

**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Ярославский государственный университет им. П.Г. Демидова»**

Кафедра математического анализа

Сдано на кафедру  
« \_\_\_\_ » \_\_\_\_\_ 2018 г.  
Заведующий кафедрой  
д. ф.-м. н., доцент  
\_\_\_\_\_ М.В. Невский

Выпускная квалификационная работа

**Нормы интерполяционных проекторов и экстремальные симплексы**  
(Направление подготовки бакалавров 01.03.02 Прикладная математика и  
информатика)

Научный руководитель  
канд. ф.-м. н., доцент  
\_\_\_\_\_ А.Ю. Ухалов  
« \_\_\_\_ » \_\_\_\_\_ 2018 г.

Студент группы ПМИ-41БО  
\_\_\_\_\_ А. В. Лютенков  
« \_\_\_\_ » \_\_\_\_\_ 2018 г.

Ярославль 2018 г.

## Реферат

Объем 22 с., 3 гл., 2 табл., 10 источников, 2 прил.

**Ключевые слова:** невырожденный симплекс, интерполяционный проектор, норма проектора, минимальная норма проектора, минимизация функции, Dlib, Eigen, Boost, алгоритм Бройдена-Флетчера-Голдфарба-Шанно

Объектом исследования является задача о построении минимального проектора (проектора, имеющего минимальную норму) при интерполяции непрерывной на кубе функции с помощью полиномов  $n$  переменных степени не выше единицы. Оценки для норм минимальных проекторов играют важную роль в теории приближений. Точные значения минимальной нормы проектора в настоящее время известны только для  $n=1,2,3,7$ .

Цель работы —разработать компьютерную программу для численной минимизации функции многих переменных и применить ее для решения задачи о минимальном проекторе.

В результате была реализована компьютерная программа для получения верхних оценок минимальной нормы проектора. Удалось улучшить оценки норм минимальных проекторов в случаях  $n=5,6,10,14,18,20$ .

# Содержание

<b>Введение</b>	<b>4</b>
<b>1. Задача линейной интерполяции на n-мерном кубе</b>	<b>5</b>
1.1. Интерполяционный проектор . . . . .	5
1.2. Норма интерполяционного проектора. Минимальная норма проектора	6
<b>2. Компьютерная программа для расчета <math>\theta_n</math></b>	<b>7</b>
2.1. Dlib . . . . .	7
2.2. Eigen . . . . .	7
2.3. Boost . . . . .	8
2.4. Реализация программы . . . . .	8
2.4.1. Алгоритм Бroyдена-Флетчера-Голдфарба-Шанно (BFGS) . . . .	8
<b>3. Результаты</b>	<b>10</b>
<b>4. Заключение</b>	<b>11</b>
<b>Приложение 1</b>	<b>13</b>
<b>Приложение 2</b>	<b>20</b>

# Введение

В данной ВКР рассматривается задача о построении минимального проектора (проектора, имеющего минимальную норму) при интерполяции непрерывной на кубе функции с помощью полиномов  $n$  переменных степени не выше единицы. Оценки для норм минимальных проекторов играют важную роль в теории приближений. Неравенство Лебега связывает норму проектора с величиной наилучшего приближения функции многочленами соответствующей степени. Этим, в частности, и обусловлен интерес к изучению минимальных проекторов и к получению оценок их норм. Точные значения минимальной нормы проектора в настоящее время известны только для  $n=1,2,3,7$ .

Также описывается реализованная в рамках данной работы компьютерная программа, которая решает задачу численной оптимизации функции многих переменных для нахождения верхних оценок минимальной нормы проектора. С помощью реализованной программы удалось улучшить верхние оценки минимальных норм проектора при  $n=5,6,10,14,18,20$ .

# 1. Задача линейной интерполяции на n-мерном кубе

Положим  $Q_n := [0..1]^n$ , где  $n \in \mathbb{R}^n$ ,  $Q_n$  — n-мерный куб, множество вершин куба будем обозначать как  $ver(Q_n)$ .  $\Pi_1(\mathbb{R}^n)$  — совокупность многочленов n переменных степени  $\leq 1$ . Пусть  $S$  — невырожденный симплекс в  $\mathbb{R}^n$ , вершины симплекса задаются, как  $x^{(j)} = (x_1^{(j)}, \dots, x_n^{(j)})$ ,  $j = 1, \dots, n$ . Рассмотрим матрицу  $A$ :

$$A := \begin{pmatrix} x_1^{(1)} & \dots & x_n^{(1)} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{(n+1)} & \dots & x_n^{(n+1)} & 1 \end{pmatrix} \quad (1.1)$$

Скажем, что набор точек  $x^{(j)}$  — допустим для интерполяции многочленами из  $\Pi_1(\mathbb{R}^n)$ . Это условие эквивалентно тому, что матрица  $A$  является невырожденной.

$\Delta := \det(A)$ , определитель, который получается из  $\Delta$  заменой j-й строки на строку  $(x_1, \dots, x_n, 1)$ . Многочленный  $\lambda_j(x) := \Delta_j(x)/\Delta$  из  $\Pi_1(\mathbb{R}^n)$  называются базисными многочленами Лагранжа симплекса  $S$  и обладают свойством  $\lambda_j(x^k) = \delta_j^k$ , где  $\delta_j^k$  — символ Кронекера.  $\lambda_j = l_{1j}x_1 + \dots + l_{nj}x_n + l_{n+1j}$ , коэффициенты  $l_{ij}$  составляют столбцы матрицы

$$A^{-1} = \begin{pmatrix} \dots & l_{1,j} & \dots \\ \vdots & \vdots & \vdots \\ \vdots & l_{n,j} & \dots \\ \vdots & l_{n+1,j} & \dots \end{pmatrix} \quad (1.2)$$

Так как  $\lambda_j(x^k) = \delta_j^k$  любой многочлен  $p \in \Pi_1(\mathbb{R}^n)$  удовлетворяет равенству

$$p(x) = \sum_{j=1}^{n+1} p(x^{(j)})\lambda_j(x) \quad (1.3)$$

Так как  $\det(A) \neq 0$ , то для любой  $f \in C(Q_n)$ , где  $C(Q_n)$  — совокупность  $f : Q_n \rightarrow \mathbb{R}$  найдется единственный многочлен  $p \in \Pi_1(\mathbb{R}^n)$  удовлетворяющий условиям:

$$p(x^{(j)}) = f(x^{(j)}) \quad (1.4)$$

## 1.1. Интерполяционный проектор

Введем в рассмотрение оператор  $P : C(Q_n) \rightarrow \Pi_1(\mathbb{R}^n)$ , который далее будем называть интерполяционным проектором. Интерполяционный проектор по системе узлов  $x^{(j)}$  определяется с помощью равенств:

$$Pf(x^{(j)}) = f_j := f(x^{(j)}), j = 1, \dots, n+1 \quad (1.1.1)$$

Эти равенства показывает, что данный оператор является линейным и справедлив следующий аналог интерполяционной формулы Лагранжа:

$$Pf(x^{(j)}) = p(x) = \sum_{j=1}^{n+1} f_j \lambda_j(x) \quad (1.1.2)$$

## 1.2. Норма интерполяционного проектора. Минимальная норма проектора

Обозначим  $\|P\|$  норму оператора  $P$ . Эта величина зависит от узлов  $x^{(j)}$ .

**Лемма.** Для любого интерполяционного проектора  $P : C(Q_n) \rightarrow \Pi_1(\mathbb{R}^n)$  и симплекса  $S$  с вершинами в его узлах имеет место равенство

$$\|P\| = \max_{x \in \text{ver}(Q_n)} \sum_{j=1}^{n+1} |\lambda_j(x)| \quad (1.2.1)$$

Доказательство леммы (см. [1]).

Обозначим через  $\theta_n$  минимальную норму проектора, при условии, что все узлы принадлежат кубу  $Q_n$ :

$$\theta_n := \min_{x^{(j)} \in Q_n} \|P\| \quad (1.2.2)$$

Интерполяционный проектор  $P^*$  с нормой  $\|P^*\| = \theta_n$  назовем минимальным.

Центральной темой в данной работе является вопрос о нахождении минимальной нормы проектора по заданному  $n$ .

Стоит отметить, что минимальная норма проектора достигается на границе куба  $Q_n$ , т.е. в том случае, когда все вершины невырожденного симплекса принадлежат границе  $Q_n$ , доказательство данного факта приводится в книге М.В. Невского (см. [1])

Для  $\theta_n$  справедливы следующие оценки (см. [1]):

$$\frac{1}{4}\sqrt{n} < \theta_n < 3\sqrt{n}$$

$$3 - \frac{4}{n+1} \leq \theta_n$$

Приведем некоторые оценки для  $\theta_n$ :

Таблица 1.2.1

Примеры оценок  $\theta_n$

n	$\theta_n$
1	$\theta_n = 1$
2	$\theta_n = 1.89 \dots$
3	$\theta_n = 2$
4	$2.2 \dots \leq \theta_n \leq 2.33 \dots$
5	$2.33 \dots \leq \theta_n \leq 2.6 \dots$
6	$2.42 \dots \leq \theta_n \leq 3$
7	$\theta_n = 2.5$

Более детально оценки  $\theta_n$  описываются в книге М.В. Невского[1].

## 2. Компьютерная программа для расчета $\theta_n$

В рамках данной ВКР была реализована компьютерная программа для вычисления значений  $\theta_n$ . Задача об отыскании минимальной нормы проектора сводится к задаче отыскания минимума функции многих переменных. Норма проектора вычисляется по формуле (1.2.1), зная это зададим целевую функцию для минимизации.

$$F(A) = \max_{x \in \text{ver}(Q_n)} \sum_{j=1}^{n+1} |\lambda_j(x)| \quad (2.1.1)$$

Где  $A$  — матрица, которая имеет вид (1.1).

Так как задача минимизации функции (2.1.1) является трудоемкой, целесообразным является применение сторонних программных библиотек, поставляющих решения задач линейной алгебры, численных методов и методов оптимизации, существующих на рынке свободно распространяемого программного обеспечения.

Программа реализована на языке программирования C++ с использованием библиотек Dlib, Boost, Eigen, которые предоставляют необходимый функционал для оптимального решения поставленной задачи.

### 2.1. Dlib

Dlib - это универсальная кроссплатформенная программная библиотека, написанная на языке программирования C++. На ее дизайн в значительной степени влияют идеи проектирования по контракту и разработки программного обеспечения на основе компонентов. Таким образом, это, прежде всего, набор независимых программных компонентов. Это программное обеспечение с открытым исходным кодом, выпущенное под лицензией Software Boost.

С момента начала разработки в 2002 году, Dlib вырос до широкого спектра инструментов. По состоянию на 2016 год он содержит программные компоненты для работы с сетями, потоками, графическими пользовательскими интерфейсами, структурами данных, линейной алгеброй, машинным обучением, обработкой изображений, интеллектуальным анализом данных, анализом XML и текста, численной оптимизацией, байесовскими сетями и многими другими задачами. В последние годы большая часть развития была сосредоточена на создании широкого набора инструментов статистического машинного обучения, в 2009 году описание этого набора инструментов Dlib было опубликовано в журнале «Journal of Machine Learning Research» [2]. С тех пор он используется в широком диапазоне областей.

### 2.2. Eigen

Eigen - это высокоуровневая библиотека C++ для линейной алгебры, матричных и векторных операций, геометрических преобразований, численных вычислений и связанных с ними алгоритмов. Eigen является библиотекой с открытым исходным кодом, лицензированной в MPL2, начиная с версии 3.1.1. Более ранние версии были лицензированы под LGPL3 + [3].

Eigen реализуется с использованием метода метапрограммирования шаблонов выражений, то есть она строит деревья выражений во время компиляции и генерирует собственный код для их вычисления. Используя шаблоны выражений и модель затрат операций с плавающей запятой, библиотека выполняет свою собственную развертку и векторизацию цикла [4].

## 2.3. Boost

Boost представляет собой набор библиотек для языка программирования C++, который обеспечивает поддержку задач и структур, таких как линейная алгебра, генерация псевдослучайных чисел, многопоточность, обработка изображений, регулярные выражения и модульное тестирование. Он содержит более восьмидесяти отдельных библиотек.

Большинство библиотек Boost лицензированы по лицензии Boost Software, которая позволяет использовать Boost как с бесплатными, так и с проприетарными проектами программного обеспечения. Многие основатели Boost находятся в комитете по стандартам C++, и несколько библиотек Boost были приняты для включения в стандарт C++ Technical Report 1 и C++ 11 [5].

## 2.4. Реализация программы

Программа реализована в виде консольного приложения. Приложение завит от выше указанных программных библиотек, также использует стандартную библиотеку шаблонов STL языка C++, листинг основной части кода программы см. Приложение 1.

Так как для вычисления нормы проектора необходим перебор по вершинам куба, то асимптотика алгоритма отыскания  $\theta_n$  будет порядка  $O(2^n)$ , что требует оптимального выполнения некоторых операций, для обеспечения наилучшего времени работы программы. Для оптимальной работы с матрицами и операций над ними (например вычисление обратной матрицы) используется библиотека Eigen.

Для оптимизации целевой функции (2.1.1) используются решения, поставляемые библиотекой Dlib. Dlib предоставляет метод минимизации нелинейной функции многих переменных внутри куба  $Q_n$ , с возможностью использования различных стратегий поиска минимума функции. Для решения текущей задачи стратегиями поиска были выбраны алгоритмы: BFGS и L-BFGS.

### 2.4.1. Алгоритм Бroyдена-Флетчера-Голдфарба-Шанно (BFGS)

При численной оптимизации алгоритм Бройдена-Флетчера-Голдфарба-Шанно (BFGS) является итерационным методом решения неограниченных задач нелинейной оптимизации [6].

Метод BFGS относится к квази-ньютоновским методам, классу методов оптимизации восходящего подъема, которые ищут стационарную точку (предпочтительно дважды непрерывно дифференцируемой) функции. Для таких задач необходимым условием оптимальности является то, что градиент равен нулю. Метод Ньютона и методы BFGS не гарантируют сходимости, если функция не имеет квадратичного



разложения Тейлора вблизи оптимума. Тем не менее, BFGS доказал свою хорошую производительность даже для негладкой оптимизации [7].

В квази-Ньютоновских методах матрицу гессиана вторых производных не нужно оценивать напрямую. Вместо этого матрица гессиана аппроксимируется с использованием обновлений, определяемых оценками градиента (или приближительными оценками градиента). Квази-Ньютоновские методы являются обобщениями метода секущих для нахождения корня первой производной для многомерных задач. В многомерных задачах уравнение секущей не определяет уникальное решение, а квази-Ньютоновские методы отличаются тем, как они ограничивают решение. Метод BFGS является одним из самых популярных членов этого класса [8]. Также широко используется L-BFGS, который представляет собой версию BFGS с ограниченной памятью, которая особенно подходит для задач с очень большим количеством переменных (например,  $> 1000$ ). Вариант BFGS-B [9] обрабатывает простые ограничения например кубом.

Алгоритм назван в честь Чарльза Джорджа Бройдена, Роджера Флетчера, Дональда Голдфарба и Дэвида Шанно.

Схема алгоритма:

```

дано  $\epsilon, x_0$ 
инициализировать  $C_0$ 
 $k = 0$ 
while  $\|\nabla f_k\| > \epsilon$ 
    найти направление  $p_k = -C_k \nabla f_k$ 
    вычислить  $x_{k+1} = x_k + \alpha_k p_k$ 
    обозначить  $s_k = x_{k+1} - x_k, y_k = \nabla f_{k+1} - \nabla f_k$ 
    вычислить  $C_{k+1}$ 
     $k = k + 1$ 
end

```

### 3. Результаты

С помощью реализованной программы были получены численные верхние оценки минимальных норм проекторов для  $n = 1, \dots, 20$ . Для  $n = 5, 6, 10, 14, 18, 20$  удалось улучшить оценки приведенные в книге М.В. Невского [1], а также для  $n = 6$  была улучшена оценка указанная в статье [10]. Симплексы заданные списком узлов для улучшенных оценок приводятся в Приложении 2. Ниже приводится таблица с известными на момент написания работы оценками и оценками полученными в рамках данной ВКР.

Таблица 3.1

**Таблица сравнения известных оценок минимальной нормы проектора с полученными в данной работе**

n	Текущие оценки $\theta_n$	Оценки $\theta_n$ полученные в рамках данной работы
1	$\theta_1 = 1$	$\theta_1 = 1$
2	$\theta_2 = 1.89 \dots$	$\theta_2 = 1.89 \dots$
3	$\theta_3 = 2$	$\theta_3 = 2$
4	$2.2 \dots \leq \theta_n \leq 2.3203 \dots$	$2.2 \dots \leq \theta_n \leq 2.3204 \dots$
5	$2.33 \dots \leq \theta_n \leq 2.6 \dots$	$2.33 \dots \leq \theta_n \leq 2.44880 \dots$
6	$2.42 \dots \leq \theta_n \leq 3$	$2.42 \dots \leq \theta_n \leq 2.60004 \dots$
7	$\theta_7 = 2.5$	$\theta_7 = 2.5$
8	$2.5555 \dots \leq \theta_8 \leq 3.1428 \dots$	$2.5555 \dots \leq \theta_8 \leq 3.1428 \dots$
9	$2.6 \leq \theta_9 \leq 3.0000 \dots$	$2.6 \leq \theta_9 \leq 3.0000 \dots$
10	$2.6363 \dots \leq \theta_{10} \leq 3.8000 \dots$	$2.6363 \dots \leq \theta_{10} \leq 3.5186 \dots$
11	$2.6666 \dots \leq \theta_{11} \leq 3.0000 \dots$	$2.6666 \dots \leq \theta_{11} \leq 3.0000 \dots$
12	$2.6923 \dots \leq \theta_{12} \leq 3.4000 \dots$	$2.6923 \dots \leq \theta_{12} \leq 3.4000 \dots$
13	$2.7142 \dots \leq \theta_{13} \leq 3.7692 \dots$	$2.7142 \dots \leq \theta_{13} \leq 3.7692 \dots$
14	$2.7333 \dots \leq \theta_{14} \leq 4.1999 \dots$	$2.7333 \dots \leq \theta_{14} \leq 4.0156 \dots$
15	$2.75 \dots \leq \theta_{15} \leq 3.5 \dots$	$2.75 \dots \leq \theta_{15} \leq 3.5 \dots$
16	$2.7647 \dots \leq \theta_{16} \leq 4.2000 \dots$	$2.7647 \dots \leq \theta_{16} \leq 4.2000 \dots$
17	$2.7777 \dots \leq \theta_{17} \leq 4.0882 \dots$	$2.7777 \dots \leq \theta_{17} \leq 4.0882 \dots$
18	$2.7894 \dots \leq \theta_{18} \leq 5.5882 \dots$	$2.7894 \dots \leq \theta_{18} \leq 5.14006 \dots$
19	$2.8 \leq \theta_{19} \leq 4.0000 \dots$	$2.8 \leq \theta_{19} \leq 4.0000 \dots$
20	$2.8095 \dots \leq \theta_{20} \leq 4.7241 \dots$	$2.8095 \dots \leq \theta_{20} \leq 4.68879 \dots$

## 4. Заключение

В данной работе опробована возможность применения программных библиотек, таких как `eigen`, `Dlib`, `boost` для вычисления оценок минимальной нормы проектора. Использован алгоритм Бroyдена-Флетчера-Голдфарба-Шанно (BFGS) для минимизации величины  $\theta_n$  в  $Q_n$ . В результате были найдены новые верхние оценки для величин  $\theta_5$ ,  $\theta_6$ ,  $\theta_{10}$ ,  $\theta_{14}$ ,  $\theta_{18}$ ,  $\theta_{20}$ .

## Список литературы

- [1] Невский М.В., Геометрические оценки в полиномиальной интерполяции, Ярославль: ЯрГУ
- [2] King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. J. Mach. Learn. Res. 10 (Jul): 1755–1758
- [3] Eigen License. tuxfamily.org. Retrieved 16 Jan 2016
- [4] Guennebaud, Gaël (2013). Eigen: A C++ linear algebra library. Eurographics/CGLibs.
- [5] Library Technical Report. JTC1/SC22/WG21 - The C++ Standards Committee. 2 July 2003. Retrieved 1 February 2012
- [6] Fletcher, Roger (1987), Practical methods of optimization (2nd ed.), New York: John Wiley & Sons, ISBN 978-0-471-91547-8
- [7] Lewis, Adrian S.; Overton, Michael (2009), Nonsmooth optimization via BFGS
- [8] Nocedal, Jorge; Wright, Stephen J. (2006), Numerical Optimization (2nd ed.), Berlin, New York: Springer-Verlag, ISBN 978-0-387-30303-1
- [9] Byrd, Richard H.; Lu, Peihuang; Nocedal, Jorge; Zhu, Ciyu (1995), A Limited Memory Algorithm for Bound Constrained Optimization, SIAM Journal on Scientific Computing
- [10] Кудрявцев, И. С., Озерова Е. А., Ухалов А.Ю. Новые оценки для норм минимальных проекторов, 2017

## Листинг программы

```
1  #include <dlib/optimization.h>
2  #include <dlib/global_optimization.h>
3  #include <iostream>
4  #include <vector>
5  #include <iomanip>
6  #include <eigen3/Eigen/Dense>
7  #include <eigen3/Eigen/LU>
8  #include <boost/random.hpp>
9  #include <boost/random/normal_distribution.hpp>
10 #include <random>
11 #include <fstream>
12 #include <algorithm>
13 #include <string>
14 #include <iomanip>
15 #include <fstream>
16 #include <string>
17
18 using namespace dlib;
19 using namespace std;
20 using namespace Eigen;
21 using namespace std;
22 using namespace boost;
23
24 void LogMSG(const std::string &s)
25 {
26     return;
27     fstream out;
28     out.open("/Users/artem/diplomlogs/diplomlogs.txt", ios::out);
29     cout << s << endl;
30     out.close();
31 }
32
33
34 int SYMPLEX_DIM;
35 typedef matrix<double,0,1> column_vector;
36
37 //Генератор случайных чисел
38 double sample(double dummy)
39 {
40     using namespace std::chrono;
41     std::default_random_engine engine(
42         system_clock::to_time_t(system_clock::now()));
```

```

43         std::uniform_real_distribution<> distr(0, 1);
44         return distr(engine);
45     }
46
47     //Получение случайной матрицы
48     void getRandomMatrix( MatrixXd& a, int n)
49     {
50         using namespace std::chrono;
51         std::default_random_engine engine(
52             system_clock::to_time_t( system_clock::now()) );
53         std::uniform_real_distribution<> distr(0, 1);
54         auto gen_number = [&engine, &distr] () { return distr(engine); };
55         for( int i = 0; i <= n; ++i )
56         {
57             for( int j = 0; j < n; ++j )
58             {
59                 a(i, j) = gen_number();
60             }
61         }
62
63         for( int i = 0; i <= n; ++i )
64         {
65             a(i, n) = 1;
66         }
67     }
68 }
69
70
71 //Вычисление нормы проектора
72 double getNorm( MatrixXd& a)
73 {
74     LogMSG("getNorm/ start");
75     int dim = SYMPLEX_DIM + 1;
76     LogMSG("getNorm/ MatrixXd::inverse");
77     MatrixXd invMatrix = a.inverse();
78     //cout << a << endl;
79     //cout << "detA = " << a.determinant() << endl;
80
81     int pow2 = 1 << (dim-1);
82     double norm = 0;
83     LogMSG( "getNorm/ start search norm" );
84     LogMSG( "getNorm/ number of cube vertex = " + std::to_string( pow2 ) );
85     for(int i = 0; i < pow2; ++i)
86     {
87         std::vector<double> x(dim, 0);
88         x[dim-1]=1;

```

```

89         int ind = 0;
90         int d = i;
91         while( d )
92         {
93             x[ind] = d % 2;
94             ind++;
95             d /= 2;
96         }
97         double sum = 0;
98         for( int k = 0; k < dim; ++k )
99         {
100             double cur_sum = 0;
101             for( int j = 0; j < dim; ++j )
102             {
103                 cur_sum += invMatrix(j,k)*x[j];
104             }
105             sum += std::abs( cur_sum );
106         }
107
108         norm = std::max( sum, norm );
109     }
110     LogMSG( "getNorm/ return norm" );
111     return norm;
112 }
113
114 //Составить матрицу по вектору начального приближения
115 MatrixXd getMatixByVect( const column_vector& m )
116 {
117     LogMSG("getMatixByVect/ start");
118     MatrixXd A = MatrixXd::Zero( SYMPLEX_DIM+1,SYMPLEX_DIM+1 );
119     for(int i = 0; i <= SYMPLEX_DIM; ++i)
120     {
121         for( int j = 0; j < SYMPLEX_DIM; ++j )
122         {
123             A(i, j) = m( SYMPLEX_DIM*i + j );
124         }
125         A(i, SYMPLEX_DIM) = 1;
126     }
127     return A;
128 }
129
130 //Составить матрицу по вектору
131 column_vector getVectorByMatrix( MatrixXd& A )
132 {
133     column_vector vect( SYMPLEX_DIM*( SYMPLEX_DIM+1 ) );
134     for( int i = 0; i <= SYMPLEX_DIM; ++i )

```

```

135         {
136             for( int j = 0; j < SYMPLEX_DIM; ++j )
137             {
138                 vect(SYMPLEX_DIM*i + j) = A(i, j);
139             }
140         }
141     }
142     return vect;
143 }
144
145 //Целевая функция для оптимизации
146 double func( const column_vector& m )
147 {
148     LogMSG( "func/ start" );
149     LogMSG( "func/ getMatrixByVect" );
150     MatrixXd A =getMatixByVect( m );
151     LogMSG( "func/ getNorm" );
152     return getNorm( A );
153 }
154
155 //Метод для посчета минимальной нормы на
156 //случайно сгенерированном наборе симплексов
157 double MonteCarlo( int dimension )
158 {
159     double minimalNorm = 1e9+7;
160     for( int i = 0; i < 1e10; ++i )
161     {
162         MatrixXd a = MatrixXd::Zero( dimension+1,dimension+1 );
163         getRandomMatrix( a, dimension );
164         minimalNorm = std::min(minimalNorm, getNorm( a ));
165     }
166     return minimalNorm;
167 }
168
169 void solve( int n, column_vector& starting_point )
170 {
171     try
172     {
173         //n-мерный случай
174         SYMPLEX_DIM = n;
175         cout << starting_point << endl;
176         LogMSG( "main/ find_min_box_constrained" );
177         cout << endl << find_min_box_constrained(
178             bfgs_search_strategy(),
179             objective_delta_stop_strategy( 1e-15 ).be_verbose(),
180             func,

```



```

181         derivative( func,1e-15 ),
182         starting_point, 0, 1 );
183         cout << "-----"<< endl;
184
185         cout << "main/getMatixByVect" << endl;
186         MatrixXd A1 = getMatixByVect( starting_point );
187         cout << A1 << endl;
188         cout << "main/getNorm" << endl;
189         cout << "theta(" << n << ") = " << setprecision(11)
190         << getNorm(A1) << endl << "-----\n";
191     }
192     catch(...){
193         cout << "Процесс вычисления прерван" << endl;
194     }
195 }
196
197 void test_solve()
198 {
199     return;
200 }
201
202 void ReadDataFromConsole( int& n, column_vector& r_data )
203 {
204     std::cout << "ReadDataFromConsole" << endl;
205     std::cin >> n;
206     r_data = column_vector( n*(n+1) );
207     for( int i = 0; i <= n; ++i )
208     {
209         for( int j = 0; j < n; ++j )
210         {
211             double coord;
212             std::cin >> coord;
213             r_data(n*i + j) = coord;
214         }
215     }
216
217     return;
218 }
219
220 void ReadDataFromFile( const std::string& file_name,
221 int &n,
222 column_vector& r_data )
223 {
224     std::cout << "ReadDataFromFile" << endl;
225     std::fstream input_file( file_name, ios_base::in );
226     input_file >> n;

```

```

227         r_data = column_vector( n*(n+1) );
228         for( int i = 0; i <= n; ++i )
229         {
230             for( int j = 0; j < n; ++j )
231             {
232                 double coord;
233                 input_file >> coord;
234                 r_data( n*i + j ) = coord;
235             }
236         }
237         return;
238     }
239
240     void solve_with_random_start_point( int n )
241     {
242         MatrixXd B = MatrixXd::Zero( n+1, n+1 );
243         getRandomMatrix( B, n );
244         column_vector starting_point = getVectorByMatrix( B );
245         solve( n, starting_point );
246     }
247
248     int main() try
249     {
250
251         int TYPE_INPUT = 0;
252         // type input. Default input from console type = 0, from file type = 1;
253         std::string NAME_INPUT_FILE;
254         std::cin >> TYPE_INPUT;
255         int n;
256         column_vector starting_point;
257
258         switch( TYPE_INPUT )
259         {
260             case 0:
261                 ReadDataFromConsole( n, starting_point );
262                 break;
263             case 1:
264                 std::cin >> NAME_INPUT_FILE;
265                 ReadDataFromFile( NAME_INPUT_FILE, n, starting_point );
266                 break;
267             default:
268                 std::cout << "Error input, check type input!";
269                 return 0;
270
271         }
272

```

```

273 //Примеры решения и стартовые приближения для n = 1..6
274
275 solve( n, starting_point );
276
277 column_vector starting_point2 = {0.45, 0.45,
278     0.8, 0.69,
279     0.3, 0.9};
280 solve(2, starting_point2);
281 column_vector starting_point3 = {0.547079, 0.0126172, 0.0867465,
282     0.490059, 0.478761, 0.635784,
283     0.93395, 0.514259, 0.193468,
284     0.217008, 0.73415, 0.467616};
285 solve(3, starting_point3);
286 column_vector starting_point4 = {1, 0.292919, 0, 0,
287     1, 1, 0.707131, 1,
288     0.499911, 0, 1, 0.500076,
289     0, 0.292961, 0, 1,
290     0, 1, 0.70711, 0};
291 solve(4, starting_point4);
292
293 column_vector starting_point5 = {1, 1, 1, 0.282777, 0.662421,
294     0, 0.282777, 0.662421, 0, 1,
295     0.337578, 0, 1, 1, 0.282777,
296     0, 1, 0.282777, 0.662421, 0,
297     0.717222, 0.662421, 0, 1, 1,
298     1, 0, 0, 0, 0};
299 solve(5, starting_point5);
300 column_vector starting_point6 = {1, 0, 1, 0.091, 0, 0.4999,
301     0, 1, 1, 1, 0, 0,
302     0.5, 1, 1, 0.091, 1, 1,
303     0.9106, 0.0896, 0.0895, 1, 0.91053, 0.9106,
304     0, 0, 0.5, 0.09115, 1, 0,
305     0, 0.49999, 0, 0.09104, 0, 1,
306     1, 1, 0, 0.09105, 0.5001, 0};
307 solve(6, starting_point6);
308
309
310 }
311 catch (std::exception& e)
312 {
313     cout << e.what() << endl;
314 }
315 catch(...)
316 {
317     cout << "unknown exception"<<endl;
318 }

```

### Симплексы на которых достигаются улучшенные оценки

Далее для  $n$ , для которых улучшены верхние оценки минимальной нормы проектора  $\theta_n$  приводятся симплексы, на которых достигается данная оценка, заданные в виде списков узлов:

#### $n=5, \theta_5 = 2.4488029336$

(1, 0.99999992367, 0.99999999995, 0.28277765266, 0.66242093126), (1.9985588096e-07, 0.28277715142, 0.66242088676, 0, 0.99999999585), (0.33757909432, 1.4555570776e-12, 0.99999998881, 0.99999999384, 0.28277748041), (0, 1, 0.28277711932, 0.66242011715, 4.1747646609e-09), (0.71722297192, 0.66242085342, 0, 0.99999999527, 0.99999982627), (0.99999999766, 5.8842807186e-08, 7.8689584018e-08, 0, 1.231226984e-07)

#### $n=6, \theta_6 = 2.6000414905$

(0.99996732848, 4.404950382e-08, 0.9999803767, 0.091043705497, 1.5080287403e-05, 0.49999078839), (1.5440523041e-05, 0.99998242532, 0.99998536235, 1, 4.2979109321e-06, 5.0154875895e-06), (0.49997325566, 0.99999995279, 1, 0.091030248441, 0.99999971727, 0.99999485857), (0.91054529517, 0.089496154482, 0.089462458908, 1, 0.91053597023, 0.91054722512), (9.8896459314e-09, 5.0622915594e-06, 0.49998966987, 0.09106291499, 0.99999877544, 6.9374976785e-06), (5.5000135932e-06, 0.50000006185, 1.5126932468e-07, 0.091050184422, 1.6459243787e-05, 0.99999751282), (0.99998790427, 0.99999999534, 1.0226889906e-05, 0.091028761409, 0.50001817275, 3.3747301998e-07)

#### $n=10, \theta_{10} = 3.5186849383$

(0.000343761, 0.283091, 0.00152227, 0, 0.993474, 0.97941, 0.962042, 0.989746, 0.00673603, 0.00435528), (0.299096, 0.00174845, 1, 1, 0.00590979, 0.980404, 0.999164, 0.00352637, 0.000545441, 5.34969e-05), (0.00125624, 0.999996, 0, 0.873951, 0.0999516, 0.900134, 0.903515, 0.139397, 0.998182, 0.99672), (0, 0.939365, 0.900971, 0.997554, 0.999772, 0.0186347, 1.57508e-07, 0.971075, 0.136581, 0.143967), (0.994957, 0.0107344, 0.0496297, 0.999201, 0.999864, 0.00234734, 0.932301, 0.118349, 0.00217507, 1), (0.998827, 0.998919, 0.897163, 0.00384381, 0.0246995, 0.977905, 0.0455373, 0.938394, 0.00713544, 1), (0.996961, 0.966364, 0.908957, 0.0394159, 0.945958, 0.0765708, 0.9765, 0.0194037, 1, 0.0054306), (0.998516, 0.00311883, 0.0639222, 0.990097, 0.0869709, 0.915778, 0.00260304, 0.995842, 0.999934, 0.00539863), (0.0345267, 0.00683825, 0.994703, 0.0919666, 0, 0, 1, 1, 0.99426, 0.99797), (0.0416991, 0.00155994, 0.990757, 0.0810886, 0.989043, 0.997378, 7.50152e-05, 6.53832e-05, 0.982976, 1), (0.000212241, 0.318944, 0.000215123, 0.000392483, 0.0358272, 0, 2.01195e-05, 0.0308336, 0.0691685, 0.0227924)

#### $n=14, \theta_{14} = 4.0156094236$

(5.11975e-05, 0.0887167, 0.999972, 0.998949, 0.00174637, 0.000222371, 1, 0.00557882, 0.999992, 1.7163e-05, 0.000140542, 1, 3.02053e-05, 0.999274), (0.119939, 0.118546, 0.998817, 0.00242529, 0.999721, 0.000253437, 0.000184339, 1, 0.999745, 0.00113355, 1, 0.000454303, 0.00135587, 0.999972), (0.991502, 0.999999, 0.000333431, 0.999976, 0.0428634, 0.00192493, 0.033751, 1, 0.969658, 0.00212882, 0.999922, 0.969954, 0.954448, 0.0204009), (0.00090201, 1, 0.0471678, 0.000420019, 3.94574e-05, 0.999958, 0.975597, 0.975096, 0.999999, 0.998514, 0.0235392, 0.0115776, 0.0367109, 0.999998), (0.999866, 0.00070839, 0.989752, 0.961199, 0.0178236, 0.999985, 0.00012808, 0.963871, 0.0371852,

0.999959, 0.0162689, 0.00883974, 0.0510681, 0.993505), (0.000109535, 0.000209244, 0.0077589, 0.999939, 1, 0.000255139, 0.989474, 0.999538, 1.2471e-05, 0.923057, 0.044588, 0.0592835, 0.992639, 0.947896), (0.997899, 0.999684, 0.0456788, 0.974036, 0.999935, 0.954572, 0.999994, 0.0427239, 1.16546e-05, 0.00668326, 1, 0.0244723, 0.000320798, 0.968688) (0.00395713, 0.00715372, 0.936955, 0.0494695, 0.0390959, 1, 0.995082, 0.987239, 0.000814255, 7.60102e-06, 0.999599, 0.998657, 0.912031, 0.0037748), (1, 0.00139645, 0.982596, 0.0212883, 0.0334078, 0.053158, 0.968014, 0.000506459, 0.999927, 0.999997, 0.984547, 0.000470046, 1, 0.023132), (0.000631627, 1, 0.999879, 0.000102598, 0.00201826, 0.0444134, 0.000420926, 0.00270765, 0.0425575, 0.999999, 0.977588, 0.976265, 0.999327, 0.999794), (3.89041e-05, 1, 1, 0.980854, 0.97631, 0.999992, 0.0137901, 0.0255375, 0.973826, 0.0531728, 0.000364088, 0.00194644, 0.999845, 0.028632), (0.999962, 0.000107765, 0.0328616, 2.48067e-06, 0.983354, 0.99966, 0.000144143, 0.0307265, 0.99075, 0.0432125, 3.73111e-05, 0.957377, 1, 0.998587), (0.998942, 0.999302, 0.972642, 0.0199797, 1, 0.000174567, 0.980111, 0.999441, 0.00378433, 0.948393, 0.040194, 0.99999, 0.00144084, 1.6943e-05), (0.00454873, 0.00233364, 0.00032794, 1, 0.999156, 0.926904, 0.0564063, 0.0400909, 0.953912, 0.999986, 0.999974, 0.999884, 2.75091e-05, 3.94854e-05), (0.0871276, 0.000678347, 0, 1.34914e-06, 3.40856e-06, 0.000356143, 3.11508e-05, 0.000151653, 1.64089e-05, 0.000534129, 3.67674e-05, 0, 9.1822e-06, 2.34297e-06)

**n=18,  $\theta_{18}$  =5.140061**

(0, 6.7546e-09, 1, 0, 0.868399, 0.0497722, 0, 0.868399, 0.0497723, 1, 1, 1, 1, 0.0953266, 0.0953265, 0.0953264, 1, 1), (2.28686e-08, 0, 1, 0.868399, 0, 0.0497723, 0.868399, 0, 0.0497722, 1, 1, 1, 0.0953265, 1, 0.0953266, 1, 0.0953265, 1), (1, 1, 1, 0.0497722, 0.0497722, 1, 0.0497723, 0.0497722, 1, 0, 8.30747e-09, 8.30747e-09, 0.956133, 0.956133, 8.30748e-09, 0, 8.30748e-09, 0.956133), (0, 0.868399, 0.0497722, 0, 6.7546e-09, 1, 0, 0.868399, 0.0497723, 0.0953265, 1, 1, 1, 1, 1, 0.0953265, 0.0953266), (0.868399, 0, 0.0497723, 2.28686e-08, 0, 1, 0.868399, 0, 0.0497722, 1, 0.0953265, 1, 1, 1, 1, 0.0953265, 1, 0.0953265), (0.0497722, 0.0497722, 1, 1, 1, 1, 0.0497722, 0.0497722, 1, 0, 8.30749e-09, 0.956133, 0, 8.30748e-09, 8.30748e-09, 0.956133, 0.956133, 8.30748e-09), (0, 0.868399, 0.0497723, 0, 0.868399, 0.0497722, 0, 6.7546e-09, 1, 1, 0.0953265, 0.0953265, 0.0953265, 1, 1, 1, 1, 1), (0.868399, 0, 0.0497723, 0.868399, 0, 0.0497722, 2.28686e-08, 0, 1, 0.0953265, 1, 0.0953265, 1, 0.0953266, 1, 1, 1, 1), (0.0497723, 0.0497723, 1, 0.0497723, 0.0497722, 1, 1, 1, 1, 0.956133, 0.956133, 8.30746e-09, 0, 8.30747e-09, 0.956133, 0, 8.30747e-09, 8.30747e-09), (1, 1, 0, 0.0953265, 1, 0, 1, 0.0953265, 0.956133, 0.932512, 0.932512, 0.932512, 0.932512, 0, 0, 0.932512, 0, 0, 1), (1, 1, 9.08861e-09, 1, 0.0953265, 9.08861e-09, 0.0953265, 1, 0.956133, 0.932512, 0.932512, 0.932512, 0, 0.932512, 6.56782e-09, 0, 0.932512, 6.56781e-09), (1, 1, 9.08859e-09, 1, 1, 0.956133, 0.0953265, 0.0953265, 9.08859e-09, 0.932512, 0.932512, 0.932512, 0, 6.5678e-09, 0.932512, 0, 6.56782e-09, 0.932512), (1, 0.0953265, 0.956133, 1, 1, 0, 0.0953264, 1, 0, 0.932512, 0, 0, 0.932512, 0.932512, 0.932512, 0.932512, 0, 0), (0.0953265, 1, 0.956133, 1, 1, 9.0886e-09, 1, 0.0953265, 9.08859e-09, 0, 0.932512, 6.56781e-09, 0.932512, 0.932512, 0.932512, 0, 0.932512, 6.5678e-09), (0.0953266, 0.0953265, 9.0886e-09, 1, 1, 9.08859e-09, 1, 1, 0.956133, 0, 6.56782e-09, 0.932512, 0.932512, 0.932512, 0.932512, 0, 6.56782e-09, 0.932512), (0.0953265, 1, 0, 1, 0.0953265, 0.956133, 1, 1, 0, 0.932512, 0, 0, 0.932512, 0, 0, 0.932512, 0.932512, 0.932512), (1, 0.0953264, 9.0886e-09, 0.0953265, 1, 0.956133, 1, 1, 9.08859e-09, 0, 0.932512, 6.56781e-09, 0, 0.932512, 6.56782e-09, 0.932512, 0.932512,

0.932512), (1, 1, 0.956133, 0.0953265, 0.0953265, 9.08859e-09, 1, 1, 9.08859e-09, 0,  
6.56781e-09, 0.932512, 0, 6.56781e-09, 0.932512, 0.932512, 0.932512, 0.932512),  
(0.140881, 0.140881, 0, 0.140881, 0.140881, 0, 0.140881, 0.140881, 0, 2.3886e-08, 0, 0,  
2.3886e-08, 0, 0, 2.3886e-08, 0, 0)

**n=20,  $\theta_{20}$  =4.688796**

(0.913198, 0, 0, 0, 0.00390006, 0.00390019, 0.00390012, 0.00390016, 0.947348, 0.947348,  
0.947348, 0.947348, 0.947348, 0.947348, 0.947348, 0.947348, 0.947348, 0.947348,  
0.947348, 0.947348), (0, 0.913198, 0, 0, 0.947348, 0.947348, 0.947348, 0.947348,  
0.0039002, 0.00390012, 0.00390015, 0.00390015, 0.947348, 0.947348, 0.947348, 0.947348,  
0.947348, 0.947348, 0.947348, 0.947348), (0, 0, 0.913198, 0, 0.947348, 0.947348, 0.947348,  
0.947348, 0.947348, 0.947348, 0.947348, 0.947348, 0.0039001, 0.00390009, 0.00390011,  
0.0039001, 0.947348, 0.947348, 0.947348, 0.947348), (0, 0, 0, 0.913198, 0.947348,  
0.947348, 0.947349, 0.947348, 0.947348, 0.947348, 0.947348, 0.947348, 0.947348, 0.947348,  
0.947348, 0.947348, 0.947348, 0.00390015, 0.00390011, 0.00390015, 0.00390016),  
(0.0039001, 0.947348, 0.947348, 0.947348, 0.885422, 0.0385256, 0.0385256, 0.0385257, 1,  
1, 0, 0, 1, 1, 0, 1.99812e-08, 0, 1, 1, 2.03086e-08), (0.00390011, 0.947348, 0.947348,  
0.947348, 0.0385256, 0.885422, 0.0385256, 0.0385256, 0.0385256, 1, 0, 1, 1.69286e-08, 0, 0, 1, 1, 1,  
1.82243e-08, 0), (0.0039001, 0.947348, 0.947348, 0.947348, 0.0385256, 0.0385256,  
0.885422, 0.0385257, 1.69286e-08, 1, 0, 1, 2.08925e-08, 1, 0, 1, 1, 1.58687e-08, 0, 1),  
(0.00390025, 0.947348, 0.947348, 0.947348, 0.0385257, 0.0385257, 0.0385256, 0.885422, 0,  
0, 1, 1, 1, 0, 1, 0, 1.78263e-08, 0, 1, 1), (0.947348, 0.0039001, 0.947348, 0.947348, 0, 0, 1,  
1, 0.885422, 0.0385257, 0.0385256, 0.0385256, 1, 1.78636e-08, 0, 1, 1, 1.59377e-08, 1,  
1.91314e-08), (0.947348, 0.00390016, 0.947348, 0.947348, 1, 1, 0, 0, 0.0385256, 0.885422,  
0.0385257, 0.0385256, 1, 0, 1, 0, 1, 0, 1.92302e-08, 1), (0.947348, 0.0039001, 0.947348,  
0.947348, 1, 0, 2.05184e-08, 1, 0.0385257, 0.0385257, 0.885422, 0.0385256, 2.37411e-08, 1,  
0, 1, 0, 1, 0, 1), (0.947348, 0.00390012, 0.947348, 0.947348, 2.05184e-08, 1, 1, 0,  
0.0385256, 0.0385257, 0.0385256, 0.885422, 0, 1, 1, 0, 1.80565e-08, 1, 1, 0), (0.947348,  
0.947348, 0.00390009, 0.947348, 1, 1.78636e-08, 1, 0, 0, 0, 1, 1, 0.885422, 0.0385256,  
0.0385256, 0.0385257, 1, 1, 1.91476e-08, 0), (0.947348, 0.947348, 0.0039001, 0.947348,  
2.37411e-08, 1, 0, 1, 0, 1, 1, 2.08925e-08, 0.0385256, 0.885422, 0.0385256, 0.0385257, 1, 0,  
1, 0), (0.947348, 0.947348, 0.00390011, 0.947348, 0, 0, 1, 1, 1, 1, 0, 0, 0.0385256,  
0.0385256, 0.885422, 0.0385257, 0, 1, 0, 1), (0.947348, 0.947348, 0.00390014, 0.947348, 1,  
1, 0, 0, 1, 0, 1.99812e-08, 1, 0.0385257, 0.0385256, 0.0385257, 0.885422, 1.87415e-08, 0, 1,  
1), (0.947348, 0.947348, 0.947348, 0.00390021, 1, 1.81351e-08, 0, 1, 1, 0, 1.9251e-08, 1, 0,  
1, 1, 0, 0.885422, 0.0385255, 0.0385256, 0.0385256), (0.947348, 0.947348, 0.947348,  
0.00390019, 0, 1, 1.69101e-08, 1, 1.58538e-08, 1, 0, 1, 1, 1.6657e-08, 0, 1, 0.0385256,  
0.885422, 0.0385256, 0.0385257), (0.947348, 0.947348, 0.947348, 0.00390025, 1,  
1.6934e-08, 1, 0, 0, 1, 1, 1.68607e-08, 0, 0, 1, 1, 0.0385257, 0.0385257, 0.885422,  
0.0385256), (0.947348, 0.947348, 0.947348, 0.00390014, 0, 1, 1, 2.03644e-08, 1,  
2.03224e-08, 1, 0, 1, 1, 0, 2.13658e-08, 0.0385256, 0.0385257, 0.0385256, 0.885422), (0, 0,  
0, 0, 0, 3.38164e-08, 3.34541e-08, 3.1704e-08, 3.33852e-08, 0, 0, 3.91276e-08,  
3.33102e-08, 0, 3.50991e-08, 0, 2.97189e-08, 0, 3.56742e-08)