Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №3

Численное решение нелинейных уравнений

Выполнил: cтудент гр. 953506

Кондрашов И.Д.

Руководитель: доцент Анисимов В. Я.

Минск 2021

Содержание

Цель выполнения задания 2

Краткие теоретические сведения 3

Программная реализация 9

Выводы 11

# Цель выполнения задания

* изучить методы решения нелинейных уравнений(метод бисекции, метод хорд, метод Ньютона);
* составить алгоритм решения нелинейных уравнений указанными методами, применимый для организации вычислений на ЭВМ;
* составить программу решения нелинейных уравнений по разработанному алгоритму;
* сравнить число итераций, необходимых для достижения заданной точности вычисления разными методами
* выполнить тестовые примеры и проверить правильность работы программы.

# Краткие теоретические сведения

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, letter, email

Description automatically generated

Text

Description automatically generated with medium confidence

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated with medium confidence

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Graphical user interface, text, application, email

Description automatically generated**

**Text, letter

Description automatically generated**

A picture containing Word

Description automatically generated

# Программная реализация

Вариант 11

Text

Description automatically generated

Table

Description automatically generated with medium confidence



Теорема Штурма для нахождения корней уравнения x3 + ax2 + bx + c

def shturm\_amount\_of\_roots(poly=None, left\_border=None, right\_border=None):  
 def shturm\_range\_value(\_poly\_range\_, value):  
 counter = 0  
 size = len(\_poly\_range\_)  
 current\_sgn = \_poly\_range\_[0](value) > 0  
 for i in range(size - 1):  
 new\_sgn = \_poly\_range\_[i + 1](value) > 0  
 if new\_sgn != current\_sgn:  
 counter += 1  
 current\_sgn = new\_sgn  
 return counter  
 shturm\_sequence = []  
 shturm\_sequence.append(poly)  
 shturm\_sequence.append(sp.diff(poly))  
 sequence\_range = sp.degree(poly, gen=x)  
 for i in range(sequence\_range - 1):  
 shturm\_sequence.append(-sp.div(shturm\_sequence[i], shturm\_sequence[i + 1])[1])  
 return shturm\_range\_value(shturm\_sequence, left\_border) - shturm\_range\_value(shturm\_sequence, right\_border)

## Метод бисекции

def bisection(poly, left\_border, right\_border, accuracy):  
 counter = 0  
 while abs((left\_border - right\_border)) > accuracy:  
 average = (left\_border + right\_border)/2  
 if poly(left\_border)\*poly(average) <= 0:  
 right\_border = average  
 else:  
 left\_border = average  
 counter += 1  
 return average, counter

## Метод хорд

def chord(poly, left\_border, right\_border, accuracy):  
 temp = 0  
 average = 10\*accuracy  
 counter = 0  
 while abs(temp - average) > accuracy:  
 temp = average  
 average = left\_border - (poly(left\_border)/(poly(right\_border) - poly(left\_border)))\*(right\_border - left\_border)  
 if poly(left\_border)\*poly(average) <= 0:  
 right\_border = average  
 else:  
 left\_border = average  
 counter += 1  
 return average, counter

## Метод Ньютона

def newton(poly, entrypoint, accuracy):  
 derivative\_func = sp.diff(poly)  
 temp = entrypoint + accuracy\*10  
 counter = 0  
 while abs(entrypoint - temp) > accuracy:  
 temp = entrypoint  
 entrypoint = entrypoint - poly(entrypoint)/derivative\_func(entrypoint)  
 counter += 1  
 return entrypoint, counter

## Подтверждение работы программы:

A picture containing text, indoor, screenshot

Description automatically generated

Shape

Description automatically generated with medium confidence

Из последней картинки мы можем сделать вывод, что число итераций в методах отличается:

1. Метод бисекции – 18 итераций
2. Метод хорд – 6 итераций
3. Метод Ньютона – 5 итераций

**Выводы**

Таким образом, в ходе выполнения лабораторной работы был применён метод бисекции, метод хорд и метод Ньютона для решения нелинейных уравнений, составлены алгоритмы и созданы реализации соответствующих программ на языке Python для решения поставленной задачи.