Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Лабораторная работа №5.

Поиск корней алгебраического уравнения методом секущих

По дисциплине «Software Engineering Process»

Вариант №3

Проверила: Беккулова К.А.

Выполнил студент:

Бектурсунова Айжамал

группа ПИ-1-22 (англ)

г. Бишкек 2024

**Введение**

В данном отчете представлено решение задачи Коши для дифференциального уравнения первого порядка, имеющего вид:

y′=1+ysin(x)−y2, y(0)=0.

Решение осуществляется с использованием двух численных методов: метода Эйлера и метода Рунге-Кутты (Гюна). Целью работы является получение значений y в заданных точках x=0,0.1,0.2,…,1 с шагом h=0.1

1. **Методические рекомендации**

Работа проводилась в соответствии с методическими рекомендациями, включающими следующие этапы:

1. Разработка прототипов в Excel.
2. Создание спецификации требований.
3. Определение входных и выходных данных.
4. Показ шагов решения (итерации).
5. Проектирование алгоритма.
6. Реализация алгоритма.
7. Тестирование программы.

#### Разработка прототипов в Excel

#### 

#### Спецификация требований

#### Входные данные:

#### Начальное значение y(0)=0.

#### Параметры a=0.2 и k=1.

#### Шаг h=0.1.

#### Выходные данные:

#### Значения yn​ для каждого шага n по методам Эйлера и Гюна.

#### Функция

#### Реализация алгоритма

#### import numpy as np

#### import matplotlib.pyplot as plt

#### # Параметры

#### a = 0.2 # Значение параметра a

#### k = 1 # Значение параметра k

#### h = 0.1 # Шаг

#### x\_values = np.arange(0, 1.1, h) # Массив значений x

#### y\_euler = np.zeros(len(x\_values)) # Массив для метода Эйлера

#### y\_runge\_kutta = np.zeros(len(x\_values)) # Массив для метода Рунге-Кутты

#### # Метод Эйлера

#### for n in range(len(x\_values) - 1):

#### f = 1 + a \* y\_euler[n] \* np.sin(x\_values[n]) - k \* y\_euler[n]\*\*2

#### y\_euler[n + 1] = y\_euler[n] + h \* f

#### # Метод Рунге-Кутты

#### for n in range(len(x\_values) - 1):

#### k1 = 1 + a \* y\_runge\_kutta[n] \* np.sin(x\_values[n]) - k \* y\_runge\_kutta[n]\*\*2

#### k2 = 1 + a \* (y\_runge\_kutta[n] + h \* k1) \* np.sin(x\_values[n] + h) - k \* (y\_runge\_kutta[n] + h \* k1)\*\*2

#### y\_runge\_kutta[n + 1] = y\_runge\_kutta[n] + (h / 2) \* (k1 + k2)

#### # Визуализация результатов

#### plt.plot(x\_values, y\_euler, label='Метод Эйлера', marker='o')

#### plt.plot(x\_values, y\_runge\_kutta, label='Метод Рунге-Кутты', marker='x')

#### plt.xlabel('x')

#### plt.ylabel('y')

#### plt.title('Решение задачи Коши')

#### plt.legend()

#### plt.grid()

#### plt.show()

#### Результат программы

#### 

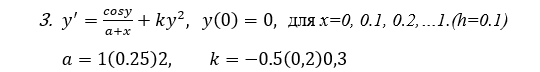
#### Диаграмма активности UML

#### 

**Отчет по выполнению задания на решение задачи Коши (свой вариант)**

**Введение**

В данном отчете описывается процесс решения задачи Коши для дифференциального уравнения первого порядка методом Эйлера и методом Рунге-Кутты. Уравнение имеет вид:

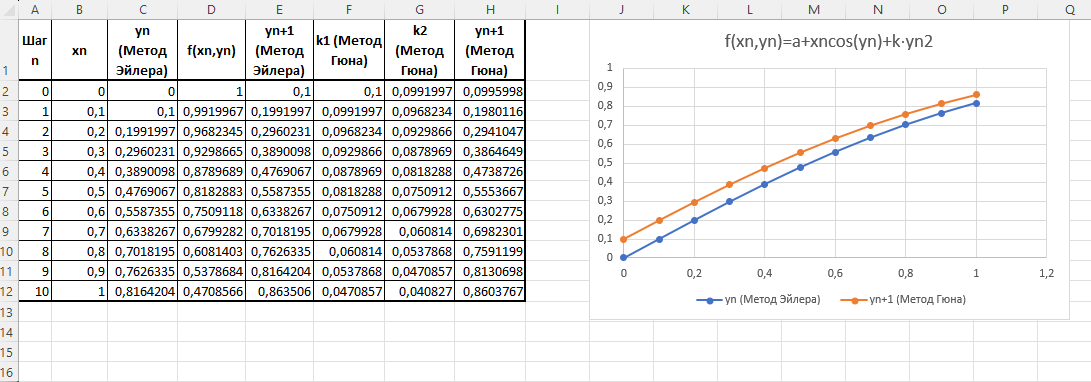


**Методические рекомендации**

Порядок выполнения работы:

1. Разработка прототипов в Excel.
2. Создание спецификации требований.
3. Определение входных и выходных данных.
4. Отображение шагов решения (итерации).
5. Проектирование алгоритма.
6. Реализация алгоритма.
7. Тестирование программы.

**Разработка прототипов в Excel**

****

**Спецификация требований**

**Входные данные:**

* **Параметр a**: от 1 до 2 с шагом 0.25
* **Параметр k**: от -0.5 до 0.3 с шагом 0.2
* **Переменная x**: от 0 до 1 с шагом h=0.1

**Выходные данные:**

* + Значения yn​ для каждого шага n по методам Эйлера и Гюна.
  + ****

**Проектирование алгоритма**

Алгоритм для метода Эйлера и метода Рунге-Кутты можно описать следующим образом:

1. Инициализация: Установите начальные значения x=0, y=0.
2. **Метод Эйлера**:

Вычисляем f(xn,yn).

Обновляем yn+1=yn+h⋅f(xn,yn)

1. **Метод Рунге-Кутты второго порядка**:

Вычисляем k1=f(xn,yn)

Вычисляем k2=f(xn+h,yn+h⋅k1)

Обновляем yn+1=yn+h2⋅(k1+k2)

1. **Цикл по значениям a и k** позволяет выполнить расчеты для всех комбинаций параметров и строит график для метода Эйлера и метода Рунге-Кутты для каждой пары значений.

**Реализация алгоритма**

import numpy as np

import matplotlib.pyplot as plt

# Параметры задачи

h = 0.1 # шаг по x

x\_values = np.arange(0, 1 + h, h) # значения x от 0 до 1 с шагом h

a\_values = np.arange(1, 2.25, 0.25) # a от 1 до 2 с шагом 0.25

k\_values = np.arange(-0.5, 0.4, 0.2) # k от -0.5 до 0.3 с шагом 0.2

# Функция для вычисления правой части уравнения y' = f(x, y)

def f(x, y, a, k):

return np.cos(y) / (a + x) + k \* y\*\*2

# Метод Эйлера

def euler\_method(a, k):

y\_euler = np.zeros(len(x\_values)) # массив для решения методом Эйлера

for i in range(len(x\_values) - 1):

f\_val = f(x\_values[i], y\_euler[i], a, k)

y\_euler[i + 1] = y\_euler[i] + h \* f\_val

return y\_euler

# Метод Рунге-Кутты второго порядка

def runge\_kutta\_method(a, k):

y\_runge\_kutta = np.zeros(len(x\_values)) # массив для решения методом Рунге-Кутты

for i in range(len(x\_values) - 1):

k1 = f(x\_values[i], y\_runge\_kutta[i], a, k)

k2 = f(x\_values[i] + h, y\_runge\_kutta[i] + h \* k1, a, k)

y\_runge\_kutta[i + 1] = y\_runge\_kutta[i] + (h / 2) \* (k1 + k2)

return y\_runge\_kutta

# Построение решений для различных значений a и k

for a in a\_values:

for k in k\_values:

y\_euler = euler\_method(a, k)

y\_runge\_kutta = runge\_kutta\_method(a, k)

# Визуализация результатов

plt.plot(x\_values, y\_euler, label=f'Эйлер, a={a}, k={k}', linestyle='--')

plt.plot(x\_values, y\_runge\_kutta, label=f'Рунге-Кутта, a={a}, k={k}', linestyle='-')

plt.xlabel('x')

plt.ylabel('y')

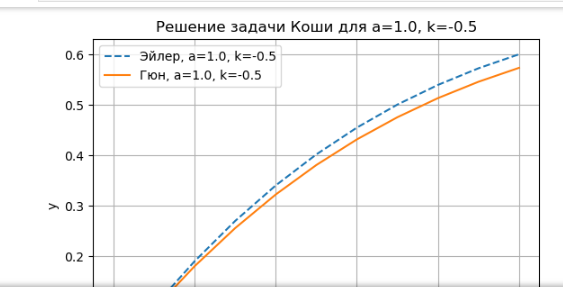
plt.title(f'Решение задачи Коши для a={a}, k={k}')

plt.legend()

plt.grid()

plt.show()

**Результат программы:**



**Диаграмма активности UML**

