МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
Московский физико-технический институт
(национальный исследовательский университет)

А.Е. Умнов

МЕТОДЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Рекомендовано Учебно-методическим объединением высших учебных заведений Российской федерации по образованию в области прикладных математики и физики в качестве учебного пособия

7-е издание
испр. и доп.

Москва 2021

УДК 519.7(075)
ББК 22.18я73
У54

Рецензенты:
Отдел "Проблем математического моделирования"
Вычислительный центр им. А.А. Дородницына РАН
(зав. отделом, д. ф.-м. н., проф. А.П. Абрамов),
д. ф.-м. н., проф. А.П. Афанасьев

УМНОВ А.Е.
У20 Методы математического моделирования:
Учебное пособие. – М.: МФТИ, 2021. 295 с.
ISBN 5-7417-0189-2

Данное учебное пособие является расширенной версией курса лекций по основам методологии математического моделирования в условиях неполного информационного обеспечения и предназначено для студентов высших учебных заведений, специализирующихся в области менеджмента наукоемких и инновационных технологий.

Учебное издание

УМНОВ Александр Евгеньевич
Методы математического моделирования

Редактор *В.А. Дружинина.* Корректор *Л.В. Себова.*

Государственное образовательное учреждение
высшего профессионального образования
Московский физико-технический институт
(национальный исследовательский университет)
РФ 141700, Моск. обл., г. Долгопрудный, Институтский пер., 9

ISBN 5-7417-0189-2

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Процесс построения *математической модели* – формализованного (то есть представленного в виде математических соотношений) описания комплекса факторов, существенно влияющих на состояние и/или функционирование исследуемого объекта, и соответствующего этому описанию информационного обеспечения – принято называть *математическим моделированием*.

Практическая полезность математического моделирования заключается в возможности получения информации о качественных свойствах и количественных характеристиках изучаемого объекта без проведения (часто сложных или дорогостоящих) экспериментов в натуре, что может оправдывать затраты на преодоление трудностей, возникающих в процессе разработки или при попытках использования математических моделей.

Основное затруднение, с которым приходится сталкиваться в математическом моделировании, заключается в обеспечении *адекватности* этой модели исследуемому объекту. Пользователю необходимо выяснить, насколько точно данная модель отражает реальную ситуацию и насколько надежные количественные оценки могут быть получены в процессе работы с этой моделью.

Опыт математического моделирования, накопленный в течение последних нескольких десятилетий, показывает, что проблема адекватности в ряде случаев может быть успешно разрешена. Примером тому служат системы компьютерной имитации многочисленных природных процессов и технических объектов. С другой стороны, попытки применения методов математического моделирования для исследования социально–экономической объектов природы убедительно демонстрируют, что, несмотря на естественное желание учесть в модели все факторы, существенно влияющие на функционирование исследуемого объекта, добиться этого исключительно трудно, а иногда даже невозможно.

В случаях, когда построение математической модели, учитывающей с приемлемой степенью точности все факторы являющиеся существенными для исследуемого объекта невозможно, приходится отказываться от стандартной методологии использования модели и пытаться действовать иными способами, основанными на *изменении постановок* решаемых задач и *включении пользователя* в процесс поиска решений. В дальнейшем, в рамках данного курса, такие модели будут называться *неполными*.

Основной целью данного курса является описание альтернативной методологии математического моделирования, не требующей выполнения условия адекватности в полной мере. При этом рассмотрение данной проблемы ограничивается классом неполных математических моделей, каждое состояние которых полностью и однозначно описывается упорядоченным *конечным* набором вещественных чисел, что, с одной стороны, позволяет при исследовании модели ограничиться классическим аппаратом математического анализа и теории конечномерных линейных пространств. С другой стороны, это ограничение не является принципиально необходимым, и рассматриваемые методы могут быть применены и для иных классов задач.

Структура пособия, имеющего своей задачей содействие подготовке разработчиков и пользователей неполных моделей, основана на интеграции (в виде одного учебного курса) сведений, традиционно преподаваемых в различных разделах высшей математики и информатики, а также ряда научных результатов, полученных в разные периоды времени, как при участии автора, так и под его руководством.

Условно учебное пособие можно разделить на три части:

- первая часть (глава 1) содержит обзор необходимых сведений из курсов линейной алгебры, выпуклого анализа и теории функций многих переменных;
- во второй части (главы 2-5) рассматриваются как классические оптимизационные задачи, так и элементы теории математического программирования, включая понятия двойственности и принципа максимума, а также их приложения. Особое внимание уделено линейным оптимизационным задачам, являющимся инструментальной основой методологии неполного моделирования;
- третья часть (главы 6-9) посвящена вопросам применения методологии неполного моделирования для класса конечномерных моделей. Детально рассматриваются проблемы корректного включения пользователя в контур принятия решений, а также различные аспекты практического использования комплекса алгоритмов и необходимых программных средств.

Набор программных средств и файлов с иллюстративными данными, необходимыми как для демонстрационных целей, так и для самостоятельного использования студентами при выполнении заданий, доступен на архивном сервере кафедры высшей математики МФТИ или на авторском сайте www.umnov.ru .

Автор также хотел бы выразить признательность за содействие и большой вклад в получение результатов, использованных при подготовке данного пособия.

В первую очередь ведущему научному сотруднику ЦЭМИ РАН, Киму К.В., выполнившему в 1985 году первую реализацию алгоритма компьютерного анализа неполных линейных моделей (на языке FORTRAN-IV, в рамках проекта "Региональное развитие" Международного института прикладного системного анализа, IIASA, Laxenburg, Austria).

Выпускникам МФТИ, сотрудникам НИИ "Научный центр": Васильченко Е.Н., Кучеренко К.А. и Кулешову А.П., принимавшим активное участие в разработке и внедрении системы неполного моделирования "Баланс-2", отмеченной в декабре 1991 года золотой медалью ВДНХ СССР.

Выпускнику МФТИ, Коротких М.П., запрограммировавшему на C++ версию основного модуля системы – программы Lc и разработавшему в 1992 году в рамках своей дипломной работы интерпретатор языка L.

Выпускникам МФТИ: Чекареву Д.А., создавшему версию интерфейсной оболочки MultiLc для среды MS Windows XP, и Умнову Е.А., выполнившему большой объем работы по подготовке и тестированию демонстрационных моделей и задач.

Автор также выражает благодарность доценту кафедры математических основ управления МФТИ Бирюкову А.Г., прочитавшему рукопись и сделавшему ряд ценных замечаний.

Наконец, автор хотел бы поблагодарить студентов МФТИ групп 431 и 531, слушавших этот курс в 2005/06 и 2006/07 учебных годах, чье упорное желание разобраться в сути изучаемого предмета явилось одним из основных стимулов подготовки данного пособия.

Глава 1

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Раздел 1.1. КОНЕЧНОМЕРНОЕ ЕВКЛИДОВО ПРОСТРАНСТВО

§ 1.1.1. Определение и основные свойства

В рамках данного учебного курса мы ограничимся рассмотрением лишь математических моделей, конкретное состояние которых описывается конечным упорядоченным набором вещественных чисел. Поэтому одним из основных компонентов используемого математического аппарата является конечномерное евклидово пространство. Напомним соответствующие определения и основные свойства.

Линейное пространство

Определение 1.1.1.1 Множество Λ, состоящее из элементов $x, y, z, \ldots$, для которых определена операция сравнения, называется *линейным пространством*, если

1°. Каждой паре элементов x, y этого множества поставлен в соответствие третий элемент этого же множества,

называемый их *суммой* и обозначаемый $x+y$, таким образом, что выполнены аксиомы:

а) $x+y=y+x$;

б) $x+(y+z)=(x+y)+z$;

в) существует *нулевой элемент* o такой, что для любого $x\in\Lambda$ имеет место $x+o=x$;

г) для каждого x существует *противоположный элемент* $(-x)$ такой, что $x+(-x)=o$.

2°. Для любого элемента x и любого числа λ существует такой принадлежащий Λ элемент, обозначаемый λx и называемый *произведением числа на элемент*, что выполнены аксиомы:

а) $1x=x$;

б) $(\lambda\mu)x=\lambda(\mu x)$.

3°. Для операций сложения элементов и умножения элемента на число выполнены аксиомы дистрибутивности:

а) $(\lambda+\mu)x=\lambda x+\mu x$;

б) $\lambda(x+y)=\lambda x+\lambda y$; $\forall x,y\in\Lambda$; *и для любых чисел* λ,μ.

Конечномерное линейное пространство

Определение 1.1.1.2. Будем называть частный вид линейного пространства n*-мерным линейным пространством* (обозначаемым как Λ^n), если в этом пространстве существует *базис* – набор n упорядоченных линейно независимых элементов $\{g_1,g_2,\ldots,g_n\}$, таких, что каждый неупорядоченный набор элементов вида $\{x,g_1,g_2,\ldots,g_n\}$ $\forall x\in\Lambda^n$ оказывается *линейно зависимым*.

В этом случае любой элемент $x \in \Lambda^n$ *может быть представлен*, и притом *единственным образом* в виде $x = \sum_{j=1}^{n} \xi_j g_j$, где числа $\{\xi_1, \xi_2, \ldots, \xi_n\}$, называемые *компонентами или координатами*, образуют столбец $\|x\|_g = \begin{Vmatrix} \xi_1 \\ \xi_2 \\ \ldots \\ \xi_n \end{Vmatrix}$ – *координатное представление* элемента x в данном базисе.

Евклидово пространство

Определение 1.1.1.3

Евклидово пространство E (по определению) является частным случаем *линейного пространства* Λ, если в нем введена операция *скалярного произведения*, в результате которой *каждой* упорядоченной паре элементов $x, y \in E$ ставится в соответствие *единственное* число (x, y) таким образом, что выполняются следующие четыре аксиомы:

1°. $(x, y) = (y, x)$;

2°. $(\lambda x, y) = \lambda (x, y)$;

3°. $(x_1 + x_2, y) = (x_1, y) + (x_2, y)$;

4°. $(x, x) \geq 0$, причем

$$(x, x) = 0 \quad \Leftrightarrow \quad x = o\,.$$

В E введем следующие числовые характеристики элементов:

1°. $|x|$ – *норма* (*длина*) элемента x, где $|x| = \sqrt{(x, x)}$.

2°. $\rho(x, y)$ – *расстояние между элементами* x и y, где

$$\rho(x, y) = |x - y| = \sqrt{(x - y, x - y)}\,.$$

Исходя из аксиом евклидова пространства, можно показать, что для любых $x, y \in E$ справедливы:

- неравенство Коши–Буняковского $\left|(x,y)\right| \le \left|x\right|\left|y\right|$ и

- неравенство "треугольника" $\left|x+y\right| \le \left|x\right| + \left|y\right|$.

Как частный случай линейного пространства, евклидово пространство также может быть конечномерным (такое пространство принято обозначать E^n), причем в нем всегда можно построить особый базис $\{e_1, e_2, \ldots, e_n\}$, называемый *ортонормированным*, такой, что

$$(e_i, e_j) = \delta_{ij} = \begin{cases} 1, i = j, \\ 0, i \neq j. \end{cases}$$

Наконец, из аксиоматик линейного и евклидова пространств следует, что в любом ортонормированном базисе для каждой пары элементов $x, y \in E^n$ скалярное произведение (x, y) имеет вид

$$(x,y) = \|x\|^{\mathrm{T}} \|y\| = \sum_{j=1}^{n} \xi_j \eta_j \,, \quad \text{где } \|x\|_e = \begin{Vmatrix} \xi_1 \\ \xi_2 \\ \ldots \\ \xi_n \end{Vmatrix} \text{ и } \|y\|_e = \begin{Vmatrix} \eta_1 \\ \eta_2 \\ \ldots \\ \eta_n \end{Vmatrix}$$

суть координатные представления (координатные столбцы) этих элементов в базисе $\{e_1, e_2, \ldots, e_n\}$[1].

Нормы и расстояния для элементов в таком базисе находятся по формулам:

1º. *норма* (*длина*) элемента x

$$|x| = \sqrt{(x,x)} = \sqrt{\|x\|^{\mathrm{T}} \|x\|} = \sqrt{\sum_{j=1}^{n} \xi_j^2} \qquad \text{и}$$

2º. *расстояние между элементами* x и y

$$\rho(x,y) = |x-y| = \sqrt{(x-y, x-y)} = \sqrt{\sum_{j=1}^{n} (\xi_j - \eta_j)^2} \,.$$

[1] Верхний индекс $^{\mathrm{T}}$ означает операцию транспонирования.

В дальнейшем (если явно не оговорено иное) мы будем всегда использовать только евклидово конечномерное пространство с ортонормированным базисом.

§ 1.1.2. Подмножества и сходимость в E^n

Напомним определения некоторых важных для приложений понятий.

Определение 1.1.2.1

Множество $U \subset E^n$ называется *ограниченным*, если существует $C < +\infty$ такое, что $|x| \le C \quad \forall x \in U$.

Множество всех элементов $x \in E^n$ таких, что $\rho(x,a) < \delta$ (для некоторого $\delta > 0$), называется δ-*окрестностью элемента* a и обозначается как $U_\delta(a)$.

Элемент $x \in U$ называется *граничным* элементом множества U, если любая окрестность этого элемента содержит как элементы U, так и элементы, не принадлежащие U.

Элемент $x \in U$ называется *внутренним* элементом множества U, если существует окрестность этого элемента, содержащая только элементы U. Совокупность всех внутренних элементов множества U будем обозначать $\operatorname{int} U$.

Множество $U \subset E^n$ называется *замкнутым*, если оно содержит все свои граничные элементы. Если же U не содержит ни одного своего граничного элемента, то U принято называть *открытым* множеством.

Множество $U \subset E^n$ называется *выпуклым*, если $\forall x_1, x_2 \in U$ элемент $x = (1-\lambda)x_1 + \lambda x_2$ принадлежит множеству $U \quad \forall \lambda \in [0,1]$.

Определение 1.1.2.2

Будем говорить, что в E^n задана *последовательность* элементов $\{x_k\}$, если каждому натуральному числу k поставлен в соответствие по некоторому правилу единственный элемент x_k.

Элемент $a \in E^n$ называется *пределом последовательности* $\{x_k\}$, если

$$\rho(x_k, a) \to 0 \text{ при } k \to \infty .$$

Множество U называется *компактным*, если оно содержит предельные элементы для *всех* своих последовательностей. В E^n каждое замкнутое, ограниченное множество является компактным.

В случае, когда предел существует, говорят, что "*последовательность* $\{x_k\}$ *сходится к элементу* a" и записывают этот факт в виде

$$\lim_{k\to\infty} x_k = a .$$

Определение 1.1.2.3

Непустое множество Ω, образованное из элементов линейного пространства Λ, называется *подпространством* этого линейного пространства, если $\forall x, y \in \Omega$ и любого числа λ множеству Ω также принадлежат элементы: $x + y$ и λx.

Совокупность всевозможных линейных комбинаций некоторого множества элементов $\{x_1, x_2, ..., x_k\}$ линейного пространства Λ называется *линейной оболочкой* этого множества.

Определение 1.1.2.4

Множество, состоящее из элементов вида $x + x_0$, где x_0 есть произвольный фиксированный элемент линейного пространства Λ, а x – любой элемент некоторого подпространства $\Omega \subset \Lambda$, называется *гиперплоскостью* (или *линейным многообразием*) в линейном пространстве Λ.

Гиперплоскостью в конечномерном евклидовом пространстве является, например, совокупность всех элементов $x \in E^n$, удовлетворяющих уравнению вида $(l, x - x^0) = 0$, где $l \neq o, x^0$ – некоторые фиксированные элементы E^n.

Каждая гиперплоскость вида $(l, x - x^0) = 0$ разделяет конечномерное евклидово пространство на два "*полупространства*"

$$E^n_+ = \{x | (l, x - x^0) \geq 0\} \quad \text{и} \quad E^n_- = \{x | (l, x - x^0) \leq 0\}.$$

Определение 1.1.2.5

Принято говорить, что гиперплоскость

$$(l, x - x^0) = 0$$

является *разделяющей для подмножеств* Θ_1 *и* Θ_2 *в* E^n, если

$$\begin{cases} \Theta_1 \subset E^n_+, \\ \Theta_2 \subset E^n_-. \end{cases}$$

Гиперплоскость $(l, x - x^0) = 0$ *называется опорной для множества* Θ *на его граничном элементе* x^*, если

$$(l, x^* - x^0) = 0 \text{ и } \left[\begin{array}{l} \Theta \subset E^n_+, \\ \Theta \subset E^n_-. \end{array} \right.$$

Определение 1.1.2.6.

Множество $K \subset E^n$ называется *конусом с вершиной в* x^0, если $\forall x \in K$ элемент $x^0 + \tau(x - x^0)$ также принадлежит множеству K $\forall \tau \geq 0$.

В общем случае конус в E^n не является выпуклым или замкнутым множеством, однако некоторые важные для приложений частные виды конусов данными свойствами обладают. Например, пересечение в E^n k полупространств

$$(l_i, x - x^0) \geq 0, \; l_i \neq o \; \forall i = [1, k]$$

задает выпуклый замкнутый конус с вершиной в x^0, называемый иногда *многогранным*.

§ 1.1.3. Проекция элемента на подмножество

Определение 1.1.3.1

Проекцией элемента $x^0 \in E^n$ *на выпуклое подмножество* $\Omega \subset E^n$ называется элемент $\bar{x} \in \Omega$ такой, что $\left| x^0 - \bar{x} \right| = \inf\limits_{x \in \Omega} \left| x^0 - x \right|$.

Неотрицательное число $\rho \equiv \inf\limits_{x \in \Omega} \left| x^0 - x \right|$ называется *расстоянием элемента* x^0 *до подмножества* Ω.

Основные свойства проекций и расстояний от элемента до подмножества в E^n могут быть сформулированы в виде следующих теорем.

Теорема 1.1.3.1

Для любого выпуклого замкнутого множества $\Omega \subset E^n$ **и любого элемента** $x^0 \in E^n$ ***существует единственный*** **элемент** $\bar{x} \in \Omega$**, являющийся проекцией** x^0 **на** Ω**.**

Доказательство.

Докажем *существование* проекции.

Если $x^0 \in \Omega$, то $\bar{x} = x^0$ и $\rho = 0$.

Пусть теперь $x^0 \notin \Omega$ и существует число $\rho = \inf\limits_{x \in \Omega} \left| x^0 - x \right|$, тогда по определению точной нижней грани существует ограниченная последовательность элементов $\{x_k\} \subset \Omega$ такая, что

$$\lim_{k \to \infty} \left| x^0 - x_k \right| = \rho .$$

Но, согласно теореме Больцано–Вейерштрасса, из ограниченной последовательности можно выделить $\{x_{k_i}\} \subset \Omega$ – сходящуюся

подпоследовательность $\{x_{k_i}\}$.

Если при этом $\lim\limits_{i\to\infty} x_{k_i} = \bar{x}$, то в силу замкнутости Ω элемент $\bar{x} \in \Omega$, и для него справедливо равенство $\rho = \left| x^0 - \bar{x} \right|$. То есть $\bar{x}$ – проекция x^0 на Ω.

Покажем теперь, что проекция *единственна*.

Без ограничения общности будем считать, что $x^0 = o$, и предположим противное: пусть в Ω существуют неравные элементы $\bar{x}_1$ и $\bar{x}_2$, для которых $\left| \bar{x}_1 \right| = \left| \bar{x}_2 \right| = \rho$.

Рассмотрим два элемента: $y = \dfrac{\bar{x}_1 + \bar{x}_2}{2}$ и $z = \dfrac{\bar{x}_1 - \bar{x}_2}{2}$, для которых очевидны равенства $\bar{x}_1 = y + z$ и $(y, z) = 0$. Тогда

$$\rho^2 = (\bar{x}_1, \bar{x}_1) = (y + z, y + z) = \left| y \right|^2 + \left| z \right|^2$$

и, следовательно, $\left| y \right|^2 < \rho^2$, поскольку согласно сделанному предположению $z \neq o$.

Наконец, учитывая, что в силу выпуклости Ω элемент

$$y = \frac{\bar{x}_1 + \bar{x}_2}{2} \in \Omega,$$

приходим к противоречию с определением 1.1.3.1, что и доказывает единственность проекции.

Теорема доказана.

Теорема 1.1.3.2

Для того чтобы элемент $\bar{x} \in \Omega$ являлся проекцией элемента x^0 на выпуклое замкнутое множество Ω, необходимо и достаточно, чтобы $\forall x \in \Omega$ выполнялось неравенство

$$(x - \bar{x}, x^0 - \bar{x}) \le 0.$$

Доказательство.

Докажем необходимость.

Пусть $\bar{x} \in \Omega$ – проекция x^0 на Ω, тогда элемент $y = \alpha x + (1-\alpha)\bar{x} \in \Omega$, $\forall x \in \Omega$ при $\forall \alpha \in [0,1]$. Для этого элемента справедлива оценка

$$\left| x^0 - y \right|^2 = \left| x^0 - (\alpha x + (1-\alpha)\bar{x}) \right|^2 =$$

$$= \left| (x^0 - \bar{x}) - \alpha(x - \bar{x}) \right|^2 =$$

$$= \left| x^0 - \bar{x} \right|^2 - 2\alpha(x^0 - \bar{x}, x - \bar{x}) + \alpha^2 \left| x - \bar{x} \right|^2.$$

В силу определения 1.1.3.1 $\left| x^0 - y \right|^2 \geq \left| x^0 - \bar{x} \right|^2$. А это в свою очередь означает, что

$$-2\alpha(x^0 - \bar{x}, x - \bar{x}) + \alpha^2 \left| x - \bar{x} \right|^2 \geq 0, \quad \forall \alpha \in [0,1].$$

Откуда очевидно выполнение неравенства

$$(x^0 - \bar{x}, x - \bar{x}) \leq 0 \text{ или } (x - \bar{x}, x^0 - \bar{x}) \leq 0.$$

Докажем достаточность.

Пусть $\forall x \in \Omega$ справедливо неравенство

$$(x - \bar{x}, x^0 - \bar{x}) \leq 0.$$

Тогда справедлива оценка

$$\left| x - x^0 \right|^2 = \left| (x - \bar{x}) - (x^0 - \bar{x}) \right|^2 =$$

$$= \left| x - \bar{x} \right|^2 - 2(x - \bar{x}, x^0 - \bar{x}) + \left| x^0 - \bar{x} \right|^2 \geq \left| x^0 - \bar{x} \right|^2.$$

Следовательно, элемент $\bar{x} \in \Omega$ является проекцией элемента x^0 на Ω.

Теорема доказана.

§ 1.1.4. Условия отделимости выпуклых подмножеств

При построении и обосновании различных методов исследования математических моделей в E^n важную роль играют следующие теоремы.

Теорема 1.1.4.1 **Пусть $\Omega \subset E^n$ – выпуклое замкнутое множество. Тогда $\forall x^0 \notin \overline{\Omega}$ существует гиперплоскость $(l, x - x^0) = 0$ с $l \neq o$ такая, что**

$$(l, x - x^0) < 0 \ \forall x \in \Omega .$$

Доказательство.

Пусть элемент $\bar{x}$ является проекцией элемента x^0 на Ω. Выберем гиперплоскость $(l, x - x^0) = 0$ с ненулевым (в силу $\forall x^0 \notin \overline{\Omega}$) $l = x^0 - \bar{x}$, тогда, используя утверждение теоремы 1.1.3.2 и равенство

$$x - x^0 = (x - \bar{x}) - l ,$$

получаем оценку

$$(l, x - x^0) =$$

$$= (x^0 - \bar{x}, x - x^0) \leq (x^0 - \bar{x}, x - \bar{x}) - (l, l) < 0,$$

так как $l \neq o$.

Теорема доказана.

Теорема 1.1.4.2 **Пусть $\Omega \subset E^n$ – выпуклое замкнутое множество. Тогда для любого граничного элемента $\bar{x}$ этого множества существует *опорная* гиперплоскость $(l, x - \bar{x}) = 0$ с $l \neq o$ такая, что $(l, x - \bar{x}) \leq 0 \ \forall x \in \Omega$.**

Доказательство.

Согласно определению граничного элемента множества $\Omega \subset E^n$ существует последовательность элементов $\{x_k\}$ таких, что:

$1^\circ.$ $\quad x_k \notin \overline{\Omega} \quad \forall k ;$

$2^\circ.$ $\quad \lim\limits_{k \to \infty} x_k = \bar{x} ;$

По теореме 1.1.4.1 для каждого k существует гиперплоскость $(l_k, x - x_k) = 0$ такая, что:

$$3^\circ. \quad l_k = \frac{x_k - \bar{x}}{|x_k - \bar{x}|};$$

$$4^\circ. \quad (l_k, x - x_k) < 0 \quad \forall x \in \Omega.$$

В силу предположения о сходимости $\{x_k\}$ будет сходиться и $\{l_k\}$. Пусть $\lim\limits_{k\to\infty} l_k = l$, тогда, принимая во внимание, что предельный переход не нарушает нестрогих неравенств (теорема "*о двух милиционерах*"), из $\lim\limits_{k\to\infty}(l_k, x - x_k) \le 0$ получаем

$$(l, x - \bar{x}) \le 0 \ \forall x \in \Omega,$$

то есть гиперплоскость $(l, x - \bar{x}) = 0$ – опорная.

Теорема доказана.

Из курса выпуклого анализа известно, что:

1º. Если $\Omega \subset E^n$ – выпуклое множество, то множества $\overline{\Omega}$ и $\operatorname{int}\Omega$ также выпуклы.

2º. Если $\Omega \subset E^n$ и $\Theta \subset E^n$ – выпуклые множества, то множества

$$\Omega \pm \Theta = \{x \in E^n : x = x_1 \pm x_2 \ , \ \forall x_1 \in \Omega \ , \ \forall x_2 \in \Theta \}$$

также выпуклы.

Теорема 1.1.4.3 (О разделяющей гиперплоскости)

Пусть $\Omega \subset E^n$ и $\Theta \subset E^n$ – выпуклые множества такие, что любая внутренняя точка Ω не принадлежит Θ. Тогда существует разделяющая множества Ω и Θ гиперплоскость

$$(l, y - x) = 0 \quad \textbf{с} \quad l \ne o$$

такая, что

$$(l, y - x) \le 0, \quad \forall x \in \Omega \ \textbf{и} \ \forall y \in \Theta.$$

Доказательство.

Рассмотрим множество $\Theta - \operatorname{int}\Omega$, состоящее из элементов вида $y - x$, $\forall x \in \operatorname{int}\Omega$ и $\forall y \in \Theta$. Это множество выпуклое и не содержит по условию теоремы нулевого элемента.

Тогда в силу теорем 1.1.4.1 и 1.1.4.2 для каждого его внешнего элемента $y^0 - x^0$ существует гиперплоскость

$$(l,(y-x)-(y^0-x^0))=0 \quad \text{с} \quad l \neq o$$

такая, что $(l,(y-x)-(y^0-x^0)) \le 0$.

Поскольку элемент $y^0 - x^0 = o$ для рассматриваемого множества является внешним, то будет справедлива оценка $(l, y-x) \le 0$.

Наконец, включив путем соответствующего предельного перехода (не нарушающего нестрогие неравенства) в рассмотрение граничные точки множества Ω, получаем утверждение теоремы.

Теорема доказана.

Теорема 1.1.4.4. (Фаркаша)

Для того чтобы $\| A \| \| x \| = \| b \|$ – система m линейных уравнений с n неизвестными имела *неотрицательное* частное решение (то есть решение $\| x^0 \| \ge \| o \|$), необходимо и достаточно, чтобы $\| y \|$ – каждое частное решение системы линейных неравенств $\| A \|^{\mathrm{T}} \| y \| \le \| o \|$ – удовлетворяло условию

$$\| b \|^{\mathrm{T}} \| y \| \le 0.$$

Доказательство.

Докажем необходимость.

Пусть система линейных уравнений $\| A \| \| x \| = \| b \|$ имет "неотрицательное" частное решение, то есть, покомпонентно удовлетворяющее условию $\| x^0 \| \ge \| o \|$. Покажем, что в этом случае для каждого решения системы линейных неравенств $\| A \|^{\mathrm{T}} \| y \| \le \| o \|$

выполнено условие $\| b \|^{\mathrm{T}} \| y \| \le 0$. Действительно,

$$\| b \|^{\mathrm{T}} \| y \| = (\| A \| x^0 \|)^{\mathrm{T}} \| y \| = \| x^0 \|^{\mathrm{T}} (\| A \|^{\mathrm{T}} \| y \|) \le 0,$$

поскольку n-компонентная строка с неотрицательными элементами $\| x^0 \|^{\mathrm{T}}$ умножается справа на n-компонентный столбец $\| A \|^{\mathrm{T}} \| y \|$ с неположительными элементами.

Докажем достаточность.

Пусть матрица $\| A \|$ задает линейное отображение вида $\hat{A} : E^n \to E^m$, столбцы $\| b \|, \| y \|$ задают элементы $b, y \in E^m$, а столбцы $\| x \|, \| x^0 \|$ – элементы $x, x^0 \in E^n$. Обозначим через Ω множество всех элементов $v \in E^m$ таких, что $v = \hat{A}x \ \ \forall x \ge o$. Оно очевидно выпуклое. Если для *каждого* решения системы линейных неравенств $\| A \|^{\mathrm{T}} \| y \| \le \| o \|$ выполнено условие $\| b \|^{\mathrm{T}} \| y \| \le 0$ и при этом $b \in \Omega$, то достаточность доказана.

Допустим, что $b \notin \Omega$. Покажем, что в этом случае не для каждого решения системы линейных неравенств $\| A \|^{\mathrm{T}} \| y \| \le \| o \|$ выполнено условие $\| b \|^{\mathrm{T}} \| y \| \le 0$.

Действительно, пусть элемент $u \in \Omega \subset E^m$ – проекция b на Ω. Заметим, что здесь (без доказательства) мы предположили замкнутость Ω, которая гарантирует существование проекции.

Тогда для элемента $y' = b - u$ справедливы оценки:

1°. В силу теоремы 1.1.4.1 $(y', v - b) < 0 \ \forall v \in \Omega$, но поскольку $o \in \Omega$, то $(y', b) > 0$;

2°. По теореме 1.1.3.2 $(v - u, b - u) \le 0 \ \forall v \in \Omega$ или $(v - u, y') \le 0 \ \forall v \in \Omega$. Очевидно, что элемент $v + u$ также

будет принадлежать множеству Ω. Тогда из последнего неравенства получаем $(v, y') \le 0 \ \forall v \in \Omega$. Откуда следует оценка

$$(v, y') = (\hat{A}x, y') = (x, \hat{A}^{+} y') \le 0 \qquad \forall x \ge o .$$

В силу произвольности $\| x \| \ge \| o \|$ имеем $\| \hat{A}^{+} y' \| \le \| o \|$. То есть из $b \notin \Omega$ вытекает существование y' такого, что

$$\begin{cases} \| A \|^{\mathrm{T}} \| y' \| \le \| o \|, \\ \| b \|^{\mathrm{T}} \| y' \| > 0, \end{cases}$$

поскольку $\| \hat{A}^{+} \|_e = \| A \|^{\mathrm{T}}$.

Теорема доказана.

Поясним геометрический смысл теоремы Фаркаша следующим примером. Пусть $\hat{A} : E^3 \to E^2$ и матрица этого отображения имеет вид $\| \hat{A} \|_e = \begin{Vmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{Vmatrix}$, где

$$\left\| \vec{a}_1 \right\|_e = \begin{Vmatrix} \alpha_{11} \\ \alpha_{21} \end{Vmatrix}; \left\| \vec{a}_2 \right\|_e = \begin{Vmatrix} \alpha_{12} \\ \alpha_{22} \end{Vmatrix}; \left\| \vec{a}_3 \right\|_e = \begin{Vmatrix} \alpha_{13} \\ \alpha_{23} \end{Vmatrix}; \quad \text{и}$$

$$\left\| \vec{b} \right\|_e = \begin{Vmatrix} \beta_1 \\ \beta_2 \end{Vmatrix}; \quad \left\| \vec{y} \right\|_e = \begin{Vmatrix} \eta_1 \\ \eta_2 \end{Vmatrix}; \quad \left\| \vec{v} \right\|_e = \begin{Vmatrix} \nu_1 \\ \nu_2 \end{Vmatrix}; \quad \left\| \vec{x} \right\|_e = \begin{Vmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{Vmatrix}$$

– координатные представления соответствующих элементов (в данном случае векторов) в E^2 и в E^3.

В этом случае множество Ω состоит из линейных комбинаций вида

$$\vec{v} = \xi_1 \vec{a}_1 + \xi_2 \vec{a}_2 + \xi_3 \vec{a}_3$$

с *неотрицательными* коэффициентами и является замкнутым конусом.

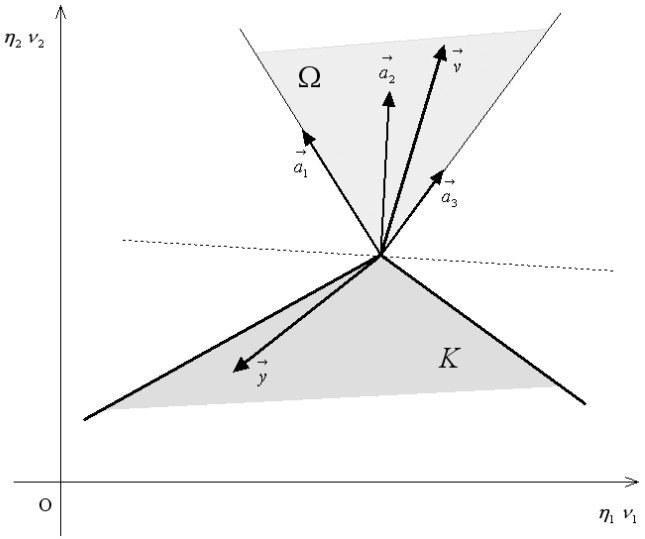


Рис. 1.1.4.1

Конусом также является и K – множество всех векторов вида

$$\{\ \vec{y}\ \mid\ \ \|A\|^{\mathrm{T}}\|y\| \le \|o\|\ \},$$

то есть образующих *неострые* углы с каждым из векторов $\vec{a}_1, \vec{a}_2$ и $\vec{a}_3$.

Тогда утверждение теоремы Фаркаша геометрически означает следующее:

для того, что *любой* вектор $\vec{y} \in K$ удовлетворял также и условию $(\vec{b}, \vec{y}) \le 0$ (то есть, чтобы угол между $\vec{b}$ и $\vec{y}$ был *неострым*) необходимо и достаточно, чтобы $\vec{b} \in \Omega$,

что иллюстрирует рис. 1.1.4.1.

Раздел 1.2. ФУНКЦИОНАЛЫ В КОНЕЧНОМЕРНОМ ЕВКЛИДОВОМ ПРОСТРАНСТВЕ

§ 1.2.1. Определение, классификация и способы задания функционалов

Определение 1.2.1.1 Будем говорить, что на множестве $\Omega \subseteq E^n$ задан *функционал* (*числовая функция*) $\varphi = F(x)$, если каждому элементу $x \in \Omega$ поставлено в соответствие единственное число φ, называемое *значением функционала на элементе* x.

Функционал в E^n можно обозначать и в координатном представлении в виде как $\varphi = F(\xi_1, \xi_2, \ldots, \xi_n)$. Напомним также некоторые, часто используемые в курсе математического анализа, определения.

Определение 1.2.1.2 *Изоуровнем* (*уровнем*, или c*–уровнем*) функционала $\varphi = F(x)$ называется множество элементов $x \in E^n$, удовлетворяющих равенству $F(x) = c$.

Графиком функционала $\varphi = F(x)$, заданного на $\Omega \subseteq E^n$, называется совокупность элементов y в E^{n+1}, имеющих координатное представление вида $\| y \|_e = \begin{Vmatrix} \| x \|_e \\ F(x) \end{Vmatrix}$.

Множество элементов y в E^{n+1}, имеющих координатное представление вида $\| y \|_e = \begin{Vmatrix} \| x \|_e \\ x_{n+1} \end{Vmatrix}$ таких, что $x_{n+1} \geq F(x)$, называется *надграфиком* функционала $u = F(x)$, заданного на $\Omega \subseteq E^n$.

Определение 1.2.1.3

Функционал $F(x)$ называется *выпуклым* (или *выпуклым вниз*) на множестве $\Omega \subseteq E^n$, если

$$\forall x_1, x_2 \in \Omega \text{ и } \forall \lambda \in [0,1]$$

справедливо неравенство

$$F((1-\lambda)x_1 + \lambda x_2) \le (1-\lambda)F(x_1) + \lambda F(x_2).$$

Если данное неравенство выполняется как строгое $\forall \lambda \in (0,1)$, то функционал называют *строго выпуклым* на множестве Ω.

Функционал $F(x)$ называется *вогнутым* (или *выпуклым вверх*) на множестве $\Omega \subseteq E^n$, если

$$\forall x_1, x_2 \in \Omega \text{ и } \forall \lambda \in [0,1]$$

справедливо неравенство

$$F((1-\lambda)x_1 + \lambda x_2) \ge (1-\lambda)F(x_1) + \lambda F(x_2).$$

Линейные, билинейные и квадратичные функционалы

Определение 1.2.1.4

Функционал $l(x)$ называется *линейным функционалом* (или *линейной формой*), если $\forall x, y \in E^n$ и любых чисел λ и μ:

$$l(\lambda x + \mu y) = \lambda\, l(x) + \mu\, l(y).$$

В евклидовом пространстве E^n с ортонормированным базисом линейный функционал $l(x)$ представим

1) в *матричном* виде $l(x) = \| l \|_e \| x \|_e$;
2) в *координатном* виде $l(x) = \sum_{j=1}^{n} \lambda_j \xi_j$;
3) в *символическом* виде $l(x) = (l, x)$,

где $\|x\|_e = \begin{Vmatrix} \xi_1 \\ \xi_2 \\ \dots \\ \xi_n \end{Vmatrix}$ и $\|l\|_e = \|\lambda_1 \quad \lambda_2 \quad \dots \quad \lambda_n\|$, – координатные представления элемента x и линейного функционала f в E^n, а $\lambda_k = l(e_k)$; $\forall k = [1, n]$.

Определение 1.2.1.5

Функционал $B(x, y)$, зависящий от двух аргументов x и y, называется *билинейным функционалом* (или б*илинейной формой*), если $\forall x, y, z \in \Lambda$ и любых чисел λ и μ справедливы равенства:

1º. $B(\lambda x + \mu y, z) = \lambda B(x, z) + \mu B(y, z)$;
2º. $B(z, \lambda x + \mu y) = \lambda B(z, x) + \mu B(z, y)$.

Функционал $B(x, y)$ называется *симметричным*, если $B(x, y) = B(y, x)$, $\forall x, y \in \Lambda$.

Определение 1.2.1.6

Функционал $\Phi(x)$ называется *квадратичным функционалом* (или *квадратичной формой*), если

$$\forall x \in \Lambda \quad \Phi(x) = B(x, x),$$

где $B(x, y)$ – некоторый симметричный билинейный функционал.

В конечномерном евклидовом пространстве E^n с ортонормированным базисом $\{e_1, e_2, \dots, e_n\}$ квадратичный функционал $\Phi(x)$ представим

1) в *матричном* виде $\Phi(x) = \|x\|_e^{\mathrm{T}} \|\Phi\|_e \|x\|_e$,

2) в *координатном* виде $\Phi(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} \phi_{ij}\xi_i\xi_j$,

3) в *символическом* виде $\Phi(x) = (x, \hat{\Phi}x)$,

где $\phi_{kj} = \Phi(e_k, e_j)$, $\forall k, j = [1, n]$, а $\hat{\Phi}$ – некоторый самосопряженный оператор, задаваемый в данном ортонормированном базисе симметрической матрицей $\|\Phi\|_e$.

В конечномерном евклидовом пространстве имеется возможность определять подмножества его элементов, накладывая ограничения на значения функционалов. Например, система условий (типа "равенство" и/или "нестрогое неравенство") вида

$$\begin{cases} f_1(x) \geq 0, \\ f_2(x) \geq 0, \\ \quad \dots \\ f_m(x) \geq 0, \end{cases}$$

где $f_1(x), f_2(x), \dots, f_m(x)$ – некоторые функционалы, задает подмножество U элементов в E^n.

Отметим, что если эти функционалы линейные и неоднородные, а ограничения выполняются как равенства, то U является *гиперплоскостью* в E^n. В случае однородных линейных функционалов U будет *подпространством* в E^n.

§ 1.2.2. Предел и непрерывность функционалов

Помимо значения функционала на элементе из области определения Ω другой важной числовой характеристикой функционала является его *предел*. Существуют два наиболее часто используемых, равносильных друг другу определения *предела функционала* в E^n.

Определение 1.2.2.1а (по Гейне)

Пусть область определения Ω функционала $F(x)$ содержит $U_\delta(a)$ кроме, быть может, самого элемента a. Тогда число α называется *пределом* этого функционала *на элементе* a, если для любой последовательности $\{x_k\}$, сходящейся к a, такой что $x_k \neq a;\ \forall k$ справедливо равенство

$$\lim_{k\to\infty} F(x_k) = \alpha.$$

Определение 1.2.2.1в (по Коши)

Число α называется *пределом* этого функционала *на элементе* a, если для любого числа $\varepsilon > 0$ существует такое число $\delta > 0$, что $\forall x : 0 < \rho(x,a) < \delta$ выполняется неравенство $\left| F(x) - \alpha \right| < \varepsilon$.

Равенство предела функционала $F(x)$ на элементе a числу α записывается в виде $\lim\limits_{x\to a} F(x) = \alpha$.

В общем случае существование и значение повторного предела зависит от порядка предельных переходов и не связано с существованием и значением предела функционала на элементе a.

Определение 1.2.2.2

Функционал $F(x)$ называется *непрерывным* на элементе a, если он определен в некоторой окрестности элемента a и $\lim\limits_{x\to a} F(x) = F(a)$.

Если функционал непрерывен на каждом элементе множества Ω, то он называется *непрерывным на множестве* Ω.

Известно (*теорема Вейерштрасса*), что непрерывный на компактном (то есть замкнутом и ограниченном) множестве элементов $\Omega \subset E^n$ функционал достигает на этом множестве как своей точной верхней, так и нижней граней.

Иначе говоря, в этом случае существуют элементы $x^* \in \Omega$ такие, что

$$F(x^*) = \sup_{x\in\Omega} F(x) = \max_{x\in\Omega} F(x)$$

или

$$F(x^*) = \inf_{x\in\Omega} F(x) = \min_{x\in\Omega} F(x).$$

Кроме того, отметим (без доказательства), что справедлива

Теорема 1.2.2.1 **Пусть $\Omega \subset E^n$ – выпуклое множество. Тогда выпуклый функционал $F(x)$, $x \in \Omega$, непрерывен в каждой внутренней точке Ω.**

Раздел 1.3. АППРОКСИМАЦИЯ ФУНКЦИОНАЛОВ В КОНЕЧНОМЕРНОМ ЕВКЛИДОВОМ ПРОСТРАНСТВЕ

§ 1.3.1. Частные производные функционалов

Частной производной первого порядка функционала $F(x) = F(\xi_1, \xi_2, ..., \xi_n)$ по переменной ξ_k на элементе x_0 с координатным представлением $\| x_0 \| = \begin{Vmatrix} \xi_{01} \\ \xi_{02} \\ \dots \\ \xi_{0n} \end{Vmatrix}$ называется число, равное пределу

$$\lim_{\Delta\xi_k \to 0} \frac{F(\xi_{01}, \xi_{02}, \dots, \xi_{0k} + \Delta\xi_k, \dots, \xi_{0n}) - F(\xi_{01}, \xi_{02}, \dots, \xi_{0k}, \dots, \xi_{0n})}{\Delta\xi_k}.$$

Частная производная первого порядка, обозначаемая как $\dfrac{\partial F}{\partial \xi_k}$, находится

для любого k по правилам дифференцирования функции одной переменной в предположении, что все компоненты элемента x, кроме ξ_k, постоянны.

Каждая частная производная $\dfrac{\partial F}{\partial \xi_k}$ сама является функционалом на элементе $x \in E^n$ и, следовательно, в координатной форме зависит от n независимых переменных $\{\xi_1, \xi_2, \ldots, \xi_n\}$.

Частную производную этого нового функционала по ξ_i называют *частной производной второго порядка по* ξ_j *и* ξ_i исходного функционала $F(x)$ и обозначают как $\dfrac{\partial^2 F}{\partial \xi_i \partial \xi_j}$. В случае $i = j$ используется запись вида $\dfrac{\partial^2 F}{\partial \xi_i^2}$. Частную производную по различным переменным принято называть *смешанной*.

Поскольку частные производные второго порядка в свою очередь также можно рассматривать как новые функционалы на элементе $x \in E^n$, то аналогичным способом возможно определение частных производных *третьего* и *более высоких* порядков. Например,

$$\frac{\partial^3 F}{\partial \xi_j^2 \partial \xi_i} = \frac{\partial}{\partial \xi_j}\left(\frac{\partial^2 F}{\partial \xi_j \partial \xi_i}\right).$$

Для смешанных производных справедливо следующее утверждение:

> Если две смешанные производные *одного и того же порядка, отличающиеся порядком дифференцирования*, непрерывны на некотором элементе $x \in E^n$, то они имеют на этом элементе *равные значения*.

§ 1.3.2. Производные по направлению. Градиент и гессиан функционала

Из определения частных производных следует, что они характеризуют величину изменения значения функционала при вариациях аргументов *коллинеарно базисным элементам*, то есть вдоль координатных осей в пространстве E^n. В этом случае представляется естественным использование *произвольного* направления вариации элемента, в окрестности которого исследуется поведение функционала.

Пусть в E^n задан фиксированный элемент w с координатным представлением $\|w\| = \|\omega_1 \quad \omega_2 \quad \dots \quad \omega_n\|^{\mathrm{T}}$ единичной длины, то есть с $\sum_{k=1}^{n} \omega_k^2 = 1$, определяющий в E^n некоторое направление, тогда можно дать

Определение 1.3.2.1 *Производной функционала* $F(x)$ *по направлению* w на элементе x_0 называется предел вида

$$\lim_{\tau \to +0} \frac{F(\xi_{01} + \tau\omega_1, \xi_{02} + \tau\omega_2, \dots, \xi_{0n} + \tau\omega_n) - F(\xi_{01}, \xi_{02}, \dots, \xi_{0n})}{\tau}.$$

Отметим, что в случае ортонормированного базиса можно использовать альтернативную форму координатного представления $\|w\|$.

Действительно, если обе части разложения $w = \sum_{j=1}^{n} \omega_j e_j$ последовательно умножить скалярно на e_i, $i = [1, n]$, то мы получим $\omega_i = (w, e_i)$, $\forall i = [1, n]$. Но поскольку $|w| = |e_i| = 1$, то будут справедливы равенства

$$\omega_i = \frac{(w, e_i)}{|w|\,|e_i|} = \cos\alpha_i, \ \forall i = [1, n],$$

где α_i – угол между единичным элементом w и базисным ортом e_i.

Таким образом, мы приходим к новому виду координатного представления элемента w, задающего направление,

$$\|w\| = \begin{Vmatrix} \cos\alpha_1 \\ \cos\alpha_2 \\ \cdots \\ \cos\alpha_n \end{Vmatrix}, \text{ причем } \sum_{j=1}^{n} \cos^2 \alpha_j = 1.$$

Производная по направлению функционала $F(x)$ в свою очередь также является функционалом, но уже зависящим как от элемента x_0, так и от элемента w. Его принято обозначать $\dfrac{\partial F}{\partial w}$. Заметим, что варьирование аргумента функционала происходит в линейном одномерном многообразии (на *гиперлуче*) вида

$$x = x_0 + \tau w; \quad \forall \tau \in [0, +\infty),$$

что позволяет рассматривать этот функционал как функцию одного переменного

$$\varphi(\tau) = F(x_0 + \tau w) \quad \forall \tau \in [0, +\infty).$$

Имеет место следующая важная для приложений

Теорема 1.3.2.1 **Пусть $\Omega \subset E^n$ – выпуклое множество. Тогда, определенный на Ω, выпуклый функционал имеет на каждом внутреннем элементе Ω производную по любому направлению**.

Доказательство.

Из курса математического анализа известно, что для любой выпуклой на некотором интервале (α, β) функции одного аргумента $\varphi(\tau)$ выражение

$$\frac{\varphi(\tau) - \varphi(\tau_0)}{\tau - \tau_0}$$

ограничено снизу на $(\tau_0, \tau) \subset (\alpha, \beta)$ и монотонно убывает при $\tau \to \tau_0 + 0$.

Поэтому для $\varphi(\tau)$ в $\tau_0 \in (\alpha, \beta)$ существует правосторонняя производная.

Если на элементе $x_0 \subset \Omega \subseteq E^n$ для выпуклого функционала $F(x)$ задано направление w, то очевидно, что

$$\varphi(\tau) = F(x_0 + \tau w), \quad \forall \tau \in [0, +\infty)$$

– выпуклая функция одного аргумента τ и для нее существует правосторонняя производная

$$\left.\frac{d\varphi}{d\tau}\right|_{\tau_0}^{+} = \frac{\partial F}{\partial w}.$$

Теорема доказана.

Получим теперь связь значения производной по направлению со значениями частных производных.

Пусть функционал $F(x)$ имеет на некотором элементе x все первые частные производные. Тогда упорядоченный набор из n чисел

$$\left\{ \frac{\partial F}{\partial \xi_1} \quad \frac{\partial F}{\partial \xi_2} \quad \cdots \quad \frac{\partial F}{\partial \xi_n} \right\}$$

будет задавать некоторый элемент в E^n с координатным представлением

$$\left\| \frac{\partial F}{\partial \xi_1} \quad \frac{\partial F}{\partial \xi_2} \quad \cdots \quad \frac{\partial F}{\partial \xi_n} \right\|^{\mathrm{T}}.$$

Этот элемент называется *градиентом* функционала $F(x)$ на элементе x и обозначается как $\operatorname{grad} F$ или ∇F, то есть,

$$\| \operatorname{grad} F \| \equiv \| \nabla F \| = \left\| \begin{array}{c} \dfrac{\partial F}{\partial \xi_1} \\ \dfrac{\partial F}{\partial \xi_2} \\ \cdots \\ \dfrac{\partial F}{\partial \xi_n} \end{array} \right\|.$$

Тогда, согласно правилу дифференцирования суперпозиции функций, формула для производной по направлению будет иметь вид

$$\frac{\partial F}{\partial w} = \sum_{j=1}^{n} \frac{\partial F}{\partial \xi_j} \cos \alpha_j, \quad \text{или} \quad \frac{\partial F}{\partial w} = (\operatorname{grad} F, w). \qquad (1.3.2.1)$$

В матричном же виде это равенство может быть записано как

$$\frac{\partial F}{\partial w} = \| \operatorname{grad} F \|^{\mathrm{T}} \| w \|.$$

Заметим, что в силу неравенства Коши–Буняковского и условия нормировки направления $|w| = 1$, из (1.3.2.1) следуют важные для вычислительной практики оценки

$$-| \operatorname{grad} F | \le \frac{\partial F}{\partial w} = (\operatorname{grad} F, w) \le | \operatorname{grad} F |.$$

Иначе говоря, $\max\limits_{w} \dfrac{\partial F}{\partial w} = | \operatorname{grad} F |$ и $\min\limits_{w} \dfrac{\partial F}{\partial w} = -| \operatorname{grad} F |$.

Рассуждая аналогично случаю производной по направлению первого порядка, можно получить формулу для $\dfrac{\partial^2 F}{\partial w^2} = \dfrac{\partial}{\partial w}\left(\dfrac{\partial F}{\partial w}\right)$. Действительно, применив правило дифференцирования сложной функции по компонентам элемента x_0 к функционалу $\dfrac{\partial F}{\partial w}$, находим, что в координатном виде

$$\frac{\partial^2 F}{\partial w^2} = \sum_{j=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} \cos \alpha_j \cos \alpha_i .$$

С другой стороны, если учесть, что в ортонормированном базисе конечномерного евклидова пространства E^n квадратная, порядка n, симметрическая матрица $\left\| \dfrac{\partial^2 F}{\partial \xi_j \partial \xi_i} \right\|$ – матрица *Гессе,* задает линейный *самосопряженный* оператор, называемый *гессианом* и обозначаемый как

$$\hat{\operatorname{Hess}} F \quad \text{или} \quad \nabla^2 F,$$

то формула для $\dfrac{\partial^2 F}{\partial w^2}$ может быть записана как в матричной, так и в символической формах:

$$\frac{\partial^2 F}{\partial w^2} = \| w \|^{\mathrm{T}} \left\| \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} \right\| \| w \| \quad \text{или} \quad \frac{\partial^2 F}{\partial w^2} = (w, \hat{\mathrm{Hess}}\, F\, w).$$

Напомним, что все собственные значения самосопряженного оператора вещественны, собственные векторы, отвечающие различным собственным значениям, ортогональны, а из его собственных векторов всегда можно образовать ортонормированный базис.

§ 1.3.3. Дифференциалы функционала

Определение 1.3.3.1 Функционал $F(x)$ называется *дифференцируемым на элементе* x_0, если $\exists g \in E^n$ такой, что в некоторой окрестности x_0 справедливо равенство

$$F(x) - F(x_0) = (g, \Delta x) + \mathrm{o}(|\Delta x|),$$

при $|\Delta x| \to 0$, где $\Delta x = x - x_0$.

Если координатные представления элементов g и Δx суть

$$\| g \| = \| \gamma_1 \quad \gamma_2 \quad \dots \quad \gamma_n \|^{\mathrm{T}} \quad \text{и} \quad \| \Delta x \| = \| \Delta\xi_1 \quad \Delta\xi_2 \quad \dots \quad \Delta\xi_n \|^{\mathrm{T}},$$

то условие дифференцируемости функционала $F(x)$ на элементе x_0 может быть записано в матричном виде как

$$F(x) - F(x_0) = \| g \|^{\mathrm{T}} \| \Delta x \| + \mathrm{o}(|\Delta x|), \quad \text{при } |\Delta x| \to 0,$$

или в координатах

$$F(\xi_1, \xi_2, \dots, \xi_n) - F(\xi_{01}, \xi_{02}, \dots, \xi_{0n}) =$$

$$= \sum_{j=1}^{n} \gamma_j \Delta\xi_j + \mathrm{o}\left(\sqrt{\sum_{j=1}^{n} \Delta\xi_j^2} \right),$$

где $\Delta\xi_j = \xi_j - \xi_{0j}$, $\forall j = [1, n]$ и $\Delta x \to o$.

В приведенных формулах следует обращать внимание на различие обозначений o и o:

$\mathrm{o}(\tau)$ – функция *одной переменной* такая, что $\lim\limits_{\tau \to 0} \dfrac{\mathrm{o}(\tau)}{\tau} = 0$,

o – *нулевой элемент* в E^n.

Первый дифференциал функционала и его представления

Определение 1.3.3.2 Если функционал $F(x)$ дифференцируем на элементе x_0, то линейный по Δx функционал

$$(g, \Delta x) = \sum_{j=1}^{n} \gamma_j \Delta\xi_j$$

называется *первым дифференциалом функционала* $F(x)$ на элементе x_0 и обозначается dF.

При этом на элементе x_0 существуют первые частные производные $\dfrac{\partial F}{\partial \xi_j}$, $\forall j = [1, n]$, и будут справедливы равенства

$$\gamma_j = \frac{\partial F}{\partial \xi_j}, \ \forall j = [1, n].$$

По определению принимается, что для *независимой переменной* ее дифференциал равен приращению, то есть $dx = \Delta x$. Тогда окончательно приходим к формуле

$$dF = \sum_{j=1}^{n} \frac{\partial F}{\partial \xi_j} d\xi_j \ .$$

Отметим, что если из дифференцируемости вытекает существование первых частных производных, то обратное, вообще говоря, неверно: функционал может иметь на некотором элементе все первые частные производные, но не быть при этом дифференцируемым (и даже не быть непрерывным) на этом элементе.

Заметим, что формула приращения значения дифференцируемого функционала может быть записана (в силу соотношения $g = \operatorname{grad} F$) в виде

$$F(x) - F(x_0) = (\operatorname{grad} F, dx) + \mathrm{o}(\left| dx \right|), \text{ при } \left| dx \right| \to 0, \quad (1.3.3.1)$$

где элемент dx считается по определению равным $\Delta x = x - x_0$. А сам дифференциал представим в символической форме $dF = (\operatorname{grad} F, dx)$ или же в матричной $dF = \| \operatorname{grad} F \|^{\mathrm{T}} \| dx \|$.

Второй дифференциал функционала и его представления. Дифференциалы высших порядков

Для функционалов в E^n также возможно определить и дифференциалы высших порядков. Действительно, первый дифференциал

$$dF = (\operatorname{grad} F, dx) = \sum_{k=1}^{n} \frac{\partial F}{\partial \xi_k} d\xi_k$$

в E^n сам по себе является некоторым функционалом, определенным на паре элементов $x,\ dx \in E^n$. Иначе говоря, dF зависит в координатном представлении от $2n$ скалярных независимых переменных

$$\{\xi_1, \xi_2, \ldots, \xi_n, d\xi_1, d\xi_2, \ldots, d\xi_n\},$$

причем от последних n – линейно.

Зафиксируем значения приращений $\{d\xi_1, d\xi_2, \ldots, d\xi_n\}$ и найдем дифференциал для dF, считая его функционалом зависящим только от $\{\xi_1, \xi_2, \ldots, \xi_n\}$ и используя в качестве значений независимых переменных $\{d\xi_1, d\xi_2, \ldots, d\xi_n\}$, *те же значения*, что использовались ранее при нахождении первого дифференциала. Построенный таким образом новый дифференциал называют *дифференциалом второго порядка* функционала F на элементе $x \in E^n$ и обозначают d^2F.

Получим теперь формулу для d^2F.

$$d^2F = d(dF) = d\Big(\sum_{j=1}^{n} \frac{\partial F}{\partial \xi_j} d\xi_j\Big) = \sum_{k=1}^{n} d\Big(\frac{\partial F}{\partial \xi_j}\Big) d\xi_j =$$

$$= \sum_{j=1}^{n} \Big(\sum_{i=1}^{n} \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} d\xi_i\Big) d\xi_j = \sum_{j=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} d\xi_j d\xi_i.$$

Откуда следует, что в E^n второй дифференциал от $F(x)$ является *квадратичным функционалом* элемента dx с координатным представлением $\| dx \| = \begin{Vmatrix} d\xi_1 \\ d\xi_2 \\ \ldots \\ d\xi_n \end{Vmatrix}$.

При помощи *гессиана* – оператора $\hat{\text{Hess}}\, F$ с матрицей $\| \text{Hess}\, F \| = \left\| \frac{\partial^2 F}{\partial \xi_k \partial \xi_i} \right\|$ полученное координатное представление для d^2F может быть записано как в матричном, так и символическом виде

$$d^2F = \| dx \|^{\mathrm{T}} \| \text{Hess}\, F \| \| dx \| \quad \text{или} \quad d^2F = (dx, \hat{\text{Hess}}\, F\, dx),$$

а, поскольку матрица линейного оператора $\hat{\mathrm{Hess}}\,F$ в рассматриваемом ортонормированном базисе симметрическая, то этот оператор *самосопряженный*. С другой стороны, можно утверждать, что гессиан[2] (как симметрическая матрица) в любом базисе конечномерного линейного пространства порождает квадратичный функционал d^2F, который, кроме того, будет иметь *диагональный* вид в базисе из собственных векторов оператора $\hat{\mathrm{Hess}}\,F$.

Наконец, естественное обобщение понятия дифференциала на случай k-го порядка имеет вид

$$d^kF = \sum_{\{\alpha_1+\alpha_2+\ldots+\alpha_n=k\}} A^k_{\alpha_1,\alpha_2,\ldots,\alpha_n}(d\xi_1)^{\alpha_1}(d\xi_2)^{\alpha_2}\ldots(d\xi_n)^{\alpha_n},$$

где

$$A^k_{\alpha_1,\alpha_2,\ldots,\alpha_n} = \frac{k!}{\alpha_1!\alpha_2!\ldots\alpha_n!}\,\frac{\partial^k F(x_0)}{\partial\xi_1^{\alpha_1}\partial\xi_2^{\alpha_2}\ldots\partial\xi_n^{\alpha_n}}$$

и $\alpha_1+\alpha_2+\ldots+\alpha_n = k \quad \forall k=[1,N]$.

§ 1.3.4. Формула Тейлора

Аналогично случаю функции одной переменной, для функционала $F(x)$ в E^n можно поставить задачу его *наилучшей аппроксимации* в малой окрестности элемента x_0 линейной комбинацией *алгебраических многочленов*, степени не выше N. В общем виде эта задача сводится к поиску таких значений коэффициентов $\sigma^m_{\alpha_1,\alpha_2,\ldots,\alpha_n}$ в линейной комбинации

---

[2] Термин *гессиан* иногда используется также для обозначения как определителя матрицы Гессе, так и квадратичного функционала d^2F..

$$F(x)-F(x_0)=$$
$$=\sum_{k=1}^{N}\sum_{\{\alpha_1+\alpha_2+\ldots+\alpha_n=k\}}\sigma^{k}_{\alpha_1,\alpha_2,\ldots,\alpha_n}(\xi_1-\xi_{01})^{\alpha_1}(\xi_2-\xi_{02})^{\alpha_2}\ldots(\xi_n-\xi_{0n})^{\alpha_n}$$
$$+R(x,x_0),$$

(1.3.4.1)

которые обеспечат выполнение предельного равенства

$$\lim_{x\to x_0}\frac{R(x,x_0)}{\left|x-x_0\right|^{N}}=0.$$

Теорема *Тейлора* утверждает, что (в предположении существования и непрерывности всех частных производных функционала $F(x)$ на элементе x_0 до порядка $N+1$ включительно) искомые значения коэффициентов равняются

$$\sigma^{k}_{\alpha_1,\alpha_2,\ldots,\alpha_n}=\frac{1}{k!}A^{k}_{\alpha_1,\alpha_2,\ldots,\alpha_n}=\frac{1}{\alpha_1!\alpha_2!\ldots\alpha_n!}\frac{\partial^k F(x_0)}{\partial\xi_1^{\alpha_1}\partial\xi_2^{\alpha_2}\ldots\partial\xi_n^{\alpha_n}},$$

где $\alpha_1+\alpha_2+\ldots+\alpha_n=k$ и $\forall k=[1,N]$.

В этом случае равенство (1.3.4.1) называется *формулой Тейлора* для функционала $F(x)$ в окрестности элемента x_0. Отметим, что эту формулу, обозначив $\rho(x,x_0)=\left|x-x_0\right|$, можно записать также в виде

$$F(x)-F(x_0)=\sum_{k=1}^{N}\frac{1}{k!}d^kF+o(\rho^N(x,x_0)).$$

Особый практический интерес представляет случай $N=2$ – квадратичной аппроксимации функционала, для которого координатная, матричная и символическая формы записи формулы Тейлора соответственно имеют вид

$$F(x) - F(x_0) =$$
$$= \sum_{j=1}^{n} \frac{\partial F}{\partial \xi_j} d\xi_j + \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} d\xi_j d\xi_i + \mathrm{o}(\rho^2(x, x_0)),$$

(1.3.4.2)

$$F(x) - F(x_0) =$$
$$= \| \operatorname{grad} F \|^{\mathrm{T}} \| dx \| + \frac{1}{2} \| dx \|^{\mathrm{T}} \| \operatorname{Hess} F \| \| dx \| + \mathrm{o}(\rho^2(x, x_0)),$$

(1.3.4.3)

$$F(x) - F(x_0) =$$
$$= (\operatorname{grad} F, dx) + \frac{1}{2} (dx, \overset{\Lambda}{\operatorname{Hess}} F\, dx) + \mathrm{o}(\rho^2(x, x_0)).$$

(1.3.4.4)

§ 1.3.5. Недифференцируемые функционалы. Субдифференциал

Необходимым условием существования тейлоровской аппроксимации функционала $F(x)$ в окрестности некоторого элемента $x_0 \in E^n$ является дифференцируемость $F(x)$ на этом элементе. Однако в большом числе практически важных задач это условие оказывается не выполненным. Например, в случаях, когда функционал $F(x)$ является "негладкой" зависимостью, определяемой, скажем, при помощи операторов максимизации (или минимизации).

Если функционал $F(x)$ имеет градиент в $U_\delta(x_0)$ – некоторой δ-окрестности элемента x_0, но при этом $\operatorname{grad} F(x_0)$ не существует, то для исследования локальных свойств $F(x)$ можно использовать так называемый обобщенный дифференциал.

Определение 1.3.5.1.

Обобщенным дифференциалом (или *субдифференциалом*) функционала $F(x)$ на элементе $x_0 \in \Omega \subset E^n$, называется множество ∂F элементов $l \in E^n$, таких, что

$$F(x) - F(x_0) \geq (l, x - x_0)\ ,\ \forall x \in \Omega\ .$$

Каждый элемент $l \in \partial F$ называется *обобщенным градиентом* (или *субградиентом*) функционала $F(x)$ на элементе x_0.

Отметим следующие, полезные для приложений, свойства субдифференциалов *выпуклых функционалов* на *выпуклых множествах*:

- если $F(x)$ – выпуклый функционал на выпуклом множестве Ω, то ∂F существует $\forall x_0 \in \operatorname{int}\Omega$ и является выпуклым и замкнутым множеством;
- если $\partial F(x_0) \neq \varnothing\ \forall x_0 \in \Omega$, то $F(x)$ – выпуклый функционал на Ω;
- если $F(x)$ дифференцируем на $x_0 \in \operatorname{int}\Omega$, то $\partial F = \{\nabla F(x_0)\}$;
- если $\partial F(x_0) = \{a\}$ (то есть $a \in E^n$ – единственный элемент субдифференциала на элементе $x_0 \in \operatorname{int}\Omega$), то $F(x)$ дифференцируем на x_0, причем $\nabla F(x_0) = a$;
- если функционалы $F_i(x), i = [1, m]$ выпуклые на выпуклом множестве Ω и $F(x) = \sum_{i=1}^{m} F_i(x)$, то $\partial F(x) = \sum_{i=1}^{m} \partial F_i(x)$.

Заметим, что для граничных точек данные свойства могут не иметь места. Например, в E^1 для выпуклого функционала $F(x) = -\sqrt{1 - \xi_1^2}$

на выпуклом множестве $[-1,1]$ субдифференциалы $\partial F(-1)$ и $\partial F(1)$ не существуют.

Понятие субдифференциала допускает следующую геометрическую интерпретацию. Пусть

$$l \in \partial F(x_0)\,,\; x_0 \in \operatorname{int}\Omega \subset E^n,$$

тогда каждый такой элемент l является нормальным для некоторой опорной гиперплоскости $(l, x - x_0) = 0$ множества

$$\{x \mid F(x) \le F(x_0)\}$$

на элементе x_0 (см. рис.1.3.5.1).

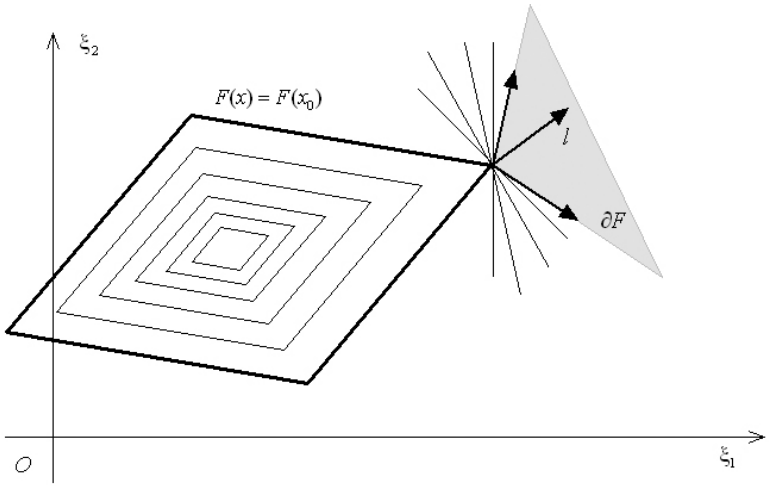


Рис. 1.3.5.1

Глава 2
КЛАССИЧЕСКИЕ ЭКСТРЕМАЛЬНЫЕ ЗАДАЧИ В E^n

Раздел 2.1. БЕЗУСЛОВНЫЙ ЭКСТРЕМУМ ФУНКЦИОНАЛА

§ 2.1.1. Определение максимума, минимума и седлового элемента

Определение 2.1.1.1

Говорят, что функционал $F(x)$ имеет *локальный максимум* (*минимум*) на элементе $x^* \in E^n$, если существует $U_\delta(x^*)$ – окрестность этого элемента такая, что

$$F(x) \le F(x^*)\ ,\ \forall x \in U_\delta(x^*)$$

(для случая *минимума*

$$F(x) \ge F(x^*)\ ,\ \forall x \in U_\delta(x^*)\ ).$$

В этом случае принята терминология: x^* – *экстремальный элемент* для $F(x)$, а $F(x^*)$ – *экстремальное значение* (*экстремум*) функционала $F(x)$. Если, кроме того, для всех элементов окрестности $U_\delta(x^*)$, $x \ne x^*$, справедливо неравенство $F(x) < F(x^*)$ (или $F(x) > F(x^*)$), то говорят о *строгом экстремуме* функционала $F(x)$.

Задачу поиска экстремального элемента

$$x^* = \arg\max_{x \in D} F(x)$$

на D – открытом множестве в области определения функционала $F(x)$, принято называть *классической экстремальной задачей*, или *задачей безусловной оптимизации*.

Определение 2.1.1.2

Седловым элементом функционала $F(x)$ называется элемент x^* такой, что при переносе начала координат в элемент x^* найдутся два подпространства Ω_1 и Ω_2 $\Omega_1 \oplus \Omega_2 = E^n$ такие, что

$$x^* = \arg\max_{x \in \Omega_1} F(x) \quad \text{и} \quad x^* = \arg\min_{x \in \Omega_2} F(x).$$

§ 2.1.2. Необходимые условия экстремума

Введем для задачи поиска максимального значения функционала понятие вариаций, "*улучшающих функционал* $F(x)$" на элементе $x^0 \in D \subseteq E^n$.

Определение 2.1.2.1

Пусть для некоторого элемента $\delta x \in E^n$ найдется число $\varepsilon > 0$ такое, что $x^0 + \tau \delta x \in D$ и

$$F(x^0 + \tau \delta x) > F(x^0), \forall \tau : 0 < \tau < \varepsilon.$$

Тогда $P(x^0)$ – совокупность всех таких элементов δx назовем *множеством улучшающих функционал вариаций на элементе* x^0.

Необходимое условие существования экстремума функционала $F(x)$ на элементе $x^* \in E^n$ формулирует следующая теорема.

Теорема 2.1.2.1 **Для того чтобы функционал $F(x)$ имел экстремум на элементе x^*, необходимо, чтобы на этом элементе градиент функционала либо *не существовал*, либо *равнялся нулевому элементу* в E^n.**

Доказательство.

Действительно, пусть $\operatorname{grad} F$ существует на максимальном элементе x^* и $\operatorname{grad} F \neq o$, тогда из соотношения

$$F(x^* + dx) - F(x^*) = (\operatorname{grad} F, dx) + \mathrm{o}(|\, dx\,|)$$

следует, что если для некоторой ненулевой, достаточно малой по норме не улучшающей вариации dx выполнено условие

$$F(x^* + dx) < F(x^*),$$

то будет верно и неравенство

$$F(x^* - dx) > F(x^*).$$

То есть, $-dx$ – улучшающая вариация и $F(x^*)$ не является максимальным значением для функционала $F(x)$.

Случай минимума рассматривается аналогично.

Теорема доказана.

Элементы в E^n, для которых $\operatorname{grad} F = o$ или не существует, называются *критическими* элементами функционала $F(x)$. Для дифференцируемых функционалов элементы в E^n, для которых $\operatorname{grad} F = o$, принято называть *стационарными*.

Следует отметить, что если на элементе x^* функционал не имеет градиента, но существует *субдифференциал*, то необходимое условие экстремума на x^* имеет вид

$$o \in \partial F(x^*).$$

Необходимое условие экстремума не является достаточным. Это следует из факта отсутствия экстремума на нулевом элементе у функционала $F(x) = \xi_1 \xi_2$, $x \in E^2$, хотя на этом элементе его градиент

$$\| \operatorname{grad} F \| = \left\| \frac{\partial F}{\partial \xi_1} \quad \frac{\partial F}{\partial \xi_2} \right\|^{\mathrm{T}} = \left\| \begin{matrix} \xi_2 \\ \xi_1 \end{matrix} \right\|$$

нулевой, то есть $\operatorname{grad} F(x) \big|_{x=o} = o$.

§ 2.1.3. Достаточные условия экстремума

Достаточные условия существования (или отсутствия) строгого экстремума функционала $F(x)$ на элементе $x^* \in E^n$ дает

Теорема 2.1.3.1 **Пусть функционал $F(x)$ дважды непрерывно дифференцируем в некоторой окрестности стационарного элемента x^*. Тогда у функционала $F(x)$ на x^*:**

1°. имеется *строгий минимум*, если на этом элементе

$$d^2 F = (dx, \hat{\operatorname{Hess}} F\, dx) > 0 \,, \ \forall dx \neq o \,;$$

2°. имеется *строгий максимум*, если на этом элементе

$$d^2 F = (dx, \hat{\operatorname{Hess}} F\, dx) < 0 \,, \ \forall dx \neq o \,;$$

3. отсутствует строгий экстремум, если на этом элементе $d^2 F = (dx, \hat{\operatorname{Hess}} F\, dx)$ не является знакопостоянным.

Доказательство.

Действительно, из формулы Тейлора (1.3.5) при условии стационарности x_0 следует, что

$$F(x_0 + dx) - F(x_0) = \frac{1}{2} \| dx \|^{\mathrm{T}} \| \operatorname{Hess} F \| \| dx \| + \mathrm{o}(| dx |^2),$$

но тогда для достаточно малых по норме dx знак приращения функционала будет зависеть от знаковой определенности *квадратичного функционала*

$$d^2 F = (dx, \hat{\operatorname{Hess}} F \, dx) > 0 ,$$

порождаемого в E^n симметрической матрицей Гессе.

Теорема доказана.

Как известно из курса линейной алгебры, исследование квадратичного функционала на знаковую определенность можно выполнять либо методом приведения его координатного представления к *диагональному виду*, построив специальный базис, либо непосредственно в исходном базисе – при помощи *критерия Сильвестра*.

Использование критерия Сильвестра позволяет утверждать, что для дважды непрерывно дифференцируемого функционала $F(x)$ на стационарном элементе:

1º. Имеет строгий минимум, если на этом элементе у матрицы Гессе $\left\| \dfrac{\partial^2 F}{\partial \xi_j \partial \xi_i} \right\|$ все главные диагональные миноры *положительные*;

2º. Имеет строгий максимум, если на этом элементе у матрицы Гессе $\left\| \dfrac{\partial^2 F}{\partial \xi_j \partial \xi_i} \right\|$ все четные главные диагональные миноры *положительные*, а все нечетные – *отрицательные.*

3º. Не имеет строго экстремума в остальных случаях.

В качестве примера рассмотрим в E^2 функционал

$$F(\xi_1, \xi_2) = \xi_1^3 + \xi_2^3 - 3\xi_1\xi_2 ,$$

градиент которого имеет координатное представление

$$\left\| \operatorname{grad} F \right\| = \left\| \begin{matrix} 3\xi_1^2 - 3\xi_2 \\ 3\xi_2^2 - 3\xi_1 \end{matrix} \right\|,$$

а гессиан представляется матрицей $\left\| \overset{\wedge}{\operatorname{Hess}} F \right\| = \left\| \begin{matrix} 6\xi_1 & -3 \\ -3 & 6\xi_2 \end{matrix} \right\|$.

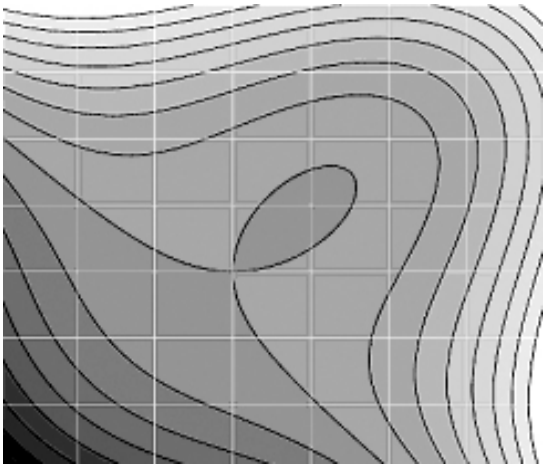


Рис. 2.1.3.1 Система изолиний для $F(\xi_1, \xi_2) = \xi_1^3 + \xi_2^3 - 3\xi_1\xi_2$

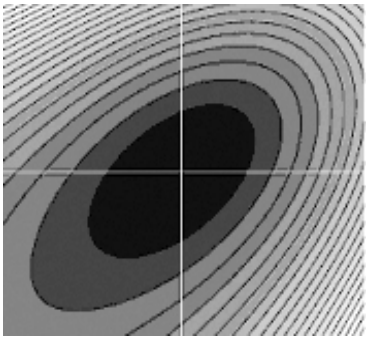


Рис. 2.1.3.2. Окрестность x_1

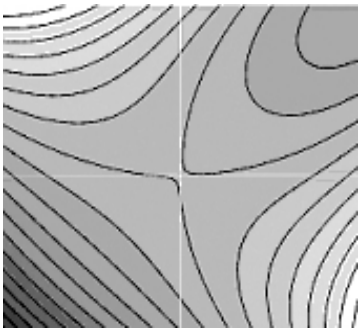


Рис. 2.1.3.3. Окрестность x_2

Легко видеть, что рассматриваемый функционал определен $\forall x \in E^2$ и имеет только два стационарных элемента:

$$\| x_1 \| = \begin{Vmatrix} 1 \\ 1 \end{Vmatrix} \quad \text{и} \quad \| x_2 \| = \begin{Vmatrix} 0 \\ 0 \end{Vmatrix},$$

причем (согласно критерию Сильвестра) на x_1 $F(\xi_1, \xi_2)$ он имеет *локальный минимум*, в то время как, x_2 – *седловой элемент*, не являющийся экстремальным.

На рис. 2.1.3.1 приведена система изолиний для данного функционала, а на рис. 2.1.3.2 и 2.1.3.3 – системы изолиний в окрестности стационарных элементов x_1 и x_2.

Условия теоремы 2.1.3.1 не являются, вообще говоря, *необходимыми условиями экстремума*. Например, в E^2 для функционала $\xi_1^4 + \xi_2^4$ в начале координат имеется строгий минимум.

Отметим, однако, что даже существование окрестности стационарного элемента с определенной локальной выпуклостью может не являться необходимым условием строгого экстремума, что иллюстрируется следующим примером. В E^1 у функционала

$$F(\xi_1) = \begin{cases} \xi_1^6 \left(2 + \sin \dfrac{1}{\xi_1} \right), & \xi_1 \neq 0, \\ 0, & \xi_1 = 0 \end{cases}$$

имеется строгий минимум при $\xi_1 = 0$, хотя в силу

$$F''(\xi_1) = \begin{cases} -\xi_1^2 \sin \dfrac{1}{\xi_1} + o(\xi_1^2), & \xi_1 \neq 0, \\ 0, & \xi_1 \neq 0 \end{cases}$$

в любой, сколь угодно малой окрестности начала координат имеются области локальной выпуклости как вниз, так и вверх.

Раздел 2.2. МЕТОДЫ ПОИСКА БЕЗУСЛОВНОГО ЭКСТРЕМУМА В E^n

§ 2.2.1. Общая схема поиска локального экстремума

Вполне очевидно, что метод поиска в E^n локальных экстремумов функционала $F(\xi_1, \xi_2, \ldots, \xi_n)$, основанный на исследовании критических его элементов (или же решений уравнения $\operatorname{grad} F(x) = o$), практически пригоден лишь для крайне ограниченного числа случаев. Реально реализуемой альтернативой является использование так называемых *итеративных численных методов*. Все эти методы также основаны на необходимых или достаточных условиях экстремума функционала в E^n и могут быть представлены в виде следующей схемы поиска, например, максимума:

1º. Для некоторого начального элемента x^0 находятся ненулевое *улучшающее направление максимизации* $w^0 \in E^n$ и положительное число $\sigma_0 < +\infty$ – *величина шага* по данному направлению – такие, что на элементе $x^1 = x^0 + \sigma_0 \cdot w^0$ верно неравенство $F(x^1) > F(x^0)$.

2º. Если задача в пункте 1º решена успешно, то элемент x^0 заменяется на x^1 и процедура пункта 1º повторяется для некоторого нового $w^1 \in E^n$. Если же оказалось, что множество улучшающих направлений состоит только из нулевого элемента или же $\sigma_0 = 0$, то элемент x^0 принимается за x^* – максимальный, либо, при $\sigma_0 = +\infty$, констатируется факт отсутствия максимума у $F(x)$.

Таким образом, поиск экстремального элемента сводится к итерационной процедуре вида

$$x^{k+1} = x^k + \sigma_k \cdot w^k, \quad k = 0,1,2,\ldots,$$

условия сходимости которой к искомому экстремальному элементу x^* (то есть, $\rho(x^k, x^*) \to 0$ при $k \to \infty$) зависят, как от способа выбора w^k и σ_k, так и от свойств максимизируемого функционала.

§ 2.2.2. Методы поиска локального гладкого экстремума

Предположим, что функционал $F(x)$ непрерывно дифференцируем в E^n и выполнены следующие условия:

1º. Начальный элемент x^0 принадлежит окрестности максимального элемента x^*, в которой функционал $F(x)$ *строго выпуклый вверх*;

2º. Элемент *направления максимизации* w^k удовлетворяет ограничению

$$(\operatorname{grad} F(x^k), w^k) \geq \alpha \cdot \left| \operatorname{grad} F(x^k) \right|^2,$$

где α – некоторое фиксированное малое положительное число;

3º. Величина шага по выбранному направлению максимизации σ_k находится путем решения одномерной задачи

$$\sigma_k = \max_{\sigma \geq 0} F(x^k + \sigma \cdot w^k).$$

В этом случае последовательность элементов $\{x^k\}$ будет сходиться к x^*.

Заметим, что сформулированное в п. 3º правило выбора шага эффективно для *непрерывно дифференцируемых* функционалов в E^n. Оптимальная величина шага σ_k по направлению w^k в этом случае может быть найдена из уравнения

$$(\operatorname{grad} F(x^k + \sigma_k w^k), w^k) = 0.$$

Это правило гарантирует наибольшее локальное увеличение функционала по направлению w^k, что иллюстрирует рисунок 2.2.2.1.

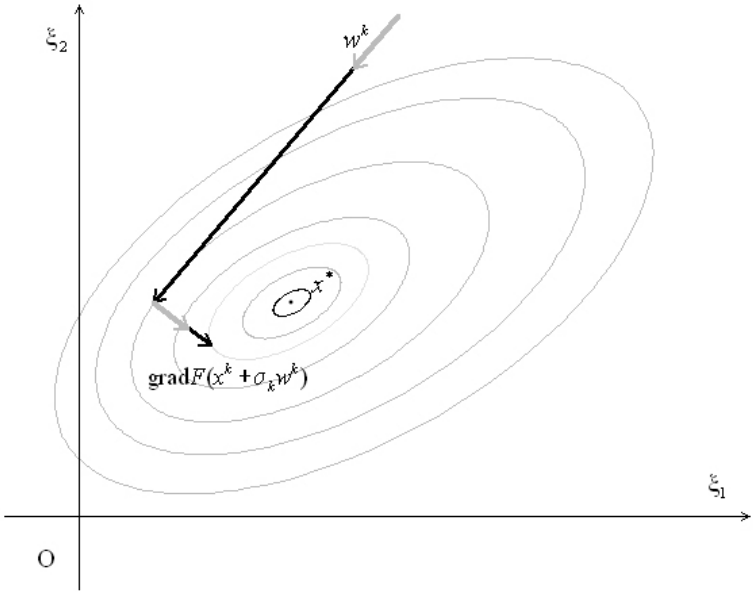


Рисунок 2.2.2.1

В случаях, когда оптимизируемый функционал более гладкий (например, имеет непрерывные частные производные первого и второго порядка), для выбора w_k можно использовать другие, более эффективные схемы, такие как:

1) *метод наискорейшего подъема*, основанный на оценке скорости возрастания функционала $F(x)$ на элементе x^k по направлению w^k, которая в силу неравенства Коши–Буняковского максимальна при $w^k = \operatorname{grad} F(x^k)$, поскольку $\forall w$ с $|w| = 1$

$$\left|\frac{\partial F}{\partial w}\right| = |(\operatorname{grad} F, w)| \le |\operatorname{grad} F \| w| = |\operatorname{grad} F|.$$

Таким образом, направление "наискорейшего" *возрастания* функционала $F(x)$ на некотором фиксированном элементе x сонаправлено с элементом $\operatorname{grad} F(x)$, а направление "наискорейшего" *убывания* с элементом $-\operatorname{grad} F(x)$.

2) *метод квадратичной аппроксимации* (*метод Ньютона*), при котором улучшающее направление w_k удовлетворяет условию

$$\hat{\operatorname{Hess}} F(x_k) w^k = -\operatorname{grad} F(x_k),$$

что в матричном и координатном представлениях есть система линейных уравнений

$$\left\| \hat{\operatorname{Hess}} F(x_k) \right\| \left\| w^k \right\| = -\left\| \operatorname{grad} F(x_k) \right\| \text{ или}$$

$$\sum_{j=1}^{n} \frac{\partial^2 F}{\partial \xi_i \partial \xi_j} \omega_j^k = -\frac{\partial F}{\partial \xi_i}, \; i = [1, n],$$

где $\left\| w^k \right\| = \begin{Vmatrix} \omega_1^k \\ \omega_2^k \\ \dots \\ \omega_n^k \end{Vmatrix}$ – координатное представление элемента w^k.

Убедимся, что для квадратичного функционала $F(x)$ стационарный (то есть такой, что $\operatorname{grad} F(x) = o$) элемент находится по методу Ньютона за одну итерацию с $\sigma = 1$.

Действительно, в этом случае стационарный элемент существует и единственный. Разложение $F(x)$ по формуле Тейлора в окрестности любого x_0 при произвольном элементе вариации $dx = w$ (то есть отклонении от x_0) записывается *без остаточного члена*

$$F(x_0 + w) = F(x_0) + \sum_{j=1}^{n} \frac{\partial F}{\partial \xi_j} \omega_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} \omega_j \omega_i .$$

Тогда условие стационарности $F(x)$ будет иметь вид

$$\frac{\partial F}{\partial \omega_i} = \frac{\partial F}{\partial \xi_i} + \sum_{j=1}^{n} \frac{\partial^2 F}{\partial \xi_j \partial \xi_i} \omega_j = 0 \quad ; \quad i = [1, n],$$

что и доказывает проверяемое утверждение. Тип стационарного элемента (максимум, минимум или "седло") в этом случае зависит от знаковой определенности $\left\| \hat{\text{Hess}}\, F(x) \right\|$.

В заключение приведем пример использования схемы *градиентного подъема* с непрерывным изменением величины "шага по направлению" (*метод Коши*). Формально поиск локального максимального элемента для функционала $F(x)$ в этом случае может быть сведен к решению следующей задачи Коши: $\frac{dx}{d\tau} = \text{grad}\, F(x);\ x(0) = x^0$.

Например, в E^2 для $F(\xi_1, \xi_2) = -2\xi_1^2 - \xi_2^2$ с $\left\| x^0 \right\| = \begin{Vmatrix} 1 \\ 1 \end{Vmatrix}$ имеем $\begin{cases} \dot{\xi}_1 = -4\xi_1, \\ \dot{\xi}_2 = -2\xi_2, \end{cases}$ что дает $\begin{cases} \xi_1(\tau) = e^{-4\tau}, \\ \xi_2(\tau) = e^{-2\tau} \end{cases}$ с фазовой траекторией, задаваемой в силу условий $\xi_1(0) = 1$; $\xi_2(0) = 1$ уравнением $\xi_1 - \xi_2^2 = 0$, которая при $\tau = 0$ выходит из начального элемента x_0 и при $\tau \to +\infty$ асимптотически стремится к максимальному элементу $\left\| x^* \right\| = \begin{Vmatrix} 0 \\ 0 \end{Vmatrix}$.

§ 2.2.3. Поиск экстремума выпуклого недифференцируемого функционала

Использование итерационной процедуры вида

$$x^{k+1} = x^k + \sigma_k \cdot w^k, \quad k = 0,1,2,\ldots,$$

оказывается возможным и в случаях, когда выпуклый функционал $F(x)$ не имеет непрерывных частных производных, но для каждого элемента из его области определения существует субдифференциал ∂F (см. определение 1.3.5.1.)

В качестве элемента улучшающего направления обычно берется $w^k \in \partial F(x_k)$ такой, что $w_k = \arg\min_{l \in \partial F(x_k)} \| l \|$, поскольку необходимое условие экстремальности элемента x^* имеет вид $o \in \partial F(x^*)$. Можно показать, что сходимость итерационной процедуры в этом случае обеспечивается выполнением следующих двух условий на выбор величины шага σ_k:

$$1)\ \lim_{k\to\infty} \sigma_k = 0; \qquad 2)\ \lim_{N\to\infty} \sum_{k=0}^{k=N} \sigma_k = +\infty.$$

Отметим, что в вычислительной практике достаточно часто используется схема с некоторым фиксированным $w^k \in \partial F(x_k)$, ограниченным лишь условием $w^k \neq o$.

Раздел 2.3. МЕТОДЫ ПОИСКА ОДНОМЕРНОГО ЭКСТРЕМУМА В E^n

§ 2.3.1. Метод дихотомии

Как было отмечено, одной из возможных (и достаточно часто применяемых на практике) процедур выбора величины σ_k – шага по улучшающему направлению – является правило

$$\sigma_k = \arg\max_{\sigma \geq 0} F(x^k + \sigma \cdot w^k).$$

Рассмотрим эту задачу подробнее.

Пусть в E^n задан функционал $F(x)$ и Ω – совокупность элементов $x \in E^n$ таких, что $x = x_0 + \tau w;\ \forall \tau \in (-\infty,+\infty)$, а w – некоторый *ненулевой* элемент в E^n. Обозначим

$$f(\tau) = F(x_0 + \tau w);\ \forall \tau \in (-\infty,+\infty).$$

Задачу отыскания экстремального для $f(\tau)$ числа $\tau^* \in E^1$ принято называть задачей *одномерной оптимизации* $F(x)$ (или *поиска экстремума* $F(x)$) *по направлению* w.

Действительно, в координатной форме эта задача сводится к нахождению экстремальных значений функции

$$f(\tau) = F(\xi_1^0 + \tau\omega_1, \xi_2^0 + \tau\omega_2, \ldots, \xi_n^0 + \tau\omega_n),$$

зависящей от одного аргумента τ, где $\|w\| = \begin{Vmatrix} \omega_1 \\ \omega_2 \\ \cdots \\ \omega_n \end{Vmatrix}$ – координатное представление элемента w в E^n.

Для решения задач одномерной оптимизации возможно использование необходимых и достаточных условий существования экстремума функции одной переменной, основанных на теоремах 2.1.2.1 и 2.1.3.1 для частного случая $n = 1$. Однако в вычислительной практике также достаточно широко применяются схемы одномерного поиска, не требующие нахождения производных, а использующие только значение $f(\tau)$.

Определение 2.3.1.1 Отрезок $[\alpha,\beta]$ называется *отрезком локализации экстремума* функции $f(\tau)$, если τ^* – аргумент экстремального значения $f(\tau)$ – принадлежит $[\alpha,\beta]$, при этом само значение τ^* может быть не известно.

Определение 2.3.1.2 Функция $f(\tau)$ называется *унимодальной*, если она имеет единственный экстремум на отрезке $[\alpha,\beta]$.

Будет справедлива

Теорема 2.3.1.1 **Пусть функция $f(\tau)$ определена и унимодальна (на минимум) для $\tau\in[\alpha,\beta]$. Тогда для фиксированных $\lambda,\mu\in[\alpha,\beta]$ и λ таких, что $\lambda<\mu$:**

из условия $f(\lambda)>f(\mu)$ следует, что

$$f(\tau)\geq f(\lambda),\ \forall\tau\in[\alpha,\lambda],$$

а из условия $f(\lambda)<f(\mu)$ следует, что

$$f(\mu)\leq f(\tau),\ \forall\tau\in[\mu,\beta].$$

Доказательство.

Рассмотрим случай, когда $f(\lambda)>f(\mu)$ для $\tau\in[\alpha,\lambda)$, и предположим, что существует $\tau\in[\alpha,\lambda)$ такое, что

$$f(\tau)\leq f(\lambda).$$

Однако это противоречит унимодальности функции $f(\tau)$ на $\tau\in[\alpha,\beta]$.

Второй случай доказывается аналогично.

Теорема доказана.

Рассмотрим наиболее часто применяемые на практике вычислительные схемы, основанные на идее последовательного сужения отрезка локализации за счет вычисления значения $f(\tau)$ в некоторых дополнительных точках отрезка $[\alpha,\beta]$ и использующие различные критерии эффективности поиска экстремума на $[\alpha,\beta]\subset E^1$.

В дальнейшем будем считать, что значения минимизируемой функции $f(\tau)$ найдены в двух новых точках λ и μ, принадлежащих $[\alpha,\beta]$, причем таких, что $\alpha<\lambda<\mu<\beta$.

Одной из возможных схем "одномерной оптимизации" является "метод дихотомии", основанный на следующих рассуждениях.

Ясно, что искомое значение τ^* принадлежит либо $[\alpha,\mu]$, либо $[\lambda,\beta]$ и длина нового отрезка локализации будет зависеть от выбора λ и μ. Организуем этот выбор таким образом, чтобы длина максимального из двух отрезков $[\alpha,\mu]$ и $[\lambda,\beta]$ оказалась как можно меньшей. Иначе говоря, необходимо исследовать в E^2 на экстремум функционал вида $L(\lambda,\mu)=\min\limits_{\lambda}\max\limits_{\mu}\{\mu-\alpha,\beta-\lambda\}$, вид изолиний которого показан на рис. 2.3.1.1.

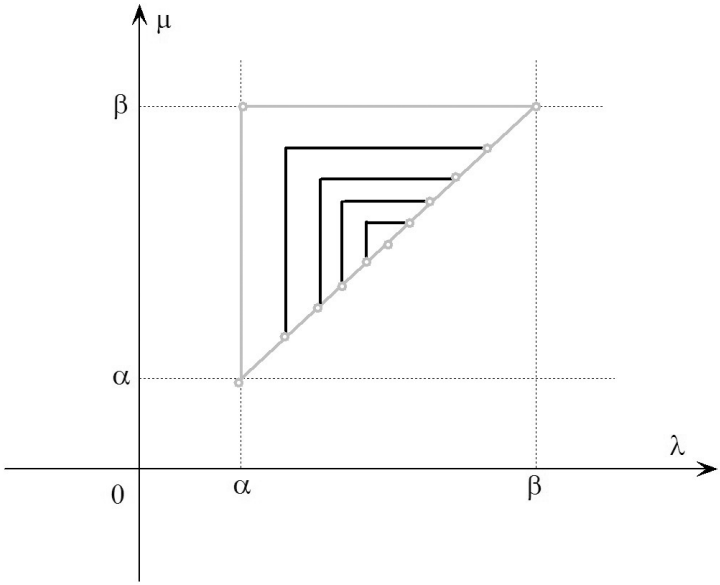


Рис. 2.3.1.1

Очевидно, что минимальное значение $L(\lambda,\mu)$ достигается при $\lambda=\mu=\dfrac{\alpha+\beta}{2}$, однако это запрещено условием $\lambda<\mu$. Действительно, при $\lambda=\mu$ дополнительное значение $f(\tau)$ находится только для *одной* точки из $[\alpha,\beta]$, что не позволяет сократить отрезок неопределенности.

На практике рекомендуется обходить эту проблему, выбирая

$$\lambda = \frac{\alpha+\beta}{2} - \frac{\varepsilon}{2} \quad \text{и} \quad \mu = \frac{\alpha+\beta}{2} + \frac{\varepsilon}{2},$$

где $\varepsilon > 0$ число достаточно малое, но гарантирующее различие значений $f(\lambda)$ и $f(\mu)$ при заданной допустимой погрешности вычислений.

Процедуру уменьшения длины отрезка локализации можно проводить итеративно, приняв $\alpha_0 = \alpha$, $\beta_0 = \beta$ и используя рекуррентные соотношения

$$\lambda_k = \frac{\alpha_k+\beta_k}{2} - \frac{\varepsilon}{2} \quad \text{и} \quad \mu_k = \frac{\alpha_k+\beta_k}{2} + \frac{\varepsilon}{2}, \quad k = 0,1,2,\ldots.$$

При этом из теоремы 2.3.1 следует, что если на *k*-м шаге

$$f(\lambda_k) > f(\mu_k),$$

то нужно выбирать $\alpha_{k+1} = \lambda_k$ и $\beta_{k+1} = \beta_k$, иначе полагать $\alpha_{k+1} = \alpha_k$ и $\beta_{k+1} = \mu_k$.

Отметим, что длина отрезка локализации после k-ой итерации будет равна

$$\beta_{k+1} - \alpha_{k+1} = \frac{\beta-\alpha}{2^k} + \varepsilon\left(1 - \frac{1}{2^k}\right),$$

а число N – количество вычислений функций, необходимых в методе дихотомии для получения длины отрезка локализации меньшей чем Δ, определяется из условий (с учетом $\varepsilon << \Delta$)

$$\left(\frac{1}{2}\right)^{\frac{N}{2}} < \frac{\Delta}{\beta-\alpha} \quad \text{или же} \quad N > 2\log_2 \frac{\beta-\alpha}{\Delta}.$$

§ 2.3.2. Метод "золотого сечения"

Метод дихотомии требует вычисления на каждом шаге двух новых значений $f(\tau)$ на отрезке локализации. В тех случаях, когда для этого не требуются затраты значительных вычислительных ресурсов, алгоритмическая простота данного метода является основным аргументом в пользу дихотомии.

Однако на практике достаточно часто возникает ситуация, когда нахождение значения $f(\tau)$ само по себе является сложной и/или ресурсоемкой задачей. В этих случаях рекомендуется применение методов одномерной оптимизации, требующих на каждом шаге, начиная со второго, вычисления *одного* значения $f(\tau)$ на отрезке локализации. Одной из таких схем является метод "золотого сечения".

Геометрическая задача нахождения "золотого сечения" – разбиения данного отрезка на две неравные части так, чтобы

отношение длин меньшей и большей частей равнялось отношению длины большей части к длине исходного отрезка

– рассматривалась еще Евклидом, а ее алгебраическое решение сводится к решению квадратного уравнения вида $\rho^2 - 3\omega\rho + \omega^2 = 0,$ где ρ – длина меньшей части исходного отрезка длины ω.

Идея использования метода "золотого сечения" в одномерном поиске экстремума заключается в следующем.

Поскольку исходный отрезок локализации $[\alpha_k, \beta_k]$ разбивается точками λ_k и μ_k на три части, а новый промежуток неопределенности $[\alpha_{k+1}, \beta_{k+1}]$ согласно теореме 2.3.1.1 выбирается по правилу:

для $f(\lambda_k) > f(\mu_k)$ $[\alpha_{k+1}, \beta_{k+1}] = [\lambda_k, \beta_k]$,

иначе $[\alpha_{k+1}, \beta_{k+1}] = [\alpha_k, \mu_k]$,

то для уменьшения числа новых значений $f(\tau)$, необходимых для сокращения отрезка локализации, следует потребовать, чтобы для каждой новой итерации либо $\lambda_{k+1} = \mu_k$, либо $\mu_{k+1} = \lambda_k$.

Из рисунка 2.3.2.1 очевидно, что добиться этого можно, если длина $[\alpha_{k+1}, \beta_{k+1}]$ не зависит от того, будет ли

$$f(\lambda_k) \geq f(\mu_k) \text{ или же } f(\lambda_k) \leq f(\mu_k);$$

то есть, будет справедливо равенство $\beta_k - \lambda_k = \mu_k - \alpha_k$.

Введем в рассмотрение параметр $0 < \rho < 1$, такой что:

1º. $\lambda_k = \alpha_k + \rho(\beta_k - \alpha_k)$.

2º. $\mu_k = \alpha_k + (1-\rho)(\beta_k - \alpha_k)$ и

3º. $(\beta_{k+1} - \alpha_{k+1}) = (1-\rho)(\beta_k - \alpha_k)$.

Если оказывается, что $f(\lambda_k) < f(\mu_k)$, то полагаем:

4º. $\alpha_{k+1} = \alpha_k$ и 5º. $\beta_{k+1} = \mu_k$.

И учтем, наконец, что:

6º. $\mu_{k+1} = \lambda_k$.

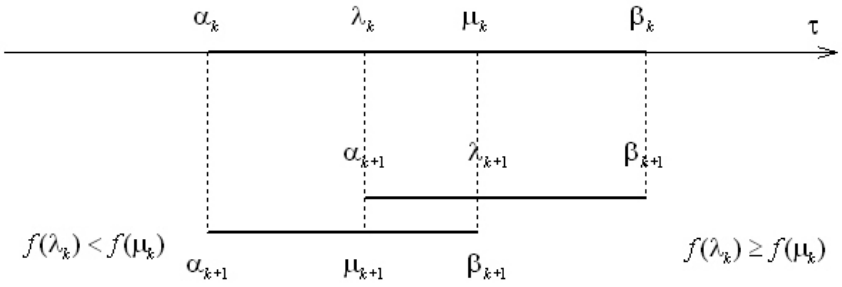


Рис. 2.3.2.1

Из равенства 2º, верного для любого k, получаем

$$\mu_{k+1} = \alpha_{k+1} + (1-\rho)(\beta_{k+1} - \alpha_{k+1}),$$

что в силу 4º, 5º и 6º означает

$$\lambda_k = \alpha_k + (1-\rho)(\mu_k - \alpha_k).$$

Подставляя в это равенство выражения 1º и 2º, получаем

$$\alpha_k + \rho(\beta_k - \alpha_k) = \alpha_k + (1-\rho)((\alpha_k + (1-\rho)(\beta_k - \alpha_k)) - \alpha_k),$$

а после упрощений

$$\rho(\beta_k - \alpha_k) = (1-\rho)^2(\beta_k - \alpha_k).$$

Наконец, поскольку это соотношение должно быть верным для любого k, то коэффициент ρ является корнем квадратного уравнения

$$\rho^2 - 3\rho + 1 = 0 \ ,$$

являющегося уравнением задачи нахождения "золотого сечения", и, кроме того, $\rho < 1$, то

$$\rho = \frac{3-\sqrt{5}}{2} \quad \text{и} \quad 1-\rho = \frac{\sqrt{5}-1}{2}.$$

Случай $f(\lambda_k) \geq f(\mu_k)$ приводит к точно такому же результату (проверьте это самостоятельно).

Для метода "золотого сечения" число N – вычислений значения функции, необходимых для получения длины отрезка локализации меньшей, чем Δ, в силу соотношения 3º, определяется неравенством

$$\frac{\beta-\alpha}{\tau^{N-1}} < \Delta \quad \text{или же} \quad N > \log_\tau \frac{\tau(\beta-\alpha)}{\Delta}, \text{ где } \tau = \frac{1+\sqrt{5}}{2}.$$

§ 2.3.3. Метод Фибоначчи

Метод "золотого сечения" является более эффективным по сравнению с методом дихотомии потому, что отношение $\frac{\beta_{k+1}-\alpha_{k+1}}{\beta_k-\alpha_k}$ – коэффициент сжатия отрезка локализации – для "золотого сечения" лучше, чем для дихотомии (поскольку $\frac{\sqrt{5}-1}{2} \approx 0.618 < \frac{1}{\sqrt{2}}$).

Для данных методов не требуется, вообще говоря, определять число шагов, обеспечивающих достижение требуемой точности. При их использовании достаточно на каждом шаге проверять выполнение условия $\beta_k - \alpha_k \leq \Delta$, где Δ – требуемая точность.

Существуют, однако, алгоритмы одномерной оптимизации, для которых необходимо заранее найти значение N – числа шагов, обеспечивающих заданную точность. К таким методам относится алгоритм, предложенный и обоснованный в 1953 году Дж. Кифером, и получивший название метода Фибоначчи.

Теоретическая оценка коэффициента сжатия в методе Фибоначчи несколько лучше, чем для метода "золотого сечения", в то время как реализация метода Фибоначчи сложнее и потому в вычислительной практике этот метод используется реже. Однако отметим, что при $N \to +\infty$ эффективность обеих схем асимптотически сближается.

Алгоритм метода Фибоначчи основан на использовании "чисел Фибоначчи", являющихся членами числовой последовательности $\{F_k\}$, где

$$F_0 = F_1 = 1;$$
$$F_k = F_{k-1} + F_{k-2};\ k = 2,3,\ldots$$

или же в явном виде

$$F_k = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{k+1}\right],$$

описывается следующими рекуррентными соотношениями

$$\lambda_k = \alpha_k + \frac{F_{N-k-1}}{F_{N-k+1}}(\beta_k - \alpha_k),$$
$$\mu_k = \alpha_k + \frac{F_{N-k}}{F_{N-k+1}}(\beta_k - \alpha_k).$$

Покажем, что в методе Фибоначчи либо $\lambda_{k+1} = \mu_k$, либо $\mu_{k+1} = \lambda_k$, то есть на каждой новой итерации требуется вычисление лишь одного нового значения $f(\tau)$.

Рассмотрим для примера случай $f(\lambda_k) \geq f(\mu_k)$, тогда $\alpha_{k+1} = \lambda_k$ и $\beta_{k+1} = \beta_k$. Нам надо показать, что $\lambda_{k+1} = \mu_k$. Действительно, заменив в выражении для λ_k k на $k+1$, получим

$$\lambda_{k+1} = \alpha_{k+1} + \frac{F_{N-k-2}}{F_{N-k}}(\beta_{k+1} - \alpha_{k+1}) = \lambda_k + \frac{F_{N-k-2}}{F_{N-k}}(\beta_k - \lambda_k).$$

Еще раз подставляем выражение для λ_k в правую часть последнего равенства. Получаем

$$\lambda_{k+1} = \left(\alpha_k + \frac{F_{N-k-1}}{F_{N-k+1}}(\beta_k - \alpha_k)\right) +$$
$$+ \frac{F_{N-k-2}}{F_{N-k}}\left[\beta_k - \left(\alpha_k + \frac{F_{N-k-1}}{F_{N-k+1}}(\beta_k - \alpha_k)\right)\right] =$$
$$= \alpha_k + R(\beta_k - \alpha_k),$$

где $R = \dfrac{F_{N-k-1}}{F_{N-k+1}} + \dfrac{F_{N-k-2}}{F_{N-k}} - \dfrac{F_{N-k-2}}{F_{N-k}} \dfrac{F_{N-k-1}}{F_{N-k+1}}$.

Поскольку для членов последовательности Фибоначчи справедливы равенства

$$F_{N-k+1} = F_{N-k} + F_{N-k-1} \quad \text{и} \quad F_{N-k} = F_{N-k-1} + F_{N-k-2},$$

то выражение для R может быть упрощено следующим образом

$$R = \frac{F_{N-k-1}F_{N-k} + F_{N-k-2}F_{N-k+1} - F_{N-k-2}F_{N-k-1}}{F_{N-k}F_{N-k+1}} =$$

$$= \frac{F_{N-k-1}F_{N-k} + F_{N-k-2}F_{N-k} + F_{N-k-2}F_{N-k-1} - F_{N-k-2}F_{N-k-1}}{F_{N-k}F_{N-k+1}} =$$

$$= \frac{F_{N-k-1} + F_{N-k-2}}{F_{N-k+1}} = \frac{F_{N-k}}{F_{N-k+1}}.$$

И окончательно получаем

$$\lambda_{k+1} = \alpha_k + \frac{F_{N-k}}{F_{N-k+1}}(\beta_k - \alpha_k) = \mu_k .$$

Проверьте самостоятельно, что в случае $f(\lambda_k) \le f(\mu_k)$ имеет место $\mu_{k+1} = \lambda_k$.

Полное число шагов метода Фибоначчи N фиксировано, что позволяет получить несколько лучшую (по сравнению с методом "золотого сечения") оценку эффективности. Непосредственная проверка показывает, что

$$\beta_{N-1} - \alpha_{N-1} = \frac{2(\beta - \alpha)}{F_N}.$$

Тогда, приняв за последнее приближение точку, близкую к середине отрезка $[\alpha_{N-1}, \beta_{N-1}]$, приходим к оценке

$$\beta_N - \alpha_N \approx \frac{\beta - \alpha}{F_N}, \quad \text{где } F_N \approx \frac{\tau^{N+1}}{\sqrt{5}}.$$

Таким образом, показано, что алгоритм вычислений в методе Фибоначчи практически аналогичен схеме "золотого сечения", показанной на рис. 2.3.2.1, за двумя исключениями:

1) коэффициент сжатия в нем равен $\frac{F_{N-k}}{F_{N-k+1}}$, то есть меняется от итерации к итерации, стремясь, однако, асимптотически к значению $\frac{\sqrt{5}-1}{2} \approx 0.618$.

2) для нахождения значений λ_k и μ_k необходимо знать заранее N – полное число шагов метода Фибоначчи, которое удовлетворяет оценкам:

$$\frac{\beta-\alpha}{F_N} < \Delta \quad \text{или же} \quad N > \log_\tau \frac{(\beta-\alpha)\sqrt{5}}{\tau\Delta}.$$

Глава 3
ЗАДАЧИ ПОИСКА ЭКСТРЕМУМА ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ

Раздел 3.1. ОБЩАЯ ПОСТАНОВКА ЗАДАЧ НА УСЛОВНЫЙ ЭКСТРЕМУМ

Рассмотрим следующую задачу:

на множестве $R \subset E^n$ *найти элемент* x^* *такой, что*

$$F(x^*) \geq F(x) \ \forall x \in R, \tag{3.1.1}$$

где $F(x)$ *– некоторый функционал в* E^n.

В другом виде эта задача иногда формулируется как:

найти $\max\limits_{x \in R} F(x)$ или *найти* $\arg\max\limits_{x \in R} F(x)$.

По исторически сложившимся причинам эту задачу принято называть *задачей на условный экстремум*, или же *задачей математического программирования*.

Основное отличие задачи (3.1.1) от классической оптимизационной задачи (2.1.1) состоит в том, что для нее поиск экстремального элемента осуществляется не по всей области определения *целевого функционала* $F(x)$, а по, вообще говоря, более узкому множеству R, которое обычно называют *допустимым множеством* задачи (3.1.1).

В зависимости от свойств и способа описания допустимого множества и целевого функционала, задачи вида (3.1.1) терминологически подразделяются на специальные классы, например:

- линейного программирования;
- квадратичного программирования;
- выпуклого программирования;
- параметрического программирования;
- динамического программирования;
- дискретного (целочисленного) программирования и т.п.

Раздел 3.2. УСЛОВИЯ ОПТИМАЛЬНОСТИ В ЗАДАЧАХ НА УСЛОВНЫЙ ЭКСТРЕМУМ

Исследование и разработка алгоритмов решения задач математического программирования составляет предмет сравнительно нового (хотя и насчитывающего уже более чем полувековую историю) раздела математики, носящего название "Методы оптимизации".

Вполне очевидно, что при отсутствии каких-либо предположений о свойствах допустимого множества R и целевого функционала $F(x)$, методу решения задачи (3.1.1), использующему лишь определение экстремального элемента (проще говоря, полному перебору), нет альтернативы. Однако существует большое число практически эффективных методов решения задач математического программирования ряда конкретных классов, алгоритмической основой которых служат как необходимые, так и достаточные условия оптимальности.

Для анализа необходимых и достаточных условий существования экстремального элемента $x^* \in R$ введем для допустимого элемента $x^0 \in R$ *множество допустимых вариаций*.

Определение 3.2.1. Если для некоторого элемента $\delta x \in E^n$ найдется число $\varepsilon > 0$ такое, что

$$x^0 + \tau \delta x \in R; \ \forall \tau : 0 < \tau < \varepsilon,$$

то такой элемент называется *допустимой вариацией на* x^0 *в* R.

Совокупность всех таких допустимых вариаций образует в E^n конус с вершиной в x^0, называемый *конусом допустимых в* R *вариаций на элементе* $x^0 \in R$.

Конус допустимых вариаций будем обозначать $K(x^0)$. Его использование позволяет формулировать условия оптимальности для тех случаев, когда $K(x^0)$ – выпуклое и имеющее внутренние элементы множество.

В то же время определение 3.2.1 для ряда практически важных случаев оказывается излишне жестким: если R не имеет внутренних точек (например, является нелинейной гиперповерхностью в E^n), то конус $K(x^0)$ будет состоять из единственного элемента $\delta x = o$. В этой ситуации можно обобщить понятия допустимой вариации на элементе $x^0 \in R$ следующим образом.

Определение 3.2.2. Пусть для некоторого элемента $\delta x \in E^n$ найдутся

- число $\varepsilon > 0$ и
- непрерывно зависящий от ε элемент $o(\varepsilon) \in E^n$, удовлетворяющий соотношению

$$\lim_{\varepsilon \to +0} \frac{|o(\varepsilon)|}{\varepsilon} = 0$$

такие, что

$$x^0 + \tau \delta x + o(\varepsilon) \in R; \ \forall \tau : 0 < \tau < \varepsilon.$$

Тогда совокупность всех таких элементов δx назовем *конусом касательных* (*или обобщенных*) *допустимых вариаций на элементе* $x^0 \in R$, обозначаемым $M(x^0)$.

Можно показать, что конус $M(x^0)$ совпадает с замыканием конуса $K(x^0)$ в случаях, когда множество R выпукло или имеет непустую внутренность $(\operatorname{int} R)$.

Введем понятие "улучшающих функционал $F(x)$ вариаций" на элементе $x^0 \in R$.

Определение 3.2.3. Пусть для некоторого элемента $\delta x \in E^n$ найдется число $\varepsilon > 0$ такое, что

$$F(x^0 + \tau \delta x) > F(x^0) \quad \forall \tau : 0 < \tau < \varepsilon .$$

Тогда $P(x^0)$ – совокупность всех таких элементов δx – назовем *множеством улучшающих функционал вариаций на элементе* $x^0 \in R$.

Сформулируем теперь, основываясь на понятиях допустимых и улучшающих вариаций, условия оптимальности в задаче математического программирования вида

$$\textit{найти} \max_{x \in R} F(x),$$

если функционал $F(x)$ допускает аппроксимацию по формуле Тейлора первого порядка в окрестности произвольного элемента $x^0 \in R$.

Теорема 3.2.1. **Пусть функционал $F(x)$ на множестве R принимает максимальное значение на элементе x^*. Тогда**

$$(\operatorname{grad} F(x^*), \delta x) \le 0 \quad \forall \delta x \in \overline{K}(x^*),$$

где $\overline{K}(x^*)$ – непустое замыкание конуса допустимых вариаций на элементе x^*.

Доказательство.

Если множество $\overline{K}(x^*)$ состоит только из нулевого элемента, то утверждение теоремы очевидно.

Допустим теперь, что в $\overline{K}(x^*)$ имеется *ненулевой* элемент δx такой, что

$$(\operatorname{grad} F(x^*), \delta x) > 0 .$$

Тогда в силу свойств конуса $\overline{K}(x^*)$ и вытекающего из (3.2.1) соотношения

$$F(x^* + \tau\delta x) - F(x^*) = \tau(\operatorname{grad} F(x^*), \delta x) + o(\tau),$$

найдется $\varepsilon > 0$ такое, что элемент

$$x^* + \tau\delta x \in R, \ \forall \tau : 0 < \tau < \varepsilon \quad \text{и} \quad F(x^* + \tau\delta x) > F(x^*) .$$

Что в свою очередь означает: множество улучшающих вариаций на элементе x^* содержит ненулевой элемент, и, следовательно, $F(x^*)$ не является максимальным значением.

Полученное противоречие доказывает справедливость утверждения теоремы.

Теорема доказана.

Утверждение теоремы 3.2.1 допускает также следующую интерпретацию:

> *пересечение множества улучшающих вариаций и конуса допустимых вариаций на оптимальном элементе* $x^* \in E^n$ *содержит только нулевой элемент*.

Полученные условия оптимальности позволяют решать исходную задачу поиска элементов вида $x^* = \arg\max_{x \in R} F(x)$, итеративно осуществляя переход от одного элемента в E^n к другому по схеме, близкой к описанной в § 2.2.

$$x^{k+1} = x^k + \sigma_k \cdot w^k, \quad k = 0,1,2,\ldots , \qquad (3.2.1)$$

При этом в качестве элемента w_k используются элементы, принадлежащие пересечению множества улучшающих и конуса допустимых вариаций. Процесс решения продолжается, пока это пересечение содержит ненулевые элементы.

Раздел 3.3. ПРИНЦИП МАКСИМУМА

Предположим дополнительно, что R выпукло в $U_\varepsilon(x^*)$ – некоторой ε-окрестности элемента x^*, и $\operatorname{grad} F(x^*) \neq o$, (в этом случае гиперплоскость $(l^*, x - x^*) = 0$ – очевидно опорная для множества $U_\varepsilon(x^*) \cap R$ на элементе x^*), то линейный функционал (l^*, x) будет достигать на этом множестве максимальное значение при $x = x^*$, равное (l^*, x^*).

Действительно, утверждение теоремы 3.2.1 о том, что на элементе x^*, максимизирующем значение функционала $F(x)$ на множестве R,

$$(\operatorname{grad} F(x^*), \delta x) \le 0\,; \quad \forall \delta x \in \overline{K}(x^*),$$

приняв во внимание, что $\delta x = x - x^*$, можно записать в виде

$$(l^*, x - x^*) \le 0\,; \quad \forall x \in R \subset E^n, \text{ где } l^* = \operatorname{grad} F(x^*).$$

Это неравенство в свою очередь будет равносильно условию

$$\max_{x \in R} (l^*, x - x^*) = 0\,,$$

причем этот максимум будет достигаться на элементе x^*.

Ужесточим условия, налагаемые на R, потребовав *выпуклости* всего этого множества. Тогда будет справедлива

Теорема 3.3.1. (Принцип максимума)[3]

Пусть функционал $F(x)$ на выпуклом множестве R принимает максимальное значение на элементе x^*. Тогда на этом элементе достигает максимума на R и линейный функционал $(\operatorname{grad} F(x^*), x)$.

[3] В вычислительной практике используются также и иные формы принципа максимума, формулируемые для различных классов оптимизационных задач. Например, "принцип максимума Л.С. Понтрягина" для задач оптимального управления в банаховом пространстве (см. § 5.1.2).

Доказательство.

Предположим противное: в R существует элемент $\bar{x}$, такой, что $(l^*, \bar{x}) > (l^*, x^*)$. Тогда $(l^*, \delta x) > 0$, где $\delta x = \bar{x} - x^*$.

В силу предположения о выпуклости R и условия $\bar{x}, x^* \in R$ имеем $\forall \tau : 0 \le \tau \le 1$:

$$x = \tau\bar{x} + (1-\tau)x^* = x^* + \tau\delta x \in R.$$

Значит δx – допустимая вариация и, кроме того, улучшающая в силу соотношения

$$F(x) - F(x^*) = \tau(l^*, \delta x) + o(\tau)$$

значение функционала $F(x)$ на элементе x^*, что противоречит условию теоремы: $x^* = \arg\max_{x \in R} F(x)$.

Теорема доказана.

Отметим, что в случае, когда $\operatorname{grad} F(x^*) \neq o$, гиперплоскость $(l^*, x - x^*) = 0$ на элементе x^* в E^n является опорной не только к R, но и ко множеству $\{x \mid F(x) \le F(x^*)\}$, при условии выпуклости вверх $F(x)$; $\forall x \in U_\varepsilon(x^*)$.

Теорема 3.3.1 утверждает, что достижение линейным функционалом (l^*, x) своего максимума на $x^* \in R$ является *необходимым* условием для $x^* = \arg\max_{x \in R} F(x)$ в случае выпуклости допустимого множества R. *Достаточное* условие можно получить, наложив более жесткие условия не только на R, но и на целевой функционал $F(x)$.

Потребуем, чтобы на всем допустимом множестве R целевой функционал был *выпуклым вверх*, то есть $\forall x_1, x_2 \in R$ и $\forall \tau \in [0,1]$ справедливо неравенство

$$F((1-\tau)x_1 + \tau x_2) \ge (1-\tau)F(x_1) + \tau F(x_2).$$

Тогда будет справедлива

Теорема 3.3.2. **Пусть на элементе x^* выпуклого множества R линейный функционал (l^*, x) достигает своего максимума. Тогда выпуклый вверх функционал $F(x)$ принимает максимальное значение на том же элементе x^*.**

Доказательство.

Предположим противное: в R существует элемент $\bar{x}$ такой, что $F(\bar{x}) > F(x^*)$. В силу предположения о выпуклости R и условия $\bar{x}, x^* \in R$ элемент

$$x = \tau\bar{x} + (1-\tau)x^* = x^* + \tau\delta x \in R,\ \forall\tau : 0 \le \tau \le 1,$$

где $\delta x = \bar{x} - x^*$. Значит, δx – *допустимая* вариация и, кроме того, *улучшающая* значение функционала $F(x)$ на элементе x^*.

С другой стороны, в силу выпуклости функционала $F(x)$ при сделанных ограничениях на параметр τ справедлива оценка

$$F(x) \ge \tau F(\bar{x}) + (1-\tau)F(x^*) = \\ = F(x^*) + \tau(F(\bar{x}) - F(x^*)).$$

Тогда $F(x) > F(x^*)$ и, в силу соотношения

$$F(x) - F(x^*) = \tau(l^*, x - x^*) + o(\tau)$$

имеем $(l^*, x - x^*) > 0$ или $(l^*, x) > (l^*, x^*)$, что противоречит условию теоремы: $x^* = \arg\max_{x \in R}(l^*, x)$.

Теорема доказана.

Таким образом теорема 3.3.2 формулирует *достаточные* условия оптимальности элемента $x^* \in E^n$ для выпуклых R и $F(x)$, при условии $l^* = \operatorname{grad} F(x^*) \ne o$.

Подчеркнем еще раз, что в условиях теоремы 3.3.2 гиперплоскость $(l^*, x - x^*) = 0$ на элементе x^*, являющаяся *опорной* к R и ко множеству $\{x \mid F(x) \geq F(x^*)\}$, будет также и *разделяющей* гиперплоскостью для этих множеств.

Раздел 3.4. ДВОЙСТВЕННЫЕ (СОПРЯЖЕННЫЕ) ЗАДАЧИ

Поскольку устанавливаемые теоремой 3.3.2 условия оптимальности являются достаточными (для случая выпуклых R и $F(x)$), то оказывается возможным использование иного (по сравнению со схемой, описанной в §3.2) метода решения исходной задачи – отыскания элемента $x^* = \arg\max_{x \in R} F(x)$.

Будем предполагать, что решение этой задачи существует, единственно и принадлежит границе допустимого множества R. Отметим также, что для любого фиксированного ненулевого элемента l, удовлетворяющего условию теоремы 3.3.2, элемент вида $\tilde{x}(l) = \arg\max_{x \in R}(l, x)$ (хотя быть может не единственный) очевидно существует и принадлежит границе R.

Заметим, что искомый элемент x^* может быть найден из соотношения $l^* = \operatorname{grad} F(x^*)$, если известен элемент l^*, который в свою очередь является решением задачи $l^* = \arg\max_{l} F(\tilde{x}(l))$.

В этом случае оказываются справедливыми соотношения

$$\max_{x \in R} F(x) = \max_{l} F(\tilde{x}(l)) \text{ и } l^* = \arg\max_{l}(l, \tilde{x}(l)).$$

Принципиально допустимую неединственность решения задачи $\tilde{x}(l) = \arg\max_{x \in R}(l, x)$ можно устранить, выбрав, например, из всех таких элементов один, максимизирующий значение функционала $F(x)$.

Отметим, что, если задача поиска $x^* = \arg\max_{x \in R} F(x)$ решается в исходном конечномерном евклидовом пространстве E^n, то задача $l^* = \arg\max_{l} F(\widetilde{x}(l))$ решается в евклидовом пространстве линейных функционалов E^{n+} – сопряженном (двойственном) к E^n. Поэтому задачу $l^* = \arg\max_{l} F(\widetilde{x}(l))$ будем называть *двойственной* (*сопряженной*) к *прямой* задаче $x^* = \arg\max_{x \in R} F(x)$.

Наконец, в силу существования тождественного изоморфизма между пространствами E^n и E^{n++} можно утверждать, что задача двойственная к двойственной совпадает с исходной прямой задачей.

Действительно, как было отмечено в §1.2.1, в E^n с ортонормированным базисом $\{e_1, e_2, \ldots, e_n\}$ линейный функционал $f(x)$ имеет вид $f(x) = \sum_{j=1}^{n} \lambda_j \xi_j$, где

$$\|x\|_e = \|\xi_1 \quad \xi_2 \quad \ldots \quad \xi_n\|^{\mathrm{T}} \text{ и } \|l\|_e = \|\lambda_1 \quad \lambda_2 \quad \ldots \quad \lambda_n\|$$

– координатные представления соответственно элемента x и линейного функционала f в E^n.

Если в пространстве линейных функционалов E^{n+} выбрать базис $\{h_1, h_2, \ldots, h_n\}$, взаимный (биортогональный) к $\{e_1, e_2, \ldots, e_n\}$, то верно равенство $\|l\|_h = \|l\|_e^{\mathrm{T}}$. Это позволяет установить изоморфизм пространств E^n и E^{n+}, сопоставляя их элементы во взаимнооднозначное соответствие по правилу $\|l\|_h \leftrightarrow \|x\|_e^{\mathrm{T}}$.

Рассуждая аналогично для пары пространств E^{n+} и E^{n++}, приходим к их изоморфизму с соответствием вида $\|l\|_h^{\mathrm{T}} \leftrightarrow \|X\|_e$, где $X \in E^{n++}$, что в силу $\|x\|_e \leftrightarrow \|X\|_e$ и означает тождественный изоморфизм E^n и E^{n++}, позволяющий считать их одним и тем же линейным пространством.

В заключение следует отметить, что решение двойственной задачи в ряде случаев удается упростить за счет учета специфических свойств допустимого множества R. Например в случае, когда множество допустимых вариаций на элементе x^* будет выпуклым, замкнутым многогранным конусом вида $\forall \delta x: (a_i^*, \delta x) \geq 0$, $\forall i = [1, m]$, оказывается возможным представление l^* в виде специальной линейной комбинации с неотрицательными коэффициентами элементов $\{-a_i^*, \forall i = [1, m]\}$, порождающих этот конус.

Существование такой комбинации гарантирует альтернативная формулировка теоремы 1.1.4.4 (Фаркаша):

Теорема 1.1.4.4 (Фаркаша)

Для того чтобы существовали $\lambda_i \geq 0$, $i = [1, m]$ **такие, что**

$$l^* = -\sum_{i=1}^{m} \lambda_i a_i^*,$$

необходимо и достаточно, чтобы для каждой допустимой на x^* **вариации, то есть,**

$$\forall \delta x: (a_i^*, \delta x) \geq 0; \forall i = [1, m],$$

выполнялось неравенство $(l^*, \delta x) \leq 0$**, где**

$$\delta x = x - x^*.$$

Раздел 3.5. ЗАДАЧА МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

По исторически сложившимся причинам *задачей математического программирования* принято называть задачу поиска в E^n экстремальных элементов функционала $F(x)$, на множестве элементов x удовлетворяющих условиям вида:

$$f_i(x) \geq 0, i = [1, m],$$

или в развернутой форме

$$\begin{cases} f_1(x) \ge 0, \\ f_2(x) \ge 0, \\ \quad \dots \\ f_m(x) \ge 0. \end{cases}$$

Соответственно в координатной форме эта задача записывается:

Найти максимум $F(\xi_1, \xi_2, \dots, \xi_n)$ по $\{\xi_1, \xi_2, \dots, \xi_n\}$,

при условиях: (3.5.1)

$$\begin{cases} f_1(\xi_1, \xi_2, \dots, \xi_n) \ge 0, \\ f_2(\xi_1, \xi_2, \dots, \xi_n) \ge 0, \\ \quad \dots \\ f_m(\xi_1, \xi_2, \dots, \xi_n) \ge 0. \end{cases}$$

Отметим, что ограничения типа «равенство» могут присутствовать в условии этой задачи в форме двух «встречных» неравенств, например, условие $f_i(x) = 0$ представимо в виде системы $\begin{cases} f_i(x) \ge 0 \\ -f_i(x) \ge 0 \end{cases}$.

§ 3.5.1. Необходимые условия разрешимости задачи математического программирования

Дадим вначале

Определение 3.5.1.1 Ограничение $f_i(x) \ge 0$ называется *активным* на элементе x^*, если $f_i(x^*) \le 0$.

Пусть функционалы $\{ F(x), f_i(x), i = [1, m] \}$ непрерывно дифференцируемы на элементе $x^* \in E^n$, и $J(x^*)$ – множество индексов активных на элементе x^* ограничений, набор элементов

$\{\text{grad}\, f_i(x^*), i \in J(x^*)\}$ линейно независим и, наконец, $\delta x = x - x^*$, тогда оказывается справедливой

Теорема 3.5.1.1. **Пусть функционал $F(x)$ на множестве R принимает максимальное значение на элементе x^*. Тогда необходимое условие оптимальности имеет вид**

$$(\text{grad}\, F(x^*), \delta x) \le 0 \quad \text{и} \quad \forall \delta x \in M(x^*):$$
$$(\text{grad}\, f_i(x^*), \delta x) \ge 0, \; i \in J(x^*).$$

Доказательство.

В силу теоремы 3.2.1 и определения 3.2.2 для доказательства достаточно показать, что конус допустимых вариаций на элементе x^* определяется системой условий

$$(\text{grad}\, f_i(x^*), \delta x) \ge 0; \quad i \in J(x^*).$$

Действительно, из условий теоремы следует, что каждое из активных на x^* ограничений $f_i(x) \ge 0$ в малой окрестности элемента x^* может быть линейно аппроксимировано опорной гиперплоскостью

$$(\text{grad}\, f_i(x^*), x - x^*) = 0.$$

Тогда множество элементов $\delta x = x - x^*$ – допустимых на x^* вариаций – задается системой неравенств

$$(\text{grad}\, f_i(x^*), \delta x) \ge 0 \; i \in J(x^*).$$

Теорема доказана.

Утверждение теоремы 3.5.1.1 при замене δx на $x - x^*$ имеет вид

$$(\text{grad}\, F(x^*), x) \le (\text{grad}\, F(x^*), x^*) \qquad (3.5.1.1)$$
$$\forall x: (\text{grad}\, f_i(x^*), x) \ge (\text{grad}\, f_i(x^*), x^*), i \in J(x^*),$$

что, согласно *теореме Фаркаша*, равносильно существованию набора чисел $\{\lambda_1^*, \lambda_2^*, ..., \lambda_m^*\}$ таких, что

$$\lambda_i^* \geq 0 \quad \text{и} \quad \sum_{i \in J(x^*)} \lambda_i^* \operatorname{grad} f_i(x^*) = \operatorname{grad} F(x^*).$$

Действительно, если систему неравенств (3.5.1.1) привести к виду

$$(\operatorname{grad} F(x^*), \delta x) \leq 0$$

$$\forall \delta x : (-\operatorname{grad} f_i(x^*), \delta x) \leq 0, \ \ i \in J(x^*)$$

или, в координатной форме

$$\sum_{j=1}^{n} \frac{\partial F}{\partial x_j} \delta x_j \leq 0$$

$$\forall \delta x : \ -\sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} \delta x_j \leq 0 \ \ \forall i \in J(x^*)$$

и предположить, что каждое решение второй системы удовлетворяет первому неравенству, то по теореме Фаркаша должно сушествовать решение системы линейных уравнений вида

$$-\sum_{i \in J(x^*)} \frac{\partial f_i}{\partial x_j} \lambda^*{}_i = \frac{\partial F}{\partial x_j} \qquad \forall j = [1, n],$$

такое, что $\lambda^*{}_i \geq 0 \ \forall i \in J(x^*)$.

Иначе говоря, выполняются условия:

$$\lambda_i^* \geq 0 \quad \text{и} \qquad -\sum_{i \in J(x^*)} \lambda_i^* \operatorname{grad} f_i(x^*) = \operatorname{grad} F(x^*).$$

Заметим, что, если использовать обозначения из теоремы 1.1.4.4, то будут справедливы равенства:

$$\| b \| = \left\| \frac{\partial F}{\partial x_j} \right\| \quad \text{и} \quad \| A \| = \left\| -\frac{\partial f_i}{\partial x_j} \right\|.$$

Если доопределить $\lambda_i^* = 0$, $\forall i \notin J(x^*)$ и ввести в рассмотрение элемент $\Lambda \in E^m$, то последнее равенство принимает вид

$$\operatorname{grad} F(x^*) + \sum_{i=1}^{m} \lambda_i^* \operatorname{grad} f_i(x^*) = o;$$
$$\lambda_i^* f_i(x^*) = 0; \quad \lambda_i^* \geq 0; \quad \forall i = [1, m].$$

Таким образом оказывается справедливой

Теорема 3.5.1.2. (Куна–Таккера) **Пусть функционал $F(x)$ на множестве R принимает максимальное значение на элементе x^* и элементы $\operatorname{grad} f_i(x^*), i \in J(x^*)$ линейно независимы. Тогда существуют числа λ_i^*, $\forall i = [1, m]$ такие, что**

$$\operatorname{grad} F(x^*) + \sum_{i=1}^{m} \lambda_i^* \operatorname{grad} f_i(x^*) = o;$$
$$\lambda_i^* f_i(x^*) = 0; \quad \lambda_i^* \geq 0; \quad \forall i = [1, m].$$

Условие *линейной независимости* набора элементов $\{-\operatorname{grad} f_i(x^*)\}$, для которых $\lambda_i > 0$, равносильно требованию регулярности множества допустимых вариаций на x^*. Отметим также, что, равенства

$$\lambda_i^* f_i(x^*) = 0, \quad \forall i = [1, m]$$

принято называть условиями *дополняющей нежесткости*.

Полученные выше необходимые условия экстремальности удобнее формулировать, введя в рассмотрение специальный функционал, зависящий как от x, так и от $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ и имеющий вид:

$$L(x, \Lambda) = F(x) + \sum_{i=1}^{m} \lambda_i f_i(x).$$

В этом случае необходимые условия экстремальности формулируются как

Теорема 3.5.1.3. **Пусть функционал $F(x)$ на множестве R принимает максимальное значение на элементе x^* и элементы**

$$\operatorname{grad} f_i(x^*)\ ;\ i \in J(x^*)$$

линейно независимы. Тогда существуют числа $\lambda_i^* \geq 0\ \forall i = [1,m]$ такие, что

$$\operatorname{grad}_x L(x^*, \Lambda^*) = o\ ;$$

$$\lambda_i^* f_i(x^*) = 0; \quad \forall i = [1,m].$$

В теории математического программирования числа $\lambda_i^* \geq 0$, $i = [1,m]$ принято называть *множителями Лагранжа*, а функционал $L(x,\Lambda)$ – *функцией Лагранжа*.

§ 3.5.2. Функция Лагранжа и ее свойства

При исследовании свойств функции Лагранжа полезными оказываются следующие теоремы.

Теорема 3.5.2.1. **Справедливо равенство**

$$\max_{x \in R} F(x) = \max_x \min_{\Lambda \geq o} L(x,\Lambda)\,.$$

Доказательство.

В силу неотрицательности чисел λ_i, $i = [1,m]$ и структуры функции Лагранжа справедливо равенство

$$\min_{\Lambda \geq o} L(x,\Lambda) = \begin{cases} F(x) & x \in R, \\ -\infty & x \notin R. \end{cases}$$

Тогда, если x^* – решение исходной задачи, то $x^* \in R$ и

$$F(x^*) = \max_{x \in R} F(x) = \max_x \min_{\Lambda \geq o} L(x,\Lambda).$$

Обратно: если x^* – оптимальный элемент функционала

$$\max_x \min_{\Lambda \geq o} L(x,\Lambda)\,,$$

то $x^* \in R$ и справедливо $\max\limits_{x\in R} F(x) = \max\limits_{x} \min\limits_{\Lambda\geq o} L(x,\Lambda)$.

Теорема доказана.

Следствие 3.5.2.1. **Задачи поиска $\max\limits_{x\in R} F(x)$ и $\max\limits_{x} \min\limits_{\Lambda\geq o} L(x,\Lambda)$ равносильны.**

Вновь используя терминологию раздела 3.4, задачу поиска

$$\min_{\Lambda\geq o} \max_{x} L(x,\Lambda)$$

логично назвать *двойственной* к *прямой* задаче

$$\max_{x} \min_{\Lambda\geq o} L(x,\Lambda).$$

Теорема 3.5.2.2. **Справедливо соотношение**

$$\min_{\Lambda\geq o} \max_{x} L(x,\Lambda) \geq \max_{x} \min_{\Lambda\geq o} L(x,\Lambda).$$

Доказательство.

Очевидно, что $L(x,\Lambda) \geq \min\limits_{\Lambda\geq o} L(x,\Lambda)$.

Но тогда, как частный случай, верна оценка

$$\forall \Lambda \geq o:\ \max_{x} L(x,\Lambda) \geq \max_{x} \min_{\Lambda\geq o} L(x,\Lambda),$$

из которой следует, что

$$\min_{\Lambda\geq o} \max_{x} L(x,\Lambda) \geq \max_{x} \min_{\Lambda\geq o} L(x,\Lambda).$$

Теорема доказана.

Теоремы 3.5.2.1 и 3.5.2.2 справедливы без каких-либо предположений о выпуклости прямой задачи. Их утверждения позволяют получать в общем случае *верхнюю* оценку ее решения. Наложение же дополнительных условий, приводит к более сильным оценкам, таким как

Теорема 3.5.2.3. **Пусть $F(x)$ *выпукла вверх* на *выпуклом* множестве R, имеющем внутренние элементы (условие регулярности Слейтера), тогда верны равенства**

$$\min_{\Lambda\geq o} \max_{x} L(x,\Lambda) = L(x^*,\Lambda^*) = \max_{x} \min_{\Lambda\geq o} L(x,\Lambda).$$

Доказательство.

Поскольку утверждение теоремы 3.5.2.2 справедливо и в рассматриваемом случае, для доказательства достаточно убедиться в справедливости оценки

$$\min_{\Lambda \geq o} \max_{x} L(x, \Lambda) \leq \max_{x} \min_{\Lambda \geq o} L(x, \Lambda).$$

Рассмотрим евклидово пространство E^{m+1} с элементами y такими, что $\| y \| = \begin{Vmatrix} \eta_0 \\ \eta_1 \\ \dots \\ \eta_m \end{Vmatrix}$. В этом пространстве выделим множество Ω с элементами, удовлетворяющими системе неравенств

$$\begin{cases} F(x) \geq \eta_0 \\ f_1(x) \geq \eta_1 \\ \dots \\ f_m(x) \geq \eta_m \end{cases}$$

для некоторого фиксированного элемента x, а также множество Θ с элементами, удовлетворяющими системе условий вида

$$\begin{cases} F(x^*) < \eta_0 \\ \eta_1 = 0 \\ \dots \\ \eta_m = 0 \end{cases}.$$

Множества Ω и Θ по условию теоремы выпуклы и, что очевидно, по построению не имеют общих элементов в E^{m+1}.

Поэтому в силу теоремы 1.1.4.3 можно утверждать, что существует разделяющая множества Ω и Θ гиперплоскость. Например, вида

$$(l^*, y - y^*) = 0,$$

где $l^* \neq o$ и $\left\| y^* \right\| = \begin{Vmatrix} \eta_0^* \\ \eta_1^* \\ \dots \\ \eta_m^* \end{Vmatrix} = \begin{Vmatrix} F(x^*) \\ 0 \\ \dots \\ 0 \end{Vmatrix}$.

В этом случае $\forall y \in \Omega$ будет справедлива оценка $(l^*, y) \le (l^*, y^*)$, причем $l^* \ge o$, поскольку множеству Ω принадлежат элементы со сколь угодно большими по модулю и отрицательными по знаку координатами.

Если координатное представление $\left\| l^* \right\| = \begin{Vmatrix} \lambda_0^* \\ \lambda_1^* \\ \dots \\ \lambda_m^* \end{Vmatrix}$, то последнее неравенство можно записать как

$$\lambda_0^* \eta_0 + \sum_{i=1}^{m} \lambda_i^* \eta_i \le \lambda_0^* F(x^*); \quad \forall x.$$

А поскольку это неравенство верно $\forall y \in \Omega$, то оно будет верным и для элемента $\left\| y \right\| = \begin{Vmatrix} \eta_0 \\ \eta_1 \\ \dots \\ \eta_m \end{Vmatrix} = \begin{Vmatrix} F(x) \\ f_1(x) \\ \dots \\ f_m(x) \end{Vmatrix}$, что дает оценку

$$\lambda_0^* F(x) + \sum_{i=1}^{m} \lambda_i^* f_i(x) \le \lambda_0^* F(x^*); \quad \forall x.$$

Заметим также, что $\lambda_0^* > 0$. Действительно, из предположения $\lambda_0^* = 0$ в силу $\lambda_j^* \ge 0 \ \forall j = [1, m]$ получаем, что

$$\sum_{j=1}^{m} \lambda^*_j f_j(x) \le 0\,;\ \forall x\,,$$

а это противоречит условию регулярности Слейтера.

Если $\lambda^*_0 > 0$, то можно пронормировать неравенство

$$\lambda^*_0 F(x) + \sum_{i=1}^{m} \lambda^*_i f_i(x) \le \lambda^*_0 F(x^*)\,;\quad \forall x\,,$$

поделив обе его части на λ^*_0. Тогда, сохранив прежние обозначения для пронормированных компонентов элемента Λ, получаем оценку

$$F(x) + \sum_{i=1}^{m} \lambda^*_i f_i(x) \le F(x^*)\,;\quad \forall x\,.$$

При этом в силу произвольности x будет справедливо и неравенство

$$\max_x \left[F(x) + \sum_{i=1}^{m} \lambda^*_i f_i(x) \right] \le F(x^*)\,,$$

то есть, $\max\limits_x L(x, \Lambda^*) \le F(x^*)$.

Наконец, из очевидного

$$\max_x L(x, \Lambda^*) \ge \min_{\Lambda \ge o} \max_x L(x, \Lambda)\,,$$

учитывая, что $F(x^*) = \max\limits_{x \in R} F(x) = \max\limits_x \min\limits_{\Lambda \ge o} L(x, \Lambda^*)$, получаем требуемое

$$\max_x \min_{\Lambda \ge o} L(x, \Lambda) \ge \min_{\Lambda \ge o} \max_x L(x, \Lambda)\,.$$

И в сочетании с ранее полученным (теорема 3.5.2.2)

$$\min_{\Lambda \ge o} \max_x L(x, \Lambda) \ge \max_x \min_{\Lambda \ge o} L(x, \Lambda)$$

приходим к равенству

$$\min_{\Lambda \ge o} \max_x L(x, \Lambda) = L(x^*, \Lambda^*) = \max_x \min_{\Lambda \ge o} L(x, \Lambda)\,.$$

Теорема доказана.

Следствие 3.5.2.2. **Пара элементов $\{x^*, \Lambda^*\}$ есть (см. определение 2.1.2) *седловая точка* функции Лагранжа.**

§ 3.5.3. Достаточные условия разрешимости задачи математического программирования

Исходя из вышеизложенного (подобно случаю поиска безусловного экстремума), для задачи математического программирования можно сформулировать *достаточные условия второго порядка* существования экстремума на элементе x^*.

Теорема 3.5.3.1. **Пусть существует элемент x^* и набор чисел**

$$\lambda_i^* \geq 0 \,;\, i = [1, m]$$

таких, что

$$\begin{aligned} & f_i(x^*) \geq 0\,, \;\; i = [1, m]; \\ & \lambda_i^* f_i(x^*) = 0\,, \, i = [1, m] \\ & \operatorname{grad}_x L(x^*, \Lambda^*) = o \end{aligned} \quad \textbf{и}$$

и пусть для любого элемента $z \in E^n$ такого, что

$$(\operatorname{grad}_x f_i(x^*), z) = 0\,, \; \forall i \in J^*,$$

$$\text{где } J^* : \left\{ i \,\middle|\, \lambda_i^* > 0 \right\}$$

и

$$(\operatorname{grad}_x f_i(x^*), z) \geq 0\,, \; \forall i \in G,$$

$$\text{где } G : \left\{ i \,\middle|\, \lambda_i^* = 0 \bigcap f_i(x^*) \geq 0 \right\}$$

выполняется неравенство

$$\left(z,\ \hat{\operatorname{Hess}} L(x^*, \Lambda^*)\, z \right) < 0\,,$$

тогда x^* – решение исходной задачи математического программирования.

Доказательство.

Предположим противное. Пусть существует сходящаяся к x^* последовательность $\{x_k\}$ $(x_k \neq x^*, \forall k)$ такая, что

$$F(x_k) \geq F(x^*).$$

Тогда, в силу соотношений дополняющей нежесткости, имеем

$$L(x_k, \Lambda^*) = F(x_k) + \sum_{i=1}^{m} \lambda_i^* f(x_k) \geq F(x^*) =$$

$$= F(x^*) + \sum_{i=1}^{m} \lambda_i^* f(x^*) \quad \Rightarrow \quad L(x_k, \Lambda^*) \geq L(x^*, \Lambda^*).$$

По формуле Тейлора

$$L(x_k, \Lambda^*) = L(x^*, \Lambda^*) + \left(\operatorname{grad}_x L(x^*, \Lambda^*), (x_k - x^*)\right) +$$

$$+ \frac{1}{2}\left((x_k - x^*), \overset{\wedge}{\operatorname{Hess}} L(x^*, \Lambda^*)(x_k - x^*)\right) + o\left(\left| x_k - x^* \right|^2\right)$$

Тогда, в силу соотношений

$$\operatorname{grad}_x L(x^*, \Lambda^*) = o \quad \text{и} \quad L(x_k, \Lambda^*) \geq L(x^*, \Lambda^*),$$

получаем

$$\frac{1}{2}\left((x_k - x^*), \overset{\wedge}{\operatorname{Hess}} L(x^*, \Lambda^*)(x_k - x^*)\right) + o\left(\left| x_k - x^* \right|^2\right) \geq 0. \tag{3.5.3.1}$$

Рассмотрим вспомогательную последовательность с общим членом $z_k = \dfrac{x_k - x^*}{\left| x_k - x^* \right|}$. Очевидно, что $z_k \to z$ при $k \to +\infty$, такому, что $\left| z \right| = 1$.

В этом случае, при $\forall i \in J$

$$\lim_{k \to +\infty} \frac{f_i(x_k) - f_i(x^*)}{\left| x_k - x^* \right|} =$$

$$= \lim_{k \to +\infty} \frac{\left(\text{grad}_x f_i(x^*), (x_k - x^*)\right) + o(\left| x_k - x^* \right|)}{\left| x_k - x^* \right|} =$$

$= \left(\text{grad}_x f_i(x^*), z\right) = 0$, поскольку $\forall i \in J^*$:

$$f_i(x_k) = f_i(x^*) = 0.$$

Аналогично получаем, что при $\forall i \in G$

$$\lim_{k \to +\infty} \frac{f_i(x_k) - f_i(x^*)}{\left| x_k - x^* \right|} = \left(\text{grad}_x f_i(x^*), z\right) \geq 0.$$

Наконец, поделив на $\left| x_k - x^* \right|$ почленно неравенство (3.5.3.1) и перейдя к пределу при $k \to +\infty$, получим

$$\left(z,\ \hat{\text{Hess}}\, L(x^*, \Lambda^*)\, z\right) \geq 0,$$

что противоречит условию теоремы.

Теорема доказана.

Раздел 3.6 О МЕТОДАХ РЕШЕНИЯ ЗАДАЧ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Поиск решения задачи математического программирования общего вида представляет собой в большинстве случаев чрезвычайно сложную вычислительную проблему. В связи с чем, представляется целесообразным выделение некоторых классов задач, для которых имеются практически эффективные инструментальные средства поиска и анализа их решений.

В первую очередь это задачи линейного программирования, которым целиком посвящена глава 4, а также нелинейные задачи с ограничениями типа "равенство", рассматриваемые ниже.

§ 3.6.1. Задачи математического программирования с ограничениями типа «равенство»

В практически важном частном случае множество R может состоять из элементов E^n, удовлетворяющих системе уравнений вида

$$f_i(x) = 0;\ \forall i = [1, m]. \qquad (3.6.1.1)$$

Данную задачу в E^n принято называть задачей *поиска условного экстремума с ограничениями типа "равенство"*. При этом обычно предполагается, что $m < n$ и система элементов

$$\{\text{grad}\, f_1(x), \text{grad}\, f_2(x), \ldots, \text{grad}\, f_m(x),\}$$

линейно независима для некоторых $x \in R$.

В координатной форме задачу отыскания условного экстремума принято записывать в виде

найти максимум $F(\xi_1, \xi_2, \ldots, \xi_n)$ *по* $\{\xi_1, \xi_2, \ldots, \xi_n\}$,

при условиях:

$$\begin{cases} f_1(\xi_1, \xi_2, \ldots, \xi_n) = 0, \\ f_2(\xi_1, \xi_2, \ldots, \xi_n) = 0, \\ \qquad \ldots \\ f_m(\xi_1, \xi_2, \ldots, \xi_n) = 0. \end{cases} \qquad (3.6.1.2)$$

Удобная и применимая для достаточно широкого класса задач форма записи необходимых и достаточных условий существования условного экстремума основана на использовании *функции Лагранжа* – вспомогательного функционала вида, определенного ранее в §3.5.1

$$L(x, \Lambda) = F(x) + \sum_{i=1}^{m} \lambda_i f_i(x),$$

где каждое из условий вида $f_i(x) = 0$ можно заменить равносильной системой неравенств $\begin{cases} f_i(x) \geq 0 \\ -f_i(x) \geq 0 \end{cases}$, а сама функция Лагранжа приводится к виду $L(x, \Lambda) = F(x) + \sum_{i=1}^{m} (\lambda_i^+ - \lambda_i^-) f_i(x)$.

Поскольку в задаче математического программирования числа λ_i^+ и λ_i^- должны быть неотрицательными, то коэффициенты $\lambda_i = \lambda_i^+ - \lambda_i^-$ не имеют ограничения на знак, и условия экстремальности существенно упрощаются.

Будем предполагать, что функционалы $F(x)$ и $f_i(x)$, $\forall i = [1,m]$ *непрерывно дифференцируемы*, а элементы $\operatorname{grad} f_i(x)$, $\forall i = [1,m]$ *линейно независимы* на экстремальном элементе x^*. Заметим, что последнее условие в координатном представлении равносильно равенству $\operatorname{rg}\| J \| = m$, где

$$\| J \| = \begin{Vmatrix} \frac{\partial f_1}{\partial \xi_1} & \frac{\partial f_1}{\partial \xi_2} & \cdots & \frac{\partial f_1}{\partial \xi_n} \\ \frac{\partial f_2}{\partial \xi_1} & \frac{\partial f_2}{\partial \xi_2} & \cdots & \frac{\partial f_2}{\partial \xi_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial \xi_1} & \frac{\partial f_m}{\partial \xi_2} & \cdots & \frac{\partial f_m}{\partial \xi_n} \end{Vmatrix}$$

– *матрица Якоби* системы функций $\{f_1(x), f_2(x), \ldots f_m(x)\}$.

В методе множителей Лагранжа *необходимые* условия существования условного экстремума на элементе x^* будут иметь вид: существует Λ^* такое, что

$$\begin{cases} \left.\dfrac{\partial L}{\partial \xi_j}\right|_{x = x^*,\ \Lambda = \Lambda^*} = 0\ , \ \forall j = [1,n], \\ f_i(x^*) = 0\ , \qquad \forall i = [1,m]. \end{cases} \tag{3.6.1.3}$$

Иначе говоря, пара элементов $\{x^*, \Lambda^*\}$ является стационарной для $L(x,\Lambda)$ и удовлетворяет всем условиям связи

$$f_i(x^*) = 0\ , \ \forall i = [1,m]\ .$$

Рассмотрим теперь *достаточные* условия.

Пусть функционалы $F(x)$ и $f_i(x)$, $\forall i = [1, m]$ дважды непрерывно дифференцируемы на элементе x^*, для которого справедливы сформулированные выше необходимые условия, тогда из *положительной* (*отрицательной*) *определенности* второго дифференциала

$$d^2 L = (dx, \hat{\text{Hess}}\, L(x^*, \Lambda^*) dx)$$

на множестве допустимых вариаций dx, удовлетворяющих соотношениям

$$(\text{grad}\, f_i(x^*), dx) = 0\,,\ \forall i = [1, m]\,, \qquad (3.6.1.4)$$

следует, что x^* есть решение задачи на локальный условный минимум (максимум).

Важно отметить, что знаковая определенность второго дифференциала

$$d^2 L = \sum_{k=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 L}{\partial \xi_k \partial \xi_i} d\xi_k d\xi_i$$

исследуется не на множестве произвольных вариаций $\{d\xi_1, d\xi_2, \ldots, d\xi_n\}$, а на $n - m$-мерном подпространстве в E^n, определяемом (в координатном представлении) однородной системой m *линейных* уравнений с n неизвестными:

$$\sum_{k=1}^{n} \frac{\partial f_j}{\partial \xi_k} d\xi_k = 0\,,\ \forall j = [1, m]\,,$$

ранг основной матрицы которой равен m.

Таким образом, решение задачи на условный экстремум состоит из:

1º. Решения нелинейной системы уравнений (3.6.1.3), то есть определения стационарных элементов задачи (3.6.1.2).

2º. Исследования каждой из этих стационарных точек на экстремальность в подпространстве, задаваемом системой линейных уравнений (3.6.1.4).

Стоит отметить, что при решении задачи (3.6.1.1) теоретически допустимой альтернативой методу множителей Лагранжа является тривиальный метод исключения части переменных при помощи соотношений (3.6.1.2), хотя недостатки этого подхода вполне очевидны.

Основной особенностью, осложняющей решение задач математического программирования вида (3.5.1), является задание ограничений на компоненты $\{\xi_1, \xi_2, \ldots, \xi_n\}$ координатного представления элемента $x \in E^n$ в форме системы неравенств

$$\begin{cases} f_1(\xi_1, \xi_2, \ldots, \xi_n) \geq 0 \\ f_2(\xi_1, \xi_2, \ldots, \xi_n) \geq 0 \\ \qquad \ldots \\ f_m(\xi_1, \xi_2, \ldots, \xi_n) \geq 0 \end{cases}, \tag{3.6.1.5}$$

что не позволяет однозначно выражать одни компоненты $\{\xi_1, \xi_2, \ldots, \xi_n\}$ через другие, и в свою очередь, делает принципиально невозможным использование прямых методов типа исключения.

С другой стороны, использование формализма Лагранжа для задач математического программирования не позволяет
получить априорную информацию об их стационарных точках. Поиск этой информации требует (как минимум) выделения *множества активных ограничений*, то есть разбиения всей системы ограничений на ограничения типа "*равенство*" и "*строгое неравенство*".

Полное число вариантов таких разбиений, каждое из которых сводит исходную задачу математического программирования (3.5.1) к "более простой" задаче на *условный экстремум*, как нетрудно оценить, составляет 2^m. Для вполне реальных значений, например $m = 100$, данное число является величиной порядка 10^{30}, что демонстрирует, мягко говоря, "малоэффективность" данного подхода.

Для практического решения задач математического программирования обычно используются схемы типа (3.2.1), хотя в каждом конкретном случае сходимость такой итерационной процедуры требует обоснования, а ее эффективность – специального исследования.

Ниже, в качестве иллюстрации, будет детально рассмотрен один из таких методов, используемых для решения задач математического программирования, так называемый метод "штрафных функций".

Раздел 3.7. МЕТОД ШТРАФНЫХ ФУНКЦИЙ

§ 3.7.1. Описание алгоритма

Метод штрафных функций в значительном числе случаев считается наиболее подходящим средством решения задач математического программирования алгоритмом, хотя и обладающим рядом свойств, ограничивающих его применение.

Идея метода штрафных функций, сформулированная впервые Курантом в 1942 году, заключается в том, что вместо исходной задачи математического программирования (3.5.1) решается задача поиска экстремума специального вспомогательного функционала *без каких-либо ограничений на* $x \in E^n$.

Этот вспомогательный функционал, который будем обозначать $A(\tau, x)$, выбирается равным целевому функционалу исходной задачи $F(x)$, к которому добавлены слагаемые $P(\tau, \alpha)$, "штрафующие" нарушение каждого из условий $f_i(x) \geq 0$, $i = [1, m]$. Механизм штрафа заключается в том, что эта добавка мала, если соответствующее ограничение не нарушено, но отрицательна и велика по модулю, если $f_i(x) < 0$, $i = [1, m]$ на элементе x.

При использовании метода штрафных функций предполагается, что добавка $P(\tau, \alpha)$, где $\tau > 0$ – некоторый параметр[4], штрафующая нарушение ограничения вида $\alpha \geq 0$, удовлетворяет при каждом фиксированном значении α предельному соотношению

$$\lim_{\tau \to +0} P(\tau, \alpha) = \begin{cases} 0, & \alpha \geq 0, \\ +\infty, & \alpha < 0. \end{cases} \qquad (3.7.1.1)$$

При этом поиск максимума вспомогательного функционала осуществляется при *фиксированном* значении параметра τ, однако его значение можно менять и использовать τ как регулятор меры штрафа за "единицу нарушения ограничения".

---

[4] Обычно $P(\tau, \alpha)$ называют *штрафной функцией*, а параметр τ – *коэффициентом штрафа*.

Таким образом, решение задачи математического программирования

найти максимум $F(x)$ по $x \in E^n$,

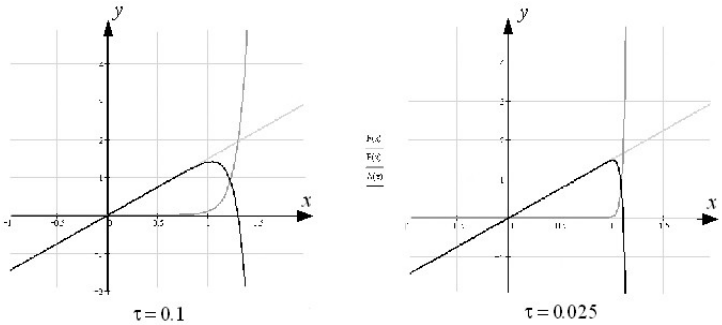
при условиях: $f_i(x) \geq 0, \forall i = [1, m]$ (3.7.1.2)

сводится к максимизации вспомогательной функции

$$A(\tau, x) = F(x) - \sum_{i=1}^{m} P(\tau, f_i(x))$$

без каких-либо ограничений.

Элемент $\bar{x}(\tau)$, на котором вспомогательная функция $A(\tau, x)$ достигает своего максимума, является приближенным решением задачи (3.7.1.2), причем величина погрешности будет стремиться к нулю при $\tau \to +0$. Рис.3.7.1.1 иллюстрирует изменение вида вспомогательной функции при уменьшении коэффициента штрафа.



$\tau = 0.1$ $\tau = 0.025$

Рисунок 3.7.1.1.

В силу соотношения (3.7.1.1), для *любой* числовой последовательности положительных чисел $\{\tau_k\} \to +0$ при $k \to +\infty$, $k \in N$, будут выполняться предельные равенства:

$$\lim_{k \to +\infty} A(\tau_k, \bar{x}(\tau_k)) = F(x^*),$$

$$\lim_{k \to +\infty} \bar{x}(\tau_k) = x^*,$$

где $\bar{x}(\tau_k)$ – экстремальный по x элемент функционала $A(\tau, x)$ при фиксированном, положительном значении параметра $\tau = \tau_k$.

Если достаточно гладкая штрафная функция $P(r,\alpha)$ удовлетворяет также и неравенствам

$$\frac{\partial P}{\partial \alpha} < 0 \quad \text{и} \quad \frac{\partial^2 P}{\partial \alpha^2} > 0, \forall \alpha,$$

то, можно показать, что, согласно определению предела по Гейне, данные предельные соотношения равносильны условиям

$$\begin{aligned} &\lim_{\tau \to +0} A(\tau, \bar{x}(\tau)) = F(x^*), \\ &\lim_{\tau \to +0} \bar{x}(\tau) = x^*. \end{aligned} \tag{3.7.1.3}$$

Если же $A(\tau, x)$ достаточно гладкий в E^n функционал, имеющий изолированный локальный экстремальный по x элемент, то неявно заданная *элемент-функция* $\bar{x}(\tau)$ определяется равенством:

$$\operatorname{grad}_x A(\tau, \bar{x}(\tau)) = o\,. \tag{3.7.1.4}$$

Координатное представление последнего равенства будет иметь вид: $\frac{\partial A}{\partial \xi_k} = 0$, $\forall k = [1,n]$, причем согласно *теореме о неявных функциях*, компоненты элемента $\frac{d\bar{x}(\tau)}{d\tau}$ могут быть найдены из системы линейных уравнений

$$\sum_{j=1}^{n} \frac{\partial^2 A}{\partial \xi_k \partial \xi_j} \frac{d\bar{\xi}_j}{d\tau} = -\frac{\partial^2 A}{\partial \xi_k \partial \tau}, \forall k = [1,n].$$

§ 3.7.2. Проблема точности

При использовании метода гладких штрафных функций возникает так называемая проблема точности. Поясним суть этой проблемы на следующем примере.

Рассмотрим задачу: *найти максимум* $3\xi_1 + 2\xi_2$ на $\{\xi_1, \xi_2\} \in E^2$,

$$\textit{при условиях}: \begin{cases} \xi_1 \le 2, \\ \xi_2 \le 1. \end{cases} \tag{3.7.2.1}$$

В качестве штрафной функции выберем $P(\tau,\alpha)=\tau e^{-\frac{\alpha}{\tau}}$, тогда вспомогательная функция будет иметь вид

$$A(\tau,\xi_1,\xi_2)=3\xi_1+2\xi_2-\tau e^{-\frac{2-\xi_1}{\tau}}-\tau e^{-\frac{1-\xi_2}{\tau}},$$

условия стационарности на элементе $\|\bar{x}\|=\left\|\begin{matrix}\bar{\xi}_1\\ \bar{\xi}_2\end{matrix}\right\|$ которой:

$$\begin{cases}\dfrac{\partial A}{\partial\xi_1}=3-e^{-\frac{\bar{\xi}_1-2}{\tau}}=0,\\ \dfrac{\partial A}{\partial\xi_2}=2-e^{-\frac{\bar{\xi}_2-1}{\tau}}=0.\end{cases}$$

Откуда находим, что $\bar{\xi}_1(\tau)=2-\tau\ln 3$ и $\bar{\xi}_2(\tau)=1-\tau\ln 2$, и, следовательно, $\xi_1^*=\lim\limits_{\tau\to+0}\bar{\xi}_1(\tau)=2$; $\xi_2^*=\lim\limits_{\tau\to+0}\bar{\xi}_2(\tau)=1$, а точное экстремальное значение функционала на этом элементе равно 8. Таким образом, для данной задачи метод штрафных функций дает решение с погрешностью порядка величины параметра τ.

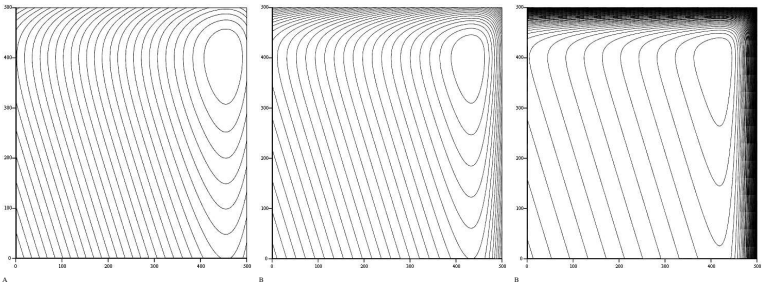


Рис. 3.7.2.1.

На рис.3.7.2.1 приведены системы изолиний вспомогательного функционала для значений коэффициента штрафа

$$\tau=0.5,\ \tau=0.3\ \text{ и }\ \tau=0.17.$$

А поскольку густота изолиний пропорциональна норме градиента, то данный рисунок наглядно демонстрирует увеличение "штрафа" за нарушение ограничений при $\tau \to +0$.

Чтобы оценить порядок величины вносимой погрешности рассмотрим разложение функции $\bar{x}(\tau)$ по формуле Тейлора в окрестности некоторого $\tau > 0$ до $o(\Delta\tau)$:

$$\bar{x}(\tau + \Delta\tau) = \bar{x}(\tau) + \frac{\partial \bar{x}}{\partial \tau}\Delta\tau + o(\Delta\tau),$$

из которого следует оценка при $\Delta\tau \to -\tau$ сверху:

$$\bar{x}(0) = \bar{x}(\tau) - \frac{\partial \bar{x}}{\partial \tau}\tau + o(\tau). \qquad (3.7.2.2)$$

Последнее соотношение означает, что погрешность метода по порядку малости будет совпадать с τ при условии, что вектор-функция $\bar{x}(\tau)$ непрерывно дифференцируема по τ.

С другой стороны, вектор-функция $\frac{\partial \bar{x}}{\partial \tau}\tau$ в формуле (3.7.2.2) может рассматриваться как корректирующая поправка, уменьшающая порядок погрешности, вносимой методом гладких штрафных функций. Компоненты этой вектор-функции являются решением следующей системы линейных уравнений:

$$\sum_{i=1}^{n} \frac{\partial^2 A}{\partial \xi_i \partial \xi_j} \frac{\partial \bar{\xi}_i}{\partial \tau} = -\frac{\partial^2 A}{\partial \xi_j \partial \tau}; \quad \forall j = [1, l],$$

получаемой дифференцированием соотношений (3.7.1.4) по параметру τ.

§ 3.7.3. Проблема сходимости

Отмеченную выше проблему погрешности можно, казалось бы, легко решить, используя свойство (3.7.1.3), то есть, полагая значение коэффициента штрафа в процедуре максимизации вспомогательной функции достаточно малым. Однако, следующий пример наглядно демонстрирует возникающие при этом осложнения.

Пусть требуется решить задачу математического программирования:

найти минимум $F(x)=\xi_1^2+2\xi_2^2+3\xi_3^2$ *по* $\{\xi_1,\xi_2,\xi_3\}$,

при условиях:

$$\begin{aligned} 2\xi_1 \quad +\xi_2 &= 3,\\ \xi_1+3\xi_3 &= 4. \end{aligned} \qquad (3.7.3.1).$$

Как было показано в §3.7,1 метод штрафных функций заключается в замене исходной задачи на условный экстремум последовательностью задач без ограничений, экстремальные элементы которых сходятся к решению исходной задачи.

Функцию штрафа выберем $P(\tau,\alpha)=\dfrac{\alpha^2}{2\tau}$, тогда вспомогательный функционал для рассматриваемой демонстрационной задачи будет иметь вид

$$A(\tau,x)=\xi_1^2+2\xi_2^2+3\xi_3^2+\frac{(2\xi_1+\xi_2-3)^2}{2\tau}+\frac{(\xi_1+3\xi_3-4)^2}{2\tau}.$$

Соответственно, условия его стационарности по $\{\xi_1,\xi_2,\xi_3\}$ на элементе $\bar{x}(\tau)$ будут

$$\begin{cases} \dfrac{\partial A}{\partial\xi_1}=2\bar{\xi}_1+\dfrac{2(2\bar{\xi}_1+\bar{\xi}_2-3)}{\tau}+\dfrac{\bar{\xi}_1+3\bar{\xi}_3-4}{\tau}=0,\\ \dfrac{\partial A}{\partial\xi_2}=4\bar{\xi}_2+\dfrac{2\bar{\xi}_1+\bar{\xi}_2-3}{\tau}=0,\\ \dfrac{\partial A}{\partial\xi_3}=6\bar{\xi}_3+\dfrac{3(\bar{\xi}_1+3\bar{\xi}_3-4)}{\tau}=0 \end{cases}$$

или

$$\begin{cases} (5+2\tau)\bar{\xi}_1 \quad +2\bar{\xi}_2 \quad +3\bar{\xi}_3=10,\\ 2\bar{\xi}_1+(1+4\tau)\bar{\xi}_2 \quad =3,\\ \bar{\xi}_1 \quad +(3+\tau)\bar{\xi}_3=4. \end{cases} \qquad (3.7.3.2)$$

Отметим, что переход к пределу при $\tau\to+0$ здесь невозможен, ибо в этом случае основная матрица системы выродится. Более того, чем

меньше значение положительного параметра штрафа τ, тем ближе к нулю детерминант основной матрицы этой системы и тем значительнее вычислительные затруднения при ее решении – в первую очередь необходимость повышения точности вычислений, например, при использовании теоремы Крамера или метода Гаусса.

Преодолеть эти затруднения можно, используя опять-таки формулу Тейлора. Из системы линейных уравнений (3.7.3.2) найдем $\overline{\xi}_1(\tau)$, подставив выражения для $\overline{\xi}_2(\tau)$ и $\overline{\xi}_3(\tau)$ через $\overline{\xi}_1(\tau)$ в первое уравнение. Тогда

$$(5+2\tau)\overline{\xi}_1 + 2\frac{3-2\overline{\xi}_1}{1+4\tau} + 3\frac{4-\overline{\xi}_1}{3+\tau} = 10\,,$$

что для $\overline{\xi}_1(\tau)$ дает

$$\overline{\xi}_1(\tau) = \frac{\frac{80}{3}\tau + o(\tau)}{\frac{56}{3}\tau + o(\tau)} \quad \text{или} \quad \overline{\xi}_1(\tau) = \frac{10 + \frac{o(\tau)}{\tau}}{7 + \frac{o(\tau)}{\tau}} = \frac{10}{7} + O(\tau)\,.$$

И вот теперь, переходя к пределу при $\tau \to +0$, получаем, что $\xi_1^* = \frac{10}{7}$. Аналогично находим, что $\xi_2^* = \frac{1}{7}$ и $\xi_3^* = \frac{6}{7}$.

§ 3.7.4. Связь с методом множителей Лагранжа

Из условий (3.7.1.3) и предположений о свойствах (в данном случае, непрерывности) штрафной функции следует существование пределов

$$\lim_{\tau \to +0} \frac{\partial P(\overline{x}(\tau))}{\partial f_i}\,, \quad \forall i = [1, m]\,.$$

Если теперь сопоставить условие $\operatorname{grad}_x A(\tau, \overline{x}(\tau)) = o$, записанное для задачи на минимум в форме

$$\operatorname{grad} F(x) + \sum_{i=1}^{m} \left(-\frac{\partial P}{\partial f_i}\right) \operatorname{grad} f_i = o$$

с утверждением теоремы 3.5.1.2 (Куна–Таккера), то в силу существования и единственности множителей Лагранжа $\lambda_i^*, \forall i = [1,m]$, можно прийти к заключению, что для изолированного локального решения x^* справедливы следующие предельные соотношения:

$$\lim_{\tau \to +0} \frac{\partial P(\bar{x}(\tau))}{\partial f_i} = -\lambda_i^* \quad \forall i = [1,m]$$

Иначе говоря, пределы вида $\lim_{\tau \to +0} \frac{\partial P(\bar{x}(\tau))}{\partial f_i}$ совпадают со значениями множителей Лагранжа на оптимальном элементе, когда последние существуют и единственны. Следовательно, величины $\frac{\partial P(\bar{x}(\tau))}{\partial f_i}$ можно использовать для оценки величин множителей Лагранжа.

В качестве примера рассмотрим задачу, двойственную к задаче (3.7.2.1):

найти минимум $2\lambda_1 + \lambda_2$, *при условиях*: $\begin{cases} \lambda_1 \geq 3, \\ \lambda_2 \geq 2, \end{cases}$

решение которой $\begin{cases} \lambda_1^* = 3, \\ \lambda_2^* = 2, \end{cases}$ можно выразить также и при помощи предельных соотношений (см. задачу (3.7.2.1)).

$$\begin{cases} \lim\limits_{\tau \to +0} e^{-\frac{\bar{\xi}_1(\tau) - 2}{\tau}} = 3, \\ \lim\limits_{\tau \to +0} e^{-\frac{\bar{\xi}_2(\tau) - 1}{\tau}} = 2 \end{cases}$$

с экстремальным значением двойственного функционала

$$2\lambda_1^* + \lambda_2^* = 8.$$

Аналогичным образом соотношения

$$\lim_{\tau \to +0} \bar{\lambda}_j(\tau) = \lambda_j^*, \text{ где } \bar{\lambda}_j(\tau) = -\frac{\partial P(\bar{x}(\tau))}{\partial f_j}; \quad \forall j = [1,m].$$

можно применить для оценки значений множителей Лагранжа в задаче (3.7.3.1)

$$\frac{\partial P(\bar{x}(\tau))}{\partial f_1} = -\frac{2\bar{\xi}_1(\tau) + \bar{\xi}_2(\tau) - 3}{\tau},$$

$$\frac{\partial P(\bar{x}(\tau))}{\partial f_2} = -\frac{\bar{\xi}_1(\tau) + 3\bar{\xi}_3(\tau) - 4}{\tau},$$

и поэтому получаем

$$\bar{\lambda}_1(\tau) = -\frac{4}{7} + \mathrm{O}(\tau) \quad \text{и} \quad \bar{\lambda}_1(\tau) = -\frac{12}{7} + \mathrm{O}(\tau).$$

Глава 4
ЗАДАЧА ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Раздел 4.1. ПОСТАНОВКИ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Единственный класс задач математического программирования, для которого разработаны практически эффективные методы решения, составляют так называемые задачи *линейного программирования* (ЛП), то есть задачи отыскания в E^n экстремумов целевого *линейного* функционала $F(x)$ при условиях типа "нестрогое неравенство"[5], накладываемых на значения некоторого конечного набора *линейных* функционалов $\{f_j(x), j=[1,m]\}$, который задает допустимое множество задачи ЛП.

Рассмотрим конкретные формы постановки задач линейного программирования.

Пусть некоторый элемент $x \in E^n$ имеет в ортонормированном базисе $\{e_1, e_2, \ldots, e_n\}$ координатное представление

$$\|x\| = \begin{Vmatrix} \xi_1 \\ \xi_2 \\ \ldots \\ \xi_n \end{Vmatrix}.$$

[5] Включающих как частный случай и условия типа "равенство".

Как уже отмечалось в §1.2, каждый линейный функционал в E^n (в том числе и целевой функционал $F(x)$) полностью и однозначно задается в базисе $\{e_1, e_2, \ldots, e_n\}$ n-компонентной строкой

$$\| c \| = \| \sigma_1 \quad \sigma_2 \quad \ldots \quad \sigma_n \|,$$

где $\sigma_j = F(e_j)$, $\forall j = [1, n]$, а его значение на элементе x находится по одной из следующих равносильных формул:

$F(x) = (c, x)$ – символическая форма;

$F(x) = \| c \| \| x \|$ – матричная форма;

$F(x) = \sum_{j=1}^{n} \sigma_j \xi_j$ – координатная форма.

В задаче ЛП допустимое множество R задается системой неравенств вида

$$\begin{cases} f_1(x) \geq \beta_1, \\ f_2(x) \geq \beta_2, \\ \quad \ldots \\ f_m(x) \geq \beta_m, \end{cases}$$

где $f_i(x) = (a_i, x)$ – i-ый ограничивающий линейный функционал, для которого, кроме символического, также допустимы матричное и координатное представления

$$f_i(x) = \| a_i \| \| x \|,$$

$$f_i(x) = \sum_{j=1}^{n} \alpha_{ij} \xi_j,$$

где n-компонентная строка $\| a_i \|$ есть координатное представление линейного функционала $f_i(x)$ в E^n, то есть

$$\| a_i \| = \| \alpha_{i1} \quad \alpha_{i2} \quad \ldots \quad \alpha_{in} \|, \text{ где } i = [1, m].$$

Заметим, что совокупность строк $\left\| \alpha_{j1} \quad \alpha_{j2} \quad \dots \quad \alpha_{jn} \right\|$ при необходимости может быть также записана и в виде матрицы размера $m \times n$

$$\| A \| = \begin{Vmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \dots & \dots & \dots & \dots \\ \alpha_{m1} & \alpha_{m2} & \dots & \alpha_{mn} \end{Vmatrix}.$$

Введем обозначение: $\| b \| = \begin{Vmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_m \end{Vmatrix}$. Тогда множество R – допустимых состояний – может быть задано системой условий в одной из следующих форм

$$\begin{cases} (a_1, x) \ge \beta_1, \\ (a_2, x) \ge \beta_2, \\ \dots \\ (a_m, x) \ge \beta_m, \end{cases} \qquad \begin{cases} \| a_1 \| \| x \| \ge \beta_1, \\ \| a_2 \| \| x \| \ge \beta_2, \\ \dots \\ \| a_m \| \| x \| \ge \beta_m, \end{cases}$$

$$\begin{cases} \sum_{j=1}^{n} \alpha_{1j} \xi_j \ge \beta_1, \\ \sum_{j=1}^{n} \alpha_{2j} \xi_j \ge \beta_2, \\ \dots \\ \sum_{j=1}^{n} \alpha_{mj} \xi_j \ge \beta_m, \end{cases} \text{ или же } \qquad \| A \| \| x \| \ge \| b \|.$$

Формы постановки задачи линейного программирования при необходимости могут быть изменены при помощи правил равносильности неравенств и очевидных свойств операций нахождения максимума или минимума. Наиболее часто используются отношения следующего вида

$$\alpha \geq \beta \quad \Leftrightarrow \quad -\alpha \leq -\beta$$

$$\alpha = \beta \quad \Leftrightarrow \quad \begin{cases} \alpha \geq \beta, \\ \alpha \leq \beta, \end{cases}$$

$$\alpha \geq \beta \quad \Leftrightarrow \quad \begin{cases} \alpha + \gamma = \beta, \\ \gamma \leq 0, \end{cases} \tag{4.1.1}$$

$$\max_x (c, x) = \min_x (-c, x).$$

Рассмотрим теперь стандартные постановки задач линейного программирования.

Первой канонической формой задачи ЛП, к которой может быть сведена любая задача линейного программирования, принято называть задачу:

Найти максимум $\sum_{j=1}^{n} \sigma_j \xi_j$ *на* $\{\xi_1, \xi_2, \ldots, \xi_n\} \in E^n$,

при условиях: $\xi_j \geq 0$, $j = [1, n]$,

$$\sum_{j=1}^{n} \alpha_{ij} \xi_j \leq \beta_i, \quad i = [1, m].$$

Под *второй канонической формой* задачи ЛП, понимают задачу:

Найти максимум $\sum_{j=1}^{n} \sigma_j \xi_j$ *на* $\{\xi_1, \xi_2, \ldots, \xi_n\} \in E^n$,

при условиях:

$$\xi_j \geq 0, \quad j = [1, n],$$

$$\sum_{j=1}^{n} \alpha_{ij} \xi_j = \beta_i, \quad i = [1, m].$$

Обычно вторая каноническая форма используется для задач ЛП таких, что $m < n$.

Достаточно часто используются *матричные формы* представления задач ЛП, например для первой канонической формы условие задачи ЛП можно записать в виде

Найти максимум $\| c \| \| x \|$ *на* $x \in E^n$,

при условиях:

$$\| x \| \geq \| o \|,$$
$$\| A \| \| x \| \leq \| b \|,$$

где каждое матричное неравенство определяется как система соответствующих покомпонентных числовых неравенств.

Отметим, что при любом соотношении между натуральными m и n множество R элементов, удовлетворяющих *всем ограничениям* задачи ЛП *одновременно* либо *пусто*, либо является *выпуклым многогранником* в E^n, границы которого принадлежат гиперплоскостям вида

$$\xi_j = 0,\ j = [1, n]; \quad \sum_{j=1}^{n} \alpha_{ij} \xi_j = \beta_i,\ i = [1, m].$$

Если множество R замкнуто и ограничено (то есть является компактом), тогда по теореме Вейерштрасса функционал $\sum_{j=1}^{n} \sigma_j \xi_j$ достигает своего экстремального значения. В случае неограниченного R конечное решение задачи ЛП может не существовать.

Будем обозначать решение задачи ЛП (если оно существует) как x^* с координатным представлением $\| x^* \| = \| \xi_1^* \quad \xi_2^* \quad \dots \quad \xi_n^* \|^{\mathrm{T}}$.

Определение 4.1.1. Принято говорить, что

- элемент $x^0 \in E^n$ *допустимый*, если на нем выполнены *все* ограничения задачи ЛП, то есть $x^0 \in R$;
- ограничение типа "неравенство" задачи ЛП на допустимом элементе $x^0 \in E^n$ называется *активным*, если на этом x^0 данное ограничение выполняется как равенство;

- ограниченный элемент x^* называется *решением*, если он удовлетворяет всем ограничениям, а целевой функционал имеет на нем экстремальное значение;
- ограниченное решение x^* задачи ЛП называется *переопределенным*, если число ограничений, активных на x^*, больше, чем размерность пространства E^n;
- задача ЛП *несовместна*, если множество R пусто (система линейных ограничений противоречива).

Раздел 4.2. УСЛОВИЯ ОПТИМАЛЬНОСТИ ДЛЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

§ 4.2.1. Прямые условия оптимальности

Рассмотрим условия оптимальности для задачи ЛП в следующей формулировке:

$$\begin{aligned} &\textit{найти элемент } x^* \in E^n, \\ &\textit{максимизирующий по } x \textit{ функционал } (c,x), \\ &\textit{при условиях } (a_i,x) \geq \beta_i \ , \forall i=[1,m], \end{aligned} \qquad (4.2.1.1)$$

считая, что линейные функционалы (c,x) и (a_i,x), $\forall i=[1,m]$ соответственно задаются ненулевыми элементами c и a_i , $\forall i=[1,m]$. принадлежащими E^{n+}.

Предположим вначале, что выпуклое и замкнутое множество $R \subseteq E^n$, заданное системой линейных неравенств вида

$$(a_j,x) \geq \beta_j \ ; \forall j=[1,m],$$

имеет *внутренние элементы*, то есть

$$\exists x^0 \in R : (a_j,x^0) > \beta_j \ ; \ \forall j=[1,m].$$

Данное условие часто называют *условием регулярности Слейтера*, оно предполагает существование в R элемента x^0, на котором все ограничения задачи ЛП неактивны. Кроме того, очевидно, что на x^0 конус допустимых вариаций $\delta x \in K(x^0)$ совпадает с E^n.

Из вышесказанного следует, что существует допустимая вариация $\delta x^0 \in K(x^0)$, которая одновременно является и улучшающей для целевого функционала (c,x), то есть x^0 не может быть решением данной задачи ЛП.

Действительно, в силу определения 3.2.3 из произвольности $\delta x^0 = x - x^0$ условие

$$F(x^0 + \tau\delta x) > F(x^0) \quad \forall \tau : 0 < \tau < \varepsilon$$

для рассматриваемого случая принимает вид $\tau(c, \delta x^0) > 0$ и в качестве улучшающей вариации можно, например, принять $\delta x^0 = \alpha \cdot c$, где α – некоторое положительное число.

Таким образом, доказана

Теорема 4.2.1.1 **Элемент x^*, являющийся решением задачи ЛП, принадлежит границе замкнутого выпуклого множества R.**

Для дальнейшего анализа нам потребуется формулировка теоремы 3.2.1 для задачи линейного программирования. Поскольку в этом случае $F(x) = (c,x)$, то $\operatorname{grad} F(x)$ не зависит от x и $\operatorname{grad} F(x) = c$. Следовательно, справедлива

Теорема 4.2.1.2. **Пусть функционал (c,x) на множестве R принимает максимальное значение на элементе x^*. Тогда**

$$(c, x - x^*) \le 0; \quad \forall x \in R.$$

Если рассматривается некоторый элемент $\bar{x}$, принадлежащий границе множества R, то конус $K(x^*)$ уже не будет совпадать с E^n.

Пусть $J(\bar{x})$ – множество из k индексов ограничений, активных на граничном элементе $\bar{x}$, тогда определение $\overline{K}(\bar{x})$ в терминах условия исходной задачи линейного программирования дает

Теорема 4.2.1.3. **Множество $\overline{K}(\bar{x})$ – допустимых на элементе $\bar{x}$ вариаций для задачи (4.2.1) – задается системой линейных неравенств вида**

$$\begin{cases} (a_{j_1}, \delta x) \geq 0, \\ (a_{j_2}, \delta x) \geq 0, \\ \quad \dots \\ (a_{j_k}, \delta x) \geq 0, \end{cases}$$

где $J^*(\bar{x}) \equiv \{i_1, i_2, \dots, i_k\}$ – множество индексов активных на элементе $\bar{x}$ ограничений.

Доказательство.

В §1.1 было показано, что каждое уравнение

$$(a_i, x - \bar{x}) = 0; \forall i \in J^*(\bar{x})$$

задает в E^n гиперплоскость, которой принадлежит элемент $\bar{x}$. Поэтому каждое условие $(a_i, x - \bar{x}) \geq 0, \forall i \in J^*(\bar{x})$ определяет в E^n полупространство E^n_{i+} (см. определение 1.1.5).

Тогда, по определению $J^*(\bar{x})$, существует окрестность $U_\varepsilon(\bar{x})$ такая, что множества

$$U_\varepsilon(\bar{x}) \cap R \quad \text{и} \quad U_\varepsilon(\bar{x}) \cap \Big(\bigcap_{i \in J^*(\bar{x})} E^n_{i+} \Big)$$

совпадают.

Но тогда совпадают $\overline{K}(\bar{x})$ и $\bigcap_{i\in J^*(\bar{x})} E^n_{i+}$. Поскольку последнее множество задается системой неравенств

$$\begin{cases} (a_{j_1}, x-\bar{x}) \geq 0, \\ (a_{j_2}, x-\bar{x}) \geq 0, \\ \cdots \\ (a_{j_k}, x-\bar{x}) \geq 0, \end{cases}$$

а $\delta x = x - \bar{x},$ то мы приходим к утверждению теоремы.

Теорема доказана.

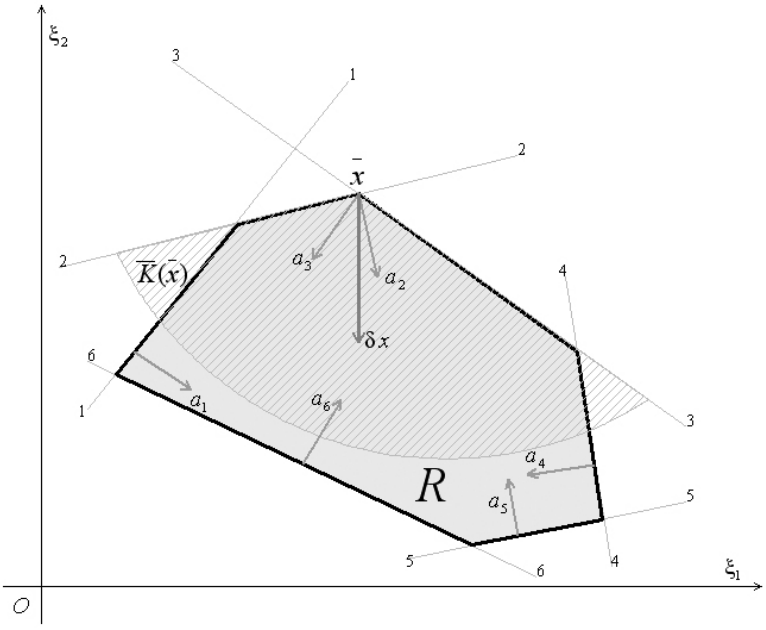


Рис. 4.2.1.1.

На рис.4.2.1.1 приведена иллюстрация, поясняющая утверждение теоремы 4.2.1.3. на примере задачи ЛП в E^2. В данном примере $n = 2$, а $m = 6$. Каждая из ограничивающих гиперплоскостей представляется отрезком прямой, значение индекса i которой указано у концов соответствующего отрезка. Элементы δx и $a_i, i = [1,6]$ представляются двухкомпонентными векторами на плоскости $\{O, \xi_1, \xi_2\}$. Ориентация вектора $a_i, i = [1,6]$ определяет положение полупространства E^2_{i+}.

Отметим, что для выбранного граничного элемента $\bar{x}$ активными являются ограничения с индексами 2 и 3. Остальные ограничения на этом элементе неактивны. Множество допустимых элементов R отмечено серым цветом, а конус допустимых направлений $\overline{K}(\bar{x})$ заштрихован.

Утверждение теоремы 4.2.1.1 геометрически означает, что любая допустимая вариация δx на элементе $\bar{x}$ является в E^2 вектором, образующим не тупой угол со всеми нормальными, ориентированными внутрь R, векторами *всех активных* на $\bar{x}$ ограничениях.

Основываясь на утверждениях теорем 4.2.1.2 и 4.2.1.3, приходим к заключению, что справедлива и

Теорема 4.2.1.4. **Функционал (c, x) на множестве R принимает максимальное значение на элементе x^* тогда и только тогда, когда $(c, \delta x) \le 0$ для каждого δx, удовлетворяющего системе линейных неравенств**

$$\begin{cases} (a_{j_1}, \delta x) \ge 0, \\ (a_{j_2}, \delta x) \ge 0, \\ \quad \dots \\ (a_{j_k}, \delta x) \ge 0, \end{cases}$$

где $J(x^*) \equiv \{j_1^*, j_2^*, \dots, j_k^*\}$ – множество индексов активных на элементе x^* ограничений.

Эта теорема дает *прямые необходимые и достаточные условия оптимальности* для рассматриваемой задачи линейного программирования.

§ 4.2.2. Двойственные условия оптимальности

Для задач линейного программирования как частного случая задач, рассмотренных в главе 3, необходимые и достаточные условия оптимальности могут быть получены также и в альтернативной, двойственной форме (Куна–Таккера), путем применения теоремы 1.1.4.4 (Фаркаша). Однако, специфическая форма постановки задачи ЛП позволяет получить эти условия в виде *другой задачи линейного программирования*, сформулированной в двойственном (сопряженном) пространстве.

Рассмотрим исходную задачу ЛП в упрощенной (без вписанных в явном виде условий $\xi_i \geq 0$, $i = [1, n]$) первой канонической форме:

$$\textit{Найти максимум} \sum_{i=1}^{n} \sigma_i \xi_i \quad \textit{на} \quad \{\xi_1, \xi_2, \ldots, \xi_n\} \in E^n,$$

$$\textit{при условиях}: \sum_{i=1}^{n} \alpha_{ji} \xi_i \leq \beta_j, \quad j = [1, m].$$

или же, в символическом виде

$$\begin{aligned} &\textit{Найти максимум} && (c, x) \quad \textit{на} \quad x \in E^n, \\ &\textit{при условиях:} && Ax \leq b. \end{aligned}$$

Функция Лагранжа для этой задачи будет иметь вид

$$L(x, \Lambda) = (c, x) + (\Lambda, b - Ax)$$

Напомним, что *прямая* задача в этом случае формулируется как

$$\max_{x} \min_{\Lambda \geq o} L(x, \Lambda),$$

а *двойственная* ей задача имеет вид $\min\limits_{\Lambda \geq o} \max\limits_{x} L(x, \Lambda)$.

Для того, чтобы получить стандартную формулировку двойственной задачи, преобразуем функцию Лагранжа к виду

$$L(x, \Lambda) = (\Lambda, b) + (c - A^{\mathrm{T}} \Lambda, x) \qquad (4.2.2.1)$$

В силу произвольности $x \in E^n$, из этой формулы (в предположении конечности Λ^*) следует, что,

$$\max_x L(x,\Lambda) = \begin{cases} (\Lambda, b), \text{ если } c - A^{\mathrm{T}}\Lambda = o, \\ +\infty, \quad \text{если } c - A^{\mathrm{T}}\Lambda \neq o. \end{cases}$$

а, значит, задача $\min\limits_{\Lambda \geq o} \max\limits_x L(x,\Lambda)$ равносильна задаче ЛП:

Найти минимум (Λ, b) *по* $\Lambda \in E^m$,

при условиях: $A^{\mathrm{T}}\Lambda = c$, $\Lambda \geq o$,

или, в координатной форме в евклидовом пространстве E^m (с элементами Λ) в виде: *минимизировать по* $\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$

линейный функционал $\sum\limits_{i=1}^{m} \beta_i \lambda_i$,

при условиях $\lambda_i \geq 0$, $\forall i = [1, m]$ и

$$\sum_{i=1}^{m} \alpha_{ij} \lambda_i = \sigma_i \quad \forall j = [1, n],$$

или же в матричной форме

минимизировать по $\| \Lambda \|$ *линейный функционал* $\| b \|^{\mathrm{T}} \| \Lambda \|$,

при условиях $\| \Lambda \| \geq \| o \|$ и $\| A \|^{\mathrm{T}} \| \Lambda \| = \| c \|$,

или, наконец, символически:

минимизировать по Λ *линейный функционал* (b, Λ),

при условиях $\Lambda \geq 0$ и

$$(\mathrm{A}_j, \Lambda) = \sigma_j \,;\, \forall j = [1, n],$$

где

$$\| \Lambda \| = \begin{Vmatrix} \lambda_1 \\ \lambda_2 \\ \cdots \\ \lambda_m \end{Vmatrix} \quad \text{и} \quad \mathrm{A}_j \in E^m, \quad \| \mathrm{A}_j \| = \| \alpha_{j1} \quad \alpha_{j2} \quad \ldots \quad \alpha_{jm} \|.$$

Формулу (4.2.2.1) можно использовать для построения двойственных задач и для других форм записи прямой задачи. В качестве примера рассмотрим следующую задачу ЛП:

$$\textit{найти максимум} \sum_{j=1}^{n} \sigma_j \xi_j \quad \textit{на} \quad \{\xi_1, \xi_2, \ldots, \xi_n\} \in E^n,$$

$$\textit{при условиях}: \qquad \xi_j \geq 0, \quad j = [1, n], \tag{4.2.2.2}$$

$$\sum_{j=1}^{n} \alpha_{ij} \xi_j \leq \beta_i, \quad i = [1, m]$$

Или же в символическом виде

$$\textit{Найти максимум} \qquad (c, x) \quad \textit{на} \quad x \in E^n,$$

$$\textit{при условиях:} \qquad x \geq o, \quad Ax \leq b.$$

Формула для функции Лагранжа в этом случае остается той же

$$L(x, \Lambda) = (\Lambda, b) + (c - A^{\mathrm{T}}\Lambda, x),$$

однако решение задачи $\max\limits_{x} L(x, \Lambda)$ в силу условия $x \geq 0$ имеет иной вид:

$$\max_{x} L(x, \Lambda) = \begin{cases} (\Lambda, b), \textit{ если } c - A^{\mathrm{T}}\Lambda \leq o, \\ +\infty, \quad \textit{если } c - A^{\mathrm{T}}\Lambda > o. \end{cases}$$

Поэтому двойственная задача $\min\limits_{\Lambda \geq o} \max\limits_{x} L(x, \Lambda)$ будет иметь следующую формулировку:

$$\textit{Найти минимум} \qquad (\Lambda, b) \quad \textit{по} \quad \Lambda \in E^m,$$

$$\textit{при условиях:} \qquad A^{\mathrm{T}}\Lambda \geq c, \quad \Lambda \geq o.$$

или в координатной форме:

$$\textit{найти минимум} \sum_{i=1}^{m} \beta_i \lambda_i \quad \textit{на} \quad \{\lambda_1, \lambda_2, \ldots, \lambda_n\} \in E^m,$$

$$\textit{при условиях}: \qquad \lambda_i \geq 0; \quad i = [1, m], \tag{4.2.2.3}$$

$$\sum_{i=1}^{m} \alpha_{ij} \lambda_i \geq \sigma_j \ , \quad j = [1, n].$$

Преимуществом записи прямой задачи линейного программирования (4.2.2.2) является *симметричность* формы записи двойственной ей задачи (4.2.2.3).

По аналогичной схеме можно получать формулировки двойственных задач и для других форм постановки прямой задачи. При этом стоит отметить, что переменные прямой задачи не входят в постановку двойственной только в *линейном* случае.

Примером, иллюстрирующим практическое значение использования двойственной задачи, может служить следующая модельная проблема.

Пусть имеется некоторый производственный процесс, в котором потребляются два вида ресурсов – *A* и *B*, и выпускаются два вида продукции – 1 и 2. Количественные характеристики этого процесса следующие:

1) на выпуск единицы продукции 1 расходуется α_{A1} единиц ресурса *A* и α_{B1} единиц ресурса *B*;
2) на выпуск единицы продукции 2 расходуется α_{A2} единиц ресурса *A* и α_{B2} единиц ресурса *B*;
3) объем доступного ресурса А составляет β_A единиц, а ресурса В – β_B единиц.

Наконец, предполагая, что цена продукции 1 составляет σ_1 за единицу, продукции 2 – σ_2 за единицу, требуется найти допустимые объемы выпуска продукции обоих видов, при которых стоимость реализации максимальна.

Если обозначить искомые объемы выпуска продукции как ξ_1 и ξ_2, то сформулированная задача сводится к следующей задаче линейного программирования:

$$\begin{aligned} &\textit{найти максимум } \sigma_1\xi_1 + \sigma_2\xi_2 \textit{ на } \{\xi_1, \xi_2\} \in E^2, \\ &\quad \textit{при условиях}: \\ &\qquad \xi_i \geq 0 \ , \quad j = [1,2], \\ &\qquad \alpha_{A1}\xi_1 + \alpha_{A2}\xi_2 \leq \beta_A, \\ &\qquad \alpha_{B1}\xi_1 + \alpha_{B2}\xi_2 \leq \beta_B. \end{aligned} \qquad (4.2.2.4)$$

Задача двойственная к данной и, согласно (4.2.2.3), записываемая как:

найти минимум $\beta_A\lambda_A + \beta_B\lambda_B$ *на* $\{\lambda_1, \lambda_2,\} \in E^2$,

при условиях: (4.2.2.5)

$$\lambda_i \geq 0, \quad j = [1,2],$$
$$\alpha_{A1}\lambda_A + \alpha_{B1}\lambda_B \geq \sigma_1,$$
$$\alpha_{A2}\lambda_A + \alpha_{B2}\lambda_B \geq \sigma_2,$$

может иметь следующую интерпретацию:

определить λ_A и λ_B – значения цен на ресурсы *A* и *B*, при которых минимизируются полные затраты на приобретение ресурсов в условиях бесприбыльного производства.

Действительно, в §4.3.2 будет показано, что, если ξ_1^* и ξ_2^* ограниченные оптимальные решения прямой задачи (4.2.2.4), то существуют ограниченные оптимальные решения λ_1^* и λ_2^* двойственной задачи, причем

$$\sigma_1\xi_1^* + \sigma_2\xi_2^* = \beta_1\lambda_1^* + \beta_2\lambda_2^* .$$

Раздел 4.3. ВЗАИМОДВОЙСТВЕННЫЕ ПАРЫ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

§ 4.3.1. Связь между условиями и решениями двойственной пары задач

Используя соотношения (4.1.1), условия как прямой, так и двойственной задач можно изменять, исходя из естественного стремления, как упростить запись постановки задач, так и облегчить исследование полученных решений.

В тех случаях, когда рассматривается пара, образованная прямой и двойственной задачами, часто оказывается удобным использовать симметричные формы записи их условий.

Например, следующую пару задач ЛП:

***задачу* (*P*)**:

найти максимум $\sum_{j=1}^{n} \sigma_j \xi_j$ *на* $\{\xi_1, \xi_2, \ldots, \xi_n\} \in E^n$,

при условиях:

$$\xi_j \geq 0\ ,\ j = [1, n], \qquad \sum_{j=1}^{n} \alpha_{ij} \xi_j \leq \beta_i\ ,\ j = [1, m]$$

и ***задачу* (*D*):**

найти минимум $\sum_{i=1}^{m} \beta_i \lambda_i$ *на* $\{\lambda_1, \lambda_2, \ldots, \lambda_n\} \in E^m$,

при условиях:

$$\lambda_i \geq 0\ ,\ i = [1, m], \qquad \sum_{i=1}^{m} \alpha_{ij} \lambda_i \geq \sigma_i\ ,\ i = [1, n].$$

В теории математического программирования пары задач ЛП {(*P*)–(*D*)} и {(*D*)–(*P*)} принято называть *взаимодвойственными*, поскольку задача, двойственная к двойственной, совпадает с прямой задачей.

Основные правила, связывающие условия прямой и двойственной задач таковы:

Если в условии прямой задачи,	**то в условии двойственной задачи**
Найти максимум	Найти минимум
Найти минимум	Найти максимум
Число переменных	Число ограничений
Число ограничений	Число переменных
j -й коэффициент целевого функционала	правая часть j -го неравенства
j -я неотрицательная переменная	j -е неравенство типа $\geq$
j -я неограниченная переменная	j -е равенство

j -й столбец в матрице ограничений	j -я строка в матрице ограничений
правая часть i -го неравенства	i -й коэффициент целевого функционала
i -я строка в матрице ограничений	i -й столбец в матрице ограничений
i -е неравенство типа $\leq$	i -я неотрицательная переменная
i -е равенство	i -я неограниченная переменная

§ 4.3.2. Теоремы двойственности в линейном программировании

Для каждой пары взаимодвойственных задач верна

Теорема 4.3.2.1 **Если** $\left\| x^* \right\| = \left\| \xi_1^* \quad \xi_2^* \quad \dots \quad \xi_n^* \right\|^{\mathrm{T}}$ **– оптимальное решение прямой задачи, а** $\left\| \Lambda^* \right\| = \left\| \lambda_1^* \quad \lambda_2^* \quad \dots \quad \lambda_n^* \right\|^{\mathrm{T}}$ **– оптимальное решение двойственной задачи, то справедливы равенства:**

1º. ***основное соотношение двойственности***

$$\sum_{j=1}^{n} \sigma_j \xi_j^* = \sum_{i=1}^{m} \beta_i \lambda_i^*;$$

2º. ***соотношения дополняющей нежесткости***

$$\lambda_i^* (-\beta_i + \sum_{j=1}^{n} \alpha_{ij} \xi_j^*) = 0 \; ; \; \forall i = [1, m],$$

$$\xi_j^* (-\sigma_j + \sum_{i=1}^{m} \alpha_{ij} \lambda_i^*) = 0 \; ; \; \forall j = [1, n].$$

Доказательство.

1º. Докажем вначале *основное соотношение двойственности*. Рассмотрим пару допустимых ограниченных решений x – прямой и y – двойственной задач. Тогда из соотношений

$$\xi_j \geq 0\,,\ j=[1,n];\quad \sum_{j=1}^{n}\alpha_{ij}\xi_j \leq \beta_i\,,\ i=[1,m];$$

и

$$\lambda_i \geq 0\,,\ i=[1,m];\quad \sum_{i=1}^{m}\alpha_{ij}\lambda_i \geq \sigma_j\,,\ j=[1,n]$$

имеем, что

$$\lambda_i \sum_{j=1}^{n}\alpha_{ij}\xi_i \leq \beta_i\lambda_i\,,\quad i=[1,m]$$

и

$$\xi_j \sum_{j=1}^{m}\alpha_{ij}\lambda_i \geq \sigma_j\xi_j\,,\quad j=[1,n].$$

Просуммировав полученные неравенства, первое по $i=[1,m]$, а второе по $j=[1,n]$, получим оценки

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_{ij}\xi_j\lambda_i &\leq \sum_{i=1}^{m}\beta_i\lambda_i,\\ \sum_{j=1}^{n}\sum_{i=1}^{m}\alpha_{ij}\lambda_i\xi_j &\geq \sum_{j=1}^{n}\sigma_j\xi_j\end{aligned} \qquad (4.3.2.1)$$

и, поскольку левые части этих неравенств одинаковы, то

$$\sum_{j=1}^{n}\sigma_j\xi_j \leq \sum_{i=1}^{m}\beta_i\lambda_i$$

на любой допустимой ограниченной паре решений, в том числе и оптимальной

$$\sum_{j=1}^{n}\sigma_j\xi_j^* \leq \sum_{i=1}^{m}\beta_i\lambda_i^*\,.$$

Заметим, что эта оценка является частным случаем теоремы 3.5.2.2, хотя она и получена иным способом. Кроме того, согласно теореме 3.5.2.3, для ограниченных оптимальных решений двойственной пары *линейных* (а, значит, выпуклых) задач справедлива оценка

$$\sum_{j=1}^{n}\sigma_j\xi_j^* \geq \sum_{i=1}^{m}\beta_i\lambda_i^* ,$$

что, в сочетании с $\sum_{j=1}^{n}\sigma_j\xi_j^* \leq \sum_{i=1}^{m}\beta_i\lambda_i^*$, означает выполнение равенства $\sum_{j=1}^{n}\sigma_j\xi_j^* = \sum_{i=1}^{m}\beta_i\lambda_i^*$.

2º. Покажем теперь справедливость *соотношений дополняющей нежесткости*.

Рассмотрим пару оптимальных решений x^* – прямой и y^* – двойственной задач. Тогда, в силу условий задач ЛП (P–D) справедливы следующие неравенства

$$\sum_{i=1}^{m}\lambda_i^*(-\beta_i + \sum_{j=1}^{n}\alpha_{ij}\xi_j^*) \leq 0 ;$$

$$\sum_{j=1}^{n}\xi_j^*(-\sigma_j + \sum_{i=1}^{m}\alpha_{ij}\lambda_i^*) \geq 0 ;$$

но в силу $\sum_{j=1}^{n}\sigma_j\xi_j^* = \sum_{i=1}^{m}\beta_i\lambda_i^*$ левые части этих неравенств равны, и потому возможен лишь случай, когда

$$\begin{aligned}\sum_{i=1}^{m}\lambda_i^*(-\beta_i + \sum_{j=1}^{n}\alpha_{ij}\xi_j^*) &= 0 ;\\ \sum_{j=1}^{n}\xi_j^*(-\sigma_j + \sum_{i=1}^{m}\alpha_{ij}\lambda_i^*) &= 0 ;\end{aligned} \qquad (4.3.2.2)$$

С другой стороны, из неравенств

$$\xi_j^* \geq 0\ ,\ j=[1,n]\ ; \qquad \sum_{j=1}^{n} \alpha_{ij}\xi_j^* \leq \beta_i\ ,\ i=[1,m];$$

и

$$\lambda_i^* \geq 0\ ,\ i=[1,m]\ ; \qquad \sum_{i=1}^{m} \alpha_{ij}\lambda_i^* \geq \sigma_j\ ,\ j=[1,n]$$

имеем, что все слагаемые левой части первого из равенств (4.3.2.2) неположительны, а все слагаемые левой части второго – неотрицательны. Тогда очевидно, что из равенств (4.3.2.2) следует справедливость соотношений

$$\lambda_i^*(-\beta_i + \sum_{j=1}^{n} \alpha_{ij}\xi_j^*) = 0\ ,\ \forall i=[1,m];$$

$$\xi_j^*(-\sigma_j + \sum_{i=1}^{m} \alpha_{ij}\lambda_i^*) = 0\ ,\ \forall j=[1,n].$$

Теорема доказана.

§ 4.3.3. Условия разрешимости пары взаимодвойственных задач ЛП

Основное соотношение двойственности и *условия дополняющей нежесткости* являются базовыми свойствами взаимодвойственной пары задач. В данном параграфе рассмотрим (в основном, без доказательств и с краткими комментариями) альтернативные формы зависимостей между решениями двойственной пары задач, формулируемые в виде следующих теорем.

Теорема 4.3.3.1. **Если допустимое множество одной из взаимодвойственных задач не пусто и ее целевой функционал ограничен на этом множестве, то данная задача имеет решение.**

Отметим, что теорема 4.3.3.1 справедлива лишь для задач ЛП. Проверьте самостоятельно, что, например, для *нелинейной* задачи

найти максимум ξ_2 *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях: $\xi_1 \geq 0$, $\xi_1 \xi_2 \leq -1$,

эта теорема не верна.

Теорема 4.3.3.2. **Если x^* и Λ^* допустимые элементы пары взаимодвойственных задач такие, что**

$$\sum_{j=1}^{n} \sigma_j \xi_j^* = \sum_{i=1}^{m} \beta_i \lambda_i^* ,$$

то x^* и Λ^* – решения этих задач.

Доказательство.

Поскольку $\sum_{j=1}^{n} \sigma_j \xi_j \leq \sum_{i=1}^{m} \beta_i \lambda_i$ для *любых* допустимых x и Λ, то тогда

$$\sum_{j=1}^{n} \sigma_j \xi_j \leq \sum_{i=1}^{m} \beta_i \lambda_i^* = \sum_{j=1}^{n} \sigma_j \xi_j^* .$$

что и означает, что x^* – решение прямой задачи.

Для Λ^* рассуждения аналогичные.

Теорема доказана.

Теорема 4.3.3.3. **Если обе взаимодвойственные задачи имеют непустые допустимые множества, то они обе имеют решение с равными оптимальными значениями целевых функций.**

Теорема 4.3.3.4. **Для существования конечных решений у пары взаимодвойственных задач необходимо и достаточно, чтобы была совместностна система неравенств:**

$$Ax \leq b , \quad A^{\mathrm{T}} \Lambda \geq c, \quad x \geq o, \quad \Lambda \geq o,$$

$$(c, x) \geq (b, \Lambda) \quad .$$

Следующая теорема (часто называемой в литературе *теоремой равновесия*) дает альтернативную формулировку условиям дополняющей нежесткости.

Теорема 4.3.3.5. **Если x^* и Λ^* оптимальные элементы пары взаимодвойственных задач, тогда**

из условия $\lambda_i^* > 0$ **следует** $(-\beta_i + \sum_{j=1}^{n} \alpha_{ij}\xi_j^*) = 0$,

а из $(-\beta_i + \sum_{j=1}^{n} \alpha_{ij}\xi_j^*) > 0 \rightarrow \lambda_i^* = 0, \forall i = [1, m]$.

Заметим, что утверждение, обратное утверждению теоремы 4.3.3.5, не верно: если $(-\beta_i + \sum_{j=1}^{n} \alpha_{ij}\xi_j^*) = 0$, то значение λ_i^* может быть нулевым. Аналогично, если $\lambda_i^* = 0$, то возможно что

$$(-\beta_i + \sum_{j=1}^{n} \alpha_{ij}\xi_j^*) = 0.$$

Теорема 4.3.3.6. **Если одна из взаимодвойственных задач имеет решение, то имеет решение и другая.**

Теорема 4.3.3.6 фактически утверждает, что для пары взаимодвойственных задач имеет место следующая (подтверждаемая примерами) альтернатива:

а) обе задачи имеют решение:

Найти максимум $F = 2\xi_1 + 3\xi_2$ *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях:

$$\xi_i \geq 0, \quad j = [1, 2],$$
$$\xi_1 + 2\xi_2 \leq 6,$$
$$2\xi_1 + \xi_2 \leq 6$$

с решением $\xi_1^* = 2, \xi_2^* = 2, F^* = 10$ и

Найти минимум $G = 6\lambda_1 + 6\lambda_2$ *на* $\{\lambda_1, \lambda_2\} \in E^2$,

при условиях:

$$\lambda_i \geq 0, \quad j = [1,2],$$
$$\lambda_1 + 2\lambda_2 \geq 2,$$
$$2\lambda_1 + \lambda_2 \geq 3$$

с решением $\lambda_1^* = \frac{4}{3}, \lambda_2^* = \frac{1}{3}, G^* = 10.$

б) обе задачи несовместны:

Найти максимум $F = \xi_1 + 3\xi_2$ *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях:

$$\xi_i \geq 0, \quad j = [1,2],$$
$$\xi_1 - \xi_2 \leq 3,$$
$$-\xi_1 + \xi_2 \leq -4.$$

и

Найти минимум $G = 3\lambda_1 - 4\lambda_2$ *на* $\{\lambda_1, \lambda_2\} \in E^2$,

при условиях:

$$\lambda_i \geq 0, \quad j = [1,2],$$
$$\lambda_1 - \lambda_2 \geq 1,$$
$$-\lambda_1 + \lambda_2 \geq 3.$$

в) одна задача совместна, а другая – нет:

Найти максимум $F = \xi_1$ *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях:

$$\xi_i \geq 0, \quad j = [1,2],$$
$$\xi_1 - \xi_2 \leq 1,$$

с неограниченым целевым функционалом на множестве допустимых состояний и

Найти минимум $G = \lambda_1$ *на* $\{\lambda_1\} \in E^1$,

при условиях:

$$\lambda_1 \geq 0\,,$$
$$\lambda_1 \geq 1,$$
$$-\lambda_1 \geq 0,$$

которая несовместна.

Обратите внимание, что согласно определению 4.1.1 первая из задач в пункте в) решений не имеет. Этот факт обобщает

Теорема 4.3.3.7. **Если одна из взаимодвойственных задач *недопустима*, а другая совместна, то целевая функция второй задачи *неограничена* на множестве ее допустимых элементов.**

§ 4.3.4. Единственность и переопределенность решений взаимодвойственных задач ЛП

В теоремах 4.3.3.1-4.3.3.7 рассматривался вопрос о *существовании* решений у пары взаимодвойственных задач. Их утверждения справедливы как случаев единственного, так и неединственного решения задачи ЛП. Следующие теоремы позволяют делать заключение о числе этих решений.

Теорема 4.3.4.1. **Если одна из взаимодвойственных задач имеет *единственное, непереопределенное* решение, то другая также имеет *единственное* решение.**

Напомним, что решение задачи ЛП называется переопределенным, если число активных на этом решении ограничений больше размерности пространства.

Теорема 4.3.4.2. **Если одна из взаимодвойственных задач имеет *единственное, переопределенное* решение, то другая задача имеет *неединственное* решение.**

Отметим также, что возможен случай, когда обе задачи взаимодвойственной пары имеют неединственное решение. Проиллюстрируем два последних утверждения примерами.

а) прямая задача переопределена, а двойственная имеет неединственное решение:

Найти максимум $F = 2\xi_1 + 3\xi_2$ *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях:

$$\xi_i \geq 0, \quad j = [1,2],$$
$$\xi_1 + 2\xi_2 \leq 6,$$
$$2\xi_1 + \xi_2 \leq 3$$

с решением $\xi_1^* = 0, \xi_2^* = 3, F^* = 9$

В этом случае на элементе x^* активными являются ограничения

$$\xi_1 \geq 0,$$
$$\xi_1 + 2\xi_2 \leq 6,$$
$$2\xi_1 + \xi_2 \leq 3.$$

а, поскольку их число 3>dim(E^2)=2, то это решение переопределенное,

и двойственная задача

Найти минимум $G = 6\lambda_1 + 3\lambda_2$ *на* $\{\lambda_1, \lambda_2\} \in E^2$,

при условиях:

$$\lambda_i \geq 0, \quad j = [1,2],$$
$$\lambda_1 + 2\lambda_2 \geq 2,$$
$$2\lambda_1 + \lambda_2 \geq 3$$

с решением $\lambda_1^* = t; t \in [0, \frac{4}{3}]; \quad \lambda_2^* = 3 - 2t; \quad G^* = 9.$

б) обе задачи взаимодвойственной пары имеют неединственное решение:

Найти максимум $F = \xi_1 + 2\xi_2$ *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях:

$$\xi_i \geq 0, \quad j = [1,2],$$
$$2\xi_1 + 4\xi_2 \leq 4,$$
$$3\xi_1 + 6\xi_2 \leq 6,$$

с решением $\xi_1^* = t; t \in [0,2]; \quad \xi_2^* = 1 - \frac{1}{2}t; \quad F^* = 2.$

и двойственная задача

Найти минимум $G = 4\lambda_1 + 6\lambda_2$ *на* $\{\lambda_1, \lambda_2\} \in E^2$,
при условиях:

$$\lambda_i \geq 0, \quad j = [1,2],$$
$$2\lambda_1 + 3\lambda_2 \geq 1,$$
$$4\lambda_1 + 6\lambda_2 \geq 2,$$

с решением $\lambda_1^* = t; t \in [0, \frac{1}{2}]; \quad \lambda_2^* = \frac{1}{3} - \frac{2}{3}t; \quad G^* = 2.$

Раздел 4.4. ФУНКЦИОНАЛЬНЫЕ СВОЙСТВА РЕШЕНИЙ ЗАДАЧ ЛП

Подробное обсуждение проблемы зависимости решения задач математического программирования от параметров будет проведено в §5.2. Здесь же мы рассмотрим лишь зависимость оптимального значения целевого функционала прямой задачи от векторного параметра b, компонентами которого являются правые части ограничений-неравенств прямой задачи.

Основные свойства функции $F(b) = (c, x^*(b))$ описывает

Теорема 4.4.1. **Функция $F(b) = (c, x^*(b))$ в своей области определения**

- **положительно однородна первой степени,**

  то есть

  $$F(\mu b) = \mu F(b), \forall \mu \geq 0;$$

- **выпукла вверх;**
- **непрерывна.**

Доказательство.

1º. Вначале докажем однородность функции $F(b)$.

Рассмотрим пару *новых* взаимодвойственных задач

Найти максимум (c,x) *на* $x \in E^n$,

при условиях: $x \geq o$, $Ax \leq \mu b$.

и

Найти минимум $(\mu b, \Lambda)$ *по* $\Lambda \in E^m$,

при условиях: $A^{\mathrm{T}}\Lambda \geq c$, $\Lambda \geq o$.

для некоторого $\mu \geq 0$.

Если x^* и Λ^* решения *исходной* пары задач, то μx^* допустимый элемент новой прямой задачи, а Λ^* допустимый элемент новой двойственной. Покажем, что элемент μx^* является также оптимальным для новой прямой задачи.

Действительно, из основного соотношения двойственности $(c,x^*) = (b,\Lambda^*)$ следует равенство

$$(c,\mu x^*) = (\mu b, \Lambda^*).$$

Но тогда по теореме 4.3.3.2 μx^* – решение новой прямой задачи и

$$F(\mu b) = (c,\mu x^*) = \mu(c,x^*) = \mu F(b),\ \forall \mu \geq 0$$

2º. Покажем теперь выпуклость вверх функции $F(b)$.

Пусть для b^1 и b^2 исходная прямая задача совместна и имеет решения x^{*1} и x^{*2}. Рассмотрим элемент

$$\tilde{x} = \mu x^{*1} + (1-\mu)x^{*2}, \quad \mu \in [0,1].$$

Он является допустимым для исходной прямой задачи, в которой

$$b = \mu b^1 + (1-\mu)b^2,$$

поскольку

$$A\tilde{x} = \mu A x^{*1} + (1-\mu)Ax^{*2} \leq \mu b^1 + (1-\mu)b^2.$$

Тогда окончательно получаем

$$F(\mu b^1 + (1-\mu)b^2) \geq (c, \bar{x}) = \mu(c, x^{*1}) + (1-\mu)(c, x^{*2}) =$$
$$= \mu F(b^1) + (1-\mu)F(b^2),$$

что означает выпуклость вверх функции $F(b)$.

3º. Область определения функции $F(b)$, как нетрудно убедиться, является выпуклым множеством. Поэтому, в силу теоремы 1.2.2.1 приходим к заключению о непрерывности функции $F(b)$.

Теорема доказана.

Теперь приведем некоторые сведения о дифференциальных свойствах функции $F(b)$. Во-первых, справедлива

Теорема 4.4.2. **Если решение двойственной задачи единственно, то функция $F(b)$ дифференцируема и**

$$\operatorname{grad} F(b) = \Lambda^*.$$

Доказательство.

Из теоремы 4.4.1. следует, в достаточно малой окрестности элемента b, при котором x^* и λ^* – решения взаимодвойственных задач, единственны, функция $F(b)$ линейна.

Из основного соотношения двойственности следует

$$F(b) = (c, x^*) = (b, \Lambda^*).$$

Поэтому тейлоровская аппроксимация $F(b)$ будет иметь вид

$$F(b + \Delta b) = F(b) + (\Lambda^*, \Delta b),$$

что, в силу единственности градиента, означает $\operatorname{grad} F(b) = \Lambda^*$.

Теорема доказана.

Без доказательства отметим, что в случае, когда решение двойственной задачи не единственно, справедлива

Теорема 4.4.3. **Пусть** $s \in E^m$ **– направляющий элемент с** $|s| = 1$, **тогда производная функции** $F(b)$ **по направлению** s **определяется по формуле** $\dfrac{\partial F}{\partial s} = \min\limits_{\lambda \in \Lambda^*}(s, \lambda)$, **где** Λ^* **– множество решений двойственной задачи.**

Свойствами, аналогичными свойствам функции $F(b)$, обладает и функция $E(c) = (c, x^*(c))$. В частности, она выпукла вниз и для нее (в случае единственности решения взаимодвойственной пары задач) $\operatorname{grad} E(c) = x^*$.

Раздел 4.5. МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ЛП

§ 4.5.1. Метод исключения

Существуют различные методы решения задачи ЛП. Например, для задачи во второй канонической форме применим следующий подход.

Найдем в матрице $\|A\|$ невырожденную квадратную подматрицу $\|A^*\|$ порядка r такую, что $r = \operatorname{rg}\|A^*\|$.

Без ограничения общности можно считать, что $\|A^*\|$ расположена в левом верхнем углу матрицы $\|A\|$. Тогда первые r компонент координатного представления элемента $\|x\|$, обозначенные как $\|x'\|$, можно выразить через $\|x''\|$ – оставшиеся $n - r$ компонентов.

Обозначим через $\|A^{**}\|$ матрицу размера $r \times (n - r)$, расположенную в правом верхнем углу $\|A\|$. Полученные зависимости имеют вид

$$\| x' \| = \| A^* \|^{-1} (\| b \| - \| A^{**} \| \| x'' \|)$$

и очевидно линейны по своим аргументам.

Подставляя выражения для компонент $\| x' \|$ через компоненты $\| x'' \|$ в функционал $\| c \|^{\mathrm{T}} \| x \|$, приходим к существенно более простой (по сравнению с исходной) задаче отыскания экстремума линейного функционала $\| c^* \|^{\mathrm{T}} \| x'' \|$ по $\| x'' \|$ в положительном квадранте евклидова пространства E^{n-k} и условии $\| x' \| \geq \| o \|$.

Проиллюстрируем применение описанной схемы для следующей задачи линейного программирования:

Найти максимум $2\xi_1 + 3\xi_2$ *на* $\{\xi_1, \xi_2\} \in E^2$,

при условиях:

$$\xi_i \geq 0, \quad j = [1,2],$$
$$\xi_1 + 2\xi_2 \leq 6,$$
$$2\xi_1 + \xi_2 \leq 6.$$

Приведем условие этой задачи к каноническому виду включением в условие двух дополнительных неотрицательных компонент ξ_3 и ξ_4:

Найти максимум $2\xi_1 + 3\xi_2$ *на* $\{\xi_1, \xi_2, \xi_3, \xi_4\} \in E^4$,

при условиях:

$$\xi_i \geq 0, \quad j = [1,4],$$
$$\xi_1 + 2\xi_2 + \xi_3 = 6,$$
$$2\xi_1 + \xi_2 + \xi_4 = 6.$$

Пусть $\| x' \| = \| \xi_1 \quad \xi_2 \|^{\mathrm{T}}$, тогда в силу двух последних равенств и условия неотрицательности переменных,

$$\xi_1(\xi_3, \xi_4) = 2 + \frac{1}{3}\xi_3 - \frac{2}{3}\xi_4 \geq 0 \text{ и}$$

$$\xi_2(\xi_3, \xi_4) = 2 - \frac{2}{3}\xi_3 + \frac{1}{3}\xi_4 \geq 0.$$

Что в свою очередь приводит к выражению для функционала

$$2\xi_1 + 3\xi_2 = 10 - \frac{4}{3}\xi_3 - \frac{1}{3}\xi_4,$$

из которого в силу неотрицательности ξ_3 и ξ_4 получаем, что максимальное значение функционала равно 10 на элементе $\left\| x^* \right\| = \left\| 2 \quad 2 \right\|^{\mathrm{T}}$.

Хотя описанный алгоритм принципиально применим для любой задачи ЛП, его практическое использование в значительной степени ограничено процедурой контроля знаков компонент $\left\| x' \right\|$.

Поясним сказанное следующим примером. Попытаемся применить схему исключения в задаче такого вида.

Найти максимум $2\xi_1 - 3\xi_2$ *на* $\{\xi_1, \xi_2\} \in E^2$,

$\xi_i \geq 0, \quad j = [1,2],$

при условиях: $\xi_1 + 2\xi_2 \leq 6,$

$2\xi_1 + \xi_2 \leq 6.$

Приведем условие этой задачи к каноническому виду включением в условие двух дополнительных неотрицательных компонент ξ_3 и ξ_4:

Найти максимум $2\xi_1 - 3\xi_2$ *на* $\{\xi_1, \xi_2, \xi_3, \xi_4\} \in E^4$,

при условиях: $\xi_i \geq 0, \quad j = [1,4],$

$\xi_1 + 2\xi_2 + \xi_3 = 6,$

$2\xi_1 + \xi_2 + \xi_4 = 6.$

Пусть снова $\left\| x' \right\| = \left\| \xi_1 \quad \xi_2 \right\|^{\mathrm{T}}$, тогда мы имеем

$$\xi_1(\xi_3, \xi_4) = 2 + \frac{1}{3}\xi_3 - \frac{2}{3}\xi_4 \geq 0 \text{ и } \xi_2(\xi_3, \xi_4) = 2 - \frac{2}{3}\xi_3 + \frac{1}{3}\xi_4 \geq 0.$$

Это приводит к иному выражению для целевого функционала

$$2\xi_1 - 3\xi_2 = -2 + \frac{8}{3}\xi_3 - \frac{7}{3}\xi_4.$$

Из этой формулы, следует, что, в силу неотрицательности ξ_4, оптимальное значение ξ_4 следует выбрать нулевым, а вот значение ξ_3 нужно постараться сделать как можно большим.

Из равенств

$$\xi_1(\xi_3,\xi_4)=2+\frac{1}{3}\xi_3-\frac{2}{3}\xi_4 \quad \text{и} \quad \xi_2(\xi_3,\xi_4)=2-\frac{2}{3}\xi_3+\frac{1}{3}\xi_4$$

вытекает, что с ростом ξ_3 значение ξ_1 неограниченно возрастает, а значение ξ_2 убывает, но не может стать отрицательным числом. Поэтому (учитывая, что $\xi_4=0$) мы приходим к следующему условию, определяющему максимально допустимую величину ξ_3

$$\xi_2(\xi_3,\xi_4)=2-\frac{2}{3}\xi_3\geq 0.$$

Откуда $\xi_3\leq 3$. И окончательно мы приходим к заключению, что координаты оптимального элемента в E^4 для задачи и канонической форме имеют вид

$$\xi_1^*=3;\quad \xi_2^*=0;\quad \xi_3^*=3;\quad \xi_4^*=0.$$

Следовательно, максимальное значение целевого функционала для исходной задачи оказывается равным 6 на элементе $\|x^*\|=\|3\quad 0\|^{\mathrm{T}}$.

Если, в отличие от рассмотренного выше примера, процедура контроля знаков сложна, то вместо схемы исключения в вычислительной практике для решения задач линейного программирования используется специальный, описываемый ниже, метод.

§ 4.5.2. Симплекс-метод

Основным алгоритмом, используемым для практического решения задач ЛП большой размерности во второй канонической форме:

найти максимум $\|c\|^{\mathrm{T}}\|x\|$ *на* $x\in E^n$,

при условиях:

$$\|x\|\geq\|o\|,$$

$$\|A\|\|x\|=\|b\|,$$

является так называемый *симплекс-метод*.

Симплекс-метод представляет собой некоторый специальный вариант метода исключения, идея которого заключается в следующем.

Пусть решение задачи ЛП существует и конечно. Выберем из n компонент координатного представления элемента

$$\|x\| = \|\xi_1 \quad \xi_2 \quad \dots \quad \xi_n\|^{\mathrm{T}}$$

некоторые m (называемые *базисными*) и найдем их значения при *нулевых* значениях остальных (называемых *небазисными*) компонент $\|x\|$ из системы

$$\begin{cases} \xi_j \geq 0, \quad j = [1, n], \\ \sum_{j=1}^{n} \alpha_{ij} \xi_j = \beta_i, \quad i = [1, m]. \end{cases}$$

1º. Оцениваем числовые характеристики линейной зависимости целевого функционала $\sum_{j=1}^{n} \sigma_j \xi_j$ от базисных и небазисных компонент.

2º. На основании полученных оценок выясняем:

1) За счет увеличения на единицу какой из небазисных компонент возможно максимальное приращение целевого функционала?

2) Какая из базисных компонент в процессе увеличения значения выбранной небазисной компоненты первой достигнет нулевой границы?

3º. Вводим первую из этих двух компонент в базис, а вторую выводим из него.

Процесс прерывается, если не находится небазисной компоненты, увеличивающей функционал, иначе повторяем шаги 1º.–2º.–3º.

На практике используются различные модификации симплекс-метода, позволяющие не только находить экстремумы в задачах ЛП, но и определять, например, значения двойственных переменных, устанавливать факт как отсутствия решения (несовместности), так и существования неограниченного решения этих задач.

Глава 5
ЗАДАЧИ, СВОДЯЩИЕСЯ К ЗАДАЧАМ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Раздел 5.1. ЗАДАЧИ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

Термин "задача оптимального управления" используется для обозначения специального класса задач поиска условного экстремума, возникающих при исследовании объектов или процессов, обладающих следующими свойствами:

- периодичностью[6] их математического описания;
- возможностью разбиения набора количественных показателей, описывающих каждый период, на два подмножества:
    - показателей состояния (или фазовых переменных);
    - показателей управляющих воздействий (или управлений).

  причем состояние объекта или процесса для некоторого периода зависит от состояний для других периодов, в то время как управление для каждого из периодов осуществляется независимо.

В тех случаях, когда под периодом понимается некоторый момент (или промежуток) времени, задачи данного класса принято называть "динамическими".

Наконец, в зависимости от того является ли множество периодов счетным или нет, используется уточняющее класс задачи определение − "дискретная" или "непрерывная" задача.

---

[6] Слово "периодичность" предполагает повторяемость элементов модели как во времени, так и в пространстве.

§ 5.1.1. Дискретные динамические задачи

Рассмотрим N -периодическую модель, описываемую для каждого из периодов $k=[1,N]$ совокупностью $x(k)$ – n -компонентных *векторов состояния* и $u(k)$ – r -компонентных *векторов управления*, удовлетворяющих как системе динамических связей

$$x_i(k+1)=f_{ik}(x(k),u(k))\,;\;\; i=[1,n],\, k=[1,N-1], \quad (5.1.1.1)$$

так и системе смешанных ограничений вида

$$g_{jk}(x(k),u(k))\ge 0\,;\; j=[1,m]\,;k=[1,N]\,. \quad (5.1.1.2)$$

Для этой модели возможна постановка задачи *оптимального управления*, заключающейся в определении натурального N^*,

$$\text{а также } x^*(k)\in E^n \quad \text{и} \quad u^*(k)\in E^r \;\; (\forall k=[1,N]),$$

доставляющих экстремум функционалу

$$\Phi(x(1),\ldots,x(N),u(1),\ldots,u(N),N) \quad (5.1.1.3)$$

где функции

$$\Phi(x(1),\ldots,x(N),\,u(1),\ldots,u(N),\,N)\,,$$
$$f_{ik}(x(k),u(k)) \quad \text{и} \quad g_{jk}(x(k),u(k))$$

непрерывно дифференцируемы по всем своим аргументам.

Формально задача (5.1.1.1)-(5.1.1.3) является задачей *математического программирования*, методы решения которой (как было отмечено в разделе 3.6) существенно зависят от вида функций входящих в ее условие. Здесь же мы рассмотрим специальный подкласс задач этого вида, называемых задачами *быстродействия* для *линейных* дискретных систем и имеющих следующую формулировку:

> *найти минимальное число периодов* – натуральное N и соответствующие *множества векторов состояний и управлений*, то есть $x(k)\in E^n$ и $u(k)\in E^r, k=[1,N]$, которые удовлетворяют как системе динамических связей

$$x(k+1) = A(k)x(k) + B(k)u(k) + c(k), \quad \forall k = [1, N-1] \tag{5.1.1.4}$$

так и системе смешанных ограничений

$$D_j(k)x(k) + E_j(k)u(k) + g_j(k) \geq 0, \quad \forall k = [1, N], \ \forall j = [1, m] \tag{5.1.1.5}$$

Метод решения задачи (5.1.1.4)-(5.1.1.5), рассматриваемый ниже, основан на использовании свойства ее линейности по $x(k)$ и $u(k)$, $k = [1, N]$ при фиксированном числе периодов N.

Если оказывается, что для некоторого N система (5.1.1.4)-(5.1.1.5) совместна, то значение пробного числа периодов уменьшается. Если же для данного N устанавливается факт несовместности системы (5.1.1.4)-(5.1.1.5), то значение N увеличивается и анализ совместности повторяется для нового значения N.

Процесс заканчивается, когда фиксируются два последовательных натуральных числа $N^* - 1$ и N^*, таких, что система (5.1.1.4)-(5.1.1.5) несовместна для $N^* - 1$ и совместна для N^*. То есть искомое минимальное N^* найдено.

В данной схеме оказываются существенными два следующих момента.

Во-первых, она предполагает, что пользователь может найти хотя бы одно значение N, при котором система (5.1.1.4)-(5.1.1.5) совместна. Вообще говоря, такое N^* может и не существовать. Например, система условий (5.1.1.5) уже сама по себе может оказаться противоречивой для любого N. Тогда рассматриваемая задача быстродействия решения не имеет. Достаточным условием существования ее решения является, например, выполнение неравенства $r \geq m^*$, где m^* – число активных ограничений системы (5.1.1.5) на оптимальном решении задачи (5.1.1.4)-(5.1.1.5).

Во-вторых, необходимо сформулировать критерий, позволяющий однозначно различать состояния совместности или несовместности задачи (5.1.1.4)-(5.1.1.5).

Для этого возможно, например, использование вспомогательной неотрицательной переменной κ, равной максимуму нарушения ограничений, входящих в систему (5.1.1.5). В этом случае задача нижнего уровня сводится к следующей задаче линейного программирования:

минимизировать κ, *при условиях* $\kappa \geq 0,$

$$x(k+1) = A(k)x(k) + B(k)u(k) + c(k), \quad \forall k = [1, N-1]$$

$$D_j(k)x(k) + E_j(k)u(k) + g_j(k) + \kappa \geq 0, \; \forall k = [1, N], \; \forall j = [1, m], \tag{5.1.1.6}$$

достижение нулевого оптимального κ^* решения которой, означает совместность задачи (5.1.1.4)-(5.1.1.5) при фиксированном N. Если же оказывается, что $\kappa^* > 0$, то это означает несовместность задачи (5.1.1.4)-(5.1.1.5).

Решение задачи поиска N^* (в случае установления существования верхней оценки для N, гарантирующего совместность) можно выполнять, используя дискретные аналоги схем дихотомии или "золотого сечения". Отметим также, что практический интерес представляет параметрический анализ решения задачи (5.1.1.4)-(5.1.1.5), то есть исследование зависимости минимального N^* от значений параметров, входящих в ее постановку.

§ 5.1.2. Непрерывные динамические задачи

Рассмотрим динамический (то есть, способный меняться с течением времени) объект, *состояние* которого в некоторый фиксированный момент времени $t \in [t_0, T]$ полностью и однозначно описывается упорядоченным набором чисел

$$\{ x_1(t), x_2(t), \ldots, x_n(t) \} \equiv x(t),$$

который можно считать элементом n-мерного евклидова пространства E^n.

Допустим также, что данный объект является *управляемым*, то есть для каждого момента времени $t \in [t_0, T]$ существует упорядоченный набор чисел

$$\{ u_1(t), u_2(t), \ldots, u_r(t) \} \equiv u(t) \in E^r ,$$

определяющих характер изменения состояния этого объекта.

Будем предполагать, что процесс изменения состояния рассматриваемого объекта описывается системой дифференциальных уравнений вида

$$\dot{x} = f(x,u) , \qquad (5.1.2.1)$$

причем элементы x и u должны в каждый момент времени $t \in [t_0, T]$ удовлетворять системе ограничений

$$g_s(x,u) \geq 0 , \quad s = [1,m] . \qquad (5.1.2.2)$$

Отметим, что как и дискретном случае:

- компоненты вектор-функции $x(t)$ принято называть *фазовыми переменными* или просто – *фазами*, а пространство E^n – *фазовым пространством*;
- компоненты $u(t)$ в этом случае называются *управляющими параметрами* или – *управлениями*, а пространство E^r – *пространством управлений*.

Различие между фазами и управлениями состоит в том, что только производные фаз (по времени) являются левыми частями уравнений (5.1.2.1).

Если фазы x и управления u удовлетворяют ограничениям (5.1.2.2), то их называют *допустимыми*.

Заметим также, что условия (5.1.2.2) не только обеспечивают ограниченность элементов x и u, но и реализуют зависимость между ними. Например, наличие в системе условий (5.1.2.2) уравнения вида $u = b(x)$ позволяет выбирать управления в зависимости от состояния управляемого объекта. Такого рода ограничения в теории оптимального управления принято называть *обратными связями*.

Для модели (5.1.2.1)-(5.1.2.2) представляется естественной постановка, так называемой задачи *оптимального управления*:

> найти вектор-функции $x^*(t) \in E^n$ и $u^*(t) \in E^r$, удовлетворяющие условиям (5.4.1)-(5.4.2) и доставляющие экстремум функционалу
>
> $$\Phi(x,u,T) . \qquad (5.1.2.3)$$

Приведем несложный пример задачи оптимального управления.

Пусть материальная точка массы m может перемещаться вдоль оси Ox под действием трех сил:

- ограниченной по модулю силы тяги u,
- диссипативной силы трения μv, пропорциональной скорости движения v,
- и упругой потенциальной силы κx.

Тогда согласно второму закону Ньютона движение точки будет определяться уравнением

$$m w = -\mu v - \kappa x + u \ ,$$

где w – ускорение точки.

Введем фазовые переменные $x_1 = x$, значение координаты точки на оси Ox, и $x_2 = v = \dot{x}_1$ – скорость движения точки вдоль оси. Поскольку ускорение точки $w = \ddot{x}_1 = \dot{x}_2$, то для рассматриваемой модели система уравнений (5.1.2.1) принимает вид

$$\begin{cases} \dot{x}_1 = x_2 \ , \\ \dot{x}_2 = -\dfrac{\kappa}{m} x_1 - \dfrac{\mu}{m} x_2 + \dfrac{1}{m} u \ . \end{cases} \qquad (5.1.2.4)$$

при условиях: $-c \le u \le c$, где c – некоторая положительная константа.

Требуется найти вектор-функции $x^*(t)$ и $u^*(t)$ такие, что объект, описываемый условиями (5.1.2.3) переходит из исходного состояния $x(0) = x^0$ в состояние $x_1 = x_2 = 0$ за минимально возможное время T^*.

Данное условие означает, что целевой функционал (5.1.2.3) в данном случае имеет вид: $\Phi(x,u,T) = T \to \min$. Отметим, что задачи оптимального управления с подобным целевым функционалом (как и в дискретном случае) называются задачами *быстродействия*.

Рассмотрим теперь схему решения задачи (5.1.2.1)-(5.1.2.2)-(5.1.2.3), основанную на идее *дискретизации времени* в линейных непрерывных задачах оптимального управления.

В линейном случае задача (5.1.2.1)-(5.1.2.2)-(5.1.2.3) формулируется, например, так

максимизировать по $\{x(t),u(t)\}$ $\int_0^T \left[(c(t),x(t))+(f(t),u(t))\right]dt$,

при условиях: $\dot{x} = A(t)x + B(t)u$, (5.1.2.5)

$D(t)x + E(t)u \geq o, \quad \forall t \in [0,T]$,

где зависящие от времени матрицы $A(t), B(t), C(t), D(t)$ и вектор-функции $x(t), u(t), c(t), f(t)$ имеют соответствующие друг другу размеры.

Дискретизация времени выполняется по очевидным формулам

$$\Delta t = \frac{T}{N}, \quad x^k = x\left(k\,\Delta t\right), \quad u^k = u\left(k\,\Delta t\right), \quad k = 0,1,2,...,N,$$

где N – полное число периодов. В выражениях для производных значения дифференциалов заменяются на *приращения* функций. Тогда задача (5.1.2.4) принимает вид

максимизировать по $\{x^k,u^k\}$ $\sum_{k=0}^{N}\left((c^k,x^k)+(f^k,u^k)\right)$,

при условиях $x^{k+1} - x^k = A^k x^k \Delta t + B^k u^k \Delta t$, (5.1.2.6)

$D^k x^k + E^k u^k \geq o, \quad \forall k \in [0, N-1]$.

Полученная задача (5.1.2.6) является обычной задачей линейного программирования, хотя и сравнительно высокой размерности.

Проиллюстрируем использование метода дискретизации времени в непрерывной задаче с динамическими связями (5.1.2.5) вида:

$$\begin{cases} \dot{x}_1 = -5x_1 + x_2 - 2x_3 + u_1 - 2u_2, \\ \dot{x}_2 = -x_1 - x_2 + u_1 - u_2, \\ \dot{x}_3 = 6x_1 - 2x_2 - 2x_3 - u_1 + 2u_2. \end{cases} \quad (5.1.2.7)$$

Рассмотрим вначале классическую задачу Коши, то есть случай, когда матрица $B(t)$ нулевая, при $x_1(0) = x_2(0) = x_3(0) = 1$. Здесь и далее были выбраны значения параметров: $T = 7.5,\ \Delta t = 0.01,\ N = 750.$

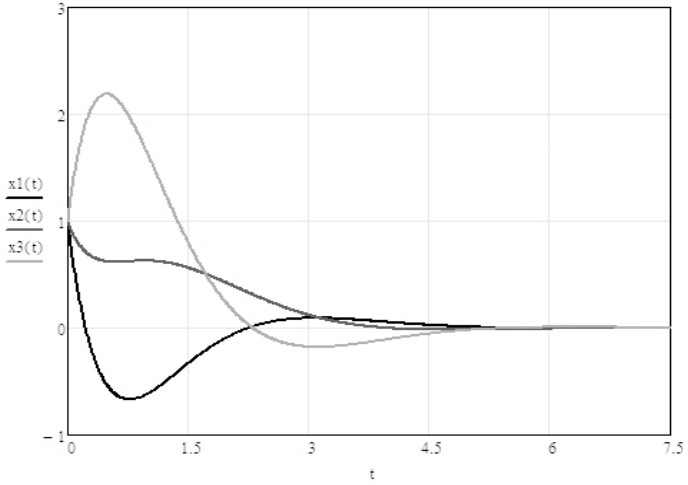


Рисунок 5.1.2.1.

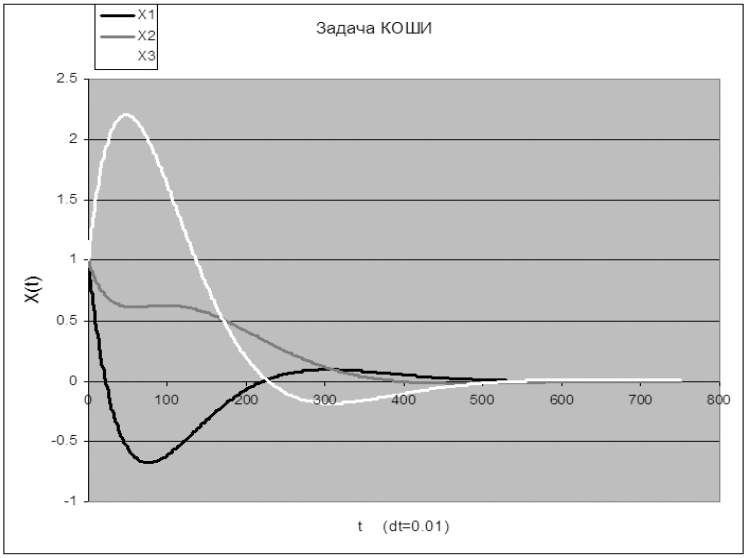


Рисунок 5.1.2.2.

$$\begin{cases} \dot{x}_1 = -5x_1 + x_2 - 2x_3, \\ \dot{x}_2 = -x_1 - x_2, \\ \dot{x}_3 = 6x_1 - 2x_2 - 2x_3. \end{cases}$$

Общее решение этой однородной системы дифференциальных уравнений имеет вид

$$\begin{Vmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{Vmatrix} = C_1 e^{-2t} \begin{Vmatrix} 1 \\ 1 \\ -1 \end{Vmatrix} + C_2 e^{-t} \begin{Vmatrix} \cos t \\ -\sin t \\ -2\cos t \end{Vmatrix} + C_3 e^{-t} \begin{Vmatrix} \sin t \\ \cos t \\ -2\sin t \end{Vmatrix},$$

причем заданные начальные значения фазовых переменных достигаются при $C_1 = 3$ и $C_2 = C_3 = -2$. Графики решений задачи Коши приведены на рис. 5.1.2.1.

Использование схемы дискретизации времени для этой задачи дает аналогичные результаты, показанные на рис. 5.1.2.2. Отметим, что в этом случае задача 5.1.2.6. является системой линейных уравнений.

В заключение приведем решения трех вариантов задачи оптимального управления (5.1.2.7) , в которых начальные условия и ограничения на управления одинаковы

$$x_1(0) = x_2(0) = x_3(0) = 1;$$
$$|u_1(t)| \le 1; \quad |u_2(t)| \le 1, \quad \forall t \in [0, T],$$

но имеются различные ограничения на фазовые переменные.

В первом варианте требуется построить оптимальные управления при граничном условии $x_1(T) \to \max$. Графики оптимальных решений для этого случая показаны на рис.5.1.2.3 и 5.1.2.4.

Для второго варианта граничные условия на фазовые переменные имели вид $x_1(T) = x_2(T) = x_3(T) = 0.2$. Графики соответствующих оптимальных решений показаны на рис.5.1.2.5 и 5.1.2.6.

Третий вариант получается из первого тем, что к краевому ограничению $x_1(T) \to \max$ добавлены условия

$$x_1(t) \ge -0.5, \; x_2(t) \ge -0.5, \; x_3(t) \ge -0.5, \quad \forall t \in [0, T].$$

Графики оптимальных решений для этого случая показаны на рис.5.1.2.7 и 5.1.2.8.

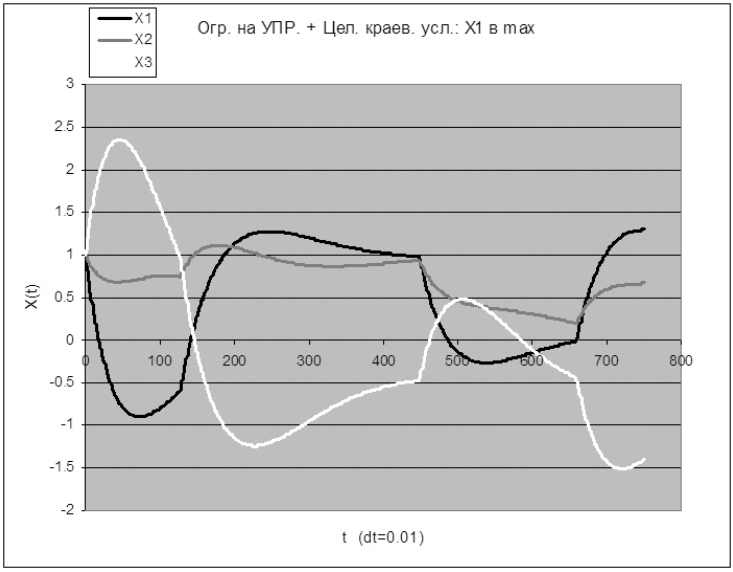


Рисунок 5.1.2.3.

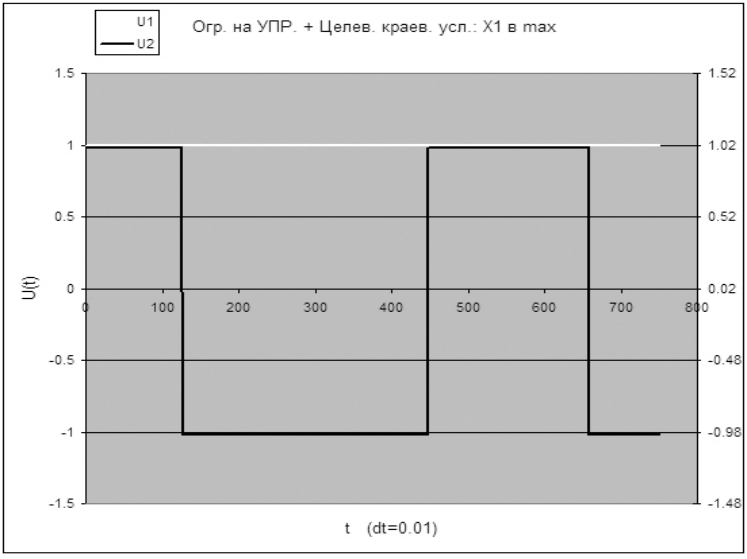


Рисунок 5.1.2.4.

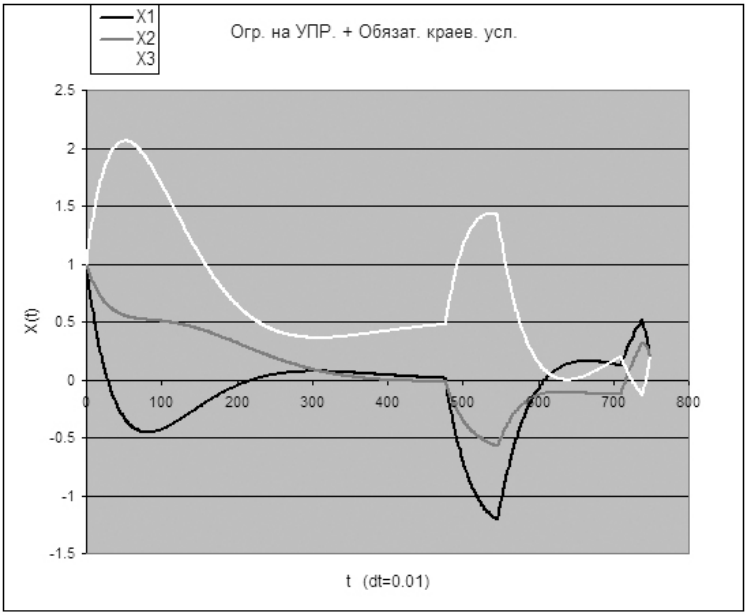


Рисунок 5.1.2.5.

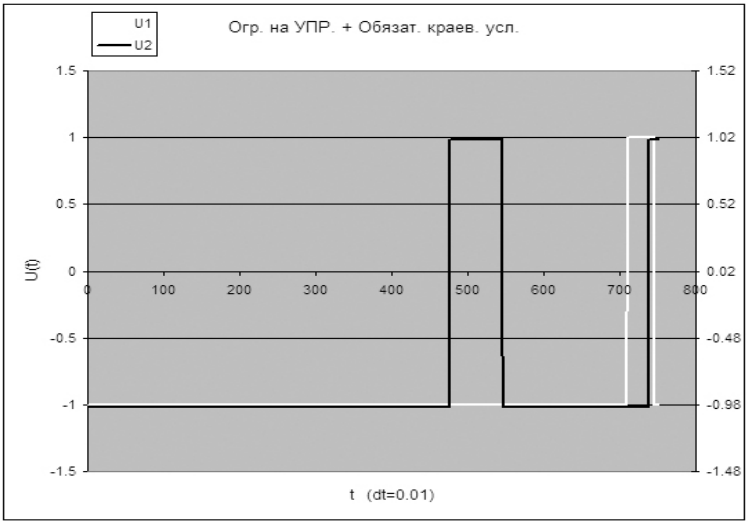


Рисунок 5.1.2.6.

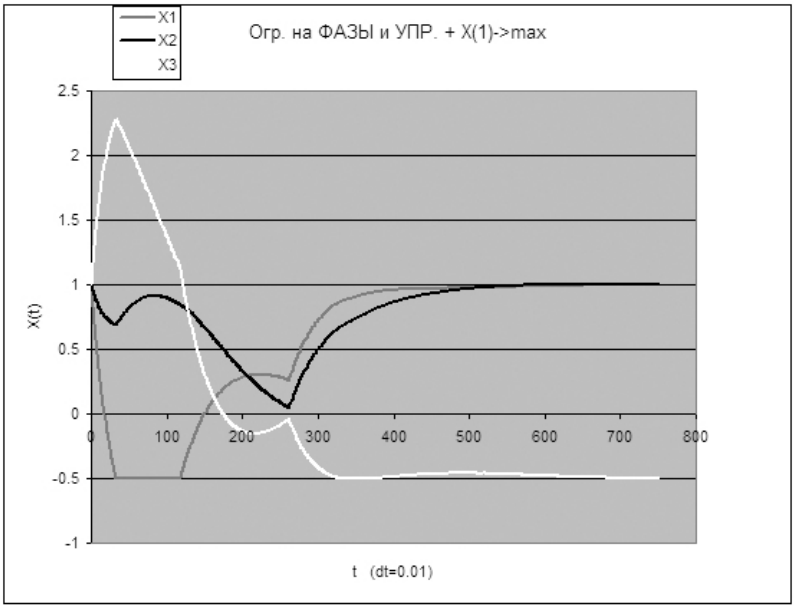


Рисунок 5.1.2.7.

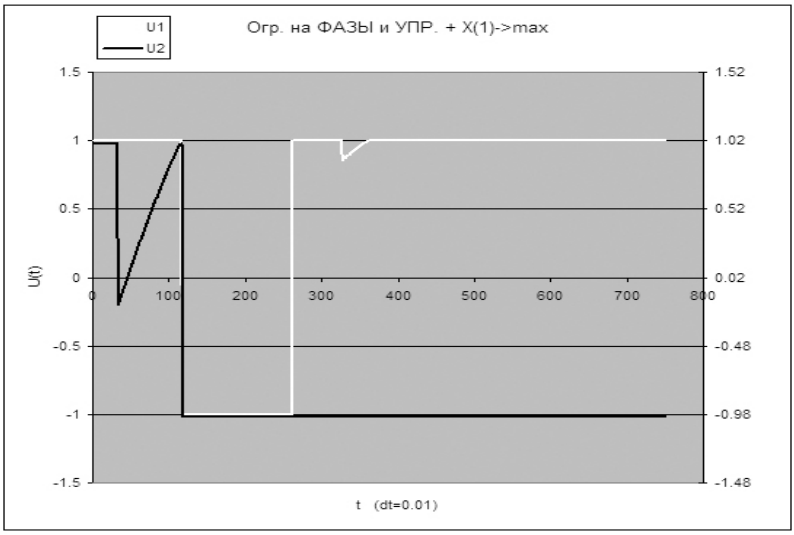


Рисунок 5.1.2.8.

Приведем без доказательства утверждение, что *при отсутствии ограничений на фазовые переменные управления находятся на своих граничных значениях, за исключением, быть может, точек «переключения»*. Этот факт иллюстрируют рис. 5.1.2.4, 5.1.2.6. и 5.1.2.8, на первых двух из которых управляющие переменные имеют свои граничные значения $\forall t \in [0,T]$, в то время как для третьего варианта (с фазовыми ограничениями) имеются интервалы времени, на которых оптимальные управления не находятся на своих границах.

§ 5.1.3. Принцип максимума Л.С.Понтрягина

Для достаточно широкого класса практически важных непрерывных задач быстродействия оказывается возможным построение оптимальных управлений без использования схемы дискретизации времени. Рассмотрим пример использования такого подхода.

Предположим, что для любого исходного x^0 – элемента фазового пространства, существует оптимальное по быстродействию управление $u^*(t)$, переводящее управляемый объект (5.1.2.1)-(5.1.2.2)-(5.1.2.3) в конечное состояние x^* за время $\tau(x^0)$.

Предположим также, что вектор-функция $f(x,u)$ непрерывно дифференцируема по x, а функционал $\tau(x)$ имеет непрерывные частные производные по всем своим аргументам до второго порядка включительно, и выведем в этих предположениях необходимые условия оптимальности процесса управления. Для большей наглядности на данном этапе рассуждений не будем принимать во внимание условия (5.1.2.2).

Пусть рассматриваемый объект начинает движение в фазовом пространстве под действием некоторого (*не обязательно оптимального*!) постоянного управления u^0, находясь при $t=0$ в состоянии x^0. Фазовую траекторию объекта в этом движении обозначим $x = y(t)$. Заметим, что вектор-функция $y(t)$ является в этом случае решением задачи Коши:

$$y' = f(y,u^0) \text{ , при условии } y(0) = x^0. \qquad (5.1.3.4)$$

Допустим, что через время Δt после начала движения объект оказывается в состоянии $x^1 = y(\Delta t)$.

Иначе говоря, на перемещение в фазовом пространстве из x^0 в x^1 объект затратил время Δt .

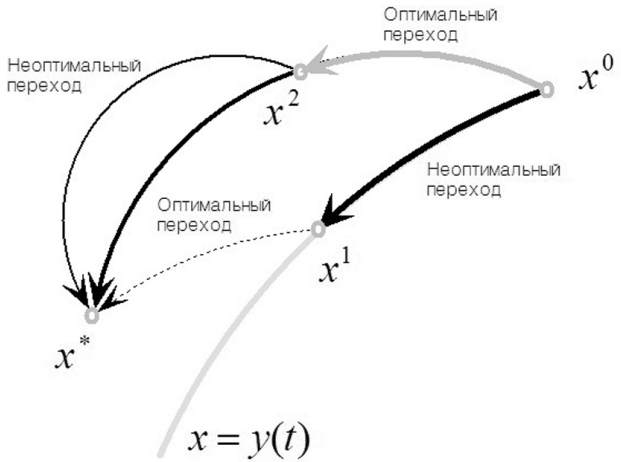


Рисунок 5.1.3.1.

Предположим теперь, что далее объект *оптимально* переходит из состояния x^1 в конечное состояние x^*. Этот переход потребует, согласно сделанным предположениям, времени $\tau(x^1)$. Поскольку оптимальный переход из x^0 в x^* потребовал бы времени $\tau(x^0)$, то в силу неоптимальности u^0 очевидно неравенство

$$\tau(x^0) \le \tau(x^1) + \Delta t \ ,$$

на основании которого, обозначив (ради формального удобства) $\omega(x) = -\tau(x)$, приходим к оценке

$$\frac{\omega(y(\Delta t)) - \omega(y(0))}{\Delta t} \le 1 ; \quad \forall \Delta t \ge 0 \ .$$

Учитывая, что предельный переход сохраняет нестрогие неравенства, при $\Delta t \to +0$ получаем

$$\frac{d}{dt}\omega(y(t))\Big|_{t=0} \le 1 \quad .$$

Далее, приняв во внимание, что $y(t)$ есть n-компонентный элемент фазового пространства E^n и использовав правило дифференцирования сложной функции

$$\omega(x) = \omega(\, y(t)\,),$$

приходим к неравенству

$$\sum_{i=1}^{n} \frac{\partial \omega}{\partial y_j}\bigg|_{y\,=\,y(0)} \cdot y_j'(0) \le 1\ ,$$

или, в силу (5.1.3.4),

$$\sum_{j=1}^{n} \frac{\partial \omega}{\partial x_j}\bigg|_{x\,=\,x_0} \cdot f_j(x_0, u_0) \le 1\ .$$

Значит, в виду произвольности x^0 и u^0, $\forall x \in E^n$ и $\forall u \in E^r$ справедлива оценка

$$\sum_{j=1}^{n} \frac{\partial \omega}{\partial x_j} \cdot f_j(x,u) \le 1\,.$$

Теперь рассмотрим оптимальный переход из x^0 в x^*[7]. Пусть точка x^2 принадлежит $x = x^*(t)$ – траектории *оптимального* перехода из x^0 в x^*, для обеспечивающей этот оптимум управляющей вектор-функции $u^*(t)$, и пусть время перехода из x^0 в x^2 равно δt. Тогда

$$\tau(x^0) = \tau(x^2) + \delta t \quad \text{или} \quad \omega(x^2) - \omega(x^0) = \delta t\,,$$

что дает оценку для оптимальной траектории

$$\frac{d}{dt}\omega(\, x^*(t)\,)\bigg|_{t\,=\,0} = 1 \quad .$$

После преобразований, аналогичных выполненным выше, получаем

$$\sum_{j=1}^{n} \frac{\partial \omega}{\partial x_j}\bigg|_{x\,=\,x^*(t)} \cdot f_j(\, x^*(t),\, u^*(t)) = 1\ . \qquad (5.1.3.5)$$

---

[7] Здесь следует не путать x^* – конкретный элемент фазового пространства, и вектор-функцию $x^*(t)$.

Введем в рассмотрение функцию, зависящую от произвольных $x(t) \in E^n$ и $u(t) \in E^r$ вида

$$B(x,u) = \sum_{j=1}^{n} \frac{\partial \omega}{\partial x_j} \cdot f_j(x,u), \qquad (5.1.3.6)$$

называемую *функцией Беллмана* [8]. Тогда будет справедлива

Теорема 5.1.3.1. **Для любых $x \in E^n$ и $u \in E^r$ справедливо неравенство:**

$$\max_{u \in E^n} B(x,u) \le 1 \quad ,$$

причем равенство

$$\max_{u \in E^n} B(x,u) = 1$$

достигается для любого оптимального процесса

$$\{x^*(t), u^*(t)\}.$$

Отметим, что утверждение теоремы (5.1.3.1) является *необходимым* условием оптимальности для задачи (5.1.2.1)-(5.1.2.2) и служит основой так называемого *метода динамического программирования Беллмана*.

Практическое применение этого метода затруднено необходимостью нахождения функции $\omega(x)$. Поэтому в первую очередь рассмотрим возможный подход к преодолению этого затруднения.

Пусть $\{x^*(t), u^*(t)\}$ – оптимальный процесс, переводящий объект в состояние $x^* = x^*(T)$ за минимально возможное время. Зафиксируем некоторый момент времени $t \in (0,T)$ и рассмотрим $\overline{B}(x, u^*(t))$ как функционал, зависящий только от вектор-функции x.

Как было показано выше, этот функционал достигает своего максимума по x при $x = x^*(t)$, и, значит,

$$\frac{\partial}{\partial x_i} \overline{B}(x, u^*(t)) = 0; \quad \forall i = [1,n].$$

[8] Уравнение (5.1.2.5) также часто называют *уравнением Беллмана*.

Применив еще раз правило дифференцирования сложной функции, с учетом (5.1.3.6) при $x = x^*(t)$ и $u = u^*(t)$ получим

$$\sum_{j=1}^{n} \frac{\partial^2 \omega}{\partial x_j \partial x_i} \cdot f_j(x,u) + \sum_{j=1}^{n} \frac{\partial \omega}{\partial x_j} \cdot \frac{\partial f_j}{\partial x_i} = 0 \; ; \forall i = [1,n] .$$

С другой стороны,

$$\frac{d}{dt}\left(\frac{\partial \omega}{\partial x_i}\right) = \sum_{j=1}^{n} \frac{\partial^2 \omega}{\partial x_j \partial x_i} \cdot x_j' = \sum_{j=1}^{n} \frac{\partial^2 \omega}{\partial x_j \partial x_i} \cdot f_j(x,u) \; ; \forall i = [1,n] ,$$

что окончательно дает

$$\frac{d}{dt}\left(\frac{\partial \omega}{\partial x_i}\right) + \sum_{j=1}^{n} \frac{\partial \omega}{\partial x_j} \cdot \frac{\partial f_j}{\partial x_i} = 0 \; ; \forall i = [1,n] .$$

Заметим, что в последние равенства входят не функция $\omega(x)$, а ее частные производные. Это позволяет упростить полученные соотношения, равно как и необходимые условия оптимальности, при помощи очевидной замены

$$\frac{\partial \omega}{\partial x_i} = \psi_i(t) \; ; \quad \forall i = [1,n]$$

к виду

$$\psi_i' = -\sum_{j=1}^{n} \psi_j \frac{\partial f_j}{\partial x_i} ; \quad \forall i = [1,n]$$

в том числе и условие (5.1.3.5)

$$\sum_{j=1}^{n} \psi_j \cdot f_j(x^*(t), u^*(t)) = 1 \; .$$

Структура этих соотношений обуславливает целесообразность введения специальной, зависящей от $2n + r$ аргументов, функции

$$H(\psi, x, u) = \sum_{j=1}^{n} \psi_j f_j(x,u) ,$$

в терминах которой необходимое условие экстремума задачи оптимального управления формулируются в виде

Если $\{x^*(t), u^*(t)\}$ – оптимальный процесс, переводящий объект в состояние $x^* = x^*(T)$ за минимально возможное время, то существует ненулевая вектор-функция $\psi(t)$ такая, что

$$\psi_i' = -\sum_{j=1}^{n} \psi_j \frac{\partial H}{\partial x_i}; \quad \forall i = [1, n]$$

и (5.1.3.7)

$$H(\psi(t), x^*(t), u^*(t)) \equiv 1; \quad \forall t \in [0, T].$$

Условия (5.1.3.7) носят название *принципа максимума*. Они были выведены в предположении существования непрерывных вторых производных функционала $\omega(x)$, которое оказывается неверным в подавляющем большинстве практически важных задач. Это обстоятельство в значительной степени ограничивает применение данных условий.

Замечательным, однако, является тот факт, что принцип максимума может быть сформулирован как

Теорема 5.1.3.2. (Принцип максимума Л.С.Понтрягина)

Для любых $x^* \in E^n$ и $u^* \in E^r$ справедливы условия:

$$H(\psi(\tau), x^*(\tau), u^*(\tau)) = \max_{v \in E^r} H(\psi(\tau), x^*(\tau), v); \forall \tau \in [0, T^*],$$

где τ есть точка непрерывности $u^*(t)$ и

$$H(\psi(T^*), x^*(T^*), u^*(T^*)) \geq 0,$$

а вектор-функция $\psi(t)$ удовлетворяет системе уравнений

$$\psi_i' = -\sum_{j=1}^{n} \psi_j \frac{\partial H}{\partial x_i}; \quad \forall i = [1, n]$$

Причем эта теорема оказывается справедливой без каких-либо предположений о непрерывности и дифференциальных свойствах функционала $\omega(x)$.

В заключение рассмотрим пример применения принципа максимума.

Рассмотрим снова управляемый объект, поведение которого описывается уравнениями (5.1.3.3). Причем для большей наглядности будем полагать, что упругая сила и сила трения отсутствуют, а масса тела единичная. То есть $m = 1;\ \kappa = 0;\ \beta = 0$, и мы приходим к следующей системе уравнений

$$\begin{cases} x_1' = x_2\ , \\ x_2' = u\ , \\ |u| \le 1\ . \end{cases} \tag{5.1.3.8}$$

Функционал $H(\psi, x, u)$, необходимый для построения оптимального по быстродействию управления, будет иметь вид

$$H(\psi, x, u) = \psi_1 x_2 + \psi_2 u\ ,$$

а вспомогательная система дифференциальных уравнений (5.1.3.7) соответственно

$$\begin{cases} \psi_1' = 0\ , \\ \psi_2' = -\psi_1\ . \end{cases} \tag{5.1.3.9}$$

Решение этой системы легко находится и $\forall t \in [0, T]$

$$\begin{cases} \psi_1(t) = c_1\ , \\ \psi_2(t) = -c_1 t + c_2\ . \end{cases}$$

Поскольку функционал $H(\psi, x, u)$ в рассматриваемой задаче линеен по u, то согласно теореме 5.1.3.2 мы получаем следующую форму условий оптимальности

$$\begin{cases} u(t) = 1, \quad \text{если } \psi_2(t) \ge 0\ , \\ u(t) = -1, \text{ если } \psi_2(t) < 0\ . \end{cases} \tag{5.1.3.10}$$

С другой стороны, функция $\psi_2(t) = -c_1 t + c_2$ линейна по времени, и, значит, меняет знак на $[0, T]$ не более одного раза. Поэтому каждое оптимальное управление есть кусочно-постоянная функция времени со значениями ± 1.

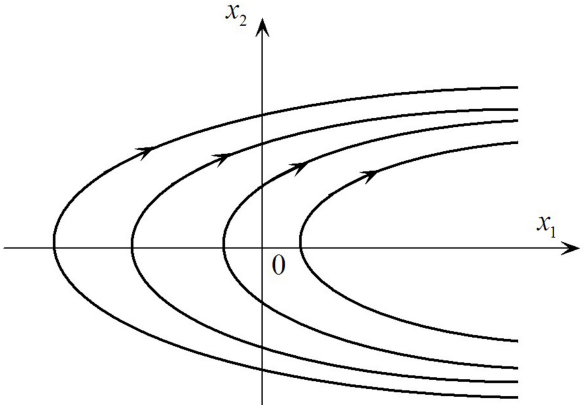


Рисунок 5.1.3.2.

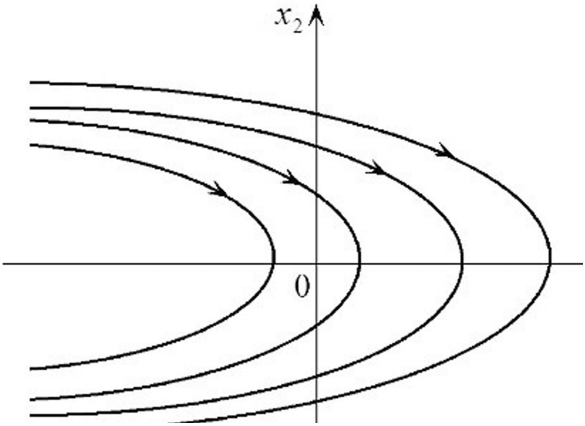


Рисунок 5.1.3.3.

Найдем теперь выражения для оптимальных (в смысле быстродействия) фазовых траекторий. Из системы (5.1.3.8) для тех t, где $u = 1$, получаем

$$\begin{cases} x_2(t) = t + d_2 \ , \\ x_1(t) = \int x_2(t)dt = \dfrac{1}{2}(t + d_2)^2 + d_1 \ , \end{cases}$$

где c_1, c_2, d_1 и d_2 – некоторые неопределенные константы. Причем из последнего равенства следует

$$x_1 = \frac{1}{2} x_2^2 + d_1 \,, \qquad (5.1.3.11)$$

то есть, искомые оптимальные фазовые траектории $x^*(t)$, отвечающие $u = 1$, должны принадлежать семейству парабол (5.1.3.11). Графики линий этого семейства показаны на рис.5.1.3.2. Направление движения вдоль этих парабол определяется условием $x_2' = u = 1 > 0$.

Рассуждая аналогичным образом, получаем семейство парабол – оптимальных фазовых траекторий для $u = -1$

$$x_1 = -\frac{1}{2} x_2^2 + e_1 \,, \qquad (5.1.3.12)$$

графики которых показаны на рис. 5.1.3.3 , где e_1 некоторая произвольная константа.

Выясним теперь как выглядит оптимальная по быстродействию траектория, переводящая объект из исходного состояния x^0 в конечное состояние x^*. Пусть координатные представления этих элементов имеют вид

$$\| x^0 \| = \begin{Vmatrix} x_1^0 \\ x_2^0 \end{Vmatrix} \quad \text{и} \quad \| x^* \| = \begin{Vmatrix} 0 \\ 0 \end{Vmatrix}.$$

Траектории семейства (5.1.3.11) (равно как и траектории семейства (5.1.3.12)) очевидно не пересекаются. Поэтому точка фазовой плоскости x^0 принадлежит лишь двум траекториям, проходящим через нее: одной из семейства (5.1.3.11) и одной из (5.1.3.12).

Согласно теореме 5.1.3.2 движение вдоль этих траекторий *необходимое* условие оптимальности процесса, причем при $u = 1$ движение будет осуществляться по траектории из (5.1.3.11), а при $u = -1$ – по траектории из (5.1.3.12).

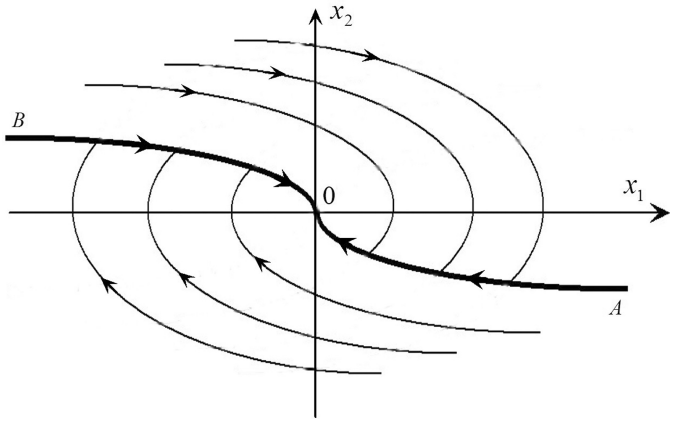


Рисунок 5.1.3.4.

С другой стороны, оптимальный процесс должен приводить объект в x^* – начало координат фазовой плоскости. Но среди траекторий семейств (5.1.3.11) и (5.1.3.12) имеются только две, приходящие в x^*: это дуги полупарабол BO и AO, определяемые соответственно условиями

$$x_1 = \frac{1}{2}x_2^2\,, \quad x_2 < 0 \qquad \text{или} \qquad x_1 = -\frac{1}{2}x_2^2\,, \quad x_2 > 0\,,$$

изображенные на рис. 5.1.3.4 жирными линиями. На этом же рисунке нежирными линиями показаны траектории, позволяющие попасть на дугу BO или дугу AO.

Поскольку для каждой точки фазовой плоскости не принадлежащей этим дугам существует только одна траектория, приводящая на BO или на AO, то мы приходим к следующему правилу построения оптимальной по быстродействию траектории.

А) Если точка x^0 расположена *ниже линии* AOB, то вначале движение осуществляется из этой точки по траектории принадлежащей семейству (5.1.3.11) (то есть при $u = 1$) до момента попадания на дугу BO в некоторой точке x^1. Затем при $u = -1$ выполняется переход по BO из x^1 в начало координат.

Б) Если же x^0 расположена *выше* AOB, то движение сначала осуществляется по траектории из семейства (5.1.3.12) с $u=-1$ до момента попадания на дугу OA. Затем с $u=1$ переходим по OA в начало координат.

В) Наконец, если x^0 принадлежит BO (или же OA), то переходим из x^0 непосредственно в начало координат по дуге BO с $u=1$ (по OA при $u=-1$).

Кроме того, можно доказать[9], что использование правил А)-Б)-В) не только необходимо, но и достаточно для оптимальности построенной траектории. Заметим также, что полученная геометрия оптимальной траектории позволяет назвать линию AOB *линией переключения управления*.

В качестве упражнения покажите самостоятельно, что:

1º. Для объекта, описываемого системой условий (5.1.3.8), $\tau(x^0)$ – минимальное время перехода из точки x^0 с $\left\| x^0 \right\| = \left\| \begin{matrix} x_1^0 \\ x_2^0 \end{matrix} \right\|$ в начало координат, дается формулой

$$\tau(x^0)=\begin{cases} x_2^0+\sqrt{x_1^0+\frac{1}{2}(x_2^0)^2}\,, & \text{если } x^0 \text{ выше } AOB, \\ -x_2^0+\sqrt{-x_1^0+\frac{1}{2}(x_2^0)^2}\,, & \text{если } x^0 \text{ ниже } AOB. \end{cases}$$

2º. $\tau(x^0)$ непрерывный, на всей фазовой плоскости функционал, однако не имеющий на линии переключения частных производных по x_1^0 и x_2^2.

Завершая рассмотрение непрерывных задач оптимального управления, следует отметить, что в вычислительной практике достаточно часто оказывается успешным подход, заключающийся *в дискретизации* (то есть

[9] Это доказательство выходит за рамки нашего курса.

в разбиении на конечное число промежутков, как в §5.1.2) отрезка $[t^0, T]$, что позволяет свести исходную задачу (5.1.3.3) к дискретной задаче вида (5.1.1.1)-(5.1.1.3), являющейся, хотя и высокоразмерной, но все же конечномерной задачей математического программирования.

Раздел 5.2. ЗАДАЧИ ПАРАМЕТРИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Рассмотренные в главах 3 и 4 методы решения задач математического программирования в общем случае не гарантируют получение решения с приемлемым уровнем затрат вычислительных ресурсов, таких, как требуемая оперативная или энергонезависимая память, процессорное время и т.д. Поэтому с методологической точки зрения представляется актуальным исследование возможных модификаций как постановок задач, так и алгоритмов поиска их решений, позволяющих снижение совокупных затрат вычислительных усилий и ресурсов.

Основой одного из таких приемов − традиционно называемого *параметрическим программированием*, является идея разделения множества переменных в задаче математического программирования на две группы: собственно *переменные* и *параметры*, причем значения последних при необходимости могут быть фиксированы или целенаправленно изменяемы, является предметом рассмотрения в данном параграфе.

§ 5.2.1. Общая постановка и примеры двухуровневых задач

Рассмотрим вначале наиболее простую постановку задачи параметрического программирования.

Пусть $x \in E^n$ – вектор переменных и $u \in E^l$ – вектор параметров являются элементами конечномерных евклидовых пространств соответственно с координатными представлениями

$$\|x\| = \|\xi_1 \quad \xi_2 \quad \cdots \quad \xi_n\|^{\mathrm{T}} \quad \text{и} \quad \|u\| = \|\nu_1 \quad \nu_2 \quad \cdots \quad \nu_l\|^{\mathrm{T}}.$$

Рассмотрим следующую задачу, которую принято называть *задачей параметрического программирования*:

$$\text{найти } \min_x f(x,u) \tag{5.2.1.1}$$

$$\text{при условиях: } y_s(x,u) \geq 0,\ s=[1,m], \tag{5.2.1.2}$$

и пусть $x^*(u)$ есть решение задачи (1)–(2) для некоторого фиксированного $u \in \Omega \subseteq E^l$.

Любую задачу, в формулировке которой используется $x^*(u)$, будем называть задачей *в пространстве параметров* или задачей *верхнего уровня*. Например, задачу

$$\min_u f(x^*(u),u) \text{ при условии } u \in \Omega. \tag{5.2.1.3}$$

В отличие от задач типа (5.2.1.3), задачу (5.2.1.1)–(5.2.1.2) будем называть задачей *нижнего уровня*.

Сразу следует отметить, что, хотя система задач (5.2.1.1)–(5.2.1.2) и (5.2.1.3) сводится к задаче математического программирования вида

$$\min_{\{x,u\}} f(x,u)$$

$$\textit{при условиях } y_s(x,u) \geq 0,\ s=[1,m],\ u \in \Omega, \tag{5.2.1.4–5.2.1.5}$$

такое сведение может приводить к неприемлемому уровню усложнения задачи.

Кроме того, число уровней в задаче параметрического программирования может превышать два, но принципиальных отличий от случая системы задач (5.2.1.1)–(5.2.1.2) и (5.2.1.3) это не порождает. Поэтому далее будет рассматриваться лишь *пара* задач нижнего и верхнего уровней.

Приведем иллюстративный пример постановки и решения задачи оптимизации по параметрам для следующей линейной модели:

на *нижнем уровне* максимизировать по $\{\xi_1,\xi_2\}$ функцию $2\xi_1+3\xi_2$,

при условиях $0 \leq \xi_1 \leq 4;\ 0 \leq \xi_2 \leq 3;$

$$\xi_1+2\xi_2 \leq \nu_1,$$

$$2\xi_1+\xi_2 \leq \nu_2,$$

для значений параметров $\{\nu_1,\nu_2\}$, являющихся решением задачи *верхнего уровня*:

максимизировать по $\{\nu_1, \nu_2\}$ функцию

$$f^*(\nu_1, \nu_2) = 2\xi_1^*(\nu_1, \nu_2) + 3\xi_2^*(\nu_1, \nu_2),$$

при условиях $0 \le \nu_1 \le 15;\ 0 \le \nu_2 \le 15$.

Для решения этой двухуровневой задачи была использована схема последовательных приближений к искомому решению u^* в пространстве E^l, описанная в следующем параграфе 5.2.2.

Таблица 5.2.1.1

Шаг №	ν_1	ν_2	$\xi_1^*(\nu_1, \nu_2)$	$\xi_2^*(\nu_1, \nu_2)$
0	1.000	1.000	0.333	0.333
1	3.000	1.500	0.000	1.500
2	6.000	3.000	0.000	3.000
3	10.000	11.000	4.000	3.000

Таблица 5.2.1.2

Шаг №	$f^*(\nu_1, \nu_2)$	ω_1	ω_2	σ
0	1.667	4.000	1.000	0.500
1	4.500	2.000	1.000	1.500
2	9.000	1.000	2.000	4.000
3	17.000	0.000	0.000	0.000

Детали реализации этой схемы будут рассмотрены в § 5.2.2. Здесь же ограничимся приведением полученных результатов в табличной и графической формах.

Рис. 5.2.1.1 представляет пошаговое изменение допустимой области задачи нижнего уровня. На рис. 5.2.1.2 показана траектория движения в пространстве параметров.

Другим примером возможного использования двухуровневой схемы решения задач параметрического программирования является проблема согласования различных целевых функций в многокритериальных математических моделях рассмотренная в разделе 5.3.

Отметим, что, задачи (5.3.1)-(5.3.2) уже сами по себе двухуровневые, поскольку в их формулировку входят решения вспомогательных однокритериальных задач. Поэтому задачу (5.3.4) – "оптимизации формы множества Парето", можно рассматривать как трехуровневый вариант задачи параметрического программирования.

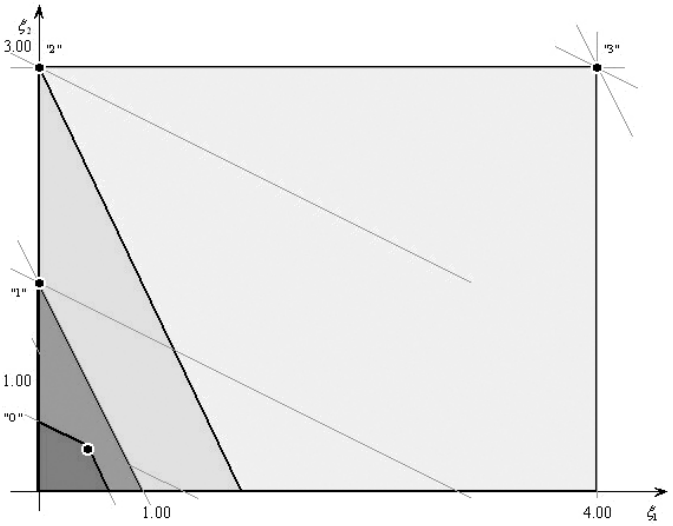


Рис. 5.2.1.1.

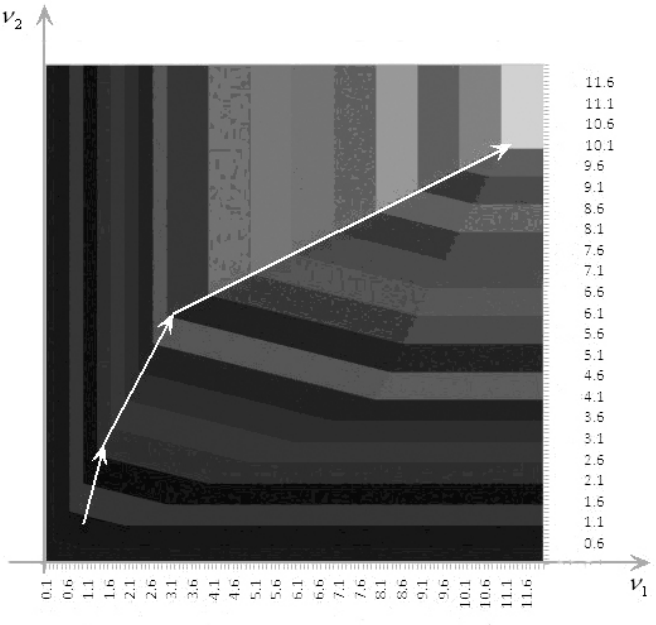


Рис. 5.2.1.2.

Другим примером использования двухуровневой схемы решения конечномерных задач математического программирования является часто встречающийся в практике частный случай N -периодической задачи дискретного оптимального управления со смешанными ограничениями, рассмотренной детально в § 5.1.1.

Действительно, пусть $x(k)$ – n-компонентный вектор состояния (фазовых переменных), а $y(k)$ – r-компонентный вектор управлений на k-м периоде, где $k=[1,N]$. На прямом произведении евклидовых пространств векторов состояний и управлений требуется найти совокупность векторов $\{x^*(k), y^*(k), k=[1,N]\}$, минимизирующих функцию

$$\sum_{k=0}^{N} F(x(k), y(k), k) \tag{5.2.1.6}$$

при условиях

$$\begin{array}{l} x_i(k+1) = f_i(x(k), y(k), k); \\ i=[1,n], \quad k=[1,N-1] \end{array}, \tag{5.2.1.7}$$

$$\begin{array}{l} g_j(x(k), y(k), k) \geq 0; \\ j=[1,m]; \ k=[1,N] \end{array}, \tag{5.2.1.8}$$

где функции $F(x(k), y(k), k)$, $f_i(x(k), y(k), k)$ и $g_j(x(k), y(k), k)$ – непрерывно дифференцируемые по фазовым переменным $x(k) \in E^n$ и управлениям $y(k) \in E^r$; $\forall k=[1,N]$.

Допустим также, что из множества компонентов вектора управления можно выделить некоторое подмножество таких, фиксация (то есть превращение в параметры) которых делает задачу (5.2.1.6)–(5.2.1.8) линейной. Образуем из этого подмножества l-компонентные векторы параметров $u(k), \forall k=[1,N]$, сохранив за остальными управлениями обозначение $y(k), \forall k=[1,N]$.

Задачу дискретного оптимального управления со смешанными ограничениями такого типа назовем *параметрически линеаризуемой*. Для ее решения также можно использовать двухуровневую схему, на *нижнем уровне* которой решаются при фиксированном $u(k), k=[1,N]$ линейные задачи оптимального управления со смешанными ограничениями, а на

верхнем уровне решается оптимизационная (вообще говоря, нелинейная) задача в пространстве параметров.

Следует отметить, что эта задача в общем случае является оптимизационной задачей математического программирования в E^l с неявно заданными условиями, пример которой детально рассматривается в §8.2.

Наконец, схема двухуровневой параметрической оптимизации может быть также использована для решения задач на системе математических моделей. Задачи этого класса возникают либо в случае связи первоначально независимых моделей в единый комплекс, либо при декомпозиции высокоразмерной модели на совокупность моделей подсистем.

Пусть имеется $k = [1, K]$ задач математического программирования следующего вида:

$$\textit{найти } \max_{x^k} f_k(x^k, u^k),\ x^k \in E^{n^k},\ u^k \in E^{l^k}, \qquad (5.2.1.9)$$

$$\textit{при условиях } y_{sk}(x^k, u^k) \geq 0,\ s = [1, m^k],$$

каждая из которых решается независимо в пространстве E^{n^k} при фиксированном векторе параметров u^k. И пусть "согласующая" задача в пространстве $E = E^N \bigcup E^L$, где

$$E^N = \bigcup_{k=1}^{K} E^{n^k},\ N = \sum_{k=1}^{K} n^k \text{ и } E^L = \bigcup_{k=1}^{K} E^{l^k},\ L = \sum_{k=1}^{K} l^k$$

имеет следующий вид:

$$\textit{найти } \max_{\{x,u\}} F(x,u),\ x = \{x^1, x^2, \ldots, x^K\},\ u = \{u^1, u^2, \ldots, u^K\} \qquad (5.2.1.10)$$

$$\textit{при условиях } Y_j(x,u) \geq 0,\ j = [1, M].$$

Тогда совокупность задач (5.2.1.8)-(5.2.1.10) можно рассматривать как *двухуровневую* систему, где задача верхнего уровня

$$\textit{найти } \max_{u} F(x^*(u), u),\ x^*(u) = \{x^{*1}(u), x^{*2}(u), \ldots, x^{*K}(u)\},$$

$$\textit{при условиях: } Y_j(x^*(u), u) \geq 0,\ j = [1, M], \qquad (5.2.1.11)$$

формулируется при помощи $x^{*k}(u)$, $k = [1, K]$ – решений задач (5.2.1.9) нижнего уровня. Условия сходимости процедуры поиска решения данной задачи в пространстве параметров, аналогичны рассмотренным выше.

Заметим, что исходная задача в силу своей специальной структуры может быть декомпозирована в двухуровневую систему задач. Такая декомпозиция может оказаться полезной, если $n^k >> l^k$, $k = [1, K]$, поскольку затраты вычислительных ресурсов на решение задач математического программирования обычно возрастают быстрее, чем растет их размерность.

Для иллюстрации схемы параметрической интеграции моделей рассмотрим две линейные оптимизационные задачи с параметрами в правых частях ограничений:

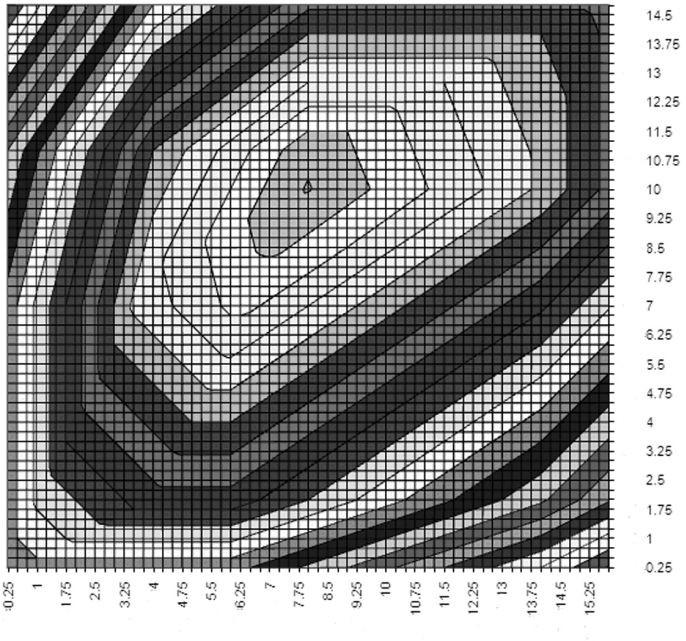


Рис. 5.2.1.3.

максимизировать по $\{\xi_1, \xi_2\}$ функцию $f^1 = 4\xi_1 + 4\xi_2$,

при условиях $0 \le \xi_1 \le 4$; $0 \le \xi_2 \le 2$;

$$\xi_1 + 2\xi_2 \le \nu_1,$$

$$2\xi_1 + \xi_2 \le \nu_2.$$

Максимизировать по $\{\xi_3, \xi_4\}$ функцию $f^2 = 4\xi_3 + 3\xi_4$,

при условиях $0 \le \xi_3 \le 4;\ 0 \le \xi_4 \le 2;$ $\begin{matrix} 2\xi_1 + \xi_2 \le \nu_3, \\ \xi_1 + 2\xi_2 \le \nu_4. \end{matrix}$

Пусть $\left\| \begin{matrix} \xi_1^*(\nu_1, \nu_2) \\ \xi_2^*(\nu_1, \nu_2) \end{matrix} \right\|$ и $\left\| \begin{matrix} \xi_3^*(\nu_3, \nu_4) \\ \xi_4^*(\nu_3, \nu_4) \end{matrix} \right\|$ – соответственно решения этих задач *нижнего уровня*, а задача *верхнего уровня* имеет следующий вид:

максимизировать по $\{\nu_1, \nu_2, \nu_3, \nu_4\}$ функцию

$$2.5 \cdot f^1(\xi_1^*(\nu_1, \nu_2), \xi_2^*(\nu_1, \nu_2)) + 2.5 \cdot f^2(\xi_3^*(\nu_3, \nu_4), \xi_4^*(\nu_3, \nu_4)) \tag{5.2.1.12}$$

при условиях $\nu_1 \ge 0, \nu_2 \ge 0, \nu_3 \ge 0, \nu_4 \ge 0,$

$\nu_1 + \nu_2 = 16, \quad \nu_3 + \nu_4 = 15.$

Система изолиний в пространстве параметров для целевого функционала (5.2.1.12) показана на рис. 5.2.1.3. Нетрудно заметить, что этот функционал (с точностью до множителя 2.5) совпадает с выражением

$$f^{*1}(\nu_1, \nu_2) + f^{*2}(\nu_1, \nu_2),$$

то есть критерии задач согласованы.

§ 5.2.2. Особенности решения задач параметрического программирования

Пусть $x^*(u)$ есть решение задачи (5.2.1.1)–(5.2.1.2) для фиксированного u, а задача верхнего уровня сформулирована в виде

$$\min_u F(x^*(u), u) \ \ u \in \Omega, \tag{5.2.2.1}$$

где $F(x, u)$ – функция, зависящая как от $x \in E^n$, так и от $u \in E^l$.

В этом случае как постановка, так и процедура решения задачи (5.2.2.1) могут в значительной степени осложняться следующими специфическими свойствами зависимости $x^*(u)$.

1. Практической невозможностью (за исключением, быть может, некоторых тривиальных случаев) аналитического решения задачи нижнего уровня (5.2.1.1)–(5.2.1.2), а, значит, также и постановки, исследования и решения в явном виде задачи верхнего уровня (5.2.2.1).

2. Несовпадением в общем случае области определения зависимости $x^*(u)$ и множества Ω, поскольку система условий задачи нижнего уровня (5.2.1.1)–(5.2.1.2) может оказаться противоречивой для некоторых $u \in \Omega \subseteq E^l$.

3. Нефункциональностью (неоднозначностью) зависимости $x^*(u)$ для тех $u \in \Omega \subseteq E^l$, при которых задача нижнего уровня (5.2.1.1)–(5.2.1.2) имеет решение, но не единственное.

4. Негладкостью зависимости $x^*(u)$ в силу того, что условия задачи "нижнего уровня" (5.2.1.1)–(5.2.1.2) могут содержать ограничения типа *неравенство*.

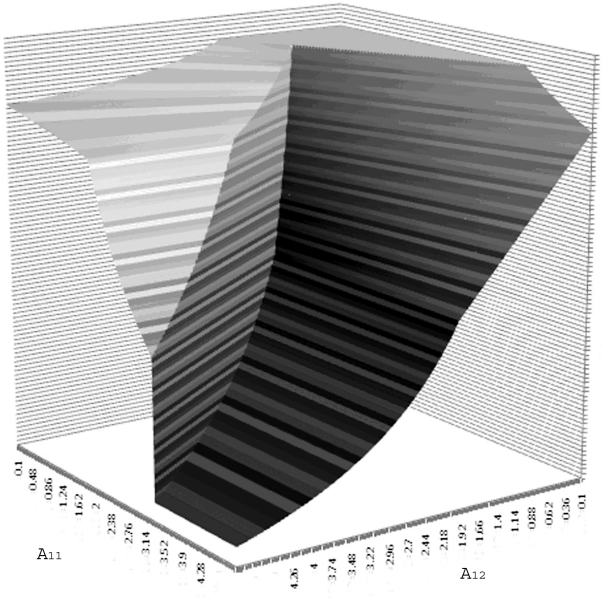


Рис. 5.2.2.1.

Более того, даже существование непрерывных производных у функций $f(x,u)$ и $y_s(x,u)$ достаточно высокого порядка не гарантирует необходимой гладкости (а иногда даже и непрерывности $x^*(u)$) и, следовательно, входящих в формулировку задачи верхнего уровня условий, что иллюстрирует рис.5.2.2.1, на котором показан типичный вид зависимости $x^*(u)$.

Принимая во внимание отмеченные специфические свойства зависимостей, можно прийти к заключению, что для практического решения задач верхнего уровня может быть использована стандартная итеративная схема последовательных приближений к искомому решению u^* в пространстве E^l k-й шаг которой состоит из следующего набора операторов:

1) для некоторого исходного приближения u_k решается задача нижнего уровня,
2) затем оцениваются как величина σ_k, так и направление шага w_k,
3) выполняется переход к точке u_{k+1} по формуле

$$u_{k+1} = u_k + \sigma_k w_k,$$

4) если в u_{k+1} условие окончания итерационного процесса не выполнено, то u_k присваивается значение u_{k+1} и делается переход к пункту 1.

В случае сходимости данной процедуры будет найдено u^* – локальное решение задачи (5.2.1.3).

В общем случае для решения задач типа (5.2.1.3) рекомендуется использование алгоритмов негладкой оптимизации, основанных на применении обобщенного дифференциала (см. §2.2.3), хотя даже для сравнительно простых и хорошо изученных классов задач эти алгоритмы оказываются достаточно сложными.

Если функционал $F(x^*(u),u)$ выпуклый на выпуклом множестве Ω, то необходимым условием того, что u^* его локальный минимум, является условие

$$\min_{\substack{w \in R \\ \|w\|=1}} \frac{\partial F}{\partial w} = 0,$$

где R – конус допустимых направлений на элементе $u^* \in \Omega$. Данный критерий в свою очередь равносилен условию

$$\partial F(x^*(u^*), u^*) \cap R^+(u^*) = \{o\},$$

использование которого базируется на следующих формулах.

1. Производная по направлению $\frac{\partial F}{\partial w} = \max_{l \in \partial F}(w, l)$.
2. Направление наискорейшего спуска для элемента $u \in \Omega$ есть $-\overline{w}$, где $\overline{w}$ – решение вспомогательной оптимизационной задачи: $\|\overline{w}\| = \min_{l \in \partial F(u)} \|l\|$.
3. Если $\Omega = E^n$, то условие оптимальности имеет вид

$$o \in \partial F(x^*(u^*), u^*).$$

Следует отметить, что если пункты 1, 3 и в большой степени 4 могут выполняться по стандартным правилам классических методов решения оптимизационных задач, то для получения оценок при выполнении пункта 2 оказывается необходимым использование специальных учитывающих отмеченные выше свойства зависимостей $x^*(u)$ процедур.

§ 5.2.3. Метод сглаживающих штрафных функций

В этом параграфе рассматривается подход, позволяющий единообразно преодолевать отмеченные выше затруднения и получать при помощи классических схем решения для достаточно широкого класса задач верхнего уровня.

Идея этого подхода заключается в замене зависимости $x^*(u)$ достаточно гладкой и определенной для всех $u \in \Omega \subseteq E^l$ и любых положительных τ функцией $\bar{x}(\tau, u)$, такой что $\lim_{r \to +0} \bar{x}(\tau, u) = x^*(u)$, причем в качестве аппроксимирующей зависимости $\bar{x}(\tau, u)$ предлагается использовать решение задачи нижнего уровня, получаемое по методу гладких штрафных функций §3.7.

Как было показано, решением задачи *нижнего уровня* (5.2.1.1) – (5.2.1.2) при использовании метода штрафных функций является

$$\bar{x}(\tau,u) = \arg\min_x A(\tau,x,u), \qquad (5.2.3.1)$$

где вспомогательная функция метода гладких штрафных функций выбирается следующего вида:

$$A(\tau,x,u) = f(x,u) + \sum_{s=1}^{m} P[\tau, y_s(x,u)] \qquad (5.2.3.2)$$

Если помимо стандартного свойства

$$\lim_{\tau\to+0} P(\tau,\alpha) = \begin{cases} +\infty, \alpha < 0, \\ 0, \alpha > 0 \end{cases}$$

достаточно гладкая штрафная функция $P(\tau,\alpha)$ удовлетворяет также и неравенствам

$$\frac{\partial P}{\partial \alpha} < 0 \quad \text{и} \quad \frac{\partial^2 P}{\partial \alpha^2} > 0, \forall\alpha, \qquad (5.2.3.3)$$

то $\bar{x}(\tau,u)$ – точка минимума вспомогательной функции A, которую можно найти из условия стационарности

$$\operatorname{grad}_x A(\tau,\bar{x},u) = o, \qquad (5.2.3.4)$$

удовлетворяет условиям сглаживания зависимости $x^*(u)$.

Теоретической основой предлагаемого подхода является использование локальных тейлоровских аппроксимаций зависимости $\bar{x}(\tau,u)$ и теоремы о неявных функциях. Действительно, если штрафная функция $P(\tau,\alpha)$ строго выпукла по α для любых $\tau > 0$ и, по крайней мере, дважды непрерывно дифференцируема по всем своим аргументам, то оказываются справедливыми следующие свойства.

1. Решение уравнений (5.2.3.4), являющихся условиями стационарности функции (5.2.3.2), $\bar{x}(\tau,u)$, существует и локально единственно для любых $u \in \Omega \subseteq E^l$ и $r > 0$, и потому зависимость $\bar{x}(\tau,u)$ является функциональной.
2. Для зависимости $\bar{x}(\tau,u)$ верно равенство
$$\lim_{\tau\to+0} \bar{x}(\tau,u) = x^*(u).$$
3. Если, наконец, функции $f(x,u)$ и $y_s(x,u)$ имеют непрерывные частные производные до второго порядка включительно, то

из теоремы о неявных функциях, примененной к условиям стационарности (5.2.3.4), следует, что функция $\bar{x}(\tau,u)$ непрерывно дифференцируема по всем своим аргументам.

Таблица 5.2.3.1.

	$\nu_1 < 0$	$\nu_1 = 0$	$\nu_1 > 0$
$\nu_2 < 0$	не существ.	$\xi_1^*(\nu_1,\nu_2) = 0$ $\xi_2^*(\nu_1,\nu_2) = 0$	$\xi_1^*(\nu_1,\nu_2) = 0$ $\xi_2^*(\nu_1,\nu_2) = 0$
$\nu_2 = 0$	не существ.	$\xi_1^*(\nu_1,\nu_2) = 0$ $\xi_2^*(\nu_1,\nu_2) = 0$	$\xi_1^*(\nu_1,\nu_2) = 0$ $\xi_2^*(\nu_1,\nu_2) \in [0,\nu_1]$
$\nu_2 > 0$	не существ.	$\xi_1^*(\nu_1,\nu_2) = 0$ $\xi_2^*(\nu_1,\nu_2) = 0$	$\xi_1^*(\nu_1,\nu_2) = 0$ $\xi_2^*(\nu_1,\nu_2) = \nu_1$

Продемонстрируем применение описанной сглаживающей процедуры на примере следующей задачи математического программирования.

Найти минимум по $\{\xi_1,\xi_2\}$ функции

$$f(\xi_1,\xi_2) = \xi_1 - 2\nu_2\xi_2 \qquad (5.2.3.5)$$

при условиях: $\xi_1 \ge 0,\ \nu_1 \ge \xi_2 \ge 0$

для произвольных значений параметров ν_1 и ν_2.

При фиксированных значениях параметров задача (5.2.3.5) является задачей линейного программирования, решение которой обладает всеми осложняющими особенностями, отмеченными в п. 2, приведено в таблице 5.2.3.1.

Отметим также, что при $\nu_1 \ge 0$ и $\forall\ \nu_2$ функция

$$f(\xi_1^*(\nu_1,\nu_2),\xi_2^*(\nu_1,\nu_2))$$

хотя и определена, но не является дифференцируемой. Воспользуемся описанной выше процедурой со штрафной функцией

$$P(\tau,\alpha)P = \tau\exp(-\alpha/\tau)\ .$$

Вспомогательная функция для задачи (5.2.3.5) будет иметь вид

$$A(\tau,\xi_1,\xi_2,\nu_1,\nu_2) = \\ = \xi_1 - 2\nu_2\xi_2 + \tau e^{-\frac{\xi_1}{\tau}} + re^{-\frac{\xi_2}{\tau}} + re^{-\frac{\nu_1-\xi_2}{\tau}}, \qquad (5.2.3.6)$$

а условия стационарности соответственно

$$\frac{\partial A}{\partial \xi_1} = 1 - e^{-\frac{\overline{\xi}_1}{\tau}} = 0,$$

$$\frac{\partial A}{\partial \xi_2} = -2\nu_2 - e^{-\frac{\overline{\xi}_2}{\tau}} + e^{-\frac{\nu_1-\overline{\xi}_2}{\tau}} = 0.$$

Откуда получаем $\overline{\xi}_1(\tau,\nu_1,\nu_2) = 0$ и

$$\overline{\xi}_2(\tau,\nu_1,\nu_2) = \nu_1 + \tau\cdot\ln\left(\nu_2 + \sqrt{\nu_2^2 + e^{-\frac{\nu_1}{\tau}}}\right),$$

которые определены и бесконечно дифференцируемы для любых значений параметров задачи (5.2.3.5). Графические представления зависимостей

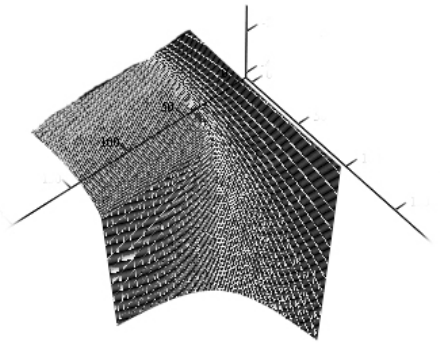


Рис. 5.2.3.1.

$A\left(\tau,\bar{\xi}_1(\tau,\nu_1,\nu_2),\bar{\xi}_2(\tau,\nu_1,\nu_2)\right)$ и $\bar{\xi}_2(\tau,\nu_1,\nu_2)$ приведены на рис. 5.2.3.2 и рис. 5.2.3.3 соответственно.

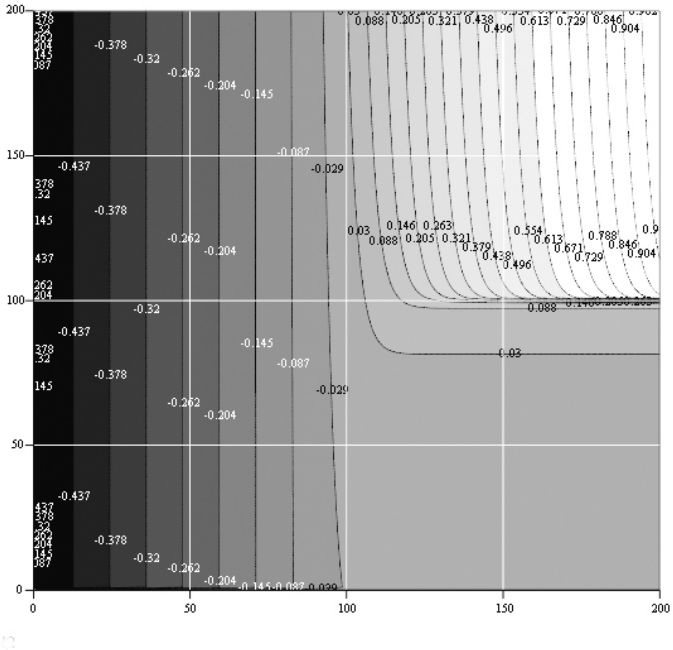


Рис. 5.2.3.2.

Двухуровневая схема "сглаживания" при решении задач параметрического программирования

Как было отмечено в разделе 3.7, основное свойство метода штрафных функций состоит в том, что

$$\lim_{\tau\to+0} A(\tau,\bar{x}(\tau,u),u) = f(x^*(u),u),$$

где $x^*(u)$ – решение задачи (5.2.3.1)–(5.2.3.2). Поэтому в качестве целевого функционала, итерационная оптимизация которого позволяет получить оценку решения задачи верхнего уровня, можно принять

$$A(\tau,\bar{x}(\tau,u),u).$$

Как было отмечено в п.2, основным затруднением при реализации процедуры

$$u_{k+1} = u_k + \sigma_k w_k,\ k = 0,1,2,\ldots \qquad (5.2.3.7)$$

является оценка w_k – направления улучшающей вариации и σ_k – величины шага по данному направлению, а главным преимуществом − возможность использования численных методов, не требующих явного аналитического представления для $A(\tau,u) = A(\tau,\bar{x}(\tau,u),u)$.

Далее, если решение задачи верхнего уровня сведено к оптимизации функционала $A(\tau,u)$, то при фиксированном $r > 0$ для определения w_k можно применить, например, метод Ньютона−Рафсона (см. §2.2.2).

Данный метод использует квадратичную тейлоровскую аппроксимацию, требующую значений как оптимизируемого функционала, так и его градиента и гессиана в точке $u_k \in \Omega \subset E^l$. Поскольку в данном случае функция $\bar{x}(\tau,u)$ определена неявно уравнениями

$$\frac{\partial A}{\partial \xi_i} = 0 \quad \text{или} \quad \frac{\partial f}{\partial \xi_i} + \sum_{s=1}^{m} \frac{\partial P}{\partial y_s}\frac{\partial y_s}{\partial \xi_i} = 0,\ \forall i = [1,n], \qquad (5.2.3.8)$$

то для компонент градиента $E(r,u)$ мы имеем

$$A'_{\nu_j} = \frac{\partial A}{\partial \nu_j} + \sum_{i=1}^{n} \frac{\partial A}{\partial \xi_i}\frac{\partial \bar{\xi}_i}{\partial \nu_j},\ \forall j = [1,l],$$

а в силу (5.2.3.8) окончательно получаем

$$E'_{\nu_j} = \frac{\partial \varepsilon}{\partial \nu_j},\ \forall j = [1,l]. \qquad (5.2.3.9)$$

Формула для компонент матрицы Гессе будет несколько более сложной:

$$A''_{\nu_j \nu_t} = \frac{\partial^2 A}{\partial \nu_j \partial \nu_t} + \sum_{i=1}^{n} \frac{\partial^2 A}{\partial \xi_i \partial \nu_j}\frac{\partial \bar{\xi}_i}{\partial \nu_t},\ \forall j,t = [1,l]. \qquad (5.2.3.10)$$

Необходимые для использования (5.2.3.10) значения $\frac{\partial \bar{\xi}_i}{\partial \nu_t},\ \forall i = [1,n],\ \forall j = [1,l]$ можно найти, решив систему линейных уравнений:

$$\sum_{i=1}^{n} \frac{\partial^2 A}{\partial \xi_i \partial \xi_j} \frac{\partial \overline{\xi}_i}{\partial \nu_t} = -\frac{\partial^2 A}{\partial \xi_j \partial \nu_t}, \ \forall j,t = [1,l], \qquad (5.2.3.11)$$

которая получается при дифференцировании (5.2.3.8) по всем компонентам $u \in E^l$.

Наконец, искомые оценки w_k и σ_k находим по формулам

$$\| w_k \| = -\| A'' \|^{-1} \| A' \| \quad \text{и} \quad \sigma_k = \arg\min_{\sigma>0} A(u_k + \sigma w_k).$$

Раздел 5.3. ЗАДАЧИ МНОГОКРИТЕРИАЛЬНОЙ ОПТИМИЗАЦИИ

Достаточно часто формализация отношений предпочтения для состояний моделируемого объекта порождает несколько независимых друг от друга целевых функционалов. По исторически сложившейся традиции в этом случае принято говорить о *задаче многокритериальной оптимизации*, хотя такой термин формально логически противоречив.

Многокритериальной мы будем называть задачу поиска в E^n экстремальных элементов функционалов $F_k(x)$ $x \in R \subset E^n$, $k = [1,K]$ на множестве элементов x, удовлетворяющих условиям вида:

$$f_j(x) \geq 0, \ j = [1,m].$$

Некорректность в общем случае подобной постановки задачи очевидна, поскольку экстремальный для одного из целевых функционалов элемент, вообще говоря, не является таковым для других. По этой причине набор независимых целевых функционалов $F_k(x), k = [1,K]$ заменяют одним, называемым *сверткой*, приходя таким образом к стандартной задаче математического программирования.

Наиболее простым примером свертки может служить задача максимизации линейной комбинации (с неотрицательными коэффициентами) исходных целевых функционалов

$$\sum_{k=1}^{K} w_k F_k(x)$$

в предположении, что каждый из них должен был максимизироваться.

Другим, менее тривиальным примером построения свертки, являются задачи поиска минимума *параметра рассогласования* κ для следующей многокритериальной модели:

в E^n *найти максимальные элементы*
функционалов $F_k(x)$ $x \in R \subset E^n$, $k = [1, K]$
и минимальные элементы
функционалов $Q_i(x), i = [K+1, K+M]$
на множестве элементов x, *удовлетворяющих условиям*:
$f_j(x) \geq 0$, $j = [1, m]$.

сводящиеся к задаче математического программирования вида:

для *абсолютной* оценки рассогласования
найти минимум κ *по совокупности* $\{\kappa, x\}$, *при условиях:*

$$\begin{aligned} &\kappa \geq 0, \\ &F_k(x) \geq F_k(x_k^*) - \kappa, \; k = [1, K], \\ &Q_i(x) \leq Q_i(x_i^*) + \kappa, \; i = [K+1, K+M], \\ &f_j(x) \geq 0, \; j = [1, m]; \end{aligned} \tag{5.3.1.}$$

для *относительной* оценки рассогласования
найти минимум κ *по совокупности* $\{\kappa, x\}$ *при условиях*:

$$\begin{aligned} &\kappa \geq 0, \; F_k(x) \geq (1-\kappa) F_k(x_k^*), \; k = [1, K], \\ &Q_i(x) \leq (1+\kappa) Q_i(x_i^*), \; i = [K+1, K+M], \\ &f_j(x) \geq 0, \; j = [1, m], \end{aligned} \tag{5.3.2.}$$

где $x^*_{\{k,i\}}$ – решения *вспомогательных задач* математического программирования с положительными оптимальными значениями целевых функционалов:

максимизировать $F_k(x)$
(*или, соответственно, минимизировать* $Q_i(x)$)
при условиях: $f_j(x) \geq 0$, $j = [1, m]$.

Решение задачи (5.3.1) будем обозначать $\{\kappa^w, x^w\}$, а решение задачи (5.3.2) – $\{\kappa^v, x^v\}$.

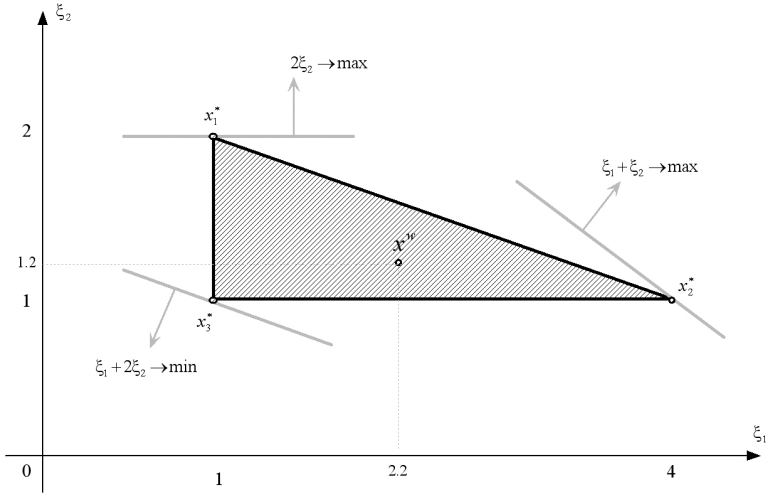


Рис. 5.3.1

Наконец, отметим, что при решении задач многокритериальной оптимизации достаточно часто оказывается возможным разделение множества всех допустимых элементов $\{ x \mid f_j(x) \geq 0,\ j = [1, m] \}$ на два подмножества, для первого из которых существуют допустимые вариации, улучшающие *все* целевые функционалы одновременно. В то время, как для второго подмножества при любой допустимой вариации имеется целевой функционал с ухудшающимся значением.

Определение 5.3.1. Множество допустимых элементов, для которых при любой допустимой вариации, улучшающей значение хотя бы одного целевого функционала, имеется целевой функционал с ухудшающимся значением, называется *множеством конкурентного равновесия по Парето*, или просто *паретовским множеством.*

Проиллюстрируем использование схем выбора параметра рассогласования на примере следующей многокритериальной модели.

Пусть в E^2 заданы целевые функционалы $F_1(x) = \xi_1 + \xi_2$, $F_2(x) = 2\xi_2$ и $Q_3(x) = \xi_1 + 2\xi_2$, которые надо оптимизировать при условиях:

$$\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7.$$

Соответствующие однокритериальные задачи легко решаются, и мы находим, что для задачи

$$F_1(x) = \xi_1 + \xi_2 \to \max$$

при условиях: $\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7$

$$\left\| x_1^* \right\| = \left\| \begin{matrix} 4 \\ 1 \end{matrix} \right\|, \text{ с } F_1(x_1^*) = 5.$$

Аналогично, для задачи $F_2(x) = 2\xi_2 \to \max$ при условиях: $\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7$

$$\left\| x_2^* \right\| = \left\| \begin{matrix} 1 \\ 2 \end{matrix} \right\|, \text{ с } F_2(x_2^*) = 4.$$

Наконец, для задачи $Q_3(x) = \xi_1 + 2\xi_2 \to \min$ при условиях: $\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7$

$$\left\| x_3^* \right\| = \left\| \begin{matrix} 1 \\ 1 \end{matrix} \right\|, \text{ с } Q\ (x_3^*) = 3.$$

Таким образом, задача определения наименьшего значения параметра рассогласования *абсолютного типа* принимает вид:

найти минимум κ по совокупности $\{\kappa, \xi_1, \xi_2\}$

при условиях:

$$\begin{aligned} &\kappa \geq 0, \\ &\xi_1 + \xi_2 \geq F_1(x_1^*) - \kappa, \\ &2\xi_2 \geq F_2(x_2^*) - \kappa, \\ &\xi_1 + 2\xi_2 \leq Q_3(x_3^*) + \kappa, \\ &\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7, \end{aligned}$$

или

$$\begin{aligned}
&\kappa \geq 0,\\
&\xi_1 + \xi_2 \geq 5 - \kappa,\\
&2\xi_2 \geq 4 - \kappa,\\
&\xi_1 + 2\xi_2 \leq 3 + \kappa,\\
&\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7,
\end{aligned} \tag{5.3.3}$$

решение которой будет $\left\| x^w \right\| = \begin{Vmatrix} 2.2 \\ 1.2 \end{Vmatrix}$ при $\kappa^w = 1.6$ и

$$F_1(x^w) = 3.4\ , F_2(x^w) = 2.4\ , \quad Q_3(x^w) = 4.6\,.$$

Соответственно задача определения наименьшего значения параметра рассогласования *относительного типа* принимает вид:

найти минимум κ по совокупности $\{\kappa, \xi_1, \xi_2\}$ при условиях:

$$\begin{aligned}
&\kappa \geq 0,\\
&\xi_1 + \xi_2 \geq (1 - \kappa) F_1(x_1^*),\\
&2\xi_2 \geq (1 - \kappa) F_2(x_2^*),\\
&\xi_1 + 2\xi_2 \leq (1 + \kappa) Q_3(x_3^*),\\
&\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7,
\end{aligned}$$

или

$$\begin{aligned}
&\kappa \geq 0,\\
&\xi_1 + \xi_2 \geq 5 - 5\kappa,\\
&2\xi_2 \geq 4 - 4\kappa,\\
&\xi_1 + 2\xi_2 \leq 3 + 3\kappa,\\
&\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7,
\end{aligned}$$

решение которой будет $\left\| x^v \right\| = \begin{Vmatrix} 1.8 \\ 1.2 \end{Vmatrix}$ при $\kappa^v = 0.4$ и

$$F_1(x^v) = 3\ , F_2(x^v) = 2.4\ , Q_3(x^v) = 4.2\,.$$

На рис. 5.3.1 приведена графическая интерпретация получаемых решений.

Покажите самостоятельно, что решение задачи минимизации параметра рассогласования (если оно существует) всегда находится на множестве Парето для исходной многокритериальной задачи, совпадающее в рассмотренном примере с допустимой областью

$$\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7.$$

В заключение отметим, что минимальное значение параметра рассогласования в общем случае зависит от формы множества Парето. Проиллюстрируем этот факт следующим примером.

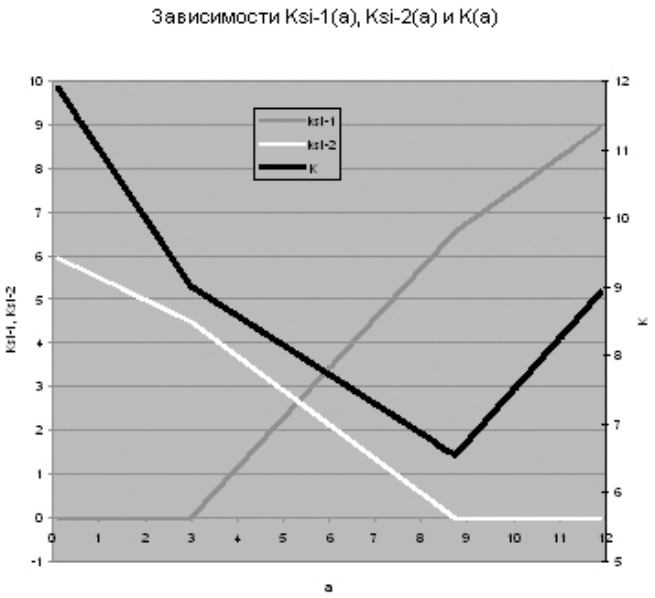


Рис. 5.3.2.

Пусть в E^2 заданы целевые функционалы $F_1(x) = 3\xi_1$, $F_2(x) = 2\xi_2$ и $Q_1(x) = \xi_1 + 2\xi_2$, которые требуется оптимизировать при условиях:

$$\xi_1 \geq 0, \quad \xi_2 \geq 0, \quad \xi_1 + \frac{a}{L-a}\xi_2 \leq a, \qquad (5.3.4)$$

где $0 < a < L$.

Геометрически множество допустимых состояний данной модели является прямоугольным треугольником с вершиной прямого угла в начале координат и катетами, лежащими на координатных осях. Сумма длин катетов постоянна и равна L, в то время, как длина катета, лежащего на оси $O\xi_1$, равна a. Оценим зависимость значения параметра рассогласования а от величины a.

Решения однокритериальных задач будут для данной модели иметь вид:

для задачи $F_1(x) = 3\xi_1 \to \max$ при условиях (5.3.4):

$$\left\| x_1^* \right\| = \left\| \begin{matrix} a \\ 0 \end{matrix} \right\|, \text{ с } F_1(x_1^*) = 3a;$$

для задачи $F_2(x) = 2\xi_2 \to \max$ при условиях (5.3.4):

$$\left\| x_2^* \right\| = \left\| \begin{matrix} 0 \\ L \end{matrix} \right\|, \text{ с } F_2(x_2^*) = 2(L-a);$$

и для задачи $Q_3(x) = \xi_1 + 2\xi_2 \to \min$ при условиях (5.3.4):

$$\left\| x_3^* \right\| = \left\| \begin{matrix} 0 \\ 0 \end{matrix} \right\|, \text{ с } Q_3(x_3^*) = 0.$$

Соответственно задача определения наименьшего значения параметра рассогласования по абсолютному изменению целевых функционалов в данном случае принимает вид:

найти минимум κ *по совокупности* $\{\kappa, \xi_1, \xi_2\}$

при условиях: $0 < a < L$

$$\kappa \geq 0,$$

$$3\xi_1 \geq 3a - \kappa,$$

$$2\xi_2 \geq 2(L-a) - \kappa,$$

$$\xi_1 + 2\xi_2 \leq \kappa,$$

$$\xi_1 \geq 0, \quad \xi_2 \geq 0, \quad \xi_1 + \frac{a}{L-a}\xi_2 \leq a,$$

для которой графики зависимостей $\xi_1(a), \xi_2(a), \kappa(a)$ приведены на рис. 5.3.2.

Как можно видеть, существует оптимальное значение a (а значит, и оптимальная форма множества Парето), при котором значение параметра рассогласования оказывается минимальным.

Глава 6

МАТЕМАТИЧЕСКИЕ МОДЕЛИ В E^n И ПРИНЦИПЫ ИХ ИСПОЛЬЗОВАНИЯ

Раздел 6.1. ОПРЕДЕЛЕНИЯ ОСНОВНЫХ ТЕРМИНОВ И ПОНЯТИЙ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Полезность математического моделирования заключается в возможности получения информации о свойствах и характере поведения изучаемого объекта без проведения (часто сложных или дорогостоящих) экспериментов в натуре, что может оправдывать затраты на преодоление затруднений, возникающих в процессе разработки или при попытках использования математических моделей.

Основное затруднение, преодоление которого оказывается необходимым в математическом моделировании, заключается в обеспечении *адекватности* этой модели исследуемому объекту. Другими словами, пользователю необходимо выяснить, насколько точно данная модель отражает реальную ситуацию и насколько надежные количественные оценки могут быть получены в процессе работы с этой моделью.

Опыт математического моделирования, накопленный в течение последних нескольких десятилетий, показывает, что проблема адекватности в ряде случаев может быть успешно преодолена. Примером служат многочисленные решенные физические и технические задачи. С другой стороны, попытки применения методов математического моделирования для исследования объектов социально-экономической природы приводят к заключению: несмотря на естественное желание учесть в модели все факторы, существенно влияющие на функционирование исследуемого объекта, добиться этого исключительно трудно, а иногда даже невозможно.

Если построение математической модели, учитывающей с достаточной степенью точности все факторы, являющиеся существенными для исследуемого объекта, оказывается невозможным, то следует отказаться от стандартной методологии ее использования и попытаться действовать иным способом, основанном на изменении постановок решаемых задач и включении пользователя в процесс поиска решений. Предметом дальнейшего рассмотрения в нашем курсе является описание альтернативной методологии математического моделирования, не требующей выполнения условия адекватности в полной мере.

Определение 6.1.1. Под *математической моделью* понимается формализованный (то есть, представленный в виде математических соотношений) набор правил, описывающих факторы, существенно влияющие на состояние и функционирование исследуемого объекта, и соответствующее этому набору информационное обеспечение.
Процесс построения и использования математической модели для решения с ее помощью конкретных задач принято называть *математическим моделированием*.

Описание математической модели выполняется в терминах количественных характеристик – *показателей* (*переменных*, *неизвестных*), значения которых подлежат определению и *параметров*, величины которых априорно известны.

Значения показателей, как правило, должны удовлетворять системе условий, задаваемых в виде ограничений, связей, целей и т.п., являющихся математической формулировкой существенных факторов.

Определение 6.1.2. Условия, выполнение которых безусловно необходимо, называются *обязательными* (*необходимыми*), тогда как условия, выполнение которых желательно, но необязательно, называются *целевыми* (*желательными*).

Как было отмечено во введении, в рамках настоящего курса рассматриваются только математические модели, описание которых выполняется при помощи конечного числа показателей и налагаемых на их значения условий.

Набор конкретных значений всех показателей и параметров математической модели называется *состоянием* модели. Набор условий (как обязательных, так и целевых) для конкретного списка показателей и значений параметров будем называть *задачей* для математической модели.

Те состояния модели, которые удовлетворяют набору условий задачи, являются ее *решением*. Процедура поиска значений показателей математической модели при заданных конкретных условиях и значениях ее параметров называется *процессом решения* задачи.

Определение 6.1.3. Состояние математической модели (или решение задачи),

- для которого не нарушено ни одно из обязательных ограничений или условий связи, называется *допустимым* или *совместным*.
- нарушающее хотя бы одно из обязательных ограничений или условий связи между показателями, называется *недопустимым* или *несовместным*.
- удовлетворяющее всем содержащимся в ее описании целевым условиям, называется *желательным* или *целевым*.

Вполне очевидно, что в общем случае может не существовать состояния математической модели, которое было бы одновременно и допустимым, и желательным. В этом случае достаточно часто возникает задача оценки степени нарушения целевых условий для различных ее допустимых состояний.

Определение 6.1.4. Допустимое состояние модели, в минимальной (в некотором, заранее определенном смысле) степени нарушающее содержащиеся в ее описании целевые условия, называется *оптимальным*.

Наконец, дадим

Определение 6.1.5. Математические модели, в которых учтены с приемлемой степенью точности все факторы, существенно влияющие на функционирование исследуемого объекта, назовем *полными*.
Неполными будем называть математические модели, в состав которых включены описания лишь некоторой части, существенно влияющих на функционирование исследуемого объекта факторов.

Прочие существенные факторы (не формализуемые с достаточной степенью точности или неизвестные вовсе) не учитываются в неполной модели.

Таким образом, предполагается, что все факторы, существенно влияющие на функционирование моделируемого объекта, можно разделить на три следующие группы:

- факторы, поддающиеся формализации с приемлемой степенью точности;
- факторы известные, но не формализуемые с необходимой точностью;
- факторы, существенно влияющие на характер функционирования исследуемого объекта, однако неучтенные или неизвестные вовсе;

в неполную математическую модель включаются только факторы, относящиеся к первой группе.

Раздел 6.2. СРАВНЕНИЕ ПОЛНЫХ И НЕПОЛНЫХ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Полные математические модели могут применяться для решения задач имитации (прогнозирования) функционирования или оптимизации характеристик исследуемого объекта или явления. Заметим, что попытки же имитирования поведения исследуемого объекта, построения прогноза, поиска его оптимальных или равновесных состояний для неполных математических моделей являются некорректными.

Некорректность использования методологии полного моделирования может быть обусловлена:

- неточностью математических описаний (формулировок) отдельных элементов модели, например из-за трудностей получения достоверных значений их параметров или неполнотой информационного обеспечения модели;
- некорректностью применения математического аппарата для формализованного описания некоторых из факторов, существенно влияющих на поведение моделируемого объекта;
- подозрением о наличии неизвестных существенных факторов.

В случае, когда нет возможности гарантировать адекватность модели, альтернативные корректные постановки задач могут быть получены, исходя из естественного требования достоверности решений, путем замены

традиционной проверки выполнения *достаточных* условий достоверности проверкой выполнения ее *необходимых* условий.

Например, в задачах имитации функционирования некоторого объекта выполнение достаточных условий достоверности позволяет ответить на вопрос: *Что произойдет в том случае, если...?* С другой стороны, выполнение необходимых условий достоверности гарантирует корректность лишь вопроса: *Чего не может произойти в том случае, если...?* Использование вопроса первого типа в случае работы с неполной моделью может приводить к недостоверному результату, в то время, как ответ на второй вопрос всегда будет достоверен.

Действительно, пусть было установлено, что одно из состояний некоторой модели допустимо, а другое – недопустимо. Если в эту модель включить дополнительное условие (неучтенное или неизвестное в момент получения решений), то решение, идентифицированное ранее как недопустимое, таковым и останется. Допустимое же решение после пополнения рассматриваемой модели, вообще говоря, допустимым уже может и не быть. Отсюда следует, что ответ на вопрос первого рода достоверен лишь для полных математических моделей, а ответ на вопрос второго рода достоверен как для полных, так и для неполных моделей.

Аналогичная ситуация имеет место и для оптимизационных задач. Поясним это несложным примером. Допустим, что требуется распределить поровну между десятью сотрудниками некоторой фирмы сумму в 10 000 рублей так, чтобы каждый получил как можно больше.

Решение, при котором на одного сотрудника приходится по 1500 рублей, является недопустимым, а в то время, как решение: выдать каждому по 1000 рублей – оптимальным. Если же теперь в постановку задачи добавить условие, заключающееся в необходимости выплаты 15-ти процентного налога, то решение о выплате каждому из сотрудников по 1000 рублей уже не будет оптимальным – оно станет недопустимым. Первое же решение (о выплате по 1500 рублей) осталось недопустимым. Оно является таковым независимо от факта, было ли учтено в модели условие выплаты налога или нет.

Таким образом, в силу вышеизложенного, процедура решения задач при помощи неполных моделей может быть разделена на два этапа:

- на первом – в автоматическом режиме все состояния неполной математической модели разделяются на два множества: множество недопустимых состояний и множество всех остальных, последнее из которых будем называть *множеством условно допустимых состояний*, ибо, строго говоря, они лишь "подозреваются на допустимость";

- затем пользователь анализирует это "подозрительное" множество (привлекая в случае необходимости не формализуемые или даже интуитивные соображения), сужает его, пополняя модель новыми обязательными условиями.

Процедурная пара "работа ЭВМ" – "анализ пользователя" может применяться при необходимости неоднократно до тех пор, пока пользователь не обнаружит состояние, которое он сочтет "приемлемым решением" или же установит факт отсутствия такового.

Основное преимущество этой схемы решения состоит в том, что пользователь (а не ЭВМ!) выбирает состояние модели, принимаемое за решение. ЭВМ лишь контролирует его

выбор, не позволяя пользователю осуществлять данный выбор на множестве недопустимых состояний.

Таким образом, если в случае полной модели пользователь получает автоматически рассчитанное компьютером окончательное решение в виде оптимального вектора, варианта прогноза или имитирующей траектории, то для неполных моделей функция ЭВМ заключается лишь в отделении множества состояний модели, идентифицированных как недопустимые. Роль пользователя неполной модели заключается в последующем уточнении результата (возможно, до полной однозначности), исходя из собственных, не обязательно формализуемых или, быть может, даже интуитивных соображений. Нужно заметить, что в традиционной практике математического моделирования, основанной на применении полных моделей, достаточно часто автоматически находимое решение вызывает у пользователя (явно или неявно) негативное отношение, порождаемое именно претензией модели на учет всех существенных факторов и, как следствие, претензией решения на оптимальность. Здравый смысл подсказывает, что для объектов социально-экономической природы, например, не следует полагаться только на формальные методы и не стоит пренебрегать опытом и интуицией пользователя, но и даже предпочтительно поощрять использование последних.

Схема неполного моделирования в этом смысле выглядит существенно более привлекательной, ибо ЭВМ (при помощи математических методов) всего лишь сужает множество состояний модели, из которых пользователь должен сделать выбор окончательного решения. Иными словами, программно-аппаратный комплекс не имеет своей функцией поиск окончательного решения вместо пользователя, а только предохраняет последнего от возможных количественных ошибок в процессе учета существенных факторов, поддающихся формализации.

Раздел 6.3. ИНТЕРАКТИВНЫЙ ПРОЦЕСС РЕШЕНИЯ ЗАДАЧ ДЛЯ НЕПОЛНЫХ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

§6.3.1. Процедура сужения множества условно допустимых состояний

Сужение множества условно допустимых состояний для неполной математической модели состоит в пополнении набора формализуемых ограничений некоторыми дополнительными условиями. Процедуру включения в математическую модель некоторого числа дополнительных условий (связей, ограничений и т.п.) будем называть *пополнением* этой модели.

Пополнение, очевидно, возможно как для полных, так и для неполных моделей. Однако, отличие этих случаев состоит в том, что включение в полную математическую модель некоторого добавочного условия (ограничения, связи и т.п.) может превратить решение, идентифицированное как допустимое (или оптимальное) в недопустимое, в то время как решение, определенное в неполной модели как недопустимое, останется таковым в случае включения в нее любого дополнительного условия или ограничения.

Следует отличать пополнение математической модели от ее *модификации*. Последняя допускает изменение формулировок отдельных условий, уже входящих в описание исходной модели. Пополнение же заключается во включении в состав модели некоторого дополнительного условия без какого-либо изменения других ее элементов. Модификация математической модели часто оказывается необходимой как на этапе разработки, так и в процессе ее эксплуатации, однако результат модификации всегда должен рассматриваться как некоторая новая неполная модель. В равной степени это относится и к удалению какого-либо условия из модели.

§6.3.2. Общая схема решения задач для неполных математических моделей

Резюмируя все изложенное выше, можно сказать, что методологическая основа неполного моделирования заключается в сочетании автоматического анализа неполных моделей и вовлечения пользователя непосредственно в процесс поиска решения.

В целом процесс решения разбивается на шаги – итерации, для каждой из которых ЭВМ выполняет количественный анализ неполной математической модели, отыскивая некоторое ее состояние (решение задачи), удовлетворяющее всей совокупности обязательных условий, включенных пользователем в формализованное описание на данной итерации, после чего пользователь осуществляет пополнение или модификацию модели.

Пополнение или модификация выполняются пользователем на основе неформализуемых критериев или оценок до тех пор, пока не будет получено состояние, идентифицируемое как приемлемое решение, или же установлен факт отсутствия такого решения.

Отметим, что при этом центральной технической проблемой оказывается задача отделения множества недопустимых состояний модели.

Для успешного использования схемы неполного моделирования на практике необходимо преодолеть два технических затруднения, возникающих в процессе анализа множества состояний модели, которые не были отнесены к недопустимым (то есть множества условно допустимых состояний):

- во-первых, следует решить проблему представления этого множества в ЭВМ в виде, удобном для анализа,
- во-вторых, необходимо предусмотреть возможность возникновения ситуации, когда анализируемое множество оказывается пустым.

Известно, что в компьютере достаточно легко представляются лишь отдельные состояния математической модели, например, последовательным перечислением значений всех ее показателей. В случае полной модели такое ограничение не имеет принципиального характера, поскольку автоматически рассчитываемое оптимальное решение или имитационный прогноз обычно являются именно конкретным состоянием модели. Однако в схемах неполного моделирования возможность анализа множества условно допустимых состояний в целом является принципиально необходимой. Пользователь должен иметь в своем распоряжении инструмент, позволяющий выполнять такой анализ для случая множеств достаточно сложной структуры.

Отмеченное затруднение можно преодолеть, воспользовавшись специальным приемом, который заключается в применении *метрики* – способа количественной оценки близости множеств допустимых и целевых состояний неполной модели.

Как и множество допустимых состояний, множество желательных состояний задается пользователем в математической модели набором условий, связей или ограничений, налагаемых на значения ее показателей и

отражающих цели, приоритеты или же отношения предпочтения пользователя.

В процессе автоматической работы ЭВМ не только отделяет множество недопустимых состояний, но и находит среди оставшихся (условно допустимых) состояний решение, наименее "удаленное" от множества желательных состояний.

Применение метрики, дающей количественную оценку степени близости множеств допустимых и целевых состояний позволяет пользователю на каждом шаге работы с математической моделью ограничиваться анализом лишь отдельной точки множества допустимых состояний, не рассматривая это множество целиком. Свобода выбора состояний из множества допустимых состояний может быть реализована пользователем при помощи пополнения или коррекции наборов обязательных и/или целевых условий математической модели.

При этом предполагается, что пользователь может сформулировать свою оценку текущего значения каждого из показателей модели, используя, быть может, не формализуемые или даже интуитивные критерии в одном из следующих видов:

- значение показателя приемлемо;
- значение показателя желательно изменить;
- значение показателя необходимо изменить.

Если значения всех показателей признаны пользователем приемлемыми, то решение получено. В противном случае необходимо продолжить пополнение неполной математической модели или ее корректирование.

Необходимость пополнения модели обуславливается не наличием каких-либо ошибок (хотя такую возможность исключать не следует), а принципиальной неполнотой анализируемой модели, которая выявляется, как правило, лишь в процессе решения задач.

Наиболее простой и эффективный способ пополнения математической модели состоит в изменении множества допустимых состояний, заключающемся во введении новых условий, связей или ограничений, то есть просто в сужении этого множества.

Если в результате включения дополнительных условий множество допустимых состояний оказалось непустым, то пополнение считается завершенным успешно. Однако достаточно часто добавление новых условий или связей превращает систему формализованных обязательных условий в противоречивую, то есть в систему, для которой не существует ни одного допустимого состояния.

В этом случае пользователю следует вернуться к предыдущему шагу процесса поиска решения и выполнить пополнение не во множестве *допустимых*, а для *желательных* состояний. Найденное значение метрики покажет величину возникшего в процессе пополнения рассогласования обязательных условий модели.

Последовательно выполняемое пополнение математической модели приводит или к выявлению полностью приемлемого состояния, или же устанавливает факт его отсутствия.

Глава 7
ЛИНЕЙНЫЕ НЕПОЛНЫЕ МОДЕЛИ

Раздел 7.1. ОПИСАНИЕ ЛИНЕЙНЫХ НЕПОЛНЫХ МОДЕЛЕЙ

§7.1.1. Структура линейной неполной математической модели

Основой описания неполной математической модели является упорядоченный n-компонентный *список показателей* – величин, значения которых характеризуют состояние модели. (Термин "показатель" может быть заменен словами *переменная* или *неизвестная*).

Помимо значений показателей описание математической модели включает также дополнительные числовые характеристики этих показателей – *атрибуты*, частично задаваемых пользователем в качестве "исходных данных" и частично рассчитываемых в автоматическом режиме как "решение"[10].

Факторы, включаемые пользователем в неполную математическую модель, могут формализоваться при помощи:

- *линейных ограничений* типа "равенство" или "неравенство", налагаемых на произвольные линейные комбинации значений показателей;
- операции поиска минимума или максимума этих комбинаций.

[10] Определение каждого типа атрибутов приводится ниже в данном параграфе.

К атрибутам, задаваемым пользователем, относятся:

- границы обязательного (допустимого) диапазона значений показателей;
- границы желательного (целевого) диапазона значений показателей;
- коэффициенты ранжирования границ желательного диапазона;
- коэффициенты линейных зависимостей между показателями.

К автоматически рассчитываемым атрибутам относятся:

- значения показателей;
- величины нарушения границ обязательного диапазона;
- двойственные оценки для границ обязательного диапазона;
- величины нарушения границ диапазона желательных значений;
- индекс группы целевых границ.

Кроме того, в неполной модели допускается использование некоторых вспомогательных характеристик (как задаваемых пользователем, так и вычисляемых автоматически), относящихся, однако, не к отдельным показателям, а ко всей математической модели в целом.

§7.1.2. Обязательные ограничения и связи

Одной из главных количественных характеристик каждого показателя математической модели являются границы диапазона его *допустимых значений*. Конкретно диапазон допустимых значений определяется заданием нижней и верхней границ этого диапазона.

В режиме автоматического расчета значений показателей в первую очередь выполняется поиск такого состояния модели, для которого ни одна из границ обязательного диапазона не оказывается нарушенной. Такие состояния модели мы назвали *допустимыми* (или *совместными*). Если же таких состояний не обнаруживается, то автоматическая обработка модели завершается. Состояния модели, в которых нарушена хотя бы одна граница допустимого диапазона, называются *недопустимыми* (или *несовместными*).

Формально множество допустимых значений показателей модели определяется системой двусторонних ограничений следующего вида. Пусть общее число показателей равно n, а значение показателя с номером k обозначается ξ_k, где $k = [1, n]$, то есть k принимает значения от 1 до n.

Обозначим через d_k и D_k соответственно *величины нижних и верхних границ допустимого диапазона значений* k-го показателя ξ_k. Тогда множество допустимых состояний модели будет задаваться системой неравенств: $d_k \le \xi_k \le D_k$, где $k = [1, n]$.

Значения границ допустимых (обязательных) диапазонов величин показателей могут быть любыми вещественными числами, удовлетворяющими неравенствам: $d_k \le D_k$, $k = [1, n]$. Данное условие означает, что верхняя граница допустимого диапазона не может быть меньше нижней, ибо это автоматически означает отсутствие допустимых состояний модели. Равенство верхней и нижней границ возможно, однако в этом случае показатель сможет принимать лишь единственное значение. Безусловное требование, чтобы нижние границы диапазонов значений показателей модели не превосходили верхних, очевидно не является существенным ограничением для пользователя.

Часть границ допустимых (обязательных) диапазонов может отсутствовать или же быть установленной "по умолчанию". Если значение некоторого показателя не ограничено сверху, то предполагается равенство этой границы "плюс бесконечности", моделируемой достаточно большим положительным числом. Аналогично, в случае отсутствия нижней границы предполагается ее равенство "минус бесконечности", представляемой достаточно большим по абсолютной величине отрицательным числом.

По умолчанию для всех показателей модели нижние границы допустимого диапазона устанавливаются равными нулю, а верхние предполагаются отсутствующими. Таким образом, пользователю необходимо указывать в модели допустимые границы лишь в тех случаях, когда эти границы реально существуют.

Показатели в неполной математической модели могут быть связаны друг с другом *линейными* зависимостями. Условие линейности означает, что зависимости между величинами показателей являются линейными однородными функциями, то есть один показатель представим в виде суммы некоторых других показателей, умноженных на некоторые постоянные коэффициенты.

По соображениям повышения практической эффективности каждый показатель модели может явно зависеть только лишь от стоящих ранее него в списке показателей, то есть от показателей, номера которых меньше, чем номер данного показателя. Это означает, что линейная функция, выражающая зависимость между показателями, имеет следующий вид:

$$\xi_k = \sum_{i=1}^{k-1} A_{ki}\xi_i, \text{ где } k = [2, n],$$

где величины A_{ki} являются фиксированными *коэффициентами линейных связей между показателями* рассматриваемой модели.

Введенное ограничение на упорядоченность списка показателей не является излишне жестким, поскольку любая линейная система уравнений может быть приведена к требуемой форме записи путем введения дополнительных показателей и, возможно, изменением порядка их следования в списке. Для пользователя это может иногда являться причиной некоторого неудобства, однако, выигрыш от достигаемого таким образом улучшения структурированности модели, повышения надежности и эффективности системы в целом, как показывает практика, вполне себя оправдывает.

Пользователю рекомендуется выполнять структуризацию постановки задачи, размещая в начале списка определения всех показателей и относя описание связей между ними в конец. Например, уравнение вида $x = y$ можно ввести в модель при помощи трех показателей с именами x, y и f, задав связь между ними по формуле $f = x - y$ и установив для показателя f нулевые обязательные границы.

Отметим, что выполнение равенств связи между показателями обеспечивается независимо от того, является ли состояние модели совместным или нет. Наконец, для упрощения ввода данных значения коэффициентов в уравнениях связи полагаются по умолчанию равными нулю и, следовательно, пользователю достаточно задать лишь ненулевые коэффициенты функций связи между показателями модели.

§7.1.3. Целевые ограничения

При помощи целевых диапазонов в математической модели формализуются предпочтения и приоритеты пользователя. Причем их существенное отличие от обязательных диапазонов заключается в том, что формализуемые цели и предпочтения могут конфликтовать как друг с другом,

так и с обязательными условиями, то есть быть логически противоречивыми. Иначе говоря, пользователь может задавать границы, определяющие множество *желательных* (*целевых*) состояний модели. Как и в случае допустимого диапазона, целевой диапазон определяется заданием его нижней и верхней границ для каждого показателя.

Формально множество целевых значений показателей модели определяется системой двусторонних ограничений типа неравенств следующего вида.

Обозначим через r_k и R_k соответственно *величины нижних и верхних границ целевого диапазона значений* k-го показателя ξ_k. Тогда множество целевых состояний модели будет задаваться системой неравенств: $r_k \le \xi_k \le R_k$, где $k = [1, n]$. Значения границ допустимых (обязательных) диапазонов значений показателей могут быть любыми вещественными числами.

Для значений границ целевых диапазонов показателей также предполагается выполнение условий: $r_k \le R_k$, где $k = [1, n]$. Последнее условие не представляется излишне жестким, поскольку маловероятно, чтобы здравомыслящий пользователь стремился найти значение показателя, удовлетворяющее одновременно двум, явно противоречащим друг другу, критериям.

Заметим, что нарушение границ желательных диапазонов возможно и для совместных состояний модели. Однако эти нарушения сводятся к минимуму по правилам, которые будут разъяснены ниже.

Как и в случае задания допустимых границ, часть границ желательных диапазонов может отсутствовать или же быть присвоенной по умолчанию. По умолчанию для всех показателей модели как нижние, так и верхние границы целевого диапазона предполагаются отсутствующими. То есть пользователю необходимо указывать в модели целевые границы лишь в тех случаях, когда они реально существуют.

§7.1.4. Расстояние между множествами допустимых и целевых состояний

Процедура решения задач для конкретной неполной математической модели обычно состоит из нескольких шагов, на каждом из которых пользователь получает результат автоматического расчета, анализирует его и выполняет необходимое пополнение или коррекцию этой модели.

Согласно методологии неполного моделирования главной задачей на этапе автоматического расчета является отделение недопустимых состояний математической модели. Множество оставшихся (условно допустимых) состояний пользователь должен проанализировать, исходя из собственных неформализуемых и, быть может, даже интуитивных соображений, имея целью этого анализа поиск приемлемых состояний модели.

Условие линейности зависимостей между показателями означает, что анализируемое множество будет выпуклым многогранником в линейном n-мерном пространстве, формализованное представление которого в ЭВМ является весьма сложной задачей. Как было отмечено ранее, для облегчения работы пользователя применена специальная схема, позволяющая анализировать лишь одно решение из множества условно допустимых состояний, не теряя, однако, при этом возможности переходить, в случае необходимости, к другим условно допустимым решениям.

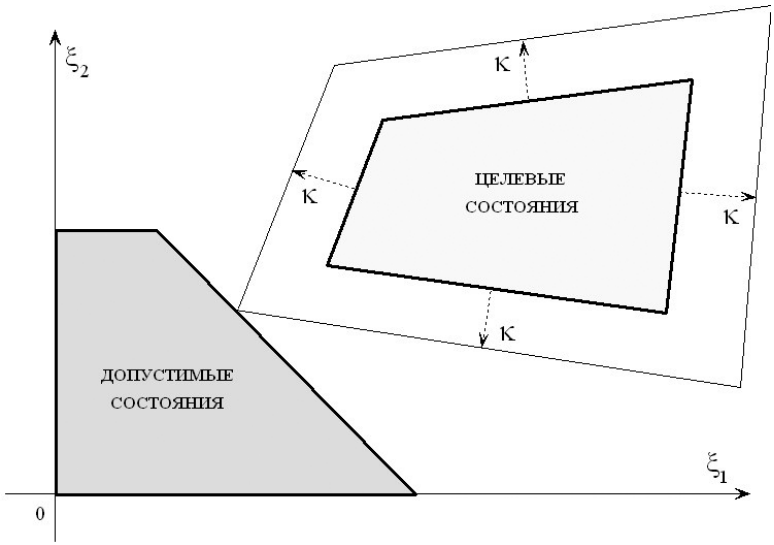


Рис. 7.1.4.1. Оценка степени близости множеств допустимых и целевых состояний в неполной математической модели.

Более конкретно: в режиме автоматического расчета значений показателей находится условно допустимое состояние неполной модели, наименее удаленное от множества целевых состояний. Иначе говоря, из всех состояний, не нарушающих ни одной границы допустимых диапазонов

значений показателей, выбирается состояние, минимально нарушающее границы целевых диапазонов.

На практике используются два различных метода оценки расстояния между множествами, называемые соответственно *абсолютной* и *относительной метриками*.

Известно, что способы оценки близости двух множеств могут отличаться от обычного метода измерения геометрических расстояний.

В рассматриваемом случае количественной характеристикой "близости" двух множеств допустимых и целевых состояний служит минимально возможная величина расширения всех целевых диапазонов, для которой найдется хотя бы одно состояние модели, принадлежащее множествам допустимых и целевых состояний одновременно. Рис. 7.1.4.1 иллюстрирует используемый способ оценки близости множеств, называемый *абсолютной метрикой*.

Приведем математическую формулировку способа измерения абсолютной метрики – количественной оценки близости множеств допустимых и целевых состояний модели. За величину абсолютной метрики принимается оптимальное значение целевой функции следующей задачи:

минимизировать κ (*по совокупности* κ *и всех* ξ_i, *где* $i = [1, n]$),

при условиях:

$$\kappa \geq 0,\ d_k \leq \xi_k \leq D_k,\quad k = [1, n],$$
$$r_k - \kappa \leq \xi_k \leq R_k + \kappa,\quad k = [1, n], \qquad (7.1.4.1)$$
$$\xi_k = \sum_{i=1}^{k-1} A_{ki}\xi_i,\quad k = [2, n].$$

Решение этой задачи (точнее, значение κ) показывает какое минимальное расширение границ целевых диапазонов модели необходимо для появления хотя бы одного общего элемента у множеств допустимых и целевых состояний.

Относительная метрика определяется аналогично абсолютной с тем единственным отличием, что "близость" множеств допустимых и целевых состояний оценивается в долях от абсолютных величин значений границ целевых диапазонов. Задача поиска величины относительной метрики имеет вид:

минимизировать κ (*по совокупности* κ *и всех* ξ_i, *где* $i = [1, n]$),

при условиях:

$$\kappa \geq 0,\ d_k \leq \xi_k \leq D_k,\quad k = [1, n],$$

$$r_k - \kappa \mid r_k \mid \le x_k \le R_k + \kappa \mid R_k \mid, \quad k = [1, n], \quad (7.1.4.2)$$

$$\xi_k = \sum_{i=1}^{k-1} A_{ki} \xi_i, \quad k = [2, n].$$

Относительная метрика оценивает минимальное относительное изменение границ желательных диапазонов для показателей модели, при котором, по крайней мере, одно из допустимых состояний оказывается принадлежащим также и множеству целевых состояний.

Заметим, что задачи (7.1.4.1) и (7.1.4.2) можно рассматривать как частный случай соответственно задач (5.4.1) и (5.4.2), решения которых позволяют найти состояния на множестве Парето, наилучшим образом согласованные по различным целевым функционалам многокритериальной модели.

Использование абсолютной и относительной метрик, а не обычного для геометрии евклидового расстояния, обусловлено сравнительной простотой их содержательной интерпретации в задачах исследования объектов социально-экономической природы. Например, относительная метрика может трактоваться как количественная мера структурной перестройки объекта, необходимой для достижения поставленных целей.

В заключение заметим, что из приведенного определения способа измерения метрики следует, что:

- при использовании относительной метрики ни одна из границ целевых диапазонов показателей модели не может иметь нулевого значения;
- как абсолютная, так и относительная метрики имеют смысл лишь для совместных состояний математической модели.

Раздел 7.2. АНАЛИЗ РЕШЕНИЙ, ПОЛУЧАЕМЫХ ПРИ ПОМОЩИ НЕПОЛНЫХ МОДЕЛЕЙ

§7.2.1. Сопоставление величин нарушения допустимых границ

Описанный выше способ измерения расстояния между множествами допустимых и целевых состояний предполагает, что решаемая задача совместна, то есть существует хотя бы один набор значений показателей неполной математической модели, удовлетворяющий всем обязательным

границам и условиям связи. Иначе говоря, ограничения и связи, определяющие множество допустимых состояний, не должны быть противоречивыми.

Часто на практике это условие не выполняется, и пользователь должен иметь возможность выявлять возникающее в модели противоречие и определять причину его возникновения.

Этой цели служит вывод вспомогательной информации, облегчающей пользователю анализ и разрешение конфликтов для несовместных моделей. В состав этих данных включены *величины нарушения обязательных границ* и значения *двойственных оценок* неравенств, задающих границы множества допустимых состояний.

Величины нарушения нижней и верхней обязательных границ y_k и Y_k вычисляются соответственно по формулам:

$$y_k = d_k - \xi_k \text{ и } Y_k = \xi_k - D_k \text{ для всех } k = [1, n],$$

причем в этих формулах значения величин y_k и Y_k заменяются нулями в том случае, если правые части равенств имеют отрицательное значение.

Значения двойственных оценок g_k и G_k для нижних и верхних обязательных границ соответственно находятся из решения двойственной задачи при помощи симплекс-метода.

В силу соотношений *дополняющей нежесткости* (см. §4.3.2), ненулевые значения величин нарушения границ обязательных диапазонов и ненулевые двойственные оценки будут иметь место лишь для активных ограничений как совместных, так и несовместных состояний математической модели.

В процессе анализа решения можно пользоваться значениями двойственных оценок как для идентификации условий, порождающих противоречивость модели и последующего ее устранения, так и для констатации факта отсутствия условно допустимых состояний на некотором шаге процесса пополнения неполной модели.

§7.2.2. Группировка целевых границ

Используемая в рассматриваемом подходе метрика обладает специфической особенностью: для оптимального значения κ, находимого в процессе решения задач оценки значения метрики, компоненты вектора x^*, определяющие состояние неполной математической модели, могут быть неоднозначными, то есть иметь не единственные компоненты.

Подобная ситуация, неприятная для пользователя тем, что значения некоторых показателей модели могут принять бессмысленные с практической точки зрения значения, возникает в тех случаях, когда число "расширенных" целевых ограничений, определяющих минимальное значение κ, оказывается меньше разности между n (размерностью линейного пространства, в котором сформулирована математическая модель исследуемого объекта) и числом активных ограничений.
При этом пользователь, конечно, имеет возможность пополнить модель и добиться устранения данного эффекта за счет введения дополнительных связей или ограничений, однако практически это может потребовать значительных затрат усилий и времени.

Для упрощения процесса анализа неполной математической модели возможно применение специальной процедуры, позволяющей устранить отмеченный выше дефект используемой метрики – находить состояния с *однозначно определяемыми* величинами показателей.

Эта процедура, называемая *группировкой границ целевых диапазонов*, заключается в последовательном решении задач нахождения величины метрики. На каждом шаге данной процедуры из условия задачи (7.1.4.1) исключаются ограничения, содержащие переменную κ путем превращения ее в константу в тех активных неравенствах, которые определяют оптимальное значение метрики. Отметим, что все приводимые ниже рассуждения в равной степени справедливы как для абсолютной, так и для относительной метрик.

Рассматриваемая процедура группировки целей состоит в следующем. Допустим, что в результате решения задачи получено значение метрики $\kappa = \kappa_1$, зависящее лишь от *активных* ограничений задачи. Тогда значения показателей неполной модели могут определяться, вообще говоря, неоднозначно, например, зависеть от алгоритма решения, от выбора начальной точки и т. п.

Для устранения возникшей неоднозначности следует выполнить фиксацию переменной κ (т.е. превращение ее в константу со значением κ_1) во всех ограничениях, оказавшихся активными на данном шаге процедуры. Затем процесс решения задачи повторяется.

При этом находится новое значение метрики $\kappa = \kappa_2 < \kappa_1$ и снова выделяется множество активных ограничений, в которых переменная κ полагается равной постоянной κ_2 и так далее. Ясно, что число ограничений, содержащих κ, должно при этом уменьшиться.

Процесс последовательной фиксации завершается, если:

- либо все ограничения, содержащие κ, оказались зафиксированными;
- либо на некотором шаге процедуры последовательной фиксации значение κ оказалось равным нулю.

Первый случай означает, что число активных ограничений достигло возможного для данной задачи максимума и решение однозначно. Во втором случае оказывается, что для оставшихся (незафиксированных) ограничений, содержащих κ, изменение границ для достижения общей точки множеств допустимых и целевых состояний не требуется.

Остающаяся во втором случае неоднозначность решения не должна представлять методологической проблемы для пользователя, поскольку неоднозначно определяемые значения показателей находятся в желательных диапазонах и, следовательно, содержательно интерпретируемы.

В итоге процедура группировки выполняет разбиение множества желательных ограничений на группы (отсюда – ее название), каждая из которых обладает своей собственной количественной оценкой "близости" целей и возможностей. Для большей наглядности эти величины именуются "расстоянием".

Строго говоря, метрикой, определяющей расстояние между множествами допустимых и целевых состояний, является не значение переменной κ, а набор неотрицательных, монотонно убывающих чисел $\kappa_1, \kappa_2, \kappa_3, ..., \kappa_T$, где T есть число групп фиксации, каждой из которых соответствует свое расстояние до цели.

Раздел 7.3. СРЕДСТВА УПРАВЛЕНИЯ ПРОЦЕССОМ РЕШЕНИЯ ЗАДАЧ ДЛЯ НЕПОЛНЫХ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Проиллюстрируем возникновение неоднозначности решения и ее устранение путем процедуры группировки следующим несложным примером. (Пусть для большей простоты и наглядности рассматриваемая модель зависимостей между показателями не содержит).

Рассмотрим математическую модель с двумя показателями ξ_1 и ξ_2, диапазоны допустимых значений которых описывается условиями:

$$0 \le \xi_1 \le 3,$$

$$0 \le \xi_2 \le 5,$$

а множество целевых состояний – условиями:

$$5 \le \xi_1 \le 11,$$

$$6 \le \xi_2 \le 8,$$

тогда задача нахождения значения метрики принимает вид:

минимизировать κ (*по совокупности* κ *и всех* ξ_1, ξ_2) *при условиях*:

$$\kappa \ge 0,$$

$$0 \le \xi_1 \le 3,$$

$$0 \le \xi_2 \le 5,$$

$$5 - \kappa \le \xi_1 \le 11 + \kappa,$$

$$6 - \kappa \le \xi_2 \le 8 + \kappa .$$

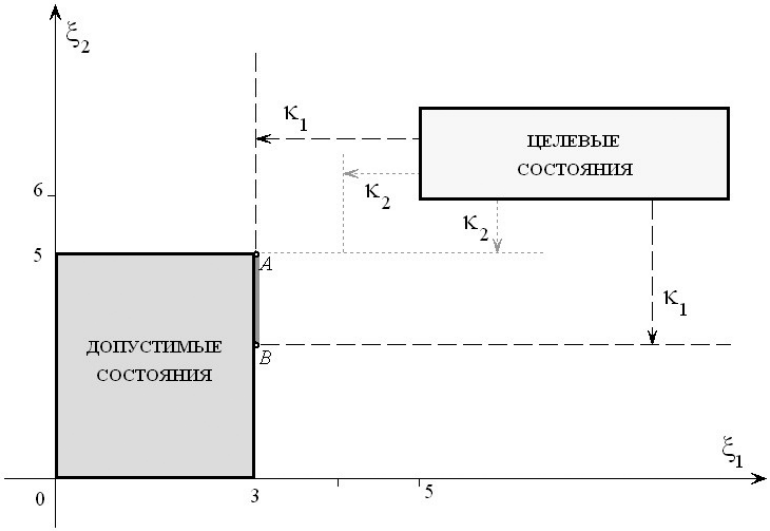


Рис. 7.3.1. Процедура группировки целевых ограничений.

Эта задача легко решается графически, и из рис. 7.3.1 мы находим очевидный ответ: $\kappa = 2$, $\xi_1 = 3$, $\xi_2 = [4,5]$. Заметим, что переменная ξ_2 определяется в этом случае *неоднозначно* и соответствующее множество точек на рисунке отмечено утолщенным отрезком *AB*.

На следующем шаге процедура группировки целевых границ требует замену переменной κ константой 2 в активных ограничениях задачи. Такое ограничение в рассматриваемом случае единственное: $5 - \kappa \le \xi_1$. Выполнив данную подстановку, получаем, что на втором шаге процедуры необходимо решать задачу вида:

минимизировать κ (*по совокупности* κ *и всех* ξ_1, ξ_2) *при условиях*:

$$\begin{aligned} &\kappa \ge 0, \\ &0 \le \xi_1 \le 3, \\ &0 \le \xi_2 \le 5, \\ &3 \le \xi_1 \le 11 + \kappa, \\ &6 - \kappa \le \xi_2 \le 8 + \kappa . \end{aligned} \qquad (7.3.1)$$

Решение и этой задачи также находится графически, оно имеет вид: $\kappa = 1$, $\xi_1 = 3$, $\xi_2 = 5$.

Полученное решение определено однозначно, и поэтому процесс группировки закончен. В приведенном примере разбиение системы ограничений, описывающих множество целевых состояний на группы по их "удаленности" от множества допустимых состояний имеет вид, приведенный в таблице 7.3.1.

Таблица 7.3.1.

Пример разбиения множества целей на группы

Группа	Расстояние до цели	Целевые ограничения в группе
1	2.	$5 - \kappa \le \xi_1$
2	1.	$6 - \kappa \le \xi_2$
0	0.	$\xi_1 \le 11 + \kappa, \quad \xi_2 \le 8 + \kappa$

§7.3.1. Управление процессом группировки целей

Расчет значений показателей неполной модели выполняется по правилам, с одной стороны, гарантирующим соблюдение обязательных ограничений и обеспечивающим, с другой стороны, наиболее полный учет целевых границ. Кроме того, выполняется группировка целевых ограничений по "степени их удаленности" от множества допустимых состояний.

При этом распределение целевых ограничений по группам может оказаться различным. Например, может случиться, что каждая группа содержит по одной цели, но вполне допустимо, что в некоторую группу попадает больше, чем одно ограничение.

Конкретное разбиение целей зависит от свойств используемой неполной математической модели и однозначно ими определяется. Вместе с тем, пользователь имеет возможность изменять структуру целевых групп путем соответствующего подбора значений ранжирующих коэффициентов.

Следующий пример иллюстрирует подобную процедуру.

Пусть множество допустимых состояний математической модели определяется системой неравенств:

$$0 \le \xi_1, 0 \le \xi_2,$$
$$\xi_1 + \xi_2 \le 3,$$

а множество целевых состояний:

$$4 \le \xi_1, 10 \le \xi_1 + 3\xi_2 .$$

Задача отыскания значения метрики для этой модели принимает вид:

минимизировать κ (*по совокупности* κ *и всех* ξ_1, ξ_2) *при условиях*:

$$\begin{aligned} &\kappa \ge 0, \\ &0 \le \xi_1, \quad 0 \le \xi_2, \\ &\xi_1 + \xi_2 \le 3, \\ &4 - \kappa \le \xi_1, \\ &10 - \kappa \le \xi_1 + 3\xi_2 . \end{aligned} \qquad (7.3.2)$$

Легко видеть, что решением задачи являются значения

$$\kappa = 3, \ \xi_1 = 1, \ \xi_2 = 2,$$

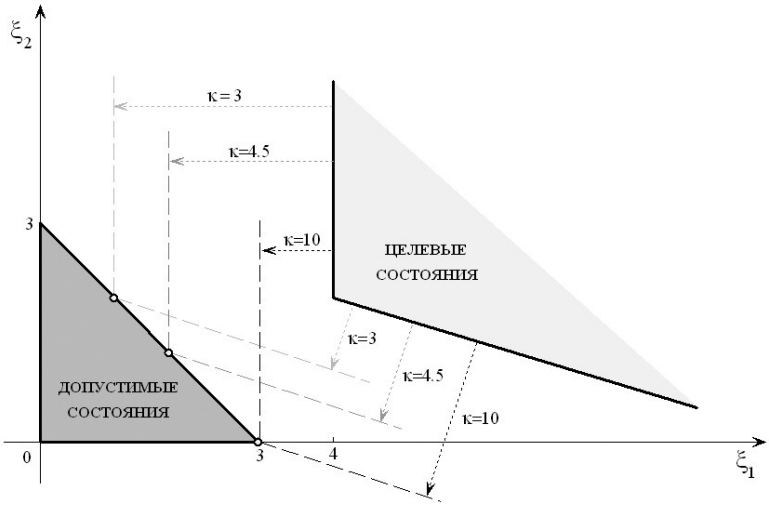


Рис. 7.3.2. Пример процесса управления группировкой целей

Геометрическая интерпретация этой задачи показана на рис. 7.3.2. а соответствующее ему, найденное в ходе процедуры группировки целей, разбиение множества целевых границ на группы имеет вид:

Таблица 7.3.2.

Разбиение целевых границ на группы задачи (7.3.2)

Группа	Расстояние до цели	Целевые ограничения в группе
1	3.	$4-\kappa \le \xi_1, \quad 10-\kappa \le x_1+3\xi_2$.
0	0.	нет

Факт попадания нескольких ограничений в одну группу означает, что эти целевые ограничения "вступают в конфликт" друг с другом при попытке уменьшения величины κ.

Как уже отмечалось, процедура расчета значения метрики приводит в точку конкурентного равновесия паретовского типа, в которой уменьшение κ для любого из "конфликтующих" целевых условий невозможно

без увеличения этой величины хотя бы для одного из остальных. Иначе говоря, процесс минимизации идет до тех пор, пока не будет выделена группа целевых ограничений, в которой дальнейшее улучшение κ невозможно, после чего все попавшие в данную группу целевые ограничения "фиксируются": переменная κ в условии задачи заменяется в этих ограничениях константой со значением, равным найденному минимуму.

Справедливость утверждения, что определяемое положение равновесия относится к *паретовскому* типу, следует непосредственно из определения используемого типа метрики.

Ранее также было отмечено, что положение равновесия на множестве Парето может быть смещено подходящей нормировкой ограничений, задающих множество целевых состояний. Эта возможность реализуется путем подбора пользователем неотрицательных ранжирующих (весовых) коэффициентов для целевых границ.

Например, заменив в задаче (7.3.2) условие $4 \le \xi_1$ на равносильное ему неравенство $8 \le 2\xi_1$ (то есть использовав ранжирующий коэффициент 2), решение будет смещено в точку $\xi_1 = 1.75$, $\xi_2 = 1.25$ со значением $\kappa = 4.5$. Новое положение равновесия несколько сместится на множестве Парето относительно исходного (см. рис. 7.3.2), но разбиение целевых ограничений по группам остается прежним.

Иной результат получится, если ранжирующий коэффициент выбрать равным 10. При замене в задаче (7.3.2) условия $4 \le \xi_1$ равносильным неравенством $40 \le 10\xi_1$ решение станет $\xi_1 = 3$, $\xi_2 = 0$ со значением $\kappa = 10$. При этом возникает иное разбиение целевых границ на группы:

Таблица 7.3.3.

Разбиение целей на группы для задачи (7.3.2) с ранжирующим коэффициентом для первой целевой границы равным 10

Группа	Расстояние до цели	Целевые ограничения в группе
1	10.	$40 - \kappa \le 10\xi_1$
2	7.	$10 - \kappa \le \xi_1 + 3\xi_2$
0	0.	нет

§7.3.2. Сравнение абсолютной и относительной метрик

Выбор типа метрики определяется только содержательной постановкой решаемых задач, поскольку с формальной
точки зрения все способы измерения расстояния между множествами равноправны – все они вводятся аксиоматически, то есть по определению.

Тем не менее, пользователь неполной модели должен представлять, что использование разных типов метрики для одной и той же математической модели может приводить (хотя и необязательно) к различным решениям.

Проиллюстрируем возможные, возникающие при замене типа метрики, случаи.

Использование относительной метрики в модели (7.3.1) приводит практически к тому же самому решению. Отличаться будут лишь значения минимальных расстояний до целевых ограничений. Действительно, задача (7.3.1) в случае использования относительной метрики принимает вид:

минимизировать κ (*по совокупности* κ *и всех* ξ_1, ξ_2) *при условиях*:

$$\kappa \geq 0,$$
$$0 \leq \xi_1 \leq 3,$$
$$0 \leq \xi_2 \leq 5,$$
$$5 - 5\kappa \leq \xi_1 \leq 11 + 11\kappa,$$
$$6 - 6\kappa \leq \xi_2 \leq 8 + 8\kappa .$$

Эта задача имеет решение: $\kappa = 0.4$, $\xi_1 = 3$, $\xi_2 = [4,5]$. Компоненты элемента x определяются, как и в случае абсолютной метрики, неоднозначно. На втором шаге процедуры группировки целей, решив задачу:

минимизировать κ (*по совокупности* κ *и всех* X_1, X_2) *при условиях*:

$$\kappa \geq 0, \quad 0 \leq \xi_1 \leq 3, \quad 0 \leq \xi_2 \leq 5,$$
$$3 \leq \xi_1 \leq 11 + 11\kappa,$$
$$6 - 6\kappa \leq \xi_2 \leq 8 + 8\kappa ,$$

находим решение: $\kappa = 0.1666667$, $\xi_1 = 3$, $\xi_2 = 5$.

Таблица 7.3.4 содержит разбиение целевых ограничений на группы для относительной метрик, и мы видим, что разбиение целевых ограничений на группы в точности повторяет приведенную в таблице 7.3.1 структуру.

Таблица 7.3.4.

Разбиение множества целевых границ на группы при использовании относительной метрики в задаче (7.3.1)

Группа	Расстояние до цели	Целевые ограничения в группе
1	0.4	$5-5\kappa \le \xi_1$
2	0.1666667	$6-6\kappa \le \xi_2$
0	0.	$\xi_1 \le 11+11\kappa, \quad \xi_2 \le 8+8\kappa$

Приведем теперь пример ситуации, когда использование разных типов метрики приводит к решениям, в которых число групп, а также распределение целевых ограничений по группам могут быть различными.

Внесем небольшие изменения в описание модели (7.3.1) так, что множество допустимых состояний будет определяться условиями:

$$0 \le \xi_1 \le 1, \quad 0 \le \xi_2 \le 3,$$

а множество целевых состояний – условиями:

$$1.5 \le \xi_1 \le 4, \qquad 4 \le \xi_2 \le 5.$$

Тогда задача для случая абсолютной метрики принимает вид:

минимизировать κ (*по совокупности* κ и всех ξ_1, ξ_2)

при условиях:

$$\begin{aligned} &\kappa \ge 0, \\ &0 \le \xi_1 \le 1, \\ &0 \le \xi_2 \le 3, \\ &1.5-\kappa \le \xi_1 \le 4+\kappa, \\ &4-\kappa \le \xi_2 \le 5+\kappa\,. \end{aligned} \tag{7.3.3}$$

Процедура группировки целей в задаче (7.3.3) приводит к решению $\xi_1 = 1$, $\xi_2 = 3$ и распределению целевых ограничений по группам,

показанному в таблице 7.3.5, причем после *первого* шага группировки переменная ξ_1 определяется неоднозначно (соответствующее множество точек на рисунке 7.3.5 выделено утолщенным отрезком *AB*).

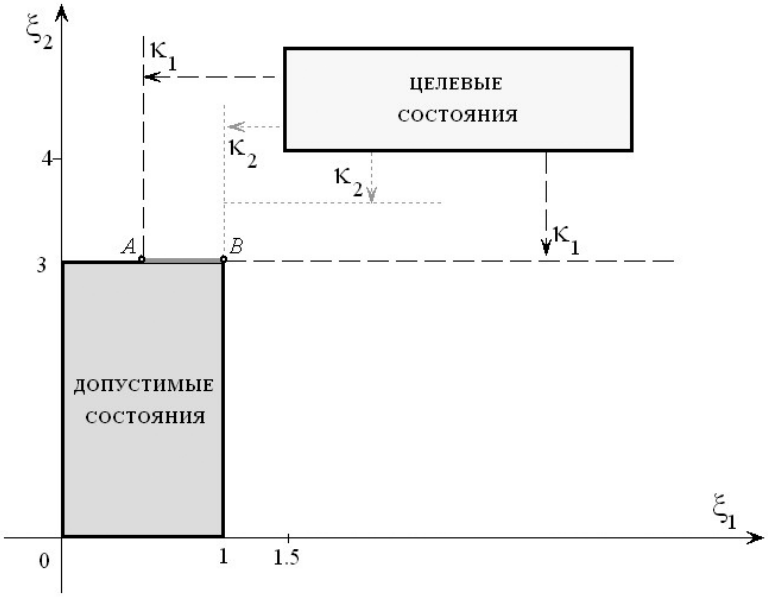


Рис. 7.3.5. Геометрическая интерпретация задачи (7.3.3)

Таблица 7.3.5.

Разбиение целей на группы для модели (7.3.3)

Группа	Расстояние до цели	Целевые ограничения в группе
1	1.	$4-\kappa \le \xi_2$
2	0.5	$1.5-\kappa \le \xi_1$
0	0.	$\xi_1 \le 4+\kappa, \quad \xi_2 \le 5+\kappa$

Использование же относительной метрики для измененной модели приводит к задаче (7.3.4):

минимизировать κ (*по совокупности* κ *и всех* ξ_1, ξ_2) *при условиях*:

$$\begin{aligned} &\kappa \geq 0, \;\; 0 \leq \xi_1 \leq 1, \\ &\qquad 0 \leq \xi_2 \leq 3, \\ &1.5 - 1.5\kappa \leq \xi_1 \leq 4 + 4\kappa, \\ &4 - 4\kappa \leq \xi_2 \leq 5 + 5\kappa\,, \end{aligned} \tag{7.3.4}$$

Таблица 7.3.6.

Разбиение множества целевых ограничений на группы в задаче (7.3.4)

Группа	Расстояние до цели	Целевые ограничения в группе
1	0.3333333	$1.5 - 1.5\kappa \leq \xi_1$
2	0.25	$4 - 4\kappa \leq \xi_2$
0	0.	$\xi_1 \leq 4 + 4\kappa, \quad \xi_2 \leq 5 + 5\kappa$

решение которой то же самое, что и для задачи (7.3.3): $\xi_1 = 1$, $\xi_2 = 3$, но с иным распределением целевых ограничений по группам, приведенным в таблице 7.3.6.

§7.3.3. Ранжирование целей

В неполных математических моделях допускается, чтобы цели, включаемые пользователем в формулировку модели, как конфликтовали друг с другом, так и противоречили ее обязательным границам и связям. Такое допущение оказывается необходимым, ибо достаточно часто пользователь не имеет возможности a priori ответить на вопрос: достижимы ли поставленные им цели, а если достижимы, то какова их относительная важность.

Формально конфликтность целей означает, что попытка уменьшить расстояние от выбранного допустимого состояния по одной из целей приводит к увеличению расстояния по другой. Совокупность всех состо-

ний, обладающих этим свойством, согласно определению 5.4.1 является множеством *конкурентного равновесия паретовского типа*, или просто множеством *Парето*.

Из определения способа измерения метрики следует, что в режиме автоматического расчета значений показателей модели делается попытка найти одно из таких состояний конкурентного равновесия, для которых:

- уменьшение значения κ для любого из конфликтующих целевых условий невозможно без его увеличения хотя бы для одного из остальных;
- значение величины κ минимально.

Положение равновесия на множестве Парето можно смещать при помощи *нормировки* условий, задающих целевые границы, что эквивалентно изменению их относительной важности для пользователя. Эта возможность реализуется путем введения в условие задачи вычисления значения метрики неотрицательных *ранжирующих коэффициентов* w_k и W_k. Для строго положительных рангов целей задача поиска значения абсолютной метрики будет иметь вид:

минимизировать κ (*по совокупности* κ *и всех* ξ_i, *где* $i = [1,n]$) *при условиях*:

$$\kappa \geq 0,\ d_k \leq \xi_k \leq D_k,\ k = [1,n],$$
$$r_k - \kappa / w_k \leq \xi_k \leq R_k + \kappa / W_k,\ k = [1,n],$$
$$\xi_k = \sum_{i=1}^{k-1} A_{ki}\xi_i,\ k = [2,n].$$

Аналогично, задача расчета относительной метрики будет иметь вид:

минимизировать κ (*по совокупности* κ *и всех* ξ_i, *где* $i = [1,n]$) *при условиях*:

$$\kappa \geq 0,\ d_k \leq \xi_k \leq D_k,\ k = [1,N],$$
$$r_k - \kappa \left| r_k \right| / w_k \leq \xi_k \leq R_k + \kappa \left| R_k \right| / W_k,\ k = [1,N],$$
$$\xi_k = \sum_{i=1}^{k-1} A_{ki}\xi_i,\ k = [2,n].$$

Чем больше значение коэффициентов w_k и W_k, тем выше ранг соответствующего целевого ограничения в модели. Если же ранжирующий коэффициент выбран равным нулю, то соответствующее ему целевое ограничение просто игнорируется и исключается из условия задачи.

Меняя величины ранжирующих коэффициентов, пользователь оказывается в состоянии "перемещать" на множестве Парето положение равновесия, имеющее минимальное значение метрики. По умолчанию значения ранжирующих коэффициентов устанавливаются равными единице.

§7.3.4. Замечания о роли линейности в неполном математическом моделировании

Помимо линейных связей между показателями в неполные математические модели можно включать и некоторые нелинейные, точнее говоря, кусочно-линейные зависимости. Такое включение оказывается возможным, поскольку в распоряжении пользователя в процессе построения неполной математической модели имеется полный набор операций над объектами стандартного линейного пространства, дополненный операцией поиска в этом пространстве экстремума линейной функции на выпуклом многограннике.

Поясним сказанное на примере реализации функции модуля (абсолютной величины) действительного числа. Из курса элементарной математики известно, что абсолютная величина числа равна самому числу, если оно неотрицательное, и этому числу, взятому с обратным знаком, если оно отрицательное. В виде формулы данное определение записывается так:

$$\mathrm{abs}(x) = x \text{, если } x \geq 0,$$
$$\mathrm{abs}(x) = -x \text{, если } x < 0.$$

При помощи операции отыскания экстремума сделанное определение можно переписать в эквивалентной форме

$$\mathrm{abs}(x) = \max\{x,-x\},$$

иначе говоря, абсолютная величина числа x равна максимальному из двух чисел: x и $-x$. Эта функция модуля может быть реализована, например, следующим образом.

Пусть в списке показателей математической модели последовательно находятся показатели с *именами* $x, \mathrm{abs}(x), y1$ и $y2$. Установим следующие линейные связи между этими показателями:

$$y1 = \mathrm{abs}(x) - x\ ;\ \ y2 = \mathrm{abs}(x) + x$$

и следующие обязательные границы:

$$y1 \geq 0 \quad ; \quad y2 \geq 0 \quad ; \quad \mathrm{abs}(x) \geq 0,$$

а также желательное условие $\mathrm{abs}(x) \leq 0$. Наконец, величина показателя x задается присвоением одновременно нижней и верхней обязательным границам этого показателя конкретного значения аргумента вычисляемой функции.

Аналогичным образом можно реализовать практически все типы выпуклых кусочно-линейных зависимостей между показателями. Вместе с тем необходимо сделать существенную оговорку: использование таких зависимостей может оказаться некорректным, если желательные границы показателей, используемых для описания нелинейных связей, вступают в конфликт с другими целевыми условиями модели. Возникающие в этом случае затруднения могут быть преодолены путем применения специальных параметрических или двухуровневых математических моделей.

Вместе с тем, предлагаемый подход позволяет в большом числе практически важных случаев ограничиться использованием линейных и кусочно-линейных неполных математических моделей. Подобное ограничение является весьма жестким в случае традиционной методологии, базирующейся на применении полных моделей, однако не будет таковым при использовании методологии неполного моделирования.

Основанием для подобного утверждения является возможность построения выпуклой аппроксимирующей кусочно-линейной оболочки для произвольного нелинейного выпуклого множества допустимых или целевых состояний, причем качество аппроксимации может неограниченно улучшаться в процессе пополнения.

Наконец, за счет введения в неполную математическую модель множества желательных состояний удается преодолеть затруднения, обусловленные как достаточно часто имеющей место неоднозначностью допустимого решения задачи, так и возможностью образования противоречивой системы обязательных условий на некотором шаге процедуры пополнения.

Глава 8
ПРИМЕРЫ ПРАКТИЧЕСКОГО ИСПОЛЬЗОВАНИЯ НЕПОЛНЫХ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Раздел 8.1. РЕШЕНИЕ ЗАДАЧ НЕПОЛНОГО МОДЕЛИРОВАНИЯ ПРИ ПОМОЩИ СПЕЦИАЛИЗИРОВАННЫХ ЭЛЕКТРОННЫХ ТАБЛИЦ

Рассмотренные в предыдущих параграфах данной главы задачи не требовали существенных затрат вычислительных усилий на получение их решений. Однако "графический" метод решения очевидно применим лишь в случаях, когда размерность пространства состояний неполной модели не превосходит трех или, если условия задач (7.1.3.1)–(7.1.3.2) имеют специфическую структуру (например, ограничения-связи между показателями отсутствуют).

Как правило, условия задач (7.1.4.1)–(7.1.4.2) не обладают данными свойствами, потому использованию вычислительной техники для их решения нет альтернативы. Более того, в случаях достаточно высокой размерности пространства состояний сложной для реализации процедурой оказывается не только поиск решений задач (7.1.4.1)–(7.1.4.2), но также ввод исходной информации и анализ получаемых результатов.

В рамках данного курса будут рассмотрены два возможных подхода к проблеме преодоления этих затруднений.

Первый, основанный на использовании специализированных электронных таблиц, предполагает, что число показателей неполной модели относительно невелико и в процессе работы пары "ЭВМ–пользователь" пополняемые задачи (7.1.4.1)–(7.1.4.2) решаются последовательно и независимо друг от друга.

Второй подход, предназначенный для работы с высокоразмерными и сложными по структуре моделями, использует автоматизацию при помощи специализированного языка программирования процедур:

- ввода и коррекции исходных данных,
- решения (быть может, неоднократного) задач (7.1.4.1)–(7.1.4.2),
- анализа получаемых на каждом шаге процедуры пополнения, решений.

В данном разделе демонстрируется применение первого подхода, в то время как второй подход будет рассмотрен в разделах 8.2. и 8.3. В качестве инструмента неполного математического моделирования используется фрагмент системы содействия принятию управленческих решений "Баланс", оформленное как `MS-DOS`-приложение с именем `KBM.exe`[11], представляющий собой электронную таблицу специальной структуры и снабженную набором встроенных функций, необходимых для решения задач (7.1.4.1)–(7.1.4.2) при помощи одной из версий симплекс-метода (см. § 4.4.2.)

Следует отметить, что для этой цели также возможно использование электронных таблиц типа `MS Excel`, однако это может оказаться не очень простой процедурой, поскольку универсальные электронные таблицы требуют специальной предварительной настройки.

Использование специализированной электронной таблицы продемонстрируем на примере решения задачи (5.4.3), описание которой дано в разделе 5.4.

Найти минимум κ *по совокупности* $\{\kappa, \xi_1, \xi_2\}$,

при условиях

$$\kappa \geq 0,$$
$$\xi_1 + \xi_2 \geq 5 - \kappa,$$
$$2\xi_2 \geq 4 - \kappa,$$
$$\xi_1 + 2\xi_2 \leq 3 + \kappa,$$
$$\xi_1 \geq 1, \quad \xi_2 \geq 1, \quad \xi_1 + 3\xi_2 \leq 7.$$

Приведем постановку данной задачи к виду удобному для использования электронной таблицы. Вначале составим упорядоченный список показателей (см. таблицу 8.1.1.1) и соответствующих им формул и границ,

---

[11] На момент подготовки 4-го издания настоящего пособия было разработано (функционально эквивалентное `KBM.exe`) приложение, допускающее работу с ним под 32- и 64-битными версиями операционной системы `MS Windows`.

причем упорядоченность должна отражать особенность структуры ограничений–связей в задачах (7.1.4.1)–(7.1.4.2), когда в записи формулы каждого из них используются только *находящиеся выше в этом списке* показатели.

Отметим также, что нулевое значение верхней желательной границы показателя κ реализует условие минимизации значения этого показателя.

Таблица 8.1.1.1

Имя показателя	Формула	Нижняя обяз. гр.	Верхняя обяз. гр.	Нижняя жел. гр.	Верхняя жел. гр.
Ksi1	ξ_1	0	$+\infty$	$-\infty$	$+\infty$
Ksi2	ξ_2	0	$+\infty$	$-\infty$	$+\infty$
Ksi1+3Ksi2	$\xi_1 + 3\xi_2$	$-\infty$	7	$-\infty$	$+\infty$
Kappa	κ	0	$+\infty$	$-\infty$	0
Func1	$\xi_1 + \xi_2 + \kappa$	5	$+\infty$	$-\infty$	$+\infty$
Func2	$2\xi_2 + \kappa$	4	$+\infty$	$-\infty$	$+\infty$
Func3	$\xi_1 + 2\xi_2 - \kappa$	$-\infty$	3	$-\infty$	$+\infty$

Приведем теперь детальное пошаговое описание процедуры решения задачи (5.5.3). Для облегчения использования демонстрационного программного обеспечения можно в любой момент работы приложения нажатием клавиши **F1** осуществлять вызов встроенной системы подсказок. При этом выдается список доступных в данный момент команд и описание способа их вызова.

После запуска приложения `KBM.exe`, пользователь попадает в архивный раздел таблицы, предназначенный для создания, хранения и редактирования задач вида (7.1.4.1)–(7.1.4.2) (Рис. 8.1.1.1).

Отметим, что 16-битная версия допускает сохранение или вызов файлов задач только при работе с гибким диском (или любым его эмулятором). Выбор раздела архива из списка доступных осуществляется при помощи клавиш $\rightarrow$ или $\leftarrow$ с последующим нажатием клавиши **Enter**.

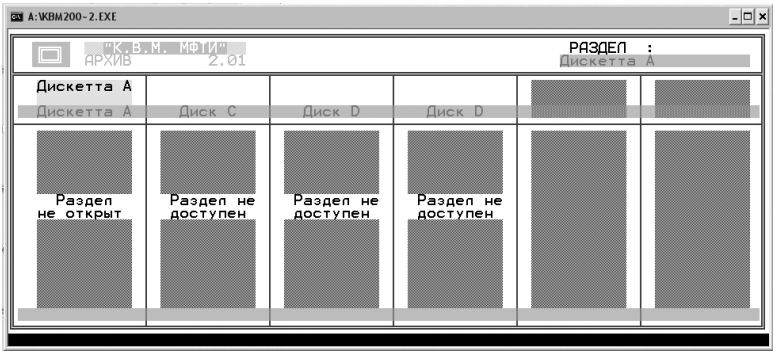


Рис. 8.1.1.1.

Выбрав раздел `Дискетта A` и нажав клавишу **`Enter`**, получаем доступ к каталогу задач на гибком диске. Выбор задачи из списка уже имеющихся осуществляется при помощи клавиш ↑ или ↓ с последующим нажатием клавиши **`Enter`**.

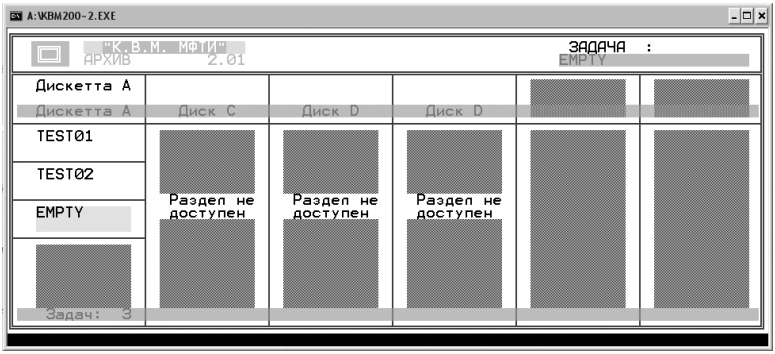


Рис. 8.1.1.2

Создадим новую задачу в текущем разделе под именем `MultiCRT`. Выберем задачу с именем `EMPTY` (рисунок 8.1.1.2), в которой нет никаких данных, и, *не нажимая* клавиши **`Enter`**, создадим ее копию с именем `MultiCRT`. Для той цели, нажав клавишу **`ESC`**, перейдем в командное меню (рисунок 8.1.1.3) и выберем команду `КОПИРОВАНИЕ`.

Нажав клавишу **`Enter`**, получим запрос на ввод имени новой задачи (рис. 8.1.1.4), и после ввода имени `MultiCRT` с последующим нажатием клавиши **`Enter`** в текущем разделе архива будет создана задача с этим именем (рис. 8.1.1.5).

Новая задача является копией задачи EMPTY и не содержит никаких данных. Для начала ввода данных необходимо перейти из подсистемы АРХИВ в подсистему ДАННЫЕ. Этот переход выполняется из состояния, показанного на рис. 8.1.1.5, нажатием клавиши **Enter**. Последующее состояние экрана показано на рис. 8.1.1.6.

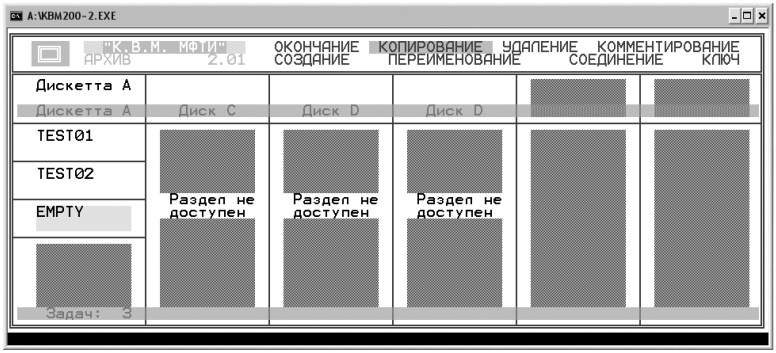


Рис. 8.1.1.3

Поскольку в решаемой задаче, согласно таблице 8.1.1.1, потребуется семь показателей, создадим их, нажав шесть раз комбинацию клавишей **ALT** и **I**. В результате экран перейдет в состояние, показанное на рис. 8.1.1.7.

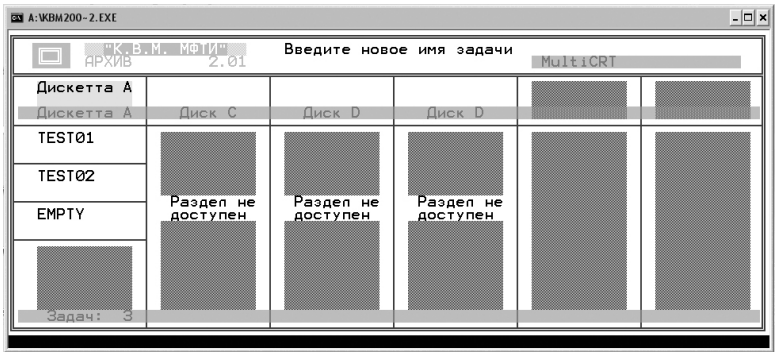


Рис. 8.1.1.4

Теперь, в соответствии с таблицей 8.1.1.1, после нажатия клавиши **TAB**, можно ввести желаемые имена показателей (рис. 8.1.1.8).

Нажав еще раз клавишу **TAB**, перейдем из окна экрана, содержащего имена показателей, в окно, содержащее значение их атрибутов, и введем

согласно таблице 8.1.1.1 значения обязательных границ (рис. 8.1.1.9). Для ввода значения конкретного атрибута необходимо при помощи клавиш ↑, ↓, → и ← установить курсор на ячейку, содержащую значение этого атрибута, и нажать клавишу **Enter**.

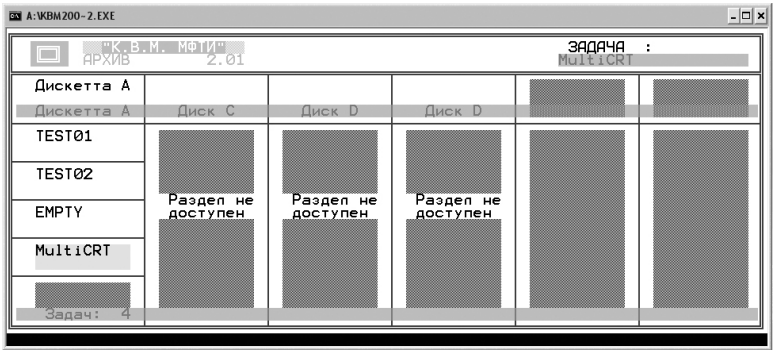


Рис 8.1.1.5

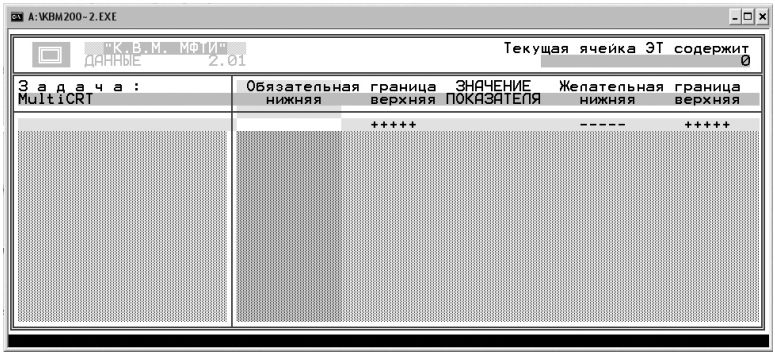


Рис. 8.1.1.6

При этом активизируется окно редактора ячеек электронной таблицы, расположенное в правом верхнем углу экрана.

Нажав еще раз клавишу **TAB**, перейдем из окна экрана, содержащего имена показателей, в окно, содержащее значение их атрибутов, и введем согласно таблице 8.1.1.1 значения обязательных границ (рис. 8.1.1.9.) Для ввода значения конкретного атрибута необходимо при помощи клавиш ↑, ↓, → и ← установить курсор на ячейку, содержащую значение

этого атрибута, и нажать клавишу **Enter**. При этом активизируется окно редактора ячеек электронной таблицы, расположенное в правом верхнем углу экрана.

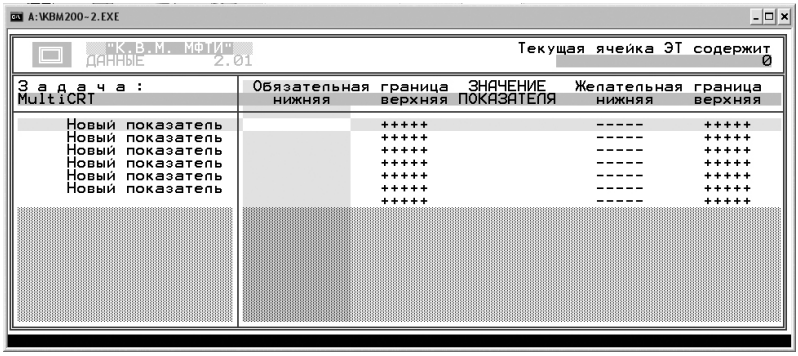


Рис. 8.1.1.7

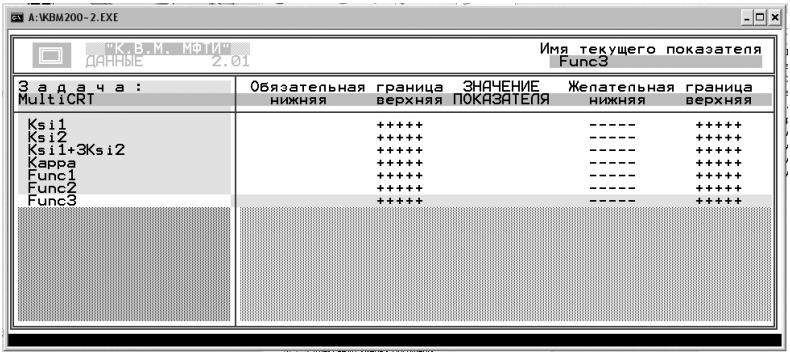


Рис. 8.1.1.8

Следует обратить внимание на то, что нулевые значения атрибутов в поле электронной таблицы изображены пробелами, хотя в окне редактора содержимого ячеек (в верхнем правом углу экрана) электронной таблицы значения ячеек представлены в явном виде и с максимальной возможной точностью. Цепочка символов `------` изображает $-\infty$, а цепочка `++++++` изображает $+\infty$.

Далее, переместившись на правый край электронной таблицы в ее "треугольную" часть, введем согласно таблице 8.1.1.1 значения коэффициентов-связей (рис. 8.1.1.10). Заметим, что данный специальный

"треугольный" вид не позволяет нарушать правило упорядоченности списка показателей.

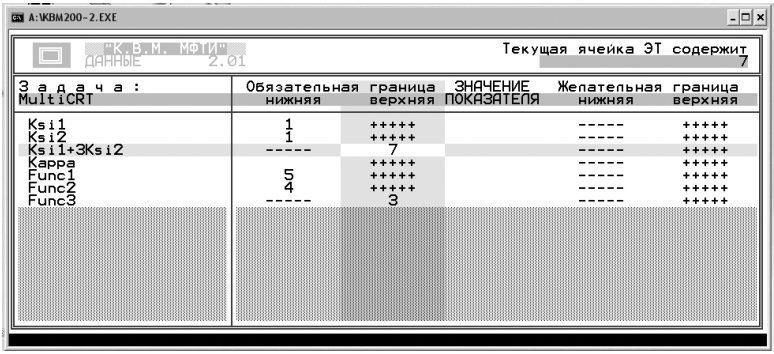


Рис. 8.1.1.9

Наконец, введем значения целевых (желательных) границ для значений показателей (рис. 8.1.1.11), что и завершает ввод условий задачи.

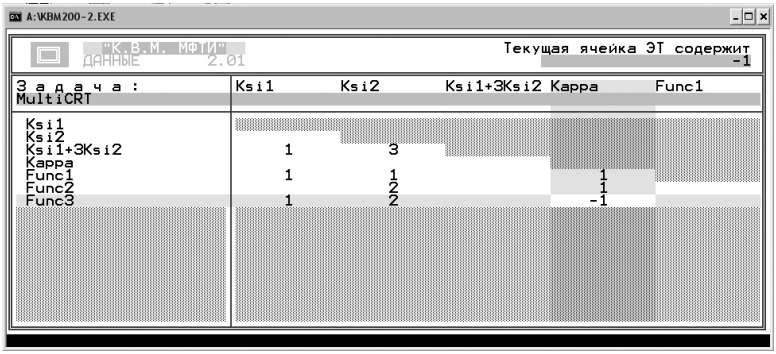


Рис. 8.1.1.10

Следующий этап работы с неполной моделью состоит в определении значений остальных атрибутов в автоматическом режиме. Для этого необходим переход в подсистему РЕШЕНИЕ и запуск "решающего" модуля системы.

Нажав клавишу **ESC**, перейдем в командное меню (рис. 8.1.1.12) и выберем команду РЕШЕНИЕ. После нажатия клавиши **Enter** решающий

модуль запустится автоматически выдавая по ходу решения серии задач линейного программирования (7.1.4.1)–(7.1.4.2) информацию о суммарных величинах нарушения обязательных ограничений и полученных оценках значения метрики.

Рис. 8.1.1.11

Процедура завершается определением статуса найденного решения и предложением сохранить его для использования в качестве начального приближения для последующих шагов модификации или пополнения модели (рис. 8.1.1.13).

Рис. 8.1.1.12

После чего выполняется возврат в подсистему ДАННЫЕ. Значения автоматически рассчитываемых атрибутов будут подставлены в соответствующие ячейки электронной таблицы и отображены на экране (рис. 8.1.1.14).

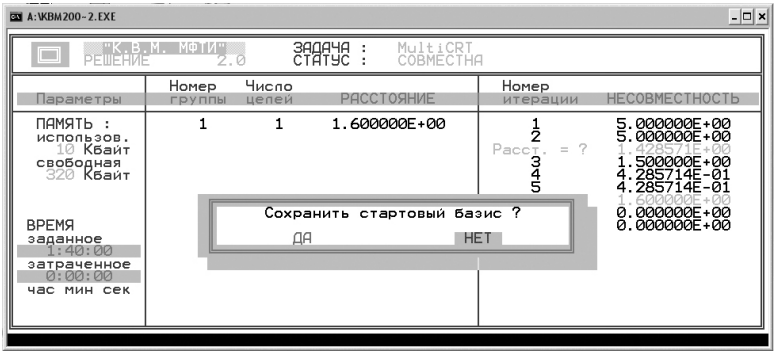


Рис8.1.1.13

Одновременно в электронной таблице представляются результаты процедуры группировки целевых границ: число образованных групп и распределение целевых границ по группам с соответствующими им оптимальными значениями метрики (рис. 8.1.1.15).

Рис. 8.1.1.14

Завершение работы приложения выполняется автоматически после нажатия клавиши **ESC,** перехода в командное меню (рис. 8.1.1.16) и выбора команды ОКОНЧАНИЕ с последующим нажатием клавиши **Enter.**

Как не трудно заметить, полученное решение задачи (атрибут ЗНАЧЕНИЕ ПОКАЗАТЕЛЯ) совпадает с решением, приведенным в разделе 5.4 для задачи 5.4.3.

Рис. 8.1.1.15

Рис8.1.1.16

Раздел 8.2. ПРИМЕНЕНИЕ ИНТЕРПРЕТАТОРА ЯЗЫКА L В ЗАДАЧЕ "АНАЛИЗ ЭФФЕКТИВНОСТИ ИНВЕСТИЦИОННЫХ ОПЕРАЦИЙ НА РЫНКЕ ЦЕННЫХ БУМАГ"

В данном разделе демонстрируется применение программных средств и методологии неполного моделирования для решения более сложных задач.

В качестве примера рассмотрим процедуру получения маргинальных оценок количественных характеристик инвестиционной деятельности на

рынке ценных бумаг. Хотя единицы измерения этих характеристик выбраны условно, рассматриваемый пример в достаточно полной мере демонстрирует как концептуальную специфику неполных математических моделей, так и возможные пути преодоления практических трудностей их использования.

§8.2.1. Содержательная постановка задачи

Будем рассматривать следующую операцию на рынке ценных бумаг. В течении N периодов (например, банковских дней) приобретается и продается определенный вид ценных бумаг с известной доходностью, оплата приобретения которых осуществляется заемными средствами, привлекаемыми по кредитной линии с фиксированной процентной ставкой.

Доходность ценных бумаг и стоимость кредита для каждого из периодов постоянны и известны. Максимальные объемы привлекаемых-возвращаемых средств по кредитной линии за один период, а также объемы приобретаемых-продаваемых ценных бумаг в течение одного периода ограничены. Также предполагается, что объем портфеля ценных бумаг и суммарная задолженность по кредиту как в начале операции, так при ее окончании равны нулю.

Требуется оценить возможную эффективность данной операции по максимальной величине остатка средств на счете операции при ее завершении при фиксированном заранее числе периодов. Необходимо также рассчитать динамику привлечения-возврата заемных средств, равно как и приобретения-продаж ценных бумаг, обеспечивающую эту эффективность. Схема операции приведена на рис.8.2.1.1.

§8.2.2. Построение списков показателей и их атрибутов

Используем для формализованного описания исследуемой операции (являющимся частным случаем задачи *дискретного оптимального управления*, см. § 5.1.1) следующие количественные характеристики для каждого периода $k = [1, N]$:

- $S(k)$ – остаток средств на счете операции в период k;
- $Q(k)$ – объем средств, привлеченных (или возвращенных) за период k;

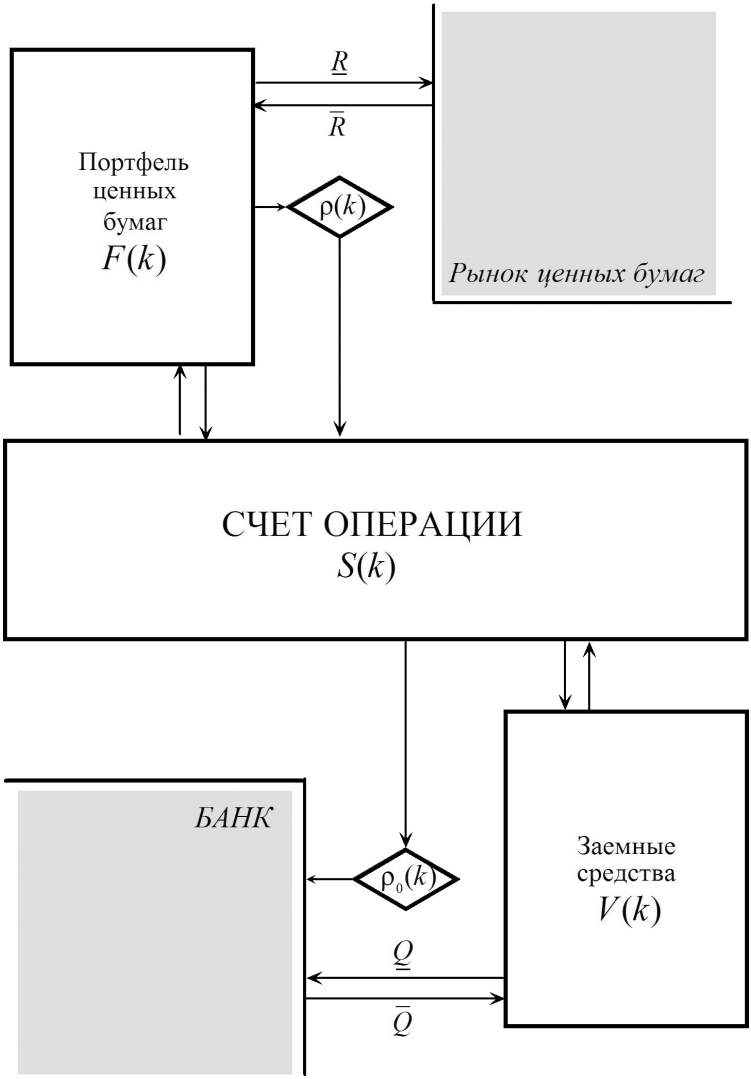


Рис. 8.2.1.1.

- $R(k)$ – стоимость ценных бумаг, купленных (или проданных) за период k ;

- $F(k)$ – объем портфеля ценных бумаг в период k;
- $V(k)$ – величина задолженности по кредиту в период k;
- $\rho_0(k)$ – процентная ставка по кредиту в период k;
- $\rho(k)$ – доходность ценных бумаг в период k.

Эти характеристики будут связаны следующими динамическими соотношениями для каждого $k = [1, N-1]$:

1°. Динамика счета операции:

$$S(k+1) = \\ = S(k) + Q(k) - R(k) + \rho(k) \cdot F(k) - \rho_0(k) \cdot V(k).$$

2°. Динамика объема портфеля ценных бумаг

$$F(k+1) = F(k) + R(k). \qquad (8.2.2.1)$$

3°. Динамика задолженности по кредиту

$$V(k+1) = V(k) + Q(k).$$

В системе разностных уравнений (8.2.2.1) переменные $S(k),\ V(k), F(k)$ являются *фазовыми переменными*, а переменные – $Q(k)$ и $R(k)$ *управлениями.* На фазовые переменные и управления при этом накладываются следующие ограничения:

- $S(k) \geq 0;\ \forall k \in [1, N]$, то есть в ходе операции овердрафт по счету операции не допускается;
- $V(k) \geq 0;\ \forall k \in [1, N]$ и $F(k) \geq 0;\ \forall k \in [1, N]$ – очевидны;
- управления $Q(k)$ и $R(k)$ произвольны по знаку и должны удовлетворять двусторонним неравенствам, отражающими как технические, так и экономические ограничения (например, по ликвидности ценных бумаг)

  $$\underline{Q}(k) \leq Q(k) \leq \overline{Q}(k);\ \ \underline{R}(k) \leq R(k) \leq \overline{R}(k);$$

  $$\forall k \in [1, N];$$
- начальные условия $S(1) = V(1) = F(1) = 0$, равно как и конечные условия $V(N) = F(N) = 0$ вытекают из условий проведения операции.

Наконец, целевое условие операции имеет вид $S(N) \to \max$.

Данная совокупность обязательных и желательных условий формально является задачей линейного программирования, в которой показатели

$$S(k), F(k), V(k), R(k), Q(k), \quad \forall k = [1, N]$$

являются переменными, а число периодов N и величины

$$\rho(k),\ \rho_0(k),\ \underline{R}(k),\ \overline{R}(k),\ \underline{Q}(k),\ \overline{Q}(k), \qquad \forall k = [1, N]$$

– параметрами, где величины $S(k), F(k), V(k)$ имеют размерность "у.е.", величины $R(k), Q(k), \underline{R}(k), \overline{R}(k), \underline{Q}(k)$ и $\overline{Q}(k)$ измеряются в "у.е./период", а $\rho(k)$ и $\rho_0(k)$ безразмерные.

Поскольку в этой модели не отражен ряд существенных, но не формализуемых факторов (например, таких как: внешние экономические и политические условия, а также всевозможные риски, обусловленные нестабильностью рынка ценных бумаг или стоимости кредита), то ее следует отнести к классу неполных моделей, позволяющей получить лишь маргинальную (верхнюю) оценку максимальной эффективности рассматриваемой операции.

§8.2.3. Формулировка условий базового варианта задачи на языке L

Расчет базового варианта задачи был выполнен для следующего набора значений параметров: $N = 300$ и $\forall k = [1, N]$

$$\underline{Q}(k) = -20.0, \quad \overline{Q}(k) = 40.0,$$
$$\underline{R}(k) = -50.0, \quad \overline{R}(k) = 40.0,$$
$$\rho(k) = 0.008, \quad \rho_0(k) = 0.004.$$

Поскольку для каждого из периодов для рассматриваемой модели необходимо, следуя правилам, описанным в §7.1, использовать 12 показателей, то общее число показателей модели для базового варианта составит 3600, что приводит к *необходимости автоматизации* процедур ввода данных в ЭВМ и анализа полученных решений.

В рамках настоящего курса для решения этих проблем оказалось достаточным использовать:

1) интерпретатор языка L, позволяющий автоматизировать формирование условий и поиск решения для линейных задач оптимального управления;
2) программу MS Excel для обработки результатов расчетов.

Язык L является модифицированным подмножеством языка C++. Полное описание состава языка L, синтаксиса команд и правил использования дается в главе 9. Ниже приводится закомментированный текст входного файла для интерпретатора языка L, позволяющий ввести исходные данные, решить задачу линейного программирования и сформировать файл выходных данных для базового варианта задачи.

```
//
//
//              Модель Invest04.l
//
// L-файл для модели оценки эффективности инвестиционных
// операций на рынке ценных бумаг
// В этом варианте модели доходность ценных бумаг и стоимость
// кредита постоянны
//
//
// Блок параметров
//
// Задается число периодов
const int N = 300;
//
// Задается имя варианта задачи
string name = "Invest04";
//
// Задаются границы для управлений Q(k) и R(k)
const double Qmin = -20.00;
const double Qmax =  40.00;
const double Rmin = -50.00;
const double Rmax =  40.00;
```

```
// Задается стоимость кредита
const double r0 = 0.004;
// Задается доходность ценных бумаг
const double rk = 0.008;
//
// Создаются массивы значений p(k) и p0(k)
double po[N+1];
double pk[N+1];
int j;
      for( j=0; j<=N; j++ )
            { po[j]=r0; pk[j]=rk; };

// Формируем имя для служебного файла
string datName = name ".dat";
//
// Формируем имя для выходного файла
string txtName = name ".txt";

// Определяется структура данных для периода модели
struct Period
// Задаем список показателей для периода
{     real F;
      real V;
      real S;
      real Q;
      real R;
      real dF;
      real dV;
      real dS;
// Последние три показателя нужны для
// формирования связей между фазовыми
// переменными в разных периодах
};
//
```

```
// Создается массив описаний периодов
Period p[N+1];
//
// Далее выполняется инициализация показателей модели для всех
// периодов
  int i;
  for( i=0; i<= N; i++ )
     {
      SetName( "F (" _itoa(i) ")", p[i].F  );
      SetName( "V (" _itoa(i) ")", p[i].V  );
      SetName( "S (" _itoa(i) ")", p[i].S  );
      SetName( "Q (" _itoa(i) ")", p[i].Q  );
      SetName( "R (" _itoa(i) ")", p[i].R  );
      SetName( "dF(" _itoa(i) ")", p[i].dF );
      SetName( "dV(" _itoa(i) ")", p[i].dV );
      SetName( "dS(" _itoa(i) ")", p[i].dS );
     };
// Задаются начальные условия (для периода k=1)
      p[0].F.lOblBound = p[0].F.uOblBound = 0;
      p[0].V.lOblBound = p[0].V.uOblBound = 0;
      p[0].S.lOblBound = p[0].S.uOblBound = 0;
//
// Устанавливаются связи и ограничения для показателей для
// периода с номером k
// Задаются связи между фазовыми переменными
  int k;
  for( k=1; k<=N; k++ )
     {
p[k].dF :=
      p[k-1].F.link - p[k].F.link + p[k].R.link;
p[k].dV :=
      p[k-1].V.link - p[k].V.link + p[k].Q.link;
```

```
p[k].dS :=
      p[k-1].S.link - p[k].S.link
    - po[k] * p[k].V.link
    + pk[k] * p[k].F.link
    + p[k].Q.link - p[k].R.link;
//
// Уравнения динамических связей
p[k].dF.lOblBound = p[k].dF.uOblBound = 0;
p[k].dV.lOblBound = p[k].dV.uOblBound = 0;
p[k].dS.lOblBound = p[k].dS.uOblBound = 0;
//
// Устанавливаются ограничения на управления
      p[k].Q.lOblBound = Qmin;
      p[k].Q.uOblBound = Qmax;
      p[k].R.lOblBound = Rmin;
      p[k].R.uOblBound = Rmax;
     };
//
// Задаются конечные условия (для периода k=N)
      p[N].F.lOblBound = p[N].F.uOblBound = 0;
      p[N].V.lOblBound = p[N].V.uOblBound = 0;
//
// Задается целевое ограничение: (максимизация S(N))
      p[N].S.lDesBound = 15000;
//
// Устанавливаются значения допустимых погрешностей решения
      epsZero = epsBound = epsDeriv = 1e-9;
      epsIncon = 1e-9;
//
// Начало процесса решения (запуск решателя)
      print("\nSolving...\n");
      solve();
```

```
// Формирование выходных файлов
      print("\nSaving\n");
      SetProblemName(name);
      SaveProblem(datName);
//
// Перенаправление печати данных в выходной файл
      redirect(txtName);
//
// Определяется символ табуляции при печати
string s = "\t";

// Печать строки с именами показателей
print( "po" s "pk" s "F" s "V" s "S" s "R" s "Q"
       s "dF" s "dV" s "dS\n" );
//
// Печать найденных значений показателей для всех периодов
      for( i=0; i<=N; i++ )
            {
            print(
                  _ftoa(po[i]) s
                  _ftoa(pk[i]) s
                  _ftoa(p[i].F.value)  s
                  _ftoa(p[i].V.value)  s
                  _ftoa(p[i].S.value)  s
                  _ftoa(p[i].R.value)  s
                  _ftoa(p[i].Q.value)  s
                  _ftoa(p[i].dF.value) s
                  _ftoa(p[i].dV.value) s
                  _ftoa(p[i].dS.value) "\n"
                  );
            };
```

```
// Создание выходного параметра с оптимальным
// значением целевого функционала
double result;
      result=p[N].S.value;
//
// Закрытие выходного файла
      redirect("-");
      print("\nExiting\n");
```

§8.2.4. Анализ решения базового варианта

На рис. 8.2.4.1 и 8.2.4.2 приведены графики показателей – фазовых переменных $S(k), F(k), V(k), \forall k = [1, N]$ и показателей – управлений $R(k), Q(k), \forall k = [1, N]$, отражающие динамику их значений, обеспечивающую максимизацию остатка средств на счете операции при ее завершении для базового варианта расчета.

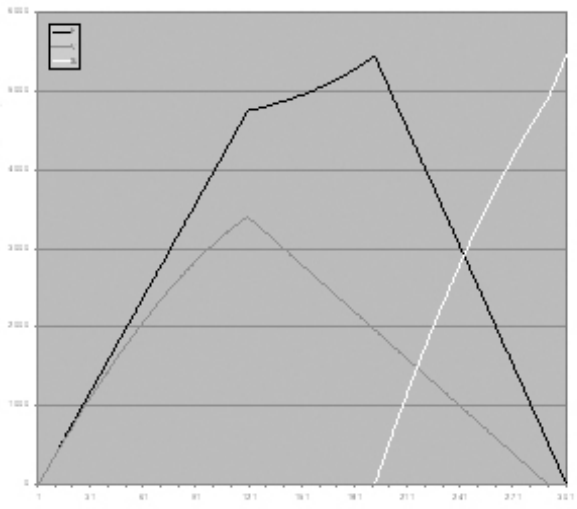


Рис. 8.2.4.1

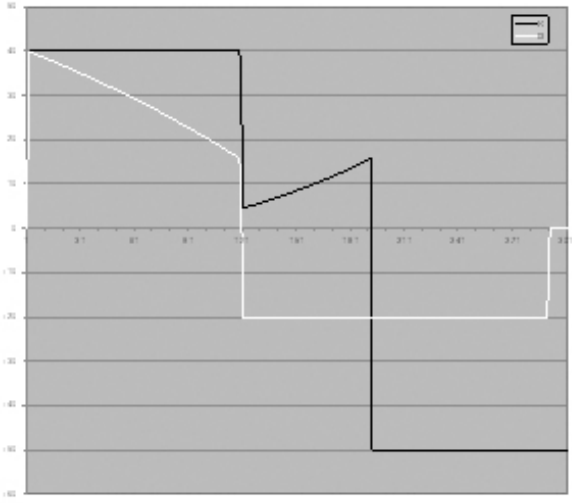


Рис. 8.2.4.2

Для определенности будем считать, что начальные условия для фазовых переменных фиксируются для $k = 0$, а конечные – для $k = 300$.

Описание оптимальной динамики значений показателей модели удобно выполнить, разделив множество всех периодов на четыре непересекающихся подмножества, как показано в таблице 8.2.4.1.

Детальные выходные данные для данного варианта находятся в файле `Invest04_Result.xls`.

Таблица 8.2.4.1.

№	Значения k	Описание динамики
1	$1 \le k \le 120$	Для данной группы периодов основной характеристикой оптимальной динамики является увеличение объема портфеля ценных бумаг $F(k)$ с максимально возможной скоростью. Величина управления $R(k)$ – скорости приобретения-продажи ценных бумаг – находится на верхней границе его допустимых значений.

		Оплата приобретаемых бумаг осуществляется как за счет привлекаемых заемных средств, так и реинвестирования дивидендов от уже приобретенных бумаг. Причем по мере роста объема портфеля скорость привлечения заемных средств $Q(k)$ снижается, хотя и остается положительной. Остаток средств на счете операции $S(k)$ для данных периодов нулевой
2	$121 \le k \le 191$	Для этой группы периодов характерной особенностью является начало возврата привлеченных заемных средств, снижение текущего объема задолженности $V(k)$ с максимально возможной скоростью. Значение соответствующего управления – $Q(k)$ скорости привлечения-возврата заемных средств – отрицательна и находится на нижней границе допустимых значений. На этом этапе рост объема ценных бумаг оказывается еще возможным, но со скоростью меньшей, чем допустимый максимум. Дивиденды тратятся в первую очередь на возврат займа, а их остающаяся часть реинвестируется в ценные бумаги. Поэтому остаток средств на счете операции для данных периодов по-прежнему нулевой.
3	$192 \le k \le 291$	Эта группа периодов характеризуется началом процесса реализации портфеля ценных бумаг, объем которого уменьшается с максимально возможной скоростью, необходимой для выполнения краевого условия $F(N) = 0$. Значения обоих управлений находятся на своих нижних допустимых границах. Средства, получаемые от реализации ценных бумаг, и дивиденды от оставшейся части портфеля аккумулируются на счете операции, остаток которого $S(k)$ растет.

4	$292 \le k \le 300$	К началу этой группы периодов объем задолженности становится равным нулю и управление $Q(k)$ принимает очевидно оптимальное в данном случае нулевое значение. Реализация оставшейся части портфеля ценных бумаг продолжается с максимально возможной скоростью, позволяющей выйти в последнем периоде на нулевой объем портфеля. Скорость аккумулирования средств на счете операции при этом несколько возрастает.

Важно отметить, что полученная оценка максимальной эффективности операции $S(N) = 5456.45$ является "верхней оптимистичной", поскольку добавление в неполную модель пополняющих ограничений сужает множество условно допустимых состояний (точнее, не расширяет его), что может привести к уменьшению значения оценки эффективности. Изменение же значений параметров или структуры связей должно рассматриваться как модификация неполной модели, требующей пересчета полученных результатов.

Как видно из графиков на рис. 8.2.4.1 зависимости $V(k)$ и $Q(k)$ нелинейны для первой группы периодов. Аналогичное заключение можно сделать о зависимостях $F(k)$ и $R(k)$ для второй группы периодов, а также о $S(k)$ для третьей и четвертой групп. Определим характер этих нелинейностей, использовав аппроксимации данных зависимостей кусочно-непрерывными функциями $V(t), Q(t), F(t), R(t)$ и $S(t)$, для значений $t \in [0, T]$.

Выберем масштаб времени так, чтобы конец k –го периода совпадал с моментом $t = k$. Тогда $T = N$ и можно принять, что

$$F(k+1) - F(k) = F'(t); \quad V(k+1) - V(k) = V'(t) \quad \text{и}$$
$$S(k+1) - S(k) = S'(t),$$

а система разностных уравнений (8.2.2.1) может быть аппроксимирована системой дифференциальных уравнений

$$\begin{cases} F' = R\,, \\ V' = Q\,, \\ S' = \rho F - \rho_0 V - R + Q\,, \end{cases} \qquad (8.2.4.1)$$

при условиях:

$$F(0) = V(0) = S(0) = 0\,; \qquad F(T) = V(T) = 0\,;$$
$$F(t) \ge 0\,; \quad V(t) \ge 0\,; \quad S(t) \ge 0\,; \quad \forall t \in [0,T]\,;$$
$$\underline{R} \le R(t) \le \bar{R}\,; \quad \underline{Q} \le Q(t) \le \bar{Q}\,.$$

Наконец, целевой функционал будет иметь вид:

$$S(T) \to \max_{R,Q}\,.$$

Рассмотрим *первую группу* периодов. Здесь мы имеем

$$S(t) = 0\,; \quad R(t) = \bar{R} \quad \text{и} \quad F(t) = \bar{R}t\,.$$

Поэтому система (8.2.4.1) упрощается и принимает вид

$$\begin{cases} V' = Q\,, \\ 0 = \rho \bar{R} t - \rho_0 V - \bar{R} + Q\,, \end{cases}$$

что приводит к задаче Коши для $V(t)$

$$\begin{cases} V' = \rho_0 V - \rho \bar{R} t + \bar{R}\,, \\ V(0) = 0\,, \end{cases}$$

решением которой является функция

$$V(t) = \kappa e^{\rho_0 t} + \alpha t + \beta\,,$$

где α, β и κ – однозначно определяемые константы.

Рассмотрим теперь *вторую группу* периодов, предполагая, что первая группа завершилась в момент времени t_1 со значениями $V(t_1) = V_1$ и $F(t_1) = F_1$. Для второй группы имеем

$$S(t) = 0\,; \quad Q(t) = \underline{Q} \quad \text{и} \quad V(t) = V_1 + \underline{Q}(t - t_1)\,,$$

поэтому

$$\begin{cases} F' = R\,, \\ 0 = \rho F - \rho_0 (V_1 + \underline{Q}(t - t_1)) - R + \underline{Q}\,, \end{cases}$$

что приводит к задаче Коши для $F(t)$

$$\begin{cases} F' = \rho F - \rho_0 (V_1 + \underline{Q}(t - t_1)) + \underline{Q}\,, \\ F(t_1) = F_1\,, \end{cases}$$

с решением вида $F(t) = \mu e^{\rho t} + \gamma t + \delta$, где γ, δ и μ – однозначно определяемые константы.

В заключение рассмотрим *третью*[12] *группу* периодов, предположив, что вторая группа завершается в момент времени t_2 со значениями $V(t_2) = V_2$ и $F(t_2) = F_2$. Для третьей группы имеем

$$Q(t) = \underline{Q} \quad \text{и} \quad V(t) = V_2 + \underline{Q}(t - t_2)\,,$$

$$R(t) = \underline{R} \quad \text{и} \quad F(t) = F_2 + \underline{R}(t - t_2)\,,$$

поэтому

$$\begin{cases} F' = \underline{R}\,, \\ V' = \underline{Q}\,, \\ S' = \rho(F_2 + \underline{R}(t - t_2)) - \rho_0 (V_2 + \underline{Q}(t - t_2)) - \underline{R} + \underline{Q}\,. \end{cases}$$

Окончательно мы получаем задачу Коши вида

[12] Исследование четвертой группы периодов полностью аналогично исследованию третьей.

$$\begin{cases} S' = \rho(F_2 + \underline{R}(t - t_2)) - \rho_0(V_2 + \underline{Q}(t - t_2)) - \underline{R} + \underline{Q}, \\ S(t_2) = 0, \end{cases}$$

решением которой является функция $S(t) = \theta t^2 + \vartheta t + \sigma$, где θ, ϑ и σ – однозначно определяемые константы.

§8.2.5. Вариант расчета для случая изменяющихся уровней доходности и стоимости кредита

В данном варианте расчета значения доходности и стоимости кредита предполагались непостоянными и задаваемыми функциями вида:

$$\rho_0(k) = \begin{cases} a_0, & k < k_0, \\ b_0, & k_0 \le k \le k_0 + \Delta, \\ a_0, & k > k_0 + \Delta, \end{cases}$$

$$\rho(k) = \begin{cases} a, & k < k_1, \\ b, & k_1 \le k \le k_1 + \Delta, \\ a, & k > k_1 + \Delta \end{cases} \qquad (8.2.5.1)$$

с фиксированными значениями параметров

$$a_0 = 0.004,\ b_0 = 0.006,\quad a = 0.008,\ b = 0.01$$

и $$k_0 = 50,\ k_1 = 150,\ \Delta = 50.$$

Входной файл этого варианта модели (`Invest06.l`) получается добавлением в L-файл базового варианта (`Invest04.l`) следующих строк:

```
// амплитуда скачка
const double dr = 0.002;
//
// номер периода начала скачка доходности
const int tpskach = 150;
//
```

```
// номер периода начала скачка стоимости кредита
const int toskach = 50;
//
// длительность скачков (в периодах)
const int lnskach = 50;
//
// номер периода конца скачков доходности и стоимости кредита
int tpend;
int toend;
tpend = tpskach + lnskach;
toend = toskach + lnskach;
//
// Изменение величин pk[j] и po[j]
for( j=tpskach; j<=tpend; j++ )
      { pk[j]=rk + dr; };
for( j=toskach; j<=toend; j++ )
      { po[j]=r0 + dr ; };
```

Как и в базовом варианте описание оптимальной динамики значений показателей модели удобно выполнить, разделив множество всех периодов на непересекающиеся подмножества, как показано в таблице 8.2.5.2.

На рис. 8.2.5.3 и 8.2.5.4 приведены графики, отражающие динамику изменения значения показателей, обеспечивающую максимизацию остатка средств на счете операции при ее завершении. Подробные результаты расчетов для данного варианта находятся в файле `Invest06_Result.xls`.

Таблица 8.2.5.2.

№	Значения k	Описание динамики
1	$1 \le k \le 50$	Для данной группы периодов, как и в базовом варианте, основной специфической характеристикой оптимальной динамики является увеличение объема портфеля ценных бумаг $F(k)$ с максимально возможной скоростью.

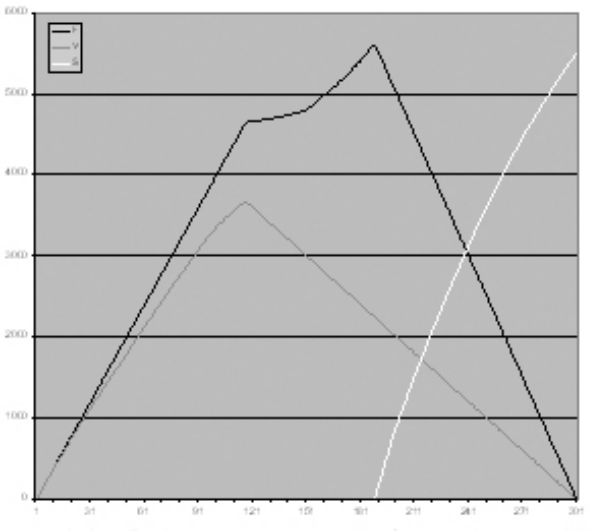


Рис. 8.2.5.3

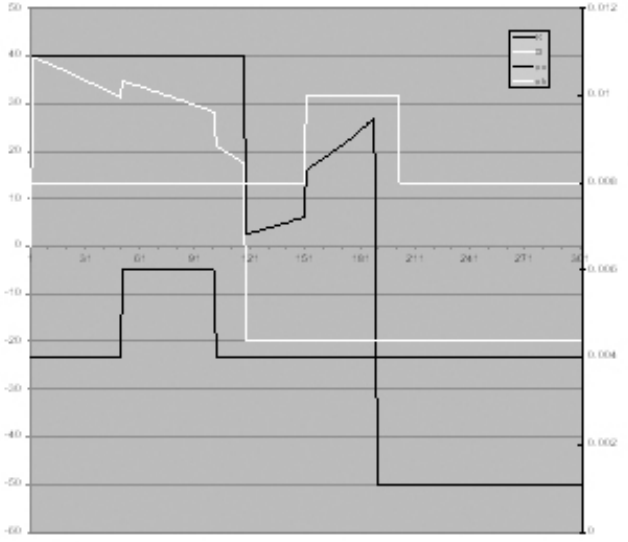


Рис. 8.2.5.4

		Величина управления $R(k)$ – скорости приобретения-продажи ценных бумаг – находится на верхней границе его допустимых значений. Оплата приобретаемых бумаг осуществляется как за счет привлекаемых заемных средств, так и реинвестирования дивидендов от уже приобретенных бумаг. Причем, по мере роста объема портфеля, скорость привлечения заемных средств $Q(k)$ снижается, хотя и остается положительной. Остаток средств на счете операции $S(k)$ для данных периодов нулевой.
2	$51 \le k \le 100$	В этой группе периодов, как следствие увеличенной стоимости кредита, скорость привлечения заемных средств выше по сравнению с базовым вариантом. Однако увеличение объема портфеля ценных бумаг $F(k)$ происходит по-прежнему с максимально возможной скоростью, и величина управления $R(k)$ – скорости приобретения-продажи ценных бумаг – находится на верхней границе его допустимых значений. Оплата приобретаемых бумаг осуществляется как за счет привлекаемых заемных средств, так и реинвестирования дивидендов от уже приобретенных бумаг. По мере роста объема портфеля скорость привлечения заемных средств $Q(k)$ снижается, но остается положительной, причем существенно большей, чем в базовом варианте. Остаток средств на счете операции $S(k)$ для данных периодов нулевой.

3	$101 \le k \le 117$	После завершения скачка стоимости кредита динамика фаз и управлений снова демонстрирует характерные особенности первой группы периодов.
4	$118 \le k \le 150$	Для этой группы периодов, как и в базовом варианте, характерной особенностью является начало возврата привлеченных заемных средств, снижение текущего объема задолженности $V(k)$ с максимально возможной скоростью. Значение соответствующего управления – $Q(k)$ скорости привлечения-возврата заемных средств – отрицательно и находится на нижней границе допустимых значений. На этом этапе рост объема ценных бумаг оказывается еще возможным, но со скоростью меньшей, чем допустимый максимум. Дивиденды тратятся в первую очередь на возврат займа, а их остающаяся часть реинвестируется в ценные бумаги. Остаток средств на счете операции для данной группы периодов по-прежнему нулевой.
5	$151 \le k \le 190$	В начале этой группы периодов происходит скачок доходности ценных бумаг. По этой причине скорость увеличения объема портфеля возрастает, который достигает своего абсолютного максимума на конечном периоде группы. Заемные средства возвращаются с максимально возможной скоростью. Аккумулирование свободных средств не происходит.
6	$191 \le k \le 200$	Эта группа, состоящая из сравнительно небольшого числа периодов, характеризуется, во-первых, началом процесса ликвидации портфеля ценных бумаг, объем которого уменьшается с максимально возможной скоростью, и, во-вторых, завершением скачка доходности.

		Существенно отметить, что для данной группы периодов повышенный уровень доходности не позволяет увеличить объем портфеля, поскольку это привело бы к нарушению краевого условия $F(N) = 0$. Значения обоих управлений находятся на своих нижних допустимых границах. Начинается процесс аккумулирования средств, получаемых от реализации ценных бумаг и дивидендов от нереализованной части портфеля, на счете операции, остаток которого $S(k)$ растет.
7	$201 \le k \le 300$	Для данной группы периодов значения обоих управлений находятся на своих нижних допустимых границах, что позволяет в последнем периоде получить нулевые значения фазовых переменных $F(k)$ и $V(k)$. Таким образом, завершается как процесс реализации портфеля, так и погашения задолженности по кредитной линии. Аккумулирование средств на счете операции завершается в последнем периоде с максимальным значением $S(k)$.

Верхняя оценка эффективности $S(N)$ в рассматриваемом случае равна 5497.88, то есть несколько отличается от базового варианта, и хотя данное отличие (и даже его знак) не является очевидным, изменения оптимальной динамики фаз и управлений могут служить для "лиц принимающих решения" определенным ориентиром в процессе анализа множества условно допустимых состояний.

Раздел 8.3. РЕШЕНИЕ СЕРИЙ ЗАДАЧ НЕПОЛНОГО МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

§8.3.1. Интерфейсная оболочка MultiLC

Сравнивая верхние оценки эффективности для трех рассмотренных вариантов, можно прийти к заключению, что значения этих оценок

существенно зависят от величин варьируемых параметров – амплитуд и начальных периодов скачков доходности и стоимости заемных средств.

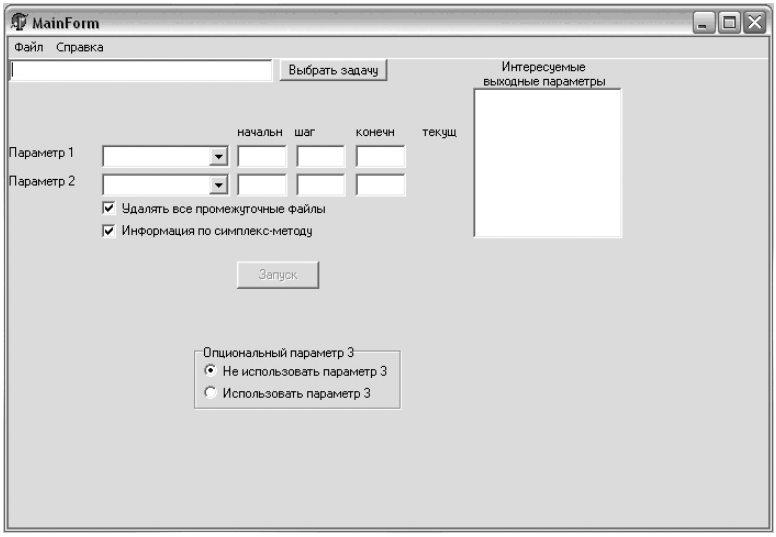


Рис. 8.3.1.1

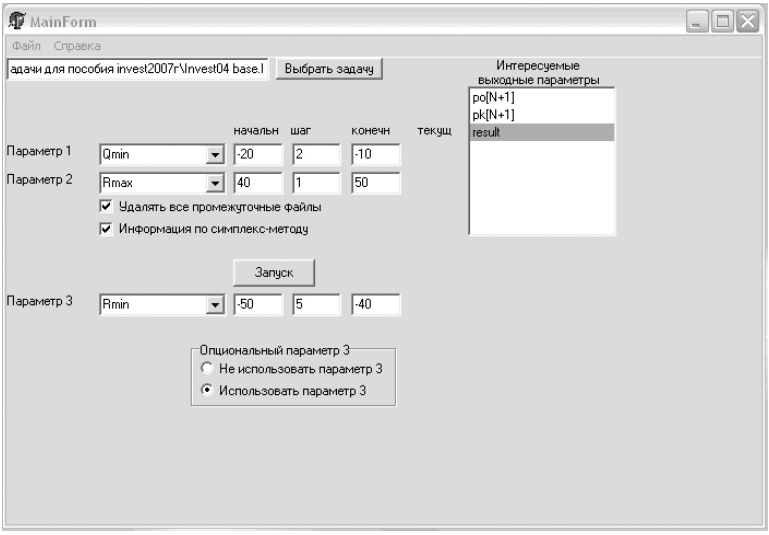


Рис. 8.3.1.2

Поскольку изменения значений параметров неполных моделей подобного рода (согласно классификации, данной в §6.3.1) следует относить к модификации, то в рассматриваемом случае объектом исследования является не одна неполная модель, а некоторое их множество. При этом на этапе автоматического анализа мы естественным образом приходим к задаче получения целого массива оценок для набора модифицированных неполных моделей, отличия между которыми задаются значениями некоторого набора параметров, изменяющихся по заранее заданным правилам.

Для решения данной проблемы было разработано специальное математическое обеспечение – программа `MultiLc`, являющейся в среде `MS Windows` интерфейсной оболочкой интерпретатора языка `L` – программы `Lc`, которая позволяет осуществлять многократный запуск интерпретатора с автоматическим внесением изменений значений выбранных параметров во входном `L`-файле, а также записью получаемых результатов в файлы выходных данных. Вид этой интерфейсной оболочки при запуске программы показан на рисунке 8.3.1.1.

Для использования программы `MultiLc` необходимо выбрать при помощи выпадающих меню:

1. входной `L`-файл, содержащий описание базового варианта неполной модели, при нажатии кнопки "**Выбрать задачу**" в стандартном окне "`Open`";
2. до трех варьируемых параметров из предлагаемых списков "**Параметр 1**", "**Параметр 2**" и "**Опциональный параметр 3**" с указанием величины шага и диапазона изменения для каждого из них;
3. любой (непустой) набор выходных параметров из списка "**Интересуемые выходные параметры**" (см. рис. 8.3.1.2).

После чего нажатием кнопки "**Запуск**" начать процесс решения задач, в ходе которого последовательно изменяются значения варьируемых параметров со следующим вложением циклов:

```
Цикл по параметру 3 (Цикл по параметру 1 (Цикл
          по параметру 2) ) ).
```

Вид интерфейсной оболочки при выполнении программы показан на рис. 8.3.1.3.

При работе с программой `MultiLc` следует иметь в виду, что:

1) если начальное и конечное значение варьируемого параметра совпадают, то выполняется только один шаг соответствующего цикла при начальном значении параметра;

2) в список параметров, из которых можно выбирать параметры для варьирования, включаются все параметры входного `L`-файла, каждый из которых объявлен отдельной командой (не списком!) как `const`;

3) в список параметров, из которых можно выбирать выходные параметры, включаются все параметры входного `L`-файла, каждый из которых объявлен отдельной командой как `double`;

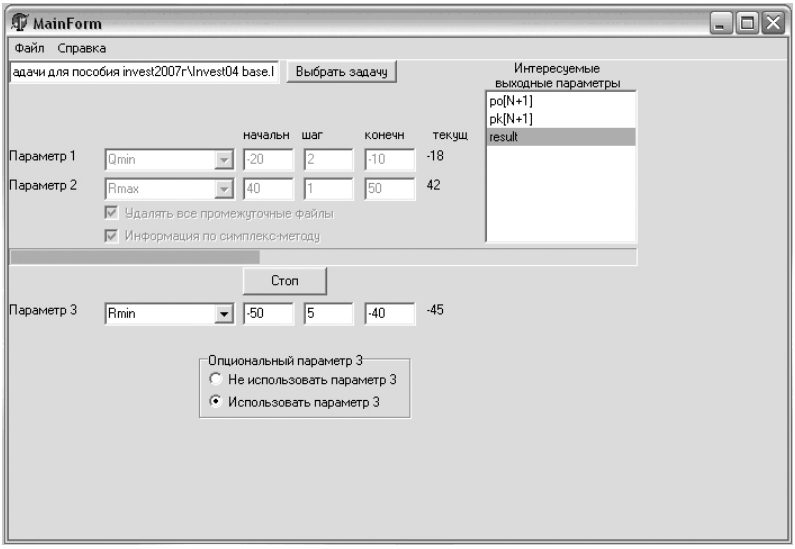


Рис. 8.3.1.3

4) выходные данные, включающие для каждой из решенных задач:
   - значения варьируемых и выходных параметров;
   - диагностику полученного решения (`оптимальное`, `несовместное`, `неограниченное`, `найденное с потерей точности` или `зацикленное`);

- а также (опционально) затраты вычислительных ресурсов, таких как: число выполненных итераций, использованная оперативная память, затраченное процессорное время – записываются в файл `MultiLcOutput.txt`.

§8.3.2. Параметрический анализ модели инвестиций на рынке ценных бумаг

В качестве примера опишем процедуру анализа зависимости маргинальной оценки величины остатка средств на счете операции при ее завершении от значений параметров k_0 и k_1 – начальных периодов для скачков стоимости кредита и доходности на рынке ценных бумаг.

Для модели, описанной в §8.2.1, требовалось исследовать зависимость $S(N)$ от k_0 и k_1. При условии, что $\rho_0(k)$ и $\rho(k)$ являются кусочно-постоянными функциями вида:

$$\rho_0(k) = \begin{cases} a_0\ , & k < k_0, \\ b_0\ , & k_0 \le k \le k_0 + \Delta, \\ a_0\ , & k > k_0 + \Delta, \end{cases}$$

$$\rho(k) = \begin{cases} a\ , & k < k_1, \\ b\ , & k_1 \le k \le k_1 + \Delta, \\ a\ , & k > k_1 + \Delta. \end{cases}$$

На рис.8.3.1.4 и 8.3.1.5 показаны графические представления искомых зависимостей, полученные при помощи программы `MultiLc`, для следующих диапазонов значений варьируемых параметров:

$$a_0 = 0.004\ ,\ b_0 = 0.006\ ,\quad a = 0.008\ ,\ b = 0.01$$

и $\quad a_0 = 0.004\ ,\ b_0 = 0.002\ ,\quad a = 0.008\ ,\ b = 0.01\ ,$

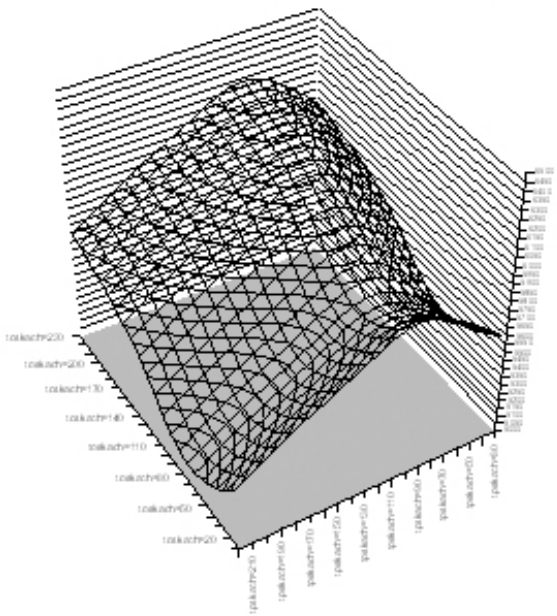


Рис. 8.3.1.4.

причем в обоих случаях выбирались $N = 300, \Delta = 50$, а

$$20 \le k_0 \le 230, \quad 30 \le k_1 \le 220$$

с постоянным шагом изменения, равным 10. Результаты численных расчетов были взяты из демо-файлов `Invest06_param1.xls` и `Invest06_param2.xls`.

В качестве иллюстрации приведем краткий качественный анализ полученных результатов.

Этот анализ показывает, что в случае нестабильного поведения рынка ценных бумаг, проявляющегося в скачке дохода вида (8.2.5.1) в момент времени k_1 существует, и притом единственная, реакция рынка кредитных линий, заключающаяся в скачке стоимости кредита в момент времени k_0, при которой достигается максимальная эффективность операции.

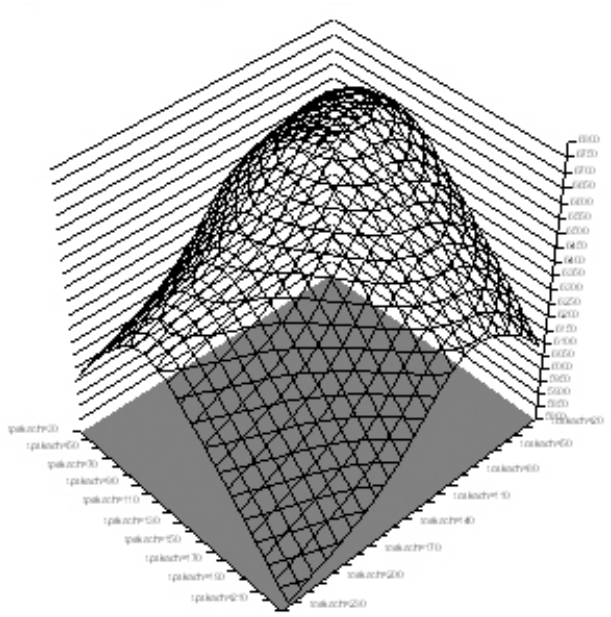


Рис. 8.3.1.5.

Иначе говоря, оптимальное значение k_0 оказывается некоторой поддающейся оценке функцией от k_1, и это обстоятельство может быть использовано регулирующими органами для улучшения условий функционирования финансовых рынков в целом или, например, в целях оптимизации налогообложения.

В заключение укажем на возможное естественное обобщение модели, состоящее во включении в число неизвестных показателей $\rho_0(k)$ и $\rho(k)$. Формально это приводит к следующей оптимизационной задаче:

максимизировать $S(N)$

при условиях:

$$S(k+1) = S(k) + Q(k) - R(k) + \rho(k) \cdot F(k) - \rho_0(k) \cdot V(k),$$
$$F(k+1) = F(k) + R(k),$$

$$V(k+1) = V(k) + Q(k), \quad \forall k = [1, N-1], \tag{8.3.1.1}$$

и $\forall k \in [1, N]$,

$$S(k) \geq 0, \quad V(k) \geq 0, \quad F(k) \geq 0,$$
$$\underline{\underline{Q}}(k) \leq Q(k) \leq \overline{Q}(k), \quad \underline{R}(k) \leq R(k) \leq \overline{R}(k),$$
$$S(1) = V(1) = F(1) = 0, \quad V(N) = F(N) = 0,$$

являющейся задачей нелинейного программирования (см. раздел 3.5.)

Размерность этой задачи слишком высока, чтобы можно было непосредственно применить для ее решения какой-либо из стандартных методов. Однако, если принять во внимание *специфику структуры* задачи (8.3.1.1), заключающуюся в том, что эта задача *линейна* при фиксированных $\rho_0(k)$ и $\rho(k)$, оказывается возможным, например, использование следующей *двухуровневой* схемы решения.

На *верхнем* уровне по методам, рассмотренным в разделе 5.5, решается задача:

максимизировать $S(N)$ *по совокупности*
допустимых значений $\rho_0(k)$ *и* $\rho(k)$.

При этом для каждой итерации при пересчете $\rho_0(k)$ и $\rho(k)$ может потребоваться (и, возможно, неоднократное) решение задачи *нижнего* уровня (8.3.1.1).

Глава 9.
ЯЗЫК ПРОГРАММИРОВАНИЯ L – СРЕДСТВО ОПИСАНИЯ ДАННЫХ И ОПЕРАЦИЙ ДЛЯ НЕПОЛНЫХ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ

Раздел 9.1. ЯЗЫК L. ВЫЗОВ ИНТЕРПРЕТАТОРА

Вызов интерпретатора в среде MS-DOS

Программа находится в одном модуле (исходном файле Lc.exe). При ее вызове вначале выполняется фаза лексического анализа, а затем фаза интерпретации. В текущей директории должен находиться служебный файл `newtask.ld`. Вызов в среде `MS-DOS` имеет следующий формат:

```
Lc <options> filename.l
```

со значениями опций:

- `-v0` – не выдавать информации о ходе решения
- `-v1` – выдавать информацию:
  - статус задачи и затраченное время
- `-v2` – выдавать информацию:
  - статус задачи и затраченное время
  - информация о созданных группах целевых границ
- `-v3` – выдавать информацию:
  - статус задачи и затраченное время
  - информация о созданных группах целевых границ
  - информация о расстоянии

`-v4`	–	(значение по умолчанию) выдавать информацию:
	–	статус задачи и затраченное время
	–	информация о созданных группах целевых границ
	–	информация о расстоянии
	–	информация об итерациях
`-son`	–	(значение по умолчанию) использовать звук
`-soff`	–	не использовать звук, –

позволяющими задавать степень детализации диагностики для автоматически выполняемых фаз работы программы.

Если `filename` = '`-`', то интерпретатор будет обрабатывать входной поток (`stdin`).

В случае обнаружения ошибок в исходном тексте или при возникновении системных или внутренних ошибок выдается диагностика вида:

```
Error in line <linenum> : <описание ошибки>
```

Вызов интерпретатора в среде MS Windows

В среде `MS Windows` имеются две возможности использования интерпретатора языка `L`:

1. Вызов файла `Lc.exe` в командном окне интерпретатора `MS-DOS` по правилам, описанным выше в данном параграфе;
2. Запуск интерфейсной оболочки `MultiLc.exe` – приложения `MS Windows`, предназначенного для многократного вызова программы `Lc`, например для выполнения параметрического анализа неполных математических моделей. Правила пользования программой `MultiLc` приведены в § 7.5. Запуск этого приложения может выполняться из любого каталога, содержащего (как минимум) файлы

```
MultiLc.exe
Lc.exe
newtask.ld.
```

При этом L-файлы с описаниями неполных моделей могут находиться в любом доступном каталоге.

Раздел 9.2. Язык L. РУКОВОДСТВО ПРОГРАММИСТА

Лексические правила

Используемые лексемы

В языке L используются следующие классы лексем:

```
идентификаторы,
ключевые слова,
константы,
стринговые литералы,
операторы и
разделители.
```

Пробел, горизонтальные и вертикальные табуляции, символ перехода на новую строку (newlines) и комментарии, имеющие общее название "пробельные литеры" – (whitespaces) рассматриваются интерпретатором только как разделители лексем и в остальном на результат интерпретации влияния не оказывают. Любая из пробельных литер годится, чтобы отделить друг от друга соседние идентификаторы, ключевые слова и константы. Если предположить, что входной поток разбит на лексемы, то следующей лексемой будет являться самая длинная цепочка символов, которая может быть лексемой.

Комментарии

Литеры /* открывают комментарий, а литеры */ закрывают его. Комментарии такого (C-подобного) типа могут быть вложенными.

Другой тип комментариев открывается литерами // и заканчивается символом перехода на новую строку. Внутри комментариев /* */ игнорируются литеры //, внутри комментариев // игнорируются литеры

/* и */. Комментарии нельзя помещать внутри строк. Будучи помещенными внутри строк, литеры /*, */ и // воспринимаются как часть строки.

Идентификаторы

Идентификатор – последовательность букв и цифр. Первой литерой должна быть буква, знак подчеркивания _ считается буквой. Буквы нижнего и верхнего регистра различаются. Допускается использование только букв латинского алфавита.

Идентификаторы могут иметь длину до 1000 символов.

На языке LEX'а определение идентификатора можно записать следующим образом:

```
IDENTIFICATOR [_a-zA-Z][_a-zA-Z0-9]*
```

Ключевые слова

Следующие идентификаторы зарезервированы в качестве ключевых слов и в другом смысле использоваться не могут:

```
const    static    volatile    register    int
double   short     string      real        struct
if       else      for         while       do
return

lOblBound    uOblBound    lDesBound    uDesBound
lRang        uRang        value        dualEst
lFixLev      uFixLev      link         string
real
```

Константы

Существует несколько видов констант. Каждая из них имеет свой тип данных (типы данных рассматриваются ниже).

Целые константы

Целая константа, состоящая из последовательности цифр, воспринимается как восьмеричная, если она начинается с `0` (цифры `0`), и как десятичная – в противном случае. Восьмеричная константа не содержит цифр `8` и `9`. Последовательность цифр, перед которой стоит `0x` или `0X`, рассматривается как шестнадцатеричное целое.

На языке `LEX`'а определение целой константы можно записать следующим образом:

```
D                                [0-9]
H                                [0-9a-fA-F]
INT_CONSTANT                         {D}+   |
                                   0X{H}+   |
                                      0x{H}+
```

Константы с плавающей точкой

Плавающая константа состоит из целой части, десятичной точки, дробной части, `e` или `E` и целого (возможно, со знаком), представляющего экспоненту. И целая, и дробная часть представляют собой последовательность цифр. Либо целая часть, либо дробная часть (но не обе вместе) могут отсутствовать; также могут отсутствовать десятичная точка или `E` с экспонентой (но не обе одновременно).

На языке LEX'а определение плавающей константы можно записать следующим образом:

```
D                        [0-9]
E                        [DEde][-+]?{D}+
FLOAT_CONSTANT           {D}+"."{D}*({E})? |
                         {D}*"."{D}+({E})? |
                                    {D}+{E}
```

Стринговые константы

Стринговая константа – это последовательность литер, заключенная в двойные кавычки (например, `"..."`). Рядом написанные стринговые литералы объединяются в один стринг. Внутри литерной контанты допускаются следующие `ESC`-последовательности:

newline (linefeed)	NL (LF)	\n
horisontal tab	HT	\t
vertical tab	VT	\v
backspace	BS	\b
carriage return	CR	\r
formfeed	FF	\f
bell	BEL	\a
backslash	\	\\
question mark	?	\?
single quote	'	\'
double quote	"	\"

Синтаксис

При описании синтаксиса используется форма Бакуса–Наура (Backus-Naur Form, BNF). Предполагается знакомство читателя с этим способом записи. Пометка opt свидетельствует о необязательности параметра.

Класс памяти

Все переменные, объявляемые в программе, являются статическими. Допускается при определении переменных использовать ключевые слова static и register. В данной версии интерпретатора эти ключевые слова игнорируются; впоследствии планируется сделать более быстрым обращение к "регистровым" переменным.

Запрещены к использованию ключевые слова extern и auto.

Базовые типы

Существует несколько базовых типов: два целых типа (int и short), один плавающий тип (double) и один тип для работы со строками (string). Кроме того, имеется тип real, определяющий показатели неполных математических моделей.

Размер числа типа int – 16 бит, также как и размер числа типа short – 16 бит.

Размер числа типа double – как минимум 8 байт. Размер типа определяется эффективностью различных типов для данной микропроцессорной архитектуры. Для данной реализации размер double – 8 байт (формат IEEE 8) под DOS, Unix и 12 байт (формат Think C Universal, соединение форматов MC68881 Extended и SANE Extended) – под Macintosh.

Тип string предназначен для работы со строками. Он поддерживает операции соединения и присваивания:

```
string a, b, c = "c" ;
a = "a" ;
b = a "b" ;
print( b c "\n" );
print( "\a" );
```

Тип real предназначен для работы с показателями задач неполного математического моделирования. Можно работать с типом real как с псевдоструктурой:

```
   struct real /* pseudo-struct */
   {
// Значения атрибутов, задаваемые  пользователем:
  double lOblBound;   // Нижняя обязательная граница
  double uOblBound;   // Верхняя обязательная граница
  double lDesBound;   // Нижняя желательная граница
  double uDesBound;   // Верхняя желательная граница
  double lRang;       // Ранг нижней желательной
                      // границы

  double uRang;       // Ранг верхней желательной
                      // границы

// Значения, которые находятся автоматически:

  double value;       // Значение показателя
  double dualEst;     // Двойственная оценка
  int    lFixLev;     // Номер группы, в которую
                      // попала нижняя граница
```

```
  int     uFixLev;      // Номер группы, в которую
                        // попала  верхняя граница
  link;                 // Указатель связи между
                        // показателями типа  real
            };
```

Назначение поля `link` объясняется ниже, в пункте "Псевдоструктура типа `real`". Остальные поля используются в соответствии с методикой применения неполных математических моделей.

Выводимые типы

Помимо базовых типов существует практически неограниченный класс выводимых типов, которые формируются из уже существующих и которые описывают следующие конструкции:

- Массивы объектов заданного типа. Массивы могут иметь любую размерность.
- Структуры, содержащие последовательность объектов, возможно, различных типов
  (включая типа `real`, другие структуры и массивы).

В общем случае приведенные методы конструирования объектов могут применяться рекурсивно.

Квалификаторы типов

Тип объекта может снабжаться квалификатором. Декларация объекта с квалификатором `const` указывает на то, что его значение далее не будет изменяться; квалификатор `volatile` можно использовать, но он не оказывает никакого влияния. Ни один из квалификаторов не влияет на диапазоны значений и арифметические свойства объектов.

Объекты

Объект – эта некоторая именованная область памяти. С объектом связывается `lvalue` – выражение, ссылающееся на объект. Очевидным примером `lvalue` является имя переменной.

Преобразования

Некоторые операторы в зависимости от своих операндов могут вызывать преобразования их значений из одного типа в другой.

Целочисленные преобразования

Типы `short` и `int` рассматриваются интерпретатором как эквивалентные.

Преобразования между целыми и плавающими типами

При преобразовании из плавающего типа в целый дробная часть отбрасывается; если полученное при этом значение нельзя представить в заданном целом типе, то результат не

определен. Если значение преобразуется из целого в величину с плавающей точкой и она находится в допустимом диапазоне, но представляется в новом типе неточно, то результатом будет одно из двух значений нового типа, ближайшего к исходному.

Арифметические преобразования

Во многих операциях преобразование типов осуществляется по одним и тем же правилам. Они состоят в том, что операнды приводятся к некоторому общему типу, который является и типом результата.

- Если операнд имеет тип `short`, он всегда вначале приводится к типу `int`.
- Если какой-либо из операндов имеет тип `double`, то другой операнд приводится к типу `double`.
- В противном случае оба операнда имеют тип `int`.

Выражения

Приоритеты описываемых операторов имеют тот же порядок, что и подразделы данного параграфа (от высших к низшим). В каждом подразделе описываются операторы, имеющие одинаковый приоритет, и

указывается их ассоциативность (левая или правая). Приоритеты и ассоциативность всех операторов полностью соответствуют языку C и описаны в § 8.3.

Порядок вычисления выражений в данной реализации интерпретатора – слева направо. Рекомендуется использование скобок при записи сложных выражений.

В процессе вычисления осуществляется контроль за делением на нуль. Контроль за переполнением и остальными исключительными ситуациями не осуществляется; поведение интерпретатора в таких ситуациях не определено.

Первичные выражения

Первичные выражения – это идентификаторы, константы, символьные цепочки (стринги) и выражения в скобках.

```
первичное_выражение
    : идентификатор
    | константа
    | стринг
    | ( выражение )
    ;
```

Идентификатор, если он был должным образом декларирован – первичное выражение. Идентификатор есть `lvalue`, если он обозначает объект типа `int`, `short`, `real`, `double` или "`struct`".

Константа – первичное выражение. Ее тип зависит от формы записи.

Стринг – первичное выражение. Его тип – `string`.

Выражение в скобках – первичное выражение. Его тип и значение соответствуют типу и значению этого же выражения без скобок. Наличие или отсутствие скобок не влияет на то, является ли данное значение `lvalue` или нет.

Постфиксные выражения

В постфиксных выражениях операторы выполняются слева направо.

```
постфиксное_выражение
: первичное_выражение
| постфиксное_выражение [ выражение ]
| постфиксное_выражение
( список_аргументов opt )
             | int ( выражение )
             | double ( выражение )
             | постфиксное_выражение .
               идентификатор
             | постфиксное_выражение ++
             | постфиксное_выражение --
             ;

          список_аргументов
              : выражение_присваивание
              | список_аргументов ,
                выражение_присваивание
              ;
```

Ссылки на элемент массива

Постфиксное выражение, за которым следует выражение в квадратных скобках, есть постфиксное выражение, обозначающее ссылку в индексируемый массив. Выражение должно иметь тип `int` или `short`, выход за пределы массива проверяется в момент обращения к массиву (`run-time bounds checking`). Если массив имеет тип `массив элементов T`, то тип результата индексирования – `T`. Для массивов строк этот оператор используется следующим образом:

```
string result, myString = "My String" ;
string strArray[ 3 ] = { "a", "b", "c" };
         result = myString ;
         print( result "\n" );

         result = (string) strArray[ 2 ];
         print( result "\n" (string)
                strArray[ 0 ] "\n" );
```

то есть для получения ссылки на элемент массива строк нужно добавить (string).

Вызов функции

Вызов функции есть постфиксное выражение, за которым следуют скобки, содержащие (возможно, пустой) список разделенных запятыми выражений-присваиваний, представляющих собой аргументы этой функции. Допускается только вызов функций, описанных в библиотеке стандартных функций. Язык L не предусматривает возможности определения других функций. Сами функции, вызываемые в программе, реализованы на языке C с использованием стандартных библиотечных функций C и выполняются максимально эффективно. При вызове функции аргументы передаются по значению. Порядок вычисления аргументов – слева направо. Все аргументы вычисляются полностью до входа в функцию.

Преобразование типа

Постфиксное выражение вида int(выражение) или double(выражение) является постфиксным выражением. Значением такого выражения является значение его операнда, преобразованного к типу int или double соответственно. Результат преобразования не является lvalue. Выражение должно иметь арифметический тип.

Ссылки на члены структуры

Постфиксное выражение, за которым стоит точка с последующим идентификатором, является постфиксным выражением. Выражение первого операнда должно быть структурой или real, а идентификатор – именем члена структуры или именем члена "псевдоструктуры" real. Значе ние –

именованный член структуры (кроме члена link для real). Выражение является lvalue, если тип второго выражения – не массив.

Постфиксное инкрементирование и декрементирование

Постфиксное выражение, за которым следует ++ или --, есть постфиксное выражение. Значением такого выражения является значение его операнда. После того как значение было взято, операнд увеличивается (++) или уменьшается (--) на 1. Операнд должен быть lvalue и иметь

арифметический тип. Результат инкрементирования или декрементирования не есть `lvalue`.

Унарные операторы

Выражения с унарными операторами выполняются справа налево.

```
унарное_выражение
    : постфиксное_выражение
    | ++ унарное_выражение
    | -- унарное_выражение
    | унарный_оператор выражение
    ;
унарный_оператор
    : +
    | -
    | ~
    | !
    ;
```

Префиксные инкрементирование и декрементирование

Унарное выражение, перед которым стоит `++` или `--`, есть унарное выражение. Операнд увеличивается (`++`) или

уменьшается (`--`) на `1`. Значением выражения является значения его операнда после увеличения (уменьшения). Операнд всегда – `lvalue`. Результат инкрементирования и декрементирования не есть `lvalue`.

Оператор унарный плюс

Операнд унарного "+" должен иметь арифметический тип, результат – значение операнда.

Оператор унарный минус

Операнд унарного "-" должен иметь арифметический тип, результат – значение операнда с противоположным знаком.

Оператор обращения разрядов

Операнд оператора ~ должен иметь целочисленный тип, результат – дополнение операнда до единиц по всем разрядам. Результат не является lvalue.

Оператор логического отрицания

Операнд оператора ! должен иметь арифметический тип. Результат равен 1, если сравнение операнда с 0 дает истину, и равен 0 – в противном случае. Тип результата – int.

Мультипликативные операторы

Мультипликативные операторы *, / и % выполняются слева направо.

```
мультипликативное_выражение
    : унарное_выражение
    | мультипликативное_выражение *
      унарное_выражение

    | мультипликативное_выражение /
      унарное_выражение
    | мультипликативное_выражение %
      унарное_выражение
    ;
```

Операнды операторов * и / должны быть арифметического типа, оператора % – целочисленного типа. Над операндами осуществляются обычные арифметические преобразования, которые приводят их значения к типу результата. Бинарный оператор * обозначает умножение, бинарный оператор / получает частное, а % – остаток от деления первого операнда на второй; если второй операнд есть 0, то результат не определен.

Аддитивные операторы

Аддитивные операторы + и – выполняются слева направо.

```
аддитивное_выражение
    : мультипликативное_выражение
    | аддитивное_выражение + мультипли-
кативное_выражение
    | аддитивное_выражение + мультипли-
кативное_выражение
    ;
```

Операнды операторов + и – должны быть арифметического типа (за исключением работы с полем `link` типа `real`). Над операндами осуществляются обычные арифметические преобразования, которые приводят их значения к типу результата. Бинарный оператор + обозначает сложение, бинарный оператор – обозначает вычитание.

Операторы сдвига

Операторы сдвига << и >> выполняются слева направо. Для обоих операторов каждый операнд должен иметь целочисленный тип. Результат не определен, если правый операнд отрицателен или его значение превышает число битов в типе левого выражения или равно ему (в данной версии интерпретатора >= 16).

```
сдвиговое_выражение
    : аддитивное_выражение
    | сдвиговое_выражение >>
      аддитивное_выражение
    | сдвиговое_выражение <<
      аддитивное_выражение
    ;
```

Значение `E1 << E2` равно значению `E1` (рассматриваемому как цепочка битов), сдвинутому влево на `E2` бит; при отсутствии переполнения такая операция эквивалентна умножению на 2 в степени `E2`. Значение `E1 >> E2` равно значению `E1` (рассматриваемому как цепочка битов), сдвинутому вправо на `E2` бит; если `E1` имеет неотрицательное значение, то такая операция эквивалентна делению на 2 в степени `E2`.

Операторы отношения

Операторы отношения выполняются слева направо. Согласно грамматике языка выражение `a < b < c` трактуется так же, как `(a < b) < c`, а результат вычисления `a < b` всегда есть `0` или `1`.

```
выражение_отношения
    : сдвиговое_отношение
    | выражение_отношения <
      сдвиговое_отношение

    | выражение_отношения >
      сдвиговое_отношение
    | выражение_отношения <=
      сдвиговое_отношение
    | выражение_отношения >=
      сдвиговое_отношение
    ;
```

Операторы < (меньше), > (больше), <= (меньше или равно), >= (больше или равно) выдают `0`, если отношение `ложно` и `1`, если `истинно`. Тип результата – `int`. Операнды должны быть арифметического типа; над ними выполняются обычные арифметические преобразования.

Операторы равенства

```
выражение_равенства
    : выражение_отношения
    | выражение_равенства == выраже-
ние_отношения
    | выражение_равенства != выраже-
ние_отношения
    ;
```

Операторы == (`равно`) и != (`не равно`) являются аналогами операторов отношения с той лишь разницей, что они имеют более низкий приоритет.

Оператор побитового И

```
И_выражение
     : выражение_равенства
     | И_выражение &
       выражение_равенства
     ;
```

Выполняются обычные арифметические преобразования; результат – побитовое И операндов. Операнды должны быть целочисленного типа.

Оператор побитового исключающего ИЛИ

```
ислючающее_ИЛИ_выражение
     : И_выражение
     | исключающее_ИЛИ_выражение ^
       И_выражение
     ;
```

Выполняются обычные арифметические преобразования; результат – побитовое исключающее ИЛИ операндов. Операнды должны быть целочисленного типа.

Оператор побитового ИЛИ

```
ИЛИ_выражение
     : исключающее_ИЛИ_выражение
     | ИЛИ_выражение |
       исключающее_ИЛИ_выражение
     ;
```

Выполняются обычные арифметические преобразования; результат – побитовое ИЛИ операндов. Операнды должны быть целочисленного типа.

Оператор логического И

```
логическое_И_выражение
     : ИЛИ_выражение
     | логическое_И_выражение &&
       ИЛИ_выражение
     ;
```

Операторы && выполняются слева направо. В отличие от языка C оцениваются оба операнда. Операнды оцениваются слева направо. Оператор && выдает 1, если оба операнда не равны 0, и 0 – в противном случае. Операнды должны быть арифметического типа. Тип результата – int.

Оператор логического ИЛИ

```
логическое_ИЛИ_выражение
    : логическое_И_выражение
    | логическое_ИЛИ_выражение ||
      логическое_И_выражение
    ;
```

Операторы || выполняются слева направо. В отличие от языка C оцениваются оба операнда. Операнды оцениваются слева направо. Оператор || выдает 1, если по крайней мере один из операндов не равен нулю, и 0 – в противном случае. Операнды должны быть арифметического типа. Тип результата – int.

Условный оператор

```
условное_выражение
    : логическое_ИЛИ_выражение
    | логическое_ИЛИ_выражение ?
      выражение : условное_выражение
    ;
```

Вычисляется первое выражение, включая все побочные эффекты; если оно не равно 0, то результат есть значение второго выражения, в противном случае – значение третьего выражения. В отличие от языка C оцениваются все выражения. Порядок вычисления операндов – слева направо. Операнды должны быть арифметического типа; тип результата равен типу операнда, значение которого является результатом.

Выражение присваивания

Существуют несколько операторов присваивания; они выполняются справа налево.

```
выражение
    : условное_выражение
    | унарное_выражение
      опер_присваивания выражение
    ;

опер_присваивания
       /* оператор присваивания */
    :  =
    |  *=
    |  /=
    |  %=
    |  +=
    |  -=
    |  <<=
    |  >>=
    |  &=
    |  ^=
    |  |=
    ;
```

Операторы присваивания в качестве левого операнда требуют `lvalue`, причем модифицируемое; это значит, что оно не может быть массивом, или структурой, или типом с модификатором const. Тип выражения присваивания – тип его левого операнда, а значение – значение его левого операнда сразу после присваивания.

В простом присваивании с оператором = значение выражения замещает объект, на который ссылается `lvalue`. При этом должно выполняться одно из следующих условий: оба операнда имеют арифметический тип (если типы операндов разные, правый операнд приводится к типу левого операнда); оба операнда есть строки.

Выражение `E1 opt = E2` эквивалентно выражению `E1 = E1 opt (E2)` с одним исключением: `E1` вычисляется только один раз.

Оператор запятая

Оператор запятая, применяющийся в `C`, не реализован. При попытке его применения будет выдаваться синтаксическая ошибка.

Декларации

То, каким образом интерпретируется каждый идентификатор, специфицируется декларациями (declarations); они не всегда резервируют память для описываемых ими идентификаторов. Декларации, резервирующие память, называются определениями (definitions) и имеют следующий вид:

```
            декларация
                : спецификаторы_декларации спи-
сок_иниц_деклараторов opt ;
                ;
```

Декларатор в список_иниц_деклараторов содержат декларируемые идентификаторы; спецификаторы_декларации представляют собой последовательности, состоящие из спецификаторов типа и класса памяти.

```
            спецификаторы_декларации
                : спецификатор_класса_памяти
                  спецификаторы_декларации opt
                | спецификатор_типа
                  спецификаторы_декларации opt
                | квалификатор_типа
                  спецификаторы_декларации opt
                ;
            список_иниц_деклараторов
                : иниц_декларатор
                | список_иниц_деклараторов ,
                  иниц_декларатор
                ;
            иниц_декларатор
                : декларатор
                | декларатор = инициализатор
                ;
```

Декларатор содержат подлежащие описанию имена. Декларация должна иметь по меньшей мере один декларатор. Пустая декларация незаконна.

Спецификаторы класса памяти

Класс памяти специфицируется следующим образом:

```
спецификатор_класса_памяти
    : register
    | static
    ;
```

Смысл классов памяти пояснялся ранее. Декларация может содержать не более одного спецификатора памяти.

Спецификаторы типа

Спецификаторы типа определяются следующим образом:

```
спецификатор_типа
    : short
    | int
    | double
    | real
    | string
    | идентификатор_структуры
    ;
```

Декларация не может содержать более одного спецификатора типа. В декларации должен содержаться ровно один спецификатор типа (в отличие от языка C, где по умолчанию типом считается int).

Для указания особых свойств декларируемых объектов предназначаются квалификаторы:

```
квалификатор_типа
    : const
    | volatile
    ;
```

Квалификаторы типа могут употребляться с любым спецификатором типа. Const-объект разрешается инициализировать, однако присваивать ему что-либо в дальнейшем запрещается.

Декларации структур

Структура – это объект, состоящий из последовательности именованных членов различных типов. Декларация структуры имеет следующий вид:

```
декларация_структуры
    : struct идентификатор_структуры
     { список_структ_декл } ;
    ;
```

Список_структ_декл является непустой последовательностью деклараций членов структуры:

```
список_структ_декл
    : структ_декларация
    | список_структ_декл
      структ_декларация
    ;
структ_декларация
    : список_спецификаторов
      список_структ_деклараторов ;
    ;
список_спецификаторов
    : спецификатор_типа
      список_спецификаторов opt
    | квалификатор_типа
      список_спецификаторов opt
    ;

список_структ_деклараторов
    : структ_декларатор
    | список_структ_деклараторов ,
      структ_декларатор
    ;
```

Имена членов разных структур не конфликтуют между собой. Однако они конфликтуют с именами переменных, функций и именами структур (т.е. выведенных типов). Таким образом, имя члена структуры не должно использоваться для обозначения переменной, функции, структуры и не быть ключевым словом. Имя члена нельзя упоминать дважды в одной и той же структуре. Член структуры может быть объектом любого типа, включая массивы любой размерности и структуры.

Декларатор

Декларатор имеют следующий синтаксис:

```
декларатор
    : идентификатор
    | ( декларатор )
    | декларатор
      [ целое_выражение opt ]
    ;
```

Список деклараторов располагается сразу после спецификаторов типа и указателя класса памяти. Главный элемент любого декларатора – это объявляемый им идентификатор; в простейшем случае декларатор из одного его и состоит, что отражено в первой строке продукции грамматики с именем декларатор. Спецификаторы класса памяти относятся непосредственно к идентификатору, а его тип зависит от вида декларатора.

В декларации "`T D`", где `T` – тип, а `D` – просто идентификатор, тип идентификатора есть `T`.

В декларации "`T D`", где `D` имеет вид (`D1`), тип идентификатора в `D1` тот же, что и в `D`. Скобки не изменяют тип.

Декларатор массивов

В декларации `T D`, где `D` имеет вид

```
D1 [ целое_выражение opt ]
```

и где тип идентификатора декларации `T D1` есть `модификатор_типа T`, тип идентификатора `D` есть `модификатор_типа массив из T`. Если `целое_выражение` присутствует, то оно должно быть целочисленным и больше `0`. Отметим, что в отличие от языка `C` `целое_выражение` не должно быть константой, что компенсирует "урезанную" возможность использования динамических массивов. Если `целое_выражение` отсутствует, то массив имеет незавершенный вид.

Массив можно конструировать из объектов арифметического типа, строк, структур, `real` и других массивов (генерируя при необходимости и многомерные массивы). Любой тип, из которого конструируется массив, должен быть завершенным; он не может быть массивом незавершенного типа. Это значит, что для многомерного массива пустой может быть

только первая размерность. Незавершенный тип массива получает свое завершение при его инициализации; декларация незавершенного массива без инициализации незаконна:

```
int  ia[][2] = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
                // так правильно
double  da[]; // а это – ошибка
```

Инициализация

При помощи иниц_декларатора можно указать начальное значение декларируемого объекта. Инициализатору, представляющему собой выражение или список инициализаторов, заключенный в фигурные скобки, предшествует знак =. Этот список может завершаться запятой; ее назначение – сделать форматирование более четким.

```
инициализатор
    : выражение_присваивания
    | { список_инициализаторов }
    | { список_инициализаторов , }
    ;
список_инициализаторов
    : инициализатор
    | список_инициализаторов ,
      инициализатор
    ;
```

На инициализаторы не накладывается требование константности. Инициализаторы должны иметь тип, соответствующий объекту.

Объект арифметического типа, инициализация которого явно не указана, инициализируется 0. Массивы объектов арифметического типа не инициализируются.

Инициализатор массива – это список инициализаторов его членов, заключенный в фигурные скобки. Если размер массива не известен, то он считается равным числу инициализаторов, при этом тип его становится завершенным. Если

размер массива известен, то число инициализаторов не должно превышать число элементов массива.

Инициализация структур интерпретатором не поддерживается.

Инструкции

Инструкция – далеко не лучший перевод слова `statement`. В качестве еще одного (столь же далекого от идеала) перевода можно предложить слово утверждение. В дальнейшем в целях соответствия с русским переводом книги `K&R` будет употребляться слово инструкция.

За исключением оговоренных случаев инструкции выполняются в том порядке, как они написаны. Инструкции не имеют значений и выполняются, чтобы произвести определенные действия. Все виды инструкций можно разбить на следующие группы:

```
инструкция
    : инструкция_выражение
    | составная_инструкция
    | особая_инструкция
    | инструкция_выбора
    | циклическая_инструкция
    | инструкция_решателю
    ;
```

Инструкция-выражение

Наиболее употребительный вид инструкции – это инструкция_выражение :

```
инструкция_выражение
    : выражение opt ;
    | return ;
    ;
```

Чаще всего инструкция_выражение – это присваивание или вызов функции. Все действия, реализующие побочный эффект выражения, завершаются, прежде чем начнет выполняться следующая инструкция. Если выражение в инструкции опущено, то она называется пустой; пустая инструкция часто используется для обозначения пустого тела циклической инструкции. Если встречается инструкция

```
return ;
```

то интерпретация программы прекращается.

Составная инструкция

Так как в местах, где по синтаксису полагается одна инструкция, иногда возникает необходимость выполнить несколько, предусматривается возможность задания составной инструкции (которую также называют блоком).

```
составная_инструкция
    : { список_инструкций }
    ;

список_инструкций
    : инструкция
    | список_инструкций инструкция
    ;
```

Декларации внутри блока запрещены. Пространство имен в программе одно, общее для всех блоков.

Особая инструкция

Особая инструкция определяется следующим образом:

```
особая_инструкция
    : инструкция_выражение
    | составная_инструкция
    ;
```

Интерпретатор находит особую инструкцию следующим образом: если первый лексический символ – `{`, то особая инструкция является составной инструкцией, оканчивающейся символом `}`, соответствующим первому лексическому символу. В противном случае особой инструкции пренадлежат все символы до точки с запятой (`;`) включительно.

Инструкции выбора

Инструкции выбора осуществляют отбор одной из нескольких альтернатив, определяющих порядок выполнения инструкций.

```
инструкция_выбора
    : if ( выражение ) особая_инструкция
    | if ( выражение ) особая_инструкция
```

```
else особая_инструкция
                ;
```

Оба вида `if`-инструкций содержат выражение, которое должно иметь арифметический тип. Сначала вычисляется выражение в скобках со всеми его побочными эффектами и результат сравнивается с `0`. В случае несовпадения с `0` выполняется первая подинструкция. В случае совпадения с `0` для второго типа `if` выполняется вторая подинструкция. Отличие от `C` состоит в том, что подвыражения имеют тип не инструкция, а особая_инструкция. Таким образом, следующая программа неверна:

```
            if ( i != 0 )
                if ( j == 2 )
                    k = 3 ;

                else
                    k = 4 ;
```

Подобный код должен быть записан следующим образом:

```
            if ( i != 0 )
            {
                if ( j == 2 )
                    k = 3 ;
                else
                    k = 4 ;
            }
```

Циклические инструкции

Циклические инструкции специфицируют циклы:

```
            циклическая_инструкция
                : while ( выражение ) осо-
бая_инструкция
                | do особая_инструкция while ( выра-
жение )
                | for ( выражение opt ; выражение ;
выражение )
                        особая_инструкция
                ;
```

В инструкциях `while` и `do` выполнение подинструкции повторяется до тех пор, пока значение выражения не станет `0`. Выражение должно

иметь арифметический тип. В while вычисление выражения со всеми побочными эффектами и проверка осуществляется перед каждым выполнением инструкции, а в do – после.

В инструкции for первое выражение вычисляется один раз, тем самым осуществляется инициализация цикла. Само выражение может отсутствовать, но если оно присутствует, то тип этого выражения должен быть арифметическим. Второе и третье выражения должны присутствовать всегда и иметь арифметический тип. Побочные эффекты всех трех выражений заканчиваются по завершении их вычислений. Следует отметить, что C накладывает менее жесткие ограничения: в C первое выражение может иметь любой тип, а второе и третье могут отсутствовать.

Второе выражение вычисляется перед каждой итерацией. Как только его значение становится равным 0, for прекращает свою работу. Третье выражение вычисляется после каждой итерации и, следовательно, выполняет повторную инициализацию цикла. В целом конструкция

```
for ( выражение1 ; выражение2 ; выражение3 ) инструкция
```

эквивалентна конструкции

```
выражение1 ;
while ( выражение2 )
{
    инструкция
    выражение3 ;
}
```

Псевдоструктура типа real

При объявлении переменной типа real создаются внутренние структуры данных интерпретатора, содержащие информацию об атрибутах показателей, описанных в § 7.1. Напрямую эти структуры недоступны. Для работы с ними необходимо использовать специальную псевдоструктуру:

```
        struct real /* pseudo-struct */
        {
// Атрибуты показателя, значения которых
// задает пользователь
            double lOblBound;
            // Нижняя обязательная граница
            double uOblBound;
            // Верхняя обязательная граница
            double lDesBound;
            // Нижняя желательная граница
            double uDesBound;
            // Верхняя желательная граница
            double lRang;
             // Ранг нижней желательной границы
            double uRang;
            // Ранг верхней желательной границы
// Атрибуты показателя,
// значения которых вычисляются автоматически
            double value;
            // Значение показателя
            double dualEst;
            // Двойственная оценка
            int    lFixLev;
            // Номер группы, в которую попала
            // нижняя граница
            int    uFixLev;
            // Номер группы, в которую попала
            // верхняя граница
            link   ;
            // указатель связи между показателями
        };
```

При создании показателя первые 6 полей по умолчанию инициализируются со следующими значениями:

lOblBound	Нижняя обязательная граница	0
uOblBound	Верхняя обязательная граница	+inf
lDesBound	Нижняя желательная граница	-inf
uDesBound	Верхняя желательная граница	+inf

lRang	Ранг нижней желательной границы	1
uRang	Ранг верхней желательной границы	1

В случае, если требуются другие значения этих атрибутов, можно изменить их, воспользовавшись стандартным синтаксисом, например:

```
real r ;
r.lOblBound = 5 ;
```

Значения следующих четырех полей рассчитываются автоматически при помощи вызова функции `solve()` . Доступ к ним можно получить также стандартным образом:

```
double need, have ;
need = r.lDesBound ;
have = r.value ;
```

Последнее поле позволяет создать "связку" (`link`). Оно служит для задания связей между показателями:

```
инструкция_решателю
    : выражение_показатель :=
      список_связей ;
    ;
список_связей
    : коэффициент_связи
    | список_связей +
      коэффициент_связи
    | список_связей -
      коэффициент_связи
    ;
коэффициент_связи
    : выражение_показатель.link
    | выражение *
      выражение_показатель.link
    ;
```

Здесь `выражение_показатель` должен иметь тип `real`, а `выражение` – арифметический тип со значением *отличным от нуля*. Приведем пример:

```
real a, b, c ;
c := 2 * b.link - 24.5 * c.link ;
```

Особенностью использования данного синтаксиса является требование выполнения условия: `выражение_показатель` по правую сторону от `:=` должен быть определен в программе РАНЬШЕ, чем каждое `выражение_показатель` по левую сторону. Это требование определяется необходимостью иметь треугольную матрицу коэффициентов связей между показателями, определенную в § 7.1.

Пример использования переменных типа real

В качестве примера использования языка `L` приведем текст программы, находящей для каждой из пяти пар вещественных чисел выбрать максимальное. Сравниваемые числа имеют вид $x_1(k) = \cos(\frac{\pi k}{10})$; $x_2(k) = \sin(\frac{\pi k}{10})$, где k – номер пары.

Решаемая задача может быть сведена к набору из пяти задач линейного программирования следующего вида:

`минимизировать` $MAX(k)$ `при условиях:`

$$MAX(k) \geq X1(k),$$
$$MAX(k) \geq X2(k),$$
$$X1(k) = x_1(k),$$
$$X2(k) = x_2(k), \quad k = [1,5],$$

поскольку оптимальное значение показателя $MAX(k)$ будет очевидно совпадать с максимальным числом в *k*-й паре сравниваемых чисел.

Отметим также, что первые два неравенства равносильны условиям неотрицательности вспомогательных показателей

$$Y1(k) = MAX(k) - X1(k) \text{ и } Y2(k) = MAX(k) - X2(k).$$

Для каждой пары чисел в программе создается структура, содержащая как сами числа *x*1 и *x*2, так и показатели, необходимые для решения этой задачи решателем: *X*1 и *X*2, *MAX* и два вспомогательных показателя *Y*1 и *Y*2. Затем все структуры инициализируются в цикле, после чего ищется решение и выполняется форматированная распечатка результатов.

```
struct MaxType  // Определяем структуру типа MaxType
      {
          double x1;
          double x2;
          real   X1;
          real   X2;
          real   MAX;
          real   Y1;
          real   Y2;
      };

const int SIZE = 5 ;  // Задаем число пар сравниваемых чисел
MaxType max[ SIZE ] ; // Создаем массив из пяти объектов типа MaxType
double rad = M_PI / ( 2 * SIZE );

// Инициализируем структуры в цикле
          int i;
          for( i = 0 ; i < SIZE ; i++ )
          {
    // Присваиваем значения сравниваемым числам
              max[ i ].x1 = cos( i * rad ) ;
              max[ i ].x2 = sin( i * rad ) ;
    // Устанавливаем обязательные границы
    // для показателей X1 и X2
       max[ i ].X1.lOblBound =
       max[ i ].X1.uOblBound = max[ i ].x1 ;
       max[ i ].X2.lOblBound =
       max[ i ].X2.uOblBound = max[ i ].x2 ;

    // Отменяем установленное по умолчанию условие
    // неотрицательности для MAX
       max[ i ].MAX.lOblBound = -1e7;

    // Задаем целевое условие (минимизировать MAX)
       max[ i ].MAX.uDesBound = -1e7;

    // Задаем связи между показателями
       max[ i ].Y1 :=
       max[ i ].MAX.link - max[ i ].X1.link ;
```

```
    max[ i ].Y2 :=
    max[ i ].MAX.link - max[ i ].X2.link ;
        };
// Инициализация закончена

        solve(); // Решаем задачу

        print("RESULTS :\n");
        // Распечатываем результаты
        for( i = 0 ; i < SIZE ; i++ )
        {
            print(     "  Max of    "
            formatDouble("%8g and",
            max[ i ].x1 )
            formatDouble("%8g   is ",
            max[ i ].x2 )
            formatDouble("%8g\n",
            max[ i ].MAX.value )
        );
        };
```

Раздел 9.3. ЯЗЫК L. СПРАВОЧНИК ПО СТАНДАРТНОЙ БИБЛИОТЕКЕ

В данном параграфе описаны стандартные функции, доступные при интерпретации программы. Кроме того, здесь описаны предопределенные переменные и константы.

Стандартные математические функции

Все стандартные математические функции работают с аргументами типа `double`. Углы в тригонометрических функциях задаются в радианах. Все функции в точности соответствуют таким же функциям языка `C`.

`double acos (double x)` – арккосинус x,
`double asin (double x)` – арксинус x,
`double atan (double x)` – арктангенс x,
`double ceil (double x)` – наименьшее целое в виде `double`, которое $\geq x$,

`double cos (double x)` – косинус x,
`double cosh (double x)` – гиперболический косинус x,
`double exp (double x)` – экспоненциальная функция e^x,
`double fabs (double x)` – абсолютное значение $|x|$,
`double floor (double x)` – наибольшее целое в виде `double`, которое $\le x$,
`double log (double x)` – натуральный логарифм $\ln(x)$, $x > 0$,
`double log10 (double x)` – десятичный логарифм $\lg(x)$, $x > 0$,
`double sin (double x)` – синус x,
`double sinh (double x)` – гиперболический синус x,
`double sqrt (double x)` – квадратный корень из x, $x \ge 0$,
`double tan (double x)` – тангенс x,
`double tanh (double x)` – гиперболический тангенс x.

Математические константы

Приведенные ниже предопределенные константы имеют тип `const double`. Все константы определены с точностью до 21 знака.

```
M_E        =  2.71828182845904523536     - e
M_LOG2E    =  1.44269504088896340736      -
                                  log2( e )
M_LOG10E   =  0.43429448190325182765      -
                                 log10( e )
M_LN2      =  0.69314718055994530942      -
                                    ln( 2 )
```

```
M_LN10      =  2.30258509299404568402      -
                                      ln( 10 )
M_PI        =  3.14159265358979323846    - PI
M_PI_2      =  1.57079632679489661923  - PI/2
M_PI_4      =  0.78539816339744830961 - PI/4
M_1_PI      =  0.31830988618379067153 - 1/PI
M_2_PI      =  0.63661977236758134308 - 2/PI

M_1_SQRTPI  =  0.56418958354775628695      -
                                1 / sqrt( PI )
M_2_SQRTPI  =  1.12837916709551257390      -
                                2 / sqrt( PI )
M_SQRT2     =  1.41421356237309504880      -
                                   sqrt( 2 )
M_SQRT_2    =  0.70710678118654752440      -
                                sqrt( 2 ) / 2
```

Переменные – параметры решающего модуля

Путем изменения указанных ниже переменных пользователь может настраивать параметры решающего модуля программы `Lc.exe`. При вызове функции `solve()` проверяется корректность установки параметров и находится решение задач (7.1.1) и (7.1.2).

Тип	*Имя*	*Описание*
`int`	`FixLevNum`	Количество образованных групп

Допуски на:

`double`	`epsZero`	ноль
`double`	`epsBound`	границу
`double`	`epsIncon`	несовместность
`double`	`epsDeriv`	производную
`double`	`plusInfinity`	машинная бесконечность
`double`	`sheetZero`	0-диапазон электронной таблицы

```
double   MaxTime        максимально
                        допустимое
                        время решения
int      isRelBasis     не 0, если используется
                        метрика
int      state          статус решения задачи
```

Переменная `state` указывает статус полученого решения задачи:

```
const int   ST_CONSIST   СОВМЕСТНА
const int   ST_INCONSIST НЕСОВМЕСТНА
const int   ST_GREATMAX
                         НЕОГРАНИЧЕННОЕ
                         РЕШЕНИЕ
const int   ST_INCONSISTFIX
                         ПОТЕРЯ ТОЧНОСТИ
const int   ST_CYCLING
                         ПОТЕРЯ ТОЧНОСТИ:
                         ЗАЦИКЛИВАНИЕ
const int   ST_TIMEHALT  ИСЧЕРПАН ЛИМИТ
                         ВРЕМЕНИ
```

Функции для работы с именами параметров

При создании параметра ему присваивается имя, соответствующее имени переменной. Например, в следующем фрагменте кода:

```
real a ;
real b[ 2 ];
```

показателям присваиваются имена `a, b[1] и b[2].`

Описанные ниже функции предназначены для работы с именами показателей.

```
string  GetName( real ) -
                возвращает имя показателя
string SetName( string newName, real )
                - устанавливает новое имя
          показателя, которое и возвращает.
```

Функции для работы с задачами

Функции, описанные ниже, позволяют решать задачу, сохранять ее в стандартном формате, получать и устанавливать имя задачи, а также получать строку, описывающую состояние решения задачи.

```
int     solve()
```

- Запуск решающего модуля программы. При решении выдается диагностика, полнота которой регулируется опциями. Возвращает 0 в случае отсутствия ошибок, не 0 – в случае каких-либо ошибок (нехватка памяти, etc).

```
int     SaveProblem( string filename )
```

- Сохраняет текущую задачу в файл с именем filename. Возвращает 0 в случае успешного завершения, не 0 – в случае ошибок (нет места на диске, etc).

```
string  GetProblemName()
```
– возвращает имя задачи

```
int     SetProblemName( string newName )
```
– устанавливает новое имя задачи, всегда возвращает 0.

```
string  GetProblemState()
```
– возвращает строку, описывающую статус решения задачи:

Русская версия

```
"СОВМЕСТНА"
"НЕСОВМЕСТНА"
"НЕОГРАНИЧЕННОЕ РЕШЕНИЕ"
"ПОТЕРЯ ТОЧНОСТИ"
"ПОТЕРЯ ТОЧНОСТИ: ЗАЦИКЛИВАНИЕ"
"ИСЧЕРПАН ЛИМИТ ВРЕМЕНИ"
"НЕ РЕШАЛАСЬ"
```

Латинская версия

```
"FEASIBLE"
"INFEASIBLE"
"UNBOUNDED SOLUTION"
"LOST OF ACCURACY"
"LOST OF ACCURACY: CYCLING"
"TIME LIMIT EXCESS"
"WAS NOT SOLVED"
```

Функции общего назначения

`int strlen( string )` - вычисляет длину строки

`int print ( string )` - печатает строку

`int redirect ( string filename )`
- перенаправляет весь вывод в файл filename. Если `filename == "-"`, то вывод перенаправляется в `stdout` (т.е. обычно на экран).

`double atof( string )`
- вводит число типа `double` из строки

`int atoi( string )`
- вводит число типа `int` из строки

`string _ftoa( double )`
- распечатывает `double` в строку по формату `%g`

`string _itoa( int )`
- распечатывает `int` в строку по формату `%d`

Следующие функции распечатывают соответственно `double`, `int` или `строку` в соответствии с форматной строкой `format` :

```
string formatDouble
            ( string format, double value );
```

```
string formatInt
                ( string format, int  value );
string  formatString
   ( string format, string formattedString );
```

Строка `format` должна строиться в соответствии с правилами, принятыми в языке `C` для форматных строк.