ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Факультет кибербезопасности и управления

Кафедра ??

**ОТЧЕТ**

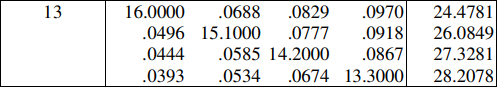
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №** **2**

по дисциплине Числовые методы

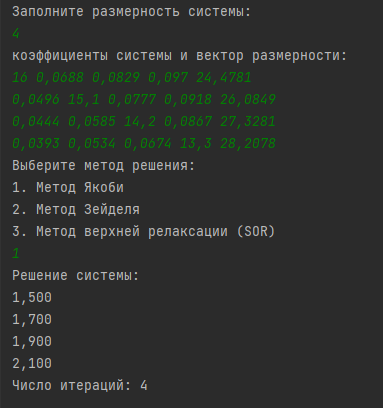
|  |  |  |
| --- | --- | --- |
| **ВЫПОЛНИЛ** | | |
| студент | РПИС-11  (группа) | Ларкин М.Л.  (ФИО) |
|  | **ПРОВЕРИЛ** |  |
| Преподаватель  (должность) | | Осанов В А.  (ФИО) |

Самара 2023

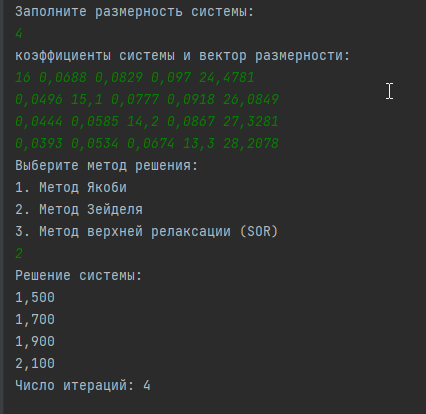
Данные для 13 варианта:



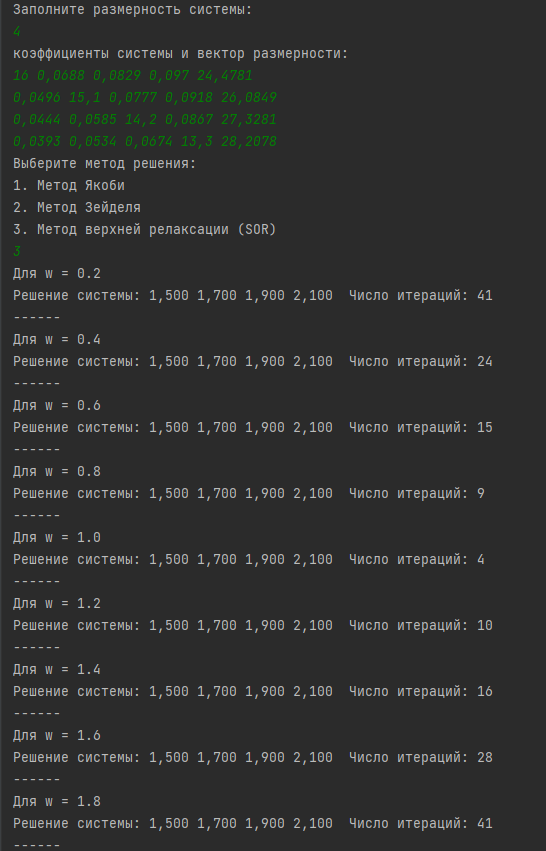
Решение для метода Якоби:



Решение для метода Зейделя:



Решение через метод верхней релаксации:



Код программы:

import java.util.\*  
  
fun main() {  
 val scanner = Scanner(System.*`in`*)  
  
 *println*("Заполните размерность системы:")  
 val Nn: Int = *readLine*()!!.*toInt*()  
  
 *println*("коэффициенты системы и вектор размерности:")  
 val Aa = Array(Nn) **{** DoubleArray(Nn) **}** val Bb = DoubleArray(Nn)  
  
 for (i in 0 *until* Nn) {  
 for (j in 0 *until* Nn) {  
 Aa[i][j] = scanner.nextDouble()  
 }  
 Bb[i] = scanner.nextDouble()  
 }  
  
 *println*("Выберите метод решения:")  
 *println*("1. Метод Якоби\n2. Метод Зейделя\n3. Метод верхней релаксации (SOR)")  
 val choice = *readLine*()?.*toInt*() ?: 1  
  
 when (choice) {  
 2 -> {  
 val (solution, iterations) = *seidel*(Aa, Bb)  
 *println*("Решение системы:")  
 for (value in solution) {  
 *println*("%.3f".*format*(value))  
 }  
 *println*("Число итераций: $iterations")  
 }  
 3 -> {  
 for (w in 2..18 *step* 2) {  
 val relaxFactor = w / 10.0  
 val (solution, iterations) = *sor*(Aa, Bb, relaxFactor)  
 *println*("Для w = $relaxFactor")  
 *print*("Решение системы: ")  
 for (value in solution) {  
 *print*("%.3f ".*format*(value))  
 }  
 *println*(" Число итераций: $iterations")  
 *println*("------")  
 }  
 }  
 else -> {  
 val (solution, iterations) = *jacobi*(Aa, Bb)  
 *println*("Решение системы:")  
 for (value in solution) {  
 *println*("%.3f".*format*(value))  
 }  
 *println*("Число итераций: $iterations")  
 }  
 }  
}  
fun jacobi(A: Array<DoubleArray>, b: DoubleArray, epsilon: Double = 1e-5): Pair<DoubleArray, Int> {  
 val n = A.size  
 var x = DoubleArray(n) **{** 1.0 **}** var xNew = DoubleArray(n)  
 var iterations = 0  
 var difference: Double  
  
 do {  
 iterations++  
 for (i in 0 *until* n) {  
 var sum = 0.0  
 for (j in 0 *until* n) {  
 if (i != j) sum += A[i][j] \* x[j]  
 }  
 xNew[i] = (b[i] - sum) / A[i][i]  
 }  
  
 difference = 0.0  
 for (i in 0 *until* n) {  
 difference += Math.abs(xNew[i] - x[i])  
 x[i] = xNew[i]  
 }  
  
 } while (difference > epsilon && iterations <= n \* 10)  
  
 return Pair(x, iterations)  
}  
  
fun seidel(A: Array<DoubleArray>, b: DoubleArray, epsilon: Double = 1e-5): Pair<DoubleArray, Int> {  
 val n = A.size  
 var x = DoubleArray(n) **{** 1.0 **}** var xPrev = DoubleArray(n)  
 var iterations = 0  
 var difference: Double  
  
 do {  
 xPrev = x.*copyOf*()  
 for (i in 0 *until* n) {  
 var temp = b[i]  
 for (j in 0 *until* n) {  
 if (i != j) {  
 temp -= A[i][j] \* x[j]  
 }  
 }  
 x[i] = temp / A[i][i]  
 }  
  
 difference = 0.0  
 for (i in 0 *until* n) {  
 difference += Math.abs(x[i] - xPrev[i])  
 }  
 iterations++  
  
 } while (difference > epsilon && iterations <= n \* 10)  
  
 return Pair(x, iterations)  
}  
fun sor(A: Array<DoubleArray>, b: DoubleArray, w: Double, epsilon: Double = 1e-5): Pair<DoubleArray, Int> {  
 val n = A.size  
 var x = DoubleArray(n) **{** 1.0 **}** var xPrev = DoubleArray(n)  
 var iterations = 0  
 var difference: Double  
  
 do {  
 xPrev = x.*copyOf*()  
 for (i in 0 *until* n) {  
 var temp = b[i]  
 for (j in 0 *until* n) {  
 if (i != j) {  
 temp -= A[i][j] \* x[j]  
 }  
 }  
 x[i] = (1 - w) \* x[i] + (w / A[i][i]) \* temp  
 }  
  
 difference = 0.0  
 for (i in 0 *until* n) {  
 difference += Math.abs(x[i] - xPrev[i])  
 }  
 iterations++  
  
 } while (difference > epsilon && iterations <= n \* 10)  
  
 return Pair(x, iterations)  
}