*Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники

–

Лабораторная работа №1 по дисциплине «Вычислительная математика»
Вариант 12

Выполнил:
Студент группы P3212
Медведев Ярослав Александрович
Преподаватель:
Машина Екатерина Сергеевна

г. Санкт-Петербург
2025

Цель работы

Реализация итерационного метода Гаусса - Зейделя для решения СЛАУ на языке программирования Golang

Описание метода

Метод Гаусса-Зейделя является модификацией метода простой итерации и обеспечивает более быструю сходимость к решению систем уравнений.

Так же как и в методе простых итераций строится эквивалентная СЛАУ и за начальное приближение принимается вектор правых частей (как правило, но может быть выбран и нулевой, и единичный вектор): $x_i^0 = (d_1, d_2, \dots, d_n)$.

$$
\begin{aligned}
x_1 &= c_{11}x_1 + c_{12}x_2 + \dots + c_{1n}x_n + d_1 \\
x_2 &= c_{21}x_1 + c_{22}x_2 + \dots + c_{2n}x_n + d_2 \\
&\dots\ \dots \\
x_n &= c_{n1}x_1 + c_{n2}x_2 + \dots + c_{nn}x_n + d_n
\end{aligned}
$$

Идея метода: при вычислении компонента $x_i^{(k+1)}$ на (k+1)-й итерации используются $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{i-1}^{(k+1)}$, уже вычисленные на ($k$+1)-й итерации.

Значения остальных компонент $x_{i+1}^{(k+1)}, x_{i+2}^{(k+1)}, \dots, x_n^{(k+1)}$ берутся из предыдущей итерации.

Схема для $k = 1$:

$x_1^1 \to x_2^0 \quad x_3^0 \quad x_4^0$

$x_2^1 \to x_1^1 \quad x_3^0 \quad x_4^0$

$x_3^1 \to x_1^1 \quad x_2^1 \quad x_4^0$

$x_4^1 \to x_1^1 \quad x_2^1 \quad x_3^1$

Схема для $k = 2$:

$x_1^2 \to x_2^1 \quad x_3^1 \quad x_4^1$

$x_2^2 \to x_1^2 \quad x_3^1 \quad x_4^1$

$x_3^2 \to x_1^2 \quad x_2^2 \quad x_4^1$

$x_4^2 \to x_1^2 \quad x_2^2 \quad x_3^2$

Тогда приближения к решению системы методом Зейделя определяются следующей системой равенств:

$$x_1^{(k+1)} = c_{11}x_1^{(k)} + c_{12}x_2^{(k)} + \cdots + c_{1n}x_n^{(k)} + d_1$$
$$x_2^{(k+1)} = c_{21}x_1^{(k+1)} + c_{22}x_2^{(k)} + \cdots + c_{2n}x_n^{(k)} + d_2$$
$$x_3^{(k+1)} = c_{31}x_1^{(k+1)} + c_{32}x_2^{(k+1)} + c_{33}x_3^{(k)} \ldots + c_{3n}x_n^{(k)} + d_3$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$x_n^{(k+1)} = c_{n1}x_1^{(k+1)} + c_{n2}x_2^{(k+1)} + \cdots + c_{n\,n-1}x_{n-1}^{(k+1)} + c_{nn}x_n^{(k)} + d_n$$

Рабочая формула метода Гаусса-Зейделя:

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

Итерационный процесс продолжается до тех пор, пока:

$$|x_1^{(k)} - x_1^{(k-1)}| \le \varepsilon, \quad |x_2^{(k)} - x_2^{(k-1)}| \le \varepsilon, \quad |x_3^{(k)} - x_3^{(k-1)}| \le \varepsilon$$

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

$$|a_{ii}| \ge \sum_{j \ne i} |a_{ij}|, \qquad i = 1, 2, \ldots, n$$

При этом хотя бы для одного уравнения неравенство должно выполняться строго.

Листинг программы

Полный код программы:
https://github.com/gr1ckly/Computional_Mathematic/tree/master/Lab_1

Реализация преобразования для диагонального преобладания и метода Гаусса-Зейделя

```
type Matrix struct {
   size, maxIter int
   data          [][]float64
   vecB          []float64
```

```
    accuracy      float64
}

func (matrix *Matrix) IsDiagonallyDominant() bool {
    for i := 0; i < matrix.size; i++ {
        sum := 0.0
        for j := 0; j < matrix.size; j++ {
            if i != j {
                sum += math.Abs(matrix.data[i][j])
            }
        }
        if math.Abs(matrix.data[i][i]) <= sum {
            return false
        }
    }
    return true
}

func (matrix *Matrix) MakeDiagonallyDominant() bool {
    for i := 0; i < matrix.size; i++ {
        maxRow := i
        maxVal := math.Abs(matrix.data[i][i])
        for k := i + 1; k < matrix.size; k++ {
            if math.Abs(matrix.data[k][i]) > maxVal {
                maxVal = math.Abs(matrix.data[k][i])
                maxRow = k
            }
        }
        if maxRow != i {
            matrix.data[i], matrix.data[maxRow] =
matrix.data[maxRow], matrix.data[i]
            matrix.vecB[i], matrix.vecB[maxRow] =
matrix.vecB[maxRow], matrix.vecB[i]
        }
    }
    if matrix.IsDiagonallyDominant() {
        return true
    } else {
        return false
    }
}

func (matrix *Matrix) GaussSeidel() ([]float64, int, []float64,
error) {
    x := make([]float64, matrix.size)
    for idx, _ := range x {
        x[idx] = matrix.vecB[idx] / matrix.data[idx][idx]
    }
```

```
   errors := make([]float64, matrix.size)

   for iter := 0; iter < matrix.maxIter; iter++ {
      maxDiff := 0.0
      for i := 0; i < matrix.size; i++ {
         sum := 0.0
         for j := 0; j < matrix.size; j++ {
            if j != i {
               sum += matrix.data[i][j] * x[j]
            }
         }
         newX := float64(matrix.vecB[i]-sum) /
float64(matrix.data[i][i])
         errors[i] = math.Abs(newX - x[i])
         if errors[i] > maxDiff {
            maxDiff = errors[i]
         }
         x[i] = newX
      }
      if maxDiff <= matrix.accuracy {
         return x, iter + 1, errors, nil
      }
   }
   return nil, matrix.maxIter, errors, fmt.Errorf("Решения за %d
итераций при указанной точности не найдено\n", matrix.maxIter)
}
```

Примеры работы программы

Работа программы при вводе данных через консоль для матрицы с диагональным преобладанием:

```
Введите любой символ, если хотите ввести данные из файла:
Введите размер матрицы: 3
Введите матрицу:
4 -1 1
4 -8 1
-2 1 5
Введите вектор b:
7 -21 15
Введите максимальное количество итераций: 50
Введите точность: 0.05
Диагональное преобразование найдено:
4 -1 1 7
4 -8 1 -21
-2 1 5 15
Норма матрицы: 13.000000
Решено за 3 итераций
Отклонения:
0.012460937499999991
0.018798828125
0.0012246093750003517
Решение:
1.9952734375
3.997314453125
2.998646484375
```

Работа программы при вводе из файла input.txt матрицы без диагонального преобладания:

Содержимое файла:

```
3
1 2 3
4 5 6
7 8 10
6 15 25
1000
0.01
```

Работа программы:

```
Введите любой символ, если хотите ввести данные из файла: итпа
Введите название файла: input.txt
Диагональное преобразование невозможно
Норма матрицы: 25.000000
Решено за 41 итераций
Отклонения:
0.004023338001721166
0.009894750696977095
0.005255387797410971
Решение:
0.9313527700190056
1.168826786746455
0.9103312188293614
```

Вывод

В процессе выполнения данной лабораторной работы я смог реализовать итерационный метод Гаусса - Зейделя для решения СЛАУ на языке программирования Golang.