A picture containing logo

Description automatically generated

Лабораторная работа №1

по Вычислительной Математики

Вариант 13

Выполнил:

Пурэвсурэн Билгуун

Группа Р3213

Преподователь:

Малышева Татьяна Алексеевна

г. Санкт-Петербург 2022г

1. **Цель работы:**

Научиться искать решение СЛАУ при помощи численных методов, написать программу, которая будет совершать приближенные вычисления и находить решение, получая на вход матрицу из файла или консоли.

1. **Задание лабораторной работы:**
2. № варианта определяется как номер в списке группы согласно ИСУ. (13)
3. В программе численный метод должен быть реализован в виде отдельной подпрограммы или класса, в который входные/выходные данные передаются в качестве параметров.
4. Размерность матрицы n <=20 (задается из файла или с клавиатуры - по выбору конечного пользователя).
5. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

**Метод простых итераций, должно быть реализовано:**

* Точность задается с клавиатуры/файла
* Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
* Вывод вектора неизвестных:
* Вывод количества итераций, за которое было найдено решение.
* Вывод вектора погрешностей:

1. **Описание метода, расчетные формулы:**

Метод простых итераций - итерационный метод, находящий решение путем последовательного приближения значений переменных. За начальное (нулевое) приближение выбирают вектор свободных членов: 𝑥(0) = 𝐷 или нулевой вектор: 𝑥(0) = 0.

При очередной итерации вычисляются новые значение для всех переменных путем использования в уравнении перменных полученных на предыдущей итерации, то есть для , а для .   
Конечная формула метода простых итераций:

A picture containing diagram

Description automatically generated

1. **Реализация расчета**
2. public static void iteration(){  
    for (int i = 0; i < *SIZE*; i++){  
    *matrixX1*[i][0] = *matrixX2*[i][0];  
    }  
    for (int i = 0 ; i < *SIZE*; i++){  
    double sumOfOther = 0;  
    for (int j = 0; j < *SIZE*; j++){  
    if(j!=i){  
    sumOfOther += *matrixA*[i][j]\**matrixX1*[j][0];  
    }  
    }  
    sumOfOther += *matrixB*[i][0];  
    *matrixX2* [i][0] = sumOfOther;  
    }  
   }  
   public static boolean checkAllNewX(){  
    for(int i = 0; i < *SIZE*; i++){  
    if(Math.*abs*(*matrixX2*[i][0] - *matrixX1*[i][0]) > *epsilon*){  
    return false;  
    }  
    }  
    return true;  
   }  
     
   public static void start(){  
    int count = 0;  
     
    do {  
    *iteration*();  
    count++;  
    } while (!*checkAllNewX*() && count < *M*);  
     
    System.*out*.println("\nПосле работы программы");  
    for(int i = 0; i < *SIZE*; i++){  
    System.*out*.println("X" + (i+1) + " = " + *matrixX2*[i][0]);  
    }  
     
    if (count>=*M*){  
    System.*out*.println("\nИтерации не сходятся(на заданном максимальном их количестве)");  
    }  
    else{  
    System.*out*.println("\nОбщее количество итерации = " + count + "\n");  
    }  
     
    for( int i = 0; i < *SIZE*; i ++){  
    System.*out*.println("Вектор погрешности вектора X\_" + (i+1) + " = " + Math.*abs*(*matrixX2*[i][0] - *matrixX1*[i][0]));  
    }  
   }
3. **Реализация установки начальных значений**

public static void initMatrixX1andX2(){  
 *matrixX2* = new double[*SIZE*][1];  
 *matrixX1* = new double[*SIZE*][1];  
 for (int i = 0; i < *SIZE*; i++){  
 *matrixX2*[i][0] = *matrixB*[i][0];  
 }  
}

public static void setMatrixAandB(double[][] mainMatrix){  
 *matrixA* = new double[*SIZE*][*SIZE*];  
 *matrixB* = new double[*SIZE*][1];  
 *maxes* = new ArrayList<Double>(*SIZE*);  
  
 for(int i = 0; i < *SIZE*; i++){  
 for(int j = 0; j < *SIZE*; j++){  
 *matrixA*[i][j] = mainMatrix[i][j];  
 }  
 *matrixB*[i][0] = mainMatrix[i][*SIZE*];  
 }  
  
 System.*out*.println("До перестановки: ");  
 for(int i = 0; i < *SIZE*; i++){  
 for (int j = 0; j < *SIZE*; j ++){  
 System.*out*.print(*matrixA*[i][j] + " ");  
 }  
 System.*out*.print(*matrixB*[i][0]);  
 System.*out*.println();  
 }  
 *setDiagonalDominance*();  
 System.*out*.println("После перестановки: ");  
 for(int i = 0; i < *SIZE*; i++){  
 for (int j = 0; j < *SIZE*; j ++){  
 System.*out*.print(*matrixA*[i][j] + " ");  
 }  
 System.*out*.print(*matrixB*[i][0]);  
 System.*out*.println();  
 }  
  
 System.*out*.println("\nПосле извлечение матрицу C:");  
 for (int i = 0; i < *SIZE*; i++){  
 for (int j = 0; j < *SIZE*; j++){  
 if(*matrixA*[i][j] / *maxes*.get(i) != 1D){  
 *matrixA*[i][j] /= -*maxes*.get(i);  
 }  
 else {  
 *matrixA*[i][j] = 0;  
 }  
 }  
 *matrixB*[i][0] /= *maxes*.get(i);  
 }  
 for(int i = 0; i < *SIZE*; i++){  
 for (int j = 0; j < *SIZE*; j ++){  
 System.*out*.print(*matrixA*[i][j] + " ");  
 }  
 System.*out*.print(*matrixB*[i][0]);  
 System.*out*.println();  
 }  
  
}

1. **Примеры работы программы:**

**Calendar

Description automatically generated**

Text

Description automatically generated