**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 №1

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

на тему: «Разработка программ по методам решения нелинейных 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 решения нелинейных уравнений.

**ЗАДАНИЕ**

**Вариант 30**

Разработать алгоритмы и написать программы, реализующие следующие методы решения нелинейных уравнений:

1. Метод половинного деления (дихoтомии).

2. Метод хорд (пропорциональных частей).

3. Метод касательных (метод Ньютона).

4. Метод простой итерации (последовательных приближений).

Нелинейное уравнение согласно варианту, представлено на рисунке 1.



Рисунок 1 – Уравнение согласно варианту

Решение уравнения с помощью калькулятора представлено на рисунке 2.

Решение уравнения методом половинного деления представлено на рисунке 3.

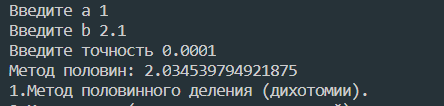


Рисунок 3 – Решение уравнения с помощью калькулятора

Решение уравнения методом хорд представлено на рисунке 4.

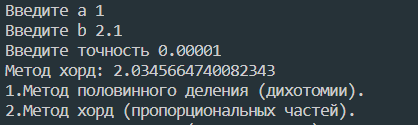


Рисунок 4 – Решение уравнения методом хорд

Решение уравнения методом Ньютона представлено на рисунке 5.

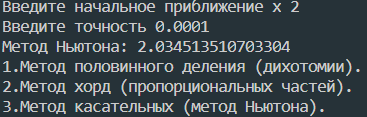


Рисунок 5 – Решение уравнения методом Ньютона

Решение уравнения методом итераций представлено на рисунке 6.

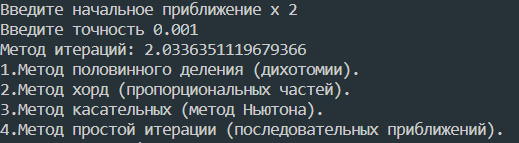


Рисунок 6 – Решение уравнения методом итераций

График функции представлен на рисунке 7.

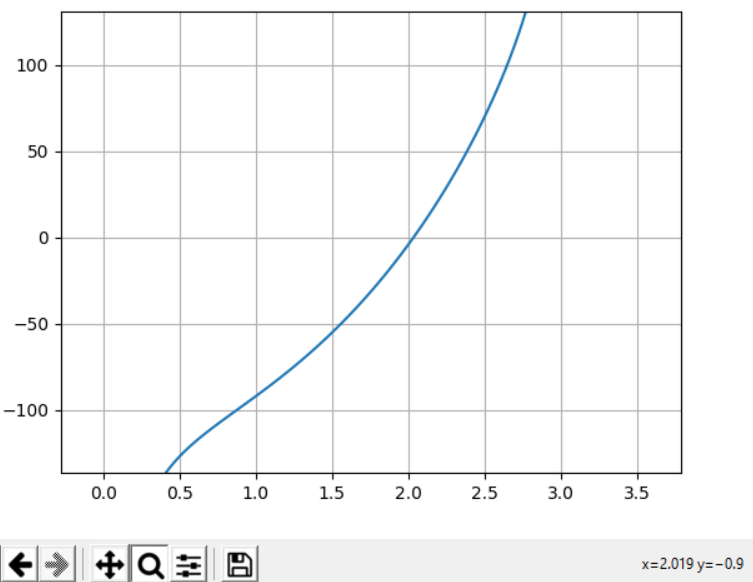


Рисунок 7 – График функции

Сравнивая время выполнений вычислений можно сделать вывод, что метод хорд самый быстрый (рисунок 8).

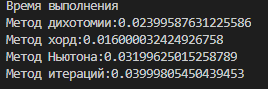


Рисунок 8 – Время вычислений различными методами

Подставляя полученные значения корня в уравнение можно сделать вывод, что метод хорд производит самые точные вычисления (рисунок 9).

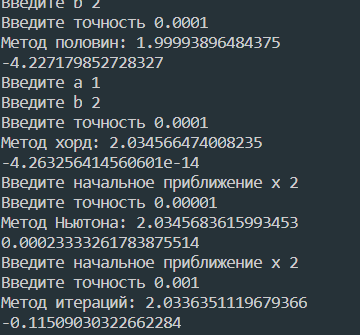


Рисунок 9 – Точность вычислений различных методов

**Вывод**: изучена разработка алгоритмов численных методов и программное обеспечение для решения нелинейных уравнений, а также произведено сравнение методов вычисления корня нелинейного уравнения. При сравнении точности вычислений для метода дихотомии получено значение: -4,27, для метода хорд: -4,232\*10-14, для метода Ньютона: 0.0002333, для метода итераций: -0.115, из этого можно сделать вывод что метод хорд самый точный. При сравнении времени выполнения для метода дихотомии получено значение: 0.02 с., для метода хорд: 0.01 с., для метода Ньютона: 0.03 с., для метода итераций: 0.04 с., из этого можно сделать вывод что метод хорд самый быстрый.

# **ПРИЛОЖЕНИЕ А**

**Графические схемы алгоритмов**



Рисунок A.1 – Графическая схема вычисления корня методом дихотомии

a,b – отрезок на котором расположено начальное приближение корня

e – заданная точность

fa – значение функции на конце отрезка

fс – значение функции на конце отрезка

с- приближение значения корня



Рисунок A.2 – Графическая схема вычисления корня методом хорд

a,b – отрезок на котором расположено начальное приближение корня

e – заданная точность

fa – значение функции на конце отрезка

fс – значение функции на конце отрезка

с- приближение значения корня



proizFunc()

Рисунок A.3 – Графическая схема вычисления корня методом Ньютона

x – ­начальное приближение корня

e – заданная точность

fс – значение функции с заданным значением

с – следующее приближение значения корня



secondFunc()

Рисунок A.4 – Графическая схема вычисления корня методом простых итераций

x – ­начальное приближение корня

e – заданная точность

fс – значение функции с заданным значением

с – следующее приближение значения корня

# **ПРИЛОЖЕНИЕ Б**

**Листинг программы**

import math

import matplotlib.pyplot as plt

import numpy as np

import os

#Метод половинного деления (дихoтомии).

def dichotomyMethod():

a=float(input("Введите a "))

b=float(input("Введите b "))

e=float(input("Введите точность "))

fa=func(a)

while (b-a)>=2\*e: #выполняется пока модуль разности отрезка больше удвоенной точности

c=(a+b)/2 #следущее приближение

fc=func(c) #вычислем значение функции с текущим приближением

if fa\*fc>0: #отбрасываем один из отрезков в зависимости от знака

a=c

else:

b=c

c=(a+b)/2

print("Метод половин: "+str(c))

#Метод хорд (пропорциональных частей).

def chordMethod():

a=float(input("Введите a "))

b=float(input("Введите b "))

e=float(input("Введите точность "))

fa=func(a)

while (abs(b-a))>e: #выполняется пока разность приближений больше точности

c=a-(b-a)/(func(b)-func(a))\*func(a) #точка пересечения хорды с осью абсцисс

fc=func(c) #подставляем значение в функцию

if fa\*fc>0: #отбрасываем один из отрезков в зависимости от знака

a=c

else:

b=c

c=(a+b)/2 #найденный отрезок делится пополам

print("Метод хорд: "+str(c))

#Метод касательных (метод Ньютона).

def newtonMethod():

x=float(input("Введите начальное приближение x "))

e=float(input("Введите точность "))

c=x

while True: #прекращается когда результаты двух последовательных итераций близки с опрделенной точностью

x=c-func(c)/derivedFunc(c) #нахождение следущего приближнения корня как абсциссы точки пересечения с осью x

if(abs(x-c))<=e: #осуществляет выход из цикла если значений итераций меньше точности

break

else:

c=x

print("Метод Ньютона: "+str(c))

#Метод простой итерации (последовательных приближений).

def iterationMethod():

x=float(input("Введите начальное приближение x "))

e=float(input("Введите точность "))

while True: #прекращается когда результаты двух последовательных итераций близки с опрделенной точностью

c=x #сохранение предыдущей итерации для нахождения следущей

x=convertedFunc(c) #вычисление значения функции записанной в виде x=f(x)

if(abs(x-c))<=e: #осуществляет выход из цикла если разность значений итераций меньше точности

break

print("Метод итераций: "+str(c))

#Вычисляет значение функции при заданном x

def func(x):

return x+math.cos(((x)\*\*0.52)+2)

#Вычисляет значение преоразованной для метода простых итераций функции при заданном x

def convertedFunc(x):

return -math.cos(((x)\*\*0.52)+2)

#Вычисляет значение производной функции для метода Ньютона при заданном x

def derivedFunc(x):

return 1-math.sin(x\*\*0.52+2)\*0.52/x\*\*0.48

#Строит график заданной функции

def showGraphic():

xmin=0.01

xmax=2

xlist=np.linspace(xmin,xmax,100)

ylist=[func(x) for x in xlist]

plt.grid()

plt.plot(xlist,ylist)

plt.show()

#Функция меню программы

def menu():

key=0

while key!=6: #выполняется до выбора пункта меню выход

key=selectKey()

os.system('cls' if os.name == 'nt' else 'clear')

if key==1:

dichotomyMethod()

elif key==2:

chordMethod()

elif key==3:

newtonMethod()

elif key==4:

iterationMethod()

elif key==5:

showGraphic()

#Функция получения выбора пользователя

def selectKey():

return int(input("Меню:\n1.Вычислить методом половинного деления (дихoтомии)\n2.Вычислить методом хорд (пропорциональных частей)\n3.Вычислить методом касательных (метод Ньютона)\n4.Вычислить методом простой итерации (последовательных приближений)\n5.Отобразить график\n6.Выход\nВыберите "))

menu()

**Вывод:** в ходе выполнения лабораторной работы были изучены и разработаны алгоритмы численных методов и программное обеспечение для решения нелинейных уравнений.