САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОЙ ТЕОРИИ МОДЕЛИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ

Колосов Игорь Андреевич

Выпускная квалификационная работа специалиста

ТЕСТИРОВАНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ПОСТРОЕНИЯ ВЫПУКЛОЙ ОБОЛОЧКИ

Направление СМ.116.2006

Прикладная математика и информатика

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

Санкт-Петербург
2016

Оглавление

Стр.

1. Введение

Во многих задачах вычислительной геометрии и на некоторых этапах реализации алгоритмов недифференцируемой оптимизации возникает задача по выявлению крайних точек выпуклой оболочки конечного числа точек. Например, кодифференциал функции f суммы от кодифференцируемых функций f_k представляет собой сумму (по Минковскому) кодифференциалов df_k, т.е. сумму выпуклых многогранников. Естественно, что количество элементов в кодифференциале df функции f возрастает экспоненциально. В таком случае для эффективной работы численных алгоритмов негладкого анализа необходимо производить «чистку», т.е. отбросить не информативные точки и оставить только крайние.

Из известных методов решающих рассматриваемую задачу является алгоритм QuickHull [3] — построение выпуклой оболочки для заданного набора точек. Однако, его применение сопряжено со многими ограничениями. Разработчики пакета MatLab обращают внимание на то, что данная процедура дает гарантированный результат для набора точек из пространства размерностей не более 6-9.

В работе рассматривается программная реализация по выявлению крайних точек у множества, которое получается в результате сложения многогранника и отрезка. Теоретической базой данного алгоритма являются результаты работы [1]. Предполагается, что многогранник задан крайними точками, а также известны грани и их нормали. В частности, данную информацию о многограннике можно получить в результате применения алгоритма QuickHull.

2. Постановка задачи

Напомним определения выпуклой оболочки и крайней точки.

Определение 1. *Выпуклой оболочкой* $\operatorname{conv} A$ множества A называется наименьшее выпуклое множество, содержащее A.

Определение 2. Точка p выпуклого множества $S \subset \mathbb{R}^n$ называется *крайней*, если не существует пары точек $a, b \in S$ таких, что p лежит в относительной внутренности отрезка $\operatorname{conv}\{a,b\}$.

Рассмотрим множество A, состоящее из трех точек на плоскости.

$$A = \left\{ \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \begin{pmatrix} -2 \\ 6 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \end{pmatrix} \right\}.$$

На рисунке 2.1 треугольник со своей внутренностью соответствует множеству $\operatorname{conv} A$, а вершины являются крайними точками.

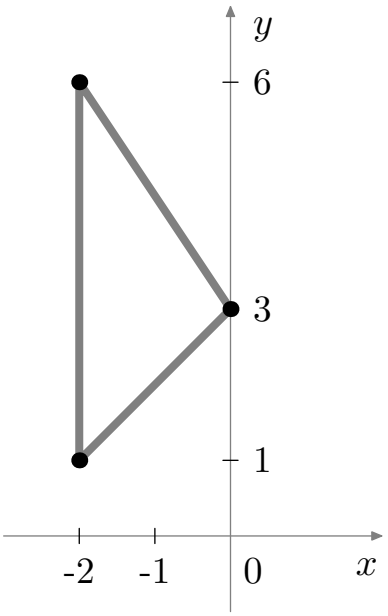
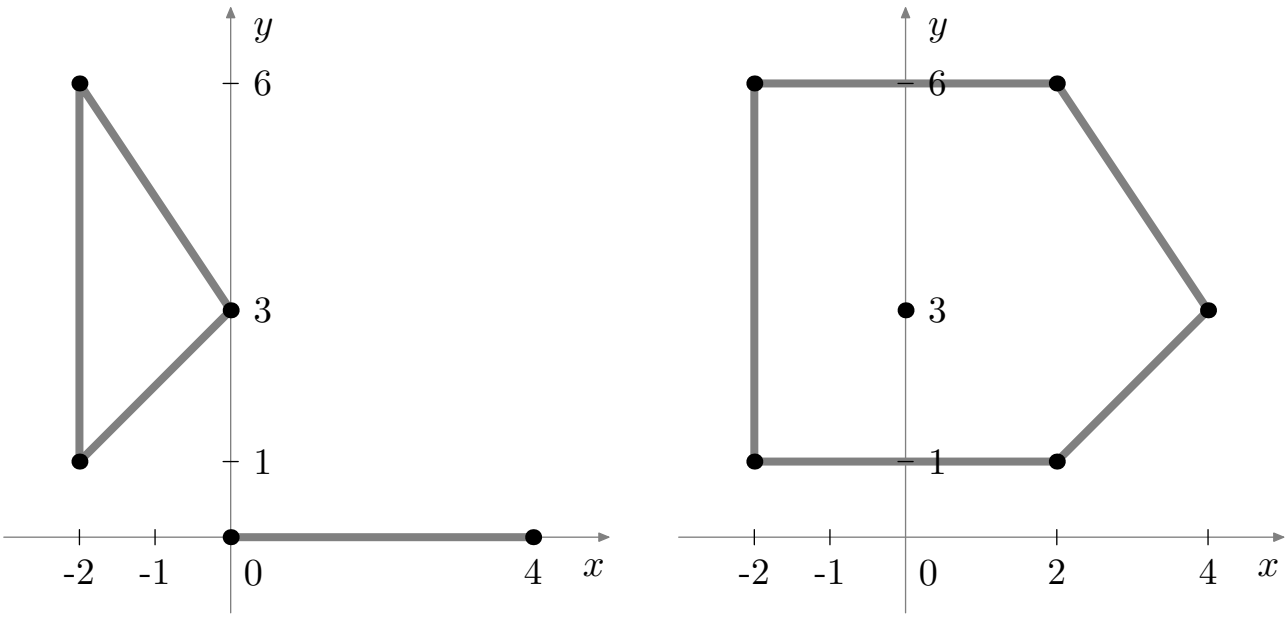


Рисунок 2.1 — Выпуклая оболочка и крайние точки множества A

Изобразим (см. рис. 2.2) сумму многогранника $\operatorname{conv} A$ и отрезка T, с вершинами в точках $(0,0)^T$ и $(4,0)^T$.

Видно, что точка $(0,3)^T$ не является крайней для нового многоугольника (рис. 2.2, б)). Она не является существенной для задания результата суммы. Точку $(0,3)^T$ следует идентифицировать и удалить.



а) треугольник $\operatorname{conv} A$ и отрезок T б) $\operatorname{conv} A + T$

Рисунок 2.2 — Сумма многогранника и отрезка

Ставится задача проверки крайности точек, которые получаются в результате суммы произвольных многогранника и отрезка в конечномерном пространстве.

3. Вспомагательные сведения

Напомним одно из определений выпуклого конуса.

Определение 3. Пусть подмножество C является частью пространства $\mathbb{R}^n$. Множество

$$K(C) = \{\lambda x \mid \lambda > 0,\, x \in C\}.$$

называется *конической оболочкой* подмножества C.

Если множество C — выпуклое, то $K(C)$ является выпуклым конусом.

Заметим, что $K(C)$ не обязательно содержит начало координат. Тогда замыкание $K(C)$ задается следующим образом:

$$\operatorname{cone} C := \overline{K(C)} = 0_n \bigcup K(C).$$

Полученный оператор cone совпадает по определению с одноименным оператором из [2]. Определим несколько необходимых нам множеств.

Пусть задано множество точек $A = \{a_0, a_1, \ldots, a_m\}$, где все $a_i \in \mathbb{R}^n$ являются крайними для многогранника $\operatorname{conv} A$. Не умаляя общности предположим, что размерность аффинной оболочки множества $\operatorname{conv} A$ равняется n, а также все точки a_i отличны друг от друга.

Зададим следующие вспомогательные множества. Вычтем a_i из остальных точек A и зададим матрицу с размерностью $n \times m$

$$A(a_i) := (a_0 - a_i, a_1 - a_i, \ldots, a_{i-1} - a_i, a_{i+1} - a_i, \ldots, a_m - a_i).$$

Заметим, что в матрице $A(a_i)$ отсутствует нулевой столбец. В силу определения множества A, нулевой столбец в $A(a_i)$ может быть получен только как разность $a_i - a_i$.

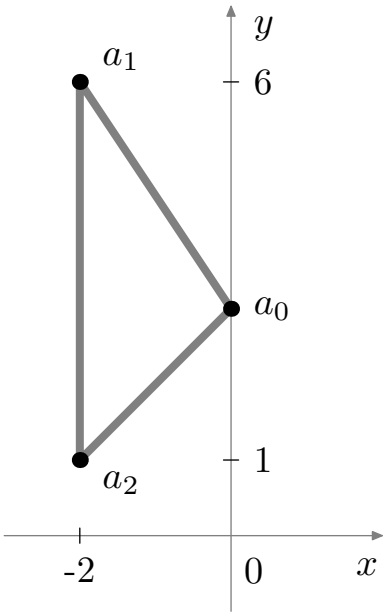
Обозначим выпуклый многогранный конус, натянутый на множество $\{\operatorname{conv} A - a_i\} \setminus 0_n$, следующим образом

$$K(a_i) := K(\operatorname{conv} \bigcup_{j \in 1:m} A(a_i)_j),$$

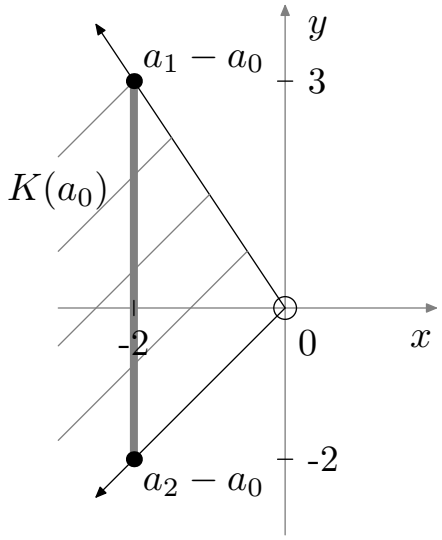
где $A(a_i)_j$ — j-й столбец матрицы $A(a_i)$.

Для иллюстрации описанных множеств рассмотрим пример:

$$A = \left\{ \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \begin{pmatrix} -2 \\ 6 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \end{pmatrix} \right\}, \qquad A(a_0) = \left\{ \begin{pmatrix} -2 \\ 3 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix} \right\}. \tag{3.1}$$



а) $\operatorname{conv} A$



б) $K(a_0)$

Рисунок 3.1 — Изображения выпуклой оболочки множества A и конуса $K(a_0)$

Формализуем поставленную задачу. Рассмотрим сложение многогранника $\operatorname{conv} A$ с отрезком $\operatorname{conv}\{b_0, b_1\}$, $b_0, b_1 \in \mathbb{R}^n$, $b_0 \neq b_1$:

$$\operatorname{conv} A + \operatorname{conv}\{b_0, b_1\} =$$
$$= \operatorname{conv}\{a_0 + b_0, a_1 + b_0, \ldots, a_m + b_0, a_0 + b_1, a_1 + b_1, \ldots, a_m + b_1\}. \tag{3.2}$$

Выражение (3.2) можно представить, как сумму точки и множества

$$\operatorname{conv} A + \operatorname{conv}\{b_0, b_1\} = b_0 + \operatorname{conv} A_v, \tag{3.3}$$

где $A_v = \{a_0, a_1, \ldots, a_m, a_0 + v, a_1 + v, \ldots, a_m + v\}$, $v = b_1 - b_0$.

Свяжем описанные множества с приведенными примерами. Для рис. 2.2 имеем $b_0 = (0, 0)^T$, $b_1 = (4, 0)^T$, множество A из (3.1). Тогда выпуклая оболочка $\operatorname{conv} A_v$ изображена на рис. 2.2 б). Рассмотрим другое (3.3) представление суммы многогранника и отрезка. Множество $\operatorname{conv} A_v$ можно получить смещая $\operatorname{conv} A$ на вектор v, при этом «запоминая» весь след смещения. В новых обозначениях

получим рис. 3.2.

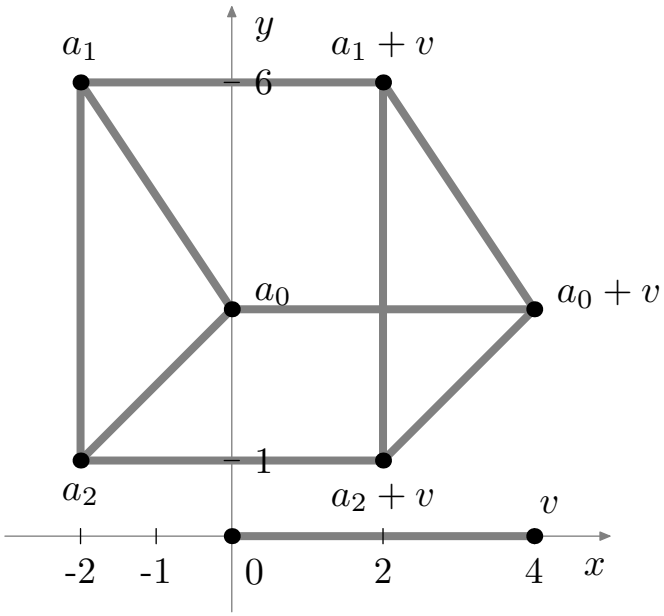


Рисунок 3.2 — Множество A_v и вектор смещения v

Выявим крайние точки множества $\operatorname{conv} A_v$. Справедлива следующая лемма [**ANG**].

Лемма 1. *Для каждой точки a_i из множества A справедливо:*

1. *Если $v \notin K(a_i)$, то $a_i + v$ является крайней для* $\operatorname{conv} A_v$.
   В противном случае $a_i + v$ не является крайней для $\operatorname{conv} A_v$.
2. *Если $-v \notin K(a_i)$, то a_i является крайней для* $\operatorname{conv} A_v$.
   В противном случае a_i не является крайней для $\operatorname{conv} A_v$.

Замечание 1. Лемма (1) задает необходимое и достаточное условие крайности точки для суммы многогранника и отрезка.

Проверим графически условия леммы для точек a_0 и $a_0 + v$ из рис. 3.2. Изобразим конус $K(a_0)$ и векторы v, $-v$ (см. рис. 3.3).

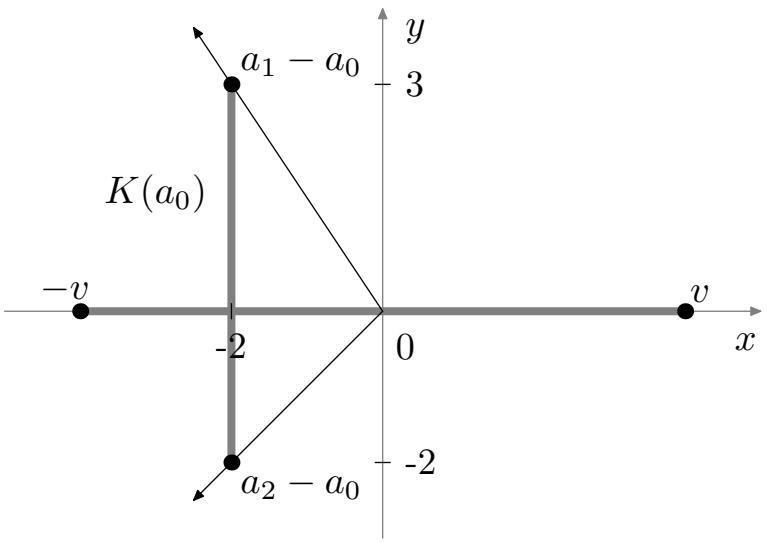


Рисунок 3.3 — Проверка условий леммы

На рис. 3.3 видно, что $v \notin K(a_0)$. Получаем, что точка $a_0 + v$ является крайней. Теперь рассмотрим второй пункт леммы. Вектор $-v$ принадлежит конусу $K(a_0)$. Тогда a_0 не является крайней.

4. Решение задачи

4.1 Алгоритм проверки условий крайности

Пусть все нормали к граням множества $\operatorname{conv} A$ направлены наружу относительно к $\operatorname{conv} A$. Многогранник $\operatorname{conv} A$ имеет грани F_k, с нормалями n_k, $k \in 1 : m_{F(A)}$. Грани F_k задаются своими крайними точками, т.е.

$$F_k = \operatorname{conv}\{a_{k_1}, \ldots, a_{k_{m(F_k)}}\},$$

где все $a_{k_i} \in A$, $m_{F(A)}$ — количество крайних точек грани F_k.

Зададим множество индексов активных граней, в которых участвует точка $a_i \in A$, $i \in 1 : m$:

$$F(a_i) = \{k \in 1 : m_{F(A)} \,|\, a_i \in F_k\}.$$

Запишем принципиальный алгоритм, который проверяет условия леммы 1 для точек a_i и $a_i + v$, $i \in 0 : m$.

Алгоритм

Для всех $i \in 0 : m$ выполнить:

1. Выберем точку $a_i \in A$. Найдем множество активных граней $F(a_i)$.
2. Вычислим для всех $k \in F(a_i)$ скалярное произведение

$$s_k := \langle n_k, v \rangle.$$

   Если хотя бы для одного индекса $k \in F(a_i)$ число $s_k > 0$, то $v \notin K(a_i)$. Переходим к шагу 3.

   В противном случае переходим к шагу 4.
3. Точка $a_i + v$ является крайней для $\operatorname{conv} A_v$. Переходим к шагу 5.
4. Точка $a_i + v$ не является крайней для $\operatorname{conv} A_v$.
5. Для всех $k \in F(a_i)$ имеем

$$s_k := \langle n_k, -v \rangle = -s_k.$$

Если хотя бы для одного индекса $k \in F(a_i)$ выполняется $s_k > 0$, то $-v \notin K(a_i)$. Переходим к шагу 6.

В противном случае переходим к шагу 7.

6. Точка a_i является крайней для $\operatorname{conv} A_v$. Переходим к шагу 8.
7. Точка a_i не является крайней для $\operatorname{conv} A_v$.
8. Конец рассмотрения индекса i.

Проиллюстрируем работу алгоритма на примере (3.1).

Пример работы алгоритма. Пусть

$$A = \left\{ \bigcup_{i=0:2} a_i \right\} = \left\{ \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \begin{pmatrix} -2 \\ 6 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \end{pmatrix} \right\}, \quad v = \begin{pmatrix} 4 \\ 0 \end{pmatrix},$$

$$A_v = \left\{ \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \begin{pmatrix} -2 \\ 6 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right\}.$$

Треугольник $\operatorname{conv} A$ имеет стороны F_k, с нормалями n_k, $k \in 1 : 3$:

$$F_1 = \operatorname{conv}\{a_0, a_1\} \qquad F_2 = \operatorname{conv}\{a_1, a_2\}, \qquad F_3 = \operatorname{conv}\{a_2, a_0\},$$

$$n_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \qquad n_2 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \qquad n_3 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

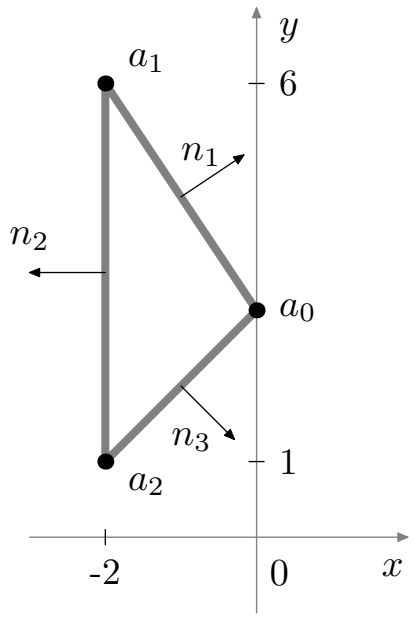


Рисунок 4.1 — Нормали к граням $\operatorname{conv} A$

Случай $i = 0$:

1. Имеем $F(a_0) = \{1, 3\}$.

2.

$$s_1 = \langle n_1, v\rangle = 12,$$
$$s_3 = \langle n_3, v\rangle = 4.$$

Отсюда $v \notin K(a_0)$.

3. Точка $a_0 + v$ является крайней для $\operatorname{conv} A_v$.

5.

$$s_1 = \langle n_1, -v\rangle = -12,$$
$$s_3 = \langle n_3, -v\rangle = -4.$$

Отсюда $-v \in K(a_0)$.

7. Точка a_0 не является крайней для $\operatorname{conv} A_v$.

8. Конец рассмотрения индекса $i = 0$.

Случай $i = 1$:

1. Имеем $F(a_1) = \{1, 2\}$.

2.

$$s_1 = \langle n_1, v\rangle = 12,$$
$$s_2 = \langle n_2, v\rangle = -4.$$

Отсюда $v \notin K(a_1)$.

3. Точка $a_1 + v$ является крайней для $\operatorname{conv} A_v$.

5.

$$s_1 = \langle n_1, -v\rangle = -12,$$
$$s_2 = \langle n_2, -v\rangle = 4.$$

Отсюда $-v \notin K(a_1)$.

6. Точка a_1 является крайней для $\operatorname{conv} A_v$.

8. Конец рассмотрения индекса $i = 1$.

Случай $i = 2$:

1. Имеем $F(a_2) = \{3, 2\}$.

2.

$$s_3 = \langle n_3, v \rangle = 4,$$
$$s_2 = \langle n_2, v \rangle = -4.$$

Отсюда $v \notin K(a_0)$.

3. Точка $a_2 + v$ является крайней для $\operatorname{conv} A_v$.

5.

$$s_3 = \langle n_3, -v \rangle = -4,$$
$$s_2 = \langle n_2, -v \rangle = 4.$$

Отсюда $-v \notin K(a_0)$.

6. Точка a_2 является крайней для $\operatorname{conv} A_v$.

8. Конец рассмотрения индекса $i = 2$.

По окончанию алгоритма идентифицируем точку a_0, как не являющейся крайней для многоугольника $\operatorname{conv} A_v$. Удалим ее из A_v и получим

$$\operatorname{conv} A_v = \operatorname{conv} \left\{ \begin{pmatrix} -2 \\ 6 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right\}.$$

4.2 Программная реализация

Опишем общую схему программной реализации.

1. Генерируется или задается множество точек M в пространстве $\mathbb{R}^n$.
2. Вычисляется выпуклая оболочка множества M. Набор полученных крайних точек записывается в множество A. Сохраняется информация о гранях F_k многогранника $\operatorname{conv} A$ и соответствующих нормалях n_k, $k \in 1 : m_{F(A)}$.
3. Вводится вектор смещения $v \in \mathbb{R}^n$. Строится массив allPoints, коотрый соответствует множеству $A_v = \{A \bigcup \{A + v\}\}$.
4. Для всех точек из A_v проверяется лемма 1. Выводится затраченное время на этот пункт. Крайние точки записывается в массиве resultPoints.

5. Вычисляется выпуклая оболочка множества A_v. Полученные крайние точки сравниваются с resultPoints. Если есть несовпадения, выводится информация о конкретных нестыковках.

Алгоритм проверки леммы реализован на языке программирования JavaScript с использованием библиотек quickhull3d [5] версии >= 2.0.0 для построения выпуклой оболочки в трёхмерном пространстве и t3-boilerplate версии >= 0.2.7 для отображения модели. Для случая четырёхмерного, пяти-мерного и шести-мерного пространств для построения выпуклой оболочки использовалась библиотека Qhull [3] реализованная на языке программирования C.

В случае трёхмерного пространства для удобства задания первоначального набора данных можно задать как параметры авто-генерации точек пространства (по умолчанию), либо загрузить в корневую директорию проекта файл points.js с набором точек в формате JSON. Пример содержания файла points.js:

```
var DIMENSIONS = 3;
var points = [ [0, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1] ];
```

Реализация алгоритма проверки леммы в трёх-мерном пространстве с построением 3D модели находится в файле libs/scripts.js, в папке libs находятся так же вспомогательные библиотеки такие как JQuery версии 1.12.3 [6] и THREE [7]. Для запуска следует открыть файл index.html в любом современном браузере например Chrome или FireFox. Важно, чтобы браузер поддерживал современные инструменты вроде canvas. Для получения подробной информации о ходе выполнения проверки следует запустить панель отладки браузера (Chrome, FireFox: Ctrl + Shift + i). Настройки параметров авто-генерации и другие настройки находятся в теле файла index.html. При загрузке файла index.html происходит построение выпуклой оболочки с помощью библиотеки quickhull3d и отображение результата на сцене с помощью библиотек t3-boilerplate и THREE, а так же отображается набор нормалей для граней выпуклой оболочки. Пользователю предлагается задать смещение v, относительно осей координат x, y, z, точек отобранных для построения выпуклой оболочки алгоритмом Quickhull. После указания смещения следует нажать кнопку “Построить”. В результате отобразится исходная выпуклая оболочка с заданным смещением. Далее для запуска алгоритма построения результирующей выпуклой оболочки с помощью леммы, с включением исходных точек и точек смещения (множество A_v) следует нажать кнопку “Проверить”.

Описание алгортма.

На первоначальном этапе исходные точки (M) записываются в массив points глобального контекста

```
// исходный набор точек
var points = [];
```

По исходному набору точек просходит построение выпуклой оболочки $\mathrm{conv}\, A$ с помошью библиотеки quickhull3d (см. рис. (4.2))

```
faces = qh(points);
```

Массив faces (F_k) представляет из себя массив граней выпуклой оболочки с нормалями (n_k) к ним. Далее выполняется построение 3D модели по заданным точкам и граням выпуклой обочки с нормалями. В процессе построения заполняются массивы normals - массив нормалей, pointNormals (нобор нормалей для каждой из точек граней) и массив qhPointsLinks — ссылки на точки выпуклой оболочки в исходном массиве points.

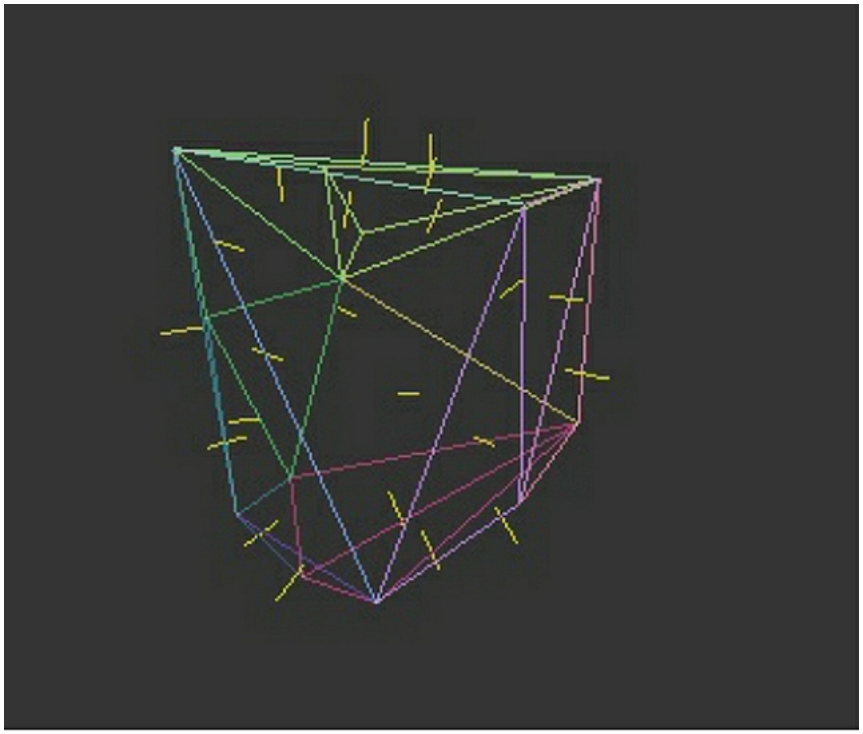


Рисунок 4.2 — Выпуклая оболочка множества A

На этапе построения смещения относительно осей координат, заполняются массивы newPoints - точки для построения выпулой оболчки смещения и массив allPoints (A_v) - точки включающие в себя исходный набор и точки смещения.

```
var i;
newPoints = points.slice();
```

```
allPoints = points.slice();
var x = parseInt(document.querySelector("#coefficientX").value);
var y = parseInt(document.querySelector("#coefficientY").value);
var z = parseInt(document.querySelector("#coefficientZ").value);
if (isNaN(x)) {
        x = 0;
}
if (isNaN(y)) {
        y = 0;
}
if (isNaN(z)) {
        z = 0;
}
for (i = 0; i < qhPointLinks.length; i += 1) {
        var point = [
        parseFloat(points[qhPointLinks[i]][0]) + x,
        parseFloat(points[qhPointLinks[i]][1]) + y,
        parseFloat(points[qhPointLinks[i]][2]) + z
        ];
        newPoints[qhPointLinks[i]] = point;
        allPoints.push(point);
}
```

Далее происходит построение смещённой выпуклой оболочки по точкам массива newPoints аналогично первому этапу (см. рис. (4.3)), при этом нормали и набор pointNormals заполняется заново, сами нормали остаются неизменными как и ссылки на точки выпуклой оболочки в этих массивах.

На завершающем этапе проверки используется лемма 1 для отбора точек выпуклой оболчки и происходит проверка этих точек посредством сравнения каждой точки с точкой отобранной по алгоритму Quickhull для набора точек allPoints. По условиям леммы для каждой точки граней pointNormals проверям скалярное произведение вектора (v) построенного из точка A исходной выпуклой оболочки в точку A’ смещённой выпуклой оболочки.

```
var result;
var i;
```

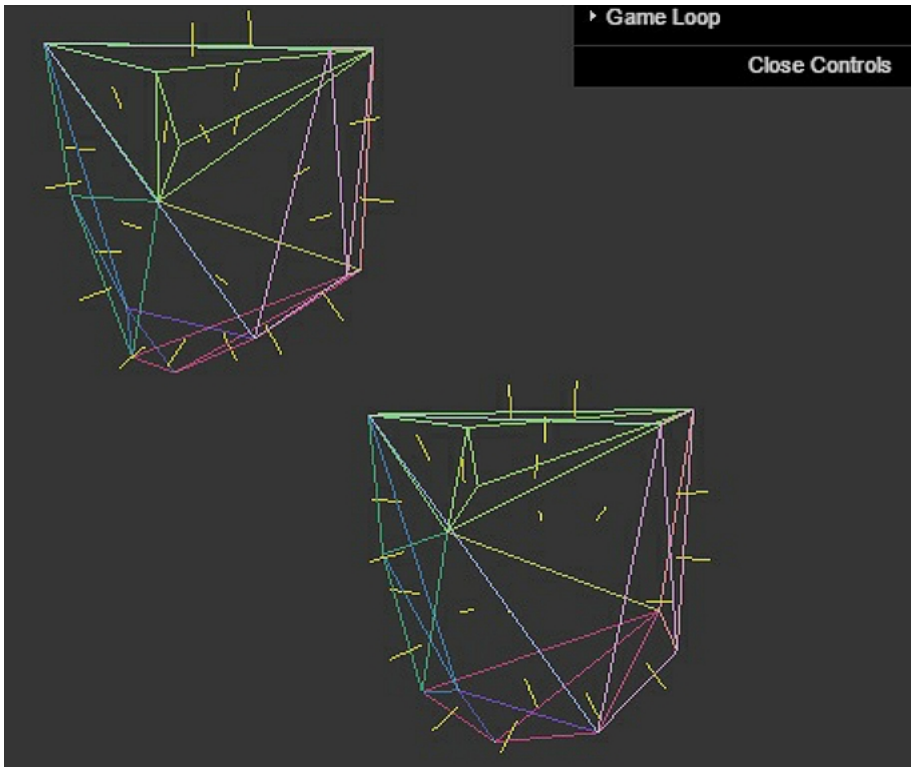


Рисунок 4.3 — Выпуклые оболочки множеств A и $A + v$

```
var a, b;
//Строим вектор AA' (v) где A' (ai + v) точка смещения
a = [newPoints[index][0] - points[index][0], newPoints[index][1] - points[index][1], newPoints[index][2] - points[index][2]];
//Проверяем условия леммы
for (i = 0; i < item.length; i += 1) {
	//Вычисляем скалярное произведения вектора AA' (v) и нормалей к граням которые содержат точку A
	result = (a[0]*item[i].x) +
	(a[1]*item[i].y) +
	(a[2]*item[i].z);
//Если скалярное произведение < 0 то прерываем цикл
	if (result < 0) {
		break;
	}
}
//Если скалярное произведение < 0 добавляем точку в результирующий массив
if (result < 0) {
```

```
        resultPoints.push(points[index]);
} else {
        excludedPoints.push(points[index]);
}
```

Аналогично производится проверка ветора A'A (-v) в результате мы получаем набор точек resultPoints - крайние точки множества $\mathrm{conv}\, A_v$ и набор excludedPoints - точки не удовлетворяющие условиям отбора. На рис. 4.4 некрайние точки показаны синими кружками. Далее по точкам allPoints (A_v) происходит построение выпуклой оболочки. Сравниваются крайние точки последней с resultPoints.

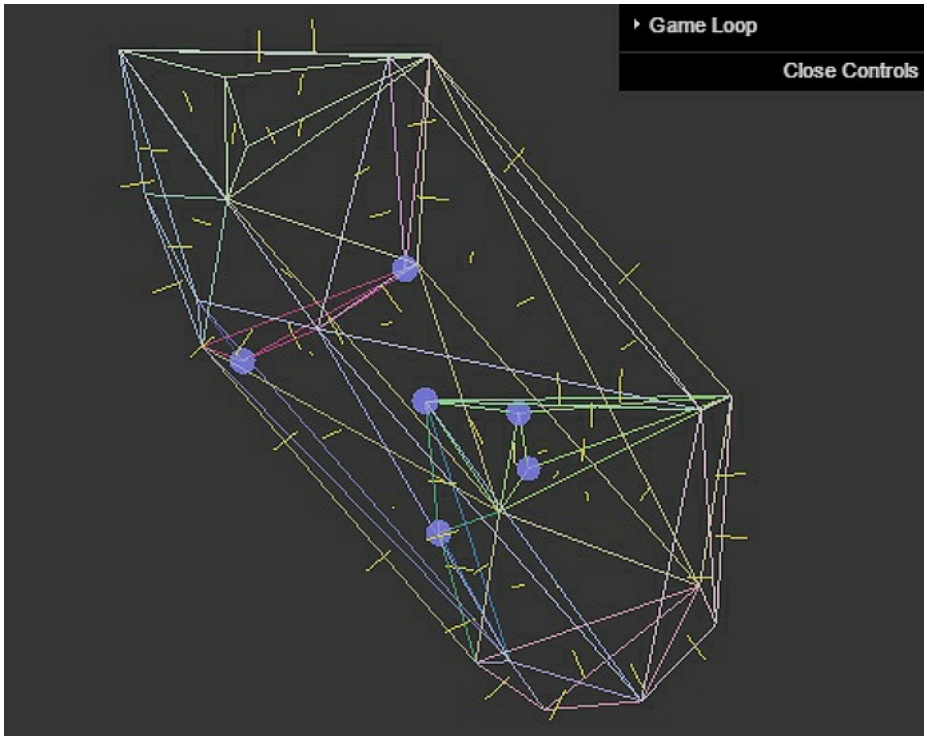


Рисунок 4.4 — Идентифицированные точки леммой и $\mathrm{conv}\, A_v$

```
        console.log('Построение выпуклой оболочки смещения');
var resultFaces = qh(resultPoints);
buildHull(this, resultPoints, resultFaces)
console.log('Количество точек', qhPointLinks.length, resultPoints.length);
resultFaces = qh(allPoints);
buildHull(this, allPoints, resultFaces)
console.log('Проверка количества точек', qhPointLinks.length);
for (var i = 0; i < resultPoints.length; i++) {
```

```
        var noMatch = true;
        for (var j = 0; j < qhPointLinks.length; j++) {
            if (allPoints[qhPointLinks[j]][0] === resultPoints[i][0] &&
            allPoints[qhPointLinks[j]][1] === resultPoints[i][1] &&
            allPoints[qhPointLinks[j]][2] === resultPoints[i][2]
            ) {
                console.log('match: ' + i, resultPoints[i]);
                noMatch = false;
                break;
            }
        }
        if (noMatch) {
            console.log('ERROR: ' + i, resultPoints[i]);
        }
    }
    if (excludedPoints.length > 0) {
        for (var i = 0; i < excludedPoints.length; i += 1) {
            console.log('Excluded: ' + i, excludedPoints[i]);
        }
    }
```

Для проверки работы алгоритма в 4х 5ти и 6ти мерном пространствах, будет использован сервис [4], который представляет собой API для доступа к пакету qhull.org. Сервис принимает на вход два не обязательных параметра. Первый принимает значение вида D1 - D6 (http://freestel.ru/api/qhull/D4), которое определяет размерность пространства. Второй аргумент указывает количество точек, если используется сценарий автоматической генерации точек на стороне сервиса через утилиту rbox (http://freestel.ru/api/qhull/D4/100) из пакета qhull. Сервис отдаёт набор граней и нормалей к ним, используя результат работы утилиты qconvex из пакета qhull. Все остальные пункты проверки леммы аналогичны пунктам для 3-х мерного пространства.

5. Заключение

В работе разработан программный пакет на языке JavaScript для численной проверки необходимого и достаточного условия крайности точек суммы двух многогранников. Крайние точки сравнивались с результатами нахождения выпуклой оболочки этой суммы. Численные эксперименты подтверждают теоретические результаты. В трехмерном случае удалось наглядно изобразить каждый этап программной реализации.

Список литературы

[1] Ангелов Т. А. Нахождение крайних точек суммы двух политопов. Семинар «CNSA & NDO». Избранные доклады. 05 мая 2016.

[2] Рокафеллар Р. Выпуклый анализ // М.: Мир, 1973. 472 с.

[3] Barber C. B., Dobkin D. P., Huhdanpaa H. T. The Quickhull algorithm for convex hulls // ACM Trans. on Mathematical Software. 1996. 22. 469–483 (http://www.qhull.org).

Электронные ресурсы:

[4] http://freestel.ru/api/qhull

[5] https://github.com/maurizzzio/quickhull3d

[6] http://jquery.com

[7] http://treejs.org