Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 4
З дисципліни «Алгоритми та методи обчислень»

На тему «Розв’язання нелінійних рівнянь на комп’ютері»

Виконав:
студент 2 курсу ФІОТ
групи ІВ-71
Мазан Я. В.
Залікова – 7109

Перевірив:
ст.вик. Порєв В. М.

Київ – 2019

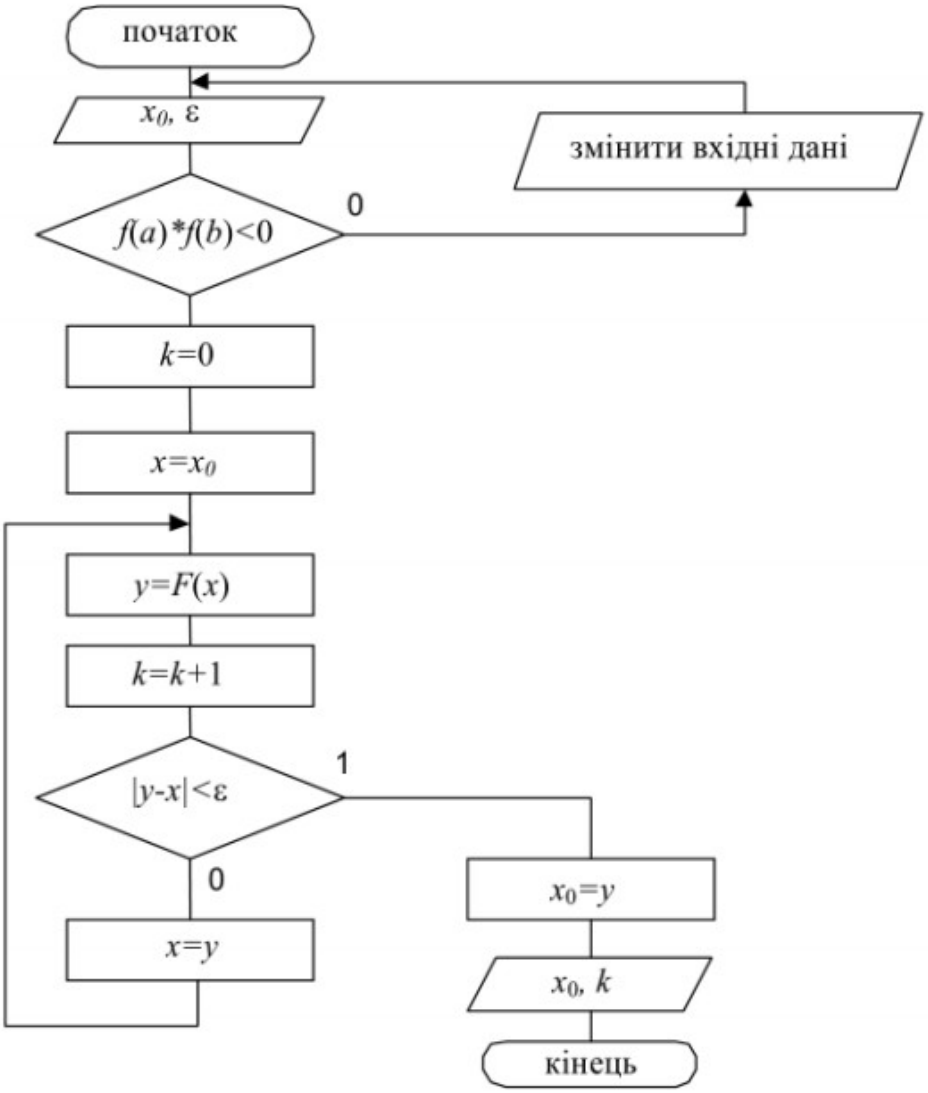
Мета:
Метою даного заняття є ознайомлення з методиками та вивчення різних алгоритмів розв’язання нелінійних рівнянь на комп’ютері

Завдання:
Закріплення знань студентів при вирішенні практичних завдань з розв’язування нелінійних рівнянь. Оволодіння методами і практичними навичками розв’язування нелінійних рівнянь на комп’ютері. Набуття умінь і навичок при програмуванні та налагодженні програм для розв’язування нелінійних рівнянь на комп'ютері.

Індивідуальне завдання:

Метод	**№ варіанту**	**Рівняння**	**Примітка**
Метод дотичних	9	$x^3 – x - 3 = 0$	1.672

Блок-схема методу:



Код програми:

```
package com.labworks.amc_lab4
import android.graphics.Color
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Toast
import com.jjoe64.graphview.LegendRenderer
import com.jjoe64.graphview.series.*
import kotlinx.android.synthetic.main.activity_main.*
import android.support.design.widget.Snackbar
import java.text.DecimalFormat
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
    fun onClick(view: View) {
        val epsilon: Double
        val a: Double
        val b: Double
        try {
            epsilon = epsilonInput.text.toString().toDouble()
            a = minInput.text.toString().toDouble()
            b = maxInput.text.toString().toDouble()
        } catch (e: Exception) {
            Toast.makeText(this, "Input correct values to calculate roots!", Toast.LENGTH_SHORT).show()
            return
        }
        plot(view, a,b)
        val rootVal: Double
        try {
            rootVal = tangentMethod(a,b,epsilon)
        } catch (e: RuntimeException) {
            Toast.makeText(this, "Root value is outside the input range!", Toast.LENGTH_SHORT).show()
            return
        }
        val foundX = PointsGraphSeries<DataPoint>(arrayOf(DataPoint(rootVal, function(rootVal))))
        foundX.setColor(Color.GREEN)
        foundX.setShape(PointsGraphSeries.Shape.POINT)
        foundX.setTitle("X")
        foundX.setOnDataPointTapListener( object: OnDataPointTapListener {
            val pattern: String = "#.####"
            val format = DecimalFormat(pattern)
            override fun onTap(series: Series<DataPointInterface>, dataPointInterface: DataPointInterface) {
                format.format(rootVal)
                Snackbar.make(view, "x = ${format.format(rootVal)}\ny = ${format.format(function(rootVal))}", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show()
            }
        })
        graph.addSeries(foundX)
    }
    fun plot(view: View, a: Double, b: Double) {
        graph.removeAllSeries()
        graph.getViewport().setMinX(a)
        graph.getViewport().setMaxX(b)
        val x_delta = (b-a)/500
        val x_vals = List(500, {i -> a+ i*x_delta})
        val y_vals = x_vals.map({x -> function(x) })
        val function = LineGraphSeries<DataPoint>(Array<DataPoint>(500, {i -> DataPoint(x_vals[i],y_vals[i])}))
        function.setColor(Color.RED)
        function.setThickness(3)
        function.setTitle("Графік функції");
        graph.addSeries(function)
        graph.getLegendRenderer().setVisible(true)
        graph.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.TOP)
    }
    companion object Calculate {
        fun function(x: Double): Double {
            return Math.pow(x, 3.0) - x - 3
        }
        fun derivative(x: Double): Double {
            return 3 * Math.pow(x, 2.0) - 1
        }
        fun secondDerivative(x: Double): Double {
            return 6 * x
        }
        fun tangentMethod(a: Double, b: Double, epsilon: Double): Double {
```

```
        val x0 = if (function(b) * secondDerivative(b) > 0) b else a
        val x1 = x0 - function(x0) / derivative(x0)
        val x2 = x1 - function(x1) / derivative(x1)
        val values: MutableMap<String, Double> = mutableMapOf("Xi-1" to x1, "Xi" to x2)
        while (Math.abs(values["Xi-1"]!! - values["Xi"]!!) > epsilon) {
            val xi = values["Xi"]!! - function(values["Xi"]!!) / derivative(values["Xi"]!!)
            values["Xi-1"] = values["Xi"]!!
            values["Xi"] = xi
        }
        if (a <= values["Xi"]!! && b >= values["Xi"]!!)
            return values["Xi"]!!
        else
            throw RuntimeException("Equation's root doesn't belong to the range!")
    }
  }
}
```

Результати виконання програми:

Висновок:

Створена мною програма знаходить корені нелінійних рівнянь з вказаною точністю з указанням приближеного значення кореня. Я реалізував метод ітерацій класичний і той де ітерації реалізовані за допомогою формули дотичних Ньютона. В моїй програмі функції можна вводити при виконанні.