0.1 Введение

Данный проект представляет собой компьютерную симуляцию полёта снаряда (шара) с реалистичной динамикой. В модели учтены:

- Гравитация;
- Аэродинамическое сопротивление воздуха;
- Влияние ветра;
- Отскок от поверхности (с учётом коэффициента восстановления).

Симуляция реализована на языке Python с использованием библиотеки `pygame`. Все численные интегрирования выполнены вручную без использования готовых решателей ОДУ.

0.2 Методы интегрирования

В задаче используются два метода численного интегрирования системы дифференциальных уравнений.

0.2.1 RK4 (Метод Рунге–Кутты 4-го порядка)

При каждом шаге рассчитываются:

$$
\begin{aligned}
k_1 &= f\big(\mathbf{y}(t)\big),\\
k_2 &= f\Big(\mathbf{y}(t) + \frac{\Delta t}{2}\, k_1\Big),\\
k_3 &= f\Big(\mathbf{y}(t) + \frac{\Delta t}{2}\, k_2\Big),\\
k_4 &= f\Big(\mathbf{y}(t) + \Delta t\, k_3\Big),
\end{aligned}
$$

а итоговое приближение:

$$
\mathbf{y}(t+\Delta t) \approx \mathbf{y}(t) + \frac{\Delta t}{6}\Big(k_1 + 2k_2 + 2k_3 + k_4\Big).
$$

Здесь $\mathbf{y}(t) = [x,\, y,\, v_x,\, v_y]$ — вектор состояния системы.

0.2.2 Dormand–Prince (Dopri5)

Метод Dormand–Prince использует 7 промежуточных этапов $k_1, \ldots, k_7$ с коэффициентами, взятыми из таблицы Бутчера. Итоговое обновление состояния производится по формуле:

$$\mathbf{y}(t + \Delta t) \approx \mathbf{y}(t) + \Delta t\Big(b_1 k_1 + b_2 k_2 + b_3 k_3 + b_4 k_4 + b_5 k_5 + b_6 k_6 + b_7 k_7\Big),$$

где коэффициенты a_{ij} и b_i задают вклад каждого этапа и обеспечивают высокую точность интегрирования (порядка 5-го).

0.3 Физическая модель и дифференциальные уравнения

Симуляция описывается системой ОДУ, вытекающих из второго закона Ньютона.

0.3.1 Уравнения движения

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y,$$
$$\frac{dv_x}{dt} = a_{\text{drag},x}, \quad \frac{dv_y}{dt} = -g + a_{\text{drag},y}.$$

0.3.2 Сила сопротивления воздуха

Сила сопротивления рассчитывается по формуле:

$$\mathbf{F}_{\text{drag}} = -\frac{1}{2}\, C_d\, \rho\, A\, |\mathbf{v}_{\text{rel}}|\, \hat{\mathbf{v}}_{\text{rel}},$$

где:

- C_d — коэффициент аэродинамического сопротивления,

- ρ — плотность воздуха,
- $A = \pi r^2$ — площадь поперечного сечения снаряда (при радиусе r),
- $\mathbf{v}_{\text{rel}} = \mathbf{v} - \mathbf{v}_{\text{wind}}$ — относительная скорость снаряда относительно ветра,
- $|\mathbf{v}_{\text{rel}}|$ — модуль относительной скорости,
- $\hat{\mathbf{v}}_{\text{rel}}$ — единичный вектор по направлению $\mathbf{v}_{\text{rel}}$.

После деления силы на массу m получаем ускорение:

$$\mathbf{a}_{\text{drag}} = -\frac{1}{2m} C_d \, \rho \, A \, |\mathbf{v}_{\text{rel}}| \, \mathbf{v}_{\text{rel}}.$$

0.3.3 Гравитация

Ускорение свободного падения g действует вниз, поэтому его вклад в уравнение для v_y равен $-g$.

0.4 Особенности реализации

- **Ручная реализация методов интегрирования.** Методы RK4 и Dormand–Prince реализованы самостоятельно, без использования готовых функций вроде `odeint` или `solve_ivp`.
- **Параметры модели.** Изменяемые параметры (размер снаряда, масса, угол выстрела, сила выстрела, высота пушки, скорость и направление ветра) задаются в словаре `params`, что позволяет легко настраивать симуляцию.
- **Отскок от поверхности.** При достижении снарядом земли ($y \leq 0$) реализован отскок с инвертированием вертикальной скорости, умноженным на коэффициент восстановления.

- **Визуализация и сравнение методов.** С помощью `pygame` отображаются траектории, полученные методами RK4 и Dormand–Prince, а также рассчитывается и выводится ошибка (разница в позициях) между ними.