**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**МЕТОДЫ ЧИСЛЕННОГО АНАЛИЗА**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

***“Решение задачи Коши для ОДУ-II явным методом трапеций и методом Рунге-Кутта 4-го порядка”***

**ВОЛКОВ ЕВГЕНИЙ, 3 КУРС, 9 ГРУППА**

**ПРЕПОДАВАТЕЛЬ: ПОЛЕВИКОВ В.К.**

**ПОСТАНОВКА ЗАДАЧИ**

Решить задачу Коши для ОДУ-II явным методом трапеций и методом Рунге-Кутта 4 порядка для заданных *a, b, , , f(•):*

**ИСХОДНЫЕ ДАННЫЕ**

Точное решение получим следующую задачу:

**КРАТКАЯ ТЕОРИЯ**

Для сведения исходного ДУ 2-го порядка к системе ДУ 1-го порядка выполним следующую замену:

Рассмотрим задачу Коши **(\*)**

**Явный метод трапеций.**

Проинтегрируем исходное ДУ **(\*)** на отрезке и заменим интеграл по КФ трапеций:

.

Равенство будет достигаться на некоторой близкой функции :

.

Это неявная формула относительно , поэтому с помощью ЯМЭ найдём . Окончательно запишем:

Запишем ЯМТ для нашей системы ДУ 1-го порядка, где

:

**Метод Рунге-Кутта 4-го порядка (q=3).**

Проинтегрируем исходное ДУ **(\*)** на отрезке и заменим интеграл линейной комбинацией:

, где

*……*

При q=3 получаем следующие коэффициенты:

Тогда метод Р.-К. 4-го порядка имеет вид:

Где

;

Для нашей системы метод имеет вид:

Где

**КОД ПРОГРАММЫ**

import math

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

a = 0

b = math.pi/2

u01 = 0

u02 = 1

h = 0.01

N = int((b-a)/h)

h = (b-a)/N

def f2(\_x, \_y1, \_y2):

  return (\_y2 - \_y1 + (\_x-2)\*math.sin(\_x) - math.cos(\_x))

def trapezium():

  y1 = []

  y2 = []

  x = []

  xi = a

  x.append(xi)

  y\_1i = u01

  y\_2i = u02

  y1.append(y\_1i)

  y2.append(y\_2i)

  for i in range(1,N+1):

    \_y\_2iplus1\_ = y\_2i + h\*f2(xi, y\_1i, y\_2i) ## Y\_2,i+1 с чертой

    y\_1iplus1 = y\_1i + h/2\*(y\_2i +  \_y\_2iplus1\_) ## Y\_1,i+1

    y1.append(y\_1iplus1)

    \_y\_1iplus1\_ = y\_1i + h\*y\_2i ## Y\_1,i+1 с чертой

    y\_2iplus1 = y\_2i + h/2\*(f2(xi, y\_1i, y\_2i)

+f2(xi+h, \_y\_1iplus1\_, \_y\_2iplus1\_))  ## Y\_2,i+1

    y2.append(y\_2iplus1)

    y\_1i = y\_1iplus1

    y\_2i = y\_2iplus1

    xi += h

    x.append(xi)

  return x,y1

def runge\_kutta():

  y1 = []

  y2 = []

  xi = a

  y\_1i = u01

  y\_2i = u02

  y1.append(y\_1i)

  y2.append(y\_2i)

  for i in range(1,N+1):

    fi01 = y\_2i

    fi02 = f2(xi, y\_1i, y\_2i)

    fi11 = y\_2i + h/2\*fi02

    fi12 = f2(xi+h/2, y\_1i + fi01\*h/2, y\_2i + fi02\*h/2)

    fi21 = y\_2i + fi12\*h/2

    fi22 = f2(xi+h/2, y\_1i + fi11\*h/2, y\_2i + fi12\*h/2)

    fi31 = y\_2i + h\*fi22

    fi32 = f2(xi+h, y\_1i + fi21\*h, y\_2i + fi22\*h)

    y\_1iplus1 = y\_1i + (fi01 + 2\*fi11 + 2\*fi21 + fi31)\*h/6

    y\_2iplus1 = y\_2i + (fi02 + 2\*fi12 + 2\*fi22 + fi32)\*h/6

    y1.append(y\_1iplus1)

    y2.append(y\_2iplus1)

    y\_1i = y\_1iplus1

    y\_2i = y\_2iplus1

    xi+=h

  return y1

x, y = trapezium()

y\_rk = runge\_kutta()

u = []

for i in x:

  u.append(i\*math.cos(i))

f = plt.figure()

f.set\_figwidth(11)

f.set\_figheight(6.5)

plt.plot(x, y, color='black')

plt.plot(x, y\_rk, color = 'blue')

plt.plot(x, u, color='red')

plt.show()

df = pd.DataFrame(u)

df[1] = y

df[2] = y\_rk

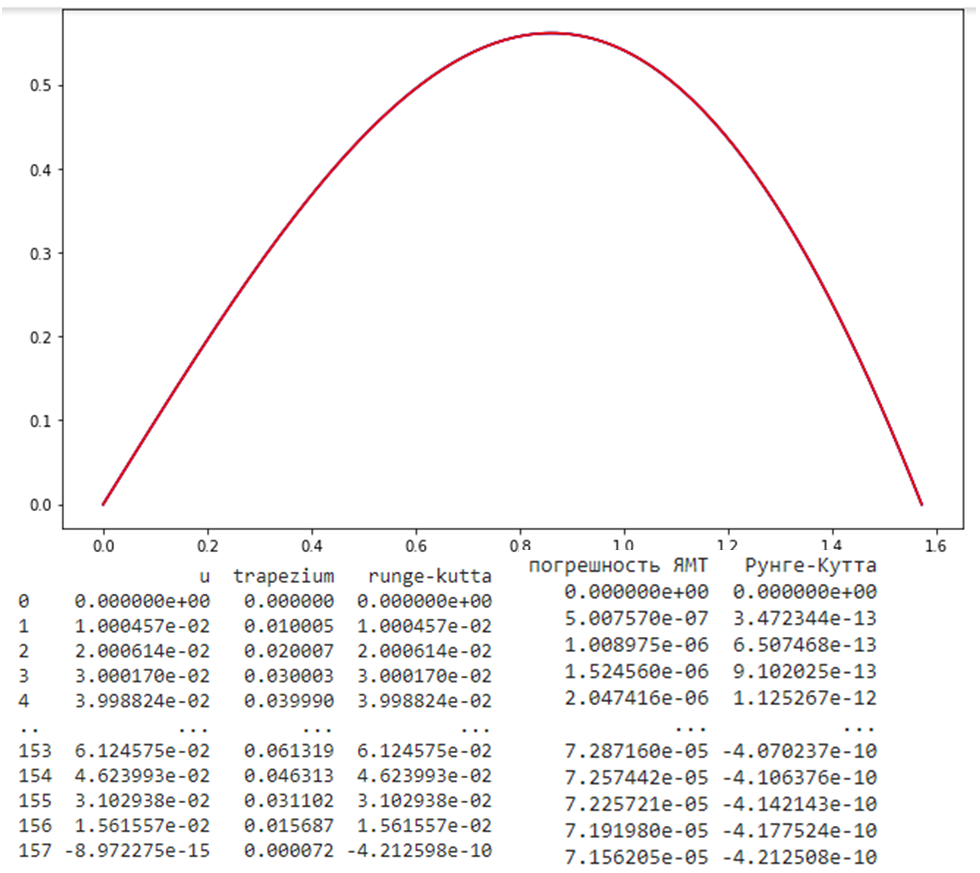
df = df.rename(columns={0:'u', 1:'trapezium', 2:'runge-kutta'})

print(df)

**РЕЗУЛЬТАТЫ**

Изобразим три графика: точное решение и приближённые решения, полученные с помощью явного метода трапеций и метода Рунге-Кутта 4 порядка.

Для сравнения приведём также значения этих функций в узлах сетки. Т.к. количество узлов N=157, то ограничимся только первыми и последними 5 узлами.

****