Университет ИТМО, факультет ПИиКТ

Лабораторная работа №1

Дисциплина: Вычислительная математика

Вариант 19

Выполнил: Щелыкалов Виктор

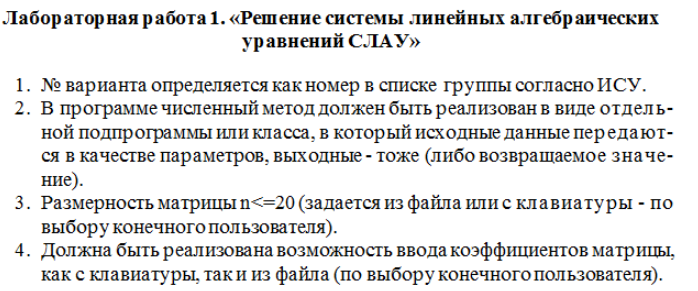
Группа: Р3214

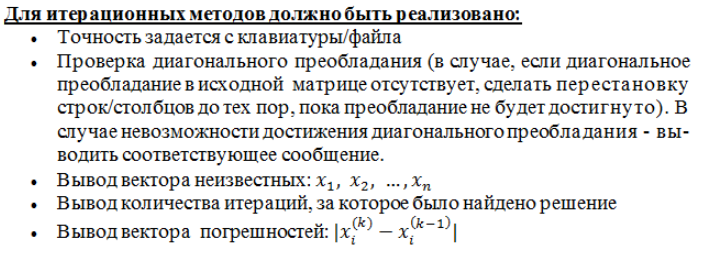
Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург, 2020 год

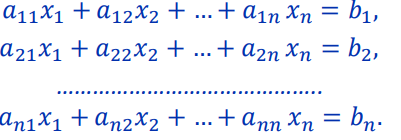
**Текст задания:**





**Цель работы:** изучить метод простых итераций для решения СЛАУ. Перевести математический алгоритм на машинный язык(Java).

**Ход работы:**

(1)

Итерационные методы дают возможность для системы (1) построить последовательность векторов 𝑥 (0) , 𝑥 (1) , … , 𝑥 (𝑘) , пределом которой должно быть точное решение 𝑥 (∗) :

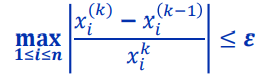
𝑥 (∗) = lim 𝑘→∞ 𝑥 (𝑘)

На практике построение последовательности заканчивается, как только достигается желаемая точность. Критерии окончания итерационного процесса:

Критерий по абсолютным отклонениям, наиболее простой и часто используемый способ – это сравнение между собой соответствующих неизвестных по двум соседним итерациям (k) и (k-1):



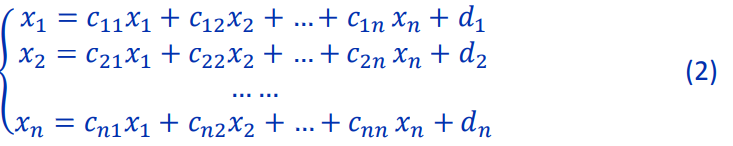
Критерий по относительным разностям:



Критерий по невязке, показывающий насколько полученное приближение 𝑥 𝑘отличается от точного решения:

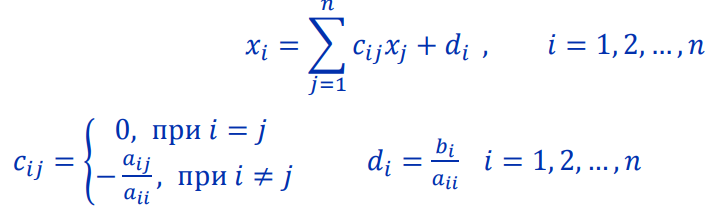


Систему(1) необходимо привести к виду:

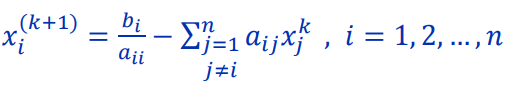


Или в векторно-матричном виде: 𝑥 = 𝐶𝑥 + 𝐷, где x – вектор неизвестных, 𝐶 ‒ матрица коэффициентов преобразованной системы размерности n\*n, 𝐷 ‒ вектор правых частей преобразованной системы.

Систему (2) представим в сокращенном виде:



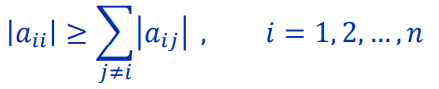
Рабочая формула метода простой итерации:



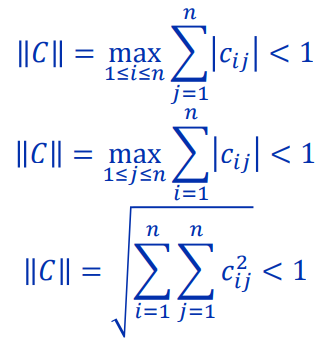
где k – номер итерации. За начальное (нулевое) приближение выбирают вектор свободных членов: 𝑥 (0) = 𝐷 или нулевой вектор. Следующее приближение:

𝑥 (1) = 𝑐𝑥 (0) + 𝑑 , 𝑥 (2) = 𝑐𝑥 (1) + 𝑑 … 𝑥 (𝑘) = 𝑐𝑥 (𝑘−1) + 𝑑

*Теорема.* Достаточным условием сходимости итерационного процесса к решению системы при любом начальном векторе 𝑥𝑖 (0) является выполнение условия преобладания диагональных элементов или доминирование диагонали:



*Теорема.* Достаточным условием сходимости итерационного метода к решению системы при любом начальном векторе 𝑥𝑖 (0) является требование к норме матрицы С: |𝐶 |< 1 (по строкам, или по столбцам, или по всей матрице)



**Листинг:**

public void doMethod() {

// Введем вектор значений неизвестных на предыдущей итерации,

// размер которого равен числу строк в матрице, т.е. size,

// причем согласно методу изначально заполняем его нулями

previousVariableValues = new double[size];

for (int i = 0; i < size; i++) {

previousVariableValues[i] = 0.0;

}

//Вектор погрешностей

epsVector = new double[size];

//число итераций

amount = -1;

// Будем выполнять итерационный процесс до тех пор,

// пока не будет достигнута необходимая точность

while (true) {

amount++;

// Введем вектор значений неизвестных на текущем шаге

double[] currentVariableValues = new double[size];

// Посчитаем значения неизвестных на текущей итерации

// в соответствии с теоретическими формулами

for (int i = 0; i < size; i++) {

// Инициализируем i-ую неизвестную значением

// свободного члена i-ой строки матрицы

currentVariableValues[i] = matrix[i][size];

// Вычитаем сумму по всем отличным от i-ой неизвестным

for (int j = 0; j < size; j++) {

if (i != j) {

currentVariableValues[i] -= matrix[i][j] \* previousVariableValues[j];

}

}

// Делим на коэффициент при i-ой неизвестной

currentVariableValues[i] /= matrix[i][i];

}

// Посчитаем текущую погрешность относительно предыдущей итерации

double error = -1.0;

for (int i = 0; i < size; i++) {

//System.out.println(previousVariableValues[i] + " " + currentVariableValues[i]);

epsVector[i] = Math.abs(previousVariableValues[i] - currentVariableValues[i]);

if (Math.abs(previousVariableValues[i] - currentVariableValues[i]) > error) {

error = Math.abs(previousVariableValues[i] - currentVariableValues[i]);

}

}

// Если необходимая точность достигнута, то завершаем процесс

if (error < eps) {

if (!Double.isNaN(currentVariableValues[0])) {

previousVariableValues = currentVariableValues;

}

break;

}

// Переходим к следующей итерации, так что текущие значения неизвестных

// становятся значениями на предыдущей итерации

previousVariableValues = currentVariableValues;

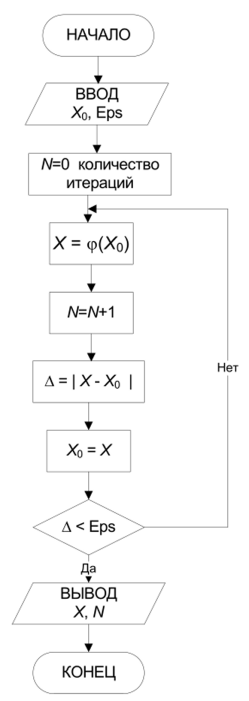
}

}

**Примеры работы программы:**

|  |  |
| --- | --- |
| Файл(1) или консоль(2):  2  Введите размер матрицы:  3  Введите элементы матрицы:  10 2 3 5  5 50 1 6  2 4 20 10  Введите точность:  0.0001  Имеем матрицу:  10.0 2.0 3.0 5.0  5.0 50.0 1.0 6.0  2.0 4.0 20.0 10.0  Вектор неизвестных: 0,34984 0,07601 0,44980  Вектор погрешности: 0,00007 0,00003 0,00004  Число итераций: 7 | Файл(1) или консоль(2):  2  Введите размер матрицы:  3  Введите элементы матрицы:  1 50 1 1  100 1 2 1  9 8 8 1  Введите точность:  0.0001  Имеем матрицу:  100.0 1.0 2.0 1.0  1.0 50.0 1.0 1.0  9.0 8.0 8.0 1.0  Вектор неизвестных: 0,00785 0,01788 0,09828  Вектор погрешности: 0,00001 0,00001 0,00006  Число итераций: 5 |
| Файл(1) или консоль(2):  1  Имеем матрицу:  10.0 1.0 2.0 1.0 1.0 3.0  1.0 6.0 1.0 1.0 1.0 2.0  2.0 3.0 4.0 1.0 2.0 2.0  1.0 1.0 3.0 7.0 1.0 4.0  1.0 2.0 3.0 1.0 8.0 1.0  Вектор неизвестных: 0,21048 0,21008 0,16673 0,45582 -0,06721 Вектор погрешности: 0,00374 0,00443 0,00999 0,00640 0,00619 Число итераций: 388 (точность 0.01) | Файл(1) или консоль(2):  1  Имеем матрицу:  9.0 1.0 2.0 1.0 1.0 2.0  2.0 33.0 2.0 1.0 2.0 1.0  3.0 4.0 9.0 2.0 2.0 2.0  1.0 3.0 4.0 9.0 3.0 1.0  2.0 9.0 1.0 1.0 7.0 2.0  Вектор неизвестных: 0,17229 -0,00090 0,11959 -0,04109 0,22074  Вектор погрешности: 0,00440 0,00174 0,00714 0,00833 0,00716  Число итераций: 15 (точность 0.01) |

**Блок-схема:**



**Вывод:** в результате выполнения работы я познакомился с методом простых итераций, теоремами необходимыми для сведения решения к данному методу и нюансами, при которых решение не является верным. Основной сложностью стал учет условий, налагаемых на СЛАУ.