Санкт-Петербургский государственный университет

Кафедра математической теории моделирования систем управления

Осипов Анатолий Александрович

Выпускная квалификационная работа бакалавра

Экспериментальная проверка линейной скорости сходимости МДМ-метода

Направление 010400

Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Тамасян Г. Ш.

Санкт-Петербург

2017

Содержание

Введение

Математическое программирование — это раздел математики, который разрабатывает теорию, численные методы решения многомерных экстремальных задач с ограничениями. Её отличием от классической математики является то, что оно занимается методами решения задач нахождения из всех возможных вариантов наилучших. Наилучшими вариантами называют те варианты, при которых достигается, например, минимальные затраты или убытки, максимальная прибыль или производительность. Основным инструментом для решения таких задач является математичекое моделирование. Математическая модель – это «перевод» изучаемой ситуации и всех существующих условий и сведений на язык математики в виде уравнений, неравенств и тождеств. И если все эти соотношения линейные, то вся задача называется задачей линейного программирования.

МДМ-метод, который является основным рассматриваемым объектом данной работы, возник как решение вспомогательной задачи нахождения ближайшей точки выпуклого многогранника к началу координат в оптимальном управлении [1] и негладкой оптимизации [2].

Формальная постановка задачи

Пусть у нас есть конечное множество точек в пространстве. На эти точки натянута выпуклая оболочка. Очевидно, что эта оболочка является замкнутым выпуклым множеством. Необходимо найти точку выпуклой оболочки, которая является ближайшей к началу координат. В работе рассмотрен один из методов, а именно МДМ-метод, позволяющих найти такую точку. Но в некоторых случаях МДМ-метод плохо сходится, когда начало координат принадлежит многограннику. В связи с этим лучше всего сначала решить вспомогательную задачу проверки принадлежности начала координат выпуклой оболочке заданных точек, а после переходить непосредственно к основному методу, которому посвящена данная работа. Один из способов проверки принадлежности начала координат выпуклому многограннику – это сведение её к решению задачи линейного программирования.

Так же в работе представлены результаты численных экспериментов, задачей которых было проверить теоретически доказанную линейную зависимость скорости сходимости МДМ-метода от количества точек в пространстве.

Математическая постановка задачи

Пусть в m-мерном евклидовом пространстве E_m задано конечное множество точек $H = \{a_i\}_{i=1}^n$. Обозначим через G выпуклую оболочку множества H. Перед нами стоит задача: *найти точку из G, ближайшую (в евклидовой норме) к началу координат.* Задачу можно записать так:

$$\|v\|^2 \to \min_{v \in G} \tag{1}$$

Задача (1) имеет решение и оно единственно (см. [6]). Обозначим его v_*.

Как уже отмечалось выше, для проверки принадлежности начала координат множеству G необходимо решить вспомогательную задачу ли-

нейного программирования в пространстве векторов $W = (p_1, \ldots, p_n, u)$:

$$\begin{gathered}
\min u \\
\sum_{i=1}^{n} p_i a_k^{(i)} - u \le 0,\ k \in [1:m], \\
-\sum_{i=1}^{n} p_i a_k^{(i)} - u \le 0,\ k \in [1:m], \\
\sum_{i=1}^{n} p_i = 1;\ p_i \ge 0, i \in [1:n];\ u \ge 0,
\end{gathered} \tag{2}$$

где $a_i = (a_1^{(i)}, \ldots, a_m^{(i)})$ — точки из H.

Обозначим через $w_0 = (p_1^{(0)}, \ldots, p_n^{(0)}, u^{(0)})$ решение этой задачи. Нетрудно проверить (см. [6]), что $\mathbb{O} \in G$ тогда и только тогда, когда $u^0 = 0$, и если это так, то $v_* = 0$, если же $u^{(0)} > 0$, то необходимо использовать МДМ-метод (см. [5]) и в качестве начального приближения использовать точку

$$v_0 = \sum_{i=1}^{n} p_i^{(0)} a_i,$$

где $(p_1^{(0)}, \ldots, p_n^{(0)})$ — состоит из первых n координат вектора w_0.

Таким образом, для решения поставленной задачи необходимо рассмотреть методы решения вспомогательной задачи - ЗЛП (2) и МДМ-метод.

1 Задача линейного программирования

В линейном программировании существует огромное количество методов – это и прямой симплекс-метод, и всевозможные модифицированные симплекс-методы с дополнительными правилами и улучшениями, которые создаются при естественных соображениях охватить как можно более широкий класс задач, чтобы в случае каких-то затруднений и проблем метод всё равно бы работал с конечным числом шагов. Выбор рассмотренных методов решения задач линейного программирования основан на статье И. В. Агафонова и В. А. Даугавет «Вырожденность в задачах линейного программирования» [3]. Это один из наиболее удачных методов решения ЗЛП, который обеспечивает решение задачи за конечное число шагов.

1.1 Терминология

Терминология описываемых методов взята из книги В. А. Булавского, Р. А. Зверягиной и М. А. Яковлевой «Численные методы линейного программирования» [4]. Но поскольку она принята не только в данной книге, я буду использовать необходимые термины без приведения определений, за исключением случаев, когда будут требоваться пояснения.

1.2 Особенности задач линейного программирования

Необходимо заметить, что очень часто задача, требуемая решения, имеет некоторые особенности, которые нельзя не озвучить в данной работе. Такими особенностями являются:

- ситуация вырождения;
- зацикливание;
- выбор нарушения условий оптимальности;
- вычислительная устойчивость метода;
- изменение условий задачи;

- смягчение ограничений.

Все эти особенности подробно описаны в вышеуказанной книге [4], поэтому мы не будем останавливаться на этом. Но при решении конкретной задачи необходимо быть очень внимательным и предвидеть проблемы, которые могут ждать в дальнейшем.

1.3 Описание методов

1.3.1 Простой симплекс-метод

Рассмотрим общую идею метода преобразования задачи (простой симплекс-метод)(см. [4]). В этом пункте будем рассматривать лишь задачу, поставленную в канонической форме

$$f(x) := c[N] \times x[N] \to \min_{x \in \Omega}, \tag{3}$$

где

$$\Omega = \{x[N] \mid A[M,N] \times x[N] = b[M],\ x[N] \geq \mathbb{O}\},$$
$$M = \{1, 2, \ldots, m\}, \qquad N = \{1, 2, \ldots, n\}.$$

Это связано с тем, что для общей задачи схема выглядит более громоздко, а во-вторых, — эта вычислительная схема в задачах линейного программирования применяется сравнительно редко ввиду большого объёма вспомогательной информации. Кроме того, любую задачу можно свести к вышеназванной форме.

Итак, вначале метода мы проводим некоторые преобразования и переходим к задаче, эквивалентной первоначальной. Далее мы находим допустимый столбец решений, который при выполнении определённых условий является и оптимальным решением исходной задачи, но если он не будет являться оптимальным решением, то производится пересчёт информации и дальнейшее улучшение решения. Метод последовательного улучшения, таким образом, можно трактовать как поиск базисного множества, при котором упомянутое выше эквивалентное преобразование приводит к задаче с очевидным решением.

1.3.2 Модифицированный симплекс-метод

В докладе [3] представлен конечный алгоритм для решения ЗЛП в первой канонической форме, основанный на модифицированном симплекс-методе, называемом также симплекс-методом с обратной матрицей. Алгоритм не требует ни предварительных расчётов, ни наложения каких-либо дополнительных условий на матрицы A, b, c из (3), в том числе условия невырожденности задачи, обеспечивающего конечную сходимость симплекс-метода. Конечную сходимость рассматриваемого алгоритма гарантирует применение правила Блэнда для предотвращения зацикливания.

Алгоритм состоит из двух последовательных этапов, на которых симплекс-методом решаются необходимые задачи линейного программирования: вспомогательная задача — на первом этапе и некоторая задача, равносильная исходной — на втором. Ниже приведено краткое описание одной итерации модифицированного симплекс-алгоритма с применением правила Блэнда. Решается исходная задача в канонической форме. В начале итерации имеются:

- базис;
- соответствующий этому базису базисный план;
- соответствующая этому базису обратная базисная матрица.

Проводятся действия:

1. Проверка базисного плана на оптимальность.
2. Определение направления убывания целевой функции.
3. Определение длины шага.
4. Пересчёт плана.
5. Смена базиса (в этом пункте применяетс правило Блэнда, описанное в [3]).

1.4 Линейное программирование в Matlab

В работе применяется функция linprog программы Matlab. Данная функция решает задачу линейного программирования в форме:

$$\begin{gathered} f^T \cdot x \to \inf \\ A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub. \end{gathered} \tag{4}$$

Основными входными данными linprog являются: вектор коэффициентов целевой функции f, матрица ограничений-неравенств A, вектор правых частей ограничений-неравенств b, матрица ограничений-равенств Aeq, вектор правых частей ограничений-равенств beq, вектор lb, ограничивающий план x снизу, вектор ub, ограничивающий план x сверху. На выходе функция linprog даёт оптимальный план x задачи (4) и экстремальное значение целевой функции $fval$.

1.5 Пример работы функции linprog

Рассмотрим подробнее функцию linprog на конкретном примере. Найдём решение задачи линейного программирования вида:

$$\begin{gathered} \min u \\ p_1\begin{pmatrix}2\\1\end{pmatrix} + p_2\begin{pmatrix}-2\\1\end{pmatrix} + p_3\begin{pmatrix}0\\-2\end{pmatrix} - u \leq 0, \\ p_1\begin{pmatrix}-2\\-1\end{pmatrix} + p_2\begin{pmatrix}2\\-1\end{pmatrix} + p_3\begin{pmatrix}0\\2\end{pmatrix} - u \leq 0, \\ p_1 + p_2 + p_3 = 1;\ p_{1,2,3} \geq 0,\ u \geq 0. \end{gathered} \tag{5}$$

Разберёмся что есть что, опираясь на систему (4) из предыдущего параграфа. Получаем соотношения:

$$x = \begin{pmatrix} p_1 & p_2 & p_3 & u \end{pmatrix}^T,$$

$$f = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix},$$

$$A = \begin{pmatrix} 2 & -2 & 0 & -1 \\ 1 & 1 & -2 & -1 \\ -2 & 2 & 0 & -1 \\ -1 & -1 & 2 & -1 \end{pmatrix},$$

$$b = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}^T,$$

$$Aeq = \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix},$$

$$beq = (1),$$

$$lb = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}^T.$$

Ограничения на вектор x нет, поэтому нет необходимости задавать вектор ub (см. (4)). Теперь введём нужные вектора и матрицы в систему Matlab. Соответствующая программа (m-файл)* выглядит так:

```
clear all
close all
clc % удаляются все текущие переменные из памяти MATLAB, закрываются все графические окна, очищается экран консоли
f=[0 0 0 1] % задаётся вектор длины четыре
A=[2 -2 0 -1; 1 1 -2 -1; -2 2 0 -1; -1 -1 2 -1] % строки матрицы разделяются точкой с запятой
b=zeros(4,1) %  задаётся нулевой вектор длины четыре
Aeq=[1 1 1 0]
beq=[1]
lb=zeros(4,1)
[x,fval]=linprog(f,A,b,Aeq,beq,lb);
x
fval
```

Запустив программу получим сообщение:

```
Optimization terminated.
x=
  0.3333
  0.3333
```

0.3333

0.0000

fval=

2.3148e-13

Если какой-то из входных параметров отсутствует, на его место следует поставить квадратные скобки [], за исключением случая, когда это последний параметр в списке. Например, если нужно решить задачу с верхними ограничениями на x, то оператор вызова функции linprog будет выглядеть так:

[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub).

А если нужно решить задачу без ограничений-равенств, то оператор вызова функции linprog будет выглядеть так:

[x,fval] = linprog(f,A,b,[ ],[ ],lb,ub).

Итак, проанализируем результат нашей работы. Мы знаем, что если $u = 0$, то $\mathbb{O} \in G$. Как мы видим, программа выдала нам вектор $x = [0.3333\ 0.3333\ 0.3333\ 0.0000]^T$, в котором значению u соответствует последний элемент вектора, равный как раз нулю. Из этого мы можем сделать вывод, что выпуклая оболочка точек $a_1 = \binom{2}{1}$, $a_2 = \binom{-2}{1}$, $a_3 = \binom{0}{-2}$ содержит начало координат, в чём мы так же можем убедиться, если нарисуем данные точки на плоскости с декартовой системой координат.

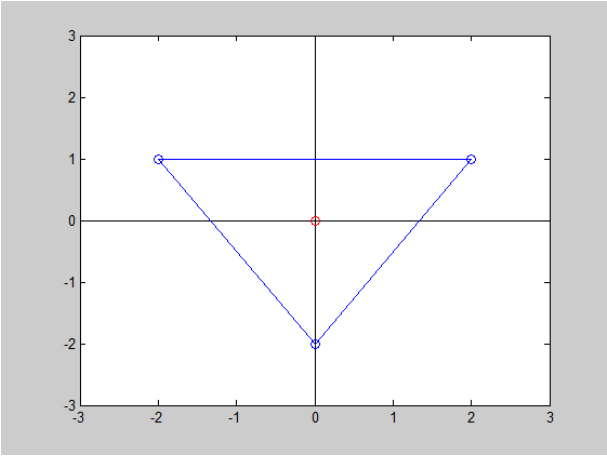

Рис. 1: Выпуклая оболочка точек a_1, a_2, a_3

2 МДМ-метод

Как уже было сказано во введении, МДМ-метод возник как решение вспомогательной задачи нахождения ближайшей точки выпуклого многогранника к началу координат в оптимальном управлении [1] и негладкой оптимизации [2]. Он получил название от трёх известных личностей, создавших этот метод, Митчелла Б. Ф., Демьянова В. Ф. и Малоземова В. Н.

2.1 Предварительные сведения

Напомним постановку задачи. Пусть в m-мерном евклидовом пространстве E_m задано конечное множество точек $H = \{a_i\}_{i=1}^n$. Обозначим через G выпуклую оболочку множества H. Перед нами стоит задача: *найти точку из G, ближайшую (в евклидовой норме) к началу координат.* Задачу можно записать так:

$$\|v\|^2 \to \min_{v \in G}. \tag{6}$$

Задача (6) имеет решение и оно единственно. Обозначим его v_*. Предположим, что мы проверили принадлежность начала координат нашей выпуклой оболочке. Далее нам необходимо найти v_* МДМ-методом.

Обозначим через A матрицу со столбцами $a_1, \ldots, a_n$. Тогда любой вектор v из выпуклой оболочки G множества H допускает представление

$$v = Ap, \quad p \geq \mathbb{O}, \quad \sum_{i=1}^{n} p[i] = 1. \tag{7}$$

Множество векторов p, удовлетворяющих условиям, указанным в формуле (7), назовём P. Носитель вектора коэффициентов p обозначим $M_+(p)$, так что

$$M_+(p) = \{i \in \ 1:n \mid p[i] > 0\}.$$

Введём величину

$$\Delta(p) = \max_{i \in M_+(p)} \langle a_i, v \rangle - \min_{i \in 1:n} \langle a_i, v \rangle,$$

где $v = Ap, \ p \in P$.

Приведём необходимые для алгоритма МДМ-метода леммы из докладов [6] и [8].

Лемма 2.1. *При любом $v = Ap,\ p \in P$, справедливо неравенство*

$$\|v - v_*\|^2 \leq \Delta(p). \tag{8}$$

Из этого так же следует, что

$$\Delta(p) \geq 0 \quad \forall p \in P. \tag{9}$$

Лемма 2.2. *Равенство в* (9) *достигается тогда и только тогда, когда вектор $v = Ap$ является решением задачи* (6).

2.2 Алгоритм МДМ-метода

Возьмём начальное приближение $v_0 \in G$. Каким образом мы можем взять это начальное приближение? Мы можем выбрать любую точку из A, или, например, можем выбрать точку $v_0 = v$, для которой

$$\langle v, v\rangle \to \min_{v \in G}, \tag{10}$$

или, что то же самое

$$\langle a_i, a_i\rangle \to \min_{i \in [1:n]}. \tag{11}$$

В программе было использовано условие (11) для выбора v_0.

Теперь нам необходимо найти $p_0 \in P$, удовлетворяющее условиям (7) и $v_0 = Ap_0$. Так как наше $v_0 = a_{i_0}$, с таким индексом i_0, для которого выполнено условие (11), то вектором p будет являться вектор, у которого будет $(n-1)$ нулей и на $i_0 -$ месте единица. Зная индекс i_0, мы легко можем составить такой вектор. Но эту процедуру необходимо делать только на начальном шаге алгоритма, так как на всех следющих шагах p будет пересчитываться по специальному правилу.

Общий алгоритм МДМ-метода.

Пусть у нас уже имеется k-е приближение $v_k = Ap_k, p_k \in P$. Опишем построение v_{k+1}.

Найдём индексы $i'_k \in M_+(p_k)$ и $i''_k \in 1:n$, такие, что

$$\max_{i \in M_+(p)} \langle a_i, v_k\rangle = \langle a_{i'_k}, v_k\rangle, \tag{12}$$

$$\min_{i \in 1:n} \langle a_i, v_k\rangle = \langle a_{i''_k}, v_k\rangle. \tag{13}$$

Переобозначим

$$a_{i'_k} = a'_k, \quad a_{i''_k} = a''_k.$$

Тогда

$$\Delta_k := \Delta(p_k) = \langle a'_k - a''_k, v_k \rangle. \tag{14}$$

Если $\Delta_k = 0$, то, согласно лемме 2.2, v_k — решение задачи (6). Процесс закончен.

Пусть $\Delta_k > 0$. Введём вектор

$$v_k(t) = v_k - t p'_k (a'_k - a''_k), \quad t \in [0, 1],$$

где $p'_k = p_k[i'_k]$. Найдём t'_k по формуле

$$t'_k = \frac{\Delta_k}{p'_k \|a'_k - a''_k\|^2}.$$

Для t_k по этой формуле действует правило

$$t_k = \begin{cases} t'_k, & \text{если } t'_k < 1, \\ 1, & \text{если } t'_k \geq 1. \end{cases}$$

Теперь $v_{k+1} = v_k(t)$. Пересчёт p производится по формуле

$$p_{k+1}[i] = \begin{cases} p_k[i], & \text{при } i \neq i'_k,\ i \neq i''_k, \\ (1 - t_k) p_k[i'_k], & \text{при } i = i'_k, \\ p_k[i''_k] + t_k p_k[i'_k], & \text{при } i = i''_k. \end{cases} \tag{15}$$

Описание МДМ-метода завершено.

Построена последовательность $v_0, v_1, \ldots$ точек из G. Если она конечна, то последний её элемент является решением задачи (6). О случае бесконечности последовательности $\{v_k\}$ рассказано в докладе [6]. Но в программе используется условие остановки алгоритма $\Delta(p) < \delta$, где $\delta > 0$ и достаточно малое, что позволяет МДМ-методу гарантированно проецировать начало координат на выпуклый многогранник.

2.3 Пример проецирования начала координат на определённый выпуклый многогранник

Чтобы разобраться, как работает алгоритм МДМ-метода, возьмём для примера шесть точек на плоскости oxy:

$$a_1 = \begin{pmatrix}1\\2\end{pmatrix}, a_2 = \begin{pmatrix}2\\1\end{pmatrix}, a_3 = \begin{pmatrix}3\\1\end{pmatrix}, a_4 = \begin{pmatrix}3\\3\end{pmatrix}, a_5 = \begin{pmatrix}1\\3\end{pmatrix}, a_6 = \begin{pmatrix}2\\2\end{pmatrix}$$

Таким образом матрица $A = \begin{pmatrix} 1 & 2 & 3 & 3 & 1 & 2 \\ 2 & 1 & 1 & 3 & 3 & 2 \end{pmatrix}$.

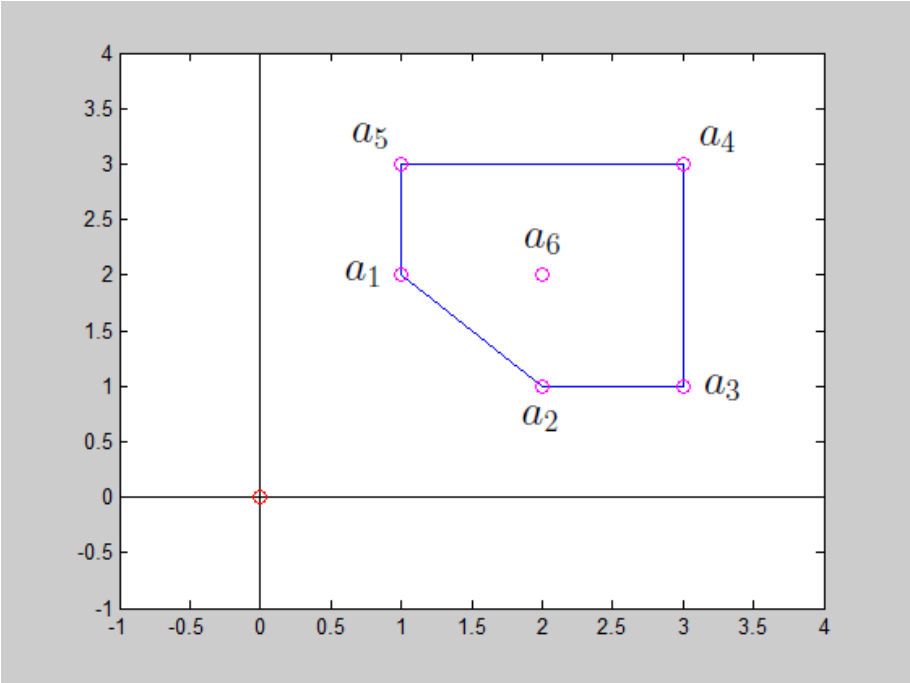

Рис. 2: Выпуклая оболочка точек a_1,a_2,a_3,a_4,a_5,a_6.

Следуя вышеописанному алгоритму, нам необходимо выбрать вектор v_0. В соответствии с условием (11), $v_0 = a_1$. Как мы видим $i_0 = 1$, поэтому вектор $p_0 = (1\ 0\ 0\ 0\ 0\ 0)$. Условия (7) выполнены. Теперь составим множество $M_+(p_0)$. Оно состоит из индексов вектора p, где элементы его положительны. Следовательно $M_+(p) = \{1\}$. Теперь выберем $i'_0 \in M_+(p_0)$ и $i''_0 \in 1 : m$, удовлетворяющие соответствующим условиям (12), (13). Т. к. множество $M_+(p_0)$ состоит из одного элемента, то $i'_0 = 1$. Теперь разберёмся со вторым индексом. Его нужно найти из условия (13) или, что то же

самое

$$\min_{i\in 1:n}\langle a_i, a_1\rangle = \langle a_{i''_k}, a_1\rangle.$$

Получаем $i''_0 = 2$. Теперь переобозначим

$$a_1 = a'_0, \quad a_2 = a''_0.$$

По формуле (14) имеем, $\Delta_0 = \langle (a'_0 - a''_0), v_0\rangle = 1 > 0$, следовательно алгоритм продолжается.

Вычислим

$$t'_0 = \frac{\Delta_0}{p'_0\|a'_0 - a''_0\|^2} = \frac{1}{2}.$$

Так как $t'_0 < 1$, то $t_0 = \frac{1}{2}$.

Далее вычислим

$$v_0(t_0) = v_0 - t_0 p'_0(a'_0 - a''_0) = \begin{pmatrix}1\\2\end{pmatrix} + \begin{pmatrix}\frac{1}{2}\\-\frac{1}{2}\end{pmatrix} = \begin{pmatrix}1.5\\1.5\end{pmatrix}.$$

Положим

$$v_1 = v_0(t) = \begin{pmatrix}1.5\\1.5\end{pmatrix}.$$

Проводим пересчёт вектора p_1 по правилу (15). Получаем

$$p_1 = (\tfrac{1}{2}\ \tfrac{1}{2}\ 0\ 0\ 0\ 0).$$

Теперь составляем множество $M_+(p_1)$, которое, как мы помним, состоит из индексов положительных элементов вектора p_1. Мы видим, что это индексы 1 и 2. Следовательно $M_+(p_1) = \{1, 2\}$. Выбираем i'_1 исходя из условия

$$\max_{i\in M_+(p)}\langle a_i, v_1\rangle = \max_{i\in 1:2}\langle a_i, v_1\rangle = \langle a_{i'_1}, v_1\rangle.$$

Максимум равен 4.5 для обоих индексов, выберем любой из них, например первый, т. е. $i'_1 = 1$.

Индекс i''_1 выбираем из условия

$$\min_{i\in 1:6}\langle a_i, v_1\rangle = \langle a_{i''_1}, v_1\rangle.$$

Минимум 4.5 достигается на 1 и 2 индексах, так же мы можем выбрать любой. Выберем первый из них, т. е. $i''_1 = 1$. Переобозначим

$$a_1 = a'_1, \quad a_1 = a''_1.$$

Т.к. $\Delta_1 = \langle (a'_1 - a''_1), v_1 \rangle = 0$, следовательно процесс закончен.

В результате у нас построена последовательность векторов $v_0 = \binom{1}{2}$, $v_1 = \binom{1.5}{1.5}$, последний из которой и является решением нашей задачи.

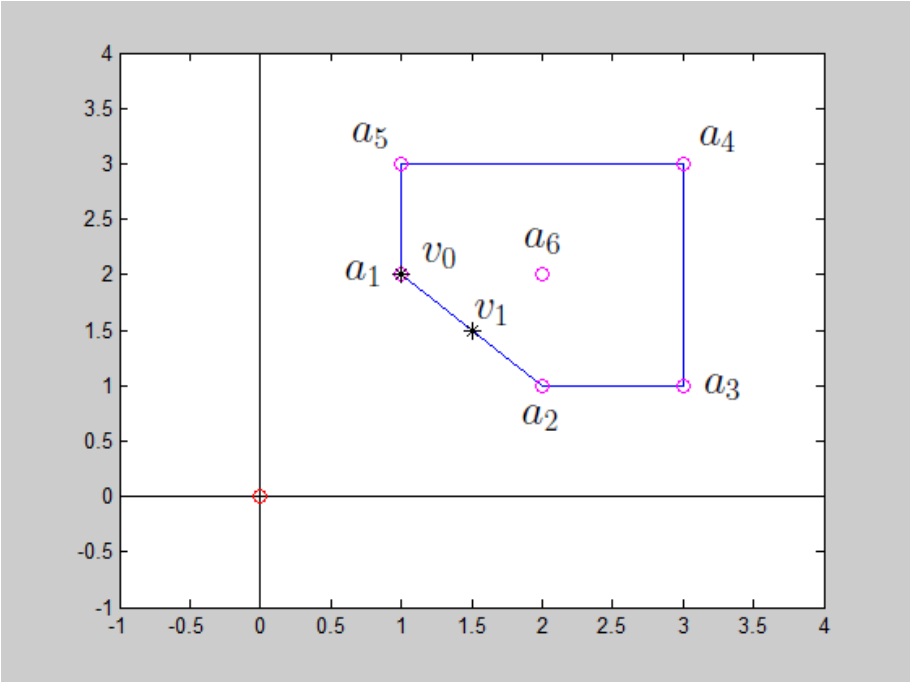

Рис. 3: Проекция начала координат на выпуклый многогранник равна v_1.

3 Численные эксперименты

Как уже было сказано прежде, основной задачей данной работы была установка зависимости между временем работы программы МДМ-метода и количеством точек в пространстве, которые нам заданы.

Численные эксперименты были разбиты на две части — проверка времени работы процедуры linprog и проверка времени работы функции МДМ-метода (код которой расположен в приложении).

3.1 Время работы процедуры linprog

Чтобы гарантировать, что начало координат принадлежит выпуклой оболочке заданных точек, были заданы $2m$ точек, где m – это размерность пространства. Они были заданны следующим образом. В двумерном случае это 4 точки, две из которых расположены случайным образом на положительно-направленных единичных ортах осей координат, а другие две –это предыдущие точки, взятые с противоположным знаком. Так же и с более высокоразмерностными пространствами. Эти $2m$ точек гарантируют нам содержание начала координат в выпуклой оболочке. Остальные точки добавляются совершенно случайным образом. Такими манипуляциями было измеряно время работы процедуры linprog для пространств 2, 3, 5, 10, 100, 500 и 1000 размерностей для наборов точек от 1000 до 10000 с шагом в 1000. На каждом наборе точек было проведено по 200 экспериментов с последующим усреднением времени работы программы. Так же необходимо оговорить тот момент, что для больших размерностей, таких как 100, 500 и 1000, время работы программы оказалось достаточно большим, поэтому на этих размерностях для каждого набора точек были проведены по 10 экспериментов, также с последующем усреднением. Результаты зависимости времени работы процедуры linprog от количества заданных точек представлены на графиках ниже (см. рис. 4).

Как мы видим, при увеличении количества точек время работы программы увеличивается линейно. Как можно заметить в пространствах 500

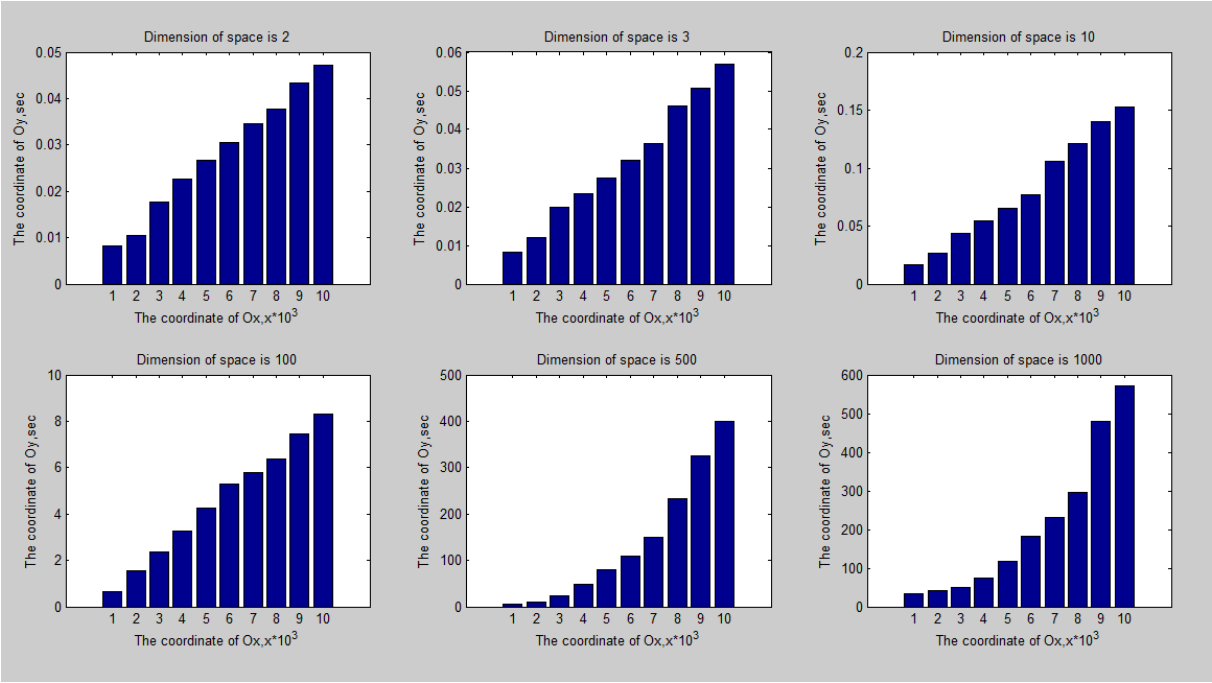

Рис. 4: Время работы процедуры linprog.

и 1000 размерностей есть некая экспоненциальная зависимость между временем и количеством точек. Возможно это связано с тем, что всё же было проведено не по 200 экспериментов на каждом наборе точек, а всего по 10, но над этим остаётся только размышлять, поскольку доказательств такой зависимости, ни теоретических, ни практических (с достаточной точностью) нам не известно.

Так же из полученных данных были построены графики зависимости времени от размерности пространства для фиксированного числа точек (рисунки 5 и 6).

Как уже говорилось выше, время работы программы при размерностях 500 и 1000 существенно превышает время работы меньших размерностей. Это отчетливо видно на этих диаграммах.

3.2 Время работы МДМ-метода

Теперь предполагается, что начало координат не содержалось в выпуклой оболочке любого набора точек. Для этого мы произведём следующие действия. Построим случайную матрицу размерности m на n, с помощью

функции $rand(m, n)$. То есть у нас будет матрица $M_{m\times n} = \{m_{ij}\}$, где $m_{ij} \in [0, 1]$. Далее, мы смещаем каждую точку на величину $\alpha + (\beta - \alpha)m_{ij}$, где $1 < \alpha < \beta$. В итоге мы переместили наши точки из единичного квадрата с левым нижним углом в начале координат, в квадрат от α до β.

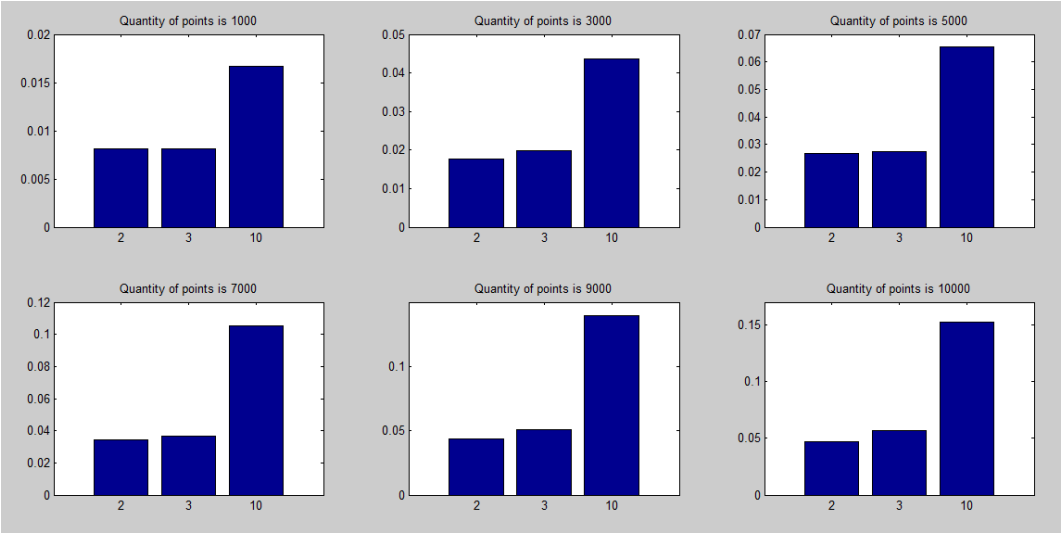

Рис. 5: Время работы процедуры linprog при 2, 3, 10 размерностях пространства.

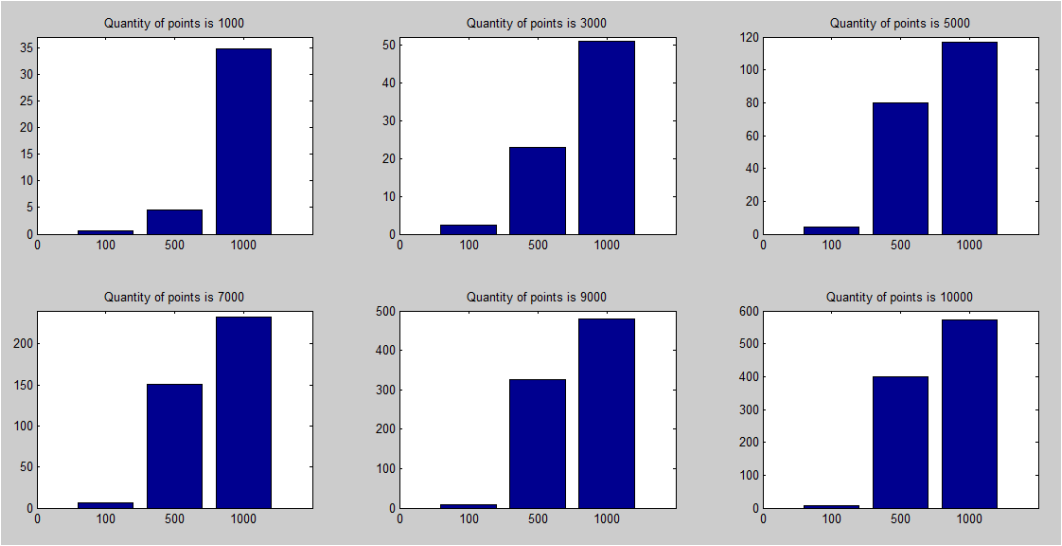

Рис. 6: Время работы процедуры linprog при 100, 500, 1000 размерностях пространства.

Размерности пространств были взяты такие же: 2, 3, 5, 10, 100, 500 и 1000. Для каждого набора точек из отрезка от 1000 до 10000 с шагом 1000, было проведено по 200 экспериментов, но время работы было не усреднено, а суммировано, так как время работы программы МДМ-метода существен-

но меньше, чем время работы процедуры linprog. Результаты экпериментов представлены на рисунке ниже:

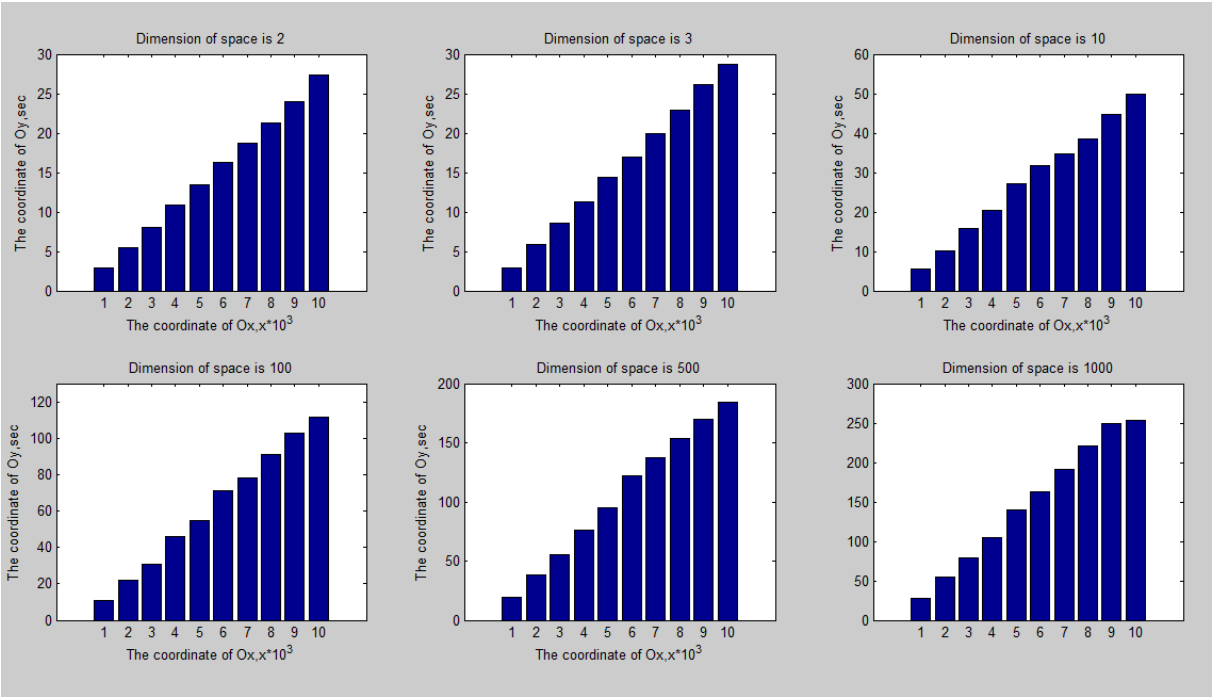

Рис. 7: Время работы функции МДМ-метода.

Мы видим, что зависимость времени работы программы от количества точек оказалась практически идеально линейной, нет видимых скачков или прогибов столбцов диаграмм.

Так же было интересно посмотреть на графики зависимости времени работы программы от размерностей пространства при фиксированном числе точек (рисунок 8).

Здесь, в отличии от результатов такой же зависимости процедуры linprog, мы видим более плавный рост времени работы метода на выбранных размерностях пространств.

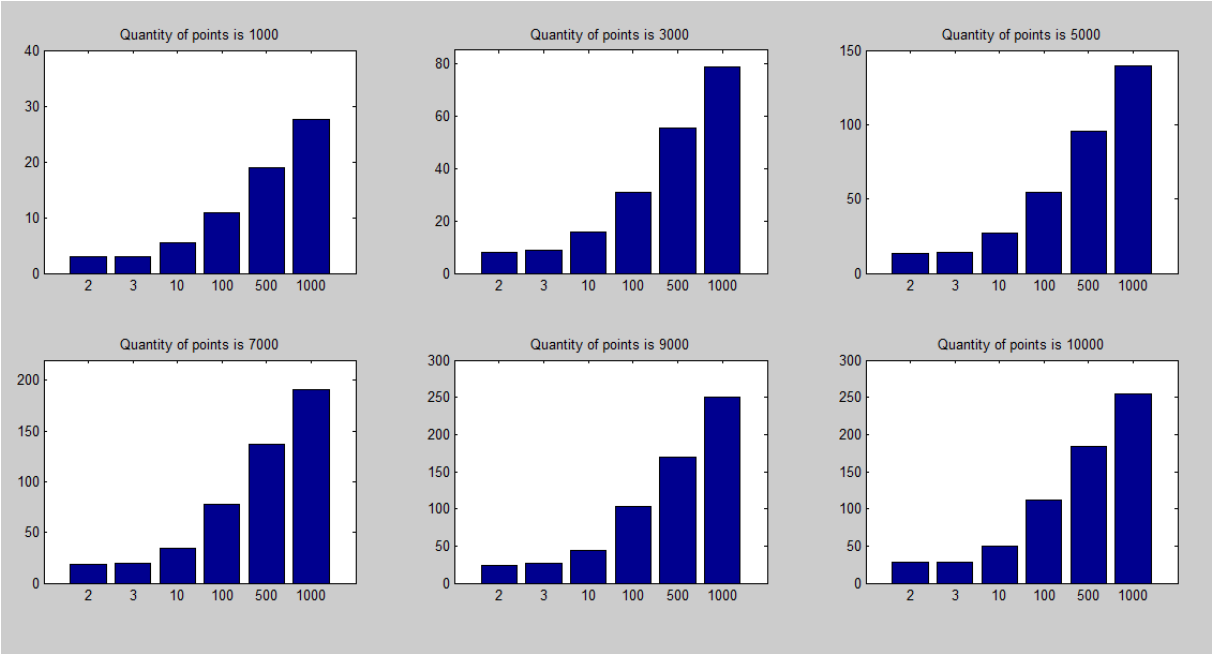

Рис. 8: Время работы функции МДМ-метода.

Заключение

Таким образом, в работе были рассмотрены методы решения задачи линейного программирования и МДМ-метод. Также экспериментально была подтверждена линейная скорость сходимости МДМ-метода, что является основной и главной частью работы, поскольку теоретическое доказательство линейной скорости сходимости уже было проведено Соловьёвой Н. А. [8], а вот экспериментальной нет.

ЛИТЕРАТУРА

[1] Gilbert E. G. *An iterative procedure for computing the minimum of quadratic form on a convex set* // J.SIAM Control. 1966. Vol. 4. No. 1. P. 61-80.

[2] Demyanov V. F. *Algorithms for some minimax broblems* // J. Computer and System Sciences. 1968. Vol. 2. No. 4. P. 342-380.

[3] Агафонова И. В. и Даугавет В. А. «Вырожденность в задачах линейного программирования». 11.12.2010г. [Электронный ресурс] `URL:http://dha.spb.ru/PDF/lpDegeneracy.pdf` (дата обращения: 12.05.2017).

[4] Булавский В. А., Зверягина Р. А. и Яковлева М. А. Численные методы линейного программирования(специальные задачи) под редакцией Л.В.Канторовича. Главная редакция физико-математической литературы издательства Наука. Москва. 1977г.

[5] Демьянов В. Ф., Малоземов В. Н. Введение в минимакс. М.: Наука, 1972. 368 с.

[6] Малозёмов В. Н. «МДМ-методу 40 лет» // Семинар CNSA & NDO. Избранные доклады. 10.12.2011 г. 10 с. [Электронный ресурс] `URL:http://dha.spb.ru/PDF/MDMMethod.pdf` (дата обращения: 12.05.2017)

[7] Сергеев А. Н., Соловьёва Н. А., Чернэуцану Е. К. Решение задач линейного программирования в среде MATLAB. 12.02.2011 г. [Электронный ресурс] `URL:http://www.apmath.spbu.ru/cnsa/programm/12022011_MatLabLP.pdf` (дата обращения: 12.05.2017).

[8] Соловьёва Н. А. Линейная скорость сходимости МДМ-метода // Семинар CNSA & NDO. Избранные доклады. 29.09.2016 г. 14 с. [Электронный ресурс] `URL:http://www.apmath.spbu.ru/cnsa/pdf/2016/LinRateMDM.pdf` (дата обращения: 12.05.2017)

Приложение

Код программы МДМ-метода(основной функции).

```
function [ otv ] = MDMf( X,n )
stop=0;
eps=0.001;
for i=1:n
z(i)=dot(X(:,i),X(:,i));
end
[~,b]=min(z);
v=X(:,b);
a1=X(:,b);
p=zeros(1,n);
p(b)=1;
for i=1:n
z(i)=dot(X(:,i),v);
end
[~,d]=min(z);
a2=X(:,d);
del=dot(a1-a2,v);
if (del<eps)
otv=v;
else
t=del/dot(a1-a2,a1-a2);
if (t>=1)
t=1;
end
v2=v-t*(a1-a2);
v=v2;
for i=1:n
if (i==b)
r(i)=(1-t)*p(b);
```

```
else if (i==d)
r(i)=p(d)+t*p(b);
else
r(i)=p(i);
end
end
end
p=r;

while stop==0
M=[];
for i=1:n
if p(i)>0
M=[M,i];
end
end
if length(M)==1
b=M(1);
else
for i=1:length(M)
w(i)=dot(X(:,i),v);
end
[~,nom]=max(w);
b=M(nom);
w=[];
end
a1=X(:,b);
for i=1:n
q(i)=dot(X(:,i),v);
end
[~,d]=min(q);
```

```
a2=X(:,d);

del=dot(a1-a2,v);
if (del<eps)
stop=1;
otv=v;
else
t=del/(p(b)*dot(a1-a2,a1-a2));
if (t>=1)
t=1;
end
v2=v-t*p(b)*(a1-a2);
v=v2;
for i=1:n
if (i==b)
r(i)=(1-t)*p(b);
else if (i==d)
r(i)=p(d)+t*p(b);
else
r(i)=p(i);
end
end
end
p=r;
end
end
end
end
```