Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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

по курсу «Моделирование»

«Решение баллистической задачи»

Студент группы ИУ9-81Б Яровикова А. С.

Преподаватель Домрачева А. Б.

Москва 2024

1 Цель работы

Целью работы является изучение методов Галилея и Ньютона для решения баллистической задачи, а также определение дальности полета брошенного тела.

2 Постановка задачи

Тело брошено под углом под углом α к горизонту со скоростью v_0. Необходимо найти дальность полета двумя способами:

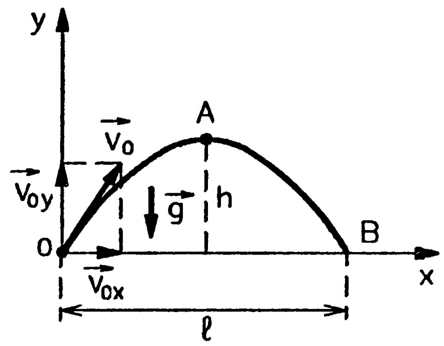
1. методом Галилея (кривизной Земли и сопротивлением воздуха пренебречь)

2. методом Ньютона (кривизной Земли пренебречь)

Начальные данные: чугунный шарик радиусом $r = 10$ см брошен под углом $\alpha = \pi/4$ с начальной скоростью $v_0 = 100$ м/с, $g = 9.8$ м/c^2.

3 Теоретические сведения

3.1 Модель Галилея



Как видно по рисунку к модели Галилея, траектория брошенного баллистического заряда описывается параболой:

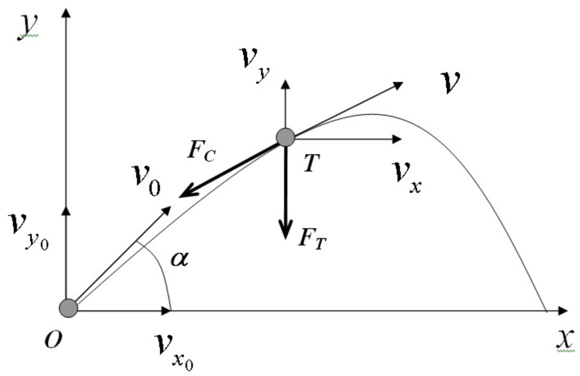
$$y = -\frac{g}{2v_0^2 cos^2(\alpha)}x^2 + (\tan\alpha)x, \tag{1}$$

где g - ускорение свободного падения, а $\alpha \in (0; \frac{\pi}{2}]$.

Следовательно для вычисления дальности полета, т.е. конечной координаты x используем формулу:

$$x = \frac{2(\tan\alpha)v_0^2 cos^2(\alpha)}{g}, \tag{2}$$

3.2 Модель Ньютона



Основное отличие модели Ньютона от модели Галилея заключается в том, что мы учитываем силу сопротивления воздуха $F_c = -\beta v^2$, коэффициент β которой вычисляется по формуле:

$$\beta = \frac{CS\rho}{2}, \tag{3}$$

где C - константа ($C \approx 0.15$),

S – площадь поперечного сечения брошенного шара ($S = \pi r^2$),

ρ - плотность воздуха ($\rho = 1,29$ кг/м^3).

Данный метод заключается в решении системы дифференциальных уравнений:

$$\begin{cases} \dfrac{du}{dt} = -\dfrac{\beta}{m} u\sqrt{u^2 + w^2} \\ \dfrac{dw}{dt} = -\dfrac{\beta}{m} w\sqrt{u^2 + w^2} - g \\ \dfrac{dx}{dt} = u \\ \dfrac{dy}{dt} = w \end{cases}, \tag{4}$$

при следующих начальных условиях:

$$\begin{cases} u(0) = v_0 cos(\alpha) \\ w(0) = v_0 sin(\alpha) \\ x(0) = 0 \\ y(0) = 0 \end{cases}. \tag{5}$$

Для решения первой системы используется метод Рунге-Кутта 4-го порядка точности.

4 Практическая реализация

Далее приведена реализация программы на языке Python, которая вычисляет дальность полета тела, брошенного под углом к горизонту методом Галилея и методом Ньютона.

Исходный код программы представлен в листинге 1.

Листинг 1: Исходный код

```
import math

radius = 0.1
rho = 7874
air_rho = 1.29
v_0 = 100
g = 9.8
alpha = 45

# m = pV
def find_massa(radius, p):
  V = 4/3 * math.pi * (radius ** 3)
   return p * V

# B = Csp / 2
def find_betta(radius, p, C=0.15):
   s = math.pi * (radius ** 2)
   return C * s * p/ 2

def galileo(speed, angle):
   alpha = math.radians(angle)
   return math.tan(alpha) * 2 * (math.cos(alpha) * speed) ** 2 / g

# u = V_x = V*cos(a)
def find_w(v, angle, t):
   return v * math.cos(math.radians(angle))

# w = V_y = V*sin(a)
def find_u(v, angle, t):
   return v * math.sin(math.radians(angle)) - g * t

# du/dt
def find_u_deriv(betta, m, u, w):
   return -betta * u * math.sqrt(u ** 2 + w ** 2) / m

# dw/dt
```

```
def find_w_deriv(betta, m, u, w):
    return -betta * w * math.sqrt(u ** 2 + w ** 2) / m - g

def newton(speed, radius, rho, angle, h=0.01):
    massa = find_massa(radius, rho)
    betta = find_betta(radius, air_rho) # rho
    v = [(find_u(speed, angle, 0), find_w(speed, angle, 0))]
    # The Runge Kutta method of the 4th order
    coords = [(0, 0)]
    while coords[-1][1] >= 0:
        cur_v = v[-1]
        cur_u = cur_v[0] # u = V_x
        cur_w = cur_v[1] # w = V_y

        k1_u = h * find_u_deriv(betta, massa, cur_u, cur_w)
        k1_w = h * find_w_deriv(betta, massa, cur_u, cur_w)

        k2_u = h * find_u_deriv(betta, massa, cur_u + k1_u / 2, cur_w + k1_w / 2)
        k2_w = h * find_w_deriv(betta, massa, cur_u + k1_u / 2, cur_w + k1_w / 2)

        k3_u = h * find_u_deriv(betta, massa, cur_u + k2_u / 2, cur_w + k2_w / 2)
        k3_w = h * find_w_deriv(betta, massa, cur_u + k2_u / 2, cur_w + k2_w / 2)

        k4_u = h * find_u_deriv(betta, massa, cur_u + k3_u / 2, cur_w + k3_w / 2)
        k4_w = h * find_w_deriv(betta, massa, cur_u + k3_u / 2, cur_w + k3_w / 2)

        cur_u += (k1_u + 2 * k2_u + 2 * k3_u + k4_u) / 6
        cur_w += (k1_w + 2 * k2_w + 2 * k3_w + k4_w) / 6

        v.append((cur_u, cur_w))
        cur_coords = coords[-1]
        cur_coord_x = cur_coords[0] + h * cur_u
        cur_coord_y = cur_coords[1] + h * cur_w
        coords.append((cur_coord_x, cur_coord_y))
    return coords[-1][0]

if __name__ == "__main__":
    print("Galileo method\n", galileo(v_0, alpha))
    print("Newton method\n", newton(v_0, radius, rho, alpha))
```

5 Результаты

Для заданных начальных условий ($r = 0.1$ м, $\alpha = \frac{\pi}{4}$, $v_0 = 100$ м/с) были получены следующие значения дальности полета:

Модель	Дальность полета, м
модель Галилея	1020.40816
модель Ньютона	950.84778

6 Вывод

В ходе выполнения лабораторной работы были изучены модели Галилея и Ньютона для решения баллистической задачи, их реализация была написана на языке Python.

В методе Ньютона для решения системы дифференциальных уравнений использовался метод Рунге-Кутта 4-го порядка точности. Однако важно иметь в виду, что окончательный результат программы может несколько отличаться от реального из-за вычислительных погрешностей, поскольку все расчеты были выполнены на ЭВМ.

Важно также отметить, что метод Ньютона более точен по сравнению с методом Галилея, поскольку модель учитывает широкий спектр факторов, включая сопротивление воздуха, плотность материала, размер и форму объекта. Поэтому метод Ньютона может привести к более реалистичным результатам, что подтверждается меньшей дальностью полета по сравнению с методом Галилея по результатам сравнения.