Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 1

«Сравнение модели Галилея и модели Ньютона»

по курсу «Моделирование»

Студент группы ИУ9-81Б: Филимонов М. Д.

Преподаватель: Домрачева А. Б.

Москва 2026

1 Постановка задачи

Требуется сравнить две модели движения тела, брошенного под углом к горизонту из точки $(0, 0)$ при параметрах $\alpha = 57^\circ$ и $v_0 = 160$ м/с:

- модель Галилея без сопротивления воздуха;
- модель Ньютона с квадратичным сопротивлением воздуха.

Для модели Ньютона используется система ОДУ по переменным x, y, u, w, коэффициент сопротивления

$$\beta = \frac{cS\rho}{2}, \quad c = 0.15,\ S = 3,\ \rho = 1.225,$$

радиус шара задан напрямую: $r = 0.25$ м, материал шара — медь ($\rho_{copper} = 8960$ кг/м^3).

Необходимо получить и сравнить дальность полета, время полета и максимальную высоту, сформировать таблицу сравнения и график траекторий.

2 Исследовательский этап

В модели Галилея сопротивление воздуха отсутствует, поэтому движение описывается аналитически:

$$x(t) = x_0 + v_0 \cos\alpha\, t, \quad y(t) = y_0 + v_0 \sin\alpha\, t - \frac{gt^2}{2}.$$

Точка приземления находится из условия $y(t) = 0$.

В модели Ньютона учитывается сопротивление воздуха, пропорциональное квадрату скорости, поэтому используется численное решение системы ОДУ методом Рунге–Кутты 4-го порядка. Момент приземления уточняется линейной интерполяцией между шагами, когда координата y меняет знак.

Итоговые значения:

- Галилей: $x_{land} = 2383.971836$ м, $t_{land} = 27.357246$ с, $y_{max} = 917.747757$ м;
- Ньютон: $x_{land} = 1253.623389$ м, $t_{land} = 22.331246$ с, $y_{max} = 614.885546$ м;

- разница дальности: $\Delta x = -1130.348448$ м.

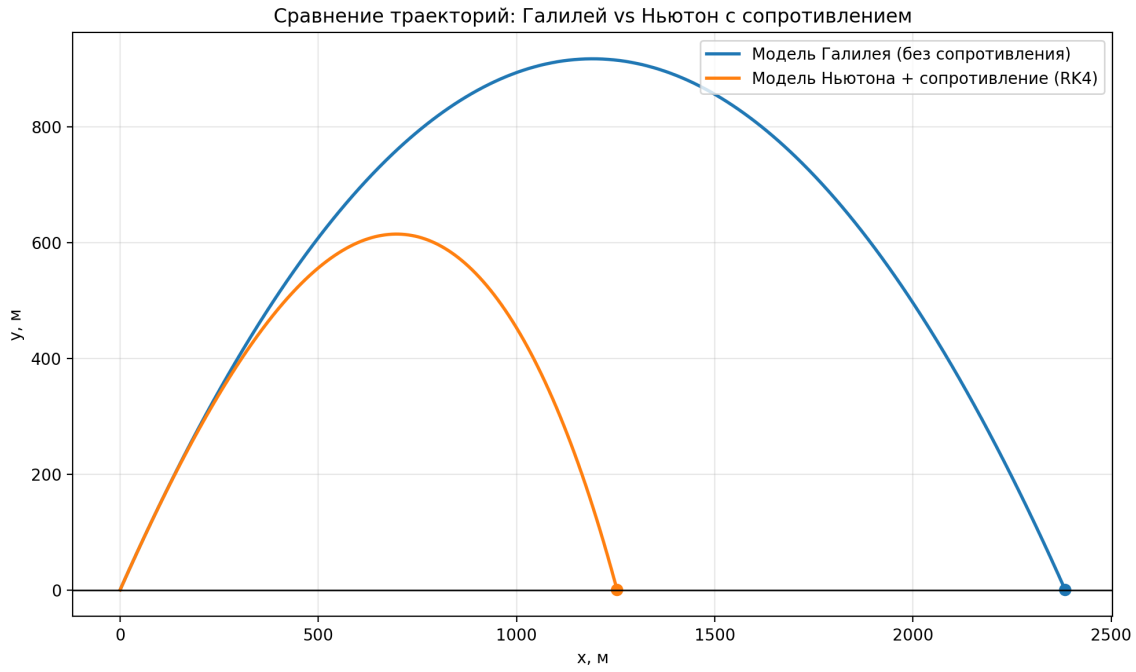

Рис. 1 — Сравнение траекторий в моделях Галилея и Ньютона

3 Конструкторский этап

Реализация в `solution.py` декомпозирована на блоки:

- структура параметров модели;
- аналитический расчет траектории Галилея;
- правая часть системы Ньютона и шаг RK4;
- численное моделирование до события приземления с интерполяцией;
- формирование таблицы сравнения и сохранение CSV;
- построение и сохранение графика;
- вывод итоговых метрик в консоль.

Для воспроизводимости используются фиксированные параметры и явное сохранение артефактов в файлы `comparison.csv` и `trajectory_comparison.png`.

4 Технологический этап

Проектные артефакты лабораторной `lab1`:

- исполняемый скрипт: `solution.py`;
- эквивалентный ноутбук: `solution.ipynb`;
- результаты выполнения: `comparison.csv`, `trajectory_comparison.png`.

Запуск в `uv`-окружении:

```
uv venv
uv pip install numpy matplotlib jupyter
uv run python solution.py
uv run jupyter nbconvert --to notebook --execute solution.ipynb
```

Полный исходный код проекта приведен ниже:

```python
import csv
import math
from dataclasses import dataclass
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np


def configure_matplotlib_for_cyrillic() -> None:
    plt.rcParams["font.family"] = "sans-serif"
    plt.rcParams["font.sans-serif"] = [
        "DejaVu Sans",
        "Arial",
        "Liberation Sans",
        "Noto Sans",
    ]
    plt.rcParams["axes.unicode_minus"] = False


@dataclass
class ModelParams:
    g: float = 9.81
    alpha_deg: float = 57.0
    v0: float = 160.0
    x0: float = 0.0
    y0: float = 0.0
    c: float = 0.15
    s: float = 3.0
    rho_air: float = 1.225
    rho_copper: float = 8960.0
    r_ball: float = 0.25
    material_name: str = "медь"

    @property
    def alpha_rad(self) -> float:
        return math.radians(self.alpha_deg)

    @property
    def beta(self) -> float:
        return self.c * self.s * self.rho_air / 2.0

    @property
    def mass(self) -> float:
        volume = 4.0 / 3.0 * math.pi * self.r_ball ** 3
        return self.rho_copper * volume


def galileo_landing(alpha_rad: float, v0: float, g: float, x0: float, y0: float, n_points:
↪  int = 1000) -> dict:
    cos_a = math.cos(alpha_rad)
    sin_a = math.sin(alpha_rad)
    if abs(cos_a) < 1e-12:
        raise ValueError("cos(alpha) слишком близок к нулю, формула y(x) неустойчива.")

    # y(t) = y0 + v0*sin(a)*t - g*t^2/2 = 0
    a_q = -0.5 * g
    b_q = v0 * sin_a
    c_q = y0
    disc = b_q * b_q - 4.0 * a_q * c_q
    if disc < 0:
        raise ValueError("Для заданных параметров нет реального времени падения.")

    sqrt_disc = math.sqrt(disc)
    t_candidates = [(-b_q + sqrt_disc) / (2.0 * a_q), (-b_q - sqrt_disc) / (2.0 * a_q)]
    t_land = max(t for t in t_candidates if t >= 0.0)
```

5 Тестирование, измерения и выводы

5.1 Проверка соответствия требованиям

По данным `validation_report.md` все обязательные пункты закрыты (PASS):

- реализованы обе физические модели (Галилей и Ньютон);
- метод RK4 применен для модели Ньютона;
- коэффициент β вычисляется по требуемой формуле;
- учтена масса медного шарика;
- точка приземления уточняется интерполяцией;
- сформированы `comparison.csv` и `trajectory_comparison.png`;
- подготовлены `solution.py`, `solution.ipynb`, `explanation.md`, `simple.md`.

5.2 Результаты тестирования

Валидация включает базовый запуск и 3 тест-кейса:

- базовый запуск `python solution.py` — PASS;
- тест-кейс 1 (новое условие, $\alpha = 57°$) — PASS;
- тест-кейс 2 ($\beta = 0$, совпадение с Галилеем в пределах численной ошибки) — PASS;
- тест-кейс 3 ($\alpha = 30°$, устойчивость результата) — PASS.

Проверка выполнения в `uv`-среде также прошла успешно: `uv run python solution.py`, `uv run jupyter nbconvert -to notebook -execute solution.ipynb`.

5.3 Обнаруженные проблемы

Критические проблемы не обнаружены.

5.4 Рекомендации

- при необходимости повысить точность контроля $\beta = 0$ можно уменьшить шаг интегрирования;
- расширить анализ графиками компонент скорости $u(t)$ и $w(t)$.

5.5 Итоговые выводы

Реализация лабораторной `lab1` соответствует обновленным coding-only требованиям: программа запускается без ошибок, результаты воспроизводимы, а сравнение моделей выполнено в табличной и графической форме.