МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

Кафедра автоматизированных систем и технологий

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

Дисциплина Численные методы

# Лабораторная №2.

# Решение НАУ (Вариант 19).

Выполнил студент

Копосов Лев Владимирович

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома

**Постановка задачи.**

Написать программу на языке PYTHON для нахождения действительных корней уравнения *f(x)=0* с точностью до 0.001, предварительно определив интервал [a, b], на котором существует решение уравнения.

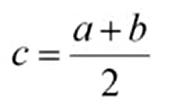
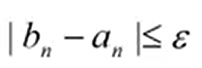


рис. 1. Задача N19

**Краткая теория используемых методов.**

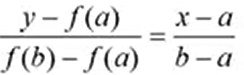
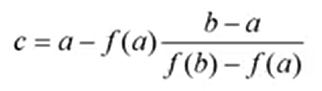
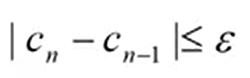
1. Метод половинного деления.

Метод половинного деления (метод бисекции) — это один из простейших численных методов для решения нелинейных уравнений. Он основан на теореме о промежуточных значениях и работает для непрерывных функций, у которых на концах отрезка изменяется знак.

Основная идея метода заключается в том, что если функция непрерывна на отрезке [a, b] и принимает значения f(a) и f(b) с разными знаками , то на этом отрезке существует корень уравнения . Метод половинного деления заключается в последовательном делении отрезка пополам  и выборе той половины , на концах которой функция имеет разные знаки. Потом выбираем начальное приближение  и продолжаем процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие , где bn - значение на оси OX в правой части промежутка, an - значение на оси OX в левой части промежутка, ε - заданная точность.

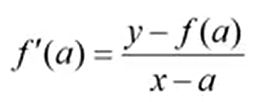
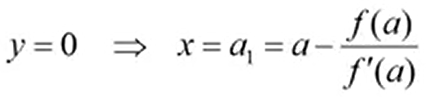
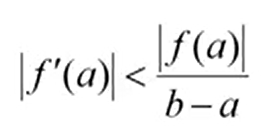
1. Метод хорд.

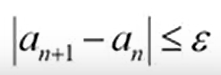
Метод хорд (метод секущих) — численный метод решения нелинейных уравнений. Он основан на линейной аппроксимации функции в окрестности корня уравнения и позволяет найти приближенное значение корня.

Основная идея метода хорд заключается в замене производной в методе Ньютона разностным приближением . Данную формулу преобразуем к виду . Вместо касательной к графику функции используется хорда. Потом выбираем начальное приближение  и продолжаем процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие , где cn, cn-1 - значения на оси OX, ε - заданная точность.

1. Метод касательных.

Метод касательных (метод Ньютона) — это численный метод решения нелинейных алгебраических уравнений, который основан на локальной аппроксимации функции в окрестности корня с помощью касательной. Он является одним из наиболее эффективных и широко используемых численных методов для нахождения корней уравнений.

Основная идея метода заключается в использовании локальной линейной аппроксимации функции в окрестности предполагаемого корня . Данную формулу преобразуем к виду , где y = 0. Сначала с помощью критерия  выбирается один из концов отрезка. В данном случае, если выполняется условие, начинаем итерацию с левого конца отрезка . Иначе - с правого конца отрезка . Потом выбираем начальное приближение  и продолжаем процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие

1) для левой части: , an, an+1 - значения на оси OX в левой части промежутка, ε - заданная точность.

2) для правой части , bn, bn+1 - значения на оси OX в правой части промежутка, ε - заданная точность.

**Алгоритм решения задачи НАУ.**

**Алгоритм отделения корней.**

Шаг 1. Делим отрезок [a, b] на n частей;

Шаг 2. Вычисляем значения f(a) и f(b);

Шаг 3. Если sign(f(a)) != sign(f(b)), то переходим к уточнению значения корня.

Иначе Переходим к Шаг 4;

Шаг 4. Вычисляем значения f(xi-1), f(xi) и f(xi+1) и находим разность Δyi = f(xi) - f(xi-1) и Δyi+1 = f(xi+1) - f(xi) на отрезке [xi, xi+2];

Шаг 5. Если sign(Δyi) != sign(Δyi+1), то используем методы для уточнения корней.

Иначе Берем сегмент из множества интервалов.

Для уточнения корней используются следующие методы:

1. **Метод половинного деления.**

Шаг 1. Задаем исходную функцию и множество интервалов, где будем проверять ее решение, а также точность вычисления ε;

Шаг 2. Берем сегмент из множества интервалов;

Иначе Конец.

Шаг 3. Если |b - a| < ε, то выводим корень решения функции x = (a + b) / 2, переходим к Шаг 2.

Шаг 4. Присваиваем новое значение c = (a + b) / 2

Шаг 5. Если sign(f(a)) != sign(f(c)), то присваиваем новое значение a = c；

Иначе присваиваем новое значение b = c;

Шаг 6. Переходим к Шаг 3.

|  |
| --- |
| Отделение корней |

рис. 2. Метод половинного деления

1. **Метод хорд.**

Шаг 1. Задаем исходную функцию и множество интервалов, где будем проверять ее решение, а также точность вычисления ε;

Шаг 2. Берем сегмент из множества интервалов;

Иначе Конец.

Шаг 3. Если |b - a| < ε, то выводим корень решения функции x = a, переходим к Шаг 2.

Шаг 4. Присваиваем новое значение c = a - (b - a) / (F(b) - F(a)) \* F(a);

Шаг 5. Если sign(f(a)) != sign(f(c)), то присваиваем новое значение b = c.

Иначе присваиваем новое значение a = c.

Шаг 6. Переходим к Шаг 3.

|  |
| --- |
| Отделение корней |

рис. 3. Метод хорд

1. **Метод касательных.**

Шаг 1. Задаем исходную функцию и множество интервалов, где будем проверять ее решение, а также точность вычисления ε;

Шаг 2. Берем сегмент из множества интервалов, [a, b];

Иначе Конец.

Шаг 3. Если sign(f(a)) != sign(f(c)), переходим к Шаг 4；

Иначе переходим к Шаг 2.

Шаг 4. Если выполняется условие |f`(a)| < |f(a)| / (b - a), то берем точку c = a;

Иначе берем точку c = b;

Шаг 5. Присваиваем новое значение c = c - f(c) / f`(c);

Шаг 6. Если |c - c\_prev| < ε, где c\_prev - значение c из предыдущей итерации, то выводим корень решения функции x = c, переходим к Шаг 2.

Иначе переходим к Шаг 5.

|  |
| --- |
| Отделение корней |

рис. 4. Метод касательных

**Вывод результата решения задачи.**

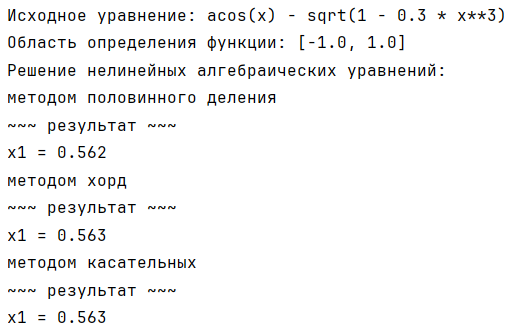
****

рис. 5. Вывод результата задачи

**Проверка правильности решения.**

Правильность решений была проверена с помощью unit тестов.

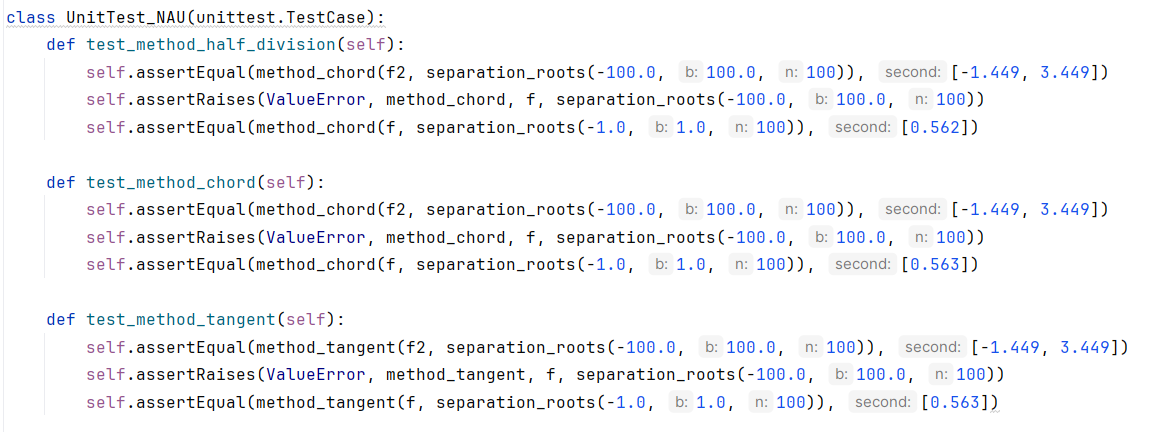
****

рис. 6. Проверка корректности программы с помощью unit тестов

**Выводы.**

Написали программу на языке PYTHON для нахождения действительных корней уравнения *f(x)=0* с точностью до 0.001 методами половинного деления, хорд и касательных, предварительно определив интервал [a, b], на котором существует решение уравнения. Предусмотрели наличие нескольких корней и проверили решение с помощью unit тестов.

**Приложение: код программы.**

| **import math**  ***# Отделение корней на отрезке [a,b]***  **def Dx(i):**  ***"""область определения функции проверяет значение"""***  **return (i >= -1.0 and i <= 1.0) and (1 - 0.3 \* i\*\*3 >= 0)**  **def D\_ab(a, b):**  ***"""область определения функции всего диапазона"""***  **min\_v = a**  **max\_v = b**  **while True:**  **min\_v = round(min\_v, 1)**  **max\_v = round(max\_v, 1)**  **if (Dx(min\_v) and Dx(max\_v)):**  **a = round(min\_v, 2)**  **b = round(max\_v, 2)**  **return (a, b)**  **if not Dx(min\_v):**  **min\_v += 0.1**  **if not Dx(max\_v):**  **max\_v -= 0.1**  **def separation\_roots(a, b, n):**  ***"""выделение n диапазонов"""***  **segment\_length = (b - a) / n**  **segments = []**  **for i in range(n):**  **start = a + i \* segment\_length**  **end = a + (i + 1) \* segment\_length**  **segments.append((start, end))**  **return segments**  **def f(x):**  ***"""исходная функция"""***  **return math.acos(x) - math.sqrt(1 - 0.3 \* x\*\*3)**  **def derivative1(func, x, accuracy = 1e-3):**  ***"""первая производная"""***  **return (func(x + accuracy) - func(x)) / accuracy**  ***# Уточнение корней методом половинного деления***  **def method\_half\_division(func, segments,**  **accuracy = 1e-3, MAX\_ITER = 100):**  ***"""решение нелинейного алгебраического уравнения методом половинного деления"""***  **root\_count = 0**  **for a\_segm, b\_segm in segments:**  ***# проверка на разные знаки***  **if func(a\_segm) \* func(b\_segm) > 0:**  **continue**  **iteration\_count = 1**  **while iteration\_count < MAX\_ITER:**  **c = (a\_segm + b\_segm) / 2**  **a\_val = func(a\_segm)**  **c\_val = func(c)**  **if abs(c\_val) < accuracy:**  **root\_count += 1**  **print(f"x{root\_count} = {round(c, 3)}")**  **break**  **if a\_val \* c\_val > 0:**  **a\_segm = c**  **else:**  **b\_segm = c**  **iteration\_count += 1**  ***# Уточнение корней методом хорд***  **def method\_chord(func, segments,**  **accuracy = 1e-3, MAX\_ITER = 100):**  ***"""решение нелинейного алгебраического уравнения методом хорд"""***  **a\_gl = segments[0][1]**  **b\_gl = segments[-1][0]**  **if abs(derivative1(func, a\_gl)) < abs(func(a\_gl)) / (b\_gl - a\_gl):**  **c1 = a\_gl**  **else:**  **c1 = b\_gl**  **root\_count = 0**  **for a\_segm, b\_segm in segments:**  ***# проверка на разные знаки***  **if func(a\_segm) \* func(b\_segm) > 0:**  **continue**  **iteration\_count = 1**  **while iteration\_count < MAX\_ITER:**  **a\_val = func(a\_segm)**  **b\_val = func(b\_segm)**  **c2 = a\_segm - (a\_val \* (a\_segm - b\_segm)) / (a\_val - b\_val)**  **c\_val = func(c2)**  **if abs(c1 - c2) < accuracy:**  **root\_count += 1**  **print(f"x{root\_count} = {round(c2, 3)}")**  **break**  **if a\_val \* c\_val > 0:**  **a\_segm = c2**  **else:**  **b\_segm = c2**  **c1 = c2**  **iteration\_count += 1**  ***# Уточнение корней методом касательной***  **def method\_tangent(func, segments,**  **accuracy = 1e-3, MAX\_ITER = 100):**  ***"""решение нелинейного алгебраического уравнения методом касательных (Ньютона)"""***  **root\_count = 0**  **for a\_segm, b\_segm in segments:**  ***# проверка на разные знаки***  **if func(a\_segm) \* func(b\_segm) > 0:**  **continue**  **if abs(derivative1(func, a\_segm)) < abs(func(a\_segm)) / (b\_segm - a\_segm):**  **c = a\_segm**  **else:**  **c = b\_segm**  **iteration\_count = 1**  **while iteration\_count < MAX\_ITER:**  **c\_prev = c**  **c = c - func(c) / derivative1(func, c)**  **if abs(c - c\_prev) < accuracy:**  **root\_count += 1**  **print(f"x{root\_count} = {round(c, 3)}")**  **break**  **iteration\_count += 1**  **if root\_count == 0:**  **print("корней нет")**  **if \_\_name\_\_ == '\_\_main\_\_':**  **print("Исходное уравнение: acos(x) - sqrt(1 - 0.3 \* x\*\*3)")**  **a, b = D\_ab(-100, 100)**  **print(f"Область определения функции: [{a}, {b}]")**  **segments = separation\_roots(a, b, 100)**  **print("Решение нелинейных алгебраических уравнений:")**  **print("методом половинного деления")**  **print(f"~~~ результат ~~~")**  **res1 = method\_half\_division(f, segments)**  **print("методом хорд")**  **print(f"~~~ результат ~~~")**  **res2 = method\_chord(f, segments)**  **print("методом касательных")**  **print(f"~~~ результат ~~~")**  **res3 = method\_tangent(f, segments)** |
| --- |