Міністерство освіти і науки України  
НТУУ «КПІ ім. І. Сікорського»

Кафедра цифрових технологій в енергетиці

Лабораторна робота № 3

«Комп’ютерне моделювання»

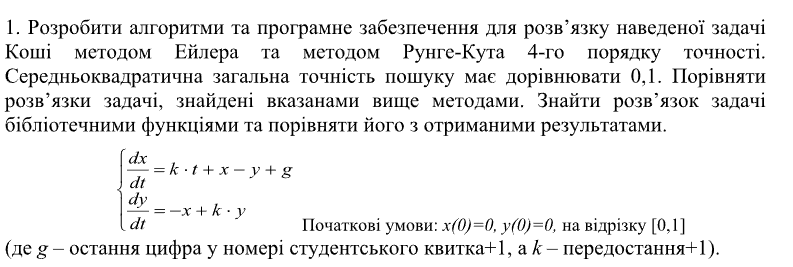
Варіант №25

Тема: «Комп’ютерне моделювання нестаціонарних процесів»

Виконав:  
студент 3-го курсу НН ІАТЕ  
групи ТР-12  
Руденко Владислав Ігорович

Київ – 2023

**Завдання**

*****Коефіцієнти згідно до варіанту: g-6, k-4*

**Результати виконання**

**Завдання 1:**

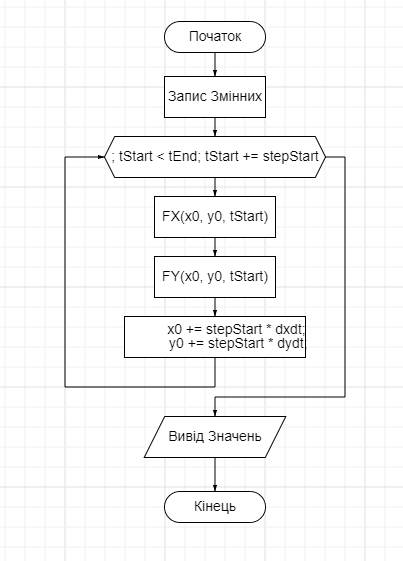


Рисунок Блок Схема Методу Ейлера

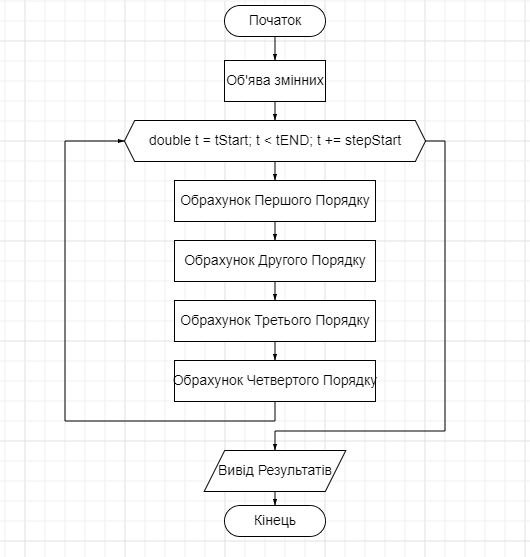


Рисунок Блок-Схема Методу Рунге Кутта 4го Порядку

**Завдання 2:**

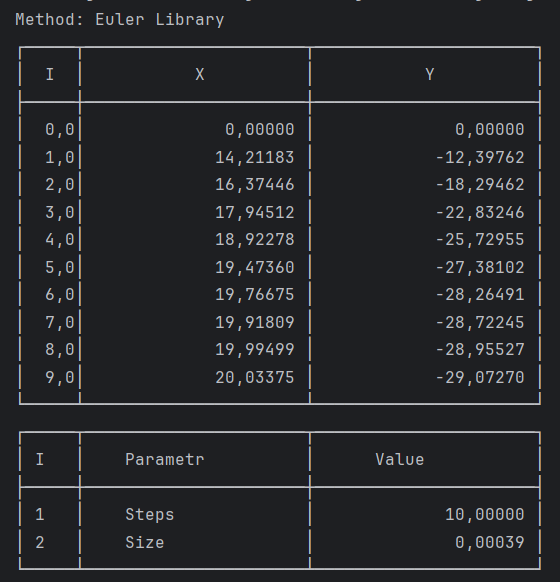


Рисунок Виконання Бібліотечного модулю методу Ейлера

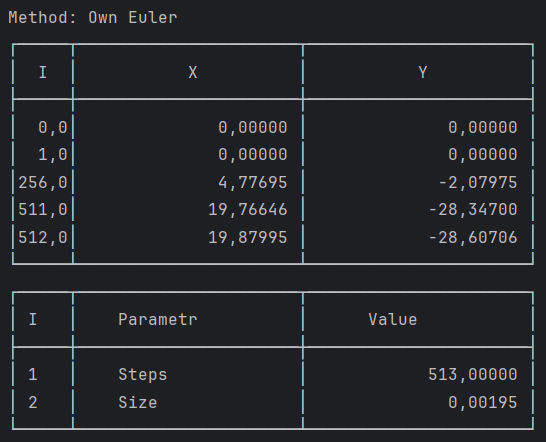


Рисунок Результат виконання власного методу Ейлера

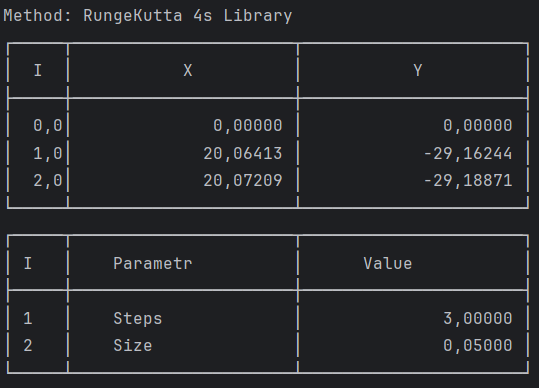


Рисунок Результат Виконання методу Рунге-Кутта 4го Порядку

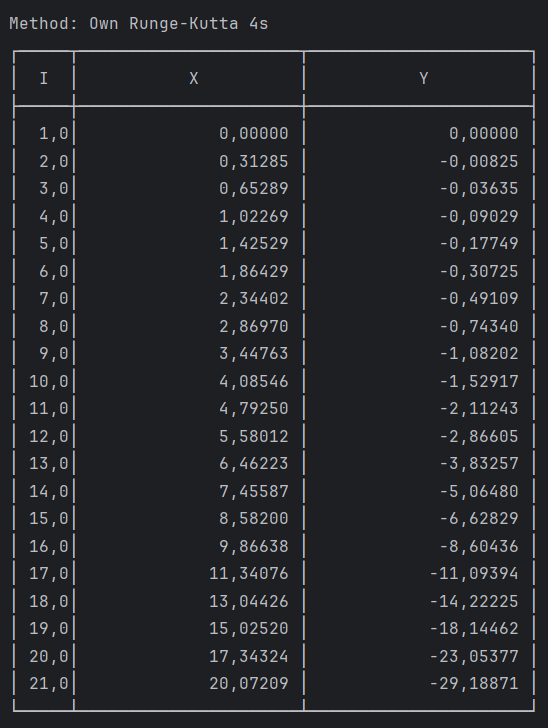


Рисунок Результат ВИконання Власного методу Ранге-Кутта 4го Порядку

**Лістинг Коду**

import org.apache.commons.math3.ode.FirstOrderDifferentialEquations;  
import org.apache.commons.math3.ode.FirstOrderIntegrator;  
import org.apache.commons.math3.ode.nonstiff.EulerIntegrator;  
import org.apache.commons.math3.ode.nonstiff.ClassicalRungeKuttaIntegrator;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
 public static void main(String[] args) {  
 double x0 = 0, y0 = 0, tStart = 0, tEnd = 1, stepStart = 0.1;  
 double precision = 0.1;  
 var equations = new RealisedFirstOrderDifferentialEquations();  
 stepStart = *solveEuler*(equations,x0,y0,tStart,tEnd,stepStart, precision);  
 x0 = 0; y0 = 0; tStart = 0;  
 *Euler*(x0,y0,tStart,tEnd,stepStart);  
 x0 = 0; y0 = 0; tStart = 0; stepStart = 0.1;  
 stepStart = *solveRK4*(equations,x0,y0,tStart,tEnd,stepStart, precision);  
 x0 = 0; y0 = 0; tStart = 0;  
 *RK4*(x0,y0,tStart,tEnd,stepStart);  
  
 }  
  
 private static double FX(double x0, double y0, double t)  
 {  
 return 4\*t+x0-y0+6;  
 }  
 private static double FY(double x0, double y0, double t1)  
 {  
 return -x0+4\*y0;  
 }  
  
 private static double solveEuler(FirstOrderDifferentialEquations equations, double x0, double y0, double tStart, double tEND, double stepStart, double precision){  
 double[]eulerResult = {x0,y0}, oldEulerRes;  
 List<double[]> res = new ArrayList<>();  
 int itter = 0;  
 res.add(new double[]{itter,eulerResult[0],eulerResult[1]});  
 do {  
 oldEulerRes = eulerResult;  
 eulerResult = *solveEquation*(new EulerIntegrator(stepStart), equations,x0, y0,tStart,tEND);  
 stepStart /= 2;  
 itter++;  
 res.add(new double[]{itter,eulerResult[0],eulerResult[1]});  
 }while (*calculateRootMSquereError*(eulerResult,oldEulerRes) > precision);  
 *print*(res,stepStart\*2,"Euler Library");  
 return stepStart;  
 }  
 private static void Euler(double x0, double y0, double tStart, double tEnd, double stepStart) {  
 stepStart\*=10;  
 double[] eulerRes = {x0, y0};  
 int INTER = 0;  
 List<double[]> result = new ArrayList<>();  
 result.add(new double[]{INTER, eulerRes[0], eulerRes[1]});  
  
 for (; tStart < tEnd; tStart += stepStart) {  
 double dxdt = *FX*(x0, y0, tStart);  
 double dydt = *FY*(x0, y0, tStart);  
  
 x0 += stepStart \* dxdt;  
 y0 += stepStart \* dydt;  
  
 eulerRes = new double[]{x0, y0};  
 INTER++;  
  
 result.add(new double[]{INTER, eulerRes[0], eulerRes[1]});  
 }  
  
 *print*(result, stepStart, "Own Euler");  
 }  
  
  
  
 private static void RK4(double x0, double y0, double tStart, double tEND, double stepStart){  
 List<double[]> res = new ArrayList<>();  
 res.add(new double[]{1, x0, y0});  
  
 for (double t = tStart; t < tEND; t += stepStart) {  
 double k1x = stepStart \* *FX*(x0, y0, t);  
 double k1y = stepStart \* *FY*(x0, y0, t);  
  
 double k2x = stepStart \* *FX*(x0 + k1x / 2, y0 + k1y / 2, t + stepStart / 2);  
 double k2y = stepStart \* *FY*(x0 + k1x / 2, y0 + k1y / 2, t + stepStart / 2);  
  
 double k3x = stepStart \* *FX*(x0 + k2x / 2, y0 + k2y / 2, t + stepStart / 2);  
 double k3y = stepStart \* *FY*(x0 + k2x / 2, y0 + k2y / 2, t + stepStart / 2);  
  
 double k4x = stepStart \* *FX*(x0 + k3x, y0 + k3y, t + stepStart);  
 double k4y = stepStart \* *FY*(x0 + k3x, y0 + k3y, t + stepStart);  
  
 x0 += (k1x + 2 \* k2x + 2 \* k3x + k4x) / 6;  
 y0 += (k1y + 2 \* k2y + 2 \* k3y + k4y) / 6;  
  
 res.add(new double[]{res.size() + 1, x0, y0});  
 }  
  
 *print*(res, stepStart, "Own Runge-Kutta 4s");;  
 }  
 private static double solveRK4(FirstOrderDifferentialEquations equations, double x0, double y0, double tStart, double tEND, double stepStart, double precision){  
 double[] rungeKuttaRes = {x0,y0}, oldRungeKuttaRes;  
 List<double[]> res = new ArrayList<>();  
 int INTER = 0;  
 res.add(new double[]{INTER,rungeKuttaRes[0],rungeKuttaRes[1]});  
  
 do {  
 oldRungeKuttaRes = rungeKuttaRes;  
 rungeKuttaRes = *solveEquation*(new ClassicalRungeKuttaIntegrator(stepStart), equations,x0, y0,tStart,tEND);  
 stepStart/=2;  
 INTER++;  
 res.add(new double[]{INTER,rungeKuttaRes[0],rungeKuttaRes[1]});  
 }while (*calculateRootMSquereError*(rungeKuttaRes, oldRungeKuttaRes) > 0.1);  
 *print*(res,stepStart\*2,"RungeKutta 4s Library");  
 return stepStart\*2;  
 }  
  
  
 private static double calculateRootMSquereError(double[] current, double[] precision){  
 double sumSquaredDifferences = 0.0;  
 for (int i = 0; i < current.length; i++) {  
 double difference = current[i] - precision[i];  
 sumSquaredDifferences += difference \* difference;  
 }  
 double meanSquaredDifference = sumSquaredDifferences / current.length;  
 return Math.*sqrt*(meanSquaredDifference);  
 }  
  
 private static double[] solveEquation(  
 FirstOrderIntegrator integrator,  
 FirstOrderDifferentialEquations equations,  
 double initialX,  
 double initialY,  
 double initialTime,  
 double endTime  
 ) {  
 double[] initialValues = new double[]{initialX, initialY};  
 integrator.integrate(equations, initialTime, initialValues, endTime, initialValues);  
 return initialValues;  
 }  
  
  
 private static void print(List<double[]> list, double step, String method) {  
 System.*out*.println("Method: "+method);  
 System.*out*.println("┌─────┬──────────────────────┬──────────────────────┐");  
 System.*out*.println("│ I │ X │ Y │");  
 System.*out*.println("├─────┼──────────────────────┼──────────────────────┤");  
  
 if (list.size() > 50) {  
 System.*out*.printf("│%5.1f│ %20.5f │ %20.5f │%n", list.get(0)[0], list.get(0)[1], list.get(0)[1]);  
 System.*out*.printf("│%5.1f│ %20.5f │ %20.5f │%n", list.get(1)[0], list.get(1)[2], list.get(1)[2]);  
 System.*out*.printf("│%5.1f│ %20.5f │ %20.5f │%n", list.get(list.size()/2)[0], list.get(list.size()/2)[1], list.get(list.size()/2)[2]);  
 System.*out*.printf("│%5.1f│ %20.5f │ %20.5f │%n", list.get(list.size()-2)[0], list.get(list.size()-2)[1], list.get(list.size()-2)[2]);  
 System.*out*.printf("│%5.1f│ %20.5f │ %20.5f │%n", list.get(list.size()-1)[0], list.get(list.size()-1)[1], list.get(list.size()-1)[2]);  
 } else {  
 for (double[] data : list) {  
 System.*out*.printf("│%5.1f│ %20.5f │ %20.5f │%n", data[0], data[1], data[2]);  
 }  
 }  
 System.*out*.println("└─────┴──────────────────────┴──────────────────────┘");  
 System.*out*.println("┌─────┬──────────────────────┬──────────────────────┐");  
 System.*out*.println("│ I │ Parametr │ Value │");  
 System.*out*.println("├─────┼──────────────────────┼──────────────────────┤");  
 System.*out*.printf("│ 1 │ Steps │ %20.5f │%n", (double)list.size());  
 System.*out*.printf("│ 2 │ Size │ %20.5f │%n", step);  
 System.*out*.println("└─────┴──────────────────────┴──────────────────────┘");  
 }  
}