Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы численного анализа

**ОТЧЁТ**

к лабораторной работе

на тему

Методы Эйлера и Рунге-Кутта

Выполнил: студент группы 153503

Киселёва Елизавета Андреевна

Проверил: Анисимов Владимир Яковлевич

Минск 2022

**Оглавление**

Цели выполнения задания 3

Краткие теоретические сведения 4

Задание 7

Алгоритм задания 8

Программная реализация 9

Результат выполнения программы 11

Тестовые примеры 13

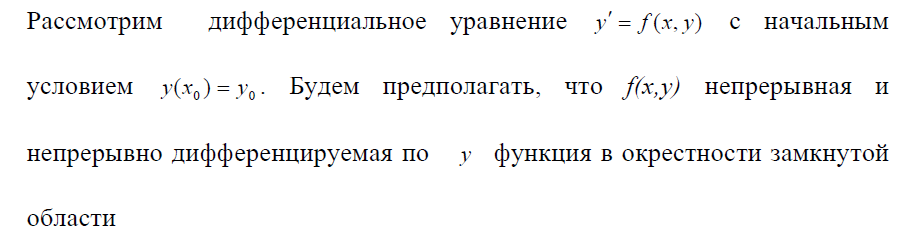
Выводы 13

Вариант 9

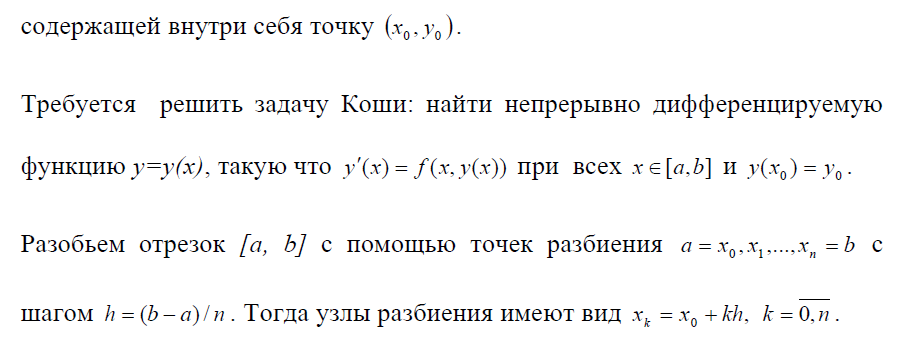
# **Цели выполнения задания**

* Изучить решение задачи Коши для обыкновенных дифференциальных уравнений методом Эйлера;
* Изучить решение задачи Коши для обыкновенных дифференциальных уравнений методом Рунге-Кутта;
* С помощью метода Эйлера найти решения уравнения;
* С помощью метода Рунге-Кутта найти решения уравнения;
* Сравнить полученные результаты;
* Сделать выводы на основе выполненного задания.

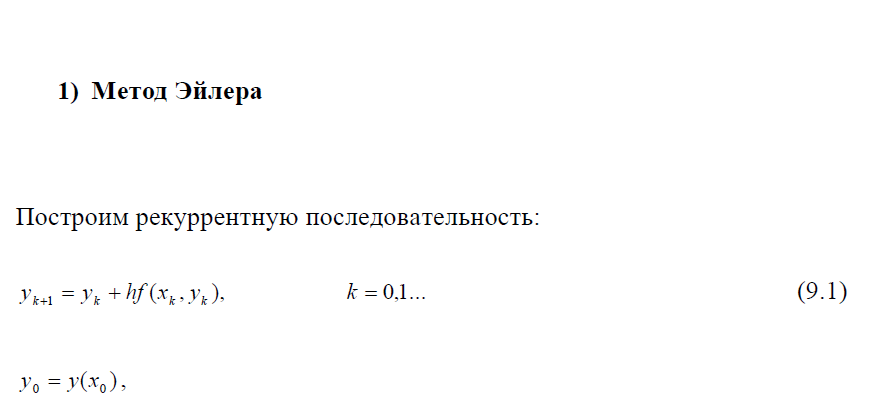
# **Краткие теоретические сведения**

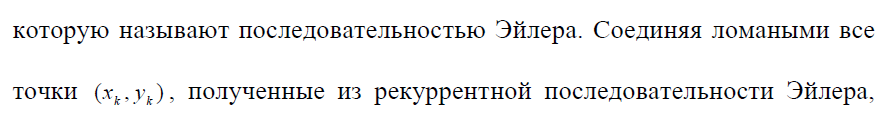


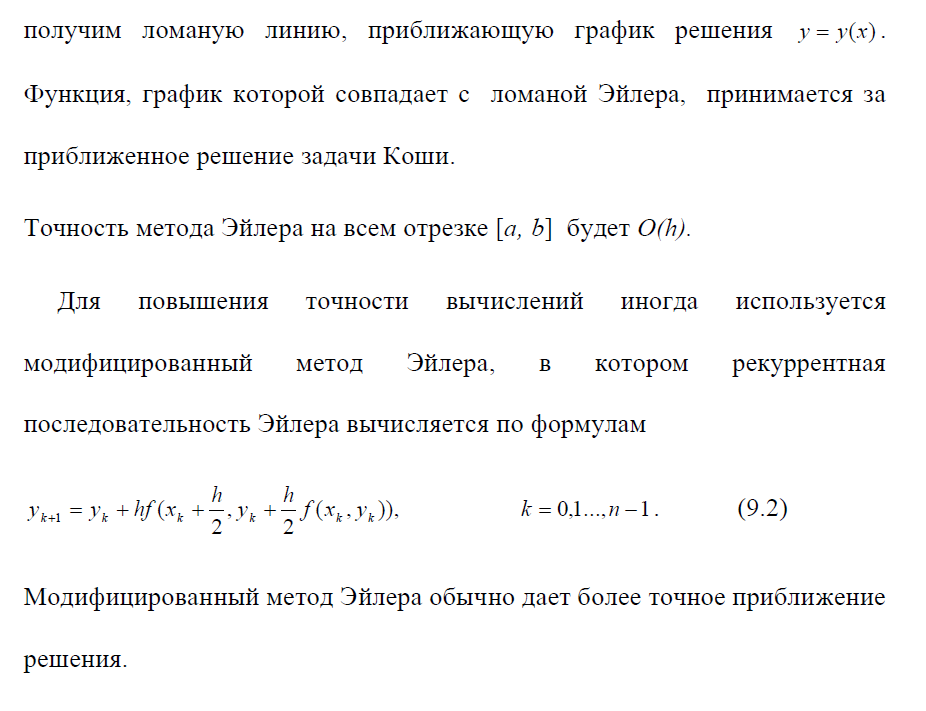


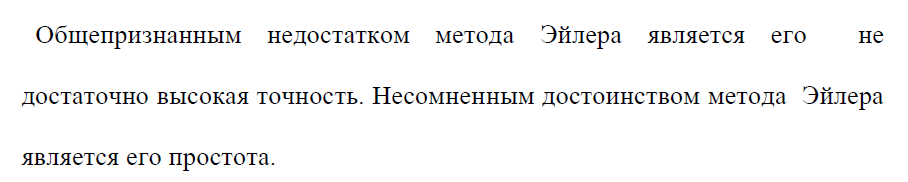


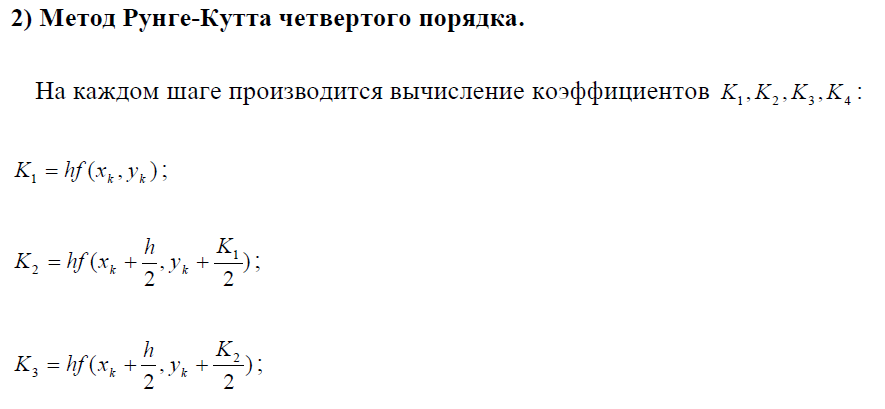


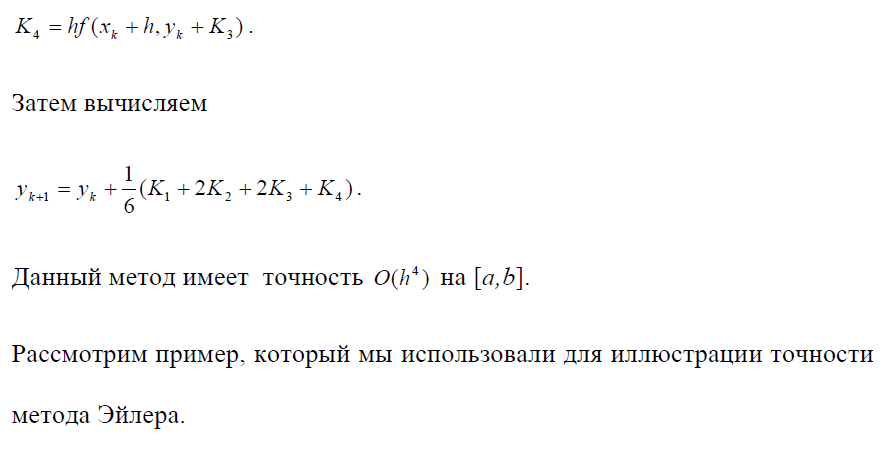


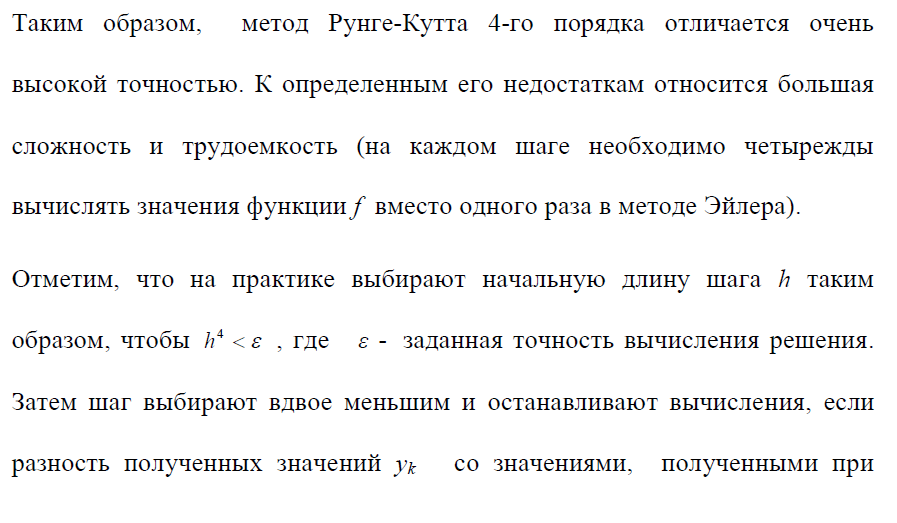


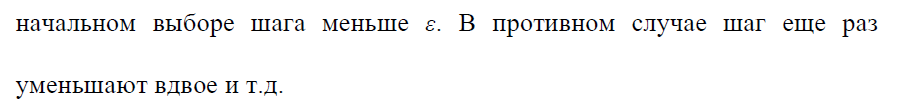






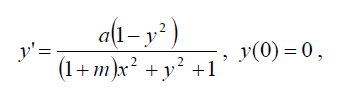






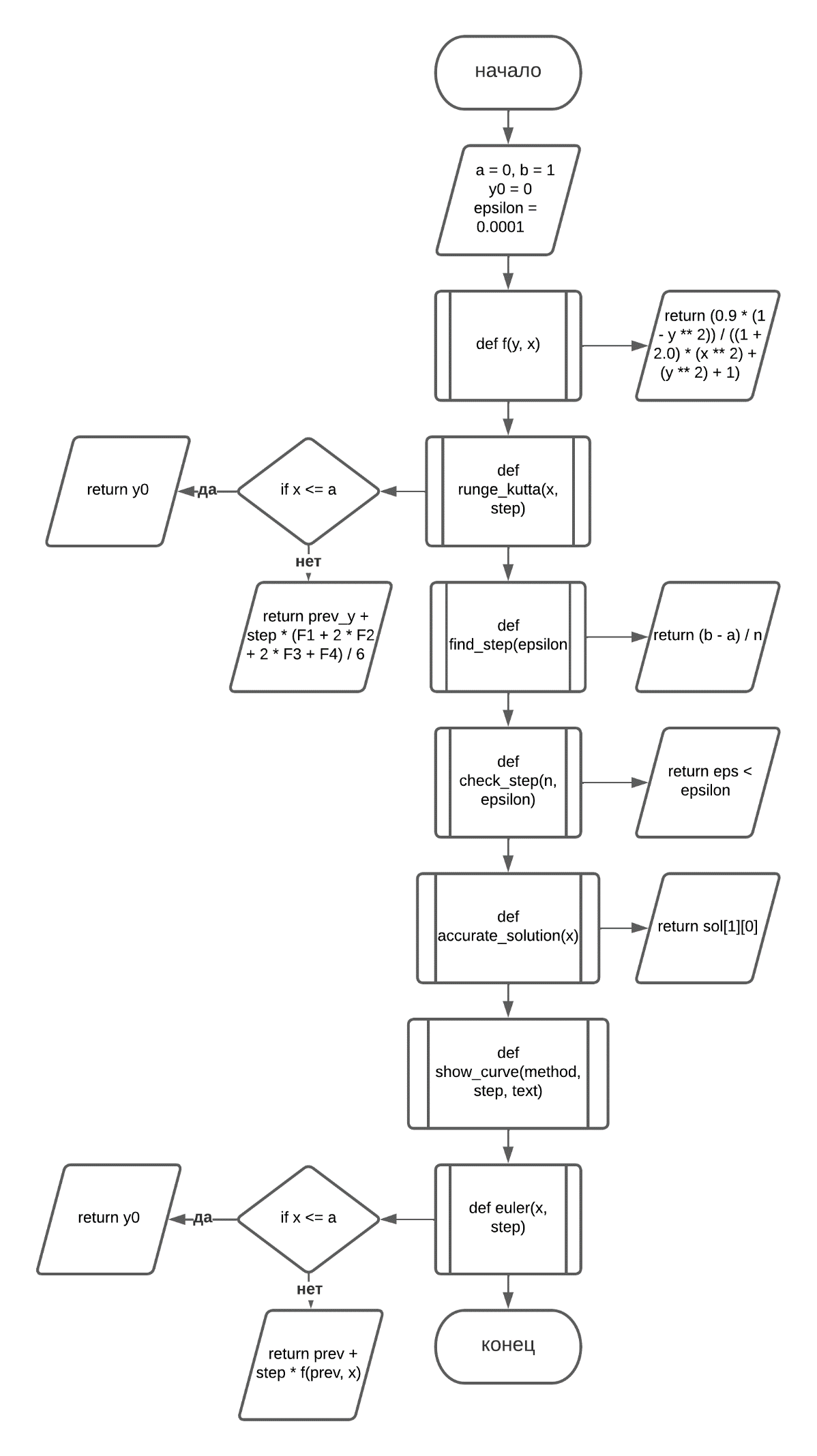
**Задание**

С помощью метода Эйлера, а затем метода Рунге-Кутта найти решения следующих уравнений на отрезке [0; 1].



где а = 1.1, m = 2.0.

Сравнить полученные результаты.

**Алгоритм задания**

**Программная реализация**

import math  
from scipy.integrate import odeint  
import numpy  
from matplotlib import pylab  
  
a = 0  
b = 1  
y0 = 0  
epsilon = 0.0001  
  
  
def f(y, x):  
 return (1.1 \* (1 - y \*\* 2)) / ((1 + 2) \* (x \*\* 2) + (y \*\* 2) + 1)  
  
  
def runge\_kutta(x, step):  
 if x <= a:  
 return y0  
 prev\_x = x - step  
 prev\_y = runge\_kutta(prev\_x, step)  
 F1 = f(prev\_y, prev\_x)  
 F2 = f(prev\_y + (step / 2) \* F1, prev\_x + step / 2)  
 F3 = f(prev\_y + (step / 2) \* F2, prev\_x + step / 2)  
 F4 = f(prev\_y + step \* F3, x)  
 return prev\_y + step \* (F1 + 2 \* F2 + 2 \* F3 + F4) / 6  
  
  
def find\_step(epsilon):  
 h0 = epsilon \*\* (1 / 4)  
 n = int((b - a) // h0)  
 if n % 2 != 0:  
 n += 1  
 while check\_step(n, epsilon):  
 n = n // 4 \* 2  
 while not check\_step(n, epsilon):  
 n += 2  
 return (b - a) / n  
  
  
def check\_step(n, epsilon):  
 h = (b - a) / n  
 y2 = runge\_kutta(a + 2 \* h, h)  
 y2e = runge\_kutta(a + 2 \* h, h \* 2)  
 eps = (1 / 15) \* abs(y2 - y2e)  
 return eps < epsilon  
  
  
step = find\_step(epsilon)  
print("Шаг интегрирования: ", step)  
  
  
def accurate\_solution(x):  
 sol = odeint(f, y0, [a, x])  
 return sol[1][0]  
  
  
def show\_curve(method, step, text):  
 pylab.cla()  
 xlist = numpy.arange(a, b + step, step)  
 ylist = []  
 x = a  
 while x <= b:  
 r = method(x, step)  
 ylist.append(r)  
 x += step  
 pylab.plot(xlist, ylist, label=text)  
 print(f"Методом Рунге-Кутта: {ylist}")  
 pylab.grid(True)  
 pylab.legend()  
 pylab.savefig("1.png")  
 ylist2 = ylist  
 ylist = []  
 x = a  
 while x <= b:  
 r = accurate\_solution(x)  
 ylist.append(r)  
 x += step  
 pylab.cla()  
 pylab.plot(xlist, ylist, label="Точное решение")  
 print(f"Точное решение: {ylist}")  
 pylab.grid(True)  
 pylab.legend()  
 pylab.savefig("2.png")  
  
""" for item in range(len(ylist)):  
 rez.append(abs(ylist[item]-ylist2[item]))  
 print(f"Погрешность между зачениями: {rez}")"""  
  
show\_curve(runge\_kutta, step, "кривая методом Рунге-Кутта")  
  
def euler(x, step):  
 if x <= a:  
 return y0  
 prev = euler(x - step, step)  
 return prev + step \* f(prev, x)  
  
  
xlist = numpy.arange(a, b + step, step)  
runge\_kutta\_points = []  
euler\_points = []  
accurate\_solution\_points = []  
x = a  
while x <= b:  
 r = runge\_kutta(x, step)  
 r1 = euler(x, step)  
 r2 = accurate\_solution(x)  
 runge\_kutta\_points.append(r)  
 euler\_points.append(r1)  
 accurate\_solution\_points.append(r2)  
 x += step  
print(f"Методом Эйлера: {euler\_points}")  
"""rez = []  
for item in range(len(euler\_points)):  
 rez.append(abs(euler\_points[item]-runge\_kutta\_points[item]))"""  
  
pylab.cla()  
pylab.plot(xlist, runge\_kutta\_points, label="кривая методом Рунге-Кутта ", color=(0, 0, 1))  
pylab.plot(xlist, euler\_points, label="кривая методом Эйлера", color=(1, 0, 0))  
pylab.plot(xlist, accurate\_solution\_points, label="точное решение", color=(0, 1, 0))  
pylab.grid(True)  
pylab.legend()  
pylab.savefig("combine.png")

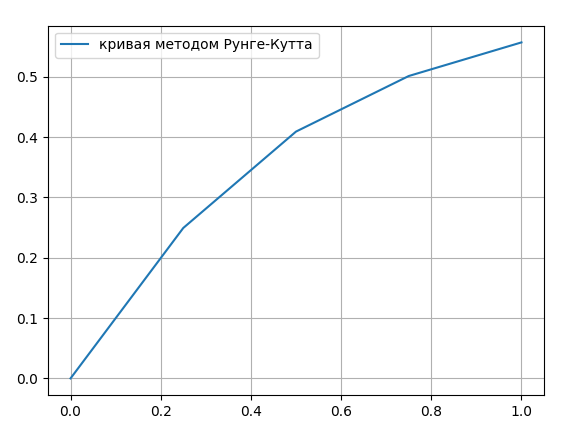
**Результат выполнения программы**

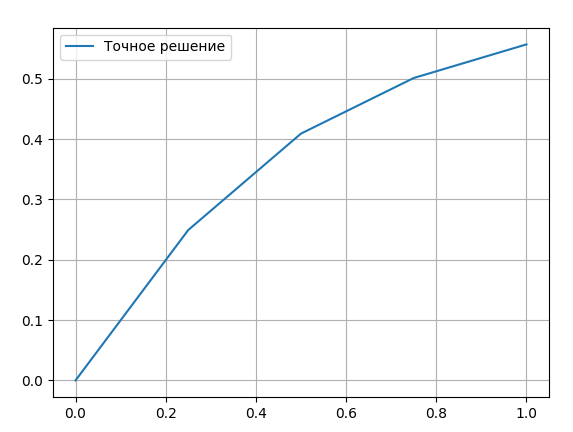
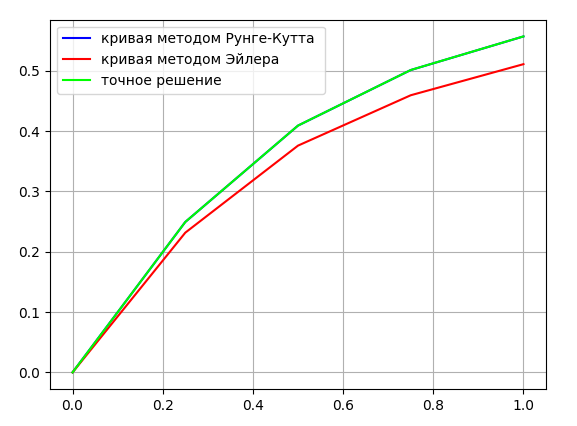
Шаг интегрирования: 0.25

Методом Рунге-Кутта: [0, 0.24930371583841257, 0.4090397815889765, 0.5011441472477581, 0.5568598076716768]

Точное решение: [0.0, 0.24929688306060954, 0.40909125563612636, 0.5011965333743809, 0.5569075211410992]

Методом Эйлера: [0, 0.23157894736842108, 0.3758725383131155, 0.4593530284289772, 0.5108783907003314]

****

****

**Тестовые примеры**

|  |  |  |
| --- | --- | --- |
| a = 0.5, m = 1.0 | a = 0.7, m = 1.5 | a = 0.9, m = 2.0 |
| Методом Рунге-Кутта: [0, 0.11906316442208519, 0.2117663033511144, 0.2758683376187522, 0.31957892301345225]  Точное решение: [0.0, 0.11905395673132853, 0.21176715767388754, 0.275873142038548, 0.319584136979079]  Методом Эйлера: [0, 0.1111111111111111, 0.19274376417233557, 0.24840882965881186, 0.28671642529486013] | Методом Рунге-Кутта: [0, 0.16381361541829662, 0.28227025965041136, 0.35818390829103985, 0.4072910913563986]  Точное решение: [0.0, 0.16379851395180434, 0.28227942216582835, 0.3581980382031534, 0.4073048651188567]  Методом Эйлера: [0, 0.15135135135135133, 0.2551140093477476, 0.3213173242981984, 0.3648703337957529] | Методом Рунге-Кутта: [0, 0.2066057936587693, 0.3451799293821097, 0.42811381924965936, 0.47947534636069117]  Точное решение: [0.0, 0.20658873449503273, 0.3452062570859286, 0.428144095847743, 0.47950360549703114]  Методом Эйлера: [0, 0.18947368421052632, 0.3109375982183748, 0.383938031876859, 0.4301917614728529] |

**Выводы**

В ходе выполнения лабораторной работы были изучены методы Эйлера и Рунге-Кутта для решения задачи Коши обыкновенных дифференциальных уравнений. Также было наглядно продемонстрировано, что, не смотря на простоту метода Эйлера, он является не точным и чем больше было взято значений, тем сильнее расходились кривые в сравнении с методом Рунге-Кутта, который оказался более точным и довольно близок к точному решению. Также при уменьшении шага в 2 раза наблюдается уменьшении разницы между методами в 2 раза (приблизительно), что означает линейную зависимость.