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

прямыми методами»

Выполнил

студент гр.5030102/20001 Дрекалов Н.С.

Преподаватель Добрецова С.Б.

Санкт-Петербург

2023

Оглавление

[**Формулировка задачи** 3](#_Toc147173981)

[**Формализация** 3](#_Toc147173982)

[**Теорема о верхней границе** 3](#_Toc147173983)

[**Метод половинного деления** 3](#_Toc147173984)

[**Метод простых итераций** 4](#_Toc147173985)

[**Предварительный анализ задачи** 5](#_Toc147173986)

[**Тестовый пример к методам** 7](#_Toc147173987)

[**Контрольные тесты** 8](#_Toc147173988)

[**Модульная структура программы** 8](#_Toc147173989)

[**Численный анализ методов** 10](#_Toc147173990)

[**Вывод** 12](#_Toc147173991)

## **Формулировка задачи**

Найти решение СЛАУ с помощью LDLT разложения.

## **Формализация**

* Пусть F(x) : Rn −> Rn = Ax - b.
* Требуется: найти x∗ такое, что F (x∗) = 0. Корень x∗ ищется для заданного числа обусловленности

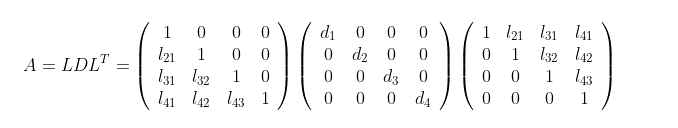
**Построение LDLT разложения:**

*Условия применимости:*

* A – симметричная матрица.

*Алгоритм метода:*

* Необходимо представить матрицу A = LDLT, где L – нижняя треугольная матрица с единицами на главной диагонали, а D – диагональная матрица. Просто решает уравнения для каждого столбца, идя слева направо, а по каждому столбцу – сверху вниз.



Идем по столбцам слева направо и сверху вниз.

1. Для 0 < i <= dimA:
   1. Для 0 < j <= dimA:
      1. Считаем сумму:
         1. sum = aij.,
         2. для 1 <= k < j:
            1. sum -= ljk \* dk \* lik.
      2. Если i = j:
         1. di  = sum,
         2. lij = 1.
      3. Иначе:
         1. lij = sum / dj.
2. Полученные L, D такие, что A = LDLT.

**Решение СЛАУ с помощью LDLT разложения:**

*Условия применимости:*

* A – симметричная матрица.

*Алгоритм метода:*

1. Представим матрицу A = LDLT.

Решим уравнение L \* c = b, где c = DLTx

1. Для 0 < i <= dimA:
   1. ci  = bi
   2. Для 0 < j < i:
      1. ci -= lij \* cj.

Построим c’ = D-1 \* c:

1. Для 0 < i <= dimA:
   1. ci /= di.

Идем по строкам снизу вверх справа налево.

1. Для dimA >= i > 0:
   1. xi = ci,
   2. Для dimA >= j > i:
      1. xi -= lji \* xj.
2. Полученный X – решение СЛАУ.

## **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 = [3 4 5 6 3 2 9 11 -1 -3]
* Ортогональная матрица Q строилась с помощью преобразование Хаусхолдера:

*Построение СЛАУ:*

* Берем единичную матрицу и делаем преобразование e11 = 10^i для 1 <= i < 15. Это будет примерно равно числу обусловленности матрицы.
* Строим симметричную матрицу по формуле A = QE’QT.
* Находим свободный член по формуле b = Ax.

*Проверка условий для методов:*

*LDLT разложение:*

* А – симметричная по построению

*Нахождения корня с помощью LDLT разложения.*

* А – симметричная по построению

**Тестовый пример к методам**

**Контрольные тесты**

*Для каждого из уравнений построим графики зависимостей*

* Зависимость числа итераций от заданной точности ε = [10−2, 10−15]
* Зависимости фактической ошибки (разности точного и найденного значений корня) от заданной точности ε = [10−2, 10−15]. На график нанести линию заданной точности.

**Модульная структура программы**

/// Класс, который имеет два поля - для хранения числа итераций и для хранения корня

class Result {

public:

explicit Result(unsigned iterNumber, double root) : iterNumber(iterNumber), root(root) {};

const unsigned iterNumber = 0;

const double root;

};

#define MAX\_ITER\_NUM ULLONG\_MAX

/\*\*

\* **\brief** Функция находит корень в заданном промежутке методом ПД с точностью эпсилон

\* **@param** f функция, с у которой ищется корень

\* **@param** a левая граница

\* **@param** b правая граница

\* **@param** epsilon точность

\* **@return** тип Result в котором содержатся два поля - корень и число итераций

\*/

Result findRoot(double(&f)(double), double a, double b, double epsilon);

/\*\*

\* **\brief** Функция находит корень в заданном промежутке методом простых итераций с точностью эпсилон для полинома

\* **@param** f полином

\* **@param** epsilon точность

\* **@return** тип Result в котором содержатся два поля - корень и число итераций

\*/

Result findRootForPolynom(double(&f)(double), double epsilon);

/\*\*

\* **\brief** Функция находит корень в заданном промежутке методом простых итераций с точностью эпсилон для трансцендентой функции

\* **@param** f трансцендентная функция

\* **@param** epsilon точность

\* **@return** тип Result в котором содержатся два поля - корень и число итераций

\*/

Result findRootForComplex(double(&f)(double), double epsilon);

/\*\*

\* **\brief** функция, считающая значения полинома в точке x

\* **\param** x переменная

\* **\return** значение полинома в точке x

\*/

double polyVar11(const double x);

/\*\*

\* **\brief** функция, считающая значения транс фукнции в точке x

\* **\param** x переменная

\* **\return** значение функции в точке x

\*/

double complexVar11(const double x);

/\*\*

\* **\brief** функция, считающая корень для полинома для различной точности и выводящая всё на экран

\*/

void testFirstFunc();

/\*\*

\* **\brief** функция, считающая корень трансцедентной фукнции для различной точности и выводящая всё на экран

\*/

void testSecondFunc();

## **Численный анализ методов**

*Рассмотрим зависимости количества итераций от точности приближения.*



На графике видно, что быстрее всего работает МПИ для трансцендентного уравнения (это связано с удачно выбранным интервалом корня), остальные три выполняются примерно одинаково по скорости, с логарифмической сложностью.

*Рассмотрим зависимость ln разности с истинным корнем от точности*



Из графика видно, что во всех случаях достигается нужная точность (значение погрешности не превышает значение заданной точности). То есть результаты получены верно.

## **Вывод**

В лабораторной работе я нашел корни уравнений методом половинного деления и методом простых итераций.

В ходе исследования сравнены эффективность методов. Метод половинного деления стабильно сходится к корню, метод простых итераций же сходится к корню с адекватной скоростью только при ряде условий.

Можно сделать вывод, что для наиболее эффективного решения уравнения, нужно его исследовать (узнать поведение функции и её производных на промежутке) и на основе полученной предварительной точности выбрать один из методов.