ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Факультет кибербезопасности и управления

Кафедра ??

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №** **4**

по дисциплине Числовые методы

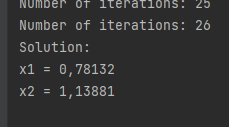
|  |  |  |
| --- | --- | --- |
| **ВЫПОЛНИЛ** | | |
| студент | РПИС-11  (группа) | Ларкин М.Л.  (ФИО) |
|  | **ПРОВЕРИЛ** |  |
| Преподаватель  (должность) | | Осанов В А.  (ФИО) |

Самара 2023

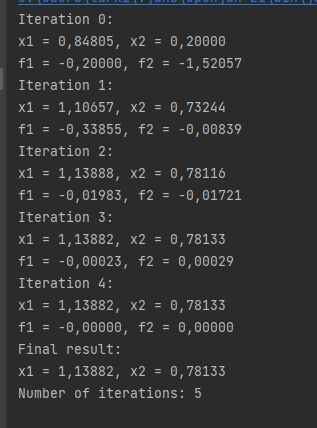
Данные для 13 варианта:

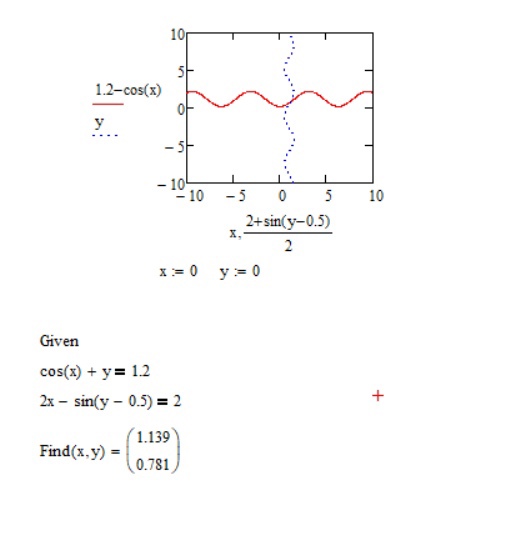


Пример решения через метод простых итераций:



Пример для метода Ньютона:





Код для программы по простым итерациям:

import kotlin.math.cos  
import kotlin.math.sin  
import kotlin.math.abs  
  
const val *eps* = 1e-5  
const val *maxIterations* = 1000  
  
fun phi1(x: Double): Double {  
 return 1.2 - *cos*(x)  
}  
  
fun phi2(y: Double): Double {  
 return (2 + *sin*(y - 0.5)) / 2  
}  
  
fun solveByFixedPointIteration(): Pair<Double, Double>? {  
 var x = 0.0  
 var y = 0.0  
 var newX = 0.0  
 var newY = 0.0  
 var iterations = 0  
  
 while (iterations < *maxIterations*) {  
 newX = *phi1*(y)  
 newY = *phi2*(x)  
  
 if (*abs*(newX - x) < *eps* && *abs*(newY - y) < *eps*) {  
 return Pair(newX, newY)  
 }  
  
 x = newX  
 y = newY  
 iterations++  
 }  
  
 return null  
}  
  
fun main() {  
 val solution = *solveByFixedPointIteration*()  
  
 if (solution != null) {  
 *println*("Solution:")  
 *println*("x1 = %.5f".*format*(solution.first))  
 *println*("x2 = %.5f".*format*(solution.second))  
 } else {  
 *println*("No solution found within the maximum number of iterations.")  
 }  
}

Код для программы по методу Ньютона:

import kotlin.math.cos  
import kotlin.math.sin  
import kotlin.math.abs  
  
const val *EPSILON* = 1e-5  
const val *MAX\_ITERATIONS* = 100  
  
fun main() {  
 var x1 = 0.0  
 var x2 = 0.0  
  
 var iter = 0  
 var dx1: Double  
 var dx2: Double  
  
 do {  
 val f1 = *cos*(x1) + x2 - 1.2  
 val f2 = 2 \* x1 - *sin*(x2 - 0.5) - 2  
  
 val jacobian11 = -*sin*(x1)  
 val jacobian12 = 1.0  
 val jacobian21 = 2.0  
 val jacobian22 = -*cos*(x2 - 0.5)  
  
 val determinant = jacobian11 \* jacobian22 - jacobian12 \* jacobian21  
  
 dx1 = (jacobian22 \* f1 - jacobian12 \* f2) / determinant  
 dx2 = (-jacobian21 \* f1 + jacobian11 \* f2) / determinant  
  
 x1 -= dx1  
 x2 -= dx2  
  
 *println*("Iteration $iter:")  
 *println*("x1 = %.5f, x2 = %.5f".*format*(x1, x2))  
 *println*("f1 = %.5f, f2 = %.5f".*format*(f1, f2))  
  
 iter++  
 } while ((*abs*(dx1) > *EPSILON* || *abs*(dx2) > *EPSILON*) && iter < *MAX\_ITERATIONS*)  
  
 *println*("Final result:")  
 *println*("x1 = %.5f, x2 = %.5f".*format*(x1, x2))  
 *println*("Number of iterations: $iter")  
}