**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по дисциплине: «Численные методы математической физики**»**

на тему: «Разработка программ по методам решения обыкновенных дифференциальных уравнений»

Выполнил: студент гр. ИТП-22

Расшивалов Н.И.  
 Принял: доцент

Стародубцев Е.Г.

Гомель 2021

**Цель работы:** научиться разрабатывать алгоритмы численных методов и программное обеспечение для численного решения обыкновенных дифференциальных уравнений (ОДУ).

**ЗАДАНИЕ**

1) Разработать алгоритмы и написать программы, реализующие следующие методы численного решения ОДУ:

а) метод Эйлера;

б) модифицированный метод Эйлера;

в) метод Рунге-Кутты 4-го порядка.

Значение шага сетки задать самостоятельно, обеспечив хорошую точность решений.

2) Оценить погрешности решений каждым из методов, сравнив с точным аналитическим решением ОДУ.

3) Оценить погрешности решений ОДУ каждым из методов по правилу Рунге, выбрав значение шага сетки, уменьшенное в 2 раза по сравнению с исходным шагом.

**Вариант 30.**

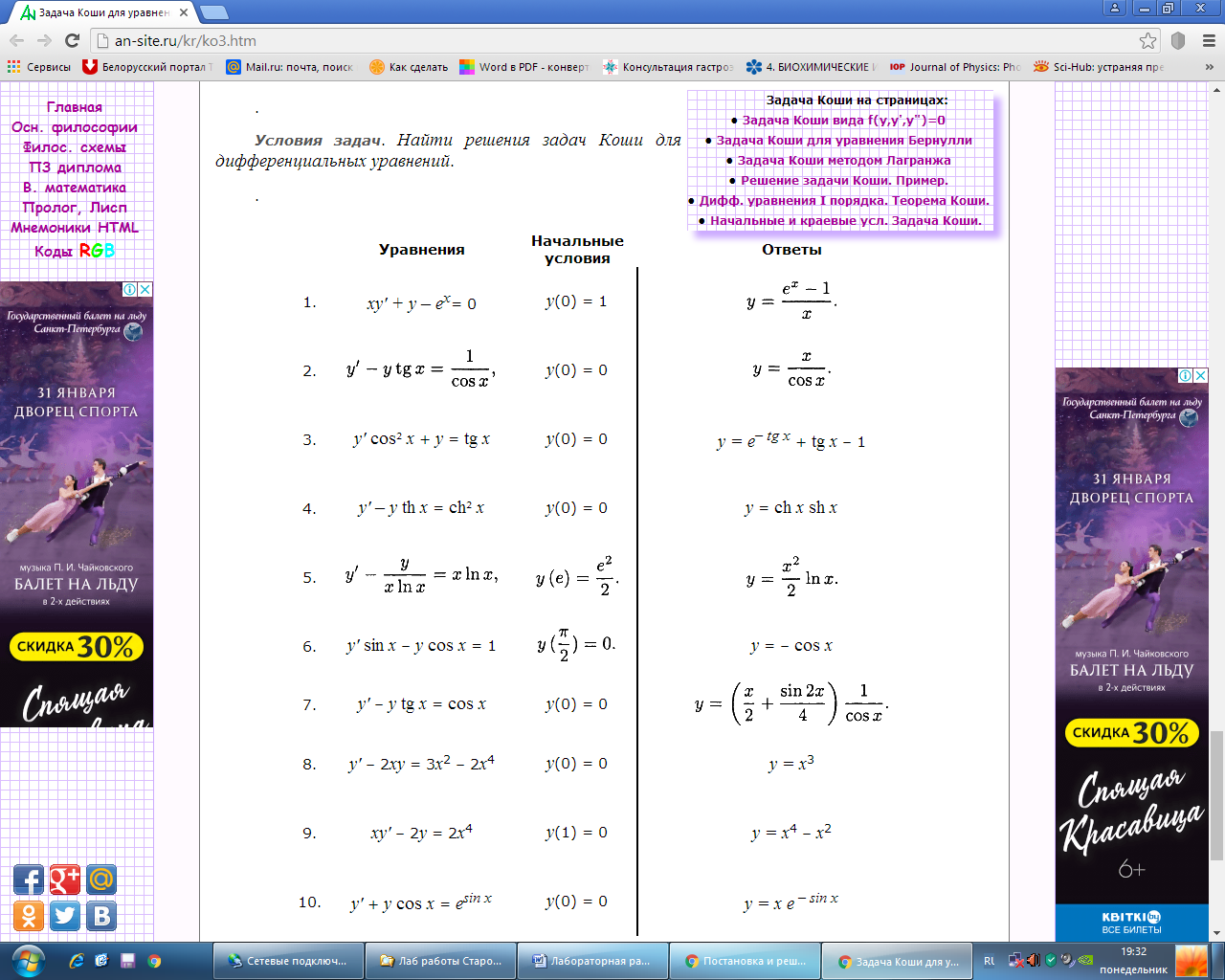


Рисунок 1 – Условие по варианту

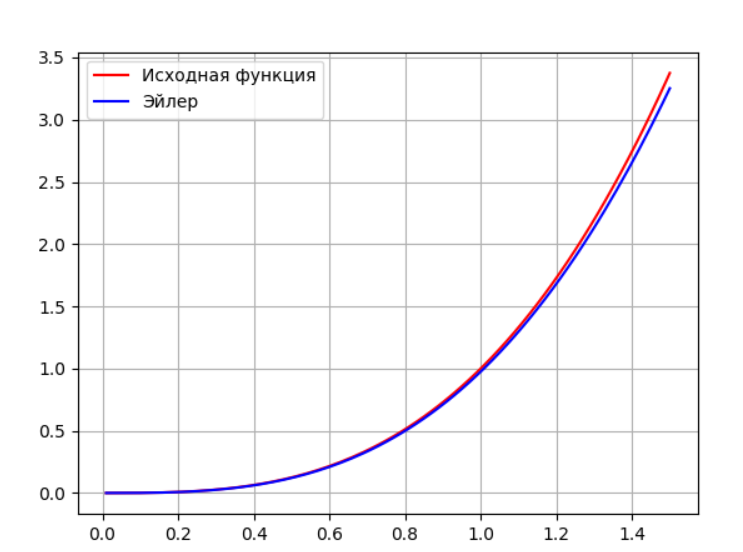


Рисунок 2 – График исходной функции и приближения по методу Эйлера

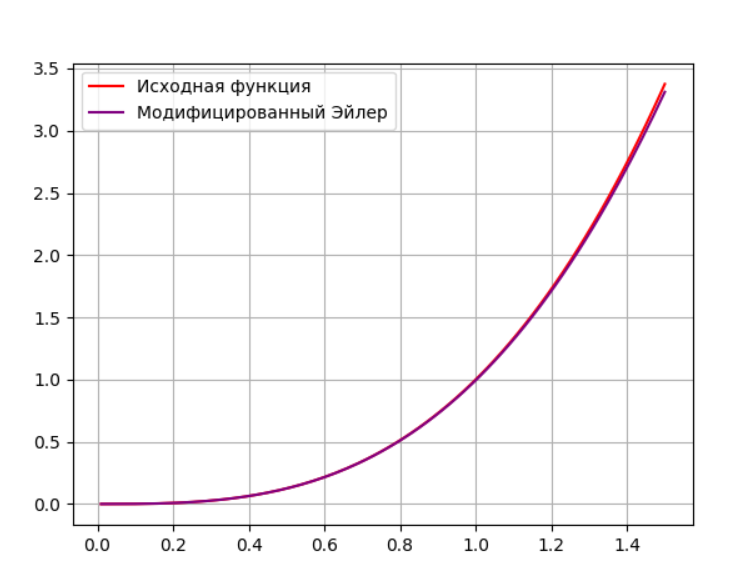


Рисунок 3 – График исходной функции и приближения по модифицированному методу Эйлера

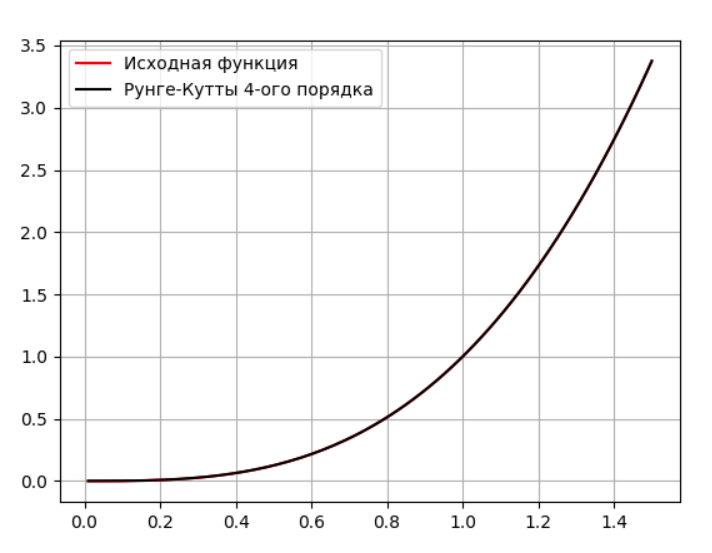


Рисунок 4 – График исходной функции и приближения по методу Рунге-Кутты

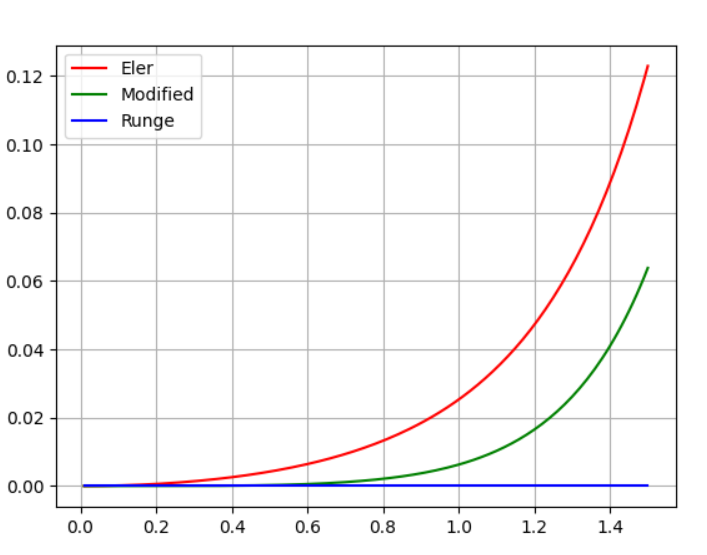


Рисунок 5 – График оценки погрешностей приближений от аналитического решения

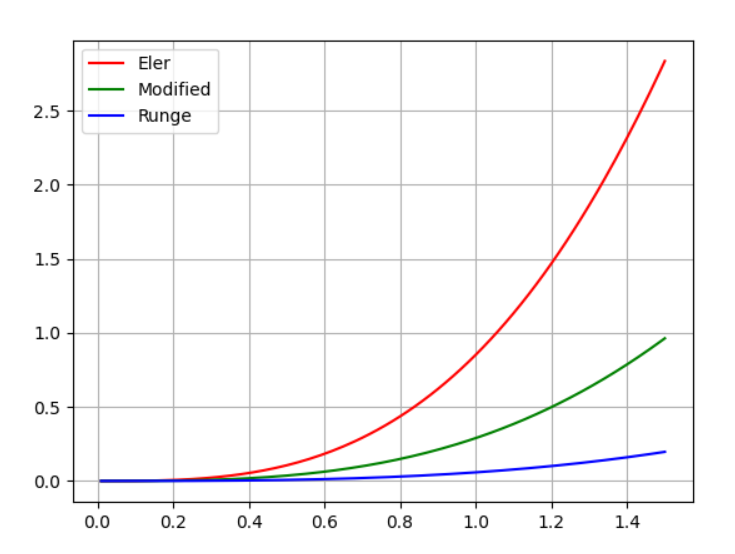


Рисунок 5 – График оценки погрешностей приближений от аналитического решения по правилу Рунге

**Вывод**: изучена разработка алгоритмов численного решения ОДУ. По результатам сравнения самым точным оказался метод Рунге-Кутты, а метод Эйлера самым не точным, по правилу Рунге также самый точный метод Рунге-Кутты, а самый не точный метод Эйлера.

# **ПРИЛОЖЕНИЕ А**

**Графические схемы алгоритмов**

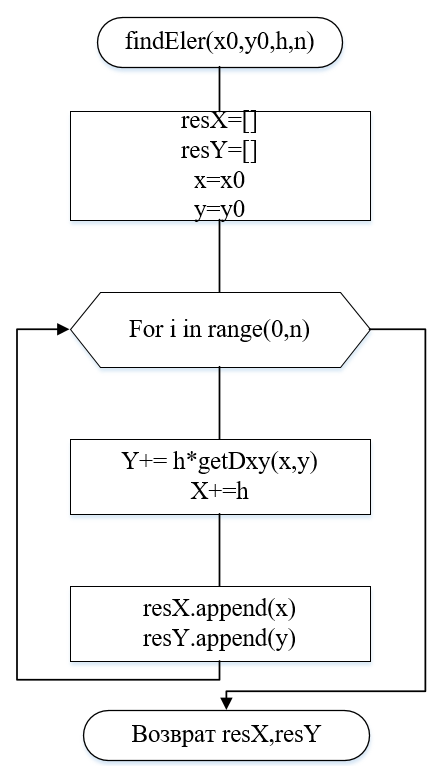


Рисунок A.1 – Графическая схема алгоритма по методу Эйлера

y – текушее значение y

x – текущее значение x

resx – массив значений x

resy – массив значений y

h ­­­­­– шаг

n ­­­­­­– количество значений

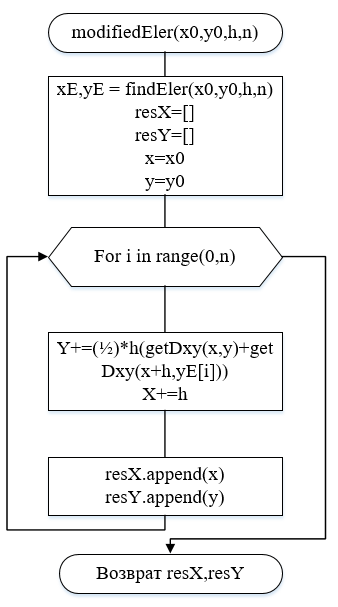


Рисунок A.2 – Графическая схема алгоритма по модифицированному методу Эйлера

y – текушее значение y

x – текущее значение x

resx – массив значений x

resy – массив значений y

h ­­­­­– шаг

n ­­­­­­– количество значений

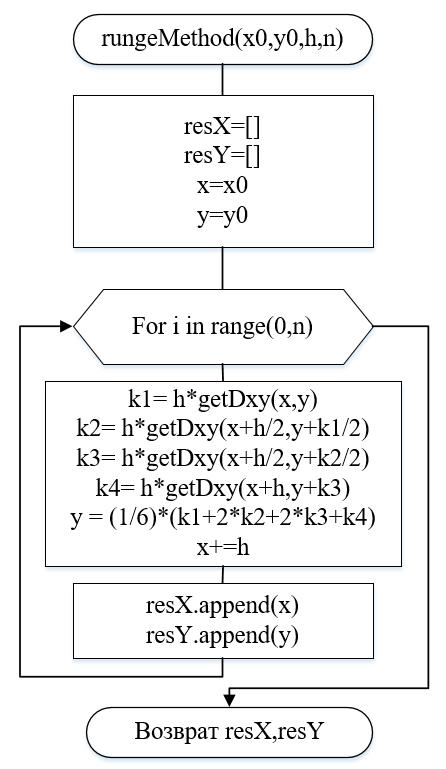


Рисунок A.3 – Графическая схема алгоритма по методу Рунге-Кутты

k1,k2,k3,k4 – коэффиценты по формулам Рунге-Кутты

y – текушее значение y

x – текущее значение x

resx – массив значений x

resy – массив значений y

h ­­­­­– шаг

n ­­­­­­– количество значений

# **ПРИЛОЖЕНИЕ Б**

**Листинг программы**

import math

import matplotlib.pyplot as plt

def main():

n = 150

xE,yE = findEler(0,0,0.01,n)

xM,yM = modifiedEler(0,0,0.01,n)

xR,yR = rungeMethod(0,0,0.01,n)

ix,iy = initial(xE,n)

showGraphic(ix, iy, "red", xE, yE, "blue","Эйлер")

showGraphic(ix, iy, "red", xM, yM, "green","Модифицированный Эйлер")

showGraphic(ix, iy, "red", xR, yR, "yellow","Рунге-Кутты 4-ого порядка")

plt.grid()

plt.plot(ix,findEps(iy, yE),color="green")

plt.plot(ix,findEps(iy, yM),color="blue")

plt.plot(ix,findEps(iy, yR),color="red")

plt.legend(['Eler','Modified','Runge'])

plt.show()

rungeRule(0,0,0.01,150)

#функция определения погрешности по правилу Рунге

def rungeRule(x0,y0,h,n):

xE,yE = findEler(x0,y0,h,n)

xE2,yE2 = findEler(x0,y0,h/2,n)

xM,yM = modifiedEler(x0,y0,h,n)

xM2,yM2 = modifiedEler(x0,y0,h/2,n)

xR,yR = rungeMethod(x0,y0,h,n)

xR2,yR2 = rungeMethod(x0,y0,h/2,n)

ix,iy = initial(xE,n)

y1 = dif(yE,yE2,1)

y2 = dif(yM,yM2,2)

y3 = dif(yR,yR2,4)

plt.grid()

plt.plot(ix,y1,color="green")

plt.plot(ix,y2,color="blue")

plt.plot(ix,y3,color="red")

plt.legend(['Eler','Modified','Runge'])

plt.show()

#формула Рунге для правила Рунге

def dif(y1,y2,p):

y = []

for i in range(0,len(y1)):

y.append((abs(y1[i]-y2[i]))/(pow(2,p)-1))

return y

#определение разностей приближенного значения от аналитического решения

def findEps(iy,my):

eps = []

for i in range(0,len(iy)):

eps.append(abs(iy[i]-my[i]))

return eps

#вычисление по методу Эйлера

def findEler(x0,y0,h,n):

resX = []

resY = []

x = x0

y = y0

for i in range(0,n):

y += h\*getDxy(x,y) #вычиление по формуле Эйлера

x += h

resX.append(x)

resY.append(y)

return resX,resY

#вычисление по модифицированному методу Эйлера

def modifiedEler(x0,y0,h,n):

xE,yE = findEler(x0,y0,h,n)

resX = []

resY = []

x = x0

y = y0

for i in range(0,n):

y += (1/2)\*h\*(getDxy(x,y)+getDxy(x+h,yE[i])) #Вычисление по модифицированнному методу

x += h

resX.append(x)

resY.append(y)

return resX,resY

#вычисление по методу Рунге-Кутты

def rungeMethod(x0,y0,h,n):

resX = []

resY = []

x = x0

y = y0

for i in range(0,n):

k1 = h \* getDxy(x,y) #вычисление коэффицентов по формулам Рунге-Кутты

k2 = h \* getDxy(x+h/2,y+k1/2)

k3 = h \* getDxy(x+h/2,y+k2/2)

k4 = h \* getDxy(x+h,y+k3)

y += 1/6\*(k1+2\*k2+2\*k3+k4) #вычисление значения по формуле Рунге-Кутты

x += h

resX.append(x)

resY.append(y)

return resX,resY

#вычисление аналетического решения

def initial(xList,n):

y = []

for x in xList:

y.append(x/math.cos(x))

return xList,y

#вычисление f(x,y)

def getDxy(x,y):

return 1/math.cos(x)+y\*math.tan(x)

#вывод графика

def showGraphic(ix,iy,selectedColor,x,y,selectedColor2,label):

plt.grid()

plt.plot(ix,iy,color=selectedColor)

plt.plot(x,y,color=selectedColor2)

plt.legend(['Исходная функция',label])

plt.show()

main()