Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 3

з дисципліни «Чисельні методи»

Тема: «Метод дихотомії»

Виконала:

студентка 3 курсу

групи КС-32

Дібцева Анна Миколаївна

Перевірила: доцент

Чуб Ольга Ігорівна

Харків – 2021

# ХОД РАБОТЫ

Целью данной работы является исследование алгоритма метода дихотомии для решения нелинейных уравнений и его программная реализация.

Входными данными являются количество коэффициентов при х, коэффициенты при х, диапазон на котором у функции имеются корни, точность вычислений.

В данном методе решения уравнений сначала отделяются корни (участки, где присутствует экстремум), а потом происходит уточнение корней – последовательное увеличение точности до заданной с нахождением корня.

Точность считается достигнутой, когда модуль разности двух последовательных приближений корня меньше удвоенного значения заданной точности.

Программа скомпилирована в исполняемый файл.

Результат работы программы приведен ниже.

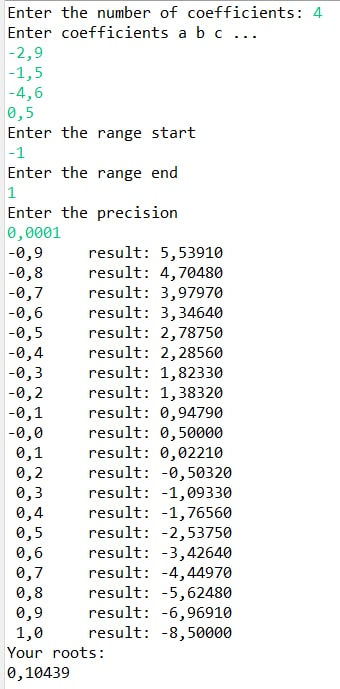


Рисунок 1 – Результат выполнения программы

# ВЫВОДЫ

Таким образом, в ходе выполнения данной лабораторной работы был рассмотрен алгоритм дихотомии для решения нелинейных уравнений и написана программа, которая позволяет решать степенные уравнения используя метод.

# ЛИСТИНГ ПРОГРАММЫ

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.Scanner;

import static java.lang.Math.\*;

public class DichotomyMain {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

double startIntervalPoint;

double endIntervalPoint;

double precision;

ArrayList<Double> coeff = new ArrayList<Double>();

System.out.print("Enter the number of coefficients: ");

int coefAmount = 0; //количество коэффициентов

while(true){

try {

coefAmount = scan.nextInt();

break;

} catch (Exception e) {

scan.next();

System.out.print("Input error! Try again: ");

}

}

if (coefAmount <= 0) {

System.out.println("error in coefficients amount");

System.exit(-1);

}

//ввод коэффициентов

System.out.println("Enter coefficients a b c ...");

for (int i = 1; i <= coefAmount; i++) {

coeff.add(inputValue(scan));

}

System.out.println("Enter the range start");

startIntervalPoint = inputValue(scan);

System.out.println("Enter the range end");

endIntervalPoint = inputValue(scan);

System.out.println("Enter the precision");

precision = inputValue(scan);

ArrayList<Double> equationRoots = dichotomyMethod(startIntervalPoint, endIntervalPoint, precision, coeff);

if (equationRoots.isEmpty()) {

System.out.println("No roots");

System.exit(0);

}

printRoots(equationRoots);

scan.close();

}

private static double inputValue(Scanner scan) {

double value;

while(true){

try {

value = scan.nextDouble();

break;

} catch (Exception e) {

scan.next();

System.out.print("Input error! Try again: ");

}

}

return value;

}

private static void printRoots(ArrayList<Double> roots) {

if (roots.isEmpty()) {

System.out.println("No roots");

System.exit(0);

}

System.out.println("Your roots:");

for (Double root : roots) {

System.out.printf("%4.5f\n", root);

}

}

/\*\*\*

\* Метод дихотомії

\* @param startIntervalPoint начало рассматриваемого интервала

\* @param endIntervalPoint конец рассматриваемого интервала

\*/

public static ArrayList<Double> dichotomyMethod(double startIntervalPoint, double endIntervalPoint, double precision, ArrayList<Double> coefficients) {

if (startIntervalPoint > endIntervalPoint) {

System.out.println("Error in diapason");

System.exit(-1);

}

ArrayList<Double> equationRoots = new ArrayList<Double>();

double prevFuncVal = substituteInTheEquation(startIntervalPoint, coefficients);

if(prevFuncVal == 0) equationRoots.add(startIntervalPoint);

for (double i = startIntervalPoint + 0.1; i <= endIntervalPoint; ) {

System.out.printf("%4.1f", i);

double newFuncVal = substituteInTheEquation(i, coefficients);

System.out.printf("\t result: %4.5f\n", newFuncVal);

if(newFuncVal == 0) {

equationRoots.add(i);

prevFuncVal = substituteInTheEquation(i+0.1, coefficients);

i += 0.2;

continue;

}

else if(signum(newFuncVal) != signum(prevFuncVal)) {

equationRoots.add(rootFinding(i - 0.1, i, precision, coefficients));

}

prevFuncVal = newFuncVal;

i += 0.1;

}

return equationRoots;

}

/\*\*\*

\* метод для подстановки x в уровнение

\* @param x

\* @return f(x)

\*/

private static double substituteInTheEquation (double x, ArrayList<Double> coefficients) {

double result = 0;

for (int i = 1; i <= coefficients.size(); i++) {

result += coefficients.get(i-1) \* pow(x, coefficients.size() - i);

}

return result;

}

/\*\*\*

\* нахождение корня в диапазоне

\* @param lowerVal начало диапазона

\* @param greaterVal конец диапазона

\* @return

\*/

private static double rootFinding(double lowerX, double greaterX, double precision, ArrayList<Double> coefficients) {

if(lowerX > greaterX) {

System.out.println("Error in root finding");

System.exit(-1);

}

double middleX;

middleX = (lowerX + greaterX) / 2;

if (substituteInTheEquation(middleX, coefficients) == 0)

return middleX;

//если знаки значений функций при этих корнях не равны, то корень между ними

if(signum(substituteInTheEquation(lowerX, coefficients)) != signum(substituteInTheEquation(middleX, coefficients))) {

if (abs(middleX - lowerX) < 2 \* precision)

return ((lowerX + middleX) / 2);

return rootFinding(lowerX, middleX, precision, coefficients);

}

else {

if (abs(greaterX - middleX) < 2 \* precision)

return ((middleX + greaterX) / 2);

return rootFinding(middleX, greaterX, precision, coefficients);

}

}

}