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 2](#_Toc167399726)

[Программная реализация 3](#_Toc167399727)

[Результат работы 4](#_Toc167399728)

[Выводы 6](#_Toc167399729)

[Список использованной литературы 6](#_Toc167399730)

[Листинг 7](#_Toc167399731)

# **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.

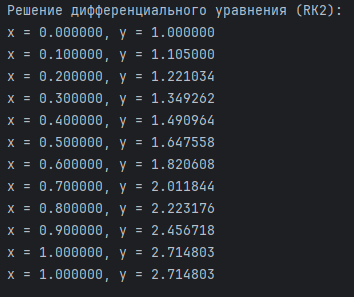
def newton\_interpolation(xs, ys, x)

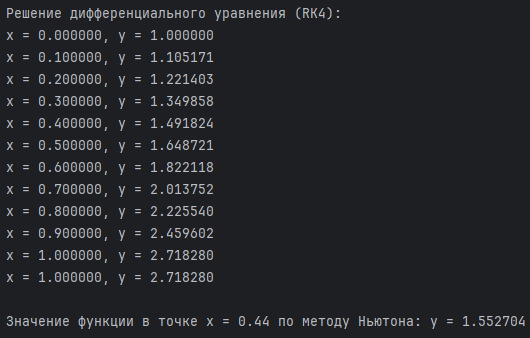
Функция, реализующая интерполяцию методом Ньютона.

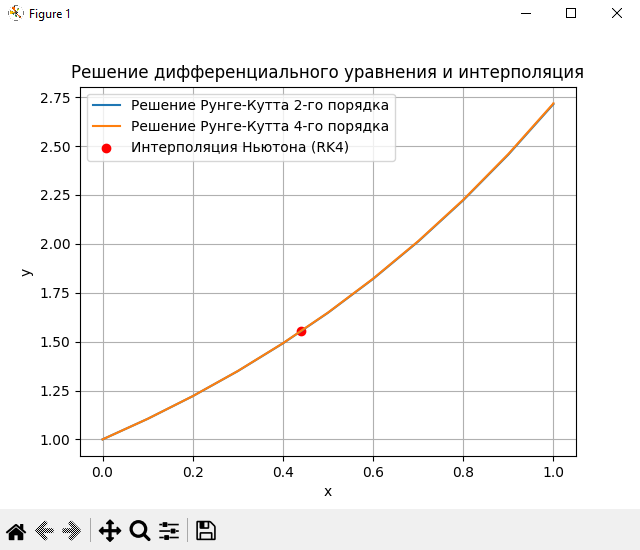
def shooting\_method(f, x0, y0, x\_end, y\_end, h, tol=1e-6, method='RK4')

Функция, реализующая уточнение методом стрельб.

# **Результат работы**







# **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()