МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ПЕТРОВ АНДРЕЙ АЛЕКСАНДРОВИЧ

Отчет по лабораторной работе № 2
вариант 16
(«Методы вычислений»)

Выполнил:
студент 2 курса 14 группы
Петров Андрей Александрович

Преподавать:
Бондарь И.В.

Минск
2021

УСЛОВИЕ

Задание 4. Симметричный метод Гаусса-Зейделя

Симметричный метод Гаусса-Зейделя (ГЗ) представляет собой композицию обычного и обратного методов ГЗ. То есть, для вычисления x^{k+1} сначала находится промежуточное приближение $x^{k+\frac{1}{2}}$ стандартным методом ГЗ, а затем, начиная с $x^{k+\frac{1}{2}}$ выполняется одна итерация обратного метода ГЗ (в котором обновление компонент осуществляется в обратном порядке, с n-й по 1-ю). Полученный результат и будет новым приближением x^{k+1}.

Постановка задачи

1. Записать симметричный метод ГЗ в виде стационарного итерационного метода $x^{k+1} = Bx^k + g$ (найти вид матрицы B и вектора g).
2. Для указанной в варианте матрицы из списка 1 (см. ниже) теоретически исследовать сходимость симметричного и простого методов ГЗ.
3. Подтвердить теоретические результаты экспериментально путем построения логарифмической диаграммы сходимости обоих методов.

15. $\begin{pmatrix} -2 & -2 & -3 \\ 0 & -4 & 0 \\ 4 & 0 & -1 \end{pmatrix}$

ХОД ВЫПОЛНЕНИЯ

ЗАДАНИЕ 4.1.

Выразим вектор g и матрицу B для симметричного метода Гаусса-Зейделя:

$$A = L + D + R$$

$$x_i^{k+\frac{1}{2}} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+\frac{1}{2}} - \sum_{j=i+1}^{n} a_{ij} x_j^{k}\right.$$

$$x^{k+\frac{1}{2}} = D^{-1}(b - Lx^{k+\frac{1}{2}} - Rx^k$$

$$Dx^{k+\frac{1}{2}} = b - Lx^{k+\frac{1}{2}} - Rx^k$$

$$(D+L)x^{k+\frac{1}{2}} = b - Rx^k$$

$$x^{k+\frac{1}{2}} = -(D+L)^{-1}Rx^k + (D+L)^{-1}b$$

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

$$x^{k+1} = D^{-1}(b - Rx^{k+1} - Lx^{k+\frac{1}{2}})$$

$$Dx^{k+1} = b - Rx^{k+1} - Lx^{k+\frac{1}{2}}$$

$$(D+R)x^{k+1} = b - Lx^{k+\frac{1}{2}}$$

$$x^{k+1} = -(D+R)^{-1}Lx^{k+\frac{1}{2}} + (D+R)^{-1}b$$

$$x^{k+1} = -(D+R)^{-1}L(-(D+L)^{-1}Rx^k + (D+L)^{-1}b) + (D+R)^{-1}b$$

$$x^{k+1} = (D+R)^{-1}L(D+L)^{-1}Rx^k - (D+R)^{-1}L\cdot(D+L)^{-1}b + (D+R)^{-1}b$$

$$x^{k+1} = (D+R)^{-1}L(D+L)^{-1}Rx^k - (D+R)^{-1}(L(D+L)^{-1} + I)b$$

$$B = (D+R)^{-1}L(D+L)^{-1}R$$

$$g = -(D+R)^{-1}(L(D+L)^{-1} + I)b$$

ЗАДАНИЕ 4.2.

Найдем матрицы B для матрицы из условия A.

Для стандартного метода Гаусса-Зейделя:

$$\begin{pmatrix} 0 & -1 & -3/2 \\ 0 & 0 & 0 \\ 0 & -4 & -6 \end{pmatrix}$$

Для симметричного метода Гаусса-Зейделя:

$$\begin{pmatrix} 0 & 6 & 9 \\ 0 & 0 & 0 \\ 0 & -4 & -6 \end{pmatrix}$$

У обеих этих матриц спектральный радиус больше единицы, поэтому для матрицы данной в условии ни стандартный, ни симметричный методы Гаусса-Зейделя не сходятся.

ЗАДАНИЕ 4.3.

Реализуем стандартный метод Гаусса-Зейделя. currentSum хранит сумму, посчитанную на основе компонент вектора x, уже вычисленных на этой итерации, а previousSum – на прошлой итерации. Затем для вычисления компоненты вектора x из соответствующего свободного члена вычислим посчитанные до этого суммы и сделаем на соответствующий диагональный элемент матрицы. Повторим итерации до выполнения условия остановки.

Код:

```
public static double[] StandardGaussSeidel(double[,] matrix, double[]
freeMembers, double[] startState, double accuracy)
{
    double[] state = startState;
    int len = freeMembers.Length;
    while (!StopCondition(matrix, freeMembers, state, accuracy)) {
        double[] previousState = state;
        for (int i = 0; i < len; i++) {
            double currentSum = 0;
            double previousSum = 0;
            for (int j = 0; j < i; j++) {
                currentSum += matrix[i, j] * state[j];
            }

            for (int j = i + 1; j < len; j++) {
                previousSum += matrix[i, j] * previousState[j];
            }

            state[i] = (freeMembers[i] - currentSum - previousSum) /
matrix[i, i];
        }
    }
    return state;
}
```

Реализуем симметричный метод Гаусса-Зейделя. Данная реализация состоит из двух частей. Первая часть аналогична стандартному методу Гаусса-Зейделя, описанному выше, а вторая часть реализует обратный метод Гаусса-Зейделя, она аналогична первой, за исключением того, что компоненты вектора x вычисляются в обратном порядке.

Код:

```
public static double[] SymmetricGaussSeidel(double[,] matrix, double[]
freeMembers, double[] startState, double accuracy)
{
    double[] state = startState;
    int len = freeMembers.Length;
    while (!StopCondition(matrix, freeMembers, state, accuracy)) {
        double[] previousState = state;
        for (int i = 0; i < len; i++) {
            double currentSum = 0;
            double previousSum = 0;
            for (int j = 0; j < i; j++) {
                currentSum += matrix[i, j] * state[j];
            }
            for (int j = i + 1; j < len; j++) {
                previousSum += matrix[i, j] * previousState[j];
            }
            state[i] = (freeMembers[i] - currentSum - previousSum) /
                             matrix[i, i];
        }
        previousState = state;
        for (int i = len-1; i >=0; i--) {
            double currentSum = 0;
            double previousSum = 0;
            for (int j = 0; j < i; j++) {
                previousSum += matrix[i, j] * previousState[j];
            }
            for (int j = i + 1; j < len; j++) {
                currentSum += matrix[i, j] * state[j];
            }
            state[i] = (freeMembers[i] - currentSum - previousSum) /
                             matrix[i, i];
        }
    }
    return state;
}
```

В качестве условия остановки используем данную функцию, где рассчитаем норму невязки и сравним ее с заданной точностью.

Код:

```
private static bool StopCondition(double[,] matrix, double[]
freeMembers, double[] state, double accuracy)
{
    double norm=0;
    int len = freeMembers.Length;
    for (int i = 0; i < len; i++) {
        double currentValue=0;
        for (int j = 0; j < len; j++) {
            currentValue += matrix[i, j] * state[j];
        }
        norm += Math.Pow(freeMembers[i] - currentValue,2);
    }
    return norm < accuracy;
}
```

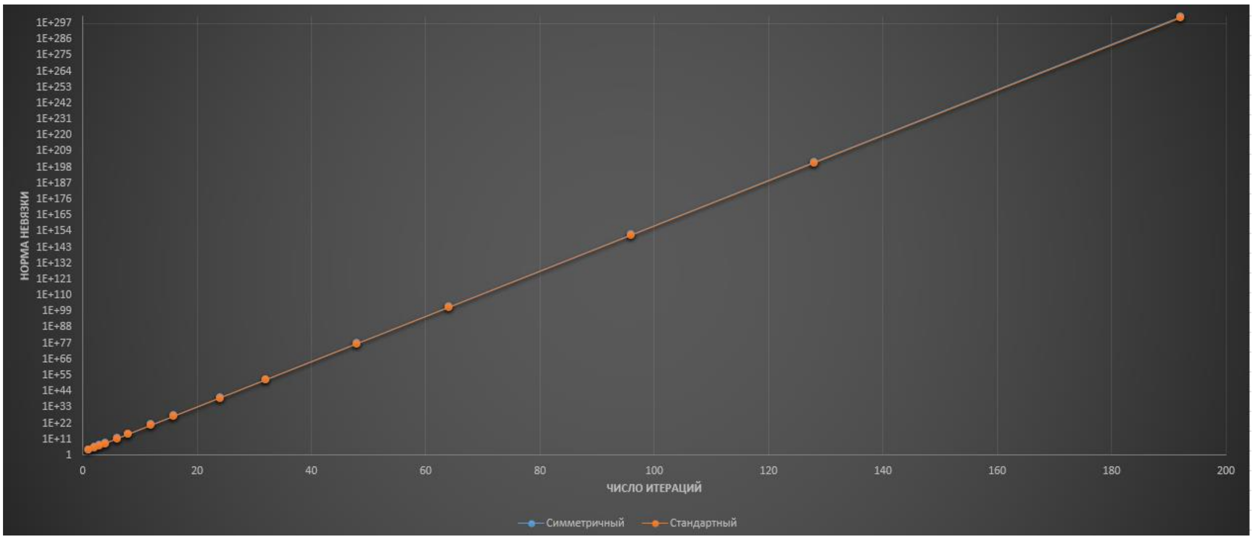
Также для реализованных методов создадим два теста, которые выполняются успешно.

Код:

```
[Test]
public void StandardTest()
{
    double[,] matrix =
    {
        {4,1,-1},
        {2,-2,1},
        {1,-1,2}
    };
    double[] freeMembers = {3, 1, 5};
    double[] expectedAnswer = {1, 2, 3};
    double[] startState = {2, 2, 2};
    double accuracy = 1e-8;
    double[] realAnswer = IterationMethods.StandardGaussSeidel(matrix,
                          freeMembers, startState, accuracy);
    double norm = 0;
    int len = realAnswer.Length;
    for (int i = 0; i < len; i++)
        norm += Math.Pow(realAnswer[i] - expectedAnswer[i], 2);
    Assert.IsTrue(norm<accuracy);
}

[Test]
public void SymmetricTest()
{
    double[,] matrix =
    {
        {4,1,-1},
        {2,-2,1},
        {1,-1,2}
    };
    double[] freeMembers = {3, 1, 5};
    double[] expectedAnswer = {1, 2, 3};
    double[] startState = {2, 2, 2};
    double accuracy = 1e-8;
    double[] realAnswer = IterationMethods.SymmetricGaussSeidel(matrix,
                                 freeMembers, startState, accuracy);
    double norm = 0;
    int len = realAnswer.Length;
    for (int i = 0; i < len; i++)
        norm += Math.Pow(realAnswer[i] - expectedAnswer[i], 2);
    Assert.IsTrue(norm<accuracy);
}
```

Для матрицы из условия построим диаграмму сходимости с логарифмической шкалой на оси ординат:

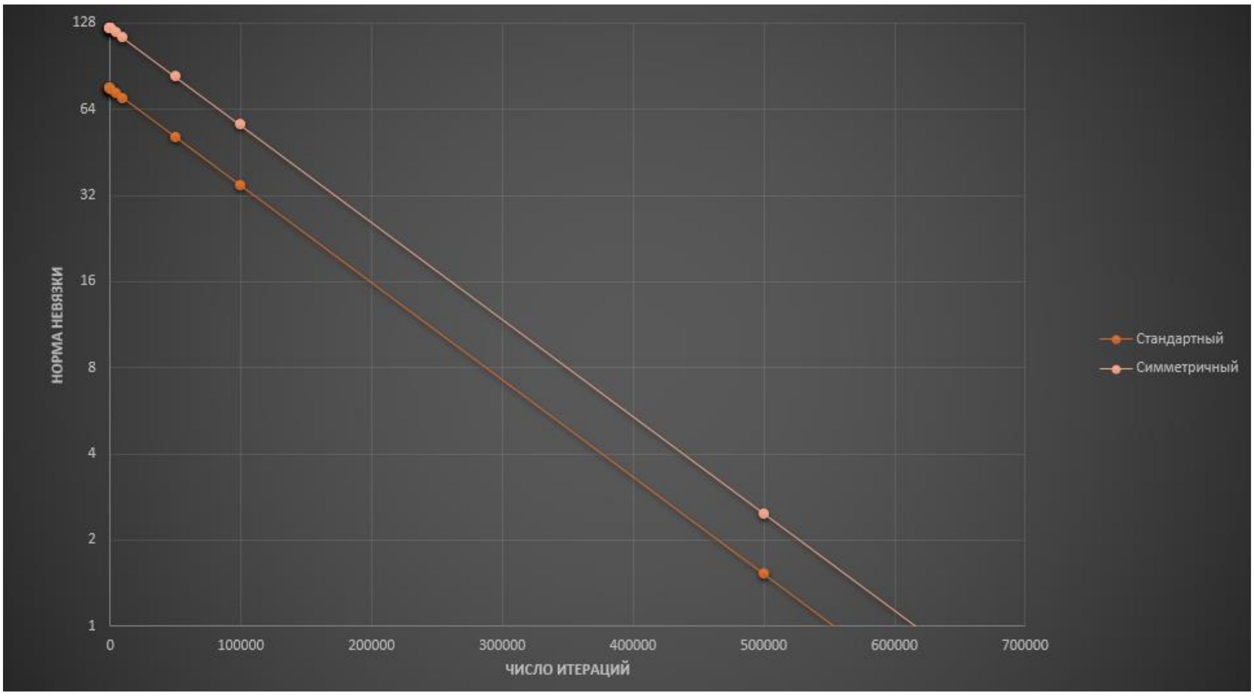


На диаграмме представлены графики и для стандартного, и для симметричного методов Гаусса-Зейделя, которые оказались близки друг к другу. Видно, что с числом итераций норма невязки растет, что подтверждает, что методы расходятся для этой матрицы.

Также подберем матрицу, для которой методы сходятся, но очень медленно:

$$\begin{pmatrix} 4 & -5 & 5 \\ 5 & -4 & 0 \\ 1 & -5.7599 & 4 \end{pmatrix}$$

Для данной матрицы логарифмическая диаграмма сходимости выглядит так:



Видно, что с ростом числа итераций норма невязки уменьшается и становится почти равной нулю.