**Поиск корней алгебраических уравнений методом простых итераций**

**Цель работы**

Целью данной лабораторной работы является проектирование и реализация алгоритма для поиска корней уравнения f(x)=x2−cos(x)=0 с использованием метода простых итераций. Метод основан на преобразовании уравнения в эквивалентный вид, который позволяет использовать итерационный процесс для нахождения корня.

**Краткие теоретические сведения**

Метод простых итераций предполагает, что уравнение можно представить в виде x=φ(x). Это преобразование позволяет вычислять последовательные приближения к корню уравнения. x=φ(x).

Формула итераций:

xn=φ(xn−1),n=1,2,3,…

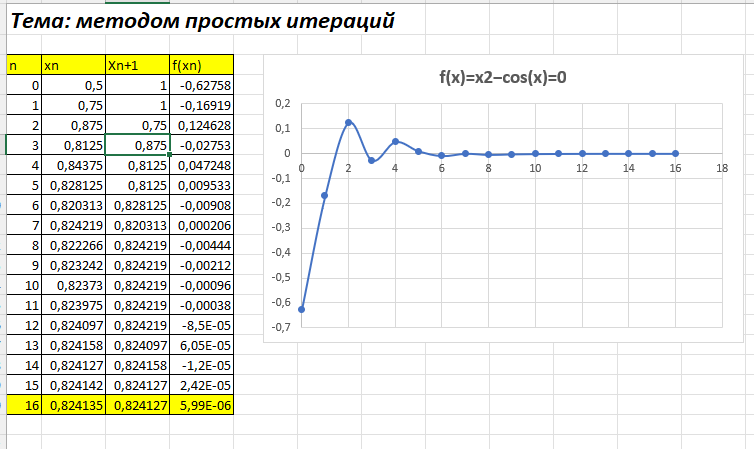
**Практическая часть**

1. **Инициализация**: Устанавливается начальное значение x0=0.5 и задана точность ϵ=0.01.
2. **Итерационный процесс**:
   * Вычисляются последовательные значения xn​ и xn+1​ до тех пор, пока не будет достигнута заданная точность.
   * Результаты каждой итерации сохраняются и выводятся.

**Задание**

Решить уравнение **f(x)=x2−cos(x)=0** методом простых итераций с заданной точностью **e=0.01**

**Результаты итераций представлены в таблице:**

****

**Код программы**

import math

def f(x):

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

def phi(x):

return math.sqrt(math.cos(x))

def simple\_iterations(x0, epsilon, max\_iterations=100):

x\_n = x0

iterations = []

for n in range(max\_iterations):

x\_n1 = phi(x\_n)

iterations.append((n, x\_n, x\_n1, f(x\_n)))

if abs(x\_n1 - x\_n) < epsilon:

break

x\_n = x\_n1

return iterations

# Параметры

x0 = 0.5 # Начальное значение

epsilon = 0.01 # Точность

# Запуск метода простых итераций

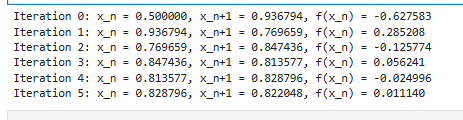
results = simple\_iterations(x0, epsilon)

# Вывод результатов

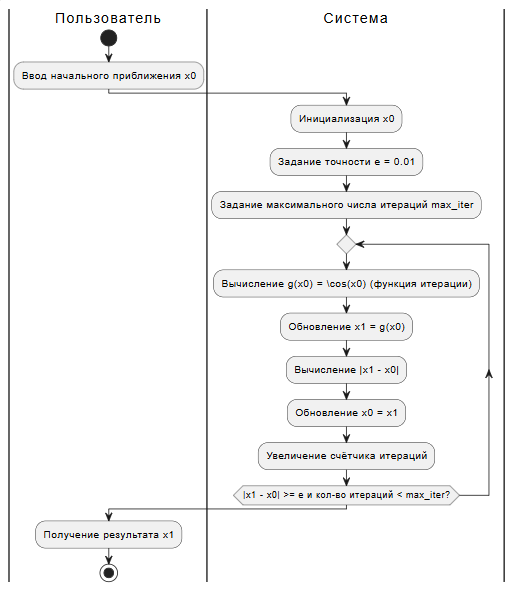
for n, x\_n, x\_n1, f\_x\_n in results:

print(f"Iteration {n}: x\_n = {x\_n:.6f}, x\_n+1 = {x\_n1:.6f}, f(x\_n) = {f\_x\_n:.6f}")

**Результат программы**



**UML Диаграмма активности**



Метод простых итераций для поиска корней уравнений.

**Цель:** Найти корень уравнения **φ(x) = (4x^2 - 10x + 10) / x^2**

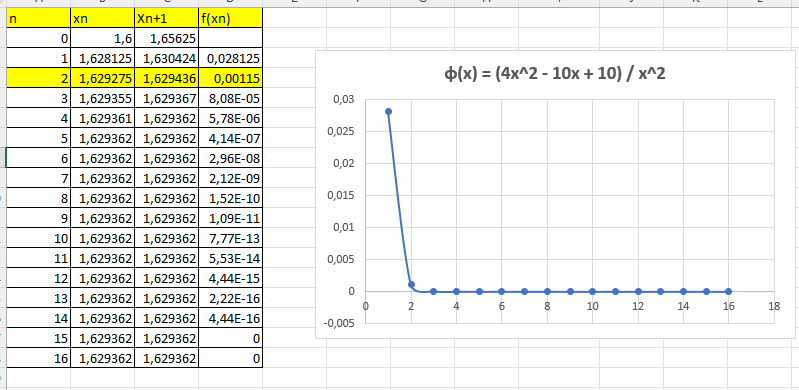
методом простых итераций с заданной точностью.

**Результаты на Excel**

В ходе выполнения индивидуального задания была успешно применена метод простых итераций для нахождения корней уравнения

**φ(x) = (4x^2 - 10x + 10) / x^2**

с заданной точностью ε=0.000001.



**Код программы**

import numpy as np

import matplotlib.pyplot as plt

# Определяем функцию g(x), которая представляет метод простых итераций: g(x) = sqrt(cos(x))

def g(x):

return np.sqrt(np.cos(x))

# Функция для простых итераций

def simple\_iteration(x0, tolerance, max\_iterations):

iteration = 0

x1 = g(x0)

# Сохраняем данные для анализа сходимости

convergence\_data = []

while np.abs(x1 - x0) >= tolerance and iteration < max\_iterations:

x0 = x1

x1 = g(x0)

iteration += 1

convergence\_data.append([iteration, x0, x1, np.abs(x1 - x0)])

return x1, iteration, convergence\_data

# Входные данные

x0 = 0.5 # начальное приближение

tolerance = 0.01 # точность

max\_iterations = 100 # максимальное количество итераций

# Запуск метода простых итераций

solution, iterations, data = simple\_iteration(x0, tolerance, max\_iterations)

# Вывод результатов

if iterations < max\_iterations:

print(f"Решение найдено: x = {solution}, за {iterations} итераций")

else:

print("Метод не сошелся за максимальное количество итераций")

# Печать данных о сходимости

for row in data:

print(f"Итерация: {row[0]}, x0: {row[1]:.6f}, x1: {row[2]:.6f}, Разница: {row[3]:.6f}")

# Построение графика сходимости

iterations = [row[0] for row in data]

errors = [row[3] for row in data]

plt.plot(iterations, errors, marker='o', label='|x1 - x0|')

plt.xlabel('Итерации')

plt.ylabel('Ошибка (|x1 - x0|)')

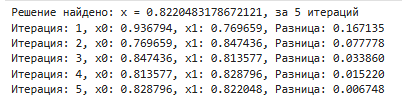
plt.title('График сходимости метода простых итераций')

plt.grid(True)

plt.legend()

plt.show()

**Результат программы**



**UML Диаграмма активности**

