ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра прикладной
математики и кибернетики

КУРСОВАЯ РАБОТА

по дисциплине «Вычислительная математика»

Выполнил:
студент группы ИП-216
Русецкий Артём Сергеевич

Проверил:
Чупрыно Л.А.

Новосибирск, 2024

Оглавление

Постановка задачи

Решить дифференциальное уравнение на интервале [0,1] методами Рунге Кутта 2 и 4 порядка с точностью 10^(-6) стартовый шаг h=0,1.

y''=(e^x+y+y')/3

y(0) = 1

y(1) = 2,718

Проинтерполировать найденное решение методом Ньютона или кубического сплайна по узлам интерполяции 0.0, 0.2, 0.4, 0.6, 0.8, 1.0.

Программная реализация

def f(x, y, dy)

Функция исходного уравнения.

def runge_kutta_2nd_order(f, x0, y0, dy0, h, x_end)

Функция, реализующая метод Рунге – Кутта 2-го порядка.

def runge_kutta_4th_order(f, x0, y0, dy0, h, x_end)

Функция, реализующая метод Рунге – Кутта 4-го порядка.

def newton_interpolation(xs, ys, x)

Функция, реализующая интерполяцию методом Ньютона.

def shooting_method(f, x0, y0, x_end, y_end, h, tol=1e-6, method='RK4')

Функция, реализующая уточнение методом стрельб.

Результат работы

```
Решение дифференциального уравнения (RK2):
x = 0.000000, y = 1.000000
x = 0.100000, y = 1.105000
x = 0.200000, y = 1.221034
x = 0.300000, y = 1.349262
x = 0.400000, y = 1.490964
x = 0.500000, y = 1.647558
x = 0.600000, y = 1.820608
x = 0.700000, y = 2.011844
x = 0.800000, y = 2.223176
x = 0.900000, y = 2.456718
x = 1.000000, y = 2.714803
x = 1.000000, y = 2.714803
```

```
Решение дифференциального уравнения (RK4):
x = 0.000000, y = 1.000000
x = 0.100000, y = 1.105171
x = 0.200000, y = 1.221403
x = 0.300000, y = 1.349858
x = 0.400000, y = 1.491824
x = 0.500000, y = 1.648721
x = 0.600000, y = 1.822118
x = 0.700000, y = 2.013752
x = 0.800000, y = 2.225540
x = 0.900000, y = 2.459602
x = 1.000000, y = 2.718280
x = 1.000000, y = 2.718280

Значение функции в точке x = 0.44 по методу Ньютона: y = 1.552704
```

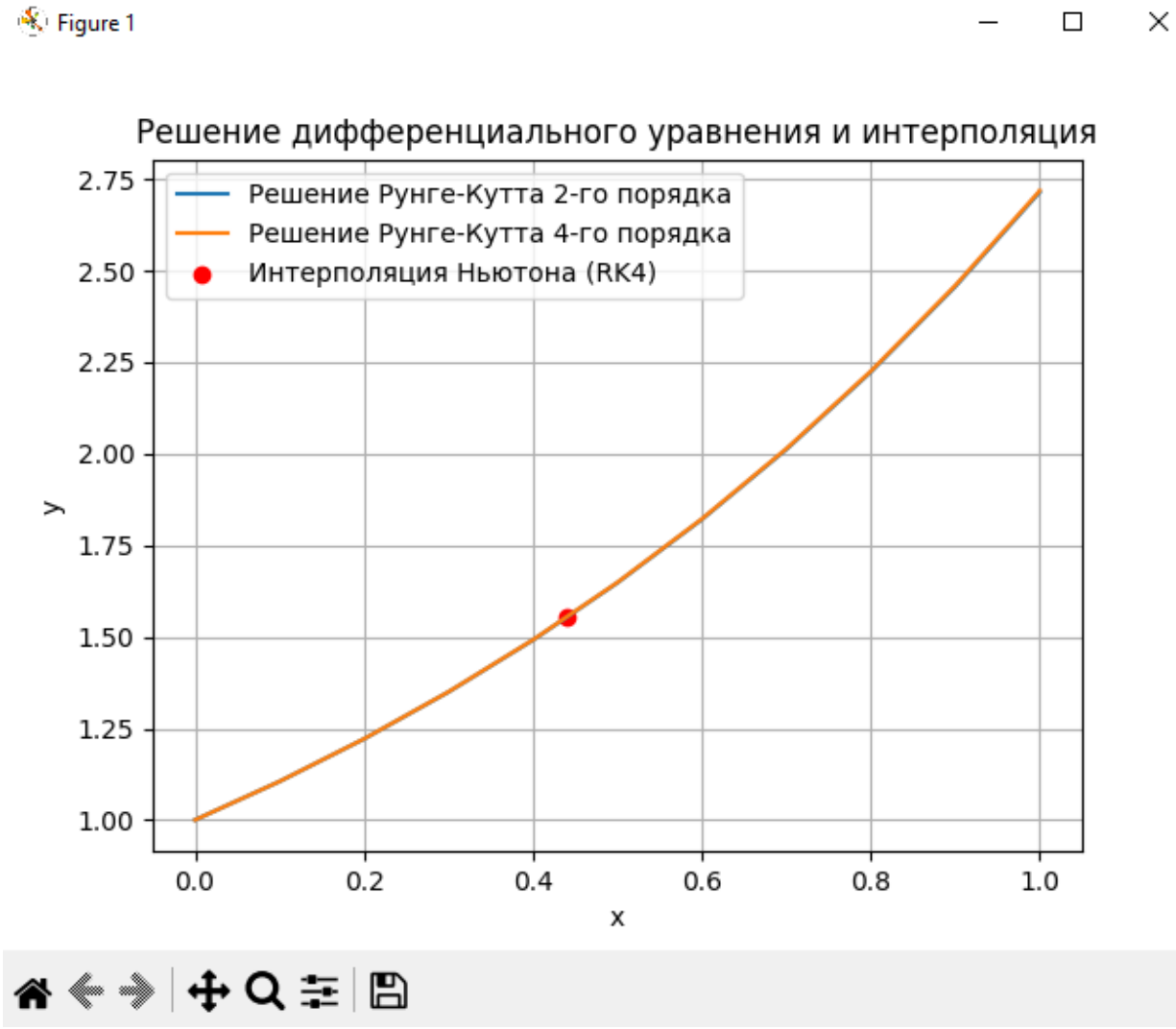
Figure 1
Решение дифференциального уравнения и интерполяция
Решение Рунге-Кутта 2-го порядка
Решение Рунге-Кутта 4-го порядка
Интерполяция Ньютона (RK4)
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
y
0.0
0.2
0.4
0.6
0.8
1.0
x

Выводы

В рамках курсовой работы была реализована программа, которая решает ДУ методами Рунге Кутта 2 и 4 порядка, а также позволяет интерполировать найденное решение методом Ньютона.

Список использованной литературы

1. ЧИСЛЕННЫЕ МЕТОДЫ [Учебное пособие]. Барон Л. А.
2. https://numpy.org/
3. https://matplotlib.org/stable/index.html
4. https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%81%D1%82%D1%80%D0%B5%D0%BB%D1%8C%D0%B1%D1%8B

Листинг

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

#y"=(e^x+y+y')/3
#y(0) = 1
#y(1) = 2,718

# Определение правой части уравнения
def f(x, y, dy):
    return (np.exp(x) + y + dy) / 3


# Метод Рунге-Кутта 2-го порядка
def runge_kutta_2nd_order(f, x0, y0, dy0, h, x_end):
    xs = [x0]
    ys = [y0]
    dys = [dy0]
    x, y, dy = x0, y0, dy0

    while x < x_end:
        if x + h > x_end:
            h = x_end - x

        k1 = h * dy
        l1 = h * f(x, y, dy)
        k2 = h * (dy + l1)
        l2 = h * f(x + h, y + k1, dy + l1)

        y += (k1 + k2) / 2
        dy += (l1 + l2) / 2
        x += h

        xs.append(x)
        ys.append(y)
        dys.append(dy)

    return np.array(xs), np.array(ys), np.array(dys)


# Метод Рунге-Кутта 4-го порядка
def runge_kutta_4th_order(f, x0, y0, dy0, h, x_end):
    xs = [x0]
    ys = [y0]
    dys = [dy0]
    x, y, dy = x0, y0, dy0

    while x < x_end:
        if x + h > x_end:
            h = x_end - x

        k1 = h * dy
        l1 = h * f(x, y, dy)
        k2 = h * (dy + 0.5 * l1)
        l2 = h * f(x + 0.5 * h, y + 0.5 * k1, dy + 0.5 * l1)
        k3 = h * (dy + 0.5 * l2)
        l3 = h * f(x + 0.5 * h, y + 0.5 * k2, dy + 0.5 * l2)
        k4 = h * (dy + l3)
        l4 = h * f(x + h, y + k3, dy + l3)

        y += (k1 + 2 * k2 + 2 * k3 + k4) / 6
        dy += (l1 + 2 * l2 + 2 * l3 + l4) / 6
        x += h
```

```
        xs.append(x)
        ys.append(y)
        dys.append(dy)

    return np.array(xs), np.array(ys), np.array(dys)


# Метод Ньютона для интерполяции
def newton_interpolation(xs, ys, x):
    n = len(xs)
    divided_differences = np.zeros((n, n))
    divided_differences[:, 0] = ys

    for j in range(1, n):
        for i in range(n - j):
            divided_differences[i, j] = (divided_differences[i+1, j-1] -
divided_differences[i, j-1]) / (xs[i+j] - xs[i])

    interpolated_value = divided_differences[0, 0]
    product_term = 1.0

    for j in range(1, n):
        product_term *= (x - xs[j-1])
        interpolated_value += divided_differences[0, j] * product_term

    return interpolated_value


# Граничные условия
x0, y0 = 0, 1
x_end, y_end = 1, np.e


# Метод стрельбы для уточнения
def shooting_method(f, x0, y0, x_end, y_end, h, tol=1e-6, method='RK4'):
    dy0_low, dy0_high = 0, 1  # Начальное приближение

    while abs(dy0_high - dy0_low) > tol:
        dy0_mid = (dy0_low + dy0_high) / 2
        if method == 'RK2':
            _, ys_low, _ = runge_kutta_2nd_order(f, x0, y0, dy0_low, h,
x_end)
            _, ys_mid, _ = runge_kutta_2nd_order(f, x0, y0, dy0_mid, h,
x_end)
        else:
            _, ys_low, _ = runge_kutta_4th_order(f, x0, y0, dy0_low, h,
x_end)
            _, ys_mid, _ = runge_kutta_4th_order(f, x0, y0, dy0_mid, h,
x_end)

        if (ys_low[-1] - y_end) * (ys_mid[-1] - y_end) < 0:
            dy0_high = dy0_mid
        else:
            dy0_low = dy0_mid

    dy0_final = (dy0_low + dy0_high) / 2
    if method == 'RK2':
        xs, ys, dys = runge_kutta_2nd_order(f, x0, y0, dy0_final, h, x_end)
    else:
        xs, ys, dys = runge_kutta_4th_order(f, x0, y0, dy0_final, h, x_end)
    return xs, ys, dys
```

```
# Начальное значение шага
h = 0.1

# Решение задачи методом стрельбы для Рунге-Кутта 2-го порядка
xs_rk2, ys_rk2, dys_rk2 = shooting_method(f, x0, y0, x_end, y_end, h,
method='RK2')

# Решение задачи методом стрельбы для Рунге-Кутта 4-го порядка
xs_rk4, ys_rk4, dys_rk4 = shooting_method(f, x0, y0, x_end, y_end, h,
method='RK4')



# Вывод результатов
print("Решение дифференциального уравнения (RK2):")
for x, y in zip(xs_rk2, ys_rk2):
    print(f"x = {x:.6f}, y = {y:.6f}")

print("\nРешение дифференциального уравнения (RK4):")
for x, y in zip(xs_rk4, ys_rk4):
    print(f"x = {x:.6f}, y = {y:.6f}")

interpolation_nodes = np.arange(0, 1, 0.2)
interpolation_values = [newton_interpolation(xs_rk4, ys_rk4, x) for x in
interpolation_nodes]
# Узлы интерполяции
x_target = 0.44
y_interpolated = newton_interpolation(interpolation_nodes,
interpolation_values, x_target)
print(f"\nЗначение функции в точке x = {x_target:.2f} по методу Ньютона: y =
{y_interpolated:.6f}")

# Построение графиков
plt.plot(xs_rk2, ys_rk2, label='Решение Рунге-Кутта 2-го порядка')
plt.plot(xs_rk4, ys_rk4, label='Решение Рунге-Кутта 4-го порядка')
plt.scatter(x_target, y_interpolated, color='red', label='Интерполяция
Ньютона (RK4)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Решение дифференциального уравнения и интерполяция')
plt.grid(True)
plt.show()
```