the Russian Academy of Sciences (https://iaaras.ru/dept/ephemeris/epm/) Viewed: 14.04.2023.
19. F2PY user guide and reference manual // Website of the F2PY program of the numpy package (https://numpy.org/doc/stable/f2py/) Viewed: 14.04.2023.
20. Intel oneAPI Toolkits // Intel oneAPI system website (https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html) Viewed: 14.04.2023.
21. Продукция ОКБ Факел // OKB Fakel website page (https://fakel-russia.com/produkciya) Viewed: 14.04.2023.
22. KIAM Astrodynamics Toolbox Wiki // Page on the GitHub web service (https://github.com/shmaxg/KIAMToolbox/wiki) Viewed: 14.04.2023.