Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
УЧЕБНЫЙ ЦЕНТР ОБЩЕЙ ФИЗИКИ ФТФ

УНИВЕРСИТЕТ ИТМО

Группа M3202 К работе допущен

Студент Фадеев А. В. Работа выполнена

Преподаватель Тимофеева Э. О. Отчет принят

Отчет по моделированию №1

Маятник Фуко

1. Цель работы.
   - Освоить навык комплексного решения физических и инженерных задач, используя методы численного моделирования процессов.
   - Смоделировать работу Маятника Фуко

2. Задачи, решаемые при выполнении работы.
   - Написание программы для моделирования физического процесса
   - Визуализация результата
   - Поиск траектории движения конца колеблющего маятника на платформе

3. Рабочие формулы.

2. Уравнения движения. На маятник действуют силы Кориолиса, центробежная сила и сила тяжести. Уравнения движения в векторной форме запишутся в виде

$$m\frac{d\vec{v}}{dt} = \vec{F}_T + \vec{F}_K + \vec{F}_{\text{ц.б.}} \qquad (1)$$

или

$$m\frac{d\vec{v}}{dt} = mG\frac{\vec{r}}{L} + 2m[\vec{v}\vec{\omega}] + m\omega^2\vec{r} \qquad (2)$$

Здесь ω – относительная частота; v – относительная скорость. Расписывая по проекциям x и y, получим

$$\frac{dv_x}{dt} = 2v_y\omega + \omega^2 x - g\frac{x}{L},$$
$$\frac{dv_y}{dt} = -2v_x\omega + \omega^2 y - g\frac{y}{L} \qquad (3)$$
$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y.$$

Ускорение свободного падения у поверхности Земли зависит от широты. Приблиз

$$g = 9{,}780318(1 + 0{,}005302 \sin\varphi - 0{,}000006 \sin^2 2\varphi) - 0{,}000003086h,$$

где φ — широта рассматриваемого места,

h — высота над уровнем моря в метрах.

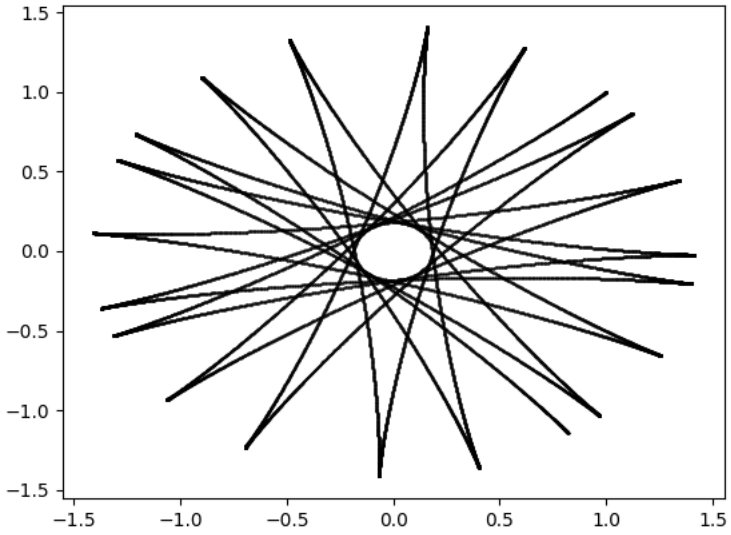
4. Программный код

```
import math
import matplotlib.pyplot as plt
import numpy


class FoucaultPendulum:
    _lst_x = numpy.array([])
    _lst_y = numpy.array([])

    def __init__(self, x, y, vx=0.0, vy=0.0, w=0.04, l=100, dt=0.01, fi=0, h=0):
        self._x, self._y = x, y
        self._vx, self._vy = vx, vy
        self._w, self._l, self._dt = w, l, dt
        self._g = 9.780318 * (1 + 0.005302 * math.sin(fi) - 0.000006 * math.sin(2
* fi) ** 2) + 0.000003086 * h

    def update(self):
        self._vx += (2 * self._vy * self._w + self._w ** 2 * self._x
                     - self._g * self._x / self._l) * self._dt
        self._vy += (-2 * self._vx * self._w + self._w ** 2 * self._y
                     - self._g * self._y / self._l) * self._dt
        self._x += self._vx * self._dt
        self._y += self._vy * self._dt
        self._lst_x = numpy.append(self._lst_x, self._x)
        self._lst_y = numpy.append(self._lst_y, self._y)

    def draw(self, rep=10000):
        for _ in range(rep):
            self.update()

        plt.scatter(self._lst_x, self._lst_y, s=0.15, c='black')
        plt.show()
```

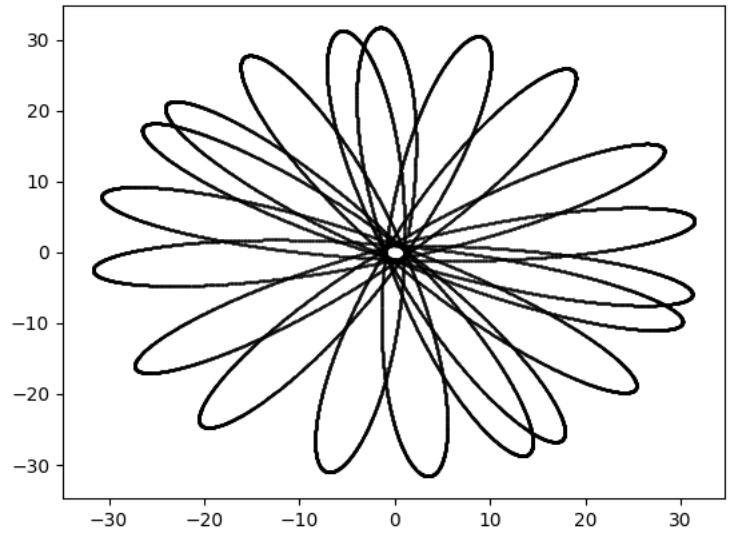
5. Примеры:

g = 9.832 for all

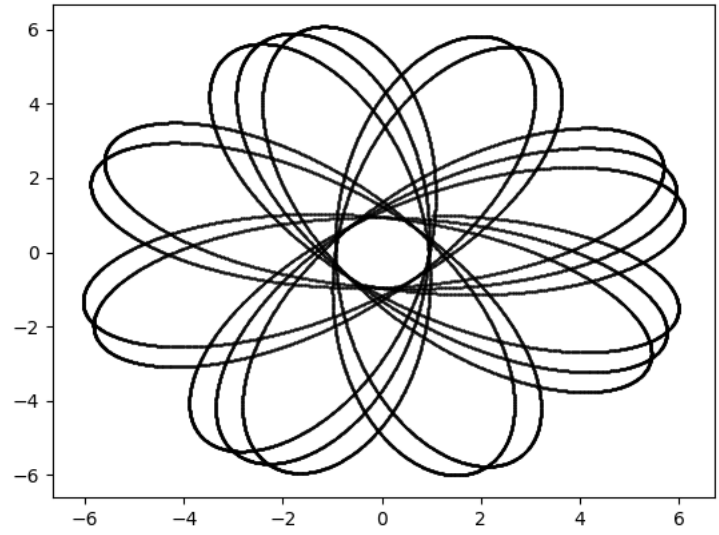
x = 1, y = 1, vx = 0, vy = 0, w = 0.04, l = 100



x = 1, y = 1, vx = 10, vy = 0, w = 0.04, l = 100



x = 1, y = 1, vx = 2, vy = 0, w = 0.08, l = 100



x = 1, y = 1, vx = 2, vy = 0, w = 0.08, l = 300