Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Лабораторная работа №2.

Поиск корней алгебраического уравнения методом секущих

По дисциплине «Software Engineering Process»

Вариант №3

Проверила: Беккулова К.А.

Выполнил студент:

Бектурсунова Айжамал

группа ПИ-1-22 (англ)

г. Бишкек 2024

**Тема: Проектирование и программная реализация решения задачи поиска корней алгебраического уравнения методом касательных (Ньютона) с заданной точностью ε**

1. **Общие сведения**

Данная работа направлена на создание программы для численного нахождения корней алгебраического уравнения методом Ньютона (касательных), используя язык программирования Python в среде Jupyter Notebook. Программа должна позволять пользователю вводить исходные данные, такие как начальное приближение, точность (погрешность) и максимальное количество итераций. В процессе работы программа будет вычислять приближённое значение корня, выводить итерационные значения в таблице и строить график, показывающий сходимость метода.

**2. Цель**

Целью проекта является разработка инструмента для численного анализа и визуализации метода Ньютона, который может быть использован в учебных и исследовательских целях для решения уравнений вида f(x)=0f(x) = 0f(x)=0. Программа должна демонстрировать процесс сходимости метода и обеспечивать удобный интерфейс для ввода исходных данных и анализа результатов.

**3. Функциональные требования**

**3.1 Входные данные:**

* Начальное приближение x0 для нахождения корня.
* Максимально допустимая погрешность: ϵ\epsilonϵ (tolerance).
* Максимальное количество итераций: Nmax.

**3.2 Выходные данные:**

* Таблица, содержащая результаты каждой итерации:
  + Номер итерации.
  + Значение текущего приближения xn​.
  + Значение функции f(xn) и её производной f′(xn).
  + Текущая погрешность ϵn \ epsilon\_nϵn.
* Найденное приближённое значение корня с указанием числа итераций, за которое была достигнута заданная точность.
* График, отображающий сходимость метода (приближение корня на каждой итерации).

**3.3 Обработка ошибок:**

* Если производная функции f′(xn) равна нулю, программа должна выводить сообщение об ошибке.
* Если процесс не сойдётся за указанное количество итераций, программа должна выводить сообщение об отсутствии сходимости.

**Структура программы:**

1. **Ввод данных**:
   * Пользователь вводит начальное приближение x0, точность ϵ\epsilonϵ, и максимальное количество итераций Nmax.
2. **Реализация метода Ньютона**:
   * Программа выполняет итерации метода касательных по следующей формуле:



* + Итерации продолжаются до тех пор, пока не будет достигнута заданная точность ϵ\epsilonϵ или не будет превышено максимальное количество итераций.

1. **Вывод результатов**:
   * Таблица с информацией по каждой итерации: значения  текущая погрешность.
   * Конечное значение корня с указанием числа итераций.
2. **Графическая визуализация**:
   * Построение графика, отображающего сходимость метода по итерациям (основываясь на приближениях 
3. **Обработка ошибок**:
   * Проверка на случай, если что приведёт к невозможности продолжения итераций.
   * Вывод сообщения об отсутствии сходимости, если решение не найдено за указанное число итераций.

Общее задание:

1.Исследование задачи. Построение прототипа:

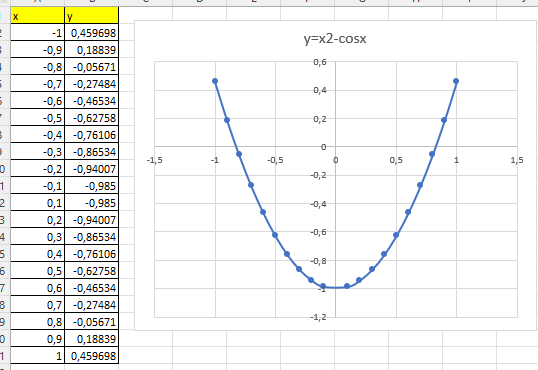


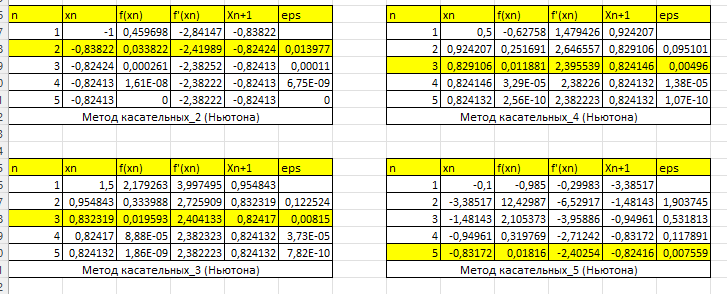
Рис.1 График функции *y*= *x2-cosx*



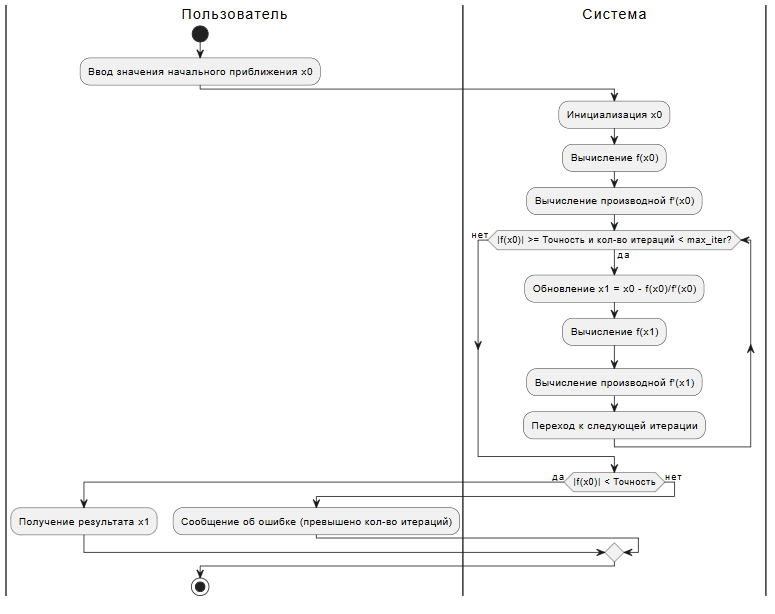
Рис.2 Результаты уравнения методом касательных с заданной точностью в EXCEL

2. Подготовка тестовых вариантов

Входные данные: *nmax,* *eps, x0.*

**Рис.3 Результаты тестовых вариантов

3. Алгоритм, построенный в виде диаграммы деятельности:

**

4. Код программы:

import numpy as np

import matplotlib.pyplot as plt

def f(x):

"""Функция, определяющая левую часть уравнения."""

return x\*\*2 - np.cos(x)

def df(x):

"""Производная функции f(x)."""

return 2\*x + np.sin(x)

def newton\_raphson(x0, tol=1e-6, max\_iter=100):

""" Функция реализует метод Ньютона-Рафсона.

Args:

x0: Начальное приближение корня.

tol: Точность вычислений.

max\_iter: Максимальное число итераций.

Returns:

Корень уравнения или None, если не сошелся.

"""

x = x0

for i in range(max\_iter):

fx = f(x)

if abs(fx) < tol:

return x

dfx = df(x)

if dfx == 0:

print("Деление на ноль!")

return None

x = x - fx/dfx

print("Превышено максимальное число итераций.")

return None

# Задаем начальное приближение

x0 = 1.0

# Вызываем функцию и получаем корень

root = newton\_raphson(x0)

if root is not None:

print("Найденный корень:", root)

# Построим график функции и касательной в найденной точке

x = np.linspace(-2, 2, 100)

y = f(x)

plt.plot(x, y, label='f(x)')

plt.plot(x, df(root)\*(x-root) + f(root), label='Касательная')

plt.scatter(root, 0, color='red', label='Корень')

plt.legend()

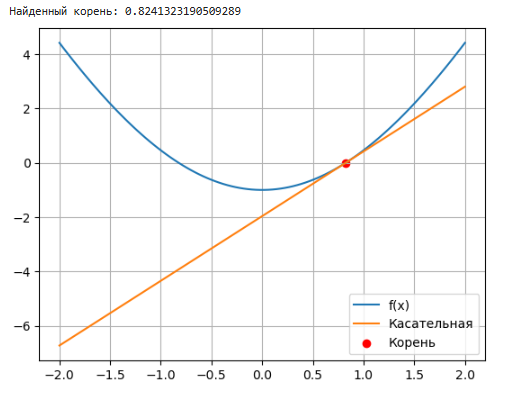
plt.grid(True)

plt.show()

else:

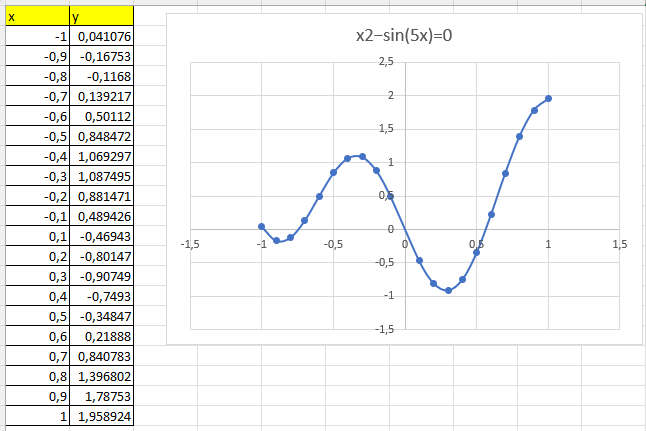
print("Корень не найден.")

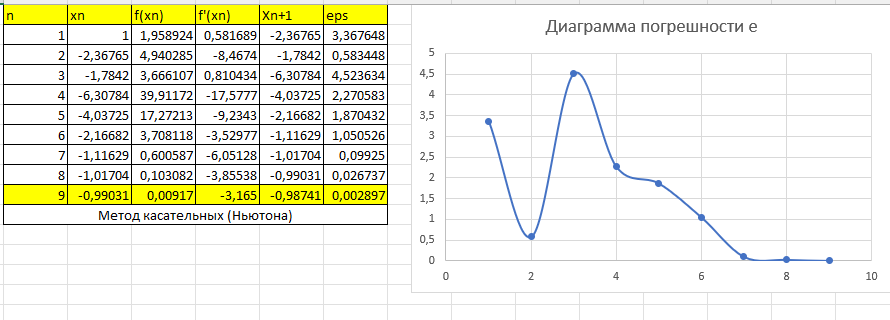
5. Результат:



Индивидуальное задание:

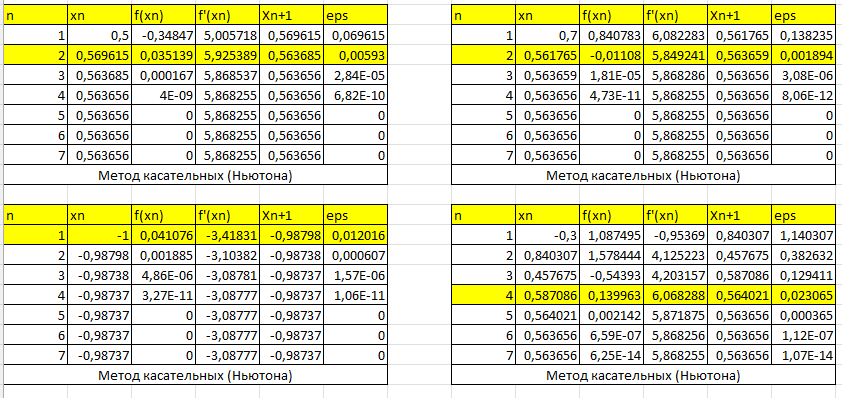
1.Исследование задачи. Построение прототипа:

 Рис.5 График функции *y* = x^2-Sin(5\*x)

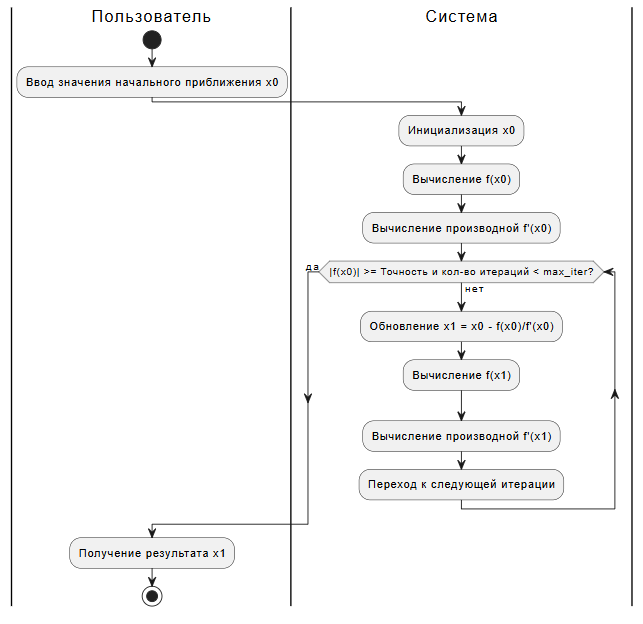
** Рис.2 Результаты уравнения методом касательных с заданной точностью в EXCEL

2. Подготовка тестовых вариантов

Входные данные: *nmax,* *eps, x0.*

 Рис.7 Результаты тестовых вариантов

3. Алгоритм, построенный в виде диаграммы деятельности:



4. Код программы:

import numpy as np

import matplotlib.pyplot as plt

def f(x):

"""Функция, определяющая левую часть уравнения."""

return x\*\*2 - np.sin(5\*x)

def df(x):

"""Производная функции f(x)."""

return 2\*x - 5\*np.cos(5\*x)

def newton\_method(x0, tol=1e-6, max\_iter=100):

"""

Метод Ньютона для поиска корня уравнения.

Args:

x0: Начальное приближение корня.

tol: Точность вычислений.

max\_iter: Максимальное число итераций.

Returns:

Корень уравнения или None, если не сошелся.

"""

for i in range(max\_iter):

fx = f(x0)

dfx = df(x0)

if abs(fx) < tol:

return x0

x0 -= fx / dfx

print("Превышено максимальное число итераций.")

return None

# Задаем начальное приближение

x0 = 0.5

# Вызываем функцию и получаем корень

root = newton\_method(x0)

if root is not None:

print("Найденный корень:", root)

# Построим график функции

x = np.linspace(-2, 2, 100)

y = f(x)

plt.plot(x, y, label='f(x)')

plt.plot(x, df(root)\*(x-root) + f(root), label='Касательная')

plt.scatter(root, 0, color='red', label='Корень')

plt.legend()

plt.grid(True)

plt.show()

else:

print("Корень не найден.")

Результат программы:

