Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1

«Решение системы линейных алгебраических уравнений СЛАУ»

Вариант 10

Выполнил:

Марьин Григорий Алексеевич

Группа P3212

Проверил:

Доцент ПИиКТ, кандидат технических наук

Малышева Татьяна Алексеевна

Санкт-Петербург

2026

Оглавление

Цель работы:

Изучить численные методы решения систем линейных уравнений, узнать различные методы, реализовать один из них.

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). В своем решении за начальное приближение я брал нулевой вектор.

Идея метода:

При вычислении компонента $x_i^{(k+1)}$ на (k+1)-й итерации используются $x_1^{(k+1)}$, x_2^{k+1} ..., x_{i-1}^{k+1}, уже вычисленные на (k+1)-й итерации. Значения остальных компонент x_{i+1} берутся из предыдущей итерации.

Рабочая формула:

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

Итерационный процесс продолжается до тех пор, пока:

$$\left| x_i^{(k)} - x_i^{(k-1)} \right| \leq \epsilon$$

Листинг программы:

```
x = [0.0] * self.n
        iterations = 0
        errors = [0.0] * self.n

        while True:
            x_new = x[:]
            current_errors = [0.0] * self.n

            for i in range(self.n):
                s1 = sum(self.A[i][j] * x_new[j] for j in range(i))
                s2 = sum(self.A[i][j] * x[j] for j in range(i + 1, self.n))

                if self.A[i][i] == 0:
                    print("\nНулевой элемент на главной диагонали, метод неприменим.")
                    return None, iterations, None

                val = (self.b[i] - s1 - s2) / self.A[i][i]

                current_errors[i] = abs(val - x[i])
```

```
            x_new[i] = val


        x = x_new
        iterations += 1
        errors = current_errors
        if max(errors) < self.epsilon:
            return x, iterations, errors
```

Примеры и результаты работы программы:

1. Входные данные:

```
3
10 1 1
2 10 1
2 2 10
12 13 14
0.0001
```

Результат:

```
Выберите режим ввода данных:
1. С клавиатуры
2. Из файла
Ваш выбор (1 или 2): 2
Введите имя файла: 1.txt
Норма матрицы: 14.0000

=== Ответ: ===
Количество итераций: 5

Вектор неизвестных x:
x1 = 1.000000
x2 = 1.000000
x3 = 1.000000

Вектор погрешностей |x(k) - x(k-1)|:
e1 = 0.0000228931
e2 = 0.0000002354
e3 = 0.0000046257
```

2. Входные данные:

```
5
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 29
21 22 23 24 25
100 101 102 103 104
0.00001
```

Результат:

```
Выберите режим ввода данных:
1. С клавиатуры
2. Из файла
Ваш выбор (1 или 2): 2
Введите имя файла: 2.txt
Диагональное преобладание отсутствует.
Невозможно достичь диагонального преобладания перестановкой строк.
Норма матрицы: 115.0000

=== Ответ: ===
Количество итераций: 150

Вектор неизвестных x:
x1 = 4.952346
x2 = -109.304661
x3 = 104.552289
x4 = 0.000006
x5 = 0.000017

Вектор погрешностей |x(k) - x(k-1)|:
e1 = 0.0000034756
e2 = 0.0000098886
e3 = 0.0000089943
e4 = 0.0000005934
e5 = 0.0000016615
```

Вывод:
В ходе лабораторной работы я познакомился с численными методами решения системы линейных уравнений. Реализовал один из них на Python.