МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ТЭВН

Лабораторная работа № 3

“Решение дифференциальных уравнений”

Вариант № 19

Факультет: ФЭН

Группа: ЭН2-31

Студент: Полозов А.А.

Преподаватель: Петрищев А.В.

Новосибирск 2024

# 1. Цель работы

Познакомиться с численными методами решения обыкновенных дифференциальных уравнений (ОДУ) и систем ОДУ. Научиться решать ОДУ в Python. Путём решения системы дифференциальных уравнений, рассчитать переходный процесс в контуре с двумя реактивными элементами.

# 2. Задание № 1

Написать программу, решающую заданное в приложении № 1 дифференциальное уравнение:

1. выполнить решение при помощи численного метода, указанного в варианте и при помощи встроенной функции (метод BDF);
2. построить графики решений, полученных указанными методами и график аналитического решения (приведено в приложении № 1) и сделать вывод о правильности найденного решения.

# 3. Пояснения к заданию № 1

Условия задания, соответствующие 19 варианту, приведены в таблице № 1.

Таблица № 1 – Условия задания № 1

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Дифференциальное уравнение и начальное условие** | **Частное аналитическое решение** | **Программируемый метод решения** |
| 19 |  |  | Неявный метод Эйлера |

Форма Коши для данного уравнения будет выглядеть следующим образом:

Блок-схема алгоритма решения представлена на рис. № 1.



Рис № 1 – Блок-схема алгоритма решения

## Листинг программы

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

import locale

# Задание правой части дифф. уравнения

def f(t, y):

    return 2\*np.square(t) - 1 - 2\*y

# Задание шага решения

h = 0.1

# Задание интервала решения

t\_max = 5

# Задание точности решения

e = np.power(10.0, -2)

# Задание массива точек

t = np.arange(0, t\_max+h, h)

# Задание массива значений

y = t\*0

# Задание начального условия

y[0] = 2

# Неявный метод Эйлера

for k in range(1, len(t)):

    z = 1

    while np.abs(z-y[k]) > e:

        y[k] = z

        z = y[k-1] + h\*f(t[k], z)

# Решение с помощью библиотеки

sol = solve\_ivp(f, [0, t\_max], [y[0]])

# Постройка графика

# Настройки графика

fig, ax = plt.subplots()

locale.setlocale(locale.LC\_NUMERIC, "de\_RU")

font = {'family': 'Times New Roman',

        'size': 12}

plt.rc('font', \*\*font)

ax.ticklabel\_format(useLocale=True)

ax.grid(linewidth = 0.3)

ax.spines['left'].set\_position('zero')

ax.spines['right'].set\_color('none')

ax.yaxis.tick\_left()

ax.spines['bottom'].set\_position('zero')

ax.spines['top'].set\_color('none')

ax.xaxis.tick\_bottom()

ax.plot(t, y, color = 'red', label = 'Решение с помощью программы')

ax.plot(sol.t, sol.y[0], color = 'green', label = 'Решение с помощью библиотеки')

y = np.square(t) - t

ax.plot(t, y, color = 'blue', label = ' Аналитическое решение')

plt.legend()

plt.show()

## Результаты работы программы (рис. 2)

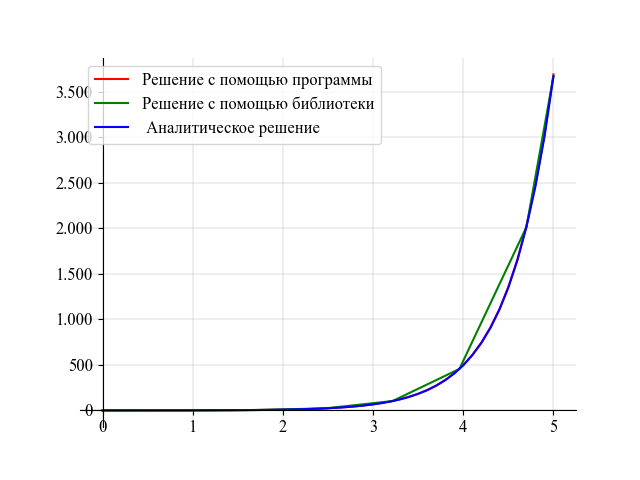


Рис. 2 – Результат работы программы

# 4. Задания № 2

Для заданной в приложении № 2 электрической схемы выполнить расчёт переходного процесса (тока в индуктивности и напряжения на ёмкости). Для решения уравнений использовать функцию , метод BDF. Параметры электрической схемы приведены в таблице 3 приложения.

# 5. Пояснения к заданию № 2

Условия задания, соответствующие 19 варианту, приведены в таблице № 2.

Таблица № 2 – Условия задания № 2

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Электрическая схема** | **Система ОДУ для расчёта переходного процесса** |
| 19 |  |  |

Приведём уравнения к форме Коши:

## Листинг программы

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

import locale

# Параметры схемы

R1 = 400

R2 = 40

L = 2

C = 40

E = 200

# Функция с правыми частями системы уравнений

def f(t, y):

    global R1, R2, L, C, E

    u = y[0]

    i = y[1]

    du = (R1\*i - u)/(C\*(R1 + R2))

    di = (1/L)\*(E - u - R2\*(R1\*i - u)/(R1 + R2))

    return [du, di]

# Интервал решения

t\_max = 0.5

# Начальные условия

u\_0 = 0

i\_0 = 0

# Решение с помощью библиотеки

sol = solve\_ivp(f, [0, t\_max], [u\_0, i\_0], method = 'BDF')

t = sol.t

u = sol.y[0, :]

i = sol.y[1, :]

# Постройка графика

fig, ax = plt.subplots()

locale.setlocale(locale.LC\_NUMERIC, "de\_RU")

font = {'family': 'Times New Roman',

        'size': 12}

plt.rc('font', \*\*font)

ax.ticklabel\_format(useLocale=True)

ax.grid(linewidth = 0.3)

ax.spines['left'].set\_position('zero')

ax.spines['right'].set\_color('none')

ax.yaxis.tick\_left()

ax.spines['bottom'].set\_position('zero')

ax.spines['top'].set\_color('none')

ax.xaxis.tick\_bottom()

ax.plot(t, u, color = 'red', label = 'График напряжения на конденсаторе')

plt.legend()

fig, ax = plt.subplots()

locale.setlocale(locale.LC\_NUMERIC, "de\_RU")

font = {'family': 'Times New Roman',

        'size': 12}

plt.rc('font', \*\*font)

ax.ticklabel\_format(useLocale=True)

ax.grid(linewidth = 0.3)

ax.spines['left'].set\_position('zero')

ax.spines['right'].set\_color('none')

ax.yaxis.tick\_left()

ax.spines['bottom'].set\_position('zero')

ax.spines['top'].set\_color('none')

ax.xaxis.tick\_bottom()

ax.plot(t, i, color = 'green', label = 'График тока в катушке')

plt.legend()

plt.show()

## Результаты работы программы (рис. 3)

|  |  |
| --- | --- |
|  |  |

Рис. 3 – Результат работы программы

# 6. Выводы

Были изучены численные методы решения обыкновенных дифференциальных уравнений и системы ОДУ в Python.