Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Лабораторная работа №1.

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 вида *f(x)=0* методом секущих с заданной точностью *е.*

**1. Общие сведения**

Данная разработка направлена на создание программы для численного нахождения корня уравнения методом секущих, используя язык программирования Python в среде Jupyter Notebook. Программа должна позволять пользователю вводить исходные данные, такие как начальные приближения, точность (погрешность) и максимальное количество итераций, после чего рассчитывать приближённое решение уравнения. Программа будет выводить итерационные значения в таблице и строить график сходимости.

**2.Цель**

Целью данной работы является разработка инструмента для численного анализа и визуализации метода секущих, который может быть использован в образовательных или исследовательских целях.

**3. Функциональные требования**

**3.1 Входные данные**

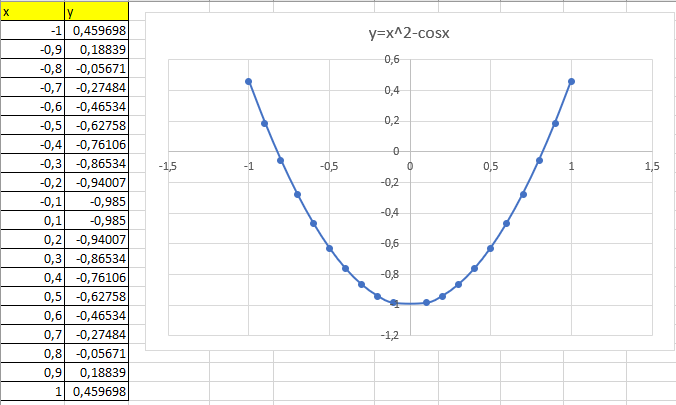
* Начальные приближения для решения уравнения: *x0, x1*
* Максимальная допустимая погрешность (tolerance, eps).
* Максимальное количество итераций *nmax*

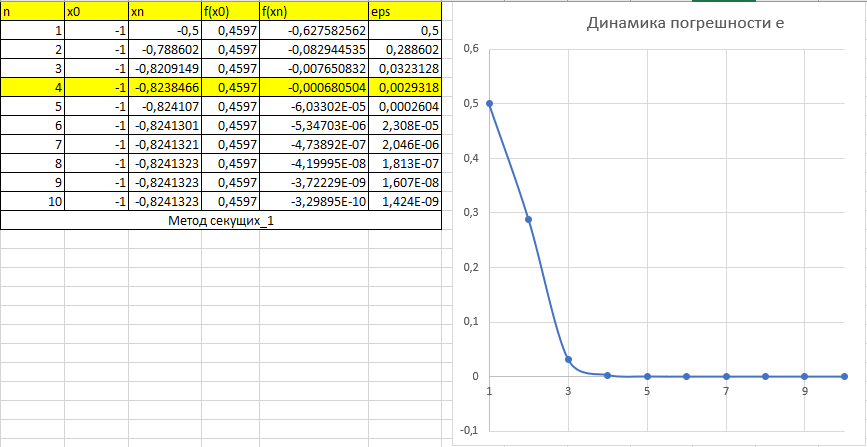
**3.2 Выходные данные**

* Таблица, содержащая результаты каждой итерации:
  + Номер итерации.
  + Значения *x0 и xn* на каждой итерации.
  + Значения функций f(*x0*) и f(*xn*) на каждой итерации.
  + Текущая погрешность eps
* Конечный результат (найденный корень) с указанием числа итераций, при которых достигнута заданная погрешность.
* Сообщение об ошибке, если:
  + Начальные значения *x0*​ и *x1*​ равны (деление на ноль).
  + Процесс не сходитcя за указанное количество итераций.

Общее задание:

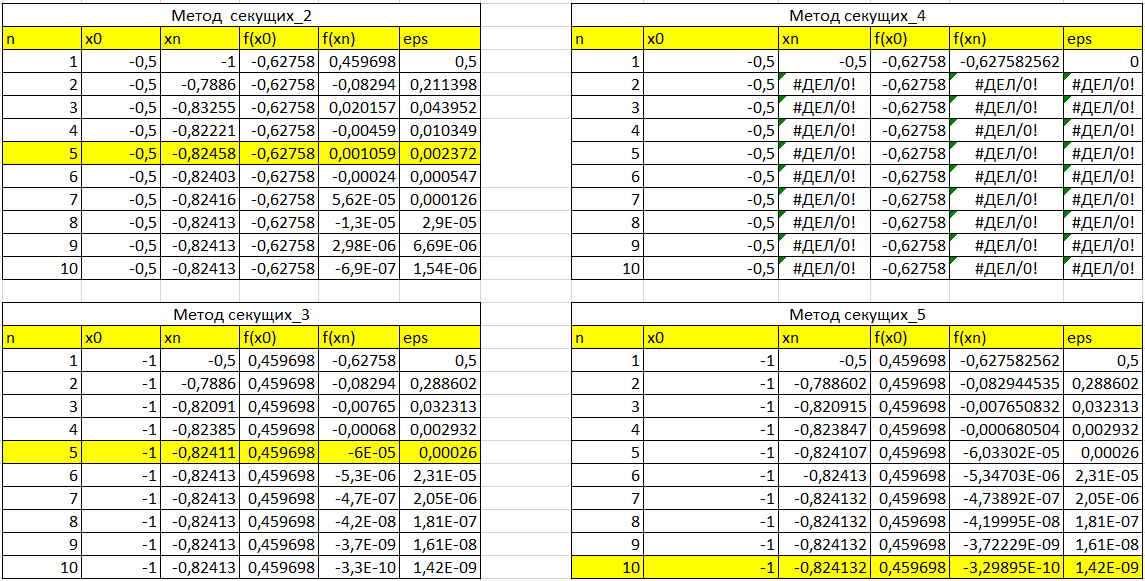
1.Исследование задачи. Построение прототипа:

Рис.1 График функции *y*= *x2-cosx*

 Рис.2 Результаты моделирования метода секущих в EXCEL

2. Подготовка тестовых вариантов

Входные данные: *nmax,* *eps, x0, x1.*

 Рис.3 Результаты тестовых вариантов

3. Алгоритм, построенный в виде диаграммы деятельности:



4. Код программы:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Function f(x) = x^2 - cos(x)

def f(x):

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

# Secant method function

def secant\_method(x0, x1, tolerance, max\_iterations):

iterations\_data = []

for iteration in range(max\_iterations):

f\_x0 = f(x0)

f\_x1 = f(x1)

# Calculate the new value of xn

if f\_x1 - f\_x0 == 0:

print("Error: Division by zero")

return None

xn = x1 - (f\_x1 \* (x1 - x0)) / (f\_x1 - f\_x0)

# Store the iteration data

eps = abs(x1 - x0)

iterations\_data.append([iteration + 1, x0, x1, f\_x0, f\_x1, eps])

# Check if tolerance is met

if eps < tolerance:

break

# Update x0 and x1 for next iteration

x0 = x1

x1 = xn

return iterations\_data, xn, iteration + 1

# Input parameters

x0 = -1.0 # Initial guess x0

x1 = -0.5 # Initial guess x1

tolerance = 0.01 # Desired tolerance

max\_iterations = 10 # Maximum number of iterations

# Run the Secant method

result, final\_xn, total\_iterations = secant\_method(x0, x1, tolerance, max\_iterations)

# If we have results, display them in a table and plot the convergence

if result:

# Convert the result into a DataFrame for pretty display

columns = ['n', 'x0', 'xn', 'f(x0)', 'f(xn)', 'eps']

df = pd.DataFrame(result, columns=columns)

# Display the result in tabular form

print(df)

print(f"\nSolution found: xn = {final\_xn}, after {total\_iterations} iterations")

# Plot the convergence of xn

plt.plot(df['n'], df['xn'], marker='o', color='b', label='xn')

plt.axhline(y=final\_xn, color='r', linestyle='--', label=f'Solution xn={final\_xn:.6f}')

plt.xlabel('Iteration')

plt.ylabel('xn value')

plt.title('Convergence of xn in Secant Method')

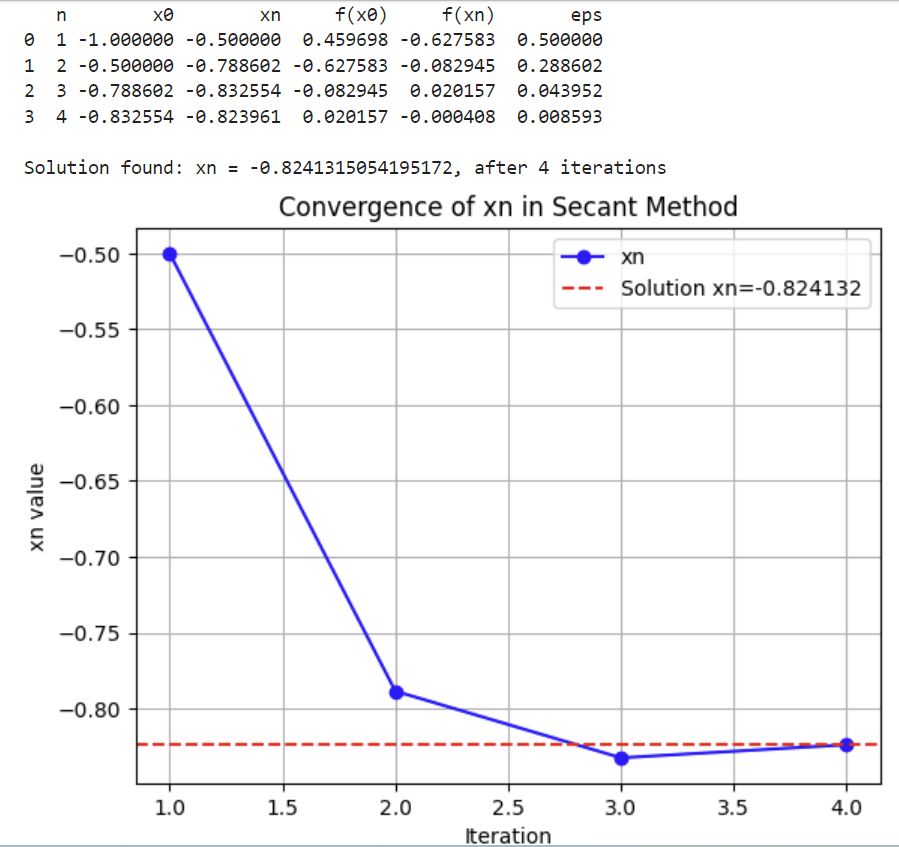
plt.grid(True)

plt.legend()

plt.show()

else: print("No solution found due to error.")

5. Результат:

 Рис.4 Результат графика функции *y*= *x2-cosx*

Индивидуальное задание:

1.Исследование задачи. Построение прототипа:

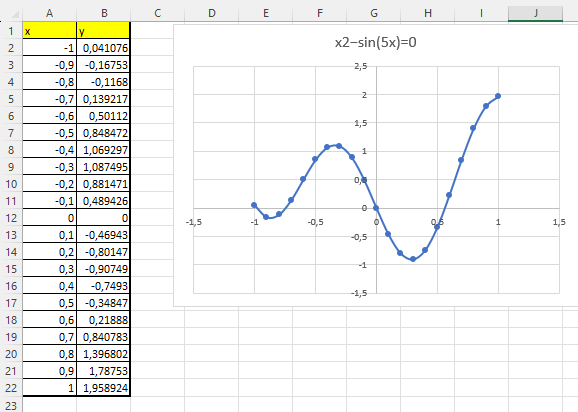
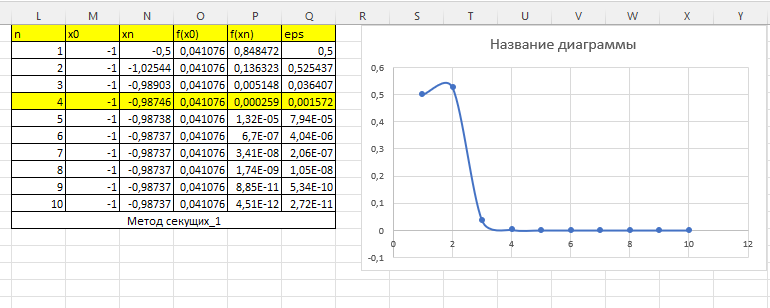
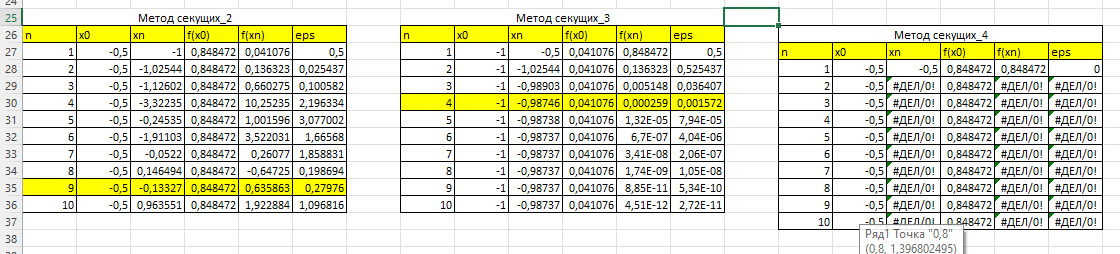


Рис.5 График функции *y* = x^2-Sin(5\*x)

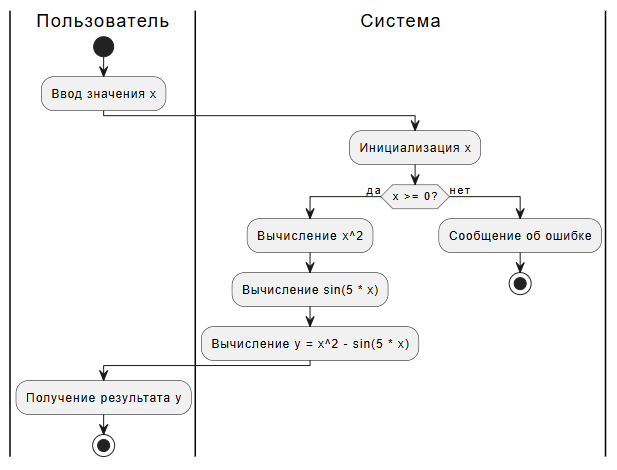
 Рис.6 Результаты моделирования метода секущих в EXCEL

2. Подготовка тестовых вариантов

Входные данные: *nmax,* *eps, x0, x1.*  Корень, который нам предстоит найти методом секущих, находится в интервале от -1 до 1.

 Рис.7 Результаты тестовых вариантов

3. Алгоритм, построенный в виде диаграммы деятельности:



4. Код программы:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Функция f(x) = x^2 - sin(5x)

def f(x):

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

# Функция для модифицированного метода секущих с фиксированным x0

def secant\_method\_fixed\_x0(x0, x1, tolerance, max\_iterations):

iterations\_data = []

for iteration in range(max\_iterations):

f\_x0 = f(x0)

f\_x1 = f(x1)

# Проверка на деление на ноль

if f\_x1 - f\_x0 == 0:

print("Ошибка: Деление на ноль")

return None

# Вычисление нового xn

xn = x1 - (f\_x1 \* (x1 - x0)) / (f\_x1 - f\_x0)

# Сохранение данных об итерации

eps = abs(f(xn)) # Обновляем ошибку на основе значения функции

iterations\_data.append([iteration + 1, x0, x1, f\_x0, f\_x1, eps])

# Проверка, достигнута ли точность

if eps < tolerance:

break

# Обновляем x1 для следующей итерации, x0 остаётся фиксированным

x1 = xn

return iterations\_data, xn, iteration + 1

# Входные параметры

x0 = -1 # Фиксированное значение x0

x1 = -0.5 # Начальное предположение для x1

tolerance = 0.01 # Желаемая точность

max\_iterations = 10 # Максимальное количество итераций

# Запуск метода секущих с фиксированным x0

result, final\_xn, total\_iterations = secant\_method\_fixed\_x0(x0, x1, tolerance, max\_iterations)

# Если результаты есть, выводим их в таблицу и строим график сходимости

if result:

# Преобразование результатов в DataFrame для удобного отображения

columns = ['n', 'x0', 'xn', 'f(x0)', 'f(xn)', 'eps']

df = pd.DataFrame(result, columns=columns)

# Вывод результата в табличной форме

print(df)

print(f"\nРешение найдено: xn = {final\_xn}, после {total\_iterations} итераций")

# Построение графика сходимости eps (ошибки)

plt.plot(df['n'], df['eps'], marker='o', color='b', label='eps (ошибка)')

plt.xlabel('Итерация')

plt.ylabel('eps (ошибка)')

plt.title('Динамика ошибки в методе секущих')

plt.grid(True)

plt.legend()

plt.show()

else:

print("Решение не найдено из-за ошибки.")

Результат программы:

