Забавная горка

Выполнил: Лышов В.А.

Научный руководитель: Байгашов А.С.

Аннотация

В работе рассматривается решение задачи о движении тела по горке. Изучается возможность численного моделирования такого движения. Для этого используется среда Python. Приводятся результаты моделирования в виде рисунков различных этапов движения тела.

Введение

Многие физические законы достаточно сложно представить себе наглядно. Это мешает хорошему освоению физики как школьной дисциплины. К счастью, современные технологии позволяют наглядно показывать многие процессы и явления, которые сложно продемонстрировать в качестве опыта или лабораторной работы. Обычно для этого используются численные модели – программы, рассчитывающие результат применения того или иного физического закона к конкретной ситуации. Несмотря на то, что численное моделирование основано на сложных математических законах и формулах, современные средства программирования позволяют во многом автоматизировать процесс моделирования.

Целью этой работы является моделирование движения шарика, с начальной скоростью закатывающегося на горку. Для этого необходимо разобраться с общими принципами численного моделирования и языком программирования Python, на котором будет написана программа.

Постановка задачи

Движение тела вверх по горке под действием начальной скорости описывается дифференциальным уравнением:

$$\begin{cases} \dfrac{dx}{dt} = v_x \\ \dfrac{dv_x}{dt} = -0{,}1 \cdot \lambda \\ \dfrac{dy}{dt} = v_y \\ \dfrac{dv_y}{dt} = -g + \lambda \end{cases}$$

где g - ускорение свободного падения, а параметр определяется следующим образом:

$$\lambda = \frac{0{,}1 \cdot x \cdot v_x^2 + g}{1{,}1}$$

Для его решения необходимо задать форму горки, а также определить начальное положение и скорость шарика. Горку представим в виде ограниченного подъёма, переходящего в плоскость, параллельную земле. Рассматривая двумерный случай, горку можно описать как график функции y(x):

$$\begin{cases} y = 0, & \text{при } x < 0 \\ y = 0{,}05 \cdot y^2, & \text{при } x \in (0{,}10) \\ y = 10, & \text{при } x > 10 \end{cases}$$

Шарик разместим у подножья подъёма и зададим такую начальную скорость, чтобы он сумел подняться по горке как минимум до её наивысшей точки. Тут следует учесть, что шарик может отрываться от поверхности горки, двигаясь по инерции и отскакивая от неё после падения. Удары о поверхность будем считать неупругими, просто разворачивая вертикальную скорость шарика при ударе в противоположную сторону. С целью упрощения задачи пренебрежём сопротивлением воздуха и силой трения.

Результат моделирования

Используя язык программирования Python была создана программа, численно решающая указанное дифференциальное уравнение. Её результат был показан с помощью анимации движения шарика, которую можно разбить на несколько рисунков в разное время.

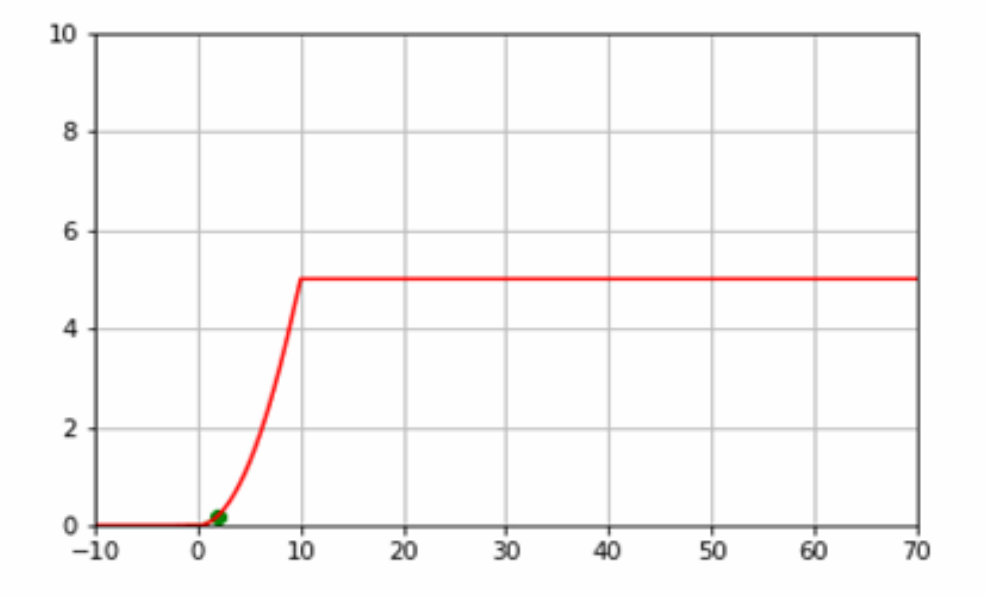

Рис 1: Тело у подножья горы, предана скорость.

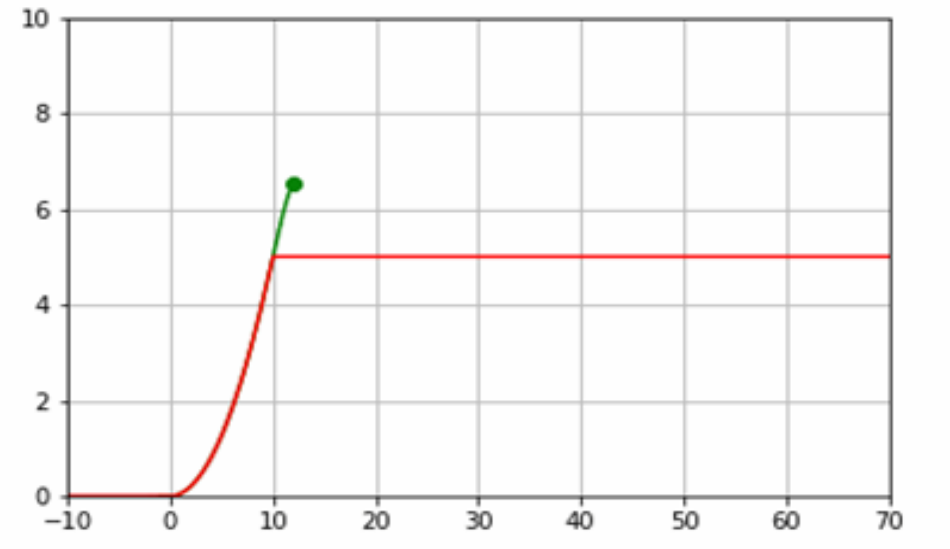

Рис 2: Тело у вершины горки, вылетает за ее пределы.

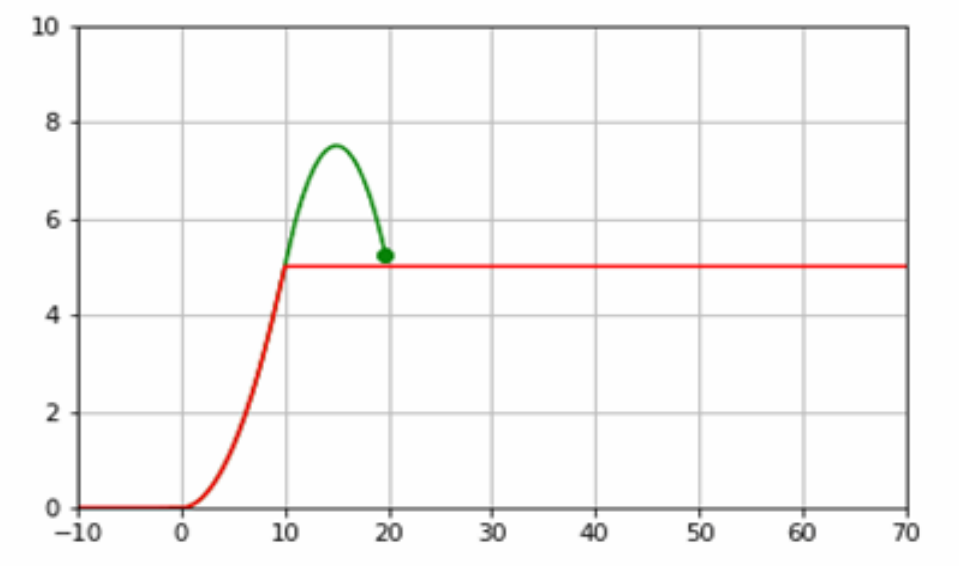

Рис 3: Шарик может отрываться от поверхности горки, двигаясь по инерции и отскакивая от неё после падения.

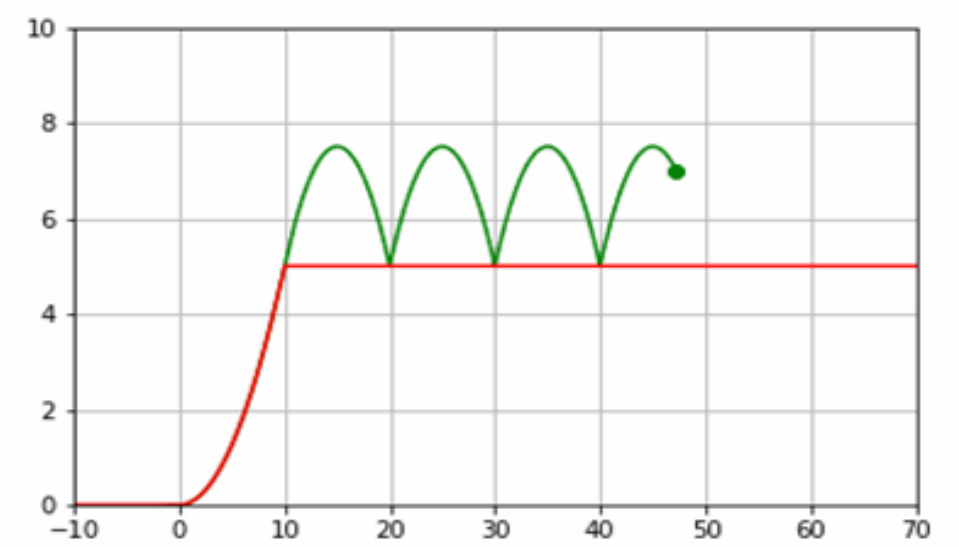

Рис 4: Неупругие удары о поверхность.

Заключение

Поставленная задача была успешно решена, была получена анимация движения шарика по горке. Можно сделать вывод о том, средства численного решения уравнений в языке программирования Python позволяют моделировать сложные процессы с помощью сравнительно простых программ. Это весьма удобно во многих случаях, когда нужно наглядно показать действие того или иного физического закона.

Листинг кода решения задачи:

```
#funny slide(заабвная горка)

import numpy as np
import math
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.axes3d as p3
from matplotlib import animation
from matplotlib.animation import ArtistAnimation
from sympy import sin, tan, cos, pi, symbols, diff, Heaviside

x1, y1, D1, D2, Y1 = symbols('x1 y1 D1 D2 Y1')

# МОЖНО ИССЛЕДОВАТЬ ФОРМУ ГОРКИ
h = 5 # Высота подъема горки
k = 10 # Длина горки по горизонтали
D0 = 0.05 * x1**2 # Уравнение параболы

def mountain(x, L1, L2):
    """
    Функция возвращает первые и вторые производные (f1,f2) уравнений
    горки на разных участках и высоту горки
    """
    if x < L1:
        f1, f2 = 0, 0
        Y = 0
    if x > L2:
        f1, f2 = 0, 0
        Y = h
    if L1 <= x <= L2:
        D1 = diff(D0, x1)
        D2 = diff(diff(D0, x1), x1)
        f1 = D1.subs(x1, x) # Команда подстановки значения x вместо x1 в D1
        f2 = D2.subs(x1, x) # Команда подстановки значения x вместо x1 в D2
        Y = D0.subs(x1, x) # Команда подстановки значения x вместо x1 в D0
    return f1, f2, Y
```

```python
# Определяем функцию для системы диф. уравнений
def move_func(s, t):
    x, v_x, y, v_y = s

    res = mountain(x, 0, k)

    f1 = res[0]
    f2 = res[1]

    # множитель Лагранжа
    lam = (f2 * v_x**2 + g) / (1 + f1**2)

    # модуль скорости
    v = np.sqrt(v_x**2 + v_y**2)

    dxdt = v_x
    dv_xdt = - lam * f1
    if y > h:#условие возможного отрыва на пологом участке
        dv_ydt = - g
    else:
        dv_ydt = - g + lam
    dydt = v_y

    return dxdt, dv_xdt, dydt, dv_ydt

# Определяем начальные значения и параметры, входящие в систему диф. уравнений
x0 = -1
v_x0 = 14

y0 = 0
v_y0 = 0

g = 9.8

s0 = x0, v_x0, y0, v_y0

X = []
Y = []

N = 5000
```

```
tau = np.linspace(-10, 70, 500)

for i in range(500):
    X.append(tau[i])
    res = mountain(tau[i], 0, k)
    Y.append(res[2])

Px = []
Py = []

T = np.linspace(0,7,N)

# Столкновение с поверхностью
for i in range(N-1):
    t = [T[i], T[i+1]]

    sol = odeint(move_func, s0, t)
    Px.append(sol[1,0])
    Py.append(sol[1,2])

    x0 = sol[1,0]
    vx0 = sol[1,1]
    y0 = sol[1,2]
    vy0 = sol[1,3]

    if np.abs(h-y0) <= 0.01 and x0 > k + 0.01:
        vy0 = - vy0

    s0 = x0, vx0, y0, vy0

# Построение фигуры
fig = plt.figure()
body = []
step = int(N/100)

for i in range(0,len(T)-1,step):
    body1, = plt.plot(Px[i], Py[i], 'o', color='g')
    body1_line, = plt.plot(Px[:i], Py[:i],'-',color='g')

    body.append([body1, body1_line])

ani = ArtistAnimation(fig,body,interval=5)

plt.plot(X,Y,color='r')
plt.xlim(-10,70)
plt.ylim(0,10)
plt.grid()

# plt.show()
ani.save('Mountain.gif')
```