Лабораторная работа №4

**Тема:** Численные методы решения нелинейных уравнений

**Язык:** Python 3.7.4

**Постановка задачи:** Реализовать численные методы решения нелинейных уравнений.



**Переменные:**

Модуль main.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| main\_menu | Содержит значения пунктов главного меню | tuple |
| main\_cond | Показывает какой пункт в главном меню выбрал пользователь | int |
| methods\_cond | Показывает какой пункт в меню Решения ДУ выбрал пользователь | int |
| cond\_checker | Выводит меню и возвращает введенное  пользователем значение. | function |

Модуль methods.py

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Значение** | **Тип** |
| func(x) | Нелинейное уравнение | function |
| func\_diff(x) | Производная от f(x) | function |
| newton() | Метод Ньютона | function |
| bisection() | Метод деления отрезка пополам | function |
| chord() | Метод хорд | function |

**Код программы:**

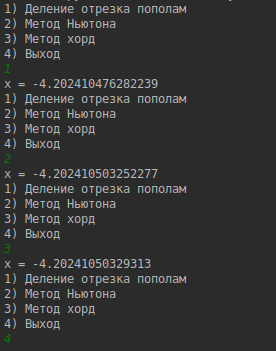
Модуль main.py

from methods import \*  
  
  
def cond\_checker(menu: list):  
 *"""  
 На вход поступает меню в виде списка.  
 Функция выводит меню и возвращает введенное  
 пользователем значение.  
 """* for each in menu:  
 print(each)  
 condition = int(input())  
 return condition - 1  
  
  
def main():  
 main\_cond = -1  
 while main\_cond != 3:  
 main\_cond = cond\_checker(main\_menu)  
 if main\_cond == 0:  
 bisection()  
 if main\_cond == 1:  
 newton()  
 if main\_cond == 2:  
 chord()  
 if main\_cond == 3:  
 pass  
  
  
main\_menu = ('1) Деление отрезка пополам', '2) Метод Ньютона', '3) Метод хорд', '4) Выход')  
main()

Модуль methods.py

import math  
  
a = -10  
b = 10  
acc = 10e-6  
  
  
def func(x):  
 return x \*\* 4 - 18 \* x \*\* 2 + 6  
  
  
def func\_diff(x):  
 return 4 \* x \* (x \*\* 2 - 9)  
  
  
def newton():  
 x\_prev = a  
 x\_curr = x\_prev - func(x\_prev) / func\_diff(x\_prev)  
 while abs(x\_curr - x\_prev) >= acc:  
 x\_prev = x\_curr  
 x\_curr = x\_prev - func(x\_prev) / func\_diff(x\_prev)  
 print('x =', x\_curr)  
  
  
def bisection():  
 x\_prev = a  
 x\_curr = b  
 if abs(x\_curr) == abs(x\_prev):  
 x\_curr += a/2  
 x = (x\_prev + x\_curr) / 2  
 while abs(func(x)) > acc:  
 x = (x\_prev + x\_curr) / 2  
 if func(x\_prev) \* func(x) < 0:  
 x\_prev, x\_curr = x\_prev, x  
 else:  
 x\_prev, x\_curr = x, x\_curr  
 print('x =', (x\_prev + x\_curr) / 2)  
  
  
def chord():  
 x\_prev = a  
 x\_curr = b  
 if abs(x\_prev) == abs(x\_curr):  
 x\_curr += a/2  
 while abs(x\_curr - x\_prev) >= acc:  
 x = x\_curr - ((x\_curr - x\_prev) \* func(x\_curr)) / (func(x\_curr) - func(x\_prev))  
 x\_prev = x\_curr  
 x\_curr = x  
 print('x =', x\_curr)

**Полученные значения:**

****

**Вывод:**

В результате вычислений мы получили примерно равные значения для x. Метод хорд и деления отрезка пополам может не работать при условии если границы отрезка в котором определена функция равны по модулю.