Министерство науки и высшего образования Российской Федерации
федеральное

государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Имитационное моделирование робототехнических систем

Практическая работа №1

Студент, группа:

Сергеева Е. С., R4150

Tg:@serkaterina

Преподаватель:

Ракшин Егор Александрович

Санкт-Петербург,

2025

Цель работы: практическое применение численных методов решения для расчета обыкновенных дифференциальных уравнений, включая явный и неявный методы Эйлера, а также метод Рунге-Кутты четвертого порядка. Проведении сравнительного анализа полученных результатов с аналитическим решением заданного линейного дифференциального уравнения второго порядка.

Начальные условия:

Однородное дифференциальное уравнение: $a \cdot \ddot{x} + b \cdot \dot{x} + c \cdot x = d$

Коэффициенты на основе варианта:

a = 6.39

b = −3.67

c = −2.56

d = 0.05

Аналитическое решение ОДУ:

1. Решение однородного уравнения: $6.39 \cdot \ddot{x} + -3.67 \cdot \dot{x} + -2.56 \cdot x = 0.05$
   Характеристическое уравнение: $6.39 \cdot x^2 + -3.67 \cdot x + -2.56 = 0$
   $\lambda_{1,2} = \frac{3.67 \pm \sqrt{(-3.67)^2 - 4 \cdot (-2.56) \cdot 6.39}}{2 \cdot 6.39}$
   $\lambda_1 \approx -0.4079;\ \lambda_2 \approx 0.9822$
   На основании $\lambda_2 > 0$ можем сделать вывод о неустойчивости системы
2. Частное решение ОДУ:
   Вводим константу x_p: $\mathrm{c} \cdot x_p = d$
   $-2.56 \cdot x_p = 0.05$
   $x_p = -0.0195$
3. Общее решение ОДУ: $x(t) = C_1 e^{-0.4079t} + C_1 e^{0.9822t} + x_p$
   $x(t) = C_1 e^{-0.4079t} + C_1 e^{0.9822t} - 0.0195$
4. Вычисление констант:
   $\begin{cases} C_1 + C_2 = x_0 - x_p \\ \lambda_1 C_1 + \lambda_2 C_2 = \dot{x_0} \end{cases}; \begin{cases} C_1 = \frac{(x_0 - x_p)\lambda_2 - \dot{x_0}}{\lambda_2 - \lambda_1} \\ C_2 = \frac{\dot{x_0} - (x_0 - x_p)\lambda_1}{\lambda_2 - \lambda_1} \end{cases}$
   $\lambda_2 - \lambda_1 = 1.3901$
5. Пример (при начальных условиях $x_0 = 0.1; \dot{x_0} = 0$)
   $y_0 = x_0 - x_p = 0.1195$
   $C_1 = \frac{0.1195 \cdot 0.9822}{1.3901} = 0.0844$
   $C_2 = \frac{0 - 0.1195 \cdot (-0.4079)}{1.3901} \approx 0.0351$
   $x(t) \approx -0.0195 + 0.0844^{-0.4079t} + 0.0351^{0.9822}$

Численные методы

Адаптируем код для маятника для решения представленного уравнения $a \cdot \ddot{x} + b \cdot \dot{x} + c \cdot x = d$, дополним его начальными условиями и известными из аналитического решения данными (код вынесен в Приложение 1). Дополнительно для симуляции введем значения:
Время моделирования – T = 0.5 сек.;

Шаг интегрирования – h = 0.01 сек.;

Визуально сравним полученные результаты:

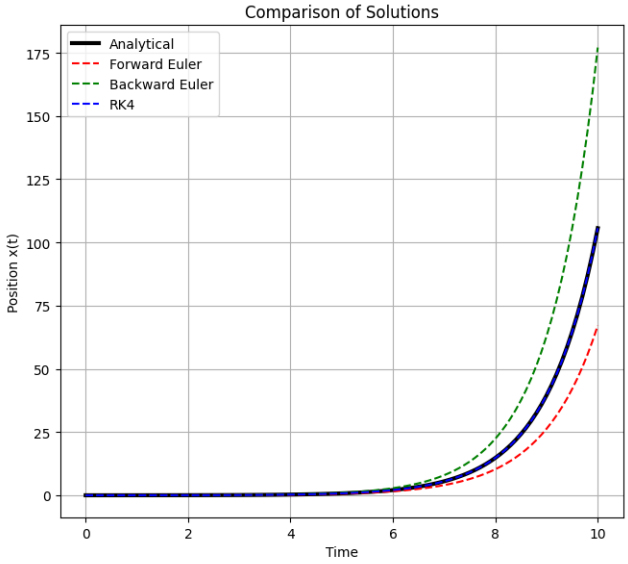

График 1 – График решений (положений x)

Отметим, что все методы корректно отражают экспоненциальный рост, что обусловлена положительным значением λ_2. Однако можно выделить, что метод Рунге-Кутты показывает наиболее точный результат, близкий к аналитическому решению.

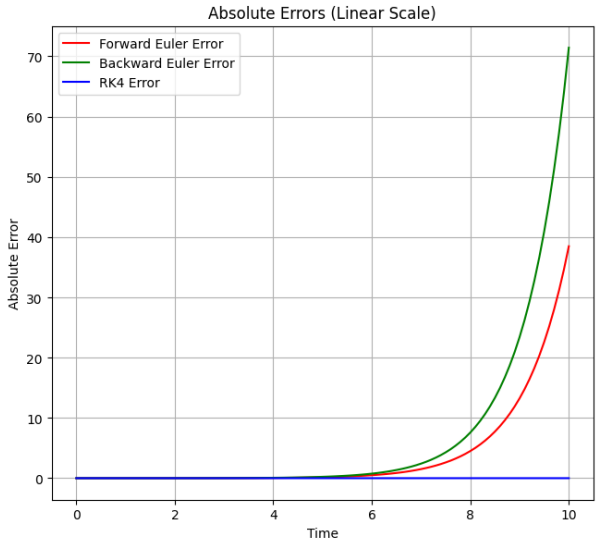

График 2 – График значения линейной ошибки

Визуально рост ошибок в течение времени схож у Явного и Неявного методов Эйлера, однако Неявный показывает более высокую скорость роста ошибок. График метода Рунге-Кутта является стабильным и сохраняет значение ошибки приближенное к 0 на протяжении наблюдаемого времени.

Таблица 1 – Сравнительная таблица максимальной и средней ошибки численных методов

Метод	Ошибка max	Ошибка ср.
Явный метод Эйлера	$1.11 \cdot 10^{-1}$	$1.82 \cdot 10^{-2}$
Неявный метод Эйлера	$1.18 \cdot 10^{-1}$	$1.94 \cdot 10^{-2}$
Метод Рунге-Кутта	$1.36 \cdot 10^{-3}$	$2.47 \cdot 10^{-4}$

Метод Рунге-Кутта показывает более высокую точность в сравнении с другими.
Выводы: Проведён аналитический расчёт исследуемой системы: $a \cdot \ddot{x} + b \cdot \dot{x} + c \cdot x = d$, найден положительный корень, приводящий к расходимости решения. Численное решение методом Рунге-Кутты четвёртого порядка практически совпало с аналитическими результатами. Данный метод являет предпочтительным при выбранном шаге интегрирования из-за минимального значения ошибки. В отличие от него, Явный и Неявный методы Эйлера показали значительные отклонения, показав более низкую точность и приближенные решения рассматриваемой задачи.

Основной код для вычисления численных методов и вывод графиков:

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

# Коэффициенты в соответствии с вариантом
a = 6.39
b = -3.67
c = -2.56
d = 0.05

def system_dynamics(x):

    x_pos = x[0]
    x_vel = x[1]

    x_acc = (d - b*x_vel - c*x_pos) / a

    return np.array([x_vel, x_acc])

def analytical_solution(t, x0):

    lambda1 = -0.4079
    lambda2 = 0.9822

    # Частное решение
    x_particular = -0.0195

    # Начальные условия: x(0) = 0.1, x'(0) = 0
    x0_pos = x0[0]  # 0.1
    x0_vel = x0[1]  # 0

    A = np.array([[1, 1],
                  [lambda1, lambda2]])
    B = np.array([x0_pos - x_particular, x0_vel])
    C1, C2 = np.linalg.solve(A, B)

    print(f"Аналитические параметры:")
    print(f"λ1 = {lambda1}, λ2 = {lambda2}")
    print(f"Частное решение x_p = {x_particular}")
    print(f"Константы: C1 = {C1:.6f}, C2 = {C2:.6f}")
    print(f"Проверка: C1 + C2 = {C1 + C2:.6f} (должно быть {x0_pos -
x_particular:.6f})")

    return C1*np.exp(lambda1*t) + C2*np.exp(lambda2*t) + x_particular

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

            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([0.1, 0.0])
Tf = 5.0
h = 0.01

print("Численное решение...")
x_fe, t_fe = forward_euler(system_dynamics, x0, Tf, h)
x_be, t_be = backward_euler(system_dynamics, x0, Tf, h)
x_rk4, t_rk4 = runge_kutta4(system_dynamics, x0, Tf, h)
```

```
print("\nАналитическое решение...")
x_analytical = analytical_solution(t_fe, x0)

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)

plt.figure(figsize=(20, 12))

# График 1: Сравнение решений
plt.subplot(2, 3, 1)
plt.plot(t_fe, x_analytical, 'k-', linewidth=2, label='Analytical')
plt.plot(t_fe, x_fe[0, :], 'r--', label='Forward Euler')
plt.plot(t be, x be[0, :], 'g--', label='Backward Euler')
plt.plot(t_rk4, x_rk4[0, :], 'b--', label='RK4')
plt.xlabel('Time')
plt.ylabel('Position x(t)')
plt.legend()
plt.title('Comparison of Solutions (UNSTABLE SYSTEM)')
plt.grid(True)


# График 2: Ошибки (линейная шкала)
plt.subplot(2, 3, 4)
plt.plot(t_fe, error_fe, 'r-', label='Forward Euler Error')
plt.plot(t_be, error_be, 'g-', label='Backward Euler Error')
plt.plot(t_rk4, error_rk4, 'b-', label='RK4 Error')
plt.xlabel('Time')
plt.ylabel('Absolute Error')
plt.legend()
plt.title('Absolute Errors (Linear Scale)')
plt.grid(True)

# Вывод численных результатов
print("\n" + "=" * 60)
print("СРАВНЕНИЕ ЧИСЛЕННЫХ МЕТОДОВ")
print("=" * 60)
print(f"Уравнение: {a}·x'' + {b}·x' + {c}·x = {d}")
print(f"Начальные условия: x(0) = {x0[0]}, x'(0) = {x0[1]}")
print(f"Шаг времени: h = {h}")
print(f"Время симуляции: Tf = {Tf}")
print(f"СИСТЕМА НЕУСТОЙЧИВА (λ2 = 0.9822 > 0)")
print()

print("МАКСИМАЛЬНЫЕ АБСОЛЮТНЫЕ ОШИБКИ:")
print(f"  Явный Эйлер:    {np.max(error_fe):.2e}")
print(f"  Неявный Эйлер:  {np.max(error_be):.2e}")
print(f"  Рунге-Кутта 4:  {np.max(error_rk4):.2e}")
print()

print("СРЕДНИЕ АБСОЛЮТНЫЕ ОШИБКИ:")
```

```
print(f"  Явный Эйлер:     {np.mean(error_fe):.2e}")
print(f"  Неявный Эйлер:   {np.mean(error_be):.2e}")
print(f"  Рунге-Кутта 4:   {np.mean(error_rk4):.2e}")
print()

print(f"ЗНАЧЕНИЯ В КОНЕЧНЫЙ МОМЕНТ t = {Tf}:")
print(f"  Аналитическое:   {x_analytical[-1]:.6f}")
print(f"  Явный Эйлер:     {x_fe[0, -1]:.6f}")
print(f"  Неявный Эйлер:   {x_be[0, -1]:.6f}")
print(f"  Рунге-Кутта 4:   {x_rk4[0, -1]:.6f}")


# Подбор размера шага на основании характеристики устойчивости
print("\n" + "=" * 60)
print("АНАЛИЗ УСТОЙЧИВОСТИ С РАЗНЫМИ ШАГАМИ")
print("=" * 60)

time_steps = [0.001, 0.01, 0.05, 0.1]
for h_test in time_steps:
    x_fe_test, t_fe_test = forward_euler(system_dynamics, x0, 2.0, h_test)
    x_analytical_test = analytical_solution(t_fe_test, x0)
    error = np.max(np.abs(x_fe_test[0, :] - x_analytical_test))

    stability = "УСТОЙЧИВО" if error < 1.0 else "НЕУСТОЙЧИВО"
    print(f"h = {h_test:.3f}: Макс. ошибка = {error:.2e} ({stability})")
```