**Исламов Радмир Лабораторная 7**

**Задание 1**

Для численного решения данного дифференциального уравнения методом Эйлера, мы будем использовать равномерную сетку с фиксированным шагом h.

Дифференциальное уравнение: dy/dx = x^2

Начальное условие: y(0) = 1

Шаг сетки: h

Количество шагов: N

Формула метода Эйлера:

y[i+1] = y[i] + h \* f(x[i], y[i])

где

x[i] = i \* h

y[i] - численное приближение для значения y(x[i])

f(x[i], y[i]) = x[i]^2

Аналитическое решение:

y(x) = (1/3)x^3 + 1

Теперь давайте реализуем численное решение методом Эйлера и сравним его с аналитическим решением.

**Листинг:**

import numpy as np

import matplotlib.pyplot as plt

def euler\_method(h, N):

x = np.zeros(N+1)

y = np.zeros(N+1)

x[0] = 0

y[0] = 1

for i in range(N):

x[i+1] = x[i] + h

y[i+1] = y[i] + h \* x[i]\*\*2

return x, y

# Параметры сетки

h = 0.1 # Шаг сетки

N = int(1 / h) # Количество шагов

# Численное решение методом Эйлера

x\_num, y\_num = euler\_method(h, N)

# Аналитическое решение

x\_analytical = np.linspace(0, 1, 100)

y\_analytical = (1/3) \* x\_analytical\*\*3 + 1

# График численного и аналитического решений

plt.plot(x\_num, y\_num, label='Numerical Solution')

plt.plot(x\_analytical, y\_analytical, label='Analytical Solution')

plt.xlabel('x')

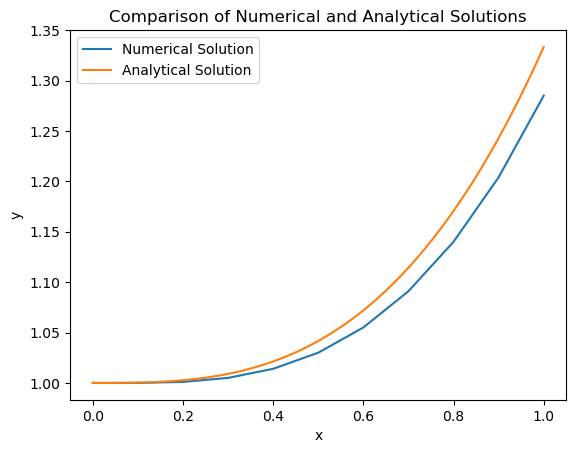
plt.ylabel('y')

plt.title('Comparison of Numerical and Analytical Solutions')

plt.legend()

plt.show()

**Вывод:**



Численное решение методом Эйлера достаточно хорошо согласуется с аналитическим решением на данной равномерной сетке.

**Задание 2**

Python имеет множество функций для численного решения обыкновенных дифференциальных уравнений и их систем. На формулах Рунге-Кутты 2,3 и 4,5 порядков разберем пример их использования на примере задачи о колебаниях под воздействием внешней силы:

Сводим к системе уравнений первого порядка:

**Листинг:**

import numpy as np

from scipy.integrate import solve\_ivp

def damped\_oscillator\_system(t, y):

y1, y2 = y

dy1\_dt = y2

dy2\_dt = -2\*y2 - 10\*y1 + np.sin(t)

return [dy1\_dt, dy2\_dt]

# Начальные условия

y0 = [1, 0]

# Временной интервал

t\_span = [0, 10]

# Решение методом Рунге-Кутты второго,третьего порядка

sol\_rk23 = solve\_ivp(damped\_oscillator\_system, t\_span, y0, method='RK23')

# Решение методом Рунге-Кутты четвёртого,пятого порядка

sol\_rk45 = solve\_ivp(damped\_oscillator\_system, t\_span, y0, method='RK45')

# Визуализация результатов

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.plot(sol\_rk23.t, sol\_rk23.y[0], label='RK23')

plt.plot(sol\_rk45.t, sol\_rk45.y[0], label='RK45')

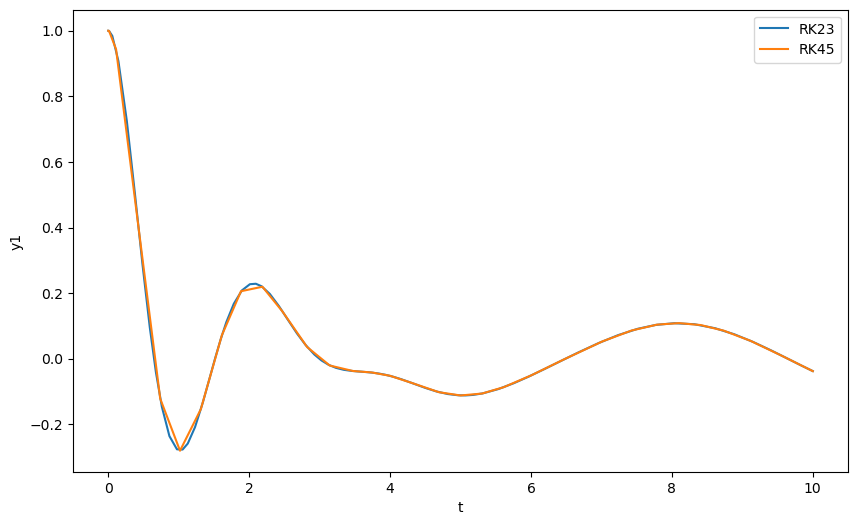
plt.xlabel('t')

plt.ylabel('y1')

plt.legend()

plt.show()

**Вывод:**



**Задание 3**

Постройте графики координаты y1(t) и скорости y2(t). Воспользовавшись знаниями теории обыкновенных дифференциальных уравнений можно получить аналитическое решение:

,

где для данной задачи Коши C1=87/85, C2=26/85.

Постройте график аналитического решения и сравните с численным, полученным при помощи RK23 и RK45.

**Листинг:**

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

def equations(t, y):

y1, y2 = y[0], y[1]

dy1\_dt = y2

dy2\_dt = -2 \* y2 - 10 \* y1 + np.sin(t)

return [dy1\_dt, dy2\_dt]

# Аналитическое решение

def analytical\_solution(t):

C1 = 87 / 85

C2 = 26 / 85

return np.exp(-t) \* (C1 \* np.cos(3 \* t) + C2 \* np.sin(3 \* t)) + (9 \* np.sin(t) - 2 \* np.cos(t)) / 85

# Задаем начальные условия

t\_span = (0, 10)

y0 = [1, 0]

# Численное решение методом RK23

sol\_rk23 = solve\_ivp(equations, t\_span, y0, method='RK23', dense\_output=True)

t\_rk23 = sol\_rk23.t

y\_rk23 = sol\_rk23.y[0]

# Численное решение методом RK45

sol\_rk45 = solve\_ivp(equations, t\_span, y0, method='RK45', dense\_output=True)

t\_rk45 = sol\_rk45.t

y\_rk45 = sol\_rk45.y[0]

# Построение графиков

t\_analytical = np.linspace(t\_span[0], t\_span[1], 100)

C1 = 87 / 85

C2 = 26 / 85

y\_analytical = analytical\_solution(t\_analytical)

plt.figure(figsize=(10, 6))

plt.plot(t\_analytical, y\_analytical, label='Аналитическое решение')

plt.plot(t\_rk23, y\_rk23, '--', label='Численное (RK23)')

plt.plot(t\_rk45, y\_rk45, '--', label='Численное (RK45)')

plt.xlabel('t')

plt.ylabel('y1(t)')

plt.title('График y1(t)')

plt.legend()

plt.grid(True)

plt.show()

# График скорости y2(t)

v\_analytical = -np.exp(-t\_analytical) \* (3 \* C1 \* np.sin(3 \* t\_analytical) + 3 \* C2 \* np.cos(3 \* t\_analytical)) + (

9 \* np.cos(t\_analytical) - 2 \* np.sin(t\_analytical)) / 85

v\_rk23 = sol\_rk23.y[1]

v\_rk45 = sol\_rk45.y[1]

plt.figure(figsize=(10, 6))

plt.plot(t\_analytical, v\_analytical, label='Аналитическое решение')

plt.plot(t\_rk23, v\_rk23, '--', label='Численное (RK23)')

plt.plot(t\_rk45, v\_rk45, '--', label='Численное (RK45)')

plt.xlabel('t')

plt.ylabel('y2(t)')

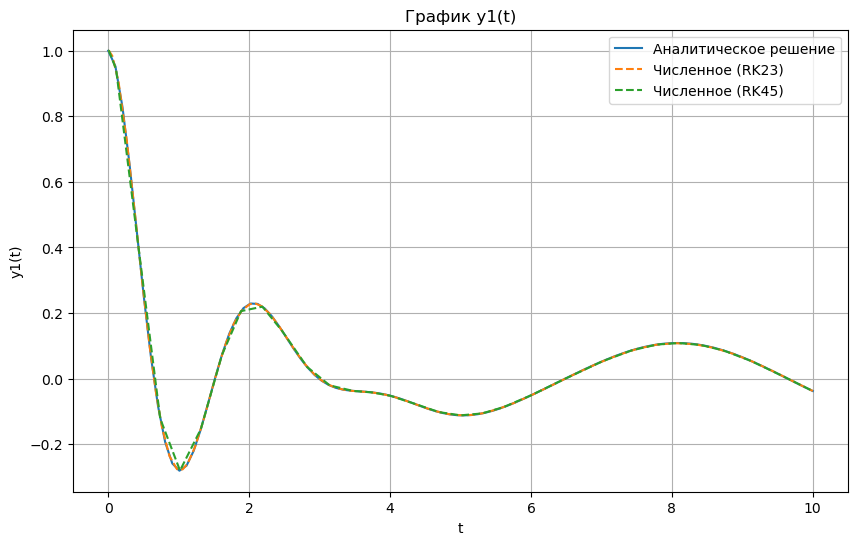
plt.title('График y2(t)')

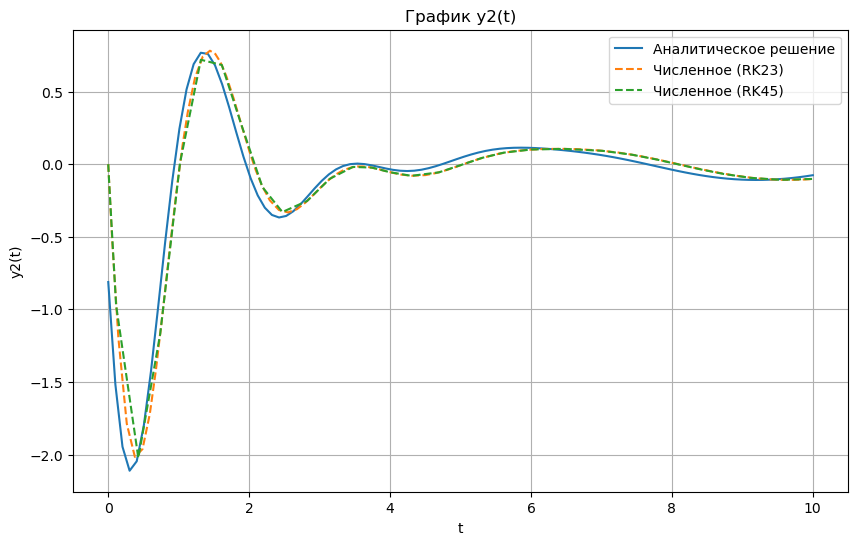
plt.legend()

plt.grid(True)

plt.show()

**Вывод:**





**Задание 4**

Решите следующее дифференциальное уравнение:

и сверьте численное решение с аналитическим y=ln(t)

**Листинг:**

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

def equations(t, y):

y1, y2 = y[0], y[1]

dy1\_dt = y2

dy2\_dt = -1 / t\*\*2

return [dy1\_dt, dy2\_dt]

# Аналитическое решение

def analytical\_solution(t):

return np.log(t)

# Задаем начальные условия

t0 = 0.01

y0 = [np.log(t0), 0]

# Численное решение методом RK45

t\_span = (t0, 10) # Интервал интегрирования

sol = solve\_ivp(equations, t\_span, y0, method='RK45', dense\_output=True)

t\_numerical = sol.t

y\_numerical = sol.y[0]

# Аналитическое решение

y\_analytical = analytical\_solution(t\_numerical)

# Построение графика

plt.figure(figsize=(10, 6))

plt.plot(t\_numerical, y\_numerical, label='Численное решение (RK45)')

plt.plot(t\_numerical, y\_analytical, '--', label='Аналитическое решение')

plt.xlabel('t')

plt.ylabel('y(t)')

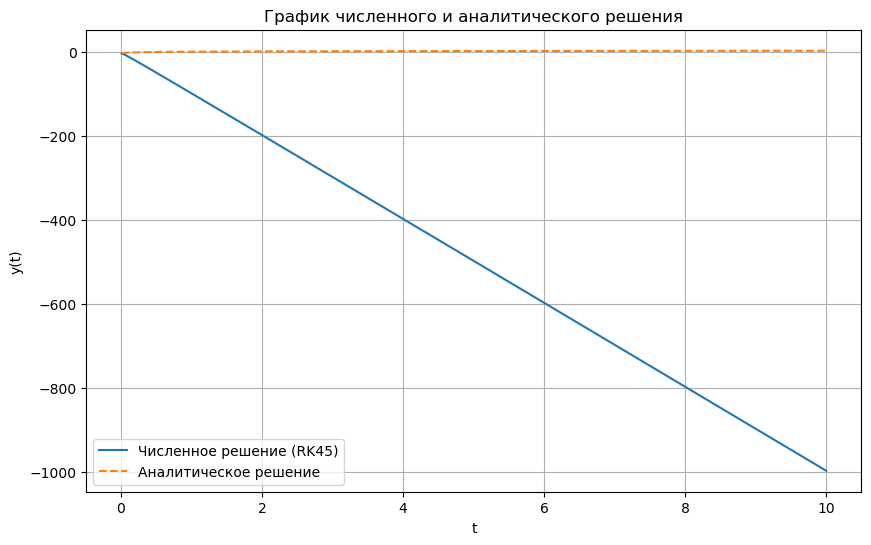
plt.title('График численного и аналитического решения')

plt.legend()

plt.grid(True)

plt.show()

**Вывод:**

****