XXVII Международный конкурс научно-технических работ школьников
«Старт в Науку»

Секция «Математическое и компьютерное моделирование биологических, физических и химических процессов. Ядерные и плазменные технологии»

Научно-исследовательская работа.

«Компьютерное моделирование задачи N тел на примере движения кометы C/2023 A3»

Автор работы:
Клопов Илья Иванович,
учащийся 11 класса
МБОУ «СОШ №31» г. Калуги

Научный руководитель:
Алексеев Александр Юрьевич,
научный сотрудник
физико-астрономического
отдела ГМИК им. К.Э. Циолковского

Москва 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Гравитационная задача N тел является классической проблемой небесной механики и гравитационной динамики Ньютона.

Она формулируется следующим образом. В пустоте находится N материальных точек, массы которых известны: m_i. Пусть попарное взаимодействие точек подчинено закону тяготения Ньютона, и пусть силы гравитации аддитивны[1]. Пусть известны начальные (на момент времени $t = 0$) положения и скорости каждой точки $\vec{r}_{i0}$, $\vec{v}_{i0}$. Требуется найти положения точек для всех последующих моментов времени. Аналитического решения задачи для N > 2 в общем случае не существует.

С появлением компьютерной техники появилась реальная возможность изучать свойства систем гравитирующих тел путём численного решения системы уравнений движения.

Численные методы сталкиваются с теми же проблемами, что и аналитические — при тесных сближениях тел необходимо уменьшать шаг интегрирования, а при этом быстро растут численные ошибки [1].

Актуальность

Моделирование взаимодействия тел по физическим законам позволяет прогнозировать поведение реальных систем. Для этой задачи не существует аналитического решения, но с появлением компьютерной техники стало возможным проводить расчеты с помощью численных методов.

Проблема

По результатам анализа доступных источников, было выявлено, что на данный момент не существует программы с открытым исходным кодом, предоставляющей все необходимые инструменты для моделирования движения тел, его визуализации и анализа, а также прогнозирования положений тел в будущем для астрономических наблюдений.

Гипотеза

Предполагается, что компьютерное моделирование гравитационной задачи N тел позволит рассчитать движение кометы C2023/A3 (Цзыцзиньшань - ATLAS), а также поведение многих других систем.

Цель проекта

Разработать программу для компьютерного моделирования задачи N тел.

[1] Аддитивность (от лат. additivus — прибавляемый), свойство величин, состоящее в том, что значение величины, соответствующее целому объекту, равно сумме значений величин, соответствующих его частям при любом разбиении объекта на части.

Объект исследования: Моделирование физического процесса с помощью компьютерной техники.

Предмет исследования: Компьютерное моделирование задачи N тел с использованием численных методов.

Задачи

1. Разработать теоретическую модель гравитационного взаимодействия N тел.
2. Выбрать оптимальный язык программирования и необходимые библиотеки для реализации численного моделирования.
3. Реализовать расчет:
   - траекторий тел;
   - барицентра системы;
   - кинетической, потенциальной, полной механической энергии системы;
   - динамического шага по времени.
4. Создать модуль для визуализации результатов моделирования, включая отображение положения тел на карте звёздного неба.
5. Проверить соблюдение законов физики в моделируемых системах и сопоставить результаты моделирования с реальными астрономическими данными.

Методы исследования

В ходе работы были использованы следующие методы исследования:

Метод анализа — метод исследования, характеризующийся выделением и изучением отдельных частей объектов исследования. В рамках работы метод анализа использовался при изучении точности и производительности вычислительных алгоритмов, а также при изучении физических параметров моделируемых систем.

Метод сравнения — метод сопоставления двух и более объектов, выделение в них общего и различного с целью классификации и типологии. В данном исследовании метод сравнения применялся для оценки эффективности различных вычислительных алгоритмов, реализованных в программе.

Новизна и преимущества решения перед альтернативными

Ключевым преимуществом данной компьютерной программы перед альтернативными будет являться наличие комплекса инструментов для расчета и перевода данных для астрономических наблюдений и визуализаций.

Данный подход исключает необходимость использования сторонних программных средств, что существенно упрощает рабочий процесс пользователя.

ГЛАВА 1. ТЕОРИЯ

1.1 Математическая модель

При изучении литературных источников была определена математическая модель для моделирования задачи N тел.

Задача Коши — одна из основных задач теории дифференциальных уравнений (обыкновенных и с частными производными); состоит в нахождении решения (интеграла) дифференциального уравнения, удовлетворяющего так называемым начальным условиям (начальным данным).

Задача Коши обычно возникает при анализе процессов, определяемых дифференциальным законом эволюции и начальным состоянием (математическим выражением которых и являются уравнение и начальное условие). Этим мотивируется терминология и выбор обозначений: начальные данные задаются при t = 0 а решение отыскивается при t > 0 [2].

Для решения данной задачи был выбран метод ломаных Эйлера. Существуют более точные методы численного интегрирования, например, метод Рунге-Кутты, однако в данной работе метод Эйлера был выбран из-за простоты в реализации.

Метод ломаных Эйлера — простейший конечно-разностный метод численного решения обыкновенных дифференциальных уравнений. Пусть дано дифференциальное уравнение:

$$\frac{dx}{dy} = f(x, y) \qquad (*)$$

С начальным условием

$$y(x_0) = y_0$$

Выбирается достаточно малый шаг по оси x, строятся точки

$$x_i = x_0 + ih, \qquad i = 0,1,2, \ldots,$$

и искомая интегральная кривая заменяется ломаной (ломаная Эйлера), звенья которой прямолинейны на отрезках $[x_i, x_{i+1}]$, а ординаты определяются по формулам

$$y_{i+1} = y_i + hf(x_i, y_i), \qquad i = 0,1,2, \ldots.$$

Если правая часть $f(x, y)$ уравнения (*) непрерывна, то на фиксированном отрезке $[x_0, x_0 + H]$ последовательность ломаных Эйлера при $h \to 0$ равномерно стремится к искомой интегральной кривой $y(x)$. При $f(x, y)$ дифференцируемой метод Эйлера имеет погрешность на шаге $O(h^2)$ и в целом $O(h)$ [3].

В следующем пункте определены формулы для описания задачи N тел с помощью метода Эйлера.

1.2 Физика процесса

Движение тел.

Перемещение тел в программе будет вычисляться с помощью следующих формул.

В симуляции тела притягиваются друг к другу с силой, которая находится по закону всемирного тяготения Ньютона в векторном виде:

$$\vec{F}_{12}(\vec{r}_1,\vec{r}_2) = Gm_1m_2\frac{\vec{r}_2-\vec{r}_1}{|\vec{r}_2-\vec{r}_1|^3},$$

где $\vec{r}_1$ и $\vec{r}_2$ – радиус-векторы тел, G – гравитационная постоянная, m_1, m_2 – массы тел.

Запишем ускорение тела через второй закон Ньютона:

$$\vec{a}_i = \frac{\vec{F}_{ij}}{m_i} = \sum_{j\neq i}^{n} Gm_j\frac{\vec{r}_j-\vec{r}_i}{\left|\vec{r}_j-\vec{r}_i\right|^3}$$

В дифференциальной форме:

$$\frac{d\vec{v}_i}{dt} = \vec{a}_i(\vec{r}_i,\vec{r}_j) = \sum_{j\neq i}^{n} Gm_j\frac{\vec{r}_j-\vec{r}_i}{\left|\vec{r}_j-\vec{r}_i\right|^3},$$

где $\vec{v}_i$ – скорость тела, t – время, G – гравитационная постоянная, m_j – масса тела, $\vec{r}_j$ и $\vec{r}_i$ – радиус-векторы тел.

Скорость тела будет равна:

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i,$$

где $\vec{r}_i$ – радиус-вектор тела.

Таким образом, движение тел описывается уравнениями:

$$\frac{d\vec{v}_i}{dt} = \sum_{j\neq i}^{n} Gm_j\frac{\vec{r}_j-\vec{r}_i}{\left|\vec{r}_j-\vec{r}_i\right|^3}$$

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i$$

Вычисление перемещений всех тел системы имеет сложность: $O(N^2)$.

Барицентр.

Для вычисления параметров барицентра системы необходимы следующие формулы:

Масса барицентра находится как сумма масс всех тел системы:

$$M_b = \sum_{i}^{n} m_i$$

Радиус-вектор барицентра задается формулой:

$$\overrightarrow{r_b} = \frac{\sum_i^n \vec{r}_i m_i}{M_b}$$

Скорость барицентра задается формулой:

$$\overrightarrow{v_b} = \frac{\sum_i^n \vec{v}_i m_i}{M_b}$$

Расчет барицентра системы имеет сложность: $O(N)$.

Расчет полной механической энергии системы.

Для проверки соблюдения закона сохранения механической энергии в симуляции понадобятся формулы для нахождения кинетической, потенциальной и полной механической энергии системы.

Кинетическая энергия системы находится как алгебраическая сумма кинетических энергий всех тел системы:

$$E_k = \sum_i^n \frac{m_i v_i^2}{2}$$

Потенциальная энергия системы находится как алгебраическая сумма попарных потенциальных энергий для всех тел системы:

$$E_p = \sum_{i=0}^{n} \sum_{j=i+1}^{n} \left(-G \frac{m_i m_j}{|\vec{r}_j - \vec{r}_i|} \right)$$

Полная механическая энергия системы находится как сумма кинетической и потенциальной энергии системы:

$$E_{total} = E_k + E_p$$

Вычисление полной механической энергии системы имеет сложность: $O(N^2)$.

Небесная сфера

Для перевода координат тел из декартовой во вторую экваториальную систему координат необходимы следующие формулы сферической тригонометрии:

Геоцентрическая эклиптическая долгота тела λ на небесной сфере находится по формуле:

$$\lambda = \tan^{-1} \frac{y}{x}$$

Геоцентрическая эклиптическая широта тела β на небесной сфере находится по формуле:

$$\beta = \tan^{-1} \frac{z}{\sqrt{x^2 + y^2}},$$

где x, y, z – координаты тел в геоцентрической декартовой системе координат.

Если ε – угол наклона оси вращения Земли к плоскости эклиптики, то:

Прямое восхождение тела α на небесной сфере находится по формуле:

$$\alpha = \tan^{-1}\frac{cos\varepsilon\ sin\lambda - sin\varepsilon\ tan\beta}{cos\lambda}$$

Склонение тела δ на небесной сфере находится по формуле:

$$\delta = \sin^{-1}(cos\varepsilon\ sin\beta + sin\varepsilon\ cos\beta sin\lambda)$$

1.2 Частный случай задачи N тел

Одним из частных случаев задачи N тел является движение N тел по окружности, так как в таком случае силы притяжения тел компенсируют друг друга, и система находится в равновесии. Все тела имеют одинаковую массу M и в начальный момент времени находятся в вершинах правильного многоугольника, вписанного в окружность радиуса R.

Для моделирования этой системы необходимо рассчитать ее начальные параметры. Выведем формулу для скорости, которую должно иметь тело массы M, для того, чтобы двигаться по окружности радиуса R под действием сил притяжения к остальным телам.

При движении по окружности модуль центростремительного ускорения одного тела можно выразить по формуле:

$$a_{\text{цс}} = \frac{{v_{full}}^2}{R}$$

где v_{full} — модуль скорости движения тела, R — радиус данной окружности. Отсюда скорость тела:

$$v_{full} = \sqrt{a_{\text{цс}} R}$$

Ускорение тела, вызванное притяжением к другим телам будет вычисляться по формуле:

$$a_{\text{цс}} = \sum_{i=1}^{n-1} a_{i\text{цс}}$$

Подставив ускорение в формулу скорости тела получим:

$$v_{full} = \sqrt{\sum_{i=1}^{n-1} a_{i\text{цс}}\, R}$$

где $a_{i\text{цс}}$ — центростремительное ускорение, вызванное притяжением к телу с индексом i.

Ускорение a_i можно найти через закон всемирного тяготения Ньютона:

$$a_i = \frac{F_i}{M} = \frac{GMM}{Mr_i^2} = G\frac{M}{r_i^2},$$

где M — масса каждого тела, r_i — расстояние между нулевым и i-м телом, G — гравитационная постоянная.

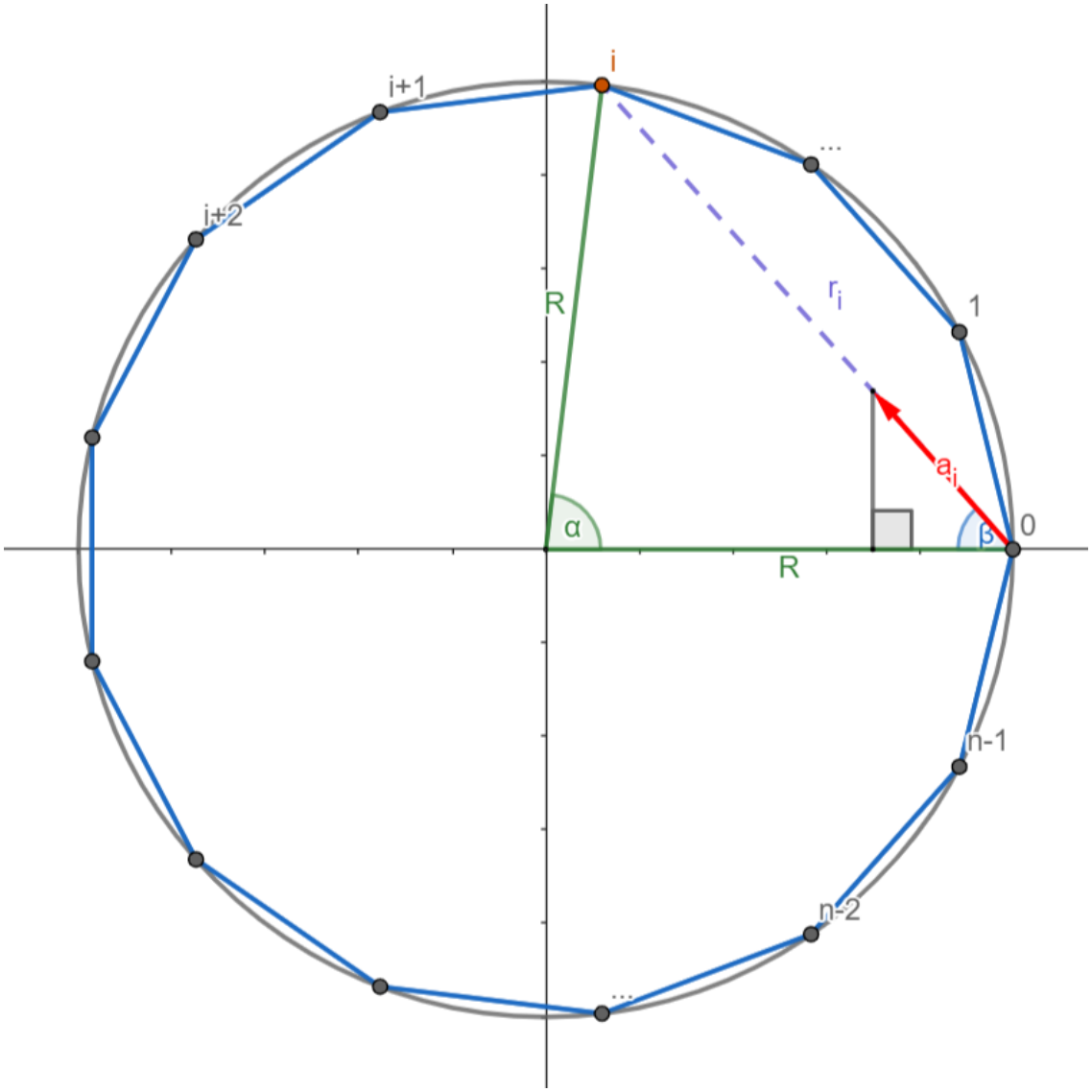


Рисунок 1 — Положение материальных точек на окружности при частном случае задачи N тел.

Угол от центра окружности между нулевым и i-м телом α будет равен:

$$\alpha = \frac{2\pi i}{n}$$

Квадрат расстояния r между нулевым и i-м телом по теореме косинусов будет равен:

$$r_i{}^2 = R^2 + R^2 - 2RR\cos\frac{2\pi i}{n} = 2R^2\left(1 - \cos\frac{2\pi i}{n}\right)$$

По формуле приведения:

$$\sin^2\frac{x}{2} = \frac{1-\cos x}{2}$$

$$r_i{}^2 = 4R^2\sin^2\frac{\pi i}{n}$$

Подставив квадрат расстояния в выражение для ускорения, получим:

$$a_i = \frac{GM}{4R^2\sin^2\frac{\pi i}{n}}$$

Для получения из полного ускорения центростремительную составляющую, необходимо спроектировать его на ось, соединяющую центр окружности и данное тело, умножив на косинус угла β между направлениями от данного тела на i-е тело и на центр окружности:

$$\beta = \frac{\pi-\alpha}{2} = \frac{\pi-\frac{2\pi i}{n}}{2} = \frac{\pi}{2} - \frac{\pi i}{n}$$

$$\cos\beta = \cos\left(\frac{\pi}{2} - \frac{\pi i}{n}\right) = \sin\frac{\pi i}{n}$$

$$a_{i\text{цс}} = \frac{GM}{4R^2\sin^2\frac{\pi i}{n}}\cos\beta = \frac{GM}{4R^2\sin^2\frac{\pi i}{n}}\sin\frac{\pi i}{n} = \frac{GM}{4R^2\sin\frac{\pi i}{n}}$$

При подстановке в формулу полной скорости получаем:

$$v_{full} = \sqrt{\sum_{i=1}^{n-1} a_{i\text{цс}}\,R} = \sqrt{\sum_{i=1}^{n-1}\frac{GM}{4R^2\sin\frac{\pi i}{n}}R} = \frac{1}{2}\sqrt{\frac{GM}{R}\sum_{i=1}^{n-1}\left(\sin\frac{\pi i}{n}\right)^{-1}}$$

Таким образом, полная скорость каждого тела вычисляется по формуле:

$$v_{full} = \frac{1}{2}\sqrt{\frac{GM}{R}\sum_{i=1}^{n-1}\left(\sin\frac{i\pi}{n}\right)^{-1}},$$

где R — радиус окружности, по которой движутся тела, M — масса одного тела, G — гравитационная постоянная.

Найдем проекции для радиус-вектора и для полной скорости на оси X и Y.

Радиус-вектор тела с индексом i в начальный момент времени имеет координаты:

$$\vec{r}_i\left(Rcos\frac{2\pi i}{n};\ Rsin\frac{2\pi i}{n}\right)$$

Скорость тела с индексом i в начальный момент времени имеет координаты:

$$\vec{v}_i\left(-v_{full}\sin\frac{2\pi i}{n};\ v_{full}\cos\frac{2\pi i}{n}\right)$$

Моделирование данного частного случая будет использовано для проверки расчетов программы.

1.3 Определение необходимых инструментов

Для разработки необходимой программы был выбран язык программирования Python версии 3.11, обладающий следующими характеристиками:

- *Простой синтаксис* облегчит и ускорит разработку программного кода;
- *Широкий выбор библиотек и модулей* позволит быстро решать поставленные задачи;
- *Переносимость* позволит запускать программу на различных платформах и операционных системах, включая Windows, macOS и Linux.

При написании кода программы были использованы следующие библиотеки языка программирования Python:

- Numpy — это фундаментальный пакет для научных вычислений на Python. Это библиотека Python, которая предоставляет объект многомерного массива, различные производные объекты и набор процедур для быстрых операций с массивами [4]. Numpy будет использоваться для различных математических вычислений.
- Matplotlib — это всеобъемлющая библиотека для создания статических, анимированных и интерактивных визуализаций [5]. С помощью Matplotlib в программе будут визуализироваться полученные из расчетов данные.
- Skyfield — это астрономический пакет на чистом языке Python, который позволяет легко получать высокоточные исследовательские координаты планет и спутников Земли. Skyfield понадобится для получения координат звезд и созвездий.

Также будет использоваться FFmpeg — набор свободных библиотек с открытым исходным кодом, которые позволяют записывать, конвертировать и передавать цифровые аудио- и видеозаписи в различных форматах.

С помощью FFmpeg программа будет сохранять полученные с помощью библиотеки Matplotlib визуализации в виде файлов с разными расширениями.

ГЛАВА 2. НАПИСАНИЕ ПРОГРАММЫ

GitFlic - первый российский сервис для хранения исходного кода и работы с ним. Основан на системе контроля версий Git[2].

Был создан онлайн репозиторий на Gitflic с исходным кодом проекта. Репозиторий доступен по ссылке: https://gitflic.ru/project/gs_team/gravity_simulator.

2.1 Расчет траекторий тел

Для расчета положений тел можно использовать стандартные циклы Python, объектно-ориентированное программирование (ООП) в Python с использованием массивов Numpy, а также целые массивы данных библиотеки Numpy.

С целью определения наиболее эффективного подхода было проведено сравнение скоростей вычислений для циклов Python, ООП, Numpy массивов. Результаты теста представлены в виде графика (Рисунок 2).

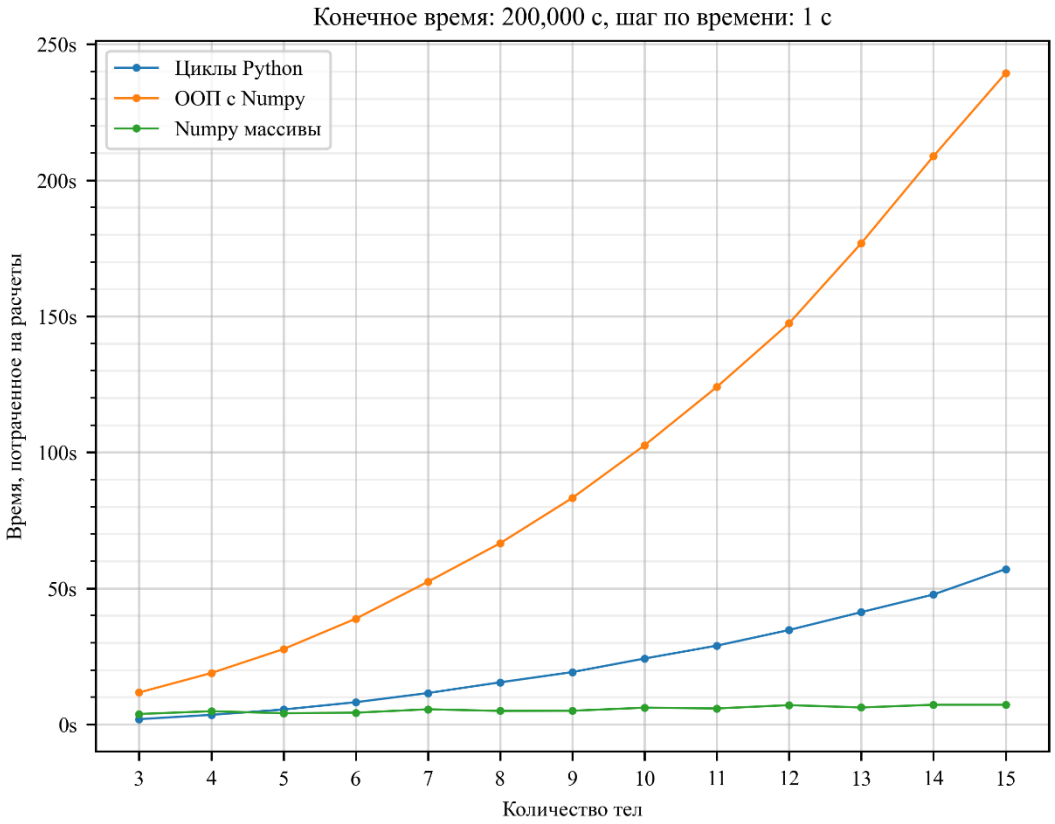


Рисунок 2 — График зависимости скорости различных алгоритмов от количества тел в симуляции.

[2] Git — это бесплатная распределенная система управления версиями с открытым исходным кодом, предназначенная для быстрой и эффективной обработки любых проектов: от небольших до очень крупных.

В тестах при использовании циклов Python (график Циклы Python) математические операции выполнялись над типом данных dict (словарь). При использовании ООП в Python (график ООП с Numpy), каждое тело рассматривалось как набор отдельных массивов данных Numpy. В другом случае (график Numpy массивы), вся система тел рассматривалась как несколько больших массивов данных Numpy, каждый из элементов которых является параметром единичного тела.

Результаты показали, что использование массивов данных Numpy обеспечивает наименьшее время вычислений при увеличении количества тел в симуляции, что обусловлено оптимизацией данной библиотеки для работы с большими массивами данных. В связи с этим в программе используется данный подход.

Был написан код для расчета координат, скоростей и ускорений тел. Все математические операции в коде производятся над массивами данных Numpy. Это позволяет ускорить процесс расчетов, а также улучшить читаемость кода.

В коде все операции производятся над объектом класса System, который обладает такими атрибутами, как массив масс тел, массив координат тел, массив скоростей тел, массив ускорений тел, конечное время расчета, шаг по времени и прочие параметры системы.

Код функции update_positions для расчета координат тел:

```
    def update_positions(self) -> None:
        self.positions += self.velocities * self.time_step
```

Код функции update_velocities для расчета скоростей тел:

```
    def update_velocities(self) -> None:
        self.velocities += self.accelerations * self.time_step
```

Код функции update_accelerations для расчета ускорений тел:

```
    def update_accelerations(self) -> None:
        distances = self.positions - self.positions[:, None]
        squared_norms = np.einsum('ijk->ij', distances ** 2)
        squared_norms[squared_norms == 0] = 1
        self.accelerations = np.einsum('ijk,ij,j->ik', distances, squared_norms
** -1.5, self.gravity_parameters)
```

В приведенном коде используется функция einsum библиотеки Numpy, позволяющая представлять простым способом многие распространенные операции с многомерными линейными алгебраическими массивами [4].

В коде вычисления параметров тел идут в следующем порядке:

1) Вычисление координат.
2) Вычисление ускорений.
3) Вычисление скоростей.

Более подробный процесс работы кода изложен в блок-схеме в Приложении 1.

Чтобы изучать и анализировать рассчитанные в программе результаты, необходимо было их визуализировать.

2.2 Визуализация

Библиотека Matplotlib была выбрана благодаря широкому спектру возможностей визуализации данных, включая создание статических, анимированных и интерактивных графиков, тесной интеграции с библиотекой Numpy и высокому качеству получаемых изображений.

Был создан отдельный файл rendering.py, который отвечает за визуализацию рассчитанных данных.

Используя Matplotlib и FFmpeg, программа может сохранять визуализации расчетов в виде файлов со следующими расширениями: gif, mp4, png, svg, jpg.

Программа может сохранять как видео, так и изображения.

С примерами визуализаций результатов можно ознакомиться в Приложениях 2 и 3.

2.3 Барицентр системы

Для расчета барицентра системы был создан класс Barycenter, обладающий всеми соответствующими параметрами.

Код функции calculate_mass для расчета массы барицентра:

```
    def calculate_mass(self, masses: np.ndarray) -> None:
        self.mass = np.einsum('i->', masses)
```

Код функции calculate_position для расчета координат барицентра:

```
    def calculate_position(self, positions: np.ndarray, masses: np.ndarray)
-> None:
        self.position = np.einsum('id,i->d', positions, masses) / self.mass
```

Код функции calculate_velocity для расчета скорости барицентра:

```
    def calculate_velocity(self, velocities: np.ndarray, masses: np.ndarray)
-> None:
        self.velocity = np.einsum('id,i->d', velocities, masses) / self.mass
```

В программе присутствует опция перехода в систему отсчета барицентра в начальный момент времени. Для этого из координат всех тел системы вычитаются координаты барицентра, аналогично происходит со скоростями, а сам барицентр переходит в начало координат и имеет в этой системе скорость, равную нулю:

```
    system.positions -= barycenter.position
system.velocities -= barycenter.velocity
barycenter.switch_reference_frame()
```

Переход в систему отсчета барицентра позволяет зафиксировать положение системы тел, так как ее импульс в этой системе отсчета будет равен нулю.

Абсолютная ошибка в вычислении координат барицентра находится как разность между конечным и начальным положением барицентра:

$$\Delta\vec{r} = \vec{r} - \vec{r}_0,$$

где $\vec{r}$ — радиус-вектор барицентра в конечный момент времени, $\vec{r}_0$ — радиус-вектор барицентра в начальный момент времени.

Относительная ошибка в вычислении координат барицентра находится как отношение абсолютной ошибки в вычислении координат барицентра к начальному размеру системы:

$$\varepsilon = \frac{\Delta\vec{r}}{systemsize},$$

где $\Delta\vec{r}$ — абсолютная ошибка в вычислении координат барицентра, $systemsize$ — начальный размер системы.

Начальный размер системы вычисляется как максимум от разностей максимума и минимума измерений радиус-векторов тел:

$$systemsize = \max_i \left(\max_{xyz} \vec{r}_i - \min_{xyz} \vec{r}_i \right)$$

Так как в симуляции отсутствует влияние внешних сил на тела системы, то барицентр должен двигаться прямолинейно и равномерно, в соответствии с законом сохранения импульса. Чтобы это проверить, была рассчитана траектория барицентра в конфигурации 2_stars.

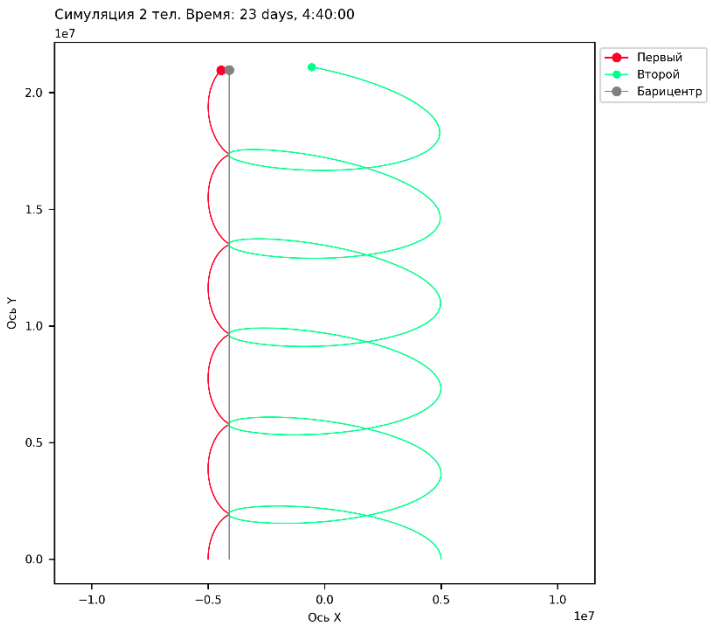


Рисунок 3 — Визуализация прямолинейного движения барицентра системы двух тел. Конфигурация — 2_stars.

Полученное из результатов прямолинейное движение барицентра системы тел подтверждает правильность расчетов программы (Рисунок 3).

2.4 Энергия системы

Был написан код для расчета различных энергий системы в соответствии с формулами, указанными в теоретической части работы.

Код функции calculate_kinetic_energy для расчета кинетической энергии:

```
    def calculate_kinetic_energy(self) -> None:
        self.kinetic_energy = np.einsum('id,i->', self.velocities ** 2,
self.masses / 2)
```

Код функции calculate_potential_energy для расчета потенциальной энергии:

```
    def calculate_potential_energy(self) -> None:
        distances = self.positions - self.positions[:, None]
        squared_norms = np.einsum('ijk->ij', distances ** 2)
        squared_norms[squared_norms == 0] = 1
        mass_matrix = np.triu(self.masses * self.masses[:, None], 1)
        self.potential_energy = -self.gravitational_constant * np.einsum('ij,ij-
>', mass_matrix, squared_norms ** -0.5)
```

Код функции calculate_total_energy для расчета полной механической энергии:

```
    def calculate_total_energy(self) -> None:
        self.total_energy = self.kinetic_energy + self.potential_energy
```

Абсолютная ошибка в вычислении полной механической энергии находится как разность между конечным и начальным значениями полной механической энергии:

$$\Delta E = E - E_0,$$

где E —значение полной механической энергии в конечный момент времени, E_0 — значение полной механической энергии в начальный момент времени.

Относительная ошибка в вычислении полной механической энергии находится как отношение абсолютной ошибки в вычислении полной механической энергии к начальному значению полной механической энергии:

$$\varepsilon = \frac{\Delta E}{E_0},$$

где ΔE —абсолютная ошибка в вычислении полной механической энергии, E_0 — значение полной механической энергии в начальный момент времени.

По значению полной механической энергии можно определить, произошла ли значительная ошибка в расчетах при сильном сближении тел.

Из-за слишком большой величины шага по времени в некоторых конфигурациях происходит «вылет» тел из системы при их сильном сближении.

Для определения ошибки в расчете механической энергии при «вылете» тела из системы была рассчитана конфигурация free_fall_1. Построен график зависимости различных энергий системы от времени в симуляции (Рисунок 4).

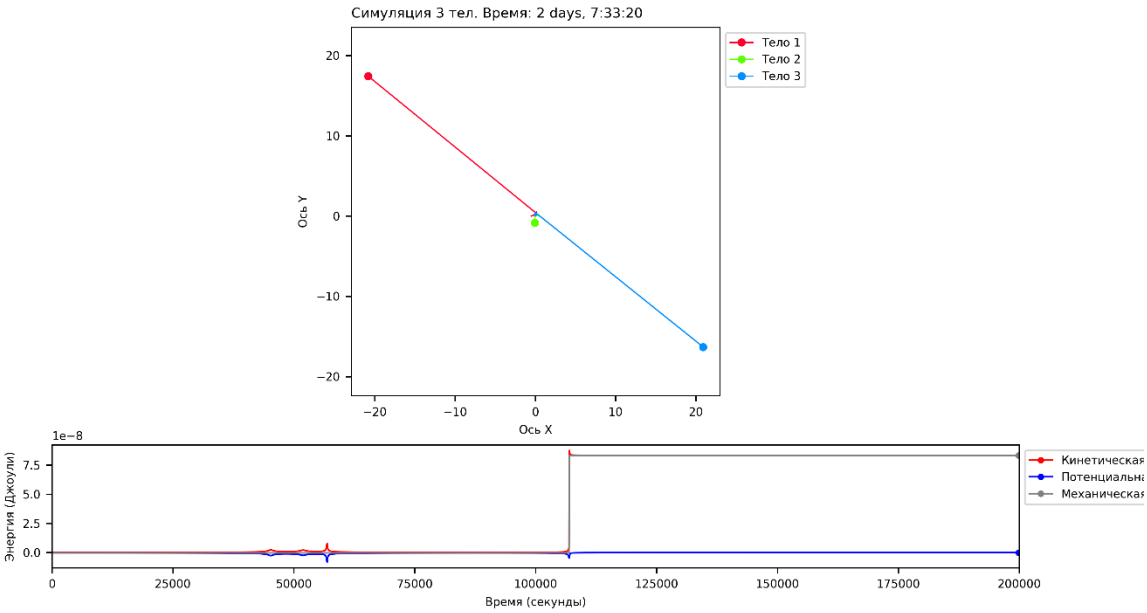


Рисунок 4 — Ошибочное изменение полной механической энергии в симуляции. Конфигурация — free_fall_1.

О «вылете» тела из системы свидетельствует сильное изменение полной механической энергии на графике. Для предотвращения резких изменений полной механической энергии можно уменьшить шаг по времени, однако при этом увеличится время, требуемое для расчетов. Другим способом решения данной проблемы является использование динамического шага по времени, который был введен далее в работе.

2.5 Небесная сфера

Для отображения траекторий тел на карте звездного неба, необходимо перевести декартовые координаты тел в сферические с учетом наклона оси вращения Земли к плоскости эклиптики.

Декартовые координаты тел в системе координат, в которой началом координат является тело с индексом POV_body_index находятся как разности координат тел и координат тела с индексом POV_body_index:

```
relative_positions = positions_array - positions_array[POV_body_index]
```

Далее находится эклиптическая широта и долгота для всех тел в соответствии с формулами:

Код расчета эклиптической долготы:

```
    ecliptic_longitude = np.arctan2(relative_positions[:, :, 1],
relative_positions[:, :, 0])
```

Код расчета эклиптической широты:

```
      square_root = np.sqrt(relative_positions[:, :, 0] ** 2 +
relative_positions[:, :, 1] ** 2)
ecliptic_latitude = np.arctan2(relative_positions[:, :, 2], square_root)
```

После этого находятся прямые восхождения и склонения тел на небесной сфере с учетом наклона оси вращения Земли:

Код расчета прямого восхождения:

```
      right_ascensions = np.arctan2(np.cos(ecliptic_obliquity) *
np.sin(ecliptic_longitude) - np.sin(ecliptic_obliquity) *
np.tan(ecliptic_latitude), np.cos(ecliptic_longitude))
```

Код расчета склонения:

```
            declinations = np.arcsin(np.cos(ecliptic_obliquity) *
      np.sin(ecliptic_latitude) + np.sin(ecliptic_obliquity) *
      np.cos(ecliptic_latitude) * np.sin(ecliptic_longitude))
```

После получения сферических координат тел, становится возможной их визуализация. Для представления небесной сферы на плоскости в разработанной программе предусмотрена реализация нескольких картографических проекций:

- Цилиндрическая равнопромежуточная проекция.
- Азимутальная проекция.
- Проекция Мольвейде.

Для создания этих проекций был использован набор стандартных проекций графиков библиотеки Matplotlib.

Для отображения звезд на карте неба использована библиотека Skyfield.

Данные для координат звезд получены из следующих звездных каталогов:

- *Hipparcos* содержит положения, собственные движения и тригонометрические параллаксы более чем 100 000 звезд.
- *Tycho* содержит свыше 1 миллиона звезд.
- *Tycho-2* содержит положения и собственные движения, а также двухполосную фотометрию 2.5 миллионов звезд. [6]

Данные для отображения астеризмов взяты из программы Stellarium — свободный виртуальный планетарий с открытым исходным кодом [7].

С помощью Matplotlib создана карта звездного неба, с которой можно ознакомиться в Приложении 4.

Реализовано преобразование и отображение координат небесных тел на карте звездного неба. Данные результаты будут использованы в дальнейших исследованиях.

2.6 Динамический шаг по времени

В связи с разнообразием моделируемых в программе систем, для минимизации погрешности численного интегрирования требуется индивидуальный подбор шага по

времени для каждой системы. В общем случае был реализован расчет шага по времени на каждой итерации моделирования. Такой шаг называется динамическим.

Динамический шаг по времени Δt рассчитывается по формуле [9]:

$$\Delta t = \mu \min_i \left(\frac{|\vec{a}_i|}{|\vec{j}_i|} \right),$$

где μ — коэффициент пропорциональности, $\vec{a}_i$ — модуль ускорения i-го тела системы, $\vec{j}_i$ — модуль рывка (производная от ускорения) i-го тела системы.

Для предотвращения принятия шагом по времени экстремальных значений, его величина ограничивается диапазоном $[\Delta t_{min};\ \Delta t_{max}]$:

$$\Delta t_{min} \leq \Delta t \leq \Delta t_{max},$$

где Δt_{max} должен быть меньше периода записи данных в файлы, во избежание пропуска записи данных.

Из данной формулы следует, что шаг по времени остается неизменным, когда отношение ускорения к изменению ускорения обратно коэффициенту пропорциональности μ:

$$\Delta t_1 = \mu \frac{a}{\frac{\Delta a}{\Delta t_0}} = \mu \frac{a \Delta t_0}{\Delta a}$$

$$\Delta t_1 = \Delta t_0 \Rightarrow \mu \frac{a}{\Delta a} = 1 \Rightarrow \mu = \frac{\Delta a}{a},$$

где Δa — изменение ускорения за шаг, a — конечное ускорение, Δt_0 — начальный шаг по времени, Δt_1 — конечный шаг по времени.

Для определения оптимального значения коэффициента пропорциональности μ был построен график зависимости относительной ошибки в расчете полной механической энергии системы от шага по времени Δt для конфигурации model_solar_system (Рисунок 5).

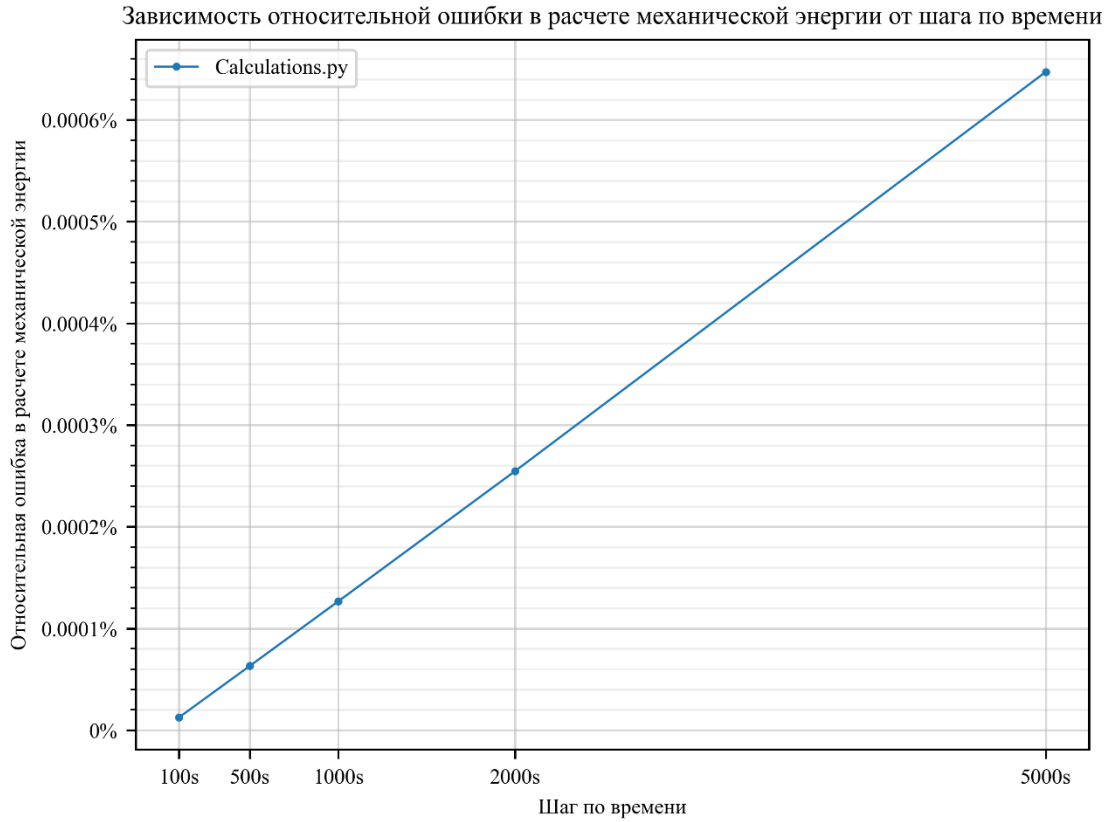


Рисунок 5 — График зависимости относительной ошибки в расчете полной механической энергии системы от шага по времени.

При анализе графика была выявлена линейная зависимость относительной ошибки от шага по времени. Для обеспечения приемлемой точности для большинства систем, установленной на уровне 0.0001% при шаге по времени приблизительно в 1000 секунд был определен коэффициент пропорциональности μ. Для этого было рассчитано среднее отношение изменения ускорения Δa к ускорению a в конфигурации model_solar_system. В результате расчетов было получено следующее отношение:

$$\mu = \frac{\Delta a}{a} = 0.00016547004635007496 \approx 0.0002$$

Отсюда следует значение коэффициента μ = 0.0002. Данное значение было внесено в программу.

Для оценки эффективности использования динамического шага было проведено сравнение точности и производительности вычислений с использованием постоянного и динамического шага для конфигурации free_fall_2, в которой наблюдается значительное сближение тел. Результаты измерений представлены в виде столбчатой диаграммы (Рисунок 6).

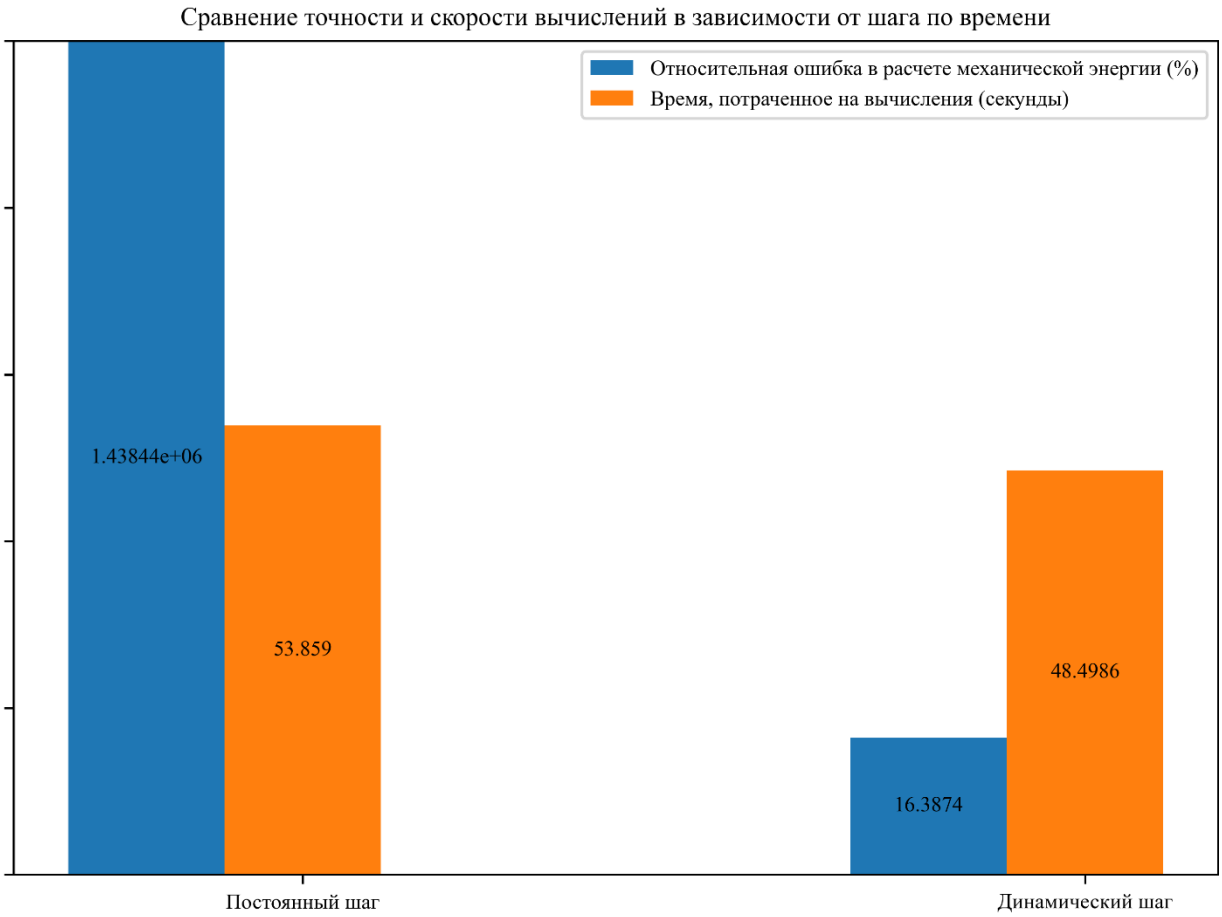


Рисунок 6 — Столбчатая диаграмма результатов измерения точности и скорости вычислений в зависимости от шага по времени.

Результаты, представленные на рисунке 6 демонстрируют, что для некоторых конфигураций использование динамического шага по времени позволяет достичь более высокой точности при сопоставимом времени вычислений по сравнению с постоянным шагом.

2.7 Проверка результатов расчетов программы

После написания программы и подготовки условий для проведения тестов, была проведена проверка точности расчётов.

1. *Проверка с помощью частного случая задачи N тел*

В качестве тестового случая, допускающего аналитическое решение, рассмотрено движение 3 тел по окружности. Начальные условия были заданы на основе выведенных ранее формул. Визуализация результатов моделирования представлена на рисунке 7.

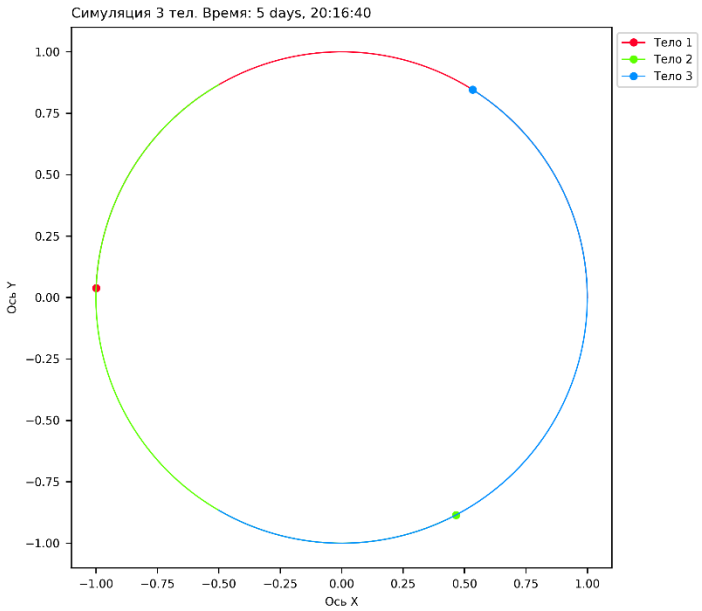


Рисунок 7 — Рассчитанное в программе движение трех тел по окружности. Конфигурация — 3_body_circle.

Результаты показали устойчивое движение 3 тел по окружности в течение 73 моделируемых дней, что свидетельствует о корректности расчета гравитационного взаимодействия в программе.

2. *Проверка с помощью движения кометы C2023/A3 (Цзыцзиньшань - ATLAS)*

Для определения точности расчетов программы на реальном примере было использовано движение кометы C/2023 A3 (Цзыцзиньшань - ATLAS).

Исходные данные о положении планет и кометы в заданную дату были получены из системы вычислений данных и эфемерид солнечной системы JPL Horizons [8].

Моделирование движения кометы проводилось в период с 2022 года по 2028 год. Полученная траектория отображена на карте звездного неба (Рисунок 8). Рассчитанная траектория совпадает с реальной (Рисунок 9).

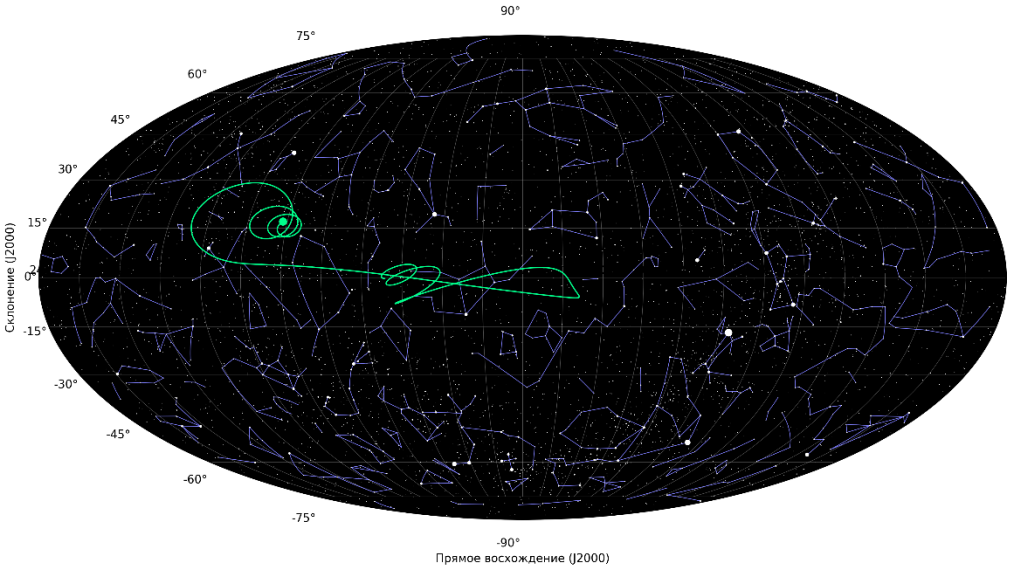


Рисунок 8 — Рассчитанная в программе траектория кометы C/2023 A3 (Цзыцзиньшань - ATLAS) на карте звездного неба.

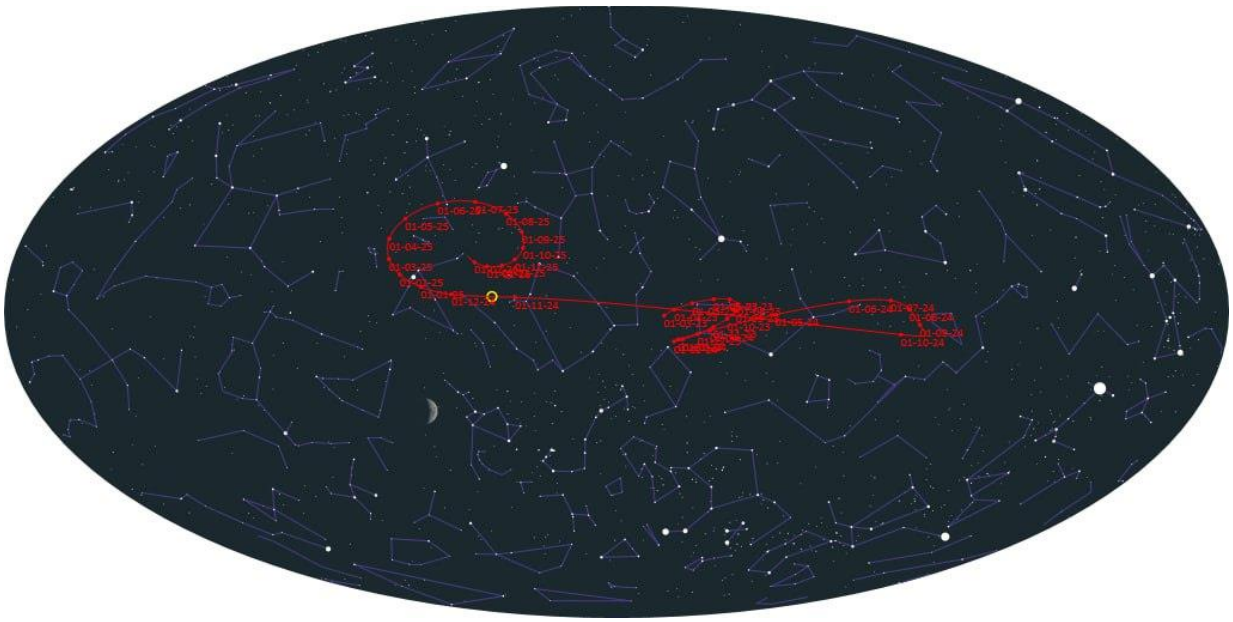


Рисунок 9 — Реальная траектория кометы C/2023 A3 (Цзыцзиньшань - ATLAS) на карте неба.

Для более точной проверки расчетов было проведено моделирование движения кометы в солнечной системе с мая по октябрь 2024 года с постоянным шагом по времени в 1 секунду. Результаты были сопоставлены с фотографиями, полученными при помощи наблюдений в телескоп в астрономической обсерватории ГМИК им. К.Э. Циолковского г. Калуги. Сопоставление проводилось путем сравнения положения кометы относительно опорных звезд на карте неба и на полученных фотографиях.

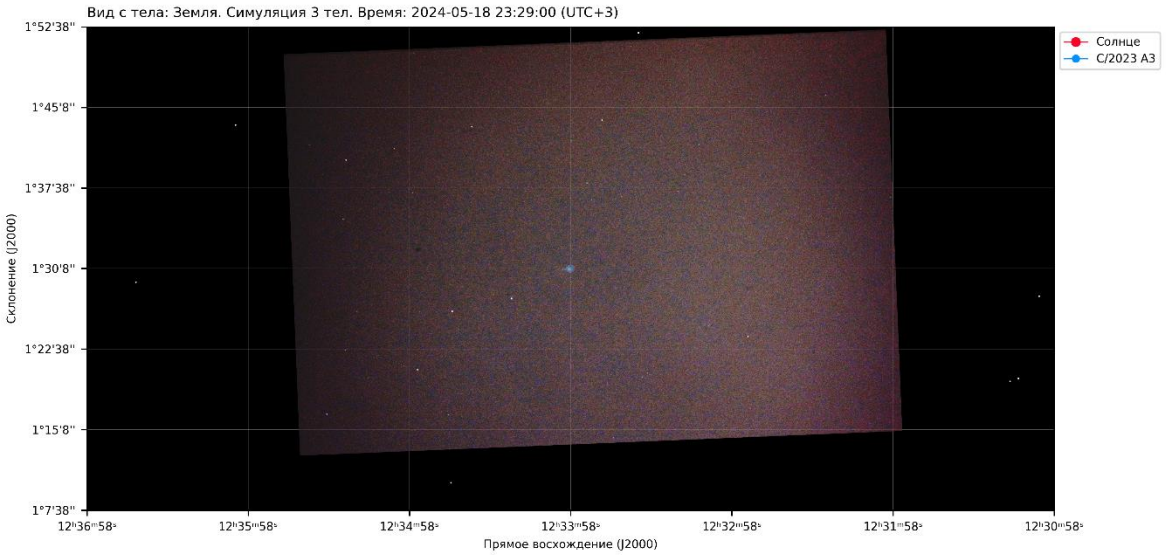


Рисунок 10 — Сопоставление положений кометы из расчётов и на фотографии 18 мая 23:29 2024 года.

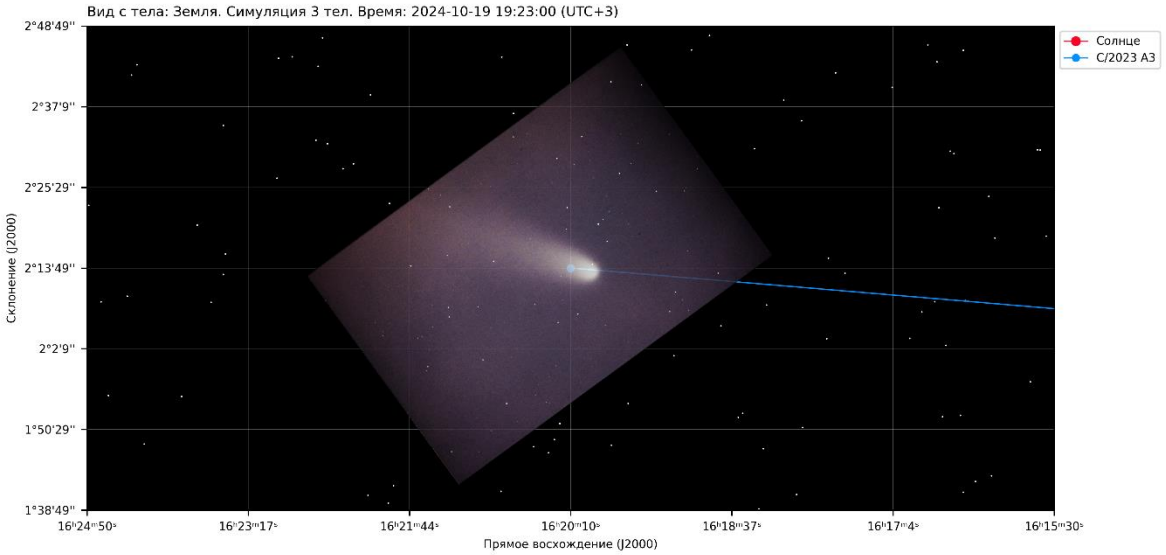


Рисунок 11 — Сопоставление положений кометы из расчётов и на фотографии 19 октября 19:23 2024 года.

На начальном этапе, 18 мая, расхождение между заданным и наблюдаемым положением кометы составляло около 1 часовой секунды, что свидетельствует о высокой точности начальных данных (Рисунок 10).

В октябре отклонение достигло 9 часовых секунд (Рисунок 11). Для увеличения точности расчетов, можно уменьшить шаг по времени или уточнить начальные данные.

Текущая погрешность в вычислениях является приемлемой, поскольку предсказанное положение кометы попадает в поле зрения телескопа относительно истинного, что позволяет использовать программу для практических целей.

ЗАКЛЮЧЕНИЕ

С использованием численного метода Эйлера, был реализован расчет траекторий тел на языке Python, а также визуализация результатов моделирования с использованием библиотеки Matplotlib.

Реализован расчет барицентра системы тел. Для анализа точности определения его положения реализованы методы расчёта абсолютной и относительной ошибок в вычислении координат барицентра. Анализ движения барицентра системы показал, что при достаточной величине шага по времени, барицентр движется прямолинейно и равномерно, в соответствии с законом сохранения импульса.

При изучении вычисления кинетической, потенциальной и полной механической энергии в программе, было выявлено, что при приемлемом шаге по времени полная механическая энергия системы остается неизменной, следовательно, в моделируемой симуляции соблюдается закон сохранения механической энергии.

Использование динамического шага по времени для некоторых систем позволило достичь увеличения точности моделирования без существенного увеличения времени расчёта. Было вычислено значение коэффициента $\mu = 0.0002$, который используется при расчете динамического шага по времени.

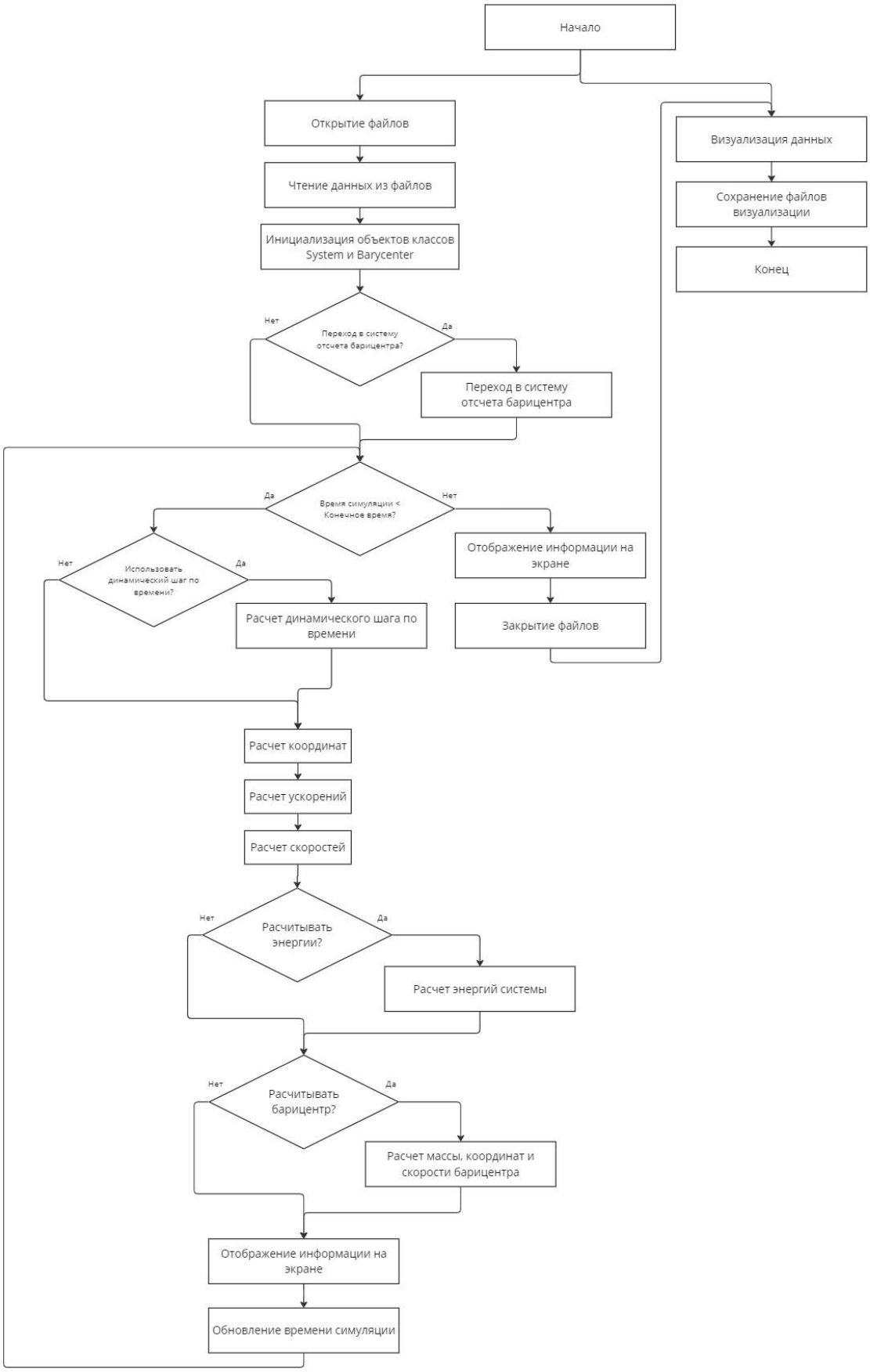
Разработанная программа успешно протестирована и демонстрирует корректные результаты при моделировании различных гравитационных систем, включая движение кометы C/2023 A3 в солнечной системе.

Проведенные тесты подтвердили правильность корректность реализованных алгоритмов и применимость программы для астрономических исследований.

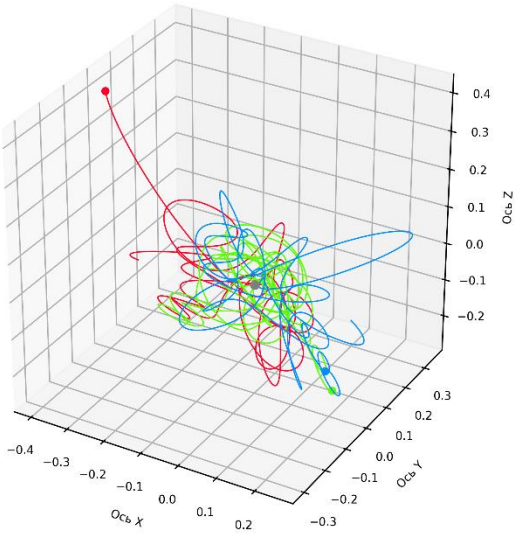
СПИСОК ЛИТЕРАТУРЫ

1. Справочно-информационный интернет-портал «Википедия»: Гравитационная задача N тел. URL: https://ru.wikipedia.org/wiki/Гравитационная_задача_N_тел (дата обращения: 02.12.2024).
2. Справочно-информационный интернет-портал «Википедия»: Задача Коши. URL: https://ru.wikipedia.org/wiki/Задача_Коши (дата обращения: 02.12.2024).
3. Метод Эйлера. Математический энциклопедический словарь. — М.: «Сов. энциклопедия», 1988. — С. 641. URL: https://www.mathedu.ru/text/matematicheskiy_entsiklopedicheskiy_slovar_1988/p641/ (дата обращения: 02.12.2024).
4. NumPy documentation. Version: 2.1. URL: https://numpy.org/doc/2.1/ (дата обращения: 02.12.2024).
5. Matplotlib 3.9.3 documentation. URL: https://matplotlib.org/stable/index.html (дата обращения: 02.12.2024).
6. Цветков А.С. Руководство по работе с каталогом Tycho-2: учебно-метод. пособие. – СПб., 2005. – 132 с.
7. Zotti, G., Hoffmann, S. M., Wolf, A., Chéreau, F., & Chéreau, G. (2021). The Simulated Sky: Stellarium for Cultural Astronomy Research. Journal of Skyscape Archaeology, 6(2), 221–258. DOI: 10.1558/jsa.17822 (дата обращения: 02.12.2024).
8. Онлайн-сервис вычислений данных и эфемерид солнечной системы JPL Horizons: Horizons System – NASA. URL: https://ssd.jpl.nasa.gov/horizons/app.html (дата обращения: 02.12.2024).
9. A new timestep criterion for N-body simulations // arXiv. URL: https://arxiv.org/html/2401.02849v1 (дата обращения: 02.12.2024).

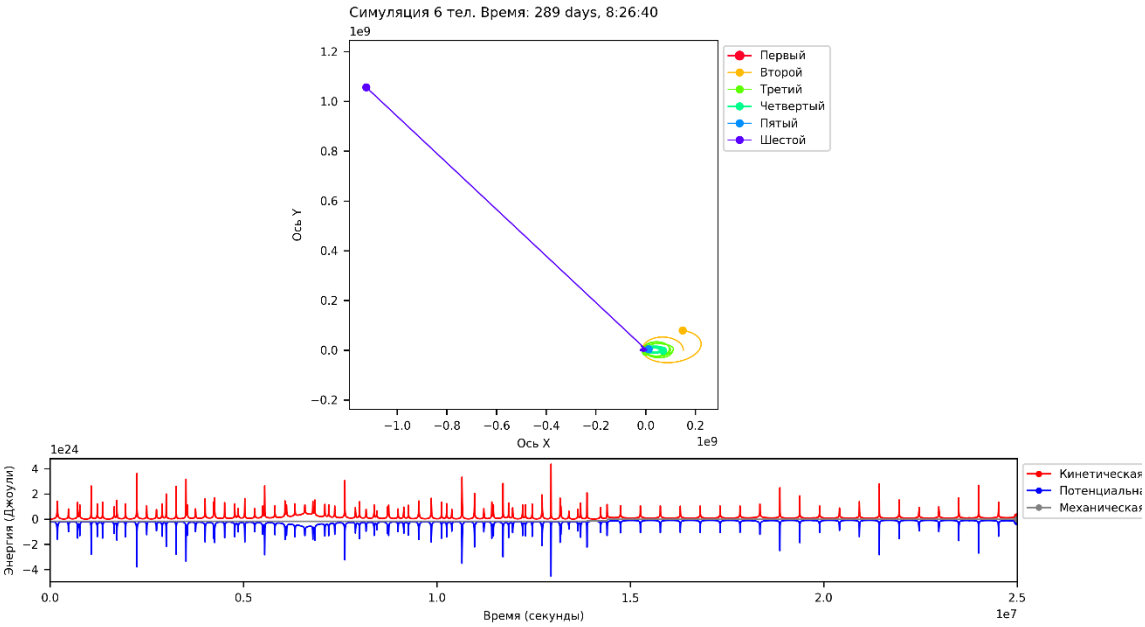
ПРИЛОЖЕНИЯ



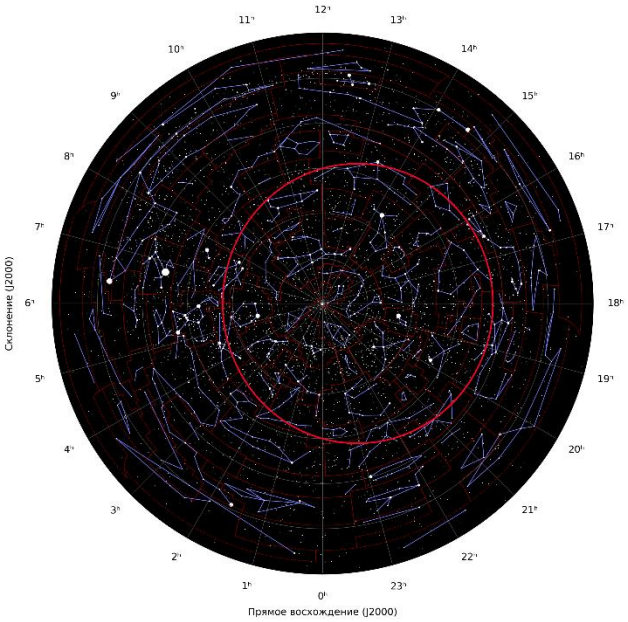
Приложение 1. Блок-схема исходного кода программы.



Приложение 2. Визуализация результатов расчетов программы.



Приложение 3. Визуализация результатов расчетов программы.



Приложение 4. Карта звездного неба, созданная с помощью Matplotlib.