МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

Направление подготовки: «Прикладная математика и информатика»

Профиль подготовки: «Вычислительная математика и суперкомпьютерные технологии»

Отчет по лабораторной работе №1

**«Блочное LU разложение»**

**Выполнил:** студент группы 381903-3м

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Панов А.А.

Подпись

Нижний Новгород  
2020

Содержание

[Введение 2](#_Toc40036094)

[1. Постановка задачи 3](#_Toc40036095)

[2. Алгоритм LU разложения и его свойства 4](#_Toc40036096)

[2.1 Связь метода Гаусса и LU разложения 4](#_Toc40036097)

[2.2 Алгоритмическая сложность 4](#_Toc40036098)

[3. Наивная реализация 5](#_Toc40036099)

[4. Блочная реализация 5](#_Toc40036100)

[5. Численный эксперимент 5](#_Toc40036101)

# 

# Введение

LU разложение − представление матрицы A в виде A=LU. L − нижняя треугольная матрица с диагональными элементами, равными единице, а U − верхняя треугольная матрица с ненулевыми диагональными элементами.

LU разложение используется для решения СЛАУ вида Ax = b. LU разложение особенно полезно если имеется «несколько правых частей». Алгоритм решения в таком случае следующий:

1. Находится LU разложение матрицы A:
2. Для каждой «правой части» bi из B:
   1. // вид системы после подстановки
   2. // L – нижняя треугольная матрица, ищется обратным ходом Гаусса
   3. // U – верхняя треугольная матрица, ищется обратным ходом Гаусса

LU разложение достаточно вычислительно сложная задача (алгоритмическая сложность пропорциональна n3) и для её эффективного решения полезно использовать параллельные вычисления.

# Постановка задачи

Реализовать блочное LU-разложение для квадратной матрицы, используя технологию OpenMP, то есть представить матрицу A в виде произведения двух матриц:  
A=LU, где L — нижняя треугольная матрица, а U — верхняя треугольная матрица.

Программа на языке C++ должна реализовывать функцию со следующим заголовком:

void LU\_Decomposition(double \* A, double \* L, double \* U, int n);

Функция получает в аргументах следующие переменные:  
A – указатель на массив, в котором по строкам хранится матрица A размера n×n  
n – размерность матрицы

L – указатель на массив, в котором по строкам необходимо записать матрицу L размера n×n,

U – указатель на массив, в котором по строкам необходимо записать матрицу U размера n×n.  
Ответ считается корректным, если:

Размерность матрицы n ≤ 3000.

# Алгоритм LU разложения и его свойства

## Связь метода Гаусса и LU разложения

Рассмотрим СЛАУ вида Ax = b, с невырожденной матрицей A. С помощью метода Гаусса систему можно представить в виде: Ux = y = L-1b

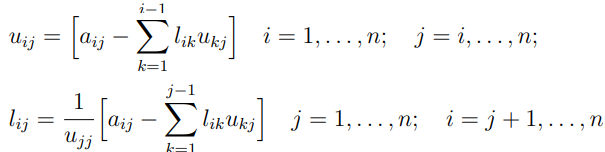
Ux = L-1b, умножим обе части на L

LUx = b

Таким образом коэффициенты U можно получить в результате применения к матрице A метода Гаусса. Матрица L будет состоять из коэффициентов метода Гаусса. Существует теорема (доказательство можно посмотреть в «Численных методах» В. Б. Андреева), согласно которой LU-разложение существует и единственно, если главные миноры матрицы A отличны от нуля. Также для любой невырожденной матрицы A существует матрица перестановок P такая, что матрица PA имеет ненулевые главные миноры. Следовательно систему Ax = b можно представить в виде PAx = LUx = Pb.

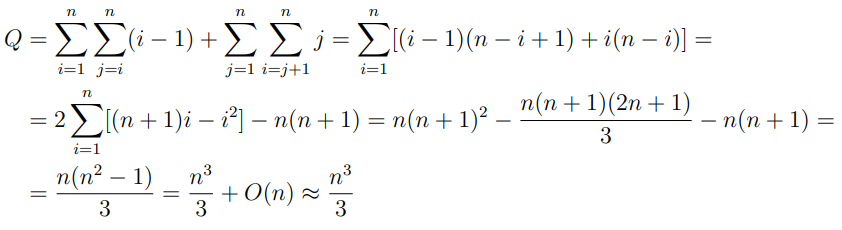
## Алгоритмическая сложность

Вычисление коэффициентов L и U выполняется по следующим формулам:



Формула 1. Коэффициенты матриц LU

Количество умножений и делений можно подсчитать по следующей формуле:



Для получения LU разложения потребуется ) умножений и делений.

# Наивная реализация

Реализация LU разложения по формуле 1 достаточно простая. При этом код легко параллелится.

#define indx(i, j, n) ((i)\*(n)+(j))

for (int ved = 0; ved < n; ved++)

{

double el = U[indx(ved, ved, n)];

#pragma omp parallel for

for (int i = ved + 1; i < n; i++)

{

double coeff = U[indx(i, ved, n)] / el;

L[indx(i, ved, n)] = coeff;

#pragma ivdep

for (int j = ved + 1; j < n; j++)

U[indx(i, j, n)] -= coeff \* U[indx(ved, j, n)];

}

}

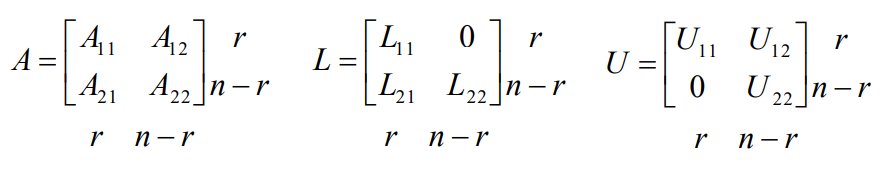
Недостаток данного метода – плохое использование кэш памяти процессора.

## Погрешность решения

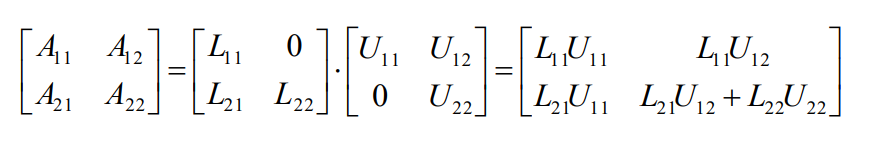
Пусть на компьютере выполнено LU разложение с ошибкой вычислений :

, где эквивалентное возмущение. Тогда выполняется неравенство:

# Блочная реализация

Чтобы улучшить работу с кэшем процессора представим исходную матрицу A и искомые матрицы L и U в следующем виде (на рисунке ниже r это размер блока): 

Подставив в формулу A = LU получим:



Алгоритм решения в таком случай стовится таким:

1. Находим L11, U11 используя стандартное LU разложение
2. Находим U12 из СЛАУ c несколькими правыми частями A12 = L11U12
3. Находим L21 из СЛАУ c несколькими правыми частями A21 = L21U11
4. Находим матрицу Ᾱ22 = A22 – L21U12, Ᾱ22 = L22U22, далее алгоритм применяется рекурсивно.

Так как матрицы L11 и U11 треугольные, то для поиска U12 и L21 можно применить обратный ход Гаусса. Алгоритмическая сложность первого этапа алгоритма , второго и третьего на k шаге, третьего ). Всего шагов , так что можно оценить сложность алгоритма как Есть более точная оценка . Можно заметить, что если , то умножение матриц будет наиболее «трудоемкой» операцией. Общую долю матричных умножений можно оценить величиной . Так как матричное умножение можно эффективно распараллелить, то большая доля матричных умножений позволит распараллелить LU разложение очень эффективно.

# Вычислительные эксперименты

# Заключение