Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и информатики

Лабораторная работа по дисциплине

Численные методы

«Решение систем линейных алгебраических уравнений

итерационными методами»

Выполнил

студент гр.5030102/20001 Дрекалов Н.С.

Преподаватель Добрецова С.Б.

Санкт-Петербург

2023

Оглавление

[**Формулировка задачи** 3](#_Toc151461771)

[**Формализация** 3](#_Toc151461772)

[**Предварительный анализ задачи** 4](#_Toc151461773)

[**Тестовый пример к методам** 4](#_Toc151461774)

[**Контрольные тесты** 5](#_Toc151461775)

[**Модульная структура программы** 5](#_Toc151461776)

[**Численный анализ методов** 7](#_Toc151461777)

[**Вывод** 9](#_Toc151461778)

## **Формулировка задачи**

Найти решение СЛАУ с помощью метода Зейделя с определённой точностью.

## **Формализация**

* Пусть F(x) : Rn −> Rn = Ax - b.
* Требуется: найти xk+1 приближение такое, что . xk+1 будет являться корнем этого уравнения

**Метод Зейделя с оптимальным параметром :**

*Условия применимости:*

* A – симметричная матрица с диагональным преобладанием.

*Алгоритм метода:*

* Считаем k+1 приближение x из k-го по формуле
* Делаем это до тех пор, пока
* Полученное k + 1-е приближение – корень уравнения Ax=b с точностью эпсилон.

## **Предварительный анализ задачи**

*Построение корня:*

* Корень X находился по формуле: xi = ln(i + 1) \* i / 2 \* sqrt(i) \* sin(i).

*Построение ортогональной матрицы:*

* Для матрицы использовался вектор w = [0.3 0.4 0.5 0.6 0.3 0.2 0.9 0.11 -0.1 -0.3]
* Ортогональная матрица Q строилась с помощью преобразование Хаусхолдера:

*Построение СЛАУ:*

* Строим диагональную матрицу с числами минимум на порядок больше чисел ортогональной матрицы Q.
* Строим симметричную матрицу по формуле A = QE’QT.
* Находим свободный член по формуле b = Ax.

*Проверка условий для методов:*

*Метод Зейделя с оптимальным параметром:*

* А – симметричная с диагональным преобладанием по построению

**Тестовый пример к методам**

,

Собственные числа матрицы:

Коэффициент

Формула итерации:

Начальное приближение:

*0.252489*

*1.7618*

*0.66912*

*1.20874*

*1.19791*

*1.05407*

*1.04686*

*1.08249*

*0.999092*

**Контрольные тесты**

*Построим графики зависимостей*

* Нормы фактической ошибки и нормы невязки от точности.
* Числа итераций от определителя.

**Модульная структура программы**

// Класс одного линейного уравнения

typedef std::vector<std::vector<long double>> Matrix;

typedef std::vector<long double> LinMatrix;

// Utility функции для произведения операций над матрицами

LinMatrix operator\*(Matrix A, LinMatrix x);

LinMatrix operator+(LinMatrix A, LinMatrix B);

LinMatrix operator-(LinMatrix A, LinMatrix B);

LinMatrix operator/(LinMatrix A, long double a);

long double norm(LinMatrix A);

class LinEquation {

// A \* x = b

Matrix A;

LinMatrix b;

LinMatrix x;

size\_t dim;

// Коэффициент альфа в итерациях

long double a;

// Точность

long double epsilon;

/\*\*

\* \brief Функция, считающая новое приближение на основе старого

\* \param x - старое приближение

\* \return новое приближение

\*/

LinMatrix doOneIteration(LinMatrix& x);

/\*\*

\* \brief функция, проверяющая, больше ли норма разности между приближениями, чем эпсилон

\* \param x1 - новое приближение

\* \param x0 - старое приближение

\* \return больше ли норма всё еще, чем эпсилон

\*/

bool isConditionMet(LinMatrix& x1, LinMatrix& x0);

public:

/\*\*

\* \brief Конструктор по умолчанию

\*/

LinEquation() = default;

/\*\*

\* \brief Конструктор класса

\* \param A - матрица системы

\* \param b - свободный член

\* \param l1 - первое собственное число

\* \param ln - последнее собственное число

\*/

LinEquation(Matrix& A, LinMatrix& b, long double l1, long double ln);

/\*\*

\* \brief Функция, возвращающая матрицу A

\* \return матрица A

\*/

const Matrix& getA() const;

/\*\*

\* \brief Функция, возвращающая матрицу b

\* \return матрица b

\*/

const LinMatrix& getb() const;

/\*\*

\* \brief Функция, возвращающая корень

\* \return корень уравнения

\*/

const LinMatrix& getx() const;

/\*\*

\* \brief Фукнция, считающая корень СЛАУ с помощью метода Зейделя

\* \param epsilon - точность нахождения

\*/

void solve(long double epsilon);

};

// Класс решения

class Solution {

// Массив всех СЛАУ

std::vector<LinEquation> linEquations;

// Файл, в из которого сичтываются СЛАУ

const std::string inFilename;

// Файл, в который записываются СЛАУ

const std::string outFilename;

bool initialized = false;

int eMin = 0, eMax = 0;

int equationsCount = 0;

/\*\*

\* \brief Функция, сичтывающая матрицы из заданного файла

\*/

void readEquationsFromFile();

/\*\*

\* \brief Функция, записывающая матрицы в заданный файл

\*/

void writeMatrices();

/\*\*

\* \brief Функция обработки ошибок

\* \param error - ошибка

\*/

void parseError(const std::string& error);

public:

/\*\*

\* \brief Конструктор класса

\* \param inFilename - файл, из которого считываются матрицы

\* \param outFilename - файл, в который записываются матрицы

\*/

explicit Solution(const std::string& inFilename, const std::string& outFilename);

/\*\*

\* \brief Функция, считающая корни у всех СЛАУ

\*/

void begin();

/\*\*

\* \brief Функция, вызываемая при завершении вычислений

\*/

void end();

/\*\*

\* \brief Деструктор класса

\*/

~Solution();

};

/\*\*

\* \brief Точка входа

\*/

int main()

## **Численный анализ методов**

*Рассмотрим зависимость нормы фактической ошибки и нормы невязки от числа итераций.*

**

На графике видно, что ошибка и норма невязки уменьшаются в соответствии с точностью. Причем зависимость логарифмов растет линейно.

*Рассмотрим зависимость числа итераций от определителя.*



Из графика видно, что N = f(log(|A|), причем число итераций увеличивается с увеличением определителя.

## **Вывод**

В лабораторной работе мне удалось найти корни СЛАУ с заданной точности с помощью итерационного стационарного метода Зейделя.

В ходе исследования была проанализирована зависимость нормы невязки и фактической ошибки от точности, отклонений замечено не было. Также была исследована зависимость числа итераций от определителя матрицы системы, она оказалась вида N = C \* log(k|A|) + b, причем C > 0.