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»

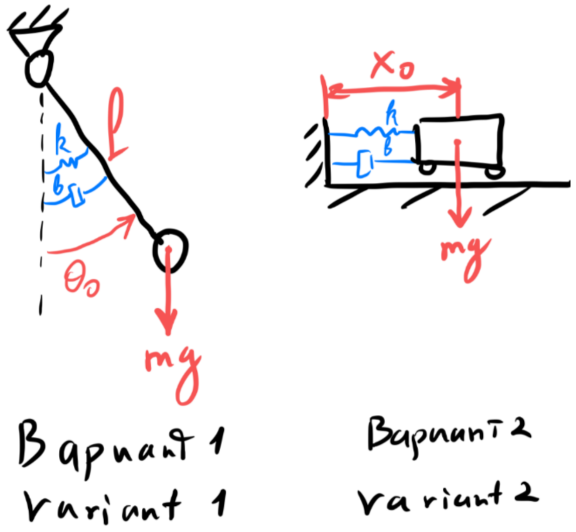
Студен:
Группа R4134c *И. Ковылин*

Преподаватель:
Ассистент *Е.А. Ракшин*

Санкт-Петербург 2025 г.

ЦЕЛЬ РАБОТЫ.

- Решить в численном виде ОДУ для рисунка 2;



- Решить ОДУ системы в аналитическом виде;
- Сравнить результаты методов с аналитическим решением.

АНАЛИТИЧЕСКОЕ РЕШЕНИЕ ОДУ ВТОРОГО ПОРЯДКА.

Для решения системы в аналитическом виде нужно составить лагранжиан системы.

$$\mathcal{L}(x,\dot{x}) = \mathcal{K}(x,\dot{x}) - \mathcal{P}(x) = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2. \quad (1)$$

Уравнение Лагранжа:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = \mathcal{Q}. \quad (2)$$

Где:

$$\frac{\partial \mathcal{L}}{\partial \dot{x}} = m\dot{x}; \quad (3)$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) = m\ddot{x}; \quad (4)$$

$$\frac{\partial \mathcal{L}}{\partial x} = -kx; \quad (5)$$

$$\mathcal{Q} = -b\dot{x}. \quad (6)$$

Подставив данные уравнения в (2), было получено:

$$m\ddot{x} + b\dot{x} + kx = 0. \quad (7)$$

Согласно таблице – m = 0.1, b = 0.025, k = -14.6.

Решение будет производиться в общем виде, составляется характеристический многочлен:

$$\begin{gathered} a \cdot \lambda^2 + b \cdot \lambda + c = 0; \\ \lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \\ \lambda_1 = -0.125 - 12.0824i; \\ \lambda_2 = -0.125 + 12.0824i; \end{gathered} \quad (8)$$

Так как нет действительных корней, то общее решение ОДУ:

$$x = e^{\alpha t}(C_1 \cos(\beta x) + C_2 \sin(\beta x)). \quad (9)$$

Где C_1 и C_2 – постоянные величины, $\alpha = -0.125$, $\beta = 12.0824$.

Нахождение значений коэффициентов C_1 и C_2 при начальных условиях $x_0 = 0.54, \dot{x}_0 = 0$.

$$x(0) = C_1 = x_0;$$

$$\dot{x}(0) = \alpha C_1 + \beta C_2 = 0;$$

$$\begin{cases} C_1 = x_0 \\ C_2 = \dfrac{-\alpha C_1}{\beta} \end{cases}. \tag{10}$$

$$\begin{cases} C_1 = 0.54 \\ C_2 = 0.006 \end{cases}. \tag{11}$$

Подставив в (9) и в (11) получено:

$$x = e^{-0.125t}(0.54 \cdot \cos(12.0824x) + 0.006 \cdot \sin(12.0824x)) \tag{12}$$

– решение системы при заданных начальных условиях.

РЕШЕНИЕ ОДУ ВТОРОГО ПОРЯДКА ЧИСЛЕННЫМИ МЕТОДАМИ.

Функция *pendulum_dynamics* из файла Integrators.ipunb была изменена для решения ОДУ из формулы 7.

```
3 usages
def pendulum_dynamics(x):

    m = 0.1
    k = 14.6
    b = 0.025

    temp = x[0] #x
    dx = x[1] #z

    ddx = (- b * dx - k * temp) / m

    return np.array([dx, ddx])
```

Рисунок 1 – функция *pendulum_dynamics*

Также были добавлены 2 функции расчета аналитического решения ОДУ.

```
1 usage
def analytical_solution(Tf, h):

    t = np.arange(0, Tf + h, h)
    alpha = -0.125
    omega = 12.082

    C1 = 0.54
    C2 = 0.006

    return [np.exp(alpha * t) * (C1 * np.cos(omega * t) + C2 * np.sin(omega * t)) , t]
```

```
1 usage
def analytical_solution_derivative(Tf, h):

    t = np.arange(0, Tf + h, h)
    alpha = -0.125
    omega = 12.0824
    return [np.exp(alpha * t) * (-6.5261 * np.sin(omega * t) + 0.005 * np.cos(omega * t)), t]
```

Рисунок 2 – функции аналитического решения

Функция *analytical_solution* аналитически считает координату x, а функция *analytical_solution_derivative* считает $\dot{x}$.

Также добавлены вызовы этих функций:

```python
x_vals, t_analytic = analytical_solution(Tf, h)
x_vals_derivative, t_analytic_derivative = analytical_solution_derivative(Tf, h)

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

plt.subplot(1, 3, 1)
plt.plot(t_analytic, x_vals, 'b-',  linewidth=3, label='Analytical Solution')

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')
plt.ylabel('Angle (rad)')
plt.legend()
plt.title('Pendulum Angle vs Time')

plt.subplot(1, 3, 2)
plt.plot(t_analytic_derivative, x_vals_derivative, 'b-', linewidth=3, label='Analytical Solution')
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')
plt.ylabel('Angular Velocity (rad/s)')
plt.legend()
plt.title('Angular Velocity vs Time')

plt.subplot(1, 3, 3)
plt.plot(x_vals, x_vals_derivative, 'b-', linewidth=3, label='Analytical Solution')
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')
```

Рисунок 3 – вызовы функций *analytical_solution* и *analytical_solution_derivative*

Далее будут показаны графики решений.

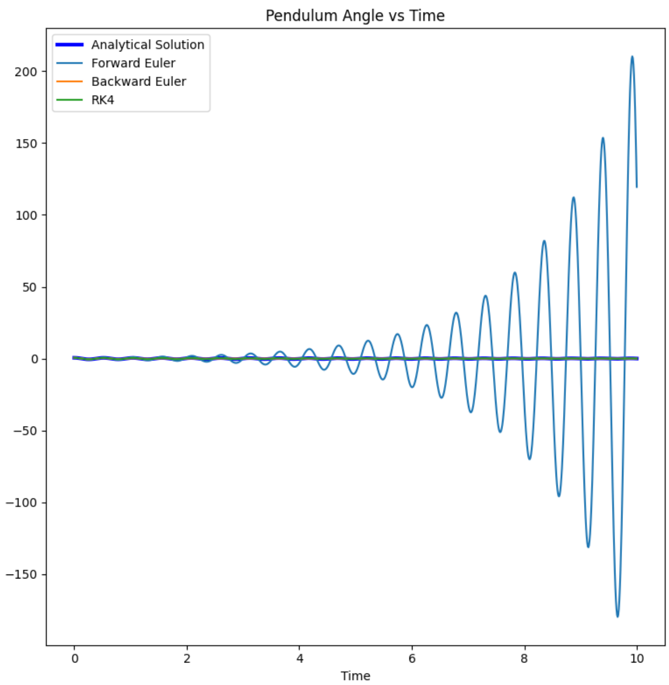


Рисунок 4 – зависимость положения от времени

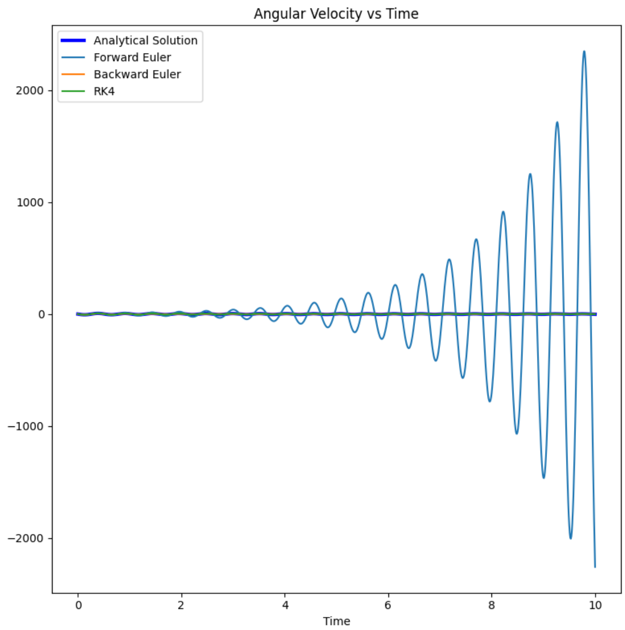


Рисунок 5 – зависимость скорости от времени

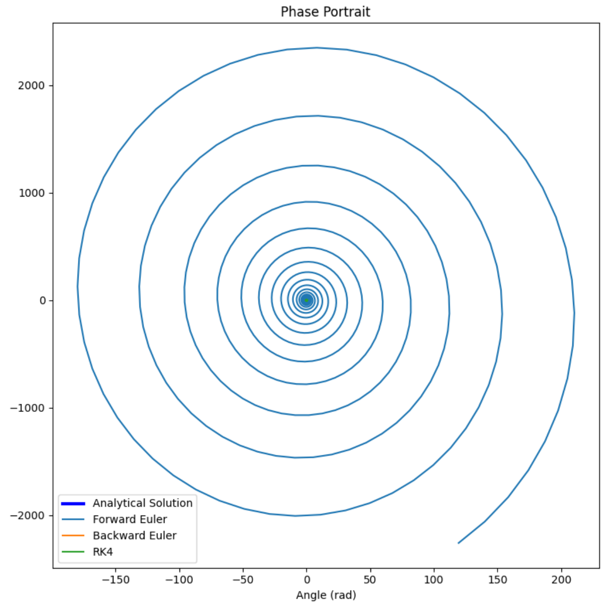


Рисунок 6 – фазовый портрет

Видно, что явный метод Эйлера работает очень плохо. Для более объективной оценки методов был уменьшен шаг интегрирования в 10 раз.

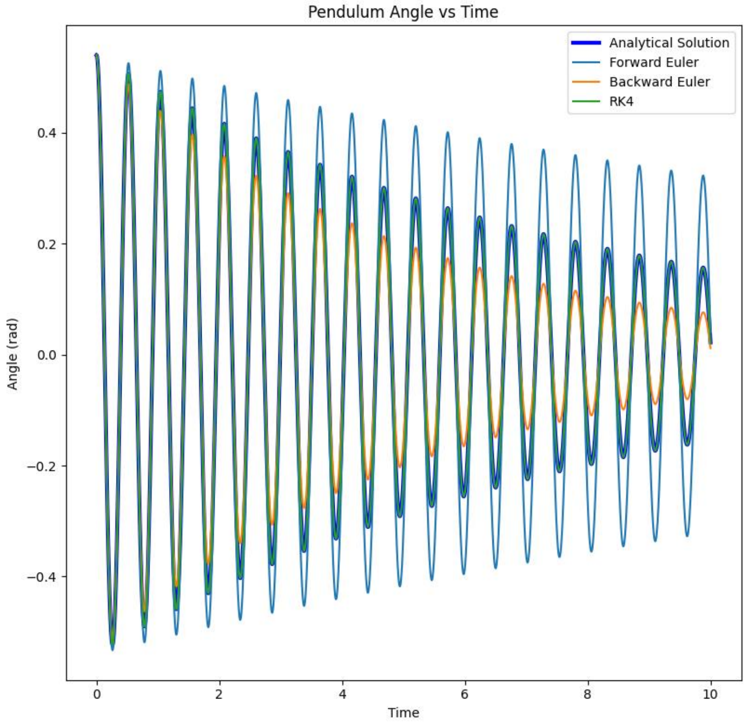


Рисунок 7 – зависимость положения от времени при h = 0.001

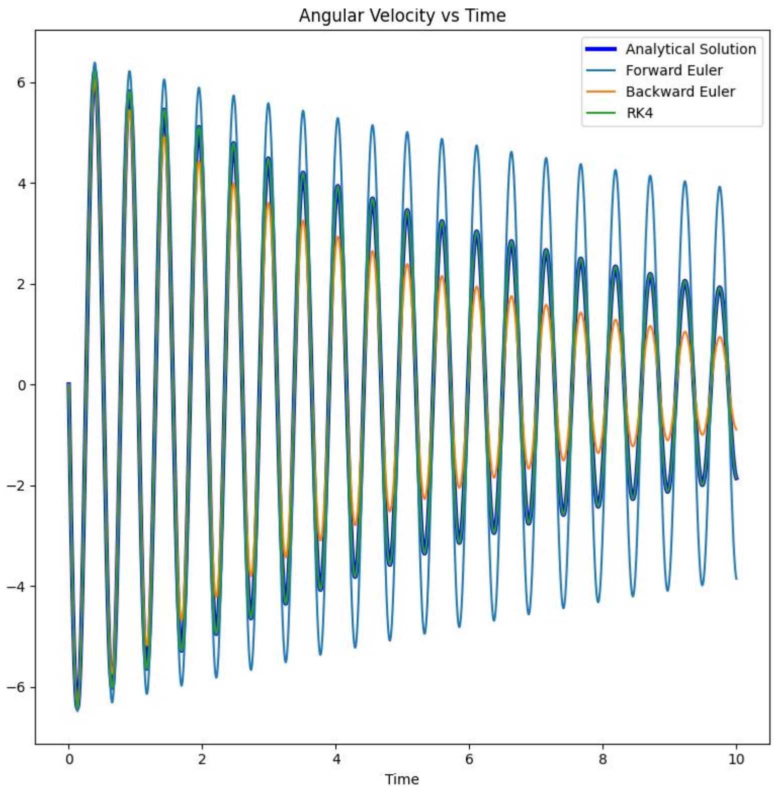


Рисунок 8 – зависимость скорости от времени при h = 0.001

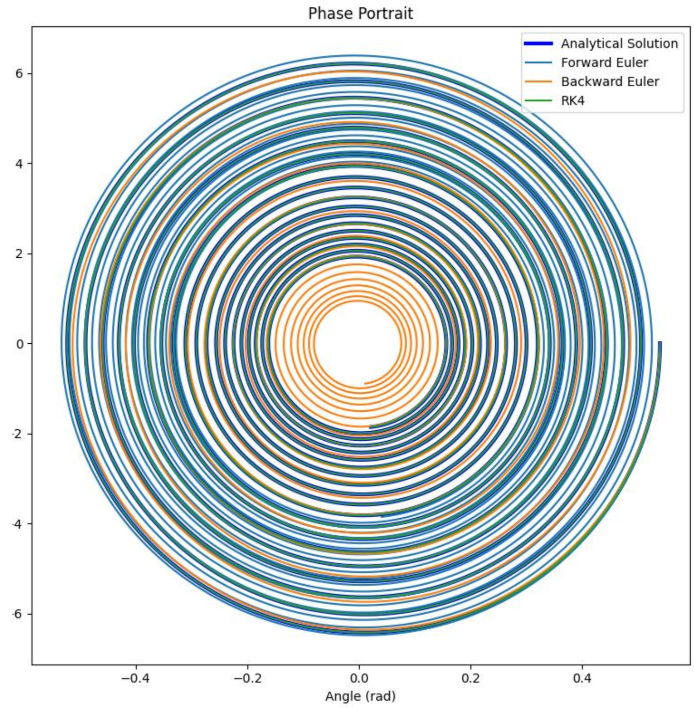


Рисунок 9 – фазовый портрет при h = 0.001

Из рисунков 7, 8, 9 видно, что теперь явный метод Эйлера работает лучше, но все еще хуже обратного метода Эйлера и Рунге – Кутты.

ВЫВОД.

Согласно полученным данным, можно сделать вывод о том, что из всех приведенных методов численного интегрирования метод Рунге – Кутты является самым точным, так как он практически идеально совпадает с данными, полученные аналитическим методом расчета.

Также можно сделать вывод о том, что для более – менее стабильной работы явного метода Эйлера нужно выставлять достаточно маленький шаг интегрирования, в противном же случае его результат будет абсолютно неточным.