Министерство науки и высшего образования Российской Федерации
федеральное
государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Имитационное моделирование робототехнических систем

Практическая работа №2

Вариант 1

Студент, группа:
Сергеева Е. С., R4150
Tg:@serkaterina

Преподаватель:
Ракшин Егор Александрович

Санкт-Петербург,
2025

Цель работы: составить дифференциальное уравнение движения маятника, оснащённого пружиной и демпфером, применяя уравнения Лагранжа второго рода, решить уравнение аналитическим способом (если возможно). Выполнить численные решения уравнения тремя различными методами: методом явного Эйлера, методом неявного Эйлера и методом Рунге-Кутты четвёртого порядка. Провести сравнительный анализ полученных результатов численных решений.

Начальные условия:

m, kg	k, N/m, Nm/rad	b, N*s/m, Nm*s/rad	l, m	theta_0, rad
1	8	0.025	0.6	1.539428212

Математическое моделирование

Вводим Лагранжиан как разницу кинетической и потенциальной энергии:

$$\mathcal{L}(\theta,\dot{\theta}) = \mathrm{K}(\theta,\dot{\theta}) - \mathrm{P}(\theta)$$

Выразим кинетическую и потенциальную энергию:

$$\mathrm{K}(\theta,\dot{\theta}) = \frac{1}{2}ml^2\dot{\theta}^2$$

$$\mathrm{P}(\theta) = mgl(1-\cos\theta) + \frac{1}{2}k\dot{\theta}^2$$

Тогда для расчета Лагранжиана используем:

$$\mathcal{L}(\theta,\dot{\theta}) = \frac{1}{2}ml^2\dot{\theta}^2 - mgl(1-\cos\theta) + \frac{1}{2}k\dot{\theta}^2$$

Используем уравнение Эйлера-Лагранжа для выражения уравнения динамики:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}}\right) - \frac{\partial\mathcal{L}}{\partial\theta} = Q$$

$$Q = -b\dot{\theta}$$

Найдем частную производную по угловой скорости

$$\frac{\partial\mathcal{L}}{\partial\dot{\theta}} = ml^2\dot{\theta}$$

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}}\right) = ml^2\ddot{\theta}$$

Найдем частную производную по углу:

$$\frac{\partial\mathcal{L}}{\partial\theta} = -mgl\sin\theta - k\theta$$

Используем подстановку в уравнение Эйлера-Лагранжа:

$$ml^2\ddot{\theta} + mgl\sin\theta + k\theta = -b\dot{\theta}$$

Однородное дифференциальное уравнение динамики:

$$\ddot{\theta} = -\frac{1}{ml^2}\left(mgl\sin\theta + k\theta + b\dot{\theta}\right)$$

Приведем выражение к канонической форме:

Частоту выразим как $\omega_n^2 = \frac{k}{ml^2} + \frac{q}{l}$, тогда коэффициент демпфирования: $2\zeta = \frac{b}{ml^2}$

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0$$

Аналитическое решение

При начальных условиях $\theta(0) = \theta_0, \dot{\theta}(0) = \dot{\theta}_0$

1) Недостаточные демпфирование ($0 \le \zeta < 1$): $\omega_d = \omega_n\sqrt{1-\zeta^2}$, тогда:

$$\theta(t) = e^{-\zeta\omega_n t}\left(\theta_0\cos(\omega_d t) + \frac{\dot{\theta}_0 + \zeta\omega_n\theta_0}{\omega_d}\sin(\omega_d t)\right)$$

2) Критическое демпфирование ($\zeta = 1$)

$$\theta(t) = \left[\theta_0 + \left(\dot{\theta}_0 + \omega_n\theta_0\right)t\right]e^{-\omega_n t}$$

3) Избыточное демпфирование ($\zeta > 1$)

$$\lambda_{1,2} = -\omega_n(\zeta \pm \sqrt{\zeta^2 - 1}$$

$$\theta(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}$$

$$ml^2\ddot{\theta} + mgl\sin\theta + k\theta = -b\dot{\theta}$$

$$ml^2\ddot{\theta} + mgl\sin\theta + k\theta + b\dot{\theta} = 0$$

Для малых углов $\sin\theta = \theta$, тогда

$$ml^2\ddot{\theta} + mgl\theta + k\theta + b\dot{\theta} = 0$$

$$ml^2\ddot{\theta} + b\dot{\theta} + (mgl + k)\theta = 0$$

Аналитическое решение ОДУ

$$0{,}6^2\ddot{\theta} + 0.025\dot{\theta} + (9{,}8 \cdot 0.6 + 8)\theta = 0$$

$$0.36\theta^2 + 0.025\theta + 13.88 = 0$$

$\lambda_1 \approx -0.0347 - 6.209i, \lambda_2 \approx -0.0347 + 6.209i$

Общее решение:

$$\theta(t) = C_1 e^{-0.0347t} + C_2\sin(6.209t)$$

Вычисление констант произведем внутри кода:

```
def analytical_solution(t, x0):
    omega0 = np.sqrt(k_total / I)
    zeta = b / (2 * np.sqrt(I * k_total))
    omega_d = omega0 * np.sqrt(1 - zeta**2)

    theta0 = x0[0]
    theta_dot0 = x0[1]

    C1 = theta0
    C2 = (theta_dot0 + zeta * omega0 * theta0) / omega_d

    return np.exp(-zeta * omega0 * t) * (C1 * np.cos(omega_d * t) + C2 *
np.sin(omega_d * t))
```

Таким образом,
$C_1 = 1.539428$
$C_2 = 0.008609$
$\theta(t) = e^{-0.0347t}(1.539428 \cdot \cos(6.208149t) + 0.008614 \cdot \sin(6.209t))$

Численные методы

Используем начальные условия выполним решение уравнение с помощью метода Явного Эйлера:

```
def forward_euler(fun, x0, Tf, h):
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0
    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k] + h * fun(x_hist[:, k])
    return x_hist, t
```

Метода Невного Эйлера:

```
def backward_euler(fun, x0, Tf, h, tol=1e-8, max_iter=100):
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k] + h * fun(x_hist[:, k])

        for i in range(max_iter):
            x_next = x_hist[:, k] + h * fun(x_hist[:, k + 1])
            error = np.linalg.norm(x_next - x_hist[:, k + 1])
            x_hist[:, k + 1] = x_next
            if error < tol:
                break
    return x_hist, t
```

Метода Рунге-Кутта 4

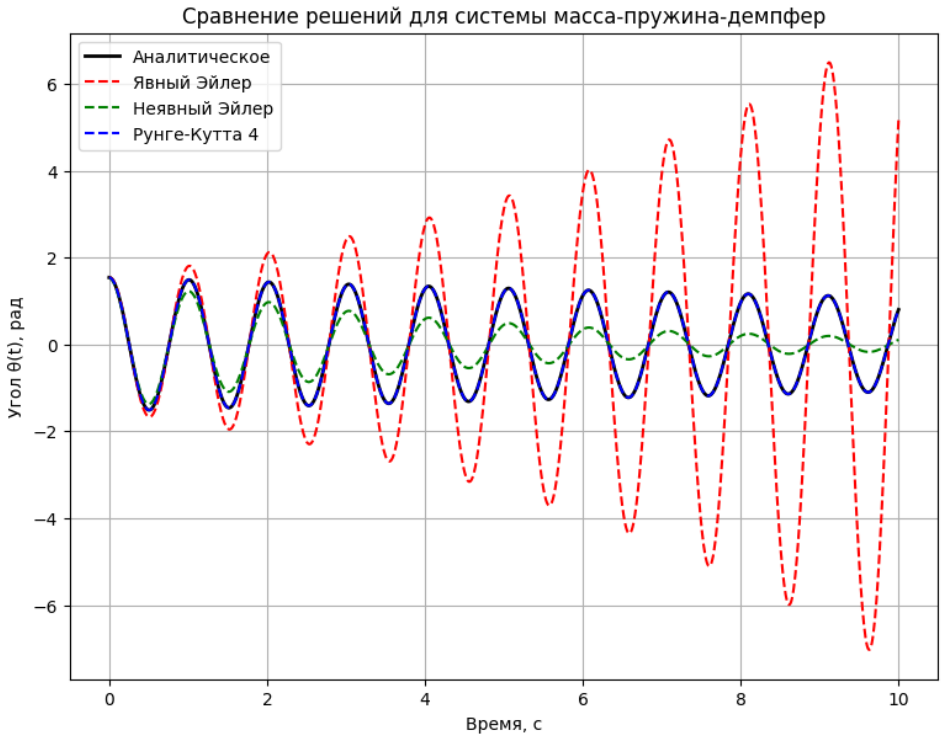
```
def runge_kutta4(fun, x0, Tf, h):
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        k1 = fun(x_hist[:, k])
        k2 = fun(x_hist[:, k] + 0.5 * h * k1)
        k3 = fun(x_hist[:, k] + 0.5 * h * k2)
        k4 = fun(x_hist[:, k] + h * k3)

        x_hist[:, k + 1] = x_hist[:, k] + (h / 6.0) * (k1 + 2*k2 + 2*k3 +
k4)
    return x_hist,
```

В качестве параметров симуляции используем время симуляции `Tf = 10.0`
И шаг интегрирования `h = 0.01`

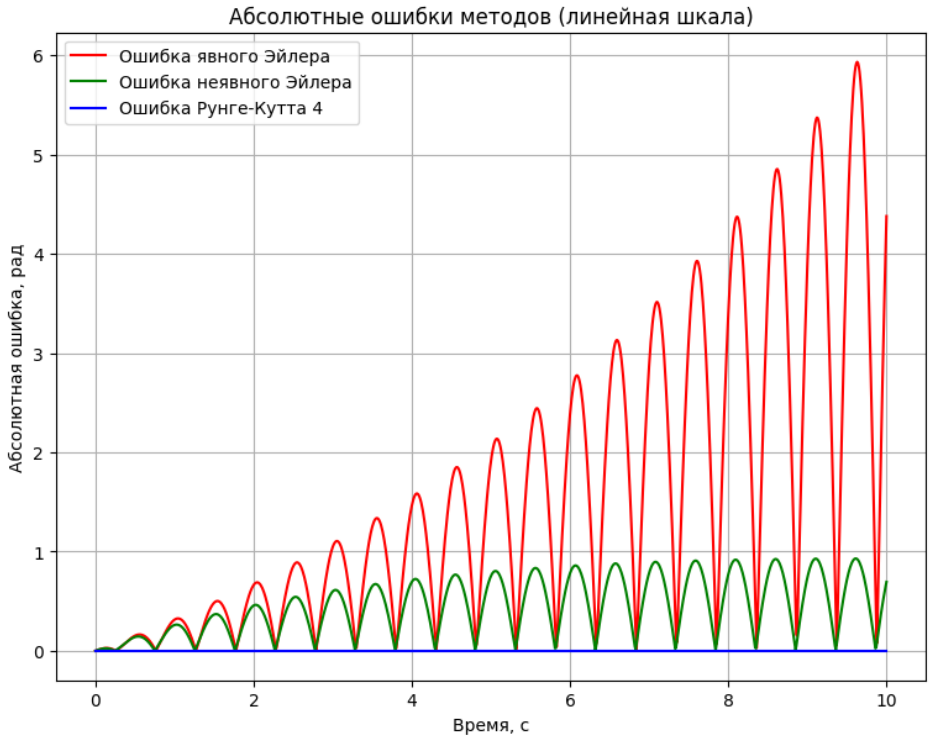
Построим график по заданным параметрам (полный код программы представлен в Приложении 1), получим:



Точки аналитического решения и решения методам Рунге-Кутта 4 находятся в значительной близости, что выделяет метод Рунге-Кутта 4 как наиболее точный и стабильный.

Характеризуя Неявный метод Эйлера, выделим, что полученные при решении значения близки к значениям полученным аналитическим путем, график с течением времени характеризуется затуханием и увеличение погрешности. В это время Явный Эйлер демонстрирует более значительные отклонения и со временем только усиливает колебания, что также негативно отражается на точности модели.

Подтвердим полученные выводы путем анализа графика ошибок:



Решения, найденные с помощью методы Рунге-Кутта 4 близки к значениям, полученным с помощью аналитического решения, и с течением времени сохраняют значение ошибки близкое к 0. Значения же, полученные с помощью методов Явного и Неявного Эйлера, характеризируется нестабильностью и увеличиваются в течение времени, особенно сильно это демонстрирует метод Явного Эйлера.

Вывод: Выполненная работа позволила получить уравнение движения механической системы, включающей маятник, пружину и демпфер. Модель углового движения маятника была построена на основе уравнений Лагранжа второго рода. Проведен анализ уравнения методами аналитического и численного решения, среди последних были использованы явный и неявный методы Эйлера, а также метод Рунге–Кутты 4-го порядка.

Сравнительный анализ показал превосходящую точность метода Рунге–Кутты над методами Эйлера. Полученное численное решение хорошо согласуется с аналитическими расчетами, что свидетельствует о корректности построенной модели и правильности выбранных алгоритмов решения.

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

m = 1.0
k = 8.0
b = 0.025
l = 0.6
theta_0 = 1.539428212
g = 9.8


I = m * l**2
k_total = m * g * l + k

print(f"\nДИФФЕРЕНЦИАЛЬНОЕ УРАВНЕНИЕ:")
print(f"{I:.2f}·θ'' + {b:.3f}·θ' + {k_total:.2f}·θ = 0")

def system_dynamics(x):

    theta = x[0]
    theta_dot = x[1]
    theta_ddot = - (b/I) * theta_dot - (k_total/I) * theta
    return np.array([theta_dot, theta_ddot])

def analytical_solution(t, x0):

    omega0 = np.sqrt(k_total / I)
    zeta = b / (2 * np.sqrt(I * k_total))
    omega_d = omega0 * np.sqrt(1 - zeta**2)

    theta0 = x0[0]
    theta_dot0 = x0[1]

    C1 = theta0
    C2 = (theta_dot0 + zeta * omega0 * theta0) / omega_d

    print(f"\nКОНСТАНТЫ ИЗ НАЧАЛЬНЫХ УСЛОВИЙ:")
    print(f"C₁ = θ₀ = {C1:.6f}")
    print(f"C₂ = (θ₀' + ζω₀θ₀)/ω_d = ({theta_dot0} +
{zeta:.6f}·{omega0:.6f}·{theta0:.6f})/{omega_d:.6f} = {C2:.6f}")

    # Аналитическое решение
    return np.exp(-zeta * omega0 * t) * (C1 * np.cos(omega_d * t) + C2 *
np.sin(omega_d * t))

def forward_euler(fun, x0, Tf, h):
    """Явный метод Эйлера"""
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0
```

```
    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k] + h * fun(x_hist[:, k])
    return x_hist, t

def backward_euler(fun, x0, Tf, h, tol=1e-8, max_iter=100):
    """Неявный метод Эйлера"""
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k] + h * fun(x_hist[:, k])

        for i in range(max_iter):
            x_next = x_hist[:, k] + h * fun(x_hist[:, k + 1])
            error = np.linalg.norm(x_next - x_hist[:, k + 1])
            x_hist[:, k + 1] = x_next
            if error < tol:
                break
    return x_hist, t

def runge_kutta4(fun, x0, Tf, h):
    """Метод Рунге-Кутта 4-го порядка"""
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        k1 = fun(x_hist[:, k])
        k2 = fun(x_hist[:, k] + 0.5 * h * k1)
        k3 = fun(x_hist[:, k] + 0.5 * h * k2)
        k4 = fun(x_hist[:, k] + h * k3)

        x_hist[:, k + 1] = x_hist[:, k] + (h / 6.0) * (k1 + 2*k2 + 2*k3 +
k4)
    return x_hist, t


x0 = np.array([theta_0, 0.0])
Tf = 10.0
h = 0.01

print("\n" + "=" * 70)
print("ПАРАМЕТРЫ СИМУЛЯЦИИ")
print("=" * 70)
print(f"Начальные условия: θ(0) = {x0[0]:.6f} рад, θ'(0) = {x0[1]:.1f}
рад/с")
print(f"Шаг времени: h = {h}")
print(f"Время симуляции: Tf = {Tf} с")

# Численные решения
```

```
print("\n" + "=" * 70)
print("ВЫЧИСЛЕНИЕ ЧИСЛЕННЫХ РЕШЕНИЙ")
print("=" * 70)

print("Явный метод Эйлера...")
x_fe, t_fe = forward_euler(system_dynamics, x0, Tf, h)
print("Неявный метод Эйлера...")
x_be, t_be = backward_euler(system_dynamics, x0, Tf, h)
print("Метод Рунге-Кутта 4-го порядка...")
x_rk4, t_rk4 = runge_kutta4(system_dynamics, x0, Tf, h)

# Аналитическое решение
print("\n" + "=" * 70)
print("ВЫЧИСЛЕНИЕ АНАЛИТИЧЕСКОГО РЕШЕНИЯ")
print("=" * 70)
x_analytical = analytical_solution(t_fe, x0)

# Вычисление ошибок
error_fe = np.abs(x_fe[0, :] - x_analytical)
error_be = np.abs(x_be[0, :] - x_analytical)
error_rk4 = np.abs(x_rk4[0, :] - x_analytical)

# Построение графиков
plt.figure(figsize=(20, 15))

# График 1: Сравнение решений
plt.subplot(2, 2, 1)
plt.plot(t_fe, x_analytical, 'k-', linewidth=2, label='Аналитическое')
plt.plot(t_fe, x_fe[0, :], 'r--', label='Явный Эйлер')
plt.plot(t_be, x_be[0, :], 'g--', label='Неявный Эйлер')
plt.plot(t_rk4, x_rk4[0, :], 'b--', label='Рунге-Кутта 4')
plt.xlabel('Время, с')
plt.ylabel('Угол θ(t), рад')
plt.legend()
plt.title('Сравнение решений для системы масса-пружина-демпфер')
plt.grid(True)

# График 2: Ошибки (линейная шкала)
plt.subplot(2, 2, 3)
plt.plot(t_fe, error_fe, 'r-', label='Ошибка явного Эйлера')
plt.plot(t_be, error_be, 'g-', label='Ошибка неявного Эйлера')
plt.plot(t_rk4, error_rk4, 'b-', label='Ошибка Рунге-Кутта 4')
plt.xlabel('Время, с')
plt.ylabel('Абсолютная ошибка, рад')
plt.legend()
plt.title('Абсолютные ошибки методов (линейная шкала)')
plt.grid(True)
# Вывод результатов
print("\n" + "=" * 70)
print("СРАВНЕНИЕ ЧИСЛЕННЫХ МЕТОДОВ")
print("=" * 70)
```