Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 1**

«Розв’язування нелінійних рівнянь методом дихотомії та методом хорд»

**з дисципліни «Чисельні методи ПЗ»**

**Лектор:**

доцент кафедри ПЗ

Мельник Н.Б.

**Виконав:**

студ. групи ПЗ-15

Марущак А. С.

**Прийняв:**

асистент кафедри ПЗ

Гарматій Г.Ю

«\_\_\_» \_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема роботи:** Розв’язування нелінійних рівнянь методом дихотомії та методом хорд.

**Мета роботи:** Ознайомлення на практиці з методами відокремлення дійсних ізольованих коренів нелінійних рівнянь. Вивчення методу дихотомії та методу хорд уточнення коренів.

**Теоретичні відомості**

Наступні методи розв’язування нелінійних рівнянь дозволяють знайти розв’язок для наступної задачі:Розглянемо рівняння , у якому є неперервною нелінійною функцією. На відрізку  дана функція є монотонною та диференційованою, на ньому міститься єдиний корінь  заданого рівняння, тобто . Потрібно знайти значення кореня  з заданою похибкою .

Процедура знаходження наближеного розв’язку нелінійного рівняння складається з двох етапів:

• локалізації коренів – визначення інтервалів, що містять єдиний корінь;

• уточнення коренів.

Локалізувати корені можна 2 способами: графічним (використовуючи графік функції) та аналітичним (провівши розрахунки).

Далі нам необхідно уточнити значення кореня одним із наведених нижче методів.

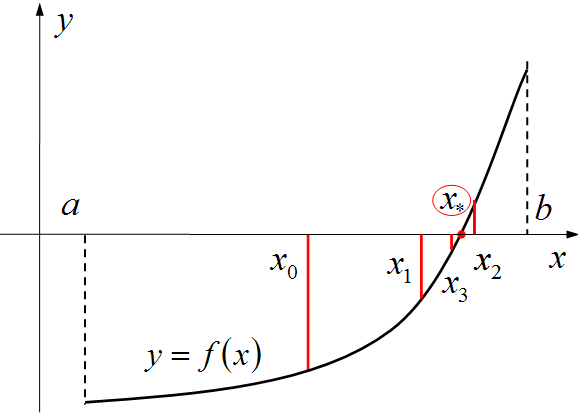
**Метод поділу відрізка навпіл**

Покладемо ,  і обчислимо . Якщо , то , у протилежному випадку, якщо , то чинимо так:

 (1)  (2)

 (3)

і обчислюємо . Якщо , то ітераційний процес завершуємо і вважаємо, що , а коли , то продовжуємо ітераційний процес (1)-(3).



*Рис 1.1. Графічна інтерпретація методу поділу відрізку навпіл*

Серед переваг даного методу потрібно відзначити простоту реалізації та надійність. Недоліком наведеного методу є невелика швидкість його збіжності.

**Опис алгоритму:**

1. Приймаємо значення start, end, precision.
2. Перевіряємо, чи значення функції в точках start та end мають різні знаки. Якщо так, то повідомляємо, що корені не можна знайти.
3. Шукаємо середнє значення між start та end
4. Якщо фукнція в цій точці має той самий знак, що і у точці start , тоді start = сер. знач., в противному випадку - функція має такий самий знак, що і у точці end, і тому end = сер.знач.
5. Повторюємо кроки 3,4 допоки |end –start| > precision і f(end) 0;
6. Повертаємо із функції значення end, а також к-ть ітерацій.

**Метод хорд (інтерполювання, або пропорційних частин)**

Суть методу хорд полягає в тому, що на відрізку  малої довжини дугу функції  замінюють хордою , яка її стягує. За наближене значення кореня приймають абсцису точки перетину хорди з віссю .

Для довільного -го наближення точного значення кореня для заданого рівняння використовують формулу

, , де x­­­0 = a

Дугу кривої стягують хордою доти, поки шуканий наближений корінь не досягне точності, тобто

,

де  – наближені значення кореня рівняння , відповідно на-му та ()-му ітераційному кроці

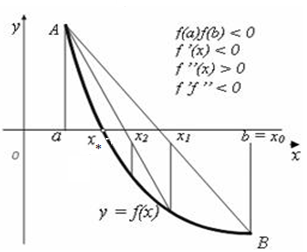
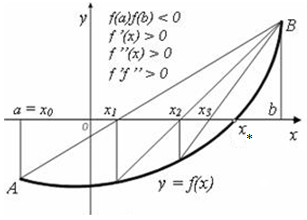
.

Рис 1.2. Графічна інтерпретація методу хорд

Для автоматизованого вибору рухомого кінця хорди і відповідно визначення співвідношення для обчислення наближеного значення кореня існує певне правило: нерухомим кінцем відрізка є той, для якого знак функції співпадає зі знаком її другої похідної. Якщо *f(b)f”(b) > 0*, то нерухомим є кінець(), інакше, якщо *f(a)f”(a) > 0*, то нерухомим є кінець ()*.*

**Опис алгоритму:**

1. Отримуємо значення start, end, precision.
2. Перевіряємо, чи значення функції в точках start та end мають різні знаки. Якщо так, то повідомляємо, що корені не можна знайти.
3. Якщо f(start)\*f’’(start) > 0, продовжуємо, інкаше ідемо до кроку 6
4. Рухаємо точку end у точку перетину лінії, проведену через точки (start, f(start)), (end, f(end)) з віссю абсцис, а попереднє значення end зберігаємо, як end\_prev.
5. Повторюємо крок 4, допоки |end–end\_previous| > precision, інакше повертаємо результат і кількість ітерацій.
6. Рухаємо точку start у точку перетину лінії, проведену через точки (start, f(start)), (end, f(end)) з віссю абсцис, а попереднє значення start зберігаємо, як start\_prev.
7. Повторюємо крок 6, допоки |start–start\_previous| > precision, інакше повертаємо результат і кількість ітерацій.

**Примітка:** для збільшення наочності в алгоритмі було прописано 2 варіанти: коли рухомим є початок відрізку або його кінець. Проте в скороченому вигляді, можна просто поміняти їх значення місцями і отримати бажаний результат.

**Індивідуальне завдання**

1. Ознайомитися з теоретичним матеріалом.

1. Відокремити дійсні корені рівняння графічним та аналітичним способами і скласти програму його розв’язування методом дихотомії та методом хорд..

**Хід роботи**

**Етап локалізації коренів**

**Графічний метод:**

Представимо рівняння у вигляді .

Побудуємо графіки функцій та :

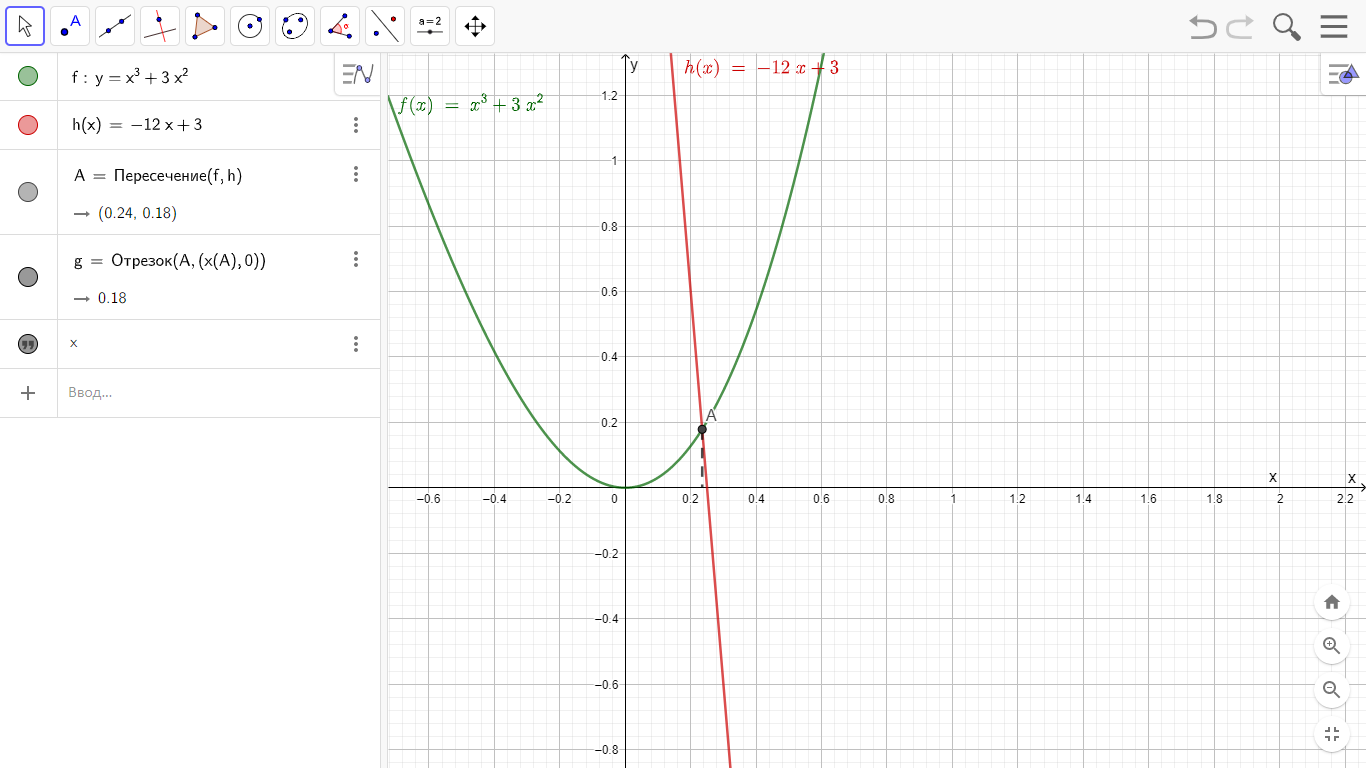


Рис 1.3. Графічний метод

З рисунка 1.3 можна точно встановити, що корінь локалізований на відрізку від 0 до 1.

**Аналітичний метод:**

Для аналітичного розв’язку визначимо монотонність функції f(x), для цього розв’яжемо рівняння f’(x) = 0 та знайдемо інтервали монотонності, враховуючи, що функція визначена на всій множині дійсних чисел:

= 0

Це рівняння не має розв'язків в дійсних числах. f’(x) > 0 на всій множині дійсних чисел, тому можемо зробити висновок, що функція має лише один корінь. Більшу інформацію отримати аналітичним методом у цьому випадку неможливо. Тоді протабулюємо значення функції на відрізку [-5; 5]:

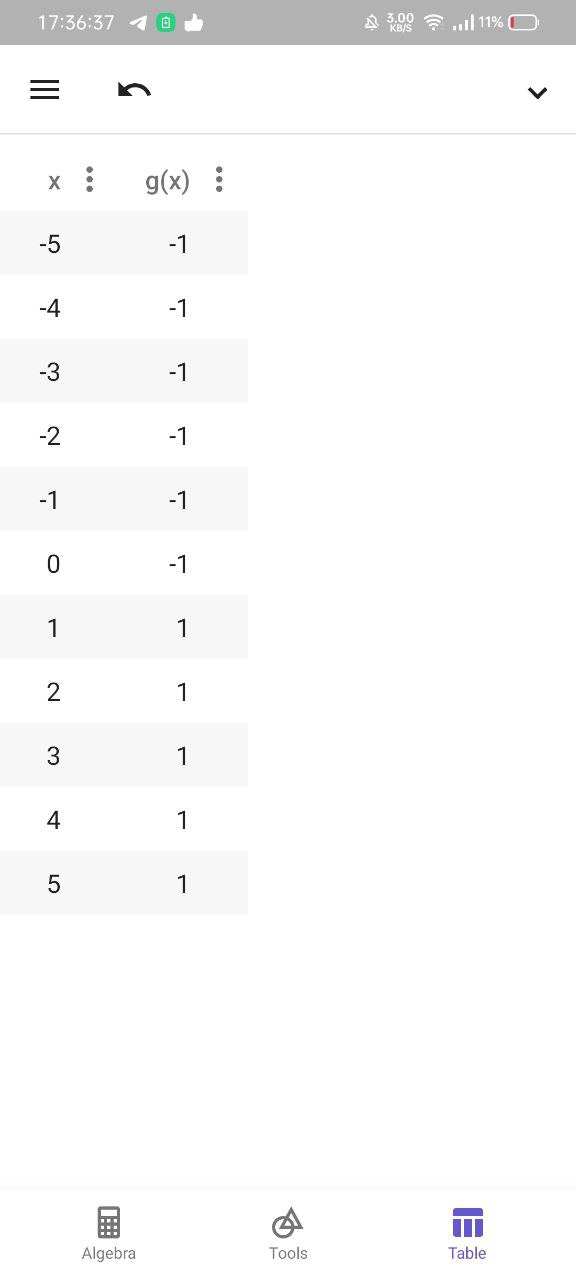


Рис 1.4 Результати табуляції

Як бачимо з рис. 1.4, функція змінює знак на відрізку [0; 1], що співпадає з результатом графічного методу, тому корінь шукатимемо там.

**Етап уточнення коренів**

Для подальшої роботи я використовуватиму функції f та d2f , які відповідають функції із завдання та її другій похідній. Їх код має наступний вигляд:

double f(double x)

{

    return x\*x\*x + 3\*x\*x + 12\*x - 3;

}

double d2f(double x)

{

    return 6\*x+6;

}

**Метод дихотомії(бісекції):**

Код функції:

double findDyhotomea(double start, double end, double precision, int& iterations)

{

    // Визначаємо, чи корені наявні.

    if(f(start)\*f(end) >= 0) throw "No roots!";

    else

    {

        iterations = 0;

        // допоки ми не досягнемо потрібної точності, або, що не варто виключати, випадково потрапимо в корінь...

        while(fabs(start-end) > precision && f(end) != 0)

        {

            double midpoint = (end+start)/2; // Вираховуємо середню точку.

            // Визначаємо, з яким кінцем у неї співпадає знак функції і робимо відповідне присвоєння.

            if(f(start)\*f(midpoint) > 0) start = midpoint;

            else end = midpoint;

            iterations++;

        }

        // Повертаємо відповідь

        return end;

    }

}

**Метод хорд:**

Код функції:

double findHordes(double start, double end, double precision, int& iterations)

{

    // Перевіряємо наявність коренів

    if(f(start)\*f(end) >= 0) throw "No roots!";

    // Обираємо рухомий кінець

    else if(f(start)\*d2f(start) > 0) // Рухаємо кінець відрізку

    {

        iterations = 0;

        double end\_prev = 0;

        // Допоки різниця попереднього і насупного члену не стане меншою за задану...

        do

        {

            // Зберігаємо попереднє значення кінця відрізку

            end\_prev = end;

            // Пересуваємо кінець у точку перетину хорди з віссю абсцис

            end -= f(end)\*(end-start)/(f(end)-f(start));

            iterations++;

        } while(fabs(end-end\_prev) > precision);

        // Повертаємо відповідь.

        return end;

    }

    else // Рухаємо початок відрізку

    {

        iterations = 0;

        double start\_prev = 0;

        // Допоки різниця попереднього і насупного члену не стане меншою за задану...

        do

        {

            // Зберігаємо попереднє значення почтаку відрізку

            start\_prev = start;

            // Пересуваємо початок у точку перетину хорди з віссю абсцис

            start -= f(start) \* (end-start)/(f(end) - f(start));

            iterations++;

        } while (fabs(start-start\_prev) > precision);

        // Повертаємо відповідь.

        return start;

    }

}

**Результат** виконання цих функцій(точність ):

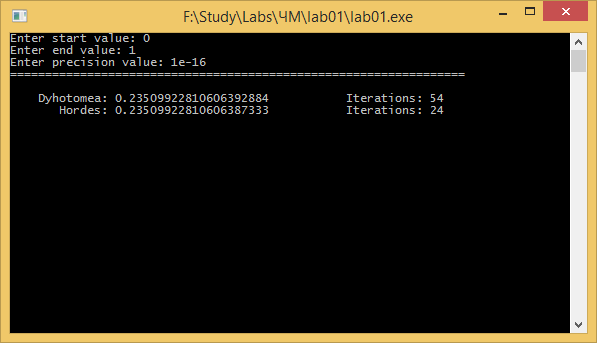


Рис 1.5. Результати виконання програми.

**Аналіз результатів:**

Як бачимо з рисунку 1.4, ми отримали однакові значення кореня рівняння обома методами аж до 15 значущої цифри включно. Це свідчить про те, що результат з високою вірогідністю є вірним. Перевіримо його за допомогою сервісу WolframAlpha:



Отже, пошук коренів заданого рівняння було виконано правильно та ефективно. Також, можна помітити, що для заданого випадку метод хорд відпрацював швидше, ніж метод дихотомії.

**Висновок:**

Я ознайомився на практиці з методами знаходження коренів нелінійних рівнянь та розробив функції для уточнення коренів на заданих проміжках на основі отриманих знань. Використовуючи графічний та аналітичний метод, а також табулювання функції, я встановив, що корінь рівняння знаходиться на відрізку [0; 1]. Потім, використавши метод хорд та бісекції я уточнив локацію кореня і вирахував його наближене значення: .