Информационно-коммуникационные технологии в педагогическом образовании, 2025. № 2 (95). *infed.ru*

УДК 004.43: 629.78

Д. Н. Стахин, Н. А. Стахин

D. N. Stakhin, N. A. Stakhin

Стахин Дмитрий Николаевич, аспирант, ТГПУ, г. Томск, Россия.

Стахин Николай Александрович, к. ф.-м. н., доцент кафедры информатики, ТГПУ, г. Томск, Россия.

Stakhin Dmitry Nikolaevich, Postgraduate student, Tomsk State Pedagogical University, Tomsk, Russia.

Stakhin Nikolay Aleksandrovich, Candidate of Physico-mathematical Sciences, Associate Professor, Tomsk State Pedagogical University, Tomsk, Russia.

ЛЕТИМ НА ЛУНУ. ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ТРАЕКТОРИИ РАКЕТЫ СРЕДСТВАМИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

WE ARE GOING TO THE MOON. NUMERICAL SIMULATION OF THE ROCKET TRAJECTORY USING THE PYTHON PROGRAMMING LANGUAGE

Аннотация. *Численно рассчитана баллистическая траектория ракеты, летящей от Земли к Луне. Предложен алгоритм оптимизации начальной скорости и угла старта ракеты для мягкой посадки ракеты на поверхность Луны. Алгоритм реализован на языке программирования Python. Приведены: график траектории ракеты и график изменения скорости ракеты во время полёта.*

Annotation. *The ballistic trajectory of a rocket flying from the Earth to the Moon has been calculated numerically. An algorithm for optimizing the initial velocity and launch angle of a rocket for a soft landing of a rocket on the surface of the Moon is proposed. The algorithm is implemented in the Python programming language. The following are shown: a graph of the rocket trajectory and a graph of the rocket velocity change during the flight.*

Ключевые слова: *траектория ракеты, численное моделирование, метод Рунге-Кутта, метод Powell, язык программирования Python, законы Ньютона.*

Keywords: *rocket trajectory, numerical simulation, Runge-Kutta method, Powell method, Python programming language, Newton's laws.*

Введение. Изучение Луны продолжает оставаться ключевой задачей современной космонавтики и науки о космосе. С момента первых шагов человека по её поверхности в 1969 году интерес к Луне значительно возрос. Всего было совершено 24 пилотируемых полёта на Луну, успешно закончились высадкой космонавтов на её поверхность только шесть полётов. Последней успешной посадкой стала миссия «Аполлон-17» в 1972 году. Всего слетали к Луне 24 человека, трое из них слетали дважды. Все пилотируемые полёты к Луне были частью программы НАСА «Аполлон». В СССР до 1972 года состоялись четыре испытательных запуска. К сожалению, все они закончились неудачей.

Создание баз на Луне, изучение её геологии и ресурсов, а также подготовка к дальним межпланетным миссиям, стартующим с поверхности Луны, требуют разработки точных моделей движения космических аппаратов.

Одной из важнейших задач является расчет траектории ракеты между Землёй и Луной. Эта задача, известная как задача трёх тел, является сложной проблемой классической механики, не имеющей аналитического решения. Гравитационное взаимодействие Земли, Луны и ракеты исключает возможность аналитического решения задачи трёх тел. Однако современные численные методы позволяют с высокой точностью моделировать такие траектории численно и оптимизировать начальные параметры полёта.

Python, благодаря своим библиотекам ‘numpy‘, ‘scipy‘ и ‘matplotlib‘, стал одним из наиболее популярных инструментов для проведения численных экспериментов и визуализации результатов [5]. В данной работе демонстрируется, как современные численные алгоритмы языка программирования Python позволяют решить сложную задачу динамики космических аппаратов.

Исторический контекст

С самого начала космической эры оптимизация траектории космических аппаратов вдохновляла инженеров и ученых на разработку компьютеров и их использование в решении различных задач космонавтики. Программа «Аполлон» потребовала выполнения сложных расчётов, занимавших недели вычислительного времени на громоздких мейнфреймах [6]. Сегодня аналогичные расчеты можно выполнить на персональном компьютере за минуты, благодаря развитию алгоритмов и вычислительных мощностей компьютеров.

Некоторые авторы используют в расчетах траектории ракеты методы, основанные на аналитических решениях и упрощениях, таких как разбиение траектории на два отдельных участк а (геоцентрическую и селеноцентрическую фазы). Примером так ой работы является исследование, выполненное *Т. В. Старостиной и О. Л. Стариновой*, в котором для расчёта траектории перелета Земля-Луна малого космического аппарата применяются упрощённые модели двух тел и метод конических сечений [9]. Эти методы позволяют проводить анализ траекторий с минимальными вычислительными затратами, что особенно важно для малых аппаратов с ограниченными ресурсами. Тем не менее, такие подходы имеют ограничения при учёте гравитационных взаимодействий и требуют дополнительных допущений для расчёта оптимальных параметров.

В отличие от цитируемых методов, предложенный в данной работе подход использует численные методы для моделирования траектории ракеты, учитывающие совместное воздействие на ракету гравитационного поля и Земли, и Луны. Это обеспечивает моделирование без упрощений, поэтому предложенный подход оказывается применимым для широкого спектра задач – от образовательных целей до подготовки реальных космических миссий.

В отличие от методов прошлого века, современные подходы используют интеграцию численных методов с алгоритмами оптимизации, что позволяет достичь высокой точности даже для сложных задач.

Цели и задачи работы

Цель работы – разработать численную модель траектории ракеты с использованием современных библиотек Python. Для достижения этой цели решаются следующие задачи:

1. Построение модели движения ракеты в гравитационном поле двух планет: Земли и Луны.

2. Расчет траектории ракеты как решение системы дифференциальных уравнений.
3. Применение численных методов оптимизации для выбора начальных параметров стартующей ракеты.
4. Построение графиков: траектории ракеты и падения скорости ракеты в полете.
5. Анализ временных и пространственных характеристик полёта.

Программные инструменты

В работе используются следующие стандартные библиотеки языка программирования Python [1]:

- **‘numpy’** – для работы с массивами и решения задач линейной алгебры [12];
- **‘scipy’** – для численного решения дифференциальных уравнений и оптимизации [5];
- **‘matplotlib’** – для визуализации траектории и анализа данных [8];
- **‘pandas’** – для представления результатов в виде таблиц [13].

Эти инструменты обеспечивают гибкость алгоритма и высокую производительность при моделировании.

Структура статьи

1. **Введение**, где описаны задачи и цели работы.
2. **Физическая модель –** содержит основные уравнения.
3. **Численное решение** – описывает алгоритмы интеграции и оптимизации.
4. **Результаты** – содержит графики, таблицы скорости и анализа траектории ракеты.
5. **Заключение** – рассматриваются результаты работы и перспективы новых исследований.

Инновационная составляющая

Данная работа использует современный подход при решении задачи межпланетной динамики. Показано, что язык программирования Python и его открытые библиотеки, позволяют моделировать сложную траекторию ракеты с высокой точностью. В отличие от громоздких методов прошлого, представленный алгоритм выполняет расчеты быстро и эффективно. Что открывает перспективы для использования модели в образовательных, исследовательских и прикладных задачах, связанных с практической космонавтикой.

Физическая модель

В работе рассматривается движение ракеты от Земли к Луне по законам классической механики. Ось декартовой системы координат находится в центре Земли, ракета удалена от центра Земли на расстояние R_{Rocket}. Луна находится на оси абсцисс и удалена от Земли на расстояние R_{Moon} [2]. Модель учитывает гравитационное воздействие Земли и Луны на ракету, а использует численные методы для расчета траектории.

Гравитационные силы

Сила притяжения между двумя телами массой m_1 и m_2 определяется законом всемирного тяготения Ньютона [10]:

Условия завершения полёта

Полёт завершится при выполнении одного из условий:

- Ракета достигает поверхности Луны, когда расстояние до её центра r_{Moon} становится меньше или равно радиусу Луны R_{Moon}.
- При выполнении этого условия, движение ракеты останавливается ($v_x = v_y = 0$).

Начальные условия

Начальные условия задаются следующим образом:

$$F = G \cdot \frac{m_1 \cdot m_2}{r^2},$$

где:

- F – сила гравитационного притяжения, Н;
- $G = 6.67430 \cdot 10^{-11}$ м^{3}кг$^{-1}$с$^{-2}$ – гравитационная постоянная;
- m_1, m_2 – массы тел, кг;
- r – расстояние между центрами масс тел, m_1, m_2.

На ракету действуют две основные силы, для проекций которых пишем:

1. Для гравитации Земли:

$$F_{Earth,x} = -G \cdot \frac{M_{Earth} \cdot m_{rocket} \cdot x}{r_{Earth}^3}, F_{Earth,y} = -G \cdot \frac{M_{Earth} \cdot m_{rocket} \cdot y}{r_{Earth}^3},$$

где $r_{Earth} = \sqrt{x^2 + y^2}$ — расстояние от ракеты до центра Земли.

2. Для гравитации Луны:

$$F_{Moon,x} = -G \cdot \frac{M_{Moon} \cdot m_{rocket} \cdot (x - R_{Earth-Moon})}{r_{Moon}^3}, F_{Moon,y} = -G \cdot \frac{M_{Moon} \cdot m_{rocket} \cdot y}{r_{Moon}^3},$$

где $r_{Moon} = \sqrt{(x - R_{Earth-Moon})^2 + y^2}$ — расстояние от ракеты до центра Лун

Уравнения движения

Согласно второму закону Ньютона:

$$\vec{a} = \vec{F}/m,$$

где $\vec{F}$ – суммарная сила, $\vec{a}$ – ускорение.

Уравнения движения ракеты в декартовой системе координат имеют вид:

$$\frac{dx}{dt} = v_x, \frac{dy}{dt} = v_y, \frac{dv_x}{dt} = \frac{F_{Earth,x} + F_{Moon,x}}{m_{rocket}}, \frac{dv_y}{dt} = \frac{F_{Earth,y} + F_{Moon,y}}{m_{rocket}}.$$

Условия завершения полёта

Полёт завершится при выполнении одного из условий:

- Ракета достигает поверхности Луны, когда расстояние до её центра r_{Moon} становится меньше или равно радиусу Луны R_{Moon}.

- При выполнении этого условия, движение ракеты останавливается ($v_x = v_y = 0$).

Начальные условия

Начальные условия задаются следующим образом:

$$x(0) = 0, y(0) = R_{Earth}, v_x(0) = v_0 cos\theta, v_y(0) = v_0 sin\theta,$$

где:

- v_0 – начальная скорость ракеты, м/с;
- θ – угол старта ракеты относительно горизонтальной плоскости.

Алгоритм расчёта

Расчеты выполняются по следующему алгоритму:

1. Рассчитать гравитационные силы Земли и Луны для каждой точки траектории.
2. Вычислить ускорения a_x , a_y на основе сил со стороны планет.
3. Решить систему дифференциальных уравнений методом Рунге-Кутта 8-го порядка (DOP853).
4. Проверить выполнение условий завершения полёта.

Коррекция траектории

Для минимизации остаточной скорости при посадке используется метод оптимизации Powell. Алгоритм выполняется в два этапа:

- Грубая оптимизация с широкими границами параметров.
- Уточнённая оптимизация в узком диапазоне.

Расчётные ограничения

Модель предполагает, что:

- Влияние других небесных тел (например, Солнца) пренебрежимо мало.

- Сопротивление атмосферы на высотах выше земной атмосферы отсутствует.
- Масса ракеты не изменяется в течение полёта.

Интеграция и оптимизация

Для решения дифференциальных уравнений используется интегратор DOP853, который обеспечивает высокую точность даже при сложных траекториях [7]. Оптимизация выполняется методом Powell, что позволяет эффективно находить минимальные скорости на поверхности Луны. Модель и подход адаптированы для образовательных и научных приложений, позволяя проводить численные эксперименты с высокой степенью точности.

Численное решение

Для численного моделирования движения ракеты между Землёй и Луной используется современный подход, включающий решение дифференциальных уравнений и численную оптимизацию. Основными инструментами являются язык Python и научные библиотеки, такие как ‘scipy‘ и ‘numpy‘.

Решение дифференциальных уравнений

Движение ракеты в гравитационных полях Земли и Луны описывается системой дифференциальных уравнений, определяющей изменение координат и скоростей (x, y, v_x, v_y). Решение этой системы выполняется с использованием функции solve_ivp из библиотеки scipy.integrate.

Основным методом интеграции выбран Dormand-Prince (DOP853) , адаптивный метод Рунге-Кутта 8-го порядка.

Преимущества используемого метода:

- Высокая точность при моделировании гравитационных сил.
- Адаптивный шаг времени, обеспечивающий стабильность и точность при численных расчётах.

- Возможность определения событий (events) для остановки интеграции, например, при достижении ракеты поверхности Луны.

Условия завершения полета

Для моделирования посадки на Луну или её прохождения мимо используются следующие условия завершения:

- Если расстояние до поверхности Луны (r_{Moon} – R_{Moon}) становится меньше или равно нулю, ракета достигает поверхности.
- Если ракета пересекает поверхность Луны, её движение останавливается путём обнуления скоростей ($v_x = 0$, $v_y = 0$).

Контроль точности

Метод DOP853 обеспечивает адаптивный контроль точности за счёт настройки параметров:

- $rtol = 10^{-12}$ – относительная точность.
- $atol = 10^{-14}$ – абсолютная точность.

Эти параметры позволяют учитывать малейшие изменения скорости и положения ракеты.

Оптимизация начальных параметров

Целью оптимизации является нахождение начальной скорости (v_0) и угла вылета (θ_0), минимизирующих функцию стоимости. Эта функция включает штраф за остаточную скорость ракеты и за отклонение от поверхности Луны в точке завершения траектории.

$$Cost(v_0, \theta_0) = Velocity\ Penalty + Distance\ Penalty,$$

Функция стоимости

Функция стоимости учитывает два компонента:

$$Velocity\ Penalty = k_v \cdot v_{landing}^2,$$

$Distance\ Penalty = k_d \cdot d^2_{surface}$, где

- **Velocity Penalty** – штраф за остаточную скорость ракеты при посадке:

$v_{landing}$ – скорость ракеты при контакте с поверхностью Луны,

k_V – коэффициент штрафа.

- **Distance Penalty** – штраф за отклонение от поверхности Луны:

$d_{surface}$ – отклонение от поверхности Луны, k_d – коэффициент штрафа.

Используемый метод оптимизации

Для оптимизации параметров (v_0, θ_0) используется метод **Powell**, метод обладает следующими преимуществами:

- Не требует расчёта градиентов, что упрощает реализацию при наличии сложной функции стоимости [4].
- Эффективен для задач с ограниченным числом параметров.
- Высокая точность нахождения минимума при условии корректной постановки задачи.

Этапы оптимизации

Оптимизация проводится в два этапа:

$$v_0 \in [6{,}000, 30{,}000]\ \text{м/с}, \quad \theta_0 \in [-\pi, \pi].$$

$$v_0 \in [v_{coarse} - 3{,}000, v_{coarse} + 3{,}000], \quad \theta_0 \in [\theta_{coarse} - 0.6, \theta_{coarse} + 0.6].$$

1. **Грубая оптимизация:** выполняется поиск оптимальных параметров в широких пределах.
2. **Точная оптимизация:** уточняются параметры в узком диапазоне вокруг результатов грубой оптимизации.

Численные результаты

В результате оптимизации были получены параметры (v_0, θ_0), которые обеспечивают минимальную скорость ракеты при посадке. Решение уравнений и визуализация траектории позволяет:

- Рассчитать полный путь ракеты с высокой точностью.
- Построить график изменения скорости в зависимости от времени.
- Оценить изменения расстояний до Земли и Луны на разных этапах полёта.

Особенности реализации в Python

Использование Python даёт следующие преимущества:

- Интеграция численных методов (solve_ivp) с оптимизацией (scipy.optimize.minimize).
- Простота настройки параметров и добавления новых условий.
- Высокая точность вычислений благодаря использованию современных алгоритмов, таких как DOP853.
- Визуализация результатов с помощью библиотеки matplotlib.

Представленный подход демонстрирует эффективность применения Python для сложных задач моделирования и оптимизации, включая расчёт межпланетных траекторий.

Программный код на Python

В данном разделе представлен полный код для расчётов траектории полёта на Луну. Код разделён на логические блоки для удобства восприятия.

Импорт библиотек и определение констант

```
# Импорт необходимых библиотек

import numpy as np

from scipy.integrate import solve_ivp
```

```
from scipy.optimize import minimize

import matplotlib.pyplot as plt

import pandas as pd

# Физические константы

G = 6.67430e-11  # Гравитационная постоянная, м^3 кг^-1 с^-2

M_earth = 5.972e24  # Масса Земли, кг

M_moon = 7.348e22  # Масса Луны, кг

R_earth = 6.371e6  # Радиус Земли, м

R_moon = 1.737e6  # Радиус Луны, м

R_earth_moon = 3.844e8  # Расстояние между Землёй и Луной, м

# Параметры симуляции

t_max = 300000  # Максимальное время моделирования, секунды

t_eval = np.linspace(0, t_max, 30000)  # Шаги времени для интеграции

Гравитационные силы Земли и Луны

# Сила гравитации со стороны Земли

def force_earth(x, y):

    r = np.sqrt(x**2 + y**2)

    f_x = -G * M_earth * x / r**3

    f_y = -G * M_earth * y / r**3

    return f_x, f_y


# Сила гравитации со стороны Луны
```

```
def force_moon(x, y):
    r = np.sqrt((x - R_earth_moon)**2 + y**2)
    f_x = -G * M_moon * (x - R_earth_moon) / r**3
    f_y = -G * M_moon * y / r**3
    return f_x, f_y
```

Дифференциальные уравнения движения

```
# Уравнения движения с остановкой при достижении поверхности Луны
def equations_with_stop_at_surface(t, y):
    x, y_position, vx, vy = y
    # Гравитационные силы
    fx_earth, fy_earth = force_earth(x, y_position)
    fx_moon, fy_moon = force_moon(x, y_position)
    # Расстояние до поверхности Луны
    distance_to_surface = np.sqrt((x - R_earth_moon)**2 + y_position**2) - R_moon
    if distance_to_surface <= 0:  # Если пересекаем поверхность Луны
        return [0, 0, 0, 0]  # Останавливаем движение
    # Обычные уравнения движения
   return [vx, vy, fx_earth + fx_moon, fy_earth + fy_moon]
```

Функция стоимости

```
# Функция стоимости для оптимизации параметров
def cost_with_surface_distance(v0, theta0):
    x0, y0 = 0, R_earth
    vx0 = v0 * np.cos(theta0)
```

```
    vy0 = v0 * np.sin(theta0)
    initial_state = [x0, y0, vx0, vy0]
    solution = solve_ivp(
        equations_with_stop_at_surface,
        (0, t_max),
        initial_state,
        t_eval=t_eval,
        method='DOP853',
        rtol=1e-12,
        atol=1e-14)
    if not solution.success:
        return np.inf
    # Минимальное расстояние до Луны
    x, y = solution.y[0], solution.y[1]
    distances_to_moon = np.sqrt((x - R_earth_moon)**2 + y**2)
    closest_distance = np.min(distances_to_moon) - R_moon
    if closest_distance <= 0:  # Если достигаем поверхности Луны
        closest_index = np.argmin(distances_to_moon)
        vx_closest, vy_closest = solution.y[2]closest_index], solution.y[3][closest_index]
        velocity_penalty = (vx_closest**2 + vy_closest**2) * 500
        return velocity_penalty
    closest_index = np.argmin(distances_to_moon)
    vx_closest, vy_closest = solution.y[2][closest_index], solution.y[3][closest_index]
```

```
    distance_penalty = closest_distance**2
    velocity_penalty = (vx_closest**2 + vy_closest**2) * 500
    return distance_penalty + velocity_penalty
```

Оптимизация параметров

```
# Функция для оптимизации начальных параметров
def optimize_trajectory():
    """
    Оптимизация начальной скорости и угла вылета.
    """
    # Грубая оптимизация
    print("Начало грубой оптимизации...")
    initial_guess = [1.12e4, np.pi / 4]
    coarse_bounds = [(8000, 25000), (0, np.pi)]
    result_coarse = minimize(
        lambda vars: cost_with_surface_distance(vars[0], vars[1]),
        x0=initial_guess,
        bounds=coarse_bounds,
        method='Powell')
    v0_coarse, theta0_coarse = result_coarse.x
    print(f"Грубая оптимизация завершена: v0 = {v0_coarse:.
2f}, theta = {theta0_coarse:.4f}")
    # Точная оптимизация
    print("Начало точной оптимизации...")
    fine_bounds = [
        (v0_coarse - 1000, v0_coarse + 1000),
```

```
        (theta0_coarse - 0.2, theta0_coarse + 0.2)
    ]
    result_fine = minimize(
        lambda vars: cost_with_surface_distance(vars[0], vars[1]),
        x0=[v0_coarse, theta0_coarse],
        bounds=fine_bounds,
        method='Powell')
    v0_fine, theta0_fine = result_fine.x
    print(f"Точная оптимизация завершена: v0 = {v0_fine:.2f},
theta = {theta0_fine:.4f}")
    return v0_fine, theta0_fine
      Визуализация траектории
# Функция для построения траектории
def visualize_trajectory(v0, theta0):
    """
    Построение графика оптимизированной траектории с
реальными масштабами Земли и Луны.
    """
    initial_state = [0, R_earth, v0 * np.cos(theta0), v0 *
np.sin(theta0)]
    # Решение уравнений движения
    solution = solve_ivp(
        equations_with_stop_at_surface,
        (0, t_max),
        initial_state,
        t_eval=t_eval,
```

```
        events=surface_event,
        method='DOP853',
        rtol=1e-12,
        atol=1e-14)
    x, y = solution.y[0], solution.y[1]
    # Построение графика
    plt.figure(figsize=(12, 6))  # Прямоугольный график
    plt.plot(x / 1e6, y / 1e6, label="Траектория", color="green")
    # Реальный масштаб Земли
    earth_circle = plt.Circle((0, 0), R_earth / 1e6, color="blue",
label="Земля", alpha=0.7)
    plt.gca().add_artist(earth_circle)
    # Реальный масштаб Луны
    moon_circle = plt.Circle((R_earth_moon / 1e6, 0), R_moon /
1e6, color="gray", label="Луна", alpha=0.7)
    plt.gca().add_artist(moon_circle)
    # Настройка осей
    plt.xlabel("x (тыс. км)")
    plt.ylabel("y (тыс. км)")
    plt.title("Оптимизированная траектория до поверхности
Луны")
    plt.legend()
    plt.grid(True)
    # Настройка масштаба осей
    plt.axis("scaled")  # Сохранение пропорций
```

```
    # Динамическая корректировка границ оси Y на основе траектории и положения Земли/Луны
    buffer = 0.1 * 1e8 / 1e6  # Небольшой буфер для эстетики
    y_min = min(y.min() / 1e6, -R_earth / 1e6) - 2 * buffer
    y_max = max(y.max() / 1e6, R_moon / 1e6) + buffer
    plt.gca().set_ylim([y_min, y_max])  # Динамическая установка границ оси Y
    # Сохранение графика в файл
    filename = 'trajectory.png'
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    print(f"График сохранён в файл: {filename}")
    plt.show()
      Расчёт таблицы скоростей
# Функция для вычисления таблицы скоростей
def calculate_speed_table(v0, theta0, num_points=100):
    """
    Рассчитать таблицу скорости, расстояния от Земли и Луны.
    """
    initial_state = [0, R_earth, v0 * np.cos(theta0), v0 * np.sin(theta0)]
    solution = solve_ivp(
        equations_with_stop_at_surface,
        (0, t_max),
        initial_state,
        t_eval=t_eval,
```

```
        method='DOP853',
        rtol=1e-12,
        atol=1e-14)
    if not solution.success:
        raise ValueError("Интеграция не удалась!")
    # Извлечение данных
    x, y = solution.y[0], solution.y[1]
    vx, vy = solution.y[2], solution.y[3]
    times = solution.t
    total_points = len(times)
    selected_indices = np.linspace(0, total_points - 1,
num_points, dtype=int)
    selected_times = times[selected_indices].astype(int)
    speeds = np.sqrt(vx[selected_indices]**2 +
vy[selected_indices]**2)
    distances_from_earth = (
        np.sqrt(x[selected_indices]**2 + y[selected_indices]**2) -
R_earth
    ) / 1000  # В км
    distances_to_moon = (
        np.sqrt((x[selected_indices] - R_earth_moon)**2 +
y[selected_indices]**2) - R_moon
    ) / 1000  # в км
    speed_table = pd.DataFrame({
      "Время (с)": selected_times,
       "Скорость (м/с)": speeds,
```

```
        "Расстояние от Земли (км)": distances_from_earth,
        "Расстояние до Луны (км)": distances_to_moon
    })
    return speed_table
```

Запуск расчётов и визуализации

```
# Запуск оптимизации
v0_opt, theta0_opt = optimize_trajectory()
# Визуализация траектории
visualize_trajectory(v0_opt, theta0_opt)
# Расчёт и вывод таблицы скоростей
speed_table = calculate_speed_table(v0_opt, theta0_opt, num_points=100)
print(speed_table)
```

Результаты

Результаты численного моделирования и оптимизации начальных параметров ракеты демонстрируют успешное достижение поверхности Луны и минимальной скорости на финальном участке. На основе численного расчета траектории были визуализированы положение ракеты и изменение скорости ракеты во времени.

Оптимизированная траектория

На рисунке 1 представлена оптимизированная траектория ракеты от Земли (слева) к Луне (справа). Траектория описывает движение ракеты в гравитационных полях двух небесных тел и показывает, что ракета достигает поверхности Луны с минимальным отклонением.

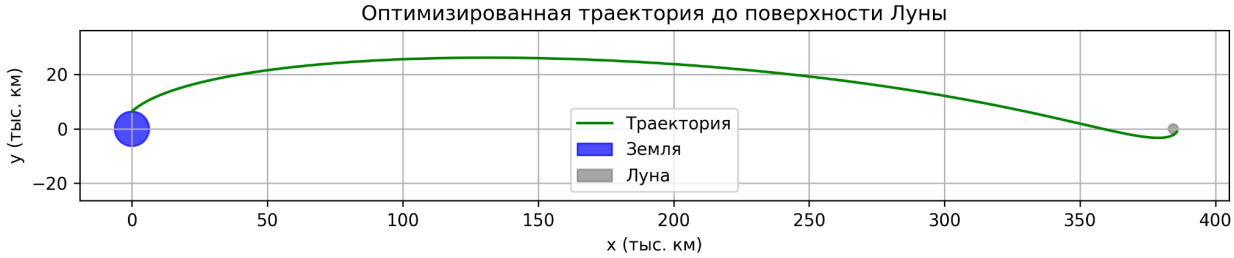

Рисунок 1. Оптимизированная траектория ракеты для достижения поверхности Луны

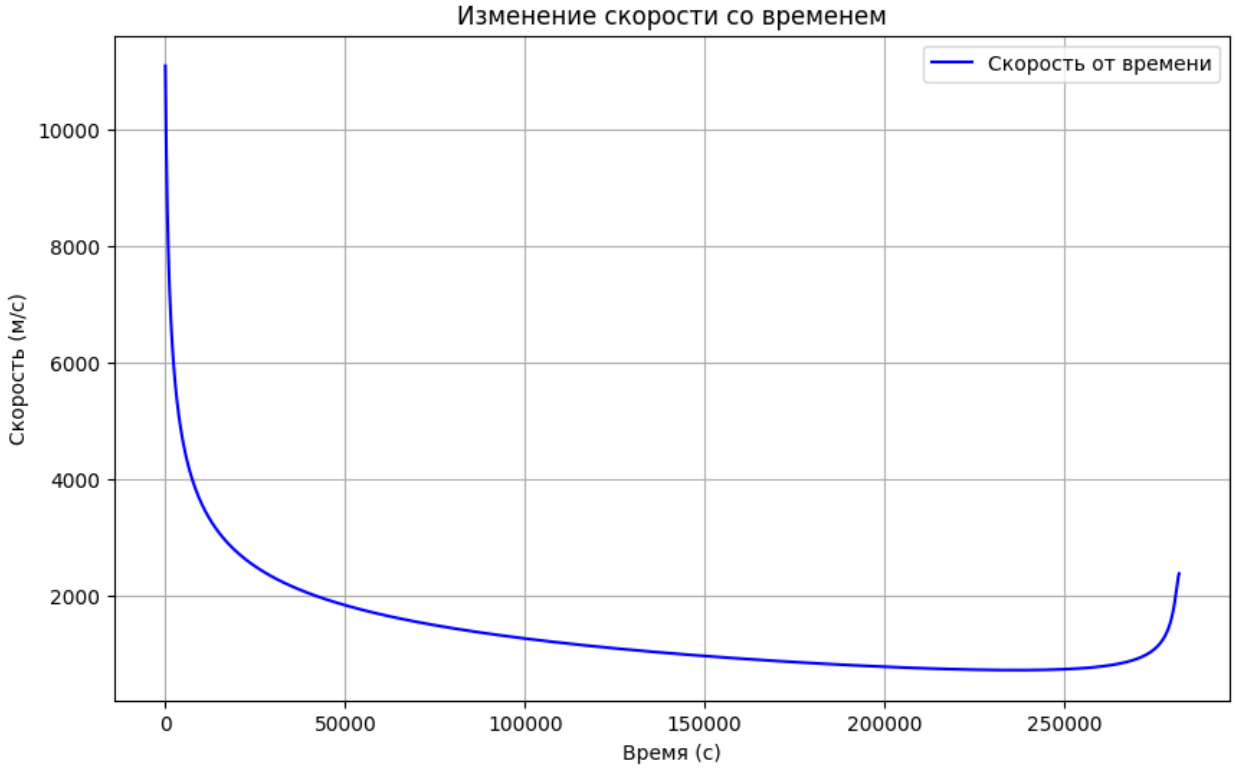

Рисунок 2 График изменения скорости ракеты во времени

Заключение

В данной работе был разработан алгоритм численного моделирования траектории ракеты от Земли к Луне средствами языка программирования Python. Основные результаты демонстрируют эффективное использование современных численных методов для решения сложных задач небесной механики и оптимизации межпланетных траекторий.

Основные результаты

- Построена физическая модель движения ракеты, основанная на законах Ньютона, учитывающая гравитационное притяжение ракеты со стороны Земли и Луны.
- Использованы современные численные методы: метод Рунге-Кутта 8-го порядка (DOP853) для решения системы дифференциальных уравнений [3] и метод оптимизации Powell для нахождения оптимальных параметров старта ракеты.
- Визуализированы траектория ракеты и изменение скорости во времени (рис. 2), что подтверждает достижение цели – минимизации скорости на поверхности Луны при точном попадании на её поверхность.
- Оптимизированы начальная скорость и угол старта ракеты, что обеспечивает реализацию траектории с минимальными отклонениями и соответствуют баллистическому сценарию полета, не требующему активного торможения. Это делает предложенный метод применимым для анализа экономичных траекторий космических аппаратов.

Практическая значимость

Разработанный алгоритм и его реализация на языке Python обладают высокой практической значимостью. Основные преимущества:

- Использование открытых инструментов, таких как numpy, scipy и matplotlib, позволяет легко воспроизводить результаты и модифицировать модель.
- Код подходит для достижения образовательных целей, включая изучение численных методов решения дифференциальных уравнений и оптимизации.
- Алгоритм может быть адаптирован для решения более сложных задач, включая анализ траекторий реальных космических миссий.

Представленный подход открывает возможности для детального расчета межпланетных траекторий и демонстрирует роль численных методов в современном исследовательском процессе.

Направления для дальнейших исследований

Данное исследование закладывает, на наш взгляд, основу для расчета дальних более сложных моделей межпланетных полетов. Потенциальные направления дальнейших исследований включают:

- Учет гравитационного влияния других небесных тел, таких как Солнце, при моделировании более сложных дальних траекторий [11].
- Включение атмосферы Земли в модель для анализа траекторий, начинающихся с её поверхности.
- Интеграцию активного торможения и корректирующих маневров для моделирования более реалистичных сценариев космических полетов.
- Оптимизацию траекторий для различных целей, таких как выход на окололунную орбиту или возврат на Землю.

Таким образом, данное исследование демонстрирует хороший потенциал использования современных численных методов для решения задач космонавтики и астрофизики. Представленный подход может служить основой для будущих исследований и разработки миссий к Луне и другим небесным телам.

Список литературы

1. Interactive courses Python. – URL: https:// www.codecademy.com/learn/learn-python-3 (дата обращения : 01.02.2025). – Текст : электронный.
2. Moon. // Wikipedia : [сайт]. – URL: https://en.wikipedia.org/ wiki/Moon (дата обращения 01.02.2025). – Текст : электронный.
3. Powell's_method // Wikipedia : [сайт]. – URL: https:// en.wikipedia.org/wiki/Powell's_method (дата обращения : 01.02.2025). – Текст : электронный.

4. Runge–Kutta methods // Wikipedia : [сайт]. – URL: https://en.wikipedia.org/wiki/Runge–Kutta_methods. (дата обращения : 01.02.2025). – Текст : электронный.
5. SciPy documentation. – URL: https://docs.scipy.org/doc/scipy/ (дата обращения : 01.02.2025). – Текст : электронный.
6. List of Apollo missions. // Wikipedia : [сайт]. – URL: https://en.wikipedia.org/wiki/List_of_Apollo_missions (дата обращения : 01.02.2025). – Текст : электронный.
7. Scipy.integrate DOP853. – URL: https://www.unige.ch/~hairer/software.html (дата обращения : 01.02.2025). – Текст : электронный.
8. Matplotlib. – URL: https://matplotlib.org/stable/contents.html (дата обращения : 01.02.2025). – Текст : электронный.
9. Старостина, Т. В. Расчёт траектории перелёта Земля-Луна малого космического аппарата / Т. В. Старостина, О. Л. Старинова – Текст : электронный. // Международная молодёжная научная конференция "XVI Королёвские чтения", посвящённая 60-летию полёта в космос Ю. А. Гагарина. – 2021. – Т. 1. – С. 114-115. – URL: https://rusneb.ru/catalog/000199_000009_010952634/ (дата обращения : 01.02.2025).
10. Gravity // Wikipedia : [сайт]. – URL: https://en.wikipedia.org/wiki/Gravity. (дата обращения : 01.02.2025) . – Текст : электронный.
11. Ельников, Р. В. Гравитационный маневр у луны при межпланетных перелетах космического аппарата с малой тягой. / Р. В. Ельников. – URL: https://en.wikipedia.org/wiki/Interplanetary_trajectory (дата обращения: 01.02.2025). – Текст : электронный.
12. NumPy user guide. NumPy quickstart. // Basic operations. – URL: https://numpy.org/doc/stable/user/quickstart.html (дата обращения : 01.02.2025). – Текст : электронный.
13. Pandas documentation. – URL: https://pandas.pydata.org/docs/ (дата обращения : 01.02.2025). – Текст : электронный.

© Стахин Д. Н., Стахин Н. А., 2025

Д. Н. Стахин, Н. А. Стахин 2025-03-27