Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №1

Метод Гаусса-Зейделя

Преподаватель: Малышева Татьяна Алексеевна

Выполнил: Ле Фан Фу Куок

P3212

Санкт-Петербург
2022

Описание метода

Метод Гаусса-Зейделя один из самых распространённых метод решения СЛАУ вида

$$Ax = b,$$

относится к категории итерационных методов. Их преимущество состоит уменьшении погрешности при вычислении вектора неизвестных до заданного значения. Недостатком является достаточно жёсткое условие сходимости итераций:

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

(при этом хотя бы для одного уравнения неравенство должно выполнятся строго). После проверки на сходимость нужно задать начальные приближение вектора x (обычно задают равными 0), и менять их на каждой итерации согласно формуле:

$$x_i^{(k)} = \frac{1}{a_{ii}}(b_k - a_{ij}x_{i+1}^{(k-1)} - a_{ij}x_{i+2}^{(k-1)})$$

Особенностью метода является использование предыдущих приближений $(k-1)$ при вычислении следующих (k) приближений. Итерации прекращаются, когда погрешность вычислений приблизится к заданной точности

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

Цель работы

Познакомиться с задачами, решаемыми в рамках дисциплины “Вычислительная математика”, на примере решения систем линейных алгебраических уравнений с помощью итерационного метода ГауссаЗейделя. Проанализировать полученные результаты и оценить погрешность.

Задание

Создать программу, которая решает систему линейных алгебраических

уравнений методом Гаусса-Зейделя.

Листинг программы

https://github.com/secarB/Compmath1

```java
import static java.lang.Math.abs;

public class Matrix {
    private int range;
    private int M = 1000;
    private int k = 1;
    private double[][] a;
    private double[] b;
    private double[] x;
    private double[] d = new double[M];
    private double efr;
    private double maxD = 0;
    private double D;
    private double s;
    private double x_I;
    public Matrix(int range, double[][] a, double[] b, double efr) {
        this.range = range;
        this.a = a.clone();
        this.b = b.clone();
        this.efr = efr;
        this.x = new double[range];
        for (int i = 0; i < range; i++) {
            x[i] = 0;
        }
        System.out.println("Matrix is created ");
        getMatrix();
    }
    public void solve() {
        if (isConverge(a,b)) {
            startSeidel();
        } else {
            System.out.println("Can't achieve diagonal dominance");
        }
    }
    private void startSeidel() {
        maxD = 0;
        for (int i = 0; i < range; i++) {
            s = b[i];
            for (int j = 0; j < range; j++) {
                if (j != i)
                {
                    s -= a[i][j] * x[j];
                }
            }
            x_I = s / a[i][i];
            D = abs(x_I - x[i]);
            if (D > maxD) maxD = D;
            x[i] = x_I;
        }
        d[k - 1] = maxD;
        if (maxD < efr) {
            System.out.println(maxD);
            getVectorX(x);
            getIteration(k);
            getError(d);
        } else if (k < M) {
            k++;
            startSeidel();
        } else {
            System.out.println("Iterations diverge.");
        }
```

```
    }

    public boolean isConverge(double[][] array,double[] bArray) {
        Diagonal diagonal = new Diagonal(array, bArray);
        boolean f = diagonal.findNewMatrix();
        if (f) {
            a = diagonal.getNewMatrix().clone();
            b = diagonal.getNewBMatrix().clone();
            System.out.println("Matrix with diagonal dominance:");
            getMatrix();
        }
        return f;
    }

    private void getVectorX(double[] x) {
        System.out.println("Solution X: ");
        for (int i = 0; i < range; i++) {
            System.out.printf("%10.4f", x[i]);
            System.out.println();
        }
    }

    private void getIteration(int k) {
        System.out.println("Number of iterations = " + k);
    }

    private void getError(double[] d) {
        System.out.println("Error vector: ");
        for (int i = 0; i < k; i++) {
            System.out.printf("%10.10f", d[i]);
            System.out.println();
        }
    }

    private void getMatrix() {
        for (int i = 0; i < range; i++) {
            for (int j = 0; j < range; j++) {
                System.out.printf("%6.1f", a[i][j]);
            }
            System.out.print(" | ");
            System.out.printf("%6.1f", b[i]);
            System.out.println();
        }
    }
}
```

Примеры

Enter type of input 1: from console 2: from file

1

Enter the range of the matrix

3

Enter the matrix

3 -2 1 6

-1 2 4 9

1 3 2 2

Enter error function

0.00000001

Matrix is created

```
  3.0  -2.0   1.0 |    6.0
 -1.0   2.0   4.0 |    9.0
  1.0   3.0   2.0 |    2.0
```

Matrix with diagonal dominance:

```
  3.0  -2.0   1.0 |    6.0
  1.0   3.0   2.0 |    2.0
 -1.0   2.0   4.0 |    9.0
```

8.12195835586671E-9

Solution X:

```
   0.1220
  -1.3415
   2.9512
```

Number of iterations = 20

Error vector:

2.7500000000

1.5277777778

1.1967592593

0.1685742455

0.1292206076

0.0186000352

0.0138411776

0.0024511477

0.0014698212

0.0003131781

0.0001546141

0.0000390924

0.0000160934

0.0000047906

0.0000016551

0.0000005783

0.0000001678

0.0000000689

0.0000000167

0.0000000081

Выводы.

На лабораторных работах я познакомился с итерационным методом Гаусса-Зейделя для решения СЛАУ.