Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

Институт Информационных и Вычислительных Технологий

Кафедра Математического моделирования

КУРСОВАЯ РАБОТА

по дисциплине «Численные методы»

ТЕМА: ДВИЖЕНИЕ В ЦЕНТРАЛЬНОМ ПОЛЕ

Студенты:
Ахмедов К.Т.

Преподаватель:
Амосова Ольга Алексеевна

Группа:
А-16-20

Москва, 2022

Содержание

1.ПОСТАНОВКА ЗАДАЧИ

1.1 Условия задачи

Цель: определить движение точечного тела массы m в поле тяготения, создаваемой силой притяжения, равной:

$$\vec{F}=\gamma\, Mmr^{\alpha}\frac{\vec{r}}{r},$$

где: $r=\sqrt{x^2+y^2}$ – радиус-вектор тела,

(x,y) - декартовы координаты в системе с центром

координат, расположенном в центре притяжения).

В классической Ньютоновской механике показатель $\alpha=-2$

Гравитационная постоянная $\gamma=6.67\cdot 10^{-11}\left[\frac{м^3}{кг\cdot с^3}\right]$

Пусть $m=1$ и в начальный момент времени $t=0$ тело находится на расстоянии R_0 и имеет скорость v_0, перпендикулярную радиус-вектору. В качестве источника притяжения возьмем Землю, но будем считать, что вся ее масса сосредоточена в центре.

1.2 Используемые константы

$\gamma=6.67\cdot 10^{-11}\left[\frac{м^3}{кг\cdot с^3}\right]$ – Гравитационная постоянная

$M=5.974\cdot 10^{24}[кг]$ – Масса Земли

$R=6371\cdot 10^{3}[м]$ – Радиус Земли

$m=1[кг]$ – Масса тела

$\alpha=-2$ – коэффициент

1.3 Вывод дифференциального уравнения и постановка задачи Коши

Используя второй закон Ньютона, составим дифференциальные уравнения для описания изменения декартовых координат тела. Используя указанные выше начальные данные, поставим задачу Коши для полученной системы ОДУ:

По второму закону Ньютона, зная силы, действующие на некоторую точку, мы можем определить движение данной материальной точки:

$$m\vec{a}=\vec{F} \qquad (1.1)$$

где m – масса тела, F – равнодействующая приложенных сил, a – ускорение тела, совпадающее по направлению с равнодействующей силой F

Так как вектор ускорения точки равен производной от вектора скорости $\vec{v}$ или второй производной от радиус-вектора r точки по времени, т.е.:

$$\vec{a}=\vec{v}'=\vec{r}''$$

Тогда уравнение (1.1) может быть записано в виде:

$$m\vec{v}'=\vec{F}$$

Или

$$m\vec{r}''=\vec{F} \qquad (1.2)$$

Проектируя обе части уравнения (1.2) на оси системы координат, можно получить дифференциальные уравнения движения точки в этой системе:

$$\begin{cases} m x''=F_x \\ m y''=F_y \end{cases}$$

Перенесем в правую часть m и подставим вместо F нашу заданную формулу силы притяжения:

$$\begin{cases} x''=\gamma M r^{\alpha}\dfrac{x}{r} \\ y''=\gamma M r^{\alpha}\dfrac{y}{r} \end{cases}$$

Или что то же самое:

$$\begin{cases} \dfrac{d^2x}{dt^2}=\gamma M r^{\alpha}\dfrac{x}{r} \\ \dfrac{d^2y}{dt^2}=\gamma M r^{\alpha}\dfrac{y}{r} \end{cases}$$

Учитывая, что в начальный момент времени t = 0 тело находится на расстоянии R_0 и имеет скорость v_0, параллельную радиус – вектору и учитывая $a=v'$, поставим задачу Коши:

$$\begin{cases} x'' = \gamma M r^{\alpha} \frac{x}{r} \\ y'' = \gamma M r^{\alpha} \frac{y}{r} \\ x = R_0, v_x = 0 \\ y = 0, v_y = v_0 \end{cases}$$

Для удобства примем, что R_0 лежит целиком в оси Ох, тогда вектор v_0 целиком лежит на оси Оу

Мы видим, что оба уравнения данной системы являются ОДУ 2 порядка, тогда для численного решения данной задачи методом Рунге – Кутты необходимо свести каждое из этих уравнений к уравнению 1 порядку

Пусть для первого уравнения $x_1 = x, x_2 = x_1'$, тогда:

$$\begin{cases} x_1' = x_2 \\ x_2' = \gamma M r^{\alpha} \frac{x_1}{r} \\ x_1 = R_0, v_x = 0 \end{cases}$$

Для второго уравнения $y_1 = y, y_2 = y_1'$:

$$\begin{cases} y_1' = y_2 \\ y_2' = \gamma M r^{\alpha} \frac{y_1}{r} \\ y_1 = 0, v_y = v_0 \end{cases}$$

Итого, запишем полученную систему:

$$\begin{cases} \begin{cases} x_1' = x_2 \\ x_2' = \gamma M r^{\alpha} \frac{x_1}{r} \\ x_1 = R_0, v_x = 0 \end{cases} \\ \begin{cases} y_1' = y_2 \\ y_2' = \gamma M r^{\alpha} \frac{y_1}{r} \\ y_1 = 0, v_y = v_0 \end{cases} \end{cases}$$

2.Численное решение задачи

2.1 Метод решения

Для решения задачи будем использовать метод Рунге-Кутты 3 порядка. Для начала запишем его:

¿

Где $\alpha_1=0.1,\alpha_2=0.3$

Выведем соответствующий метод с данными коэффициентами

Разложение Тейлора:

$$f(t+a,y+b)=f+\alpha f_t+bf_y+\frac{1}{2}\left(a^2f_{tt}+2abf_{ty}+b^2f_{yy}\right)$$

Разложение точного решения:

$$y_{i+1}=y_i+ha_1f+h^2/2\left(f_t+f_yf\right)+h^3/6\left(f_{tt}+2f_{ty}f+f_tf_y+f_{yy}f^2+f_yf_yf\right)$$

Разложение k_i по Тейлору:

$$y_{i+1}=y_i+hc_1f+hc_2\left[f+\alpha_1hf_t+\beta_{11}hf_yf+h^2/2\left(\alpha_1f_{tt}+\beta_{11}^2f_{yy}f^2+2f_{ty}\alpha_1\beta_{11}f+O\left(h^3\right)\right)\right]+hc_3\left[f+\alpha_2hf_t+f_y\left(\beta_{21}\right.\right.$$

(*)

Теперь группируем:

$$y_{i+1}=y_i+h\left(c_1+c_2+c_3\right)f+h^2\left[c_2\left(\alpha_1f_t+\beta_{11}f_yf\right)+c_3\left(\alpha_2f_t+\beta_{21}ff_y+\beta_{22}ff_y\right)\right]+h^3\left[\frac{c_2}{2}\left(f_{tt}\alpha_1^2+f_{yy}\beta_{11}^2f^2+2f_{ty}\alpha_1\beta_{11}\right.\right.$$

(**)

Сопоставим (*) и (**):

1) $c_1+c_2+c_3=1$
2) $\alpha_1c_2+\alpha_2c_3=\frac{1}{2}$
3) $\beta_{11}c_2+c_3\left(\beta_{21}+\beta_{22}\right)=\frac{1}{2}$
4) $c_2\alpha+1^2+c_3\alpha_2^2=\frac{1}{3}$
5) $c_2\alpha_{1_{\beta_{11}}}+c_3\alpha_2\left(\beta_{21}+\beta_{22}\right)=\frac{1}{3}$
6) $c_3\beta_{22}\alpha_1=\frac{1}{6}$

7) $c_2\beta_{11}^2+c_3(\beta_{21}+\beta_{22})^2=\frac{1}{3}$

8) $c_3\beta_{22}\beta_{11}=\frac{1}{6}$

Разделим (6) на (8) и получим: $\beta_{11}=\alpha_1=0.1$
Вычтем (7) из (5) и получим $\alpha_2=\beta_{21}+\beta_{22}$
С учетом этого, (2) = (3) и (4) = (5)

$\alpha_1=0.1, \alpha_2=0.3$
Тогда получим из (1), (2), (4):

$$\begin{cases} C_1+C_2+C_3=1 \\ 0.1C_2+0.3C_3=\frac{1}{2} \\ 0.1^2C_2+0.3^2C_3=\frac{1}{3} \end{cases}$$

Решив СЛАУ, получим: $(C_1, C_2, C_3)=\left(\frac{49}{9}, -\frac{55}{6}, \frac{85}{18}\right)$
Из (6): $\beta_{22}=\frac{1\cdot 18}{6\cdot 0.1\cdot 85}=\frac{6}{17}\approx 0.353$
Тогда: $\beta_{21}=\alpha_2-\beta_{22}=\frac{9}{176}\approx -0.053$
Итого, метод Рунге – Кутты 3 порядка с коэффициентами $\alpha_1=0.1, \alpha_2=0.3$ будет выглядеть следующим образом:

¿

2.2 Тесты

При данных в условии константах, а также произвольно определенном расстоянии тела от Земли, равном $R=4.716*R_з[м]$, будем менять лишь начальную скорость:
Также будем рассматривать количество оборотов спутника вокруг Земли и меру близости замкнутой траектории к окружности, определенную следующей формулой:

$$\mu=1-\frac{d}{R},$$
$$d=\max_{Г} ¿¿$$

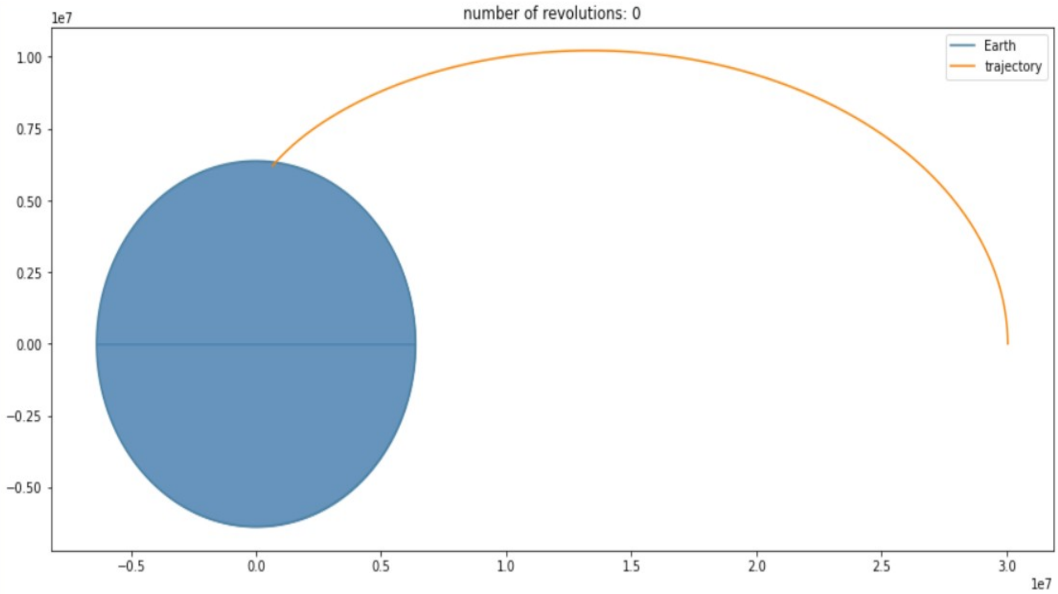
где d – максимальное расстояние между соответсвенными точками траектории и окружности,
$Г-¿$ множество точек траектории полета тела.
Коэффициент μ построен таким образом, что при $d\to 0:\mu\approx 1$ (точки траектории сливаются с точками окружности),

Также для удобства рассматривать меру близости будем лишь для первого оборота

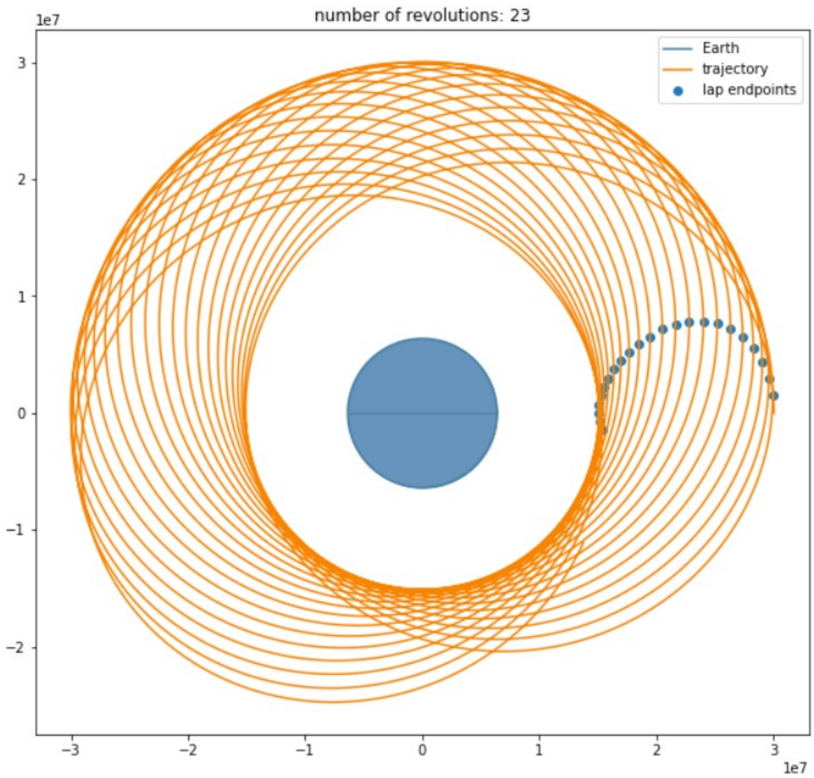
Для круговой траектории период определяется стандартной формулой:

$$T = \frac{2\pi \cdot R}{v_0}$$

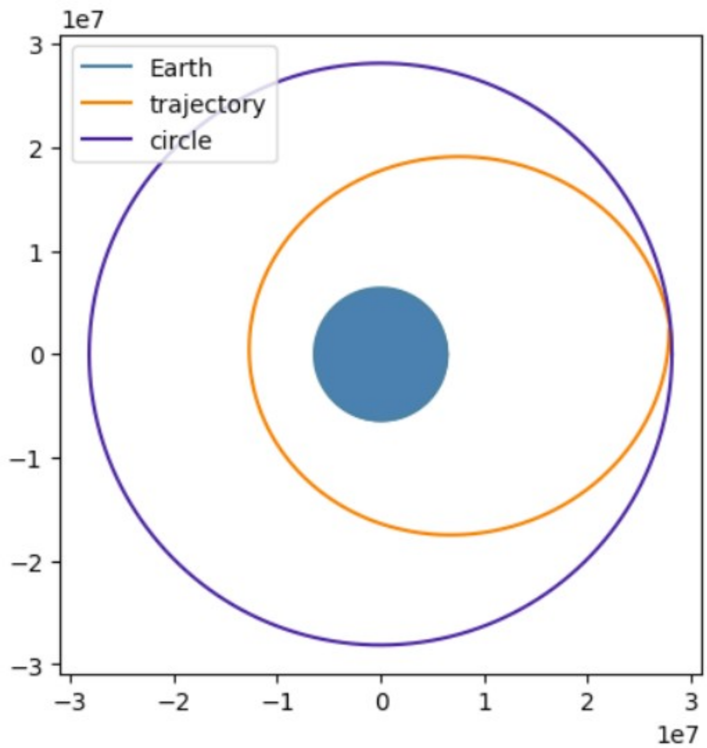
1) при $v_0 = 2000 \left[\frac{м}{с} \right]$ тело совершит жесткую посадку на Землю:



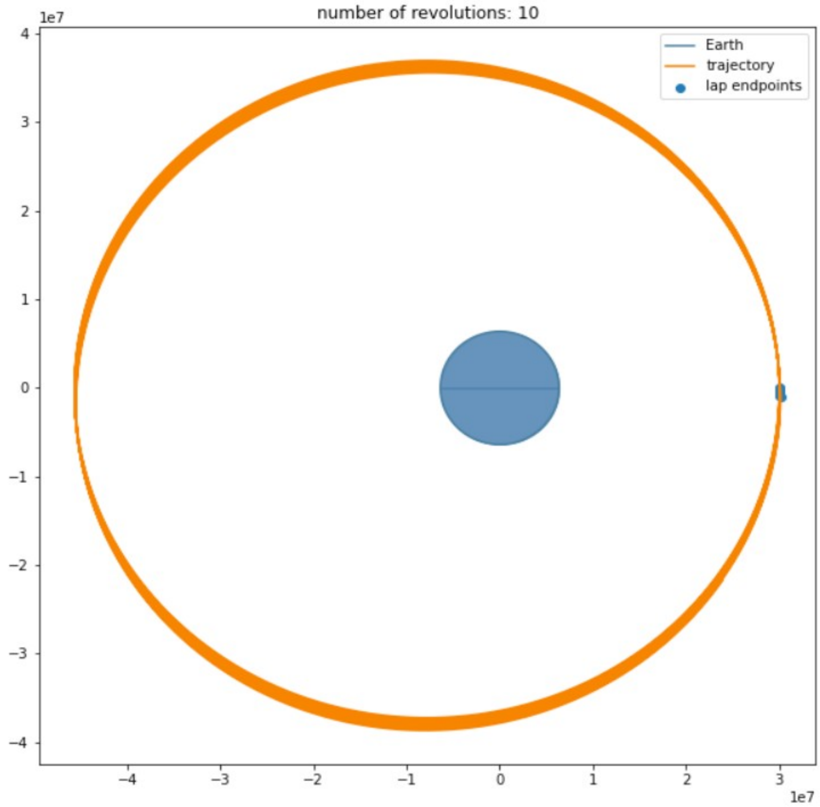
2) При $v_0 = 3000 \left[\frac{м}{с} \right]$ происходит инфинитное движение по незамкнутой траектории:



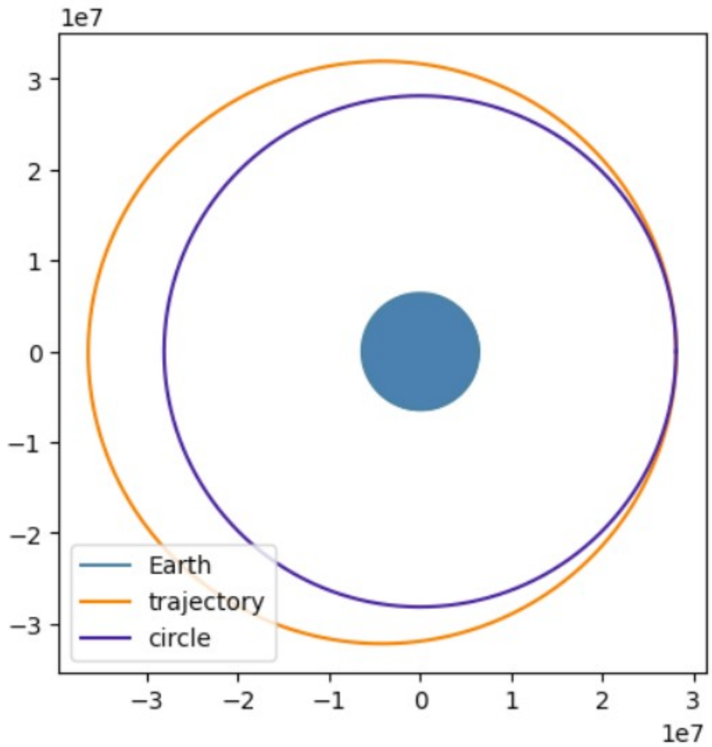
При данной скорости тело совершает 23 оборота вокруг Земли
Период T = 62927 с
Мера близости: $\mu = 0.2670694$



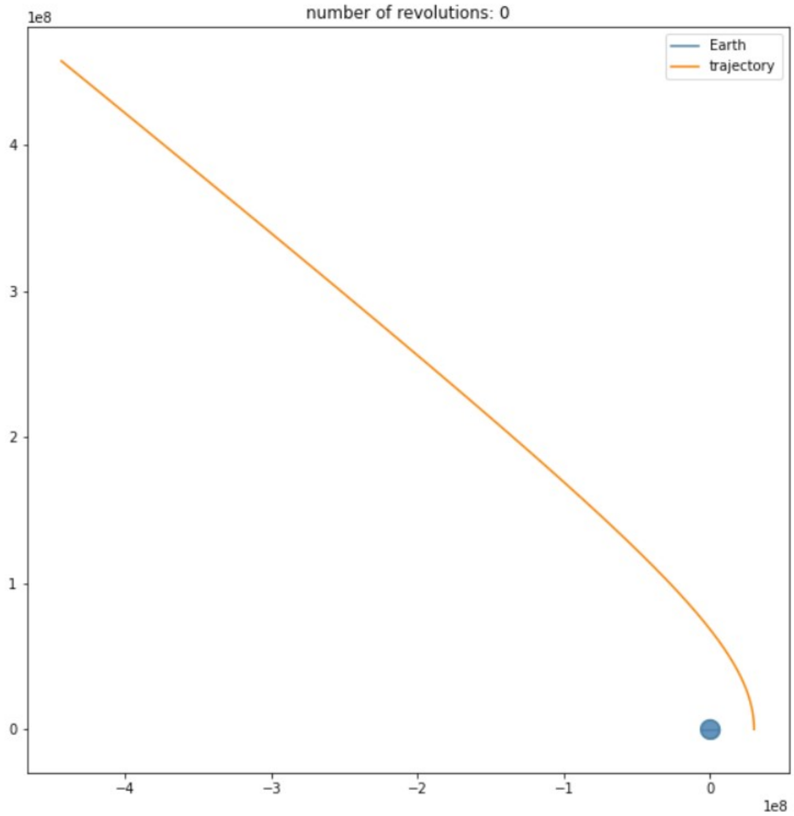
3) При скорости $v_0 = 4000\,\frac{м}{с}$ орбита уже близка к круговой, что удовлетворяет нашим требованиям:



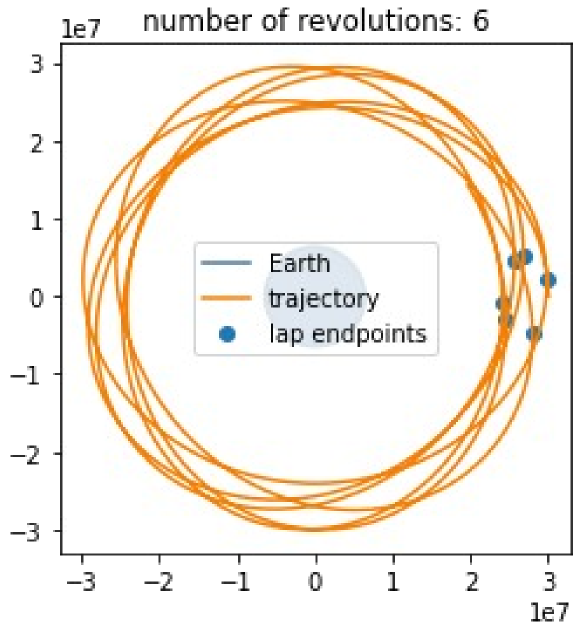
Тело совершает 10 оборотов
Период T = 47195 с
Степень близости: $\mu = 0.6656361$



4) С начальной скоростью $v_0 = 5500\left[\frac{м}{с}\right]$ спутник удаляется более чем на $100 \cdot R_з$ и дальнейшие вычисления прекращаются.



5) Траектория полета с показателем $\alpha = 1$ и $v_0 = 3070\left[\frac{м}{с}\right]$:

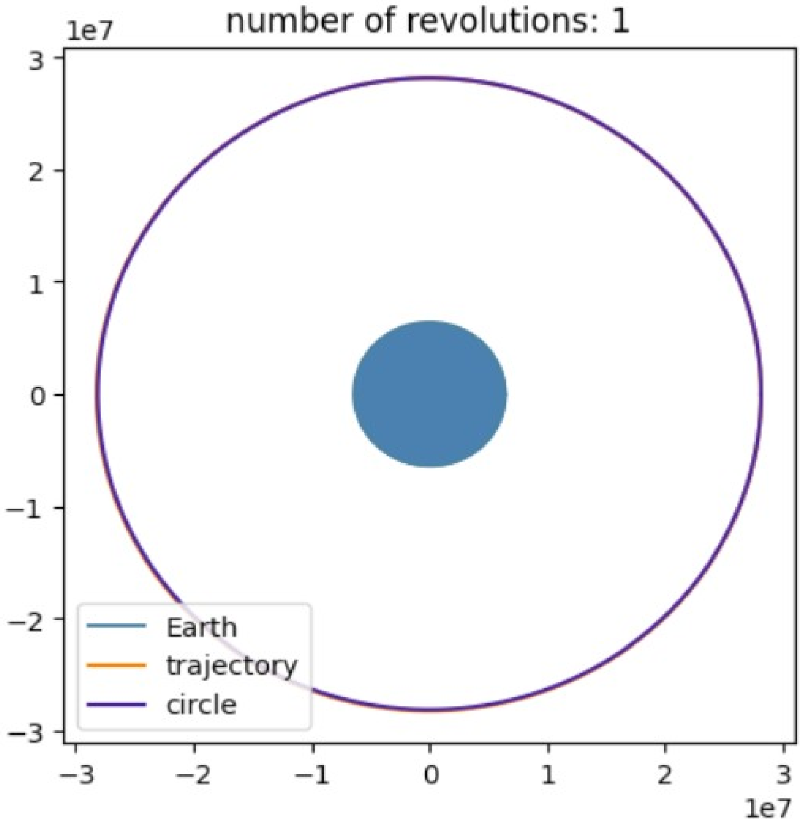


6) Теперь воспользуемся функцией для нахождения скорости, при которой траектория является круговой
Очевидно, также что эта скорость должна лежать на отрезке [3000 ; 4000] м/с

```
find_opt_speed(3000)
```

✓ 18.3s

```
Коэффициент близости:0.9514026472973189
Начальная скорость: 3730
Период T = 47392.29130896883
```



Траектория полета максимально близка к окружности

Заключение

В настоящей работе было рассмотрено гравитационное взаимодействие двух тел, а также движение одного тела относительно источника гравитационного поля – Земли. В соответствии с заданием было проведено математическое моделирование этого процесса: поставлена и решена задача Коши для системы ДУ, использован численный метод решения задачи (Рунге-Кутты), доказана корректность этого метода, произведена подготовка начальных данных для повышения эффективности и скорости работы метода. Также предоставлены фазовые портреты решений, найденных указанным методом. Проведен анализ полученных решений, и, в соответствии с введенной метрикой качества получено наилучшее решение п.5 задания. Также смоделирован процесс движения при различных показателях α (п.6 задания). В результате получилась программа, эффективно решающая поставленную задачу.

Список использованных источников

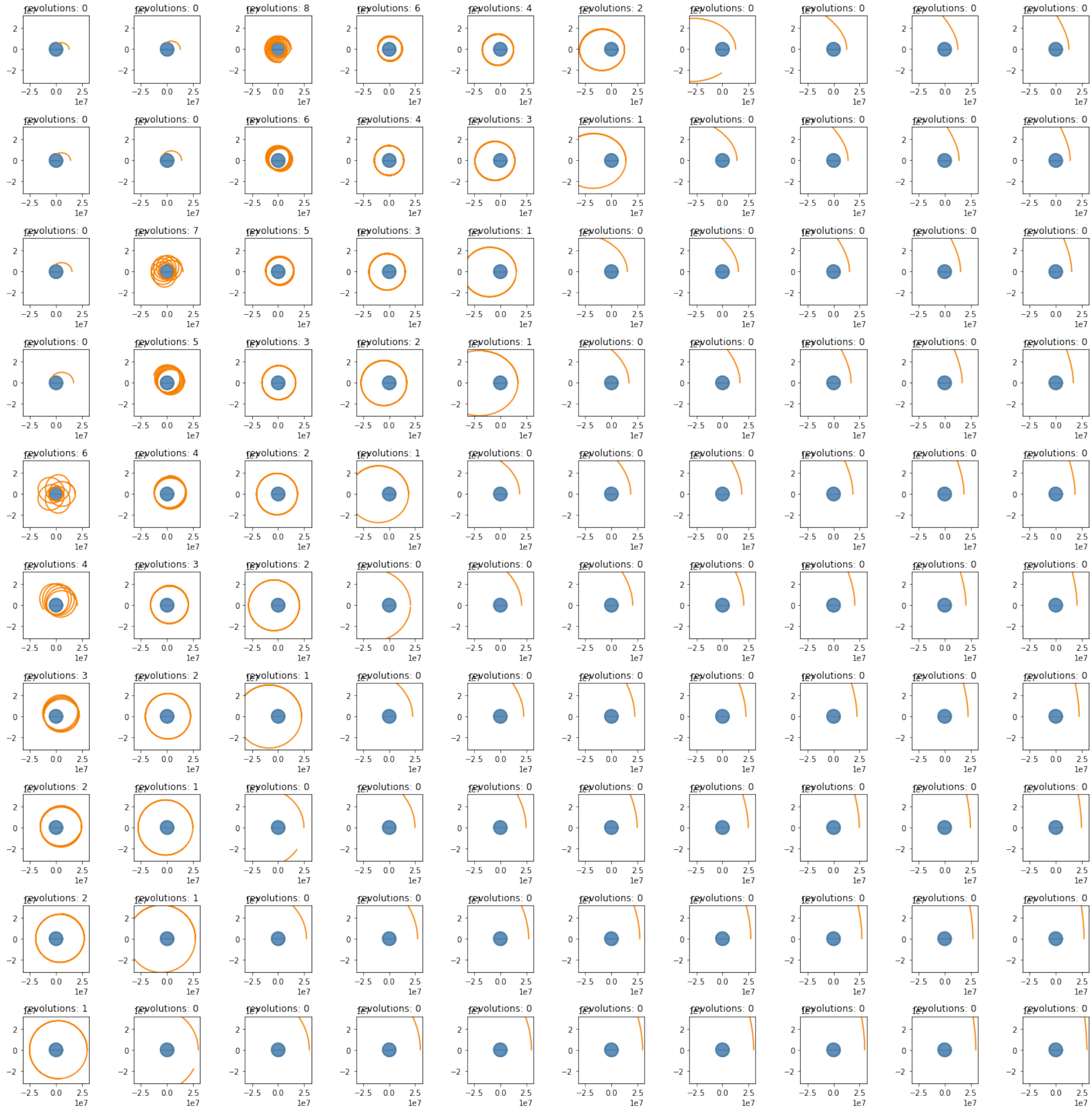
1. **Амосов А. А.** Вычислительные методы: учебное пособие / Амосов А. А., Дубинский Ю. А., Копченова Н.В. – 2-е изд. – СПб. : Издательство «Лань», – 672 с. – ISBN 978-5-8114-1623-3
2. **Амосова О.А.** Численные методы на языке Python: учеб. пособие для вузов / О.А. Амосова, А.Е. Вестфальский, А.В. Князев, Н.Е. Крымов. – М.: Издательство МЭИ, 2022. – 80 с. – ISBN 978-5-7046-2617-6
3. **Никитин Н. Н.** Курс теоретической механики: Учеб. для машиностроит. и приборостроит. спец. вузов. – 5-е изд., перераб. и доп. – М.: Высш. шк., 1990. – 607 с.: ил. ISBN 5-06-000695-6

Приложение

Траектории движения тела с различными начальными данными:

По столбцам начальная скорость изменяется от 3490 до 9 770 м/с

По строкам начальный радиус-вектор изменяется $2 \cdot R_0$ до $5.2 \cdot R_0$



Код программы

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from numpy import sqrt, abs, ceil

G = 6.67 * 10 ** (-11)  # Гравитационная постоянная
M = 5.974 * 10 ** (24)  # масса Земли
m = 1  # масса тела
ALPHA = 3
R0 = 6371 * 10 ** 3


def f_right(u_vec):
    r = sqrt(u_vec[0] ** 2 + u_vec[1] ** 2)
    new_x = (-G * M) * (u_vec[0] / r ** ALPHA)
    new_y = -G * M * (u_vec[1] / r ** ALPHA)
    return np.array((new_x, new_y))


def f_give(u_vec): return u_vec


def fun_x(u_vec):
    r = sqrt(u_vec[0] ** 2 + u_vec[1] ** 2)
    return -G * M * (u_vec[0] / r ** ALPHA)


def fun_y(u_vec):
    r = sqrt(u_vec[0] ** 2 + u_vec[1] ** 2)
    return -G * M * (u_vec[1] / r ** ALPHA)


def K1(u_vec, fun):
    return fun(u_vec)


def K2(u_vec, h, K1, fun):
    return fun(u_vec + 0.1 * h * K1)


def K3(u_vec, h, K1, K2, fun):
    return fun(u_vec - 0.053 * h * K1 + 0.353 * h * K2)


def give_x(u_vec):
    return u_vec[0]


def give_y(u_vec):
    return u_vec[1]


def RK(h, fun, u_vec, proj, norm_k=1):
    K1ex = K1(u_vec, fun) * norm_k
    K2ex = K2(u_vec, h, K1ex, fun) * norm_k
    K3ex = K3(u_vec, h, K1ex, K2ex, fun) * norm_k
    res = proj + (h / 3) * (49 / 3 * K1ex - 55 / 2 * K2ex + 85 / 6 * K3ex)
    return res
```

```python
def solve_RK(h, a, b, velocity, R):
    n = int(ceil(abs(b - a) / h))

    r = np.zeros((n, 2))  # coordinates  # velocities
    r[0, 0] = R  # x
    r[0, 1] = 0  # y

    v = np.zeros((n, 2))  # velocities
    v[0, 0] = 0  # x
    v[0, 1] = velocity  # y

    # normalization

    T0 = R / velocity
    r[0, 0] = 1
    v[0, 1] = 1
    h = h / T0

    norm_k = 1 / (R * velocity ** 2)

    revolutions = 0  # laps counter

    r_dots = []
    for i in range(1, n):

        v[i] = RK(h, f_right, r[i - 1], v[i - 1], norm_k)
        r[i] = RK(h, f_give, v[i], r[i - 1])

        # count revolutions

        if v[i, 1] > 0 and (v[i - 1, 0] > 0 and v[i, 0] <= 0):
            # revolution
            revolutions += 1
            r_dots.append([r[i, 0] * R, r[i, 1] * R])

        if sqrt(r[i, 0] ** 2 + r[i, 1] ** 2) < R0 / R:
            print('Falling')
            return r[:i + 1] * R, v[:i + 1] * velocity, revolutions, np.array(r_dots)

        if sqrt(r[i, 0] ** 2 + r[i, 1] ** 2) > 100 * R0 / R:
            print('Flown away...')
            return r[:i - 1] * R, v[:i - 1] * velocity, revolutions, np.array(r_dots)

    return r * R, v * velocity, revolutions, np.array(r_dots)


a = 0
b = 800000
dt = 30
v0 = 2000
R = 4.716 * R0

rs, vs, revols, r_dots = solve_RK(dt, a, b, v0, R)

# draw Earth
t = np.arange(0, 2 * np.pi, 0.01)
x_earth = np.array(R0 * np.cos(t))
y_earth = np.array(R0 * np.sin(t))

# plt.style.use("dark_background")
fig, ax = plt.subplots(figsize=(10, 10))
```

```python
ax.plot(x_earth, y_earth, label='Earth', c='#457B9D')
ax.fill_between(x_earth, y_earth, color='#457BAD', alpha=1)
ax.plot(rs[:, 0], rs[:, 1], label='trajectory', c='#F77F00')
ax.set(aspect=1)
ax.set_title(f'number of revolutions: {revols}')

if r_dots.size > 0:
    ax.scatter(r_dots[:, 0], r_dots[:, 1], label='lap endpoints')
    # ax.scatter(rs[-1:, 0],r_dots[-1:,1], label='endpoint')

ax.legend()
T_period = 2 * np.pi * R / v0
T_period


def solve_RK_2(h, a, b, velocity, R):
    n = int(ceil(abs(b - a) / h))

    r = np.zeros((n, 2))  # coordinates  # velocities
    r[0, 0] = R  # x
    r[0, 1] = 0  # y

    v = np.zeros((n, 2))  # velocities
    v[0, 0] = 0  # x
    v[0, 1] = velocity  # y

    # normalization

    T0 = R / velocity
    r[0, 0] = 1
    v[0, 1] = 1
    h = h / T0

    norm_k = 1 / (R * velocity ** 2)

    revolutions = 0  # laps counter

    r_dots = []  # endpoints coordinates
    i_dots = []  # endpoints time
    for i in range(1, n):

        v[i] = RK(h, f_right, r[i - 1], v[i - 1], norm_k)
        r[i] = RK(h, f_give, v[i], r[i - 1])

        # count revolutions

        if v[i, 1] > 0 and (v[i - 1, 0] > 0 and v[i, 0] <= 0):
            # revolution
            revolutions += 1
            r_dots.append([r[i, 0] * R, r[i, 1] * R])
            i_dots.append(i)  # get

        if sqrt(r[i, 0] ** 2 + r[i, 1] ** 2) < R0 / R:
            print('Falling')
            return r[:i + 1] * R, v[:i + 1] * velocity, revolutions, np.array(r_dots)

        if sqrt(r[i, 0] ** 2 + r[i, 1] ** 2) > 100 * R0 / R:
            print('Flown away...')
            return r[:i - 1] * R, v[:i - 1] * velocity, revolutions, np.array(r_dots)

    return r * R, v * velocity, revolutions, np.array(r_dots), np.array(i_dots)
```

```python
def find_opt_speed(v0):
    a = 0
    b = 72000
    dt = 30
    R = 4.416 * R0

    while True:
        rs, vs, revols, r_dots, i_dots = solve_RK_2(dt, a, b, v0, R)

        n_points = rs.shape[0]
        t = np.linspace(0, 2 * np.pi, i_dots[0])
        x_earth = np.array(R0 * np.cos(t))
        y_earth = np.array(R0 * np.sin(t))

        x_circle = np.array(R * np.cos(t))

        y_circle = np.array(R * np.sin(t))

        dists = (x_circle - rs[:i_dots[0], 0]) ** 2 + (
                y_circle - rs[:i_dots[0], 1]) ** 2  # dist between traj and circle points
        d = np.max(np.sqrt(dists))
        sim_koef = 1 - d / R
        if sim_koef < 0.95:
            v0 += 10
        elif v0 > 4000:
            print('Скорость превышена')
            break
        else:
            print(f'Коэффициент близости:{sim_koef}\nНачальная скорость: {v0}')
            T_period = 2 * np.pi * R / v0
            print(f'Период T = {T_period}')
            break


find_opt_speed(3000)

plt.style.use("default")
fig, ax = plt.subplots()
ax.plot(x_earth, y_earth, label='Earth', c='#457B9D')
ax.fill_between(x_earth, y_earth, color='#457BAD', alpha=1)

ax.plot(rs[:i_dots[0], 0], rs[:i_dots[0], 1], label='trajectory', c='#F77F00')

ax.plot(x_circle, y_circle, label='circle', c='#451F9D')

ax.set(aspect=1)
ax.set_title(f'number of revolutions: {revols}')

ax.legend(loc='lower left')
```