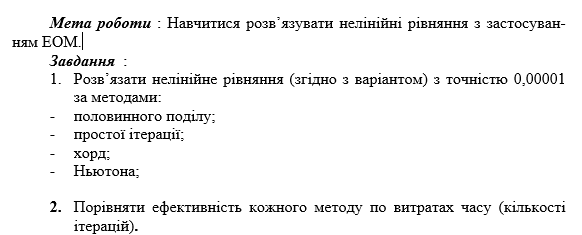
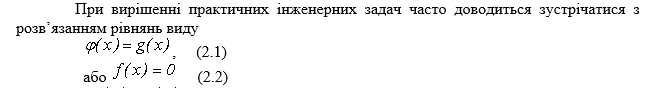
**Лабораторна робота №2**

Розв’язання нелінійних рівнянь за чисельними методами

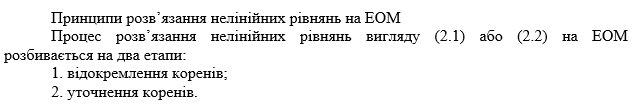


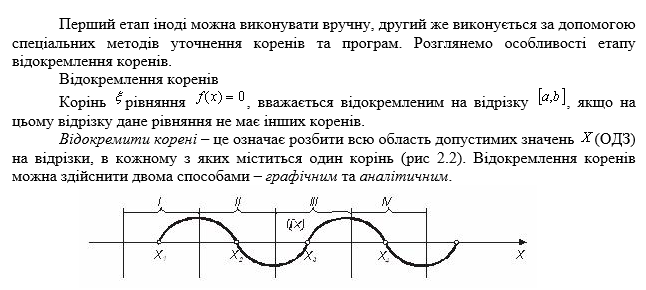


Короткі теоретичні відомості

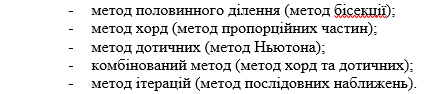








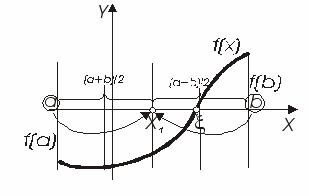




*Метод половинного ділення*

*Постановка задачі*

Нехай маємо рівняння http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image132.png, де http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image133.png– неперервна, монотонна нелінійна функція, яка має на відрізку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image135.pngєдиний корінь http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image137.png, тобто добуток http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image139.png, причому http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image141.png, де http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image143.png– задана похибка обчислень. Потрібно знайти значення кореня http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image145.pngз заданою похибкою http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image146.png(рис. 2.9).



1. На відрізку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image156.pngвибираємо точку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image157.png, яка розділяє його на два рівних відрізки http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image159.pngі http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image161.png, довжина яких рівна і знаходиться за формулою

http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image163.png

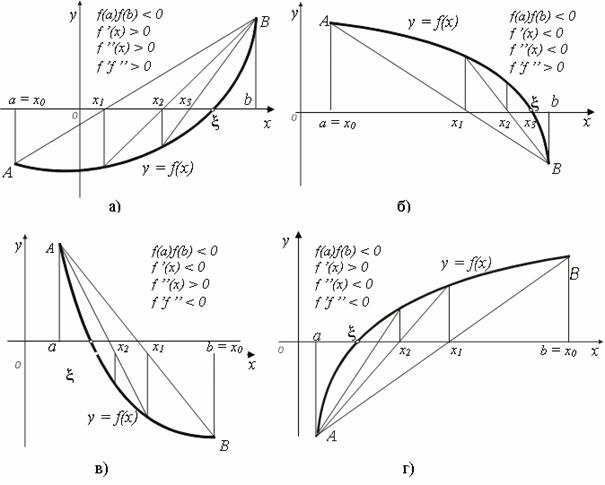
2. Перевіряємо чи http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image165.png, якщо так, то http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image167.png– точний корінь початкового рівняння і переходимо до пункту 6.

3. У випадку, коли http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image169.png, то з двох отриманих відрізків http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image171.pngі http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image173.pngвибираємо той, на кінцях якого функція http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image175.pngприймає значення протилежних знаків, тобто, якщо http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image177.png, тоді залишаємо відрізок http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image179.pngі точку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image181.pngпереносимо в точку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image183.png(http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image185.png); якщоhttp://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image187.png, то залишаємо відрізок http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image189.pngі переносимо точку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image191.pngв точку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image193.png(http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image194.png) і переходимо до пункту 1.

4. Процес ділення відрізка навпіл виконується доти, поки на якомусь етапі, або середина відрізка буде коренем, або буде виконана умова закінчення ітераційного процесу: http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image196.png (http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image198.png).

*Метод хорд*

*Суть методу* хорд полягає в тому, що на достатньо малому відрізку http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image216.pngдуга функції http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image217.pngзамінюється хордою *ab*, яка її стягує. За наближене значення кореня приймається точка *х1* перетину хорди з віссю http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image218.png

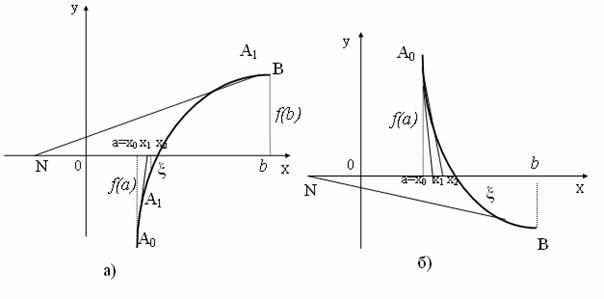


http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image241.png

*Правило 1*. Нерухомим кінцем відрізка є той, для якого знак функції співпадає із знаком другої похідної. Якщо http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image260.png, то нерухомим є кінець http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image262.png, а всі наближення до кореня http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image263.pngлежать зі сторони кінця http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image264.png. Якщо http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image266.png, то нерухомим є кінець http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image268.png, а всі наближення до кореня http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image269.pngлежать зі сторони кінця http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image270.png

*Правило 2.* Якщо добуток першої на другу похідну функції http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image271.pngбільший за нуль: http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image272.png, то рухомий кінець http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image274.png; якщо добуток першої на другу похідну менший за нуль: http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image275.png, то рухомий кінець http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image277.png.

*Метод Ньютона (метод дотичних)*



http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image323.png

При виборі початкового наближення кореня необхідно використовувати наступне правило: *за початкову точку слід вибирати той кінець відрізка [a, b], в якому знак функції співпадає зі знаком другої похідної.*

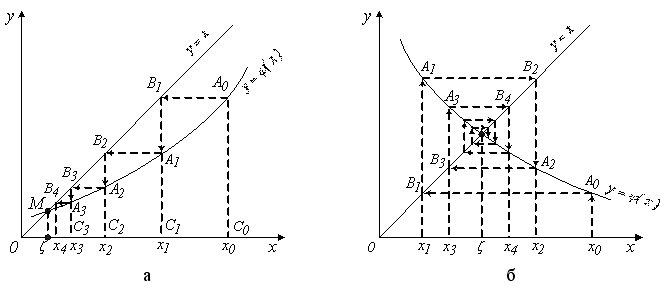
*Метод ітерацій (метод послідовних наближень)*

Суть методу полягає у заміні початкового рівняння

http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image404.png

еквівалентним йому рівнянням

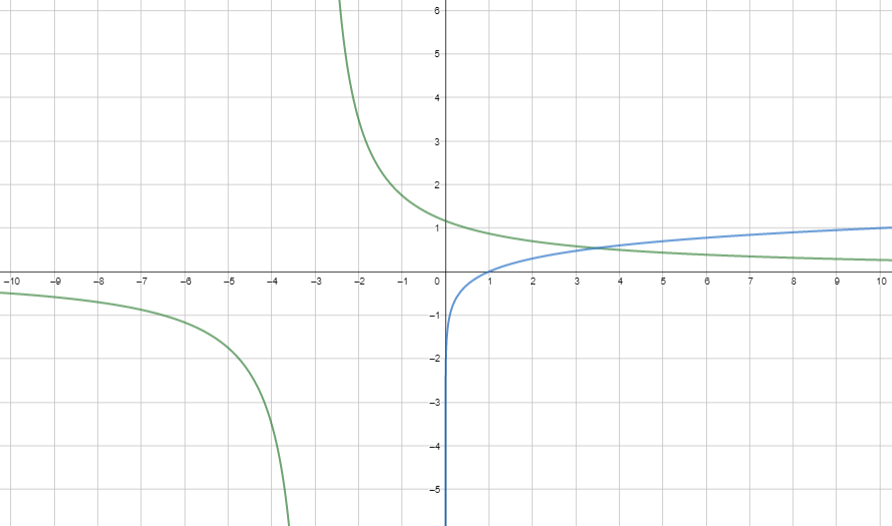
http://posibnyky.vntu.edu.ua/met/m1_t1_lecture4_src/m1_t1_lecture4_image405.png,



**Хід роботи**

1. Схема розрахунків

Відокремлення коренів (графічнім методом):



Рівняння має 1 корінь, що знаходиться на відрізку: [3; 4].

a = 3, b = 4

Перевірка правильності інтервалу:

1. На відрізку [a; b] функція нерозривна та зростаюча, тому похідна не змінює знак.

Визначення g(x) для методу ітерацій:

1. Текст програми

#include <iostream>

#include <string>

#include <cmath>

const double DEF\_EPSILON = 0.00001;

int iterativeCounter = 0;

// Возвращает значение функции в точке х

double getF(double x) {

return log10(x) - 7 / (2 \* x + 6);

}

// Возвращает значение производной в точке х

double getFPrime(double x) {

return 1 / (x \* log(10)) + 14 / ((2 \* x + 6) \* (2 \* x + 6));

}

// Возвращает значение производной в точке х

double getFDoublePrime(double x) {

double p = (2 \* x + 6);

return -1/(x\*x\*log(10)) - 56 / (p \* p \* p);

}

// Преобразованная функция F для метода итераций

double getG(double x) {

return x - 4.13 \* getF(x);

}

double bisection(double l, double r, double eps = DEF\_EPSILON);

double bisection\_recursive(double l, double r, double eps = DEF\_EPSILON);

double falsePosition(double l, double r, double eps = DEF\_EPSILON);

double falsePosition\_recursive(double l, double r, double eps = DEF\_EPSILON);

double newtonRaphson(double l, double r, double eps = DEF\_EPSILON);

double newtonRaphson\_recursive(double l, double r, double eps = DEF\_EPSILON);

double iterative(double x0, double eps = DEF\_EPSILON);

double iterative\_recursive(double x0, double eps = DEF\_EPSILON);

void processRes(std::string method, double res);

void checkRoots(double l, double r);

void clearIterativeCounter();

void printHello();

void run();

template <typename T>

T prompt(const char label[]);

int main() {

printHello();

while (true) {

try {

run();

}

catch (std::runtime\_error err) {

std::cout << err.what() << std::endl;

}

if (prompt<std::string>("Repeat (0 - no): ") == "0") {

break;

}

std::cout << "\n- - - - - - - - - - - - - - - -\n\n";

}

return 0;

}

void run() {

double eps = prompt<double>("Epsilon (if 0 default - 0.00001): ") / 10;

if (eps <= 0) eps = DEF\_EPSILON;

std::cout << "Enter [a, b]:" << std::endl;

double l = prompt<double>("a: ");

double r = prompt<double>("b: ");

checkRoots(l, r);

std::cout << "Results: " << std::endl << std::endl;

processRes("Bisection Method (iterative)", bisection(l, r, eps));

processRes("Bisection Method (recursive)", bisection\_recursive(l, r, eps));

std::cout << std::endl;

processRes("False-Position Method (iterative)", falsePosition(l, r, eps));

processRes("False-Position Method (recursive)", falsePosition\_recursive(l, r, eps));

std::cout << std::endl;

processRes("NewtonRaphson Method (iterative)", newtonRaphson(l, r, eps));

processRes("NewtonRaphson Method (recursive)", newtonRaphson\_recursive(l, r, eps));

std::cout << std::endl;

processRes("Iterative Method (iterative)", iterative((l + r) / 2, eps));

processRes("Iterative Method (recursive)", iterative\_recursive((l + r) / 2, eps));

std::cout << std::endl;

}

void processRes(std::string method, double res) {

std::cout << method << ": " << res << "; Iterations: " << iterativeCounter << std::endl;

clearIterativeCounter();

}

void printHello() {

std::cout << " \* \* \* Numerical Methods - Root Finding \* \* \* " << std::endl << std::endl;

}

double bisection(double l, double r, double eps) {

checkRoots(l, r);

double m, my;

while (true) {

iterativeCounter++;

m = (l + r) / 2;

my = getF(m);

if (abs(my) <= eps || abs(r - l) < eps) {

break;

}

if (my \* getF(l) < 0) r = m;

else l = m;

}

return m;

}

double bisection\_recursive(double l, double r, double eps) {

checkRoots(l, r);

iterativeCounter++;

double m = (l + r) / 2;

double my = getF(m);

if (abs(my) < eps || abs(r - l) < eps) return m;

if (my \* getF(l) < 0) return bisection\_recursive(l, m);

else return bisection\_recursive(m, r);

}

double falsePosition(double l, double r, double eps) {

checkRoots(l, r);

double ly, ry, mx, my;

while (true) {

iterativeCounter++;

ly = getF(l);

ry = getF(r);

mx = l - ly \* (r - l) / (ry - ly);

my = getF(mx);

if (abs(my) <= eps || abs(r - l) < eps) {

break;

}

if (ly \* my < 0) r = mx;

else l = mx;

}

return mx;

}

double falsePosition\_recursive(double l, double r, double eps) {

checkRoots(l, r);

iterativeCounter++;

double ly = getF(l);

double ry = getF(r);

double mx = l - ly \* (r - l) / (ry - ly);

double my = getF(mx);

if (abs(my) <= eps || abs(r - l) < eps) {

return mx;

}

if (ly \* my < 0) return falsePosition\_recursive(l, mx);

else return falsePosition\_recursive(mx, r);

}

double newtonRaphson(double l, double r, double eps) {

checkRoots(l, r);

double x0 = (getF(l) \* getFDoublePrime(l) > 0) ? l : r;

while (true) {

iterativeCounter++;

double x = x0 - getF(x0) / getFPrime(x0);

if (abs(x - x0) < eps) break;

x0 = x;

}

return x0;

}

double newtonRaphson\_recursive(double l, double r, double eps) {

iterativeCounter++;

if (getF(r) \* getFDoublePrime(r) > 0) std::swap(l, r);

double x = l - getF(l) / getFPrime(l);

if (abs(x - l) < eps) return x;

return newtonRaphson\_recursive(x, r, eps);

}

double iterative(double x0, double eps) {

while (true) {

iterativeCounter++;

double x1 = getG(x0);

if (abs(x1 - x0) <= eps) break;

x0 = x1;

}

return x0;

}

double iterative\_recursive(double x0, double eps) {

iterativeCounter++;

double x1 = getG(x0);

if (abs(x1 - x0) <= eps) return x0;

return iterative\_recursive(x1);

}

void checkRoots(double l, double r) {

if (getF(l) \* getF(r) < 0) return;

throw std::runtime\_error("No roots on this range");

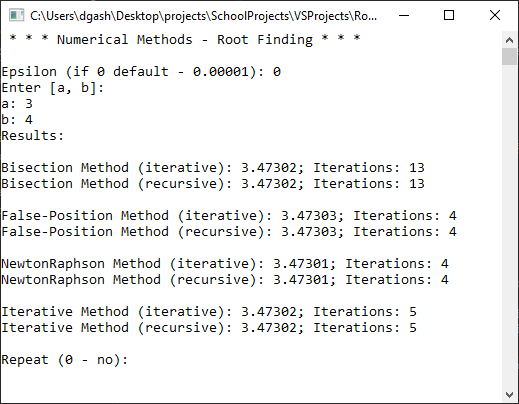
}

void clearIterativeCounter() {

iterativeCounter = 0;

}

1. Результат виконання



**Висновок:** на цій лабораторній роботі навчився розв’язувати нелінійні рівняння на ЕОМ. Проаналізувавши результати видно, що найповільніший метод – це метод половинного ділення (bisection method). Інші методи показали близькі результати, але метод Ньютона-Рафсона показав себе найкраще.