Санкт-Петербургский государственный университет

Эмпирический анализ алгоритма построения разложения Холецкого.

По дисциплине: «Алгоритмы и анализ сложности»
Направление: «Фундаментальная информатика и информационные технологии»

Выполнил студент 3 курса группы 20.Б11-ПУ
Киселев Владимир Александрович

Санкт-Петербург
2022 г.

Содержание

1. Описание алгоритма построения разложения Холецкого для положительно определенных симметричных матриц.

1.1. История

Разложение Холецкого впервые предложено французским офицером и математиком Андре-Луи Холецким в конце Первой Мировой войны, незадолго до его гибели в бою в августе 1918 г. Идея этого разложения была опубликована в 1924 г. его сослуживцем. Потом оно было использовано Т. Банашевичем в 1938 г. В советской математической литературе называется также методом квадратного корня.

1.2. Область применения

Первоначально разложение Холецкого использовалось исключительно для плотных симметричных положительно определенных матриц. В настоящее время его использование гораздо шире. Оно может быть применено также к эрмитовым матрицам. Для повышения производительности вычислений часто применяется блочная версия разложения.

Для разреженных матриц разложение Холецкого также широко применяется в качестве основного этапа прямого метода решения линейных систем. В этом случае используют специальные упорядочивания для уменьшения ширины профиля исключения, а следовательно и уменьшения количества арифметических операций. Другие упорядочивания используются для выделения независимых блоков вычислений при работе на системах с параллельной организацией.

Варианты разложения Холецкого нашли успешные применения в итерационных методах для построения переобусловливателей разреженных симметричных положительно определенных матриц.

Разложение Холецкого также используются в методах Монте-Карло для генерации коррелированных случайных величин.

1.3. Математическое описание алгоритма

Разложение Холецкого эрмитовой положительно определенной матрицы A является разложением вида: $A = LL^T$, где L — нижняя треугольная матрица с вещественными и положительными диагональными элементами.

Каждая эрмитова положительно определенная матрица имеет уникальное разложение Холецкого.

Выходные данные: действительно положительно определённая симметрическая матрица A (элементы a_{ij}).

Объём входных данных: $\frac{n(n+1)}{2}$ (в силу симметричности достаточно хранить только диагональ и над/поддиагональные элементы).

Выходные данные: нижняя треугольная матрица L (элементы l_{ij}).

Объём выходных данных: $\frac{n(n+1)}{2}$ (в силу треугольности достаточно хранить только ненулевые элементы).

Элементы матрицы L можно вычислить, начиная с верхнего левого угла матрицы A, по формулам:

$$l_{11} = \sqrt{a_{11}}, \qquad l_{i1} = \frac{a_{i1}}{l_{11}}, \ i \in [2, n],$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \ i \in [2, n],$$

$$l_{ij} = \left(a_{ij} - \sum_{p=1}^{k-1} l_{ip} l_{jp}\right) / l_{ii}, \ j \in [2, n-1], \ i \in [j+1, n].$$

Выражение под квадратным корнем всегда положительно, если A — вещественная симметричная положительно определённая матрица.

Вычисление происходит сверху вниз, слева направо, т.е. сначала вычисляется $L_{ij} (j < i)$, а уже затем L_{ii}. Вычисления обычно проводятся в одной из следующих последовательностей.

- Алгоритм Холецкого-Банашевича или просто алгоритм Холецкого, когда вычисления начинаются с верхнего левого угла матрицы L и проводятся по строкам. Этот вариант разложения используется наиболее часто, особенно при использовании построчного формата хранения элементов матрицы L.
- Краут-вариант алгоритма Холецкого, когда вычисления также начинаются с верхнего левого угла матрицы L, но проводятся по столбцам. Этот вариант разложения используется несколько реже, применяется он при использовании столбцевого формата хранения элементов матрицы L, а также когда необходимо проводить коррекцию ведущих элементов при выполнении приближенного разложения.

Существует также блочная версия метода, однако в данном описании разобран точечный метод.

2. Математический анализ алгоритма

2.1. Последовательная сложность алгоритма

Для разложения Холецкого матрицы порядка n в последовательном (наиболее быстром) варианте требуется:

- n вычислений квадратного корня,

- $\frac{n(n-1)}{2}$ делений,
- $\frac{n^3-n}{6}$ умножений,
- $\frac{n^3-n}{6}$ сложений (вычитаний).

Умножения и сложения (вычитания) составляют основную часть алгоритма.

При этом использование режима накопления требует совершения умножений и вычитаний в режиме двойной точности, что ещё больше увеличивает долю умножений и сложений/вычитаний во времени, требуемом для выполнения разложения Холецкого. Таким образом, при классификации по последовательной сложности разложение Холецкого относится к алгоритмам с кубической сложностью.

3. Входные данные и единицы измерения трудоемкости

3.1. Характеристики

Входными данными для алгоритма является матрица A размера $n \times n$, $n \in [64, 2048]$

Так как количество базовых операций заранее известно для любой матрицы размера $n \times n$, я выбрал единицами измерения трудоемкости — время выполнения программы.

Перед запуском алгоритма Холецкого, а также после его выполнения замеряется время на компьютере. Разницей этих двух замеров и будет время работы разложения.

Количество тестов — 10. в каждом из которых 21 раз запускается алгоритм Холецкого для разного n.

3.2. Генерация входных данных

Теорема:

Для того, чтобы матрица A ($dim(A) = n \times n$) была симметричной и положительно определенной, необходимо и достаточно, чтобы она была представима в следующем виде: $A = LL^T$, где L — нижнетреугольная матрица.

Основываясь на этой теореме, я решил сгенерировать матрицу L со случайными значениями, с помощью чего будет высчитываться матрица $A = LL^T$.

```
108     void matrixGenerator(double lowerLimit = -10, double upperLimit = 10) {
109         /* ... */
116         srand(time(0));
117
118         for (int i = 0; i < n; ++i) {
119             for (int j = 0; j <= i; ++j) {
120                 double v = lowerLimit + ((double)rand() / (double)(RAND_MAX)) * (upperLimit - lowerLimit);
121                 exactMatrix[i][j] = v;
122             }
123         }
124         double** transposedExactMatrix = matrixTranspose(exactMatrix);
125         inputArr = matrixMultiplication(exactMatrix, transposedExactMatrix);
126
127         deleteArr(transposedExactMatrix, n);
128     }
```

(Рисунок 1. Функция, генерирующая симметрическую положительно определенную матрицу inputArr)

4. Реализация алгоритма

Для программной реализации был выбран язык C++

```
164     void optimized_decompose() {
165         auto begin = std::chrono::steady_clock::now();
166         for (int i = 0; i < n; ++i) {
167             for (int j = 0; j <= i; ++j) {
168                 double res = inputArr[i][j];
169                 for (int k = 0; k < j; ++k) {
170                     res -= outputArr[i][k] * outputArr[j][k];
171                 }
172 
173                 if (i == j) {
174                     outputArr[i][j] = sqrt(res);
175                 } else {
176                     outputArr[i][j] = res / outputArr[j][j];
177                 }
178             }
179         }
180         auto end = std::chrono::steady_clock::now();
181         auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
182         operatingTime = elapsed_ms.count();
183     }
```

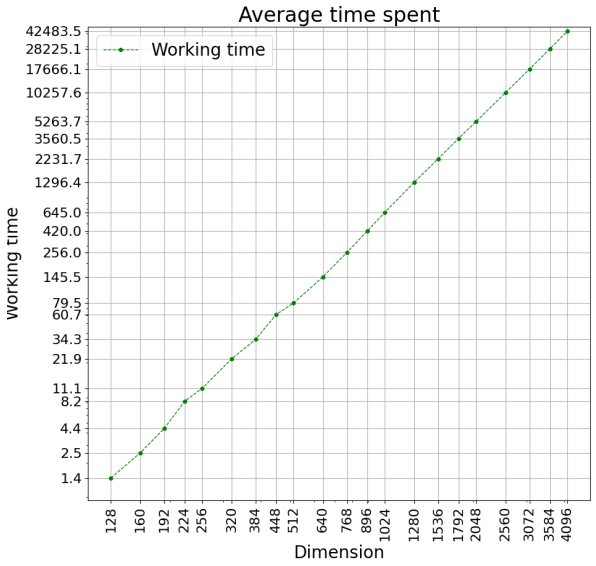
(Рисунок 2. Функция разложения матрицы inputArr алгоритмом Холецкого)

С полной реализацией программы можно ознакомиться по следующей ссылке: https://github.com/CroccoRush/the-Kholetsky-algorithm

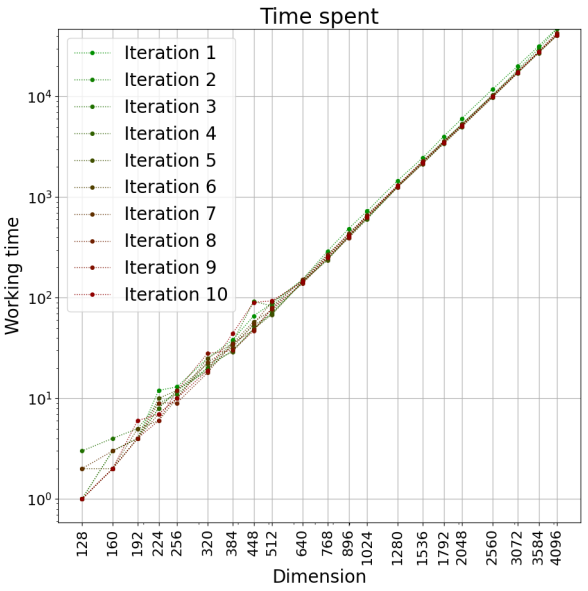
5. Вычислительный эксперимент

В соответствии с входными данными был проведен вычислительный эксперимент. На графике представлен результат: зависимость времени работы алгоритма от размерности входной матрицы. Наглядно заметно, что в среднем отклонение невелико, а некоторые выбросы могут быть обусловлены вмешательством в вычисления сторонних процессов.

График зависимости времени работы алгоритма от размерности входной матрицы (см. Рисунок 3 и Рисунок 5). Следовательно, алгоритм действительно принадлежит классу $O(n^3)$.



(Рисунок 3. Среднее время работы алгоритма Холецкого в логарифмической шкале)



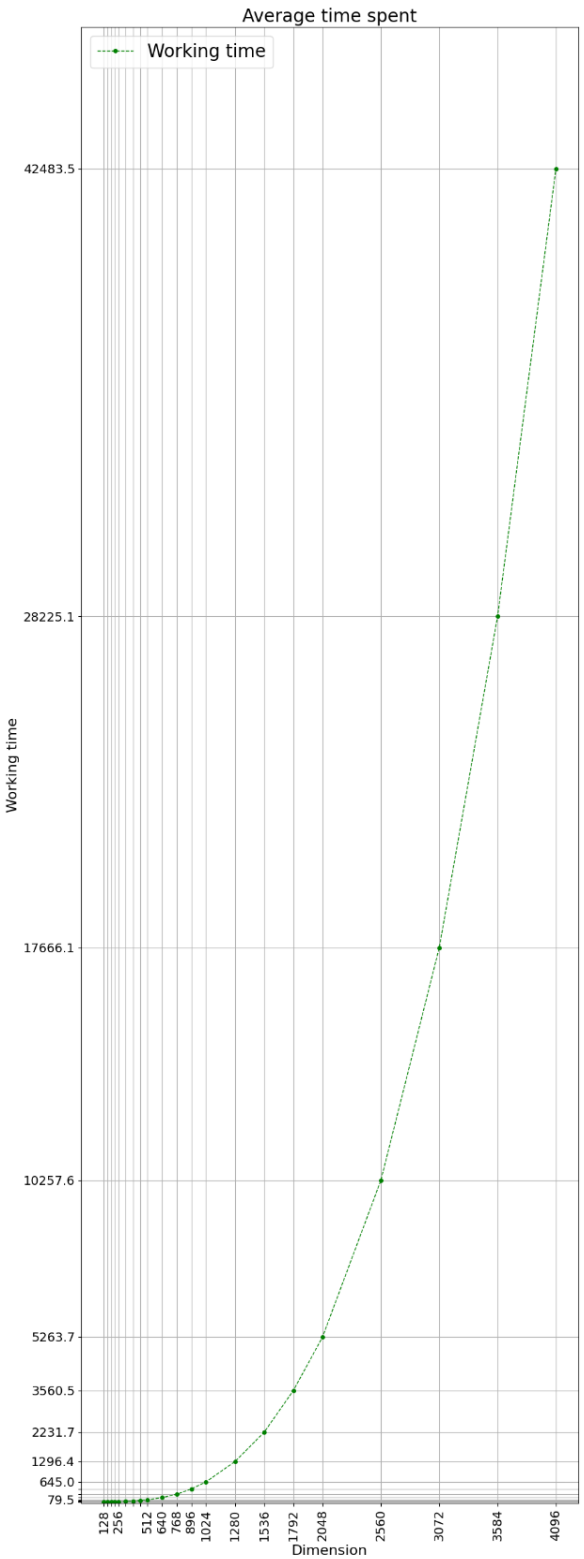
(Рисунок 4. Время работы каждой итерации алгоритма Холецкого в логарифмической шкале)

Для рассмотрения отношения значений измеренной трудоемкости при удвоении размера входных данных можем воспользоваться уже имеющимися результатами.

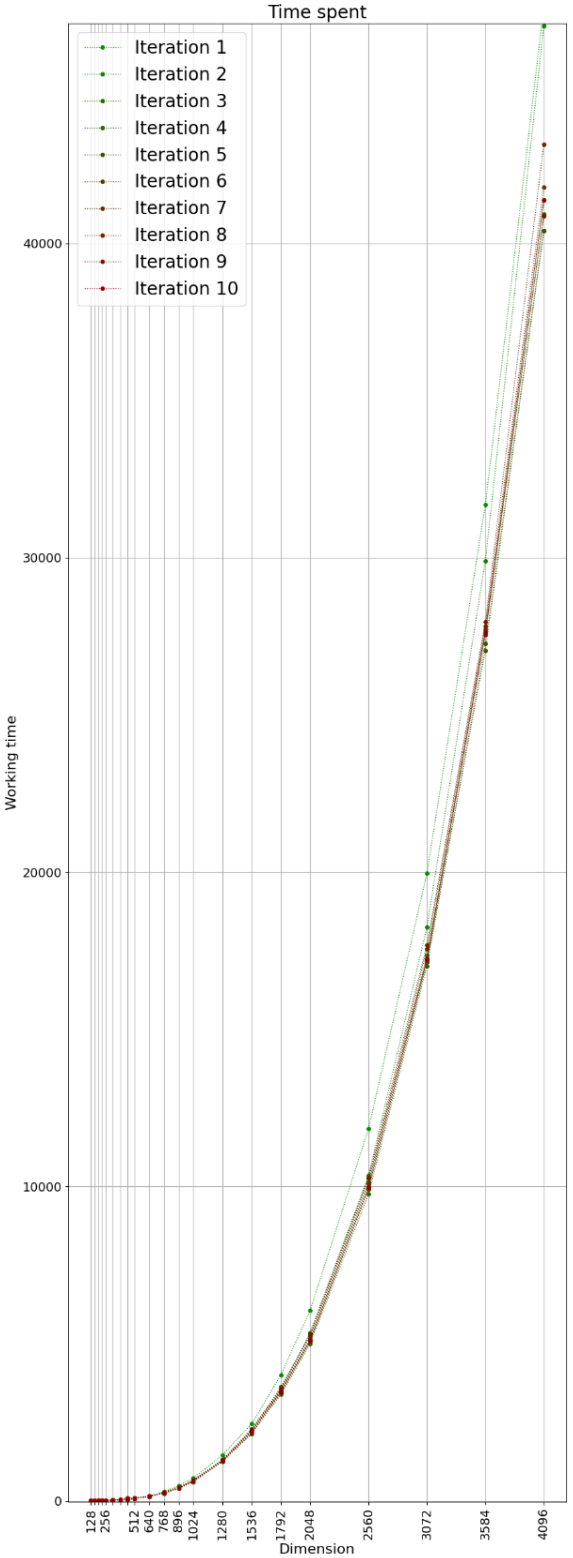
У нас есть вычисления при $n = \{128, 256, 512, 1024, 2048, 4096\}$ полученные усреднением 10 итераций для каждого n. Таким образом, $n_i^3 = (2n_{i-1})^3 = 8n_{i-1}^3$, $i \in [2, 6]$.

Получаем усредненный результат:

- $T_1 = T(n_1^3) = 1.4$ мс.
- $T_2 = T(n_2^3) = 11.1$ мс.
- $T_3 = T(n_3^3) = 79.5$ мс.
- $T_4 = T(n_4^3) = 645$ мс.
- $T_5 = T(n_5^3) = 5263.7$ мс.
- $T_6 = T(n_6^3) = 42483.5$ мс.



(Рисунок 5. Среднее время работы алгоритма Холецкого)



(Рисунок 6. Время работы каждой итерации алгоритма Холецкого)

Получается, что $\frac{T_2}{T_1} = 7.93$, $\frac{T_3}{T_2} = 7.16$, $\frac{T_4}{T_3} = 8.11$, $\frac{T_5}{T_4} = 8.16$, $\frac{T_6}{T_5} = 8.03$

Теоретическая сложность должна быть примерно O(n^3) - то есть кубическая. Вычислим к чему стремиться теоретическая оценка $\frac{T(2n)}{T(n)}$ и сравним с полученными значениями. $\frac{O((2n)^3)}{O(n^3)} = \frac{O(8n^3)}{O(n^3)} = 8$. Оценки приблизительно равны, следовательно, коэффициент увеличения по времени примерно равен 8.

Источники

http://omega.sp.susu.ru/books/conference/PaVT2015/full/058.pdf
Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. - Алгоритмы. Построение и анализ. Издание 3-е (2013)
William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery - Numerical Recipes in C (2002)

Характеристики вычислительной среды и оборудования

Вычислительная среда: Microsoft Visual Studio 2019
Оборудование:

- AMD Ryzen 7 3700U 2.30 GHz
- 8 ГБ оперативной памяти