МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Ничипорук Роман Олегович

Численное интегрирование

Отчет по лабораторной работе № 3
Вариант 10
Численные методы
студента 3-го курса 4-ой группы

Преподаватель:
Левчук Е.А.

2024

ОГЛАВЛЕНИЕ

ГЛАВА 1
Введение

Лабораторная работа была выполнена с использованием языка програмирования Python.

При реализации функций были применены следующие модули:

- math - модуль предоставляет обширный функционал для работы с числами;

- matplotlib - популярная Python-библиотека для визуализации данных. Она используется для создания любых видов графиков: линейных, круговых диаграмм, построчных гистограмм и других - в зависимости от задач;

- numpy - это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами.

ГЛАВА 2

Задание 1

2.1 Условие

Написать программу, которая находит решение задачи Коши указанными в варианте методами с шагом $h = 10^{-3}$. Вывести графики полученных решений, а также модуль разности решений в крайней правой точке интервала, на котором задана задача.

$$\begin{cases} u' = u + v - w + \sqrt{x} \\ v' = -3u - 3v + 3w \\ w' = -2u - 2v + 2w + x^2 \end{cases}, \ u(0) = 2, \ v(0) = 3, \ w(0) = 0, \ x \in [0; 1].$$

Используя неявный метод Эйлера.

2.2 Использованная теория

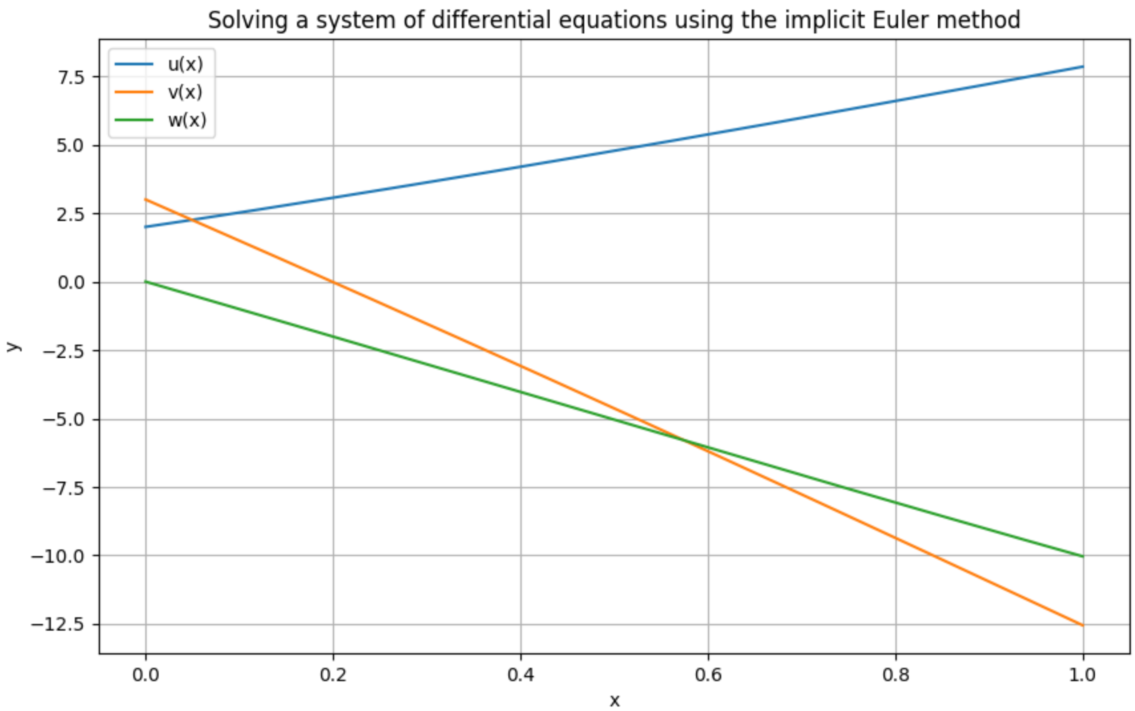
Случай неравноотстоящих узлов

Неявный метод Эйлера:

$$y_{i+1} = y_i + hf(x_{x+1}, y_{y+1}), \ y_0 = u_0 \tag{2.1}$$

Под p понимается порядок погрешности использованного численного метода.

2.3 Результаты




```
Modulus of the difference of solutions at the rightmost point of the interval:
    [4.71183199e-04 4.35413345e-05 5.29860890e-04]
```

Листинг 2.1 terminal

ГЛАВА 3

Задание 2

3.1 Условие

Написать программу, которая находит решение задачи Коши указанными в варианте методами с шагом $h = 10^{-3}$. Вывести графики полученных решений, а также модуль разности решений в крайней правой точке интервала, на котором задана задача.

$$\begin{cases} u' = u + v - w + \sqrt{x} \\ v' = -3u - 3v + 3w \\ w' = -2u - 2v + 2w + x^2 \end{cases}, \; u(0) = 2, \; v(0) = 3, \; w(0) = 0, \; x \in [0; 1].$$

Используя метод Рунге-Кутта 4-го порядка.

3.2 Использованная теория

Случай неравноотстоящих узлов

Метод Рунге-Кутта 4-го порядка:

$$y_{i+1} = y_i + \frac{h}{6}(k_0 + 2k_1 + 2k_2 + k_3), \tag{3.1}$$

$$k_0 = f(x_i, y_i), \tag{3.2}$$

$$k_1 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_0), \tag{3.3}$$

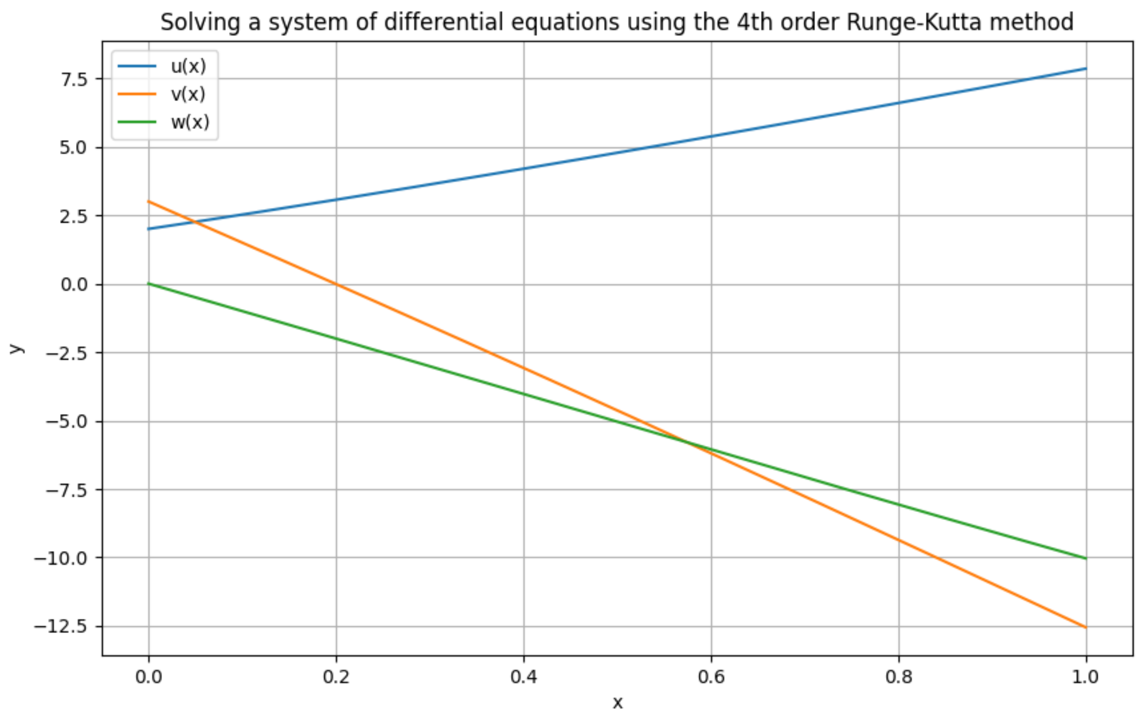
$$k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1), \tag{3.4}$$

$$k_3 = f(x_i + h, y_i + hk_2), \tag{3.5}$$

$$y_0 = u_0. \tag{3.6}$$

Под p понимается порядок погрешности использованного численного метода.

3.3 Результаты




```
Modulus of the difference of solutions at the rightmost point of the interval:
    [4.71183199e-04 4.35413345e-05 5.29860890e-04]
```

Листинг 3.1 terminal

ГЛАВА 4

Листинг программы

```python
from Lab_3.implicit_euler_method import *
from Lab_3.runge_kutta_method_4th_order import *
from Lab_3.utility import *
from Lab_3.input_data import *


x_values, y_values = implicit_euler(f, x_values, u0, h, num_steps)
display_data(x_values, y_values,
             'Solving a system of differential equations using the implicit Euler method')

y_imt_right = y_values[-1]

x_values, y_values = runge_kutta_fourth_order(f, h, u0, x_values)
display_data(x_values, y_values,
             'Solving a system of differential equations using the 4th order Runge-Kutta method')

y_rk_right = y_values[-1]

print("Modulus of the difference of solutions at the rightmost point of the interval: ",
      abs(y_imt_right-y_rk_right))
```

Листинг 4.1 main.py

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


def f(x, y):
    u, v, w = y
    return np.array([u + v - w + math.sqrt(x),
                     -3 * u - 3 * v + 3 * w,
                     -2 * u - 2 * v + 2 * w + math.pow(x, 2)])


def display_data(x_values, y_values, title):
    u_values = y_values[:, 0]
    v_values = y_values[:, 1]
    w_values = y_values[:, 2]
    plt.figure(figsize=(10, 6))
    plt.plot(x_values, u_values, label='u(x)')
    plt.plot(x_values, v_values, label='v(x)')
    plt.plot(x_values, w_values, label='w(x)')
    plt.title(title)
    plt.xlabel('x')
    plt.ylabel('y')
```

```
    plt.legend()
    plt.grid(True)
    plt.show()
```

Листинг 4.2 utility.py

```
import numpy as np


u0 = np.array([2, 3, 0])
h = 0.001
x_range = [0, 1]
num_steps = int((x_range[1] - x_range[0]) / h)
x_values = np.linspace(x_range[0], x_range[1], num_steps + 1)
```

Листинг 4.3 input_data.py

```
import numpy as np


def implicit_euler(f, x_values, y0, h, num_iterations):
    x = np.zeros(num_iterations + 1)
    y = np.zeros((num_iterations + 1, len(y0)))
    y[0] = y0

    for i in range(num_iterations):
        x[i+1] = x[i] + h
        y_guess = y[i] + h * f(x[i+1], y[i])
        y[i+1] = y[i] + h * f(x[i+1], y_guess)

    return x, y
```

Листинг 4.4 implicit_euler_method.py

```
import numpy as np


def runge_kutta_fourth_order(f, h, y0, x_values):
    y_values = np.zeros((len(x_values), len(y0)))
    y_values[0] = y0
    for i in range(1, len(x_values)):
        xi = x_values[i]
        yi_prev = y_values[i - 1]
        ki0 = f(xi, yi_prev)
        ki1 = f(xi + h / 2, yi_prev + h / 2 * ki0)
        ki2 = f(xi + h / 2, yi_prev + h / 2 * ki1)
        ki3 = f(xi + h, yi_prev + h * ki2)
        yi_new = yi_prev + h / 6 * (ki0 + 2 * ki1 + 2 * ki2 + ki3)
        y_values[i] = yi_new
    return x_values, y_values
```

Листинг 4.5 runge_kutta_method_4th_order.py

ГЛАВА 5

Вывод

В результате оба метода продемонстрировали отличную точность, что связано с входными данными.

В среднем же Метод Рунге-Кутта является более точным, чем метод Эйлера, т.к. чем выше порядок, тем точнее расчет приращения Δy.