Санкт-Петербургский Национальный Исследовательский Университет
ИТМО

Факультет Технологического Менеджмента и Инноваций

Отчет по проектной работе:

“Визуализация маятника Капицы на Python”

Выполнили:
Богданова К.П.
Титкова А.Д.

Санкт-Петербург,
2025

Личный вклад авторов

1. Богданова К.П.: разработка визуализации, подготовка графиков и анимации;
2. Титкова А.Д.: теоретический анализ, анализ устойчивости, оформление отчета.

Цели и задачи

Цель: создание программной модели, демонстрирующей поведение маятника Капицы - возможность устойчивого равновесия в перевернутом положении при высокочастотных колебаниях точки подвеса.

Задачи:

1. Теоретический анализ системы маятника Капицы и вывод уравнений движения;
2. Численное решение полученных дифференциальных уравнений методами SciPy;
3. Разработка на Python алгоритмов анализа устойчивости системы;
4. Визуализация траектории движения маятника в 3D-пространстве на Python;
5. Анализ характерных режимов движения системы.

Актуальность

Маятник Капицы представляет собой яркий пример системы с динамической стабилизацией, демонстрирующий парадоксальное на первый взгляд поведение - возможность устойчивого равновесия в перевернутом положении при определенных условиях. Этот эффект был открыт советским физиком П.Л. Капицей в 1951 году. Теоретическая значимость проекта заключается в демонстрации возможностей стабилизации механических систем за счет высокочастотных колебаний. С практической точки зрения, принципы, лежащие в основе эффекта Капицы, используются при проектировании современных систем стабилизации, в робототехнике (особенно при создании балансирующих роботов), в конструкции различных измерительных приборов и даже в некоторых медицинских устройствах. Исследование подобных систем важно для развития методов управления сложными динамическими системами. Кроме того, данный проект имеет образовательную ценность для студентов, изучающих теоретическую механику и нелинейную динамику.

Этапы работы над проектом

1. Теоретический анализ

1.1. Физическая постановка задачи

Маятник Капицы представляет собой обычный математический маятник, точка подвеса которого совершает высокочастотные вертикальные колебания с частотой ω и амплитудой a. В отличие от классического маятника, такая система может демонстрировать устойчивое равновесие в перевернутом положении (когда груз находится выше точки подвеса).

1.2. Вывод уравнений движения

Уравнения движения были получены из Лагранжева формализма. Положение маятника описывается углами φ (отклонение от вертикали) и θ (азимутальный угол). Учитывая, что точка подвеса совершает колебания по закону $z(t) = a \cdot \cos(\omega t)$, кинетическая и потенциальная энергии системы записываются как:

$$T = \frac{1}{2}ml^2(\dot{\varphi}^2 + \dot{\theta}^2\sin^2\varphi) + ma\omega l \cdot \sin(\omega t) \cdot \sin\varphi \cdot \dot{\varphi}$$
$$U = -mgl \cdot \cos\varphi - ma\omega^2 l \cdot \cos(\omega t) \cdot \cos\varphi$$

После применения уравнений Лагранжа получаем систему дифференциальных уравнений, реализованную в функции lagrangian_equations:

```python
def lagrangian_equations(self, t, y):
    """Уравнения Лагранжа для маятника Капицы"""
    phi, theta, phi_dot, theta_dot = y
    # Ускорение точки подвеса (вторая производная от z0 = a cos(ωt))
    z_acc = -self.a * self.omega**2 * np.cos(self.omega * t)
    # Уравнения движения
    phi_ddot = (self.g/self.l) * np.sin(phi) - (self.a*self.omega**2/self.l) * np.sin(phi) * np.cos(self.omega * t)
    theta_ddot = 0  # Для простоты рассматриваем только движение в плоскости
    return [phi_dot, theta_dot, phi_ddot, theta_ddot]
```

Рисунок 1 – Уравнения движения маятника Капицы

1.3. Условия устойчивости

Теоретический анализ показывает, что перевернутое положение становится устойчивым при выполнении критерия:

$$(a\omega)^2/(2gl) > 1$$

Это условие проверяется в коде при каждом запуске моделирования:

```
omega0 = np.sqrt(self.g / self.l)
        # Условие устойчивости
        self.is_stable = (self.a / self.l) * (self.omega / omega0) > np.sqrt(2)
```

Рисунок 2 – Анализ устойчивости

2. Численное моделирование

Для решения системы дифференциальных уравнений использовался метод DOP853 из библиотеки SciPy, который обеспечивает высокую точность вычислений. Особое внимание было уделено выбору начальных условий - маятник запускается из положения, близкого к перевернутому ($\varphi = \pi + 0.01$), чтобы наблюдать его поведение вблизи точки предполагаемого равновесия.

```
sol = solve_ivp(
    self.lagrangian_equations,
    [0, t_max],
    [initial_angle, 0.0, 0.0, 0.0],  # Начальные условия близкие к вертикали
    method='DOP853',
    rtol=1e-8
)
```

Рисунок 3 – Решение системы дифференциальных уравнений

Реализована функция для автоматического определения амплитуды колебаний подвеса, обеспечивающей границу устойчивости согласно критерию Капицы:

```
def calculate_amplitude(self):
    """Автоматический расчет амплитуды"""
    omega0 = np.sqrt(self.g / self.l)
    # Согласно формуле: (a/l)*(ω/ω0) > √2
    # Для демонстрации эффекта возьмем амплитуду, соответствующую границе устойчивости
    return self.l * np.sqrt(2) * omega0 / self.omega
```

Рисунок 4 – Автоматический расчет амплитуды

3. Визуализация результатов

3.1. Трехмерная анимация

Основная визуализация представляет собой 3D-модель маятника, где:

- Синяя точка отмечает положение точки подвеса;
- Красная точка - груз маятника;
- Зеленая линия показывает траекторию движения груз.

```python
def animate(i):
    # Обновление положения маятника
    self.line.set_data([0, self.x[i]], [0, self.y[i]])
    self.line.set_3d_properties([self.z_pivot[i], self.z[i]])

    self.bob.set_data([self.x[i]], [self.y[i]])
    self.bob.set_3d_properties([self.z[i]])

    self.pivot.set_data([0], [0])
    self.pivot.set_3d_properties([self.z_pivot[i]])

    # Обновление траектории
    self.history[0].append(self.x[i])
    self.history[1].append(self.y[i])
    self.history[2].append(self.z[i])
```

Рисунок 5 – Визуализация маятника

3.2. Аналитические графики

Дополнительно построены:

- График эффективного потенциала
- График усредненного движения
- Фазовый портрет системы
- Быстрая компонента

```python
# График эффективного потенциала
phi_range = np.linspace(0, 2*np.pi, 100)
omega0 = np.sqrt(self.g / self.l)
A = 0.25 * (self.a/self.l)**2 * (self.omega/omega0)**2  # Коэффициент
U_eff = -self.g * self.l * np.cos(phi_range) + self.g * self.l * A * np.sin(phi_range)**2
self.ax_pot.plot(phi_range, U_eff, 'b-')
self.ax_pot.set_title(r"Эффективный потенциал $U_{eff}(\phi)$")
self.ax_pot.set_xlabel(r'Угол $\phi$ (рад)')
self.ax_pot.set_ylabel('Потенциальная энергия')
```

Рисунок 6 – График эффективного потенциала

```python
# Усреднение по быстрым колебаниям (разделение на медленную и быструю составляющие)
# Усреднение методом скользящего среднего
window_size = int(2 * np.pi / (self.omega * (self.t_eval[1] - self.t_eval[0])))
if window_size % 2 == 0:
    window_size += 1

self.X = np.convolve(self.phi, np.ones(window_size)/window_size, mode='same')
self.xi = self.phi - self.X
```

Рисунок 7 – Усредненное движение

```python
self.xi = self.phi - self.X
```

Рисунок 8 – Выделение быстрой компоненты

```
self.ax_fast.plot(self.t_eval, self.xi, 'r-')
self.ax_fast.set_title("Быстрая составляющая ξ(t)")
self.ax_fast.set_xlabel('Время (с)')
self.ax_fast.set_ylabel('Угол (рад)')
```

Рисунок 9 – Построение графика быстрой компоненты движения

```
dX_dt = np.gradient(self.X, self.t_eval)
self.ax_phase.plot(self.X, dX_dt, 'b-')
self.ax_phase.set_title("Фазовый портрет X(t)")
self.ax_phase.set_xlabel('Угол (рад)')
self.ax_phase.set_ylabel('Угловая скорость (рад/с)')
```

Рисунок 10 – Расчет производной усредненного движения и построение фазового портрета

Выносимые на защиту результаты

1. **Подтверждение теоретического критерия устойчивости**
   Численное моделирование подтвердило, что перевернутое положение маятника действительно становится устойчивым при выполнении условия $(a\omega)^2/(2gl) > 1$. В программе это отображается изменением цвета заголовка на зеленый при выполнении критерия устойчивости.
2. **Характерные режимы движения**
   Обнаружены три характерных режима:

- Неустойчивый (маятник быстро падает);
- Переходный (маятник совершает сложные колебания);
- Устойчивый (маятник сохраняет положение вблизи перевернутого состояния).

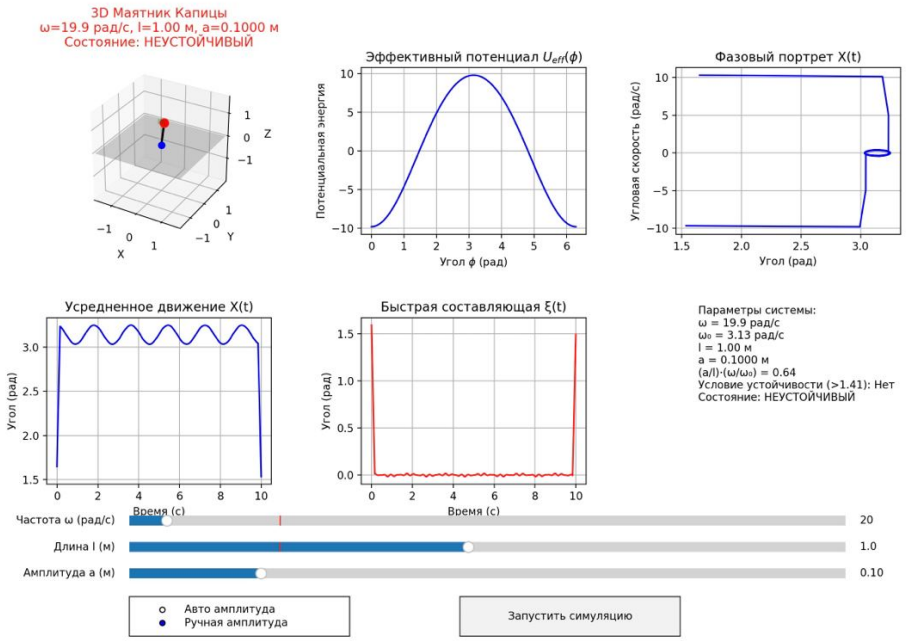

Рисунок 11 – Неустойчивый режим

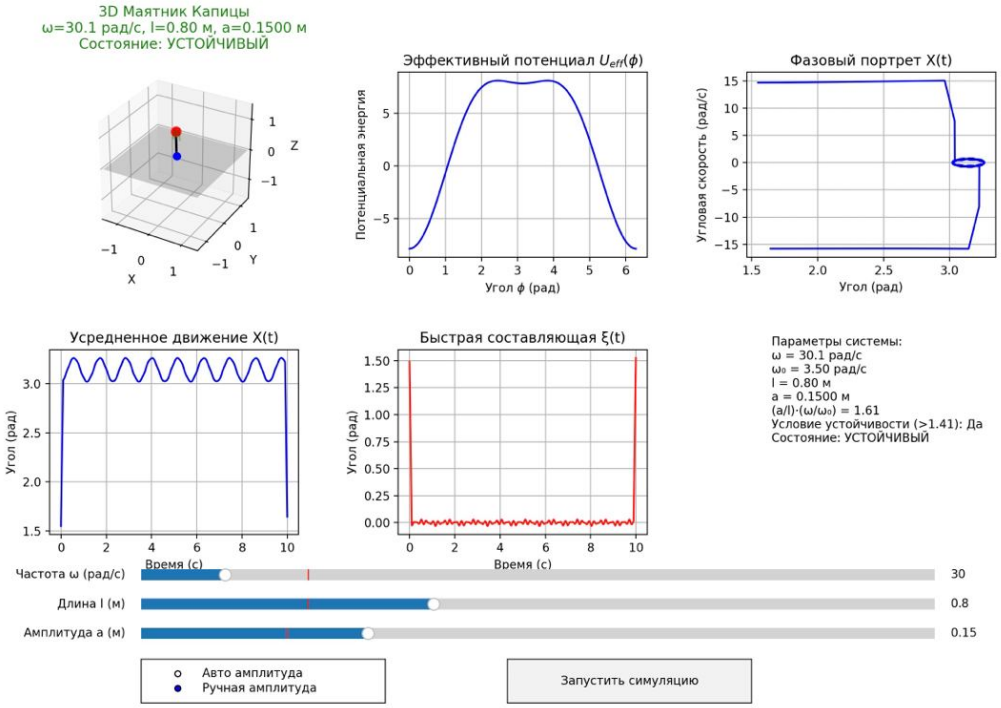

Рисунок 12 – Граничный случай

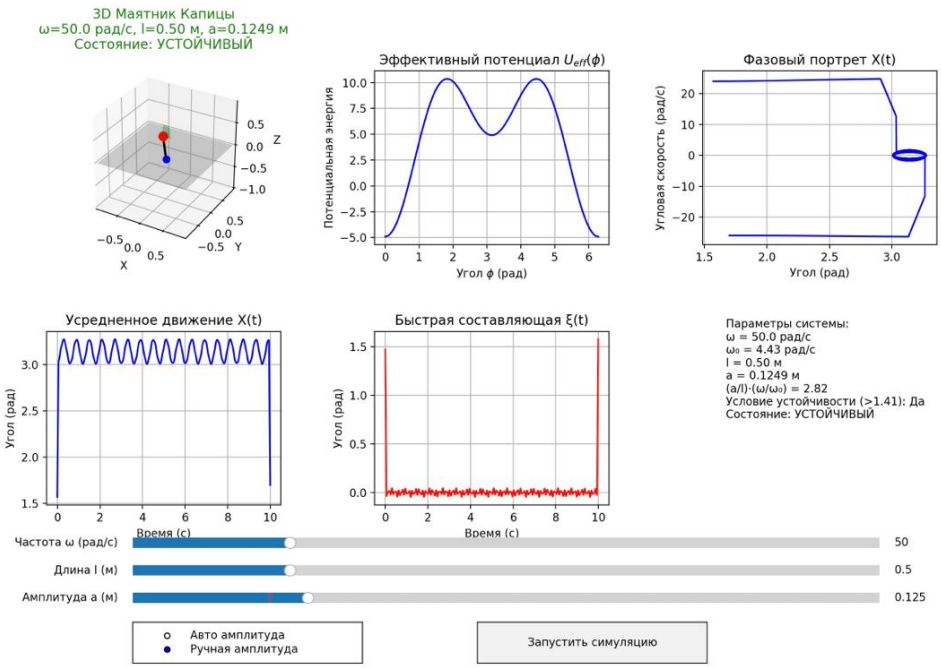

Рисунок 13 – Устойчивый режим

3. **Эффективный потенциал**
   Анализ эффективного потенциала U_eff(φ) = -g·cosφ + (a²ω²/4l)sin²φ показал, что при выполнении условия устойчивости в точке φ = π действительно образуется минимум потенциала, что соответствует устойчивому положению равновесия.
4. **Практическая значимость**
   Разработанная модель позволяет:

- Наглядно демонстрировать эффект динамической стабилизации;
- Исследовать влияние параметров системы (ω, l, a) на устойчивость;
- Служить основой для модификаций (учет трения, пространственный случай);
- Использоваться как учебное пособие по нелинейной динамике.