МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)
Факультет системы управления и робототехники

ОТЧЕТ
по дисциплине
«Задачи и подходы современной робототехники»

по теме:
Лабораторная работа 2 - Аналитическое решение ОДУ

Студент:
Группа № R4133c *Шумкарбек кызы Нурзада*

Предподаватель:
инженер, ассистент *Ракшин Е. А.*

Санкт-Петербург 2025

СОДЕРЖАНИЕ

Основная часть

2 - вариант

Даны коэффициенты:

m, kg=0.6,

k, N/m, Nm/rad = 6.8,

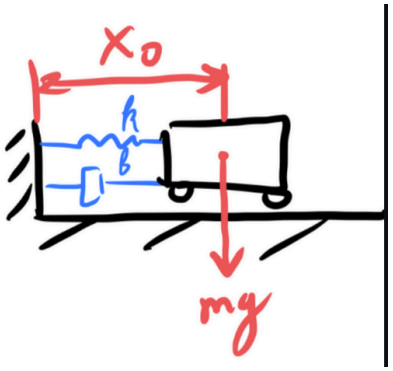
b, N*s/m, Nm*s/rad = 0.045,

l, m = 0.36

theta_0, rad = 0.754005655

x_0, m = 0.83

Имеет блок схему



Алгоритм решения ОДУ аналитическим методом

Уравнение движения масса-пружина-демпфер

$m \cdot \ddot{x} + mg + k \cdot x = -b \cdot \dot{x}$

$m \cdot \ddot{x} + b \cdot \dot{x} + k \cdot x = -mg$

$0.6 \cdot \ddot{x} + 0.045 \cdot \dot{x} + 6.8 \cdot x = -0.6 \cdot 9.81$

$0.6 \cdot \ddot{x} + 0.045 \cdot \dot{x} + 6.8 \cdot x = -5.886$

Однородное уравнение:

$0.6 \cdot \ddot{x} + 0.045 \cdot \dot{x} + 6.8 \cdot x = 0$

Характеристическое уравнение:

$0.6s^2 + 0.045s + 6.8 = 0$

$D = (0.045)^2 - 4 \cdot 0.6 \cdot 6.8 = 0.002025 - 16.32 = -16.317975$

Поскольку D<0, корни уравнения являются комплексными сопряженными числами, решение однородного уравнения будет иметь вид затухающих колебаний, поскольку коэффициент a будет отрицательным. Общее решение однородного уравнения будет выражаться через комплексные числа (в виде комбинации синусов и косинусов для действительных функций:

$s_{1,2} = a \pm ib$

где a = -b\2a, b = √-D\2a

a = -0.045\2.06 = -0.035

b = 4.03\2*0.6= 3.366

$s_{1,2}$ = -0.0375 ± 3.366i

Общее решение однородного уравнения:

x□(t) = e^(at)·[C_1·cos(bt) + C_2·sin(bt)]

x□(t) = e^(-0.0375t)·[C_1·cos(3.366t) + C_2·sin(3.366t)]

x(t) = e^(-0.0375t)·[C_1·cos(3.366t) + C_2·sin(3.366t)] - 0.8656

Частное решение:

Правая часть постоянная ⇒ ищем x□ = A

0.6·0 + 0.045·0 + 6.8·A = -5.886

6.8A = -5.886

A = -0.8656

Общее решение:

x(t) = e^(-0.0375·t)·[C_1·cos(3.366t) + C_2·sin(3.366t)] - 0.8656

Начальные условия: x(0) = 0.83, ẋ(0) = 0

x(0) = e^0·[C_1·cos0 + C_2·sin0] - 0.8656 = 0.83

C_1 - 0.8656 = 0.83

C_1 = 1.6956

ẋ(t) = -0.0375·e^(-0.0375t)·[1.6956·cos(3.366t) + C_2·sin(3.366t)]

+ e^(-0.0375t)·[-1.6956·3.366·sin(3.366t) + C_2·3.366·cos(3.366t)]

ẋ(0) = -0.0375·1.6956 + 3.366·C_2 = 0

-0.063585 + 3.366·C_2 = 0

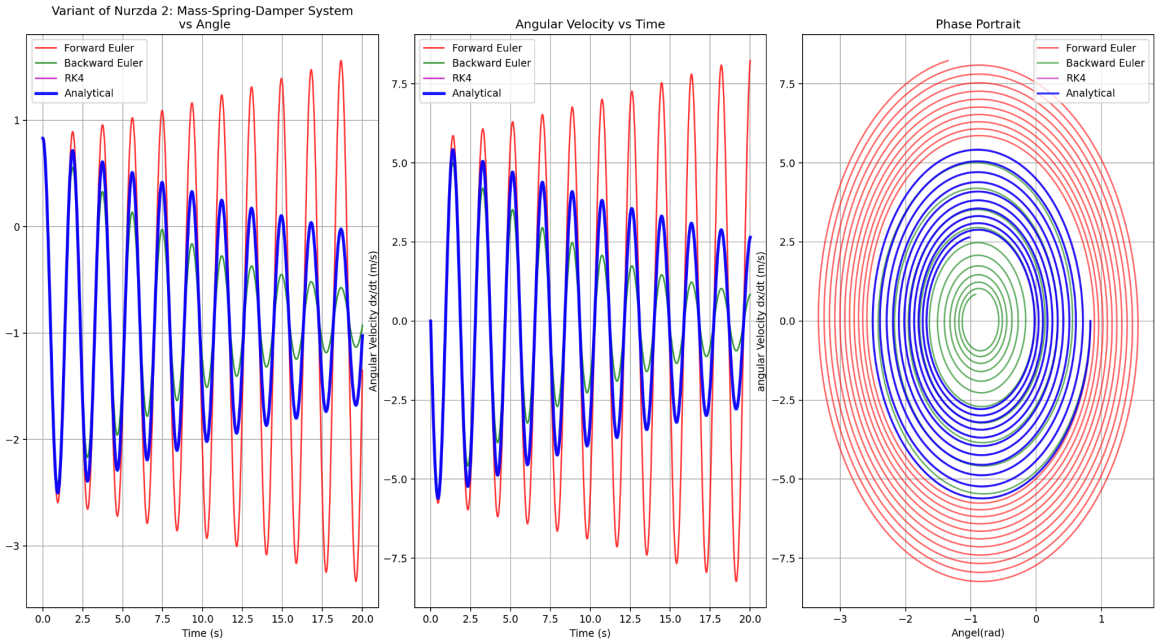
C_2 = 0.01889

Аналитическое решение:

x(t) = e^(-0.0375·t)·[1.6956·cos(3.366t) + 0.01889·sin(3.366t)] - 0.8656

Алгоритм решения ОДУ численным методом

Используя интеграторы в файле "Integrators.ipynb" явным/неявным методом Эйлера метод Рунга-Кутты, выведем графики:



Ниже приведен код реализации явного/неявного методов Эйлера и метода Рунга-Кутты с уравненем масса-пружина-демпфера, с аналитическим решением:

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

m = 0.6
b = 0.045
k = 6.8
g = 9.81
l = 0.36
theta_0 = 0.754005655
x_0 = 0.83

def mass_spring_damper_system(x):

    x_pos = x[0]    # x
    x_vel = x[1]    # dx/dt

    # x'' = (-m·g - b·x' - k·x) / m
    x_acc = (-m*g - b*x_vel - k*x_pos) / m

    return np.array([x_vel, x_acc])

def analytic_solution(t):


    delta = 0.0375
    omega = 3.366
    C1 = 1.6956
    C2 = 0.01889
    x_steady = -0.8656
    return np.exp(-delta * t) * (C1 * np.cos(omega * t) + C2 * np.sin(omega * t)) + x_steady
```

```python
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

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]
        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
```

```python
            if error < tol:
                break
    return x_hist, t

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, t

x0 = np.array([x_0, 0.0])  # Начальные условия: x(0)=0.83, x'(0)=0.0
Tf = 20.0
h = 0.01


x_fe, t_fe = forward_euler(mass_spring_damper_system, x0, Tf, h)
x_be, t_be = backward_euler(mass_spring_damper_system, x0, Tf, h)
x_rk4, t_rk4 = runge_kutta4(mass_spring_damper_system, x0, Tf, h)

plt.figure(figsize=(25, 8))

plt.subplot(1, 3, 1)
plt.plot(t_fe, x_fe[0, :], 'r-', label='Forward Euler', linewidth=1.5, alpha=0.8)
plt.plot(t_be, x_be[0, :], 'g-', label='Backward Euler', linewidth=1.5, alpha=0.8)
plt.plot(t_rk4, x_rk4[0, :], 'm-', label='RK4', linewidth=1.5, alpha=0.8)
plt.plot(t_fe, analytic_solution(t_fe), 'b-', label='Analytical', linewidth=3, alpha=0.9)
```

```
plt.xlabel('Time (s)')
plt.ylabel('Angle (rad)')
plt.title('Variant of Nurzda 2: Mass-Spring-Damper System\n vs Angle')
plt.legend()
plt.grid(True)

plt.subplot(1, 3, 2)
plt.plot(t_fe, x_fe[1, :], 'r-', label='Forward Euler', linewidth=1.5, alpha=0.8)
plt.plot(t_be, x_be[1, :], 'g-', label='Backward Euler', linewidth=1.5, alpha=0.8)
plt.plot(t_rk4, x_rk4[1, :], 'm-', label='RK4', linewidth=1.5, alpha=0.8)

plt.plot(t_fe, analytic_derivative(t_fe), 'b-', label='Analytical', linewidth=3, alpha=0.9)
plt.xlabel('Time (s)')
plt.ylabel('Angular Velocity dx/dt (m/s)')
plt.title('Angular Velocity vs Time')
plt.legend()
plt.grid(True)

plt.subplot(1, 3, 3)
plt.plot(x_fe[0, :], x_fe[1, :], 'r-', label='Forward Euler', alpha=0.6)
plt.plot(x_be[0, :], x_be[1, :], 'g-', label='Backward Euler', alpha=0.6)
plt.plot(x_rk4[0, :], x_rk4[1, :], 'm-', label='RK4', alpha=0.6)

plt.plot(analytic_solution(t_fe), analytic_derivative(t_fe), 'b-',
         label='Analytical', linewidth=2, alpha=0.8)
plt.xlabel('Angel(rad)')
plt.ylabel('angular Velocity dx/dt (m/s)')
plt.title('Phase Portrait')
plt.legend()
plt.grid(True)
```

```
plt.tight_layout()
plt.savefig('variant2_results.png', dpi=300, bbox_inches='tight')
plt.show()
```

Можно сделать вывод, что аналитическое, методом Рунга-Кутты 4-го порядка, неявный метод Эйлера дают наиболее точный результат.

Заключение

В ходе лабораторной работы проведено сравнение численных методов решения дифференциальных уравнений на примере масса-пружины-демпферной системы с аналитическим решением.
На основе анализа графиков установлен следующий порядок методов по степени близости к аналитическому решению для каждого метода:

- Метод Рунге-Кутты 4-го порядка (RK4) - демонстрирует наивысшую точность и минимальное расхождение с аналитическим решением
- Неявный метод Эйлера (Backward Euler) - обеспечивает устойчивость, но обладает значительным численным затуханием
- Явный метод Эйлера (Forward Euler) - имеет наибольшую погрешность и склонность к неустойчивости

Экспериментально подтверждено, что метод Рунге-Кутты 4-го порядка является оптимальным выбором для моделирования динамики масс-пружин-демпферных систем, обеспечивая наилучшее соответствие аналитическому решению при разумных вычислительных затратах.