Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки:

Системное и Прикладное Программное Обеспечение

(09.03.04 Программная инженерия)

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №1

Вариант №10

Студент

Карташев Владимир Сергеевич,
группа P3215

Преподаватель / Практик

Малышева Татьяна Алексеевна

г. Санкт-Петербург, 2024 г.

Цель работы

Написать программу, решающую СЛАУ методом простых итераций.

Описание метода, расчетные формулы

Метод простой итерации (или метод простых итераций) — это численный метод решения систем линейных алгебраических уравнений (СЛАУ). Он основан на итеративном приближении к решению.

Формулы:

$$x_1^{k+1} = -0{,}1x_2^k - 0{,}1x_3^k + 1{,}2$$
$$x_2^{k+1} = -0{,}2x_1^k - 0{,}1x_3^k + 1{,}3$$
$$x_3^{k+1} = -0{,}2x_1^k - 0{,}2x_2^k + 1{,}4$$

Листинг программы

```
currentValues := make([]float64, len(resultVector))  // значения x1, x2, x3, ..., xn на текущей итерации
previousValues := make([]float64, len(resultVector)) // значения x1, x2, x3, ..., xn с предыдущей итерации

var iteration int
for iteration = 1; iteration <= 1000; iteration++ { // Максимальное количество итераций
    var maxDiff float64 // Максимальная абсолютная величина разности

    for i := range resultVector {
        sum := 0.0

        // формула, реализующаяся в цикле:
        // new_x1 = x1 - b1 - c1 / r1
        // new_x2 = a2 - x2 - c2 / r2
        // new_x3 = a3 - b3 - x3 / r3
        // ...

        for j := range resultVector {
            // найдем сумму элементов строки без диагонального элемента
            if i != j {
                sum += coefficientMatrix[i][j] * previousValues[j]
            }
        }

        // подставляем значения строки в формулу
        newValue := (resultVector[i] - sum) / coefficientMatrix[i][i]
        // сохраняем новые значения x1, x2, x3, ..., xn в массив
        currentValues[i] = newValue

        diff := math.Abs(newValue - previousValues[i]) // находим абсолютную величину разности
        // сравниваем с максимальной. Если больше -> новая максимальная величина
        if diff > maxDiff {
            maxDiff = diff
        }
    }

    fmt.Printf( format: "Итерация %d: %v\n", iteration, currentValues)

    if maxDiff < epsilon {
        break
    } // если максимальная абсолютная величина разности < вероятности -> стоп, мы нашли ответ
    // иначе - продолжаем
```

Примеры и результаты работы программы

Пример входных данных:
файл с названием input

содержимое файла input:

```
0.01
20 2 3 7 5
1 12 -2 -5 4
5 -3 13 0 -3
0 0 -1 15 7
```

Первые n-1 значений - коэффициенты, последний - чему равно уравнение

Работа программы:

```
Выберите тип ввода
1. Ввод из файла
2. Ввод с консоли
Введите цифру:
1

Введите название файла:
input

Матрица коэффициентов:
20      2       3       7
1       12      -2      -5
5       -3      13      0
0       0       -1      15

Вектор результатов уравнения:
5
4
-3
7

Epsilon: 0.01

Вектор неизвестных:
X1 X2 X3 X4
```

Расчет:
Итерация 1: [0.25 0.3333333333333333 -0.2307692307692308 0.4666666666666667]
Итерация 2: [0.08794871794871795 0.4684829059829060 -0.25 0.4512820512820513]
Итерация 3: [0.08270299145299145 0.47237179487179487 -0.1564842209072978 0.45]
Итерация 4: [0.06873545364891517 0.4878607138943675 -0.1535691978961209 0.4562343852728468]
Итерация 5: [0.06456727344948499 0.4921081730769230 -0.1446227020431902 0.4564287201402586]

Решение сошлось после 5 итераций
Результат: [0.06456727344948499 0.4921081730769230 -0.1446227020431902 0.4564287201402586]

Выводы

В данной лабораторной работе я познакомился и испытал на практике решения СЛАУ методом простых итераций. Также я познакомился с языком Golang (затронутый синтаксис: циклы, срезы, двумерные срезы).