Чесной Владислав Сергійович

КН-45-5/1

Варіант 18

Лабораторна робота 1-2

**Тема:** Чисельне розв’язання нелінійних рівнянь.

**Мета:** Навчитися розв’язувати задачі одновимірної оптимізації за допомогою

методу простої ітерації та методу Ньютона.

**Завдання**

Розв’язати нелінійне рівняння e-2x-2x+1=0 з точністю ϵ=0,001.

1. Відокремити корені графічно і аналітично. Розв’язати за допомогою методу простої ітерації або методу Ньютона.

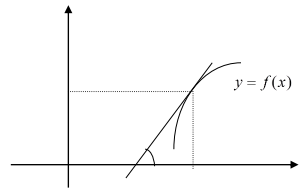
2. Порівняти аналітичні розрахунки з результатами чисельного розв’язання за допомогою програмного пакета Maple або MathCad.

3. Розв’язати нелінійне рівняння за допомогою онлайн-калькулятора https://math.semestr.ru/optim/newton.php.

4. Виконати програмну реалізацію методу простої ітерації або методу Ньютона.

**Теоретичні відомості**

Метод Ньютона (метод дотичних) для наближеного розв’язку рівняння f(x)=0 полягає в побудові ітераційної послідовності {xn}, n = 0,1,2,…, що збігається до кореня рівняння на відрізку [a,b] його локалізації.



На рисунку зображено спосіб отримання першого наближення за методом дотичних: x1 є точка перетину дотичної, проведеної до кривої в точці з координатами (x0,f(x0)). З прямокутного трикутника, гострий кут якого α, маємо

, звідки х1=х0 - .

Достатні умови збіжності такі.

Нехай f(x) – визначена і двічі диференційована на [a,b], причому похідні f'(x), f''(x) зберігають знак на [a,b]. Тоді, виходячи з початкового наближення x₀ ∈ [a,b], що задовольняє нерівність f(x₀)f''(x₀)>0, ітераційна послідовність

xₙ₊₁ = xₙ – f(xₙ)/f'(xₙ), n = 0,1,2,...

збігається до єдиного на [a,b] розв’язку ξ рівняння f(x)=0.

Аналітично наближені корені рівняння f(x)=0 можна знайти за таким алгоритмом:

1. Відокремлюємо корені графічно, тобто встановлюємо інтервали, що містять один корінь рівняння.

2. На кожному інтервалі [a,b] вибираємо початкове наближення кореня x₀.

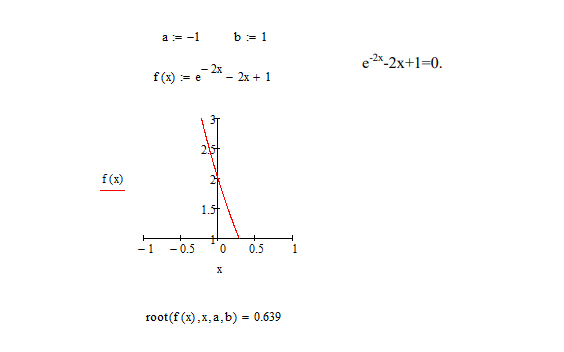
3. Перевіряємо достатні умови збіжності методу Ньютона: функція f(x) визначена та двічі диференційована на [a,b], причому похідні f'(x), f''(x) зберігають знак на [a,b]; відрізку [a,b] належить тільки один корінь, тобто f(a)·f(b)<0; початкове наближення задовольняє умову f(x₀)·f''(x₀)>0.

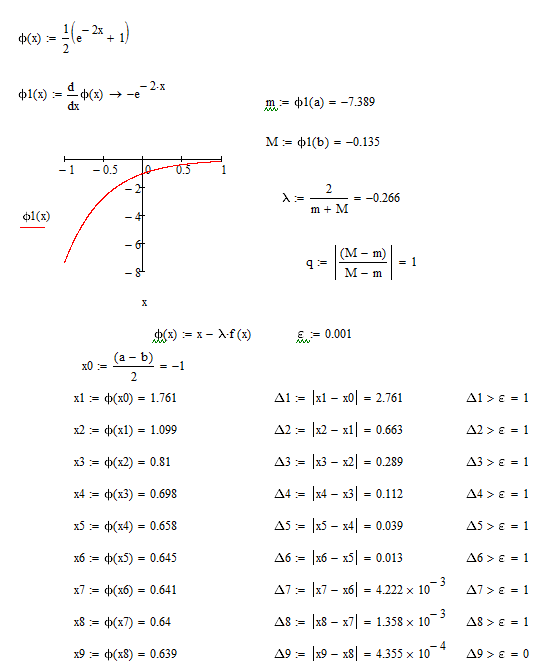
4. Знаходимо корінь рівняння на [a,b] із заданою точністю, будуючи ітераційну послідовність

xₙ₊₁ = xₙ – f(xₙ)/f'(xₙ).1.

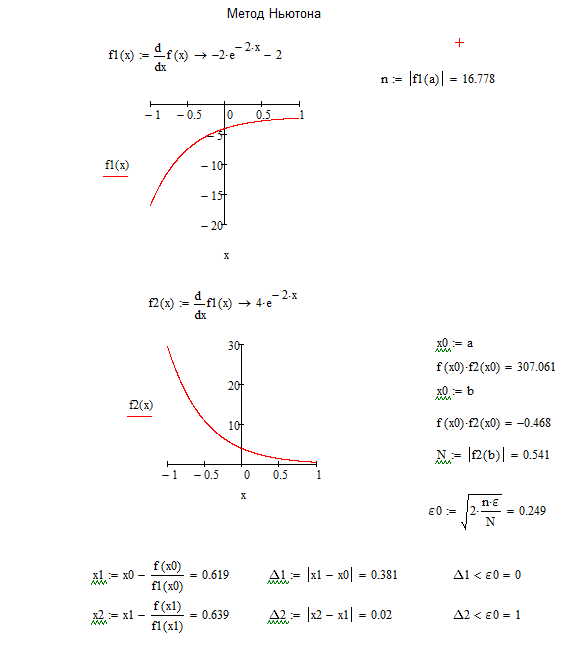
**Чисельне розв’язання за допомогою програмного пакета Mathcad**

Метод простих ітерацій





Метод Ньютона



**Програмна реалізація**

Лістинг програми

#include <iostream>

#include <Windows.h>

#include <cmath>

#include <stdexcept>

using namespace std;

#define MAXIMUM\_ITERATIONS 1000

double equation\_function(double x) {

return exp(-2.0 \* x) - 2.0 \* x + 1.0;

}

double equation\_derivative(double x) {

return -2.0 \* exp(-2.0 \* x) - 2.0;

}

struct SolutionData {

double solution\_value;

int iteration\_count;

};

double compute\_iteration\_coefficient(double start\_point, double end\_point) {

double derivative\_at\_start = equation\_derivative(start\_point);

double derivative\_at\_end = equation\_derivative(end\_point);

return 2.0 / (derivative\_at\_start + derivative\_at\_end);

}

SolutionData iterative\_method(double initial\_guess, double boundary, double precision) {

int step\_counter = 0;

double current\_approximation = initial\_guess;

double previous\_approximation;

double iteration\_coeff = compute\_iteration\_coefficient(initial\_guess, boundary);

while (true) {

previous\_approximation = current\_approximation;

current\_approximation = previous\_approximation - iteration\_coeff \* equation\_function(previous\_approximation);

step\_counter++;

if (step\_counter > MAXIMUM\_ITERATIONS) {

throw runtime\_error("Перевищено максимальну кількість ітерацій!");

}

if (fabs(current\_approximation - previous\_approximation) <= precision) {

break;

}

}

SolutionData result;

result.solution\_value = current\_approximation;

result.iteration\_count = step\_counter;

return result;

}

SolutionData newton\_raphson\_method(double initial\_guess, double precision) {

int step\_counter = 0;

double current\_approximation = initial\_guess;

double previous\_approximation;

while (true) {

previous\_approximation = current\_approximation;

double function\_value = equation\_function(previous\_approximation);

double derivative\_value = equation\_derivative(previous\_approximation);

current\_approximation = previous\_approximation - function\_value / derivative\_value;

step\_counter++;

if (step\_counter > MAXIMUM\_ITERATIONS) {

throw runtime\_error("Перевищено максимальну кількість ітерацій!");

}

if (fabs(current\_approximation - previous\_approximation) <= precision) {

break;

}

}

SolutionData result;

result.solution\_value = current\_approximation;

result.iteration\_count = step\_counter;

return result;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "=== РОЗВ'ЯЗАННЯ НЕЛІНІЙНОГО РІВНЯННЯ ===" << endl;

cout << "Рівняння: f(x) = e^(-2x) - 2x + 1 = 0" << endl;

cout << endl;

cout << "Доступні методи розв'язання:" << endl;

cout << "[1] Метод послідовних наближень" << endl;

cout << "[2] Метод Ньютона-Рафсона" << endl;

cout << endl;

int method\_selection;

cout << "Оберіть метод (1 або 2): ";

cin >> method\_selection;

double starting\_point, accuracy;

cout << "Початкове наближення (x0): ";

cin >> starting\_point;

cout << "Точність обчислень (ε): ";

cin >> accuracy;

try {

SolutionData computation\_result;

switch (method\_selection) {

case 1: {

double upper\_bound = 1.0;

computation\_result = iterative\_method(starting\_point, upper\_bound, accuracy);

cout << endl << "=== РЕЗУЛЬТАТ (Метод послідовних наближень) ===" << endl;

break;

}

case 2: {

computation\_result = newton\_raphson\_method(starting\_point, accuracy);

cout << endl << "=== РЕЗУЛЬТАТ (Метод Ньютона-Рафсона) ===" << endl;

break;

}

default: {

cout << "Помилка: Невірний номер методу!" << endl;

return EXIT\_FAILURE;

}

}

cout << "Корінь рівняння: " << computation\_result.solution\_value << endl;

cout << "Кількість кроків: " << computation\_result.iteration\_count << endl;

cout << "Перевірка: f(" << computation\_result.solution\_value << ") = "

<< equation\_function(computation\_result.solution\_value) << endl;

}

catch (const exception& error\_message) {

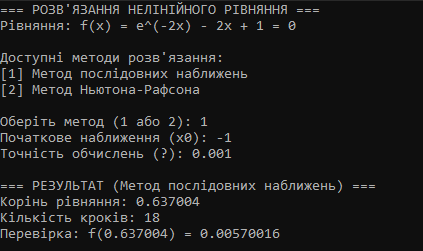
cout << "Помилка виконання: " << error\_message.what() << endl;

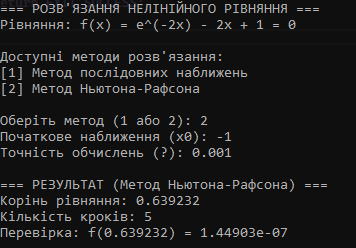
return EXIT\_FAILURE;

}

return EXIT\_SUCCESS;

}





Висновки:

Під час виконання даної лабораторної роботи для заданого моїм варіантом рівняння при похібці 0.001 я отримав наступні наближенні значення коренів:

1) Root (Mathcad) = 0.639

2) Метод простих ітерацій (Mathcad) = 0.639

3) Метод Ньютона (Mathcad) = 0.639

5) Метод простих ітерацій (Visual Studio) = 0.639232

6) Метод Ньютона (Visual Studio) = 0.639232

Всі результати показали практично ідентичний результат обчисленого кореню.