Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Ярославский государственный университет им. П.Г. Демидова"

Кафедра теории функций и функционального анализа

«Допустить к защите»
Зав.кафедрой,
к.ф.-м.н., доцент
_________ М.В.Невский
«__» ___________ 20__ г.

Выпускная квалификационная работа

Вычисление геометрических характеристик n-мерного симплекса

Научный руководитель
к.ф.-м.н., доцент
_________ М.В.Невский
«__» ___________ 20__ г.

Студентка группы ПМИ-51СО
_________ А.А.Князева
«__» ___________ 20__ г.

Ярославль 2015 г.

Содержание

1 Введение. Постановка задачи

Данная работа посвящена решению ряда задач, связанных с невырожденным симплексом $S \subset \mathbb{R}^n$. Эта тема является актуальной и в последнее время ей было посвящено множество работ, например [1], [3], [4] и [5]. Основными задачами являются вычисление осевых диаметров n-мерного симплекса, минимального положительного коэффициента гомотетии при поглощении симплексом единичного куба, а также вычисление в симплексе максимального отрезка заданного направления.

Целью работы является изучение методов решения поставленных задач и реализация их на ЭВМ. Программа должна подсчитывать осевые диаметры, коэффициенты λ_j базисных многочленов Лагранжа, норму интерполяционного проектора и величину α по заданным вершинам невырожденного симплекса S, максимальный отрезок заданного направления (направление задаётся вектором размерности n) и некоторые другие характеристики.

Теоретическая часть работы содержит необходимые сведения по геометрии симплекса, взятые из [1] - [6]. Представленные в этой части работы формулы используются для нахождения осевых диаметров n-мерного симплекса, базисных многочленов Лагранжа и других величин, связанных с поставленными задачами. Вторая часть включает в себя описание программы. Затем рассмотрены некоторые примеры, подтверждающие точность полученных в завершении работы программы результатов. Обобщает все это заключение, в котором подводится итог всей проделанной работе. В приложении представлен код реализованной программы. Материалы данной работы частично опубликованы в статье автора [7].

2 Геометрические характеристики n-мерного симплекса

2.1 Некоторые свойста базисных многочленов Лагранжа

Пусть $n \in \mathbb{N}$. Элемент $x \in \mathbb{R}^n$ будем записывать в виде $x = (x_1, \dots, x_n)$. Положим $Q_n := [0,1]^n$. Через $C(Q_n)$ обозначим пространство непрерывных функций $F : C(Q_n) \to \mathbb{R}$ с нормой $\|f\|_{C(Q_n)} := \max_{x \in Q_n} |f(x)|$, а через $\Pi_1(\mathbb{R}_n)$ – совокупность многочленов от n переменных степени $\leqslant 1$.

Рассмотрим невырожденный симплекс $S \subset \mathbb{R}^n$, то есть такой, что $vol(S) \neq 0$. Обозначим вершины S через $x^{(j)} = \left(x_1{}^{(j)}, \dots, x_n{}^{(j)}\right), j = 1, \dots, n+1$. Из координат вершин $x^{(j)}$ составим матрицу

$$\mathbf{A} := \begin{pmatrix} x_1{}^{(1)} & \dots & x_n{}^{(1)} & 1 \\ x_1{}^{(2)} & \dots & x_n{}^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_1{}^{(n+1)} & \dots & x_n{}^{(n+1)} & 1 \end{pmatrix}.$$

Пусть $\Delta := det(\mathbf{A})$, а определитель $\Delta_j(x)$ получается из Δ заменой j-й строки на строку $(x_1, \dots, x_n, 1)$. Введем в рассмотрение многочлены, определяемые равенством $\lambda_j(x) = \Delta_j(x)/\Delta$. Многочлены $\lambda_j(x) = 0$ задают $(n-1)$-мерные грани S. Имеет место представление

$$S = \{x \in \mathbb{R}^n : \lambda_j(x) \geq 0, j = 1, \dots, n+1\}. \tag{2.1}$$

В дальнейшем используется запись

$$\lambda_j(x) = l_{1j}x_1 + \dots + l_{nj}x_n + l_{n+1,j}. \tag{2.2}$$

Обозначим через $d_i(S)$ максимальную длину отрезка, содержащегося в S и параллельного оси x_i. Величину $d_i(S)$ будем называть i-м осевым диаметром S. В [1] доказано, что для любого $i = 1, \dots, n$ справедливо равенство

$$\frac{1}{d_i(S)} = \frac{1}{2} \sum_{j=1}^{n+1} |l_{ij}|\,. \tag{2.3}$$

В симплексе S существует ровно один отрезок длины $d_i(S)$, параллельный оси x_i. Концы $y_+^{(i)}$ и $y_-^{(i)}$ этого отрезка суть

$$y_+^{(i)} = \sum_{j=1}^{n+1} s_{ij} x^{(j)}, s_{ij} := \frac{|l_{ij}| + l_{ij}}{\sum\limits_{k=1}^{n+1} |l_{ik}|}; \tag{2.4}$$

$$y_-^{(i)} = \sum_{j=1}^{n+1} t_{ij} x^{(j)}, t_{ij} := \frac{|l_{ij}| - l_{ij}}{\sum\limits_{k=1}^{n+1} |l_{ik}|}. \tag{2.5}$$

Обозанчим за $\Sigma_i(S)$ $(n-1)$-меру проекции S на гиперплоскость $x_i = 0$. Имеет место равенство

$$vol(S) = \frac{d_i(S) \cdot \Sigma_i(S)}{n}, \tag{2.6}$$

Если $Q_n \subset S$, то справедливо неравенство

$$\sum_{i=1}^{n} \frac{1}{d_i(S)} \leq 1. \tag{2.7}$$

Из (2.7) следует, что в случае $Q_n \subset S$ для некоторого $i = 1, \ldots, n$ симплекс S содержит отрезок длины n, параллельный оси x_i.

Если $S \subset Q_n$, то многочлены λ_j могут применяться для вычисления величины $\xi(S) := \min\{\sigma \geq 1 : Q_n \subset \sigma S\}$, где σS есть результат гомотетии S относительно центра тяжести с коэффициентом σ.

$$\xi(S) = (n+1) \max_{1 \leq j \leq n+1} \max_{x \in ver(Q_n)} (-\lambda_j(x)) + 1. \tag{2.8}$$

Здесь и далее $ver(Q_n)$ есть совокупность вершин Q_n.

Пусть $P : C(Q_n) \to \Pi_1(\mathbb{R}^n)$ – интерполяционный проектор, узлы которого совпадают с вершинами $S \subset Q_n$. Этот проектор определяется равенствами $Pf\left(x^{(j)}\right) = f_j := f\left(x^{(j)}\right)$. Справедлив следующий аналог интерполяционной формулы Лагранжа:

$$Pf(x) = p(x) := \sum_{j=1}^{n+1} f_j \lambda_j(x), \tag{2.9}$$

в связи с чем многочлены λ_j мы называем базисными многочленами Лагранжа. Из (2.9) получается, что норма P как оператора из $C(Q_n)$ в

$C(Q_n)$ может быть найдена по формулам

$$\|P\| = \max_{x \in ver(Q_n)} \sum_{j=1}^{n+1} |\lambda_j(x)| = \max_{f_j = \pm 1} \max_{x \in ver(Q_n)} |p(x)|. \qquad (2.10)$$

Как установлено в [1], в случае $S \subset Q_n$ введённые нами величины связаны двойным неравенством

$$\sum_{i=1}^{n} \frac{1}{d_i(S)} \leq \xi(S) \leq \frac{n+1}{2}(\|P\| - 1) + 1. \qquad (2.11)$$

Пусть C - выпуклое тело в $\mathbb{R}^n$, т.е. компактное выпуклое подмножество $\mathbb{R}^n$ с непустой внутренностью. Через σC обозначим результат гомотетии C относительно центра тяжести с коэффициентом σ. Символом $d_i(C)$ обозначим i-й осевой диаметр C. Через Q_n обозначим n-мерный единичный куб $[0,1]^n$. Под транслятором будем понимать результат параллельного переноса.

Для выпуклых тел $C_1, C_2 \subset \mathbb{R}^n$ обозначим через $\alpha(C_1; C_2)$ минимальное $\sigma > 0$, для которого C_1 принадлежит транслятору σC_2. В [3] доказано, что для любого выпуклого тела C справедливо неравенство

$$\alpha(Q_n; C) \leq \sum_{i=1}^{n} \frac{1}{d_i(C)}. \qquad (2.12)$$

Если же C представляет собой невырожденный симплекс S, то это соотношение обращается в равенство, т.е. имеет место

$$\alpha(Q_n; S) = \sum_{i=1}^{n} \frac{1}{d_i(S)}. \qquad (2.13)$$

(см. [4, теорема 4]).

Из (2.13) и (2.3) получается, что

$$\alpha(Q_n; S) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} |l_{ij}|. \qquad (2.14)$$

2.2 Вычисление максимального в симплексе отрезка данного направления

Пусть v – ненулевой n-мерный вектор. Обозначим через $d^v(S)$ максимальную длину отрезка, принадлежащего S и параллельного v. Далее представлены формулы для вычисления величины $d^v(S)$ и концов максимального отрезка по координатам v и вершинам S. В случае, когда v коллинеарен i-й координатной оси, положим $d_i(S) := d^v(S)$, где $d_i(S)$ – i-ый осевой диаметр S.

Обозначим через $v_1, \ldots v_n$ координаты данного нам вектора v. Введем в рассмотрение числа $(1 \leq j \leq n{+}1)$

$$m_j := \sum_{k=1}^{n} l_{kj} v_k, \tag{2.15}$$

$$\alpha_j := \frac{|m_j| - m_j}{\sum\limits_{k=1}^{n+1} |m_k|}, \quad \beta_j := \frac{|m_j| + m_j}{\sum\limits_{k=1}^{n+1} |m_k|}. \tag{2.16}$$

Через $\|\cdot\|$ обозначим евклидову норму в $\mathbb{R}^n$. В [5] доказано следующее.

Теорема. *Величина $d^v(S)$ удовлетворяет равенству*

$$d^v(S) = \frac{2\|v\|}{\sum\limits_{j=1}^{n+1} |m_j|}. \tag{2.17}$$

Концы единственного отрезка максимальной длины, принадлежащего S и параллельного v, есть точки

$$a = \sum_{j=1}^{n+1} \alpha_j x^{(j)}, \quad b = \sum_{j=1}^{n+1} \beta_j x^{(j)}. \tag{2.18}$$

Для доказательства данной теоремы в [5] используются следующие вспомогательные предположения.

Лемма 1. *Пусть I – отрезок, параллельный v и расположенный в S таким образом, что каждый $(n-1)$-мерная грань S содержит хотя бы один из его концов. Тогда длина I совпадает с правой частью (2.17).*

Лемма 2. *В S существует единственный отрезок, параллельный вектору v и расположенный таким образом, что каждая $(n-1)$-мерная грань S содержит хотя бы один из его концов. Этот отрезок является единственным отрезком из S максимальной длины, параллельным v.*

С привлечением результатов, описанных в [1] и [2] можно получить следующие следствия. Пусть $\Sigma(S; v)$ есть $(n-1)$-мерная мера проекции симплекса S на гиперплоскость, ортогональную вектору v.

Следствие 1. *Имеют места равенства*

$$\Sigma(S; v) = \frac{n * vol(S)}{d^v(S)} = \frac{|det(A)|}{2(n-1)!\|v\|} \sum_{j=1}^{n+1} |m_j|. \tag{2.19}$$

Через σS обозначим образ S при гомотетии с коэфиициентом σ и центром гомотетии в центре тяжести S. Пусть V – невырожденный параллелепипед в $\mathbb{R}^n$, рёбра которого задаются линейно независимыми векторами $v^{(1)}, \ldots, v^{(n)}$. Через $\alpha(V; S)$ обозначим минимальное $\sigma > 0$ такое, что V одержится в трансляте сиплекса σS. Вычислим величину

$$M := \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n+1} \left| \sum_{k=1}^{n} l_{kj} v_k^{(i)} \right|. \tag{2.20}$$

Следствие 2. *Справедливы равенства*

$$\alpha(V; S) = \sum_{i=1}^{n} \frac{\|v^{(i)}\|}{d^{v^{(i)}}(S)} = M. \tag{2.21}$$

Следствие 3. *Неравенство $M \leq 1$ эквивалентно тому, что V содержится в трансляте симплекса S. Равенство $M = 1$ эквивалентно тому, что некоторый транслят S' симплекса S описан вокруг V, т.е. $V \subset S'$ и каждая $(n-1)$-мерная грань S' содержит вершину V.*

2.3 Симплекс и куб в $\mathbb{R}^n$

В данном разделе рассматривается задача о вычислении для симплекса S такой точки $x \in \mathbb{R}^n$, для которой с минимально возможным коэффициентом $\sigma > 0$ справедливо включение $Q_n \subset S_{x,\sigma}$. В [6] автор доказывает, что задача имеет решение, и причём единственное, в случае $\alpha(S) \neq 1$; при этом минимальное σ как раз и равно $\alpha(S)$. Далее приведены формулы, в которых центр x минимальной положительной гомотетии вычисляется через вершины S и числа l_{ij} – коэффициенты многочленов λ_j (см (2.2)).

Пусть S - невырожденный симплекс в $\mathbb{R}^n$. Из определения $\alpha(S)$ (см. 2.14) легко следует, что некоторый транслянт симплекса описан вокруг

Q_n. Поэтому $\alpha(S) = 1$ тогда и только тогда, когда существует транслят S описанный вокруг Q_n.

Теорема 1. *Если $\sigma = \sum_{i=1}^{n} 1/d_i(S) \neq 1$, то существует единственная точка $x = (x_1, \ldots, x_n)$ такая, что $Q_n \subset S_{x,\sigma}$. Имеют место равенства*

$$x_k = \frac{1}{2(\sigma - 1)} \left[\sum_{j=1}^{n+1} \left(\sum_{i=1}^{n} |l_{ij}| \right) x_k^{(j)-1} \right], k = 1, \ldots, n. \tag{2.22}$$

Если $0 < \sigma < \sum_{i=1}^{n} 1/d_i(S)$, то для любой $x \in \mathbb{R}^n$ верно $Q_n \not\subset S_{x,\sigma}$.

В [6] приведены следующие формулы для вычисления x, в которых используются только вершины S и числа l_{ij}.

Теорема 2. *Для невырожденного симплекса $S \subset \mathbb{R}^n$ условие $\alpha(S) \neq 1$ эквивалентно*

$$\sum_{i=1}^{n} \sum_{j=1}^{n+1} |l_{ij}| \neq 2. \tag{2.23}$$

Пусть выполнено (2.23) и $\sigma := \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n+1} |l_{ij}|$. Тогда единственная точка x, для которой верно включение $Q_n \subset S_{x,\sigma}$, может быть вычислена по равенствам

$$x_k = \frac{\sum_{j=1}^{n+1} (\sum_{i=1}^{n} |l_{ij}|) x_k^{(j)} - 1}{\sum_{i=1}^{n} \sum_{j=1}^{n+1} |l_{ij}| - 2}, k = 1, \ldots, n. \tag{2.24}$$

Рассмотрим ситуацию, когда симплекс содержиться в Q_n. $d_i(S) \leq 1$, поэтому $\alpha(S) \geq n$ (равенство имеет место тогда и только тогда, когда каждый осевой диаметр равен 1). Далее идёт специальный вариант теоремы 1 для этой ситуации.

Теорема 3. *Пусть $S \subset Q_n$ и $d_1(S) = \cdots = d_n(S) = 1$. Существует единственная точка $x \in S$ такая, что $Q_n \subset S_{x,n}$. При $n > 1$ имеют место равенства*

$$x_k = \frac{1}{2(n - 1)} \left[\sum_{j=1}^{n+1} \left(\sum_{i=1}^{n} |l_{ij}| \right) x_k^{(j)-1} \right], k = 1, \ldots, n. \tag{2.25}$$

Если $0 < \sigma < n$, то для любой $x \in \mathbb{R}^n$ верно $Q_n \not\subset S_{x,n}$.

3 Описание программы

Основной целью моей работы было написание программы, реализующей алгоритм нахождения различных геометрических характеристик симплекса по заданным вершинам.

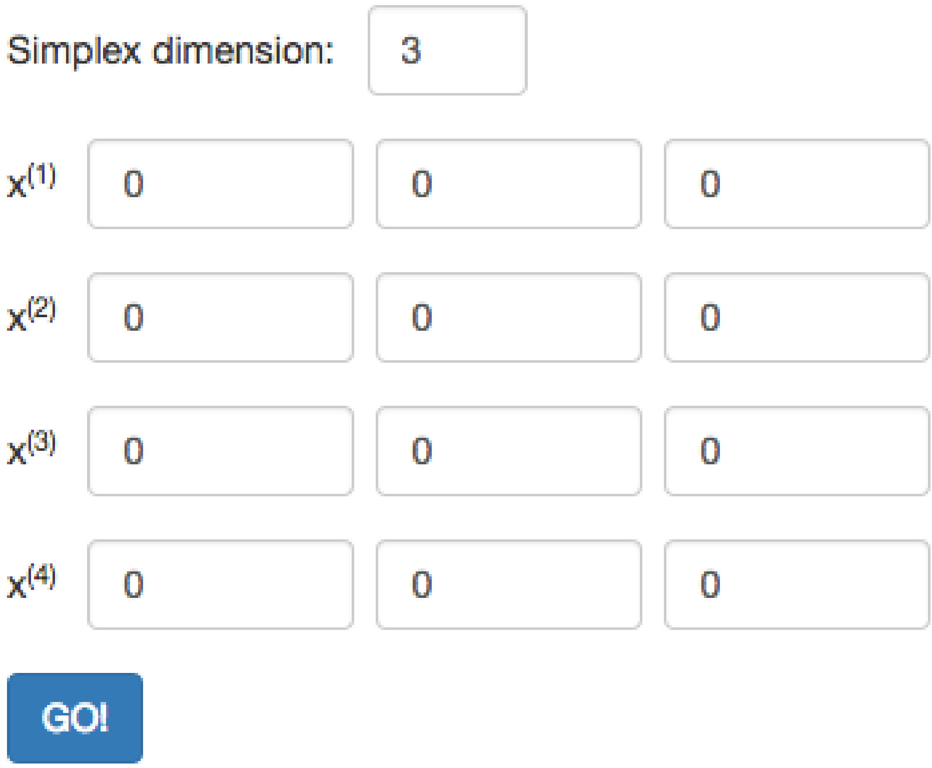
Программа написана на языке JavaScript и представляет собой веб-приложение. Серверная часть реализована на NodeJs. Для сборки клиентской части использовался GULP, пакетный менеджер на клиентской части - Bower. Программа написана с использованием популярного фрэймворка AngularJS. Так же применялся front-end фрэймворк Bootstrap. На странице приложения вводятся координаты вершин симплекса, вектор, параллелограмм (представленный n векторами). После завершения подсчета на экран выводятся осевые диаметры (длины и координаты концов), коэффициенты базисных многочленов Лагранжа, норма проектора $\|P\|$, минимальный транслятор $\alpha(Q_n; S)$, максимальный вписанный отрезок, параллельный заданному вектору (длина и координаты концов), $\Sigma(S; v)$, $\alpha(V; S)$ для заданного параллелограмма. Все исходники хранятся в репозитории на GitHub.

Программа включает в себя следующие файлы:

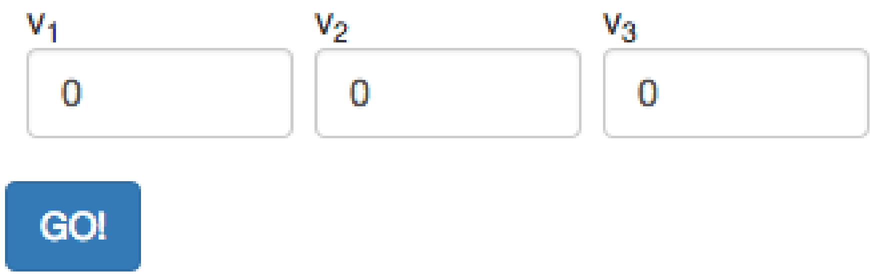
1. package.json, метаданные серверной части проекта;
2. bower.json, метаданные клиентской части проекта;
3. server.js, серверная часть проекта;
4. gulpfile.js, файл, необходимый для сборки проекта и сжатия файлов;
5. index.html, файл с версткой;
6. simplex.js, контроллер;
7. mymath.js, файл с реализацией необходимых вычислительных функций;

Интерфейс выглядит следующим образом:

1. ввод размерности и вершин симплекса;



2. ввод координат вектора;



3. ввод векторов, образующих параллелограмм;

4 Результаты счета

В этой части работы будут приведены несколько примеров, показывающих результаты вручную посчитанного алгоритма в сравнении с результатами, посчитанными программой.

4.1 Случай n = 2

Пусть $n = 2$, S - треугольник с вершинами $x^{(1)} = (1,\ 0)$, $x^{(2)} = (1/2, 1)$, $x^{(3)} = (0, 1/4)$. Тогда

$$\mathbf{A} := \begin{pmatrix} 1 & 0 & 1 \\ \frac{1}{2} & 1 & 1 \\ 0 & \frac{1}{4} & 1 \end{pmatrix}, \mathbf{A}^{-1} := \begin{pmatrix} \frac{6}{7} & \frac{2}{7} & -\frac{8}{7} \\ -\frac{4}{7} & \frac{8}{7} & -\frac{4}{7} \\ \frac{1}{7} & -\frac{2}{7} & \frac{8}{7} \end{pmatrix}.$$

Коэффициенты базисных многочленов Лагранжа составляют столбцы матрицы $\mathbf{A}^{-1}$, поэтому

$$\lambda_1(x) = \frac{6}{7}x_1 - \frac{4}{7}x_2 + \frac{1}{7}, \lambda_2(x) = \frac{2}{7}x_1 + \frac{8}{7}x_2 - \frac{2}{7}, \lambda_3 = -\frac{8}{7}x_1 - \frac{4}{7}x_2 + \frac{8}{7}. \quad (4.1)$$

Результат программы:

```
λ1 = 0.8571428571428571*x1 + 0.2857142857142857*x2 -
1.1428571428571428*x3
λ2 = -0.5714285714285714*x1 + 1.1428571428571428*x2 -
0.5714285714285714*x3
λ3 = 0.1428571428571428 5*x1 - 0.2857142857142857*x2 +
1.1428571428571428*x3
```

В соответствии с (2.1) симплекс S (с границей) задаётся системой линейных неравенств

$$\frac{6}{7}x_1 - \frac{4}{7}x_2 \geq -\frac{1}{7}, \frac{2}{7}x_1 + \frac{8}{7}x_2 \geq \frac{2}{7}, \frac{8}{7}x_1 - \frac{4}{7}x_2 \geq -\frac{8}{7}.$$

Вычисления по формуле (2.3) с учётом (2.2), (4.1) дают:

$$\frac{1}{d_1(S)} = \frac{1}{2}\left(\frac{6}{7} + \frac{2}{7} + \frac{8}{7}\right) = \frac{8}{7}, \frac{1}{d_2(S)} = \frac{1}{2}\left(\frac{4}{7} + \frac{8}{7} + \frac{4}{7}\right) = \frac{8}{7},$$

т.е. $d_1(S) = d_2(S) = 7/8$. Заметим, что совпадение $d_1(S)$ и $d_2(S)$ сразу следует из равенства длин проекций S на координатные оси – достаточно привлечь (2.6).

Результат выполнения программы:

$$d_1 = 0.875$$
$$d_2 = 0.875$$

Найдём координаты концов максимальных отрезков рассматриваемого видаю При $i = 1$ формулы (2.4) - (2.5) приводят к следующим результатам:

$$s_{11} = \frac{6/7 + 6/7}{16/7} = 1, s_{12} = \frac{2/7 + 2/7}{16/7} = \frac{1}{4}, s_{13} = \frac{8/7 - 8/7}{16/7} = 0,$$

$$y_+^{(1)} = s_{11}\, x^{(1)} + s_{12}\, x^{(2)} + s_{13}\, x^{(3)} = \left(\frac{7}{8}, \frac{1}{4}\right);$$

$$t_{11} = \frac{6/7 - 6/7}{16/7} = 0, t_{12} = \frac{2/7 + 2/7}{16/7} = 0, t_{13} = \frac{8/7 + 8/7}{16/7} = 1,$$

$$y_-^{(1)} = t_{11}\, x^{(1)} + t_{12}\, x^{(2)} + t_{13}\, x^{(3)} = \left(0, \frac{1}{4}\right).$$

Концы максимального в S отрезка, параллельного оси x_1 есть точки $y_+^{(1)} = (7/8, 1/4), y_-^{(1)} = (0, 1/4)$. При $i = 2$ аналогично получаем:

$$s_{21} = 0,\ s_{22} = 1,\ s_{23} = 0,$$

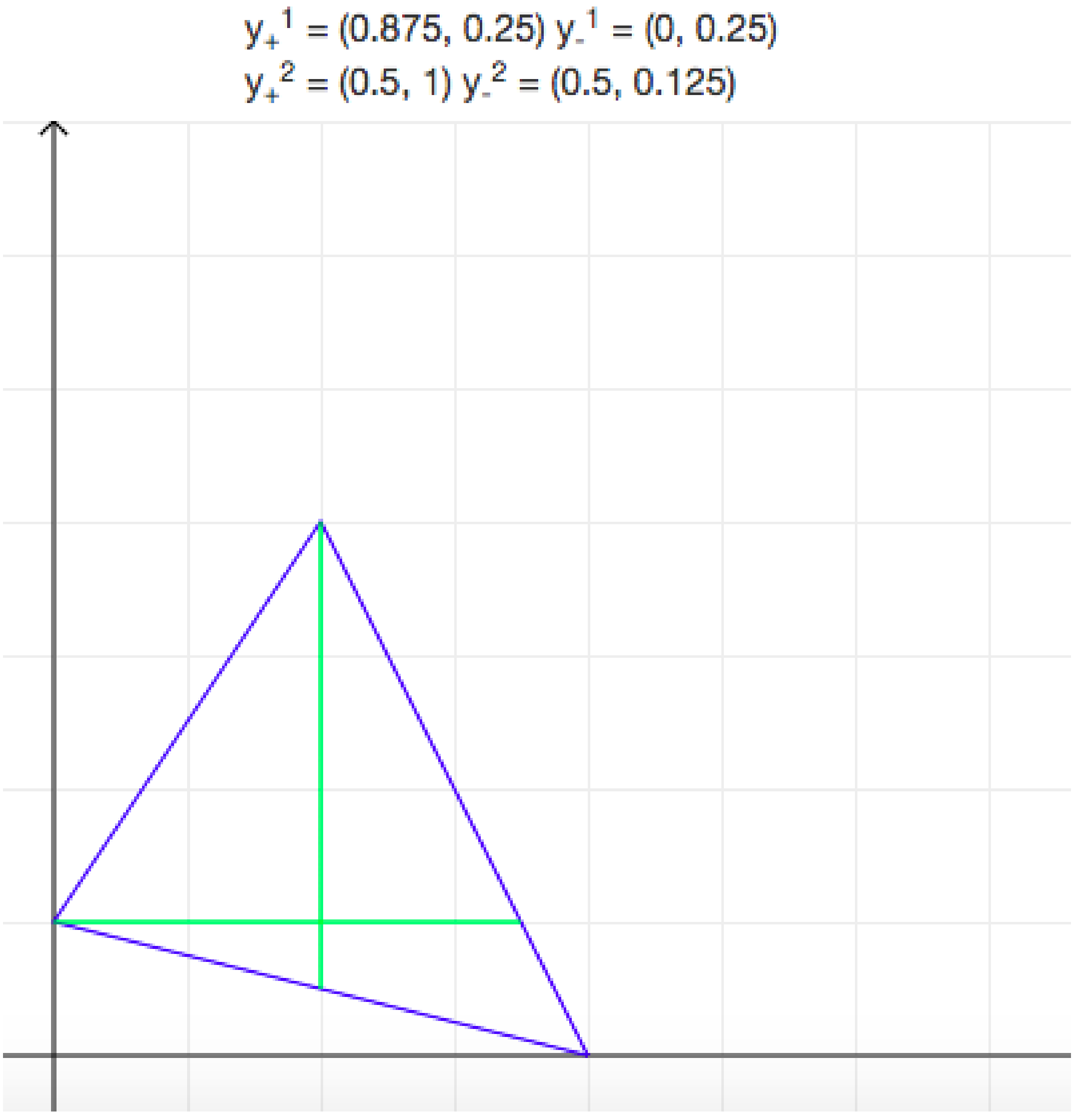
$$y_+^{(2)} = s_{21}\, x^{(1)} + s_{22}\, x^{(2)} + s_{23}\, x^{(3)} = x^{(2)} = \left(\frac{1}{2}, 1\right);$$

$$t_{21} = \frac{1}{2},\ t_{22} = 0,\ t_{23} = \frac{1}{2},$$

$$y_-^{(2)} = t_{21}\, x^{(1)} + t_{22}\, x^{(2)} + t_{23}\, x^{(3)} = \frac{1}{2}\, x^{(1)} + \frac{1}{2}\, x^{(3)} = \left(\frac{1}{2}, \frac{1}{8}\right).$$

Поэтому максимальным в S отрезком, параллельным оси x_2, является отрезок с концами $y_+^{(2)} = (1/2, 1), y_-^{(2)} = (1/2, 1/8)$.

Результат выполнения программы:



Теперь найдем $\xi(S)$. Очевидно,

$$\max_{x\in ver(Q_2)}(-\lambda_1(x)) = \frac{3}{7}, \max_{x\in ver(Q_2)}(-\lambda_2(x)) = \frac{2}{7}, \max_{x\in ver(Q_2)}(-\lambda_3(x)) = \frac{4}{7}.$$

Формула (2.8) даёт $\xi(S) = 3\cdot(4/7)+1 = 19/7$.

Найдём норму интерполяционного проектора $P : C(Q_2) \to \Pi_1(\mathbb{R}^2)$, узлы которого совпадают с вершинами S. Интерполяционная формула Лагранжа (2.9) в данном случае имеет вид

$$p(x) = Pf(x) = f_1\lambda_1(x) + f_2\lambda_2(x) + f_3\lambda_3(x).$$

Подставим сюда выражение для базисных многочленов λ_j и найдем значения p в вершинах Q_2:

$$p(0,0) = \frac{1}{7}f_1 - \frac{2}{7}f_2 + \frac{8}{7}f_3, p(1,0) = f_1,$$

$$p(0,1) = -\frac{3}{7}f_1 + \frac{6}{7}f_2 + \frac{4}{7}f_3, p(1,1) = \frac{3}{7}f_1 + \frac{8}{7}f_2 - \frac{4}{7}f_3,$$

Поэтому в соответствии с (2.10)

$$\left\|P\right\| = \max_{f_j=\pm 1} \max(\left|p(0,0)\right|, \left|p(1,0)\right|, \left|p(0,1)\right|, \left|p(1,1)\right|) = \frac{15}{7}.$$

Соотношения (2.11) принимают вид $16/7 < 19/7 = 19/7$.

Применяя (2.13) или (2.14) находим $\alpha(Q_2; S) = 16/7$.

Значения в программе:

||P|| = 2.142857142857143

α(S) = 2.2857142857142856

ξ(S) = 2.7142857142857144

2.2857142857142856 <= 2.7142857142857144 <= 2.7142857142857144

Возьмем вектор $v = (1;1)$. Найдем величины $m_1, \ldots, m_3$ с помощью формулы (2.15). Получаем

$$m_1 = \frac{2}{7}, m_2 = \frac{10}{7}, m_3 = -\frac{12}{7}$$

Далее

$$\alpha_1 = 0, \alpha_2 = 0, \alpha_3 = 1, \beta_1 = \frac{1}{6}, \beta_2 = \frac{5}{6}, \beta_3 = 0, .$$

И находим концы максимального вписанного в симплекс отрезка заданного направления

$$a = 1 * (0, \frac{1}{4}), b = \frac{1}{6} * (1,0) + \frac{5}{6} * (\frac{1}{2}, 1) = (\frac{7}{12}, \frac{5}{6},).$$

Его длина суть

$$d^v(S) = \frac{2\|v\|}{\sum_{j=1}^{n+1} |m_j|} = \frac{2\sqrt{3}}{3}.$$

Результат выполнения программы :

d^v = 0.8249579113843056}

a = (0, 0.2500000000000006)

b = (0.5833333333333334, 0.8333333333333334)

А величина $\Sigma(S, v)$ равна $\frac{3\sqrt{2}}{4}$. После выполнения программы получаем

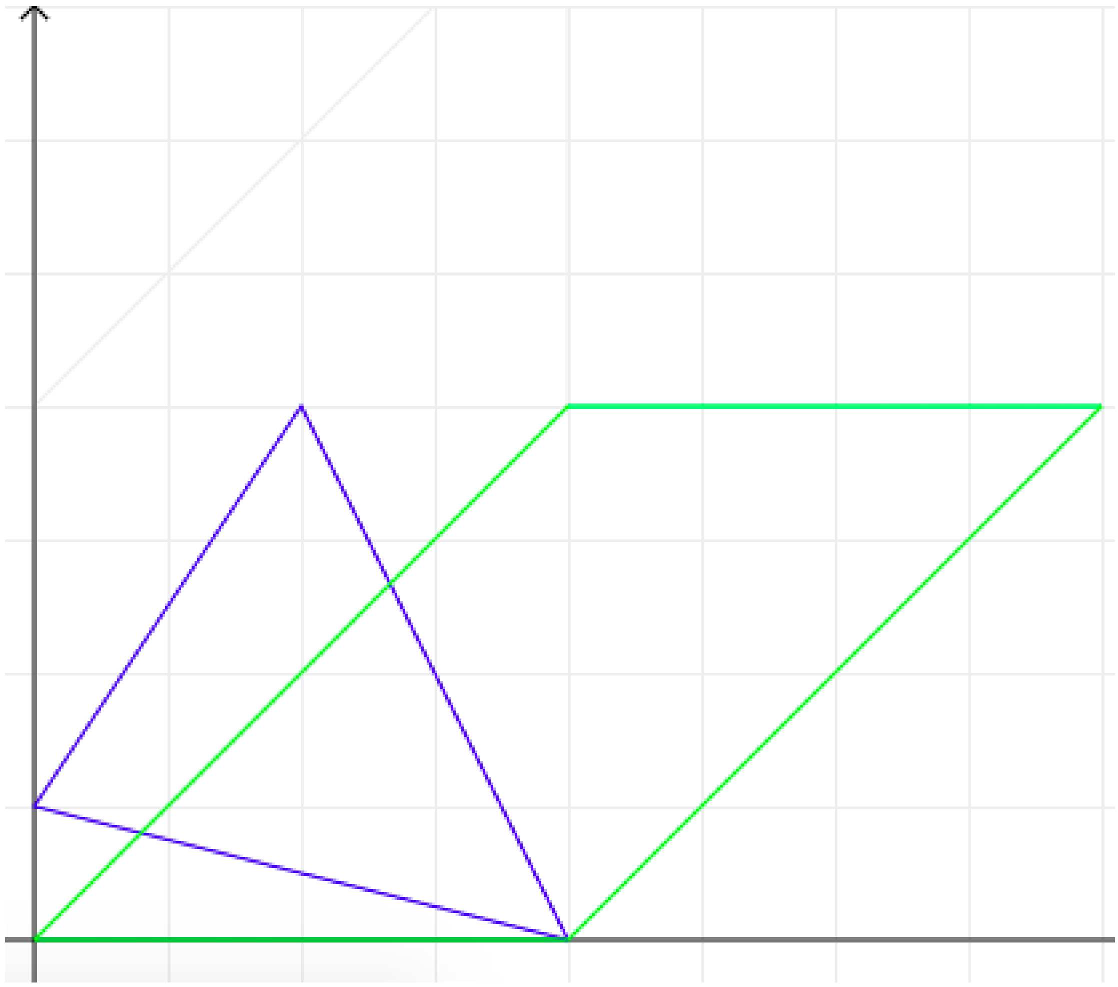
Σ(S,v) = 1.0606601717798214

Возьмем параллелограмм V, который задается векторами $v_1 = (1; 0)$, $v_2 = (1, 1)$ и посчитаем для него величину $\alpha(V, S)$ используя формулу (2.21). Т.к. v_2 есть вектор из предыдущих расчетов, а v_1 – вектор, параллельный координатной оси, то нет необходимости пересчитывать $d^{v^{(i)}}$. Получаем

$$\alpha(V, S) = \frac{1 * 8}{7} + \frac{1}{1} + \frac{12\sqrt{2}}{7\sqrt{2}} = \frac{20}{7} = M$$

Результат выполнения программы:

$\alpha(V,S) = 2.8571428571428568$



4.2 Случай n = 3

Пусть $n = 3$, S – тетраэдр с вершинами $x^{(1)} = (1,0,0), x^{(2)} = (0,1,0), x^{(3)} = (0,0,1), x^{(4)} = (1,1,1)$. Это правильный тетраэдр, вписанный в куб $Q_3 = [0,1]^3$. В рассматриваемой ситуации

$$\mathbf{A} := \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \mathbf{A}^{-1} := \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}.$$

Базисные многочлены Лагранжа –

$$\lambda_1(x) = \frac{1}{2}x_1 - \frac{1}{2}x_2 - \frac{1}{2}x_3 + \frac{1}{2}, \lambda_2(x) = -\frac{1}{2}x_1 + \frac{1}{2}x_2 - \frac{1}{2}x_3 + \frac{1}{2},$$

$$\lambda_3 = -\frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{1}{2}x_3 + \frac{1}{2}, \lambda_4 = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}.$$

Результат выполнения программы:

λ_1 = 0.5*x_1 - 0.5*x_2 - 0.5*x_3 + 0.5*x_4
λ_2 = -0.5*x_1 + 0.5*x_2 - 0.5*x_3 + 0.5*x_4
λ_3 = -0.5*x_1 - 0.5*x_2 + 0.5*x_3 + 0.5*x_4
λ_4 = 0.5*x_1 + 0.5*x_2 + 0.5*x_3 - 0.5*x_4

Поэтому симплекс S (с границей) задаётся системой линейных неравенств

$$x_1 - x_2 - x_3 \geq -1, -x_1 + x_2 - x_3 \geq -1,$$

$$-x_1 - x + 2 + x_3 \geq -1, x_1 + x_2 + x_3 \geq 1,$$

Вычислим осевые диаметры S и максимальные отрезки, параллельные координатным осям. Формула (2.3) даёт $d_1(S) = d_2(S) = d_3(S) = 1$. В программе получаем:

d_1 = 1
d_2 = 1
d_3 = 1

Факт совпадения всех $d_i(S)$ следует также из (2.6) и того, что площади проекций S на координатные плоскости одинаковы (и равны 1). Числа s_{ij} и t_{ij} оказываются следующими:

$$s_{11} = \frac{1}{2}, s_{12} = 0, s_{13} = 0, s_{14} = \frac{1}{2};$$

$$t_{11} = 0, t_{12} = \frac{1}{2}, t_{13} = \frac{1}{2}, t_{14} = 0;$$
$$s_{21} = 0, s_{22} = \frac{1}{2}, s_{23} = 0, s_{24} = \frac{1}{2};$$
$$t_{21} = \frac{1}{2}, t_{22} = 0, t_{23} = \frac{1}{2}, t_{24} = 0;$$
$$s_{31} = 0, s_{32} = 0, s_{33} = \frac{1}{2}, s_{34} = \frac{1}{2};$$
$$t_{31} = \frac{1}{2}, t_{32} = \frac{1}{2}, t_{33} = 0, t_{34} = 0.$$

Поэтому

$$y_+^{(1)} = \sum_{j=1}^{4} s_{1j}\, x^{(j)} = \frac{1}{2}\left(x^{(1)} + x^{(4)}\right) = \left(1, \frac{1}{2}, \frac{1}{2}\right),$$

$$y_-^{(1)} = \sum_{j=1}^{4} t_{1j}\, x^{(j)} = \frac{1}{2}\left(x^{(2)} + x^{(3)}\right) = \left(0, \frac{1}{2}, \frac{1}{2}\right),$$

$$y_+^{(2)} = \sum_{j=1}^{4} s_{2j}\, x^{(j)} = \frac{1}{2}\left(x^{(2)} + x^{(4)}\right) = \left(\frac{1}{2}, 1, \frac{1}{2}\right),$$

$$y_-^{(2)} = \sum_{j=1}^{4} t_{2j}\, x^{(j)} = \frac{1}{2}\left(x^{(1)} + x^{(3)}\right) = \left(\frac{1}{2}, 0, \frac{1}{2}\right),$$

$$y_+^{(3)} = \sum_{j=1}^{4} s_{3j}\, x^{(j)} = \frac{1}{2}\left(x^{(3)} + x^{(4)}\right) = \left(\frac{1}{2}, \frac{1}{2}, 1\right),$$

$$y_-^{(3)} = \sum_{j=1}^{4} t_{3j}\, x^{(j)} = \frac{1}{2}\left(x^{(1)} + x^{(2)}\right) = \left(\frac{1}{2}, \frac{1}{2}, 0\right),$$

Максимальными в S отрезками, параллельными координатным осям, оказываются отрезки единичной длины, соединяющие середины противоположных (скрещивающихся) рёбер. Например, максимальный отрезок, параллельный оси x_1, имеет концы $y_+^{(1)} = (1, 1/2, 1/2)$ и $y_-^{(1)} = (0, 1/2, 1/2)$. Указанные отрезки пересекаются в центре куба.

В программе получаем:

y_+^1 = (1, 0.5, 0.5) y_-^1 = (0, 0.5, 0.5)
y_+^2 = (0.5, 1, 0.5) y_-^2 = (0.5, 0, 0.5)
y_+^3 = (0.5, 0.5, 1) y_-^3 = (0.5, 0.5, 0)

Найдём $\xi(S)$. Нетрудно видеть, что при всех $j = 1, 2, 3, 4$

$$\max_{x \in ver(Q_3)} (-\lambda_j(x)) = \frac{1}{2},$$

Поэтому по формуле (2.8) $\xi(S) = 4 \cdot (1/2) + 1 = 3$. Пусть $P : C(Q_3) \to \Pi_1(\mathbb{R}^3)$ – интерполяционный проектор, узлы которого совпадают с вершинами S. Применяя формулу

$$p(x) = Pf(x) = f_1\lambda_1(x) + f_2\lambda_2(x) + f_3\lambda_3(x) + f_4\lambda_4(x).$$

и явный вид λ_j, найдём значения p в каждой из восьми вершин $Q_3 : p(0,0,0) = (1/2)(f_1 + f_2 + f_3 - f_4), p(1,0,0) = f_1$ и т.д. Как оказывается,

$$\|P\| = \max_{f_j = \pm 1} \max_{x \in ver(Q_n)} |p(x)| = 2.$$

Двойное неравенство (2.11) имеет форму равенства – каждая из его частей равняется 3. Применяя (2.13) или (2.14) находим $\alpha(Q_3; S) = 3$.

Выполнение программы даёт нам следующие результаты:

||P|| = 2

α(S) = 3

ξ(S) = 3

3 <= 3 <= 3

Возьмем вектор $v = (1,1,1)$. Найдем величины $m_1, \ldots, m_4$ с помощью формулы (2.15). Получаем

$$m_1 = -\frac{1}{2}, m_2 = -\frac{1}{2}, m_3 = -\frac{1}{2}, m_4 = \frac{3}{2}.$$

Далее

$$\alpha_1 = \frac{1}{3}, \alpha_2 = \frac{1}{3}, \alpha_3 = \frac{1}{3}, \alpha_4 = 0, \beta_1 = 0, \beta_2 = 0, \beta_3 = 0, \beta_4 = 1.$$

И находим концы максимального вписанного в симплекс отрезка заданного направления

$$a = \frac{1}{3}(1,0,0) + \frac{1}{3}(0,1,0) + \frac{1}{3}(0,0,1) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}), b = 1 \cdot (1,1,1) = (1,1,1).$$

Его длина суть

$$d^v(S) = \frac{2\|v\|}{\sum_{j=1}^{n+1}|m_j|} = \frac{2\sqrt{3}}{3}.$$

Результат выполнения программы:

```
dᵛ = 1.1547005383792515}

a = (0.3333333333333333, 0.3333333333333333, 0.3333333333333333)

b = (1, 1, 1)
```

А величина $\Sigma(S, v)$ равна $\frac{\sqrt{3}}{2}$. Результат выполнения программы:

```
Σ(S,v) = 0.8660254037844387
```

Возьмем параллелограмм V, который задается векторами $v_1 = (1, 1, 1)$, $v_2 = (1, 0, 0)$, $v_3 = (0, 1, 0)$ и посчитаем для него величину $\alpha(V, S)$ используя формулу (2.21). Т.к. v_1 есть вектор из предыдущих расчетов, а v_2 и v_3 параллельны координатным осям, то нет необходимости пересчитывать $d^{v^{(i)}}$. Получаем

$$\alpha(V, S) = \frac{3\sqrt{3}}{2\sqrt{3}} + \frac{1}{1} + \frac{1}{1} = \frac{7}{2} = M$$

Результат выполнения программы

```
α(V,S) = 3.5
```

5 Заключение

Подводя итог, можно сказать, что в данной работе реализованы все поставленные задачи, а именно:

- рассмотрен алгоритм нахождения осевых диаметров, максимального в симплексе отрезка данного направления и прочих велечин по заданным вершинам невырожденного симплекса, вектору и параллелограмму.

- на примерах разобраны случаи $n = 2$ (S - треугольник) и $n = 3$ (S - тетраэдр), подтверждающие корректность выполнения программы

- написана веб-программа, реализующая данные подсчеты на ЭВМ, с ипользованием популярных технологий и фрэймворков.

6 Библиография

Список литературы

[1] Невский М.В., Об одном свойстве n-мерного симплекса // Матем. заметки. 2010. Т. 87, №4. С. 580 - 593.

[2] Невский М.В., О некоторых свойствах базисных многочленов Лагранжа // Преподавание математики и компьютерных наук в классическом университете. Материалы 3-й научно-методической конференции преподавателей математического факультета и факультета информатики и вычислительной техники ЯрГУ им. П.Г.Демидова. Ярославль, 2010. С. 111 - 117.

[3] Невский М.В., Об осевых диаметрах выпуклого тела // Матем. заметки. 2011. Т. 90, №2. С. 313 - 315.

[4] Nevskii M., Properties of axial diametres of a simplex //Discrete Comput. Geom. 2011. V. 87, №2. P. 301 - 312.

[5] Невский М.В.,Вычисление максимального в сиплексе отрезка данного направления // Фундамент. и прикладная математика. 2013. Т. 18, № 2. С. 147-152. (Английский перевод: Nevskii M.V. // journal of Math Sciense. 2014. V_1 203, N 6. P. 851-854)

[6] Невский М.В., Об одной задачи для симплекса и куба в $\mathbb{R}^n$ // Модел. и анализ информ. систем. Т.20, № 3 (2013) 77 85.

[7] Толстопятенко А.А., Вычисление геометрических характеристик n-мерного симплекса. //Путь в науку. Математика: Материалы II международной молодежной научно-практической конференции /Гл. ред. Д.В. Глазков.- Ярославль ЯрГУ, 2014. с.56-59.

7 Приложение

1. Файл mymath.js (реализация необходимых для вычислений методов)

```
function transpose(a) {
    return a[0].map(function (v, i) {
        return a.map(function (r, j) {
            return a[j][i];
        });
    });
}

function getA(m) {
    for (i in m) {
        m[i].push(1);
    }
    return m;
}

function _determinant(A) {
    var N = A.length, B = [], denom = 1, exchanges = 0;
    for (var i = 0; i < N; ++i) {
        B[i] = [];
        for (var j = 0; j < N; ++j) B[i][j] = A[i][j];
    }
    for (var i = 0; i < N - 1; ++i) {
        var maxN = i, maxValue = Math.abs(B[i][i]);
        for (var j = i + 1; j < N; ++j) {
            var value = Math.abs(B[j][i]);
            if (value > maxValue) {
                maxN = j;
                maxValue = value;
            }
        }
        if (maxN > i) {
            var temp = B[i];
            B[i] = B[maxN];
            B[maxN] = temp;
            ++exchanges;
        }
        else {
            if (maxValue == 0) return maxValue;
        }
        var value1 = B[i][i];
        for (var j = i + 1; j < N; ++j) {
            var value2 = B[j][i];
            B[j][i] = 0;
            for (var k = i + 1; k < N; ++k)
                B[j][k] = (B[j][k] * value1 - B[i][k] * value2) /
    denom;
        }
```

```
        denom = value1;
    }
    if (exchanges % 2) return -B[N - 1][N - 1];
    else return B[N - 1][N - 1];
}

function _matrixCofactor(i, j, A) {
    var N = A.length, sign = ((i + j) % 2 == 0) ? 1 : -1;
    for (var m = 0; m < N; m++) {
        for (var n = j + 1; n < N; n++) A[m][n - 1] = A[m][n];
        A[m].length--;
    }
    for (var k = i + 1; k < N; k++) A[k - 1] = A[k];
    A.length--;
    return sign * _determinant(A);
}

function _adjugateMatrix(A) {
    var N = A.length, B = [], adjA = [];
    for (var i = 0; i < N; i++) {
        adjA[i] = [];
        for (var j = 0; j < N; j++) {
            for (var m = 0; m < N; m++) {
                B[m] = [];
                for (var n = 0; n < N; n++) B[m][n] = A[m][n];
            }
            adjA[i][j] = _matrixCofactor(j, i, B);
        }
    }
    return adjA;
}

function inverse(A) {
    var det = _determinant(A);
    if (det == 0) return false;
    var N = A.length, a = _adjugateMatrix(A);
    for (var i = 0; i < N; i++) {
        for (var j = 0; j < N; j++) {
            a[i][j] /= det;
            a[i][j] = a[i][j] || 0;
        }
    }
    return a;

}

function ifPointIntoSimplex(l, p) {
    var i, j, arr = [];
    for (i = 0; i < l.length; i++) {
        var t = 0;
        for (j = 0; j < l[i].length; j++) {
            t = t + l[i][j] * p[j] ? p[j] : 1;
        }
```

```
        arr.push(t);
    }
    for (i in arr) {
        if (arr[i] < 0) {
            return -1;
        }
        else if (arr[i] == 0) {
            return 0;
        }
    }
    return 1;
}

function findDiamters(l) {
    var d = [], i = 0, j = 0;
    for (i = 0; i < l.length - 1; i++) {
        d[i] = 0;
        for (j = 0; j < l[i].length; j++) {
            d[i] += Math.abs(l[i][j]);
        }
        d[i] = 2 / d[i];
    }
    return d;
}

function _findSum(l) {
    var i, j, result = [];
    for (i = 0; i < l.length - 1; i++) {
        var sum = 0;
        for (j = 0; j < l[i].length; j++) {
            sum += Math.abs(l[i][j]);
        }
        result.push(sum);
    }
    return result;
}

function _findSOrT(el, s) {
    if (s) {
        return (Math.abs(el) + el);
    }
    return (Math.abs(el) - el);
}

function endPoints(l, a) {
    var sum = _findSum(l), i, j, k, s = [],
        t = [], result = [], n = l.length;
    for (i = 0; i < n - 1; i++) {
        s[i] = [];
        t[i] = [];
        for (j = 0; j < n; j++) {
            s[i][j] = (s[i][j] || 0) + _findSOrT(l[i][j], true);
            t[i][j] = (t[i][j] || 0) + _findSOrT(l[i][j]);
```

```
        }
        s[i] = s[i].map(function (num) {
            return num / sum[i]
        });
        t[i] = t[i].map(function (num) {
            return num / sum[i]
        });
    }
    result = [[], []];
    for (k = 0; k < n-1; k++) {
        var t_arr = [];
        var s_arr = [];
        for (j = 0; j<n; j++) {
            for (i = 0; i < n-1; i++){
                s_arr[i] = (s_arr[i] || 0) + s[k][j] * a[j][i];
                t_arr[i] = (t_arr[i] || 0) + t[k][j]* a[j][i];
            }
        }
        result[0].push(s_arr);
        result[1].push(t_arr);
    }
    return result;
}

function _getQ(n, f) {
    var i, j, arr = [], m = Math.pow(2, n - 1);
    for (i = 0; i < m; i++) {
        arr.push([]);
        var m_2 = i.toString(2);
        if (m_2.length < n - 1) {
            var l = m_2.length;
            for (j = l; j < n - 1; j++) {
                m_2 = '0' + m_2;
            }
        }
        for (j = 0; j < n - 1; j++) {
            arr[i].push(f && !parseInt(m_2[j]) ? -1 :
   parseInt(m_2[j]));
        }
    }
    return arr;
}

function findKsi(l) {
    var q = _getQ(l.length), i, k, j, arr = [], lambda = [];
    for (i = 0; i < l.length; i++) {
        lambda[i] = [];
        for (j = 0; j < q.length; j++) {
            lambda[i][j] = 0;
            for (k = 0; k < l.length; k++) {
                lambda[i][j] += l[k][i] *
                (q[j] == undefined ||
                (q[j][k] == undefined) ? 1 : q[j][k])
```

```
            }
        }
    }
    for (i = 0; i < lambda.length; i++) {
        lambda[i] = lambda[i].map(function (num) {
            return -num;
        });
        arr[i] = Math.max.apply(Math, lambda[i]);
    }
    return l.length * Math.max.apply(Math, arr) + 1;
}

function findP(l) {
    var q = _getQ(l.length), i, k, j, arr = [], lambda = [];
    for (i = 0; i < l.length; i++) {
        lambda[i] = [];
        for (j = 0; j < q.length; j++) {
            lambda[i][j] = 0;
            for (k = 0; k < l.length; k++) {
                lambda[i][j] += l[k][i] * (q[j] === undefined ||
   (q[j][k] === undefined) ? 1 : q[j][k]);
            }
        }
    }
    for (i = 0; i < lambda[0].length; i++) {
        for (j = 0; j < l.length; j++)
            arr[i] = (arr[i] || 0) + Math.abs(lambda[j][i]);
    }
    return Math.max.apply(Math, arr);
}

function findAlpha(d) {
    var i, a = 0;
    for (i = 0; i < d.length; i++) {
        a += 1 / d[1];
    }
    return a;
}

function _findM(l, v) {
    var i, j, m = [];
    for (i = 0; i < l.length; i++) {
        m[i] = 0;
        for (j = 0; j < v.length; j++) {
            m[i] += v[j] * l[j][i];
        }
    }
    return m;
}

function _findMSum(m) {
    for (var s = 0, k = m.length; k; s += Math.abs(m[--k]));
    return s;
```

```
}

function vectorEndPoints(a, l, v) {
    var m = _findM(l, v), sum = _findMSum(m), i, j,
        k, alpha = [], beta = [], result = [[], []],
        n = l.length;
    for (i = 0; i < n; i++) {
        alpha[i] = _findSOrT(m[i]) / sum;
        beta[i] = _findSOrT(m[i], true) / sum;
    }
    for (k = 0; k < n - 1; k++) {
        for (j = 0; j < n; j++) {
            result[0][k] = (result[0][k] || 0) + alpha[j] * a[j][k];
            result[1][k] = (result[1][k] || 0) + beta[j] * a[j][k];
        }
    }
    return result;
}

function _findVectorNorm(v) {
    var sum = 0;
    for (var i = 0; i < v.length; i++) {
        sum += v[i] * v[i];
    }
    return Math.sqrt(sum);
}

function findVectorD(l, v) {
    var m = _findM(l, v), sum = _findMSum(m),
        norm = _findVectorNorm(v);
    return (2 * norm) / sum;
}

function findVectorKsi(l, v, a) {
    var m = _findM(l, v), sum = _findMSum(m),
        norm = _findVectorNorm(v), det = _determinant(a),
        f = [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880,
   3628800, 39916800, 479001600];
    return (Math.abs(det) * sum) / (norm * 2 * f[v.length - 1]);
}

function findVAlpha(l, v) {
    var i, j, result = 0;
    for (i = 0; i < v.length; i++) {
        result += _findVectorNorm(v[i]) / findVectorD(l, v[i]);
    }
    return result;
}

function findX(l, a) {
    var i, j, k, sum = 0, result = [];
    for (i = 0; i < l.length; i++) {
        for (j = 0; j < l.length - 1; j++) {
```

```
            sum += Math.abs(l[j][i]);
        }
    }
    for (k = 0; k < l.length - 1; k++) {
        result[k] = 0;
        for (i = 0; i < l.length; i++) {
            for (j = 0; j < l.length - 1; j++) {
                result[k] += Math.abs(l[j][i]) * a[i][k];
            }
        }
    }
    result = result.map(function (num) {
        return num / (sum - 2)
    });
    return result;
}

function findBarycentricCoords(l, x) {
    var i, j, result = [];
    for (i = 0; i < l.length; i++) {
        result[i] = 0;
        for (j = 0; j < l[i].length; j++) {
            result[i] += l[j][i] * (x[j] == undefined ? 1 : x[j]);
        }
    }
    return result;
}
```

2. Файл server.js (серверная часть)

```
var express = require('express'),
    server = express();

server.use(express.static(__dirname+'/static'));
server.listen(1337);
```

3. Файл simplex.js (контроллер)

```
/**
 * Created by Anastasia on 28/01/15.
 */

var smplxApp = angular.module('smplxApp', ['ngSanitize']);


smplxApp.controller('ShowInputFields', ['$scope', function($scope) {
    $scope.smplxDim = 2;
    //renewFields($scope);
    $scope.$watch('smplxDim', function() {
        renewFields();
        $scope.vectorResult = false;
        $scope.vResult = false;
        return $scope.polynomsResult = false;
```

```
});

$scope.calculate = function() {
    calculate($scope);
};

$scope.calculateVector = function() {
    calculateVector($scope);
};

$scope.calculateV = function() {
    calculateV($scope);
};

function renewFields() {
    var dim = $scope.smplxDim,
        vertices = [],
        vertex = [],
        vector = [],
        v = [];
        i = 0, j = 0;
    for (i = 0; i <= dim; i++) {
        v[i] = [];
        vertex = new Array(dim);
        for (j = 0; j < dim; j++) {
            vector[j] = 0;
            vertex[j] = 0;
            v[i][j] = 0;
        }
        vertices.push(vertex);
    }
    v.pop();
    $scope.vertices = vertices;
    $scope.vector = vector;
    $scope.v = v;
}
function calculate($scope) {
    var a = getA(angular.copy($scope.vertices)), i = 0;
    var l = inverse(a);
    console.table(l);
    var d = findDiamters(l);
    var end_points = endPoints(l, a);
    var ksi = findKsi(l);
    var p = findP(l);
    var alpha = findAlpha(d);
    $scope.polynomsResult = {
      polynoms: l.map(function(e) {
        return renderResult(e).join('');
      }),
      d: d,
      endPoints: end_points,
      alpha: alpha,
      ksi: ksi,
```

```
    p: p,
    something: (p - 1) * ($scope.smplxDim + 1) / 2 + 1
  };
  if ($scope.smplxDim == 2) {
      var cnv = document.getElementById('myTriangle');
      var ctx = cnv.getContext('2d');
      var width = cnv.width;
      var height = cnv.height;
      ctx.clearRect ( 0 , 0 , width, height);
      for (var x = 0.5; x < width; x += 50) {
        ctx.moveTo(x, 0);
        ctx.lineTo(x, height);
      }
      for (var y = 0.5; y < height; y += 50) {
        ctx.moveTo(0, y);
        ctx.lineTo(width, y);
      }
      ctx.strokeStyle = "#eee";
      ctx.stroke();


      ctx.beginPath();
      ctx.moveTo(0, height - 25);
      ctx.lineTo(width, height - 25);
      ctx.moveTo(width - 5, height - 20);
      ctx.lineTo(width, height - 25);
      ctx.lineTo(width - 5, height - 30);

      ctx.moveTo(50, height);
      ctx.lineTo(50, 0);
      ctx.moveTo(55, 5);
      ctx.lineTo(50, 0);
      ctx.lineTo(45, 5);
      ctx.strokeStyle = "#000";
      ctx.stroke();

      function getXCoord (x) {
          return x*200 + 50;
      };
      function getYCoord (y) {
          return height - 25 - y*200;
      };
      ctx.beginPath();
      ctx.moveTo(getXCoord(a[0][0]), getYCoord(a[0][1]));
      ctx.lineTo(getXCoord(a[1][0]), getYCoord(a[1][1]));
      ctx.lineTo(getXCoord(a[2][0]), getYCoord(a[2][1]));
      ctx.lineTo(getXCoord(a[0][0]), getYCoord(a[0][1]));
      ctx.strokeStyle = "#00f";
      ctx.stroke();
      ctx.beginPath();
      ctx.moveTo(getXCoord(end_points[0][0][0]),
getYCoord(end_points[0][0][1]));
      ctx.lineTo(getXCoord(end_points[1][0][0]),
```

```
getYCoord(end_points[1][0][1]));
          ctx.moveTo(getXCoord(end_points[0][1][0]),
getYCoord(end_points[0][1][1]));
          ctx.lineTo(getXCoord(end_points[1][1][0]),
getYCoord(end_points[1][1][1]));
          ctx.strokeStyle = "#0f0";
          ctx.stroke();
     }

     console.table(d);
     console.table(end_points);
     console.log(alpha, ksi, (p-1) * ($scope.smplxDim + 1)/2 +
1);

 }
 function calculateVector($scope) {
     var a = getA(angular.copy($scope.vertices)),
         v = angular.copy($scope.vector),
         l = inverse(a), d, ends, ksi;
     d = findVectorD(l, v);
     ends = vectorEndPoints(a, l, v);
     ksi = findVectorKsi(l ,v, a);
     console.table(d);
     console.table(ends);
     console.log(ksi);
     $scope.vectorResult = {
         d: d,
         endPoints: ends,
         ksi: ksi
     };
     if ($scope.smplxDim == 2) {
         var cnv = document.getElementById('myVector');
         var ctx = cnv.getContext('2d');
         var width = cnv.width;
         var height = cnv.height;
         ctx.clearRect ( 0 , 0 , width, height);
         for (var x = 0.5; x < width; x += 50) {
           ctx.moveTo(x, 0);
           ctx.lineTo(x, height);
         }
         for (var y = 0.5; y < height; y += 50) {
           ctx.moveTo(0, y);
           ctx.lineTo(width, y);
         }
         ctx.strokeStyle = "#eee";
         ctx.stroke();


         ctx.beginPath();
         ctx.moveTo(0, height - 25);
         ctx.lineTo(width, height - 25);
         ctx.moveTo(width - 5, height - 20);
         ctx.lineTo(width, height - 25);
```

```
            ctx.lineTo(width - 5, height - 30);

            ctx.moveTo(50, height);
            ctx.lineTo(50, 0);
            ctx.moveTo(55, 5);
            ctx.lineTo(50, 0);
            ctx.lineTo(45, 5);
            ctx.strokeStyle = "#000";
            ctx.stroke();

            function getXCoord (x) {
                return x*200 + 50;
            };
            function getYCoord (y) {
                return height - 25 - y*200;
            };

            ctx.beginPath();
            ctx.moveTo(getXCoord(a[0][0]), getYCoord(a[0][1]));
            ctx.lineTo(getXCoord(a[1][0]), getYCoord(a[1][1]));
            ctx.lineTo(getXCoord(a[2][0]), getYCoord(a[2][1]));
            ctx.lineTo(getXCoord(a[0][0]), getYCoord(a[0][1]));
            ctx.strokeStyle = "#00f";
            ctx.stroke();
            ctx.beginPath();
            ctx.moveTo(getXCoord(0), getYCoord(0));
            ctx.lineTo(getXCoord(v[0]), getYCoord(v[1]));
            ctx.strokeStyle = "#0f0";
            ctx.stroke();
            ctx.beginPath();
            ctx.moveTo(getXCoord(ends[0][0]),
getYCoord(ends[0][1]));
            ctx.lineTo(getXCoord(ends[1][0]),
getYCoord(ends[1][1]));
            ctx.strokeStyle = "#f00";
            ctx.stroke();
        }

 }
 function calculateV($scope) {
        var a = getA(angular.copy($scope.vertices)),
            v = angular.copy($scope.v),
            l = inverse(a), alpha, x, barC;
        alpha = findVAlpha(l, v);
        $scope.vResult = {
            alpha: alpha
        };
        if ($scope.smplxDim == 2) {
            var cnv = document.getElementById('myV');
            var ctx = cnv.getContext('2d');
            var width = cnv.width;
            var height = cnv.height;
            ctx.clearRect ( 0 , 0 , width, height);
```

```
        for (var x = 0.5; x < width; x += 50) {
            ctx.moveTo(x, 0);
            ctx.lineTo(x, height);
        }
        for (var y = 0.5; y < height; y += 50) {
            ctx.moveTo(0, y);
            ctx.lineTo(width, y);
        }
        ctx.strokeStyle = "#eee";
        ctx.stroke();


        ctx.beginPath();
        ctx.moveTo(0, height - 25);
        ctx.lineTo(width, height - 25);
        ctx.moveTo(width - 5, height - 20);
        ctx.lineTo(width, height - 25);
        ctx.lineTo(width - 5, height - 30);

        ctx.moveTo(50, height);
        ctx.lineTo(50, 0);
        ctx.moveTo(55, 5);
        ctx.lineTo(50, 0);
        ctx.lineTo(45, 5);
        ctx.strokeStyle = "#000";
        ctx.stroke();

        function getXCoord (x) {
            return x*200 + 50;
        };
        function getYCoord (y) {
            return height - 25 - y*200;
        };

        ctx.beginPath();
        ctx.moveTo(getXCoord(a[0][0]), getYCoord(a[0][1]));
        ctx.lineTo(getXCoord(a[1][0]), getYCoord(a[1][1]));
        ctx.lineTo(getXCoord(a[2][0]), getYCoord(a[2][1]));
        ctx.lineTo(getXCoord(a[0][0]), getYCoord(a[0][1]));
        ctx.strokeStyle = "#00f";
        ctx.stroke();
        ctx.beginPath();
        ctx.moveTo(getXCoord(v[0][0]), getYCoord(v[0][1]));
        ctx.lineTo(getXCoord(0), getYCoord(0));
        ctx.lineTo(getXCoord(v[1][0]), getYCoord(v[1][1]));
        ctx.lineTo(getXCoord(v[0][0] + v[1][0]),
getYCoord(v[0][1] + v[1][1]));
        ctx.lineTo(getXCoord(v[0][0]), getYCoord(v[0][1]));
        ctx.strokeStyle = "#0f0";
        ctx.stroke();
    }
    x = findX(l, a);
    barC = findBarycentricCoords(l, x);
```

```
    }

     renderResult = function(a) {
      return a.map(function(e, index) {
        if (index === 0) {
          return e + "*x<sub>" + (index + 1) + "</sub>";
        } else {
          if (e >= 0) {
            return " + " + e + "*x<sub>" + (index + 1) + "</sub>";
          } else {
            return " - " + (Math.abs(e)) + "*x<sub>" + (index + 1)
  + "</sub>";
          }
        }
      });
    };
    }
]);
```

4. Файл gulpfile.js (сборка проекта и сжатие файлов)

```
var gulp = require('gulp'),
    uglify = require('gulp-uglify'),
    gulpFilter = require('gulp-filter'),
    minHtml = require('gulp-minify-html'),
    minCss = require('gulp-minify-css'),
    bower = require('gulp-bower'),
    dirName = './static/assets';

var filterJs = gulpFilter(['**/*.js', '!**/*.min.js']),
    filterCss = gulpFilter('**/*.css');

gulp.task('bower', function() {
    return bower('./bower_components')
        .pipe(filterJs)
        .pipe(uglify())
        .pipe(filterJs.restore())

        .pipe(filterCss)
        .pipe(minCss({}))
        .pipe(filterCss.restore())
        .pipe(gulp.dest(dirName));
});

gulp.task('css', function() {
    gulp.src('./client/css/*.css')
        .pipe(minCss({keepBreaks:true}))
        .pipe(gulp.dest(dirName));
});

gulp.task('js', function() {
    gulp.src('./client/js/*.js')
        .pipe(uglify())
```

```
        .pipe(gulp.dest(dirName));
});

gulp.task('html', function() {
    gulp.src('./client/*.html')
        .pipe(minHtml({}))
        .pipe(gulp.dest('./static'));
});

gulp.task('watch', function() {
    gulp.watch('./client/js/*.js', ['js']);
    gulp.watch('./client/css/*.css', ['css']);
    gulp.watch('./client/*.html', ['html']);
});

gulp.task('default', ['bower', 'css', 'js', 'html', 'watch']);
```

5. Файл package.json (метаданные серверной части)

```
{
  "name": "simplex",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "bower": "^1.3.12",
    "gulp": "^3.8.10",
    "gulp-bower": "0.0.10"
  },
  "dependencies": {
    "express": "^4.11.1",
    "mathjs": "^1.2.0"
  }
}
```

6. Файл bower.json (метаданные клиентской части)

```
{
  "name": "simplex",
  "version": "0.0.0",
  "authors": [
    "Anastasia <anastasia.tolstopyatenko@gmail.com>"
  ],
  "moduleType": [
    "node"
  ],
  "license": "MIT",
```

```
  "homepage": "index.html",
  "private": true,
  "ignore": [
    "**/.*",
    "node_modules",
    "bower_components",
    "test",
    "tests"
  ],
  "dependencies": {
    "bootstrap": "~3.3.2",
    "angular": "~1.3.11",
    "mathjs": "~1.4.0",
    "angular-sanitize": "~1.3.15"
  }
}
```

7. Файл index.html (интерфейс)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8">
  <meta http-equiv="Content-Style-Type" content="text/css">
  <title>My test</title>
  <meta name="Generator" content="Cocoa HTML Writer">
  <meta name="CocoaVersion" content="1347.57">
</head>
<body>
<p class="p1"><span class="s1">Simplex dimension:<span
    class="Apple-converted-space">В </span></span></p>
<p class="p2"><span class="s2">x</span><span
    class="s1"><sup>({{$index+1}})</sup></span></p>
<p class="p1"><span class="s1"><br>
</span></p>
<p class="p3"><span class="s3">GO!</span><span class="s2"><span
    class="Apple-converted-space">В </span></span></p>
<p class="p2"><span class="s2">O»</span><span
    class="s1"><sub>{{$index+1}}</sub></span><span class="s2">
    =<span class="Apple-converted-space">В </span></span></p>
<p class="p1"><span class="s1">d</span><span
    class="s4"><sub>{{$index+1}}</sub></span><span class="s1"> =
    {{value}}<span class="Apple-converted-space">В </span></span></p>
<p class="p1"><span class="s1">y</span><span
    class="s4"><sub>+</sub><sup>{{$index+1}}</sup></span><span
    class="s1"> = ({{polynomsResult.endPoints[0][$index].join(',
    ')}}) y</span><span
    class="s4"><sub>-</sub><sup>{{$index+1}}</sup></span><span
    class="s1"> = ({{polynomsResult.endPoints[1][$index].join(',
    ')}})<span class="Apple-converted-space">В </span></span></p>
<p class="p4"><span class="s1">||P|| =
```

```
    {{polynomsResult.p}}</span></p>
<p class="p4"><span class="s1">a(S) =
    {{polynomsResult.alpha}}</span></p>
<p class="p4"><span class="s1">Os(S) =
    {{polynomsResult.ksi}}</span></p>
<p class="p4"><span class="s1">{{polynomsResult.alpha}} &lt;=
    {{polynomsResult.ksi}} &lt;=
    {{polynomsResult.something}}</span></p>
<p class="p2"><span class="s2">v</span><span
    class="s1"><sub>{{$index+1}}</sub></span><span class="s2"><span
    class="Apple-converted-space">B </span></span></p>
<p class="p3"><span class="s3">GO!</span><span class="s2"><span
    class="Apple-converted-space">B </span></span></p>
<p class="p4"><span class="s1">d</span><span
    class="s4"><sup>v</sup></span><span class="s1"> =
    {{vectorResult.d}}</span></p>
<p class="p4"><span class="s1">a =
    ({{vectorResult.endPoints[0].join(', ')}})</span></p>
<p class="p4"><span class="s1">b =
    ({{vectorResult.endPoints[1].join(', ')}})</span></p>
<p class="p4"><span class="s1">OJ(S,v) =
    {{vectorResult.ksi}}</span></p>
<p class="p2"><span class="s2">v</span><span
    class="s1"><sup>({{$index+1}})</sup></span></p>
<p class="p1"><span class="s1"><br>
</span></p>
<p class="p3"><span class="s3">GO!</span><span class="s2"><span
    class="Apple-converted-space">B </span></span></p>
<p class="p4"><span class="s1">a(V,S) = {{vResult.alpha}}</span></p>
</body>
</html>
```

8. Файл simplex.css (стили для интерфейса)

```
.dim-div {
    width: 200px;
    margin-top: 16px;
}
.dim-div .dim-input {
    width: 30%;
    display: inline-block;
    margin-left: 8px;
}
.ver-div, .btn-go{
    margin-top: 16px;
}

.ver-div .coord-div {
    display: inline-block;
    margin-left: 8px;
    min-width: 100px;
}
p.p1 {margin: 0.0px 0.0px 0.0px 0.0px; font: 12.0px Times; color:
```

```
    #000000; -webkit-text-stroke: #000000}
p.p2 {margin: 0.0px 0.0px 0.0px 0.0px; font: 8.0px Times; color:
    #000000; -webkit-text-stroke: #000000}
p.p3 {margin: 0.0px 0.0px 0.0px 0.0px; font: 11.0px 'Helvetica
    Neue'; color: #000000; -webkit-text-stroke: #000000}
p.p4 {margin: 0.0px 0.0px 12.0px 0.0px; font: 12.0px Times; color:
    #000000; -webkit-text-stroke: #000000}
span.s1 {font-kerning: none}
span.s2 {font: 12.0px Times; font-kerning: none}
span.s3 {font-kerning: none; background-color: #c0c0c0}
span.s4 {font: 8.0px Times; font-kerning: none}
```