Санкт-Петербургский национальный исследовательский
Академический университет имени Ж.И.Алферова
Российской академии наук

Отчет по проекту по курсу «Python»

Решение гравитационной задачи N-тел: сравнение методов численного интегрирования

Группа: 101.2

Преподаватель: Мсукар С.

Студент: Черепанов Кирилл

Санкт-Петербург
12 декабря 2025 г.

1 Аннотация

Данная работа посвящена численному решению гравитационной задачи N-тел и сравнительному анализу эффективности двух методов численного интегрирования: метода Рунге-Кутты 4-го порядка (RK4) и метода Leapfrog. В рамках исследования была разработана программная система на языке Python для моделирования динамики гравитационного взаимодействия небесных тел с использованием реальных данных Солнечной системы.

Проект включает реализацию вычислительного ядра для расчета гравитационных сил между множественными телами и численное интегрирование уравнений движения с применением двух различных алгоритмов. Проведено тестирование на системах различной сложности: от простой двухтельной системы Земля-Солнце до полной планетной системы с восемью и более телами.

Выполнен комплексный сравнительный анализ методов по следующим критериям: точность сохранения полной энергии системы при долгосрочном моделировании, зависимость погрешности от количества тел в системе, вычислительная производительность и масштабируемость, численная стабильность при различных шагах интегрирования по времени.

Результаты исследования показывают преимущества и недостатки каждого метода для различных сценариев применения и позволяют сформулировать рекомендации по выбору оптимального численного подхода в зависимости от требований к краткосрочной точности или долгосрочной стабильности моделирования. Программный код проекта организован в соответствии с современными практиками разработки программного обеспечения и размещен в публичном репозитории GitHub.

2 Методы численного интегрирования

2.1 Постановка задачи

Гравитационная задача N тел описывается системой обыкновенных дифференциальных уравнений второго порядка. Для i-го тела с массой m_i уравнения движения имеют вид:

$$\frac{d^2\mathbf{r}_i}{dt^2} = \mathbf{a}_i = \sum_{j \neq i}^{N} \frac{Gm_j(\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^3} \tag{1}$$

где $\mathbf{r}_i$ — радиус-вектор i-го тела, G — гравитационная постоянная. Для численного решения систему преобразуют в систему первого порядка:

$$\begin{cases} \frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i \\ \frac{d\mathbf{v}_i}{dt} = \mathbf{a}_i(\mathbf{r}_1, \ldots, \mathbf{r}_N) \end{cases} \tag{2}$$

2.2 Явный метод Эйлера

Простейший численный метод решения ОДУ. Для уравнения $\frac{dy}{dt} = f(t, y)$ имеет вид:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{3}$$

Для задачи N тел:

$$\begin{cases} \mathbf{r}_i^{n+1} = \mathbf{r}_i^n + h \cdot \mathbf{v}_i^n \\ \mathbf{v}_i^{n+1} = \mathbf{v}_i^n + h \cdot \mathbf{a}_i^n \end{cases} \tag{4}$$

Основные характеристики:

- Точность: $O(h)$
- Простая реализация
- Не сохраняет энергию системы
- Непригоден для долгосрочного моделирования

2.3 Метод Рунге-Кутты 4-го порядка (RK4)

Один из наиболее популярных методов, обеспечивающий высокую точность. Для уравнения $\frac{dy}{dt} = f(t, y)$:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5}$$

где:

$$k_1 = f(t_n, y_n) \tag{6}$$
$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \tag{7}$$
$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \tag{8}$$
$$k_4 = f(t_n + h, y_n + hk_3) \tag{9}$$

Для задачи N тел метод применяется к состоянию системы $\mathbf{y} = (\mathbf{r}_1, \ldots, \mathbf{r}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N)$.
Основные характеристики:

- Точность: $O(h^4)$
- Требует 4 вычисления ускорений на шаг
- Высокая точность на коротких интервалах
- Не является симплектическим

2.4 Метод Leapfrog

Симплектический интегратор второго порядка. Координаты и скорости вычисляются в чередующиеся моменты времени (схема kick-drift-kick):

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{h}{2}\mathbf{a}_i^n \tag{10}$$
$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + h\mathbf{v}_i^{n+1/2} \tag{11}$$
$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^{n+1/2} + \frac{h}{2}\mathbf{a}_i^{n+1} \tag{12}$$

Основные характеристики:

- Точность: $O(h^2)$
- Симплектичность — отличное сохранение энергии
- Требует 2 вычисления ускорений на шаг
- Идеален для долгосрочных симуляций

2.5 Сравнение методов

Таблица 1 – Сравнительные характеристики методов

Характеристика	**Эйлер**	**RK4**	**Leapfrog**
Порядок точности	1	4	2
Вычислений ускорений	1	4	2
Симплектичность	Нет	Нет	Да
Сохранение энергии	Плохое	Хорошее	Отличное
Долгосрочная стабильность	Низкая	Средняя	Высокая

Вычислительная сложность одного шага для системы из N тел составляет $O(N^2)$ для всех методов, однако метод Leapfrog требует в 2 раза меньше вычислений ускорений по сравнению с RK4.

3 Описание разработанной программы

3.1 Общая архитектура

Разработанная программная система для решения задачи N тел реализована на языке Python и имеет модульную архитектуру, обеспечивающую расширяемость и удобство использования. Проект организован в соответствии с принципами объектно-ориентированного программирования и разделён на следующие основные модули:

- `core` — базовые структуры данных и физические константы
- `physics` — расчёт гравитационных сил и энергии
- `integrators` — численные методы интегрирования
- `simulation` — управление симуляцией
- `visualization` — визуализация результатов
- `examples` — примеры использования

3.2 Модуль `core`

Модуль содержит фундаментальные компоненты системы:

ParticleData — класс данных для хранения состояния системы частиц, включающий:

- `masses`: массы частиц, массив формы $(n, 1)$
- `positions`: координаты частиц, массив формы $(n, 3)$

- `velocities`: скорости частиц, массив формы $(n, 3)$
- `forces`: силы, действующие на частицы, массив формы $(n, 3)$

Класс обеспечивает валидацию данных при инициализации и предоставляет метод копирования состояния системы.

Константы:

- `G = 6.6743e-11` — гравитационная постоянная [$м^3/(кг \cdot с^2)$]
- `AU = 1.496e11` — астрономическая единица [м]

3.3 Модуль `physics`

Модуль отвечает за расчёт физических взаимодействий в системе.

ForceCalculator — абстрактный базовый класс, определяющий интерфейс для вычисления сил:

- `calc_force(data)` — расчёт гравитационных сил
- `calc_potential_energy(data)` — расчёт потенциальной энергии

DirectForceCalculator — реализация прямого метода расчёта сил. Для каждой пары частиц i и j вычисляется гравитационная сила:

$$\mathbf{F}_{ij} = \frac{Gm_i m_j}{|\mathbf{r}_j - \mathbf{r}_i|^3}(\mathbf{r}_j - \mathbf{r}_i) \tag{13}$$

Вычислительная сложность составляет $O(N^2)$, где N — количество частиц.

Потенциальная энергия системы вычисляется как:

$$U = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\frac{Gm_i m_j}{|\mathbf{r}_j - \mathbf{r}_i|} \tag{14}$$

3.4 Модуль `integrators`

Модуль содержит реализацию трёх численных методов интегрирования.

Integrator — абстрактный базовый класс с методом `step(system, dt)`, выполняющим один шаг интегрирования.

ExplicitEulerIntegrator — явный метод Эйлера:

$$\mathbf{r}^{n+1} = \mathbf{r}^n + h\mathbf{v}^n \tag{15}$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + h\mathbf{a}^n \tag{16}$$

RK4Integrator — метод Рунге-Кутты 4-го порядка. Реализация выполняет четыре промежуточных вычисления ускорений (k_1, k_2, k_3, k_4) и обновляет состояние системы взвешенной суммой.

LFIntegrator — метод Leapfrog в схеме kick-drift-kick:

1. **Kick 1:** $\mathbf{v}^{n+1/2} = \mathbf{v}^n + \frac{h}{2}\mathbf{a}^n$
2. **Drift:** $\mathbf{r}^{n+1} = \mathbf{r}^n + h\mathbf{v}^{n+1/2}$
3. **Kick 2:** $\mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} + \frac{h}{2}\mathbf{a}^{n+1}$

3.5 Модуль `simulation`

Модуль управляет процессом симуляции и объединяет физику с численными методами.

ParticleSystem — класс, представляющий физическую систему частиц:

- `calc_forces()` — обновление сил через `ForceCalculator`
- `calc_kinetic_energy()` — расчёт кинетической энергии: $K = \sum_{i=1}^{N} \frac{m_i v_i^2}{2}$
- `calc_potential_energy()` — расчёт потенциальной энергии
- `calc_total_energy()` — расчёт полной энергии: $E = K + U$

SimulationParameters — структура данных, хранящая параметры симуляции:

- `force_calculator` — объект расчёта сил
- `integrator` — численный интегратор
- `dt` — шаг интегрирования по времени

Simulation — главный класс симуляции:

- `step()` — выполнение одного временного шага
- `get_data()` — получение текущего состояния системы
- `time` — текущее время симуляции

3.6 Модуль `visualization`

Модуль обеспечивает визуализацию результатов симуляции с использованием библиотеки Matplotlib.

Animator2D — класс для создания анимации движения частиц в 2D:

- Отображение траекторий частиц с настраиваемой длиной следа
- Динамическое обновление позиций в реальном времени
- Поддержка тёмной и светлой цветовых схем
- Настраиваемые цвета, размеры и имена частиц
- Отображение текущего времени симуляции
- Сохранение анимации в видеофайл (MP4, GIF)

Метод `visualize(interval, repeat, show)` запускает анимацию с заданным интервалом обновления кадров.

Plotter2D — класс для построения статических траекторий:

- Отображение полных траекторий всех частиц
- Маркировка начальных и конечных позиций
- Сохранение графиков в высоком разрешении

EnergyPlotter — класс для визуализации энергии системы:

- График кинетической, потенциальной и полной энергии
- График относительной ошибки сохранения энергии в логарифмическом масштабе
- Поддержка научной нотации для больших значений

Относительная ошибка вычисляется как:

$$\varepsilon(t) = \left| \frac{E(t) - E(0)}{E(0)} \right| \times 100\% \tag{17}$$

3.7 Примеры использования

Разработано три демонстрационных примера:
earth_sun_orbit.py — моделирование орбиты Земли вокруг Солнца:

- Система из 2 тел
- Период симуляции: 1 год (8766 шагов по 1 часу)
- Использование интегратора Leapfrog
- Создание анимации орбитального движения

solar_system.py — полная модель Солнечной системы:

- Система из 9 тел (Солнце + 8 планет)
- Период симуляции: 10 лет
- Шаг интегрирования: 1 день (86400 с)
- Визуализация траекторий всех планет
- Анализ сохранения энергии системы

energy_conservation_test.py — сравнительный тест интеграторов:

- Запуск идентичной симуляции с тремя методами
- Расчёт статистики ошибок сохранения энергии
- Построение сравнительных графиков
- Количественная оценка точности методов

3.8 Технические особенности реализации

Использование NumPy: Все вычисления выполняются с использованием векторизованных операций библиотеки NumPy, что обеспечивает высокую производительность.

Типизация: Применяется аннотация типов Python (type hints) для повышения читаемости кода и раннего обнаружения ошибок.

Модульность: Модульная архитектура позволяет легко добавлять новые интеграторы, методы расчёта сил и визуализаторы без изменения существующего кода.

Валидация данных: Класс `ParticleData` автоматически проверяет корректность размерностей массивов при инициализации.

4 Тестирование и сравнение методов

Тестирование производится на основе эфемерид ИПА РАН (Планетная теория EPM2021, начальная дата 2025-12-13 04:42:28 UTC).

4.1 Задача двух тел: Солнце - Земля

Проведём тестирование приложения на примере системы Солнце — Земля. Общее время симуляции: 1 год. Общий временной шаг: 36000 секунд.

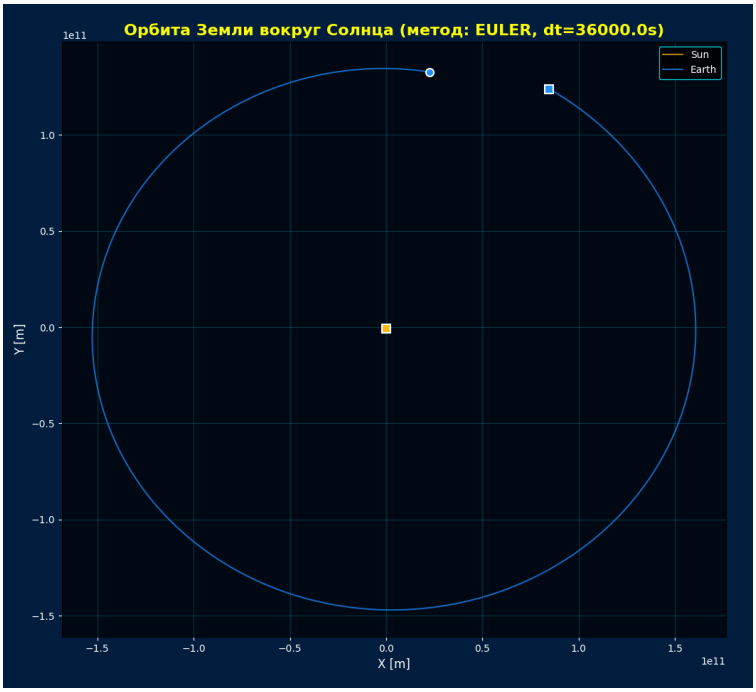

Рис. 1 – Солнце — Земля, явный метод Эйлера

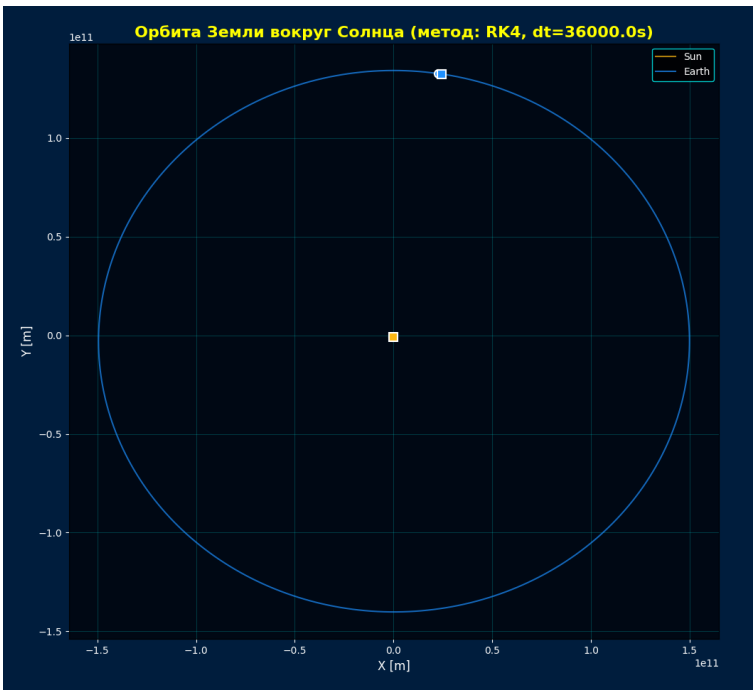

Рис. 2 – Солнце — Земля, метод Рунге-Кутты 4 порядка

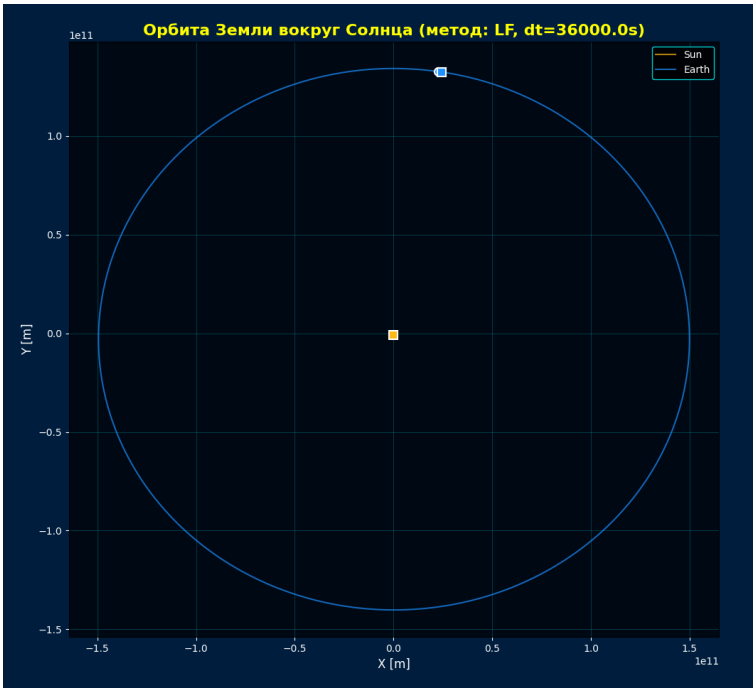

Рис. 3 – Солнце — Земля, Leap–Frog

Из полученных траекторий видно, что при данном временном шаге явный метод Эйлера даёт существенную ошибку, в то время как RK4 и LF сходятся друг с другом.

4.2 Внутренняя Солнечная система

Проведём тестирование приложения на примере внутренней Солнечной системы (Солнце—Меркурий—Венера—Земля—Марс). Общее время симуляции: 1 год. Общий временной шаг: 36000 секунд.

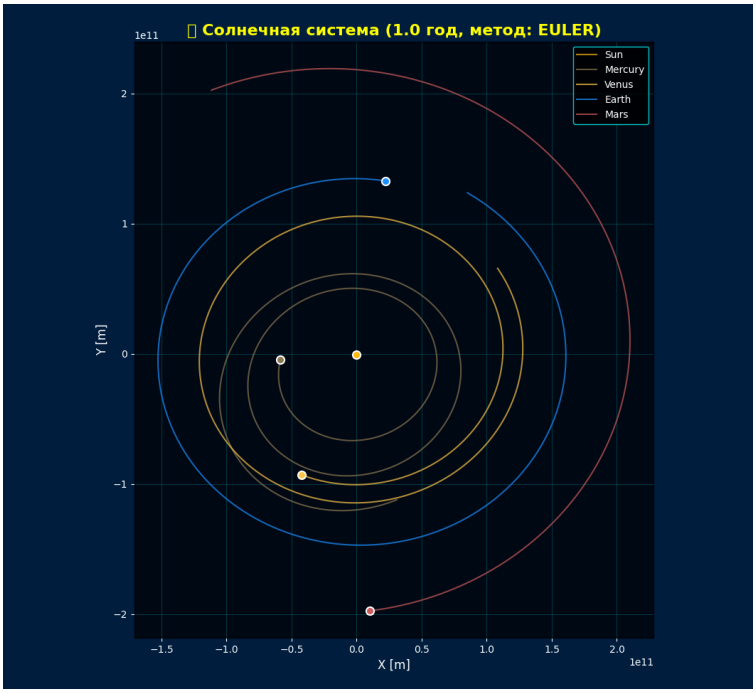

Рис. 4 – Внутренняя Солнечная система, явный метод Эйлера

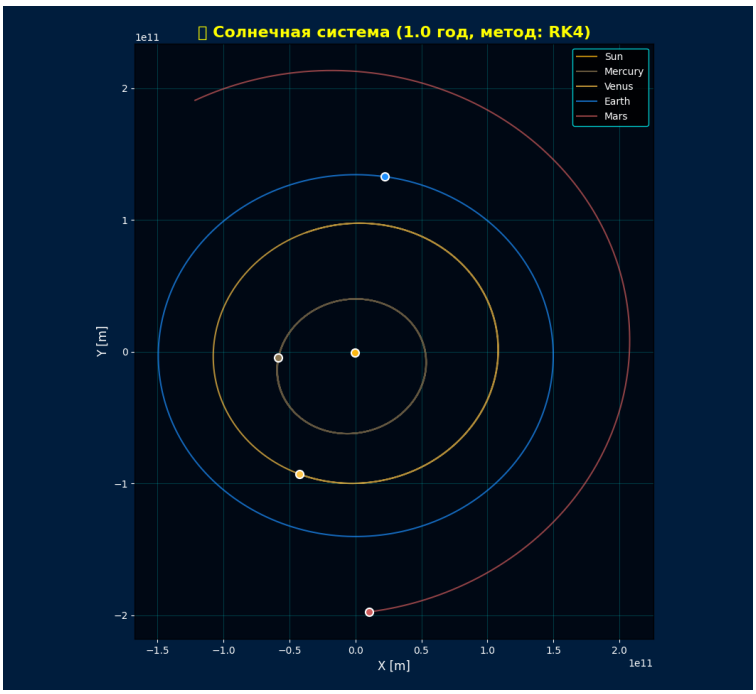

Рис. 5 – Внутренняя Солнечная система, метод Рунге-Кутты 4 порядка

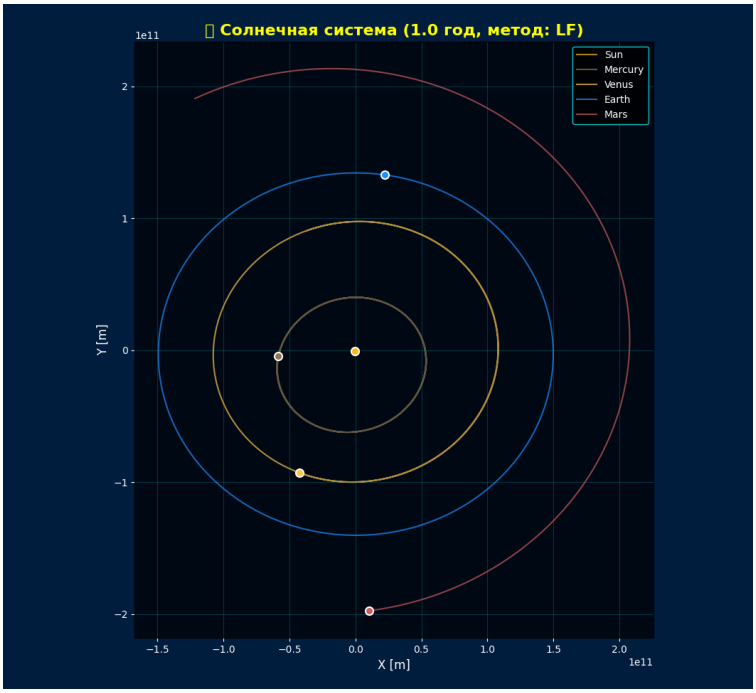

Рис. 6 – Внутренняя Солнечная система, Leap–Frog

Из-за наибольшего ускорения, действующего на Меркурий, его траектория при расчёте явным методом Эйлера искажается сильнее траекторий остальных тел.

4.3 Полная солнечная система

Проведём тестирование приложения на примере Солнечной системы, включая Плутон. Общее время симуляции: 248 лет (период Плутона). Общий временной шаг: 86400 секунд.

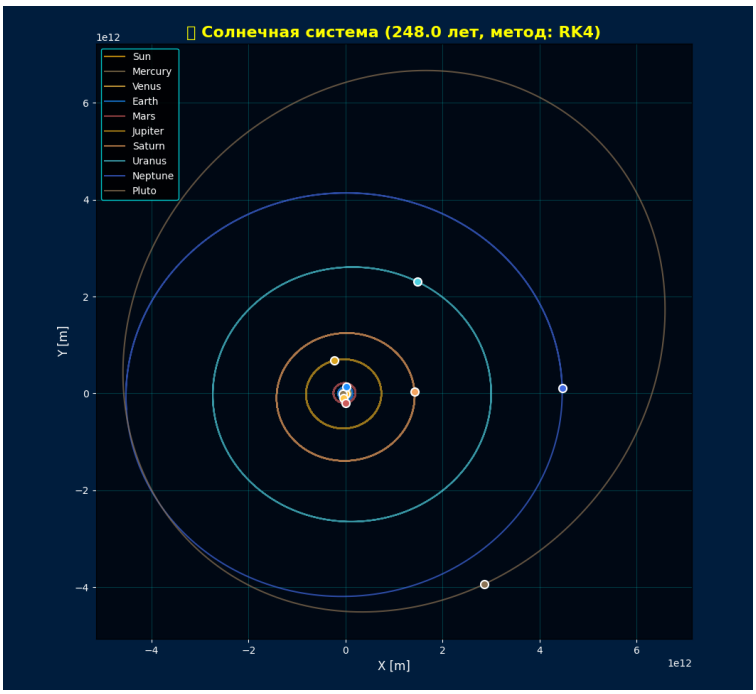

Рис. 7 – Солнечная система, метод Рунге-Кутты 4 порядка

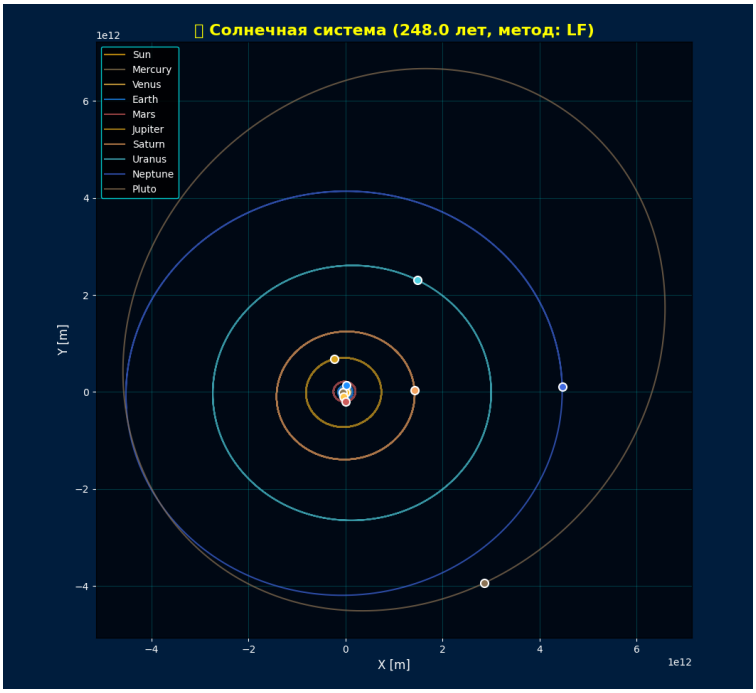

Рис. 8 – Солнечная система, Leap–Frog

Точность симуляции практически не меняется при росте числа тел Солнечной системы из-за незначительности взаимодействия между планетами.

4.4 Исследование сохранения энергии

На примере полной Солнечной системы исследуем сохранение методами энергии с ходом симуляции. Общее время симуляции: 2480 лет. Общий временной шаг: 86400 секунд.

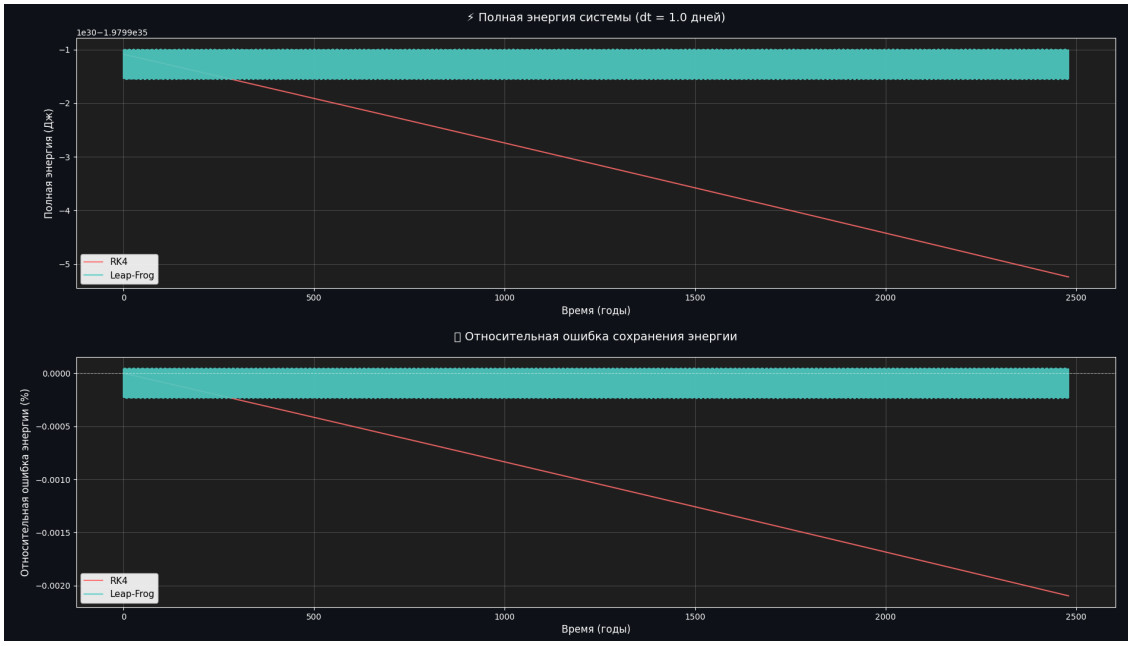

Рис. 9 – Солнечная система, полная энергия, 2480 лет

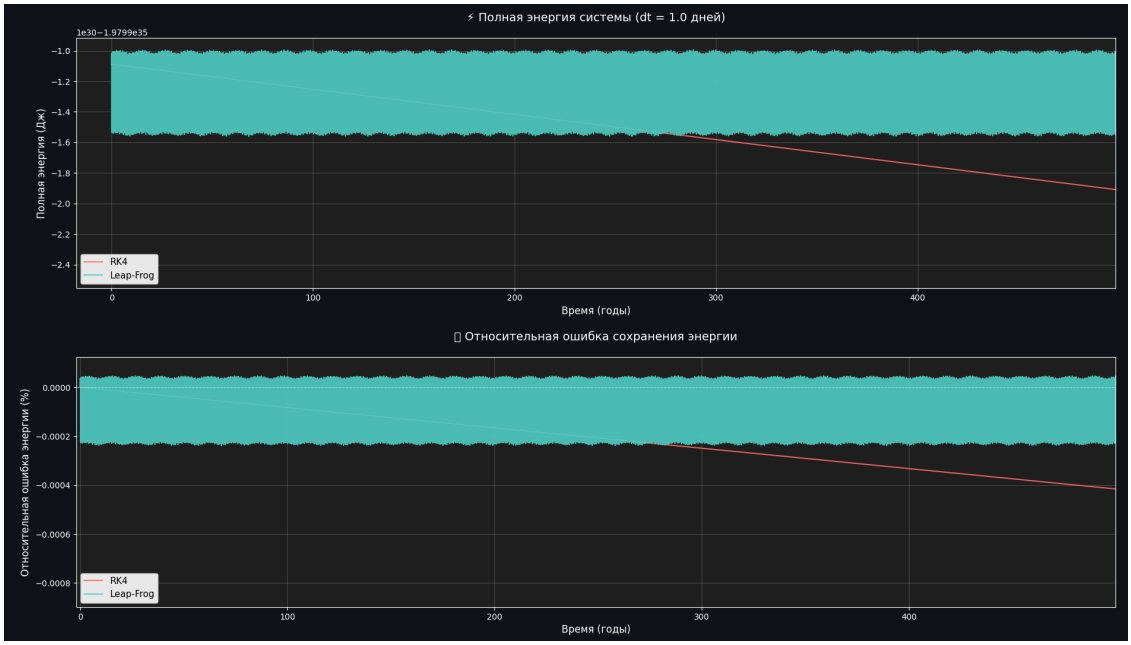

Рис. 10 – Солнечная система, полная энергия, 500 лет

Из графика видно, что RK4 сохраняет энергию лучше чем LF в краткосрочной перспективе (менее 300 лет).

LF не так точен в краткосрочной перспективе, однако на больших промежутках времени он сохраняет энергию лучше РК4.

4.5 Тесты производительности

Проведём тесты производительности для двух методов интегрирования. Результаты представим на графиках.

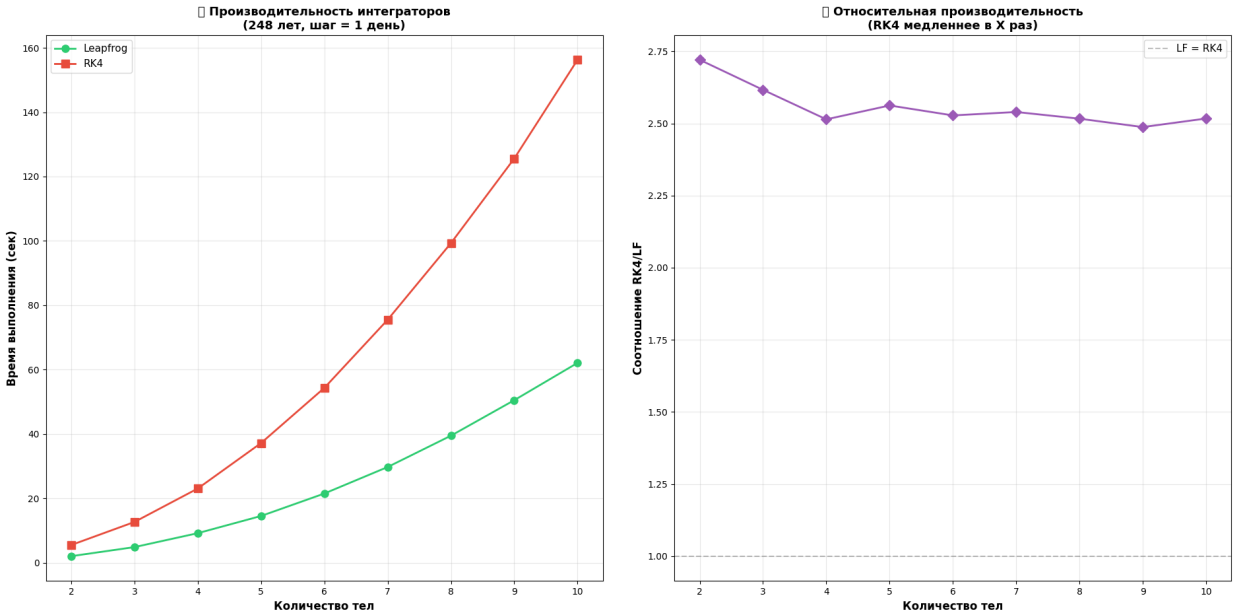

Рис. 11 – Зависимость времени симуляции от количества тел

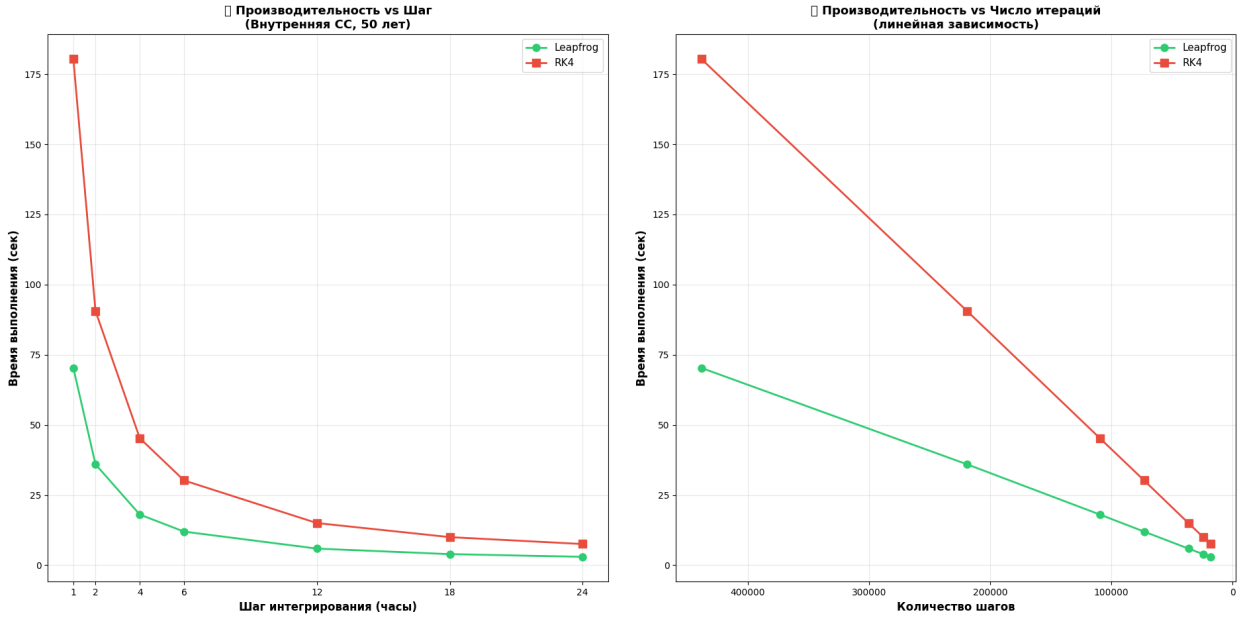

Рис. 12 – Зависимость времени симуляции от временного шага

4.6 Тест стабильности

На примере полной Солнечной системы проведём тест стабильности методов интегрирования. Для этого будем бинарным поиском искать такое значение, при котором происходит разрушение системы. Траектории планет при критическом значении временного шага показаны на рисунках.

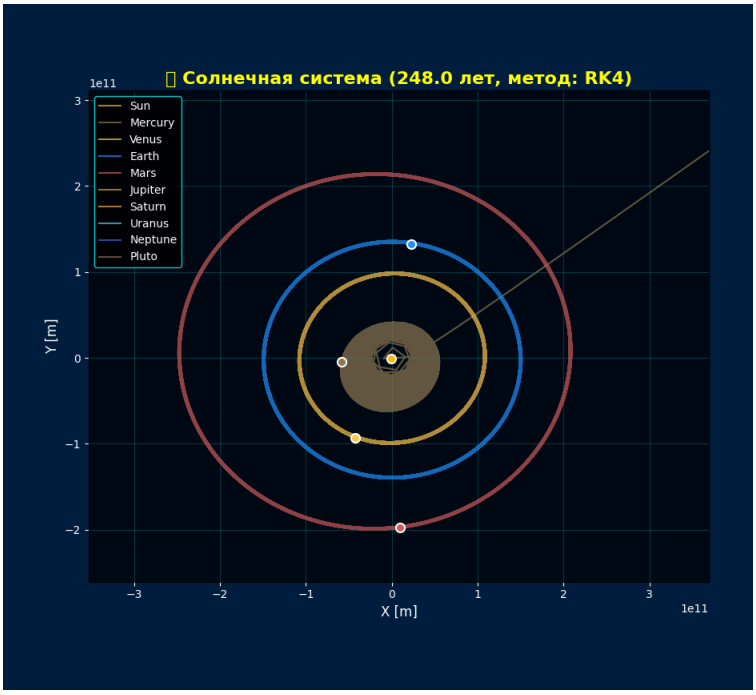

Рис. 13 – Разрушение Солнечной системы, LF, dt=250000 секунд

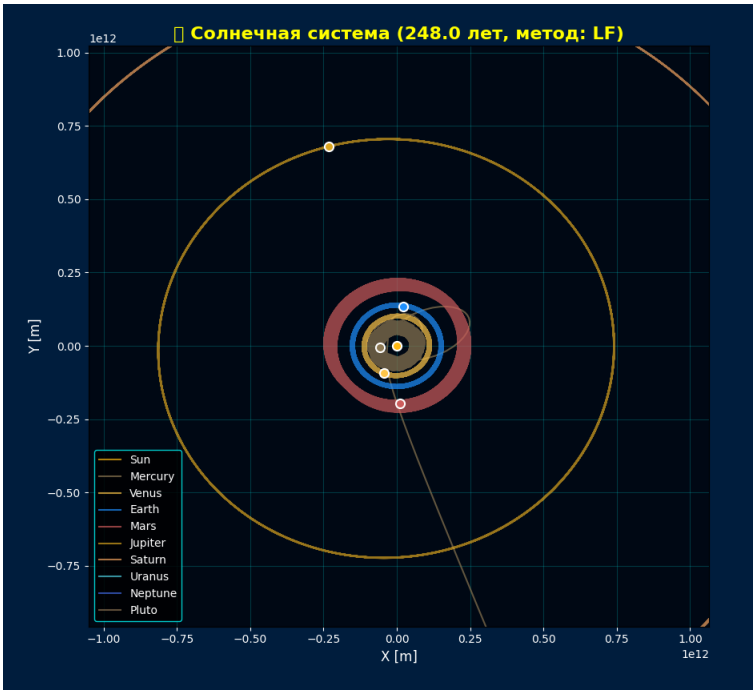

Рис. 14 – Разрушение Солнечной системы, LF, dt=1280000 секунд

Критическое значение dt для метода Рунге–Кутты в 5 раз меньше, чем для метода Leap–Frog.

5 Вывод

В рамках данного проекта была разработана программная система для численного моделирования гравитационной задачи N-тел и проведён комплексный сравнительный анализ трёх методов численного интегрирования: явного метода Эйлера, метода Рунге-Кутты 4-го порядка и симплектического метода Leap–Frog.

Основные результаты Явный метод Эйлера показал неудовлетворительные результаты даже на коротких временных интервалах. Метод демонстрирует быстрое накопление ошибок и систематическое нарушение законов сохранения энергии, что делает его непригодным для моделирования гравитационных систем на любых значимых временных масштабах.

Метод Рунге–Кутты 4-го порядка продемонстрировал высокую точность в краткосрочной перспективе (до 300 лет симуляции). Благодаря четвертому порядку точности, метод обеспечивает минимальную погрешность на коротких интервалах времени. Однако при долгосрочном моделировании (более 500 лет) наблюдается систематический дрейф полной энергии системы, что связано с несимплектичностью метода. Вычислительная стоимость RK4 в два раза выше, чем у Leapfrog, за счёт необходимости четырёхкратного вычисления ускорений на каждом шаге.

Метод Leapfrog показал наилучшие результаты для долгосрочного моделирования гравитационных систем. Симплектичность метода обеспечивает отличное сохранение полной энергии системы на временных масштабах в тысячи лет. Хотя краткосрочная точность уступает RK4, осцилляции энергии остаются ограниченными и не приводят к систематическому дрейфу. Метод также продемонстрировал в 5 раз большую численную стабильность по сравнению с RK4, позволяя использовать значительно больший шаг интегрирования без разрушения системы.

Практические рекомендации Для краткосрочных симуляций (до нескольких десятков орбитальных периодов), где требуется максимальная точность отдельных траекторий, рекомендуется использовать метод Рунге-Кутты 4-го порядка с малым шагом интегрирования.

Для долгосрочного моделирования гравитационных систем (сотни и тысячи лет) оптимальным выбором является метод Leapfrog. Его симплектичность критически важна для предотвращения накопления систематических ошибок в энергии системы.

При ограниченных вычислительных ресурсах метод Leapfrog предпочтителен благодаря выигрышу в производительности и возможности использования большего временного шага.

Заключение Разработанная программная система успешно продемонстрировала способность моделировать реальную Солнечную систему на временных масштабах в тысячи лет. Модульная архитектура программы обеспечивает лёгкую расширяемость для добавления новых численных методов и алгоритмов оптимизации вычислений. Проведённое исследование подтвердило фундаментальное значение симплектичности для численного интегрирования гравитационно связанных систем и продемонстрировало, что высокий формальный порядок точности не гарантирует превосходства при долгосрочном моделировании консервативных динамических систем.