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](#_Toc148990649)

[**Формализация** 3](#_Toc148990650)

[**Предварительный анализ задачи** 5](#_Toc148990651)

[**Тестовый пример к методам** 6](#_Toc148990652)

[**Контрольные тесты** 8](#_Toc148990653)

[**Модульная структура программы** 8](#_Toc148990654)

[**Численный анализ методов** 10](#_Toc148990655)

[**Проверка на матрице с нулевым определителем** 11](#_Toc148990656)

[**Вывод** 12](#_Toc148990657)

## **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

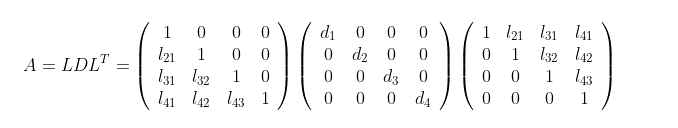
**Построение LDLT разложения:**

*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 решаем 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:*

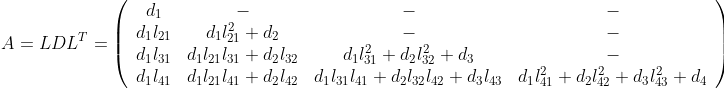
* А – симметричная по построению

*Нахождения корня с помощью LDLT разложения.*

* А – симметричная по построению

**Тестовый пример к методам**

Решим тестовый пример для уравнения Ax = b, где

1. Построим LDLT разложение:

Идем по столбцам:

Первый:

d1 = a11 = -1

l21 = a21 / d1 = 4

l31 = a31 / d1 = -5

l41 = a41 / d1 = -4

Второй:

d2 = a22 – d1l221 = 3 + 1 \* 16 = 19

l32 = (a32 – d1l21l31)/d2 = (-9 – 20) / 19 = -29/19

l42 = (a42 – d1l21l41)/d2 = (5 - 16) / 19 = -11/19

Третий:

d3 = a33 – d1l231 – d2l232 = 7 + 25 – 29 \* 29 / 19 = -233/19

l43 = (a43 – d1l31l41 – d2l32l42) / d3 = (-2 \* 19 + 20 \* 19 - 29 \* 11) / -233 = 23/-233

Четвёртый

d4 = a44 – d1l241 – d2l242 – d3l243 = 0 + 16 – 121/19 + 23 \* 23 / (233 \* 19) = = 43168/4427 = 2272/233

1. Найдем c из уравнения Lc = b, где c = DLTx:

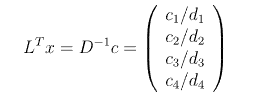
c1 = b1 = 22

c2 = b2 – l21c1 = -5 – 4 \* 22 = -93

c3 = b3 – l31c1 – l32c2 = 0 + 5 \* 22 – 29/19 \* 93 = -607/19

c4 = b4 – l41c1 – l42c2 – l43c3 = 8 + 4 \* 22 – 11/19 \* 93 – 23/233 \* 607/19 = = 172672 / 4427 = 9088 / 233

1. Умножим обе части слева на D-1:



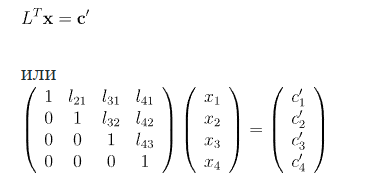
c1’ = c1 / d1 = -22

c2’ = c2 / d2 = -93/19

c3’ = c3 / d3 = 607/233

c4’ = c4 / d4 = 9088/2272 = 4

1. Решаем уравнение



x4 = c4’ = 4

x3 = c3’ – l43x4 = 607/233 + 4 \* 23/233 = 699/233 = 3

x2 = c2’ – l32x3 – l42x4 = -93/19 + 29 \* 3 / 19 + 11 \* 4 / 19 = 38/19 = 2

x1 = c1’ – l21x2 – l31x3 – l41x4 = -22 – 8 + 15 + 16 = 1

Получаем

Проверим ответ:

**Контрольные тесты**

*Построим графики зависимостей*

* Нормы фактической ошибки от числа обусловленности.
* Нормы невязки от числа обусловленности.

**Модульная структура программы**

// Класс одного линейного уравнения

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

typedef std::vector<long double> LinMatrix;

class LinEquation {

// A \* x = b

Matrix A;

LinMatrix b;

long double conditionalityNumber;

LinMatrix x;

// A = LDLt

Matrix L;

LinMatrix D;

/\*\*

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

\*/

void makeLDLt();

public:

/\*\*

\* \brief Конструктор по умолчанию

\*/

LinEquation() = default;

/\*\*

\* \brief Конструктор класса

\* \param A - матрица системы

\* \param b - свободный член

\* \param conditionalityNumber - число обуслвленности

\*/

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

/\*\*

\* \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 Функция, возвращающая число обуцсловленности

\* \return число обусловленности

\*/

const long double getConditionalityNumber() const;

/\*\*

\* \brief Фукнция, считающая корень СЛАУ с помощью LDLt разложения

\*/

void solve();

};

// 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

\*/

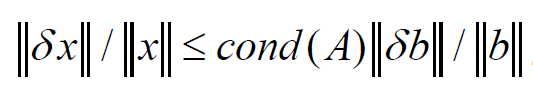
int main()

## **Численный анализ методов**

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



На графике видно, что ошибка и норма невязки растут с увеличением числа обусловленности. Причем зависимость логарифмов растет линейно.

**

*Также было проверено выполнение неравенства*

Приведу ниже код проверки.

Для iй СЛАУ:

db = 2 \* (0.5 - rand(MATRIX\_SIZE,1)) .\* b;

condNum = norm(A) \* norm(inv(A));

if ~(norm1(i) / norm(roots(:, i)) <= condNum \* norm(db) \* norm(b))

inequalityTrue = false;

end

...

if inequalityTrue

fprintf("Неравество выполняется!\n");

else

fprintf("Неравенство не выполняется!\n");

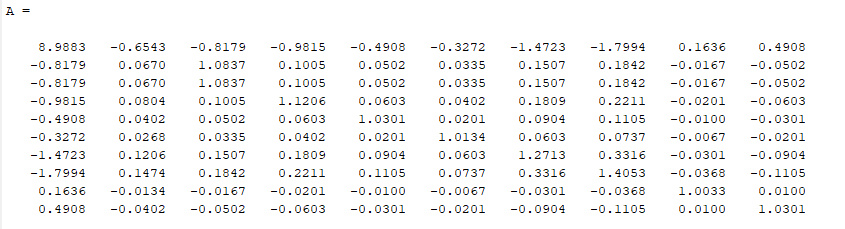
end



## **Проверка на матрице с нулевым определителем**

*Построение матрицы:*

В первой построенной ранее симметричной матрице поменяли строку и столбец на другие (в итоге матрица нетривиальная и симметричная, но с нулевым определителем)



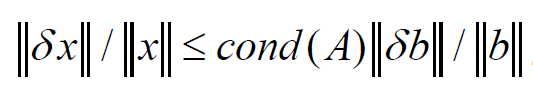
К ней и к матрице B = 10A считался свободный член, и затем матрицы «отправлялись» в программу, считающую корень.

Корни удалось получить написанным алгоритмом. Нормы фактической ошибки равны 1.3846 и 34.7297 соответственно, то есть ошибка примерно так-же, как и значения матрицы (на порядок), в то время как норма невязки равны 0.0458 и 71.4629, то есть норма увеличилась на два порядка.

То есть получилось, что в целом алгоритм может посчитать корни, но у нас нет способа проанализировать их точность (так как не определено число обусловленности из-за не определенной обратной матрицы).

## **Вывод**

В лабораторной работе мне удалось найти корни СЛАУ с помощью LDLT разложения

**В ходе исследования была проанализирована зависимость нормы невязки и фактической ошибки от числа обусловленности. Выяснилось, что ошибка растёт вместе с числом обусловленности, т. е. число обусловленности можно считать величиной, обратной к точности.

Также было проверено на истинность неравенство . Оно оказалось истинным для всех составленных СЛАУ