Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Е.С. Рябчикова
С.М. Андреев
М.Ю. Рябчиков

МЕТОДЫ И ТЕОРИИ ОПТИМИЗАЦИИ

Утверждено Редакционно-издательским советом университета
в качестве учебного пособия

Магнитогорск
2016

УДК 681.5.017

Рецензенты:

кандидат технических наук,
начальник отдела АСУ,
ЗАО «КонсОМ СКС»
А.Н. Панов

доктор технических наук, профессор,
заведующий кафедрой вычислительной техники и программирования,
ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.И. Носова»
О.С. Логунова

Рябчикова Е.С., Андреев С.М., Рябчиков М.Ю.
Методы и теория оптимизации [Электронный ресурс] : учебное пособие / Елена Сергеевна Рябчикова, Сергей Михайлович Андреев, Михаил Юрьевич Рябчиков ; ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.И. Носова». – Электрон. текстовые дан. (1,59 Мб). – Магнитогорск : ФГБОУ ВО «МГТУ им. Г.И. Носова», 2016. – 1 электрон. опт. диск (CD-R). – Систем. требования : IBM PC, любой, более 1 GHz ; 512 Мб RAM ; 10 Мб HDD ; MS Windows XP и выше ; Adobe Reader 8.0 и выше ; CD/DVD-ROM дисковод ; мышь. – Загл. с титул. экрана.

Учебное пособие по дисциплинам «Моделирование систем управления» и «Методы и теория оптимизации» для студентов очной и заочной формам обучения, обучающихся по направлениям подготовки 27.03.04 – Управление в технических системах, профиль – Системы и средства автоматизации технологических процессов и 15.04.06 – Мехатроника и робототехника, профиль – Мехатронные системы в автоматизированном производстве. В учебном пособии рассматривается постановка задач одномерной и многомерной оптимизации целевой функции, задач линейного программирования и способы их решения. Приведены практические аспекты решения оптимизационных задач, в том числе определение оптимальных коэффициентов стабилизирующего управления с использованием методов оптимизации.

УДК 681.5.017

Содержание

1. ПОСТАНОВКА И КЛАССИФИКАЦИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Оптимизация – это выбор наилучшего решения. В самом общем случае, решить оптимизационную задачу это значит найти наилучшее решение среди возможных вариантов решения.

Если оптимизация связана с расчетом оптимальных значений параметров при заданной структуре объекта, то она называется параметрической. Задача выбора оптимальной структуры является структурной оптимизацией.

Теория оптимизации представляет собой совокупность фундаментальных математических результатов и численных методов, позволяющих избежать полного перебора всех решений.

Методы оптимизации – это методы построения алгоритмов нахождения оптимального (минимального или максимального) значения некоторой функции.

Для того чтобы использовать результаты и вычислительные процедуры теории оптимизации на практике, необходимо, прежде всего, сформулировать рассматриваемую задачу на математическом языке, т.е. построить математическую модель объекта оптимизации.

Построение математических моделей оптимизации можно условно разбить на следующие основные этапы.

1.Определение границ объекта оптимизации. Необходимость этого этапа диктуется невозможностью учета и исчерпывающего описания всех сторон большинства реальных систем. Выделив главные переменные, параметры и ограничения, следует приближенно представить систему как некоторую изолированную часть реального мира и упростить ее внутреннюю структуру. Может оказаться, что первоначальные границы объекта оптимизации выбраны неудачно. Тогда в одних случаях границы системы следует расширить, а в других – сузить.

2.Выбор управляемых переменных. На этом этапе математического моделирования необходимо провести различие между теми величинами, значения которых можно варьировать и выбирать с целью достижения наилучшего результата (***управляемыми переменными***), и величинами, которые фиксированы или определяются внешними факторами. Определение тех значений управляемых переменных, которым соответствует наилучшая (оптимальная) ситуация, и представляет собой задачу оптимизации.

3.Определение ограничений на управляемые переменные. В реальных условиях на выбор значений управляемых переменных обычно наложены ограничения, которые связаны с ограниченностью имеющихся ресурсов, мощностей и т.д. При построении математической модели эти ограничения обычно записывают в виде равенств и неравенств. Совокупность всех ограничений на управляемые переменные определяет ***допустимое множество*** задачи оптимизации.

4.Выбор числового критерия оптимизации. Числовой критерий, минимальному или максимальному значению которого соответствует наилучший вариант поведения объекта оптимизации, называется ***целевой функцией***. Она полностью определяется выбранными значениями управляемых переменных.

5.Формулировка математической задачи оптимизации. Объединяя результаты предыдущих этапов построения математической модели, ее записывают в виде математической задачи оптимизации, включающей построенную целевую функцию и найденные ограничения на управляемые переменные. При записи математических задач оптимизации в общем виде обычно используется следующая символика:

$$f(x) \to \min(\max), x \in G,$$

где $f(x)$ - целевая функция, а G - допустимое множество, заданное ограничениями на управляемые переменные.

В зависимости от вида целевой функции и соотношения ограничений выделяют различные задачи оптимизации, классификация которых приведена на рис. 1.1.

Существует несколько признаков классификации. Основные критерии следующие.

1. По типу параметров задачи оптимизации. Различают непрерывные задачи оптимизации (continues optimization) и дискретные (discrete) и целочисленные (integer optimization).

2. По критерию размерности допустимого множества параметров G. Задачи оптимизации по этому критерию делятся на задачи одномерной оптимизации и задачи многомерной оптимизации.

3. Критерий наличия или отсутствия ограничений на допустимое множество G. Различают задачи условной (constrained) и безусловной (unconstrained) оптимизации. Этот признак классификации имеет место, как для одномерных, так и для многомерных задач оптимизации.

4. По характеру ограничений различают детерминированную оптимизацию и стохастическую. Если множество допустимых значений включает случайные компоненты, то имеет место стохастическое программирование. При этом стохастическая оптимизация может относиться и к дискретной задаче.

5. По виду целевой функции и виду ограничений различают линейное и нелинейное программирование. Задача линейного программирования содержит линейную целевую функцию, ограничения в задаче также линейны. При нарушении линейности целевой функции или ограничений имеет место нелинейная задача оптимизации.

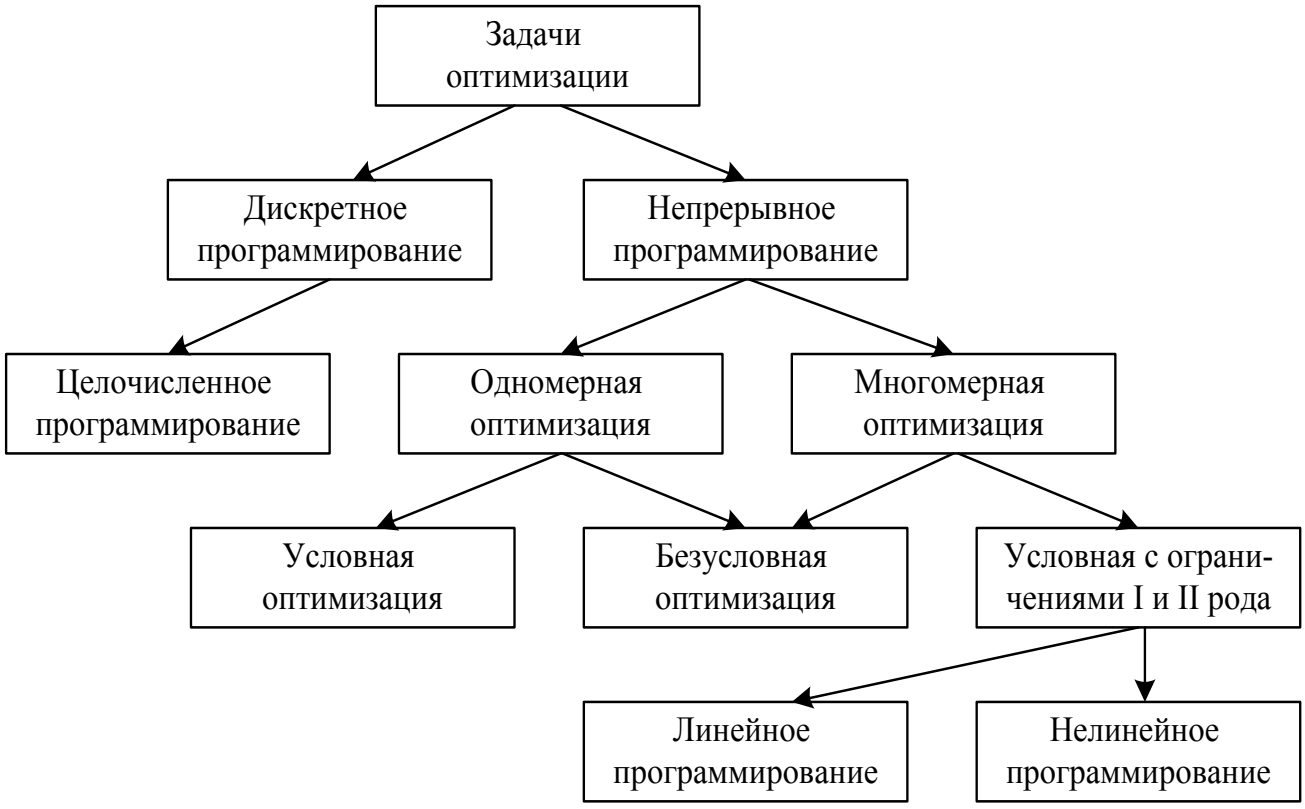


Рис. 1.1. Классификация задач оптимизации

В качестве ***примеров постановки задачи оптимизации*** рассмотрим:

Пример №1. Необходимо найти оптимальный температурный режим в трубчатом реакторе, который заполнен катализатором. На вход поступает сырье, на выходе – продукты реакции. В трубчатом реакторе протекает химическая реакция, при этом в каждом сечении реактора концентрация компонентов и температура отличаются друг от друга.

Если считать, что движение реагента не меняется по длине то уравнение состояния этого реактора можно записать:

$$\frac{dC_i}{dl} = f_i(C_i, T_l, l), \tag{1.1}$$

где C_i – концентрация i-того компонента реагирующей смеси; T_l – температура в сечении реактора; l – продольная координата трубчатого реактора.

Граничные условия:

$$C_i = C_{i0} \ \ при\, l = 0. \tag{1.2}$$

Ограничение на фазовые координаты:

$$C_i \geq 0; \tag{1.3}$$

концентрация компонентов не отрицательна (i=1,2,…,r)

$$T_0 \leq T \leq T_*. \tag{1.4}$$

Для процесса, описываемого системой (1.1)-(1.4), требуется найти температурный профиль по длине реактора T(l), при котором концентрация целевого продукта на выходе реактора будет максимальна $C_k = C_k(l) \to \max$.

Пример №2. Задача управления температурным режимом печи.

В печь подается топливо расхода G_T и состава C_T, а также воздух с расходом G_B. В печи происходит реакция горения топлива, что обеспечивает достижение некой температуры печи, которая будет зависеть от соотношения расхода воздуха и топлива:

$$\alpha = \frac{G_B}{G_T}, \tag{1.5}$$

где T=T(α) – температура (функция от α).

При этом α и T ограничены. Нижние и верхние предельно допустимые значения параметров $\alpha_0 \leq \alpha \leq \alpha_*;\ T_0 \leq T \leq T_*;$ α^0 – оптимальное значение соотношения расходов, при котором температура в печи достигает максимального значение.

Требуется найти такое соотношение расходов $\frac{G_B}{G_T}$, чтобы температура печи была максимальна.

2. МЕТОДЫ РЕШЕНИЯ ОДНОМЕРНЫХ ЗАДАЧ ОПТИМИЗАЦИИ

Одномерная задача оптимизации формулируется следующим образом: найти наименьшее или наибольшее значение целевой функции, заданной на множестве G, и определить значение проектного параметра xϵG, при котором целевая функция принимает экстремальное значение.

Существование решения поставленной задачи вытекает из теоремы Вейерштрасса:

«Всякая функция f(x), непрерывная на отрезке [a,b], принимает на этом отрезке наименьшее и наибольшее значение, т.е. на отрезке [a,b] существуют такие точки x1 и x2, что для любого значения xϵ[a,b] имеет место неравенство:

$$f(x1)<=f(x)<=f(x2)».$$

В случае периодической функции, если на данном отрезке [a,b] укладывается несколько периодов, то может быть несколько таких точек.

Теорема Вейерштрасса – это теорема существования и она говорит, что решение задачи оптимизации для непрерывной функции обязательно существует.

2.1. Решение задач одномерной оптимизации методом производной

Если функция задана аналитически и имеется возможность найти выражение для производной этой функции, то задача оптимизации решается методом производной, известным из математики.

Функция f(x) может достигать своего наибольшего или наименьшего значения либо в граничных точках отрезка *[a,b]*, либо в точках минимума или максимума.

Точки минимума или максимума называются критическими точками, т.к. в них производная равна нулю.

Поэтому для решения задачи оптимизации методом производной нужно вычислить значение функции во всех критических точках и в граничных точках отрезка, а затем сравнить полученные значения и выбрать из них наибольшее и наименьшее.

Пример: найти наибольшее и наименьшее значения функции $f(x) = x^3/3 - x^2$, заданной на отрезке [1,3].

$f'(x) = x^2-2x = 0$

$x(x-2) = 0$

$x_1=0 \notin [1,3] => x_1=0$ не рассматриваем

$x_2=2 \in [1,3]$

$a=1 \quad f(a) = -2/3$

$x_2=2 \quad f(x_2) = -4/3$

$b=3 \quad f(b) = 0$

Отсюда:

$f_{min} = f(2) = -4/3; f_{max} = f(3) = 0.$

2.2. Метод перебора значений целевой функции с равномерным распределением точек по отрезку (метод полного перебора, метод сканирования)

Пусть требуется найти наименьшее значение целевой функции $U = f(x)^*$, заданной на отрезке $[a,b]$. Для решения поставленной задачи возьмем некоторое целое число n, вычислим шаг $h = (b-a)/n$ и определим значения целевой функции $f(x)$ в точках $x_k = a + k \cdot h$ ($k = 0,1,2,\ldots n$): $U_k = f(x_k)$. После этого среди полученных чисел U_k найдем наименьшее число:

$$m_n = \min(U_0, U_1, \ldots, U_n). \qquad (2.1)$$

Число m_n можно приближенно принять за наименьшее значение функции $f(x)$ на отрезке $[a,b]$. При этом очевидно, что

$$m_n \geq m = \min f(x),$$

$$x \in [a,b].$$

Однако благодаря непрерывности функции $f(x)$ будем иметь

$$\lim_{n\to\infty} m_n = m = \min f(x),\ x \in [a,b],$$

т.е. с увеличением числа точек n ошибка, которую мы будем допускать, принимая m_n за m, стремится к нулю.

Какое же n нужно взять, чтобы погрешность определения наименьшего значения целевой функции $\delta_n = |m_n - m|$ не превышала заданной точности ε, т.е. чтобы $\delta_n \leq \varepsilon$?

Если нам известно только то, что функция $f(x)$ непрерывна на отрезке $[a,b]$, то ответить на поставленный вопрос нельзя. Какое бы n мы не взяли, и как бы не выбирали n точек на отрезке $[a,b]$, всегда можно указать такую непрерывную функцию, что для нее m_n будет отличаться от m больше, чем на ε. Справедливость такого утверждения иллюстрируется рис. 2.1.

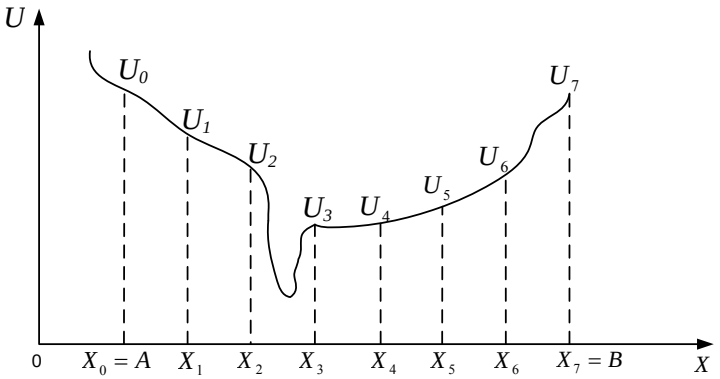


Рис. 2.1. График многоэкстремальной целевой функции

Допустим, что мы взяли $n=7$. Определяя значения функции $U_k = f(x_k)$ в точках x_k ($k=0,1,2,\dots 7$), получим, что

$$m_7 = \min(U_0, U_1, \dots, U_7) = U_3 = f(x_3).$$

Величину $m_7 = U_3$, согласно описанному методу, следует приближенно принять за наименьшее значение целевой функции на рассматриваемом отрезке. Однако, если у нас перед глазами нет графика функции, а известны только ∞ чисел U_k, то по ним невозможно установить, что функция $f(x)$ имеет между точками x_2 и x_3 узкий "язык", который опускается гораздо ниже значения $m_7 = U_3$. Из-за небольшого числа точек мы его пропустим. Если взять большее n, то данный "язык" обнаружится, но при этом может оказаться незамеченным какой-нибудь другой еще более узкий "язык".

При отсутствии дополнительной информации о свойствах целевой функции $f(x)$, о том, насколько "резкими" могут быть ее изменения, сомнения останутся всегда, какое бы большое число точек мы не взяли.

Дать строго обоснованную оценку числа точек n, необходимого для решения задачи с точностью ε, можно только сужая класс рассматриваемых функций.

Пусть априори известно, что целевая функция $U = f(x)$ имеет на отрезке $[a,b]$ только один минимум, т. е. является унимодальной (рис. 2.2). Это свойство целевой функции позволяет избежать пропуска фактического минимума при решении задач оптимизации, кроме того, объем вычислений также может быть сокращен.

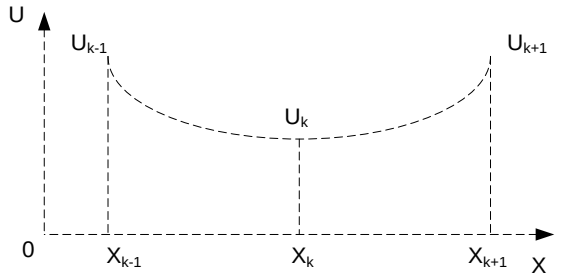


Рис. 2.2. График унимодальной целевой функции

Для решения задачи оптимизации в этом случае может быть применен следующий прием. Возьмем некоторый шаг h и будем последовательно вычислять значения функции $f(x)$ в точках $x_0 = a$; $x_1 = a+h$; $x_2 = a+2h$…, сравнивая при этом получаемые числа U_0, U_1, U_2,… Сначала они будут убывать: $U_0 > U_1 > U_2$ …, однако, в дальнейшем найдется такая точка $x_k = a+kh$, что для значения функции в этой точке $U_k = f(x_k)$ будут справедливы следующие неравенства: $U_{k-1} \ge U_k$, $U_{k+1} \ge U_k$. Это означает, что минимум функции достигается на отрезке $[x_{k-1}, x_{k+1}]$ и его приближенно можно принять равным $U_k = f(x_k)$. Для повышения точности решения задачи можно уменьшить шаг h и повторить описанную процедуру уже для отрезка $[x_{k-1}, x_{k+1}]$.

В задачах управления часто требуется найти с заданной точностью ε не само значение минимума целевой функции на отрезке $[a,b]$, а абсциссу точки минимума $x^* = \arg m$, т. к. x обычно играет роль управляющего воздействия, которое, конечно, должно быть оптимальным. В этом случае задача выбора числа n, необходимого для отыскания с заданной точностью ε абсциссы точки минимума x^*, решается так.

Следует заметить, что интервал значений x, в котором заключен оптимум целевой функции, называется интервалом неопределенности. В начале решения задачи этот интервал имеет длину $(b-a)$. При решении задачи методом перебора первоначальный интервал неопределенности сужается до двух шагов, т. е. окончательный интервал неопределенности будет равен $2h$ (рис. 2.3).

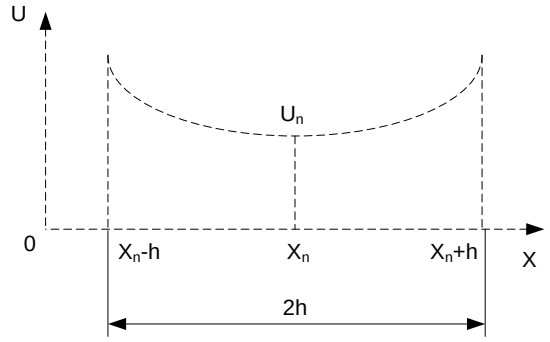


Рис. 2.3. Окончательный интервал неопределенности

Последнее означает, что $x^* \in [x_n - h, x_n + h]$, где $x_n = \arg m_n$ или, что то же самое,

$$x_n - h \le x^* \le x_n + h .$$

Поэтому

$$-h \le x^* - x_n \le h$$

или

$$\left| x^* - x_n \right| \le h .$$

Но по условию задачи требуется, чтобы $\left| x^* - x_n \right| \le \varepsilon$, следовательно, шаг h должен удовлетворять следующему условию: $h \le \varepsilon$. Или

$$(b-a)/n \le \varepsilon ;$$

$$n \ge (b-a)/\varepsilon . \qquad (2.2)$$

Данное соотношение и определяет необходимое значение числа n.

Заметим, что задача поиска может быть сформулирована и так: требуется найти абсциссу точки минимума x^*, причем длина окончательного интервала неопределенности не должна превышать заданного, достаточно малого положительного числа ε. В этом случае, очевидно, что n следует выбирать по соотношению:

$$n \ge 2(b-a)/\varepsilon .$$

Блок-схема алгоритма реализации метода полного перебора значений целевой функции представлена на рис. 2.4. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

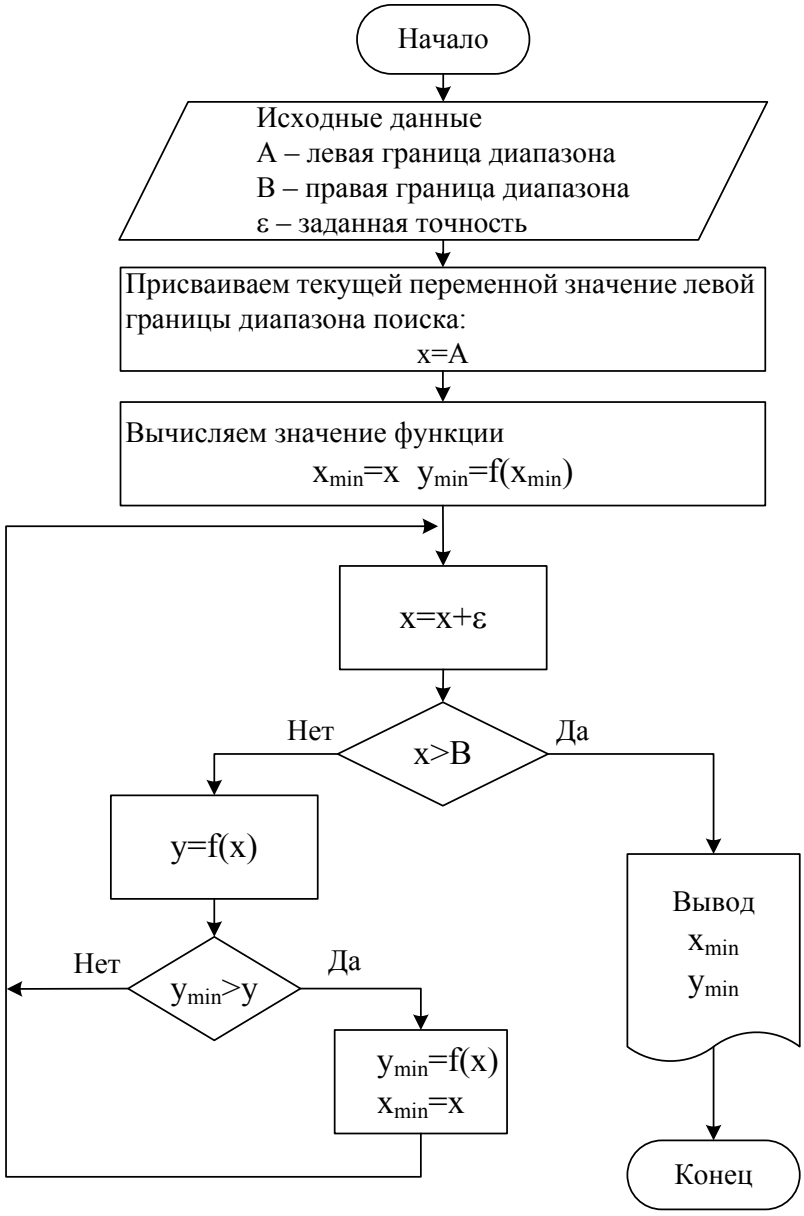


Рис. 2.4. Блок-схема алгоритма реализации метода полного перебора значений целевой функции

2.3. Одномерная оптимизация целевой функции методом золотого сечения

Как известно, ***золотым сечением отрезка*** называют деление отрезка на две части так, что отношение длины всего отрезка к длине большей части равно отношению длины большей части к меньшей.

Золотое сечение отрезка $[a,b]$ производят две симметрично расположенные точки:

$$x_1 = a + (1-\tau)\cdot(b-a); \tag{2.3}$$

$$x_2 = a + \tau\cdot(b-a), \tag{2.4}$$

где $\tau = (\sqrt{5}-1)/2 = 0{,}6180339$. Точка x_1, в свою очередь, производит золотое сечение отрезка $[a,x_2]$, а точка x_2 - золотое сечение отрезка $[x_1,b]$.

Алгоритм поиска минимального значения функции $f(x)$ на отрезке $[a,b]$ заключается в следующем.

Начальный отрезок $[a,b]$ делим точками x_1 и x_2 по "правилу золотого сечения" (2.3),(2.4) и в точках x_1 и x_2 вычисляем значения функций $f(x_1)$ и $f(x_2)$.

Если $f(x_1) < f(x_2)$, то для дальнейшего деления берем отрезок $[a,x_2]$. Если же $f(x_1) > f(x_2)$, то берем отрезок $[x_1,b]$. Покажем это.

Пусть $f(x_1) < f(x_2)$, тогда для дальнейшего рассмотрения берем отрезок $[a,x_2]$, т.е. считаем x^* расположенным левее точки x_2. Действительно, предположим, что оптимум $x^* \geq x_2$, т.е. расположен наоборот правее точки x_2, но тогда x_1 и x_2 будут находиться с одной стороны от оптимума: $x_1 < x_2 < x^*$. По определению унимодальности это означало бы, что $f(x_1) > f(x_2)$, но это противоречит условию $f(x_1) < f(x_2)$.

Аналогично, если $f(x_1) > f(x_2)$, то для дальнейшего деления берем отрезок $[x_1,b]$, т. к. оптимум x^* не может лежать левее x_1, иначе нарушается предположение об унимодальности функции $f(x)$. Действительно, если предположим, что $x^* < x_1$, то тогда будет $x^* < x_1 < x_2$. По определению унимодальности это означало бы, что $f(x_1) < f(x_2)$, но это не так, т. к. по условию $f(x_1) > f(x_2)$.

В первом случае одной из двух точек, делящих в золотом отношении отрезок $[a,x_2]$, будет точка x_1, а во втором для отрезка $[x_1,b]$ - точка x_2. Это следует из свойств золотого сечения. Таким образом, из двух значений функции, необходимых на новом шаге расчета, вычисляется всего одно: другое было получено на предыдущем шаге вычислений.

Итак, на каждом шаге поиска, начиная со второго, требуется лишь одно вычисление функции, и интервал неопределенности уменьшается в (0,618)-1 раз.

Итерации продолжаются до тех пор, пока интервал неопределенности не станет меньше некоторого заданного достаточно малого положительного числа ε.

Алгоритм поиска минимума функции $f(x)$ на отрезке $[a,b]$ поясняется на рис. 2.5. Блок-схема алгоритма реализации метода «золотого сечения» представлена на рис. 2.6. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

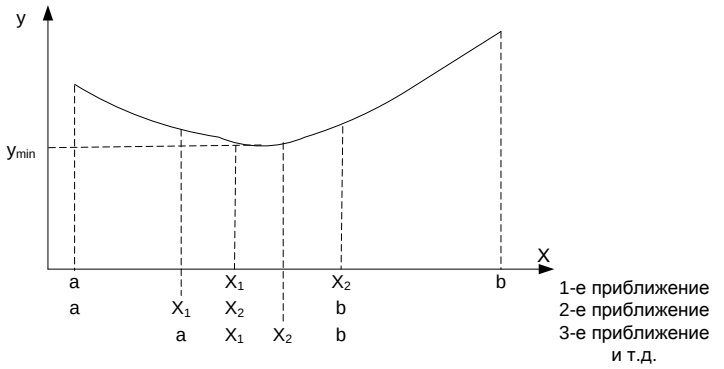


Рис. 2.5. Поиск минимума методом золотого сечения

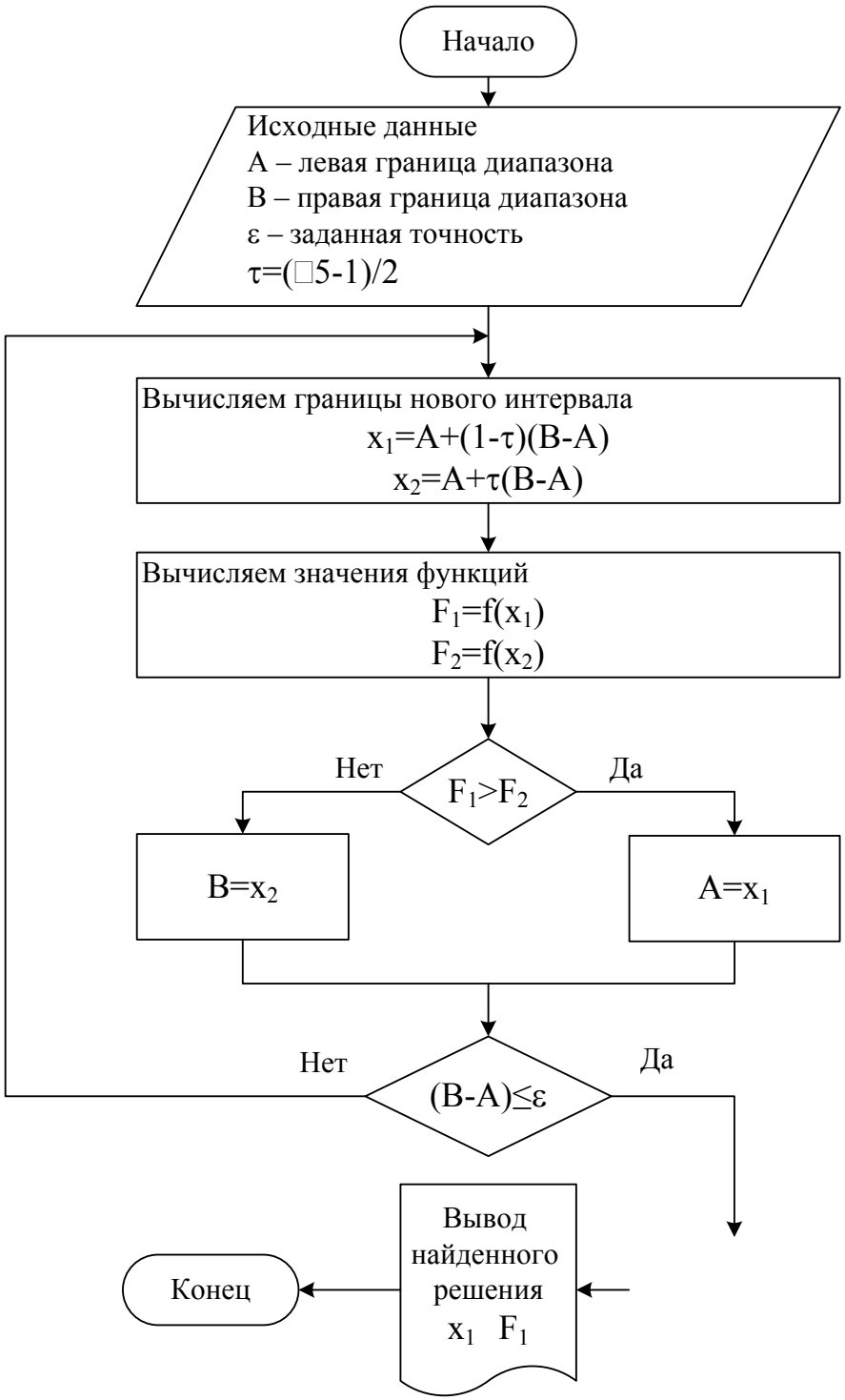


Рис. 2.6. Блок-схема алгоритма реализации метода «золотого сечения»

2.4. Одномерная оптимизация целевой функции методом квадратичной интерполяции

В этом методе целевая функция сначала заменяется квадратичным полиномом $f(x) = ax^2 + bx + c$. После этого положение минимума целевой функции определяется по положению минимума полинома, что значительно проще.

Рассмотрим аспекты практической реализации этого метода.

Пусть задана унимодальная функция $f(x)$, точка x_0 – начальная аппроксимация положения минимума и h – шаг, величина которого имеет тот же порядок, что и расстояние от точки x_0 до истинного минимума. Вычислительные процедуры в этом случае содержат следующие этапы:

1. Вычисляем значение функции в исходной точке $x_1 = x_0$, $F_1 = f(x_1)$. и $f(x_0 + h)$.

2. Определяем две другие точки для аппроксимирующего полинома: $x_2 = x_0 + h$; $F_2 = f(x_2)$;

а) если $f(x_0) < f(x_0 + h)$, то в качестве третьей аппроксимирующей точки выбирается точка $x_3 = x_0 - h$ и вычисляется $F_3 = f(x_3) = f(x_0 - h)$ (см. рис. 2.7, а);

б) если $f(x_0) > f(x_0 + h)$, то выбирается точка $x_3 = x_0 + 2h$ и вычисляется $F_3 = f(x_3) = f(x_0 + 2h)$ (см. рис. 8, б).

Такой алгоритм выбора точек обусловлен тем, что мы ищем минимум целевой функции. В случае а), сделав шаг в направлении увеличения x_0, мы обнаруживаем, что целевая функция в этом направлении возрастает, следовательно, дальнейшее продвижение в этом направлении не имеет смысла, поэтому в качестве третьей аппроксимирующей точки берется точка $(x_0 - h)$.

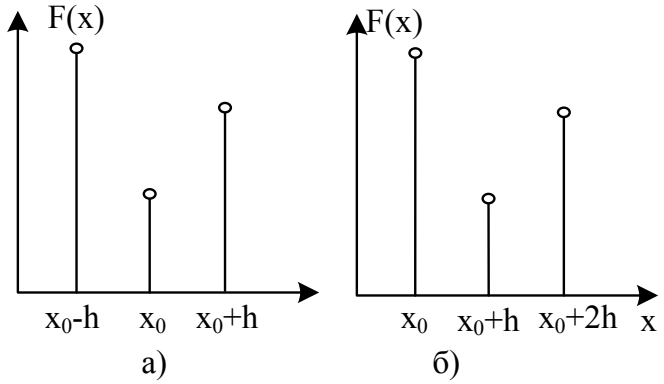


Рис. 2.7. Выбор точек для аппроксимации квадратичного полинома

Этим самым учитывается следующее обстоятельство: чем ближе три аппроксимирующие точки будут к точке минимума, тем лучше в этой области квадратичный полином будет описывать свойства целевой функции и тем точнее будет найдена точка минимума за одну итерацию.

3. Заменим целевую функцию в окрестности этих точек квадратичным полиномом $f(x) = ax^2 + bx + c$. Коэффициенты этого полинома можно определить из системы трех линейных уравнений (10), которые надо разрешить относительно коэффициентов a, b, c:

$$\begin{cases} ax_1{}^2 + bx_1 + c = f(x_1) \\ ax_2{}^2 + bx_2 + c = f(x_2) \\ ax_3{}^2 + bx_3 + c = f(x_3) \end{cases} \tag{2.5}$$

При этом непосредственно определять численные значения коэффициентов a, b, c нет необходимости. В задаче нужно только знать положение минимума квадратичного полинома. Известно, что минимум квадратичного полинома расположен в точке;

$$x^* = -\frac{b}{2a} \quad \text{, при } a > 0\,. \tag{2.6}$$

Решаем систему (10) с помощью определителей по методу Крамера:

$$\Delta = \begin{vmatrix} x_1{}^2 & x_1 & 1 \\ x_2{}^2 & x_2 & 1 \\ x_3{}^2 & x_3 & 1 \end{vmatrix} = (x_2 - x_1) + (x_3 - x_1) + (x_3 - x_2);$$

$$\Delta_1 = \begin{vmatrix} f(x_1) & x_1 & 1 \\ f(x_2) & x_2 & 1 \\ f(x_3) & x_3 & 1 \end{vmatrix} = f(x_1)(x_2 - x_3) + f(x_2)(x_3 - x_1) + f(x_3)(x_1 - x_2);$$

$$\Delta_2 = \begin{vmatrix} x_1{}^2 & f(x_1) & 1 \\ x_2{}^2 & f(x_2) & 1 \\ x_3{}^2 & f(x_3) & 1 \end{vmatrix} = \left(x_3{}^2 - x_2{}^2\right) f(x_1) + \left(x_1{}^2 - x_3{}^2\right) f(x_2) + \left(x_2{}^2 - x_1{}^2\right) f(x_3);$$

$$a = \frac{\Delta_1}{\Delta}; b = \frac{\Delta_2}{\Delta}.$$

4. Определим приближенное значение положения точки минимума целевой функции:

$$x^* = -\frac{b}{2a} = -\frac{1}{2}\frac{\Delta_2}{\Delta_1};$$

$$x^* = -\frac{1}{2}\frac{\left(x_3{}^2 - x_2{}^2\right) f(x_1) + \left(x_1{}^2 - x_3{}^2\right) f(x_2) + \left(x_2{}^2 - x_1{}^2\right) f(x_3)}{f(x_1)(x_2 - x_3) + f(x_2)(x_3 - x_1) + f(x_3)(x_1 - x_2)}. \tag{2.7}$$

5. Проведем упорядочение значений целевой функции. Упорядочить значения целевой функции – это значит присвоить точке с большим значением больший индекс (рис. 2.8).

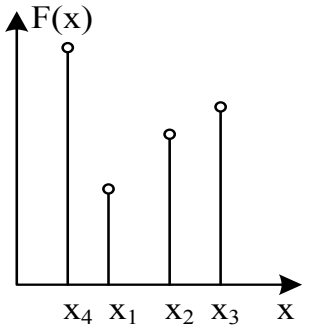


Рис. 2.8. Упорядочение значений целевой функции

6. Проверим выполнение условия окончания поиска минимума целевой функции. Если после упорядочения значений целевой функции модуль разности двух самых минимальных значений целевой функции будет меньше заданной точности ε, то вычисления прекращаются:

$$|f(x_1) - f(x_2)| \le \varepsilon . \qquad (2.8)$$

Для непрерывной целевой функции выполняется соотношение $\lim_{\Delta x \to 0} \Delta f = 0$, поэтому данное условие окончания поиска часто заменяют на следующее:

$$|x_1 - x_2| \le \varepsilon , \qquad (2.9)$$

т.е. вычисления прекращают, когда модуль разности абсцисс двух точек с наименьшими значениями будет меньше или равен ε.

Смысл такого окончания поиска заключается в следующем: очевидно, что значения целевой функции во вновь вычисленной точке x^* не может быть больше предыдущего наименьшего значения, т.к. мы ищем минимум целевой функции. Поэтому после упорядочения значений целевой функции точка x^*, найденная на данной итерации, получит имя x_1, а предыдущая точка минимума будет иметь имя x_2. Поэтому, оценивая модуль $|x_1 - x_2|$, мы тем самым определим, насколько проведение нового этапа вычислений изменит координату точки минимума. Если $|x_1 - x_2|$ меньше некоторой достаточно малой положительной величины ε, то очевидно, что дальнейшее проведение вычислений не имеет смысла, т.е. координата точки минимума будет уточнена незначительно.

7. Если точность решения задачи на четвертом этапе не достигнута, то точка x_4 с наибольшим значением целевой функции отбрасывается, и вновь возвращаются на третий этап, т.е. по оставшимся трем точкам вновь вычисляют точку минимума x^*. Но если, оставив точку с наибольшим значением функции, мы определим конечные границы интервала, в котором лежит минимум, то следует действительно эту точку оставить и затем вернуться на шаг три (рис. 2.9).

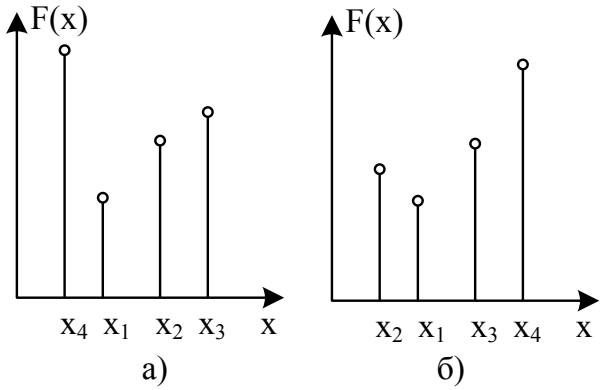


Рис. 2.9. Пояснение принципа выбора точки, которую необходимо отбросить:
а) отбрасываем точку x_4; б) отбрасываем точку x_3

8. После проведения некоторого количества вычислений абсциссы точек x_1, x_2, x_3, а также ординаты этих точек $f(x_1), f(x_2), f(x_3)$ будут достаточно близки друг к другу, поэтому относительная погрешность вычислений x^* по формуле (2.7) будет достаточно большой. Поэтому для вычислений x^* на второй и последующих интерполяциях используется формула (12) в преобразованном виде:

$$x^* = \frac{x_1 + x_2}{2} + \frac{1}{2}\frac{(x_3 - x_1)(x_2 - x_3)(f(x_1) - f(x_2))}{[f(x_1)(x_2 - x_3) + f(x_2)(x_3 - x_1) + f(x_3)(x_1 - x_2)]}. \qquad (2.10)$$

Этот метод в литературе называется методом Пауэлла.

Блок-схема алгоритма реализации метода квадратичной интерполяции представлена на рис. 2.10, 2.11. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

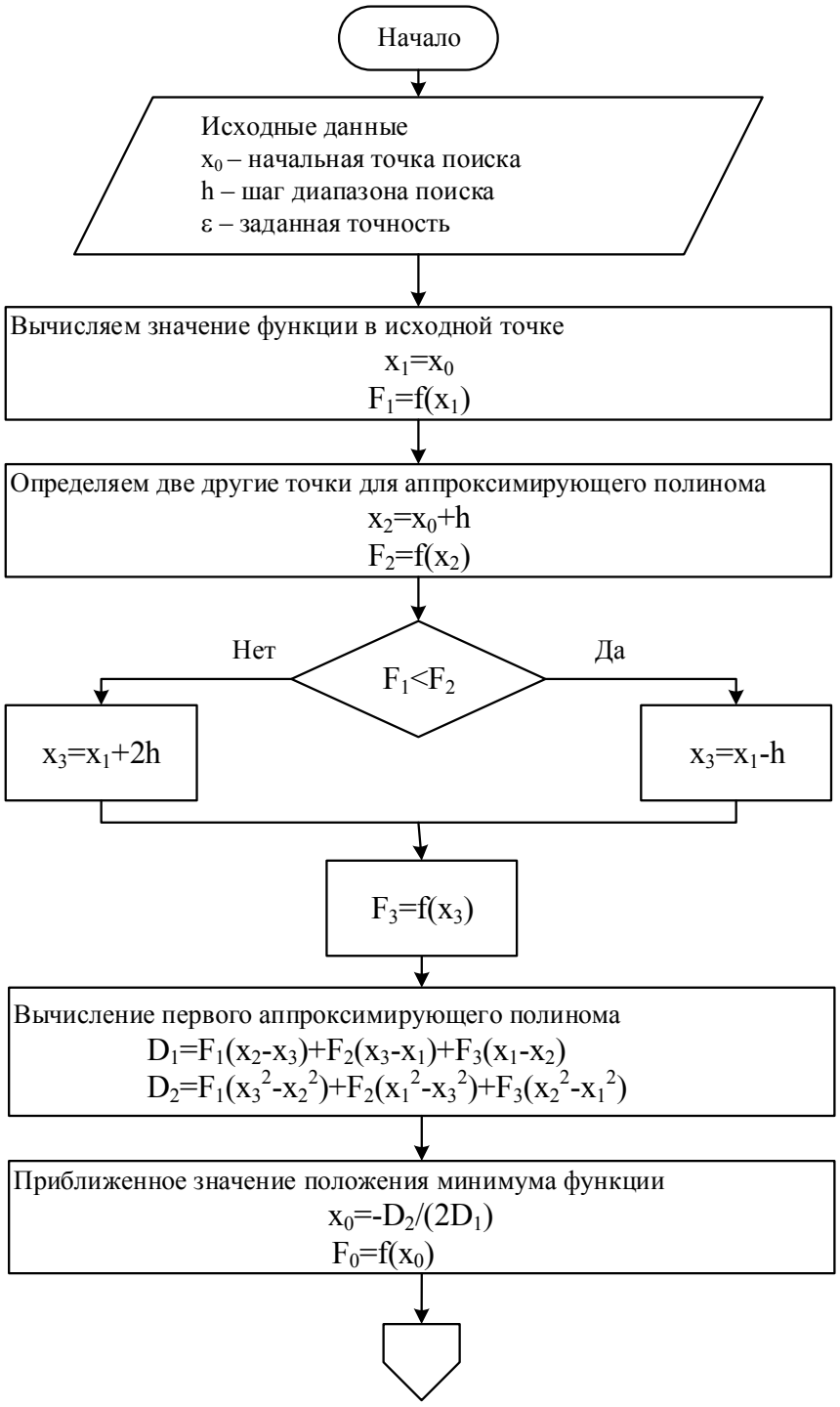


Рис. 2.10. Блок-схема алгоритма поиска минимума функции методом квадратичной интерполяции (начало)

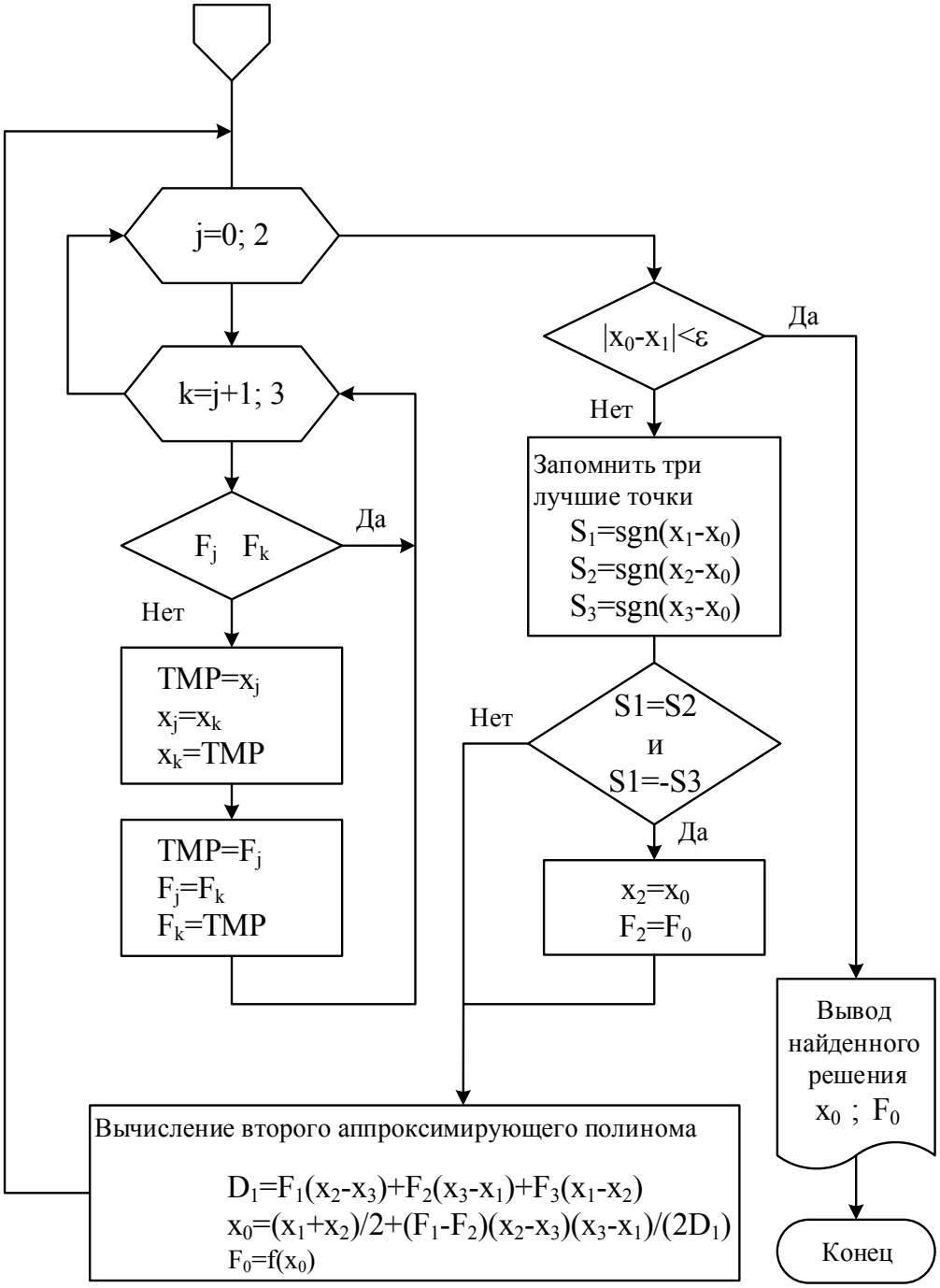


Рис. 2.11. Блок-схема алгоритма поиска минимума функции методом квадратичной интерполяции (окончание)

2.5. Контрольные вопросы

1. Суть метода сканирования (метода полного перебора значений целевой функции).
2. Что такое интервал неопределенности и каков его окончательный размер при решении задачи методом перебора?
3. Как выбирается число просматриваемых точек интервала неопределенности при решении задачи одномерной оптимизации методом перебора?
4. Какие функции называются унимодальными?
5. Что называется, золотым сечением отрезка?
6. Алгоритм поиска минимума методом золотого сечения.

7. Теоретическое обоснование метода золотого сечения.

8. На чем основан метод квадратичной интерполяции?

9. В чем заключается упорядочение значений целевой функции?

10. Каковы условия окончания поиска минимума целевой функции в методе золотого сечение и в методе квадратичной интерполяции?

2.6. Задачи для самостоятельного решения

Найти на отрезке [-10,10] абсциссу точки минимума x^* функции $U = x^2 + k_1 \cdot \exp(k_2 \cdot x)$ с абсолютной погрешностью, не превышающей 0,01.

Таблица 2.1

Варианты заданий

Номер варианта	k_1	k_2
1	1,0	-0,85
2	2,0	-0,65
3	3,0	-0,45
4	4,0	-0,25
5	5,0	-0,05
6	6,0	0,15
7	7,0	0,35
8	8,0	0,55
9	9,0	0,75
10	10,0	0,95

3. МЕТОДЫ РЕШЕНИЯ МНОГОМЕРНЫХ ЗАДАЧ ОПТИМИЗАЦИИ

В многомерных задачах оптимизации требуется найти наибольшее или наименьшее значение целевой функции *f(x)*, где $x=x_1,x_2,...x_n$, заданной на множестве *G* *n*-мерного пространства:

$$f(x) \to \min(\max), \quad x \in G \ \ (G - \text{область возможных решений}).$$

Если функция непрерывна, а область G является замкнутой ограниченной областью, то для этого случая справедлива теорема Вейерштрасса.

Решение задачи оптимизации возможно, если функция непрерывна в области G, которая является замкнутой и ограниченной.

Ограниченной называется область, если существует такое расстояние C, что расстояние до любой точки M от начала координат меньше или равно C:

$$\left|\overline{OM}\right| \le C .$$

Если функция задана в виде формулы и дифференцируема, то условия существования экстремума известны из математики.

На практике, при решении задач оптимизации, целевая функция не имеет формулы, а значения ее определяются или в результате эксперимента, или путем сложных математических расчетов. Поэтому для решения задач многомерной оптимизации применяют специальные методы.

3.1. Метод покоординатного спуска Гаусса-Зейделя

Пусть требуется найти наименьшее значение целевой функции $U = f(M) = f(x_1, x_2, \ldots, x_n)$. Здесь через M обозначена точка n-мерного пространства с координатами $x_1, x_2, \ldots, x_n$: $M = (x_1, x_2, \ldots, x_n)$.

Выберем какую-нибудь начальную точку $M_0 = (x_{10}, x_{20}, \ldots, x_{n0})$ и рассмотрим функцию f при фиксированных значениях всех переменных, кроме первой: $f(x_1, x_{20}, x_{30}, \ldots, x_{n0})$. Тогда она превратится в функцию одной переменной x_1. Изменяя эту переменную, будем двигаться от начальной точки $x_1 = x_{10}$ в сторону убывания функции, пока не дойдем до ее минимума при $x_1 = x_{11}$, после которого она начинает возрастать. Точку с координатами $(x_{11}, x_{20}, x_{30}, \ldots, x_{n0})$ обозначим через M_1,при этом $f(M_0) > f(M_1)$.

Фиксируем теперь переменные $x_1 = x_{11}$, $x_3 = x_{30}$, $x_4 = x_{40} \ldots$, $x_n = x_{n0}$ и рассмотрим функцию f как функцию одной переменной x_2: $(x_{11}, x_2, x_{30}, x_{40}, \ldots, x_{n0})$. Изменяя x_2, будем опять двигаться от начального значения $x_2 = x_{20}$ в сторону убывания функции, пока не дойдем до минимума при $x_2 = x_{21}$. Точку с координатами $(x_{11}, x_{21}, x_{30}, x_{40}, \ldots, x_{n0})$ обозначим через M_2, при этом $f(M_1) > f(M_2)$.

Проведем такую же минимизацию целевой функции по переменным $x_3, x_4, \ldots, x_n$. Дойдя до переменной x_n, снова вернемся к x_1 и продолжим.

Эта процедура вполне оправдывает название метода. С ее помощью мы строим последовательность точек $M_0, M_1, M_2, \ldots$, которой соответствует монотонная последовательность значений функции $f(M_0) > f(M_1) > f(M_2) > \ldots$.

Поиск прекращается, когда расстояние между точками M_i и M_{i-1} n-мерного вещественного пространства R_n, полученными в двух соседних итерациях, т.е. разделенными циклом вычислений по всем переменным, станет меньше некоторой, наперед заданной достаточно малой положительной величины ε.

Данный метод сводит задачу поиска наименьшего значения функции нескольких переменных к многократному решению одномерных задач оптимизации. Для решения же одномерных задач можно использовать любой известный метод, например, методы: полного перебора, производной, золотого сечения и т.д.

Пример.

Пусть целевая функция имеет вид: $U = x_1^2 + x_2^2 + 1{,}5 \cdot x_1 \cdot x_2$ и поиск начинается из точки $M_0 = (3;3)$; $f(M_0) = 31{,}5$. Точность вычислений $\varepsilon = 0{,}01$.

Сначала изменяем координату x_1, оставив x_2 постоянной и равной $x_{20} = 3$. Тогда целевая функция будет иметь вид $U = f(x_1, 3) = x_1^2 + 9 + 4{,}5 \cdot x_1$. Минимум $U = f(x_1, 3)$ найдем, приравнивая к нулю производную $\frac{df(x_1,3)}{dx_1}$, т.е.:

$$\frac{df(x_1,3)}{dx_1} = 2 \cdot x_1 + 4{,}5 = 0.$$

Отсюда первый экстремум по x_1 равен $x_{11} = -2{,}25$ и ему соответствует точка M_1 с координатами $x_{11} = -2{,}25$, $x_{20} = 3$, т.е. $M_1 = (-2{,}25;3)$; $f(M_1) = 3{,}83 < f(M_0)$.

Теперь оставим неизменной координату x_1 и равной -2,25, т.е. $x_{11} = -2{,}25$, а будем изменять x_2.

Целевая функция в этом случае примет вид $U = f(-2{,}25, x_2) = (-2{,}25)^2 + x_2^2 + 1{,}5 \cdot (-2{,}25) \cdot x_2$.

Далее находим минимум функции по x_2:

$$\frac{df(-2{,}25, x_2)}{dx_2} = 2 \cdot x_2 - 3{,}375 = 0.$$

Отсюда следует, что $x_{22} = 1{,}68$.

Данному экстремуму соответствует точка $M_2 = (-2{,}25;1{,}68)$; $f(M_2) = 2{,}21 < f(M_1) < f(M_0)$.

Далее необходимо проверить условие окончания поиска минимума – определить расстояние между точками, разделенными циклом вычислений по всем переменным, т.е. в данном случае между точками $M_0 = (3;3)$ и $M_2 = (-2{,}25;1{,}68)$, и сравнить его с заданной точностью вычислений:

$$s_1 = \sqrt{(x_{11} - x_{10})^2 + (x_{21} - x_{20})^2} = \sqrt{(-2{,}25 - 3)^2 + (1{,}68 - 3)^2} = 5{,}41 .$$

Так как $s_1 > \varepsilon$, то необходимо продолжить поиск минимума целевой функции.

Теперь опять повторяем цикл вычислений для x_1, закрепляя $x_{22} = 1{,}68$:

$$U = f(x_1, 1{,}68) = x_1^2 + 1{,}68^2 + 1{,}5 \cdot 1{,}68 \cdot x_1 ;$$

$$\frac{df(x_1, 1{,}68)}{dx_1} = 2 \cdot x_1 + 1{,}5 \cdot 1{,}68 = 0 .$$

Из последнего уравнения следует, что $x_{12} = -1{,}26$. Найденному экстремуму соответствует точка $M_3 = (-1{,}26; 1{,}68)$.

$$f(M_3) = 1{,}26 < f(M_2) < f(M_1) < f(M_0) .$$

Далее аналогично продолжаем процесс. В результате поиска получаем ломаную линию, состоящую из взаимно перпендикулярных прямых, точки излома которой - это точки $M_0, M_1, M_2, \cdots$. Они находятся в местах касания этих прямых с линиями уровня целевой функции:

$$U = f(x_1, x_2) = const .$$

Последовательность поиска минимума целевой функции, зависящей от двух переменных $U = f(x_1, x_2)$ методом покоординатного спуска иллюстрируется рис. 3.1.

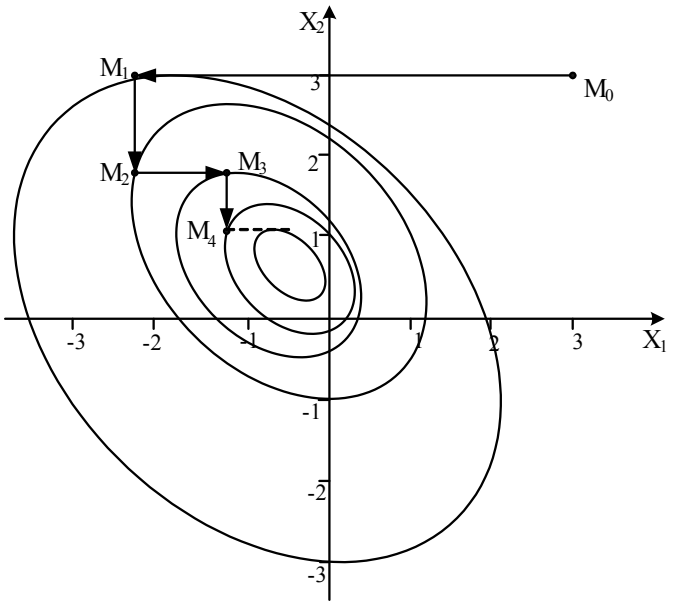


Рис. 3.1. Поиск минимума методом покоординатного спуска

Блок-схема алгоритма реализации метода покоординатного спуска представлена на рис. 3.2. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

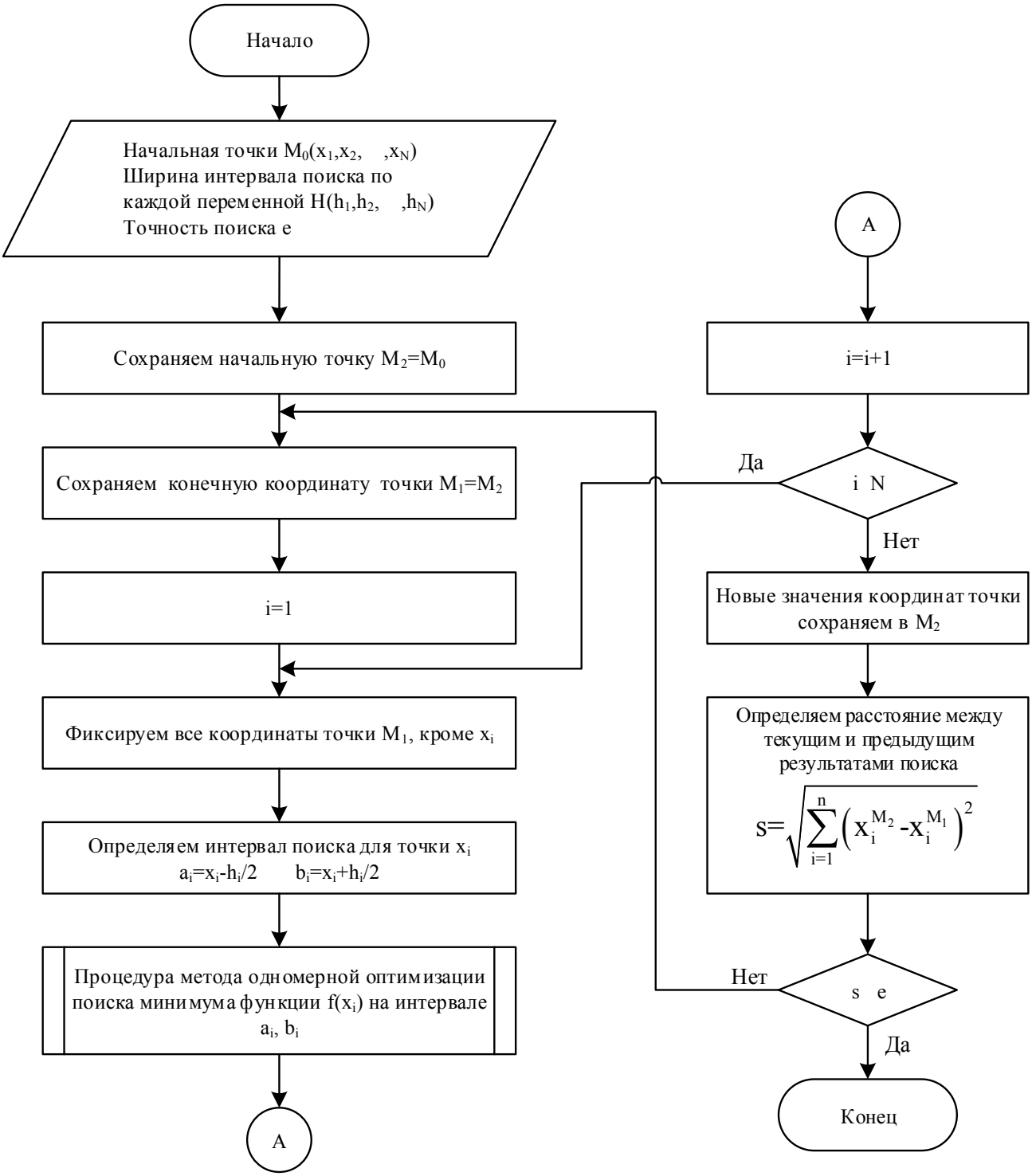


Рис. 3.2. Блок-схема алгоритма реализации метода покоординатного спуска

3.2. Решение задачи многомерной оптимизации градиентными методами

Рассмотрим функцию трех переменных $f(x,y,z)$. Вычислим частные производные: $\frac{df}{dx};\frac{df}{dy};\frac{df}{dz}$ и составим с их помощью вектор, который называется градиентом функции:

$$Grad\ f(x,y,z)=\frac{df(x,y,z)}{dx}i+\frac{df(x,y,z)}{dy}j+\frac{df(x,y,z)}{dz}k, \qquad (3.1)$$

где i,j,k – единичные векторы, параллельные координатным осям (орты).

Частные производные характеризуют изменение функции по каждой независимой переменной в отдельности.

Вектор градиента дает общее представление о поведении функции в окрестности точки x,y,z. В направлении градиента функция возрастает с максимальной скоростью. В обратном направлении, которое часто называют антиградиентным, функция убывает с максимальной скоростью.

Модуль градиента определяется по формуле:

$$|Grad\ f(x,y,z)|=\sqrt{\left(\frac{df}{dx}\right)^2+\left(\frac{df}{dy}\right)^2+\left(\frac{df}{dz}\right)^2}. \qquad (3.2)$$

Модуль градиента определяет скорость возрастания и убывания целевой функции в направление градиента и антиградиента. В других направлениях целевая функция будет изменяться с меньшей скоростью. При переходе от одной точке к другой меняется как модуль градиента, так и его направление.

Для нахождения минимума целевой функции методом градиентного спуска выбираем каким-либо способом начальную точку M_0, выделяем в ней градиент целевой функции и делаем небольшой шаг в антиградиентном направлении. В результате мы придем в точку, в которой целевая функция будет иметь меньшее значение. Во вновь полученной точке снова вычисляем градиент и снова делаем шаг в антиградиентном направлении. Продолжая этот процесс, будем двигаться в сторону уменьшения целевой функции. Выбор направления движения позволяет надеяться, что продвижение к точке минимума в этом случае будет более быстрым, чем в методе покоординатного спуска.

Поскольку градиент – это вектор, а при выполнении каких-либо операций мы должны использовать однородные математические объекты, то вместо точки n-мерного пространства будем пользоваться понятием радиуса вектора этой точки:

точка $M_0=(x_0,y_0,z_0)$;

радиус вектор $\overline{OM_0}=x_0\cdot i+y_0\cdot j+z_0\cdot k$;

$$\overline{OM_1}=\overline{OM_0}-h*Grad\ f(M_0)=i\left[x_0-h\frac{df}{dx}(M_0)\right]+j\left[y_0-h\frac{df}{dy}(M_0)\right]+k\left[z_0-h\frac{df}{dz}(M_0)\right];$$

$$M_1=\left(x_0-h\frac{df}{dx}(M_0);y_0-h\frac{df}{dy}(M_0);z_0-h\frac{df}{dz}(M_0)\right).$$

В общем виде, координата точки M_i выражается через координату точки M_{i-1}:

$$M_i=\left(x_{i-1}-h\frac{df}{dx}(M_{i-1});y_{i-1}-h\frac{df}{dy}(M_{i-1});z_{i-1}-h\frac{df}{dz}(M_{i-1})\right). \qquad (3.3)$$

Формулы для вычисления частных производных можно получить только в том случае, если целевая функция задана аналитически.

В противном случае, эти производные вычисляются с помощью численного дифференцирования.

$$\frac{df}{dx_i} = \frac{1}{\Delta x_i}\left[f\left(x_1, x_2, x_3, ..., x_i + \Delta x_i, ..., x_n\right) - f\left(x_1, x_2, ..., x_i, ..., x_n\right)\right].$$

Поиск прекращается, когда модуль градиента целевой функции станет меньше или равным некоторого малого числа ε, т.е. когда:

$$\left|Grad\ f\right| \le \varepsilon .$$

Метод наискорейшего спуска

При использовании градиентного спуска в задачах оптимизации основной объем вычислений приходится обычно на вычисление градиента целевой функции в каждой точке траектории спуска. Поэтому целесообразно уменьшить количество таких точек без ущерба для самого решения. Это достигается в методе наискорейшего спуска, который является модификацией классического градиентного метода.

Согласно этому методу, после определения в начальной точке антиградиентного направления, в этом направлении делают не один шаг, а двигаются до тех пор, пока целевая функция убывает в этом направлении, достигая таким образом минимума в некоторой точке. В этой точке снова определяют направление спуска (с помощью градиента) и ищут новую точку минимума целевой функции и т.д.

В этом методе спуск происходит гораздо более крупными шагами и градиент функции вычисляется в меньшем числе точек.

Рассмотрим пример определения минимума целевой функции по методу наискорейшего спуска.

Необходимо найти минимум целевой функции: $f\left(x_1, x_2\right) = {x_1}^2 + {x_2}^2 + 1{,}5x_1x_2$.

Начальная точка имеет координаты: $M_0 = (2,3)$. Точность вычислений $\varepsilon = 0{,}01$.

Определим частные производные целевой функции по координатам x_1, x_2:

$\frac{df}{dx_1}(M_0) = 2x_1 + 1{,}5x_2 = 2 \cdot 2 + 1{,}5 \cdot 3 = 8{,}5$:

$\frac{df}{dx_2}(M_0) = 2x_2 + 1{,}5x_1 = 2 \cdot 3 + 1{,}5 \cdot 2 = 9.$

Запишем выражения для определения координат на следующей итерации, полагая шаг h_1 неизвестным:

$$x_{11} = x_{10} - h_1 \frac{df}{dx_1}(M_0) = 2 - h_1 \cdot 8{,}5;$$

$$x_{21} = x_{20} - h_1 \frac{df}{dx_2}(M_0) = 3 - h_1 \cdot 9.$$

Подставим эти значения в целевую функцию:

$$f\left(x_{11}, x_{21}\right) = \left(2 - 8{,}5h_1\right)^2 + \left(3 - 9h_1\right)^2 + 1{,}5\left(2 - 8{,}5h_1\right)\left(3 - 9h_1\right) = 268\,{h_1}^2 - 166{,}75h_1 + 22.$$

Для определения величины оптимального шага h_1 можно воспользоваться любым методом одномерной оптимизации, минимизируя значение целевой функции, которая зависит только от шага h_1.

В данном примере воспользуемся методом производной. Возьмем частную производную по h_1 и, приравняв ее нулю, найдем величину шага h_1, при котором мы приходим в точку минимума:

$$\frac{df}{dh_1} = 536h_1 - 166{,}75 = 0; h_1 = 0{,}31;$$

$$\frac{df^2}{dh_1^2} = 536 > 0.$$

Следовательно, точка (x_{11}, x_{21}) – это точка минимума.

$x_{11} = 2 - 0{,}31 \cdot 8{,}5 = -0{,}635;$

$x_{21} = 3 - 0{,}31 \cdot 9 = 0{,}21;$

$M_1(-0{,}635;0{,}21).$

Далее необходимо проверить условие окончания поиска минимума – определить величину модуля градиента целевой функции в найденной точке $M_1(-0{,}635;0{,}21)$ и сравнить его с заданной точностью вычислений:

$$|Grad\ f(M_1)| = \sqrt{\left(\frac{df(M_1)}{dx_1}\right)^2 + \left(\frac{df(M_1)}{dx_2}\right)^2} = \sqrt{(2\cdot(-0{,}635) + 1{,}5\cdot 0{,}21)^2 + (2\cdot 0{,}21 + 1{,}5\cdot(-0{,}635))^2} = 1{,}75.$$

Так как $|Grad\ f(M_1)| > \varepsilon$, то необходимо продолжить поиск минимума целевой функции.

Повторяем процедуру, считая начальной точкой точку M_1:

$$\frac{df}{dx_1}(M_1) = 2x_1 + 1{,}5x_2 = 2\cdot(-0{,}635) + 1{,}5\cdot 0{,}21 = -0{,}955;$$

$$\frac{df}{dx_2}(M_1) = 2x_2 + 1{,}5x_1 = 2\cdot 0{,}21 + 1{,}5\cdot(-0{,}635) = -0{,}5325;$$

$$x_{12} = x_{11} - h_2\frac{df}{dx_1}(M_1) = -0{,}635 + h_2\cdot 0{,}955;$$

$$x_{22} = x_{21} - h_2\frac{df}{dx_2}(M_1) = 0{,}21 + h_2\cdot 0{,}5325;$$

$$f(x_{12}; x_{22}) = (-0{,}635 + 0{,}955h_2)^2 + (0{,}21 + 0{,}5325h_2)^2 + 1{,}5(0{,}635 + 0{,}955h_2)(0{,}21 + 0{,}5325h_2);$$

$$f(x_{12}; x_{22}) = 0{,}2473 - 1{,}19558h_2 + 1{,}9559{h_2}^2;$$

$$\frac{df}{dh_2} = -1{,}19558 + 2 \cdot 1{,}9559h_2 = 0; h_2 = 0{,}3052;$$

$$x_{12} = -0{,}635 + 0{,}3052 \cdot 0{,}955 = -0{,}34343;$$

$$x_{22} = 0{,}21 + 0{,}3052 \cdot 0{,}5325 = 0{,}37254;$$

$$M_2 = (-0{,}34343; 0{,}37254); \ldots$$

и т.д.

Блок-схема алгоритма реализации метода наискорейшего спуска представлена на рис. 3.3. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

3.3. Решение задачи многомерной оптимизации методом конфигураций (Хука-Дживса)

Градиентные методы медленно сходятся, когда поверхности уровня целевой функции $f(x_1, x_2, \ldots, x_n)$ сильно вытянуты. Это свойство известно в литературе как эффект «оврагов» или эффект «гребней».

Суть эффекта состоит в том, что небольшое изменение одних переменных приводит к резкому изменению значения целевой функции. Эта группа переменных характеризует «склон оврага». По другим же переменным, задающим направление «дна оврага», целевая функция изменяется незначительно.

Обычно траектория градиентного метода характеризуется быстрым спуском на «дно оврага» и затем медленным зигзагообразным движением по «дну оврага» к точке минимума. Метод конфигураций позволяет значительно ускорить процедуру поиска минимума целевой функции.

Поиск начинают из некоторой произвольно выбранной точки b_1 (рис. 3.4) и ведут шагами δ_i по каждой независимой переменной x_i. Под b_1 следует понимать радиус вектор этой точки, а под δ_i также вектор, все составляющие которого, кроме δ_i, равны нулю.

Сначала проверяется точка $b_1 + \delta_1$. Если целевая функция в этой точке имеет меньшее значение, чем в точке b_1, то эта точка называется временной вершиной и обозначается через t_{11}. Если целевая функция в точке $b_1 + \delta_1$ имеет большее значение, то проверяется точка $b_1 - \delta_1$. Может оказаться, что обе точки будут непригодными, т.е. целевая функция будет иметь большее значение, чем в начальной точке. В этом случае временной вершиной будет являться сама точка b_1.

Двойной индекс временной вершины t_{11} означает, что ведется 1-ый этап поиска и первые шаги выполнялись по 1-ой независимой переменной.

$$t_{11} = \begin{cases} b_1 + \delta_1, & \textit{если} \quad f(b_1 + \delta) < f(b_1); \\ b_1 - \delta_1, & \textit{если} \quad f(b_1 - \delta_1) < f(b_1) < f(b_1 + \delta_1); \\ b_1, & \textit{если} \quad f(b_1) < \min\left[f(b_1 + \delta_1); f(b_1 - \delta_1)\right]. \end{cases} \qquad (3.4)$$

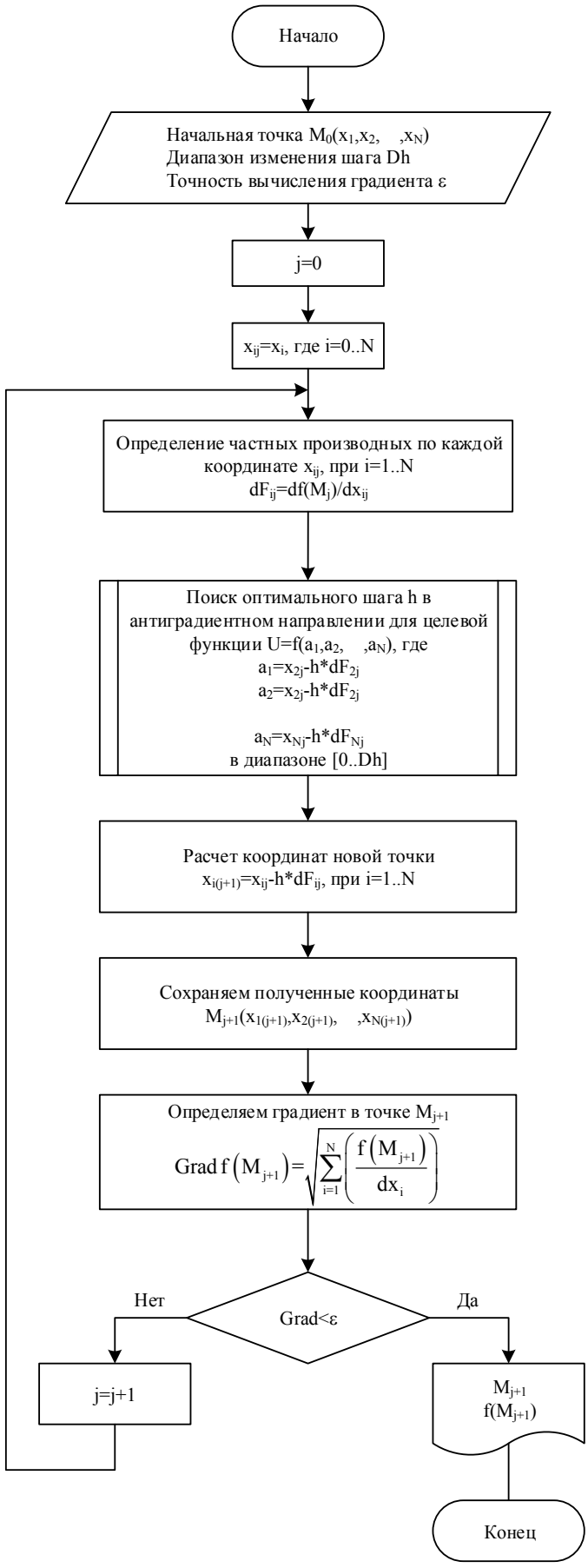


Рис. 3.3. Блок-схема алгоритма реализации метода наискорейшего спуска

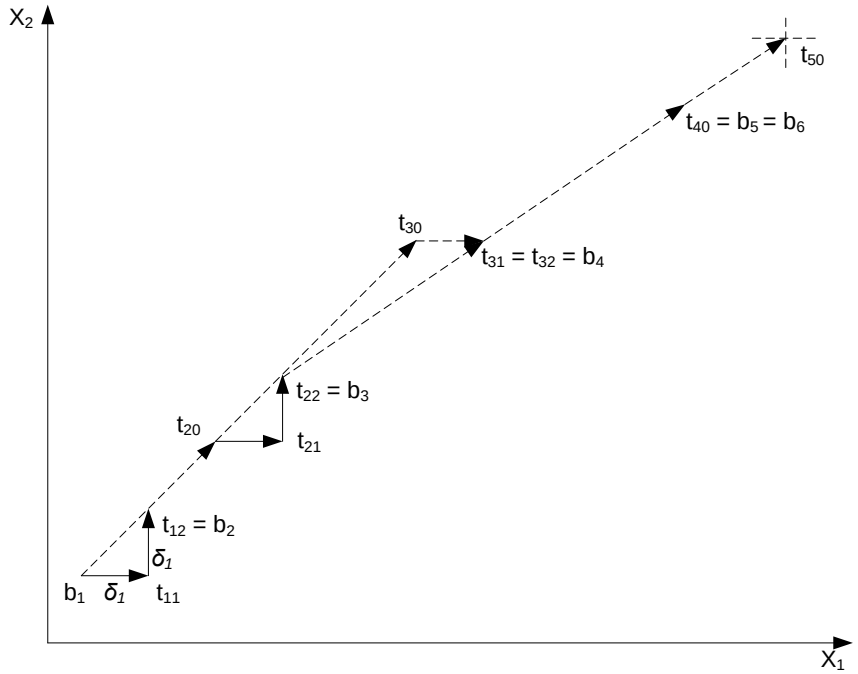


Рис.3.4. Графическое представление процесса поиска

Далее аналогичным образом выполняются пробные шаги по второй независимой переменной x_2.

В общем случае для временной вершины $t_{1,j}$, полученной продвижением из временной вершины $t_{1,j-1}$ можно записать:

$$t_{1,j} = \begin{cases} t_{1,j-1} + \delta_j, & \text{если} \quad f\left(t_{1,j-1} + \delta_j\right) < f\left(t_{1,j-1}\right); \\ t_{1,j-1} - \delta_j, & \text{если} \quad f\left(t_{1,j-1} - \delta_j\right) < f\left(t_{1,j-1}\right) < f\left(t_{1,j-1} + \delta_j\right); \\ t_{1,j-1}, & \text{если} \quad f\left(t_{1,j-1}\right) < \min\left[f\left(t_{1,j-1} + \delta_j\right); f\left(t_{1,j-1} - \delta_j\right)\right] \end{cases} \qquad (3.5)$$

В результате 1-го этапа поиска по всем независимым переменным будут последовательно найдены вершины $t_{11}, t_{12}, t_{13}, \ldots, t_{1n}$. Временную вершину t_{1n} называют второй базовой точкой и обозначают через b_2.

Дальнейшая процедура поиска основывается на следующей гипотезе. Считается, что для скорейшего продвижения к точке минимума из новой базисной точки b_2 нужно продвигаться в том же направлении, что и на предыдущем участке. Опираясь на эту гипотезу, отказываются от выполнения пробных шагов в районе базовой точки b_2, а сразу продвигаются в направлении вектора $\overline{(b_1 b_2)}$ на двойную его длину (отсчет ведется от точки b_1). В результате такого продвижения получаем вершину t_{20}:

$$t_{20} = b_1 + 2(b_2 - b_1) = 2b_2 - b_1. \qquad (3.6)$$

Двойной индекс t_{20} означает:

2 - ведется второй этап поиска;

0 - изменению не подвергается ни одна из независимых переменных.

В районе временной вершины t_{20} аналогичным образом выполняют пробные шаги по

всем независимым переменным. Временную вершину t_{22} называют третьей базовой точкой и обозначают b_3.

Далее также в районе точки b_3 отказываются от выполнения пробных шагов, а сразу, продвигаются в направлении вектора $\overline{(b_2 b_3)}$ на двойную его длину. В результате получаем точку t_{30}.

Пусть в результате выполнения пробных шагов в районе временной вершины t_{30} оказалось, что $t_{31} = t_{32} = b_4$. Это приводит к изменению направления движения к оптимуму, и в результате движения по этому направлению получаем временную вершину t_{40}.

Пусть в результате выполнения пробных шагов в районе временной вершины t_{40}, оказалось, что $t_{40} = t_{41} = t_{42}$. Кроме того, пусть $f(t_{40}) < f(b_4)$. Тогда временная вершина t_{40} называется 5-ой базисной точкой и обозначается b_5.

Далее аналогичным образом продвигаемся в направлении вектора $\overline{(b_4 b_5)}$ на двойную его длину и получаем временную вершину t_{50}.

В результате выполнения пробных шагов в районе временной вершины t_{50} оказалось, что $t_{50} = t_{51} = t_{52}$, кроме того $f(t_{50}) > f(b_5)$. В этом случае считают, что b_6 совпадает с базовой точкой b_5. На этом первый цикл поиска заканчивается.

Второй цикл поиска начинается из базовой точки b_6, причем сначала в районе базовой точки b_6 выполняются пробные шаги той же длины, что и на предыдущем цикле вычислений. Если все они будут неудачными, то размер шага по каждой независимой переменной уменьшается в несколько раз и процедура вычислений повторяется.

Поиск прекращается, как только размер шага по каждой переменной не будет превышать некоторой достаточно малой величины ε.

Блок-схема реализации поиска по образцу (исследование вокруг базисной точки) представлена на рис. 3.5.

Блок-схема основного поиска решения представлена на рис. 3.6. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

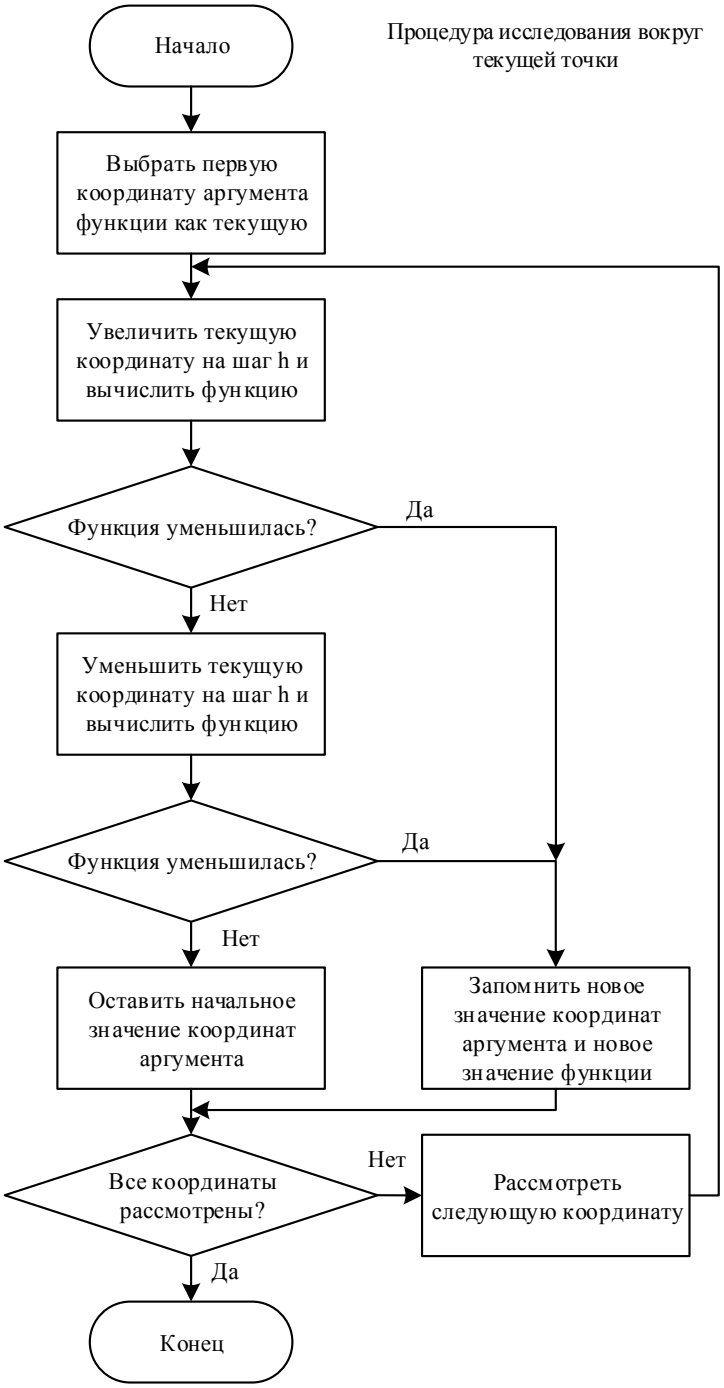


Рис. 3.5. Блок-схема реализации поиска по образцу (исследование вокруг базисной точки)

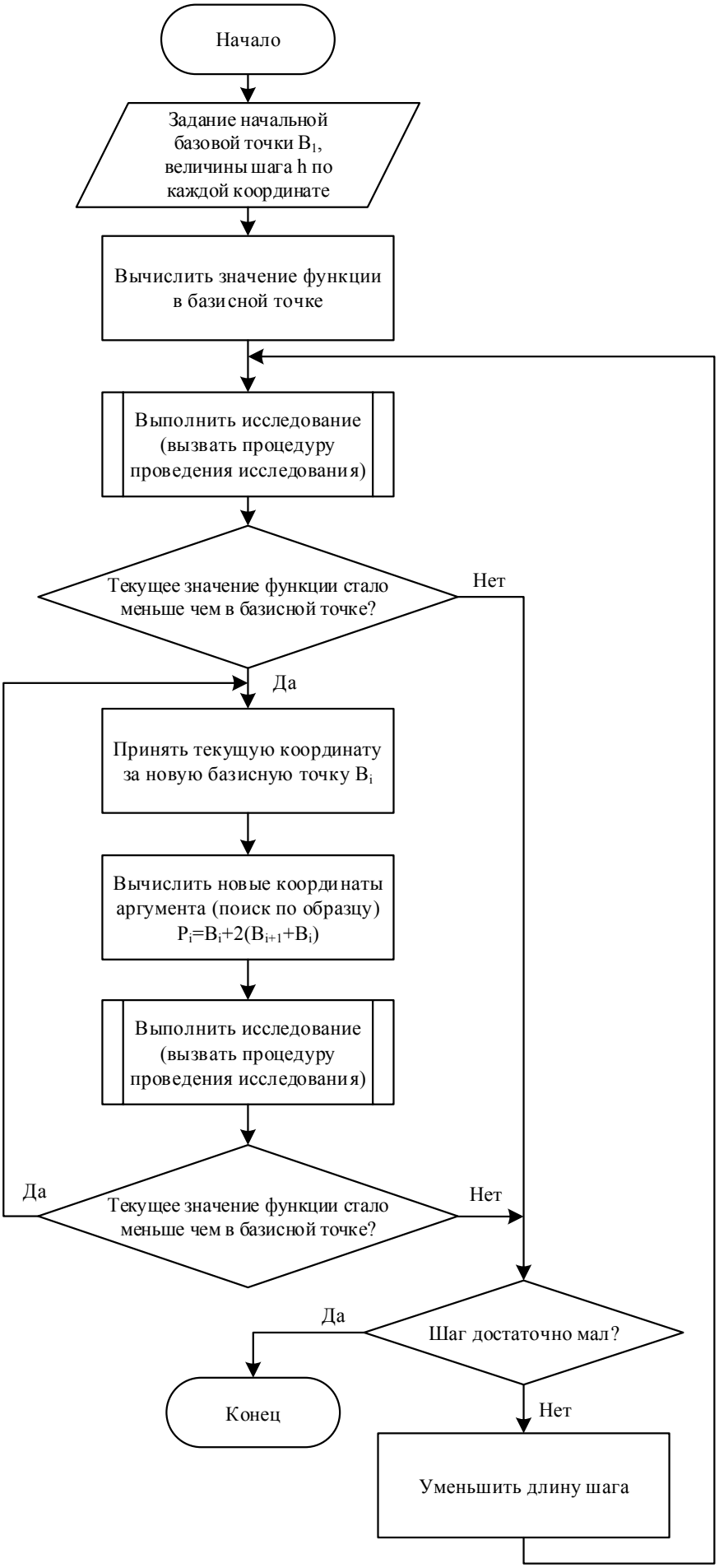


Рис. 3.6. Блок-схема основного поиска решения

3.4. Симплексный метод прямого поиска Нелдера-Мида

На разработку методов прямого поиска для определения минимума функции n переменных было затрачено много усилий. Методы прямого поиска являются методами, в которых используются только значения функции. Один из наиболее надежных метод Нелдера-Мида, являющийся одним из самых эффективных, если $n \le 6$.

Данный метод предназначен для минимизации функции n действительных переменных с использованием лишь вычисляемых на каждом шаге значений минимизируемой функции (метод нулевого порядка). Для него не требуется и в нем не используется (явно или неявно) информация о производных минимизируемой функции. Таким образом, метод Нелдера-Мида относится к общему классу прямых методов поиска минимума функции. Как и многие другие методы, относящиеся к подмножеству прямых методов, метод Нелдера-Мида на каждом шаге итеративного процесса хранит невырожденный симплекс – геометрический объект в n –мерном пространстве, являющийся выпуклой оболочкой, натянутой на $(n+1)$ вершину. В двумерном пространстве симплексом является равносторонний треугольник, а в трехмерном пространстве – правильный тетраэдр.

Идея метода состоит в сравнении значений функции в $(n+1)$ вершинах симплекса и перемещении симплекса в направлении оптимальной точки с помощью итерационной процедуры. Каждая итерация прямого симплекс-метода поиска минимума начинается с построения симплекса, который задается своими $(n+1)$-й вершинами и вычисляемыми в этих вершинах значениями функции. Затем многогранник дополняется одной либо несколькими точками вместе со значениями функции в них. Одна или несколько вершин после этого отбраковываются. Итерационный процесс завершается тогда, когда вершины симплекса и вычисленные в них значения функции при сравнении с предыдущей итерацией удовлетворяют некоторым условиям сходимости.

В данном методе симплекс перемещается с помощью четырех основных операций: отражение, растяжение, усечение и сжатие. Для полного задания симплекс метода необходимо определить три константы:

- коэффициент отражения α;
- коэффициент растяжения γ;
- коэффициент сжатия β.

Согласно оригинальному алгоритму, эти константы должны удовлетворять условиям:

$$\alpha > 0,\ \gamma > 1,\ \gamma > \alpha,\ 0 < \beta < 1.$$

Практически во всех стандартных реализациях метода коэффициенты выбираются равными:

$$\alpha = 1,\ \gamma = 2,\ \beta = ½.$$

Рассмотрим основные шаги процедуры:

Шаг 1. Выбирается первоначальный симплекс с вершинами X_1, X_2… X_{N+1}, где X_i – вектор координаты вершины $X_i = (x_1, x_2, \dots, x_n)$ и вычисляются значения функции $f_1=f(X_1)$, $f_2=f(X_2)$,…, $f_{n+1}=f(X_{N+1})$.

Шаг 2. Среди всех найденных значений определяем наибольшее значение функции f_{MAX}, наименьшее значение функции f_{MIN}, и следующее за наибольшим значение функции f_G, а также соответствующие им координаты вершин симплекса X_{MAX}, X_{MIN} и X_G соответственно.

Шаг 3. Находится центр тяжести X_0 всех, за исключением X_{MAX}, вершин симплекса, по формуле:

$$X_0 = \frac{1}{n} \cdot \sum_{i \neq \max} X_i , \tag{3.6}$$

и вычисляется $f_0=f(X_0)$.

Шаг 4. «Отражаем» точку X_{MAX} относительно точки X_0, получаем точку X_R и находим значение функции в этой точке $f_R=f(X_R)$ (рис. 3.7).

Величина отрезка X_0X_R определяется коэффициентом отражения α. При $\alpha>0$ получим:

$$X_R - X_0 = \alpha \cdot (X_0 - X_{MAX}), \tag{3.7}$$

где α - коэффициент отражения.

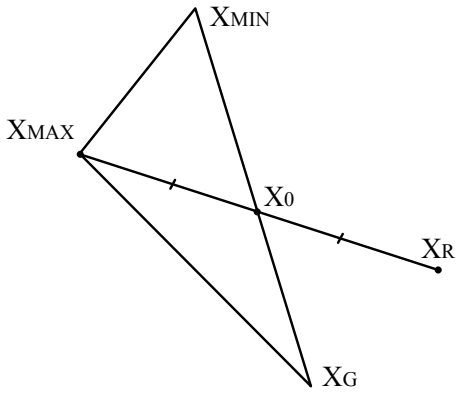


Рис. 3.7. Процедура «отражения» симплекса

Шаг 5. Сравниваются значение функции в точках X_R и X_{MIN}, т.е. производится сравнение значений f_{MIN} и f_R. Дальнейший ход поиска определяется отношением этих величин.

5.1. Если $f_R<f_{MIN}$, то это значит, что мы получили наименьшее значение функции. Направление из точки X_0 в точку X_R наиболее удобно для перемещения. В этом случае производится дальнейшее перемещение до точки X_E и находится значение функции в этой точке: $f_E=f(X_E)$. Эта процедура называется «растяжением» симплекса (рис. 3.8), а точка X_E определяется из соотношения:

$$X_E = \gamma \cdot X_R + (1-\gamma)X_0 , \tag{3.8}$$

где γ - коэффициент растяжения.

Если $f_E<f_{MIN}$, то заменяем точку X_{MAX}, на точку X_E и проверяем симплекс на сходимость к минимуму (**шаг 9**). Если сходимость достигнута, то процесс останавливается, в противном случае возвращается на **шаг 2**.

Если $f_E>f_{MIN}$, то отбрасываем точку f_E. Очевидно, перемещение было слишком большим от точки X_R к точке X_E, поэтому заменяем точку X_{MAX} на точку X_R, в которой было получено улучшение и переходим к **шагу 9** для проверки на сходимость.

5.2. Если $f_R>f_{MIN}$, но $f_R<f_G$, то X_R является лучшей точкой по сравнению с X_{MAX} и X_G. В этом случае производится замена X_{MAX} на X_R и переход к **шагу 9**.

5.3. Если $f_R>f_{MIN}$ и $f_R>f_G$, то переходим к **шагу 6**.

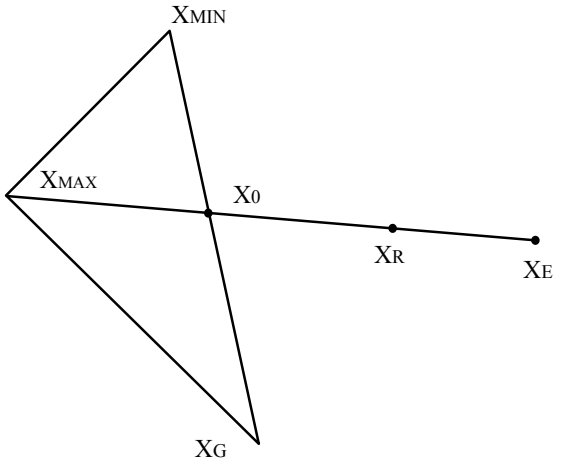


Рис. 3.8. Процедура «растяжения» симплекса

Шаг 6. Сравним значение f_R и f_{MAX}. В зависимости от соотношения этих величин дальнейший ход поиска определяется следующими шагами (рис. 3.9).

6.1. Если $f_R > f_{MAX}$, то переходим к шагу сжатия 6.2. Если $f_R < f_{MAX}$, заменяем точку X_{MAX} на точку X_R и значение функции f_{MAX} на значение f_R и переходим на шаг 6.2.

6.2. В этом случае $f_R \geq f_{MAX}$, поэтому ясно, что перемещение от точки X_{MAX} по направлению к точке X_0 было слишком большое, поэтому для исправления такой ситуации найдем точку X_C (а затем значение функции в этой точке f_C) с помощью шага «сжатия».

Если $f_R > f_{MAX}$ то находим точку X_C из соотношения:

$$\mathrm{X_i} = \left(\mathrm{X_i} + \mathrm{X_{MIN}}\right)/2$$

$$X_c = \beta \cdot X_{MAX} + (1-\beta)X_0, \qquad (3.9)$$

где β - коэффициент сжатия.

Если $f_R = f_{MAX}$ (т.е. в шаге 6.1. была проведена замена f_{MAX} на f_R), то точка X_C найдется из соотношения:

$$X_c = \beta \cdot X_R + (1-\beta)X_0. \qquad (3.10)$$

Шаг 8. На этом шаге уменьшаем размер симплекса делением пополам расстояний от каждой точки симплекса до точки X_{MIN} – точки, определяющей наименьшее значение функции. Очевидно, что центр тяжести всех точек сместится по направлению к минимуму функции. Таким образом, точки X_i заменятся на точки

$$X_i = (X_i + X_{MIN})/2. \qquad (3.11)$$

Затем вычисляются значения функции в этих точках и переходим на **шаг 9**.

Шаг 9. Проверяем сходимость метода для определения окончания итерационной процедуры. Проверка сходимости основана на том, чтобы стандартное отклонение значений функции вычисленных в вершинах симплекса было бы меньше заданного малого значения ε. Для этого вычисляется

$$\sigma^2 = \sum_{i=1}^{n+1} \left(f_i - \overline{f}\right)^2 /(n+1), \qquad (3.12)$$

где $\overline{f} = \sum_{i=1}^{n+1} f_i /(n+1)$.

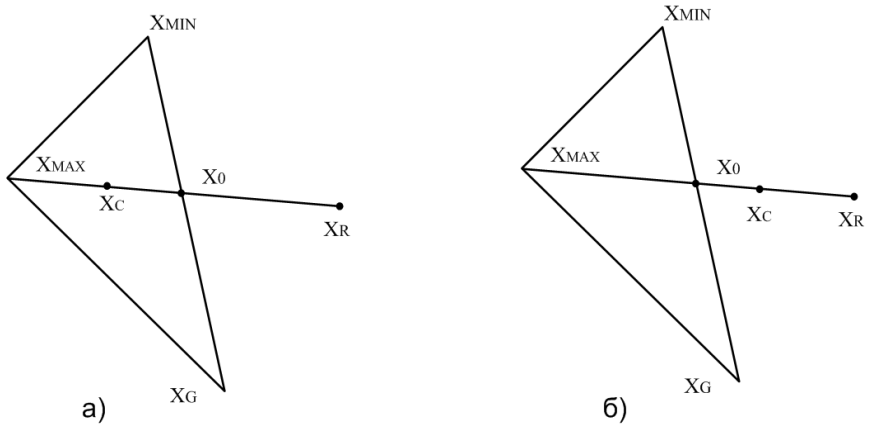


Рис. 3.9. Процедура «сжатия» симплекса

Если $\sigma<\varepsilon$, то все значения функции в вершинах симплекса очень близки к друг другу, и поэтому они, возможно, лежат вблизи точки минимума функции и процедура поиска на этом заканчивается. В противном случае поиск продолжается. Принимаем текущие координаты точек симплекса за начальные координаты и переходим на **шаг 2**.

Графическое представление поиска по данному методу показано на рис. 3.10. Блок-схема алгоритма реализации метода Нелдера-Мида представлена на рис. 3.10. Пример программной реализации метода на языке Visual Basic for Application приведен в Приложении.

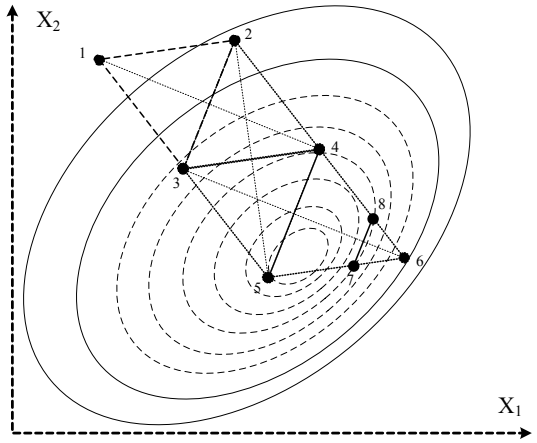


Рис. 3.10. Графическое представление поиска по методу Нелдера-Мида

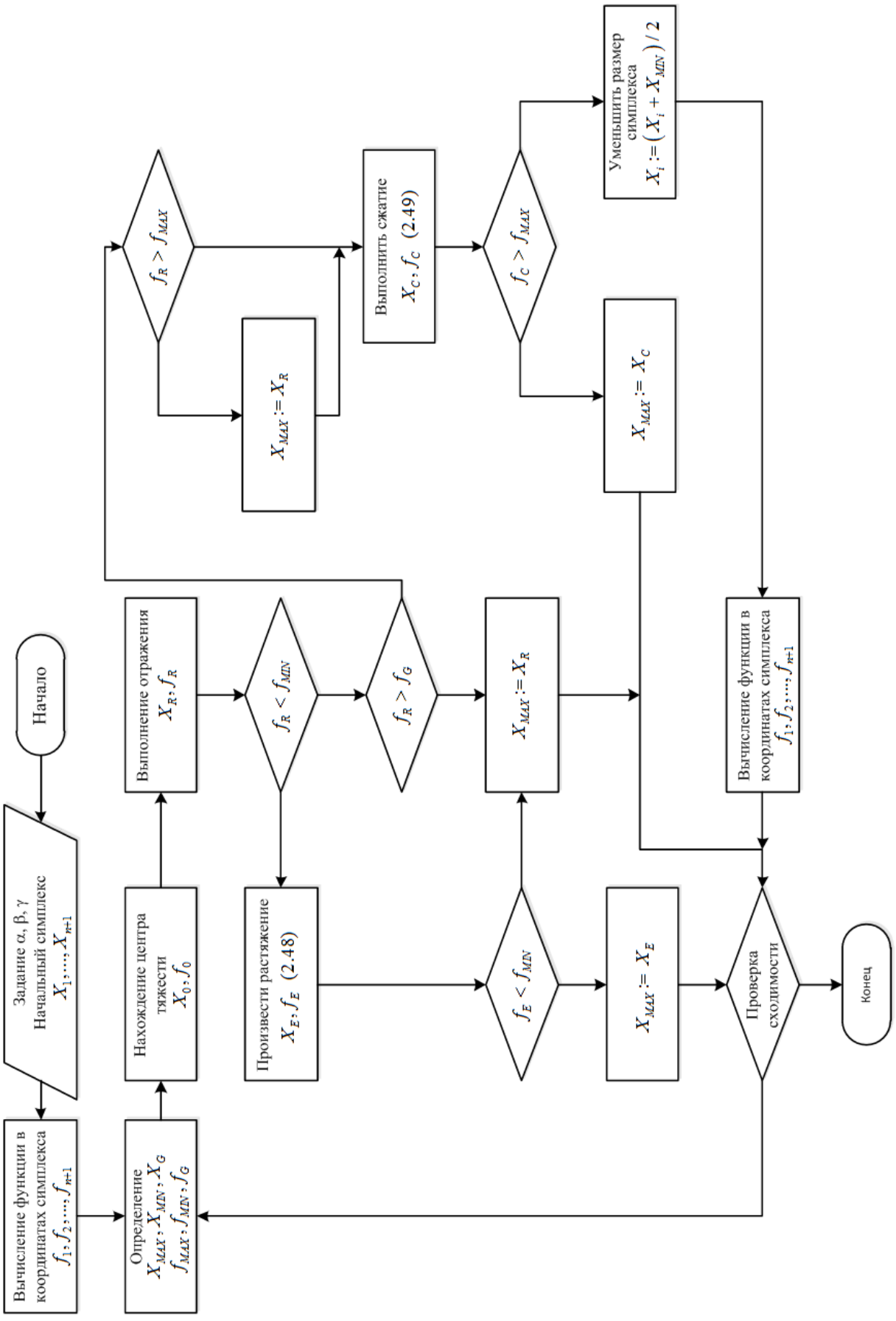


Рис. 3.11. Блок-схема алгоритма реализации метода Нелдера-Мида

3.5. Контрольные вопросы

1. Алгоритм поиска минимума методом покоординатного спуска.
2. В чем смысл условия прекращения поиска в методе покоординатного спуска?
3. Что такое градиент функции? Что он характеризует?
4. Как определяется модуль градиента? Что он определяет?
5. Сущность метода градиентного спуска.
6. Чем метод наискорейшего спуска отличается от метода градиентного спуска?
7. Условие окончание поиска в градиентных методах.
8. Что такое эффект «оврагов»?
9. Что такое поиск по образцу в методе конфигурации?
10. Когда прекращается поиск минимума в методе Хука-Дживса?
11. В чем заключается суть процедур «отражения», «растяжения» и «сжатия» в методе Нелдера-Мида?Каковы условия окончания поиска минимума целевой функции в методе золотого сечение и в методе квадратичной интерполяции?

3.6. Задачи для самостоятельного решения

Задача №1.

Необходимо минимизировать методом многомерной оптимизации целевую функцию с абсолютной погрешностью, не превышающей 0,01:

$$U = f(x_1, x_2) = a \cdot x_1 + b \cdot x_2 + \exp\left(c \cdot x_1^2 + d \cdot x_2^2\right).$$

Таблица 3.1

Варианты заданий

Номер варианта	a	b	c	d
1	1,0	-1,4	0,01	0,11
2	2,0	-1,3	0,04	0,12
3	3,0	-1,2	0,02	0,13
4	4,0	-1,1	0,16	0,14
5	5,0	-1,0	0,25	0,15
6	6,0	-0,9	0,36	0,16
7	7,0	-0,8	0,49	0,17
8	8,0	-0,7	0,64	0,18
9	9,0	-0,6	0,81	0,19
10	10,0	-0,5	0,94	0,20
11	11,0	-0,4	1,00	0,21
12	12,0	-0,3	1,21	0,22
13	13,0	-0,2	1,44	0,23
14	14,0	-0,1	1,69	0,24
15	15,0	0,0	1,96	0,25
16	16,0	0,0	1,99	0,26
17	17,0	0,1	2,56	0,27
18	18,0	0,2	2,89	0,28
19	19,0	0,3	3,24	0,29
20	20,0	0,4	3,81	0,30
21	21,0	0,5	4,00	0,31
22	22,0	0,6	5,02	0,32
23	23,0	0,7	4,84	0,33
24	24,0	0,8	5,29	0,34
25	25,0	0,9	5,76	0,35

Задача №2.

В результате эксперимента определены значения некоторой величины y – $y^{э}(x_i)$, соответствующие определенным значениям другой переменной x – x_i. При этом установлено, что между величинами y и x существует функциональная зависимость, причем вид функции $y^{т} = f(x_i) = ax_i^2 + bx_i + c$ известен. Требуется с помощью метода многомерной оптимизации определить такое значение параметров a,b,c этой функции, при которых сумма квадратов отклонений экспериментальных данных от расчетных значений будет минимальна:

$$U = \sum_{i-1}^{n} \left[y^{э}(x_i) - y^{т}(x_i) \right]^2 \to \min \quad . \tag{3.13}$$

После этого, для найденных значений коэффициентов a,b,c необходимо построить график функции $y^{т} = f(x)$ и отметить на нем экспериментальные точки. Значения параметров a,b,c следует искать с абсолютной погрешностью $\varepsilon = 0{,}01$.

Данную задачу можно представить следующим образом. Для объекта управления из каких-либо соображений выбрана структура математической модели. Требуется так подобрать параметры этой модели, чтобы ее выход был бы достаточно близок к выходу реального объекта. Мерой близости является целевая функция U. Сказанное иллюстрируется рис. 3.12.

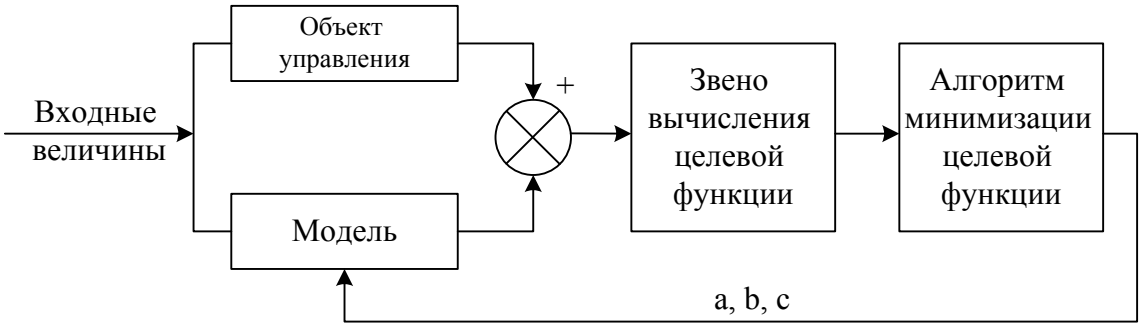


Рис. 3.12. Схема настройки модели объекта управления

Функциональная зависимость $\overline{Y} = f(x)$, определяющая с определенной достоверностью наличие статистической связи между переменными y и x, называется теоретической линией регрессии.

Метод определения коэффициентов линии регрессии, основанный на соблюдении условия (3.13), называется методом наименьших квадратов.

Этот метод расчета уравнения теоретической линии регрессии не единственный, но наиболее универсальный, физически понятный и часто применяемый в инженерной практике. Кроме этого метода, для определения уравнения $\overline{Y} = f(x)$ можно использовать интерполяционные многочлены Чебышева и интерполяционную формулу Лагранжа.

Таблица 3.2

Варианты №1-10

x_i	Значения $y_i = y(x_i)$									
	№ 1	№ 2	№ 3	№ 4	№ 5	№ 6	№ 7	№ 8	№ 9	№ 10
1	2,05	2,09	2,02	1,99	2,23	2,07	2,18	-0,10	-0,16	2,09
2	1,94	2,05	1,98	2,03	2,29	2,17	2,43	-0,21	0,01	2,31
3	1,92	2,19	1,67	2,20	2,27	2,21	2,40	0,01	0,10	2,72
4	1,87	2,18	1,65	2,39	2,62	2,31	2,43	0,05	0,16	2,77
5	1,77	2,17	1,57	2,19	2,72	2,10	2,65	-0,13	0,05	2,78
6	1,88	2,27	1,42	2,61	2,82	2,09	2,75	-0,23	0,35	2,97
7	1,71	2,58	1,37	2,35	3,13	2,12	2,67	-0,21	0,19	3,00
8	1,60	2,73	1,07	2,60	3,49	1,63	2,66	-0,43	0,50	3,51
9	1,56	2,82	0,85	2,55	3,82	1,78	2,63	-0,57	0,74	3,43
10	1,40	3,04	0,48	2,49	3,95	1,52	2,75	-0,44	1,03	3,58
11	1,50	3,03	0,35	2,50	4,22	1,16	2,41	-0,44	1,06	3,58
12	1,26	3,15	-0,30	2,52	4,48	1,07	2,24	-0,83	1,49	3,51
13	0,99	3,62	-0,61	2,44	5,06	0,85	2,12	-0,78	1,79	3,82
14	0,97	3,85	-1,20	2,35	5,50	0,56	1,74	-0,81	2,03	3,90
15	0,91	4,19	-1,39	2,26	5,68	0,10	1,57	-1,06	2,22	3,77
16	0,71	4,45	-1,76	2,19	6,19	-0,25	1,17	-1,41	2,50	3,81
17	0,43	4,89	-2,28	2,24	6,42	-0,65	0,96	-1,40	2,88	4,00
18	0,54	5,06	-2,81	2,34	7,04	-1,06	0,63	-1,70	3,21	3,97
19	0,19	5,63	-3,57	1,96	7,57	-1,66	0,25	-1,96	3,63	4,08
20	0,01	5,91	-4,06	2,19	8,10	-2,01	-0,01	-1,91	3,90	4,08

Таблица 3.3

Варианты №11-20

x_i	Значения $y_i = y(x_i)$									
	№ 11	№ 12	№ 13	№ 14	№ 15	№ 16	№ 17	№ 18	№ 19	№ 20
1	0,17	0,80	0,04	0,08	-0,02	0,14	-1,86	-1,65	-1,89	-1,84
2	0,07	0,29	0,47	0,14	0,44	0,23	-1,95	-2,00	-2,07	-1,98
3	0,17	0,52	0,78	0,37	0,51	0,44	-2,12	-1,87	-2,30	-1,72
4	0,05	0,77	1,01	0,36	0,67	0,54	-2,06	-1,89	-2,26	-1,58
5	0,12	0,93	1,19	0,44	0,69	0,72	-2,15	-1,75	-2,34	-1,59
6	0,00	1,20	1,60	0,48	1,04	0,76	-2,00	-1,59	-2,66	-1,59
7	0,01	1,20	1,93	0,27	1,14	0,37	-2,12	-1,44	-2,88	-1,58
8	-0,05	1,35	2,22	0,39	1,37	0,64	-2,31	-1,51	-2,85	-1,64
9	-0,21	1,39	2,50	0,50	1,77	0,57	-2,29	-1,00	-3,16	-1,55
10	-0,50	1,48	3,01	0,48	2,00	0,44	-2,57	-1,17	-3,49	-1,35
11	-0,50	1,52	3,22	0,69	2,12	0,41	-2,56	-0,87	-3,88	-1,33
12	-0,86	1,71	3,71	0,50	2,47	0,30	-2,86	-0,47	-4,22	-1,47
13	-1,24	1,72	4,23	0,31	2,90	-0,01	-2,85	-0,33	-4,45	-1,50
14	-1,47	1,87	4,78	0,37	3,50	-0,03	-3,03	-0,00	-4,99	-1,65
15	-1,79	1,86	5,27	0,43	3,99	-0,47	-3,25	0,34	-5,36	-1,62
16	-2,25	1,89	5,75	0,33	4,06	-0,68	-3,08	0,49	-5,71	-1,87
17	-2,55	2,04	6,16	0,31	4,54	-0,93	-3,29	0,81	-6,51	-1,61
18	-3,18	1,73	6,76	0,09	4,99	-1,28	-3,67	1,37	-6,76	-1,86
19	-3,60	2,04	7,30	0,08	5,36	-1,53	-3,70	1,72	-7,35	-1,84
20	-3,93	2,03	8,00	0,03	5,99	-1,93	-3,85	2,03	-8,02	-1,91

4. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Линейное программирование - это наука о методах исследования и отыскания наибольших и наименьших значений линейной функции, на неизвестные параметры которой наложены линейные ограничения.

Задачей линейного программирования (ЗЛП) называется задача отыскания экстремума (максимума или минимума) линейной функции от нескольких переменных при линейных ограничениях на эти переменные.

Таким образом, задачи линейного программирования относятся к задачам на условный экстремум функции.

ЗЛП является удобной математической моделью для большого числа экономических задач (планирование производства, расходование ресурсов, раскрой материалов, транспортные перевозки и т.д.).

Общая задача линейного программирования состоит в максимизации или минимизации линейной целевой функции:

$$L = f(x_1, x_2, ...x_n) = c_0 + c_1x_1 + c_2x_2 + ...c_nx_n \,. \qquad (4.1)$$

Эта функция задана в некоторой ограниченной области пространства – области допустимых решений ОДР, причем эта область Gn задается системой неравенств следующего вида:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \le b_1 \\ a_{12}x_1 + a_{22}x_2 + ... + a_{2n}x_n \le b_2 \\ ... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \le b_m \end{cases}, \qquad (4.2)$$

где n – количество переменных в целевой функции; m – число неравенств, ограничивающих целевую функцию.

Среди неотрицательных решений системы (4.2) нужно найти такое, при котором линейная целевая функция L достигает наибольшего или наименьшего значения.

Система неравенств (4.2) представляет собой многогранник допустимых решений – ОДР. ОДР не всегда бывает ограниченной. Она может быть неограниченной и даже пустой, когда система неравенств противоречива. Возможны также случаи лишних неравенств, входящих в систему (4.2): это плоскости, не имеющие общих точек с ОДР – такие неравенства отбрасываются при решении.

ОДР всегда должна быть выпуклой.

Прямая или плоскость, имеющая с ОДР по крайней мере одну общую точку, при том такую, что вся ОДР лежит по одну сторону от этой прямой или плоскости, называется опорной прямой или плоскостью по отношению к ОДР.

Геометрически задача линейного программирования представляет собой отыскание такой точки многогранника ОДР, координаты которой доставляют линейной функции минимальное значение, причем допустимыми решениями служат все точки многогранника решений.

4.1. Геометрический (графический) метод решения задачи линейного программирования

Графический метод основан на геометрической интерпретации задачи линейного программирования и применяется в основном при решении задач двумерного пространства и

только некоторых задач трехмерного пространства, так как довольно трудно построить многогранник решений, который образуется в результате пересечения полупространств. Задачу пространства размерности больше трех изобразить графически вообще невозможно.

Пусть задача линейного программирования задана в двумерном пространстве, т. е. ограничения содержат две переменные.

Если в ЗЛП ограничения заданы в виде неравенств с двумя переменными, она может быть решена графически.

Каждое неравенство системы ограничений и условие неотрицательности представляют собой полуплоскость. Пересечение полуплоскостей образует выпуклое многоугольное множество (многоугольник допустимых решений).

Целевая функция графически изображается множеством параллельных прямых, называемых линиями уровня, каждой из которых соответствует конкретное значение целевой функции.

Графический метод решения ЗЛП состоит из следующих шагов.

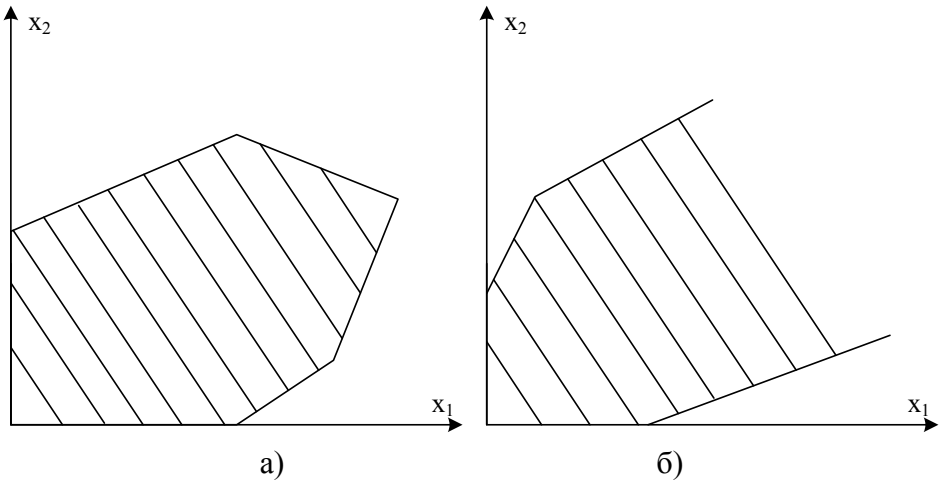
Шаг 1.

Сначала на координатной плоскости x1Ox2 строится допустимая многоугольная область (область допустимых решений, область определения), соответствующая ограничениям:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 \le b_1; \\ a_{12}x_1 + a_{22}x_2 \le b_2; \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 \le b_m; \\ x_1 \ge 0, \;\; x_2 \ge 0. \end{cases}$$

Не приводя строгих доказательств, укажем те случаи, которые тут могут получиться.

Основной случай – получающаяся область имеет вид ограниченного выпуклого многоугольника (рис. 4.1,а). Неосновной случай – получается неограниченный выпуклый многоугольник (рис. 4.1,б). Наконец, возможен случай, когда неравенства противоречат друг другу, и допустимая область вообще пуста.



а) б)

Рис. 4.1. Возможные виды области допустимых решений

Шаг 2.

Вернёмся теперь к исходной задаче линейного программирования. В ней, кроме

системы неравенств, есть еще целевая функция $L = f(x_1, x_2) = c_1 x_1 + c_2 x_2 \to \max$.

Рассмотрим прямую $L = c_1 x_1 + c_2 x_2$. Будем увеличивать L. Что будет происходить с нашей прямой?

Легко догадаться, что прямая будет двигаться параллельно самой себе в том направлении, которое дается вектором $\overline{(c_1 c_2)}$, так как это вектор нормали к нашей прямой и одновременно вектор градиента функции $f(x_1, x_2) = c_1 x_1 + c_2 x_2$ (рис. 4.2).

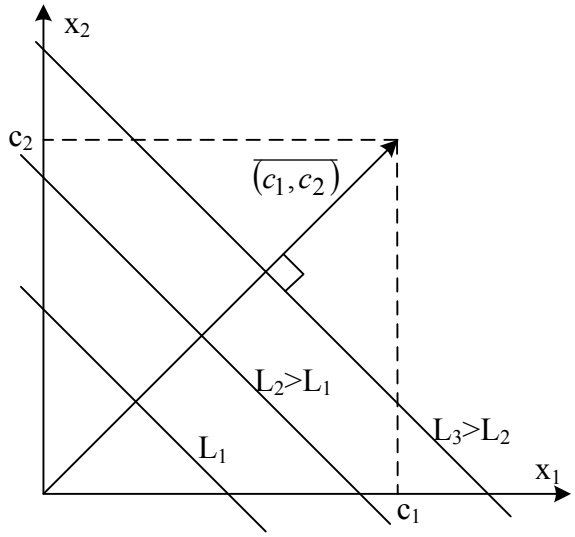


Рис. 4.2. Поведение целевой функции

Шаг 3.

Увеличивая L, мы начнем двигать нашу прямую и её пересечение с допустимой областью будет изменяться (см. рис. 4.2). В конце концов эта прямая выйдет на границу допустимой области, как правило, это будет одна из вершин многоугольника. Дальнейшее увеличение L приведёт к тому, что пересечение прямой $L = c_1 x_1 + c_2 x_2$ с допустимой областью будет пустым. Поэтому то положение прямой $L = c_1 x_1 + c_2 x_2$, при котором она вышла на граничную точку допустимой области, и даст решение задачи, а соответствующее значение L и будет оптимальным значением целевой функции (рис. 4.3).

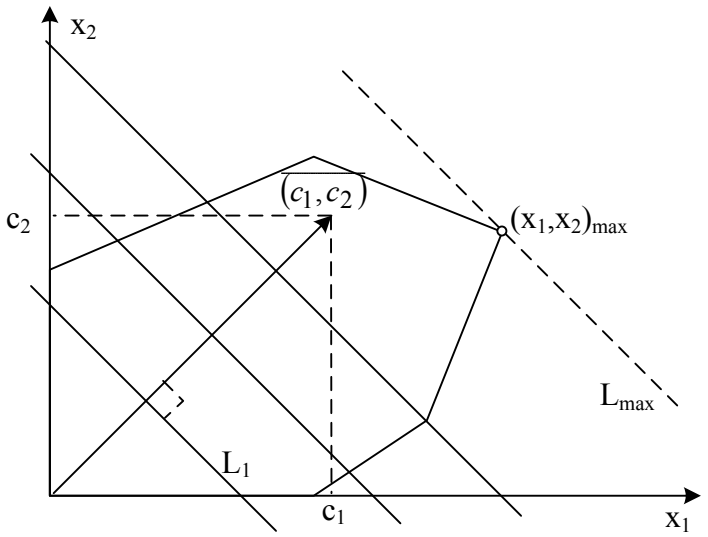


Рис. 4.3. Определение максимального значения целевой функции

Алгоритм решения ЗЛП геометрическим методом:

1. По неравенствам системы ограничений строится многоугольник допустимых решений.

2. Используя выражение для линейной целевой функции $L = f(x_1, x_2) = c_0 + c_1 x_1 + c_2 x_2$, строят вектор градиента функции, выходящий из начала координат: $\bar{c} = \{c_1, c_2\} = c_1 i + c_2 j$.

3. Проводят прямую, перпендикулярную вектору $\bar{c}$.

4. Перемещая эту прямую в направлении вектора $\bar{c}$, находят точки касания этой прямой многоугольника допустимых решений, в момент входа этой прямой в ОДР. В этих точках целевая функция принимает наименьшее значение из всех значений, принимаемых в ОДР.

5. Продолжая перемещать прямую в том же направлении вектора $\bar{c}$, находят точку выхода этой прямой из ОДР. В этой точке целевая функция будет принимать наибольшее значение из всех значений, принимаемых в ОДР.

Пример 1

$Z = 6x_1+2x_2 \rightarrow \max$

$$\begin{cases} 4x_1+5x_2 \le 61 \\ -3x_1+4x_2 \le 24 \\ 5x_1-3x_2 \le 30 \end{cases}$$

$x_1 \ge 0,\ x_2 \ge 0$

Построим область допустимых решений ЗЛП, представляющую собой множество точек, удовлетворяющих ограничениям задачи. Область допустимых решений – ограниченный многоугольник, выделенный внешней штриховкой.

Вектор-градиент целевой функции gradZ=c={6,2} определяет направление ее возрастания. Из рис. 4.4 видно, что целевая функция z принимает максимальное значение в точке пересечения прямых, задаваемых неравенствами (а) и (б):

$$\begin{cases} 4x_1+5x_2 = 61 \\ 5x_1-3x_2 = 30 \end{cases}$$

Решив эту системы уравнений, получим x1*=9, x2*=5. Максимальное значение функции Zmax=64.

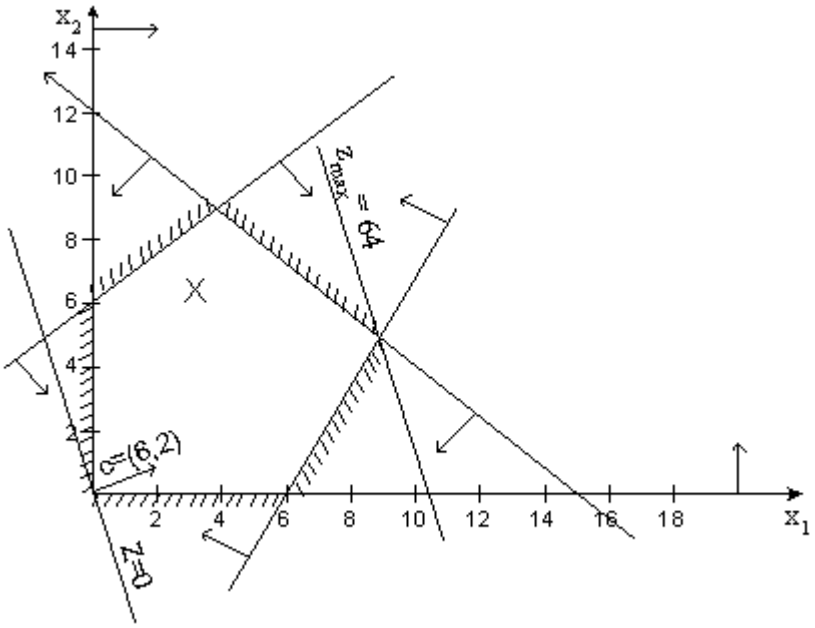


Рис. 4.4. Графическое решение примера 1

Пример 2

$Z = 8x_1+10x_2 \to \max$

$x_1+2x_2 \le 220$

$2x_1+x_2 \le 260$

$4x_1+5x_2 \le 640$

$x_1 \ge 0,\ x_2 \ge 0$

Построим область допустимых решений задачи.

Эта область предоставляет собой выпуклый многоугольник OM1M2M3M4 (рис. 4.5). Определим координаты вершин многоугольника, как точки пересечения соответствующих прямых.

Координаты вершины M_1 определим из решения системы:

$$\begin{cases} x_1 + 2x_2 = 220 \\ x_1 = 0 \end{cases}.$$

Решив эту систему, получим $M_1=\{0;110\}$.

Координаты вершины $M_2=\{60;80\}$ получим из системы: $\begin{cases} x_1 + 2x_2 = 220 \\ 4x_1 + 5x_2 = 640 \end{cases}$.

Координаты вершины $M_3=\{110;40\}$ получим из системы: $\begin{cases} 2x_1 + x_2 = 260 \\ 4x_1 + 5x_2 = 640 \end{cases}$.

Координаты вершины $M_4=\{130;0\}$ получим из системы: $\begin{cases} 2x_1 + x_2 = 260 \\ x_2 = 0 \end{cases}$.

(б

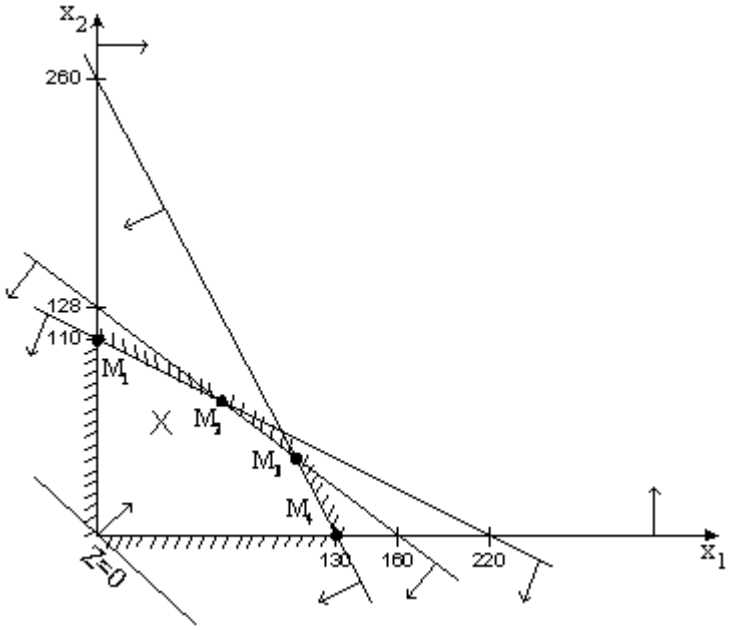


Рис. 4.5. Графическое решение примера 2

Из геометрической интерпретации ЗЛП видно, что, если задача имеет решение, то оно достигается в одной из вершин многоугольника допустимых решений. Поэтому для того, чтобы найти решение достаточно перебрать все вершины многоугольника допустимых решений. Вычислим значения целевой функции во всех вершинах этого многоугольника:

$z_0 = f(M_0) = 0$,

$z_1 = f(M_1) = 1100$,

$z_2 = f(M_2) = 1280$,

$z_3 = f(M_3) = 1280$,

$z_4 = f(M_4) = 1040$.

Отсюда видно, что целевая функция L достигает максимального значения в двух вершинах M2 и M3, т.е. в своем крайнем положении линия уровня $Z = 8x1+10x2=const=1280$ содержит вершины M2, M3 и, следовательно, все ребро (M2,M3).

Таким образом, решений бесконечно много – все точки ребра (M2,M3). При этом максимальное значение целевой функции Z равно 1280.

Пример 3

$Z=x_1+x_2 \rightarrow \max$

$x_1+x_2 \le -1$

$x_1 \ge 0, x_2 \ge 0$

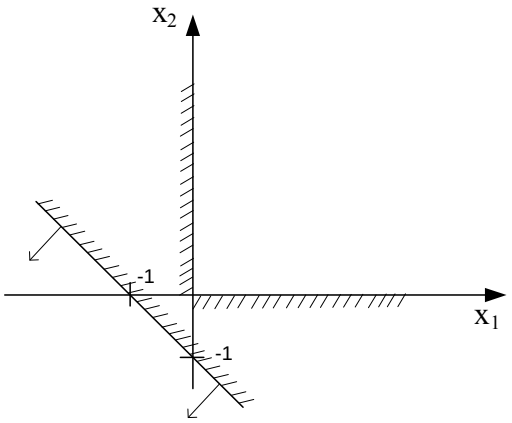


Рис. 4.6. Графическое решение примера 3

Построим область допустимых решений задачи. Из рис..4.6 видно, что нет точек, одновременно удовлетворяющих всем ограничениям, т.е. область допустимых решений пуста. Таким образом, задача не имеет решения.

Пример 4

$Z=x_1+x_2 \rightarrow \max$

$x_1-x_2 \le 1$

$x_1 \ge 0,\ x_2 \ge 0$

Построим область допустимых решений задачи. Из рис. 4.7 видно, что эта область неограниченная. Целевая функция Z на этой области допустимых решений не ограничена сверху, так при сдвиге вправо целевая функция возрастает, а сдвигать вправо можно до бесконечности. Следовательно, не существует точки, в которой функция Z достигает максимума, т.е. задача не имеет решения.

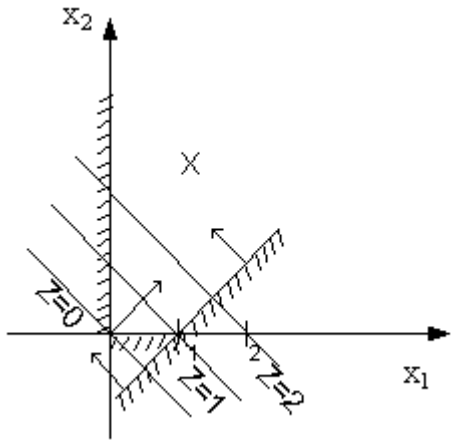


Рис. 4.7. Графическое решение примера 4

Неограниченность области допустимых решений в сочетании с неограниченностью целевой функции и приводит к тому, что задача не имеет решения. Неограниченность только области допустимых решений не является препятствием для существования решения.

4.2. Симплекс-метод решения задачи линейного программирования

Двумерные задачи линейного программирования решаются графически. Для случая N=3 можно рассмотреть трехмерное пространство и целевая функция будет достигать своё оптимальное значение в одной из вершин многогранника.

В общем виде, когда в задаче участвуют N-неизвестных, можно сказать, что область допустимых решений, задаваемая системой ограничивающих условий, представляется выпуклым многогранником в n-мерном пространстве и оптимальное значение целевой функции достигается в одной или нескольких вершинах. Решить данные задачи графически, когда количество переменных более 3 весьма затруднительно. Существует универсальный способ решения задач линейного программирования, называемый симплекс-методом.

Симплекс-метод является основным в линейном программировании. Решение задачи начинается с рассмотрений одной из вершин многогранника условий. Если исследуемая вершина не соответствует максимуму (минимуму), то переходят к соседней, увеличивая значение функции цели при решении задачи на максимум и уменьшая при решении задачи на минимум. Таким образом, переход от одной вершины к другой улучшает значение функции цели. Так как число вершин многогранника ограничено, то за конечное число шагов гарантируется нахождение оптимального значения или установление того факта, что задача неразрешима.

Этот метод является универсальным, применимым к любой задаче линейного программирования в канонической форме. Система ограничений здесь - система линейных уравнений, в которой количество неизвестных больше количества уравнений. Если ранг системы равен r, то мы можем выбрать r неизвестных, которые выразим через остальные неизвестные. Для определенности предположим, что выбраны первые, идущие подряд, неизвестные x1, x2, ..., xr. Тогда наша система уравнений может быть записана как:

$$\begin{cases} x_1 = b_1 + a_{1,r+1}x_{r+1} + ... + a_{1n}x_n, \\ x_2 = b_2 + a_{2,r+1}x_{r+1}. + .. + a_{2n}x_n, \\ ... \\ x_r = b_r + a_{r,r+1}x_{r+1} + ... + a_{mn}x_n. \end{cases} \quad (4.3)$$

К такому виду можно привести любую совместную систему, например, методом Гаусса. Правда, не всегда можно выражать через остальные первые r неизвестных (мы это сделали для определенности записи). Однако такие r неизвестных обязательно найдутся. Эти неизвестные (переменные) называются базисными, остальные свободными.

Придавая определенные значения свободным переменным и вычисляя значения базисных (выраженных через свободные), мы будем получать различные решения нашей системы ограничений. Таким образом, можно получить любое ее решение. Нас будут интересовать особые решения, получаемые в случае, когда свободные переменные равны нулю. Такие решения называются базисными, их столько же, сколько различных базисных видов у данной системы ограничений. Базисное решение называется допустимым базисным решением или опорным решением, если в нем значения переменных неотрицательны. Если в качестве базисных взяты переменные x1, x2, ..., xr, то решение {b1, b2,..., br, 0, ..., 0} будет опорным при условии, что b1, b2,..., br $\geq$ 0.

Симплекс-метод основан на теореме, которая называется фундаментальной теоремой симплекс-метода. Среди оптимальных планов задачи линейного программирования в канонической форме обязательно есть опорное решение ее системы ограничений. Если оптимальный план задачи единственен, то он совпадает с некоторым опорным решением. Различных опорных решений системы ограничений конечное число. Поэтому решение задачи в канонической форме можно было бы искать перебором опорных решений и выбором среди них того, для которого значение целевой функции Z самое большое. Но, во-

первых, все опорные решения неизвестны и их нужно находить, а, во-вторых, в реальных задачах этих решений очень много и прямой перебор вряд ли возможен. Симплекс-метод представляет собой некоторую процедуру направленного перебора опорных решений. Исходя из некоторого, найденного заранее опорного решения по определенному алгоритму симплекс-метода мы подсчитываем новое опорное решение, на котором значение целевой функции Z не меньше, чем на старом. После ряда шагов мы приходим к опорному решению, которое является оптимальным планом.

Итак, симплексный метод вносит определенный порядок как при нахождении первого (исходного) базисного решения, так и при переходе к другим базисным решениям. Его идея состоит в следующем.

Имея систему ограничений, приведенную к общему виду, то есть к системе m линейных уравнений с n переменными ($m < n$), находят любое базисное решение этой системы, заботясь только о том, чтобы найти его как можно проще.

Если первое же найденное базисное решение оказалось допустимым, то проверяют его на оптимальность. Если оно не оптимально, то, осуществляется переход к другому, обязательно допустимому базисному решению.

Симплексный метод гарантирует, что при этом новом решении линейная форма, если и не достигнет оптимума, то приблизится к нему. С новым допустимым базисным решением поступают так же, пока не находят решение, которое является оптимальным.

Если первое найденное базисное решение окажется недопустимым, то с помощью симплексного метода осуществляется переход к другим базисным решениям, которые приближают нас к области допустимых решений, пока на каком-то шаге решения либо базисное решение окажется допустимым и к нему применяют алгоритм симплексного метода, либо мы убеждаемся в противоречивости системы ограничений.

Таким образом, применение симплексного метода распадается на два этапа: нахождение допустимого базисного решения системы ограничений или установление факта ее несовместности; нахождение оптимального решения. При этом каждый этап может включать несколько шагов, соответствующих тому или иному базисному решению. Но так как число базисных решений всегда ограниченно, то ограниченно и число шагов симплексного метода.

Приведенная схема симплексного метода явно выражает его алгоритмический характер (характер четкого предписания о выполнении последовательных операций), что позволяет успешно программировать и реализовать этот метод на ЭВМ. Задачи же с небольшим числом переменных и ограничений могут быть решены симплексным методом вручную.

Не останавливаясь подробнее на сути алгоритма, опишем его вычислительную сторону. Вычисления по симплекс-методу организуются в виде симплекс-таблиц, которые являются сокращенной записью задачи линейного программирования в канонической форме. Перед составлением симплекс-таблицы задача должна быть преобразована, система ограничений приведена к допустимому базисному виду, с помощью которого из целевой функции должны быть исключены базисные переменные. Вопрос об этих предварительных преобразованиях мы рассмотрим ниже. Сейчас же будем считать, что они уже выполнены и задача имеет вид (4.4).

Здесь для определенности записи считается, что в качестве базисных переменных можно взять переменные X1, X2, ..., Xr и что при этом b1, b2,..., $br \geq 0$ (соответствующее базисное решение является опорным).

$$Z = c_0 + c_{r+1}x_{r+1} + ... + c_n x_n \to \max, \min$$
$$\begin{cases} x_1 = b_1 + a_{1,r+1}x_{r+1} + ... + a_{1n}x_n, \\ x_2 = b_2 + a_{2,r+1}x_{r+1}. + .. + a_{2n}x_n, \\ ... \\ x_r = b_r + a_{r,r+1}x_{r+1} + ... + a_{mn}x_n. \end{cases} \quad (4.4)$$
$$x_i \geq 0, i = 1,2,...,n.$$

Для составления симплекс-таблицы во всех равенствах в условии задачи члены, содержащие переменные, переносятся в левую часть, свободные оставляются справа, т.е. задача записывается в виде системы равенств:

$$\begin{cases} x_1 - a_{1,r+1}x_{r+1} - ... - a_{1n}x_n = b_1, \\ x_2 - a_{2,r+1}x_{r+1} - ... - a_{2n}x_n = b_2, \\ ... \\ x_r - a_{r,r+1}x_{r+1} - ... - a_{mn}x_n = b_r, \\ Z - c_{r+1}x_{r+1} - ... - c_n x_n = c_0. \end{cases} \quad (4.5)$$

Далее эта система оформляется в виде симплекс-таблиц.

Таблица 4.1

Исходная симплекс-таблица в общем виде

Баз. перем.	Своб. члены	x_{r+1}	x_{r+2}	...	x_n
x_1	b_1	$-a_{1,r+1}$	$-a_{1,r+2}$	...	$-a_{1n}$
x_2	b_2	$-a_{2,r+1}$	$-a_{2,r+2}$	...	$-a_{2n}$
...	...	...	...	...	...
x_r	b_r	$-a_{r,r+1}$	$-a_{r,r+2}$	...	$-a_m$
Z	c_0	$-c_{r+1}$	$-c_{r+2}$	...	$-c_n$

4.2.1. Алгоритм симплекс-метода

Шаг 1. Составление исходной симплекс-таблицы (см. табл. 4.1). Если исходная задача линейного программирования задана в стандартной форме, то необходимо привести ее к каноническому виду путем введения в каждое из ограничений-неравенств дополнительной неотрицательной переменной.

Примечание: Перед приведением задачи к каноническому виду необходимо учесть следующее. В случае, если в исходной задаче необходимо найти минимум, знаки коэффициентов целевой функции Z меняются на противоположные $c = -c$. Знаки коэффициентов ограничивающих условий со знаком "≥" также меняются на противоположные. В случае, если условие содержит знак "≤", коэффициенты запишутся без

изменений.

Шаг 2. Нахождение базисного решения.

Согласно определению базисного решения, свободные переменные равны нулю, а значения базисных переменных – соответствующим значениям свободных чисел, т.е.:

$$X = (b_1, b_2, b_3, ...b_r, 0,0,0,...0).$$

Шаг 3. Проверка совместности системы ограничений задачи линейного программирования.

Если в исходной симплекс-таблице имеется строка с отрицательным свободным числом, в которой нет ни одного отрицательного элемента (т.е. отрицательного коэффициента при свободной переменной), то согласно признаку несовместности система ограничений данной задачи не совместна (строка целевой функции при рассмотрении данного признака не учитывается).

Другими словами, система ограничения несовместна, если в любой строке (кроме строки целевой функции), имеющей отрицательное свободное число c_0, нет ни одного отрицательного элемента.

Если система ограничения несовместна, то рассматриваемая задача линейного программирования не имеет решения.

Шаг 4. Проверка ограниченности целевой функции.

Если в исходной симплекс-таблице имеется колонка с отрицательным элементом в строке целевой функции (кроме колонки свободных чисел), в которой не было бы хотя бы одного положительного элемента), то область допустимых решений целевой функции неограниченна, т.е. рассматриваемая задача линейного программирования имеет бесконечно много решений.

Шаг 5. Проверка допустимости найденного базисного решения.

Если найденное базисное решение (см. шаг 2) не содержит отрицательных компонент (при поиске максимума целевой функции), то оно является допустимым. Если же отрицательные компоненты имеются, то необходимо преобразовать симплекс-таблицу таким образом, чтобы базисное решение стало допустимым. Для этого необходимо перейти на шаг 8, пункт а).

Шаг 6. Проверка оптимальности найденного базисного решения.

Найденное базисное решение является оптимальным, если согласно признаку оптимальности в строке целевой функции нет отрицательных элементов (при поиске максимума целевой функции). При этом свободное число данной строки при рассмотрении данного признака не учитывается. Если же отрицательные компоненты имеются, то необходимо преобразовать симплекс-таблицу таким образом, чтобы базисное решение стало оптимальным. Для этого необходимо перейти на шаг 8, пункт б).

Шаг 7. Если найденное базисное решение оптимальное, то проверяют наличие альтернативного решения (чаще всего этот этап опускают и считают, что решение ЗЛП найдено).

Шаг 8. Нахождение разрешающего элемента.

а) Если получено недопустимое базисное решение, то есть в столбце свободных членов имеются отрицательные элементы, то выбираем среди них максимальный по модулю - он задает разрешающую строку k. В этой строке также находим максимальный по модулю отрицательный элемент ak,l – он задает разрешающий столбец (назовем его №1) и является *разрешающим элементом*. Переменная, соответствующая разрешающей строке, исключается из базиса (становится свободной), переменная, соответствующая разрешающему столбцу, включается в базис (становится базисной). То есть эти переменные в новой симплекс-таблице нужно поменять местами. Далее необходимо пересчитать все

элементы новой симплекс-таблицы, т.е. осуществить преобразование симплекс-таблицы согласно шагу 9.

Если после перерасчета в столбце свободных членов остались отрицательные элементы, то необходимо снова найти разрешающий элемент согласно вышеперечисленному алгоритму. Если же допустимое базисное решение найдено, то переходим к проверке его на оптимальность (шаг 6).

б) Если получено неоптимальное базисное решение, то есть в в строке Z есть отрицательные элементы, то решение требует улучшения. Выбираем среди отрицательных элементов строки Z максимальный по модулю (исключая значение b0):

$$a_{0,l}=\min\{a_{0,i}\}.$$

l – столбец, в котором находится максимальный по модулю отрицательный элемент, будет разрешающим. Для того, чтобы найти разрешающую строку, находим отношение соответствующего свободного члена и элемента из разрешающего столбца, при условии, что они неотрицательны:

$$b_k/a_{k,l}=\min\{b_i/a_{i,l}\} \text{ при } a_{i,l}>0,\ b_i>0.$$

k – строка, для которой это отношение минимально – разрешающая. Элемент ak,l – разрешающий (ведущий). Переменная, соответствующая разрешающей строке (xk), исключается из базиса, переменная, соответствующая разрешающему столбцу (xl), включается в базис. То есть эти переменные в новой симплекс-таблице нужно поменять местами. Далее необходимо пересчитать все элементы новой симплекс-таблицы, т.е. осуществить преобразование симплекс-таблицы согласно шагу 9.

Шаг 9. Преобразование симплекс-таблицы.

При составлении новой симплекс-таблицы в ней происходят следующие изменения:

- вместо базисной переменной xk записываем xl; вместо небазисной переменной xl записываем xk;
- разрешающий элемент заменяется на обратную величину: ak,l'= 1/ak,l;
- все элементы разрешающего столбца (кроме ak,l) умножаются на -1/ak,l;
- все элементы разрешающей строки (кроме ak,l) умножаются на 1/ak,l;
- оставшиеся элементы симплекс-таблицы преобразуются по формуле (4.6), согласно «правилу прямоугольника» (рис. 4.8):

$$a_{i,j}'= a_{i,j} – (a_{i,l} * a_{k,j})/ a_{k,l}. \qquad (4.6)$$

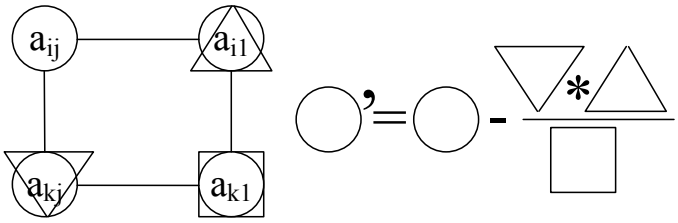


Рис. 4.8. Пояснение «правила прямоугольника»

Преобразуемый элемент **ai,j** и соответствующие ему три сомножителя являются вершинами «прямоугольника».

В результате получают новую симплекс-таблицу, отвечающую новому базисному решению.

Теперь следует просмотреть строку целевой функции (индексную), если в ней нет отрицательных значений (в задачи на нахождение максимального значения), либо положительных (в задачи на нахождение минимального значения) кроме стоящего на месте b0 (свободного столбца), то значит, что оптимальное решение получено. В противном случае, переходим к новой симплекс-таблице по выше описанному алгоритму.

Блок-схема алгоритма реализации симплекс-метода представлена на рис.4.9.

4.2.2. Пример решения ЗЛП симплекс-методом

Решить следующую задачу линейного программирования симплекс-методом:

$$\begin{cases} x_1 + 3x_2 \le 18; \\ 2x_1 + x_2 \le 16; \\ x_1 \le 5; \\ 3x_1 \le 21; \\ x_1 \ge 0; \\ x_2 \ge 0. \end{cases}$$

$F(x_1, x_2) = 2x_1 + 3x_2 \to \max$.

Шаг 1. Составление исходной симплекс-таблицы.

Исходная задача линейного программирования задана в стандартной форме. Приведем ее к каноническому виду путем введения в каждое из ограничений-неравенств дополнительной неотрицательной переменной, т.е.:

$$\begin{cases} x_1 + 3x_2 + x_3 = 18; \\ 2x_1 + x_2 + x_4 = 16; \\ x_1 + x_5 = 5; \\ 3x_1 + x_6 = 21; \\ x_j \ge 0 \left(\forall j = \overline{1,6}\right). \end{cases}$$

В полученной системе уравнений примем в качестве разрешенных (базисных) переменные x_3, x_4, x_5, x_6, тогда свободными переменными будут x_1, x_2. Выразим базисные переменные через свободные:

$$\begin{cases} x_3 = 18 - (x_1 + 3x_2); \\ x_4 = 16 - (2x_1 + x_2); \\ x_5 = 5 - (x_1); \\ x_6 = 21 - (3x_1); \\ x_j \ge 0 \left(\forall j = \overline{1,6}\right). \end{cases}$$

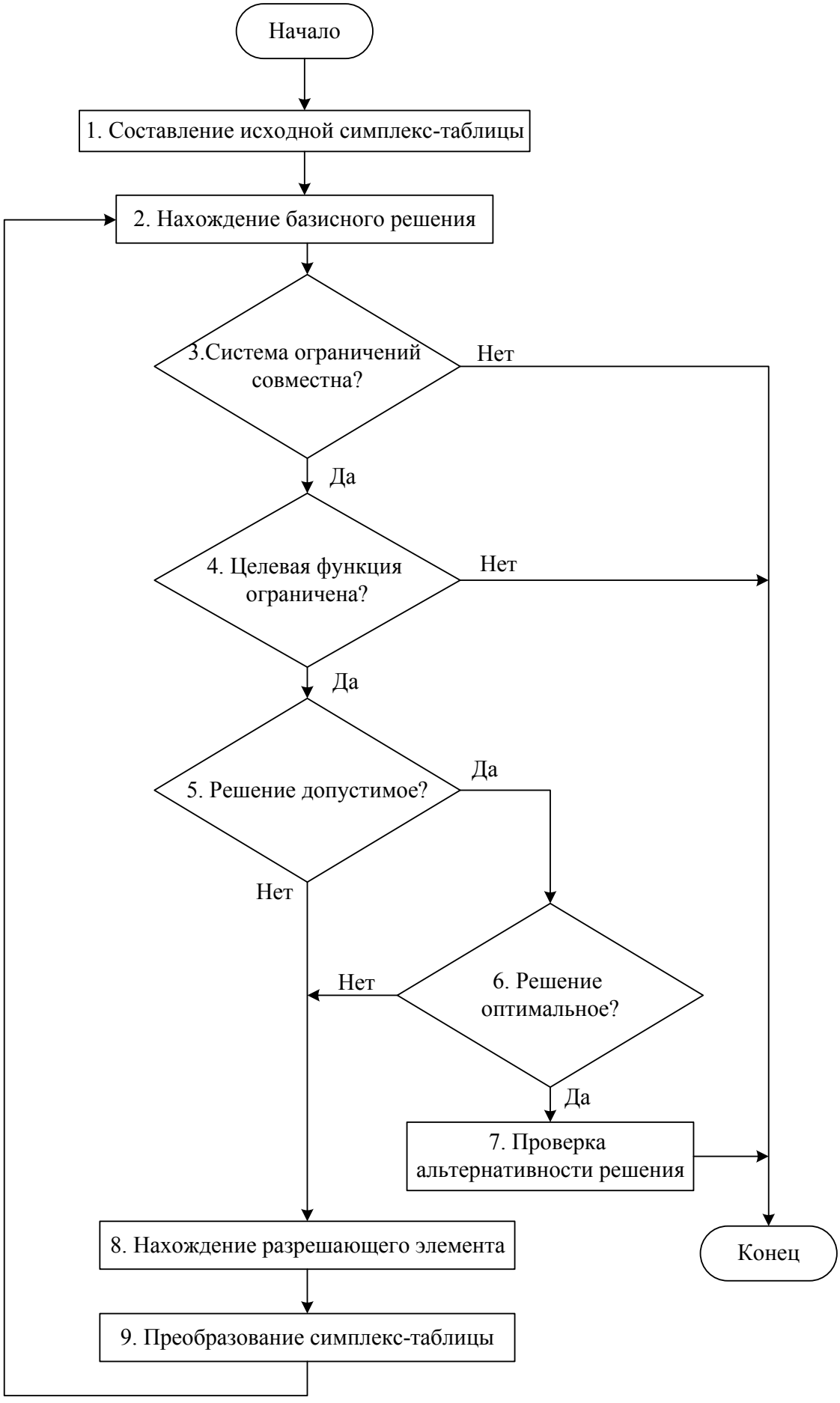


Рис. 4.9. Блок-схема алгоритма реализации симплекс-метода

Приведем целевую функцию к следующему виду:

$F(x_1, x_2) = 0 - (-2x_1 - 3x_2) \to \max$.

На основе полученной задачи сформируем исходную симплекс-таблицу (СП – свободные переменные, БП – базисные переменные):

Таблица 4.2

Исходная симплекс-таблица (1 цикл)

СП БП	b_i	x_1	x_2	Оценочные отношения
x_3	18	1	3	
x_4	16	2	1	
x_5	5	0	1	
x_6	21	3	0	
$F_{\max}$	0	–2	–3	

Шаг 2. Нахождение базисного решения.

Согласно определению базисного решения, свободные переменные равны нулю, а значения базисных переменных – соответствующим значениям свободных чисел, т.е.:

$X = (x_1, x_2, x_3, x_4, x_5, x_6) = (0,0,18,16,5,21)$.

Шаг 3. Проверка совместности системы ограничений задачи линейного программирования.

На данной итерации признак несовместности системы ограничений не выявлен (т.е. нет строки с отрицательным свободным числом (кроме строки целевой функции), в которой не было бы хотя бы одного отрицательного элемента (т.е. отрицательного коэффициента при свободной переменной)).

Шаг 4. Проверка ограниченности целевой функции.

На данной итерации признак неограниченности целевой функции не выявлен (т.е. нет колонки с отрицательным элементом в строке целевой функции (кроме колонки свободных чисел), в которой не было бы хотя бы одного положительного элемента).

Шаг 5. Проверка допустимости найденного базисного решения.

Так как найденное базисное решение не содержит отрицательных компонент, то оно является допустимым.

Шаг 6. Проверка оптимальности найденного базисного решения.

Найденное базисное решение не является оптимальным, так как согласно признаку оптимальности в строке целевой функции не должно быть отрицательных элементов (свободное число данной строки при рассмотрении данного признака не учитывается). Следовательно, согласно алгоритму симплекс-метода переходим к 8 шагу.

Шаг 8. Нахождение разрешающего элемента.

8.1. Определение разрешающего столбца.

Так как найденное базисное решение допустимое, то поиск разрешающего столбца будем производить по следующей схеме: определяем столбцы с отрицательными элементами в строке целевой функции (кроме столбца свободных чисел). Согласно таблице 4.2, таких

столбцов два: столбец «x1» и столбец «x2». Из таких столбцов выбирается тот, который содержит наименьший элемент в строке целевой функции. Он и будет разрешающим. Столбец «x2» содержит наименьший элемент (–3) в сравнении со столбцом «x1». Следовательно, его принимаем в качестве разрешающего.

8.2. Определение разрешающей строки.

Для определения разрешающей строки находим положительные оценочные отношения свободных чисел к элементам разрешающего столбца. Строка, которой соответствует наименьшее положительное оценочное отношение, принимается в качестве разрешенной.

Таблица 4.3

Определение разрешающего элемента (1 цикл)

СП БП	b_i	x_1	x_2	Оценочные отношения
x_3	18	1	3	18/3=6
x_4	16	2	1	16/1=16
x_5	5	0	1	5/1=5 - min
x_6	21	3	0	-
$F_{\max}$	0	–2	–3	-

В таблице 4.3 наименьшее положительное оценочное отношение соответствует строке «x_5», следовательно, она будет разрешающей. Элемент, расположенный на пересечение разрешающей колонки и разрешающей строки, принимается в качестве разрешающего. В нашем примере – это элемент $a_{33}=1$, который расположен на пересечении строки «x_5» и столбца «x_2».

Шаг 9. Преобразование симплекс-таблицы.

Разрешающий элемент показывает одну базисную и одну свободную переменные, которые необходимо поменять местами в симплекс-таблице, для перехода к новому «улучшенному» базисному решению. В данном случае это переменные x_5 и x_2, в новой симплекс-таблице (таблице 4.4) их меняем местами.

9.1. Преобразование разрешающего элемента.

Разрешающий элемент таблицы 4.3 преобразовывается следующим образом:

$$(a_{33})^{-1}=\left(1^{-1}\right)=1.$$

Полученный результат вписываем в аналогичную клетку таблицы 4.4.

9.2. Преобразование разрешающей строки.

Элементы разрешающей строки таблицы 4.3 делим на разрешающий элемент данной симплекс-таблицы, результаты вписываются в аналогичные ячейки новой симплекс-таблицы (таблицы 4.4). Преобразования элементов разрешающей строки приведены в таблице 4.4.

9.3. Преобразование разрешающей колонки.

Элементы разрешающей колонки таблицы 4.3 делим на разрешающий элемент данной симплекс-таблицы, а результат берется с обратным знаком. Полученные результаты

вписываются в аналогичные ячейки новой симплекс-таблицы (таблицы 4.4). Преобразования элементов разрешающей колонки приведены в таблице 4.4.

9.4. Преобразование остальных элементов симплекс-таблицы.

Преобразование остальных элементов симплекс-таблицы (т.е. элементов не расположенных в разрешающей строке и разрешающей колонке) осуществляется по правилу «прямоугольника».

К примеру, рассмотрим преобразование элемента, расположенного на пересечении строки «x_3» и колонки «b_i», условно обозначим его «x_3b_i». В таблице 4.3 мысленно вычерчиваем прямоугольник, одна вершина которого располагается в клетке, значение которой преобразуем (т.е. в клетке «x_3b_i»), а другая (диагональная вершина) – в клетке с разрешающим элементом. Две другие вершины (второй диагонали) определяются однозначно. Тогда преобразованное значение клетки «x_3b_i»будет равно прежнему значению данной клетки минус дробь, в знаменателе которой разрешающий элемент (из таблицы 4.3), а в числителе произведение двух других неиспользованных вершин, т.е.:

«x_3b_i»: $18-\frac{5*3}{1}=3$.

Аналогично преобразуются значения других клеток:

«x_3x_1»: $1-\frac{0*3}{1}=1$; «x_4b_i»: $16-\frac{5*1}{1}=11$;

«x_4x_1»: $2-\frac{0*1}{1}=2$; «x_6b_i»: $21-\frac{5*0}{1}=21$;

«x_6x_1»: $3-\frac{0*0}{1}=3$; «$F_{max}b_i$»: $0-\frac{5*(-3)}{1}=15$;

«$F_{max}x_1$»: $-2-\frac{0*(-3)}{1}=-2$.

В результате данных преобразований получили новую симплекс-таблицу (таблица 4.4).

Начинаем второй цикл вычислений.

Шаг 2. Нахождение базисного решения.

В результате проведенных симплекс-преобразований получили новое базисное решение (таблица 4.4):

$X=(x_1,x_2,x_3,x_4,x_5,x_6)=(0{,}5{,}3{,}11{,}0{,}21)$.

Как видно, при данном базисном решении значение целевой функции F_{max} =15, что больше, чем при предыдущем базисном решении.

Таблица 4.4

Преобразованная симплекс-таблица (2 цикл)

СП БП	b_i	x_1	x_5	Оценочные отношения
x_3	$18-\frac{5*3}{1}=3$	$1-\frac{0*3}{1}=1$	–(3/1)= –3	
x_4	$16-\frac{5*1}{1}=11$	$2-\frac{0*1}{1}=2$	–(1/1)= –1	

Окончание таблицы 4.4

x_2	5/1=5	0/1=0	$(1)^{-1}= 1$	
x_6	$21-\frac{5*0}{1}=21$	$3-\frac{0*0}{1}=3$	–(0/1)= 0	
$F_{\max}$	$0-\frac{5*(-3)}{1}=15$	$-2-\frac{0*(-3)}{1}=-2$	–(–3/1)= 3	

Шаг 3. Проверка совместности системы ограничений задачи линейного программирования.

Не совместность системы ограничений в таблице 4.4 не выявлена.

Шаг 4. Проверка ограниченности целевой функции.

Неограниченность целевой функции в таблице 4.4 не выявлена.

Шаг 5. Проверка допустимости найденного базисного решения.

Найденное базисное решение допустимое, так как не содержит отрицательных компонент.

Шаг 6. Проверка оптимальности найденного базисного решения.

Найденное базисное решение не оптимальное, так как в строке целевой функции симплекс-таблицы (таблица 4.4) содержится отрицательный элемент: –2 (свободное число данной строки при рассмотрении данного признака не учитывается). Следовательно, переходим к 8 шагу.

Шаг 8. Нахождение разрешающего элемента.

8.1. Определение разрешающего столбца.

Найденное базисное решение допустимое, определяем столбцы с отрицательными элементами в строке целевой функции (кроме столбца свободных чисел). Согласно таблице 4.4, таким столбцом является только один столбец: «x_1». Следовательно, его принимаем в качестве разрешающего.

8.2. Определение разрешающей строки.

Согласно полученным значениям положительных оценочных отношений в таблице 4.5, минимальным является отношение, соответствующее строке «x_3». Следовательно, ее принимаем в качестве разрешающей.

Таблица 4.5

Определение разрешающего элемента (2 цикл)

СП БП	b_i	x_1	x_5	Оценочные отношения
x_3	3	1	–3	3/1=3 – min
x_4	11	2	–1	11/2=5,5
x_2	5	0	1	–
x_6	21	3	0	21/3=7
$F_{\max}$	15	–2	3	–

Шаг 9. Преобразование симплекс-таблицы.

Преобразования симплекс-таблицы (таблицы 4.5) выполняются аналогично, как и в предыдущей итерации. Результаты преобразований элементов симплекс-таблицы приведены в таблице 4.6.

Таблица 4.6

Преобразованная симплекс-таблица (3 цикл)

СП БП	b_i	x_3	x_5	Оценочные отношения
x_1	3/1=3	$(1)^{-1}= 1$	–3/1= –3	
x_4	$11-\frac{3*2}{1}=5$	–(2/1)= –2	$-1-\frac{2*(-3)}{1}=5$	
x_2	$5-\frac{3*0}{1}=5$	–(0/1)= 0	$1-\frac{0*(-3)}{1}=1$	
x_6	$21-\frac{3*3}{1}=12$	–(3/1)= –3	$0-\frac{3*(-3)}{1}=9$	
$F_{\max}$	$15-\frac{3*(-2)}{1}=21$	–(–2/1)= 2	$3-\frac{-2*(-3)}{1}=-3$	

Начинаем третий цикл вычислений.

Шаг 2. Нахождение базисного решения.

В результате проведенных симплекс-преобразований получили новое базисное решение (таблица 4.6):

$X=(x_1,x_2,x_3,x_4,x_5,x_6)=(3{,}5,0{,}5,0,12)$.

Как видно, при данном базисном решении значение целевой функции $F_{\max}$ =21, что больше, чем при предыдущем базисном решении.

Шаг 3. Проверка совместности системы ограничений задачи линейного программирования.

Не совместность системы ограничений в таблице 4.6 не выявлена.

Шаг 4. Проверка ограниченности целевой функции.

Неограниченность целевой функции в таблице 4.6 не выявлена.

Шаг 5. Проверка допустимости найденного базисного решения.

Найденное базисное решение допустимое, так как не содержит отрицательных компонент.

Шаг 6. Проверка оптимальности найденного базисного решения.

Найденное базисное решение не оптимальное, так как в строке целевой функции симплекс-таблицы (таблица 4.6) содержится отрицательный элемент: –3 (свободное число данной строки при рассмотрении данного признака не учитывается). Следовательно, переходим к 8 шагу.

Шаг 8. Нахождение разрешающего элемента.

Найденное базисное решение допустимое, определяем столбцы с отрицательными элементами в строке целевой функции (кроме столбца свободных чисел). Согласно таблице 4.6, таким столбцом является только один столбец: «x_5». Следовательно, его принимаем в качестве разрешающего.

8.2. Определение разрешающей строки.

Согласно полученным значениям положительных оценочных отношений в таблице 4.7, минимальным является отношение, соответствующее строке «x_4». Следовательно, ее принимаем в качестве разрешающей.

Таблица 4.7

Определение разрешающего элемента (3 цикл)

СП БП	b_i	x_3	x_5	Оценочные отношения
x_1	3	1	–3	–
x_4	5	–2	5	5/5=1 – min
x_2	5	0	1	5/1=5
x_6	12	–3	9	12/9=1⅓
$F_{\max}$	21	2	–3	

Шаг 9. Преобразование симплекс-таблицы.

Преобразования симплекс-таблицы (таблицы 4.7) выполняются аналогично, как и в предыдущей итерации. Результаты преобразований элементов симплекс-таблицы приведены в таблице 4.8.

Таблица 4.8

Преобразованная симплекс-таблица (4 цикл)

СП БП	b_i	x_3	x_4	Оценочные отношения
x_1	$3-\frac{5*(-3)}{5}=6$	$1-\frac{-2*(-3)}{5}=-\frac{1}{5}$	–(–3/5)= 3/5	
x_5	5/5=1	–2/5	$(5)^{-1}$= 1/5	
x_2	$5-\frac{5*1}{5}=4$	$0-\frac{-2*1}{5}=\frac{2}{5}$	–(1/5)= –1/5	
x_6	$12-\frac{5*9}{5}=3$	$-3-\frac{-2*9}{5}=\frac{3}{5}$	–(9/5)= –9/5	
$F_{\max}$	$21-\frac{5*(-3)}{5}=24$	$2-\frac{-2*(-3)}{5}=\frac{4}{5}$	–(–3/5)= 3/5	

Начинаем четвертый цикл вычислений.

Шаг 2. Нахождение базисного решения.

В результате проведенных симплекс-преобразований получили новое базисное решение (таблица 4.8):

$X = (x_1, x_2, x_3, x_4, x_5, x_6) = (6,4,0,0,1,3)$.

Шаг 3. Проверка совместности системы ограничений задачи линейного программирования.

Не совместность системы ограничений в таблице 4.8 не выявлена.

Шаг 4. Проверка ограниченности целевой функции.

Неограниченность целевой функции в таблице 4.8 не выявлена.

Шаг 5. Проверка допустимости найденного базисного решения.

Найденное базисное решение допустимое, так как не содержит отрицательных компонент.

Шаг 6. Проверка оптимальности найденного базисного решения.

Найденное базисное решение оптимальное, так как в строке целевой функции симплекс-таблицы (таблица 4.8) нет отрицательных элементов (свободное число данной строки при рассмотрении данного признака не учитывается).

Шаг 7. Проверка альтернативности решения.

Найденное решение является единственным, так как в строке целевой функции (таблица 4.8) нет нулевых элементов (свободное число данной строки при рассмотрении данного признака не учитывается).

Ответ: оптимальное значение целевой функции рассматриваемой задачи $F_{\max} = 24$, которое достигается при $X = (x_1, x_2, x_3, x_4, x_5, x_6) = (6,4,0,0,1,3)$.

4.3. Транспортная задача линейного программирования

Под названием «транспортная задача» объединяется широкий круг задач с единой математической моделью. Данные задачи относятся к задачам линейного программирования и могут быть решены симплексным методом. Однако матрица системы ограничений транспортной задачи настолько своеобразна, что для ее решения разработаны специальные методы. Эти методы, как и симплексный метод, позволяют найти начальное опорное решение, а затем, улучшая его, получить оптимальное решение.

4.3.1. Формулировка транспортной задачи

Однородный груз сосредоточен у k поставщиков в объемах $a_1, a_2, \ldots, a_k$. Данный груз необходимо доставить и потребителям в объемах $b_1, b_2, \ldots, b_n$. Известны c_{ij} $i = 1, 2, \ldots, k$ и $j = 1, 2, \ldots, n$ – стоимости перевозки единицы груза от каждого i-го поставщика каждому j-му потребителю. Требуется составить такой план перевозок, при котором запасы всех поставщиков будут вывезены полностью, запросы всех потребителей полностью удовлетворены и суммарные затраты на перевозку всех грузов минимальны.

Исходные данные транспортной задачи обычно записываются в таблицу:

	b_1	b_2	…	b_n
a_1	c_{11}	c_{12}	…	c_{1n}
a_2	c_{21}	…	…	c_{2n}
…	…	…	…	…
a_k	c_{k1}	…	…	c_{kn}

Исходные данные задачи могут быть представлены также в виде вектора запасов поставщиков $A = (a_1, a_2, \ldots, a_k)$, вектора запросов потребителей $B = (b_1, b_2, \ldots, b_n)$ и матрицы стоимостей $C = \{c_{ij}\}$:

$$C = \begin{pmatrix} c_{11} & c_{12} & \ldots & c_{1n} \\ c_{21} & c_{22} & \ldots & c_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ c_{k1} & c_{k2} & \ldots & c_{kn} \end{pmatrix}. \qquad (4.8)$$

В транспортных задачах под поставщиками и потребителями понимаются различные промышленные и сельскохозяйственные предприятия, заводы, фабрики, склады, магазины и т.д. Однородными считаются грузы, которые могут быть перевезены одним видом транспорта. Под стоимостью перевозок понимаются тарифы, расстояния, время, расход топлива и т.п.

4.3.2. Математическая модель транспортной задачи

Переменными (неизвестными) транспортной задачи являются x_{ij} $..,i-(=1,2, \ldots, k), j= 1,2, \ldots, n$ – объемы перевозок от каждого i-го поставщика каждому j-му потребителю. Эти переменные можно записать в виде матрицы перевозок:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{k1} & x_{k2} & \dots & x_{kn} \end{pmatrix} \tag{4.9}$$

или

	a_1	a_2	…	a_n
b_1	x_{11}	x_{12}	…	x_{1n}
b_2	x_{21}	…	…	x_{2n}
…	…	…	…	…
b_k	x_{k1}	…	…	x_{kn}

Так как произведение $c_{ij}x_{ij}$ определяет затраты на перевозку груза от ***i***-го поставщика ***j***-му потребителю, то суммарные затраты на перевозку всех грузов равны $\sum_{i=1}^{k}\sum_{j=1}^{n} c_{ij}x_{ij}$.

По условию задачи требуется обеспечить минимум суммарных затрат. Следовательно, целевая функция задачи имеет вид:

$$F(x) = \sum_{i=1}^{k}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min \ . \tag{4.10}$$

Система ограничений задачи состоит из двух групп уравнений. Первая группа из ***k*** уравнений описывает тот факт, что запасы всех ***k*** поставщиков вывозятся полностью:

$$\sum_{j=1}^{n} x_{ij} = a_i \, . \tag{4.11}$$

Вторая группа из ***n*** уравнений выражает требование полностью удовлетворить запросы всех *n* потребителей:

$$\sum_{i=1}^{k} x_{ij} = b_j \, . \tag{4.12}$$

Учитывая условие неотрицательности объемов перевозок, математическую модель транспортной задачи можно записать так:

$$F(x) = \sum_{i=1}^{k}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min \ ;$$

$$\sum_{j=1}^{n} x_{ij} = a_i \, ; \ i = 1,2,...,k \, ;$$

$$\sum_{i=1}^{k} x_{ij} = b_j \, ; \ j = 1,2,...,n \, ;$$

$$x_{ij} \geq 0 \, ; \ i = 1,2,...,k \, ; \ j = 1,2,...,n \, .$$

В рассмотренной модели транспортной задачи предполагается, что суммарные запасы поставщиков равны суммарным запросам потребителей, т.е.

$$\sum_{i=1}^{k} a_j = \sum_{j=1}^{n} b_j \,. \qquad (4.13)$$

Такая задача называется задачей с правильным балансом, а ее модель — закрытой. Если же это равенство не выполняется, то задача называется задачей с неправильным балансом, а ее модель – открытой.

Математическая формулировка транспортной задачи такова: найти переменные $\boldsymbol{X=(x_{ij})}$ задачи, удовлетворяющие системе ограничений (4.11), (4.12) , условиям неотрицательности $x_{ij} \geq 0$ и обеспечивающие минимум целевой функции (4.10).

Математическая модель транспортной задачи может быть записана в векторном виде. Для этого рассмотрим матрицу $\boldsymbol{A}$ системы уравнений – ограничений задачи.

$$A = \begin{array}{c} \begin{matrix} x_{11} & x_{12} & \cdots & x_{1k} & x_{21} & x_{22} & \cdots & x_{2k} & \cdots & x_{n1} & x_{n2} & \cdots & x_{nk} \end{matrix} \\ \begin{pmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} \end{array}.$$

Сверху над каждым столбцом матрицы указана переменная задачи, коэффициентами при которой являются элементы соответствующего столбца в уравнениях системы ограничений. Каждый столбец матрицы $\boldsymbol{A}$, соответствующий переменной $\boldsymbol{x_{ij}}$.., является вектором-условием задачи и обозначается через $\boldsymbol{A_{ij}}$. Каждый вектор имеет всего $\boldsymbol{k+n}$ координат, и только две из них, отличные от нуля, равны единице. Первая единица вектора $\boldsymbol{A_{ij}}$ стоит на $\boldsymbol{i}$-м месте, а вторая - на $\boldsymbol{(k+j)}$-м месте, т.е.:

$$A_{ij} = \begin{pmatrix} 0 \\ \ldots \\ 1 \\ \ldots \\ 0 \\ 0 \\ \ldots \\ 1 \\ \ldots \\ 0 \end{pmatrix} \begin{matrix} 1 \\ \ldots \\ i \\ \ldots \\ k \\ k+1 \\ \ldots \\ k+j \\ \ldots \\ k+n \end{matrix}; \qquad A_0 = \begin{pmatrix} a_1 \\ \ldots \\ a_i \\ \ldots \\ a_k \\ b_1 \\ \ldots \\ b_j \\ \ldots \\ b_n \end{pmatrix}.$$

Таким образом, в векторной форме задача будет выглядеть так:

$$F(x) = \sum_{i=1}^{k} \sum_{j=1}^{n} c_{ij} x_{ij} \to \min \;;$$

$$\sum_{i=1}^{k}\sum_{j=1}^{n} A_{ij}x_{ij} = A_0 ;$$

$$x_{ij} \geq 0 ;\ i = 1,2,...,k ;\ \ j = 1,2,...,n .$$

Необходимое и достаточное условия разрешимости транспортной задачи:

«Для того чтобы транспортная задача линейного программирования имела решение, необходимо и достаточно, чтобы суммарные запасы поставщиков равнялись суммарным запросам потребителей:

$$\sum_{i=1}^{k} a_j = \sum_{j=1}^{n} b_j ,$$

т.е. задача должна быть с правильным балансом».

4.3.3. Алгоритм решения транспортной задачи методом потенциалов

Метод потенциалов предназначен для решения транспортной задачи в матричной постановке. Потенциалы – это двойственные переменные. Сам метод – прямой, на каждом шаге выбирается одно из двойственных ограничений, которое не выполняется и исправляется таким образом, чтобы не нарушить ограничения прямой задачи. Постепенно в двойственной задаче ограничения будут выполнены, что будет означать оптимальность в прямой задаче.

Таблица для метода потенциалов имеет следующий вид:

	b_1	b_2	…	b_j	…	b_n	**u**
a_1	c_{11}	c_{12}	…	…	…	c_{1n}	$\mathbf{u_1}$
a_2	c_{21}	…	…	…	…	c_{2n}	$\mathbf{u_1}$
…	…	…	…	…	…	…	**…**
a_i	…	…	…	c_{ij}	…	…	**…**
…	…	…	…	…	…	…	**…**
a_k	c_{k1}	c_{k2}	…	…	…	c_{kn}	$\mathbf{u_k}$
v	$\mathbf{v_1}$	$\mathbf{v_2}$	**…**	$\mathbf{v_j}$	**…**	$\mathbf{v_n}$	

Каждая ячейка (i, j) таблицы хранит информацию о цене товара $c_{ij.}$.

Алгоритм метода потенциалов для решения транспортной задачи будет заключаться в выполнении следующих шагов.

Шаг 1. Проверка на сбалансированность задачи. Необходимо сравнить общий запас груза с суммарным спросом (т.е. проверить выполнение условия (4.13)) и в случае нарушения баланса привести задачу к закрытой модели, добавив фиктивного поставщика или фиктивного потребителя.

Шаг 2. Записать условие задачи в виде транспортной таблицы.

Шаг 3. Построить начальный опорный план перевозок. Для этого можно воспользоваться методом северо-западного угла или методом минимального элемента.

Выбирается клетка в транспортной таблице. В выбранную клетку записывается максимально возможная величина поставки. При этом либо исчерпывается запас груза у поставщиков (“закрывается строка”), либо полностью удовлетворяется спрос потребителя (“закрывается столбец”). Соблюдение этого требования обеспечит заполнение (k+n-1) клеток.

Основные способы построения опорного плана:

1. Метод «Северо-западного угла». Первой загружается клетка (1;1). Если закрывается строка, то следующей загружается клетка (2;1). Если закрывается столбец, то следующей загружается (1;2).Далее загружается клетка соседняя либо по строке, либо по столбцу. Процесс заканчивается, когда распределяют весь груз.

2. Метод минимального элемента. Первой в таблице загружается клетка, которой соответствует наименьший тариф (выбирают только среди тарифов реальных поставщиков и потребителей, запасы фиктивного поставщика распределяются в последнюю очередь). Далее загружается клетка той же строки (столбца) со следующим по величине тарифом и т.д. Процесс заканчивается, когда распределяют весь груз.

На этом же этапе необходимо проверить полученный опорный план на невырожденность. Количество заполненных клеток **N** должно удовлетворять условию **N=n+k-1**. Если это условие не выполняется, то план является вырожденным. Прежде чем двигаться дальше, необходимо выбрать одну незаполненную клетку и записать в нее число ноль, осуществив так называемую нуль-загрузку. Выбирать следует такие клетки, которые не образуют циклов с другими заполненными клетками, иначе опорного плана не получится.

Циклом называется набор клеток вида (i,k), (j,k), (j,t),..., (h,i) в котором две и только две соседние клетки расположены в одном столбце или одной строке таблицы, причем последняя клетка находится в той же строке или столбце, что и первая:

$$(i_1, j_1) \to (i_1, j_2) \to (i_2, j_2) \to \ldots \to (i_s, j_s) \to (i_s, j_1).$$

Минимальный цикл – четыре клетки.

На этом же этапе можно вычислить общие затраты на перевозку всей продукции.

Шаг 4. Составить вспомогательную матрицу затрат. Она строится из исходной матрицы издержек (4.8) путем переноса только тех ячеек c_{ij}, которые соответствуют заполненным клеткам транспортной таблицы. Остальные ячейки остаются пустыми.

Каждому поставщику (каждой строке) ставят в соответствие число u_i, называемое потенциалом поставщика. Каждому потребителю (каждому столбцу) ставят в соответствие некоторое число v_j, называемое потенциалом потребителя.

Далее необходимо рассчитать значения потенциалов. Для этого положить, например, $v_1 = 0$. Остальные потенциалы можно рассчитать с помощью базисных (заполненных клеток), исходя из уравнения:

$$u_i + v_j = c_{ij}. \tag{4.14}$$

Шаг 5. Для свободных клеток рассчитать оценки по формуле:

$$s_{ij} = c_{ij} - u_i - v_j \ . \tag{4.15}$$

Каждая такая оценка показывает, на сколько изменятся общие транспортные затраты при загрузке данной клетки единицей груза. Таким образом, если среди оценок имеются отрицательные (затраты уменьшаются) то данный план можно улучшить переместив в соответствующую клетку некоторое количество продукции. Если же среди оценок нет

отрицательных - план является оптимальным, т.е. если все $s_{ij} \geq 0$, то найдено оптимальное решение. Если же имеется хоть одна отрицательная оценка, то полученный опорный план можно улучшить. Переходим на шаг 6.

Шаг 6. Из всех отрицательных оценок необходимо выбрать наибольшую по модулю, так как ее воздействие на общие затраты является максимальным. Для клетки, занятой этой оценкой, необходимо выполнить следующие действия:

6.1. Построить цикл для этой клетки. Здесь цикл – это замкнутая ломаная линия, которая чередует вертикальное и горизонтальное направления и проходит только по базисным клеткам (заполненным изначально из опорного плана). В исходной клетке необходимо поставить «+» и далее по циклу расставить, чередуя, «+» и «-». Помечать знаками «+» и «-» другие занятые числами ячейки следует таким образом, что в каждой строке и каждом столбце транспортной таблицы число знаков «+»будет равно числу знаков «-» . Это всегда можно сделать единственным образом, причем в каждой строке и каждом столбце содержится по одному «+» и «-».

6.2. Определить минимум M из всех элементов, помеченных знаком «-», и выбрать одну ячейку, где этот минимум достигается. Данная ячейка обозначает загруженную клетку, которая должна стать свободной.

Шаг 7. Нарисовать новую таблицу с пересчитанным планом перевозок: для клеток с «+» прибавить значение M к c_{ij}, а для клеток с «-» – вычесть. Остальные клетки оставляются неизменными.

Продолжать итерации необходимо до тех пор, пока не будет получен оптимальный опорный план, т.е. когда в рабочей матрице затрат будут получены только положительные оценки s_{ij} .

4.3.4. Пример решения транспортной задачи методом потенциалов

Необходимо составить оптимальный план перевозки грузов от трех поставщиков с грузами 240, 40, 110 т. к четырем потребителям с запросами 90, 190, 40 и 130 т. Тарифы на перевозку единицы груза от каждого поставщика к каждому потребителю даны матрицей затрат:

$$\begin{pmatrix} 7 & 13 & 9 & 8 \\ 14 & 8 & 7 & 10 \\ 3 & 15 & 20 & 6 \end{pmatrix}.$$

Шаг 1. Проверка на сбалансированность.

Общее число запасов на складах: 390. Общая потребность: 450. Мы видим, что общая потребность превышает общее число запасов на 60 единиц. Задача является открытой (несбалансированной), для приведения ее к закрытой введем фиктивного производителя №4 с запасом продукции, равным 60. Все издержки по доставке продукции с данного производителя (склада) любому потребителю принимаем равными нулю.

Шаг 2. Запишем условие задачи в виде транспортной таблицы. В верхней строке перечислим потребности потребителей по порядку номеров. В левом столбце перечислим имеющиеся запасы на складах. На пересечении j-го столбца и i-й строки будем записывать количество продукции, поставляемое с i-го склада j-му потребителю. Пока начальное решение не найдено, оставим эти клетки пустыми.

	a_1=90	a_2=190	a_3=40	a_4=130
b_1=240	7	13	9	8
b_4=40	14	8	7	10
b_3=110	3	15	20	6
b_4=60	0	0	0	0

Шаг 3. Отыскание начального решения. Метод северо-западного угла.

Введем вспомогательные строку и столбец, в которых будем отмечать оставшиеся нераспределенные запасы и соответственно потребности (остатки). Изначально их содержимое равно исходным запасам и потребностям, так как еще ничего не распределялось. На рисунке они представлены желтым цветом. Выберем клетку в которую будем распределять продукцию на следующей итерации, это левая верхняя клетка (северо-западный угол). На рисунке как сама клетка так и соответствующие ей остатки отображаются красным шрифтом.

	b_1=90	b_2=190	b_3=40	b_4=130	
a_1=240	X				240
a_2=40					40
a_3=110					110
a_4=60					60
	90	190	40	130	

Итерация: 1. Заполним клетку a_1,b_1. Сравним значения остатков для производителя a_1 и потребителя b_1. Нераспределенных остатков по потребностям для b_1 меньше (см. таблицу выше, красный шрифт), запишем меньшее число в клетку a_1,b_1 одновременно вычитая его из обеих клеток остатков (см. таблицу ниже). При этом клетка остатков по потребностям обнулится, указывая, что все потребности для b_1 удовлетворены (см. таблицу ниже). Поэтому исключим столбец b_1 из дальнейшего рассмотрения (серый фон). Ненулевое значение остатка по запасам для a_1 показывает, сколько единиц продукции у него осталось не потребленной.

	b_1=90	b_2=190	b_3=40	b_4=130	
a_1=240	90	X			150
a_2=40					40
a_3=110					110
a_4=60					60
	0	190	40	130	

Итерация: 2. Заполним клетку a_1,b_2. Сравним значения остатков для производителя a_1 и потребителя b_2. Нераспределенных остатков по запасам для a_1 меньше (см. таблицу выше, красный шрифт), запишем меньшее число в клетку a_1,b_2 одновременно

вычитая его из обеих клеток остатков (см. таблицу ниже). При этом клетка остатков по запасам обнулится, указывая, что все запасы производителя **a_1** использованы (см. таблицу ниже). Поэтому исключим строку **a_1** из дальнейшего рассмотрения (серый фон). Ненулевое значение остатка по потребностям для **b_2** показывает, сколько единиц продукции ему еще требуется.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$	
$a_1=240$	90	150			0
$a_2=40$		X			40
$a_3=110$					110
$a_4=60$					60
	0	40	40	130	

Итерация: 3.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$	
$a_1=240$	90	150			0
$a_2=40$		40			0
$a_3=110$			X		110
$a_4=60$					60
	0	0	40	130	

Итерация: 4.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$	
$a_1=240$	90	150			0
$a_2=40$		40			0
$a_3=110$			40	X	70
$a_4=60$					60
	0	0	0	130	

Итерация: 5.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$	
$a_1=240$	90	150			0
$a_2=40$		40			0
$a_3=110$			40	70	0
$a_4=60$				X	60
	0	0	0	60	

Итерация: 6.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$	
$a_1=240$	90	150			0
$a_2=40$		40			0
$a_3=110$			40	70	0
$a_4=60$				60	0
	0	0	0	0	

Получено допустимое начальное решение (опорный план) (см. таблицу ниже), удовлетворены нужды всех потребителей и использованы все запасы производителей:

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90	150		
$a_2=40$		40		
$a_3=110$			40	70
$a_4=60$				60

Проверим полученный опорный план на невырожденность. Количество заполненных клеток **N** должно удовлетворять условию **N=n+k-1** . В нашем случае N=6, n+k=4+4=8 , план является вырожденным. Прежде чем двигаться дальше, выберем одну незаполненную клетку и запишем в нее число ноль, осуществим так называемую нуль-загрузку. Выбирать следует такие клетки, которые не образуют циклов с другими заполненными клетками, иначе опорного плана не получится.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90	150	0	
$a_2=40$		40		
$a_3=110$			40	70
$a_4=60$				60

Вычислим общие затраты на перевозку всей продукции. Для этого запишем транспортную таблицу в которой совместим найденный опорный план с величинами издержек. В левом верхнем углу каждой клетки будем указывать количество единиц продукции а в правом нижнем затраты на перевозку единицы продукции. (см. таблицу ниже).

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90 7	150 13	0 9	
$a_2=40$		40 8		
$a_3=110$			40 20	70 6
$a_4=60$				60 0

Перемножим числа стоящие в одной клетке (для всех клеток) затем полученные произведения сложим. Получим значение суммарных затрат, для данного начального решения:

$$F_{нач} = 90*7+150*13+0*9+40*8+40*20+70*6+60*0 = 4120 .$$

Шаг 4. Составим вспомогательную матрицу затрат и рассчитаем значения потенциалов по формуле (4.14).

Порядок вычисления потенциалов был следующий:

1) пусть $v_4 = 0$; 2) $u_3 = c_{3,4} - v_4$;

3) $u_4 = c_{4,4} - v_4$; 4) $v_3 = c_{3,3} - u_3$;

5) $u_1 = c_{1,3} - v_3$; 6) $v_1 = c_{1,1} - u_1$;

7) $v_2 = c_{1,2} - u_1$; 8) $u_2 = c_{2,2} - v_2$.

	b_1	b_2	b_3	b_4	
a_1	7	13	9		$u_1=-5$
a_2		8			$u_2=-10$
a_3			20	6	$u_3=6$
a_4				0	$u_4=0$
	$v_1=12$	$v_2=18$	$v_3=14$	$v_4=0$	

Шаг 5. Для свободных клеток рассчитаем оценки по формуле (4.15).

	b_1	b_2	b_3	b_4	
a_1	7	13	9	13	$u_1=-5$
a_2	12	8	3	20	$u_2=-10$
a_3	-15	-9	20	6	$u_3=6$
a_4	-12	-18	-14	0	$u_4=0$
	$v_1=12$	$v_2=18$	$v_3=14$	$v_4=0$	

Шаг 6. Из всех отрицательных оценок выберем наибольшую по модулю (красный цвет), так как ее воздействие на общие затраты является максимальным. В нашем случае такая оценка находится в ячейке **a_4,b_2** (красный цвет), в соответствующую ячейку транспортной таблицы мы должны переместить некоторое количество продукции т.е. загрузить ее. Отметим в транспортной таблице ячейку **a_4,b_2** знаком «+» . Кроме нее, мы пометим знаками «-» и «+» другие занятые числами ячейки таким образом, что в каждой строке и каждом столбце транспортной таблицы число знаков «+» будет равно числу знаков «-».

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90	150 -	0 +	
$a_2=40$		40		
$a_3=110$			40 -	70 +
$a_4=60$		+		60 -

Затем мы определим минимум **M** из всех элементов, помеченных знаком «-», и выбираем одну ячейку, где этот минимум достигается. В нашем случае таковой является **a_3,b_3**, она обозначает загруженную клетку, которая должна стать свободной.

Число **M** при этом составляет: 40.

Шаг 7. Строим новую транспортную таблицу с пересчитанным планом перевозок:

а) в ячейку **a_4,b_2** новой таблицы записывается число **M;**

б) ячейка **a_3,b_3** остается пустой;

в) в остальных ячейках, помеченных знаками - или +, число **M** соответственно вычитается из стоящего в ячейке числа или складывается с ним; результат вносится в соответствующую ячейку новой таблицы;

г) непомеченные числа переносятся в новую таблицу без изменений; остальные ячейки новой таблицы остаются пустыми.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90	110	40	
$a_2=40$		40		
$a_3=110$				110
$a_4=60$		40		20

Начинаем вторую итерацию цикла расчета, с построения новой рабочей матрицы затрат (шаг 4 первой итерации).

Итерация: 2.

Рабочая матрица затрат с пересчитанными потенциалами и оценками:

	b_1	b_2	b_3	b_4	
a_1	7	13	9	-5	$u_1=13$
a_2	12	8	3	2	$u_2=8$
a_3	3	9	18	6	$u_3=6$
a_4	6	0	4	0	$u_4=0$
	$v_1=-6$	$v_2=0$	$v_3=-4$	$v_4=0$	

Ячейка **a_1,b_4**, транспортной таблицы должна загрузиться.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90	110 -	40	+
$a_2=40$		40		
$a_3=110$				110
$a_4=60$		40 +		20 -

Ячейка **a_4,b_4** становится свободной. M = 20.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90	90	40	20
$a_2=40$		40		
$a_3=110$				110
$a_4=60$		60		

Итерация: 3.

Рабочая матрица затрат с пересчитанными потенциалами и оценками:

	b_1	b_2	b_3	b_4	
a_1	7	13	9	8	$u_1=8$
a_2	12	8	3	7	$u_2=3$
a_3	-2	4	13	6	$u_3=6$
a_4	6	0	4	5	$u_4=-5$
	$v_1=-1$	$v_2=5$	$v_3=1$	$v_4=0$	

Ячейка **a_3,b_1**, транспортной таблицы, должна загрузиться.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$	90 -	90	40	20 +
$a_2=40$		40		
$a_3=110$	+			110 -
$a_4=60$		60		

Ячейка **a_1,b_1** становится свободной. M = 90.

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$		90	40	110
$a_2=40$		40		
$a_3=110$	90			20
$a_4=60$		60		

Итерация: 4.

Рабочая матрица затрат с пересчитанными потенциалами и оценками:

	b_1	b_2	b_3	b_4	
a_1	2	13	9	8	$u_1=8$
a_2	14	8	3	7	$u_2=3$
a_3	3	4	13	6	$u_3=6$
a_4	8	0	4	5	$u_4=-5$
	$v_1=-3$	$v_2=5$	$v_3=1$	$v_4=0$	

В приведенной выше таблице нет отрицательных оценок (план улучшить нельзя), следовательно, достигнуто оптимальное решение:

	$b_1=90$	$b_2=190$	$b_3=40$	$b_4=130$
$a_1=240$		90 13	40 9	110 8
$a_2=40$		40 8		
$a_3=110$	90 3			20 6
$a_4=60$		60 0		

Общие затраты на перевозку всей продукции, для оптимального плана составляют: $F_{\min} = 3120$.

4.4. Контрольные вопросы

1. В чем заключается геометрический смысл задачи линейного программирования?
2. Какой вид может иметь область допустимых решений при геометрическом методе решения ЗЛП?
3. Что определяет вектор-градиент целевой функции? Как его построить?
4. В чем заключается фундаментальная теорема симплекс-метода?
5. Как осуществляется переход к канонической форме записи задачи линейного программирования?
6. Как проверить совместность системы ограничений задачи линейного программирования? Какой будет вывод, если система ограничений несовместна?
7. Как проверить ограниченность целевой функции? Какой будет вывод, если целевая функция неограниченна?
8. Какое базисное решение считается допустимым в симплекс-методе? Как поступить, если базисное решение оказалось недопустимым?
9. Какое базисное решение считается оптимальным в симплекс-методе? Как поступить, если базисное решение оказалось неоптимальным?
10. Как определить разрешающий элемент в симплекс-таблице?
11. В чем заключается «правило прямоугольника» при преобразовании симплекс-таблицы?
12. Как математически сформулировать транспортную задачу линейного программирования?
13. В чем заключается необходимое и достаточное условия разрешимости транспортной задачи?
14. Как составить исходный опорный план транспортной задачи методом северо-западного угла и методом минимального элемента?
15. В чем суть метода потенциалов при решении транспортной задачи?
16. Что понимается под циклом в транспортной задаче?
17. Какой опорный план в транспортной задаче считается оптимальным?

4.5. Задачи для самостоятельного решения

Задание 1

Решить задачу линейного программирования геометрическим и симплекс-методом. Сравнить результаты.

Таблица 4.9

Варианты

№ варианта	ЗЛП
1.	$f = 2x_1 + 3x_2 \to \max$ $\begin{cases} 2x_1 + x_2 \le 10 \\ -2x_1 + 3x_2 \le 6 \\ 2x_1 + 4x_2 \ge 8 \end{cases}$ $x_1, x_2 \ge 0$

Продолжение таблицы 4.9

2.	$f = x_1 + 2x_2 \to \max$ $\begin{cases} 4x_1 - 2x_2 \le 12 \\ -x_1 + 3x_2 \le 6 \\ 2x_1 + 4x_2 \le 16 \end{cases}$ $x_1, x_2 \ge 0$
3.	$f = 2x_1 + 7x_2 \to \max$ $\begin{cases} -2x_1 + 3x_2 \le 14 \\ x_1 + x_2 \le 8 \end{cases}$ $x_1, x_2 \ge 0$
4.	$f = 2x_1 + 15x_2 \to \max$ $\begin{cases} 5x_1 + 2x_2 \le 25 \\ 3x_1 + 4x_2 \le 24 \\ -x_1 + 3x_2 \le 12 \end{cases}$ $x_1, x_2 \ge 0$
5.	$f = 5x_1 + 3x_2 \to \max$ $\begin{cases} 3x_1 + 5x_2 \le 15 \\ 5x_1 + 2x_2 \le 10 \end{cases}$ $x_1, x_2 \ge 0$
6.	$f = x_1 + x_2 \to \max$ $\begin{cases} x_1 + 2x_2 \le 10 \\ x_1 + 2x_2 \ge 2 \\ 2x_1 + x_2 \le 10 \end{cases}$ $x_1, x_2 \ge 0$
7.	$f = x_1 + x_2 \to \min$ $\begin{cases} 2x_1 + 4x_2 \le 16 \\ -4x_1 + 2x_2 \le 8 \\ x_1 + 3x_2 \ge 9 \end{cases}$ $x_1, x_2 \ge 0$
8.	$f = x_1 + x_2 \to \max$ $\begin{cases} x_1 + 2x_2 \le 14 \\ -5x_1 + 3x_2 \le 15 \\ 4x_1 + 6x_2 \ge 24 \end{cases}$ $x_1, x_2 \ge 0$
9.	$f = -2x_1 + x_2 \to \min$ $\begin{cases} 3x_1 - 2x_2 \le 12 \\ -x_1 + 2x_2 \le 8 \\ 2x_1 + 3x_2 \ge 6 \end{cases}$ $x_1, x_2 \ge 0$

Окончание таблицы 4.9

10.	$f = -2x_1 - 3x_2 \to \min$ $\begin{cases} -4x_1 + 2x_2 \ge 4 \\ x_1 + x_2 \ge 6 \end{cases}$ $x_1, x_2 \ge 0$
11.	$f = 2x_1 + 3x_2 \to \max$ $\begin{cases} -3x_1 - 2x_2 \le -6 \\ x_1 + 4x_2 \le 4 \end{cases}$ $x_1, x_2 \ge 0$
12.	$f = 2x_1 + 3x_2 \to \max$ $\begin{cases} x_1 + x_2 \le 2 \\ x_1 + x_2 \ge 1 \end{cases}$ $x_1, x_2 \ge 0$
13.	$f = x_1 + 3x_2 \to \max$ $\begin{cases} x_1 - x_2 \le 1 \\ 2x_1 + x_2 \le 2 \\ x_1 - x_2 \ge 0 \end{cases}$ $x_1, x_2 \ge 0$
14.	$f = x_1 + 2x_2 \to \max$ $\begin{cases} 2x_1 + 3x_2 \le 6 \\ 2x_1 + x_2 \le 4 \\ x_1 \le 1 \\ x_1 - x_2 \ge -1 \\ 2x_1 + x_2 \ge 1 \end{cases}$ $x_1, x_2 \ge 0$

Задание 2

Решить транспортную задачу линейного программирования.

1. На складах A, B, C имеются запасы продукции в количествах 90, 400, 110 т. соответственно. Потребители M, H, K должны получить эту продукцию в количествах 140, 300, 160 т. соответственно. Найти такой вариант прикрепления поставщиков к потребителям, при котором сумма затрат на перевозки была бы минимальной. Расходы по перевозке 1 т. продукции заданы таблицей (у.е.).

$$\begin{pmatrix} 2 & 5 & 2 \\ 4 & 1 & 5 \\ 3 & 6 & 8 \end{pmatrix}$$

2. В пунктах A и B находятся соответственно 150 и 90 т горючего. Пунктам 1, 2 и 3 требуются соответственно 60, 70, 110 т. горючего. Стоимость перевозки 1 т. горючего из пункта A в пункты 1, 2, 3 равна 60, 10, 40 ден. ед. за 1 т. соответственно, а из пункта B в

пункты 1, 2, 3 – 120, 20, 80 ден. ед. за 1 т. соответственно. Составьте план перевозок горючего, минимизирующий общую сумму транспортных расходов.

3. В трех хранилищах горючего ежедневно хранится 175, 125 и 140 т бензина. Этот бензин ежедневно получают четыре заправочные станции в количествах, равных соответственно 180, 160, 60 и 40 т. Стоимости перевозок 1 т бензина с хранилищ к заправочным станциям задаются матрицей. Составить такой план перевозок бензина, при котором общая стоимость перевозок является минимальной.

$$\begin{pmatrix} 9 & 7 & 5 & 3 \\ 1 & 2 & 4 & 6 \\ 8 & 10 & 12 & 1 \end{pmatrix}$$

4. Промышленный концерн имеет два завода и пять складов в различных регионах страны. Каждый месяц первый завод производит 40, а второй 70 ед. продукции. Вся продукция, производимая заводами, должна быть направлена на склады. Вместимость первого склада равна 20 ед. продукции; второго – 30; третьего – 15; четвертого – 27; пятого – 28 единиц. Издержки транспортировки продукции от завода до склада приведены в матрице. Распределите план перевозок из условия минимизации ежемесячных расходов на транспортировку.

$$\begin{pmatrix} 520 & 480 & 650 & 500 & 720 \\ 450 & 525 & 630 & 560 & 750 \end{pmatrix}$$

5. На трех хлебокомбинатах ежедневно производится 110, 190 и 90 т муки. Эта мука потребляется четырьмя хлебозаводами, ежедневные потребности которых равны соответственно 80, 60, 170 и 80 т. Тарифы перевозок 1 т муки с хлебокомбинатов к каждому из хлебозаводов задаются матрицей. Составить такой план доставки муки, при котором общая стоимость перевозок является минимальной.

$$\begin{pmatrix} 8 & 1 & 9 & 7 \\ 4 & 6 & 2 & 12 \\ 3 & 5 & 8 & 9 \end{pmatrix}$$

6. Груз, хранящийся на трех складах и требующий для перевозки 60, 80, 106 автомашин соответственно, необходимо перевезти в четыре магазин. Первому магазину требуется 44 машины груза, второму – 70, третьему – 50 и четвертому – 82 машины. Стоимость пробега одной машины за 1 км составляет 10 ден. Единиц. Расстояния от складов до магазинов указаны в матрице. Составьте минимальный по стоимости план перевозки груза от складов до магазинов.

$$\begin{pmatrix} 13 & 17 & 6 & 8 \\ 2 & 7 & 10 & 41 \\ 12 & 18 & 2 & 22 \end{pmatrix}$$

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сухарев, А.Г. Курс методов оптимизации [Электронный ресурс]: учеб. пособие / А.Г. Сухарев, А.В. Тимохов, В.В. Федоров – М.: Физматлит, 2011. – 384 с. – Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=2330 – Загл. с экрана. ISBN 978-5-9221-0559-0

2. Лесин, В.В Основы методов оптимизации [Текст] : учебное пособие / В. В. Лесин, Ю. П. Лисовец. - 3-е изд., испр. - СПб. и др. : Лань, 2011. - 341 с. : ил., граф., табл. - (Учебники для вузов. Спец. лит.). - ISBN 978-5-8114-1217-4.

3. Кочегурова, Е.А. Теория и методы оптимизации [Текст] / Е.А. Кочегурова. – Томск: Изд-во Томского политехнического университета, 2012. – 157 с.

4. Измаилов, А. Ф. Численные методы оптимизации [Электронный ресурс] : учебное пособие / А. Ф. Измаилов, М. В. Солодов. - М. : ФИЗМАТЛИТ, 2008. - 320 с. – Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=2184 – Загл. с экрана. ISBN *978-5-9221-0975-8*

5. Акулич, И.Л. Математическое программирование в примерах и задачах [Электронный ресурс]: учебное пособие. /И.Л. Акулич - Спб.: Изд-во «Лань», - 352 с. – Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=2027 – ISBN 978-5-8114-0916-7

6. Баженов, Н.М. Методы одномерной и многомерной оптимизации [Электронный ресурс]: практикум по дисциплине «Моделирование систем» / Н.М. Баженов, Е.С. Рябчикова. – Электрон. дан. – Магнитогорск: ФГБОУ ВПО «МГТУ им. Г.И. Носова», 2011. – 1 электрон. опт. диск (CD-R) Номер гос. регистрации 0321102638 – М.: ФГУП НТЦ «Информрегистр» –Загл. с этикетки диска

7. Пантелеев, А.В. Методы оптимизации в примерах и задачах: Учеб. пособие [Текст] / А.В. Пантелеев, Т.А. Летова. – М.: Высш. шк., 2005. – 544 с.

8. Галеев, Э.М. Оптимизация: теория, примеры, задачи [Текст] / Э.М. Галеев, В.М. Тихомиров. – М.: Эдиториал УРСС, 2000. – 320 с.

9. Уайлд, Д. Дж. Методы поиска экстремума [Текст]. – М.: Наука, 1967. – 182 с.

10. Форсайт, Дж. Машинные методы математических вычислений [Текст] / Дж. Форсайт, М. Малькольм, К.Моулер. – М: Мир, 1980. – 279 с.

11. Бояринов, А.И. Методы оптимизации в химической технологии [Текст] / А.И. Бояринов, В.В. Кафаров. – М.: Химия, 1975. – 562 с.

12. Дегтярев, Ю.И. Методы оптимизации [Текст] / Ю.И. Дегтярев. – М.: Радио и связь, 1980. – 269 с.

13. Банди, Б. Методы оптимизации. Вводный курс [Текст] / Б. Банди. – М.: Радио и связь, 1988. – 128 с.

14. Высшая математика. Решение задач и примеров Online [Электронный ресурс]. – Режим доступа: http://www.math-pr.com/index.html.

ПРИЛОЖЕНИЕ

П.1. Программная реализация метода полного перебора на языке Visual Basic for Application

```
Sub Метод_полного_перебора()

Dim A, B As Integer
Dim n, h, X, U, min, minX, e As Double

'Задаем границы целевой функции
A = -5
B = 5

e = 0.0001 ' задаем точность
n = (B - A) / e ' Определяем число точек на интервале, исходя из точности
h = (B - A) / n ' Определяем шаг расчета
min = 10 'задаем начальное значение минимума

For X = A To X >= B Step h ' перебираем точки в цикле на промежутке с шагом h
   U = X * X + 6 * Exp(0.15 * X) ' целевая функция
      If U < min Then ' если нашли точку, меньшую, чем значение минимума в данный момент
         min = U 'присваиваем координату Y как минимум
         minX = X 'присваиваем связанную с ней координату X как минимум
      End If
Next X

'записываем координаты в первые ячейки первого листа
Cells(1, 1) = minX
Cells(1, 2) = min

End Sub
```

П.2. Программная реализация метода золотого сечения на языке Visual Basic for Application

```
Dim E As Double
Dim a As Double, b As Double
Dim x1 As Double, x2 As Double, y1 As Double, y2 As Double
Dim y As Double, x As Double
Dim x0 As Double
```

```
'целевая функция
Function func(x As Double) As Double
func = x * x + 6 * Exp(0.15 * x)
End Function

Function fn1()
x1 = 0.618 * a + 0.382 * b
y1 = func(x1)
End Function

Function fn2()
x2 = 0.382 * a + 0.618 * b
y2 = func(x2)
End Function

Sub Метод_золотого_сечения()

a = -5
b = 5
'x0 = 0

E = 0.001

fn1
fn2

Do While (Abs(b - a) > E)
   If y1 > y2 Then
      a = x1
      x1 = x2
      y1 = y2
      fn2
   Else
      b = x2
      x2 = x1
      y2 = y1
      fn1
   End If
Loop
   x = (a + b) / 2

Cells(2, 1) = x
Cells(2, 2) = func(x)
```

End Sub

П.3. Программная реализация метода квадратичной интерполяции на языке Visual Basic for Application

```
Sub Квадратичная_интерполяция()
Const x0 = -5, h = 0.001, k1 = 4, k2 = -0.25, e = 0.00001
Dim i As Integer, tmpf As Single, tmpx As Single
Dim x(1 To 4) As Single
Dim f(1 To 4) As Single
Dim flag As Single
x(1) = x0
x(2) = x0 + h

f(1) = x(1) ^ 2 + k1 * Exp(k2 * x(1))
f(2) = x(2) ^ 2 + k1 * Exp(k2 * x(2))

If f(1) < f(2) Then
   x(1) = x0 - h
   x(2) = x0
   x(3) = x0 + h
Else
   x(1) = x0
   x(2) = x0 + h
   x(3) = x0 + 2 * h
End If



f(1) = x(1) ^ 2 + k1 * Exp(k2 * x(1))
f(2) = x(2) ^ 2 + k1 * Exp(k2 * x(2))
f(3) = x(3) ^ 2 + k1 * Exp(k2 * x(3))

M2:
x(4) = 0.5 * (f(1) * (x(2) ^ 2 - x(3) ^ 2) + f(2) * (x(3) ^ 2 - x(1) ^ 2) + f(3) * (x(1) ^ 2 - x(2) ^ 2)) /
(f(1) * (x(2) - x(3)) + f(2) * (x(3) - x(1)) + f(3) * (x(1) - x(2)))
f(4) = x(4) ^ 2 + k1 * Exp(k2 * x(4))

Do
flag = 0

For i = 1 To 3

   If f(i) > f(i + 1) Then
```

```
    tmpf = f(i)
    tmpx = x(i)
    f(i) = f(i + 1)
    x(i) = x(i + 1)
    f(i + 1) = tmpf
    x(i + 1) = tmpx
    flag = 1
    End If
Next i

Loop While flag = 1

If Abs(f(2) - f(1)) < e Then 'проверка конца программы
GoTo M1
End If

p1 = Abs(x(2) - x(1))
p2 = Abs(x(3) - x(1))
p3 = Abs(x(4) - x(1))


If p1 = Application.Max(p1, p2, p3) Then
x(2) = x(3)
f(2) = f(3)
x(3) = x(4)
f(3) = f(4)
ElseIf p2 = Application.Max(p1, p2, p3) Then
x(3) = x(4)
f(3) = f(4)
End If

GoTo M2
M1:
MsgBox (x(1))
MsgBox (f(1))
End Sub
```

П.4. Программная реализация метода покоординатного спуска на языке Visual Basic for Application

```
Sub Покоординатный_спуск()

Const k1 = 14, k2 = -0.1, k3 = 1.69, k4 = 0.24 'исходные параметры
```

```
Const t = (5 ^ (1 / 2) - 1) / 2 ' постоянная для метода "золотого сечения"
Const e = 0.01 ' точность вычислений
Dim xa As Single, xb As Single, a As Single, b As Single, i As Integer ' переменные для метода "золотого сечения"
Dim x1 As Single, x2 As Single, x1old As Single, x2old As Single
Dim z As Integer, g As Integer ' счётчик итераций
x1 = 1 ' задание начальной координаты x1
x2 = 1 'задание начальной координаты x2
i = 37 ' строка, с которой начинается заполнение таблицы
a = -1.5 'левая граница
b = 1.5 ' правая граница

Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1

M1:
x1old = x1 'запоминаем предшествующее значение x1
x2old = x2 'запоминаем предшествующее значение x2

xa = a + (1 - t) * (b - a)
xb = a + t * (b - a)
fa = k1 * xa + k2 * x2 + Exp(k3 * xa ^ 2 + k4 * x2 ^ 2)
fb = k1 * xb + k2 * x2 + Exp(k3 * xb ^ 2 + k4 * x2 ^ 2)

If fa < fb Then ' находим отрезок, который будем делить дальше
b = xb
Else
a = xa
End If

If Abs(a - b) > e Then GoTo M1

x1 = (xb + xa) / 2
a = -1.5
b = 1.5
Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1

M2:
xa = a + (1 - t) * (b - a)
xb = a + t * (b - a)
```

```
fa = k1 * x1 + k2 * xa + Exp(k3 * x1 ^ 2 + k4 * xa ^ 2)
fb = k1 * x1 + k2 * xb + Exp(k3 * x1 ^ 2 + k4 * xb ^ 2)
If fa < fb Then
b = xb
Else
a = xa
End If

If Abs(xb - xa) > e Then GoTo M2

x2 = (xb + xa) / 2
a = -1.5
b = 1.5
Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1

If Sqr((x1 - x1old) ^ 2 + (x2 - x2old) ^ 2) > e Then GoTo M1

MsgBox (x1)
MsgBox (x2)

End Sub
```

П.5. Программная реализация классического градиентного метода на языке Visual Basic for Application

```
Sub Градиентный_метод()

Const k1 = 14, k2 = -0.1, k3 = 1.69, k4 = 0.24 'исходные параметры
Const e = 0.01 ' точность вычислений
Const h = 0.01 ' шаг
Dim dFx1 As Single, dFx2 As Single
Dim a1 As Single, a2 As Single ' переменные для вычисления оптимального шага
Dim x1 As Single, x2 As Single ' начальная точка
Dim gradf As Single ' модуль вектора градиента
Range("A37:B291").Clear
x1 = -0.5 'задание координат начальной точки
x2 = 0.5
i = 37 ' начальная строка для заполнения таблицы
Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1
```

```
dFx1 = k1 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k3 * x1 'частная производная по x1
dFx2 = k2 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k4 * x2 'частная производная по x2

gradf = Sqr(dFx1 ^ 2 + dFx2 ^ 2) 'модуль градиента

Do While gradf > e 'условие окончания поиска

x1 = x1 - h * dFx1
x2 = x2 - h * dFx2

dFx1 = k1 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k3 * x1
dFx2 = k2 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k4 * x2

gradf = Sqr(dFx1 ^ 2 + dFx2 ^ 2)

Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1

Loop

MsgBox (x1)
MsgBox (x2)

End Sub
```

П.6. Программная реализация метода наискорейшего спуска на языке Visual Basic for Application

```
Sub Наискорейший_спуск()

Const k1 = 14, k2 = -0.1, k3 = 1.69, k4 = 0.24 'исходные параметры
Const t = (5 ^ (1 / 2) - 1) / 2 ' постоянная для метода "золотого сечения"
Const e = 0.001 ' точность вычислений
Dim dFx1 As Single, dFx2 As Single
Dim k5 As Single, k6 As Single ' переменные для вычисления оптимального шага
Dim x1 As Single, x2 As Single ' начальная точка
Dim gradf As Single ' модуль вектора градиента
Dim fa As Double, fb As Double
Range("A37:B291").Clear
x1 = -1 'задание координат начальной точки
x2 = 0.5
i = 37 ' начальная строка для заполнения таблицы
```

```
Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1
a = -1.5
b = 1.5

dFx1 = k1 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k3 * x1 'частная производная по x1
dFx2 = k2 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k4 * x2 'частная производная по x2

gradf = Sqr(dFx1 ^ 2 + dFx2 ^ 2) 'модуль градиента

Do While gradf > e 'условие окончания поиска

a = -1.5
b = 1.5
' поиск оптимального шага методом золотого сечения
   Do While Abs(a - b) > e
   ha = a + (1 - t) * (b - a)
   hb = a + t * (b - a)
   k5 = x1 - ha * dFx1
   k6 = x2 - ha * dFx2
   fa = k1 * k5 + k2 * k6 + Exp(k3 * (k5 ^ 2) + k4 * (k6 ^ 2))
   k5 = x1 - hb * dFx1
   k6 = x2 - hb * dFx2
   fb = k1 * k5 + k2 * k6 + Exp(k3 * (k5 ^ 2) + k4 * (k6 ^ 2))
   If fa < fb Then ' находим отрезок, который будем делить дальше
   b = hb
   Else
   a = ha
   End If
   Loop

h = (ha + hb) / 2

x1 = x1 - h * dFx1
x2 = x2 - h * dFx2
dFx1 = k1 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k3 * x1
dFx2 = k2 + Exp(k3 * x1 ^ 2 + k4 * x2 ^ 2) * 2 * k4 * x2
gradf = Sqr(dFx1 ^ 2 + dFx2 ^ 2)
Cells(i, "A") = x1
Cells(i, "B") = x2
i = i + 1
Loop
```

```
MsgBox (x1)
MsgBox (x2)

End Sub
```

П.7. Программная реализация метода конфигураций на языке Visual Basic for Application

```
Public Function fcn(ByVal x1 As Single, ByVal x2 As Single) As Single
fcn = 14 * x1 - 0.1 * x2 + Exp(1.69 * x1 ^ 2 + 0.24 * x2 ^ 2)
End Function

Sub Метод_конфигураций()

Dim h As Single, e As Single 'шаг вычислений; точность
Dim x1 As Single, x2 As Single ' текущая координата
Dim t11 As Single, t12 As Single 'первая временная вершина
Dim t21 As Single, t22 As Single 'вторая временная вершина
Dim flag As Boolean ' Отслеживаем момент уменьшения шага
Dim x1old As Single, x2old As Single ' для хранения координат перед x2
Dim i As Integer
i = 1
Range("A1:B100").Clear

x1 = -1 'начальные значения
x2 = 1
h = 0.1 'величина шага
e = 0.001

Range("A1:B100").Clear
Cells(i, 1) = x1
Cells(i, 2) = x2
i = i + 1

Do While h > e
flag = False
t11 = x1 ' координаты первой временной вершины
t12 = x2

If fcn(x1 + h, x2) < fcn(x1, x2) Then
x1 = x1 + h
ElseIf fcn(x1 - h, x2) < fcn(x1, x2) Then
```

```
x1 = x1 - h
End If

If fcn(x1, x2 + h) < fcn(x1, x2) Then
x2 = x2 + h
ElseIf fcn(x1, x2 - h) < fcn(x1, x2) Then
x2 = x2 - h
End If

Cells(i, 1) = x1
Cells(i, 2) = x2
i = i + 1

x1old = x1
x2old = x2
x1 = t11 + 2 * (x1 - t11)
x2 = t12 + 2 * (x2 - t12)

Cells(i, 1) = x1
Cells(i, 2) = x2
i = i + 1

t21 = x1 'вторая временная вершина
t22 = x2

If fcn(t21, t22) < fcn(t11, t12) Then flag = True ' если всё ещё идём в сторону уменьшения - шаг менять не надо

If flag = False Then
h = h / 2
x1 = x1old
x2 = x2old
End If

Loop

MsgBox (x1)
MsgBox (x2)

End Sub
```

П.8. Программная реализация метода Нелдера-Мида на языке Visual Basic for Application

```
Const N = 2 ' Размерность фунции
Dim K As Double ' Начальная размерность симплекса
Dim E As Double ' Конечный размер симплекса
Dim KN As Integer 'Число вычислений функции


Rem  *****  BASIC  *****
Rem ***** Значение функции ************************
Rem Функция записывет значение x1 и x2 в
Rem ячейки таблицы B5 и C5 соответственно
Rem и возвращает полученный результат из ячейки B7

Function func(x1 As Double, x2 As Double) As Double
Worksheets(1).Range("B5") = x1
Worksheets(1).Range("C5") = x2
func = Worksheets(1).Range("B7")
KN = KN + 1
Rem Записываем в ячейку число вычислений (G4)

Worksheets(1).Range("g4") = KN
End Function


Rem ***** Алгоритм Нелдера - Мида
Sub Nelder()
 Dim S(N + 1, N) As Double 'Координаты симплекса
Dim X(N)  As Double  ' Координата текущей исследуемой точки
Dim XH(N) As Double  ' Координата точки с максимальным значением функции
Dim XG(N) As Double  ' Координата точки со значением функции второй после
максимального
Dim XL(N) As Double  ' Координата точки с минимальным значением функции
Dim XO(N) As Double  ' Координата центра тяжести всех, кроме максимальной, точек
Dim XR(N) As Double  ' Координата отраженной точки относительно XH
Dim XC(N) As Double  ' Координата сжатия симплекса относительно XR
Dim XE(N) As Double  ' Координата растяжения симплекса относительно XR
Dim F(N + 1) As Double ' Значение функции в вершинах симплекса
Dim I As Integer     ' Переменная для перемещения между точками
Dim J As Integer     ' Переменная перемещения по координатам
Dim H As Integer     ' Номер вершины симплекса с максимальным значение функции
Dim L As Integer     ' Номер вершины симплекса с минимальным значение функции
Dim FC, FE, FH, FL, FG, FO, FR As Variant ' Значение функции в соответствующих точках
```

```
Dim S1, S2, SIG As Double ' Текущий размер симплкса
Const AL = 1.1
Const BE = 0.6
Const GA = 2.1
' Передаем начальное значение точки симплекса

S(1, 1) = Worksheets(1).Range("B5")
S(1, 2) = Worksheets(1).Range("C5")
E = Worksheets(1).Range("d5")
K = Worksheets(1).Range("d7")
KN = 0
' Строим симплекс вокруг начальной точки
For I = 2 To N + 1
  For J = 1 To N
    If J = I - 1 Then
      S(I, J) = S(1, J) + K
    Else
      S(I, J) = S(1, J)
    End If
  Next J
Next I
' Вычисляем значение функции в вершинах симплекса
For I = 1 To N + 1
  For J = 1 To N
   X(J) = S(I, J)
  Next J
 F(I) = func(X(1), X(2))
Next I
' Начинаем непосредственно изменять симплекс - главный цикл
Do
' Находим наибольшее и наименьшее значение функции в вершинах симплекса
FH = -1E+300
FL = 1E+300
For I = 1 To N + 1
 If F(I) > FH Then
  FH = F(I)
  H = I
 End If
 If F(I) < FL Then
  FL = F(I)
  L = I
 End If
Next I
```

```
' Находим второе наибольшее значение и  соответствующую ему точку
FG = -1E+300
For I = 1 To N + 1
 If (F(I) > FG) And (I <> H) Then
    FG = F(I)
    G = I
 End If
Next I

' Вычисление центра тяжести всех точек кроме xh
For J = 1 To N
  XO(J) = 0
  For I = 1 To N + 1
    If I <> H Then XO(J) = XO(J) + S(I, J)
  Next I
  XO(J) = XO(J) / N
  XH(J) = S(H, J)
  XG(J) = S(G, J)
  XL(J) = S(L, J)
Next J
For J = 1 To N
  X(J) = XO(J)
Next J
' вычисляем функцию в центре тяжести
FO = func(X(1), X(2))

' выполняем отражение
For J = 1 To N
 XR(J) = XO(J) + AL * (XO(J) - XH(J))
 X(J) = XR(J)
Next J
FR = func(X(1), X(2))

' Определяем дальнейшее поведение симплекса
' растяжение или сжатие в зависимости от условий
If FR < FL Then
   For J = 1 To N
     XE(J) = GA * XR(J) + (1 - GA) * XO(J)
     X(J) = XE(J)
   Next J
   FE = func(X(1), X(2))
   If FE < FL Then
```

```
      For J = 1 To N
        S(H, J) = XE(J)
      Next J
      F(H) = FE
    Else
      For J = 1 To N
        S(H, J) = XR(J)
      Next J
      F(H) = FR
    End If
  Else
    If FR > FG Then
      If FR < FH Then
        For J = 1 To N
          XH(J) = XR(J)
        Next J
        F(H) = FR
      End If
      For J = 1 To N
        XC(J) = BE * XH(J) + (1 - BE) * XO(J)
        X(J) = XC(J)
      Next J
      FC = func(X(1), X(2))

      If FC > FH Then
        For I = 1 To N + 1
          For J = 1 To N
            S(I, J) = (S(I, J) + XL(J)) / 2
            X(J) = S(I, J)
          Next J
        F(I) = func(X(1), X(2))
        Next I
      Else
      For J = 1 To N
        S(H, J) = XC(J)
      Next J
      F(H) = FC
      End If
    Else
      For J = 1 To N
        S(H, J) = XR(J)
      Next J
      F(H) = FR
```

```
    End If
End If
' Проверка условия окончания цикла поиска
S1 = 0
S2 = 0
For I = 1 To N + 1
  S1 = S1 + F(I)
  S2 = S2 + F(I) * F(I)
Next I
SIG = S2 - S1 * S1 / (N + 1)
SIG = SIG / (N + 1)

Loop Until SIG < E
FL = func(XL(1), XL(2))
End Sub
```

Учебное текстовое электронное издание

Рябчикова Елена Сергеевна
Андреев Сергей Михайлович
Рябчиков Михаил Юрьевич

МЕТОДЫ И ТЕОРИИ ОПТИМИЗАЦИИ

Учебное пособие

1,59 Мб
1 электрон. опт. диск

г. Магнитогорск, 2016 год
ФГБОУ ВО «МГТУ им. Г.И. Носова»
Адрес: 455000, Россия, Челябинская область, г. Магнитогорск,
пр. Ленина 38

ФГБОУ ВО «Магнитогорский государственный
технический университет им. Г.И. Носова»
Кафедра автоматизированных систем управления
Центр электронных образовательных ресурсов и
дистанционных образовательных технологий
e-mail: ceor_dot@mail.ru