

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Проект по аналитической механике:
Изучение резонанса маятника Капицы

Шевелева Дарья
Солоджук Никита
Группы Б03-301 и Б03-305

Долгопрудный, 2025 г.

Цель работы: Изучение устойчивости маятника Капицы с точки зрения аналитической механики, получение условия резонанса и проверка с помощью модели.

1 Теоретическая справка

Маятником Капицы называется система, состоящая из грузика, прикреплённого к лёгкой нерастяжимой спице, которая крепится к вибрирующему подвесу. Маятник носит имя академика и нобелевского лауреата П. Л. Капицы, построившего в 1951 году теорию для описания такой системы.

В работе рассматривался случай маятника Капицы с вертикально осциллирующим подвесом с колебаниями малой амплитуды. Модель изображена на рис. 1: ϕ - угол отклонения маятника от вертикали, y_0 - амплитуда малых колебаний, m - масса маятника, l - длина маятника, $\omega_0 = \frac{g}{l}$ - частота собственных колебаний маятника, ω - частота колебаний подвеса. Также для гашения малых отклонений в модель был добавлен коэффициент трения в шарнире γ .

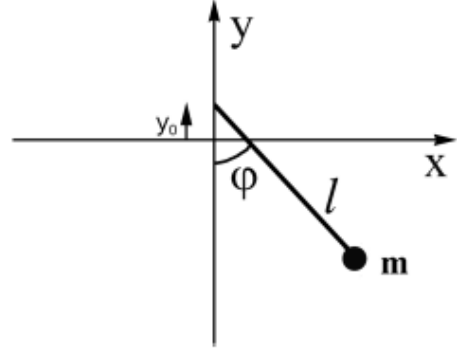


Рис. 1: Модель маятника

Перейдём в систему отсчёта, связанную с осциллирующим подвесом. Тогда на груз будет действовать вертикальная сила инерции $y_0\omega^2\cos(\omega t)$. После проекции сил на ось, перпендикулярную маятнику, уравнение движения:

$$\ddot{\phi} + 2\gamma(\dot{\phi} - \omega_p) + \left(\frac{g}{l} - \frac{y_0\omega^2}{l}\cos(\omega t)\right)\sin(\phi) = 0$$

1.1 Резонанс 0:1

Рассмотрим случай, при котором маятник стремится к положению равновесия с нулевой угловой скоростью. Тогда член с γ равен нулю.

Сделаем замену $\phi = \psi + \beta$, где ψ - медленная компонента угла, а β - быстрая. Поскольку β - быстрая компонента:

$$\begin{aligned}\ddot{\beta} &= \ddot{\phi}, \\ \ddot{\beta} &= -\left(\omega_0^2 - \frac{y_0}{l}\omega^2\cos(\omega t)\right)\sin(\psi + \beta), \\ \ddot{\beta} &= -\frac{\omega^2 y_0}{l}\sin(\psi)\cos(\omega t).\end{aligned}$$

Усредняя по периоду $T = \frac{2\pi}{\omega}$:

$$\begin{aligned}\ddot{\phi} &= \ddot{\psi}, \\ \ddot{\psi} &= -\frac{1}{T}\int_0^T \left(\omega_0^2 - \frac{y_0}{l}\omega^2\cos(\omega t)\sin(\phi)\left(\sin(\psi)\cos(\beta) + \cos(\gamma)\sin(\beta)\right)\right) dt.\end{aligned}$$

Приближения $\sin(\beta) = \beta$, $\cos(\beta) = 1 - \frac{\beta^2}{2}$:

$$\ddot{\psi} = -\frac{1}{T} \int_0^T \left(\omega_0^2 \sin(\psi) \left(1 - \frac{y_0^2 \sin(\psi)}{l^2} \cos^2(\omega t) \right) + \frac{y_0}{l} \omega^2 \left(\frac{y_0^2 \sin^2(\psi)}{2l^2} \cos^3(\omega t) + \frac{a \sin(2\psi)}{2l} \cos^2(\omega t) \right) \right) dt.$$

Учтём малость $\frac{y_0^2}{l^2}$ и проинтегрируем:

$$\ddot{\psi} = - \left(\omega_0^2 \sin(\psi) + \frac{y_0^2 \omega^2 \sin(2\psi)}{4l} \right).$$

Найдём усреднённый за период момент всех сил M :

$$\begin{aligned} ml^2 \ddot{\phi} &= M, \\ \ddot{\phi} &= \ddot{\psi}, \\ M &= -ml \left(g \sin(\psi) + \frac{y_0^2 \omega^2 \sin(2\psi)}{4l} \right). \end{aligned}$$

Введём функцию потенциала $V(\psi)$ такую, что $\frac{dV}{d\psi} = -M$:

$$\begin{aligned} V(\psi) &= ml \int \left(g \sin(\psi) + \frac{y_0^2 \omega^2 \sin(2\psi)}{4l} \right) d\psi, \\ V(\psi) &= -mgl \cos(\psi) - \frac{my_0^2 \omega^2 \cos^2(\psi)}{4} + C. \end{aligned}$$

Полученное выражение — парабола от $\cos(\psi)$. Минимум при

$$\cos(\psi) = -\frac{b}{2a} = -\frac{2gl}{(y_0\omega)^2},$$

так как $\cos(\psi) > -1$ следует $(y_0\omega)^2 > 2gl$.

Условия равновесия и устойчивости:

$$\begin{aligned} \frac{dV}{d\psi} &= -M = 0, \\ \frac{d^2V}{d\psi^2} &> 0. \end{aligned}$$

Найдём производные:

$$\begin{aligned} \frac{dV}{d\psi} &= ml \left(g \sin(\psi) + \frac{y_0^2 \omega^2 \sin(2\psi)}{4l} \right), \\ \frac{d^2V}{d\psi^2} &= ml \left(g \cos(\psi) + \frac{y_0^2 \omega^2 \cos(2\psi)}{2l} \right). \end{aligned}$$

Отсюда положения равновесия: $\psi = 0; \pi$ - устойчивые.

1.2 Резонанс 1:1

Рассмотрим случай резонанса, при котором ω_p частота вращения маятника совпадает с частотой осцилляций подвеса ω . Для этого сделаем замену $\psi = \phi - \omega t$:

$$\ddot{\psi} + 2\gamma(\dot{\psi}) + (\omega_0^2 - \frac{y_0}{l}\omega^2 \cos(\omega t)) \sin(\psi + \omega t) = 0$$

Так как рассматриваем случай совпадения угловых скоростей, член с $\dot{\psi}$ зануляется ввиду малости отклонений. Рассмотрим вклад члена с $\sin(\psi + \omega t)$ за период $T = \frac{2\pi}{\omega}$:

$$\frac{1}{T} \int_0^T \sin(\omega t + \psi) dt = 0 \rightarrow \text{вклад за период от } g = 0.$$

Член с $\sin(\psi + \omega t)\cos(\omega t)$:

$$\frac{1}{T} \int_0^T \sin(\omega t + \psi)\cos(\omega t) dt = \frac{1}{2T} \int_0^T (\sin(\psi) + \sin(2\omega t + \psi)) dt = \sin(\psi)/2$$

Уравнение движения в приближении за период:

$$\ddot{\psi} + \frac{y_0\omega^2}{2l} \sin(\psi) = 0$$

Отсюда положения равновесия $\psi = 0$ - устойчивое, $\psi = \pi$ - неустойчивое. Причем условие устойчивости первого положения равновесия - то, что маятник может сделать оборот вокруг своей оси. Из ЗСЭ:

$$\begin{aligned} mg \cdot 2l &< m \frac{\omega^2 l^2}{2} \\ \frac{\omega^2 l}{4g} &> 1 \\ \frac{\omega}{\omega_0} &> 2 \quad (\text{условие устойчивости}) \end{aligned}$$

1.3 Резонанс n:1

Рассмотрим резонансы при скорости вращения маятника больше скорости осцилляций подвеса. Уравнение движения:

$$\ddot{\phi} + (\frac{g}{l} - \frac{y_0\omega^2}{l} \cos(\omega t)) \sin(\phi) = 0$$

Замена $\phi = n\omega t + \psi$:

$$\ddot{\phi} + (\omega_0^2 - \frac{y_0\omega^2}{l} \cos(\omega t)) \sin(n\omega t + \psi) = 0$$

Усредняя за $T = \frac{2\pi}{n\omega}$:

$$\begin{aligned}
 \frac{1}{T} \int_0^T \sin(n\omega t + \psi) dt &= 0 \\
 \frac{1}{T} \int_0^T \cos(\omega t) \sin(n\omega t + \psi) dt &= \frac{1}{2T} \int_0^T [\sin((n+1)\omega t + \psi) \\
 &\quad - \sin((1-n)\omega t - \psi)] dt \\
 &= -\frac{1}{2\omega T} \left[\frac{\cos((n+1)\omega t + \psi)}{n+1} + \frac{\cos((n-1)\omega t - \psi)}{n-1} \right]_0^T \\
 &= -\frac{n}{4\pi} \left[\frac{\cos(2\pi + \psi + \frac{2\pi}{n})}{n+1} + \frac{\cos(2\pi - \psi - \frac{2\pi}{n})}{n-1} \right. \\
 &\quad \left. - \frac{\cos \psi}{n+1} - \frac{\cos \psi}{n-1} \right] \\
 &= \frac{n}{4\pi(n^2-1)} \left[2n \cos \psi - (n-1) \cos \left(2\pi + \psi + \frac{2\pi}{n} \right) \right. \\
 &\quad \left. - (n+1) \cos \left(2\pi - \psi - \frac{2\pi}{n} \right) \right] \\
 &= \frac{n^2}{2\pi(n^2-1)} [\cos \psi - \cos \left(\frac{2\pi}{n} + \psi \right)] \\
 &= \frac{n^2}{\pi(n^2-1)} \sin \left(\frac{\pi}{n} + \psi \right) \sin \left(\frac{\pi}{n} \right)
 \end{aligned}$$

Подставляя в уравнение движения:

$$\ddot{\psi} + \frac{y_0 \omega^2}{2l} \frac{n^2}{\pi(n^2-1)} \sin\left(\frac{\pi}{n}\right) \sin\left(\frac{\pi}{n} + \psi\right) = 0$$

Из уравнения видно, что при $n > 1$ резонанс возможен при сдвиге фазы на $\frac{\pi}{n}$, причем так же будет 2 противоположных положения равновесия, как в случае Резонанса 1:1, с такой же устойчивостью/неустойчивостью.

2 Пояснение кода

Мы не будем подробно останавливаться на каждом случае, а лишь рассмотрим пример для 1:1. Остальные вариации отличаются незначительно, с ними можно будет ознакомиться в приложении к работе.

2.1 Код для резонанса 1:1

2.1.1 Импорт библиотек

Для начала импортируем необходимые библиотеки:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

- `numpy` для численных вычислений
- `matplotlib.pyplot` для визуализации
- `solve_ivp` из SciPy для решения дифференциальных уравнений

2.1.2 Функция для моделирования уравнений

Для моделирования маятника необходимо написать функцию, которая содержит в себе систему дифференциальных уравнений движения для маятника.

```
def equation_of_rotation(t, y, omega0, omega, gamma, a, L):
    theta, phi = y
    dydt = [
        phi,
        -2 * gamma * (phi - omega) +
        (omega0**2 - (a/L) * omega**2 * np.cos(omega*t)) * np.sin(theta)
    ]
    return dydt
```

Где:

- `theta` - угол отклонения
- `phi` - угловая скорость
- Уравнение включает:
 - Демпфирование с коэффициентом `gamma`
 - Воздействие вибрации с амплитудой `a` и частотой `omega`
 - Гравитационное воздействие с частотой `omega0`

2.1.3 Функция для матрицы устойчивости

Для того, чтобы посмотреть на устойчивость системы, необходимо посчитать собственные значения матрицы. Эту задачу решает следующая функция:

```
def sustainability_matrix(params, T, sol_main):
    omega = params['omega']
    omega0 = params['omega0']
    gamma = params['gamma']
    a = params['a']
    L = params['L']

    def variational_eq(t, y):
        xi, dxi = y # xi = theta - theta0, dxi = phi - phi0
        theta0 = sol_main.sol(t)[0]
        coeff = (omega0**2 - (a/L)*omega**2 *
                 np.cos(omega*t)) * np.cos(theta0)

        return [
            dxi,
            -2*gamma*dxi + coeff*xi
        ]

    sol1 = solve_ivp(variational_eq, [0, T],
                     [1, 0], method='RK45', rtol=1e-8)
    sol2 = solve_ivp(variational_eq, [0, T],
                     [0, 1], method='RK45', rtol=1e-8)
```

```

M = np.array([
    [sol1.y[0, -1], sol2.y[0, -1]],
    [sol1.y[1, -1], sol2.y[1, -1]]
])
return M

```

- Решает вариационные уравнения для двух начальных условий.
- Строит матрицу из конечных состояний решений
- Возвращаемая матрица **M** позволяет определить устойчивость:
 - Если все $|\lambda(M)| < 1$ - система устойчива
 - Если $\exists |\lambda(M)| \geq 1$ - система неустойчива

2.1.4 Функции для создания графиков

Хотя детали численного решения и физических уравнений могут быть достаточно сложными, визуализационная часть представляет собой относительно стандартную задачу:

- Строятся четыре взаимосвязанных графика: временная зависимость угла отклонения, временная зависимость угловой скорости, фазовый портрет, график бесконечно малой добавки $\xi(t)$
- Используются стандартные методы matplotlib (subplot, plot, grid) и настройки оформления (подписи осей, легенды) следуют типовым шаблонам

2.1.5 Основной блок выполнения

Эта функция задает параметры модели и при запуске программы вызывает функции, описанные выше:

```

if __name__ == "__main__":
    params = {
        'omega0': np.sqrt(9.8),
        'omega': 4 * np.sqrt(9.8),
        'gamma': 0.1,
        'a': 0.01,
        'L': 1.0
    }
    y0 = [0, params['omega']]
    simulate_and_analyze(params, y0, t_span=(0, 30), save_gif=False)

```

2.2 Пример работы для двух положений равновесия для 0:1

- Рассмотрим систему начальными данными y_0 и ω (см. рис. 3). Тогда получим следующие результаты:

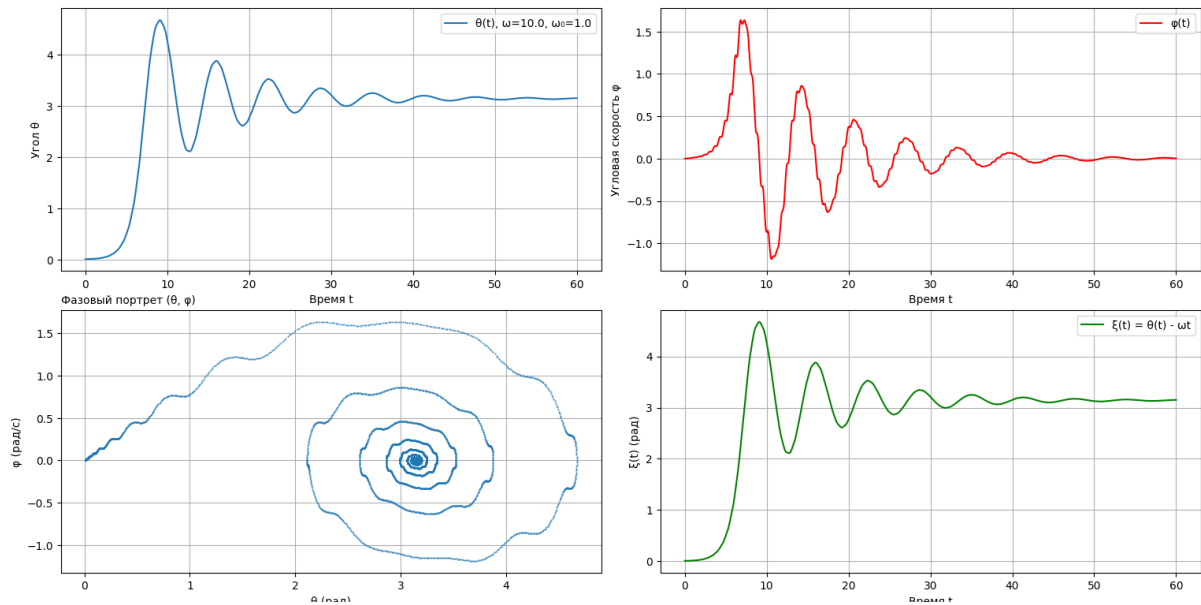


Рис. 2: Графики зависимостей для $w = 10$

Из теории вытекает, что при таких НУ система должна быть неустойчивой, что согласуется с нашим кодом:

```

if __name__ == '__main__':
    params = {
        'omega0': 1,
        'omega': 10 * 1,
        'gamma': 0.1,
        'a': 0.01,
        'L': 1.0,
        'n': 0
    }
    y0 = [0.01, params['omega'] * params['n']]
    simulate_and_analyze(params, y0, t_span=(0, 60))

```

simulate_and_analyze() > else
 script_not_ani x script_4 x
 C:\Users\User\PycharmProjects\stepik\venv\Scripts\python.exe C:\Users\User\Desktop\script_4.py
 Собственные значения: 1.7631, 0.5002
 Модули: 1.7631, 0.5002
 Система неустойчива

Рис. 3: Анализ устойчивости для $w = 10$

- Теперь возьмем другие начальные условия(см. рис. 5). Тогда получим:

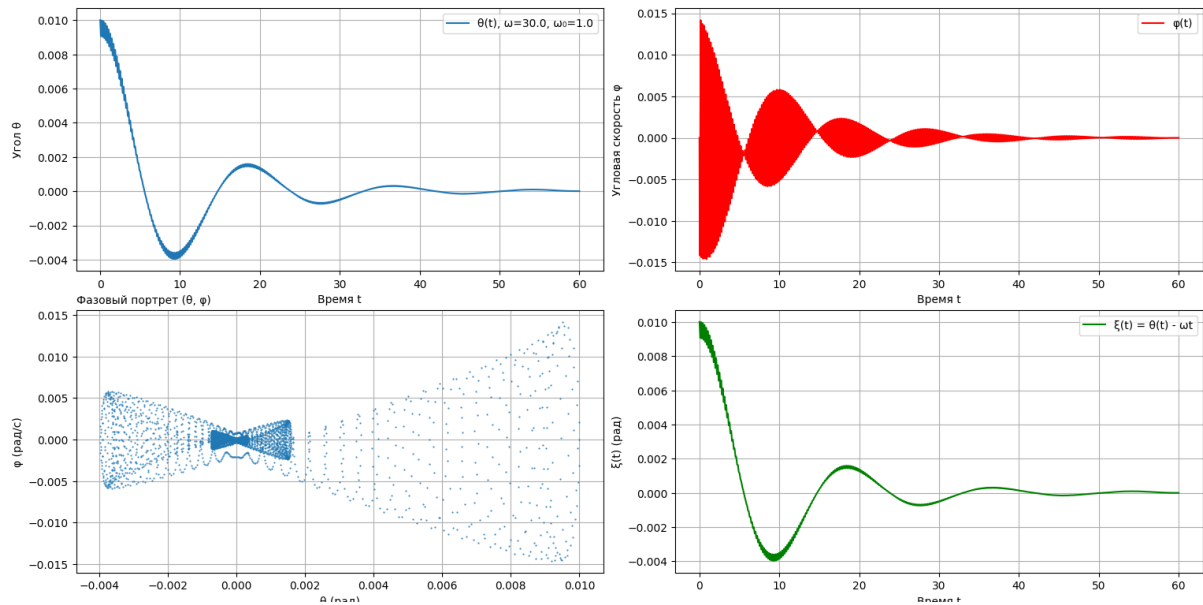


Рис. 4: Графики зависимостей для $w = 30$

Из теории вытекает, что при таких НУ система должна быть устойчивой, что вновь согласуется с нашим кодом:

```

if __name__ == "__main__":
    params = {
        'omega0': 1,
        'omega': 30 * 1,
        'gamma': 0.1,
        'a': 0.05,
        'L': 1.0,
        'n': 0
    }
    y0 = [0.01, params['omega'] * params['n']]
    simulate_and_analyze(params, y0, t_span=(0, 60))

```

script_not_ani x script_4 x
 C:\Users\User\PycharmProjects\stepik\venv\Scripts\python.exe C:\Users\User\Desktop\script_4.py
 Собственные значения: 0.9768+0.0692j, 0.9768-0.0692j
 Модули: 0.9793, 0.9793
 Система устойчива
 Process finished with exit code 0

Рис. 5: Анализ устойчивости для $w = 30$

2.3 Пример работы для двух положений равновесия для 1:1

- Рассмотрим сначала систему со следующими начальными данными: начальный угол равен 0 и начальная угловая скорость равна возбуждающей скорости. Тогда получим следующие результаты:

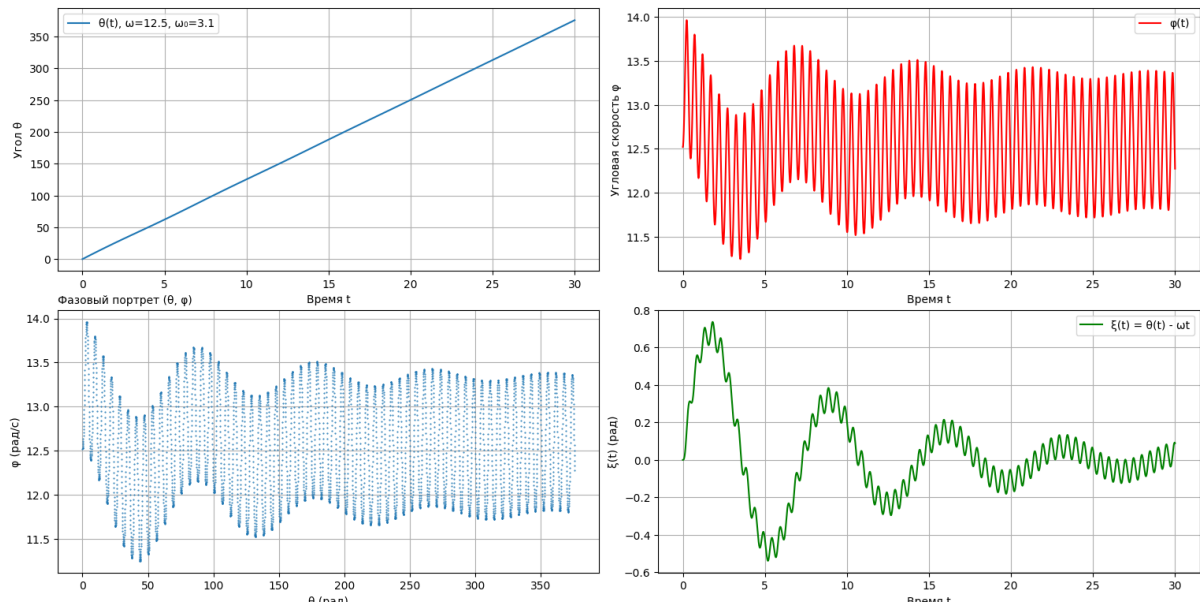


Рис. 6: Графики зависимостей для 0

Из теории вытекает, что при таких НУ система должна быть устойчивой, что согласуется с нашим кодом:

```

175 if __name__ == "__main__":
176     params = {
177         'omega0': np.sqrt(9.8), # корень из g / L
178         'omega': 4 * np.sqrt(9.8), # Частота вращ
179         'gamma': 0.1,
180         'a': 0.01,
181         'L': 1.0
182     }
183     y0 = [0, params['omega']]
184     simulate_and_analyze(params, y0, t_span=(0, 30), save_gif=False)

```

simulate_and_analyze()

```

un: script_2 x script x
C:\Users\User\PycharmProjects\stepik\venv\Scripts\python.exe C:\Users\User\Desktop\script.py
Собственные значения: 0.9092+0.2791j, 0.9092-0.2791j
Модули: 0.9511, 0.9511
Система устойчива

Process finished with exit code 0

```

Рис. 7: Анализ устойчивости для 0

- Теперь рассмотрим аналогичную систему, но изменим в ней начальный угол на π . Тогда:

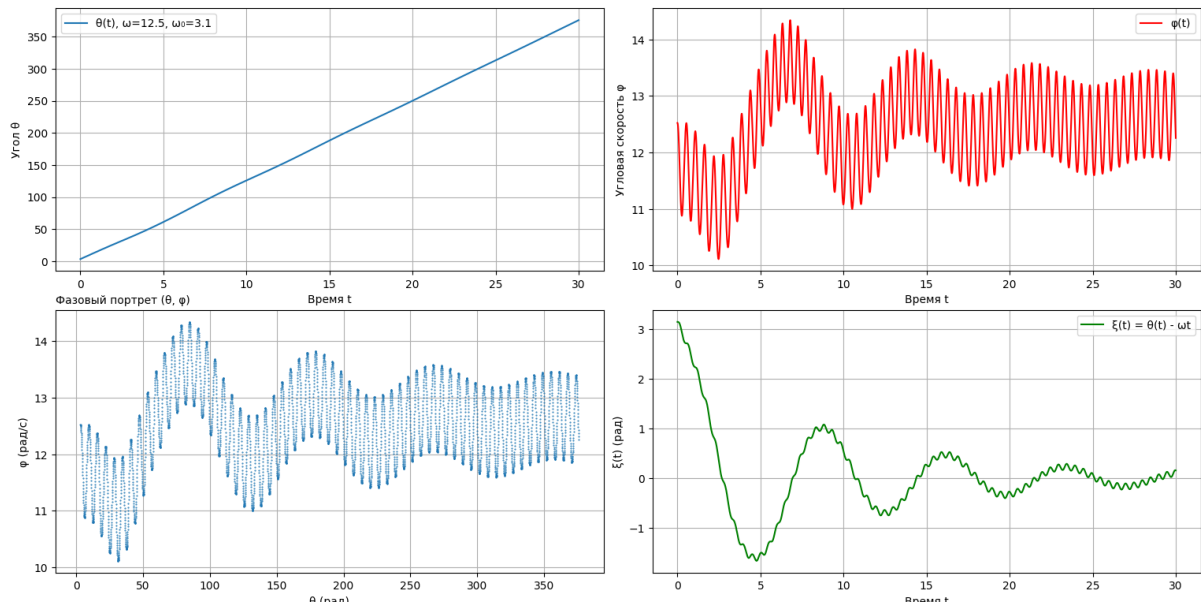


Рис. 8: Графики зависимостей для π

Из теории вытекает, что при таких НУ система должна быть неустойчивой, что также следует из нашего кода:

```

175 if __name__ == "__main__":
176     params = {
177         'omega0': np.sqrt(9.8), # корень из g / L
178         'omega': 4 * np.sqrt(9.8), # Частота вращ
179         'gamma': 0.1,
180         'a': 0.01,
181         'L': 1.0
182     }
183     y0 = [0, params['omega']]
184     simulate_and_analyze(params, y0, t_span=(0, 30), save_gif=False)

```

simulate_and_analyze()

```

un: script_2 x script x
C:\Users\User\PycharmProjects\stepik\venv\Scripts\python.exe C:\Users\User\Desktop\script.py
Собственные значения: 0.9092+0.2791j, 0.9092-0.2791j
Модули: 0.9511, 0.9511
Система устойчива

Process finished with exit code 0

```

Рис. 9: Анализ устойчивости для π

3 Приложение

3.1 Полный код для резонанса 0:1 и n:1

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.integrate import solve_ivp
4
5 def equation_of_rotation(t, y, omega0, omega, gamma, a, L, n):
6     theta, phi = y
7     dydt = [
8         phi,
9         -2*gamma*(phi - n*omega) +
10         (omega0**2 - (a/L)*omega**2*np.cos(omega*t))*np.sin(theta)
11     ]
12     return dydt
13
14 def sustainability_matrix(params, T, sol_main):
15     omega = params['omega']
16     omega0 = params['omega0']
17     gamma = params['gamma']
18     a = params['a']
19     L = params['L']
20
21     def variational_eq(t, y):
22         xi, dxi = y # xi = theta - theta0, dxi = phi - phi0
23         theta0 = sol_main.sol(t)[0]
24         coeff = (omega0**2 -
25                 100↔ (a/L)*omega**2*np.cos(omega*t))*np.cos(theta0)
26         return [dxi, -2*gamma*dxi + coeff*xi]
27
28     sol1 = solve_ivp(variational_eq, [0,T], [1,0],
29                      method='RK45', rtol=1e-8)
30     sol2 = solve_ivp(variational_eq, [0,T], [0,1],
31                      method='RK45', rtol=1e-8)
32
33     M = np.array([
34         [sol1.y[0,-1], sol2.y[0,-1]],
35         [sol1.y[1,-1], sol2.y[1,-1]]
36     ])
37     return M
38
39 def simulate_and_analyze(params, y0, t_span=(0,50)):
40     sol_main = solve_ivp(equation_of_rotation, t_span, y0,
41                          args=(params['omega0'], params['omega'],
42                                params['gamma'], params['a'], params['L'],
43                                100↔ params['n']),
44                          method='RK45', dense_output=True, rtol=1e-6)
45
46     t = np.linspace(t_span[0], t_span[1], 5000)
47     theta, phi = sol_main.sol(t)
48
49     plt.figure(figsize=(12,8))
50     plt.subplot(2,2,1)
51     plt.plot(t, theta, label=f' $\theta(t)$ ,  $\omega=\{params["omega"]:.1f\}$ ')
52     plt.xlabel('Время t'); plt.ylabel('Угол  $\theta$ ')
53     plt.grid(True); plt.legend()
54
55     plt.subplot(2,2,2)
56     plt.plot(t, phi, 'r', label=f' $\phi(t)$ ')
```

```

55 plt.xlabel('Время t'); plt.ylabel('Угловая скорость  $\phi$ ')
56 plt.grid(True); plt.legend()
57
58 plt.subplot(2,2,3)
59 plt.plot(theta, phi, '.', markersize=1)
60 plt.xlabel('θ (рад)'); plt.ylabel('φ (рад/с)')
61 plt.title('Фазовый портрет (θ, φ)'); plt.grid(True)
62
63 plt.subplot(2,2,4)
64 plt.plot(t, theta-params['n']*params['omega']*t, 'g', label='ξ(t)')
65 plt.xlabel('Время t'); plt.ylabel('ξ(t) (рад)')
66 plt.grid(True); plt.legend()
67
68 plt.tight_layout()
69 plt.show()
70
71 T = 2*np.pi/params['omega']
72 M = sustainability_matrix(params, T, sol_main)
73 eigvals = np.linalg.eigvals(M)
74 print(f"Собственные значения: {eigvals[0]:.4f}, {eigvals[1]:.4f}")
75 print(f"Модули: {np.abs(eigvals[0]):.4f}, {np.abs(eigvals[1]):.4f}")
76 print("Система устойчива" if all(np.abs(eigvals) < 1)
77       else "Система неустойчива")
78
79 if __name__ == "__main__":
80     params = {
81         'omega0': 1.0,
82         'omega': 10.0,
83         'gamma': 0.1,
84         'a': 0.01,
85         'L': 1.0,
86         'n': 0
87     }
88     y0 = [0.01, params['omega']*params['n']]
89     simulate_and_analyze(params, y0, (0,60))

```

3.2 Полный код для резонанса 1:1

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.integrate import solve_ivp
4
5 def equation_of_rotation(t, y, omega0, omega, gamma, a, L):
6     theta, phi = y
7     dydt = [
8         phi,
9         -2*gamma*(phi - omega) +
10         (omega0**2 - (a/L)*omega**2*np.cos(omega*t))*np.sin(theta)
11     ]
12     return dydt
13
14 def sustainability_matrix(params, T, sol_main):
15     omega = params['omega']
16     omega0 = params['omega0']
17     gamma = params['gamma']
18     a = params['a']
19     L = params['L']
20
21     def variational_eq(t, y):

```

```

22     xi, dxi = y
23     theta0 = sol_main.sol(t)[0]
24     coeff = (omega0**2 -
25             100→ (a/L)*omega**2*np.cos(omega*t))*np.cos(theta0)
26     return [dxi, -2*gamma*dxi + coeff*xi]
27
28 sol1 = solve_ivp(variational_eq, [0,T], [1,0],
29                 method='RK45', rtol=1e-8)
30 sol2 = solve_ivp(variational_eq, [0,T], [0,1],
31                 method='RK45', rtol=1e-8)
32
33 return np.array([
34     [sol1.y[0,-1], sol2.y[0,-1]],
35     [sol1.y[1,-1], sol2.y[1,-1]]
36 ])
37
38 def simulate_and_analyze(params, y0, t_span=(0,50)):
39     sol_main = solve_ivp(equation_of_rotation, t_span, y0,
40                         args=(params['omega0'], params['omega'],
41                             params['gamma'], params['a'], params['L']),
42                         method='RK45', dense_output=True, rtol=1e-6)
43
44     t = np.linspace(t_span[0], t_span[1], 5000)
45     theta, phi = sol_main.sol(t)
46
47     plt.figure(figsize=(12,8))
48     plt.subplot(2,2,1)
49     plt.plot(t, theta, label=f' $\theta(t)$ ,  $\omega=\{params["omega"]:.1f\}$ ')
50     plt.xlabel('Время t'); plt.ylabel('Угол  $\theta$ ')
51     plt.grid(True); plt.legend()
52
53     plt.subplot(2,2,2)
54     plt.plot(t, phi, 'r', label=f' $\phi(t)$ ')
55     plt.xlabel('Время t'); plt.ylabel('Угловая скорость  $\phi$ ')
56     plt.grid(True); plt.legend()
57
58     plt.subplot(2,2,3)
59     plt.plot(theta, phi, '.', markersize=1)
60     plt.xlabel('θ (рад)'); plt.ylabel('φ (рад/с)')
61     plt.title('Фазовый портрет (θ, φ)'); plt.grid(True)
62
63     plt.subplot(2,2,4)
64     plt.plot(t, theta-params['omega']*t, 'g', label='ξ(t)')
65     plt.xlabel('Время t'); plt.ylabel('ξ(t) (рад)')
66     plt.grid(True); plt.legend()
67
68     plt.tight_layout()
69     plt.show()
70
71     T = 2*np.pi/params['omega']
72     M = sustainability_matrix(params, T, sol_main)
73     eigvals = np.linalg.eigvals(M)
74     print(f"Собственные значения: {eigvals[0]:.4f}, {eigvals[1]:.4f}")
75     print(f"Модули: {np.abs(eigvals[0]):.4f}, {np.abs(eigvals[1]):.4f}")
76     print("Система устойчива" if all(np.abs(eigvals) < 1)
77           else "Система неустойчива")
78
79 if __name__ == "__main__":
80     params = {
81         'omega0': np.sqrt(9.8),

```

```
81         'omega': 4*np.sqrt(9.8),
82         'gamma': 0.1,
83         'a': 0.01,
84         'L': 1.0
85     }
86     simulate_and_analyze(params, [np.pi, params['omega']], (0,30))
```