Министерство науки и высшего образования Российской Федерации
Ярославский государственный университет им. П. Г. Демидова
Кафедра математического анализа

А. Ю. Ухалов

ПРАКТИКУМ ПО WOLFRAM MATHEMATICA

Практикум

Ярославль
ЯрГУ
2020

УДК 004.4(076.5)
ББК З973.2-018.2я73
У89

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2020 года

Рецензент
кафедра математичекого анализа
Ярославского государственного университета
им. П. Г. Демидова

Ухалов Алексей Юрьевич.

У89 Практикум по Wolfram Mathematica : практикум / А. Ю. Ухалов ; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2020. – 40 с.

В практикуме приведены примеры решения некоторых здач анализа и вычислительной геометрии при помощи системы компьютерной математики Wolfram Mathematica. В частности, рассмотрены задачи на отыскание экстремума функции многих переменных и задачи о вычислении объемов многомерных тел. Все примеры подробно прокомментированы. Приведены задания для самостоятельного решения.

Практикум предназначен для студентов, изучающих дисциплины «Компьютерная алгебра» и «Математический анализ».

УДК 004.4(076.5)
ББК З973.2-018.2я73

Оглавление

Введение

Система компьютерной математики Wolfram Mathematica широко распространена и используется уже более 30 лет. Популярности этой программы в значительной мере способствует удачный графический интерфейс. Панели инструментов с интуитивно-понятными пиктограммами позволяют выполнять даже довольно сложные вычисления в режиме диалога, не прибегая к программированию. Опыт преподавания данной системы показывает, что студенты уже на первом занятии начинают успешно использовать Wolfram Mathematica «в режиме калькулятора» (вычисляют интегралы, строят графики функций и т. д.). Упрощает работу с системой и встроенный учебник по Wolfram Language и встроенным функциям, где почти всегда удается найти подходящий пример, который можно доработать для решения требуемой задачи. При этом можно практически ничего не знать о языке Wolfram Language и возможностях, которые открывает его применение.

Данный практикум предназначен для пользователей, которые уже имеют опыт работы с системой Wolfram Mathematica, но хотели бы пойти несколько дальше – научиться писать небольшие программы на внутреннем языке системы для решения математических задач.

Предполагается, что читатель уже знаком с основами языка Wolfram Language. Для начального знакомства с системой существует большое количество пособий. Некоторые из таких книг приводятся в списке литературы (см. [1–4]). Не следует забывать также и про большое количество учебных пособий, доступных в сети Internet. Для начального знакомства с системой и для поиска справочной информации вполне достаточно материалов, доступных on line. Ответы на многие практические вопросы можно легко найти с помощью поисковых систем. На специализированных форумах автору неоднократно удавалось найти обсуждение особенностей системы, недостаточно ясно изложенных в официальной документации. По этим причинам представляется нецелесообразным приводить в практикуме справочные данные и подробное описание параметров функций. Главное внимание уделено

вопросам применения языка Wolfram Language для решения прикладных задач.

Выбор рассмотренных задач продиктован в основном интересами и опытом автора. В 2016–2020 годы автором совместно с М. В. Невским выполнен ряд исследований по теории линейной интерполяции и геометрии выпуклых тел. Результаты этих работ частично изложены в учебном пособии [6]. Для использования практикума знакомство с пособием не является обязательным, так как здесь приводятся все нужные определения и формулы. В то же время читатели пособия [6] могут рассматривать данный практикум как своего рода «компьютерное дополнение». В практикуме приводятся и комментируются упрощенные версии программ, разработанных для реальных научных исследований. Некоторые программы, не вошедшие в практикум, а также подготовленные нами наборы данных можно найти в архиве mendeley.com [14, 15].

Разумеется, материал, изложенный в практикуме, может быть полезен и при решении совсем других задач. Необходимость найти максимум или минимум сложной функции может возникнуть в самых разных областях математики и ее приложений.

В последней части практикума приводятся задания для самостоятельного решения. Некоторые из этих задач могут быть использованы в качестве тем для курсовых и выпускных квалификационных работ. Некоторые из предлагаемых задач до сих пор не имеют полного решения. Освоение изложенного материала может послужить основой для дальнейшей научной работы.

Автор надеется, что данное издание будет полезно читателям при использовании системы Wolfram Mathematica в собственных проектах.

Программы, описанные в пособии, хранятся в архиве `https://notebookarchive.org/`
Ссылки для скачивания кода приведены в конце соответствующих пунктов.

Глава 1

Экстремальные задачи

1. Общие замечания

Для решения задач оптимизации в системе Wolfram Mathematica имеется большой набор функций. В данном пособии мы ограничиваемся использованием функции численной минимизации NMinimize. Существует символьный вариант этой функции — Minimize, предназначенный для отыскания точного или символьного минимума. Однако найти точное значение в символьной форме удается далеко не всегда. Для рассматриваемых в данном практикуме задач нам удалось найти символьное решение только в некоторых простейших случаях. Мы ограничимся рассмотрением численной минимизации. Общий формат вызова функции имеет вид

```
NMinimize[{f(x1,...,xn), cond}, {x1,...,xn}, opt]
```

где
f(x1,...,xn) — целевая функция n переменных;
cond — логическое выражение, описывающее условия, налагаемые на переменные x1, ..., xn (область значений переменных);
{x1,...,xn} — перечень переменных, по которым производится минимизация;
opt — опции функции. С набором доступных опций можно познакомиться по документации.
В любом случае имеет смысл сначала попробовать применить набор опций выбираемых по умолчанию. Мы упомянем здесь только опцию Method, выбор которой обычно сильно влияет на качество получаемого результата. В текущей версии системы доступны следующие методы: “NelderMead”, “DifferentialEvolution”, “SimulatedAnnealing” и “RandomSearch”. У каждого из этих методов имеется свой набор дополнительных опций, позволяющий настраивать параметры метода. Метод обычно подбирается эксперименталь-

но. Так, например, для задач, описанных в п. 3., метод DifferentialEvolution показывал явно худшие результаты, чем остальные методы. Лучшие из результатов были получены с применением метода SimulatedAnnealing. Однако здесь ничего нельзя утверждать однозначно. Автор не проводил специальных исследований, посвященных сравнению методов.

Хотя функция NMinimize использует численные методы минимизации функции многих переменных, следует понимать, что она не использует передаваемую ей функцию как «черный ящик», многократно выполняя ее вычисление для различных числовых значений переменных. При вызове NMinimize целевая функция вызывается один раз как символьное выражение, производится анализ этой функции и ее вычисление оптимизируется для соответствующего численного алгоритма. В этом состоит принципиальное отличие Wolfram Mathematica от распространенных библиотек численных методов для языков типа C/C++, Pyton и др. Попытка передать функции NMinimize функцию, тело которой содержит циклы, условные выражения и другие конструкции, способ вычисления которых зависит от значения переменных, как правило, приводит к ошибкам при отыскании минимума. При этом никаких предупреждений не выдается, пользователь просто получает неверный результат. Написанные для решения важных задач программы рекомендуется тщательно тестировать на задачах с известным ответом.

В силу указанной особенности, для решения задач численной оптимизации на Wolfram Mathematica требуется построить выражение для целевой функции в явном виде. Это часто удобнее всего сделать, используя возможности символьных вычислений системы. Иногда полученное выражение оказывается очень громоздким. Следует избегать вывода больших выражений в документ Wolfram Notebook, т. к. процедура вывода и форматирования длинных выражений занимает значительное время и расходует вычислительные ресурсы системы. Само выражение при этом может быть сгенерировано довольно быстро. В некоторых случаях оказывается удобным вообще не создавать функцию в явном виде, а написать код, который возвращает целевую функцию в качестве результата работы. Этот результат можно передать, например, в функцию NMinimize. Такой подход обычно требует использования средств функционального программирования Wolfram Language.

2. Кратчайшее расстояние между кривыми

Задача. *Найти кратчайшее расстояние между двумя эллипсами. Изобразить на рисунке эллипсы и отрезок, соединяющий их ближайшие точки.*

Для решения данной задачи удобно воспользоваться параметрическими уравнениями эллипсов. Можно считать, что один из эллипсов приведен к главным осям, т. е. описывается уравнениями

$$\frac{x^2}{a_1^2}+\frac{y^2}{b_1^2}=1 \quad \text{или} \quad \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} a_1\cos v \\ b_1 \sin v \end{pmatrix},\ 0 \le u \le 2\pi.$$

Будем предполагать, что второй эллипс находится в произвольном положении и задан параметрическими уравнениями

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} a_2\cos v \\ b_2 \sin v \end{pmatrix}, \qquad 0 \le v \le 2\pi.$$

Очевидно, для отыскания ближайших точек на кривых достаточно найти, при каких u и v достигается минимум длины вектора

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}.$$

Система Wolfram Mathematica имеет встроенные функции для вычисления длины вектора (Norm) и для отыскания минимума функции многих переменных (NMinimize). Приведем программу для решения поставленной задачи.

```
a1 = 3;
b1 = 1;
a2 = 2;
b2 = 0.5;
prange = {{-6, 6}, {-6, 6}};
Manipulate[
  cen2 = {cX, cY};
 M = RotationMatrix[phi];
 e1 = ParametricPlot[{a1*Cos[u], b1*Sin[u]},
        {u, 0, 2*Pi}, PlotStyle -> Black,
         PlotRange -> prange];
 e2 = ParametricPlot[cen2 + M.{a2*Cos[v], b2*Sin[v]},
            {v, 0, 2*Pi},
```

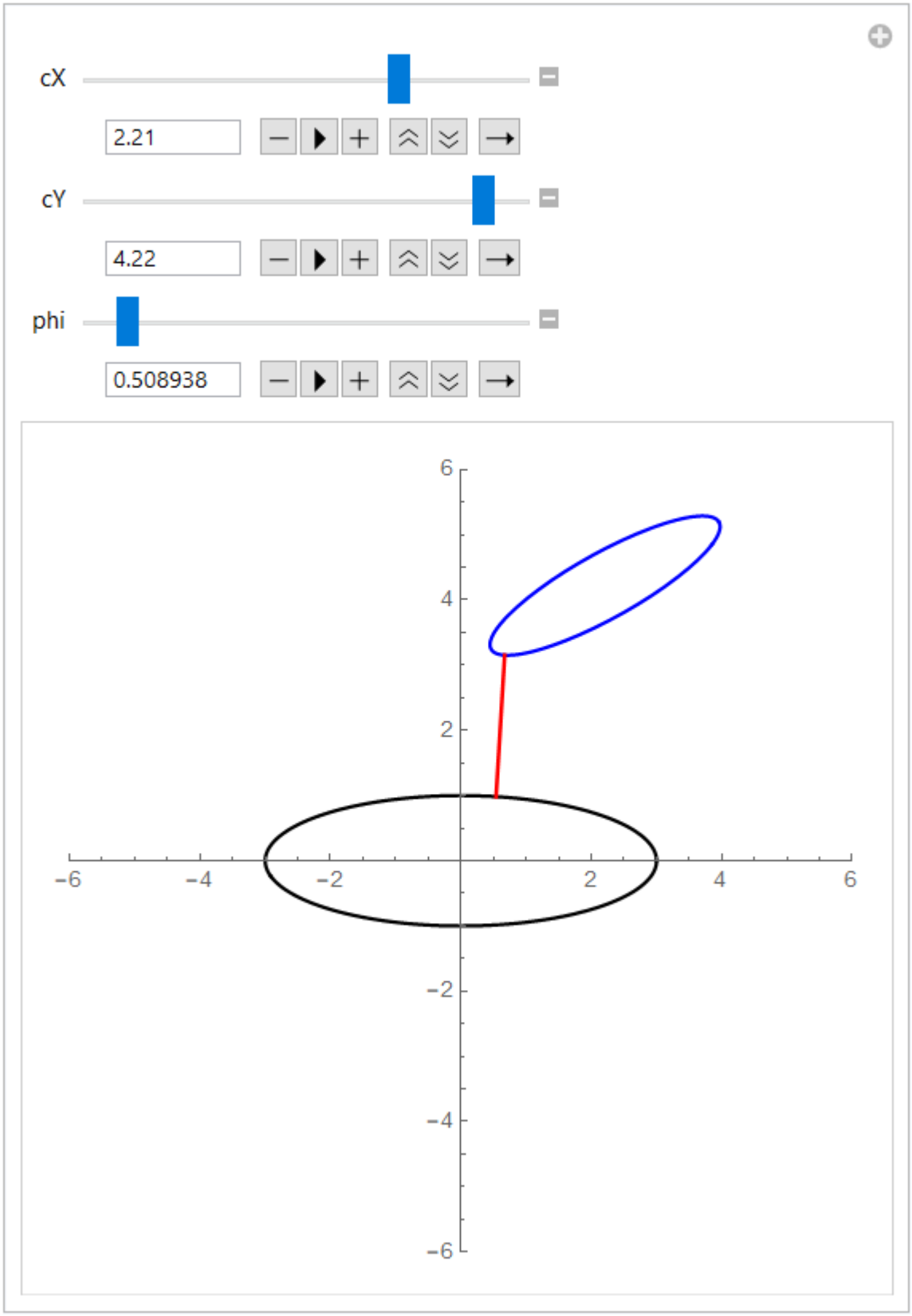

Рис. 1.1: Нахождение кратчайшего расстояния между эллипсами

```
14                PlotStyle -> Blue, PlotRange -> prange];
15    minres =
16     NMinimize[{
17       Norm[{a1*Cos[u], b1*Sin[u]} -
18       (cen2 + M.{a2*Cos[v], b2*Sin[v]})],
19       0 <= u <= 2*Pi && 0 <= v <= 2*Pi}, {u, v},
20      Method -> "RandomSearch"];
21    p1 = {a1*Cos[u], b1*Sin[u]} /. minres[[2]];
22    p2 = cen2 + M.{a2*Cos[v], b2*Sin[v]} /. minres[[2]];
23    lin = Graphics[{Red, Thickness[0.005],
24                Line[{p1, p2}]}];
25    Show[e1, e2, lin, AspectRatio -> 1],
26    {cX, -5, 5},
27    {cY, -5, 5},
28    {phi, 0, 2*Pi},
29    TrackedSymbols :> {cX, cY, phi}
30    ]
```

Результат работы программы показан на рис. 1.1.

Комментарии к коду

Строки 1 – 4. Задание значений a_1, b_1, a_2, b_2.

Строка 5. Значение переменной prange – область вывода для функции ParametricPlot, используемой для рисования эллипсов. Удобно сделать эту область одинаковой для всех выводимых кривых.

Строка 7. Переменная cen2 хранит положение центра второго эллипса. Значения переменных cX и cY определено положениями соответствующих движков (см. Рис. 1.1).

Строки 6 – 30. Вызов функции Manipulate, позволяющей изменять параметры, отвечающие за положение второго эллипса, — координаты центра $\begin{pmatrix} c_x \\ c_y \end{pmatrix}$ и угол поворота φ.

Строка 9. Формирование матрицы поворота $M = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$. Значение переменной phi в этом месте кода определено положением соответствующего движка (см. рис. 1.1).

Строки 9 – 14. Рисование эллипсов с помощью функции ParametricPlot. Следует обратить внимание на то, что после вызова функций ParametricPlot[] стоит точка с запятой. Это приводит к тому, что в данном месте кода вывод графики не производится. Соответствующие графические объекты создаются и сохраняются в переменных e1 и e2.

Строки 15 – 20. Отыскание минимума функции двух переменных с помощью функции NMinimize. Значение параметра Method -> “RandomSearch” подобрано экспериментально. Отметим, что результат, возвращенный функцией NMinimize и помещенный в переменную minres имеет вид

$$\{r_{min}, \{u \to u_{min}, v \to v_{min}\}\},$$

где r_{min} — минимум расстояния, u_{min}, v_{min} — значение параметров эллипсов, соответствующих точкам, где этот минимум достигается. Минимизируемая функция расстояния между точками на эллипсах Norm[...] зависит от двух переменных u и v, каждая из которых принимает значения от 0 до 2π. Эти ограничения переданы функции NMinimize в качестве параметров для упрощения поиска минимума.

Строки 21 и 22. Вычисление координат точек на эллипсах по значениям параметров u_{min}, v_{min}. Для этого используется операция «применение правила» \. :

<выражение> \. minres[[2]]

Значением конструкции minres[[2]] является второй элемент списка $\{r_{min}, \{u \to u_{min}, v \to v_{min}\}\}$, т. е. список $\{u \to u_{min}, v \to v_{min}\}$. Выполнение операторов в строках 21 и 22 приводит к тому, что вместо переменных u и v выражениях слева от \. подставляются величины u_{min}, v_{min} соответственно. Координаты точек на эллипсах сохраняются в переменных p1 и p2.

Строки 23 – 24. Создается графический объект — отрезок, соединяющий точки p1 и p2. Результат сохраняется в переменной lin.

Строка 25. Вывод графических объектов e1, e2 и lin на одно изображение. Установка параметра AspectRatio -> 1 гарантирует, что соотношение сторон изображения будет 1:1. Если бы выводимое изображение было не квадратным, как в нашем случае, могло бы потребоваться вычислить и установить другое значение параметра AspectRatio. При автоматическом выборе значения этого параметра изображение может быть искажено. В частности, перпендикулярные линии могут оказаться не перпендикулярными. Это может быть важно, в частности, в данной задаче. Отрезок, соединяющий две ближайшие точки на эллипсах, является общей нормалью для этих эллипсов.

Строка 29. Задание списка переменных, изменение которых будет вызывать очередное вычисление кода внутри конструкции Manipulate.

Программу можно скачать по адресу
`https://notebookarchive.org/2020-02-b235dtz/`

3. Симплекс с минимальным коэффициентом поглощения

Всюду далее $n \in \mathbb{N}$. Элемент $x \in \mathbb{R}^n$ будем записывать в виде $x = (x_1, \ldots, x_n)$. Введем обозначение $Q_n := [0,1]^n$.

Для выпуклого тела $C \subset \mathbb{R}^n$ через σC обозначим результат гомотетии C относительно центра тяжести с коэффициентом σ.

Пусть S — невырожденный симплекс в $\mathbb{R}^n$. Для симплекса S введём величину

$$\xi(S) := \min\{\sigma \geq 1 : Q_n \subset \sigma S\}.$$

Введём в рассмотрение величину

$$\xi_n := \min\{\xi(S) : \ S - n\text{-мерный симплекс}, S \subset Q_n, \ \mathrm{vol}(S) \neq 0\}.$$

Будем называть ξ_n минимальным коэффициентом поглощения куба Q_n. Симплекс $S \subset \mathbb{R}^n$, для которого $\xi(S) = \xi_n$, будем называть *экстремальным симплексом*, или *симплексом с минимальным коэффициентом поглощения.*

Величины $\xi(S)$ и ξ_n были введены М. В. Невским в связи с задачами линейной интерполяции (см. [5]). Значения ξ_n известны только для некоторых n.

Приведем необходимые для вычислений формулы. Более подробное изложение этих вопросов можно найти в главе 1 монографии [5] и в учебном пособии [6]. Обозначим вершины S через $x^{(j)} = \left(x_1^{(j)}, \ldots, x_n^{(j)}\right)$, $1 \leq j \leq n+1$, и рассмотрим матрицу

$$\mathbf{A} = \begin{pmatrix} x_1^{(1)} & \ldots & x_n^{(1)} & 1 \\ x_1^{(2)} & \ldots & x_n^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(n+1)} & \ldots & x_n^{(n+1)} & 1 \end{pmatrix}. \tag{1.1}$$

Пусть $\Delta := \det(\mathbf{A})$, тогда $\mathrm{vol}(S) = \dfrac{|\Delta|}{n!}$. Обозначим через $\Delta_j(x)$ определитель, который получается из Δ заменой j-й строки на строку $(x_1, \ldots, x_n, 1)$. Многочлены $\lambda_j(x) := \dfrac{\Delta_j(x)}{\Delta}$ из $\Pi_1(\mathbb{R}^n)$ обладают свойством $\lambda_j\left(x^{(k)}\right) = \delta_j^k$, где δ_j^k — символ Кронекера. Многочлены λ_j называются *базисными многочленами Лагранжа* симплекса S. Коэффициенты λ_j составляют j-й столбец матрицы

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

Таким образом, $\lambda_j(x) = l_{1j}x_1 + \ldots + l_{nj}x_n + l_{n+1,j}$.

Подробное описание свойств многочленов Лагранжа и доказательства приводимых ниже свойств можно найти в монографии М. В. Невского [5]. Из свойств многочленов Лагранжа следует, что числа $\lambda_1(x), \ldots, \lambda_{n+1}(x)$ являются барицентрическими координатами точки $x \in \mathbb{R}^n$. Гиперплоскость $\lambda_j(x) = 0$ содержит все вершины симплекса, кроме вершины $x^{(j)}$, гиперплоскость $\lambda_j(x) = 1$ проходит через вершину $x^{(j)}$ (см. рис. 1.2). Таким образом, симплекс S есть пересечение полупространств $\lambda_j(x) \geq 0$ или множеств $0 \leq \lambda_j(x) \leq 1$.

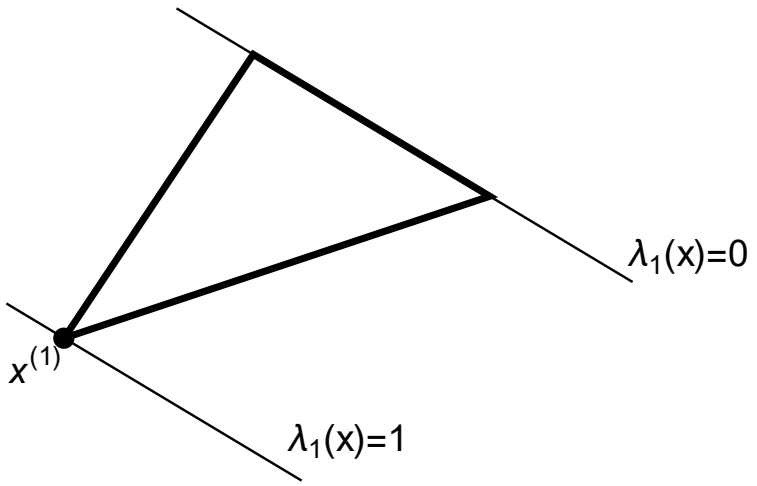

Рис. 1.2: Свойства базисных многочленов Лагранжа

Программа, позволяющая лучше проиллюстрировать свойства базисных многочленов Лагранжа, приведена в п. 1..

Пусть S — невырожденный симплекс.

Если $Q_n \not\subset S$, справедлива формула

$$\xi(S) = (n+1) \max_{1 \leq k \leq n+1} \max_{x \in \mathrm{ver}(Q_n)} (-\lambda_k(x)) + 1. \tag{1.3}$$

Задача о вычислении ξ_n сводится к задаче о минимизации функции $\xi(S)$, при условии, что $S \subset Q_n$.

Покажем, как можно приближенно вычислить ξ_n и найти экстремальный симплекс с помощью средств численной минимизации системы Wolfram Mathematica.

Проще написать программу для конкретного значения n. Сначала мы рассмотрим случай $n = 2$, а в дальнейшем покажем, как снять это ограничение.

Обозначим вершины двумерного симплекса (треугольника) S через (x1,y1), (x2,y2), (x3,y3). Функцию для вычисления величины $\xi(S)$ можно сгенерировать в явном виде с помощью следующего кода.

```
1    A={
2        {x1,y1,1},
3        {x2,y2,1},
4        {x3,y3,1}
5        };
6    L = Inverse[A];
7    l[i_,x_,y_]:=L[[1]][[i]]*x+L[[2]][[i]]*y+L[[3]][[i]];
8    T = Flatten[
9        Table[-l[i,x,y],{i,1,3,1},{x,0,1,1},{y,0,1,1}]
10                ];
11   ksi[x1_,y1_,x2_,y2_,x3_,y3_]:=1+3*Max[T];
```

Вычисления в данном фрагменте кода производятся в символьном режиме. Если убрать знаки “точка с запятой” в конце каждого оператора, будут выводиться результаты выполнения соответствующих операций. В данном случае выражения оказываются довольно громоздкими и мы их не приводим.

Естественно, при $n > 2$ выражения оказываются еще более сложными. Некоторые известные значения ξ_n были найдены с использованием аналогичных вычислений. Для некоторых значений n с помощью численной минимизации удалось уточнить известные теоретические оценки.

Комментарии к коду

Строки 1 – 5. Определение матрицы $\mathbf{A}$.

Строка 6. Вычисление обратной матрицы $L = \mathbf{A}^{-1}$.

Строка 7. Определение функций для вычисления базисных многочленов Лагранжа λ_j.

Строки 8 – 10. Формирование выражений, стоящих во втором максимуме в формуле (1.3). В выражение для λ_j подставляются координаты вершин куба (0,0), (0,1), (1,0), (1,1). Функция Flatten превращает полученную таблицу в одномерный список.

Строка 11. Определение функции ksi[] для вычисления $\xi(S)$.

Для вычисления минимума этой функции можно использовать функцию NMinimize следующим образом:

```
1   res = NMinimize[{ksi[x1,y1,x2,y2,x3,y3],
2       0 <= x1 && x1 <= 1 &&
3       0 <= y1 && y1 <= 1 &&
4       0 <= x2 && x2 <= 1 &&
5       0 <= y2 && y2 <= 1 &&
6       0 <= x3 && x3 <= 1 &&
7       0 <= y3 && y3 <= 1 },
8    {x1, y1, x2, y2, x3, y3}]
```

Результат, возвращенный функцией (и сохраненный в переменной res), имеет вид:

```
{2.34164,{x1->1.,y1->0.618034,x2->0.381966,y2->0.,
x3->0.,y3->1.}}
```

Значение ξ_2 известно: $\xi_2 = 1 + \dfrac{3\sqrt{5}}{5} = 2.341640\ldots$ (см. [5])

$$\xi_2 = 1 + \frac{3\sqrt{5}}{5} = 2.341640\ldots$$

Таким образом, нам удалось численно найти эту величину с довольно высокой точностью.

Полученный результат можно представить графически, выполнив следующий код:

```
1   ksi2 = res[[1]];
2   S = {{x1, y1}, {x2, y2}, {x3, y3}} /. res[[2]];
3   Ct = (S[[1]] + S[[2]] + S[[3]])/3;
4   ksi2S = {Ct + (S[[1]] - Ct)*ksi2,
5            Ct + (S[[2]] - Ct)*ksi2,
6            Ct + (S[[3]] - Ct)*ksi2};
7   Graphics[{
8     Thickness[0.001],
9     Line[{{0, 0}, {1, 0}, {1, 1}, {0, 1}, {0, 0}}],
10    FaceForm[None],
11    EdgeForm[Directive[Thickness[0.01], Black]],
12    Simplex[{S[[1]], S[[2]], S[[3]]}],
```

```
13     EdgeForm[Directive[Thickness[0.01], Red]],
14     Simplex[{ksi2S[[1]], ksi2S[[2]], ksi2S[[3]]}]
15             },
16   PlotRange -> {{-1, 2}, {-1, 2}},
17   Axes -> True, Ticks -> {{1}, {1}},
18   TicksStyle -> Directive["Label", 17],
19   LabelStyle -> Directive[Black, 17],
20   AxesLabel -> {"x", "y"},
21   AxesStyle -> Directive[Black, Thickness[0.005]]]
```

Результат выполнения этого кода приведен на рис 1.3.

Комментарии к коду

Строка 1. Извлечение найденного минимума – первого элемента списка res.
Строка 2. Формирование списка вершин экстремального симплекса S. Вместо переменных подставляются найденные значения аргументов функции ksi[].
Строка 3. Вычисление центра тяжести симплекса S. Результат сохранен в переменной Ct.
Строки 4 – 6. Вычисление координат вершин симплекса $\xi_2 S$. Полученный симплекс сохранен в переменной ksi2S.
Строки 7 – 21. Вывод графических объектов.
Строка 8. Задание толщины линии для рисования.
Строка 9. Вывод изображения куба (квадрата) Q_2.
Строка 10. Отключение отрисовки граней симплекса. Если этого не сделать, большой симплекс скроет все объекты, находящиеся у него внутри.
Строка 11. Настройка параметров изображения ребер симплекса S (черный цвет).
Строка 12. Вывод симплекса S.
Строка 13. Настройка параметров отображения ребер симплекса $\xi_2 S$ (красный цвет).
Строка 14. Вывод симплекса $\xi_2 S$.
Строки 16 – 21. Настройка параметров вывода графики.
Строка 16. Задание области вывода PlotRange.
Строка 17. Включение отображения осей (Axes -> True) и задание выводимых на осях отметок (Ticks -> {{1}, {1}}).
Строка 18. Установка цвета и размера шрифта для подписей к отметкам на осях (TicksStyle).
Строка 19. Определение цвета и размера вывода обозначений для осей (LabelStyle).
Строка 20. Задание текста обозначения осей (x и y).

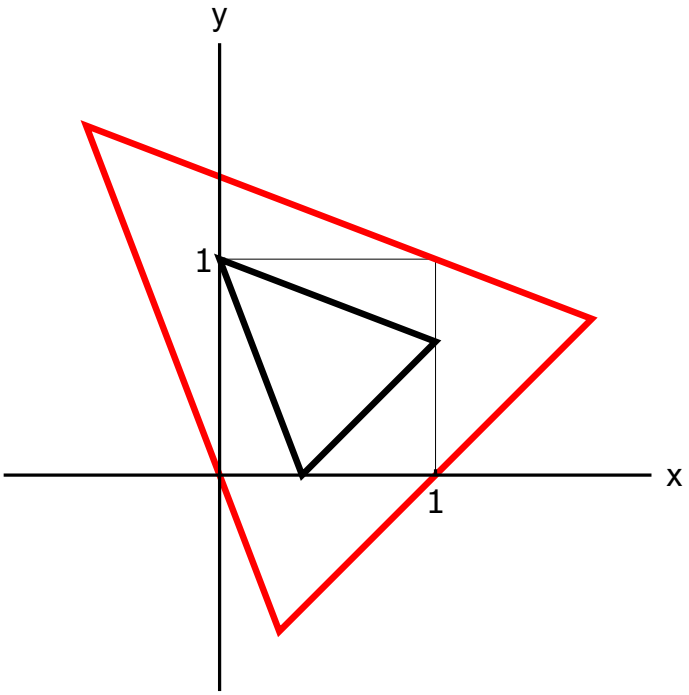

Рис. 1.3: Симплексы S и $\xi_2 S$

Строка 21. Задание толщины и цвета осей.

Программу можно скачать по адресу
`https://notebookarchive.org/2020-02-b23e65i/`

Приведенная программа для минимизации функции $\xi_(S)$, очевидно, может работать только для $n = 2$. Средства языка Wolfram Language позволяют снять эти ограничения и разработать программу, которая не содержит таких ограничений и теоретически может работать для произвольного значения n.

Ниже приводится фрагмент кода, генерирующий функцию ksi[S] для произвольного n. Для примера рассматривается случай $n = 3$

```
1   ClearAll[n,A,S,L,i,k,x,y,z,T,te,tb,i,j,bound];
2   n = 3;
3   S = Array[x, {n + 1, n}];
4   Print["S= ", S];
5   A = Map[Append[#, 1] &, S];
6   Print["A= ", A // MatrixForm];
7   L = Inverse[A];
8   (*Print[L//MatrixForm];*)
9   L = Transpose[L];
10  T = {};
```

```
11  For[i = 0, i <= 2^n - 1, i++,
12    (*Print["i=",i];*)
13    y = IntegerDigits[i, 2];
14    k = Length[y];
15    (*Print["k= ",k];*)
16    If[k < n,
17     z = ConstantArray[0, n - k];
18     y = Flatten[Prepend[y, z]];
19     (*Print["vertex= ",y];*)
20     ];
21    AppendTo[y, 1];
22    (*Print["y=",y];*)
23    te = -L.y ;
24    AppendTo[T, te];
25    ];
26  T = Flatten[T];
27  (*Print[T]*)
28  ksi[S_] := 1 + (n + 1)*Max[T];
29  bound = True;
30  For[i = 1, i <= n + 1, i++,
31    For[j = 1, j <= n, j++,
32     tb = 0 <= A[[i]][[j]] <= 1;
33     bound = And[bound, tb]
34     ]];
35  bound = And[bound];
36  Print["Bounds: ", bound];
37  Vars = Flatten[S];
38  Print["Variables:", Vars];
```

Для наглядности в коде оставлена незакомментированной часть операторов печати, предназначенных для отладки. Это позволяет лучше понять, что происходит в программе и какой вид имеют переменные, используемые для хранения данных. Ниже приводится текст, напечатанный при выполнении данного фрагмента кода.

```
S= {{x[1,1],x[1,2],x[1,3]},{x[2,1],x[2,2],x[2,3]},
   {x[3,1],x[3,2],x[3,3]},{x[4,1],x[4,2],x[4,3]}}

A= (x[1,1]      x[1,2]   x[1,3]   1
    x[2,1]      x[2,2]   x[2,3]   1
```

```
    x[3,1]        x[3,2]   x[3,3]   1
    x[4,1]        x[4,2]   x[4,3]   1   )

Bounds:
0<=x[1,1]<=1&&0<=x[1,2]<=1&&0<=x[1,3]<=1&&0<=x[2,1]<=1&&
0<=x[2,2]<=1&&0<=x[2,3]<=1&&0<=x[3,1]<=1&&0<=x[3,2]<=1&&
0<=x[3,3]<=1&&0<=x[4,1]<=1&&0<=x[4,2]<=1&&0<=x[4,3]<=1

Variables:
{x[1,1],x[1,2],x[1,3],x[2,1],x[2,2],x[2,3],
 x[3,1],x[3,2],x[3,3],x[4,1],x[4,2],x[4,3]}
```

Комментарии к коду

Строки 4, 6, 8, 12, 15, 19, 22, 27, 36, 38 содержат команды печати для отладки.

Строка 1. Очистка всех используемых в коде идентификаторов от ранее назначенных значений.

Строка 2. Полагаем n=3.

Строка 3. Формируем симплекс — двумерный массив (матрицу) из четырех строк и трех столбцов. Строки матрицы представляют собой координаты вершин симплекса.

Строка 5. Формируем матрицу A. Для этого к матрице S добавляется столбец из 1. Функция Append добавляет элемент 1 в конце одномерного массива. С помощью функции Map эта операция применяется по очереди к каждой строке матрицы S.

Строка 7. Вычисление обратной матрицы L ($L = A^{-1}$). Отметим, что обратная матрица вычисляется в символьном виде и полученное выражение становится довольно громоздким.

Строка 9. Матрица L транспонируется. Это позволяет в дальнейшем применять матричное умножение для вычисления значений в данной точке $x = (x_1, \ldots, x_n)$ сразу всех базисных многочленов Лагранжа.

Строка 9. Создание пустого списка T для храния величин, из которых выбирается максимум в формуле (1.3).

Строки 11 – 25. Формирование списка T. Для перебора всех вершин куба Q_n в (1.3) используется двоичное представление целого числа i — счетчика итераций цикла. Всего требуется сгенерировать 2^n векторов $(0, \ldots, 0) - (1, \ldots, 1)$.

Строка 13. Функция IntegerDigits[i, 2] возвращает список, состоящий из двоичных знаков числа i. Список сохраняется в переменной y.

Строка 14. В переменную k помещается длина списка y.

Строки 16 – 20. Список y дополняется спереди нулями так, чтобы его длина стала равна n.
Строка 21. Для вычисления значений базисных многочленов Лагранжа в точке $(x_1, \ldots, x_n)$ с помощью матричного умножения список дополняется в конце элементом 1. Переменная y теперь содержит выражение вида $(x_1, \ldots, x_n, 1)$.
Строка 23. Результатом умножения матрицы L^T на вектор $y = (x_1, \ldots, x_n, 1)$ является вектор, составленный из значений базисных многочленов Лагранжа $\lambda_j(x_1, \ldots, x_n)$ $(j = 1, \ldots, n+1)$.
Строка 24. Полученный на предыдущем шаге вектор добавляется к списку T.
Строка 26. Двумерный список T превращается в одномерный.
Строка 28. Определяется функция для вычисления $\xi(S)$. Она зависит от $n(n+1)$ переменной. При n=3 это переменные x[1,1],...,x[4,3] (см. список Variables в результатах выполнения кода).
Строки 29 – 35. Формирование набора ограничений для передачи в функцию минимизации NMinimize. Условию $S \subset Q_n$ соответствуют неравенства $0 \le x_i^{(j)} \le 1$, $0 \le i \le n$, $0 \le j \le n+1$. Эти выражения будут записаны в логической переменной bound. Функция And[a,b] создает из переданных в нее переменных логическое выражение a && b. Полученное в результате выполнения цикла логическое выражение можно видеть в результатах выполнения кода под заголовком Bounds.
Строка 37. Формирование списка переменных для передачи в функцию минимизации NMinimize. Для создания этого списка достаточно преобразовать список координат вершин S в одномерный список.

После выполнения приведенного выше кода можно выполнить минимизацию с помощью вызова функции NMinimize, например, следующим образом:

```
res = NMinimize[
        {ksi[S], bound},
        Vars,
        Method->"SimulatedAnnealing",
        "RandomSeed"->1]
```

В результате выполнения кода получено следующее выражение.

```
{3.,{x[1,1]->0.5,x[1,2]->1.,x[1,3]->1.,
     x[2,1]->0.5,x[2,2]->1.,x[2, 3]-> 0.,
     x[3,1]->0.,x[3,2]->0.,x[3,3]->0.5,
     x[4,1]->1.,x[4,2]->8.19807*10^-9,x[4,3]->0.5}}
```

М. В. Невский показал (см. [5]), что $\xi_3 = 3$. Следовательно, наша программа нашла практически точное значение минимума.

Значения параметров Method->“SimulatedAnnealing” и “RandomSeed”->1 при вызове функции NMinimize подобраны экспериментально. Параметр RandomSeed позволяет задать начальное значение датчика случайных чисел, используемого применяемым численным методом. Соответственно, при разных значениях этого параметра могут получиться разные результаты численной минимизации. В нашем случае, например, значение параметра “RandomSeed”->0 дало значение целевой функции, равное 3.75474, что значительно больше известного минимального значения. Естественно, в реальной ситуации, когда искомая величина заранее не известна, приходится многократно повторять поиск минимума с различными значениями параметров и выбирать наименьшее значение. Достижение точного минимума не может быть гарантировано, и удается получить лишь верхнюю границу искомого минимума.

Можно визуализировать полученный результат, изобразив найденный симплекс. Для этого достаточно выполнить следующий фрагмент кода:

```
Graphics3D[{
  FaceForm[None],
  EdgeForm[Directive[Thickness[0.01],Black]],
  Simplex[S/.res[[2]]]
  }]
```

Результат выполнения этого вызова функции Graphics3D приведен на рис. 1.4.

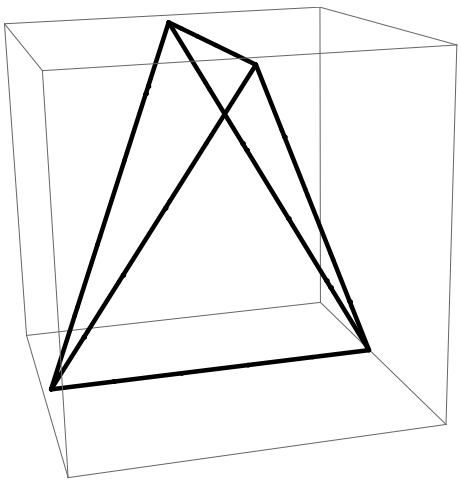

Рис. 1.4: Первый экстремальный симплекс для случая $n = 3$

Изменение параметров функции NMinimize иногда позволяет найти другой набор значений переменных, при котором достигается минимальное значение. В нашем случае изменение значения параметра RandomSeed на 3 позволяет найти другой экстремальный симплекс, для которого $\xi(S) = 3$. Вызов функции

```
res = NMinimize[
        {ksi[S], bound},
        Vars,
        Method->"SimulatedAnnealing",
        "RandomSeed"->3]
```

дает следующий результат:

```
{3.,{x[1,1]->0.,x[1,2]->0.,x[1,3]->1.,
     x[2,1]->1.,x[2,2]->1.,x[2,3]->1.,
     x[3,1]->0.,x[3,2]->1.,x[3,3]->0.,
     x[4,1]->1.,x[4,2]->0.,x[4,3]->0.}}
```

Полученный симплекс приведен на рис. 1.5

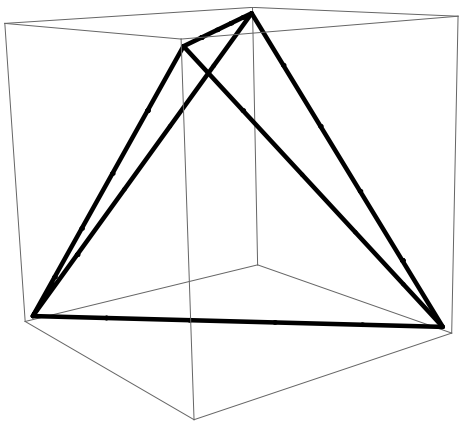

Рис. 1.5: Второй экстремальный симплекс для случая $n = 3$

Доказано (см. [5]), что любой экстремальный симплекс при $n = 3$ всегда эквивалентен одному из этих двух симплексов.

Программу можно скачать по адресу
`https://notebookarchive.org/2020-02-b23iqmr/`

4. Поиск минимума в автоматическом режиме

Значения ξ_n и новые оценки в работах [7–9] были получены с использованием программ, аналогичных приведенным выше. Естественно, при больших

значениях n требовалось организовать многократное выполнение поиска минимума в автоматическом режиме. Для этого использовались программы, аналогичные приведенной ниже.

```
1   Print[TimeObject[Now],"MINIMIZATION IS RUNNING"];
2   tmin = 100;
3   For[i = 0, i <= 100, i++,
4   Print[TimeObject[Now], " i= ", i];
5   newmin =
6   NMinimize[{ksi[S],bound},Vars,
7   Method->"SimulatedAnnealing",
8   "RandomSeed"->i];
9   If[newmin[[1]] < tmin,
10   Print[TimeObject[Now],"Attempt Number: ",i,
11   "  New Min was found!!!! Min= "];
12   Print[newmin];
13   tmin = newmin[[1]];
14   NotebookSave[]
15   ];
16   If[Mod[i, 10] == 0,
17    Print[TimeObject[Now],
18          "Attempt N ",i,"-no news..."];
19  Print["Current results: ", newmin];
20    NotebookSave[] ];
```

Фрагмент выдачи результатов работы кода приведен на рис. 1.6.

Комментарии к коду

Строка 1. Печать сообщения о начале работы программы.

Строка 2. Переменная tmin будет содержать наилучший найденный на данный момент результат.

Строки 3 – 20. Цикл попыток минимизации. Переменная цикла i будет использоваться как параметр для датчика случайных чисел, используемых алгоритмами минимизации.

Строка 4. Печать времени начала очередной итерации.

Строки 5 – 8. Вызов функции минимизации NMinimize.

Строки 9 – 15. Если найденное новое значение минимума меньше текущего минимума tmin, новый результат выводится на печать с соответствующим сообщением. Выполнение функции NotebookSave[] приводит к сохранению текущего документа Wolfram Notebook.

Строки 16 – 20. Печать текущего результата и сохранение файла каждые 10

итераций цикла.

```
12:56:27  i=  3
12:56:49  Attempt Number: 3    New Min was found!!!! Min=
{4.14755, {x[1, 1] → 0., x[1, 2] → 0.40332, x[1, 3] → 0.,
  x[1, 4] → 0.570714, x[2, 1] → 1., x[2, 2] → 0.0371553,
  x[2, 3] → 0.240156, x[2, 4] → 1.05266×10^-7,
  x[3, 1] → 0.999748, x[3, 2] → 1., x[3, 3] → 0.2404,
  x[3, 4] → 0.983199, x[4, 1] → 0.214133, x[4, 2] → 0.000151893,
  x[4, 3] → 1., x[4, 4] → 1., x[5, 1] → 0.214151,
  x[5, 2] → 1., x[5, 3] → 0.999983, x[5, 4] → 0.00210702}}
12:56:49  i=  4
12:57:16  i=  5
12:57:47  i=  6
12:58:44  Attempt Number: 6    New Min was found!!!! Min=
{4.12843, {x[1, 1] → 0.999992, x[1, 2] → 1.,
  x[1, 3] → 0.768956, x[1, 4] → 1., x[2, 1] → 0.,
  x[2, 2] → 0.229459, x[2, 3] → 8.56818×10^-8,
  x[2, 4] → 0.986207, x[3, 1] → 4.93512×10^-17,
  x[3, 2] → 0.999998, x[3, 3] → 0.768957,
  x[3, 4] → 0.000550047, x[4, 1] → 0.999117, x[4, 2] → 0.229455,
  x[4, 3] → 0., x[4, 4] → 0., x[5, 1] → 0.549693,
  x[5, 2] → 0., x[5, 3] → 1., x[5, 4] → 0.539717}}
```

Рис. 1.6: Выполнение поиска минимума в автоматическом режиме

Программу можно скачать по адресу
`https://notebookarchive.org/2020-02-b23ogar/`

Глава 2

Задачи вычислительной геометрии

1. Свойства базисных многочленов Лагранжа

Определение и основные свойства базисных многочленов Лагранжа приведены в п. 3.. Приведем код программы, позволяющей проиллюстрировать некоторые важные свойства этих многочленов. Для простоты рассмотрим случай $n = 2$.

```
x[1]={0,0};x[2]={3/2,1/2};x[3]={2/3, 1};
A = {
    {x[1][[1]], x[1][[2]], 1},
    {x[2][[1]], x[2][[2]], 1},
    {x[3][[1]], x[3][[2]], 1}
    };
L = Inverse[A];
p1[x1_,x2_]:=L[[1]][[1]]*x1+L[[2]][[1]]*x2+L[[3]][[1]];
Manipulate[
gr = Graphics[
 {
  FaceForm[None],
  EdgeForm[Directive[Thickness[0.01]]],
  Simplex[{x[1], x[2], x[3]}],
  Thickness[0.01], PointSize[0.03], Point[{0, 0}],
  Text[Style["\!\(\*SubscriptBox[\(v\), \(1\)]\)",
             Large],{-0.1, -0.10}],
  Text[
```

```
19      Style["\!\(\*SubscriptBox[\(\[Lambda]\),\(1\)]\)
20      (x)=t",Large],{2.2, 0.3}]
21  },
22   Axes -> False,
23   AxesStyle -> Directive[Thickness[0.01], 20],
24   AxesLabel -> {X1, X2}, Ticks -> None,
25   PlotRange -> {{-1, 3}, {-1, 2}}];
26   pl0=ContourPlot[p1[x1,x2]==t,
27    {x1,-1,3.0},{x2,-2.0,2.0},
28    ContourStyle->Red,
29    PlotRange->{{-1,3},{-1,2}}];
30  Show[gr, pl0],
31  {t, 0, 1}]
```

Результат работы кода показан на рис. 2.1.

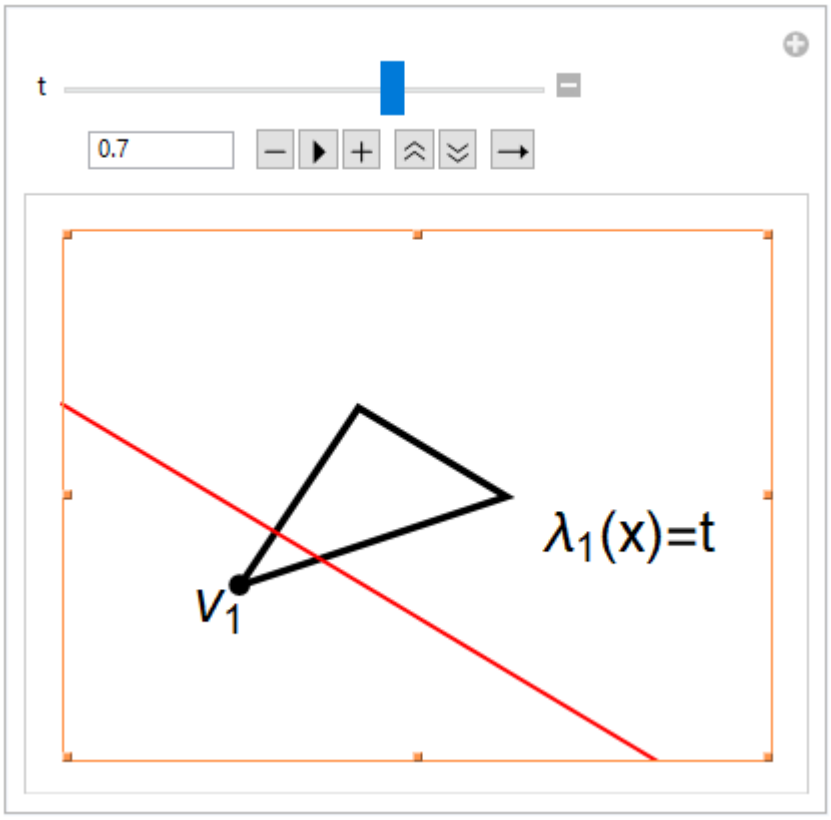

Рис. 2.1: Свойства базисных многочленов Лагранжа

Комментарии к коду

Строка 1. Задание вершин симплекса.

Строки 2 – 6. Построение матрицы A.

Строка 7. Вычисление обратной матрицы L.
Строка 8. Определение функции для вычисления многочлена λ_1.
Строки 9 – 31. Вызов функции Manipulate, позволяющей управлять параметром $t \in [0, 1]$. Код в теле функции создает изображение симплекса и прямой $\lambda_1(x_1, x_2) = t$.
Строка 10. Переменная gr будет хранить часть изображения, содержащую симплекс и надписи.
Строки 12 – 13. Настройки отображения симплекса.
Строка 14. Вывод изображения симплекса.
Строка 15. Настройка параметров изображения точки и вывод точки, отмечающей первую вершину симплекса.
Строки 16 – 20. Настройка шрифта и вывод надписей на рисунке.
Строки 22 – 25. Настройки параметров изображения.
Строки 26 – 29. Построение прямой $\lambda_1(x_1, x_2) = t$ с помощью вызова функции ContourPlot. Полученный графический объект сохраняется в переменной pl0.
Строка 30. Вывод построенных графических объектов (gr и pl0).

Программу можно скачать по адресу
`https://notebookarchive.org/2020-02-b23tbbv/`

2. Объем части куба, отсекаемой гиперплоскостью

Пусть $Q_n = [0, 1]^n$. Рассмотрим в $\mathbb{R}^n$ гиперплоскость

$$a_1x_1 + \ldots + a_nx_n = b. \tag{2.1}$$

Требуется найти объем части куба, отсекаемой гиперплоскостью (2.1).

Для вычисления объема части куба, отсекаемой гиперплоскостью, мы используем следующее утверждение из статьи [10].

Теорема. *Пусть $n > 2$, $Q'_n = [-1, 1]^n$. Рассмотрим гиперплоскость*

$$H = \{x \in \mathbb{R}^n : (a, x) = b\},$$

где $a = (a_1, \ldots, a_n) \in (\mathbb{R} \setminus \{0\})^n$, $b \in \mathbb{R}$. Тогда

$$\operatorname{vol}(Q'_n \cap H) = \frac{|a|}{\pi} \cdot 2^{n-1} \int\limits_{-\infty}^{+\infty} \left(\prod_{k=1}^{n} \frac{\sin(a_k t)}{a_k t} \right) \cdot \cos(bt)\, dt. \tag{2.2}$$

Здесь $\mathrm{vol}(Q'_n \cap H)$ означает $(n-1)$-мерный объем сечения куба Q'_n гиперплоскостью H. Если в уравнении гиперплоскости $|a| = 1$, то b есть расстояние со знаком от гиперплоскости H до начала координат. В этом случае интеграл по b в пределах от b_0 до $+\infty$ от правой части формулы (2.2) равен объему пересечения Q'_n и полупространства $\{x \in \mathbb{R}^n : (a, x) \geq b_0\}$. На практике требуется вычислить интеграл лишь по некоторому конечному промежутку, длина которого равна расстоянию от H до наиболее удаленной от нее вершины куба, лежащей в нужном полупространстве.

Так как мы рассматриваем куб $Q_n = [0,1]^n$, а не куб $Q'_n = [-1,1]^n$, в расчетные формулы потребуется внести соответствующие поправки.

Приведем код функции на языке Wolfram Language, реализующий описанный выше подход.

```
VolumeOfSegment[a_, b_] :=
 Block[{n, voln, anorm, an, bt, bn, x, y, maxd, maxx,
        k, ss, tt, FuN, FuD, Fu},
  n = Dimensions[a][[1]];
  (* Print["b= ",b]; *)
  anorm = Simplify[Sqrt[a.a]];
  (* Print["anorm= ",anorm]; *)
  an = Simplify[a/anorm];
  bt = Simplify[b/anorm];
  (* Print["an= ",an]; *)
  (* Print["bt= ",bt]; *)
  bn = Simplify[-Total[an] + 2*bt];
  (* Print["bn= ",bn]; *)
  maxd = -Sqrt[2*n];
  For[i = 0, i <= 2^n - 1, i++,
   (*Print["i=",i];*)
   x = IntegerDigits[i, 2];
   k = Length[x];
   (*Print["k= ",k];*)
   If[k < n,
    z = ConstantArray[0, n - k];
    x = Flatten[Prepend[x, z]];
    ];
   x = Map[If[# == 0, -1, 1] &, x];
   (*Print["vertex= ",x];*)
   ss = an.x - bn;
   If[Refine[ss > maxd], maxd = ss; maxx = x];
```

```
28      ];
29     (*Print["maxd= ",maxd];*)
30     (*Print["maxx= ",maxx];*)
31     FuN = 1;
32     FuD = 1;
33     For[k = 1, k <= n, k++,
34      If[Refine[an[[k]] != 0],
35           FuN *= Sin[an[[k]]*tt];
36           FuD *= an[[k]]*tt]
37      ];
38     FuN = TrigReduce[FuN];
39     FuD = Simplify[FuD];
40     Fu = FuN/FuD;
41     (* Print["Fu= ",Fu]; *)
42     voln = Integrate[(1/(2*Pi))*
43        Integrate[Fu*Cos[y*tt], {y, bn, bn + maxd},
44         Assumptions -> {Element[y, Reals]}],
45       {tt, -Infinity, +Infinity},
46       Assumptions -> {Element[y, Reals]}];
47     (*Print["Vol= ",voln];*)
48     Return[voln]]
```

Приведенная программа рассчитана на работу в символьном режиме. Это позволяет применять ее для получения точных значений объема. Допускается даже, что выражения, передаваемые в функцию, могут зависеть от параметров. В этом случае результат также может представлять собой функцию параметров. В сложных случаях может потребоваться модификация данного кода. Может потребоваться передать в функцию VolumeOfSegment[] дополнительные данные, описывающие область значений параметров. Эти параметры могут оказаться полезными при вызове функций упрощения выражений Refine и функций символьного интегрирования Integrate. Соответствующие вызовы функций могут иметь, например, вид (здесь дополнительное условие есть $-1/3 < t < 0$):

```
34   If[Assuming[-1/3<t<0,Refine[an[[k]] != 0]],
35      FuN*=Sin[an[[k]]*tt];
36      FuD*=an[[k]]*tt;]
37    ];
...
42     voln=Integrate[(1/(2*\[Pi]))*
```

```
43        Integrate[Fu*Cos[y*tt],{y,bn  bn+maxd},
44          Assumptions->{Element[y,Reals]&&-1/3<t<0}],
45          {tt,-Infinity,+Infinity},
46     Assumptions->{Element[y,Reals]&&-1/3<t<0}];
```

Если передаваемые в функцию выражения окажутся слишком сложны, программа может не справиться с их обработкой. В этом случае можно ограничиться получением численных результатов. Для этого приведенный код можно изменить, применив вместо функции символьного интегрирования Integrate функцию численного интегрирования NIntegrate.

Комментарии к коду

Строки 5, 7, 9, 10, 11, 13, 16, 19, 25, 29, 30, 38, 41, 47 содержат команды печати для отладки.

Строка 4. Функция определяет размерность пространства $\mathbb{R}^n$ по размерности первого аргумента.

Строки 6, 8, 9. Нормировка уравнения плоскости. Это нужно для правильного вычисления интеграла по отрезку [bn, bn + maxd], где maxd — максимальное расстояние от плоскости до вершины куба (интегрирование правой части (2.2) по b).

Строка 12. Вычисление переменной bn — "нового" коэффициента b — коэффициента плоскости (2.1), соответствующего сечению куба $Q'_n = [-1, 1]$. Коэффициенты a_i менять не требуется.

Строки 14 – 28. Вычисление значения maxd — максимального расстояния от плоскости до вершины куба. Отметим, что наибольшим оказывается расстояние до самой удаленной от плоскости вершины куба среди вершин, лежащих в положительной полуплоскости.

Строки 17 – 24. Формирование очередной вершины куба x. Здесь maxd — искомый максимум, ss — расстояние до очередной вершины, maxx — вершина, для которой достигается максимум расстояния. Координаты вершины maxx далее в коде не используются. Этот вектор можно распечатать при отладке программы, чтобы убедиться, что максимум найден верно.

Строки 31 – 40. Формирование подынтегрального выражения в (2.2). Здесь FuN — числитель, а FuD — знаменатель дроби.

Строка 38. Упрощение тригонометрической функции в числителе (произведения синусов). Опыт показывает, что интегрирование производится быстрее, если произведение синусов представить в виде суммы синусов кратных дуг. Это объясняется тем, что интегралы вида $\displaystyle\int_{-\infty}^{+\infty} \frac{\sin \alpha t}{t^p}\, dt$ сводятся к известным интегралам.

Строка 39. Упрощение знаменателя. В простейшем случае выражение вида $a_1 \dots a_n t^n$ не нуждается в упрощении. Однако, если коэффициенты a_i содержат радикалы, зависят от буквенных параметров и т. п., выполнение упрощения повышает шансы на то, что функция Integrate успешно справится с интегрированием полученного выражения.
Строка 40. Переменная Fu содержит готовое подынтегральное выражение.
Строки 42 – 46. Вычисление повторного интеграла. Поскольку от параметра b зависит только одно слагаемое, можно поменять порядок интегрирования. Это оправдывается тем, что несобственный интеграл в (2.2) сходится равномерно. Полученное выражение поделено на 2^n, чтобы получился объем части куба $Q_n = [0,1]^n$.

Протестировать приведенную функцию проще в стучае $n = 3$.

Рассмотрим сечение куба $Q_3 = \{(x_1, x_2, x_3) \ : \ 0 \le x_i \le 1, \ i = 1, 2, 3\}$ плоскостью $x_1 + x_2 + x_3 = \dfrac{1}{2}$. Выполнение кода

```
VolumeOfSegment[{-1,-1,-1},-1/2]
```

дает результат $\dfrac{1}{48}$. Если же вызвать функцию следующим образом:

```
VolumeOfSegment[{1,1,1},1/2]
```

результат будет $\dfrac{47}{48}$. Легко проверить, что это верные значения отсекаемых плоскостью объемов.

Программа корректно работает и если секущая плоскость параллельна одной или нескольким координатным осям. Чтобы убедиться в этом, рассмотрим тот же куб Q_3 и плоскость $x_1 + x_2 = \dfrac{1}{2}$. Вызовы функции

```
VolumeOfSegment[{1,1,0},1/2]
```

и

```
VolumeOfSegment[{-1,-1,0}  -1/2]
```

дают соответственно $\dfrac{7}{8}$ и $\dfrac{1}{8}$.

3. Объемы, отсекаемые плоскостями граней симплекса

В работах [11–13] изучалась задача об объемах, отсекаемых от куба $Q_n = [0,1]^n$ плоскостями, содержащими грани симплекса $S \subset Q_n$. Как по-

казано п. 3., уравнения этих плоскостей имеют вид $\lambda_j(x) = 0$, где $\lambda_j(x)$ — базисные многочлены Лагранжа. Используем функцию VolumeOfSegment, описанную в п. 2., для решения данной задачи. Коэффициентами многочленов $\lambda_j(x)$ являются столбцы матрицы (1.2). Для решения поставленной задачи требуется сформировать для каждого $j = 1, \ldots, n+1$ параметры функции , b и передать их в функцию VolumeOfSegment[a,b]. Для этого удобно использовать специальную функцию, которая принимает обратную матрицу L и выполняет нужные вызовы.

```
1    CutVosl[L_] := Block[{n, i, j, a, b, v},
2      Print["Cut Volumes"];
3      n = Dimensions[L][[1]] - 1;
4      For[j = 1, j <= n + 1, j++,
5       a = {};
6       For[i = 1, i <= n, i++, AppendTo[a, -L[[i]][[j]]]];
7       b = L[[n + 1]][[j]];
8       (*Print["a=",a];
9       Print["b=",b];*)
10      v = FullSimplify[VolumeOfSegment[a, b]];
11      Print["Vol_Lambda[ ", j, " ]=  ", v]];
12     ]
```

Комментарии к коду

Строка 3. Получение размерности задачи n.

Строки 4 – 12. Цикл перебора всех $n+1$-й граней симплекса.

Строки 5 – 6. Формирование вектора коэффициентов a. Знак коэффициента обратной матрицы меняется для того, чтобы вычислялся объем нужной (внешней по отношению к симплексу) части куба.

Строка 7. В переменную b помещается свободный член многочлена $\lambda_j(x)$.

Строка 10. Вызов функции VolumeOfSegment[a, b] для вычисления объема. Функция FullSimplify применяется для упрощения выражения. Выражение, возвращенное VolumeOfSegment, может иметь сложный вид, так как оно является результатом символьного интегрирования.

Строка 11. Печать очередного объема.

Для иллюстрации работы программы посчитаем объемы, отсекаемые плоскостями симплекса S с вершинами $(0,0,1)$, $(1,1,1)$, $(0,1,0)$, $(1,0,0))$ (см. рис. 1.5). Ниже приведен код для вычисления объемов.

```
S = {{0,0,1}, {1,1,1}, {0,1,0}, {1,0,0}};
A = Map[Append[#, 1] &, S];
Print["A= ", MatrixForm[A]];
```

```
L = Inverse[A];
Print["L= ", MatrixForm[L]];
CutVosl[L];
```

Результат выполнения кода приведен на рис. 2.2. Как оказывается, все грани симплексов отсекают от куба части одинаковых объемов.

$$\mathbf{A}= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{L}= \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Cut Volumes

$$\text{Vol\_Lambda}[\ 1\ ]= \frac{1}{6}$$

$$\text{Vol\_Lambda}[\ 2\ ]= \frac{1}{6}$$

$$\text{Vol\_Lambda}[\ 3\ ]= \frac{1}{6}$$

$$\text{Vol\_Lambda}[\ 4\ ]= \frac{1}{6}$$

Рис. 2.2: Вычисление отсекаемых объемов

Программу можно скачать по адресу
`https://notebookarchive.org/2020-02-b23znwl/`

Задачи и упражнения

1. Написать программу для нахождения кратчайшего расстояния между двумя эллипсоидами, предполагая, что один из эллипсоидов приведен к главным осям, а второй находится в произвольном положении.

   Указание. Использовать параметрические уравнения эллипсоида

   $$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} + \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} a\cos v\cos u \\ b\sin v\cos u \\ c\sin u \end{pmatrix},$$

   $$0 \le v \le 2\pi, \quad -\frac{\pi}{2} \le u \le \frac{\pi}{2}.$$

   Для формирования матрицы поворота использовать встроенную функцию RotationMatrix[θ,{q, r}]. Здесь θ — угол поворота, q и r — векторы, задающие плоскость. Поворот происходит в плоскости, задаваемой векторами q и r.

2. Написать функцию в процедурном стиле для вычисления функции $\xi(S)$. Если заметить, что вершина, на которой достигается второй максимум в формуле (1.3) может быть "угадана" по значениям коэффициентов j-го столбца матрицы (1.2), можно оптимизировать вычисление функции, исключив перебор вершин куба. Сравнить скорость вычисления полученной функции и функции, предложенной в п. 3..

3. Рассмотреть следующий код для вычисления функции $\xi(S)$.

```
KsiByS[S_] := Block[{A, ksi},
A = Map[Append[#, 1] &, S];
ksi = Max[
Map[Total,
Map[Select[#,#>0&]&,Drop[#,None,-1],{1}]
]
+
```

```
Flatten[Take[#, All, -1]]
]&[-Transpose[Inverse[#]]]*
Dimensions[#][[1]]+1&[A];
Return[ksi]
]
```

Протестировать работу функции. Сравнить скорость выполнения этой функции и функций, рассмотренных в задаче 2. Выяснить достоинства и недостатки каждого подхода.

4. Написать программу для отображения симплексов S и $\xi(S)S$ для случая $n = 3$.

5. С помощью программы для минимизации функции $\xi(S)$, приведенной в п. 4., найти оценку сверху для констант ξ_n для $n = 4, 5, 6, \ldots$

   Указание. Поэкспериментировать с различными алгоритмами минимизации. Для выбора другого алгоритма нужно изменить значение параметра Method. Вместо метода SimulatedAnnealing можно попробовать применить другие реализованные в системе методы: RandomSearch, NelderMead и DifferentialEvolution. У каждого из этих методов имеются свои параметры, которые можно задать изначально или изменять программно.

6. Для симплекса $S \subset \mathbb{R}^n$ обозначим через $\alpha(S)$ минимальное $\sigma > 0$, для которого Q_n принадлежит трансляту σS, т. е. гомотету с коэффициентом σ симплекса S. Для $\alpha(S)$ имеет место равенство

$$\alpha(S) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n+1} |l_{ij}| = \sum_{i=1}^{n} \frac{1}{d_i(S)}. \tag{3.1}$$

   Пользуясь формулой (3.1), написать функцию для вычисления величины $\alpha(S)$.

7. Для случаев $n = 2$ и $n = 3$ написать программу для рисования симплексов S и $\alpha(S)S$ и транслята $\alpha(S)S$, покрывающего куб Q_n (см. задачу 6).

8. Пусть S — симплекс в $\mathbb{R}^n$. Определим $d_i(S)$ как максимальную длину отрезка, содержащегося в S и параллельного оси x_i. Величину $d_i(S)$ будем называть i-м осевым диаметром симплекса S.

Пусть $1 \le i \le n$. Для i-го осевого диаметра S верно равенство

$$\frac{1}{d_i(S)} = \frac{1}{2}\sum_{j=1}^{n+1} |l_{ij}|\,. \tag{3.2}$$

В S существует ровно один отрезок длины $d_i(S)$, параллельный оси x_i. Центр этого отрезка совпадает с точкой

$$y^{(i)} := \sum_{j=1}^{n+1} m_{ij} x^{(j)}, \tag{3.3}$$

где

$$m_{ij} := \frac{|l_{ij}|}{\sum\limits_{k=1}^{n+1} |l_{ik}|}. \tag{3.4}$$

Пользуясь формулами (3.2), (3.3) и (3.4), написать программу для нахождения всех осевых диаметров n-мерного симплекса и соответствующих отрезков.

9. Для случаев $n = 2$ и $n = 3$ написать программы для рисования симплекса и отрезков, соответствующих его осевым диаметрам (см. задачу 8).

10. Пусть S — невырожденный симплекс в $\mathbb{R}^n$ и v — ненулевой n-мерный вектор. Обозначим через $d^v(S)$ максимальную длину отрезка, принадлежащего S и параллельного v. Написать программу для вычисления $d^v(S)$ и координат концов соответствующего отрезка.

Указание. Использовать формулы

$$d^v(S) = \frac{2\|v\|}{\sum\limits_{j=1}^{n+1} |m_j|}, \tag{3.5}$$

$$a = \sum_{j=1}^{n+1} \alpha_j x^{(j)}, \quad b = \sum_{j=1}^{n+1} \beta_j x^{(j)}. \tag{3.6}$$

Здесь a, b — концы отрезка максимальной длины, принадлежащего S и параллельного v,

$$m_j := \sum_{k=1}^{n} l_{kj} v_k, \quad \alpha_j := \frac{|m_j| - m_j}{\sum\limits_{k=1}^{n+1} |m_k|}, \quad \beta_j := \frac{|m_j| + m_j}{\sum\limits_{k=1}^{n+1} |m_k|}. \tag{3.7}$$

11. Для $n = 2$ и $n = 3$ написать программу для графического изображения максимального отрезка данного направления в симплексе (см. задачу 10).

12. Пусть $S(t)$ — симплекс с вершинами $(1,0,0,0,0,0,1)$, $(0,1,1,t,0,1,1)$, $(0,1,1,1-t,1,0,1)$, $(0,0,0,t,1,1,0)$, $(0,1,1,1-t,0,0,0)$, $(1,0,1,t,1,1,0)$, $(1,1,0,1-t,1,1,0)$, $(1,0,0,1,0,0,1)$. При $0 \le t \le 1$ имеем $S \subset Q_7$. Получить формулы для функций $\alpha(S(t))$ и $\xi(S(t))$ (определение функции $\alpha(S)$ и формула для ее вычисления приведены в задаче 6). Построить графики функций $\alpha(S(t))$ и $\xi(S(t))$ для $t \in [0,1]$.

13. С помощью программы из п. 3. найти объемы, отсекаемые плоскостями, содержащими грани симплекса для симплексов с вершинами

    а) $\left(\frac{1}{2},0,0\right)$, $\left(\frac{1}{2},1,0\right)$, $\left(0,\frac{1}{2},1\right)$, $\left(1,\frac{1}{2},1\right)$.

    б) $\left(\frac{1}{2},1,\frac{1}{3},1,1\right)$, $\left(\frac{1}{2},0,\frac{1}{3},1,1\right)$, $\left(\frac{1}{2},\frac{1}{2},\frac{1}{3},0,1\right)$, $\left(\frac{1}{2},\frac{1}{2},0,\frac{1}{3},0\right)$, $\left(0,\frac{1}{2},1,\frac{1}{3},0\right)$, $\left(1,\frac{1}{2},1,\frac{1}{3},0\right)$.

14. Написать программу для вычисления объема k-го слоя куба Q_n

$$T_{n,k} = \Big\{x \in Q_n \ : \ k-1 \le \sum_{i=1}^{n} x_i \le k\Big\}.$$

Убедиться, что $\mathrm{Vol}(T_{n,k}) = \dfrac{A_{n,k}}{n!}$, где $A_{n,k}$ — эйлеровы числа

$$A_{n,k} = \sum_{j=0}^{k-1} (-1)^j \binom{n+1}{j} (k-j)^n.$$

Литература

1. Wolfram S. An elementary introduction to the Wolfram Language. New York: Wolfram Media, 2015. 340 p.

2. Дьяконов В. П. Mathematica 5/6/7. Полное руководство. М.: ДМК Пресс, 2010. 624 с.

3. Климов В. С., Ухалов А. Ю. Решение задач математического анализа с использованием систем компьютерной математики. Ярославль: ЯрГУ, 2014. 96 с.

4. Mangano S. Mathematica cookbook. Cambridge: O'Reilly Media Inc. 2010. 830 p.

5. Невский М. В. Геометрические оценки в полиномиальной интерполяции. Ярославль: ЯрГУ, 2012. 218 с.

6. Невский М. В., Ухалов А. Ю. Избранные задачи анализа и вычислительной геометрии. Ч. I: учебное пособие. Ярославль: ЯрГУ, 2020. 98 с.

7. Невский М. В., Ухалов А. Ю. О числовых характеристиках симплекса и их оценках // Моделирование и анализ информационных систем. 2016. Т. 23, № 5. С. 603–619.

8. Невский М. В., Ухалов А. Ю. Новые оценки числовых величин, связанных с симплексом // Моделирование и анализ информационных систем. 2017. Т. 24, № 1. С. 94–110.

9. Nevskii M., and Ukhalov A. Perfect simplices in $\mathbb{R}^5$ // Beiträge zur Algebra und Geometrie / Contributions to Algebra and Geometry. 2018. V. 59, Issue 3. P. 501–521.

10. Frank R., Riede H. Hyperplane sections of the n-dimensional cube // American Mathematical Monthly. 2012. V. 119, N 10. P. 868–872.

11. Невский М. В., Ухалов А. Ю. Об n-мерных симплексах, удовлетворяющих включениям $S \subset [0,1]^n \subset nS$ // Моделирование и анализ информационных систем. 2017. Т. 24, № 5. С. 578–595.

12. Есипова Е. М. Геометрические характеристики симплексов, обладающих свойством равноотсечения // Современные проблемы математики и информатики: сборник научных трудов молодых ученых, аспирантов и студентов. Вып. 17. / Яросл. гос. ун-т им. П. Г. Демидова. Ярославль: ЯрГУ, 2017. С. 49–61.

13. Есипова Е. М., Ухалов А. Ю. Свойства экстремальных симплексов, связанные с отсекаемыми объемами // Современные проблемы математики и информатики: сборник научных трудов молодых ученых, аспирантов и студентов. Вып. 18. Ярославль: ЯрГУ, 2018. С. 4–17.

14. Ukhalov A., Nevskii M. Functions for checking necessary conditions for maximality of 0/1-determinant and example. Mendeley Data, v1. 2018. URL: http://dx.doi.org/10.17632/sm3x4xrb42.1

15. Ukhalov A. Matrices having the largest known determinant in machine-readable form. Mendeley Data, v1. 2019. URL: http://dx.doi.org/10.17632/hzf94h43c5.1

Учебное издание

Ухалов Алексей Юрьевич

Практикум по Wolfram Mathematica

Практикум

Редактор, корректор Л. Н. Селиванова
Компьютерная верстка А. Ю. Ухалов

Подписано в печать 10.04.2020.
Формат 60x84 1/16.
Усл. печ. л. 2,3. Уч.-изд. л. 2,0.
Тираж 3 экз. Заказ .

Оригинал-макет подготовлен
в редакционно-издательском отделе ЯрГУ.

Ярославский государственный университет им П. Г. Демидова
150003, Ярославль, ул. Советская, 14.