Университет ИТМО
ФПИиКТ

Лабораторная работа №1
по Вычислительной математике

Выполнил: Балтабаев Дамир
Группа: P3210
Вариант: 4

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург
2022

Цель работы:

Реализовать решение СЛАУ в виде отдельной подпрограммы или класса, в который входные/выходные данные передаются в качестве параметров.

Задание:

Метод простых итераций

Для итерационных методов 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: $x_1,\ x_2,\ \dots, x_n$
- Вывод количества итераций, за которое было найдено решение.
- Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$

Описание метода, расчетные формулы:

Метод итерации или **метод простой итерации** — численный метод решения системы линейных алгебраических уравнений. Суть метода заключается в нахождении по приближённому значению величины следующего приближения, являющегося более точным.

Метод позволяет получить значения корней системы с заданной точностью в виде предела последовательности некоторых векторов (в результате итерационного процесса). Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения корня.

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

$$|a_{ii}| \geq \sum_{j \neq i} |aij|, i = 1..n$$

Рабочая формула метода простой итерации:

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j^k \ , \qquad i = 1, 2, \dots, n$$

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

Следующее приближение:

$$\vec{x}^{(1)} = c\vec{x}^{(0)} + \vec{d}, \quad \vec{x}^{(2)} = c\vec{x}^{(1)} + \vec{d} \ \dots$$

Итерация продолжается до тех пор, пока погрешность не приблизится к заданной точности.

Формула: $|x^{k+1} - x^k| \leq \varepsilon$

Листинг программы:

simpleIterationMethod() - метод, отвечающий за начало работы всего алгоритма, он распоряжается порядком выполнения того или иного пункта.

```
public void simpleIterationMethod(MatrixPOJO userMatrix) {

    MatrixPOJO finalUserMatrix;
    if (!checkDiagonalDominance(userMatrix)) {
        MatrixPOJO changedMatrix = changeRows(userMatrix);

        if (!checkDiagonalDominance(changedMatrix)) {
            messenger.diagonalDominatingIsMissingMessage();
            finalUserMatrix = userMatrix;
        } else {
            finalUserMatrix = changedMatrix;
            messenger.newChangedMatrixMessage();
            messenger.introducedMatrixMessage(finalUserMatrix.getMatrix());
        }

    } else finalUserMatrix = userMatrix;

    iterate(finalUserMatrix);


}
```

Предыдущий метод отправляет матрицу на проверку диагонального преобладания в метод checkDiagonalDominance(), который возвращает булевое значение в зависимости от проверки.

```
public boolean checkDiagonalDominance(MatrixPOJO userMatrix) {
    try {
        double rowSum = 0;
        int strictInequalityCounter = 0;
        for (int i = 0; i < userMatrix.getSize(); i++) {
            for (int j = 0; j < userMatrix.getSize(); j++) {
                if (i != j) {
                    rowSum += Math.abs(userMatrix.getMatrix()[i][j]);
                }
            }
            if (!(Math.abs(userMatrix.getMatrix()[i][i]) >= rowSum)) {
                return false;
            }
            if (Math.abs(userMatrix.getMatrix()[i][i]) > rowSum) {
                strictInequalityCounter++;
            }
            rowSum = 0;

        }

        if (strictInequalityCounter == 0) return false;
    } catch (NullPointerException nullPointerException) {
        return false;
    }
    return true;
}
```

В случае отсутствия диагонального преобладания, основной метод отправляет матрицу в метод changeRows(), который меняет строки матрицы местами, пытаясь достичь диагонального преобладания. Если полученная матрица после выхода из метода не проходит проверку на диагональное преобладание – с-ма выводит сообщение о невозможности достижения диаг. Преобладания.

```
public MatrixPOJO changeRows(MatrixPOJO userMatrix) {
    double maxElement = -10000;
    int maxElementColumnIndex = 0;
    Double[][] changedMatrix = new
Double[userMatrix.getSize()][userMatrix.getSize() + 1];
    for (int i = 0; i < userMatrix.getSize(); i++) {
        for (int j = 0; j < userMatrix.getSize(); j++) {
            if (userMatrix.getMatrix()[i][j] > maxElement) {
                maxElement = userMatrix.getMatrix()[i][j];
                maxElementColumnIndex = j;
            }
        }
        int k = 0;
        while (k < userMatrix.getSize() + 1) {
            changedMatrix[maxElementColumnIndex][k] =
userMatrix.getMatrix()[i][k];
            k++;
        }
        maxElement = 0;
        maxElementColumnIndex = 0;
    }
    return new MatrixPOJO(changedMatrix, userMatrix.getAccuracy(),
userMatrix.getSize());
}
```

Метод iterate() – основной метод, отвечающий за весь функционал метода простых итераций. Метод занимается поиском приближения по основной формуле и обеспечивает итерационный механизм. Также метод считает погрешность по основной формуле и в случае соблюдения условия окончания итерации – останавливает процесс.

```
public void iterate(MatrixPOJO userMatrix) {
    double previousErrorEstimate = 0;

    Double[] results = new Double[userMatrix.getSize()];
    for (int i = 0; i < userMatrix.getSize(); i++) {
        for (int j = 0; j < userMatrix.getSize() + 1; j++) {
            results[i] = userMatrix.getMatrix()[i][j];
        }
    }

    Double[] initialApproximation = new Double[userMatrix.getSize()];
    for (int i = 0; i < userMatrix.getSize(); i++) {
        initialApproximation[i] = 0.0;
    }

    int numberOfIterations = 0;
    Double[] newInitialApproximation = new Double[userMatrix.getSize()];
```

```
    while (true) {
        messenger.numberOfIterationMessage(numberOfIterations);
        double formula = 0;
        for (int i = 0; i < userMatrix.getSize(); i++) {
            for (int j = 0; j < userMatrix.getSize(); j++) {
                if (i != j) {
                    formula = formula + (-userMatrix.getMatrix()[i][j] *
initialApproximation[j]);
                } else continue;
            }
            int indexOfX = i + 1;
            formula = (results[i] + formula) / userMatrix.getMatrix()[i][i];
            messenger.computedXMessage(indexOfX, formula);
            newInitialApproximation[i] = formula;
            formula = 0;
        }

        if (numberOfIterations != 0) {
            double maxOfErrorVector = Math.abs(newInitialApproximation[0] -
initialApproximation[0]);
            for (int i = 0; i < userMatrix.getSize(); i++) {
                double def = Math.abs(newInitialApproximation[i] -
initialApproximation[i]);
                if (def > maxOfErrorVector) {
                    maxOfErrorVector = def;
                }
            }
            messenger.currentErrorEstimateMessage(maxOfErrorVector);
            if (previousErrorEstimate != 0 && maxOfErrorVector >
previousErrorEstimate) {
                System.out.println("Погрешность увеличилась!");
                return;
            }
            previousErrorEstimate = maxOfErrorVector;
            if (maxOfErrorVector <= userMatrix.getAccuracy()) return;
        }
        numberOfIterations++;
        for (int i = 0; i < userMatrix.getSize(); i++) {
            initialApproximation[i] = newInitialApproximation[i];
        }


    }


}
```

Пример работы программы:

```
Выберите метод ввода значений:
Введите цифру 1 в консоль для ввода с клавиатуры или цифру 2 для ввода с файла
1
Вы выбрали возможность ввода данных с КЛАВИАТУРЫ
Введите размерность матрицы (n<=20):
3
Размер матрицы = 3
Введите матрицу:
2 2 10 14
10 1 1 12
2 10 1 13
Текущая матрица:
2.0 2.0 10.0 14.0
10.0 1.0 1.0 12.0
2.0 10.0 1.0 13.0


Введите точность:
0.01
Введенная точность: 0.01
Произошла перестановка строк/столбцов для достижения диагонального преобладания
Текущая матрица:
10.0 1.0 1.0 12.0
2.0 10.0 1.0 13.0
2.0 2.0 10.0 14.0
```

```
Итерация № 0
x1 = 1.2
x2 = 1.3
x3 = 1.4
Итерация № 1
x1 = 0.93
x2 = 0.9199999999999999
x3 = 0.9
Оценка погрешности: 0.4999999999999999
Итерация № 2
x1 = 1.018
x2 = 1.024
x3 = 1.03
Оценка погрешности: 0.13
Итерация № 3
x1 = 0.9945999999999999
x2 = 0.9934000000000001
x3 = 0.9916
Оценка погрешности: 0.03839999999999999
Итерация № 4
x1 = 1.0015
x2 = 1.00192
x3 = 1.0024000000000002
Оценка погрешности: 0.010800000000000143
Итерация № 5
x1 = 0.999568
x2 = 0.99946
x3 = 0.999316
Оценка погрешности: 0.0030840000000001977

Process finished with exit code 0
```

Вывод:

В ходе выполнения данной лабораторной работы, я столкнулся с числовым методом простых итераций, позволяющим решить Систему линейных алгебраических уравнений быстрым способом. Изучил понятие диагонального преобладания, необходимое для соблюдения условия сходимости матрицы. Метод простых итераций показался мне довольно эффективным и не затрагивающим большого кол-ва памяти методом, ведь каждый фрагмент необходимых формул не сохраняется в память, а перезаписывается по мере необходимости. Из недостатков можно отметить относительную сложность метода, из-за кол-ва необходимых соблюдений условий.