**Лабораторная работа № 5**

**Методы вычислительной математики для нахождения корней нелинейного уравнения**

**Выполнил:** Селуянов Данила, 932102

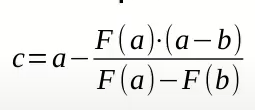
**Задание**

* Изучить теорию нахождения корней нелинейного уравнения методом хорд
* Реализовать алгоритм в программной среде языка Python
* Реализовать дополнительные проверки и исследования нелинейного уравнения, отобразить график
* Написать отчет

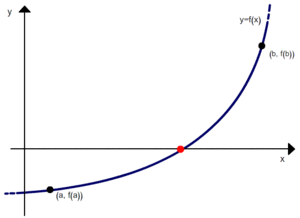
**Теоретическая часть**

Метод хорд является итерационным алгоритмом, таким образом решение уравнения заключается в многократном повторении этого алгоритма. Полученное в результате вычислений решение является приближенным, но его точность можно сделать такой, какой требуется, задав нужное значение погрешности ε. В начале вычислений методом хорд требуется указать границы области поиска корня; в общем случае эта граница может быть произвольной.

**Алгоритм:**

1. Привести выражение f(x) = g(x) к виду F(x) = 0
2. Определить начальное приближение в виде отрезка [A, B], так что (F(A) \* F(B) < 0), т.е. функции имеют разные знаки(значит в отрезке есть корень)
3. 
4. Повторять 3 пока |Cn – Cn-1| > **ε**, при этом в качестве следующего приближения выбирается тот отрезок, для которого справедливо (F(x) \* F’’(x) > 0)
5. Выводим С

**Графическое представление работы метода**



**Программный код**

##########

#  CODE  #          Лабораторная работа № 5

#   BY   #      Методы вычислительной математики для

# SAITER #      нахождения корней нелинейного уравнения

##########

import numpy as np

import matplotlib.pyplot as plt

def F(x):

    return (5 \* x - 8 \* np.log(x) - 8)

def F\_2pr(x):

    return (8 / (x\*\*2))

def half\_interval(a, b): #для дихотомии

    return (a + b) / 2

def method\_dich\_cnt(a, b, E): #МЕТОД ДИХОТОМИИ

    counter = 0

    while abs(b - a) > E:

        counter += 1

        fa = F(a)

        c = half\_interval(a, b)

        fc = F(c)

        if fa \* fc >= 0:

            a = c

        else:

            b = c

    return counter

def method\_chord(a, b, err):

    if(F(a) \* F(b) >= 0):

        print("Неверный интервал")

        exit()

    iter = 0

    c1 = a

    while True:

        c2 = a - ( F(a) \* (a-b) ) / ( F(a)- F(b) )

        if abs(c1-c2) < err:

            return [c2, iter]

        if (F(a) \* F\_2pr(a) > 0): #подбор точки для лучшей сходимости

            b = c2

        else:

            a = c2

        iter += 1

        if(iter >= 100000):

            print("Невозможно достичь заданной точности")

            exit()

        c1 = c2

def draw\_graph(x1, a, b):

    x = np.arange(-10, 10.01, 0.01)

    plt.plot(x, F(x), 0, 0) # отрисовка графа

    plt.title("y(x) = 5x – 8ln(x) - 8") #

    plt.xlabel("x")                     # оформление графа

    plt.ylabel("y")                     #

    plt.vlines(a, -10, 10, color = 'r')

    plt.vlines(b, -10, 10, color = 'r')

    plt.vlines(0, -10, 30, color = 'black')

    plt.hlines(0, -2, 10, color = 'black')

    plt.text(x1 + 0.1, F(x1) - 2, '(' + str(round(x1, 3)) + ",0)") # координаты точки

    plt.scatter(x1, F(x1), color = "red", s = 40, marker = 'o') # точка по найденному X

    plt.show()

#########################################

print("Для уравнения    5x – 8ln(x) = 8     найдем корни МЕТОДОМ ХОРД")

print()

A, B = map(float, input("Введиете интервал поиска корня [A,B]: ").split())

e = float(input("Введите точность: "))

x, iter\_chord = method\_chord(A, B, e)

print()

print("x =", x)

print("Итерации(метод Хорд):\t\t", iter\_chord)

print("Итерации(метод Дихотомии):\t", method\_dich\_cnt(A, B, e))

draw\_graph(x, A, B)

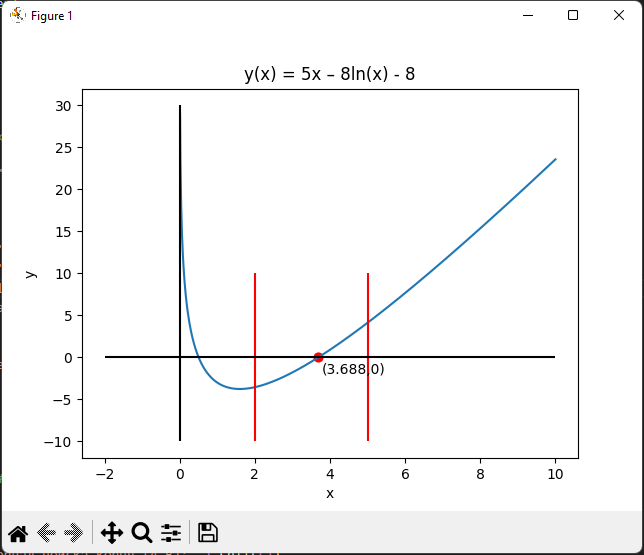
#########################################

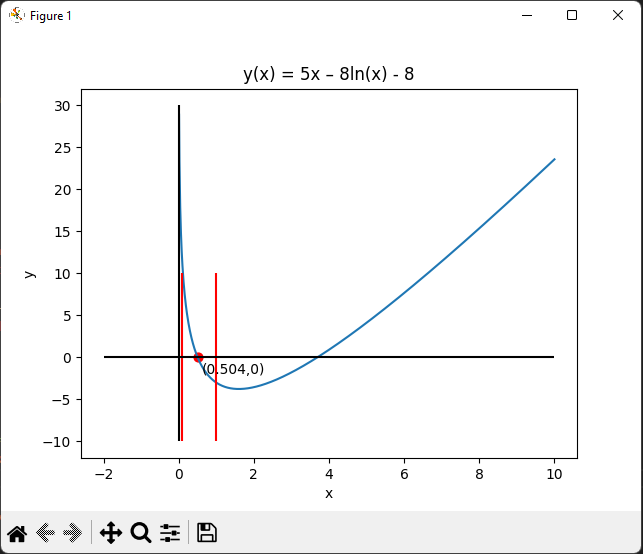
**Результаты работы программы**

**Тесты**

|  |  |  |
| --- | --- | --- |
| **Диапазон** | **Точность ε** | **Результат** |
| 2 5 | 0.1 | 3.6849697391917533 |
| 3 4 | 0.01 | 3.6882156638941512 |
| 3.4 3.8 | 0.001 | 3.6882377991638515 |
| 0.1 1 | 0.1 | 0.612754025265157 |
| 0.2 0.8 | 0.01 | 0.5089806343020169 |
| 0.4 0.6 | 0.001 | 0.5041798348409952 |

**Графики**

****

****

**Исследование скорости сходимости в зависимости от заданной точности**

|  |  |  |
| --- | --- | --- |
| **Точность ε** | **Кол-во итераций** | **Результат** |
| 0.1 | 2 | 3.6849697391917533 |
| 0.01 | 3 | 3.6879122962658664 |
| 0.001 | 4 | 3.688205765417373 |
| 0.0001 | 5 | 3.6882350099999304 |
| 0.00001 | 6 | 3.6882379240234333 |
| 0.1^10 | 12 | 3.6882382465174364 |
| 0.1^15 | 16 | 3.6882382465177517 |

**Исследование скорости сходимости в зависимости от алгоритма**

|  |  |  |  |
| --- | --- | --- | --- |
| **Диапазон** | **Точность ε** | **Кол-во итераций**  **метод Хорд** | **Кол-во итераций**  **метод Дихотомии** |
| 2 5 | 0.1 | 2 | 5 |
| 3 4 | 0.0001 | 3 | 14 |
| 1 6 | 0.0000001 | 10 | 26 |
| 0.1 1 | 0.001 | 11 | 10 |
| 0.4 0.6 | 0.0000001 | 8 | 21 |
| 0.2 0.8 | 0.000000001 | 23 | 30 |

**Вывод**

Проведя анализ теории и реализовав метод Хорд, а также сравнив его скорость сходимости с методом Дихотомии, можно сказать, что метод Хорд сходится намного быстрее, однако метод Дихотомии проще в реализации и не требует использования двойной производной.