Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра автоматического управления

А.В. Павлова, М.К. Хаджинов

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ СИСТЕМ

Методическое пособие

к лабораторным работам для студентов специальностей
53 01 03 «Автоматическое управление в технических системах»
и 53 01 07 «Информационные технологии и управление
в технических системах»
дневной и вечерней форм обучения

Минск 2003

УДК 519.85 (075.8)
ББК 22.18 я 73
П 12

Павлова А.В.

П 12 Математические основы теории систем: Метод. пособие к лаб. работам для студентов спец. 53 01 03 «Автоматическое управление в технических системах» и 53 01 07 «Информационные технологии и управление в технических системах» дневной и вечерней форм обучения / А.В. Павлова, М.К. Хаджинов. – Мн.: БГУИР, 2003. – 48 с.

ISBN 985-444-495-3.

Приведены описание и порядок выполнения четырёх лабораторных работ по курсу «Математические основы теории систем». Содержание работ определено рабочей программой курса в соответствии с учебными планами специальностей 53 01 03 «Автоматическое управление в технических системах» и 53 01 07 «Информационные технологии и управление в технических системах».

По каждой работе изложены краткие теоретические сведения, представлены варианты заданий, приведены примеры обращения к пакету MATLAB, сформулированы контрольные вопросы.

УДК 519.85 (075.8)
ББК 22.18 я 73

ISBN 985-444-495-3

Содержание

Лабораторная работа № 1

Модели линейных систем и их элементов с постоянными параметрами

Цель работы: изучение моделей линейных непрерывных и дискретных систем и их элементов с постоянными параметрами, знакомство с временными и частотными характеристиками.

Краткие теоретические сведения

Система автоматического управления (САУ) представляет собой совокупность отдельных элементов или устройств, соединенных в замкнутый контур. Это могут быть различного рода датчики, усилители, двигатели, тахогенераторы и т.д. При математическом описании их называют звеньями САУ. Для любого объекта, имеющего входную переменную $u(t)$ и выходную $y(t)$, могут быть составлены модели двух типов:

1. Модели “вход—выход”, непосредственно связывающие $y(t)$ с $u(t)$.
2. Модели “в пространстве состояний”, связывающие $y(t)$ с $u(t)$ через промежуточные переменные $x(t)$ – вектора состояния, которые отражают внутреннюю структуру модели.

В моделях ”вход—выход” обычно используется дифференциальное уравнение n-го порядка

$$a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \ldots + a_1 \frac{dy}{dt} + a_0 y = b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \Lambda + b_1 \frac{du}{dt} + b_0 u, \qquad (1.1)$$

где u – входное воздействие; y – выходное воздействие; a_i, b_i – постоянные коэффициенты, определяемые параметрами схемы.

Для отдельного звена это уравнение первого или второго порядка. При объединении их в систему порядок дифференциального уравнения увеличивается.

В инженерной практике от дифференциальных уравнений переходят в область изображений по Лапласу.

Если в уравнении (1.1) полагать, что $u(0) = u'(0) = u^{(m-1)}(0) = 0$ и $y(0) = y'(0) = y^{(n-1)}(0) = 0$, то после применения преобразования Лапласа получим алгебраическое уравнение относительно изображений $a_n s^n y(s) + a_{n-1} s^{n-1} y(s) + \ldots$ $\ldots + a_1 s y(s) + a_0 y(s) = b_m s^m u(s) + b_{m-1} s^{m-1} u(s) + \ldots + b_1 s u(s) + b_0 u(s)$.

Передаточная функция звена (системы) $W(s)$ – это отношение изображения по Лапласу выходного сигнала $y(s)$ к изображению по Лапласу входного сигнала $u(s)$ при нулевых начальных условиях:

$$W(s)=\frac{y(s)}{u(s)}=\frac{b_m s^m+b_{m-1}s^{m-1}+...+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+...+a_1 s+a_0},\quad m\le n. \tag{1.2}$$

В системе MATLAB для описания моделей систем и действий над ними используется пакет прикладных программ *Control System Toolbox*. В пакете введен класс объектов, называемый *lti*–объекты — линейные с постоянными параметрами. При создании *lti*–объекта ему присваивается имя. Передаточная функция имеет две формы представления:

tf – форма, в которой передаточная функция (2.1) задается двумя векторами–строками, составленными из коэффициентов многочленов числителя и знаменателя в порядке убывания степеней *S*. Например, оператор *W* = *tf* ([2 1], [1 3 7]) создает объект *W* подкласса *tf*, соответствующий передаточной функции $W(S)=\frac{2S+1}{S^2+3S+7}$;

zpk – форма нулей, полюсов и коэффициента усиления, которая определяет передаточную функцию (1.2) выражением вида

$$W(S)=k\frac{(S-z_1)(S-z_2)...(S-z_m)}{(S-p_1)(S-p_2)...(S-p_n)}, \tag{1.3}$$

где k – коэффициент усиления; z_1, z_2,…, z_m – нули системы, т.е. корни многочлена числителя; p_1, p_2,…, p_n – полюсы системы, т.е. корни многочлена знаменателя.

Объективно нули полосы имеют отрицательную действительную часть. В *zpk*–форме передаточная функция описывается двумя векторами–строками и одним числом, задающими нули, полюсы и коэффициент усиления системы. Например, оператор *W* = *zpk*(0, [-1-*j*, -1+*j*, -2], 5) формирует *zpk*–объект, соответствующий передаточной функции $W(S)=\frac{5S}{(S+2)(S^2+2S+2)}$.

При отсутствии нулей на их место записывается знак "пусто" [ ].

zpk–форма информативней формы *tf* для человека и удобней для вычислений на ЭВМ.

При описании элементов и систем кроме входных $u(t)$ и выходных $y(t)$ переменных можно выделить некоторые промежуточные переменные $x(t)$, которые связаны с внутренней структурой системы и называются переменными состояния. В параметрах пространства состояний система n-го порядка с одним входом и одним выходом описывается системой уравнений

$$\begin{cases} \dot{x}=Ax+Bu, \\ y=Cx+Du. \end{cases}$$

Здесь A – квадратная матрица порядка n, элементы которой определяются коэффициентами дифференциального уравнения, В – вектор-столбец $[n\cdot 1]$ постоянных коэффициентов, C – вектор–строка $[1\cdot n]$ постоянных коэффициентов, D – одноэлементная матрица.

Описание системы в параметрах пространства состояний предусматривает переход от дифференциального уравнения n-го порядка к системе из n

дифференциальных уравнений первого порядка и алгебраическому уравнению вход—состояние—выход.

ss–форма представления системы используется для формирования модели в пространстве состояний. Оператор $W = ss(A,B,C,D)$ описывает объект W четверкой матриц A, B, C, D, которые должны иметь согласованные размеры.

Функция canon позволяет сформировать две канонические *ss*–модели систем в пространстве состояний. **Функция cw=canon(w, ´modal´)** формирует *модальную форму* уравнений состояния, в которой матрица A является диагональной, а элементами главной диагонали являются действительные собственные значения матрицы A. Паре комплексно-сопряженных собственных значений соответствует блок размером 2×2, также расположенный на диагонали этой матрицы. Для системы с собственными значениями (λ_1, $a \pm jb$, λ_2) модальная матрица A имеет вид

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & a & b & 0 \\ 0 & -b & a & 0 \\ 0 & 0 & 0 & \lambda_2 \end{bmatrix}.$$

Для системы с кратными собственными значениями функцию *canon* использовать нельзя.

Функция cw=canon(w, ´companion´) формирует *присоединенную каноническую форму* системы W с характеристическим уравнением $S^n+a_{n-1}S^{n-1}+\ldots$
$\ldots+a_1S+a_0=0$. Матрица A присоединенной канонической формы имеет вид

$$A = \begin{bmatrix} 0 & 0 & 0 & \text{K} & 0 & -a_0 \\ 1 & 0 & 0 & \text{K} & 0 & -a_1 \\ 0 & 1 & 0 & \text{K} & 0 & -a_2 \\ \Lambda & \Lambda & \Lambda & \Lambda & \Lambda & \Lambda \\ 0 & 0 & 0 & \text{K} & 0 & -a_{n-2} \\ 0 & 0 & 0 & \text{K} & 1 & -a_{n-1} \end{bmatrix}.$$

Каноническим формам соответствуют определенные структуры моделей.

Присоединенная каноническая форма представляет собой последовательную цепочку интеграторов (в матрице А диагональ единиц над или под главной диагональю), охваченных отрицательными обратными связями, исходящими из одной точки (столбцовая форма) или сходящимися в одной точке (строчная форма). Коэффициенты a_i обратных связей являются коэффициентами характеристического уравнения, образующими столбец или строку матрицы А.

Модальная каноническая форма представляется параллельно соединенными интеграторами, каждый из которых охвачен обратной связью. Коэффициент обратной связи равен значению полюса p_i и совпадает с

собственным значением λ_i матрицы A. Коэффициент передачи каждой из параллельных цепей r_i. Это соответствует разложению передаточной функции $W(s)$ на сумму простых дробей $W(s)=\sum_i \frac{r_i}{S-p_i}$ с помощью команды residue.

Поскольку существует три подкласса *lti*-моделей: *tf*, *zpk* и *ss*, имеется возможность для перехода от одного подкласса модели к другой. При необходимости Matlab сам производит такое преобразование. Вычисления целесообразно производить в форме *ss*, просматривать результаты — в форме *zpk*. Если сформирована некоторая *tf*–модель с именем *W*, то для преобразования в *zpk*-форму используется оператор *zpk*(*W*), а для преобразования в *ss*-форму используется оператор *ss*(*W*), т.е. команды создания и преобразования подклассов совпадают.

Извлечение числовых значений параметров модели объекта определённого подкласса производится следующим образом.

Для одномерных систем:

функция [chisl, znam] = tfdata(w, 'v') возвращает коэффициенты полиномов числителя и знаменателя в виде числовых значений векторов–строк;

функция [z, p, k] = zpkdata(w, 'v') возвращает нули, полюсы и коэффициент усиления *lti* – объекта *W* в виде числовых значений векторов–столбцов;

функция [A, B, C, D]=ssdata(W) возвращает значения четверки матриц *A*, *B*, *C*, *D* для *lti*–объекта с именем *W*. Для дискретных систем к возвращаемым параметрам в квадратных скобках добавляется период дискретности *Ts*.

В системах, работающих под управлением ЭВМ, управление происходит периодически с периодом дискретизации *Ts*. Процессы на входе и выходе рассматриваются только в дискретные равноотстоящие моменты времени. Обозначают их $u(nTs)$ и $y(nTs)$, где *Ts* – расстояние между соседними значениями аргумента, а *n* – целое число (n=0,1,2,…). При *Ts* = 1 эти функции обозначают сокращенно $u(n)$ или $y(n)$.

Системы, в которых взаимосвязь между входом и выходом определяется только в равноотстоящие моменты времени $t = nTs$, описываются разностными уравнениями. Разностные уравнения в пространстве состояний связывают текущие значения переменных $x(n)$ с предыдущими $x(n\text{-}1)$ следующим образом:

$$\begin{cases} x(n) = A_d x(n-1) + B_d u(n-1), \\ y(n) = C_d x(n) + D_d u(n), \end{cases}$$

где n = 1, 2, 3,…; *Ad*, *Bd*, *Cd*, *Dd* — матрицы постоянных коэффициентов для дискретной системы.

Переход от непрерывной модели *Sn* к дискретной *Sd* и обратно с периодом дискретизации *Ts* производится операторами *c2d* и *d2c*: $Sd = c2d(Sn, Ts)$; $Sn = d2c(Sd, Ts)$.

При описании дискретных систем комплексная переменная *S* в выражениях (1.1) и (1.2) заменяется на *z*.

Для формирования дискретных моделей с заданным периодом дискретности к входным аргументам функций *tf*, *zpk* и *ss* необходимо добавить период дискретности *Ts*, измеряемый в секундах. Например: *W* = *tf* (*chisl*, *zman*, *Ts*); *W* = *zpk*(*z*, *p*, *k*, *Ts*); *W* = *ss*(*A*, *B*, *C*, *D*, *Ts*). Здесь аргументы функций *tf*, *zpk*, *ss* существенно отличаются от аргументов функций при формировании непрерывных объектов. Если аргумент *Ts* не указан, то система формируется как непрерывная.

К временным характеристикам линейных элементов и систем относятся две характеристики: переходная и импульсная переходная (весовая) функции.

Переходная функция $h(t)$ – это функция, определяющая изменение выходной величины системы (или отдельного элемента) при воздействии на входе единичного ступенчатого воздействия $1(t)$ и при нулевых начальных условиях.

Единичная ступенчатая функция описывается следующим образом:

$$1(t) = \begin{cases} 1, & \text{при} \quad t = 0, \\ 0, & \text{при} \quad t \neq 0, \end{cases}$$ и имеет вид, приведенный на рис. 1.1, а.

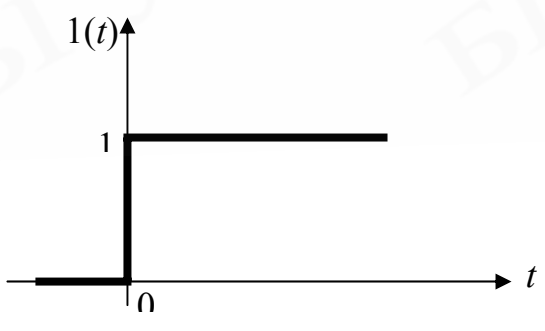
Команда *step*(*W*) строит график переходной функции для *lti*–модели *W*. Эта модель может быть непрерывной и дискретной. Команда *step*(W_1,W_2,…,W_n) позволяет на одном графике построить переходные функции для нескольких моделей W_1, W_2, … W_n. Продолжительность моделирования можно задать, или же она определяется автоматически так, чтобы отобразить основные особенности переходных процессов.

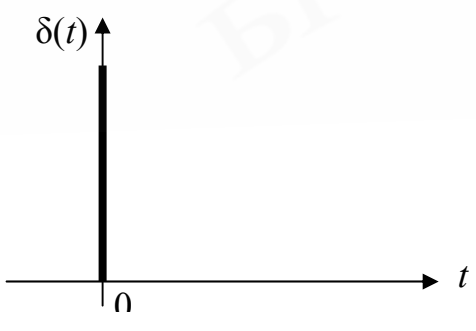
Импульсная переходная функция (функция веса) $w(t)$ – это функция, определяющая изменение выходной величины системы (или отдельного элемента) при воздействии на входе дельта–функции $\delta(t)$ и при нулевых начальных условиях.

Дельта-функция описывается следующим образом: $\delta(t) = \begin{cases} \infty, & \text{при } t = 0, \\ 0, & \text{при } t \neq 0, \end{cases}$

и имеет вид, приведенный на рис. 1.1,б. Интеграл от $\delta(t)$ равен единице.

Переходная и импульсная переходная характеристики связаны между собой следующим образом: $w(t) = \dfrac{dh(t)}{dt}$; $h(t) = \int_0^t w(\tau)d\tau$.





а б

Рис. 1.1. Единичная ступенчатая функция 1(t) (а) и дельта–функция $\delta(t)$ (б)

Команда *impulse*(*W*) строит график импульсной переходной функции для *lti*–модели *w*. Эта модель может быть непрерывной и дискретной. Команда *impulse*(W_1, W_2,…, W_n) позволяет на одном графике построить импульсные переходные функции для нескольких моделей W_1, W_2,…, W_n.

Частотные характеристики определяют динамические свойства звеньев при воздействии на них гармонических сигналов. Если входное воздействие линейной стационарной системы (отдельного элемента) является гармонической функцией вида $u(t) = A\cos(\omega t)$, то выходной сигнал после окончания переходного процесса представляет собой гармоническую функцию той же частоты, но отличающуюся в общем случае по амплитуде и фазе, т.е. $y(t)=B\cos(\omega t+\varphi)$. Комплексная передаточная функция $W(j\omega)$ – это отношение преобразования Фурье выходного гармонического сигнала к преобразованию Фурье входного гармонического сигнала: $W(j\omega)=\dfrac{Y(j\omega)}{U(j\omega)}$,

$$Y(j\omega)=\int_0^\infty y(t)e^{-j\omega t}dt,\ U(j\omega)=\int_0^\infty u(t)e^{-j\omega t}dt.$$

Формально-частотные характеристики получаются из передаточной функции $W(s)$ при замене $s=j\omega$, где ω – угловая частота, имеющая размерность [рад/с].

Функция $W(j\omega)$, зависящая от комплексной переменной, называется амплитудно-фазочастотной характеристикой (АФЧХ), она определяет изменение амплитуды и фазы выходного сигнала системы (отдельного элемента) в установившемся режиме по отношению к входному гармоническому воздействию.

Комплексная передаточная функция $W(j\omega)$ может быть представлена в виде: $W(j\omega)=P(\omega)+jQ(\omega)=A(\omega)e^{j\varphi(\omega)}$, где $A(\omega)=|W(j\omega)|=\sqrt{P^2(\omega)+Q^2(\omega)}$,

$$\varphi(\omega)=\operatorname{arctg}\frac{Q(\omega)}{P(\omega)}.$$

$A(\omega)=|W(j\omega)|$ называется амплитудно-частотной характеристикой (АЧХ), она показывает, во сколько раз амплитуда выходного гармонического сигнала отличается от амплитуды входного в зависимости от частоты сигнала. $\varphi(\omega)$ называется фазовой частотной характеристикой (ФЧХ) и определяет сдвиг фазы между выходным и входным сигналами в зависимости от частоты.

Команда *nyquist*(*W*) строит на экране график годографа $W(j\omega)$ в координатах $P(\omega)$–$Q(\omega)$.

Команда *bode*(*W*) строит АЧХ и ФЧХ с логарифмическим масштабом по оси частот, по оси ординат откладываются значения $A(\omega)$, представленные в децибелах $L(\omega)=20lgA(\omega)$ [дБ]. Значения фазы представлены в градусах.

Команда *bode*(W_1, W_2,…, W_n) позволяет построить частотные характеристики для нескольких *lti*–моделей на одном графике.

Между переходной $h(t)$, частотной $L(\omega)$ характеристиками и передаточной функцией $W(s)$ одного и того же объекта существует однозначная связь, сформулированная в теореме о предельных значениях. Согласно этой теореме $\lim\limits_{t\to\infty} h(t) = \lim\limits_{\omega\to 0} L(\omega) = \lim\limits_{s\to 0} W(s)$ и $\lim\limits_{t\to 0} h(t) = \lim\limits_{\omega\to\infty} L(\omega) = \lim\limits_{s\to\infty} W(s)$.

В инженерной практике принято строить асимптотические логарифмические частотные характеристики (ЛЧХ) в двойных логарифмических шкалах. В таких шкалах все степенные функции становятся линейными с наклоном, равным показателю степени. Асимптотические ЛАЧХ изображаются как кусочно-цельностепенные функции, т.е. состоящие из отрезков прямых линий, проведенных под определенным наклоном, кратным $20\frac{\text{дБ}}{\text{дек}}$. Изменение наклона происходит на частотах, равных по модулю нулям и полюсам передаточной функции, поэтому асимптотические ЛАЧХ удобно строить по *zpk*–форме. На высоких частотах сказывается влияние всех нулей и полюсов, поэтому $L(\omega)=K\cdot\omega^{(m-n)}$, где m – порядок числителя, n – порядок знаменателя.

Порядок выполнения работы

Задание 1

В табл. 1.1 приведены названия и дифференциальные уравнения некоторых типов звеньев.

Таблица 1.1

№	Название звена	Дифференциальные уравнения
1	Интегрирующее	$y(t) = \int_0^t ku(t)dt$ или $\frac{dy(t)}{dt} = ku(t)$
2	Апериодическое	$T\frac{dy}{dt} + y(t) = ku(t)$
3	Интегродифференцирующее	$T_2\frac{dy}{dt} + y(t) = k\left[T_1\frac{du}{dt} + u(t)\right]$
4	Колебательное	$T^2\frac{d^2y}{dt^2} + 2\xi T\frac{dy}{dt} + y(t) = ku(t)$

Здесь k – коэффициент усиления, T – постоянная времени, ξ – коэффициент демпфирования (колебательности).

Параметры k, T и ξ выбрать самостоятельно из следующих интервалов: $0{,}01 \le T \le 2$; $5 \le k \le 30$; $0{,}2 \le \xi \le 1$. Для каждого звена выполнить следующие действия.

1. Записать передаточную функцию и tf-модель. Преобразовать *tf*–модель в формы *zpk* и *ss*, выяснить связь форм на примерах.

2. Использовать цикл *for* для изменения параметров звена и построения семейства характеристик (см. пример 1).

2.1. Построить переходную характеристику, используя команду *step*(*W*). Выяснить влияние параметров звена на вид переходной характеристики, осуществляя построение $h(t)$ для нескольких моделей в одном окне. Фиксировать длительность переходного процесса и величину установившегося значения сигнала на выходе как функцию параметра звена.

2.2. Выяснить влияние параметров звена на вид амплитудно-частотной и фазочастотной характеристик, осуществляя построение $L(\omega)$ и $\varphi(\omega)$ для нескольких моделей в одном окне и используя цикл *for*. Определить влияние на вид частотных характеристик дифференцирующих или интегрирующих свойств звена.

2.3. Проследить связь между переходной, частотной характеристиками и передаточной функцией, определяемыми теоремой о предельных значениях.

2.4. Непрерывный объект преобразовать в дискретный. Сравнить характеристики непрерывного и дискретного объектов (переходную, ЛАЧХ), определить их различие и влияние периода дискретизации *Ts*, задавая изменение *Ts* в цикле *for* (см. пример 1).

Пример 1

```
% ar2d – колебательное звено. Изменяется коэффициент демпфирования ξ .
% Циклы характеристик с параметром m
% Дискретная модель, изменяется Ts - период дискретизации.
% Для исследования непрерывной модели закрыть строку формирования
% признака для преобразования в дискретный объект
clear,close all;        % Очистить память. Закрыть все окна
k=25; T=2; c='s'; % Параметры объекта, признак непрерывного объекта
c='z';  % признак для преобразование в дискретный объект
txt1=['Колебат.зв. W(',c,') = ('];          % часть заголовка
for i=-6:0;  % цикл по переменной i
  m=2^(i/2);          % m - изменяемый в цикле множитель для W(s)
  ch=[k];      %числитель передаточной функции W(s)
  zn=[T^2  2*T*m  1];      % знаменатель передаточной функции W(s)
  sk=tf(ch,zn);        % передаточная функция W(s)
 if c=='z'; % если задан признак преобразования в дискретный объект
   Ts=m*T*2; % вычисляется период дискретизации Ts
```

```
   skd=c2d(sk,Ts); % объект sk преобразуется в дискретный skd
   [ch,zn]=tfdata(skd,'v'); % числитель и знаменатель дискрет.объекта
end % конец  преобразования в дискретный объект
txt2=[txt1,poly2str(ch,c),')/(',poly2str(zn,c),')'];%заголовок
figure(1); % открыть окно № 1
hold on; % открыть режим сохранения прежних графиков в окне № 1
step(sk,T*9); % добавить переходную функцию объекта sk
title([txt2]); % нанести заголовок
hold off; % закрыть режим сохранения прежних графиков в окне № 1
pause(0), % сделать паузу, чтобы нарисовать график
figure(2);  % открыть окно № 2
hold on; % открыть режим сохранения прежних графиков в окне № 2
bode(sk); % добавить ЛАХ объекта sk
title([txt2]);  % нанести заголовок
hold off; % закрыть режим сохранения прежних графиков в окне № 2
pause, % сделать паузу, чтобы нарисовать график
end; % и так до ... окончания цикла по переменной i
zoom on; % включить режим масштабирования графика в окне № 2
```

Задание 2

Для заданной системы с передаточной функцией $W(s) = \dfrac{b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0}$ выполнить следующие действия:

1. Записать tf-модель, преобразовать *tf*–модель в формы *zpk* и *ss*.
2. Для ss-модели в параметрах пространства состояний составить систему уравнений $\begin{cases} \dot{x} = Ax + Bu, \\ y = Cx + Du, \end{cases}$ представить ее в скалярной форме и преобразовать блок-схему модели.
3. Промоделировать структуру в системе *Simulink*. Наблюдать изменение выходного сигнала $y(t)$ и переменных состояния $x1$, $x2$ и $x3$ при подаче на вход сигнала $u(t)=1(t)$, (*step*).
4. Преобразовать *ss*–модель в канонические формы с помощью команды *cw = canon*(*w*, *´modal´*), *cw=canon*(*w*, *´companion´*). Записать модели в скалярной форме и составить блок–схемы.
5. Непрерывный объект преобразовать в дискретный, сравнить характеристики непрерывного и дискретного объектов (переходную, ЛАХ), определить их различие и влияние периода дискретизации *Ts*, используя цикл *for*.

Контрольные вопросы

1. Определение передаточной функции. Передаточные функции различных соединений звеньев.

2. Определение временных характеристик. Связь между временными характеристиками.

3. Понятие о частотных характеристиках. Физический смысл АЧХ, ФЧХ и АФЧХ.

4. Правила построения асимптотических ЛАЧХ с использованием *zpk*–формы.

5. Канонические формы уравнений состояния.

6. Каковы признаки дифференцирующих, интегрирующих и позиционирующих звеньев в передаточных функциях, переходных и частотных характеристик.

7. Каковы различия непрерывных и дискретных объектов в переходных и частотных характеристиках?

Лабораторная работа № 2

Линейное программирование

Цель работы: изучение методов решения и анализа задач линейного программирования.

Краткие теоретические сведения

К задачам линейного программирования (ЛП) относятся задачи оптимизации, в которых отыскивается экстремум (максимум или минимум) заданной линейной функции многих переменных при наличии линейных ограничений в форме системы равенств и неравенств. В наиболее общем виде задача ЛП формулируется следующим образом: найти значения переменных $x_i \geq 0$, $i = \overline{1,n}$, доставляющие экстремум линейной функции цели $F(x) = \sum_{i=1}^{n} c_i x_i = C^T x$ при следующих линейных ограничениях:

$$\sum_{r=1}^{n} a_{ir} x_r \leq b_i, \quad i = \overline{1,p}; \tag{2.1}$$

$$\sum_{r=1}^{n} a_{jr} x_r \geq b_j, \quad j = \overline{p+1,q}; \tag{2.2}$$

$$\sum_{r=1}^{n} a_{kr} x_r = b_k, \quad k = \overline{q+1,m}. \tag{2.3}$$

Здесь x – n-мерный вектор переменных; C^T – транспонированный n-мерный вектор постоянных коэффициентов; a_{ji} и b_j – постоянные коэффициенты; m – общее число ограничений.

Переменные $x_i \geq 0$, удовлетворяющие совокупности заданных ограничений (2.1) – (2.3), представляют собой допустимое решение задачи и называются планом задачи. Допустимое решение, доставляющее экстремум функции цели $F(x)$, называется оптимальным решением или оптимальным планом.

Канонической формой записи задачи ЛП называют такую форму записи, в которой ограничения типа неравенств отсутствуют. Общая форма записи задачи ЛП приводится к канонической путем введения дополнительных неотрицательных переменных. В зависимости от знака неравенств (2.1) и (2.2) будут иметь место следующие выражения:

$$\begin{cases} \sum_{r=1}^{n} a_{ir} x_r + x_{n+i} = b_i, \ \ i = \overline{1, p}, \\ \sum_{r=1}^{n} a_{jr} x_r - x_{n+j} = b_j, \ \ j = \overline{p+1, q}, \\ \sum_{r=1}^{n} a_{kr} x_r = b_k, \ \ k = \overline{q+1, m}. \end{cases} \qquad (2.4)$$

Система ограничений (2.4) представляет собой систему из m линейных алгебраических уравнений с $N = n+q$ неизвестными, где q – количество введенных дополнительных переменных. Введение дополнительных переменных не изменяет функции цели и не влияет на оптимальное решение задачи.

Задача ЛП имеет смысл только при $N > m$, т.е. в том случае, когда общее число неизвестных больше количества ограничений. Одним из наиболее эффективных методов численного решения задач ЛП является симплекс–метод. Решение задач складывается из двух основных этапов. На первом этапе находят одно из решений, удовлетворяющих системе ограничений. Системы вида (2.4), в которых $N > m$, называются неопределенными. Они приводятся к определенным ($N = m$) путем приравнивания к нулю $N–m$ каких-либо переменных. Эти переменные называются свободными или небазисными. При этом остается система m уравнений с m неизвестными, которая имеет решение, если определитель системы отличен от нуля. Это решение называется базисным.

В системе из m уравнений с N неизвестными общее число базисных решений при $N>m$ определяется числом сочетаний $C_N^m = \dfrac{N!}{m!(N-m)!}$.

Базисное решение, в котором все $x_i \geq 0$, $i = \overline{1,m}$, называется допустимым базисным решением или планом. Допустимое базисное решение, доставляющее экстремум функции цели $F(x)$, называется оптимальным решением или оптимальным планом. Первый этап завершается нахождением допустимого базисного решения, хотя бы и неудачного. На втором этапе по специальным правилам переходят от одного решения к другому, более близкому к оптимальному, и так до тех пор, пока задача не будет решена.

Симплекс-метод дает оптимальную процедуру перебора базисных решений и обеспечивает сходимость к экстремальной точке за конечное число шагов.

Важную роль в математическом программировании имеет понятие двойственности. Рассмотрим две задачи линейного программирования:

$$\max \{F(x) = C^{\mathrm{T}}x \mid Ax \le B,\ x_{\mathrm{i}} \ge 0,\ i=\overline{1,n}\} \tag{2.5}$$

и

$$\min \{F(y) = B^{\mathrm{T}}y \mid A^{\mathrm{T}}y \le C,\ y_{\mathrm{j}} \ge 0,\ j=\overline{1,m}\}. \tag{2.6}$$

Задачу (2.5) называют прямой, а связанную с ней задачу (2.6) — двойственной, и они образуют симметричную пару двойственных задач. Число переменных двойственной задачи равно количеству ограничений прямой. Кроме того, при переходе от прямой задачи к двойственной векторы B и C меняются местами, матрица A коэффициентов система ограничений прямой задачи транспонируется, а знаки неравенств в ограничениях меняются на противоположные. Смысл экстремума $F(x)$ противоположен смыслу экстремума $F(y)$. Связь между задачами (2.5) и (2.6) взаимна, т.е. если прямой считать задачу (2.6), то в качестве двойственной ей будет соответствовать задача (2.5). Возможность перехода от прямой задачи к двойственной (и наоборот) устанавливается теоремой двойственности: если одна из задач (2.5) или (2.6) имеет оптимальное решение, то и другая также имеет оптимальное решение, причем оптимальные значения функций цели прямой и двойственной задач совпадают, т.е. $\max F(x) = \min F(y)$.

Если среди ограничений прямой задачи имеются равенства или на некоторые переменные не наложено условие неотрицательности, то, построив двойственную ей задачу, получим пару несимметричных двойственных задач:

$$\begin{array}{l|l}
F(x) = \sum\limits_{i=1}^{n} c_i x_i (\max) & F(y) = \sum\limits_{j=1}^{m} b_j y_j (\min) \\
\sum\limits_{i=1}^{n} a_{ji} x_i \le b_j,\ \ j = \overline{1, m_1} & \sum\limits_{j=1}^{m} a_{ij} y_j \ge c_i,\ \ i = \overline{1, n_1} \\
\sum\limits_{i=1}^{n} a_{ji} x_i = b_j,\ \ j = \overline{m_1 + 1, m} & \sum\limits_{j=1}^{m} a_{ij} y_j = c_i,\ \ i = \overline{n_1 + 1, n} \\
x_i \ge 0,\ \ i = \overline{1, n_1},\ \ n_1 \le n & y_j \ge 0,\ \ j = \overline{1, m_1},\ \ m_1 \le m.
\end{array}$$

При этом выполняются следующие правила:

1. Если на переменную x_{i} прямой задачи наложено условие неотрицательности, то i-е условие системы ограничений двойственной задачи является неравенством и наоборот.

2. Если на переменную x_i прямой задачи не наложено условие неотрицательности, то i-е ограничение двойственной задачи записывается в виде строгого равенства.

3. Если в прямой задаче имеются ограничения равенства, то на соответствующие переменные двойственной задачи не накладывается условие неотрицательности.

Линейное программирование находит широкое применение при решении многих практических задач организационно–экономического управления. Цель, как правило, заключается в том, чтобы максимизировать прибыль либо минимизировать расходы. Рассмотрим задачу рационального использования ресурсов. Пусть предприятие располагает ресурсами b_1, b_2,…, b_m, которые могут использоваться для выпуска n видов продукции. Известны норма потребления j-го ресурса на производство единицы i-й продукции – a_{ij}, а также прибыль от реализации единицы i-го вида продукции c_i (i=1,n; j=1,m). Требуется определить объем производства продукции каждого вида x_i^*, максимизирующий суммарную прибыль предприятия $F(x)=\sum_{i=1}^{n} c_i x_i$, при этом расход ресурсов не должен превышать их наличия. Математическая модель задачи имеет вид

$$\max\left\{ F(x)=\sum_{i=1}^{n} c_i x_i \,\middle|\, \sum_{i=1}^{n} a_{ij} x_i \le b_j,\ \ j=\overline{1,m};\ x_i \ge 0,\ \ i=\overline{1,n} \right\}. \qquad (2.7)$$

Под чувствительностью модели понимается зависимость оптимального решения от изменения параметров исходной задачи.

При этом можно выяснить:

а) насколько можно увеличить запас некоторого ресурса, чтобы улучшить оптимальное значение F;

б) насколько можно сократить запас некоторого ресурса при сохранении полученного оптимального значения F;

в) увеличение объема какого из ресурсов наиболее выгодно;

г) какому из ресурсов отдать предпочтение при вложении дополнительных средств;

д) в каких пределах допустимо изменение коэффициентов целевой функции, при котором не происходит изменение оптимального решения.

Ограничения, проходящие через точку оптимума, называются активными или связывающими. Ресурсы, с которыми ассоциируются эти ограничения, относятся к разряду дефицитных. Остальные ресурсы недефицитны, а соответствующие им ограничения неактивные или несвязывающие. Сокращение объема дефицитного ресурса никогда не улучшает значения целевой функции. Анализ на чувствительность придает модели динамичность, свойственную реальным процессам.

Сформулируем задачу, двойственную рассматриваемой задаче рационального использования ресурсов. Пусть некоторая организация может

закупить все ресурсы b_j, которыми располагает предприятие. Необходимо определить оптимальные цены y_j^* $(j=\overline{1,m})$ на единицу этих ресурсов при условии, что покупатель стремится минимизировать общую оценку ресурсов. При этом общая ценность ресурсов должна быть не меньше прибыли, которую может получить предприятие при организации собственного производства. Математическая модель задачи имеет вид

$$\min\left\{F(y)=\sum_{j=1}^{m}b_j y_j \,\middle|\, \sum_{j=1}^{m}a_{ji}y_j \ge C_i,\ i=\overline{1,n};\ y_j\ge 0,\ j=\overline{1,m}\right\}. \qquad (2.8)$$

Пока прибыль предприятия (задача 2.7) меньше общей ценности ресурсов (задача 2.8), решение обеих задач будет неоптимальным. Оптимум достигается в случае, когда прибыль становится равной общей ценности ресурсов, т.е. max $F(x)$ = min $F(y)$.

Двойственные оценки являются:

1) показателем дефицитности ресурсов и продукции. Величина y_j^* является оценкой j-го ресурса. Чем больше значение оценки y_j^*, тем выше дефицитность ресурса. Для недефицитного ресурса $y_j^*=0$;

2) показателем влияния ограничений на значение целевой функции. Значения переменных y_j^* численно равны частным производным $\partial F\max/\partial bj$ для исходной задачи. При незначительном приращении Δb_j y_j^* является точной мерой влияния ограничений на целевую функцию. Поэтому представляет практический интерес определение предельных значений ограничений (нижней и верхней границ), в которых оценки остаются неизменными;

3) показателем эффективности производства отдельных видов продукции с позиций критерия оптимальности. Его суть заключается в том, что в оптимальный план может быть включена лишь та продукция j-го вида, для которой выполняется условие $\sum_{j=1}^{m}a_{ji}y_j^* < c_i$;

4) инструментом сопоставления суммарных условных затрат и результатов. Это свойство следует из принципа двойственности, в котором устанавливается связь между значениями функций цели прямой и двойственной задачи.

В систему MATLAB входит пакет прикладных программ для решения задач оптимизации.

Подпрограмма *LP* предназначена для минимизации целевой функции $F(x)$ при ограничениях $Ax \le B$, т.е. $\min\left\{F(x)=\sum_{i=1}^{n}c_i x_i \,\middle|\, Ax\le B,\ x\ge 0\right\}$.

Ограничения на знак приводятся к виду $-x_i \le 0$ и добавляются к основным ограничениям задачи при вводе данных.

При подготовке к вводу данных следующей задачи:
$\max\{F(x) = -2x_1+3x_2 \mid 4x_1+3x_2 \le 12;\ x_1+2x_2 \ge 2;\ x_1 \ge 0,\ x_2 \ge 0\}$ условие преобразуется следующим образом:
$\min\{F(x) = 2x_1-3x_2 \mid 4x_1 + 3x_2 \le 12;\ \ -x_1-2x_2 \le -2;\ \ -x_1 \le 0,\ -x_2 \le 0\}$.

Обращение к подпрограмме LP осуществляется следующим образом:

F=[2 -3];
A=[4 3;-1 –2;-1 0;0 -1];
B=[12;-2;0;0];
x=*LP*(*F*, *A*, *B*).

Порядок выполнения работы

1. Решить следующую задачу ЛП.

Для изготовления четырех видов продукции (А,Б,В и Г) используются три вида ресурсов (I,II,III). Нормы расхода ресурсов на единицу продукции каждого вида, запасы ресурсов и прибыль от реализации единицы продукции приведены в табл. 2.1.

Определить, какое количество отдельных видов продукции необходимо производить, чтобы получить максимальную прибыль.

Таблица 2.1

Вид ресурса	Запас ресурса	**Норма расхода на единицу продукции**			
		А	Б	В	Г
I	220	2	1	2	1
II	60	1	0	2	1
III	320	1	2	1	1
Прибыль		4	2	3	5

Математическая модель задачи имеет вид:

$$F(x) = 4x_1+2x_2+3x_5+5x_4 \text{ (max)},$$

$$\begin{cases} 3x_1 + x_2 + 2x_3 + x_4 \le 220, \\ x_1 + 2x_3 + x_4 \le 160, \\ x_1 + 2x_3 + x_3 + x_4 \le 320, \\ x_i \ge 0, \quad i = \overline{1,4}, \end{cases}$$

или $\max\{F(x) = C^{\mathrm{T}}x | Ax \le B, x \ge 0\}$.

Различные варианты значений матриц постоянных коэффициентов C^{T}, A, B приведены в табл. 2.2.

Таблица 2.2

№ варианта	1	2	3	4	5	6	7	8	9
C^{T}	[1352]	[3251]	[6413]	[2137]	[5134]	[5321]	[4312]	[6413]	[1679]
A	$\begin{bmatrix}1232\\4105\\6013\end{bmatrix}$	$\begin{bmatrix}4106\\2132\\0245\end{bmatrix}$	$\begin{bmatrix}1053\\2340\\1314\end{bmatrix}$	$\begin{bmatrix}4112\\2025\\1341\end{bmatrix}$	$\begin{bmatrix}3501\\2311\\0612\end{bmatrix}$	$\begin{bmatrix}4640\\1205\\2123\end{bmatrix}$	$\begin{bmatrix}1113\\0214\\4140\end{bmatrix}$	$\begin{bmatrix}2230\\1245\\0316\end{bmatrix}$	$\begin{bmatrix}3131\\4220\\0351\end{bmatrix}$
B	$\begin{bmatrix}100\\250\\80\end{bmatrix}$	$\begin{bmatrix}80\\240\\300\end{bmatrix}$	$\begin{bmatrix}70\\120\\190\end{bmatrix}$	$\begin{bmatrix}250\\30\\120\end{bmatrix}$	$\begin{bmatrix}180\\200\\100\end{bmatrix}$	$\begin{bmatrix}90\\180\\320\end{bmatrix}$	$\begin{bmatrix}210\\300\\90\end{bmatrix}$	$\begin{bmatrix}280\\100\\210\end{bmatrix}$	$\begin{bmatrix}170\\90\\350\end{bmatrix}$

№ варианта	10	11	12	13	14	15
C^{T}	[4516]	[7132]	[1734]	[3574]	[5432]	[6127]

A	$\begin{bmatrix}2304\\5120\\2311\end{bmatrix}$	$\begin{bmatrix}5023\\1241\\0342\end{bmatrix}$	$\begin{bmatrix}6401\\2130\\1212\end{bmatrix}$	$\begin{bmatrix}2424\\5103\\9231\end{bmatrix}$	$\begin{bmatrix}3412\\1240\\3112\end{bmatrix}$	$\begin{bmatrix}5132\\2214\\0312\end{bmatrix}$
B	$\begin{bmatrix}300\\120\\210\end{bmatrix}$	$\begin{bmatrix}150\\75\\210\end{bmatrix}$	$\begin{bmatrix}170\\320\\150\end{bmatrix}$	$\begin{bmatrix}85\\120\\280\end{bmatrix}$	$\begin{bmatrix}250\\310\\120\end{bmatrix}$	$\begin{bmatrix}130\\75\\220\end{bmatrix}$

В соответствии с вариантом задания составить математическую модель задачи и найти оптимальное решение, используя подпрограмму $LP(F, A, B)$.

2. Выяснить, как влияют правые части ограничений на оптимальное решение задачи. Для этого вначале определить, какие ресурсы являются дефицитными (в соответствующих ограничениях должно выполняться равенство $\sum_{i=1}^{n} a_{ji} x_i = b_j$). Увеличивать поочередно величину каждого из дефицитных ресурсов и наблюдать за увеличением значения функции цели. Результаты занести в табл. 2.3. Убедиться в том, что изменение запасов недефицитных ресурсов не улучшает функцию цели.

Таблица 2.3

Ресурс	Тип ресурса (дефицитный или недефицитный)	Максимальное изменение запаса ресурса	Максимальное изменение прибыли	y_K

Пусть y_k – ценность каждой дополнительной единицы дефицитного ресурса k.

$$y_к = \frac{\text{Максимальное приращение оптимального значения } F}{\text{Максимально допустимый прирост объема ресурса } K}.$$

Дополнительные вложения следует в первую очередь направлять на увеличение того ресурса ℓ, у которого ценность y_ℓ наибольшая.

3. Выяснить диапазон изменения отдельных коэффициентов целевой функции, при котором не происходит изменения оптимального решения.

4. Составить задачу, двойственную к исходной. Найти оптимальное решение двойственной задачи. Убедиться, что величина y_j^* является оценкой j-го ресурса. Чем больше значение оценки y_j^*, тем выше дефицитность ресурса. Для недефицитного ресурса $y_j^* = 0$.

5. Объяснить, что означает равенство значений целевой функции. Записать соответствие между переменными прямой и двойственной задач.

6. Решить прямую и двойственную задачи ЛП в соответствии с вариантом задания по курсовой работе.

Контрольные вопросы

1. Постановка задачи ЛП, основные особенности задач ЛП.
2. Приведение задач ЛП к канонической форме.
3. Что называется областью допустимых решений задачи ЛП, что она представляет собой геометрически?
4. Графический метод решения задач ЛП.
5. Какое решение системы линейных алгебраических уравнений называется базисным, допустимым базисным, оптимальным?
6. Сущность симплекс-метода. Как выбирается переменная, вводимая в базис, выводимая из базиса? Какой элемент называется ведущим?
7. Признаки оптимальности симплекс–таблицы.
8. Что понимается под чувствительностью модели?
9. Двойственная задача ЛП и правила ее построения.
10. Теорема двойственности. Смысл двойственных оценок.

Лабораторная работа № 3

Нелинейное программирование. Экстремальные задачи без ограничений

Цель работы: изучение конкретных методов, используемых при поиске безусловного экстремума функций многих переменных.

Краткие теоретические сведения

В нелинейном программировании рассматриваются задачи отыскания экстремума функций многих переменных, в которых функция цели или хотя бы одно ограничение нелинейны. Простейшие задачи нелинейного программирования – это задачи без ограничений.

Задача формулируется следующим образом: найти значения переменных x_1 , x_2,…, x_n , составляющие экстремум скалярной целевой функции $F(x_1,x_2,\ldots x_n)$, если ограничения на переменные отсутствуют. Удобно предположить, что функция F и ее производные существуют и непрерывны.

Рассмотрим некоторые фундаментальные понятия, используемые при поиске экстремума.

Градиент функции F – это вектор, составляющие которого равны частным производным функции $F(x)$ по соответствующим переменным

$\nabla F(x)^T = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \text{K}, \frac{\partial F}{\partial xn}\right]$. Направление вектора градиента совпадает с направлением наискорейшего возрастания функции.

Матрица вторых производных $\nabla^2 F(x)$ – это симметрическая квадратная матрица порядка *n* вторых частных производных функции *F*(*x*). Эту матрицу называют матрицей Гессе и обозначают $H(x_1) = \nabla^2 F(x)$. Ее элемент, расположенный на пересечении *i*–й строки и *j*–го столбца, равен $\frac{\partial^2 F}{\partial x_1 \partial x_j}$.

Рассмотрим необходимые и достаточные условия существования экстремума. Пусть $x^{\text{T}}=[x_1, x_2,\ldots, x_{\text{n}}]$ – вектор переменных. Для наличия в точке *x** локального минимума (максимума) необходимо, чтобы выполнялось равенство $\nabla F(x^*) = 0$ и матрица Гессе $H(x^*) = \nabla^2 F(x^*)$ была положительно (отрицательно) полуопределенной, т.е. $x^T Hx \geq 0$ ($x^T Hx \leq 0$).

Достаточным условием существования минимума (максимума) в точке *x** является положительная (отрицательная) определённость матрицы Гессе в этой точке, т.е. $x^T Hx > 0$ ($x^T Hx < 0$).

Методы решения задач безусловной оптимизации можно разделить на три класса:

1. Методы прямого поиска, основанные только на вычислении значений целевой функции.

2. Градиентные методы, в которых используются значения первых производных *F*(*x*), вычисляемых численно или по формулам (аналитически).

3. Методы второго порядка, в которых наряду с первыми производными используются также вторые производные функции *F*(*x*).

В ряде практических инженерных задач присутствует информация только о значениях целевой функции, поэтому прямые методы имеют важное значение, хотя требуют большего количества вычислений значений функции.

Метод Нелдера—Мида (Nelder—Mead). Метод Нелдера—Мида является одним из методов прямого поиска по симплексу. Суть его в следующем.

В области допустимых значений переменных выбирается начальная точка и оценивается значение целевой функции в определённых точках, окружающих базовую. “Наилучшая” из исследуемых точек принимается начальной на следующем шаге. В качестве точек эксперимента выбираются вершины симплекса. Симплекс в *N*-мерном пространстве представляет собой многогранник, образованный *N*+1 равноотстоящими друг от друга точками–вершинами. В случае двух переменных симплексом является равносторонний треугольник, в трёхмерном пространстве симплекс представляет собой треугольную пирамиду и т.д.

На рис. 3.1,а в виде равностороннего треугольника с вершинами x^0, x^1 и x^2 приведен симплекс для функции двух переменных. В каждой из вершин симплекса оценивается значение целевой функции и при поиске минимума определяется вершина, которой соответствует наибольшее значение функции.

Пусть это будет вершина x^1. Далее находится середина отрезка x^0 x^2 – (центр тяжести) x^C, и точка x^1 проецируется через неё в новую точку x^3, которая используется в качестве вершины при построении нового симплекса (x^0 x^2 x^3).

Поиск завершается, когда разности между значениями функции в вершинах становятся достаточно малыми или диаметр симплекса становится меньше заданной величины итерационной погрешности по переменным x.

Для построения симплекса, кроме начальной точки, необходимо задать масштабный множитель (коэффициент) α. При α=1 ребра симплекса имеют единичную длину.

В N-мерном пространстве координаты вершин вычисляются по формуле

$$x_i = \begin{cases} x_j^0 + \delta^1, & \text{если } j \neq i, \\ x_j^0 + \delta^2, & \text{если } j = i, \end{cases} \qquad i = \overline{1,N}, \qquad j = \overline{1,N}.$$

Здесь

$$\delta_1 = \left[\frac{(N+1)^{1/2} + N - 1}{N\sqrt{2}} \right] \alpha; \qquad \delta_2 = \left[\frac{(N+1)^{1/2} - 1}{N\sqrt{2}} \right] \alpha.$$

Пусть x^k – точка, подлежащая отражению. Центр тяжести остальных N точек расположен в точке $x_c = \frac{1}{N} \sum_i^N x_i, \quad i \neq k$.

Отражение вершины симплекса осуществляется вдоль прямой, проходящей через x^k, x_c, и задается формулой

$$x = x^k + \lambda (x_c - x^k). \qquad (3.1)$$

При λ = 0 получаем исходную точку x^k, при λ = 1 получаем центр тяжести x_C. Для того чтобы отражение было симметричным, для получения новой вершины $x_{НОВ}$ принимают λ = 2, тогда

$$x_{\text{нов}} = 2x_c - x^k{}_{\text{предыд.}}$$

Проиллюстрируем вычислительную схему метода на примере функции, зависящей от двух переменных.

Минимизировать $F(x) = x_1^2 - 2x_1 + x_2{}^2 - 4x_2 + 5$. Пусть $x^0 = [0,0]$ и λ = 2. Тогда

$$\delta_1 = \left[\frac{\sqrt{3}+1}{2\sqrt{2}} \right] \cdot 2 = 1{,}932; \qquad \delta_2 = \left[\frac{\sqrt{3}-1}{2\sqrt{2}} \right] \cdot 2 = 0{,}518.$$

На основании этих значений вычисляются координаты двух других вершин симплекса:

$$\begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix} = \begin{bmatrix} x_1^0 + \delta_1 \\ x_2^0 + \delta_2 \end{bmatrix} = \begin{bmatrix} 0 + 0{,}518 \\ 0 + 1{,}932 \end{bmatrix} = \begin{bmatrix} 0{,}518 \\ 1{,}932 \end{bmatrix};$$

$$\begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} x_1^0 + \delta_1 \\ x_2^0 + \delta_2 \end{bmatrix} = \begin{bmatrix} 0 + 1{,}932 \\ 0 + 0{,}518 \end{bmatrix} = \begin{bmatrix} 1{,}932 \\ 0{,}518 \end{bmatrix}.$$

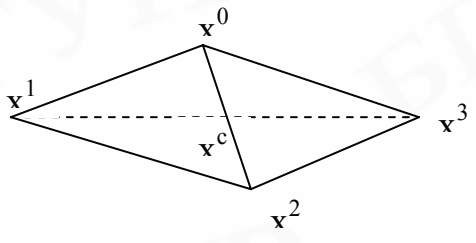
Целевая функция в вершинах симплекса определяется следующими величинами $F(x^0) = 5$; $F(x^1) = 0{,}237$; $F(x^2) = 3{,}066$. Максимальное значение функция цели имеет в вершине x^0, поэтому точку x^0 необходимо отразить относительно центра тяжести двух остальных вершин симплекса $x^c = \frac{1}{2}(x^1 + x^2)$, тогда новая точка x^3 при α=2 определится выражением $x^3 = 2x^c + x^0 = x^1 + x^2 - x^0$:

$$\begin{bmatrix} x_1^3 \\ x_2^3 \end{bmatrix} = \begin{bmatrix} 0{,}518 \\ 1{,}932 \end{bmatrix} + \begin{bmatrix} 1{,}932 \\ 0{,}518 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2{,}45 \\ 2{,}45 \end{bmatrix}.$$

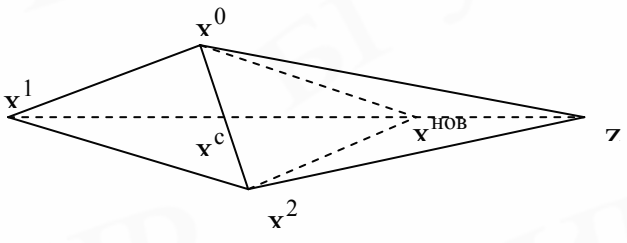
В полученной точке $F(x^3) = 2{,}3$, т.е. наблюдается уменьшение целевой функции.

В новом симплексе x^1, x^2 и x^3 наибольшее значение $F(x)$ соответствует точке x^2, поэтому её следует отразить относительно центра тяжести точек x^1 и x^3. Итерации продолжаются до тех пор, пока разности между значениями функции в вершинах становятся достаточно малыми.

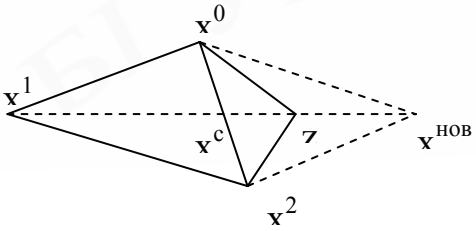
Для повышения эффективности метода используется возможность растяжения и сжатия симплекса. При этом в выражении (3.1) Нелдер и Мид рекомендуют использовать $\lambda = 3$ (растяжение) и $\lambda = 1{,}5$ (сжатие), как показано на рис. 3.1.



а



б



в

Рис. 3.1. Растяжение и сжатие симплекса:
а — нормальное отражение, $\lambda = 2$; б — растяжение, $\lambda = 3$; в — сжатие, $\lambda = 1{,}5$

К градиентным методам, изучаемым в работе, относятся:

1) метод наискорейшего спуска;
2) метод Дэвидона—Флетчера—Пауэла;
3) метод Бройдена—Флетчера—Гольдфарба—Шенно.

Во всех методах итерационная процедура поиска реализуется в соответствии с формулой $x^{k+1} = x^k + \lambda^k S^k$, где x^k – текущее приближение к решению x^*; $S^k = S(x^k)$ – направление движения в точке x^k; λ^k – параметр, характеризующий длину шага.

Способ определения S^k и λ^k на каждой итерации связан с особенностями применяемого метода.

Метод наискорейшего спуска (Steepest Descent). В методе наискорейшего спуска направление движения S при поиске минимума $F(x)$ задается вектором антиградиента $-\nabla F(x)$ функции $F(x)$ в рассматриваемой точке, т.е. $x^{k+1} = x^k - \lambda^k \nabla F(x^k)$.

Составляющие вектора градиента в точке x^k определяются значениями частных производных первого порядка функции $F(x)$ по соответствующим переменным, вычисленным в точке x^k:

$$\nabla F(x^k)^T = \left| \frac{dF(x^k)}{dx_1}; \quad \frac{dF(x^k)}{dx_2}, \quad ...., \frac{dF(x^k)}{dx_n} \right|.$$

Направление вектора градиента совпадает с направлением наискорейшего возрастания функции $F(x)$. Вектор $-\nabla F(x)$ называется антиградиентом, его направление совпадает с направлением наискорейшего убывания функции.

Предполагается, что компоненты градиента могут быть записаны в аналитическом виде или с достаточно высокой точностью вычислены при помощи численных методов.

Выбор величины шага λ^k осуществляется путем решения задачи минимизации $F(x)$ в направлении $\nabla F(x)$ с помощью одного из методов одномерного поиска. Если компоненты вектора градиента заданы аналитически, то значение λ^k, при котором достигается экстремум функции, может быть найдено из условия: $\nabla F(x^{k+1})^T \cdot \nabla F(x^k) = 0$.

Преимущество метода в том, что при переходе от шага к шагу обеспечивается выполнение неравенства $F(x^{k+1}) \le F(x^k)$, т.е. значение функции цели улучшается.

Скорость сходимости метода при решении некоторых задач является недопустимо низкой. Это связано с тем, что изменения переменных непосредственно зависят от величины градиента, которая стремится к нулю в окрестности точки минимума. Поэтому метод наискорейшего спуска эффективен при поиске на значительных расстояниях от точки минимума x^* и плохо работает в окрестности этой точки.

В алгоритме Matlab одномерный поиск происходит путем вычисления F в нескольких точках и использования полиномиальной интерполяции.

Пример

Найти максимальное значение функции $F(x) = -2x_1^2 + 18x_1 - 2x_1 \cdot x_2 - x_2^2 + 12x_2$ методом наискорейшего спуска, если $x^0 = [2;\ 1]$.

Находим составляющие вектора градиента функции

$$\nabla F(\mathrm{x}^{\mathrm{k}})^T = \left[\frac{dF(\mathrm{x}^{\mathrm{k}})}{dx_1} \quad \frac{dF(\mathrm{x}^{\mathrm{k}})}{dx_2}\right] = \left[-4x_1 + 18 - 2x_2; -2x_1 - 2x_2 + 12\right].$$

Градиент функции в точке x^0 будет $\nabla F(x^0)^T = [8,6]$.

На первом шаге движение осуществляется из точки x^0 вдоль вектора $\nabla F(x^0)$ в новую точку x^1: $\begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix} = \begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix} + \lambda^0 \begin{bmatrix} 8 \\ 6 \end{bmatrix} = \begin{bmatrix} 2 + 8\alpha^0 \\ 1 + 6\alpha^0 \end{bmatrix}$.

Величину шага λ^0 найдем из условия

$$\nabla F(x^1)^T \cdot \nabla F(x^0) = [-4(2 + 8\lambda^0) + 18 - 2(1 + 6\lambda^0);$$

$$-2(2 + 8\lambda^0) - 2(1 + 6\lambda^0) + 12]\begin{bmatrix} 8 \\ 6 \end{bmatrix} = \left[8 - 44\lambda^0;\ 6 - 28\lambda^0\right] \cdot \begin{bmatrix} 8 \\ 6 \end{bmatrix} = 100 - 520\lambda^0 = 0,$$

откуда $\lambda^0 = 0{,}192$.

В результате координаты точки x^1 получаются равными $x_1^1 = 2 + 8\lambda^0 =$ $=2 + 8 \cdot 0{,}192 = 3{,}54$; $x_2^1 = 1 + 6\lambda^0 = 1 + 6 \cdot 0{,}192 = 2{,}15$. На втором шаге движение осуществляется в направлении вектора $\nabla F(x^1)^T = [-0{,}46;\ 0{,}62]$. Координаты очередной точки x^2 определяются выражением

$$\begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} 3{,}54 \\ 2{,}15 \end{bmatrix} + \lambda^1 \begin{bmatrix} -0{,}46 \\ 0{,}62 \end{bmatrix} = \begin{bmatrix} 3{,}54 - 0{,}46\,\lambda^1 \\ 2{,}15 + 0{,}62\,\lambda^1 \end{bmatrix}.$$

Величину шага λ^1 найдем следующим образом:

$$\nabla F(x^2)^T \cdot \nabla F(x^1) = [-4(3{,}54 - 0{,}46\lambda^1) + 18 - 2(2{,}15 + 0{,}62\lambda^1);$$
$$-2(3{,}54 - 0{,}46\lambda^1) - 2(2{,}15 + 0{,}62\lambda^1) + 12]\begin{bmatrix} -0{,}46 \\ 0{,}62 \end{bmatrix} = -0{,}594 - 0{,}474\lambda^1 = 0,$$

откуда $\lambda^1 = 1{,}25$ и $x_1^2 = 2{,}965; x_2^2 = 2{,}92$.

Составляющие вектора $\nabla F(x^2)$ определяются следующими значениями: $\nabla F(x^2)^T = [0{,}32;\ 0{,}24]$. Для нахождения точки экстремума следует сделать ещё один шаг, в результате которого истинная точка экстремума определится координатами $x^* = [3,\ 3]$, $F_{\max} = 45$.

В общем случае процедура поиска продолжается до тех пор, пока все составляющие вектора градиента не станут достаточно малыми, т.е.

$$\left\|\nabla F(x^{\mathrm{k}})\right\| = \sqrt{\left(\frac{dF(x^{\mathrm{k}})}{dx_1}\right)^2 + \left(\frac{dF(x^{\mathrm{k}})}{dx_2}\right)^2} \le \varepsilon,$$

где ε – точность решения задачи.

Метод Дэвидона—Флетчера—Пауэла (ДФП) (Davidon—Fletcher—Powell). В отличие от метода наискорейшего спуска метод ДФП весьма эффективен в тех случаях, когда точка поиска x^{k} находится вблизи точки минимума.

Последовательность итераций строится в соответствии с формулой $x^{\mathrm{k}+1} = x^{\mathrm{k}} + \lambda^{\mathrm{k}} S^{\mathrm{k}}$, где S^{k} – направление поиска на k-м шаге, определяемое выражением $S^k = -A^k \cdot \nabla F(x^k)$. A^{k} – это квадратная матрица порядка n, которая изменяется на каждой итерации и носит название метрики.

Начальная точка x^0 выбирается произвольно, матрица A^0 является единичной матрицей, т.е. $A^0 = \mathrm{I}$, поэтому направление $S^0 = -\nabla F(x^0) = -g^0$. Поиск вдоль этого направления на первом шаге дает такой же результат, как в методе наискорейшего спуска: $x^1 = x^0 - \lambda^0 \nabla F(x^0) = x^0 - \lambda^0 g^0$.

Рекуррентное соотношение для вычисления матрицы A имеет вид

$$A^k = A^{k+1} + \frac{\Delta x^{k-1} \cdot (\Delta x^{k-1})^T}{(\Delta x^{k-1})^T \cdot \Delta g^{k-1}} + \frac{A^{k-1} \cdot \Delta g^{k-1} \cdot (\Delta g^{k-1})^T \cdot A^{k-1}}{(\Delta g^{k-1})^T \cdot A^{k-1} \cdot \Delta g^{k-1}}.$$

Здесь изменение переменной $\Delta x^k = x^{k-1} - x^k$, а изменение градиента при переходе от точки x^{k} к точке $x^{\mathrm{k}+1}$ равно $\Delta g^k = g^{k+1} - g^k$. Величина шага λ^{k} вычисляется на каждой итерации так же, как в методе наискорейшего спуска, из условия минимизации функции в рассматриваемом направлении.

Алгоритм обеспечивает убывание целевой функции при переходе от итерации к итерации, и метод ДФП широко используется при решении самых различных задач, возникающих на практике.

Метод Бройдена—Флетчера—Гольдфарба—Шенно (БФГШ) (Broyden—Fletcher—Coldfarb—Shanno). В методе БФГШ последовательность итераций вычисляется в соответствии с выражением $x^{k+1} = x^k - \lambda^k \cdot A^k \cdot \nabla F(x^k)$.

Полагая, что $\nabla F(\mathrm{x}^{\mathrm{k}}) = g^k$ и $A^0 = I$, рекуррентную формулу для вычисления переменной матрицы A можно записать следующим образом:

$$\mathrm{A}^{\mathrm{k+1}} = \left[\mathrm{I} - \frac{\Delta \mathrm{x}^{\mathrm{k}} \cdot (\Delta \mathrm{g}^{\mathrm{k}})^{\mathrm{T}}}{(\Delta \mathrm{x}^{\mathrm{k}})^{\mathrm{T}} \cdot \Delta \mathrm{g}^{\mathrm{k}}}\right]^{\mathrm{T}} \cdot \mathrm{A}^{\mathrm{k}} \cdot \left[\mathrm{I} - \frac{\Delta \mathrm{x}^{\mathrm{k}} \cdot (\Delta \mathrm{g}^{\mathrm{k}})^{\mathrm{T}}}{(\Delta \mathrm{x}^{\mathrm{k}})^{\mathrm{T}} \cdot \Delta \mathrm{g}^{\mathrm{k}}}\right] + \frac{\Delta \mathrm{x}^{\mathrm{k}} \cdot (\Delta \mathrm{x}^{\mathrm{k}})^{\mathrm{T}}}{(\Delta \mathrm{x}^{\mathrm{k}})^{\mathrm{T}} \cdot \Delta \mathrm{g}^{\mathrm{k}}}.$$

Величина шага λ^{k} вычисляется в результате одномерного поиска минимума функции по параметру λ, в рассматриваемом направлении S^k. К числу главных преимуществ метода относится слабая зависимость точности вычислений при проведении одномерного поиска.

К методам второго порядка, изучаемым в работе относятся: метод Гаусса—Ньютона и метод Левенберга—Марквардта.

Метод Гаусса—Ньютона(Gauss—Newton). Метод Ньютона использует информацию о вторых производных целевой функции. Вычисление каждой очередной точки осуществляется по формуле $x^{k+1} = x^k - H^{-1}(x^k) \cdot \nabla F(x^k)$, где $H(x^k) = \nabla^2 F(x^k)$ — квадратная матрица порядка n вторых производных функции $F(x)$, вычисленных в точке x^{k}. Матрицу $H(x)$ называют матрицей Гессе или гессианом функции $F(x)$. $H^{-1}(x^k)$ — матрица, обратная гессиану, вычисленному в точке x^{k}. Метод Ньютона имеет квадратичную скорость сходимости на классе выпуклых функций. Однако при исследовании неквадратичных функций метод Ньютона не отличается высокой надежностью, если точка x^0 находится на значительном расстоянии от точки x^*. Прибегают к модификации метода, чтобы обеспечить уменьшение целевой функции от итерации к итерации и осуществляют поиск вдоль направления, как в методе наискорейшего спуска. Последовательность итераций осуществляется в соответствии с формулой $x^{k+1} = x^k - \lambda^k H^{-1}(x^k) \cdot \nabla F(x^k)$.

Выбор λ^{k} осуществляется так, чтобы $F(x^{k+1})$ в рассматриваемом направлении принимала минимальное значение. Это гарантирует выполнение неравенства $F(x^{k+1}) \le F(x^k)$.

Если функция цели квадратичная, экстремум находится за один шаг.

Метод Левенберга—Марквардта (ЛМ) (Levenberg—Marquardt). Рассматриваемый метод является комбинацией метода наискорейшего спуска и метода Ньютона, в которой удачно сочетаются положительные свойства

обоих методов. Направление поиска определяется равенством $S^k = -[H^k + \lambda^k I]^{-1} \cdot \nabla F(x^k)$. При этом параметр λ позволяет не только изменять направление поиска, но и регулировать длину шага. На начальной стадии поиска параметру λ^0 присваивается достаточно больше значение (например, 10^4), так что $[H^0 + \lambda^0 I]^{-1} = [\lambda^0 I]^{-1} = \left(\frac{1}{\lambda^0}\right) \cdot I$.

Таким образом, большим значениям λ^0 соответствует направление поиска S^0, совпадающее с направлением $-\nabla F(x^0)$. При уменьшении величины λ до 0 S будет изменяться до направления, определяемого по методу Ньютона. Если в результате первого шага $F(x^1) \le F(x^0)$, следует выбрать $\lambda^1 < \lambda^0$ и реализовать ещё один шаг, в противном случае следует увеличить λ^0, т.е. положить, что $\lambda^0 = \beta\lambda^0$, где $\beta \ge 1$, и вновь реализовать предыдущий шаг.

Метод характеризуется относительной простотой, свойством убывания целевой функции при переходе от итерации к итерации, высокой скоростью сходимости в окрестности точки минимума, а также отсутствием процедуры поиска величины шага вдоль направления. Этот метод широко используется при решении задач, в которых функция $F(x)$ записывается в виде суммы квадратов, т.е. $F(x) = F_1^2(x) + F_2^2(x) + ... + F_m^2(x)$.

Описание лабораторной работы

Для исследования предлагается функция

$$F(x) = e^{x_1} \cdot (k_1 \cdot x_1{}^2 + k_2 \cdot x_2{}^2 + k_3 \cdot x_1 \cdot x_2 + k_4 \cdot x_1 + k_5 \cdot x_2),$$

где коэффициенты k_i вычисляются в соответствии с подпрограммой генерации варианта задания после введения фамилии. В дальнейшем значение k передается параметру $P1$. Все необходимые пояснения даются как комментарии в тексте.

Необходимо найти минимальное значение функции и проследить изменение координат точек поиска поочередно каждым из изложенных в теоретических сведениях методов. Открыть файл *fmi*2 и ознакомиться с текстом программы. Все необходимые пояснения даются как комментарии в тексте.

Установка метода оптимизации осуществляется функциями *FMINU*, *FMINS* и *LEASTSQ*. Вектор управляющих параметров *options* содержит 18 компонент и предназначен для настройки алгоритмов оптимизации.

Функция *FMINU* и одно из значений *options*(6) используются для реализации следующих методов:

1) по умолчанию (*options*(6)=0) – метод Broyden—Fletcher—Goldfarb-Shanno;

2) при *options*(6)=1 – метод Davidon—Fletcher—Powell;

3) при *options*(6)=2 – метод Steepest Descent (наискорейшего спуска),

функция *FMINS* используется для реализации неградиентного симплекс-метода:

4) метод Nelder—Mead,

функция *LEASTSQ* и одно из значений *options*(5) используются для реализации методов:

5) при *options*(5)=1 – метод Gauss—Neuton;

6) по умолчанию (*options*(5)=0) – метод Levenberg—Marquardt.

Функция *LEASTSQ* преобразует целевую функцию $F(x)$ в квадратичную $F^2(x)$. Если $F(x)$ является знакопеременной, то минимальные значения функций $F(x)$ и $F^2(x)$ не совпадают. Поэтому функция $F(x)$ дополняется константой $F(x)$+const для устранения знакопеременности. Иначе функция *LEASTSQ* найдет не минимум $F(x)$, а ее нулевое значение. На каждой итерации осуществляется поиск минимума исследуемой функции по величине шага λ в выбранном направлении (одномерная минимизация). Для одномерного поиска используются методы полиномиальной аппроксимации, задаваемые значением *options*(7):

7) по умолчанию (*options*(7)=0) – метод смешанной полиномиальной аппроксимации Mixed Polynomical Interpolation;

8) при *options*(7)=1 – кубическая аппроксимация Cubic Interpolation.

Режим вывода результатов устанавливается одним из значений *options*(1):

— по умолчанию (*options*(1)=0) – вывод промежуточных значений не происходит;

— при *options*(1)=1 – обычный вывод промежуточных значений;

— при *options*(1)=2 – расширенный вывод промежуточных значений.

Для контроля вектора градиента *options*(9) установить в 1.

При выводе результатов:

options(8) – значение целевой функции;

options(10) – количество выполненных итераций.

Итерационная погрешность задается значением *options*(2) для аргумента и *options*(3) для функции, значение погрешности по умолчанию 10^{-4}.

Порядок выполнения работы

1. Ввести команды *addpath c:/ work/vit* и ознакомиться с текстом программы *fmi*2. Все необходимые пояснения даются как комментарии в тексте.

2. Ввести фамилию и наблюдать линии уровня равных значений функции $F(x)$. Диапазон изменения переменных и величину шага можно менять для корректировки изображения на экране.

3. Наблюдать процедуру поиска минимума функции $F(x)$ отдельно каждым из описанных ранее методов. При изменении метода ненужные строки программы закрыть % – символом комментария.

Так, например, при реализации метода наискорейшего спуска с вычислением градиента по формуле и кубической интерполяцией при одномерном поиске должны быть открыты следующие разделы программы:

1) генерация варианта задания;
2) изображение функции $F(x)$, включая подпункты 2.1, 2.2, 2.3, 2.4;
3) минимизация функции $F(x)$, включая подпункты 3.1, 3.2, 3.3.

Далее следует открыть следующие строчки программы из подразделов 3.4, 3.5 и 3.6:

options(6) = 2; метод наискорейшего спуска;
options(7) = 1; кубическая аппроксимация;
options(1) = 2; расширенный вывод промежуточных значений

и строку из подраздела 3.8:

[*x, options*] = *fminu*(*F*, *xo*, *options*, *grad*, *P1*, *P2*, *P3*): pause.
options(8) значение функции цели;
options(10) количество выполненных итераций.

При расширенном выводе результатов на экране печатаются:

Количество вычислений функции для одномерного поиска	Значения координат точки поиска *x*1, *x*2	Значение функции в точке	Значение *grad^2*

По результатам расчетов построить траекторию движения к точке экстремума на первых пяти шагах.

Таким же образом, открывая нужные строки программ из разделов 1-3, можно проследить процедуру решения задачи каждым отдельным методом.

Раздел 4 программы предназначен для сравнительной оценки методов. При этом точки поиска для каждого метода изображаются разными цветами на одном экране; отдельно для методов 1–3 и методов 4–6.

Контрольные вопросы

1. Сформулировать необходимые и достаточные условия безусловного экстремума функций многих переменных.
2. Метод наискорейшего спуска, выбор направления и величины шага.
3. Метод Ньютона. Скорость сходимости.
4. Особенности метода Нелдера—Мида.
5. Методы одномерного поиска. Суть методов полиномиальной аппроксимации.

Пример программы

% *fmi*2 Минимизация функции двух переменных без ограничений

```
% с заданием коэффициентов  целевой функции.
% Найти значения переменных x(1) и x(2) минимизирующих функция F(x).
% Целевая функция F(x) двух переменных x(1) и x(2) имеет вид:
%                    2       2
%  F(x) = exp(x(1)).(4x(1) + 2x(2) + 4x(1).x(2) + 2x(2) + 1)
%
%  где дополнительно строкой К задаются коэффициенты при x(i).
                %       1.       ГЕНЕРАЦИЯ ВАРИАНТА ЗАДАНИЯ
clear,close all
  ng=input(' Введите свою фамилию  ','s');
 rand('state',sum([datestr(now,11),ng]));
 while ng
    k1=round((rand(1,3)-.5)*18);k=ceil(rand(1,3)*9);
    if k(1)*k(2)*4>=k(3)^2;k=[k,k1];break,end
 end;txt1=[ng,'   k  =   ',num2str(k)]
        %  2.  ИЗОБРАЖЕНИЕ ФУНКЦИИ  F(x)
        %  2.1.         ЗАДАНИЕ ДИАПАЗОНА ИЗМЕНЕНИЯ ПЕРЕМЕННЫХ
xmin=-6; xmax=2  ; ymin=-4;     ymax=3; % пределы изменений
x_shag=(xmax-xmin)/40;  y_shag=(ymax-ymin)/40;  % шаг изменений
x1=xmin:x_shag:xmax;     y1=ymin:y_shag:ymax;     %       набор значений
[xx,yy]=meshgrid(x1,y1);           % двухмерная сетка значений
        %       2.2.     ВЫЧИСЛЕНИЕ ФУНКЦИИ F(x) НА ДВУХМЕРНОЙ СЕТКЕ
FF=exp(xx).*(k(1).*xx.^2 + k(2).*yy.^2 + k(3).*xx.*yy +k(4).*xx+ k(5).*yy + k(6));
                % 2.3. ТРЁХМЕРНОЕ ИЗОБРАЖЕНИЕ ФУНКЦИИ  F(x)
figure(1)
surfl(x1,y1,FF); % surf,mesh,
                % 2.4. ИЗОБРАЖЕНИЕ ФУНКЦИИ  F(x) КОНТУРАМИ ЛИНИЙ
РАВНЫХ ЗНАЧЕНИЙ
figure(2)
 cmin=-0.1; % минимальное значение линии уровня
 cmax=100; % максимальное значение линии уровня
 nc=12; % число линии уровня
 cl=cmin+cmax*logspace(-3,0,nc);  % nc  значений уровня
%cl=[-4 -2 -1 -.5 0 .5 1 2 4 8 16 30];%уровни линий равных значений
[c,h]=contour3(x1,y1,FF,cl,'r'); %контурная поверхность красного цвета
clabel(c,h);   % подписать значения контуров
view(0,90); % повернуть поверхность к виду сверху
title(txt1);% надпись на рисунке, фамилия и значения коэффициентов К
hold on;        % сохранение рисунка для изображения точек поиска минимума
  pause
                        % 3.              МИНИМИЗАЦИЯ ФУНКЦИИ  F(x)
          % 3.1.        ЗАДАНИЕ  МИНИМИЗИРУЕМОЙ ФУНКЦИИ  F(x)
  P1=k;         % передача значений k параметру P1 целевой функции F(x)
% Задание F(x,P1,P2,P3)в форме её математического описания, т.е.INLINE
% В F(x) встроена функция xplt(x,P2,P3) для рисования точек  F(x)
% в xplt(x,P2,P3) через P2,P3 передаются цвет, маркёр, длительность паузы.
```

```
P2='b+'; % задаёт цвет и маркёр точки решения
P3=1; % задаёт паузу в секундах между точками
% минимизируемая функция F(x,P1) с встроенной функцией xplt(x,P2,P3)
F =inline('[exp(x(1)).*(P1(1).*x(1).^2+P1(2).*x(2).^2+P1(3).*x(1).*x(2)+
+P1(4).*x(1)+P1(5).*x(2)+P1(6));xplt(x,P2,P3);]',3);
% минимизируемая функция F_(x,P1) без встроенной функции xplt(x,P2,P3)
F_ =inline('[exp(x(1)).*(P1(1).*x(1).^2+P1(2).*x(2).^2+P1(3).*x(1).*x(2)
+P1(4).*x(1)+P1(5).*x(2)+P1(6)); ]',3);
    %  3.2.  ЗАДАНИЕ ГРАДИЕНТА МИНИМИЗИРУЕМОЙ ФУНКЦИИ  F(x)
% Задание ГРАДИЕНТА F(x) в форме его математического описания, т.е.INLINE
% В grad  встроена функция xplt(x) для рисования точек градиента F(x)
% xplt(x)  по умолчанию рисует красную звёздочку без паузы
grad = '[exp(x(1)).*(P1(1). *x(1).^2+P1(2). *x(2).^2+P1(3). *x(1). *x(2)+P1(4). *x(1)+
+P1(5). *x(2)+P1(6)+2*P1(1). *x(1)+P1(3). *x(2)+P1(4)); exp(x(1)). *(2*P1(2).*x(2)+
+P1(3).*x(1)+P1(5));xplt(x);]';
% В grad_ встроена функция xplt(x,P2,P3), описанная выше
grad_='[exp(x(1)).*(P1(1). *x(1).^2+P1(2). *x(2).^2+P1(3).*x(1). *x(2)+P1(4). *x(1)+
+P1(5).*x(2)+P1(6)+2*P1(1).*x(1)+P1(3).*x(2)+P1(4));exp(x(1)).*(2*P1(2).*x(2)+P1(3
)**x(1)+P1(5));xplt(x,P2,P3);]';
% Если нужно выводить точки одномерного поиска, используйте F и grad.
% Функция F_ совместно с grad_ не изображают точек одномерного поиска.
        %  3.3.       ЗАДАНИЕ НАЧАЛЬНОЙ ТОЧКИ x0
x0 = [-0.5 -1]; % начальная точка минимизации
options = []; % сброс прежних управляющих параметров оптимизации
options(14) = 40;    % ограничение числа вычислений
                    %      3.4 УСТАНОВКА МЕТОДА ОПТИМИЗАЦИИ
 % Метод минимизации задаётся одной из функций (fminu, fmins, leastsq)
% и значениями options(6) ИЛИ options(5).
% Для функции fminu установка метода оптимизации производится options(6)
% по умолчанию options(6)=0; метод Broyden—Fletcher—Goldfarb—Shanno
% options(6) = 1;    метод Davidon—Fletcher—Powell
% options(6) = 2;    метод наискорейшего спуска
% Для функции leastsq (наименьших квадратов) используется options(5)
% по умолчанию (options(5) = 0; метод Levenberg—Marquardt         '
% options(5) = 1; метод  Gauss—Newton
% Функция fmins использует неградиентный симплекс–метод Nelder—Mead
        % 3.5.       УСТАНОВКА СПОСОБА ОДНОМЕРНОГО ПОИСКА
% Поиск минимума функции F(x) по выбранному направлению происходит
% интерполяцией по нескольким вычисленным точкам.
% options(7) - определяет метод интерполяции
% Метод интерполяции по умолчанию;       1) Mixed Polynomial Interpolation
% options(7) = 1;                        2) Cubic Interpolation
        %  3.6. УСТАНОВКА РЕЖИМА ВЫВОДА ПРОМЕЖУТОЧНЫХ
ЗНАЧЕНИЙ
% по умолчанию (options(1)=0;) нет  вывода промежуточных значений
% options(1) =1; % обычный вывод промежуточных значений
```

```
options(1) =2; % расширенный вывод промежуточных значений
          % 3.7.МИНИМИЗАЦИЯ ФУНКЦИИ  F(x) С ЧИСЛЕННЫМ МЕТОДОМ
ВЫЧИСЛЕНИЯ ГРАДИЕНТА
% текст заголовка, маркёры, содержание графиков
title([txt1,'  ',P2,'кубическая интерполяция, градиент численно']);
options(7) = 1; disp('Кубической интерполяции, Cubic Interpolation')
[x, options] = fminu(F, x0, options,[],P1,P2,P3); pause
%[x, options] = leastsq(F, x0, options,[],P1,P2,P3); pause
options(7) = 0;% Метод интерполяции Mixed Polynomial Interpolation
disp('Смешанный полиномиальный метод интерполяции при поиске экстремума
по выбранному направлению.')
   P2='gx'; % изменяем цвет и маркёр точки решения
title([txt1,P2,'Mixed Polynomial Interpolation, градиент численно']);
      [x, options] = fminu(F, x0, options,[],P1,P2,P3); pause
         %[x, options] = leastsq(F, x0, options,[],P1,P2,P3); pause
hold off;       % выключаем сохранение рисунка
% ОБНОВЛЯЕМ ИЗОБРАЖЕНИЕ ФУНКЦИИ F(x) КОНТУРАМИ ЛИНИЙ
РАВНЫХ ЗНАЧЕНИЙ
       figure;
[c,h]=contour3(x1,y1,FF,cl,'r'); % контурная поверхность красного цвета
clabel(c,h);   % подписать значения контуров
view(0,90);   % повернуть поверхность к виду сверху
title(txt1)      % надпись на рисунке значения коэффициентов К
hold on;       % сохранение рисунка для изображения точек поиска минимума
  pause
  disp('МИНИМИЗАЦИЯ С ЧИСЛЕННЫМ И ФОРМУЛЬНЫМ
ВЫЧИСЛЕНИЯМИ ГРАДИЕНТА')
    disp('МИНИМИЗАЦИЯ С ЧИСЛЕННЫМ  ВЫЧИСЛЕНИЕМ ГРАДИЕНТА')
% повторяем Mixed Polynomial Interpolation, градиент численно
  P2='b+'; % изменяем цвет и маркёр точки решения
title([txt1,P2,' Mixed Polynomial Interpolation, градиент численно']);
 [x, options] = fminu(F, x0, options,[],P1,P2,P3); pause
% [x, options] = leastsq(F, x0, options,[],P1,P2,P3); pause
  % 3.8. МИНИМИЗАЦИЯ ФУНКЦИИ  F(x) С  ВЫЧИСЛЕНИЕМ ГРАДИЕНТА
ПО ФОРМУЛЕ
   disp('МИНИМИЗАЦИЯ С ФОРМУЛЬНЫМ ВЫЧИСЛЕНИЕМ ГРАДИЕНТА')
P2='gx'; % изменяем цвет и маркёр точки решения
 title([txt1,P2,'градиент формулой, Mixed Polynomial Interpolation,']);
 [x, options] = fminu(F, x0, options,grad,P1,P2,P3); pause
% [x, options] = leastsq(F, x0, options,grad,P1,P2,P3); pause
% 3.9. МИНИМИЗАЦИЯ ФУНКЦИИ  F(x) НЕГРАДИЕНТНЫМ СИМПЛЕКС–
МЕТОДОМ
[x, options] = fmins(F, x0, options,[],P1,P2,P3); pause % Nelder—Mead
  hold off;     % выключаем сохранение рисунка
                              % 4. СРАВНЕНИЕ МЕТОДОВ МИНИМИЗАЦИИ
  disp('СРАВНЕНИЕ МЕТОДОВ МИНИМИЗАЦИИ')
```

```
% 4.1. ОБНОВЛЯЕМ ИЗОБРАЖЕНИЕ ФУНКЦИИ F(x)
КОНТУРАМИ ЛИНИЙ РАВНЫХ ЗНАЧЕНИЙ
      figure;
[c,h]=contour3(x1,y1,FF,cl,'r'); %контурная поверхность красного цвета
   clabel(c,h);% подписать значения контуров
    view(0,90);         % повернуть поверхность к виду сверху
    title(txt1)  % надпись на рисунке значения коэффициентов К
  hold on;      % сохранение рисунка для изображения точек поиска минимума
  pause
                %  4.2. ЗАДАЁМ УПРАВЛЯЮЩИЕ ПАРАМЕТРЫ FOPTIONS
    % x0 = [-1 -1]; % изменение начальной точки
 options = []; % сброс прежних управляющих параметров оптимизации
  options(14) = 20;  % изменение ограничения числа вычислений
% options(7) = 1; %   Cubic Interpolation
options(1) =2; % расширенный вывод промежуточных значений
% 4.3. УСТАНАВЛИВАЕМ МЕТОДЫ МИНИМИЗАЦИИ, ЦВЕТА И МАРКЁРЫ
ТОЧЕК РЕШЕНИЙ
%options(6) = 0; метод Broyden—Fletcher—Goldfarb—Shanno
     P2='b+'; % задаёт цвет и маркёр точки решения
title([txt1,'  ',P2,' - метод  Broyden—Fletcher—Goldfarb—Shanno']);
 disp('метод  Broyden—Fletcher—Goldfarb—Shanno')
% Используем функцию F_  без маркёра, не отображаем одномерный поиск.
% Маркёр шагов решения совмещён с вычислением градиента grad_.
[x, options] = fminu(F_, x0, options,grad_,P1,P2,P3); pause
options(6) = 1; % метод  Davidon—Fletcher—Powell
  disp('метод  Davidon—Fletcher—Powell')
 P2='gx'; % задаёт цвет и маркёр точки решения
title([txt1,'  ',P2,' –  метод  Davidon—Fletcher—Powell']);
   [x, options] = fminu(F_, x0, options,grad_,P1,P2,P3); pause
options(6) = 2; disp('метод НАИСКОРЕЙШЕГО СПУСКА')
 P2='c.'; % задаёт цвет и маркёр точки решения
 title([txt1,'  ',P2,'- метод НАИСКОРЕЙШЕГО СПУСКА']);
[x, options] = fminu(F_, x0, options,grad_,P1,P2,P3); pause
hold off;       % выключаем сохранение рисунка
%4.4.ОБНОВЛЯЕМ ИЗОБРАЖЕНИЕ ФУНКЦИИ F(x) КОНТУРАМИ ЛИНИЙ
РАВНЫХ ЗНАЧЕНИЙ
figure;
[c,h]=contour3(x1,y1,FF,cl,'r'); % контурная поверхность красного цвета
clabel(c,h);   % подписать значения контуров
 view(0,90);  % повернуть поверхность к виду сверху
title(txt1)      % надпись на рисунке значения коэффициентов К
hold on;        % сохранение рисунка для изображения точек поиска минимума
  pause
 % 4.5.УСТАНАВЛИВАЕМ МЕТОДЫ МИНИМИЗАЦИИ, ЦВЕТА И МАРКЁРЫ
ТОЧЕК РЕШЕНИЙ
 %по умолчанию (options(5) = 0; метод  Levenberg—Marquardt          '
```

```
P2='cx'; % задаёт цвет и маркёр точки решения
title([txt1,'  ',P2,' – метод Levenberg—Marquardt']);
disp(' метод Levenberg—Marquardt')
  [x, options] = leastsq(F_, x0, options,grad_,P1,P2,P3); pause
       options(5) = 1; disp(' метод Gauss—Newton')
P2='b+'; % задаёт цвет и маркёр точки решения
title([txt1,'  ',P2,' –  метод  Gauss—Newton']);
  [x, options] = leastsq(F_, x0, options,grad_,P1,P2,P3); pause
P2='g.'; % задаёт цвет и маркёр точки решения
title([txt1,'  ',P2,' –  метод  Nelder—Mead ']);
disp('метод  Nelder—Mead ')
[x, options] = fmins(F, x0, options,[],P1,P2,P3); pause % Nelder—Mead
    hold off
```

Лабораторная работа № 4

Экстремальные нелинейные задачи с ограничениями

Цель работы: изучение методов, используемых при решении задач нелинейного программирования.

Краткие теоретические сведения

В нелинейном программировании (НП) рассматриваются задачи отыскания экстремума функции многих переменных при наличии ограничений на переменные в виде неравенств, при этом функция цели или хотя бы одно ограничение нелинейны. Задача НП формулируется следующим образом. Найти значения переменных x_1, x_2,...x_n, минимизирующие скалярную целевую функцию $F(x_1, x_2,...x_n)$ при наличии ограничений вида $g_j(x) \geq 0$, $j = \overline{1,m}$, где по крайней мере одна из функций $F(x)$ и $g(x)$ является нелинейной.

Методы решения задач нелинейного программирования по сравнению с задачами линейного программирования обладают большим многообразием. Экстремум в задачах НП может достигаться не только в вершине или на границе области допустимых значений переменных, но и внутри области. Допускаются любые соотношения между количеством переменных n и числом m ограничений задачи, т.е. $n<m$, $n=m$, $n>m$. Решение задач нелинейного программирования может давать два или несколько экстремумов, что требует дополнительных проверок результата решения.

Одной из важнейших теорем в теории нелинейного программирования является теорема Куна—Таккера, обобщающая классический метод неопределенных множителей Лагранжа.

Предположим, что имеется следующая задача:

$$\min\left\{F(x_1,x_2,...x_n)\middle| g_j(x_1,x_2,...x_n)\ge 0, j=\overline{1,m}, x_i\ge 0, i=\overline{1,n}\right\}, \qquad (4.1)$$

где $F(x)$ и $g_j(x)$ – выпуклые функции n переменных.

Введем функцию Лагранжа $L(x,\lambda)=F(x)+\sum_{j=1}^{m}\lambda_j g_j(x)$, используя совокупность неопределенных множителей Лагранжа $\lambda_1, \lambda_2,\ldots, \lambda_m$.

Теорема Куна—Таккера формулируется следующим образом. Пусть существует вектор x, такой, что $x_i\ge 0, i=\overline{1,n}$ и $g_j(x)\ge 0, j=\overline{1,m}$. Тогда для того, чтобы вектор x^* был оптимальным решением задачи (4.1), необходимо и достаточно, чтобы существовал неотрицательный m-мерный вектор λ^T, такой что

$$L(x^*,\lambda)\le L(x^*,\lambda^*)\le L(x,\lambda^*) \qquad (4.2)$$

для всех $x\ge 0, \lambda\ge 0$.

Выражение (4.2) означает, что функция L в точке (x^*,λ^*) при фиксированном x^* имеет глобальный максимум в области $\lambda\ge 0$ при $\lambda=\lambda^*$, а при фиксированном λ^* она имеет глобальный минимум в области $x\ge 0$ при $x=x^*$. Экстремальная точка (x^*, λ^*) с такими свойствами называется седловой точкой, а теорему Куна—Таккера часто называют теоремой о седловой точке. Итак, задаче (4.1) минимизации $F(x)$ соответствует задача нахождения седловой точки (минимаксная задача) для функции L, в которой из всех ограничений сохраняются только ограничения на знак.

Если $F(x)$ и $g_j(x)$ являются дифференцируемыми функциями, то условия теоремы Куна—Таккера записываются следующим образом:

$$\left.\begin{aligned}&\left.\frac{\partial L}{\partial x_i}\right|_{x^*,\lambda^*}\ge 0\\ &x_i^*\left.\frac{\partial L}{\partial x_i}\right|_{x^*,\lambda^*}=0\\ &x_i^*=0, i=\overline{1,n}\end{aligned}\right\}, \qquad (4.3)$$

$$\left.\begin{aligned}&\left.\frac{\partial L}{\partial \lambda_j}\right|_{x^*,\lambda^*}\le 0\\ &\lambda_j^*\left.\frac{\partial L}{\partial \lambda_j}\right|_{x^*,\lambda^*}=0\\ &\lambda_j^*=0, j=\overline{1,m}\end{aligned}\right\}. \qquad (4.4)$$

Для точек, где $x_i > 0$, в точке минимума должно выполняться условие $\left.\frac{\partial L}{\partial x_i}\right|_{x^*,\lambda^*} = 0$, а на границе области, где $x_i = 0$, первая производная для случая минимума должна быть неотрицательной, т.е. $\left.\frac{\partial L}{\partial x_i}\right|_{x^*,\lambda^*} > 0$.

Аналогично по переменной λ для внутренних точек ($\lambda > 0$) выполняется равенство нулю производной, а для граничных точек ($\lambda = 0$) первая производная для случая максимума должна быть неположительной, т.е. $\left.\frac{\partial L}{\partial \lambda_j}\right|_{x^*,\lambda^*} < 0$.

Условия (4.3) и (4.4) эквивалентны (4.2), т.е. являются необходимыми и достаточными для оптимальности x^*. Если имеется в виду седловая точка с максимумом по x и минимумом по λ, то знаки неравенств в первых строчках систем (4.3) и (4.4) изменяются на противоположные.

Одним из условий применения теоремы Куна—Таккера является требование к выпуклости $F(x)$ и всех ограничений. Рассмотрим задачу квадратичного программирования:

$$\min\left\{F(x) = C^T x + x^T Dx \middle| Ax \le B, x \ge 0\right\}, \qquad (4.5)$$

где F – квадратичная функция цели; $C^{\mathrm{T}}=[C_1, C_2,\ldots,C_{\mathrm{n}}]$ – вектор постоянных коэффициентов; D – постоянная симметричная положительно–полуопределенная матрица. Ограничения на переменные линейны.

Функция Лагранжа для задачи (4.5) имеет вид: $L(x, \lambda) = C^{\mathrm{T}}x + x^{\mathrm{T}}Dx + \lambda^{\mathrm{T}}g(x) = C^{\mathrm{T}}x + x^{\mathrm{T}}Dx + \lambda^{\mathrm{T}}(Ax-B)$. Применим условия теоремы Куна—Таккера к рассматриваемой задаче. Найдем частные производные $\frac{\partial L}{\partial x} = C + 2Dx + A^T\lambda \ge 0$, $\frac{\partial L}{\partial \lambda} = Ax - B \le 0$.

Введем дополнительные неотрицательные переменные, чтобы привести неравенства к виду равенств: $C + 2Dx + A^T\lambda - V = 0$, $Ax - B + W = 0$.

Очевидно, что $\frac{\partial L}{\partial x} = V$, $\frac{\partial L}{\partial \lambda} = -W$, следовательно,

$$x^T V = 0, \lambda^T W = 0.$$

Таким образом, решение задачи квадратичного программирования сводится к решению следующей системы уравнений:

$$\begin{cases} Ax + W = B, \\ 2Dx + A^T\lambda - V = -C, \\ x^T V = 0, \lambda^T W = 0, \\ x \ge 0, \lambda \ge 0, V \ge 0, W \ge 0. \end{cases} \qquad (4.6)$$

Существуют различные специальные методы определения x^* и λ*. Один из них реализуется в подпрограмме Q*P* пакета MATLAB, обращение к которой будет рассмотрено ниже.

Метод допустимых направлений Зойтендейка пригоден для решения задач с достаточно сложными функциями цели и ограничениями. Рассмотрим последовательность решения нелинейной задачи с линейными ограничениями, когда экстремум достигается на границе области допустимых значений переменных (ОДЗП) $\max\left\{F(x)\middle| Ax \le B, x_i \ge 0, i=\overline{1,n}, x^0 \in X\right\}$.

Из начальной точки x^0, лежащей внутри ОДЗП, движение осуществляется по направлению вектора градиента $\nabla F(x^0)$.

Как показано на рис. 4.1, движение осуществляется до тех пор, пока не достигается граница области. В общем случае двигаться до границы необязательно, так как максимум $F(x)$ может достигаться и до встречи с ней. В рассматриваемом случае $F(x)$ все время возрастает, поэтому остановиться следует в точке x^1 на граничной прямой.

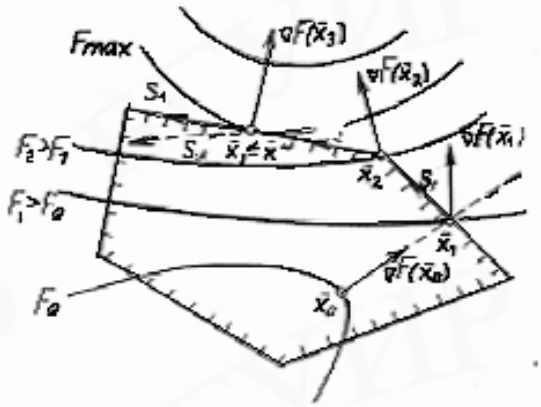


Рис. 4.1. Геометрическая интерпретация процесса решения задачи с линейными ограничениями

Координаты очередной точки $x^1 = x^0 + \alpha^0 \nabla F(x^0)$ зависят от шага α^0. Значение α^0 должно максимизировать функцию цели в рассматриваемом направлении и быть таким, чтобы очередная точка x^1 принадлежала ОДЗП.

Ограничения задачи можно представить в виде $a_j^{\mathrm{T}} x \le b$, где $a_j^{\mathrm{T}}=[a_{j1}, a_{j2},\ldots a_{jn}]$, тогда, решая систему неравенств $\begin{cases} a_j^T[x^0 + \alpha^0 \nabla F(x^0)] \le b_j, \\ x^0 + \alpha^0 \nabla F(x^0) \ge 0, \end{cases}$

определяют интервал $[\alpha_1^0, \alpha_2^0]$ значений α^0, который обеспечивает принадлежность очередной точки x^1 ОДЗП. Далее находится значение α^{0*}, максимизирующее функцию цели. Оно должно принадлежать промежутку $[\alpha_1^0, \alpha_2^0]$, тогда принимается $\alpha^0 = \alpha^{0*}$, если же α^{0*} выходит за пределы промежутка $[\alpha_1^0, \alpha_2^0]$, то принимается $\alpha^0 = \alpha_2^0$. При этом очередная точка x^1 поисковой траектории оказывается на границе области, как и показано на рис. 4.1. Вектор градиента $\nabla F(x^1)$, построенный в точке x^1, выходит за пределы ОДЗП. Поэтому находится новое направление S^1, которое должно быть таким, чтобы очередная точка $x^2 = x^1 + \alpha^1 S^1$ принадлежала ОДЗП, а функция цели при переходе к очередной точке увеличивалась максимальным образом. Направление S^1 определяется из решения следующей вспомогательной задачи математического программирования: $\max\{\nabla^T F(x^1) S^1 \,|\, a_j^T S^1 \le 0\}$ для тех j, при которых $a_j\ x^1 = b_j$. Индекс j соответствует границе ОДЗП, а условие $a_j^T S^1 \le 0$ означает, что направление S^1 следующего шага должно быть изменено и идти либо внутрь, либо по границе ОДЗП. На величину вектора S^1 накладываются ограничения, называемые условиями нормализации. Одна из разновидностей условий нормализации записывается следующим образом: $\|S^1\| = \sqrt{\sum_i (S_i^1)^2} = 1$.

Графически условию нормализации скалярного произведения $\nabla F(x^1)^T S^1$ соответствует выбор такого вектора S^1, который составляет с вектором $\nabla F(x^1)$ наименьший острый угол φ по сравнению с любым вектором, выходящим из точки x^1 и лежащим в ОДЗП.

Когда допустимое направление S^1 найдено, определяется длина шага α^1. Для нахождения α^1 используется необходимое условие экстремума $\nabla F(x^1)^T S^1 = 0$, т.е. вычисляется значение α^1, которое приводит к максимуму $F(x)$ в направлении S^1. Кроме того, точка x^2 не должна выходить за пределы ОДЗП, поэтому α^1 проверяется выражением $a_j^T (x^1 + \alpha^1 S^1) = b_j$. Из двух вычисленных значений α^1 выбирается меньшее, и находятся координаты точки x^2.

Процесс прекращается при достижении точки x^k, в которой максимум $\nabla F(x^k)^T S^k = 0$. Это значит, что соответствующие векторы взаимно перпендикулярны и дальнейшее увеличение целевой функции в данной области невозможно, а точка x^k является искомой точкой максимума функции $F(x)$.

В задачах с линейными ограничениями движение осуществляется по граничным прямым. При нелинейных ограничениях, определяющих выпуклую область, любое сколь угодно малое перемещение из граничной точки может сразу вывести за пределы ОДЗП и возникает необходимость в возвращении. При этом строятся последовательности точек, расположенных вблизи границы и внутри ОДЗП, приводящие к зигзагообразному движению вдоль границы с ее пересечением (рис. 4.2). Возврат из точки x^1 в область следует осуществлять вдоль градиента той граничной функции $g_j(x)$, которая оказалась нарушенной. Это обеспечивает движение из точки x^1 в сторону точки экстремума x^* и попадание в точку x^2. Признаком экстремума будет коллинеарность векторов $\nabla F(x)$ и $\nabla g_j(x)$.

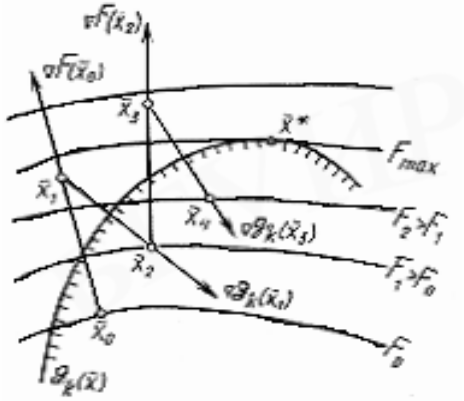


Рис. 4.2. Геометрическая интерпретация процесса решения задачи с нелинейными ограничениями

Описание лабораторной работы

В лабораторной работе рассматриваются две задачи:

1. Нахождение экстремума выпуклой нелинейной функции с линейными ограничениями

$$F(x) = k_1 x_1^{\,2} + k_2 x_2^{\,2} + k_3 x_1 x_2 + k_4 x_1 + k_5 x_2 + k_6,$$

$$\begin{cases} q_1 x_1 + q_2 x_2 \le q_5, \\ q_3 x_1 + q_4 x_2 \le q_6, \\ -x_1 \le 0, \ -x_2 \le 0. \end{cases}$$

2. Нахождение экстремума невыпуклой нелинейной функции с нелинейными ограничениями

$$F(x) = e^{x1}(k_1 x_1^{\,2} + k_2 x_2^{\,2} + k_3 x_1 x_2 + k_4 x_1 + k_5 x_2 + k_6),$$

$$\begin{cases}(x_1 - k_7)(x_2 - k_8) + k_9 \le 0,\\ -x_1 x_2 - k_{10} \le 0,\\ -x_1 \le 0,\\ -x_2 \le 0.\end{cases}$$

Коэффициенты k_i вычисляются в соответствии с подпрограммой генерации варианта задания после введения фамилии и определяются векторами-строками $k1=[k_1\ k_2\ k_3\ k_4\ k_5\ k_6]$, $q=[q_1\ q_2\ q_3\ q_4\ q_5\ q_6]$ и $k2=[k_7\ k_8\ k_9\ k_{10}]$. В дальнейшем значения k и q передаются параметру P1. Необходимо найти минимальное значение функции с учетом ограничений на переменные и проследить изменение координат точек поиска.

Для решения первой задачи используется метод на основе теоремы Куна—Таккера, реализуемый в пакете MATLAB с помощью подпрограммы Q*P* (квадратичное программирование). При выполнении работы открывается файл *fmi*4.1.

Для решения второй задачи используется совокупность рассмотренных ранее методов, реализуемая с помощью подпрограммы Constr. При выполнении работы открывается файл *fmi*4.2.

Все необходимые пояснения даются как комментарии в тексте. Возможно осуществлять минимизацию с использованием формулы вычисления градиента функции цели и без использования формулы, приближенными методами, а также без учета или с учетом условия неотрицательности переменных. Для реализации каждого отдельного случая должны быть открыты соответствующие строки и подразделы программ.

Порядок выполнения работы

1. Запустить файл *fmi*4.1, ввести фамилию и получить вариант квадратичной функции двух переменных и линейных ограничений.

2. Получить трехмерное изображение функции и линии уровня равных значений функции. Оценить координаты точки экстремума в задаче без ограничений.

3. Используя команду help Q*P*, подготовить данные для решения задачи квадратичного программирования.

4. Файл *fmi*4.1 дополнить обращением к команде Q*P* и отметкой маркером точки решения задачи на линиях уровня функции цели.

5. Запустить модифицированный файл *fmi*4.1 и найти решение задачи.

6. Запустить файл *fmi*4.2, ввести фамилию и получить вариант задания.

7. Наблюдать линии уровня равных значений функции, совмещенные с графиками ограничений. Диапазон изменения переменных и величину шага можно менять для корректировки изображения на экране.

Определить ОДЗП и выбрать начальную точку x_0 внутри области (подраздел 3.3).

8. Открыть подраздел 3.4 и наблюдать процедуру поиска минимума $F(x)$ без использования формулы вычисления градиента и учета ограничений на знак переменных. При расширенном выводе результатов на экране печатается таблица (табл. 4.1).

Таблица 4.1

Количество вычислений функции для одномерного поиска	Значения координат точки поиска	Значение функции в точке	Значение ограничения

9. Открыть подраздел 3.5 и наблюдать процедуру поиска минимума $F(x)$ с использованием формулы вычисления градиента без ограничений на знак переменных. Результаты занести в табл. 4.1

10. Открыть подраздел 3.6 и повторить пункты 2 и 3, добавляя к ограничениям условия неотрицательности переменных. Результаты занести в таблицу.

11. Изменить координаты начальной точки и повторить пункты 2, 3, 4.

12. Решить задачу квадратичного программирования в соответствии с вариантом задания по курсовой работе, используя подпрограмму Q*P*.

Контрольные вопросы

6. Перечислить особенности задач нелинейного программирования по сравнению с задачами линейного программирования.

7. В чем суть метода неопределенных множителей Лагранжа?

8. Сформулировать теорему Куна—Таккера.

9. Как записываются условия теоремы Куна—Таккера для дифференцируемых функций $F(x)$ и $q(x)$?

10. Математическая формулировка задачи квадратичного программирования.

11. Как осуществляется выбор направления S^K на очередной итерации метода допустимых направлений?

12. Выбор величины шага α^K в методе допустимых направлений.

13. Признак достижения экстремума в методе допустимых направлений Зойтендейка.

Примеры программ

```
% fmi4 Минимизация выпуклой функции двух переменных с ограничениями
% с ограничениями в виде неравенств и равенств
% с ограничениями на переменные
%       Задача :
% Найти значения переменных x(1) и x(2) минимизирующих функцию F(x)
%
% F(x)=P1(1)*x(1)^2+P1(2)*x(2)^2+P1(3)*x(1)*x(2)+P1(4)*x(1)+P1(5)*x(2)+
%      +P1(6))
% при ограничениях в виде неравенств:
%       P1(7)*x(1) + P1(8)*x(2)    <= P1(11)
%       P1(9)*x(1) +          P1(10)*x(2) <= P1(12)
% и ограничениями на переменные
%                                        x(1) >= 0
%                       x(2) >= 0
% где дополнительной строкой задаются параметры P1(i), i = 1,2,...,12.
% P1(i) – параметры целевой функции и системы ограничений.
% Параметры целевой функции задаются строкой k.
% Параметры  функций ограничений задаются строкой g.
%       P1=[k,g];
  clear,close all,%echo on
 ng=input(' Введите свою фамилию  ','s');
rand('state',sum([datestr(now,11),ng]));
while ng
   k1=round((rand(1,3)-.5)*18);k=ceil(rand(1,3)*9);
   if k(1)*k(2)*4>=k(3)^2;k=[k,k1];break,end
end
txt1=[ng,'  k = ',num2str(k),'; '];
    while ng
   k1=round((rand(1,2)-.5)*18);k3=ceil(rand(1,2)*9);
   if k1(2)*k3(1);g=[k3,k1];break,end
   end
   g(5)=g(1)*g(2);g(6)=round((g(1)*g(4)+g(2)*g(3))/2.5);
```

```
txt2=['g(x);',num2str(g(1)),'*x1+',num2str(g(2)),'*x2 <=',num2str(g(5))];
txt2=['; ',num2str(g(3)),'*x1 + ',num2str(g(4)),'*x2 <= ',num2str(g(6))];
txt2=[txt2,txt3];
P1=[k,g];
    %  2.1.    ЗАДАНИЕ ДИАПАЗОНА ИЗМЕНЕНИЯ ПЕРЕМЕННЫХ
 xmin=-3; xmax=10  ; ymin=-3;  ymax=10; % пределы изменений
 x_shag=(xmax-xmin)/80; y_shag=(ymax-ymin)/70;  % шаг изменений
 x1=xmin:x_shag:xmax;    y1=ymin:y_shag:ymax;    %       набор значений
 [xx,yy]=meshgrid(x1,y1);          % двухмерная сетка значений
                %       2.2.    ВЫЧИСЛЕНИЕ ТОЧЕК ПОВЕРХНОСТИ
zz=(k(1).*xx.^2 + k(2).*yy.^2 + k(3).*xx.*yy +k(4).*xx+ k(5).*yy + k(6));
                %       2.3.    ТРЁХМЕРНАЯ ПОВЕРХНОСТЬ ФУНКЦИИ F(x)
figure(1)
surfl(x1,y1,zz); % surf, mesh,
%2.4. ТРЁХМЕРНОЕ ИЗОБРАЖЕНИЕ ФУНКЦИИ F(x) КОНТУРАМИ ЛИНИЙ
РАВНЫХ ЗНАЧЕНИЙ
figure(2)
 cmin=-0.1; % минимальное значение линии уровня
 cmax=100; % максимальное значение линии уровня
 nc=12; % число линии уровня
 cl=cmin+cmax*logspace(-3,0,nc);  % nc  значений уровня
%cl=[-4 -2 -1 -.5 0 .5 1 2 4 8 16 30];% уровни линий равных значений
[c,h]=contour3(x1,y1,zz,cl,'r'); % контурная поверхность красного цвета
clabel(c,h);   % подписать значения контуров
 view(0,90);  % повернуть поверхность к виду сверху
title(txt1)      % надпись на рисунке, фамилия и значения коэффициентов К
hold on;% ОТКРЫВАЕМ РИСУНОК для нанесения ОГРАНИЧЕНИЙ и точек
решений
pause
                %       2.5.    ВЫЧИСЛЯЕМ И РИСУЕМ ГРАФИКИ ОГРАНИЧЕНИЙ
  g1=[]; x2=-g(2)/g(1)*x1+g(2);g2=[];x3=-g(3)/g(4)*x1+g(6)/g(4);
for i=1:length(x1);
 %выбрасываем точки за пределами рисунка
if and(x2(i)>=ymin,x2(i)<=ymax);g1=[g1,[x1(i);x2(i)]];end
%выбрасываем точки за пределами рисунка
if and(x3(i)>=ymin,x3(i)<=ymax);g2=[g2,[x1(i);x3(i)]];end
end
%графики ограничений
plot(g1(1,:),g1(2,:),g2(1,:),g2(2,:));
title([txt1,t2]);
x=[];
        %       3. ПРОГРАММА ВЫЧИСЛЕНИЯ ЭКСТРЕМАЛЬНОЙ ТОЧКИ x
% задаём начальную точку
x=[1 2]
% создаём программу вычисления экстремальной точки x

        % 4 Нанесение найденной точки x на изображение поверхности
```

```
plot(x(1),x(2),'m*')
hold off;


% fmi3 Минимизация функции двух переменных с нелинейными ограничениями
% с нелинейными ограничениями в виде неравенств и равенств
% с ограничениями на переменные
% Сравнение процессов минимизации
%       Задача :
% Найти значения переменных x(1) и x(2) минимизирующих функцию F(x)
%
% F(x)=exp(x(1))*(P1(1)*x(1)^2+P1(2)*x(2)^2+P1(3)*x(1)*x(2)+P1(4)*x(1)+
%                 +P1(5)*x(2) +P1(6)));
% при нелинейных ограничениях в виде неравенств:
%                ( x(1)-P1(7))*(x(2) -P1(8))+P1(9) <= 0;
%                                              -x(1)*x(2) - P1(10) <= 0
% и ограничениями на переменные
%                                        x(1) >= 0
%                        x(2) >= 0
% где дополнительной строкой задаются параметры P1(i), i = 1,2,...,10.
% P1(i) -- параметры целевой функции и системы нелинейных ограничений.
%         Используется программа CONSTR
%  X=CONSTR('FUN',X,OPTIONS,VLB,VUB,'GRADFUN'),
% где FUN - файл или объект класса inline. Возвращает F и G.
% [F,G]=FUN(X). F минимизируется так, чтобы G <= zeros(size(G)).
% 'GRADFUN'– файл или объект класса inline. Возвращает gf и GC –- частные
%  производные по x от функции F  и ограничений G соответственно.
%  [gf,GC] = GRADFUN(X).
clear,close all
  ng=input(' Введите свою фамилию  ','s');
 rand('state',sum([datestr(now,11),ng]));
 while ng
    k1=round((rand(1,3)-.5)*18);k=ceil(rand(1,3)*9);
    if k(1)*k(2)*4>=k(3)^2;k1=[k,k1];break,end
    end
k2=[ceil(rand*6),2+fix(rand*8),ceil(rand*6),(ceil(rand*6)+4)*4]/4;
   k=[k1,k2];
P1=k;
%P2='c+'; % задаёт цвет и маркёр точки решения
 P3=1; % задаёт паузу в секундах между точками
     %  2.1      ЗАДАНИЕ ДИАПАЗОНА ИЗМЕНЕНИЯ ПЕРЕМЕННЫХ
xmin=-10; xmax=4  ; ymin=-4;   ymax=5; % пределы изменений
x_shag=(xmax-xmin)/80;  y_shag=(ymax-ymin)/70;  % шаг изменений
x1=xmin:x_shag:xmax;     y1=ymin:y_shag:ymax;     %       набор значений
[xx,yy]=meshgrid(x1,y1);           % двухмерная сетка значений
```

```
                %       2.2     ВЫЧИСЛЕНИЕ ТОЧЕК ПОВЕРХНОСТИ
zz=exp(xx).*(k(1).*xx.^2 + k(2).*yy.^2 + k(3).*xx.*yy +k(4).*xx+ k(5).*yy + k(6));
                %       2.3     ТРЁХМЕРНАЯ ПОВЕРХНОСТЬ
 figure(1)
 hsv2=hsv; % палитра радуги
hsv3=[hsv2(11:64,:); hsv2(1:10,:)];%перестановка цветов в палитре
 surfHndl=surface(x1,y1,zz,'EdgeColor',[.8 .8 .8]);
 axis off;
 view(5,75);
colormap(hsv3);
%figure(5)
%surf(x1,y1,zz); view(10,70)
                %       2.4     ИЗОБРАЖЕНИЕ ПОВЕРХНОСТИ    ЛИНИЯМИ
РАВНЫХ ЗНАЧЕНИЙ
 figure(2)
 cmin=-0; % минимальное значение линии уровня
 cmax=400; % максимальное значение линии уровня
 nc=12; % число линии уровня
 cl=cmin+cmax*logspace(-3,0,nc);  % nc  значений уровня
 % cl=[.03 .1 .3  1 3 9  25 75  200 600];%уровни линий равных значений
 [c,h]=contour3(x1,y1,zz,cl,'r');
 clabel(c,h); % нарисовать и обозначить линий равных значений
 view(0,90);% drawnow
 hold on;% сохранение рисунка для нанесения ограничений и точек решений
   pause
                %       2.5     ВЫЧИСЛЯЕМ И РИСУЕМ ГРАФИКИ ОГРАНИЧЕНИЙ
% Формулы ограничений соответствуют  функциям в файлах fun2 и gfad2
  g1=[]; x2=-P1(10)./x1; g2=[]; x3=P1(8)-P1(9)./(x1-P1(7));
txt1=[ng,'  k = ',num2str(k1),'; '];
txt2=[' g(x);  -x1*x2 <= ',num2str(P1(10)),';  (x1 - ',num2str(P1(7))];
txt3=[')*(x2 - ',num2str(P1(8)),') + ',num2str(P1(9)),' <= 0'];
txt2=[txt2,txt3];
for i=1:length(x1);
 %выбрасываем точки за пределами рисунка
   if and(x2(i)>=ymin,x2(i)<=ymax);g1=[g1,[x1(i);x2(i)]];end
 if and(x3(i)>=ymin,x3(i)<=ymax);g2=[g2,[x1(i);x3(i)]];end
end
%графики ограничений
plot(g1(1,:),g1(2,:),g2(1,:),g2(2,:));
title([txt1,txt2]);
pause
            %  3.3         ЗАДАНИЕ НАЧАЛЬНОЙ ТОЧКИ x0
 x0 = [-.5 -1.5]; disp([' Начальная точка   X0 = ',num2str(x0)]);
  %P2='c+'; % задаёт цвет и маркёр точки решения
 P3=1; % задаёт паузу в секундах между точками
 options = []; options(1) =2; % расширенный вывод промежуточных значений
```

```
 options(14) =20; % Ограничиваем число вычислений
        % 3.4. МИНИМИЗАЦИЯ БЕЗ ИСПОЛЬЗОВАНИЯ ФОРМУЛЫ
ГРАДИЕНТА ФУНКЦИИ
 txt5=('МИНИМИЗАЦИЯ БЕЗ ИСПОЛЬЗОВАНИЯ ФОРМУЛЫ ГРАДИЕНТА
ФУНКЦИИ')
 disp(txt5);
 P2='b+'; % задаёт цвет и маркёр точки решения
[x, options] = constr(fun2, x0, options,[],[],[],P1,P2,P3);pause
        % 3.5. МИНИМИЗАЦИЯ С ИСПОЛЬЗОВАНИЕМ ФОРМУЛЫ
ГРАДИЕНТА ФУНКЦИИ
 txt6=('МИНИМИЗАЦИЯ С ИСПОЛЬЗОВАНИЕМ ФОРМУЛЫ ГРАДИЕНТА
ФУНКЦИИ')
 disp(txt6);
P2='cx'; % задаёт цвет и маркёр точки решения
[x, options] = constr(fun2, x0, options,[],[],grad2,P1,P2,P3); pause
%       3.6. ДОБАВЛЯЕМ УСЛОВИЕ НЕОТРИЦАТЕЛЬНОСТИ ПЕРЕМЕННЫХ
X
disp('ДОБАВЛЯЕМ УСЛОВИЕ        НЕОТРИЦАТЕЛЬНОСТИ ПЕРЕМЕННЫХ
X')
vlb = zeros(1,2); % ограничения снизу X >= 0
vub = [];         % нет ограничений сверху
 disp(txt5);
    P2='bx'; % задаёт цвет и маркёр точки решения
[x, options] = constr(fun2, x0, options, vlb, vub,[],P1,P2,P3); pause
 disp(txt6);
P2='c+'; % задаёт цвет и маркёр точки решения
[x, options] = constr(fun2, x0, options, vlb, vub,grad2,P1,P2,P3); pause
           %  3.7.       ЗАДАНИЕ НОВОЙ НАЧАЛЬНОЙ ТОЧКИ x0
 x0 = [2 4]; disp(['  Начальная точка   X0 = ',num2str(x0)]);
 disp(txt5);
 P2='gx'; % задаёт цвет и маркёр точки решения
[x, options] = constr(fun2, x0, options, vlb, vub,[],P1,P2,P3); pause
 disp(txt6);
P2='g+'; % задаёт цвет и маркёр точки решения
[x, options] = constr(fun2, x0, options, vlb, vub,grad2,P1,P2,P3); pause
   hold off; % выключить режим сохранения рисунка
```

Св. план 2003, поз. 18

Учебное издание

Павлова Анна Валентиновна,
Хаджинов Михаил Касьянович

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ СИСТЕМ

МЕТОДИЧЕСКОЕ ПОСОБИЕ

к лабораторным работам для студентов специальностей
53 01 03 «Автоматическое управление в технических системах»
и 53 01 07 «Информационные технологии и управление
в технических системах»
дневной и вечерней форм обучения

Редактор Т.А. Лейко
Корректор Е.Н. Батурчик
Компьютерная верстка Т.В. Шестакова

Подписано в печать 30.04.2003. Формат 60x84 1/16. Бумага офсетная.
Печать ризографическая. Гарнитура «Таймс». Усл. печ. л. 2,91.
Уч.-изд. л. 2,7. Тираж 200 экз. Заказ 25.

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Лицензия ЛП № 156 от 30.12.2002.
Лицензия ЛВ № 509 от 03.08.2001.
220013, Минск, П. Бровки, 6.