**Министерство образования Республики Беларусь**

**Белорусский Государственный Университет**

**Физический факультет**

**Лабораторная работа №2**

**«Итерационные методы решения СЛАУ*»***

***Выполнил:***

студент 2 курса, 4 группы, физического факультета БГУ

Мельников Владислав Сергеевич

Минск, 2021

**Цель:**

Решить СЛАУ  методом Якоби и Зейделя. Реализовать методы и Якоби и Зейделя. Провести численный эксперимент, в результате которого получить зависимость числа итераций от величины погрешности. Сравнить работу двух методов. Реализовать метод верхней релаксации. Найти при помощи численного эксперимента оптимальное значение параметра ω. Зависит ли оно от погрешности?

**Содержимое проекта:**

***IterativeMethods.h –*** набор необходимых объявлений для класса *IterativeMethods*

***IterativeMethods.cpp –*** реализация основной логики программы

***Util.h –*** набор необходимых объявлений для класса *Util*

***Util.cpp –*** реализация вспомогательной логики программы

***Source.cpp –*** содержит точку входа в программу

**Вывод результата:**

*В консоль*

**Основные функции:**

Реализация метода Зейделя, где slae – исходная СЛАУ, epsilon – необходимая погрешность, n – кол-во уравнений, в count помещается кол-во итераций

static double\* zeidelMethod(double\*\* slae, double epsilon, int n, int& count);

Реализация метода Якоби, где slae – исходная СЛАУ, epsilon – необходимая погрешность, n – кол-во уравнений, в count помещается кол-во итераций

static double\* yakobiMethod(double\*\* slae, double epsilon, int n, int& count);

Нахождение зависимости кол-ва итераций от введенной точности для метода Зейделя, результат помещается в ассоциативный контейнер map

static std::map<double, int> dependenceZeidel(double\*\* slae, int n);

Нахождение зависимости кол-ва итераций от введенной точности для метода Якоби, результат помещается в ассоциативный контейнер map

static std::map<double, int> dependeceYakobi(double\*\* slae, int n);

**Вспомогательный функции:**

Считывание размерности матрицы коэффициентов из файла

int dimensionOfMatrixFromFile(std::string slaepath);

Считывание СЛАУ из файла

double\*\* slaeFromFile(std::string slaepath);

Извлечение из СЛАУ nxn+1 матрицы коэффициентов nxn

double\*\* aFromSlae(double\*\* slae, int n);

Извлечение из СЛАУ массива свободных членов n

double\* bFromSlae(double\*\* slae, int n);

Генерация квадратной матрицы случайных элементов

double\*\* randMatrixA(int n);

Генерация массива случайных элементов

double\* randMatrixB(int n);

Клонирование квадратной матрицы

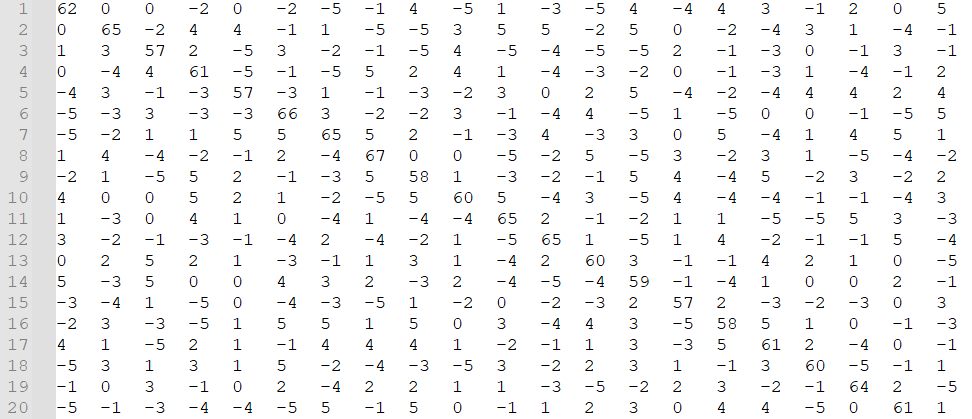
double\*\* matrixClone(double\*\* matrix, int n);

Клонирование массива

double\* arrayClone(double\* array, int n);

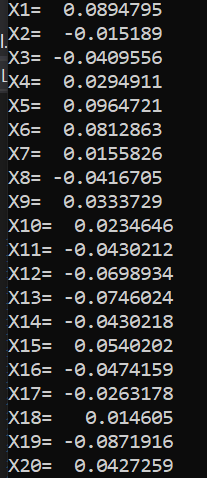
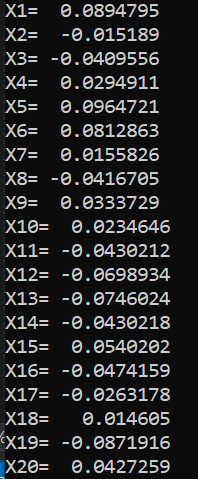
**Ход работы:**

**Исходная СЛАУ:**

****

**Результат работы методов с вводимой точностью 0.0000001:**

Зейделя Якоби

**** ****

**Кол-во итераций: 12 Кол-во итераций:** 15

Зависимость кол-ва итераций от точности для метода Якоби

Зависимость кол-ва итераций от точности для метода Зейделя

**Вывод:** метод простых итераций и метод Зейделя почти идентичны. Разница лишь в том, что в методе Зейделя расчет вектора приближений на текущей итерации происходит с использованием данных, полученных ни только на предыдущей, но и на нынешней итерации. То есть элемент x1 вычисляется на основе x2 и x3, значения которых, рассчитаны на предыдущей итерации, а следующий элемент x2 уже вычисляется за счет x1, полученного именно на текущей итерации, и x3 на предыдущей. Другими словами, данные в методе Зейделя для расчета вектора X поступают в процесс по мере их вычисления. А в методе простых итераций используются данные, строго полученные на предыдущей итерации. Рост кол-ва итераций от вводимой точности можно считать линейным.