МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

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

Институт информационных технологий математики и механики

Кафедра теоретической, компьютерной и экспериментальной механики

Направление подготовки 09.03.04 Программная инженерия

*Производственная практика*

*по получению профессиональных умений и опыта профессиональной деятельности*

**Базовые матричные алгоритмы и их применения**

Исполнитель: студент 381808-2 группы

Огнев Денис Вячеславович

Руководитель: доцент кафедры АГДМ,

Веселов Сергей Иванович

Нижний Новгород, 2020 г.

**Оглавление**

[**1** **Введение** 2](#_Toc59472743)

[**2** **Постановка задачи** 3](#_Toc59472744)

[**3** **Изучение материала** 4](#_Toc59472745)

[3.1 Определение Эрмитовой нормальной формы 4](#_Toc59472746)

[3.2 Алгоритм для матриц с полным рангом строки 4](#_Toc59472747)

[3.3 Алгоритм для общего случая 5](#_Toc59472748)

[3.4 Решение некоторых задач 5](#_Toc59472749)

[3.5 Определение проблемы ближайшего вектора и способы ее решения 6](#_Toc59472750)

[3.6 Жадный метод: Алгоритм ближайшей плоскости Бабая 7](#_Toc59472751)

[3.7 Метод Ветвей и Границ: Алгоритм Перечисления 7](#_Toc59472752)

[**4** **Заключение** 9](#_Toc59472753)

[**5** **Ссылки на источники** 10](#_Toc59472754)

# **Введение**

В данной работе рассматриваются различные решеточные алгоритмы. Описанные алгоритмы работают за полиномиальное время позволяют решать различные задачи, например проблему эквивалентности (образуют ли два базиса одинаковую решетку) или проблему включения (включение вектора в решетку). Для решения многих таких задач применяется Эрмитова нормальная форма. Эрмитова нормальная форма также используется при решении задач общей алгебры, линейного программирования, криптографии и может быть использована для решения систем линейных диофантовых уравнений.

Также в данной работе рассматривается проблема ближайшего вектора (ПБВ), для ее решения рассматриваются такие алгоритмы, как Жадный метод: Алгоритм ближайшей плоскости Бабая, который работает за полиномиальное время и решает ПБВ за полиномиальное время, но дает только приближенные решения, и метод Ветвей и Границ: алгоритм перечисления, который точно решает ПБВ, но работает за суперэкспоненциальное время

# **Постановка задачи**

Необходимо перевести и проанализировать предложенную статью, а также изучить другие материалы по данной теме, после чего написать программу, которая должна уметь вычислять Эрмитову нормальную форму (ЭНФ) и решать следующие задачи:

* Вычисление базиса для решетки L(B) от заданного множества векторов B
* Проблема эквивалентности двух решеток
* Объединение двух решеток
* Проблема содержания одной решетки в другой
* Проблема включения вектора в решетку
* Решение линейных систем Ax = b
* Получение двойственной решетки
* Пересечение двух решеток
* Нахождение наименьшей циклической решетки от заданного множества векторов B
* Проблема ближайшего вектора

# **Изучение материала**

## Определение Эрмитовой нормальной формы

*Невырожденная матрица* **B** = [**b**1*, ...,* **b***n*] ∈ *m*×*n* является Эрмитовой нормальной формой (ЭНФ), если

* Существует 1 ≤ *i*1 *< ... < ih* ≤ *m такое, что bi,j* ≠ 0 ⇒ (*j < h*) ∧ (*i* ≥ *ij*) *(строго убывающая высота столбца).*
* *Для всех , т.е. все элементы в строках ij приведены по модулю .*

## Алгоритм для матриц с полным рангом строки

Дана матрица . Предположим, что у нас есть процедура AddColumn, которая работает за полиномиальное время и принимает на вход квадратную невырожденную ЭНФ матрицы и вектор **b**, а возвращает ЭНФ матрицы [**H|b**]. ЭНФ от **B** может быть вычислена следующим образом:

1. Применить алгоритм Грама-Шмидта к столбцам **B**, чтобы найти *m* линейно независимых столбцов. Пусть **B’** – матрица размера , заданная этими столбцами.
2. Вычислить *d*=det(**B’**), используя алгоритм Грама-Шмидта или любую другую процедуру с полиномиальным временем. Пусть будет диагональной матрицей с *d* на диагонали.
3. Для *i*=1,…,*n* пусть результат применения AddColumn ко входу .
4. Вернуть .

Алгоритм AddColumn на вход принимает квадратную невырожденную ЭНФ матрицы и вектор и работает следующим образом. Если m = 0, то тут ничего не надо делать, и мы можем сразу выйти с возвращением **H**. В другом случае, пусть и и дальше:

1. Вычислить и целые x,y такие, что , используя расширенный НОД алгоритм.
2. Применить унимодулярное преобразование к первому столбцу из **H** и **b** чтобы получить
3. Добавить соответствующий вектор из к **b**’’, чтобы сократить его элементы по модулю диагональных элементов из **H’**.
4. Рекурсивно вызвать AddColumn на вход **H**’ и **b**’’ чтобы получить матрицу **H’’.**
5. Добавить соответствующий вектор из к **h**’ чтобы сократить его элементы по модулю диагональных элементов из **H’’**.
6. Вернуть

## Алгоритм для общего случая

1. Запустить процесс ортогонализации Грамма-Шмидта к строкам из **B,** и пусть – это множество индексов, такое, что . Определим операцию проецирования . Заметим, что строки линейно независимы и любая другая строка может быть выражена как линейная комбинация предыдущих строк Следовательно, однозначна, когда ограничена к , и ее инверсия может быть легко вычислена, используя коэффициенты Грама-Шмидта .
2. Определить новую матрицу **B’**=, которая полноранговая, и запустить алгоритм, данный в предыдущем пункте, чтобы найти ЭНФ **B**’’ от **B’.**
3. Применить функцию обратную операции проецирования, , к ЭНФ, определенной в предыдущем шаге(**B’’**), к данной матрице **H**. Легко заметить, что входят в ЭНФ. Следовательно, **H** является ЭНФ **B**.

## Решение некоторых задач

**Основная проблема** Дано множество рациональных векторов **B**, мы хотим вычислить базис для решетки .

Эта проблема мгновенно решается (за полиномиальное время) путем вычисления ЭНФ(**B**).

**Проблема эквивалентности** Даны для базиса **B** и **B’**, мы хотим определить, образуют ли они одинаковую решетку

Эта проблема может решена за полиномиальное время, путем вычисления **H** = ЭНФ(**B**) и **H**’ = ЭНФ(**B**’), и проверки равенства **H** и **H**’.

**Объединение решеток** Даны два базиса **B** и **B**’, мы хотим определить базис для наименьшей решетки, включающей и , и .

Легко увидеть, что эта решетка сгенерирована от [**B** | **B**’], значит базис для решетки может быть легко вычислен, как ЭНФ([**B** | **B**’]).

**Проблема содержания** Даны два базиса **B** и **B**’, мы хотим определить верно ли, что подрешетка , т.е., .

Эта проблема легко сводится к проблеме объединения и равенства: тогда и только тогда, когда . Итого, чтобы проверить включение нам нужно только вычислить ЭНФ([**B | B**’]) и ЭНФ(**B**) и проверить эквивалентность базисов ЭНФ.

**Проблема включения** Дана решетка **B** и вектор **v**, мы хотим определить верно ли, что .

Это немедленно сводится к проблеме содержания путем проверки верно ли, что . Если нам нужно проверить включение для множества векторов тогда удобно сперва вычислить **H** = ЭНФ(**B**), и потом для каждого *i* проверять **H** = ЭНФ([**H |** ]). Заметим, что ЭНФ([**H |** ]) может быть легко вычислена намного быстрее, чем ЭНФ для матричного вычисления, потому что большинство матриц уже являются Эрмитовой Нормальной Формой.

**Решение линейных систем** Пусть **Ax = b** будет системой линейных уравнений. Мы хотим найти решение **x**, или сказать, что решений нет. (Мы знаем, что это может быть выполнено за арифметических операций. Проблема в том, чтобы показать, что числа остаются малыми.)

Мы можем легко сказать, имеет ли система решения путем запуска Грама-Шмидта на [**A | b**], и проверки, что последний столбец равен **b**\* = **0.** Также, мы можем остановить наше внимание на решениях, которые используют переменные, для которых . Итак, предположим, что столбцы из **A** линейно независимы, и **Ax = b** для произвольного **x**. Мы также можем избавиться от избыточных уравнений путем запуска Грама-Шмидта на строки [**A** | **b**] и убрав уравнения, для которых соответствующий ортогонализованный вектор равен **0.**

Итак, мы свели проблему решения произвольной системы линейных уравнений к решению системы **Ax = b**, где **A** это невырожденная квадратная матрица. Потом, система может быть легко решена путем вычисления ЭНФ (или, что эквивалентно, ортогонализацией Грама-Шмидта) , чтобы получить , где **C** это треугольная матрица, и затем решить эквивалентную треугольную систему **Cx = d** обратной подстановкой. Легко увидеть, что в этом случае все задействованные числа гарантированно не будут слишком большими, и система может быть решена за полиномиальное время. (Примечание: это далеко не самый быстрый способ решать систему линейных уравнений. Известны намного более быстрые методы.)

В частном случае, это показывает, что матрица, обратная к невырожденной квадратной матрице **A** может быть вычислена за полиномиальное время путем решения уравнений . Обратная матрица дана как [].

**Двойственная решетка** Дан решеточный базис **B,** вычислить двойственный базис **D**, т.е., базис **D** такой, что и span(**B**) = span(**D**).

Эта проблема легко решается путем вычисления D как D=B〖(B^T B)〗^(-1). Заметим, что это вычисление включает в себя три матричных умножения и одно получение обратной матрицы.

**Пересечение двух решеток** Даны два базиса **B** и **B**’, мы хотим определить базис для пересечения .

Легко показать, что если и двойственные решетки от и , тогда двойственная от это **.** Итак, базис для пересечения найден путем вычисления **D**, **D**’ и **H =** ЭНФ([**D | D**’]), и потом вычисление двойственной от **H.**

**Цикличная решетка** Пусть r(**x**) циклическое вращение вектора **x**, т.е., . Дано множество векторов **B**, найти циклическую решетку, полученную от **B**, т.е. наименьшую циклическую решетку, содержащую **B**.

Эта проблема легко решается, учитывая вектора для всех и , и вычислением решетки, полученной от этих векторов.

Похожая проблема состоит в решение, является ли данная решетка циклической, т.е., верно ли, что (в таком случае, . Задача решение циклической решетки легко решается путем вычисления ЭНФ(**B**) и ЭНФ(*r*(**B**)) и проверкой равенства этих двух матриц.

## Определение проблемы ближайшего вектора и способы ее решения

Рассмотрим Проблему Ближайшего Вектора(ПБВ): Дан решеточный базис и целевой вектор , найти точку решетки **Bx** такую, что минимизировано. Это задача оптимизации (минимизации) с допустимыми решениями, заданными всеми целочисленными векторами , и целевой функцией f(**x**) = .

Пусть **B = [B**’, **b**] и **x** = (**x**’, *x*), где , **,** и .

Заметим, что если вы зафиксируете значение *x*, то задача ПБВ (**B, t**) потребует найти значение такое, что минимизировано. Это также экземпляр ПБВ (**B’, t’**) с модифицированным , и решеткой меньшего размера . В частности, пространство решений сейчас состоит из (n – 1) целочисленных переменных **x**’. Это говорит о том, что можно решить ПБВ путем установки значения **x** по одной координате за раз.

Есть несколько способов превратить этот подход к уменьшению размерности в алгоритм, используя некоторые стандартные методы алгоритмического программирования. Простейшие техники такие:

* жадный метод, который выдает приближенные значения, но работает за полиномиальное время, и
* метод ветвей и границ, который выдает точные решение за суперэкспоненциальное время.

Оба метода основаны на очень простой нижней оценке целевой функции:

## Жадный метод: Алгоритм ближайшей плоскости Бабая

Жадный метод состоит из выбора переменных по одной, определяющих пространство решений, каждый раз выбирая значение, которые выглядит более многообещающим. В нашем случае, выберем значение *x*, которое дает наименьшее возможное значение для нижней границы . Напомним, что и , и что для любого фиксированного значения *x*, ПБВ (**B, t**) сводится к ПБВ (**B’, t’**), где . Используя для нижней границы, мы хотим выбрать значение *x* такое, что

Как можно меньшее. Это очень простая 1-размерная ПБВ проблема (с решеткой и целью ), которая может быть сразу решена установкой

где компонента вектора **b**, ортогональная другим базисным векторам. Вот и все! Полный алгоритм приведен ниже:

Greedy([], **t**) = **0**

Greedy([**B** *,***b**],**t**) = *c* · **b** + Greedy(**B**, **t** − *c* · **b**)

где **b**∗ = **b**⊥**B**

*x* =

*c* =

## Метод Ветвей и Границ: Алгоритм Перечисления

Структура похожа на жадный алгоритм, но вместо жадной установки на наиболее подходящее значение (то есть на то, для которого нижняя граница расстояния минимальна), мы ограничиваем множество всех возможных значений для *x*, и затем мы переходим на каждую из них для решения каждой соответствующей подзадачи независимо. В заключении, мы выбираем наилучшее возможное решение среди возвращенных всеми ветками.

Чтобы ограничить значения, которые может принимать *x*, нам также нужна верхняя граница расстояния от цели до решетки. Ее можно получить несколькими способами. Например, можно просто использовать (расстояние от цели до начала координат) в качестве верхней границы. Обычно лучше использовать жадный алгоритм, чтобы найти приближенное решение **v** = Greedy(**B, t**), и использовать в качестве верхней границы. Как только верхняя граница *u* установлена, можно ограничить переменную *x* такими значениями, что .

Окончательный алгоритм похож на жадный и описан ниже:

Branch&Bound([],**t**) = **0**

Branch&Bound([**B***,***b**],**t**) = closest(*V* ,**t**)

где **b**∗ = **b**⊥**B**

**v** = Greedy(**B**,**t**)

*X* = {}

*V* = {*x* · **b** + Branch&Bound(**B***,***t** − *x* · **b**):*x* ∈ *X*}

где closest(*V, t*) выбирает вектор в ближайший к цели **t.**

Как и для жадного алгоритма, производительность алгоритма Ветвей и Границ может быть произвольно плохой, если мы сперва не сократим базисы.

# **Заключение**

В ходе перевода была получена информация о том, как находить ЭНФ и решать различные задачи на решетках, а также рассмотрена проблема ПБВ и решена с помощью двух алгоритмов – жадного метода и метода ветвей и границ.

# **Ссылки на источники**

* <https://cseweb.ucsd.edu/classes/sp14/cse206A-a/index.html>
* <https://en.wikipedia.org/wiki/Hermite_normal_form>
* <https://amp.ww.google-info.org/8261985/1/ermitova-normalnaya-forma.html>
* <https://en.wikipedia.org/wiki/Lattice_problem>
* <https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/cvp.pdf>