

Группа _____ М3202 _____ 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

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

или

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

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

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

4. Программный код

```
import matplotlib.pyplot as plt
import numpy

class FoucaultPendulum:
    _lst_x = numpy.array([])
    _lst_y = numpy.array([])
    _g = 9.832

    def __init__(self, x, y, vx=0.0, vy=0.0, w=0.04, l=100, dt=0.01):
        self._x, self._y = x, y
        self._vx, self._vy = vx, vy
        self._w, self._l, self._dt = w, l, dt

    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()
```