МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет систем управления и робототехники

Практическая работа №2

по дисциплине

«Имитационное моделирование робототехнических систем»

Студент:

Группа N_0 R4137c Альмахмуд Ахмад

Предподаватели:

Профессор И.И. Борисов

Ассистент Е.А. Ракшин

Санкт-Петербург 2025

1. Постановка задачи

Дана система, представленная на рисунке 1, с исходными параметрами из таблицы 1.

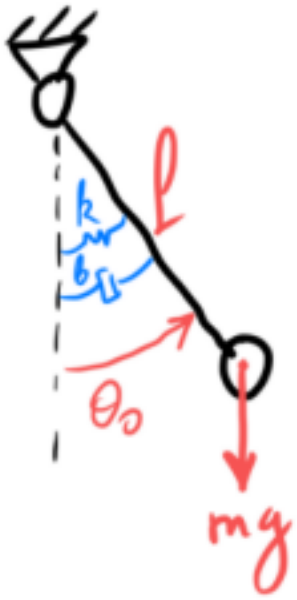


Рисунок 1. Исходная система

Таблица 1. Параметры системы

Вариант	m, kg	k, Nm/rad	b, Nm*s/rad	l, m	θ_0, rad
1	0.8	5.4	0.04	0.5	0.2042098649

2. Цель работы

Целью данной лабораторной работы является проведение аналитического и численного моделирования колебательной системы типа «масса–пружина–демпфер» с использованием различных методов интегрирования (явный и неявный методы Эйлера, метод Рунге–Кутты 4-го порядка), а также сравнение полученных численных результатов с аналитическим решением.

3. Математическое моделирование :

3.1. Составление ОДУ 2 порядка системы

Определение Лагранжиана (L):

Лагранжиан равен разности кинетической и потенциальной энергий:

$$L = T - P$$

Кинетическая энергия (T):

$$T = \frac{1}{2} m l^2 \theta'$$

Теперь найдем потенциальную энергию (P):

Потенциальная энергия силы тяжести

$$Tgrav = -mgl\cos(\theta)$$

Потенциальная энергия пружины

$$T_{spring} = \frac{1}{2}\, k\theta^2$$

Общий вид потенциальной энергии системы:

$$p = -mgl\cos(\theta) + \frac{1}{2}\, k\theta^2$$

Теперь подставим все в Лагранжиан:

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

Далее воспользуемся уравнением Эйлера-Лагранжа:

$$\frac{d}{dt}\left(\frac{dL}{d\theta'}\right) - \frac{dL}{d\theta} = Q$$

Так как наша система является неконсервативной, то в нашем случае Q является силой вязкого трения:

$$Q = -b\,\theta'$$

Теперь получим частные производные:

$$\frac{dL}{d\theta} = -mgl\ \sin(\theta)\ -\ k\theta$$

$$\frac{dL}{d\theta'} = ml^{\,2}\theta\,'$$

$$\frac{d}{dt}\left(\frac{dL}{d\theta'}\right) = ml^2\theta''$$

Подставим частные производные в уравнение Эйлера-Лагранжа:

$$ml^2\theta'' + mgl\ \sin(\theta) + k\,\theta = \ -b\theta'$$

Выразим θй получим итоговый ОДУ второго порядка:

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

1) было отмечено, что аналитическое решение общего вида невозможно, из-за присутствия непрямой функции $\sin(\theta)$. Эта функция делает уравнение нелинейным, и классические аналитические методы решения линейных уравнений здесь неприменимы. Аналитические решения доступны только для линейного вида уравнения:

$$x'' + \frac{b}{m}\,x' + \frac{k + mgl}{m^2 l}x = 0$$

4. Код (Приложение) :

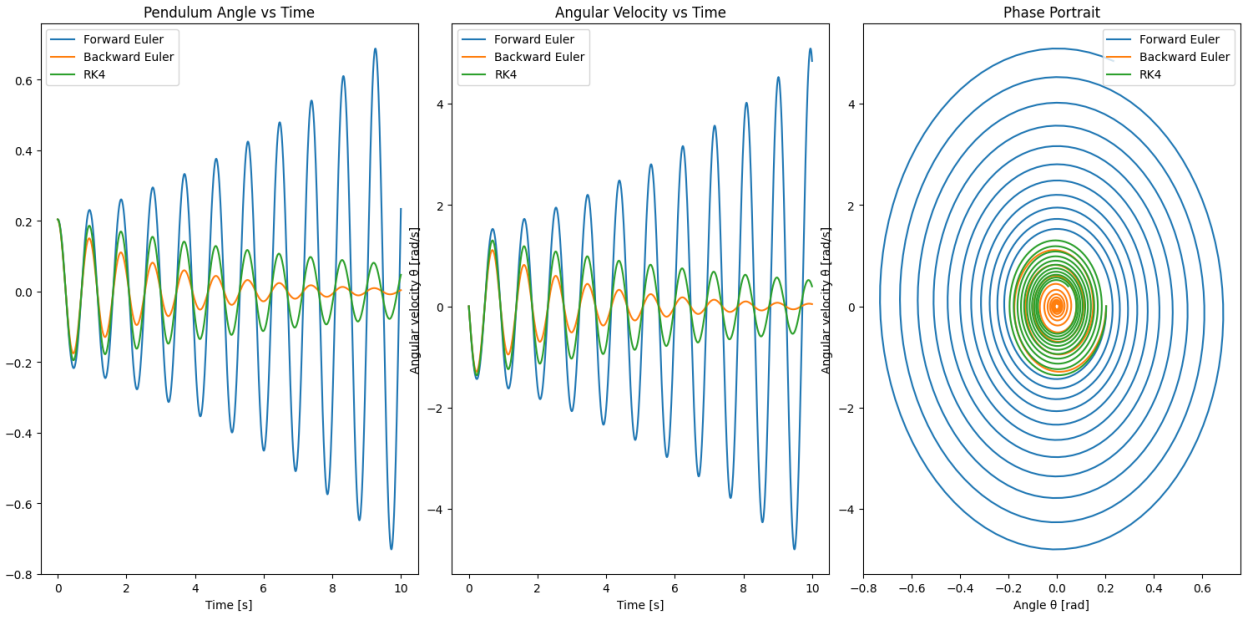
```
import numpy as np
import matplotlib.pyplot as plt

# ==========================
# Parameters
# ==========================
m = 0.8  # kg
k = 5.4  # N·m/rad
```

```
b = 0.04  # N·s/m (damping)
l = 0.5  # m
theta0 = 0.2042098649  # rad
dtheta0 = 0  # rad/s

x0 = np.array([theta0, dtheta0])
Tf = 10.0
h = 0.01



# ==========================
# System dynamics
# ==========================
def pendulum_dynamics(x):
  """
    d²ϑ/dt² = -(1/(m*l^2))*(m*g*l*sin(ϑ) + k*ϑ + b*ϑ̇)
  """
    g = 9.81
    theta = x[0]
    theta_dot = x[1]

    theta_ddot = -1 / (m * l ** 2) * (m * g * l * np.sin(theta) + k *
theta + b * theta_dot)
    return np.array([theta_dot, theta_ddot])



# ==========================
# Numerical methods
# ==========================
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]  # initial guess
        for _ 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):
    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) * (k1 + 2 * k2 + 2 *
k3 + k4)
    return x_hist, t


# ==========================
# Solve using all methods
# ==========================
x_fe, t_fe = forward_euler(pendulum_dynamics, x0, Tf, h)
x_be, t_be = backward_euler(pendulum_dynamics, x0, Tf, h)
x_rk4, t_rk4 = runge_kutta4(pendulum_dynamics, x0, Tf, h)

# ==========================
# Plot results
# ==========================
plt.figure(figsize=(24, 8))
```

```
# θ(t)
plt.subplot(1, 3, 1)
plt.plot(t_fe, x_fe[0, :], label='Forward Euler')
plt.plot(t_be, x_be[0, :], label='Backward Euler')
plt.plot(t_rk4, x_rk4[0, :], label='RK4')
plt.xlabel('Time [s]')
plt.ylabel('Angle θ [rad]')
plt.title('Pendulum Angle vs Time')
plt.legend()

# θ̇(t)
plt.subplot(1, 3, 2)
plt.plot(t_fe, x_fe[1, :], label='Forward Euler')
plt.plot(t_be, x_be[1, :], label='Backward Euler')
plt.plot(t_rk4, x_rk4[1, :], label='RK4')
plt.xlabel('Time [s]')
plt.ylabel('Angular velocity θ̇ [rad/s]')
plt.title('Angular Velocity vs Time')
plt.legend()

# Phase portrait θ vs θ̇
plt.subplot(1, 3, 3)
plt.plot(x_fe[0, :], x_fe[1, :], label='Forward Euler')
plt.plot(x_be[0, :], x_be[1, :], label='Backward Euler')
plt.plot(x_rk4[0, :], x_rk4[1, :], label='RK4')
plt.xlabel('Angle θ [rad]')
plt.ylabel('Angular velocity θ̇ [rad/s]')
plt.title('Phase Portrait')
plt.legend()

plt.tight_layout()
plt.show()
```

5. Графики и подписи



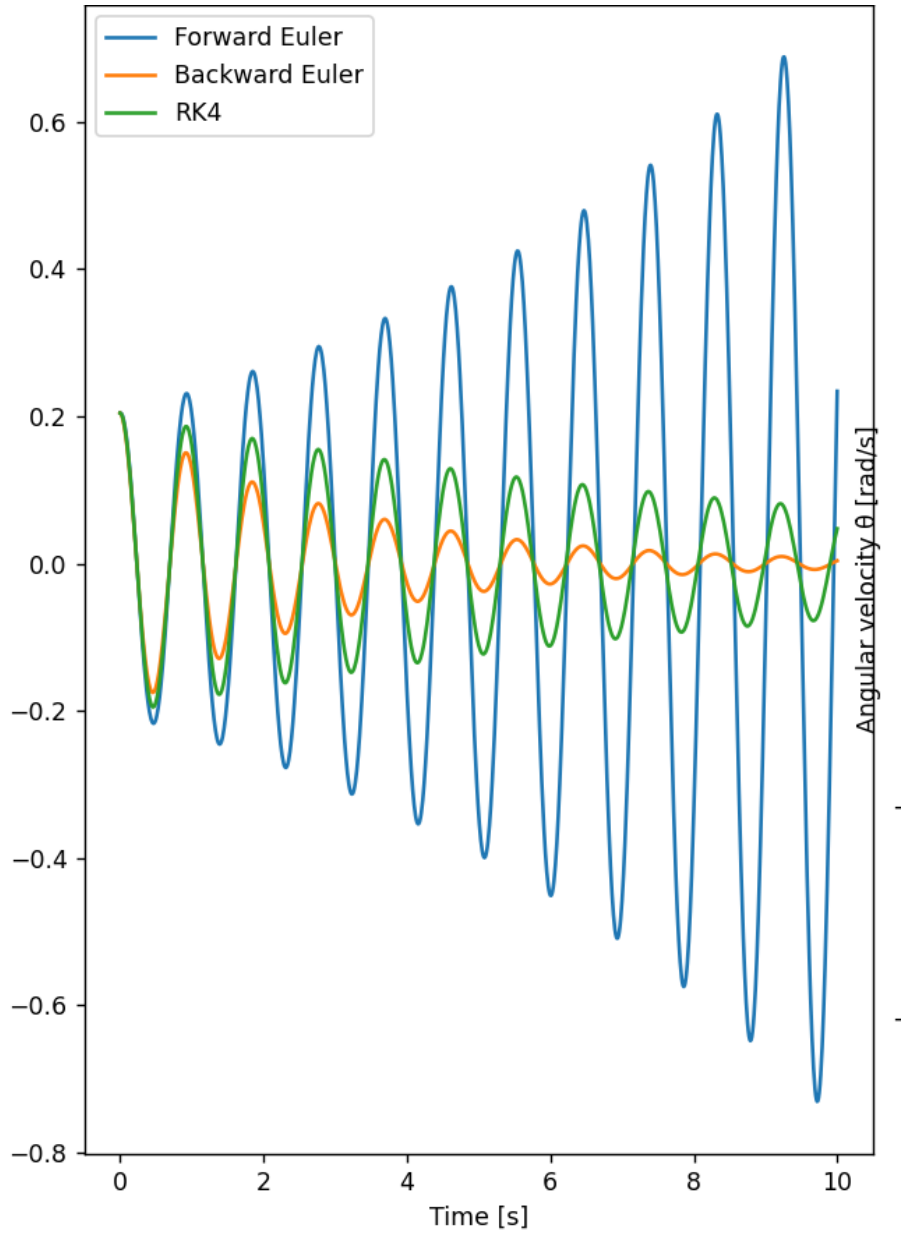


Рисунок 1 График θ для трех интеграторов

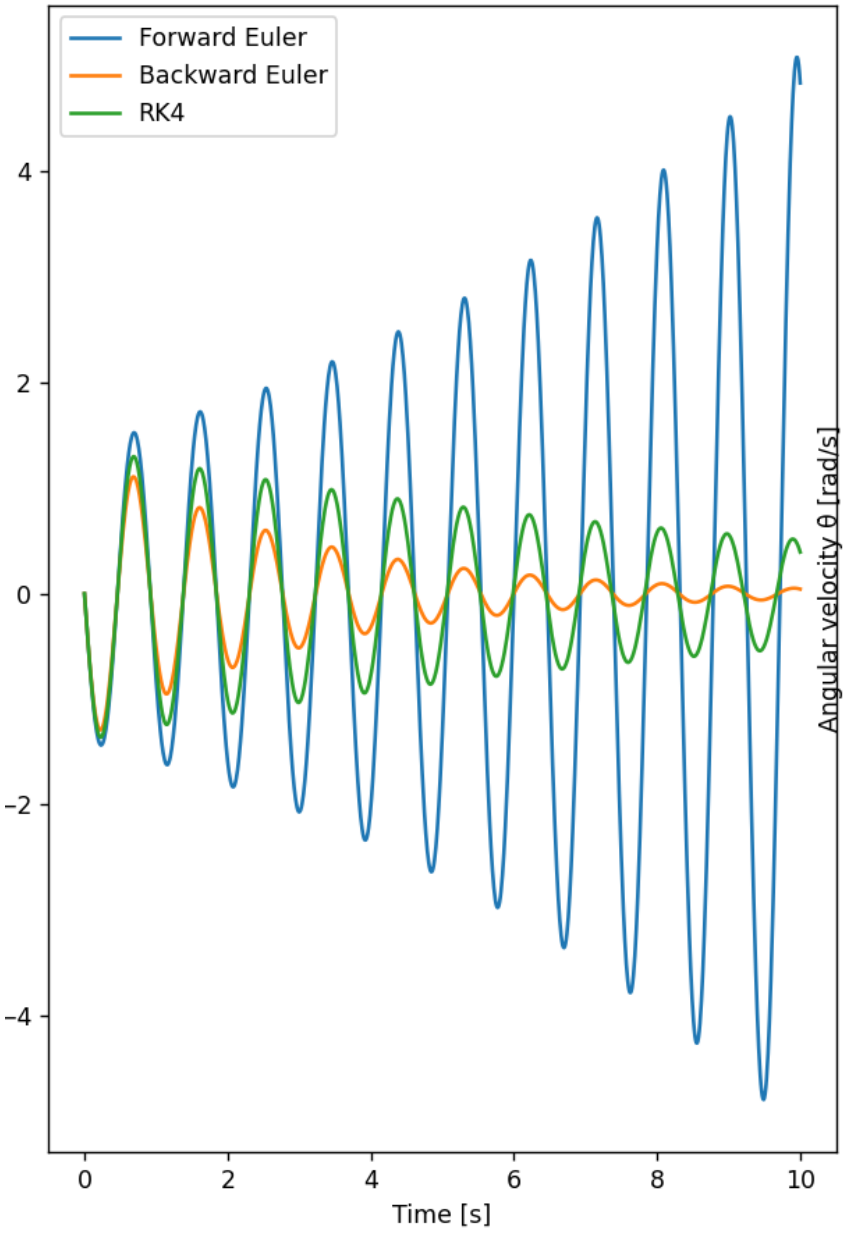


Рисунок 2 График θ для трех интеграторов

Резюме и сравнение численных методов для нелинейного уравнения

С графиков можно заметить следующее:

1. **Метод Рунге–Кутта 4-го порядка (RK4)** дает практически точное приближение решения нелинейного уравнения, точно отражая изменения угла даже при больших отклонениях.
2. **Явный метод Эйлера (Forward Euler)** обеспечивает быстрое приближение, но с меньшей точностью; с течением времени или при больших углах может отклоняться от реального решения, иногда приводя к завышенным или преувеличенным значениям угла.
3. **Неявный метод Эйлера (Backward Euler)** более устойчивый, чем явный Эйлер, но имеет тенденцию сглаживать колебания, давая менее колеблющиеся оценки по сравнению с RK4, особенно в нелинейных системах.

Вывод: Все численные методы дают примерно одинаковое направление движения системы, но **RK4 превосходит их по точности и реалистичности динамики**. Другие методы полезны для быстрого приближения или понимания общего поведения системы, с учетом различий в точности и устойчивости при нелинейных уравнениях.

2) Упрощение уравнения для малых углов:

$$\theta'' + \frac{b}{ml^2}\theta' + \frac{mgl + k}{ml^2}\theta = 0$$

Определение характеристических параметров:

a. Натуральная частота:

$$w_n = \sqrt{\frac{mgl + k}{ml^2}} = 6.83\ \frac{rad}{s}$$

b. Коэффициент затухания:

$$\zeta = \frac{b}{2\ ml^2 w_n} \approx 0.01465$$

Уравнение запишется как:

$$\theta^{\cdot\cdot} + 2\zeta w_n \theta^{\cdot} + w_n^2 \theta = 0$$

3. Определение режима затухания

Так как:

$$\zeta < 1$$

система находится в режиме малóго (недостаточного) затухания — underdamped, и аналитическое решение имеет вид:

$$\theta(t) = e^{-\zeta w_n t}(C_1 \cos(w_d t) + C_2 \sin(w_d t))$$

где:

$$w_d = w_n \sqrt{1 - \zeta^2}$$

Используя начальные условия:

$$\theta(0) = \theta_0 = 0.2042098649$$

$$\theta\cdot(0) = \theta_{\dot{0}}=0$$

получаем:

$$C_1 = \theta_0=0.2042098649$$

$$C_2 = \frac{\theta_{\dot{0}}+\zeta w_n \theta_0}{w_d}=0.002989$$

6.Код (Приложение) :

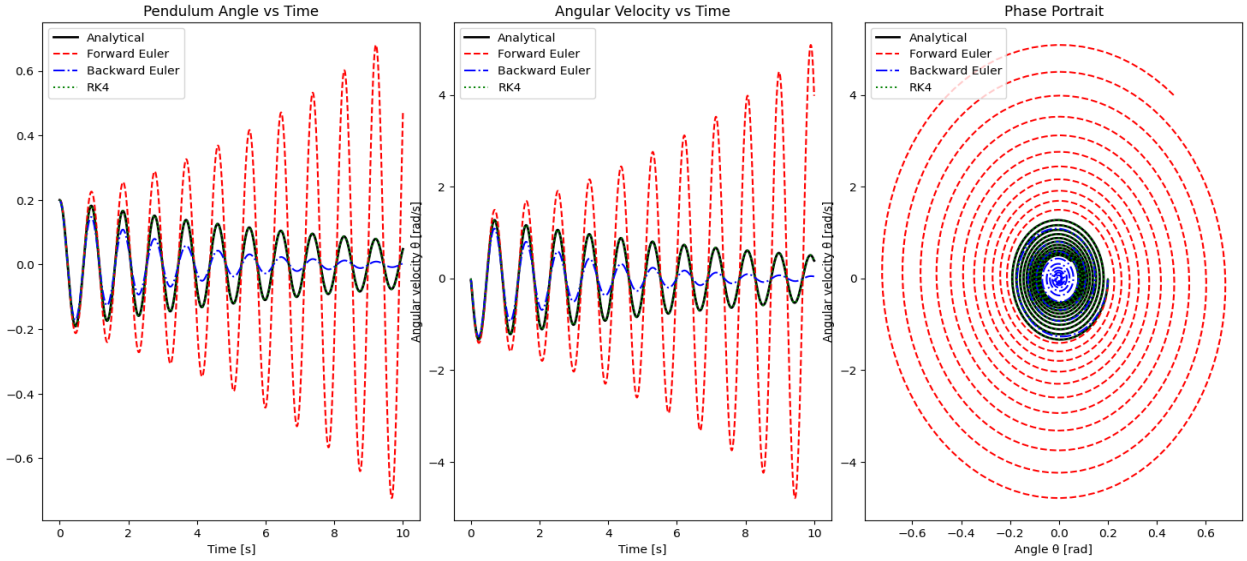
```
import numpy as np
import matplotlib.pyplot as plt

# ==========================
# Parameters
# ==========================
m = 0.8         # kg
k = 5.4         # N·m/rad
b = 0.04        # N·s/m
l = 0.5         # m
theta0 = 0.2    # rad
dtheta0 = 0     # rad/s

x0 = np.array([theta0, dtheta0])
Tf = 10.0
h = 0.01
g = 9.81
```

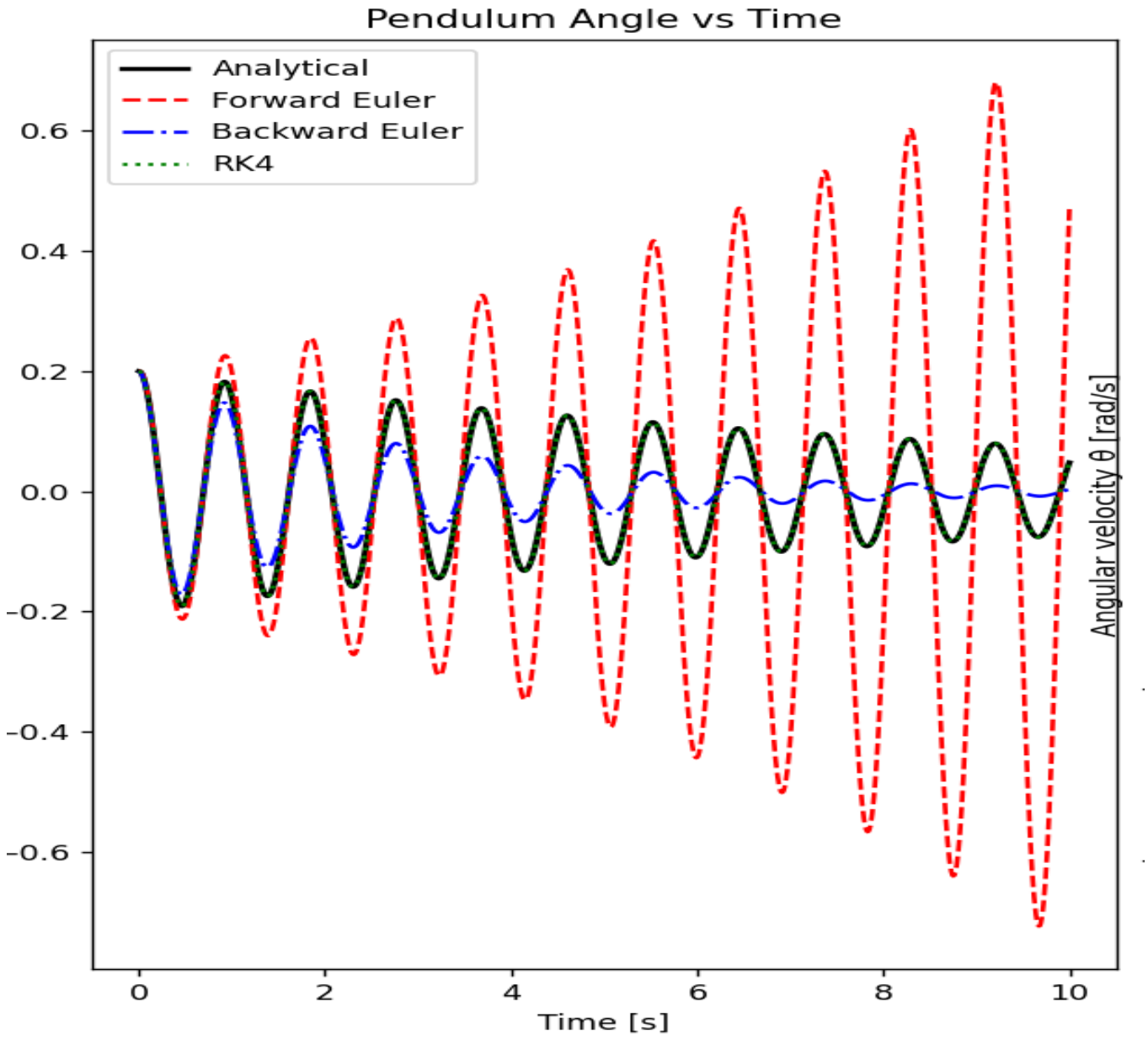
```python
# ==========================
# Linearized pendulum (small angle)
# ==========================
def pendulum_linear(x):
   theta = x[0]    theta_dot = x[1]
    theta_ddot = -1/(m*l**2) * ((m*g*l + k)*theta + b*theta_dot)
    return np.array([theta_dot, theta_ddot])

# ==========================
# Numerical methods
# ==========================
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 _ in range(max_iter):
            x_next = x_hist[:,k] + h*fun(x_hist[:,k+1])
            if np.linalg.norm(x_next - x_hist[:,k+1]) < tol:
                x_hist[:,k+1] = x_next
                break
            x_hist[:,k+1] = x_next
    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)*(k1 + 2*k2 + 2*k3 + k4)
    return x_hist, t

# ==========================
# Analytical solution (linearized)
# ==========================
omega_n = np.sqrt((m*g*l + k)/(m*l**2))
zeta = b/(2*np.sqrt(m*l**2*(m*g*l + k)))
omega_d = omega_n*np.sqrt(1 - zeta**2)

t_vals = np.linspace(0, Tf, 1000)
A = theta0
B = (dtheta0 + zeta*omega_n*theta0)/omega_d

theta_analytical = np.exp(-
zeta*omega_n*t_vals)*(A*np.cos(omega_d*t_vals) +
B*np.sin(omega_d*t_vals))
theta_analytical = np.real(theta_analytical)
theta_dot_analytical = np.gradient(theta_analytical, t_vals)

# ==========================
# Solve numerically
# ==========================
x_fe, t_fe = forward_euler(pendulum_linear, x0, Tf, h)
x_be, t_be = backward_euler(pendulum_linear, x0, Tf, h)
x_rk4, t_rk4 = runge_kutta4(pendulum_linear, x0, Tf, h)

# ==========================
# Plot results
# ==========================
plt.figure(figsize=(24,8))

# θ(t)
plt.subplot(1,3,1)
plt.plot(t_vals, theta_analytical, 'k-', linewidth=2,
label='Analytical')
```

```
plt.plot(t_fe, x_fe[0,:], 'r--', label='Forward Euler')
plt.plot(t_be, x_be[0,:], 'b-.', label='Backward Euler')
plt.plot(t_rk4, x_rk4[0,:], 'g:', label='RK4')
plt.xlabel('Time [s]')
plt.ylabel('Angle θ [rad]')
plt.title('Pendulum Angle vs Time')
plt.legend()

# θ̇(t)
plt.subplot(1,3,2)
plt.plot(t_vals, theta_dot_analytical, 'k-', linewidth=2,
label='Analytical')
plt.plot(t_fe, x_fe[1,:], 'r--', label='Forward Euler')
plt.plot(t_be, x_be[1,:], 'b-.', label='Backward Euler')
plt.plot(t_rk4, x_rk4[1,:], 'g:', label='RK4')
plt.xlabel('Time [s]')
plt.ylabel('Angular velocity θ̇[rad/s]')
plt.title('Angular Velocity vs Time')
plt.legend()

# Phase portrait θ vs θ̇
plt.subplot(1,3,3)
plt.plot(theta_analytical, theta_dot_analytical, 'k-', linewidth=2,
label='Analytical')
plt.plot(x_fe[0,:], x_fe[1,:], 'r--', label='Forward Euler')
plt.plot(x_be[0,:], x_be[1,:], 'b-.', label='Backward Euler')
plt.plot(x_rk4[0,:], x_rk4[1,:], 'g:', label='RK4')
plt.xlabel('Angle θ [rad]')
plt.ylabel('Angular velocity θ̇[rad/s]')
plt.title('Phase Portrait')
plt.legend()

plt.tight_layout()
plt.show()
```

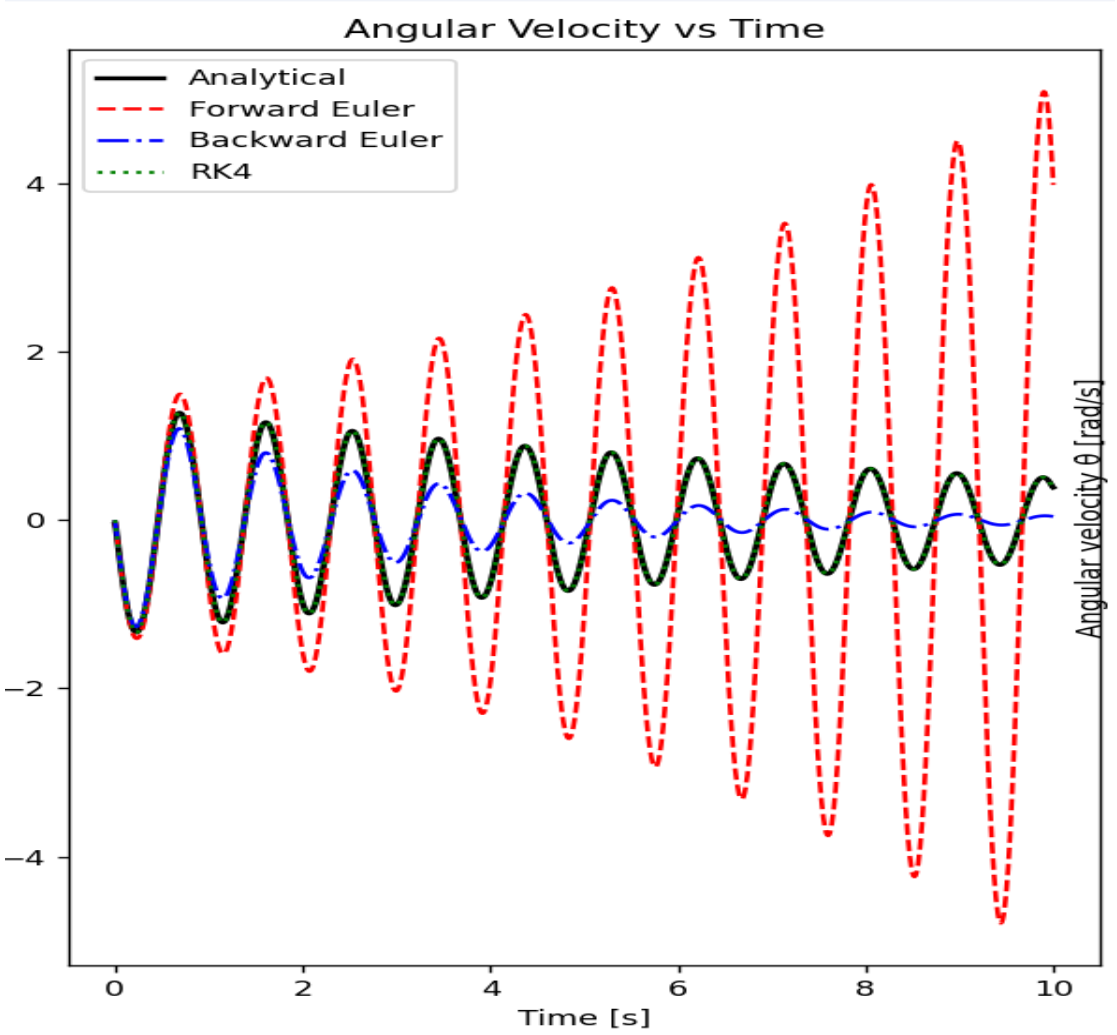
7. Графики и подписи





Сравнение углового отклонения θ(t): аналитическое и численные решения

- Этот график показывает сравнение аналитического решения θ(t) с тремя численными методами. Метод Рунге–Кутты 4-го порядка практически полностью совпадает с аналитическим решением. Явный метод Эйлера постепенно отклоняется из-за накопления ошибки. Неявный метод Эйлера остаётся устойчивым, но демонстрирует избыточное затухание, что связано с его численными свойствами.



Сравнение угловой скорости dθ/dt: аналитическое и численные решения

- Этот график показывает сравнение угловой скорости между аналитическим и численными решениями. Как и в случае θ(t), метод Рунге–Кутты обеспечивает наивысшую точность. Явный Эйлер постепенно увеличивает ошибку, в то время как неявный Эйлер стабилизирует решение, но уменьшает амплитуду скорости из-за дополнительного численного затухания.

8. Выводы

В этом проекте мы изучали решение уравнения движения маятника с демпфированием и пружиной:

$$\theta'' = -\frac{1}{m^2 l}(mgl\sin(\theta)\,\theta' + k\theta + b)$$

Во-первых, относительно общего аналитического решения:
Мы отметили, что присутствие функции $sin(\theta)$ делает уравнение нелинейным, и поэтому невозможно получить аналитическое решение классическими методами линейных уравнений. Любая попытка найти прямое решение приводит к очень сложным уравнениям, которые нельзя упростить с помощью стандартных корней или констант. Поэтому мы использовали численные методы для приближенного решения движения системы.

Во-вторых, относительно решения при малых углах:
Если предположить, что $\theta|theta$θ мало, можно использовать приближение $sin(\theta)$ ≈θ, что превращает уравнение в линейное уравнение второго порядка. В этом случае аналитическое решение возможно, и его можно сравнить с численными методами: явным Эйлером, неявным Эйлером и методом Рунге–Кутта 4-го порядка (RK4). Как видно из графиков, численные решения почти совпадают с аналитическим решением для малых углов, при этом метод RK4 демонстрирует наибольшую точность и стабильность на длительном интервале времени.

Общий вывод:

- Нелинейное уравнение требует использования численных методов, особенно при больших углах.
- Аналитическое решение возможно только при приближении малых углов.

- Метод RK4 является наиболее точным для представления движения системы.
- Явный и неявный методы Эйлера дают хорошее приближение, но с небольшими отличиями в точности и устойчивости.
- Это исследование показывает важность численных методов при анализе нелинейных физических систем с возможностью сравнения с аналитическими решениями в линейных случаях.