Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет Систем Управления и Робототехники

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

Вариант №30

ПО ДИСЦИПЛИНЕ: «ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ»

Выполнил:
Мищенко И. А.,
336835, гр. R4150
Проверил:
Ракшин Е. А.,
ассистент ФСУиР

Санкт-Петербург,
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Цель работы

Цель работы – изучить численные методы решения обыкновенных дифференциальных уравнений (явный и неявный метод Эйлера, метод Рунге-Кутты 4-го порядка) и сравнить их результаты с аналитическим решением заданного линейного ОДУ второго порядка.

2 Задачи, решаемые при выполнении работы

1) Определить по таблице коэффициенты a, b, c, d.
2) Аналитически решить уравнение второго порядка.
3) Реализовать три численных метода при помощи *«Integrators.ipynb»*.
4) Произвести симуляцию для одинаковых начальных условий.
5) Построить графики аналитического и численных решений.
6) Сравнить эффективность работы методов, сделать выводы.

ИСХОДНЫЕ ДАННЫЕ

Вариант №30

Переменная	Значение
a	-4.45
b	-3.12
c	9.36
d	-1.88

Таблица 1: Коэффициенты ОДУ согласно варианту

Дано однородное дифференциальное уравнение (ОДУ):

$$a\ddot{x} + b\dot{x} + cx = d \tag{1}$$

Подставляем коэффициенты согласно варианту:

$$-4.45 \cdot \ddot{x} - 3.12 \cdot \dot{x} + 9.36 \cdot x = -1.88 \tag{2}$$

ХОД РАБОТЫ

1 Аналитическое решение

Разделим обе части уравнения (2) на коэффициент при $\ddot{x}$:

$$\ddot{x} + 0.7011\dot{x} - 2.1034x = 0.4225 \quad (3)$$

Общее решение неоднородного уравнения представляется в виде:

$$x(t) = x_h(t) + x_p, \quad (4)$$

где $x_h(t)$ – решение соответствующего однородного уравнения,

а x_p – частное решение.

1.1 Решение однородного уравнения

Рассмотрим однородное уравнение, выраженное из (3) с помощью (4):

$$\ddot{x} + 0.7011\dot{x} - 2.1034x = 0 \quad (5)$$

Составим характеристическое уравнение из (5):

$$r^2 + 0.7011r - 2.1034 = 0 \quad (6)$$

Найдем корни (6): 0,4915

$$r_{1,2} = \frac{-0.7011 \pm \sqrt{(0.7011)^2 + 4 \cdot (2.1034)}}{2} \quad (7)$$

Численно:

$$r_1 = \frac{-0.7011 + 2.9841}{2} = 1.1415, \qquad r_2 = \frac{-0.7011 - 2.9841}{2} = -1.8426.$$

Таким образом, решение однородного уравнения:

$$x_{h(t)} = C_1 e^{1.1415t} + C_2 e^{-1.8426t} \quad (8)$$

1.2 Частное решение

Поскольку правая часть уравнения – константа d, частное решение – тоже константа:

$$x_p = \frac{d}{c} = \frac{-1.88}{9.36} = -0.20085 \tag{9}$$

1.3 Полное решение

Выразим полное решение уравнения из формулы (4), и решений (8), (9):

$$x(t) = C_1 e^{1.1415t} + C_2 e^{-1.8426t} - 0.20085 \tag{10}$$

1.4 Определение констант C_1 и C_2

Используем стандартные начальные условия практики:

$$\begin{gathered} x(0) = x_0 = 0.01, \\ x'(0) = \dot{x}_0 = 0 \end{gathered} \tag{11}$$

Запишем условие 1:

$$\begin{gathered} x(0) = C_1 + C_2 = 0.20085 + 0.01, \\ C_1 + C_2 = 0.21085 \end{gathered} \tag{12}$$

Запишем условие 2:

$$\begin{gathered} x'(t) = C_1 r_1 e^{r_1 t} + C_2 r_2 e^{r_2 t}, \\ x'(0) = C_1 r_1 + C_2 r_2 = 0 \end{gathered} \tag{13}$$

Получаем систему:

$$\begin{cases} C_1 + C_2 = 0.21085, \\ 1.1415C_1 - 1.8426C_2 = 0 \end{cases} \tag{14}$$

Из второго уравнения:

$$C_1 = 1.6142\, C_2 \tag{15}$$

Подставляем в первое:

$$1.6142C_2 + C_2 = 0.21085,$$
$$2.6142C_2 = 0.21085,$$
$$C_2 = 0.081 \tag{16}$$

Тогда:

$$C_1 = 1.6142 \cdot 0.081 = 0.131 \tag{17}$$

1.5 Итоговое аналитическое решение

$$x(t) = 0.131e^{1.1415t} + 0.081e^{-1.8426t} - 0.20085 \tag{18}$$

Положительный корень r_1 – решение содержит растущую экспоненту – система неустойчива.

2 Решение численными методами

Ниже приведен листинг программного решения задачи.

```
# Импортируем необходимые библиотеки
import numpy as np
import matplotlib.pyplot as plt

# Определяем параметры для варианта №30
a = -4.45
b = -3.12
c = 9.36
d = -1.88

# Определяем функцию для решения ОДУ
def ODE(x):
    theta = x[0]
    theta_dot = x[1]

    theta_ddot = (d - b * theta_dot - c * theta) / a

    return np.array([theta_dot, theta_ddot])

# Определяем константы для аналитического решения
C1 = 0.131
C2 = 0.081
xp = -0.20085
r1 = 1.1415
r2 = -1.8426
```

```python
# Определяем функцию для аналитического решения
def x_analytical(t):
    return C1 * np.exp(r1 * t) + C2 * np.exp(r2 * t) + xp

# Определяем функции для решения численных методов
def forward_euler(fun, x0, Tf, h):
    """
    Explicit Euler integration method
    """
    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):
    """
    Implicit Euler integration method using fixed-point iteration
    """
    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 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):
    """
    4th order Runge-Kutta integration method
    """
    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

# Расчет численных методов
```

```
x0 = np.array([0.01, 0.0])  # Начальные условия: [положение, скорость]
Tf = 10.0
h = 0.01

# Forward Euler
x_fe, t_fe = forward_euler(ODE, x0, Tf, h)

# Backward Euler
x_be, t_be = backward_euler(ODE, x0, Tf, h)

# Runge-Kutta 4
x_rk4, t_rk4 = runge_kutta4(ODE, x0, Tf, h)

# Расчет аналитического решения
t = t_fe
x_a = x_analytical(t)

# Выводим сравнительные графики методов
plt.figure(figsize=(15, 10))
plt.title('ODE solutions')
plt.plot(t, x_a, 'r-', label='Analytical')
plt.plot(t_fe, x_fe[0], 'g--', label='Forward Euler')
plt.plot(t_be, x_be[0], 'b--', label='Backward Euler')
plt.plot(t_rk4, x_rk4[0], 'k--', label='RK4')
plt.xlabel('Time')
plt.ylabel('x(t)')
plt.legend()
plt.grid()
plt.show()
```

Листинг 1: Программный код для численных методов

Программная часть работы реализована на языке Python с использованием библиотек NumPy и Matplotlib. Код состоит из нескольких логических блоков, каждый из которых отвечает за определённый этап моделирования динамической системы.

В начале программы задаются параметры дифференциального уравнения второго порядка, соответствующие варианту №30. Данные коэффициенты используются для составления правой части ОДУ, которое предварительно сведено к системе первого порядка. Для этого реализована функция ODE(x), принимающая вектор состояния и возвращающая его производные: первая компонента соответствует скорости, а вторая вычисляется по формуле, полученной из нормализованного дифференциального уравнения. Такой подход позволяет интегрировать исходное ОДУ с использованием любых методов численного решения.

После этого формируется аналитическое решение. В коде задаются заранее вычисленные константы и частное решение. Функция x_analytical(t) вычисляет значение аналитического решения в любой момент времени. Это необходимо для последующего сравнения с численными методами.

Далее реализованы три метода интегрирования: явный Эйлер, неявный Эйлер и метод Рунге–Кутты 4-го порядка. Каждый из них оформлен в отдельную функцию. Метод forward_euler использует простейшую схему интегрирования, где новое значение состояния вычисляется путём добавления шага, умноженного на производную в текущей точке. Такой метод прост и быстр, но обладает низкой точностью и слабо устойчив при моделировании неустойчивых систем.

Метод backward_euler является имплицитным, то есть производная вычисляется в конце шага. Из-за отсутствия явного выражения для следующего состояния применяется итерационный метод простой итерации (fixed-point iteration). На каждом шаге выполняется несколько итераций до достижения заданной точности, что делает метод более устойчивым, но увеличивает вычислительные затраты.

Метод runge_kutta4 реализует классическую схему Рунге–Кутты четвертого порядка. В нём последовательно вычисляются четыре промежуточных производных (k1–k4), после чего состояние обновляется взвешенной суммой этих производных. Благодаря этому метод сочетает высокую точность и достаточную устойчивость, что делает его одним из наиболее надёжных среди методов общего назначения.

На последнем этапе программы производится численная интеграция для всех трёх методов с одинаковыми начальными условиями, шагом интегрирования и длительностью моделирования. Затем строится график, на котором одновременно отображаются аналитическое решение и результаты всех численных методов. Такой подход позволяет наглядно сравнить точность и устойчивость различных схем интегрирования при моделировании неустойчивой динамической системы.

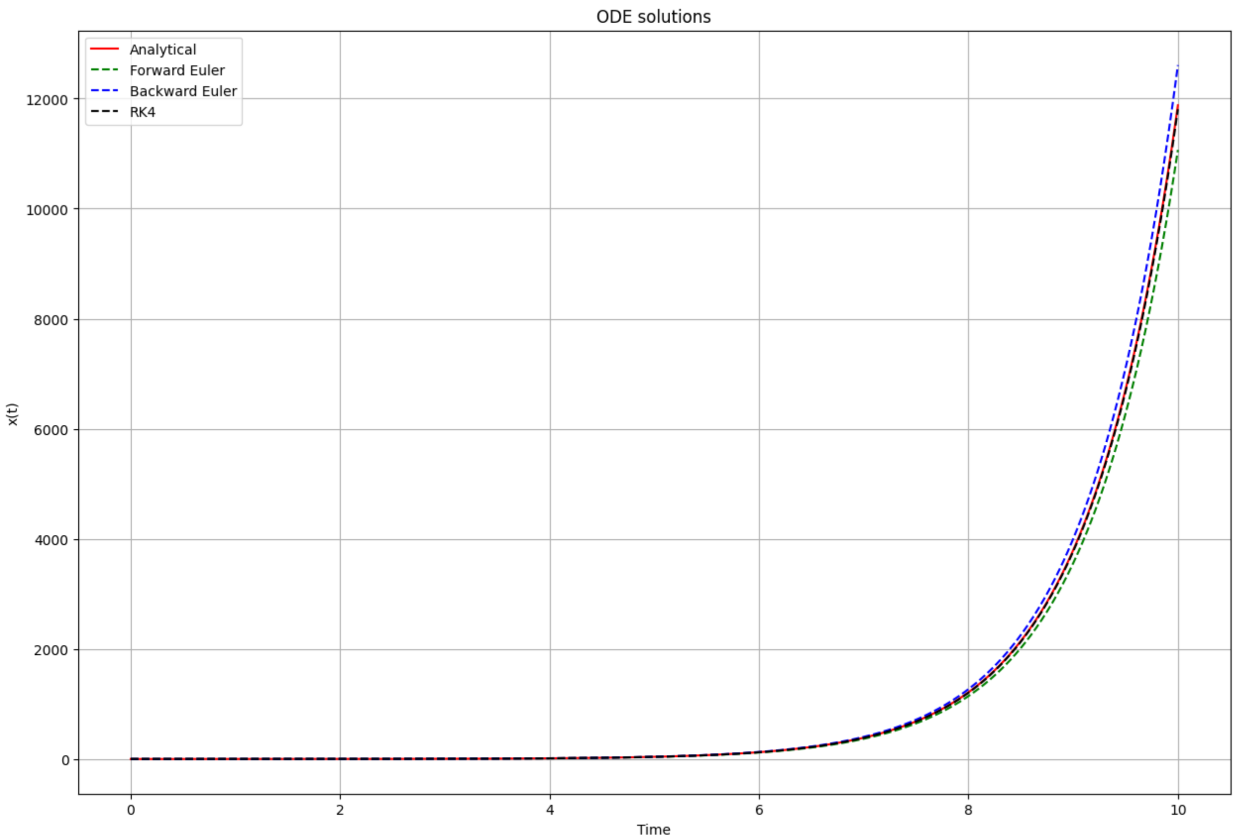

График 1: Результаты симуляции методов и аналитического решения

На итоговом графике представлены четыре кривые: аналитическое решение и результаты трёх численных методов интегрирования. Аналитическое решение показывает эталонную траекторию системы, отражающую её неустойчивый характер: наличие положительного корня характеристического уравнения приводит к росту решения с течением времени.

Метод Рунге–Кутты 4-го порядка демонстрирует наилучшее совпадение с аналитическим решением. Его высокая точность обусловлена использованием нескольких промежуточных вычислений на каждом шаге, что уменьшает локальную и глобальную погрешность.

Неявный метод Эйлера также показывает достаточно устойчивую и сглаженную траекторию, однако заметно отстаёт от аналитического решения, особенно при увеличении времени. Это связано с тем, что метод стремится подавлять рост, присущий реальной системе.

Метод явного Эйлера демонстрирует наибольшую погрешность. При интегрировании неустойчивых систем он склонен к существенным отклонениям, что особенно хорошо видно при больших значениях времени. Это подтверждает известный факт: явный Эйлер плохо подходит для жёстких и неустойчивых систем.

В целом, визуальное сравнение ясно показывает различия в точности и устойчивости методов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы было исследовано поведение линейного дифференциального уравнения второго порядка, соответствующего варианту №30. Было выполнено аналитическое решение ОДУ, найдено общее и частное решения, рассчитаны константы по заданным начальными условиям, а также проанализирована структура характеристического уравнения. Положительный корень характеристического уравнения подтвердил неустойчивость системы: одна из экспонент в решении приводит к её росту во времени.

Для численного решения задачи были реализованы три метода интегрирования: явный Эйлер, неявный Эйлер и метод Рунге–Кутты 4-го порядка. Результаты численной интеграции были сопоставлены с аналитическим решением.

Анализ показал, что:

- метод Рунге–Кутты 4-го порядка обеспечивает наилучшее совпадение с аналитическим решением даже при сравнительно небольшом шаге интегрирования;
- неявный Эйлер демонстрирует устойчивое поведение, сглаживает рост решения, однако уступает по точности RK4;
- явный Эйлер проявляет значительную ошибку и уходит от аналитического решения, что особенно заметно в неустойчивых системах.

Полученные результаты подтверждают важность выбора подходящего интегратора для моделирования динамических систем. При работе с жёсткими или неустойчивыми уравнениями простые методы, такие как явный Эйлер, демонстрируют недостаточную точность и могут привести к существенным ошибкам. Метод Рунге–Кутты 4-го порядка является оптимальным выбором в большинстве практических задач, поскольку сочетает устойчивость и высокую точность.

Таким образом, поставленные цели работы были выполнены: аналитическое решение получено, численные методы реализованы и исследованы, произведён сравнительный анализ точности и устойчивости методов интегрирования.