Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по дисциплине «Вычислительная математика» №1**

Вариант: 4

Преподаватель:   
Рыбаков Степан Дмитриевич

Выполнил: Васильченко Роман

Группа: Р32081

Санкт-Петербург, 2023г

# Цель работы

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них средствами программирования.

# Описание метода

Итерационные методы дают возможность для системы (1) построить последовательность векторов 𝑥 (0) , 𝑥 (1) , … , 𝑥 (𝑘) , пределом которой должно быть точное решение 𝑥 (∗) : 𝑥 (∗) = lim 𝑘→∞ 𝑥 (𝑘) Построение последовательности заканчивается, как только достигается желаемая точность.

Приведем систему уравнений, выразив неизвестные 𝑥1, 𝑥2, … , 𝑥𝑛 соответственно из первого, второго и т.д. уравнений системы.

# Листинг программы

## SimpleIterationMethod.java

package lab1;  
  
import lab1.Utils.PrintUtil;  
import lab1.models.Matrix;  
import lab1.models.ResultSet;  
  
import java.util.ArrayList;  
  
import static java.lang.Math.*abs*;  
  
public class SimpleIterationMethod {  
 private static final PrintUtil *pr* = new PrintUtil();  
 private static double[][] *val*;  
 public static void execute(Matrix matrix, double eps){  
 *pr*.printMatrix(matrix);  
 if (*checkDiagonal*(matrix.getMatrix(),matrix.getSize())){  
 ResultSet rs = *method\_of\_simple\_iterations\_SLAU*(matrix, eps);  
 *pr*.*println*(rs.getTable());  
 *pr*.printVector("Решение системы: ", rs.getResult());  
 *pr*.printVector("Вектор невязки: ", rs.getResiduals());  
 return;  
 }  
 *permuteMatrixHelper*(matrix,0);  
 if (*val*!= null){  
 Matrix matrix1 = new Matrix(*val*);  
 *pr*.*println*("Матрица после перестановки строк");  
 *pr*.printMatrix(matrix1);  
 ResultSet rs = *method\_of\_simple\_iterations\_SLAU*(matrix1, eps);  
 *pr*.*println*(rs.getTable());  
 *pr*.printVector("Решение системы: ", rs.getResult());  
 *pr*.printVector("Вектор невязки: ", rs.getResiduals());  
 } else {  
 *pr*.notDiagonalAll();  
 }  
 }  
  
 public static boolean checkDiagonal(double[][] matrix, int size) {  
 int i, j, k = 1;  
 double sum;  
 for (i = 0; i < size;i++) {  
 sum = 0;  
 for (j = 0; j < size;j++) {  
 sum+= *abs*(matrix[i][j]);  
 }  
 sum -= *abs*(matrix[i][i]);  
 if (sum >= *abs*(matrix[i][i])) {  
 k = 0;  
 }  
 }  
 return (k == 1);  
 }  
  
 private static void permuteMatrixHelper(Matrix matrix, int index) {  
 if(index >= matrix.getMatrix().length - 1){  
 if (*checkDiagonal*(matrix.getMatrix(), matrix.getSize())){  
 *val* = new double[matrix.getSize()][matrix.getSize()+1];  
 for (int i = 0; i < matrix.getSize();i++){  
 for (int j = 0; j < matrix.getSize()+1;j++){  
 *val*[i][j] = matrix.getMatrix()[i][j];  
 }  
 }  
 }  
 } else {  
 for (int i = index; i < matrix.getMatrix().length; i++) {  
 double[] t = matrix.getMatrix()[index];  
 matrix.getMatrix()[index] = matrix.getMatrix()[i];  
 matrix.getMatrix()[i] = t;  
  
 *permuteMatrixHelper*(matrix, index + 1);  
  
 t = matrix.getMatrix()[index];  
 matrix.getMatrix()[index] = matrix.getMatrix()[i];  
 matrix.getMatrix()[i] = t;  
 }  
 *pr*.printMatrix(matrix);  
 }  
 }  
  
 private static ResultSet method\_of\_simple\_iterations\_SLAU(Matrix matrix, double eps) {  
 ResultSet rs = new ResultSet();  
 double[] x= new double[matrix.getSize()];  
 double norma, sum, t;  
 do  
 {  
 ArrayList<Double> esps = new ArrayList<>();  
 norma = 0;  
  
 for(int i = 0; i < matrix.getSize(); i++)  
 {  
 t = x[i];  
 sum = 0;  
  
 for(int j = 0; j < matrix.getSize(); j++)  
 {  
 if(j != i)  
 sum += matrix.getMatrix()[i][j] \* x[j];  
 }  
 x[i] = (matrix.getVector()[i] - sum) / matrix.getMatrix()[i][i];  
 esps.add(*abs*(x[i] - t));  
 if (*abs*(x[i] - t) > norma)  
 norma = *abs*(x[i] - t);  
 }  
 rs.addIter(x);  
 rs.addE(esps);  
 }  
 while(norma > eps);  
  
  
 rs.setResult(x);  
  
 *//Проверка* ArrayList<Double> residuals = new ArrayList<>();  
 for(int i = 0; i < matrix.getSize(); i++)  
 {  
 double S=0;  
 for(int j = 0; j < matrix.getSize(); j++)  
 {  
 S += matrix.getMatrix()[i][j] \* x[j] ;  
 }  
 residuals.add(S - matrix.getVector()[i]);  
 }  
 rs.setResiduals(residuals);  
 return rs;  
 }  
}

## Utils/PrintUtil

package lab1.Utils;  
  
import lab1.models.Matrix;  
  
import java.math.BigDecimal;  
import java.math.MathContext;  
import java.math.RoundingMode;  
import java.util.ArrayList;  
  
public class PrintUtil {  
  
 private final MathContext context = new MathContext(3, RoundingMode.*HALF\_UP*);  
 public PrintUtil() {  
 }  
 public static void println(String message) {  
 *println*(message);  
 }  
 public static void print(String message) {  
 System.*out*.print(message);  
 }  
  
 public void printVector(String message, ArrayList<Double> list){  
 *println*(message);  
 for(int i = 0; i < list.size(); i++){  
 *println*("["+(i+1)+"] = "+String.*format*("%.15f",list.get(i)));  
 }  
 }  
  
 public void printMatrix(Matrix matrix) {  
 *println*("Матрица:");  
 if (matrix != null) {  
 for (int i = 0; i < matrix.getSize(); i++) {  
 for (int j = 0; j < matrix.getSize() + 1; j++) {  
 *print*(String.*valueOf*(new BigDecimal(matrix.getMatrix()[i][j], context)));  
 *print*(" ");  
 }  
 *println*("");  
 }  
 }  
 }  
  
 public void notDiagonalAll() {  
 *println*("Отсутствие диагонального преобладания");  
 }  
 @Override  
 public boolean equals(Object obj) {  
 return super.equals(obj);  
 }  
  
  
 @Override  
 public int hashCode() {  
 return super.hashCode();  
 }  
  
 @Override  
 public String toString() {  
 return "Printer";  
 }  
}

## Models/ConsoleCommand

package lab1.models;  
import models.ICommand;  
import java.util.Scanner;  
import lab1.SimpleIterationMethod;  
  
import static lab1.Utils.PrintUtil.*println*;  
  
public class ConsoleCommand implements ICommand {  
 private final Scanner scanner = new Scanner(System.*in*);  
  
 @Override  
 public String getMessage() {  
 return "Ввести матрицу с консоли";  
 }  
  
 @Override  
 public void execute() {  
 *println*("Вводим матрицу с консоли");  
 Matrix matrix = createMatrixFromKeyBoard();  
 double eps;  
 while (true) {  
 try {  
 *println*("Введите точность:");  
 String buffer = scanner.nextLine();  
 eps = Double.*parseDouble*(buffer);  
 break;  
 }  
 catch (Exception ignored){ }  
 }  
 SimpleIterationMethod.*execute*(matrix, eps);  
 }  
  
 public Matrix createMatrixFromKeyBoard(){  
 try {  
 *println*("Введите размерность матрицы");  
 String buffer = scanner.nextLine();  
 buffer = buffer.trim();  
 int size = Integer.*parseInt*(buffer);  
 if (size > 20 || size <= 1) {  
 throw new Exception();  
 }  
 *println*("Введите строки матрицы");  
 double [][] matrix = new double[size][size+1];  
 String [][] arr = new String[size][size+1];  
 for (int i = 0; i < size;i++) {  
 buffer = scanner.nextLine();  
 arr[i] = buffer.trim().split(" ");  
 }  
 for (int i = 0; i < size;i++){  
 for (int j = 0; j < size+1;j++) {  
 matrix[i][j] = Double.*parseDouble*(arr[i][j].trim());  
 }  
 }  
 return new Matrix(matrix);  
 } catch (Exception e) {  
 *println*("Введена неверная размерность");  
 }  
 return null;  
 }  
}

## Models/DiagonalRandomCommand

package lab1.models;  
import models.ICommand;  
import java.util.Scanner;  
import lab1.SimpleIterationMethod;  
  
import static lab1.Utils.PrintUtil.*println*;  
import static lab1.models.RandomCommand.*inputRandomMatrix*;  
  
public class DiagonalRandomCommand implements ICommand {  
 private final Scanner scanner = new Scanner(System.*in*);  
  
 @Override  
 public String getMessage() { return "Генерация рандомной матрицы с диагональным преобладанием"; }  
  
 @Override  
 public void execute() {  
 *println*("Генерация матрицы c диагональным преобладанием");  
 Number[] values = *inputRandomMatrix*();  
 Matrix matrix;  
 do {  
 matrix = createRandomMatrix((Integer) values[0]);  
 } while (!SimpleIterationMethod.*checkDiagonal*(matrix.getMatrix(), (Integer) values[0]));  
 SimpleIterationMethod.*execute*(matrix, (Double) values[1]);  
 }  
  
 public Matrix createRandomMatrix(int size) {  
 try{  
 if (size > 20 || size <= 0) {  
 throw new Exception();  
 }  
 double[][] matrix = new double[size][size + 1];  
 int min = 1, max = 10;  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix.length + 1; j++) {  
 if(i == j){  
 matrix[i][j] = Math.*random*() \* ((max\*size - min\*size) + 1) + min\*size;  
 }  
 else {  
 matrix[i][j] = Math.*random*() \* ((max - min) + 1) + min;  
 }  
 }  
 }  
 return new Matrix(matrix);  
 } catch (Exception e) {  
 *println*("Введена неверная размерность");  
 }  
 return null;  
 }  
}

## Models/FileCommand

package lab1.models;  
  
import lab1.SimpleIterationMethod;  
import models.ICommand;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class FileCommand implements ICommand {  
 private final Scanner scanner = new Scanner(System.*in*);  
  
 @Override  
 public String getMessage() {  
 return "Ввести матрицу с файла";  
 }  
  
 @Override  
 public void execute() {  
 System.*out*.println("Вводим матрицу с файла");  
 System.*out*.println("Имя файла:");  
 String path = scanner.nextLine();  
 Matrix matrix = readMatrixFromFile(path);  
 double eps;  
 while (true) {  
 try {  
 System.*out*.println("Введите точность:");  
 String buffer = scanner.nextLine();  
 eps = Double.*parseDouble*(buffer);  
 break;  
 }  
 catch (Exception ignored){ }  
 }  
 SimpleIterationMethod.*execute*(matrix, eps);  
 }  
  
 public Matrix readMatrixFromFile(String fileName) {  
 try {  
 BufferedReader file = new BufferedReader(new FileReader(new File(fileName)));  
 int size = Integer.*parseInt*(file.readLine().trim());  
 double [][] matrix = new double[size][size + 1];  
 for (int i = 0; i < size; i++) {  
 String[] row = file.readLine().trim().split(" ");  
 if (row.length > size + 1)  
 throw new ArrayIndexOutOfBoundsException();  
 for (int j = 0; j < size + 1; j++) {  
 matrix[i][j] = Double.*parseDouble*(row[j].trim());  
 }  
 }  
 return new Matrix(matrix);  
 } catch (IOException e) {  
 System.*out*.println("Ошибка ввода");  
 }  
 return null;  
 }  
}

## Models/Matrix

package lab1.models;  
  
import java.util.Arrays;  
import java.util.Objects;  
  
public class Matrix {  
  
 private final double [][] matrix;  
 private final int size;  
  
 public Matrix(double [][] matrix) {  
 this.matrix = matrix;  
 this.size = matrix.length;  
 }  
  
 public double[][] getMatrix() {  
 return matrix;  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public double[] getVector(){  
 double [] vector = new double[size];  
 for(int i = 0; i < this.size; i++){  
 vector[i]=matrix[i][size];  
 }  
 return vector;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Matrix matrix1 = (Matrix) o;  
 return size == matrix1.size &&  
 Arrays.*deepEquals*(matrix, matrix1.matrix);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = Objects.*hash*(size);  
 result = 31 \* result + Arrays.*deepHashCode*(matrix);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "lab.Matrix{" +  
 "matrix=" + Arrays.*toString*(matrix) +  
 ", size=" + size +  
 '}';  
 }  
}

## Models/RandomCommand

package lab1.models;  
import models.ICommand;  
import java.util.Scanner;  
import lab1.SimpleIterationMethod;  
  
import static lab1.Utils.PrintUtil.println;  
  
public class RandomCommand implements ICommand {  
 private static final Scanner scanner = new Scanner(System.in);  
  
 @Override  
 public String getMessage() {  
 return "Генерация рандомной матрицы";  
 }  
  
 public static Number[] inputRandomMatrix() {  
 *println*("Генерация матрицы");  
 Number[] values = new Number[2];  
 int size;  
 double eps;  
 while (true) {  
 try {  
 *println*("Введите размер:");  
 String buffer = *scanner*.nextLine();  
 size = Integer.*parseInt*(buffer);  
 *println*("Введите точность:");  
 buffer = *scanner*.nextLine();  
 eps = Double.*parseDouble*(buffer);  
 break;  
 }  
 catch (Exception ignored){ }  
 }  
 values[0] = size;  
 values[1] = eps;  
 return values;  
 }  
  
 @Override  
 public void execute() {  
 Number[] values = *inputRandomMatrix*();  
 SimpleIterationMethod.*execute*(createRandomMatrix((Integer) values[0]), (Double) values[1]);  
 }  
  
 public Matrix createRandomMatrix(int size) {  
 try{  
 if (size > 20 || size <= 0) {  
 throw new Exception();  
 }  
 double[][] matrix = new double[size][size + 1];  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix.length + 1; j++) {  
 matrix[i][j] = Math.random() \* 50 - 25;  
 }  
 }  
 return new Matrix(matrix);  
 } catch (Exception e) {  
 println("Введена неверная размерность");  
 }  
 return null;  
 }  
}

## Models/ResultSet

package lab1.models;  
  
import java.util.ArrayList;  
  
public class ResultSet {  
 ArrayList<ArrayList<Double>> iteraz = new ArrayList<>();  
 ArrayList<Double> result = new ArrayList<>();  
 ArrayList<Double> residuals = new ArrayList<>();  
 ArrayList<ArrayList<Double>> e = new ArrayList<>();  
 public void addIter(double[] iter){  
 ArrayList<Double> arrayList = new ArrayList<>();  
 for(double it : iter){ arrayList.add(it); }  
 iteraz.add(arrayList);  
 }  
 public String getTable(){  
 StringBuilder stringBuilder = new StringBuilder();  
 stringBuilder.append("№ |");  
 for(int i = 0; i < iteraz.get(0).size(); i++) {  
 stringBuilder.append("x").append(i+1).append(" |");  
 }  
 stringBuilder.append(" ###| ");  
 for(int i = 0; i < iteraz.get(0).size(); i++) {  
 stringBuilder.append("eps").append(i+1).append(" |");  
 }  
 stringBuilder.append("\n").append("0 |");  
 for(int i = 0; i < iteraz.get(0).size(); i++) {  
 stringBuilder.append("0").append(" |");  
 }  
 stringBuilder.append("\n");  
 for(int i = 0; i < iteraz.size(); i++){  
 stringBuilder.append(i+1).append(" |");  
 for(Double it : iteraz.get(i)){  
 stringBuilder.append(String.format("%.6f",it)).append(" |");  
 }  
 stringBuilder.append(" ###| ");  
 try {  
 if (!e.isEmpty() && !e.get(i).isEmpty()) {  
 for (Double it : e.get(i)) {  
 stringBuilder.append(String.format("%.6f", it)).append(" |");  
 }  
 }  
 }catch (Exception ignored){}  
 stringBuilder.append("\n");  
 }  
 return stringBuilder.toString();  
 }  
  
 public void setResult(double[] res){  
 for(double it : res){  
 result.add(it);  
 }  
 }  
  
 public ArrayList<Double> getResult() {  
 return result;  
 }  
  
 public void addE(ArrayList<Double> e) {  
 this.e.add(e);  
 }  
  
 public ArrayList<Double> getResiduals() {  
 return residuals;  
 }  
  
 public void setResiduals(ArrayList<Double> residuals) {  
 this.residuals = residuals;  
 }  
}

## Commands/SystemsOfLinearAlgebraicEquations

package lab1.commands;  
import models.ICommand;  
import lab1.models.\*;  
import modules.MenuModule;  
import сommands.Main;  
  
import java.util.ArrayList;  
  
public class SystemsOfLinearAlgebraicEquations implements ICommand {  
 @Override  
 public String getMessage() {  
 return "Решение систем линейных уравнений";  
 }  
  
 @Override  
 public void execute() {  
 ArrayList<ICommand> commands = new ArrayList<>();  
 commands.add(new ConsoleCommand());  
 commands.add(new FileCommand());  
 commands.add(new RandomCommand());  
 commands.add(new DiagonalRandomCommand());  
 commands.add(new Main());  
 MenuModule menu = new MenuModule(commands);  
 menu.execute();  
 }  
}

# Блок-схема метода

# Примеры работы программы

Вычислительная математика. Васильченко Роман, P32081

Меню:

1. Ввести матрицу с консоли

2. Ввести матрицу с файла

3. Генерация рандомной матрицы

4. Генерация рандомной матрицы с диагональным преобладанием

5. Назад <--

4

Генерация матрицы c диагональным преобладанием

Введите размер:

3

Введите точность:

0.01

Матрица:

29.9 8.61 2.02 1.80

6.61 27.7 5.13 6.68

9.21 5.78 28.0 3.51

№ |x1 |x2 |x3 | ###| eps1 |eps2 |eps3 |

0 |0 |0 |0 |

1 |0,060194 |0,227234 |0,058718 | ###| 0,060194 |0,227234 |0,058718 |

2 |-0,009259 |0,232937 |0,080389 | ###| 0,069454 |0,005703 |0,021671 |

3 |-0,012366 |0,229656 |0,082088 | ###| 0,003106 |0,003281 |0,001699 |

Решение системы:

[1] = -0,012365905421796

[2] = 0,229656048143615

[3] = 0,082088166195279

Вектор невязки:

[1] = -0,024827104524236

[2] = 0,008725504657450

[3] = 0,000000000000000

Меню:

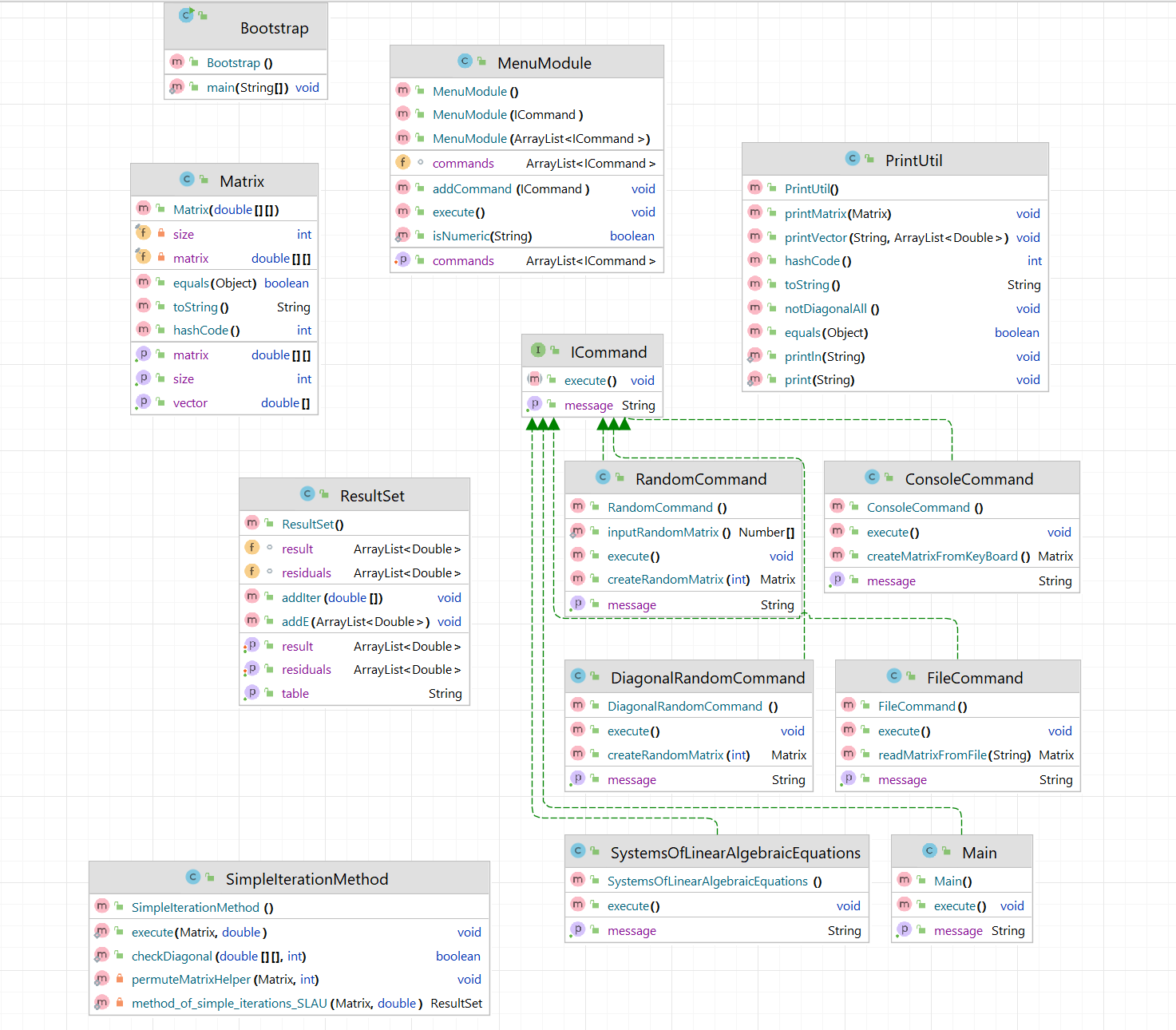
1. Ввести матрицу с консоли

2. Ввести матрицу с файла

3. Генерация рандомной матрицы

4. Генерация рандомной матрицы с диагональным преобладанием

5. Назад <--

Диаграмма классов программы

# Вывод:

В результате выполнения данной лабораторной работой я познакомилась с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав на языке программирования Java метод простых итераций.