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

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1
Вариант №12
Метод Гаусса-Зейделя

Выполнил
Путинцев Данил Денисович
Группа P3207
Проверил(а)
Преподаватель: Рыбаков Степан Дмитриевич

Санкт-Петербург 2025 год

Цель работы

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них средствами программирования

Описание метода, расчетные формулы

Метод Гаусса-Зейделя является модификацией метода простой итерации и обеспечивает более быструю сходимость к решению системы уравнений. Идея метода: при вычислении компонента $x_i^{(k+1)}$вектора неизвестных на (k+1)-й итерации используются $x_1^{(k+1)}, x_2^{(k+1)},\ldots, x_{i-1}^{(k+1)}$ уже вычисленные на (k+1)-й итерации. Значения остальных компонент $x_{i+1}^{(k+1)}, x_{i+2}^{(k+1)}, \ldots ,$ берутся из предыдущей итерации.

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 вектор): $x_i^0=(d_1,d_2,\ldots,d_n)$

$$x_1=c_{11}x_1+c_{12}x_2+\ldots+c_{1n}x_n+d_1$$

$$x_2=c_{21}x_1+c_{22}x_2+\ldots+c_{2n}x_n+d_2$$

….

$$x_n=c_{n1}x_1+c_{n2}x_2+\ldots+c_{nn}x_n+d_n$$

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

$$x_1^{(k+1)}=c_{11}x_1^{(k)}+c_{12}x_2^{(k)}+\ldots+c_{1n}x_n^{(k)}+d_1$$

$$x_2^{(k+1)}=c_{21}x_1^{(k+1)}+c_{22}x_2^{(k)}+\ldots+c_{2n}x_n^{(k)}+d_2$$

$$x_3^{(k+1)}=c_{31}x_1^{(k+1)}+c_{32}x_2^{(k+1)}+c_{33}x_3^{(k)}+\ldots+c_{2n}x_n^{(k)}+d_3$$

…

$$x_n^{(k+1)}=c_{n1}x_1^{(k+1)}+c_{n2}x_2^{(k+1)}+c_{n3}x_3^{(k+1)}+\ldots+c_{nn-1}x_{n-1}^{(k+1)}+c_{nn}x_n^{(k)}+d_n$$

Рабочая формула метода Гаусса-Зейделя:

$$x_i^{(k+1)}=\frac{b_i}{a_{ii}}-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{k+1}-\sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^{k} \quad i=1,2,3,\ldots,n$$

Итерационный процесс продолжается до тех пор, пока:

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

Листинг программы (по крайне мере, где реализован сам метод)

https://github.com/danp1t/ITMO/tree/main/comp_math/lab1

```python
def seidel(matrix, epsilon, max_iterations):
    if not is_diagonally_dominant(matrix):
        matrix = rearrange_matrix(matrix)
        if not is_diagonally_dominant(matrix):
            print("Невозможно достичь диагонального преобладания")

    a = [row[:-1] for row in matrix]
    b = [row[-1] for row in matrix]
    n = len(matrix)
    x = [0.0 for _ in range(n)]
    k = 1

    while k <= max_iterations:
        current_errors = []
        delta = 0
        for i in range(n):
            s = 0
            for j in range(n):
                if j != i:
                    s += a[i][j] * x[j]
            new_x = (b[i] - s) / a[i][i]
            d = abs(new_x - x[i])
            current_errors.append(d)

            if d > delta:
                delta = d
            x[i] = new_x

        current_errors.sort()

        if delta < epsilon:
            print(f"Решение найдено за {k} итераций.")
            return x
        k += 1

    return "Итерации расходятся"

def is_diagonally_dominant(matrix):
    n = len(matrix)
    for i in range(n):
        diag = abs(matrix[i][i])
        row_sum = sum(abs(matrix[i][j]) for j in range(n) if j != i)
        if diag <= row_sum:
            return False
    return True

def rearrange_matrix(matrix):
    n = len(matrix)
```

```python
new_matrix = [row.copy() for row in matrix]

for col in range(n):
    max_row = col
    for row in range(col, n):
        if abs(new_matrix[row][col]) > abs(new_matrix[max_row][col]):
            max_row = row

    new_matrix[col], new_matrix[max_row] = new_matrix[max_row], new_matrix[col]

return new_matrix
```

Примеры и результаты работы программы.

```
Введите команду: /input_n
Введите размерность матрицы: 2
Введите команду: /input_epsilon
Введите точность: 0.0001
Введите команду: /input_matrix

Введите матрицу 2x2 построчно. Числа разделяйте пробелами:
Строка 1: 1 -1 7
Строка 2: 3 2 16
Введите команду: /start
Невозможно достичь диагонального преобладания
Решение найдено за 25 итераций.
[5.999603978719565, -1.0000396021280435]
Введите команду: 
```

```
Строка 1: 3 5 11
Строка 2: 4 8 16
Введите команду: /info
Точность:  0.0001
Размерность:  2
Матрица:
3.0 5.0 11.0
4.0 8.0 16.0

Введите команду: /start
Невозможно достичь диагонального преобладания
Итерации расходятся
Введите команду: /input_epsilon
Введите точность: 0.5
Введите команду: /info
Точность:  0.5
Размерность:  2
Матрица:
3.0 5.0 11.0
4.0 8.0 16.0

Введите команду: /start
Невозможно достичь диагонального преобладания
Решение найдено за 2 итераций.
[4.4, -0.44000000000000002]
```

```
Введите команду: /input_n
Введите размерность матрицы: 3
Введите команду: /input_epsilon
Введите точность: 0.01
Введите команду: /input_matrix

Введите матрицу 3x3 построчно. Числа разделяйте пробелами:
Строка 1: 1 2 0 10
Строка 2: 3 2 1 23
Строка 3: 0 1 2 13
Введите команду: /info
Точность:  0.01
Размерность:  3
Матрица:
1.0 2.0 0.0 10.0
3.0 2.0 1.0 23.0
0.0 1.0 2.0 13.0

Введите команду: /start
Невозможно достичь диагонального преобладания
Решение найдено за 7 итераций.
[4.000895182291667, 2.9995524088541665, 5.000223795572917]
```

Выводы.

На этой лабораторной работе я научился реализовывать метод Гаусса-Зейделя для решения системы линейных уравнений