**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Факультет прикладной математики и информатики

Лабораторная работа №1 по курсу “ВМА”

“Метод Гаусса с выбором главного элемента по столбцам”

Вариант №3

Выполнил: Ёда Никита

3 курс, 6(а) группа

Преподаватель: Будник А.М.

2023

Содержание:

Постановка задачи...................................................................................................3

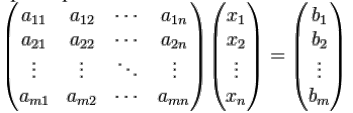
Алгоритм решения..................................................................................................4

Листинг программы.................................................................................................7

Результат и его анализ...........................................................................................13

**Постановка задачи**

В данной лабораторной работе требуется решить систему линейных алгебраических уравнений Ах=b, используя метод Гаусса с выбором главного элемента по столбцу.



План решения:

- метод Гаусса (с выбором гл. Эл. По столбцам) (1)

- решение системы (2)

- ­определитель матрицы A (3)

- обратную матрицу A^(-1) (4)

- число обусловленности v(A) (5)

- вектор невязки (6)

- матрицу невязки (7)

**Алгоритм решения**

*1. Метод Гаусса с выбором главного элемента по столбцу:*

1.1. Нахожу максимальный элемент по модулю в первом столбце. Для этого я использовал цикл от 0 до n-1, n – размер матрицы. Максимальный элемент нахожу в соответствии с номером итерации (итерация от 0 до n-1). Если 0, то первый столбец и т.д.

1.2. Переставляю строку с максимальным элементом с первой строкой (перестоновка также зависит от номера итерации, если 0, то первая строка и т.д.)

1.3. Нахожу масштабирующий множитель по формуле:

ui = , где k - шаг, i - строка, aij - матрица А

1.4. Преобразую матрицу по формуле:

ij = ij - kj \* , где k - шаг, i - строка, j - столбец, aij - матрица А

*2. Решение системы уравнений:*

Нахожу вектор x (x1, x2, x3, x4). Т.к. матрица приведена к треугольному виду, то это сделать довольно таки легко, используем mtrA и vecB.

s = b[i]

s = s - aij \* x[j]

aij = s / aij

где s - сумма, aij - матрица А, x - массив иксов

*3. Вычисление определителя матрицы A:*

Определитель mtrA (преобразованной матрицы после метода Гаусса) будет равен произведению всех элементов на диагонали. Нужно учитывать количество итераций. В моём случае, если count % 2 == 1, то умножаем на -1.

1. *Вычисление обратной матрицы A^-1:*

Для того, чтобы вычислить обратную матрицу я добавляю к исходной матрице - единичную матрицу. Таким образом:

0,3857 -0,0508 0,0102 0,0203 0,0711 | 1,0000 0,0000 0,0000 0,0000 0,0000

0,0528 0,6039 0,0000 -0,0406 0,0406 | 0,0000 1,0000 0,0000 0,0000 0,0000

0,0305 0,0000 0,4852 -0,1421 0,0812 | 0,0000 0,0000 1,0000 0,0000 0,0000

-0,0609 0,1279 0,0000 0,4711 -0,0203 | 0,0000 0,0000 0,0000 1,0000 0,0000

0,2538 0,0000 0,0914 0,0102 0,5684 | 0,0000 0,0000 0,0000 0,0000 1,0000

Теперь, необходимо привести левую матрицу, чтобы из неё получилась единичная матрица, а из единичной матрицы (та, которая справа), получится обратная матрица.

*5. Вычисление числа обусловленности v(A):*

Для того, чтобы найти число обусловленности надо умножить норму 1, изначальной матрицы, на норму 1, обратной матрицы.

Норма 1 определяется как сумма абсолютных значений элементов каждого столбца матрицы. Для матрицы размера m x n норма 1 равна максимальной сумме абсолютных значений столбцов.

*6. Нахождение вектора невязки r:*

Просто вычисляю r=Ax-b, надо учитывать, что невязка будет маленькой, т.е. \*10^-16

1. *Нахождение матрицы невязки:*

Для нахождения матрицы невязки, нужно воспользоваться формулой:

R=A^-1\*A-E, где A-исходная матрица, A^-1 - обратная матрица, E - единичная матрица

**Листинг программы**

import java.text.DecimalFormat;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 double[][] mtrA = {{0.3857, -0.0508, 0.0102, 0.0203, 0.0711},  
 {0.0528, 0.6039, 0.0000, -0.0406, 0.0406},  
 {0.0305, 0.0000, 0.4852, -0.1421, 0.0812},  
 {-0.0609, 0.1279, 0.0000, 0.4711, -0.0203},  
 {0.2538, 0.0000, 0.0914, 0.0102, 0.5684}};  
 int mtrALength = mtrA.length;  
  
 double[] vecB = {0.7613, -0.8709, 3.2074, -1.8290, 2.9537};  
 int vecBLength = vecB.length;;  
  
  
 // Метод Гаусса  
 System.out.println("|Метод Гаусса|");  
 double[][] copyMtrA1 = copyMtr(mtrA, mtrALength);  
 double[] copyVecB1 = copyVec(vecB, vecBLength);  
 int numOfStepsInGaussMet = 0; // Кол-во итераций  
 gaussMet(copyMtrA1, copyVecB1, numOfStepsInGaussMet, mtrALength, vecBLength);  
  
  
 // Нахождение Х-ов  
 double[] massX = findX(copyMtrA1, copyVecB1, mtrALength);  
 int massXLength = massX.length;  
 System.out.println("\nВектор решений X:");  
 printVecWithoutRound(massX, massXLength);  
  
  
 // Нахождение det(mtrA)  
 System.out.println("\n\n|Нахождение det(mtrA)|");  
 double detMtrA = detMtr(copyMtrA1, numOfStepsInGaussMet, mtrALength);  
 System.out.println("det(mtrA) = " + detMtrA);  
  
  
 // Нахождение обратной матрицы  
 System.out.print("\n|Нахождение обратной матрицы|");  
 double[][] singleMtr = new double[mtrALength][mtrALength]; // Создание единичной м-цы  
 for (int i = 0; i < mtrALength; i++) {  
 singleMtr[i][i] = 1;  
 }  
 double[][] copyMtrA2 = copyMtr(mtrA, mtrALength);  
 double[][] reverseMtr = reverseMtr(copyMtrA2, singleMtr, mtrALength);  
  
  
 // Нахождение числа обусловленности  
 System.out.print("\n|Нахождения числа обусловленности|");  
 double[][] copyMtrA3 = copyMtr(mtrA, mtrALength);  
 System.out.println("\nЧисло обусловленности: " + (mtrNorm(copyMtrA3, mtrALength, "mtrA") \* mtrNorm(reverseMtr, mtrALength, "reverseMtr")));  
  
  
 // Нахождение вектора невязки  
 System.out.print("\n|Нахождение невязки|");  
 double[] copyVecB2 = copyVec(vecB, vecBLength);  
 double[] massResidua = findResidua(copyMtrA3, copyVecB2, massX);  
 System.out.print("\nВектор невязки:\n");  
 printVecWithoutRound(massResidua, massXLength);  
  
  
 // Матрица матрицы невязки  
 System.out.print("\n\n|Матрица невязки|");  
 double[][] copyMtrA4 = copyMtr(mtrA, mtrALength);  
 printMtrWithoutRounding(calcResMtr(copyMtrA4, reverseMtr), mtrALength);  
 }  
 catch (ArithmeticException e) {  
 System.out.println("Ошибка: " + e.getMessage());  
 }  
 }  
  
  
 // Глубокое копирование м-цы  
 public static double[][] copyMtr(double[][] mtr, int mtrLength) {  
 double[][] copyMtrA = new double[mtrLength][mtrLength];  
  
 for (int i = 0; i < mtrLength; i++) {  
 for (int j = 0; j < mtrLength; j++) {  
 copyMtrA[i][j] = mtr[i][j];  
 }  
 }  
  
 return copyMtrA;  
 }  
  
  
 // Глубокое копирование в-ра  
 public static double[] copyVec(double[] vec, int vecLength) {  
 double[] copyVecB = new double[vecLength];  
  
 for (int i = 0; i < vecLength; i++) {  
 copyVecB[i] = vec[i];  
 }  
  
 return copyVecB;  
 }  
  
  
 // Метод Гаусса  
 public static void gaussMet (double[][] mtrA, double[] vecB, int numOfStepsInGaussMet, int mtrALength, int vecBLength) {  
 System.out.print("До:");  
 printMtrWithVec(mtrA, vecB, mtrALength, vecBLength);  
  
 while (numOfStepsInGaussMet != mtrALength - 1) {  
 // Поиск максимального элемента  
 double maxEl = Math.abs(mtrA[numOfStepsInGaussMet][numOfStepsInGaussMet]);  
 for (int i = numOfStepsInGaussMet; i < mtrALength; i++) {  
 if (Math.abs(mtrA[i][numOfStepsInGaussMet]) > maxEl) {  
 maxEl = Math.abs(mtrA[i][numOfStepsInGaussMet]);  
 }  
 }  
  
  
 // Перестановка строк  
 for (int i = numOfStepsInGaussMet; i < mtrALength; i++) {  
 if (Math.abs(mtrA[i][numOfStepsInGaussMet]) == maxEl) {  
 for (int j = 0; j < mtrALength; j++) {  
 // Для mtrA  
 double temp = mtrA[numOfStepsInGaussMet][j];  
 mtrA[numOfStepsInGaussMet][j] = mtrA[i][j];  
 mtrA[i][j] = temp;  
 }  
  
 // Для vecA  
 double temp = vecB[numOfStepsInGaussMet];  
 vecB[numOfStepsInGaussMet] = vecB[i];  
 vecB[i] = temp;  
 }  
 }  
  
  
 // Высчитываем масштабирующие множители  
 double[] massU = new double[mtrALength - 1];  
 for (int i = numOfStepsInGaussMet + 1; i < mtrALength; i++) {  
 massU[i - 1] = mtrA[i][numOfStepsInGaussMet] / mtrA[numOfStepsInGaussMet][numOfStepsInGaussMet];  
 }  
  
  
 // Преобразование матрицы  
 for (int i = numOfStepsInGaussMet + 1; i < mtrALength; i++) {  
 for (int j = numOfStepsInGaussMet; j < mtrALength; j++) {  
 mtrA[i][j] = mtrA[i][j] - (mtrA[numOfStepsInGaussMet][j] \* massU[i - 1]); // Для mtrA  
 }  
 vecB[i] = vecB[i] - (vecB[numOfStepsInGaussMet] \* massU[i - 1]); // Для vecA  
 }  
  
 numOfStepsInGaussMet++;  
 }  
  
 System.out.print("\nПосле:");  
 printMtrWithVec(mtrA, vecB, mtrALength, vecBLength);  
  
 System.out.println("\nКол-во шагов:\nk = " + (numOfStepsInGaussMet + 1));  
  
 System.out.println("Матрица приведена к треугольному виду!");  
 }  
  
  
 // Нахождение X-ов  
 public static double[] findX(double[][] mtrA, double[] vecB, int mtrALength) {  
 double[] massX = new double[mtrALength];  
  
 for (int i = mtrALength - 1; i >= 0; i--) {  
 double sum = vecB[i];  
 for (int j = mtrALength - 1; j > i; j--) {  
 sum -= mtrA[i][j] \* massX[j];  
 }  
 massX[i] = sum / mtrA[i][i];  
 }  
  
 return massX;  
 }  
  
  
 // Нахождение определителя м-цы (используя метод Гаусса)  
 public static double detMtr(double[][] mtrA, int numOfStepsInGaussMet, int mtrALength) {  
 double determinant = 1.0;  
 for (int i = 0; i < mtrALength; i++) {  
 determinant \*= mtrA[i][i];  
 }  
  
 if (numOfStepsInGaussMet % 2 == 1) {  
 return determinant \* -1;  
 }  
 else {  
 return determinant;  
 }  
 }  
  
  
 // Нахождение обратной м-цы  
 public static double[][] reverseMtr(double[][] copyMtr, double[][] singleMtr, int mtrALength) {  
 System.out.print("\nДо:");  
 printMtrWithMtr(copyMtr, singleMtr);  
  
 // Преобразование в единичную матрицу  
 for (int i = 0; i < mtrALength; i++) {  
 double pivot = copyMtr[i][i];  
  
 // Деление строки на pivot  
 for (int j = 0; j < mtrALength; j++) {  
 copyMtr[i][j] /= pivot; // Для cMtr  
 singleMtr[i][j] /= pivot; // Для singleMtr  
 }  
  
 // Вычитание других строк для получения нулей под главной диагональю  
 for (int k = 0; k < mtrALength; k++) {  
 if (k != i) {  
 double factor = copyMtr[k][i];  
 for (int j = 0; j < mtrALength; j++) {  
 copyMtr[k][j] -= factor \* copyMtr[i][j]; // Для cMtr  
 singleMtr[k][j] -= factor \* singleMtr[i][j]; // Для singleMtr  
 }  
 }  
 }  
 }  
  
 System.out.print("\nПосле:");  
 printMtrWithMtr(copyMtr, singleMtr);  
  
 System.out.print("\nОбратная м-ца:");  
 printMtr(singleMtr, singleMtr.length);  
  
 return singleMtr;  
 }  
  
  
 // Норма м-цы  
 public static double mtrNorm(double[][] mtrA, int mtrALength, String name) {  
 double maxSum = 0.0;  
  
 for (int j = 0; j < mtrALength; j++) {  
 double columnSum = 0.0;  
 for (int i = 0; i < mtrALength; i++) {  
 columnSum += Math.abs(mtrA[i][j]);  
 }  
 maxSum = Math.max(maxSum, columnSum);  
 }  
 System.out.print("\n||" + name + "|| = " + maxSum);  
  
 return maxSum;  
 }  
  
  
 // Вывод м-цы  
 public static void printMtr(double[][] mtrA, int mtrALength) {  
 System.out.println("\nВывод м-цы:");  
  
 DecimalFormat decimalFormat = new DecimalFormat("#.####");  
  
 for (int i = 0; i < mtrALength; i++) {  
 for (int j = 0; j < mtrALength; j++) {  
 System.out.print(decimalFormat.format(mtrA[i][j]));  
  
 for (int k = 0; k < 15 - decimalFormat.format(mtrA[i][j]).length(); k++) {  
 System.out.print(" ");  
 }  
 }  
 System.out.println();  
 }  
 }  
  
  
 // Вывод м-цы и в-ра  
 public static void printMtrWithVec(double[][] mtrA, double[] vecB, int mtrALength, int vecBLength) {  
 System.out.println("\nВывод м-цы и в-ра:");  
  
 DecimalFormat decimalFormat = new DecimalFormat("#.####");  
  
 for (int i = 0; i < mtrALength; i++) {  
 for (int j = 0; j < mtrALength; j++) {  
 System.out.print(decimalFormat.format(mtrA[i][j]));  
  
 for (int k = 0; k < 15 - decimalFormat.format(mtrA[i][j]).length(); k++) {  
 System.out.print(" ");  
 }  
 }  
 System.out.println("|\t" + decimalFormat.format(vecB[i]));  
 }  
 }  
  
  
 // Вывод м-цы и м-цы  
 public static void printMtrWithMtr(double[][] mtr1, double[][] mtr2) {  
 System.out.println("\nВывод м-цы и м-цы:");  
  
 DecimalFormat decimalFormat = new DecimalFormat("#.####");  
  
 for (int i = 0; i < mtr1.length; i++) {  
 for (int j = 0; j < mtr1.length; j++) {  
 System.out.print(decimalFormat.format(mtr1[i][j]));  
 for (int k = 0; k < 15 - decimalFormat.format(mtr1[i][j]).length(); k++) {  
 System.out.print(" ");  
 }  
 }  
  
 System.out.print("|\t");  
  
 for (int j = 0; j < mtr2.length; j++) {  
 System.out.print(decimalFormat.format(mtr2[i][j]));  
 for (int k = 0; k < 15 - decimalFormat.format(mtr2[i][j]).length(); k++) {  
 System.out.print(" ");  
 }  
 }  
  
 System.out.println();  
 }  
 }  
  
  
 // Вывод вектора без округления  
 public static void printVecWithoutRound(double[] vec, int vecLength) {  
 System.out.println("Вывод в-ра:");  
  
 for (int i = 0; i < vecLength; i++) {  
 System.out.print(vec[i]);  
 for (int j = 0; j < 30 - Double.toString(vec[i]).length(); j++) {  
 System.out.print(" ");  
 }  
 }  
 }  
  
  
 // Вывод м-цы без округления  
 public static void printMtrWithoutRounding(double[][] mtr, int mtrLength) {  
 System.out.println("\nВывод м-цы");  
  
 for (int i = 0; i < mtrLength; i++) {  
 for (int j = 0; j < mtrLength; j++) {  
 System.out.print(mtr[i][j]);  
 for (int k = 0; k < 30 - Double.toString(mtr[i][j]).length(); k++) {  
 System.out.print(" ");  
 }  
 }  
 System.out.println();  
 }  
 }  
  
  
 // Нахождение в-ра невязки (r = A\*x - b)  
 public static double[] findResidua(double[][] mtrA, double[] vecB, double[] massX) {  
 double[] newVec = new double[vecB.length];  
 double sum = 0;  
  
 for (int i = 0; i < mtrA.length; i++) {  
 for (int j = 0; j < mtrA.length; j++) {  
 sum += mtrA[i][j] \* massX[j];  
 }  
 newVec[i] = sum;  
 sum = 0.0;  
 }  
  
 for (int i = 0; i < vecB.length; i++) {  
 newVec[i] = newVec[i] - vecB[i];  
 }  
  
 return newVec;  
 }  
  
  
 // Нахождение м-цы невязки (r = A\*x - b)  
 public static double[][] calcResMtr(double[][] mtrA, double[][] reverseMtr) {  
 double[][] multMtr = new double[mtrA.length][reverseMtr[0].length];  
  
 for (int i = 0; i < mtrA.length; i++) {  
 for (int j = 0; j < reverseMtr[0].length; j++) {  
 for (int k = 0; k < mtrA[0].length; k++) {  
 multMtr[i][j] += mtrA[i][k] \* reverseMtr[k][j];  
 }  
 }  
 }  
  
 double[][] singleMtr = new double[mtrA.length][mtrA.length]; // Создание единичной м-цы  
 for (int i = 0; i < mtrA.length; i++) {  
 singleMtr[i][i] = 1;  
 }  
  
 double[][] subtraction = new double[multMtr.length][multMtr[0].length];  
 for (int i = 0; i < multMtr.length; i++) {  
 for (int j = 0; j < multMtr[0].length; j++) {  
 subtraction[i][j] = multMtr[i][j] - singleMtr[i][j];  
 }  
 }  
  
 return subtraction;  
 }  
}

**Результаты и его анализ**

*Результат:*

|Метод Гаусса|

До:

Вывод м-цы и в-ра:

0,3857 -0,0508 0,0102 0,0203 0,0711 | 0,7613

0,0528 0,6039 0 -0,0406 0,0406 | -0,8709

0,0305 0 0,4852 -0,1421 0,0812 | 3,2074

-0,0609 0,1279 0 0,4711 -0,0203 | -1,829

0,2538 0 0,0914 0,0102 0,5684 | 2,9537

После:

Вывод м-цы и в-ра:

0,3857 -0,0508 0,0102 0,0203 0,0711 | 0,7613

0 0,6109 -0,0014 -0,0434 0,0309 | -0,9751

0 0 0,4844 -0,1434 0,0754 | 3,1536

0 0 0 0,4834 -0,0154 | -1,5297

0 0 0 0 0,5075 | 2,0312

Кол-во шагов:

k = 5

Матрица приведена к треугольному виду!

Вектор решений X:

Вывод в-ра:

1.000161740101123 -2.0028139174367716 4.988388791426029 -3.036900748793956 4.002281144376663

|Нахождение det(mtrA)|

det(mtrA) = 0.027997864169352654

|Нахождение обратной матрицы|

До:

Вывод м-цы и м-цы:

0,3857 -0,0508 0,0102 0,0203 0,0711 | 1 0 0 0 0

0,0528 0,6039 0 -0,0406 0,0406 | 0 1 0 0 0

0,0305 0 0,4852 -0,1421 0,0812 | 0 0 1 0 0

-0,0609 0,1279 0 0,4711 -0,0203 | 0 0 0 1 0

0,2538 0 0,0914 0,0102 0,5684 | 0 0 0 0 1

После:

Вывод м-цы и м-цы:

1 0 0 0 0 | 2,7879 0,2529 0,0114 -0,0869 -0,3715

0 1 0 0 0 | -0,1351 1,6121 0,0209 0,1531 -0,0958

0 0 1 0 0 | 0,1381 -0,1212 2,1124 0,627 -0,288

0 0 0 1 0 | 0,3422 -0,4087 -0,019 2,0656 0,0629

0 0 0 0 1 | -1,2732 -0,0861 -0,3444 -0,0991 1,9704

Обратная м-ца:

Вывод м-цы:

2,7879 0,2529 0,0114 -0,0869 -0,3715

-0,1351 1,6121 0,0209 0,1531 -0,0958

0,1381 -0,1212 2,1124 0,627 -0,288

0,3422 -0,4087 -0,019 2,0656 0,0629

-1,2732 -0,0861 -0,3444 -0,0991 1,9704

|Нахождения числа обусловленности|

||mtrA|| = 0.7837

||reverseMtr|| = 4.6765518628477345

Число обусловленности: 3.6650136949137693

|Нахождение невязки|

Вектор невязки:

Вывод в-ра:

0.0 0.0 0.0 -4.440892098500626E-16 4.440892098500626E-16

|Матрица невязки|

Вывод м-цы

2.220446049250313E-16 8.673617379884035E-18 6.938893903907228E-18 8.673617379884035E-18 -5.551115123125783E-17

2.7755575615628914E-17 2.220446049250313E-16 0.0 5.204170427930421E-18 0.0

1.3877787807814457E-17 5.204170427930421E-18 0.0 -4.5102810375396984E-17 0.0

-1.734723475976807E-17 1.2576745200831851E-17 -4.336808689942018E-18 2.220446049250313E-16 6.938893903907228E-18

-2.220446049250313E-16 0.0 0.0 -6.938893903907228E-18 4.440892098500626E-16

-2.220446049250313E-16 0.0 0.0 -6.938893903907228E-18 4.440892098500626E-16

*Анализ:*

Число обусловленности небольшое (<10^2), из этого следует, что система хорошо обусловлена.

Векторы невязки очень близки к нулю (порядок 10^(-16)), это говорит о том, что ответ достаточно точен.

Общий результат состоит в том, что значения вектора невязки и матрицы невязки близки к нулю, что указывает на то, что решение или результат вычислений достаточно близки к точному значению. Однако, из-за погрешностей округления в вычислениях с плавающей точкой, некоторые значения могут не быть абсолютно точными нулями. Это нормальное явление при работе с плавающей точкой и требует учета при анализе результатов вычислений.