СПРАВОЧНИК

В. Е. БОЛНОКИН
П. И. ЧИНАЕВ

АНАЛИЗ И СИНТЕЗ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ НА ЭВМ

АЛГОРИТМЫ И ПРОГРАММЫ

МОСКВА „РАДИО И СВЯЗЬ"
1991

ББК 32.965
Б79
УДК 681.513.5.037(03)

Редакция литературы по информатике и вычислительной технике

Болнокин В. Е., Чинаев П. И.

Б 79 Анализ и синтез систем автоматического управления на ЭВМ. Алгоритмы и программы: Справочник.— М.: Радио и связь, 1991.—256 с.: ил.

ISBN 5-256-00410-7

Дается описание современных методов, алгоритмов и программ для решения на ЭВМ задач анализа устойчивости и качества динамических систем автоматического управления, синтеза оптимального управления и оценивания неизвестных параметров, а также оптимизации непрерывных и дискретных параметров систем управления. Отражен весь процесс проектирования на ЭВМ сложных технических систем: выбор модели, математического метода, разработка алгоритма, программ и, наконец, расчеты на ЭВМ.

Для инженерно-технических работников, занимающихся разработкой и исследованием современных систем управления.

Б $\frac{2402010000\text{-}031}{046(01)\text{-}91}$ 125-91 **ББК 32.965**

Справочное издание

БОЛНОКИН ВИТАЛИЙ ЕВГЕНЬЕВИЧ, ЧИНАЕВ ПЕТР ИВАНОВИЧ

АНАЛИЗ И СИНТЕЗ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ НА ЭВМ. АЛГОРИТМЫ И ПРОГРАММЫ.

Справочник.

Заведующий редакцией Г. И. Козырева. Редактор Т. М. Толмачева. Переплет художника Н. А. Пашуро. Художественный редактор Н. С. Шеин. Технический редактор Г. З. Кузнецова. Корректор З. Г. Галушкина.

ИБ № 2054

Сдано в набор 14.03.90. Подписано в печать 04.02.91. Формат $60\times88^{1}/_{16}$. Бумага офсетная № 2. Гарнитура Таймс. Печать офсетная. Усл. печ. л. 15,68. Усл. кр.-отт. 16,05. Уч.-изд. л. 14,86. Тираж 10 000 экз. Изд. № 22789. Зак. № 132.

Цена 3 р. 50 к.
Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Московская типография № 4 Госкомпечати СССР. 129041, Москва, Б. Переяславская, 46.

ISBN 5-256-00410-7

Предисловие

В инженерной практике, особенно при автоматизированном выборе наилучших проектных решений, широкое распространение получили математические методы анализа и синтеза сложных технических систем, позволяющие повысить эффективность технологических и производственных процессов в различных отраслях народного хозяйства. Применение эффективных математических методов в системах автоматического управления стало возможным с появлением быстродействующих ЭВМ. Однако использование широких возможностей, предоставляемых ЭВМ, связано с трудностями реализации математических методов на практике, и прежде всего с трудностями разработки алгоритмов и программ.

Большую помощь в создании алгоритмическо-программного обеспечения современных инженерных исследований может оказать Государственный фонд, включающий тщательно оттестированные алгоритмы и программы на различных языках программирования. Однако отсутствие методических материалов для широкого круга специалистов снижает эффективность применения современных средств вычислительной техники в проектировании систем управления. Кроме того, любой работоспособный в теоретическом отношении метод без учета его особенностей может быть неэффективен при решении конкретных задач.

Цель настоящего справочника — помочь овладеть навыками применения современного математического аппарата теории автоматического управления для расчетов на ЭВМ. Использование приведенных в книге алгоритмов и программ позволит существенно облегчить реализацию сложных оптимальных методов при практическом проведении исследований и разработок систем автоматического управления и уменьшить затраты на разработку системы.

Авторами принята следующая форма изложения: вначале кратко освещается теория (методология), затем дается алгоритм и программа, в заключение приводится пример практического применения предлагаемых алгоритмов и программ. При написании программ использовались широко распространенные языки программирования ПЛ/1 и Фортран.

В основу справочника положена книга «Анализ и синтез систем автоматического управления на ЭВМ. Алгоритмы и программы», выпущенная в 1986 г. Данное издание переработано методологически, дополнено новыми разделами, посвященными адаптивным методам выбора параметров систем, новым методам оптимизации управления. Все это позволило охватить весь процесс проектирования технических систем и практически все вопросы проектирования систем управления, встречающиеся в инженерной практике.

Введение

В.1. Роль ЭВМ при исследовании и проектировании систем управления

В существующих концепциях проектирования больших систем на передний план выдвигается комплексная автоматизация процесса проектирования, невозможная без внедрения ЭВМ во все этапы и уровни процесса проектирования. С точки зрения системного анализа в процессе проектирования технической системы можно выделить следующие основные этапы (рис. В.1):

1. Осознание потребности в проектировании системы с заданными свойствами. Формирование критериев эффективности функционирования системы.

2. Разработка концептуальной модели системы и описание ее в терминах специальных языковых и логических средств.

3. Разработка математической модели системы и ее реализация в виде программы (совокупности программ) на выбранном языке программирования и для конкретных ЭВМ.

4. Анализ математической модели путем проведения на ЭВМ многократных расчетов по программам, описывающим модель системы.

5. Оптимизация модели, которая производится с помощью специального математического и программного аппарата. В процессе оптимизации обычно происходит многократное итерационное обращение к математической модели системы. Разумеется, реализовать оптимизацию сложных систем можно лишь на быстродействующих ЭВМ.

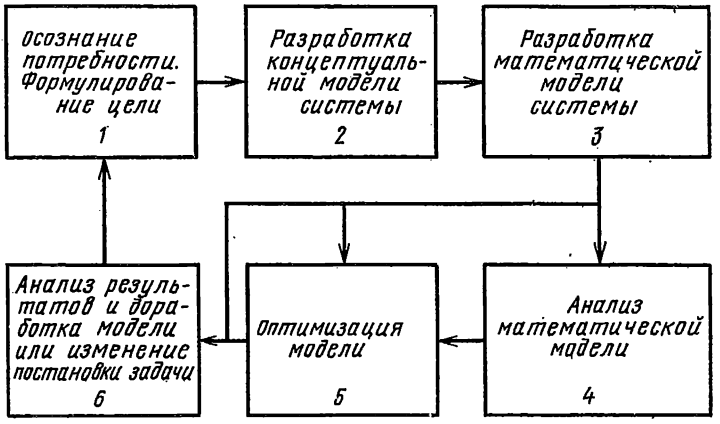


Рис. В.1

6. Анализ результатов и доработка модели или в случае необходимости изменение постановки задачи. Последняя операция, естественно, наиболее болезненна, так как возвращает процесс проектирования к исходному положению.

Представленная схема процесса проектирования может (так обычно и бывает на практике) неоднократно повторяться, пока не будет выработано приемлемое решение, которое и передается в производство. Разумеется, данная схема процесса проектирования упрощенная, в ней не отражено действие многих важных факторов. Например, наряду с математическим моделированием используется полунатурное и натурное моделирование, процесс оптимизации может осуществляться с помощью физических процессов (экспериментов, испытаний) и т. д. Большие объемы вычислений, характерные для всех этапов проектирования сложных технических систем, можно произвести только с помощью современных быстродействующих ЭВМ, обладающих достаточно большими резервами памяти, таких, как ЕС ЭВМ и «Эльбрус». При этом по мере перехода от ранних этапов проектирования (этапы 1, 2) к более поздним (этапы 3, 6) масштабы применения ЭВМ увеличиваются.

Включение ЭВМ непосредственно в контур управляющей системы позволяет расширить возможности наиболее эффективного формирования и исполнения алгоритма управления. В этом случае ЭВМ могут применяться по-разному — в автономном режиме, диалоговом или в режиме последовательной переработки информации (супервизорное управление).

Автономный режим предполагает полную автоматизацию процесса управления, когда на ЭВМ полностью возложены обработка поступающей информации и формирование требуемых управляющих воздействий.

В режиме *диалогового управления* осуществляется одновременное включение в контур и человека-оператора, который общается с ЭВМ посредством различных средств (дисплеев, графопостроителей и т. д.). Автоматическая система в диалоговом режиме управления использует команды по частям, общение с оператором и ЭВМ. Человеку и ЭВМ предоставляется возможность совместно принимать решение, что позволяет управлять объектом в сложных условиях.

В режиме *супервизорного управления* ЭВМ автоматически управляет системой, однако переход от одного этапа к другому диктуется человеком. Супервизорный контур, замыкающийся через человека-оператора, предназначен для выработки стратегии управления. Супервизорное управление существенно необходимо при наличии больших временных запаздываний в исполнении команд и в случае большой неопределенности условий функционирования системы.

Основным недостатком работы ЭВМ в контуре управления системами являлась неэффективность их использования, так как машины часто работали с недогрузкой. Переход к мини- и микроЭВМ, появление персональных ЭВМ во многом решили проблему рентабельности и ознаменовали новый подход к проектированию систем управления в целом. Теперь вместо цетрализованного управления с широко развитой сетью внешних устройств доминирующей стала идеология распределенного управления с внедрением широкой сети ЭВМ и микропроцессоров, локально встроенных в отдельные устройства.

В.2. Алгоритмы и программы

Термин «алгоритм» прочно вошел как в современный научно-технический, так и в бытовой лексикон. Но, несмотря на смысловую определенность, понятие алго-

ритма меняет свои оттенки в зависимости от того, в каком контексте оно употребляется. Поэтому следует оговорить, что мы будем понимать под алгоритмом в настоящей работе.

Обратимся к ставшему уже классическим пояснению понятия алгоритма, данному А. А. Марковым: «В математике принято под алгоритмом понимать точное предписание, определяющее вычислительный процесс... Следующие три черты характерны для алгоритмов...

а) точность предписания, не оставляющая места произволу, и его общепринятость — определенность алгоритма;

б) возможность исходить из варьируемых в известных пределах исходных данных — массовость алгоритма;

в) направленность алгоритма на получение некоторого искомого результата — результативность».

Современную точку зрения на информационную сущность понятия алгоритма выражает Н. А. Криницкий: ...Алгоритм — это прежде всего средство преобразования символьных конструкций, а вместе с ними и представленной в них информации. Это преобразование должно выполняться устройствами, для которых алгоритм — это управляющая информация... Алгоритм — средство не только описания процессов, но и создания тех операций, которые должны выполняться как шаги процессов (кстати, и необязательно последовательно)».

Таким образом, понятие алгоритма подразумевает сформулированную цель, язык, описывающий порядок действий и, возможно, физическое устройство, реализующее алгоритм. В настоящей книге мы не будем акцентировать внимание читателя на теоретических возможностях абсолютно строгого описания языка, реализующего алгоритм (современной теории алгоритмов посвящены, например, работы [5, 6]). Для нас это обычный язык формул, принятый в современной математической литературе, посвященной прикладным задачам.

Исходя из степени подробности описания алгоритма можно предложить следующую классификацию представления алгоритмов:

описание общей методики вычислений;

формальная схема последовательности операций вычислений;

алгоритмические программы для ЭВМ.

В этой классификации осуществляются постепенная специализация и детализация алгоритмов. Программа является наиболее подробным и исключающим всякую двусмысленность (речь идет, разумеется, об отлаженной программе) выполнением алгоритма, реализованного на определенном языке программирования (а часто и применительно к конкретной вычислительной системе). Схема алгоритма однозначно определяет процесс вычислений, но допускает различные реализации в виде программы (использование различных языков, в рамках одного языка его разных возможностей и способов программирования и т. д.). С другой стороны, представление алгоритма в понимании п. 1 в силу своей общности пригодно для решения широкого диапазона задач, работу его отдельных фрагментов можно оптимизировать с целью максимальной адаптации самого вычислительного метода к специфике решаемой задачи.

Авторы старались отразить в книге в основном весь процесс исследования и проектирования на ЭВМ систем автоматического управления: выбор подходящей модели математического метода, программы и, наконец, проведение численных

расчетов на ЭВМ. В то же время основная направленность книги — оказать читателю помощь при практическом решении на ЭВМ реальных задач инженерного проектирования — заставила авторов сконцентрировать свои усилия на описании алгоритмов, максимально ориентированных на использование на ЭВМ. Этому требованию наиболее удовлетворяют алгоритмы групп 2 и 3. Особое значение в практике решения задач исследования и проектирования систем автоматического управления играет программное обеспечение указанных работ. Разработка цифровой компьютерной системы управления, за исключением аппаратуры специального назначения, является неполной, а система бесполезной для практического применения, пока не будет обеспечена соответствующими программами, правильность которых подтверждена тестами. При этом следует отметить, что разработка программного обеспечения — программ применения (проблемных), операционных систем и вспомогательных сервисных программ, как и разработка аппаратуры, находится в развитии.

Следует подчеркнуть, что стоимость разработки программного обеспечения превышает (на порядок и даже больше) стоимость самой ЭВМ, для которой оно предназначено [7]. Более того, разработка программного обеспечения проводится значительно медленнее, чем разработка аппаратных средств автоматизации, требует интенсивных дополнительных трудовых затрат на доводку и сопровождение. С учетом всех этих факторов становится ясной важность создания специализированного математического обеспечения, ориентированного на машинное проектирование современных систем автоматического управления.

В.3. Описание комплекта программ

В книгу включено 17 программ, позволяющих читателю решать с помощью ЭВМ практические задачи анализа и синтеза систем автоматического управления:

GURV — анализ устойчивости линейных систем управления с помощью критерия Гурвица;

RAUS — анализ устойчивости линейных систем управления с помощью критерия Рауса;

OPR — анализ устойчивости линейных систем с бесконечно большими коэффициентами;

INTED, INTEC — вычисление математического ожидания и дисперсии выходного сигнала динамической системы с дискретным и непрерывным временем соответственно;

DIN — синтез оптимального управления методом динамического программирования;

OPTIMA — синтез оптимального управления непрерывной динамической нелинейной системы с заданным критерием качества управления;

KALMAN — фильтрация для линейных стохастических дискретных систем, т. е. синтез дискретного фильтра Калмана;

KALBUS — фильтрация для нелинейных стохастических непрерывных систем;

FMINIM — оптимизация одномерного критерия по методу золотого сечения;

DIRECT — оптимизация непрерывных параметров системы с помощью метода прямого поиска (конфигураций);

FLEXI — оптимизация непрерывных параметров системы при наличии ограничений (равенств и неравенств) с помощью метода скользящего допуска;

GOMORY — оптимизация дискретных параметров системы при наличии линейных критерия и ограничений;

VEKSP, NAPRO — решение аналогичной задачи, но для выпуклой функции критерия качества;

AODIM — оптимизация дискретных параметров для систем с нерегулярными функциями критериев качества;

OPTI — вычисление статистической оценки экстремального значения критерия, заданного на дискретном множестве параметров системы.

Часть программ разработана авторами, другая часть — переработанные для языка ПЛ/1 модификации программ на других языках, подготовленных разными авторами (ссылки на соответствующий источник приводятся при описании программы). В предлагаемых программах использовалась возможность языка ПЛ/1 по динамическому распределению памяти, т. е. все програмы, работающие с векторами и матрицами, могут оперировать массивами произвольной размерности, ограниченными в основном лишь ресурсами памяти, доступными для ЭВМ.

Одним из наиболее важных вопросов при разработке подобного рода пакетов программ является вопрос о выборе основного языка программирования. Авторы остановились на языке ПЛ/1 по следующим причинам: он обеспечивает эффективное описание сложных алгоритмов вычислений над большими совокупностями данных, представляемых в разнообразных формах; кроме того, он является широко распространенным языком, охотно используемым в инженерной практике. Это облегчает организацию совместного применения предлагаемых в данном пособии программ с программами пользователя. Более того, возможности языка ПЛ/1 позволяют осуществлять связь с программами пользователя, подготовленными на других языках, например Фортране (см. приложение 1).

Описание работы каждой из приведенных в книге программ оформлено одинаково. В описание входит информация о входных и выходных параметрах программы, режимах ее работы. В случае необходимости приводится более подробная информация о форме представления выходных данных программы. Наиболее важные фрагменты программы снабжены соответствующими комментариями.

Для удобства пользователя все приведенные программы сопровождаются типовыми примерами, иллюстрирующими возможные способы подготовки и использования программ при решении конкретных задач.

Обращение ко всем программам осуществляется с помощью стандартного оператора языка ПЛ/1 CALL либо присвоения некоторой переменной значения процедуры-функции. Обеспечить работу вызываемой программы пользователь может с помощью описания атрибутов данных и задания их начальных значений в собственной вызывающей программе.

Как известно, точность вычисления с помощью программ зависит от точности данных. В приведенных программах использовалась точность, обеспечиваемая так называемым принципом умолчания языка ПЛ/1. При необходимости пользователь может варьировать точность с помощью явного задания соответствующего атрибута. Необходимые для этого конструкции языка программирования ПЛ/1 приведены в приложении. Для иллюстрации возможности применения приведенных в книге программ рассмотрим следующий пример.

Для оптимального управления динамической системой необходимо знать ее состояние. Оценки состояния линейной динамической системы можно вычислить методом

оптимальной фильтрации по следующим формулам:

$$\dot{\hat{\mathbf{x}}}(t)=\mathbf{A}(t)\hat{\mathbf{x}}(t)+\mathbf{K}(t)[\mathbf{y}(t)-\mathbf{H}(t)\hat{\mathbf{x}}(t)];\ \dot{\mathbf{P}}(t)=\mathbf{A}(t)\mathbf{P}(t)+\mathbf{P}(t)\mathbf{A}^{\mathrm{T}}(t)+\mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^{\mathrm{T}}(t)- \tag{B1}$$
$$-\mathbf{P}(t)\mathbf{H}^{\mathrm{T}}(t)\mathbf{R}^{-1}(t)\mathbf{H}(t)\mathbf{P}(t),\ \mathbf{K}(t)=\mathbf{P}(t)\mathbf{H}^{\mathrm{T}}(t)\mathbf{R}^{-1}(t),$$

где $\hat{\mathbf{x}}(t)$ — n-мерный вектор оценок состояния системы, $\hat{\mathbf{x}}(t_0)=\hat{\mathbf{x}}_0$ ($\hat{\mathbf{x}}_0$ задано); $\mathbf{P}(t)$ — ковариационная матрица вектора $\hat{\mathbf{x}}(t)$ размером $n\times n$, $\mathbf{P}(t_0)=\mathbf{P}_0$ ($\mathbf{P}_0$ задано); $\mathbf{A}(t)$ — матрица размером $n\times n$, определяющая динамику системы; $\mathbf{G}(t)$ — матрица размером $n\times m$, характеризующая воздействие внешних возмущений по координатам вектора состояния; $\mathbf{H}(t)$ — матрица размером $l\times n$, определяющая динамику измерений; $\mathbf{y}(t)$ — l-мерный случайный вектор измерений.

Математическая модель вектора измерений:

$$\mathbf{y}(t)=\mathbf{H}(t)\mathbf{x}(t)+\mathbf{v}(t), \tag{B2}$$

где $\mathbf{v}(t)$ — l-мерный вектор шумов измерителей; $\mathbf{x}(t)$ — n-мерный вектор состояния динамической системы, $\mathbf{x}(t_0)=\mathbf{x}_0$ ($\mathbf{x}_0$ задано). Математическая модель динамической системы определяется формулой

$$\dot{\mathbf{x}}(t)=\mathbf{A}(t)\mathbf{x}(t)+\mathbf{G}(t)\mathbf{w}(t), \tag{B3}$$

где $\mathbf{w}(t)$ — m-мерный вектор возмущений; $\mathbf{Q}(t)$, $\mathbf{R}(t)$— матрицы интенсивностей внешних возмущений и погрешности измерений размером $m\times m$ и $l\times l$. Возмущения и шумы гауссовские с нулевым математическим ожиданием.

Алгоритм вычисления оценок состояния методом оптимальной фильтрации может быть разработан на основании формул (В.1)—(В.3) и оформлен как процедура на языке ПЛ/1 (см. программу KALBUS). Формальными параметрами процедуры будут следующие:

n, m, l — параметры, определяющие размерность системы;

$\mathbf{x}$, $\hat{\mathbf{x}}$, $\mathbf{P}$ — векторы состояния, оценок состояния и ковариационная матрица;

t, h — независимая переменная и шаг интегрирования;

имя процедуры вычисления матриц **A**, **G**, **H**, **Q**, **R**.

Поскольку эти матрицы являются описанием конкретной динамической системы, то алгоритм вычисления их элементов и процедуру на языке программирования разрабатывает специалист в этой области.

Из этого примера видно, что использование отлаженных и четких алгоритмов и процедуры оптимальной фильтрации может существенно сократить время специалиста на решение задачи, машинное время на отладку задачи и делает доступным использование сложного в теоретическом отношении метода оптимальной фильтрации для инженеров, имеющих недостаточную для разработки самих оптимальных методов математическую подготовку.

Глава 1.
Исследование устойчивости систем управления

Одним из основных требований, предъявляемых к качеству функционирования системы автоматического управления, является требование устойчивости. Система автоматического управления считается *устойчивой*, если она, будучи выведена из состояния установившегося движения некоторой причиной (внешним воздействием, изменением начального состояния), возвращается в исходное состояние после прекращения действия этой причины.

Вынести суждение об устойчивости систем автоматического управления позволяет анализ расположения корней характеристического уравнения, минуя вычисления самих корней, на основе критериев устойчивости. В настоящее время в теории автоматического управления критерии устойчивости делятся на две большие группы: алгебраические и частотные. *Алгебраические* критерии основаны на анализе непосредственно характеристического уравнения системы. Такими критериями являются, например, широкоизвестные критерии Рауса и Гурвица. *Частотные* критерии позволяют определить устойчивость системы автоматического управления с помощью аналитического или экспериментального исследования частотных характеристик элементов этой системы. Популярными в технических приложениях критериями этой группы являются критерии Михайлова и Найквиста. Они в той или иной форме базируются на априорном знании характеристического уравнения системы

$$a_0 p^n + a_1 p^{n-1} + \ldots + a_{n-1} p + a_n = 0, \qquad (1.1)$$

которое при изложении всех последующих алгоритмов считается заданным.

1.1. Оценка устойчивости по критерию Гурвица

Косвенные условия оценки устойчивости линейных систем, которые позволяют составить алгоритм оценки устойчивости, не прибегая к нахождению корней, нашел немецкий математик А. Гурвиц. Алгоритм Гурвица основан на построении из коэффициентов характеристического уравнения (1.1) специальной матрицы — матрицы Гурвица:

$$\begin{Vmatrix} a_1 & a_3 & a_5 & \ldots \\ a_0 & a_2 & a_4 & \ldots \\ 0 & a_1 & a_3 & \ldots \\ 0 & a_0 & a_2 & \ldots \\ 0 & 0 & a_1 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{Vmatrix}$$

В соответствии с критерием Гурвица система будет устойчивой, если определитель матрицы Гурвица и определители всех ее диагональных миноров положительные.

В первую строку матрицы Гурвица записываются коэффициенты характеристического уравнения с нечетными индексами, во вторую — с четными, последующие

пары строк получают из первых двух, смещая их элементы на 1, 2, 3,... столбца. Образующиеся свободные места слева, а также те элементы, которые находятся в строке справа от a_{n-1}, a_n, заполняются нулями. Таким образом, матрица Гурвица имеет n строк и n столбцов.

По главной диагонали располагаются коэффициенты характеристического уравнения, начиная с a_1 до a_n. Диагональные миноры Δ_k ($k=n-1, \ldots, 1$) получаются из определителя матрицы Гурвица Δ_n путем вычеркивания справа и снизу последовательно по одной строке и одному столбцу, по две строки и два столбца и т. д. Условие устойчивости системы заключается в том, чтобы при $a_0>0$ все диагональные миноры были положительные:

$$\Delta_1 = a_1 > 0, \quad \Delta_2 = \begin{vmatrix} a_1 & a_3 \\ a_0 & a_2 \end{vmatrix} > 0, \ldots, \quad \Delta_n = a_n \Delta_{n-1} > 0. \tag{1.2}$$

Таким образом, алгоритм Гурвица состоит из следующих основных этапов:

1. Составление определителя Гурвица Δ_n.
2. Вычисление определителей $\Delta_1, \ldots, \Delta_n$.
3. Анализ условия положительности определителей $\Delta_1, \ldots, \Delta_n$.

При выполнении этого условия делается вывод об устойчивости исследуемой системы, в противном случае система неустойчива.

ПРОГРАММА GURV

Назначение: анализ устойчивости линейных систем автоматического управления с помощью критерия Гурвица.

Параметры:

A — одномерный массив, составленный из коэффициентов характеристического многочлена (1.1) $a_0p^n+a_1p^{n-1}+\ldots+a_n$, $A(1)=a_0$, $A(2)=a_1$, ..., $A(N)=a_n$;

N — порядок характеристического многочлена: $N=n+1$;

IND — скалярный показатель устойчивости системы:

$$\text{IND} = \begin{cases} 0, \text{ если система неустойчива,} \\ 1, \text{ если система устойчива.} \end{cases}$$

Обращение: CALL GURV (A, N, IND);

Основные этапы работы:

1) проверка положительности коэффициентов характеристического полинома;

2) заполнение определителя Гурвица;

3) циклическая проверка положительности определителя Гурвица и всех его диагональных миноров.

В процессе работы программа GURV использует обращение к процедуре вычисления определителя произвольной квадратной матрицы DET.

Пример. Анализ устойчивости линейной системы с характеристическим полиномом $p^3+1{,}48p^2+4{,}6p+4$. В результате получено IND=1, на основании которого можно сделать вывод об устойчивости данной системы.

```
      GURV: PROCEDURE(A,N,IND);
       DECLARE A(*);
        BEGIN;
         DECLARE B(N,N);
/* 1 */
         IND=1;
         DO I=1 TO N;
          IF A(I) <= 0 THEN GO TO ME
         END;
         B=0;
/* 2 */
          DO J=1 TO N/2+1;
           J1=2*J;   J2=2*J-1;
           B(1,J)=A(J1);
           B(2,J)=A(J2);
          END;
          DO I=3 TO N BY 2;
           DO J=FLOOR(I/2)+1 TO N-1;
             B(I,J)=B(I-2,J-1);
           END;
           IF I+1 <= N THEN
           DO J=FLOOR(I/2)+1 TO N-1;
             B(I+1,J)=B(I-1,J-1);
           END;
          END;
          DO I=1 TO N-1;
           B(N,I),B(I,N)=0;
          END;
          B(N,N)=A(1);
/* 3 */
           DO I=N-1 TO 1 BY -2;
            BEGIN;
             DECLARE X(I,I);
              DO J=1 TO I;
               DO L=1 TO I;
                X(J,L)=B(J,L);
               END;
              END;
              CALL DET(X,I,DE);
              IF DE <= 0 THEN GO TO
             END;
            END;
          GO TO MET2;
MET1:     IND=0;
MET2:     END;
      END GURV;


        DET: PROCEDURE(AA,NN,DE);
         DECLARE AA(*,*);
         D=1;
         DO K=1 TO NN;
          RMAX=0;
          DO I=K TO NN;
           T=AA(I,K);
           IF ABS(T) > ABS(RMAX) THE
            DO;
             RMAX=T;   J=1;
```

```
          END;
         END;
        IF  RMAX  =  0  THEN
         DO;
          D=0;  GO  TO  FIN;
         END;
        IF  J   =  K  THEN
         DO;
          D=-D;
          DO  I=K  TO  NN;
           T=AA(J,I);
           AA(J,I)=AA(K,I);
           AA(K,I)=T;
          END;
         END;
          IF  K  >=  NN  THEN  GO  TO  MET;
          DO  I=K+1  TO NN;
           T=AA(I,K)/RMAX;
           DO  J=K+1  TO  NN;
            AA(I,J)=AA(I,J)-T*AA(K,J);
           END;
          END;
MET:      D=D*AA(K,K);
        END;
        FIN:DE=D;
        END DET;
```

```
        TEST: PROCEDURE OPTIONS(MAIN);
          DCL  A(4);
          N=3;
          A(1)=1;  A(2)=1.48;  A(3)=4.6;  A(4)=4;
          CALL  GURV(A,N,IND);
          PUT SKIP DATA(A,N,IND);
        END TEST;
```

1.2. Оценка устойчивости по критерию Рауса

На практике при анализе устойчивости систем с заданным характеристическим уравнением часто пользуются критерием Рауса. Алгоритм Рауса, как и алгоритм Гурвица, строится на основе анализа специальной таблицы (матрицы)

$$\begin{bmatrix} \tilde{a}_{11}=a_0 & \tilde{a}_{21}=a_2 & \dots \\ \tilde{a}_{21}=a_1 & \tilde{a}_{22}=a_3 & \dots \\ \tilde{a}_{31} & \tilde{a}_{32} & \dots \\ \tilde{a}_{41} & \tilde{a}_{42} & \dots \\ \cdots & \cdots & \end{bmatrix} \tag{1.3}$$

Она составляется из коэффициентов характеристического уравнения системы (1.1). В первую строку записываются коэффициенты характеристического уравнения с четными индексами (0,2, ...), во вторую — с нечетными (1, 3,...). В остальных строках элементы $\tilde{a}_{ik}(i \geqslant 3)$ вычисляются с помощью рекуррентных соотношений

$$\tilde{a}_{ik}=\tilde{a}_{i-2,\,k+1}-d_i\tilde{a}_{i-1,\,k+1}. \tag{1.4}$$

Здесь $d_i=\tilde{a}_{i-2,1}/\tilde{a}_{i-1,1}$; $\tilde{a}_{1k}=a_{2(k-1)}$, $\tilde{a}_{2k}=a_{2k-1}$, $k=1, 2, \dots$.

Эти формулы и определяют, по существу, алгоритм Рауса. Проверка устойчивости системы осуществляется в ходе заполнения таблицы Рауса. В соответствии с критерием Рауса для устойчивости необходимо и достаточно, чтобы выполнялось неравенство $d_i > 0$ ($i=3, \ldots, n+2$) при $a_0 > 0$ или (что эквивалентно) $a_0 > 0$, $a_1 > 0$, $\tilde{a}_{31} > 0$, ..., $\tilde{a}_{n+1,1} > 0$. Если какой-нибудь из коэффициентов отрицательный, то система неустойчива. Более того, число перемен знаков у коэффициентов первого столбца таблицы Рауса дает число корней характеристического уравнения системы, которые расположены в правой полуплоскости.

ПРОГРАММА RAUS

Назначение: анализ устойчивости линейных систем автоматического управления с помощью критерия Рауса.

Параметры:

A — массив, составленный из коэффициентов характеристического уравнения $a_0p^n + a_1p^{n-1} + \ldots + a_{n-1}p + a_n = 0$ системы: $A(1) = a_0$, $A(2) = a_1$, $A(N) = a_n$;

N — размер массива A: $N = n+1$;

IND — скалярный показатель устойчивости системы:

$$\text{IND} = \begin{cases} 0 & \text{для неустойчивой системы,} \\ 1 & \text{для устойчивой.} \end{cases}$$

K — число перемен знаков первого столбца таблицы Рауса.

Атрибуты всех параметров присваиваются по умолчанию.

Обращение: CALL RAUS(A, N, IND, K);

Основные этапы работы:

1) формирование из коэффициентов характеристического уравнения таблицы Рауса;

2) проверка условия положительности элементов первого столбца таблицы Рауса;

3) подсчет числа перемен знаков у элементов первого столбца таблицы Рауса.

Пример. Анализ устойчивости системы с характеристическим уравнением $p^6 + 6p^5 + 21p^4 + 44p^3 + 62p^2 + 52p + 100 = 0$ (таким образом, $N=7$). Получены следующие значения выходных параметров: IND=0 (т. е. система неустойчива), K=2.

```
RAUS: PROCEDURE(A,N,IND,K);
  DECLARE  A(*);
  IND=1; N1=(N+1)/2;
  M=FLOOR(N1);
  BEGIN;
  DECLARE  R(3:N),C(N,M);
/* 1 */
  R=0; C=0;
  DO I=1  TO  M;
    K=N-2*(I-1);
    IF  K  >  0  THEN  C(1,I)=A(K);
    K=N-2*(I-1)-1;
    IF  K  >  0  THEN  C(2,I)=A(K);
  END;
   DO  I=3  TO  N;
    R(I)=C(I-2,1)/C(I-1,1);
     DO  J=1  TO  M-1;
      C(I,J)=C(I-2,J+1)-R(I)*C(I-1,J+1);
     END;
   END;
```

```
/* 2 */ -
          DO  I=1  TO  N;
          IF  C(I,1)  <=  0  THEN
            DO;
              IND=0;  GO TO MET;
            END;
          END;
            K=0;
          GO TO MET1
/* 3 */
     MET:   K=0;
            DO  I=1  TO  N-1;
              IF  C(I,J)*C(I+1,J)  <=  0  THEN  K=K+1
            END;
     MET1:  END:
         END  RAUS;


     TEST: PROCEDURE  OPTIONS(MAIN);
        DCL  A(7);
        N=7;
        A(7)=1;  A(6)=6;  A(5)=21;  A(4)=44;
        A(3)=62; A(2)=52; A(1)=100;
        CALL  RAUS(A,N,IND,K);
     END  TEST;
```

1.3. Оценка устойчивости по критериям Михайлова и Найквиста

Оригинальный графоаналитический критерий оценки устойчивости систем автоматического управления с помощью анализа специальным образом построенных кривых, характеризующих динамические свойства исследуемой системы, предложил в 1938 г. советский ученый А. В. Михайлов.

Характеристический многочлен линейной системы n-го порядка

$$D(p) = a_0 p^n + a_1 p^{n-1} + \ldots + a_{n-1} p + a_n, \quad n \geqslant 0. \tag{1.5}$$

Подставим в него значение $p = j\omega$:

$$D(j\omega) = X(\omega) + j\,Y(\omega),$$

где $X(\omega) = a_n - a_{n-2}\omega^2 + \ldots + (-1)^n a_0 \omega^n$, $Y(\omega) = a_{n-1}\omega - a_{n-3}\omega^3 + \ldots + (-1)^{n-1} a_1 \omega^{n-1}$.

Соответствующие годографы в плоскости (X, Y) при различных n называются *кривыми Михайлова*. *Критерий Михайлова* можно сформулировать так: для устойчивости системы с характеристическим многочленом (1.5) необходимо и достаточно, чтобы при изменении величины ω от 0 до ∞ изменение аргумента функции $D(j\omega)$ равнялось $n\pi/2$. Это означает, что годограф $D(j\omega)$ должен пройти против часовой стрелки последовательно n квадрантов. Примеры кривых Михайлова, соответствующих устойчивым и неустойчивым системам, приведены на рис. 1.1, *а* и *б*.

Алгоритм, реализующий критерий Михайлова, прост и состоит в последовательном вычислении значений функций $X(\omega)$ и $Y(\omega)$ для $\omega = 0, \Delta\omega, 2\Delta\omega, \ldots$ ($\Delta\omega$ — шаг квантования). В процессе вычислений определяется изменение аргумента функции $\Delta \arg D(j\omega)$. В конце проверяется условие критерия Михайлова $\Delta \arg D(j\omega) = n\pi/2$.

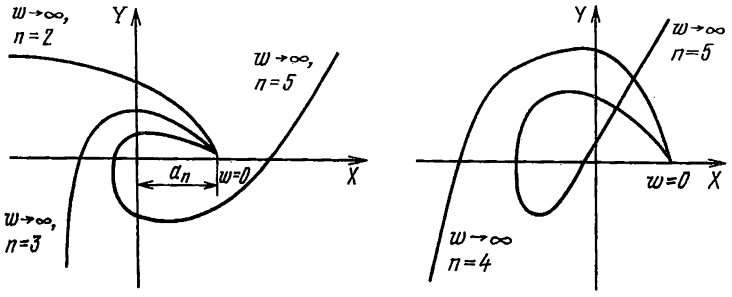


Рис. 1.1. Примеры кривых Михайлова для устойчивых (*а*) и неустойчивых (*б*) систем

При проведении расчетов бывает полезно использовать следующие свойства кривых Михайлова: при $\omega=0$ величины $X(\omega)$ и $Y(\omega)$ вычисляются по формулам $X(0)=a_n$, $Y(0)=0$. Так как сразу можно предположить, что все коэффициенты a_i ($i=0,\ldots, n$) положительные, то при $\omega\to\infty$ будем иметь либо $X(\omega)\to\infty$, $Y(\omega)\to-\infty$ при n четном, либо $X(\omega)\to-\infty$, $Y(\omega)\to\infty$ при n нечетном. Это позволяет в практических расчетах использовать асимптотические свойства кривой Михайлова для определения конечного значения параметра ω, участвующего в расчете.

Оценим устойчивость, например, электромеханической следящей системы в разомкнутом состоянии, передаточная функция которой

$$W(p)=K/[p(1+T_y p)(1+T_м p)],$$

где $K=58$ с$^{-1}$ — общий коэффициент усиления разомкнутой системы; $T_м=0{,}57$ с — постоянная времени двигателя; $T_y=0{,}01$ с — постоянная времени усилителя. Характеристический полином замкнутой системы равен сумме полиномов числителя и знаменателя передаточной функции разомкнутой системы:

$$H_з(p)=p(1+T_y p)(1+T_м p)+K=T_y T_м p^3+(T_y+T_м)p^2+p+K.$$

Для построения кривой Михайлова определим вещественную и мнимую части функции $H_з(p)$:

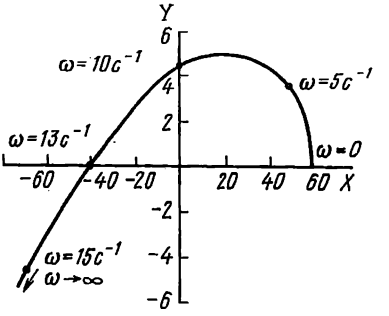
$$X(\omega)=\mathrm{Re}H_з(j\omega)=K-(T_y+T_м)\omega^2=58-0{,}58\omega^2;$$

$$Y(\omega)=\mathrm{Im}\,H_з(j\omega)=\omega-T_y T_м \omega^3=\omega-5{,}7\cdot 10^{-3}\omega^2.$$

Вычислим значения $X(\omega)$ и $Y(\omega)$ для ряда значений ω. Как видно из рис. 1.2, кривая Михайлова последовательно проходит через три квадранта, следовательно, система устойчива.

Алгоритм оценки устойчивости по *критерию Найквиста* базируется на математических моделях, описывающих частотные характеристики (в частности, переда-

Рис. 1.2. Построение кривой Михайлова



точные функции) разомкнутой цепи системы автоматического управления. На основании анализа разомкнутой цепи можно сделать вывод об устойчивости или неустойчивости замкнутой системы.

Предположим, что система автоматического управления устойчива в разомкнутом состоянии. Пусть передаточная функция разомкнутой цепи есть $W(p)$. Рассмотрим вспомогательную функцию

$$W_1(p) = W(p) + 1 = H_з(p)/H_р(p), \tag{1.6}$$

где $H_з(p)$ и $H_р(p)$— соответственно характеристические многочлены замкнутой системы и ее разомкнутой цепи. Подставим в (1.6) значение $p=j\omega$. Вследствие сделанных предположений об устойчивости разомкнутой цепи по критерию Михайлова аргумент $H_р(j\omega)$ при $0 \leqslant \omega \leqslant \infty$ изменяется на $n\pi/2$. Но для устойчивости замкнутой системы также необходимо, чтобы соответствующее изменение аргумента $H_з(j\omega)$ равнялось $n\pi/2$. Следовательно, условие устойчивости можно записать в виде

$$\Delta\arg W_1(j\omega) = \Delta\arg H_з(j\omega) - \Delta\arg H_р(j\omega) = 0.$$

Таким образом, для устойчивости замкнутой системы необходимо и достаточно, чтобы годограф $W_1(j\omega)$ не охватывал начало координат.

Возвращаясь к частотной характеристике разомкнутой системы $W(j\omega)$, можно сформулировать критерий Найквиста: в случае устойчивости разомкнутой цепи системы автоматического управления для устойчивости замкнутой системы необходимо и достаточно, чтобы годограф $W(j\omega)$ при изменении ω от 0 до ∞ не охватывал точку с координатами $(-1, 0)$.

В вычислительном аспекте алгоритм оценки устойчивости по критерию Найквиста аналогичен алгоритму по критерию Михайлова. Однако при его реализации на ЭВМ возможны трудности, связанные с алгоритмической формализацией события, заключающегося в неохвате точки $(-1, 0)$ годографом $W(j\omega)$. В этом случае может оказаться полезным следующее правило: годограф $W(j\omega)$ не охватывает точку с координатами $(-1, 0)$, если при пересечении годографом действительной оси левее точки $(-1, 0)$ число пересечений сверху вниз равно числу пересечений снизу вверх.

Определим устойчивость разомкнутой системы управления устойчивым объектом, используя критерий Найквиста. Пусть передаточная функция такой системы

$$W(p) = K(1 + \tau p)/p(1 + T_1 p)(1 + T_0^2 p^2),$$

где $K = 1$ — общий коэффициент усиления; $\tau = 0{,}1$ с — постоянная времени корректирующего устройства; $T_1 = 0{,}2$ с — постоянная времени исполнительного устройства; $T_0 = 0{,}5$ с — постоянная времени объекта. Оценим устойчивость замкнутой системы с помощью критерия Найквиста. Амплитудно-частотная характеристика (АЧХ) разомкнутой системы:

$$A(\omega) = \frac{K\sqrt{1 + (\omega\tau)^2}}{\sqrt{1 + (\omega T_1)^2}\,|1 - (\omega T_0)^2|} = \frac{\sqrt{1 + (0{,}1\omega)^2}}{\sqrt{1 + (0{,}2\omega)^2}\,|1 - (0{,}5\omega)^2|}.$$

Фазочастотная характеристика (ФЧХ):

$$\psi(\omega) = \begin{cases} \operatorname{arctg}\omega\tau - \operatorname{arctg}\omega T_1 = \operatorname{arctg}0{,}1\omega - \\ - \operatorname{arctg}0{,}2\omega \text{ при } \omega < 1/T_0 = 2\text{с}^{-1}, \\ \operatorname{arctg}\omega\tau - \operatorname{arctg}\omega T_1 - 180° = \operatorname{arctg}0{,}1\omega - \\ - \operatorname{arctg}0{,}2\omega - 180° \text{ при } \omega > 1/T_0 = 2\text{с}^{-1}. \end{cases}$$

Вычислим $A(\omega)$ и $\psi(\omega)$ для ряда значений частоты ω. Построим амплитудно-фазовую характеристику (АФХ) системы (рис. 1.3). При частоте $\omega = 1/T_0 = 2$ с$^{-1}$ АФХ имеет разрыв. Ветви АФХ, соответствующие частотам $\omega \to 1/T_0 - 0$ и $\omega \to 1/T_0 + 0$, дополним полуокружностью бесконечно большого радиуса, которую проведем по часовой стрелке от ветви АФХ, соответствующей $\omega \to 1/T_0 - 0$, к ветви, соответствующей $\omega \to 1/T_0 + 0$. Из рис. 1.3 видно, что АФХ разомкнутой системы охватывает точку $(-1, 0)$. Следовательно, замкнутая система неустойчива.

Оценить устойчивость этой системы можно и другим способом. Из выражения для ФЧХ следует, что при $\tau > T_1$ для всех частот $\psi(\omega) > -180°$. Поэтому АФХ при

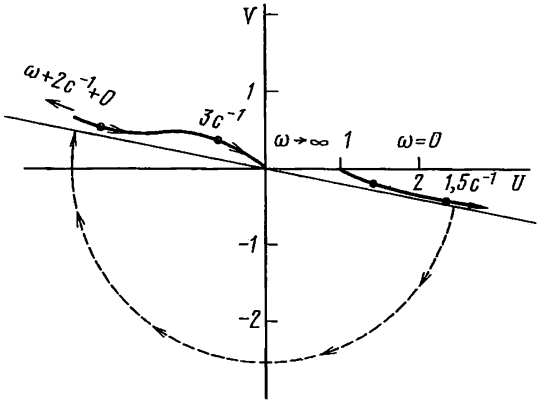


Рис. 1.3. Построение амплитудно-фазовой характеристики системы

$\tau > T_1$ не заходит в третий квадрант, и система устойчива при любых $K < 0$ и T_0. При $\tau < T_1 \psi(\omega) < -180°$ для всех частот $\omega > 1/T_0$, поэтому часть АФХ, соответствующих частотам $\omega > 1/T_0$, лежит в третьем квадранте, причем ветвь АФХ, соответствующая $\omega \to 1/T_0 + 0$, уходит в бесконечность. Поэтому при $\tau < T_1$ система неустойчива при любых K и T_0.

Критерий Найквиста был сформулирован в предположении, что разомкнутая система устойчива. Рассмотрим общий случай, когда m ($m \geqslant 0$) корней характеристического уравнения разомкнутой системы находятся в правой полуплоскости, а остальные $n - m$ — в левой. Без ограничения общности можно принять, что обратная связь системы отрицательная (если обратная связь положительная, то умножим передаточную функцию разомкнутой системы на коэффициент $K = -1$).

Теперь можно сформулировать критерий Найквиста в общем случае наличия m корней в правой полуплоскости: чтобы замкнутая система была устойчивой, необходимо и достаточно, чтобы АФХ ее разомкнутой системы при изменении ω от 0 до ∞ охватывала точку $(-1; 0)$ (при движении в положительном направлении) ровно $m/2$ раз.

ПРОГРАММА MIH

Назначение: анализ устойчивости линейных систем с помощью критерия Михайлова.

Параметры:

N — порядок характеристического полинома исследуемой системы;

A — массив коэффициентов характеристического полинома, размер (0:N);

WMAX — максимальное значение используемой в алгоритме Михайлова частоты;

DW — шаг изменения частоты;

EPS — точность вычисления изменения аргумента годографа кривой Михайлова (точность вычисления $\Delta \arg D(j\omega)$);

EPSI — нижняя граница приращения аргумента при описании годографом кривой Михайлова;

IND — индикатор устойчивости системы в зависимости от того, сколько квадрантов прошла кривая Михайлова (больше, равно или меньше N): IND=1 — система устойчива; IND=2 — система неустойчива, имеется возможность увеличить значение параметра WMAX, что позволит уточнить анализ для расширенного спектра частот; IND=0 — число квадрантов, пройденных кривой Михайлова, больше порядка характеристического уравнения системы, что невозможно теоретически (кривая Михайлова всегда проходит число квадрантов, не превышающее порядок характеристического уравнения), но тем не менее может встретиться при практических расчетах на ЭВМ, что обусловлено накоплением ошибок округления, случайного сбоя ЭВМ и т. д. При повторении исходов программисту можно рекомендовать изменить входные параметры программы (например, уменьшить DW, увеличить EPS), атрибуты параметров и повторить расчет.

Атрибуты параметров определяются по умолчанию.

Обращение: CALL MIH(A, N, WMAX, DW, EPS, EPSI, IND);

Основные этапы работы:

1) основной цикл изменения параметра частоты;

2) вычисление вещественной части годографа Михайлова;

3) вычисление мнимой части годографа Михайлова;

4) вычисление аргумента;

5) расчет приращения аргумента годографа Михайлова;

6) проверка условий устойчивости системы.

Аналогично можно разработать программу, реализующую алгоритм оценки устойчивости по критерию Найквиста. При этом изменения коснутся лишь того фрагмента MIH, в котором непосредственно осуществляется проверка условий устойчивости.

Пример. Исследование устойчивости системы второго порядка с характеристическим многочленом p^2+p+1.

Исходные данные: N=2, A(0)=A(1)=A(2)=1, WMAX=500, DW=0,5, EPS=10^{-3}, EPS=10^{-7}.

Результат: в течение 2,5 с на ЭВМ ЕС-1033 получено верное значение параметра IND=1, что соответствует устойчивости системы.

```
MIH: PROCEDURE(A,N,WMAX,DW,EPS,EPS1,IND);
     DCL A(*);
     FIS=0; P1=3.141592654:
     F=0;
     I=2; P=0; SIGNS=1; J1=1;
/* 1 */
     DO WHILE(F<=WMAX);
     P=(I-1)*DW;
/* 2 */
     J=0; X=0; J1=0;
     DO WHILE(J<=N);
     IF P=0 THEN X=0;
     ELSE
     X=X+A(N-J)*(-1)**J1*P**J;
     J1=J1+1;
     J=J+2;
     END;
/* 3 */
     J=1; Y=0; J1=0;
     DO WHILE(J<=N);
     Y=Y+A(N-J)*(-1)**J1*P**J;
     J1=J1+1;
     J=J+2;
     END;
/* 4 */
     IF X =0 THEN FI=ATAN(Y/X);
     IF X=0 & Y>0 THEN FI=0.5*PI;
     IF X=0 & Y<0 THEN FI=-0.5*PI;
/* 5 */
     SIGNN=SIGN(X*Y);
     IF(SIGNN =SIGNS) THEN
     DO;
     J1=J1+1;IF FLOOR(J1/2)*2=J1 THEN DFI=FI+PI-FIS;
     ELSE DFI=FI-FIS; F=F+DFI;
     SIGNS=SIGNN;
     FIS=FI;
     GO TO ME;
     END;
     DFI=FI-FIS; F=F+DFI; FIS=FI;
/* 6 */
ME: IF F>N*FI*0.5+EPS THEN
     DO;
     IND=0; GO TO MET;
```

```
    END;
    IF DFI<EPS;
    THEN GO TO MEE;
    I=I+1;
    END;
MEE:  IF ABS(F-N*PI*0.5) <= EPS THEN
    DO;
    IND=1; GO TO MET;
    END;
    IND=2;
MET:  END MIH;

TEST:  PROCEDURE OPTIONS(MAIN);
    DCL A(0:2);
    N=2;
    A=1;
    WMAX=500;
    EPS=0.001;
    EPS1=1.E-7;
    DW=0.5;
    CALL MIH(A,N,MMAX,DW,EPS,EPS1,IND);
END TEST;
```

1.4. Оценка устойчивости импульсных систем

Динамика импульсных систем описывается с помощью уравнений в конечных разностях или дискретного преобразования Лапласа (Z-преобразования). Как и для непрерывных систем, при оценке устойчивости импульсных систем достаточно исследовать характеристическое уравнение замкнутой импульсной системы

$$G(p) = a_0 e^{pnT} + a_1 e^{p(n-1)T} + \ldots + a_{n-1} e^{pT} + a_n \qquad (1.7)$$

(T — константа времени).

Импульсная система будет устойчива, если все нули $p_1, \ldots, p_n$ характеристического многочлена будут расположены в левой полуплоскости: $\mathrm{Re}\, p_i < 0$ ($i = 1, \ldots, n$). Если заменить переменные $e^{pT} = z$, $p = \ln z/T$, то характеристический многочлен можно записать в следующем виде:

$$\tilde{\sigma}(z) = G(\ln z/T) = a_0 z^n + a_1 z^{n-1} + \ldots + a_n.$$

При этом левая полуплоскость $\mathrm{Re}\, p < 0$ трансформируется во внутреннюю часть единичного круга плоскости z, а правая $\mathrm{Re}\, p > 0$ — во внешнюю часть. Это преобразование позволяет переформулировать условие устойчивости: импульсная система будет устойчива, если все нули $z_1 = e^{p_1 T}, \ldots, z_n = e^{p_n T}$ будут внутренними: $|z_i| < 1$ ($i = 1, \ldots, n$).

Например, пусть передаточная функция замкнутой импульсной системы

$$W_з(z) = 0{,}11/(z^2 - 17{,}8z + 0{,}89).$$

Характеристическое уравнение системы $z^2 - 17{,}8z + 0{,}89 = 0$ имеет корни $z_{1,2} = 0{,}89 \pm \pm j0{,}346$. Так как $|z_{1,2}| < 1$, то в соответствии со сформулированным критерием устойчивости данная система устойчива.

Следует отметить, что непосредственное вычисление корней характеристического уравнения — достаточно трудоемкая процедура, может быть реализована лишь с

помощью соответствующих численных методов. Поэтому при анализе устойчивости импульсных систем целесообразно использовать косвенные методы исследования расположения корней характеристического уравнения.

Покажем, как можно для этого применить алгоритмы, основанные на алгебраических критериях устойчивости. Произведем еще одну замену переменных $z=(1+\omega)/(1-\omega)$. Характеристический многочлен преобразуется к следующему виду:

$$(\omega-1)^n\tilde{\sigma}\left(\frac{\omega+1}{\omega-1}\right)=\bar{G}(\omega)=a_0(\omega+1)^n+$$

$$+a_1(\omega+1)^{n-1}(\omega-1)+\ldots+a_n(\omega-1)^n=$$

$$=b_0\omega^n+b_1\omega^{n-1}+\ldots+b_n.$$

Здесь коэффициенты $b_0, \ldots, b_n$ определяются по следующим формулам:

$$b_0=\sum_{i=0}^{n}a_ib_n=\sum_{i=0}^{n}(-1)^ia_i,$$

$$b_i=\sum_{i=1}^{n}\sum_{l=0}^{j}C_i^lC_{n-i}^{j-l}(-1)^l,\ j=1,\ldots,n-1. \qquad (1.8)$$

Это преобразование трансформирует внутренность, границу и внешность единичного круга $|z|<1$ соответственно в левую полуплоскость, мнимую ось и правую полуплоскость в плоскости ω. Теперь можно применить к многочлену $\tilde{\sigma}(\omega)$ описанные ранее алгоритмы и программы, основанные на алгебраических критериях устойчивости.

ПРОГРАММА PERES

Назначение: вычисление коэффициентов $b_0, \ldots, b_n$ по известным коэффициентам исходного характеристического уравнения $a_0, \ldots, a_n$.

Параметры:

N — порядок характеристического полинома;

A — массив, составленный из коэффициентов характеристического полинома, размер (0:N);

B — массив искомых коэффициентов b_i ($i=0, \ldots, n$), размер (0:N).

Атрибуты всех параметров определяются по умолчанию.

```
PERES:  PROCEDURE(A,B,N);
    DCL A(*),B(*);
    R0=0; RN=0;
    DO I=0 TO N;
       R0=R0+A(I);
       RN=RN+(-1)**I*A(I);
    END;
    B(0)=R0; B(N)=RN;
    DO J=1 TO N-1;
     S=0;
      DO I=1 TO N;
```

```
    R=0; N1=N-I;
      DO L=0 TO J;
       L1=J-L; W=C(L,I);
       R=R+W*C(L1,N1)*(-1)**I;
      END;
      S=S+R*A(I);
    END;
    B(J)=S;
  END;
  C:PROCEDURE(K,M);
    IF K<0 THEN RETURN(0,0);
    IF M<0 THEN RETURN(0,0);
    IF (M-K) < 0 THEN RETURN(0,0);
    IF M=0 THEN RETURN(1,0);
    Z=1.0; K2=K+1;
    DO K1=K2 TO M;
     Z=K1*Z;
    END;
    F=1.0; M1=M-K;
    IF M1>1 THEN
     DO I1=1 TO M1;
     F=F*I1;
     END;
    T=Z/F;
    RETURN(T);
   END C;
  END PERES;
```

1.5. Построение областей устойчивости

Рассмотренные алгоритмы позволяют оценить устойчивость линейных систем автоматического управления на основе исследования свойств корней характеристического уравнения системы, причем предполагалось, что коэффициенты уравнения постоянны. На практике эти коэффициенты часто зависят от ряда параметров $q_1, ..., q_r$, описывающих различные технические факторы: $a_i = a_i(q_1, ..., q_r)$ $(r \geqslant 0)$. В этом случае становится важной задача исследования области возможных значений параметров Q, чтобы выделить подобласти $Q_{уст}$ и $Q_{неуст}$, в которых система соответственно устойчива и неустойчива: $Q_{уст} \cup Q_{неуст} = Q$.

Процедуру численного анализа устойчивости системы в области возможных значений параметров Q нетрудно организовать с помощью следующего алгоритма: вся область Q квантуется с некоторым интервалом $\Delta q_1, ..., \Delta q_r$ по каждому параметру $q_1, ..., q_r$ соответственно. В каждой точке построенной таким образом аппроксимирующей решетки устойчивость системы проверяется с помощью одного из предложенных ранее методов. Повторяя эту процедуру для всех точек и объединяя в одно множество точки, в которых система устойчива, получаем оценку области устойчивости системы.

Если каждый параметр q_i $(i=1, ..., r)$ изменяется в пределах $[q_i^{min}, q_i^{max}]$, то общее число обращений к алгоритму оценки устойчивости

$$N = \prod_{i=1}^{r} \left[\left(q_i^{max} - q_i^{min} \right) / \Delta q_i + 1 \right]$$

(здесь $[\cdot]$ — символ целой части).

На практике при построении областей устойчивости систем автоматического управления очень эффективным является непосредственное определение условий,

описывающих границы областей устойчивости. Например, пусть передаточная функция разомкнутой системы

$$W(p) = K/(1 + T_1 p)(1 + T_2 p)(1 + T_3 p),$$

где $T_2 = 0{,}2$ с, $T_3 = 0{,}1$ с. Требуется построить область устойчивости системы в плоскости параметров K, T_1. Характеристический полином замкнутой системы:

$$D(p) = (1 + T_1 p)(1 + T_2 p)(1 + T_3 p) + K = \\ = 0{,}02 T_1 p^3 + (0{,}02 + 0{,}3 T_1) p^2 + (0{,}3 + T_1) p + K + 1.$$

Для получения уравнений границы области устойчивости, соответствующих наличию в характеристическом многочлене системы бесконечного и нулевого корня, приравняем нулю коэффициент при старшей степени характеристического многочлена и свободный член характеристического многочлена. В результате получим следующие уравнения границ области устойчивости: $T_1 = 0$, $K = -1$.

Уравнение для границы области устойчивости, соответствующей нахождению системы на колебательной границе устойчивости, найдем, приравняв нулю предпоследний определитель Гурвица $\Delta_{n-1} = 0$. В данном случае это условие принимает вид

$$(0{,}02 + 0{,}3 T_1)(0{,}3 + T_1) = 0{,}02 T_1 (1 + K).$$

Отсюда получаем, что

$$K = (1 + 15 T_1)(0{,}3 + T_1)/T_1 - 1.$$

В соответствии с этими соотношениями построим границы области устойчивости (рис. 1.4). Линия, соответствующая $K = -1$, практически сливается с осью абсцисс. Областью устойчивости является область A, так как для любой точки внутри нее выполняется условие устойчивости.

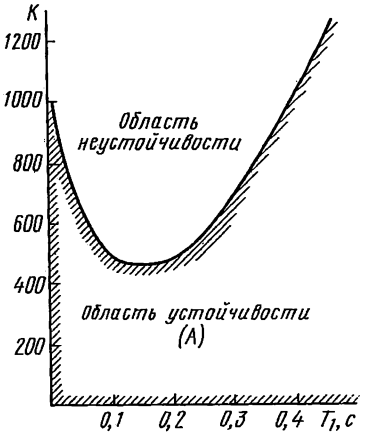


Рис. 1.4. Построение области устойчивости

Существует естественная, основанная на понятии геометрического расстояния, параметризация области устойчивости систем автоматического управления. Суть ее в следующем. В соответствии с критерием устойчивости Ляпунова совокупность корней характеристического уравнения должна быть расположена слева от мнимой оси. Сделаем перенос оси влево на расстояние от мнимой оси до ближайшего к ней корня $-v$: $d = p + v$, где d — такой же оператор, как и p, но в смещенной области. Если в характеристическое уравнение системы

$$a_0 p^n + a_1 p^{n-1} + \ldots + a_{n-1} p + a_n = 0$$

вместо p подставить $d-v$, то новое уравнение будет записано в виде

$$b_0 d^n + b_1 d^{n-1} + \ldots + b_{n-1} d + b_n = 0. \tag{1.9}$$

Установим зависимости между коэффициентами a_i и b_i $(i=0, \ldots, n)$. Пересчет коэффициентов b_i через a_i и v осуществляется с помощью следующего алгоритма. Предположим, что $a_0=b_0=1$ (это не ограничивает общности). Составим матрицу

$$\mathbf{M} = \begin{Vmatrix} a_0 & 0 & \ldots & 0 \\ -a_0 C_n^1 & -a_1 C_{n-1}^1 & \ldots & 0 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ (-1)^{n-1} a_0 C_n^{n-1} & (-1)^{n-1} a_1 C_{n-1}^{n-1} & \ldots & 0 \\ (-1)^n a_0 & (-1)^n a_1 & \ldots & (-1)^n a_n \end{Vmatrix},$$

$$\mathbf{B}^{\mathrm{T}} = \|b_0, b_1, \ldots, b_n\|^{\mathrm{T}}, \mathbf{V}^{\mathrm{T}} = \|v^n, v^{n-1}, \ldots, 1\|^{\mathrm{T}}.$$

Вектор коэффициентов $\mathbf{B}$ выражается через матрицу $\mathbf{M}$ и вектор $\mathbf{V}$: $\mathbf{B}=\mathbf{M}\cdot\mathbf{V}$. В треугольной матрице $\mathbf{M}=\|m_{ij}\|_{(n+1)\times(n+1)}$ выше главной диагонали располагаются нули, в k-й $(k=0, \ldots, n)$ строке элементы m_{kl} вычисляются по формуле $m_{kl}=a_i C_{n-i}^k (-1)^k (0 \leqslant i \leqslant k)$.

Теперь для характеристического уравнения (1.9) можно применить алгоритмы оценки устойчивости с помощью, например, алгебраических критериев. При этом, поскольку параметр v характеризует расположение ближайшего к мнимой оси корня, можно получить параметрическое условие, ограничивающее пространство параметров устойчивости системы.

ПРОГРАММА PEREB

Назначение: построение областей устойчивости; позволяет осуществить полный перебор аппроксимирующей решетки области возможных значений параметров Q, для каждой точки $q \in Q$ устойчивость системы с заданными значениями параметров $q=(q_1, \ldots, q_r)$ можно проверить с помощью одной из программ оценки устойчивости.

Параметры:

X — массив размера M×N, i-й столбец которого последовательно состоит из возможных значений компонента q_i;

N — число параметров, влияющих на устойчивость системы;

M — максимальное число возможных значений компонентов q_i $(i=1, \ldots, n)$;

IS — одномерный массив размера N, состоящий из верхних границ значений компонентов q_i $(i=1, \ldots, n)$ (заметим, что M=max IS (i)).

Атрибуты всех параметров определяются по умолчанию.

В процессе работы программы формируется N-мерный массив S, представляющий собой очередную выборку из множества Q. На основании массива вычисляются текущие значения коэффициентов характеристического многочлена, а затем осуществляется непосредственно вызов одной из программ оценки устойчивости RAUS, GURV, MIH. Расположение точки вызова указано в программе с помощью соответствующих комментариев.

```
PEREB:PROCEDURE(X,N,M,IS);
      DECLARE X(*,*),IS(*);
      BEGIN;
      DECLARE (IZX,S) (N);
      IT=1; IZT=0;
      DO I=2 TO N;
       IX(I)=1;
      END;
      IX(1)=0;
MET1: IT=0;
MET2: IT=IT+1;
      IF IX(IT)=IS(IT) THEN
      DO;
       IF IT=N THEN GOTO MK;
       IX(IT)=1; GOTO MET2;
      END;
      IX(IT)=IX(IT)+1;
      DO I=1 TO N;
       S(I)=X(IX(I),I);
      END;
     PUT SKIP DATA(S);
      GO TO MET1;
   MK:END;
      END PEREB;
```

1.6. Устойчивость систем при неограниченно возрастающих параметрах

Во многих практических задачах [10, 11] высокую точность системы по всем координатам можно обеспечить при неограниченном возрастании коэффициентов усиления в управляющем контуре. Чтобы обеспечить устойчивость многомерной системы n-го порядка при любом коэффициенте усиления по любой из координат (например, i-й), необходимо в соответствующий контур ввести воздействие по производной $(\nu_i - 2)$-го порядка (ν_i — степень собственного оператора i-го контура) [37].

Пусть уравнение движения многомерной системы задано в виде

$$[\mathbf{E}+\mathbf{A}(s)\mathbf{C}(s)]\mathbf{X}=\mathbf{A}(s)\mathbf{C}(s)\mathbf{X}^0(s)+\mathbf{H}(s)\mathbf{F}. \tag{1.10}$$

Характеристическое уравнение системы

$$|\mathbf{E}+\mathbf{A}(s)\mathbf{C}(s)|=0$$

или в развернутом виде

$$\begin{bmatrix} 1+b_{11}(s) & b_{12}(s) & \dots & b_{1m}(s) \\ b_{21}(s) & 1+b_{22}(s) & \dots & b_{2m}(s) \\ \dots & \dots & \dots & \dots \\ b_{m1}(s) & b_{m2}(s) & \dots & 1+b_{mm}(s) \end{bmatrix} = 0. \tag{1.11}$$

Раскрыв определитель (1.11), получим в общем случае оператор

$$[1+b_{11}(s)]\mathbf{A}_{11}(s)+b_{21}(s)\mathbf{A}_{21}(s)+\dots+b_{m1}(s)\mathbf{A}_{m1}(s)=0,$$

где $\mathbf{A}_{ij}(s)$ — соответствующие алгебраические дополнения. Учитывая структуру операторов $b_{ij}(s)$, можно последнее уравнение преобразовать к виду

$$F_{N_2}(s) + K_i F_{N_1}(s) = 0, \tag{1.12}$$

где K_i — коэффициент усиления i-го контура, который может принимать бесконечно большие значения; N_2, N_1 — степени многочленов F_{N_2} и F_{N_1}.

Представим уравнение (1.10) в развернутом виде:

$$\begin{aligned} &m\left(A_0 s^{N_2} + A_1 s^{N_2-1} + \ldots + A_{N_2}\right) + \\ &+\left(B_0 s^{N_1} + B_1 s^{N_1-1} + \ldots + B_{N_1}\right) = 0, \end{aligned} \tag{1.13}$$

где $m = 1/K_i$. Если m зафиксировано, то характеристическое уравнение имеет N_2 корней ($N_2 > N_1$). При $m \to 0$ N_1 корней будут приближаться к корням вырожденного уравнения

$$B_0 s^{N_1} + B_1 s^{N_1-1} + \ldots + B_{N_1} = 0,$$

остальные $N_2 - N_1$ корней стремятся к бесконечности. Это следует из того, что при конечных значениях $N_2 - N_1$ корней выражение (1.13) будет конечным, а при $s \to \infty$

$$m\left(\frac{A_0 s^{N_2} + A_1 s^{N_2-1} + \ldots + A_{N_2}}{B_0 s^{N_1} + B_1 s^{N_1-1} + \ldots + B_{N_1}}\right) \to 1.$$

Условие устойчивости будет соблюдаться только тогда, когда все корни останутся левее мнимой оси. Направление движения корней (слева или справа) зависит от соотношения между N_2 и N_1 и коэффициентами A_i и B_i. Если слева, то устойчивость всей системы при $m \to 0$ будет определяться по вырожденному уравнению (1.12). Рассмотрим основные случаи.

1. $N_2 - N_1 = \nu = 1$ (в бесконечность уходит всего один корень). Разделив многочлен $A_0 s^{N_2} + \ldots + A_{N_2}$ на $B_0 s^{N_1} + \ldots + B_{N_1}$, получим:

$$m\frac{A_0}{B_0}\left(s - C + \frac{\alpha_1 s^{N_2-1} + \alpha_2 s^{N_2-2} + \ldots + \alpha_{N_2-1}}{B_0 s^{N_1} + B_1 s^{N_1-1} + \ldots + B_{N_1}}\right) + 1 = 0, \tag{1.14}$$

а при $m \to 0$ и $s \to \infty$ имеем $m(A_0/B_0)s + 1 = 0$.

Отсюда ясно, что $s = -B_0/(A_0 m) \to \infty$, но остается слева от мнимой оси, если $B_0/A_0 > 0$.

2. $N_2 - N_1 = \nu = 2$. После деления многочленов получим уравнение второго порядка:

$$\begin{aligned} &m\frac{A_0}{B_0}s^2 + \left(A_1 - \frac{A_0}{B_0}B_1\right)\frac{m}{B_0}s + C + \\ &+\frac{\beta_0 s^{N_2-3} + \beta_1 s^{N_2-4} + \ldots + \beta_{N_2-3}}{B_0 s^{N_1} + B_1 s^{N_1-1} + \ldots + B_{N_1}} + 1 = 0, \end{aligned} \tag{1.15}$$

при $s \to \infty$ и $m \to 0$

$$m\frac{A_0}{B_0}s^2 + \frac{m}{B_0}\left(A_1 - \frac{A_0}{B_0}B_1\right)s + 1 = 0.$$

Для устойчивости системы необходимо и достаточно иметь положительные коэффициенты уравнения, поэтому $A_1/A_0 - B_1/B_0 > 0$, причем $B_0/A_0 > 0$.

3. $N_2 - N_1 = \nu \geqslant 3$. Выполнив деление, после перехода к пределу получим уравнение ν-го порядка:

$$m\left\{\frac{A_0}{B_0}s^{\nu} + \left(A_1 - \frac{A_0}{B_0}B_1\right)\frac{1}{B_0}s^{\nu-1} + \left[\left(A_2 - \frac{A_0}{B_0}B_2\right) - \right.\right.$$
$$\left.\left. - \frac{B_1}{B_0}\left(A_1 - \frac{A_0}{B_0}B_1\right)\right]\frac{1}{B_0}s^{\nu-2} + \ldots\right\} + 1 = 0. \qquad (1.16)$$

Для устойчивости системы коэффициенты уравнения должны быть положительными. Если $\nu = 3$, то это условие записывается в виде

$$\frac{A_0}{B_0} > 0;\ \frac{A_1}{A_0} - \frac{B_1}{B_0} > 0;\ \left(A_2 - \frac{A_0}{B_0}B_2\right) - \frac{B_1}{B_0}\left(A_1 - \frac{A_0}{B_0}B_1\right) > 0. \qquad (1.17)$$

После деления получим

$$m\frac{A_0}{B_0}s^3 + \left(\frac{A_1}{A_0} - \frac{B_1}{B_0}\right)m\frac{A_0}{B_0}s^2 + m\frac{A_0}{B_0}\left[\left(\frac{A_2}{A_0} - \frac{B_2}{B_0}\right) - \right.$$
$$\left. - \frac{B_1}{B_0}\left(\frac{A_1}{A_0} - \frac{B_1}{B_0}\right)\right]s + 1 = 0. \qquad (1.18)$$

Применив критерий Вышнеградского [37], убеждаемся, что при $m \to 0$ неравенство

$$\frac{A_0}{B_0}m\left(\frac{A_1}{A_0} - \frac{B_1}{B_0}\right)\left[\left(\frac{A_2}{A_0} - \frac{B_2}{B_0}\right) - \frac{B_1}{B_0}\left(\frac{A_1}{A_0} - \frac{B_1}{B_0}\right)\right] > 1 \qquad (1.19)$$

не соблюдается. Это же справедливо для $\nu \geqslant 3$.

Действительно, к вырожденному уравнению (1.13) добавляется уравнение

$$m\frac{A_0}{B_0}s^{\nu} + \frac{mA_0}{B_0}\left(\frac{A_1}{A_0} - \frac{B_1}{B_0}\right)s^{\nu-1} + \ldots + \frac{mA_0}{B_0}\left(\frac{A_{N_2}}{A_0} - \frac{B_{N_1}}{B_0}\right)s + 1 = 0.$$

При соблюдении необходимого условия устойчивости (критерия Гурвица) должны выполняться неравенства

$$\begin{vmatrix} a_1 & a_3 \\ a_0 & a_2 \end{vmatrix} > 0, \ldots, \begin{vmatrix} a_1 & a_3 & a_5 & \ldots & 0 \\ a_0 & a_2 & a_4 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \\ 0 & 0 & 0 & \ldots & a_n \end{vmatrix} > 0, \qquad (1.20)$$

в которых a_i — коэффициенты характеристического уравнения. Так как m входит множителем, то получим $m[\cdot] > \text{const}$, однако $m \to 0$, а $[\cdot] > 0$, поэтому неравенство (1.20) невозможно. Таким образом, $N_2 - N_1 = \nu \leqslant 2$.

Вернемся к исходному характеристическому уравнению (1.13)

$$\left(A_0 s^{N_2} + A_1 s^{N_2-1} + \ldots + A_{N_2}\right) + K_i\left(B_0 s^{N_1} + B_1 s^{N_1-1} + \ldots + B_{N_1}\right) = 0.$$

Введем в i-й контур, коэффициент усиления которого $K_i \to \infty$, производную $(\nu_i - 2)$-го порядка, т. е. вместо K_i подставим $K_i(s^{\nu_i - 2} + 1)$. Тогда уравнение (1.13) примет следующий вид:

$$\left(A_0 s^{N_2} + A_1 s^{N_2 - 1} + \ldots + A_{N_2}\right) + K_i\left(s^{\nu_i - 2} + 1\right)\left(B_0 s^{N_1} + B_1 s^{N_1 - 1} + \ldots + B_{N_1}\right) = 0.$$

Степень многочлена $B_0 s^{N_1} + \ldots + B_{N_1}$ возросла на $\nu_i - 2$, поэтому необходимое и достаточное условие устойчивости многомерной системы $N_2 - N_1 \leqslant \nu_i$ ($\nu_i \leqslant 3$) теперь выполняется. Применив к нему условие $N_2 - N_1 - \nu_i + 2 \leqslant \nu_i$, получим $N_2 - N_1 \leqslant 2(\nu_i - 1)$, но $\nu_i \leqslant 2$, поэтому $N_2 - N_1 \leqslant 2$.

Схема алгоритма следующая. Среди общего контура выделяется астатический контур. Составляется характеристическое уравнение и производится разложение определителя (1.11) по строкам или столбцам, которые зависят от параметров передаточной функции выделенного астатического контура. Таким образом, уравнение (1.11) приводится к виду (1.12), после чего определяется наивысший порядок многочлена. Если $N_2 - N_1 \leqslant 2$, то устойчивость многомерной системы достигается подбором коэффициентов уравнений (1.14)—(1.19). Если $N_2 - N_1 \geqslant 3$, то, введя в контур, содержащий бесконечно большой коэффициент усиления, производную $(\nu - 2)$-го порядка, снова достигаем удовлетворения условия $N_2 - N_1 \leqslant 2$, приводя задачу к рассмотренному случаю.

ПРОГРАММА OPR

Назначение: анализ устойчивости многомерных систем при неограниченно возрастающем коэффициенте усиления. Разработана на языке Фортран. Задана операторная матрица системы $\mathbf{M}s^2 + \mathbf{N}s + \mathbf{L}$, где **M**, **N**, **L** — матрицы (числовые).

Параметры:

M1, N1 — размеры операторных матриц;

M(M1, N1), N(M1, N1), L(M1, N1) — операторные матрицы размера M1×N1;

I0, J0 — индексы неограниченно возрастающих коэффициентов;

IND — индексирует факт устойчивости системы: IND=1 — система устойчива, IND=0 — система неустойчива.

Параметры M1, N1, M, N, L, I0, J0 определяются по умолчанию.

В процессе работы осуществляется обращение к программе DET — вычисление определителя матриц.

Ниже приводится листинг программы OPR с исходными (тестовыми) данными, для которых система устойчива (IND=1) при неограниченном возрастании коэффициента с индексами I0=2, J0=3.

```
      PROGRAM OPR
      DIMENSION A0(3,3)
*     A1(3,3),A2(3,3),A3(3,3),B0(3,3),B1(3,3),B1(3,3),G(3
*     D0(3,3),D1(3,3)
      INTEGER I0,J0,K,P,S,T
      REAL C1,C2,V,M(3,3),N(3,3),L(3,3)
      M(1,1)=5
      M(2,1)=0
      M(3,1)=0
      M(1,2)=0
      M(2,2)=1
```

```
      M(3,2)=1
      M(1,3)=0
      M(2,3)=0
      M(3,3)=1
      N(1,1)=1
      N(2,1)=2
      N(3,1)=3
      N(1,2)=4
      N(2,2)=5
      N(3,2)=6
      N(1,3)=7
      N(2,3)=8
      N(3,3)=9
      L(1,1)=1
      L(2,1)=2
      L(3,1)=3
      L(1,2)=4
      L(2,2)=5
      L(3,2)=6
      L(1,3)=7
      L(2,3)=8
      L(3,3)=9
      I0=2
      J0=3
      PRINT 100
100   FORMAT(1X,`INITIAL DATA`//,1X,`MATRICES M,N,L`/)
      PRINT 101,((M(I,J),J=1,3),(N(I,J),J=1,3),(L(I,J),J=1
101   FORMAT(1X,3(4X,3(F6.2,1X))/1X,3(4X,3(F6.2,1X))/,
     * 1X,3(4X,3(F6.2,1X))/)
      PRINT 102,I0,J0
102   FORMAT(/1X,`I0=`,I1,2X,`J0=`,I1///)
      DO 15 I=1,3
      A1(I,1)=M(I,1)
      A1(I,1)=M(I,2)
15    A1(I,1)=N(I,3)
      DO 16 I=1,3
      A2(I,1)=M(I,1)
      A2(I,1)=N(I,2)
16    A2(I,1)=N(I,3)
      DO 17 I=1,3
      A3(I,1)=N(I,1)
      A3(I,1)=M(I,2)
17    A3(I,1)=M(I,3)
      DO 18 I=1,3
      DO 18 J=1,3
18    A0(I,J)=M(I,J)
      I=3
      CALL DET(I,A0,DETA0)
      CALL DET(I,A1,DETA1)
      CALL DET(I,A2,DETA2)
      CALL DET(I,A3,DETA3)
      DO 21 I=1,3
      DO 21 J=1,3
21    DO(I,J)=M(I,J)
      DO 22 I=1,3
22    DO(I,J0)=L(I,J0)
      K=1
      P=1
      DO 20 I=1,3
      DO 20 J=1,3
      IF(I.EQ.I0)GOTO 20
      IF(J.EQ.J0)GOTO 20
```

```
      B0(K,P)=D0(I,J)
      P=P+1
      IF(P.NE.3)GOTO 20
      K=2
      P=1
20    CONTINUE
      DO 23 I=1,3
      DO 23 J=1,3
23    G(I,J)=M(I,J)+N(I,J)
      DO 24 I=1,3
24    G(I,J0)=2*L(I,J0)
      T=1
      S=1
      DO 25 I=1,3
      DO 25 J=1,3
      IF(I.EQ.I0)GOTO 25
      IF(J.EQ.J0)GOTO 25
      B1(T,S)=G(I,J)
      S=S+1
      IF(S.NE.3)GOTO 25
25    CONTINUE
      I=2
      CALL DET(I,B0,DETB0)
      CALL DET(I,B1,DETB1)
      IF(DETB0.EQ.0)GOTO 52
      C1=DETA0/DETB0
      C2=DETA1/DETB0-(DETA0*DETB1)/(2*DETB0)
      V=L(I0,J0)
      IF(C1*C2.LT.0)GOTO 52
      IF(C1*V.LT.0)GOTO 52
      PRINT 54
54    FORMAT(//1X,`NECESSARY  AND SUFFICIENT STABILITY`/`
     *  CONDITION FULFILED, HENCE, THIS SYSTEB IS`/`
     *  STABILE FOR UNBOUNDED INCREMENT OF COEFF. L(I0,J0)`).
      GOTO 60
52    PRINT 56
56    FORMAT(//1X,`STABILITY CRITERIA NOT FULFILED,`/`
     *  HENCE THIS SYSTEM IS NOT STABLE FOR`/`
     *  UNBOUNDED INCREMENT OF COEFF. L(I0,J0)`)
60    CONTINUE
      STOP
      END
      SUBROUTINE DET(NN,AA,DE)
      DIMENSION AA(3,3)
      D=1.
      DO 1 K=1,NN
      RMAX=0.
      DO 2 I-K,NN
      T=AA(K,I)
      IF(ABS(T).LE.ABS(RMAX))GOTO 2
      RMAX=T
      J=I
2     CONTINUE
      IF(RMAX.NE.O.)GOTO 3
      D=0.
      GOTO 4
3     CONTINUE
      IF(J.EQ.K)GOTO 5
      D=-D
      DO 6 I=K,NN
      T=AA(I,J)
      AA(I,J)=AA(I,K)
```

```
6       AA(I,K)=T
5       CONTINUE
        IF(K.GE.NN)GOTO 7
        M=K+1
        DO 8 I=M,NN
        T=AA(K,I)/RMAX
        DO 9 J=M,NN
9       AA(J,I)=AA(J,I)-T*AA(J,K)
8       CONTINUE
7       D=D*AA(K,K)
1       CONTINUE
4       DE=D
        RETURN
        END
```

Глава 2.

Статистический анализ динамических систем управления

2.1. Модели динамических систем управления и случайных входных сигналов

Основным инструментом при исследовании сложных систем управления являются методы математического моделирования. Обычно при описании функционирования исследуемой системы в пространстве состояний (фазовом пространстве) используется система дифференциальных уравнений вида

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{F}[\mathbf{x}(t),\ \mathbf{u}(t),\ \xi(t), t], \\ \mathbf{y}(t) &= \mathbf{C}[\mathbf{x}(t),\ \mathbf{u}(t),\ \eta(t), t], \end{aligned} \tag{2.1}$$

где $\mathbf{x}(t)$ — вектор состояний системы размерности n_1; $\mathbf{u}(t)$ — вектор управляющего сигнала размерности n_2; $\mathbf{y}(t)$ — вектор наблюдаемых выходных сигналов размерности n_3; $\xi(t)$, $\eta(t)$ — вектор шумов системы и погрешностей измерений n_4 и n_5 соответственно; t — скалярный параметр времени, принимающий значения из некоторого (быть может, несобственного) интервала $T=[t_0, t_к]$.

Для *линейной системы* уравнения (2.1) существенно упрощаются:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{u}(t) + \mathbf{I}(t)\xi(t), \\ \mathbf{y}(t) &= \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t) + \mathbf{R}(t)\eta(t) \end{aligned} \tag{2.2}$$

Здесь $\mathbf{F}(t)$, $\mathbf{G}(t)$, $\mathbf{I}(t)$, $\mathbf{C}(t)$, $\mathbf{D}(t)$, $\mathbf{R}(t)$ — матрицы с коэффициентами, зависящими (возможно) от времени t. При описании функционирования физических систем основные трудности возникают при попытках корректного учета влияния случайных составляющих $\xi(t)$, $\eta(t)$ в уравнениях (2.2).

Анализ *детерминированной* линейной *системы* (2.2) (т. е. при нулевых матрицах $\mathbf{I}(t)$, $\mathbf{R}(t)$) достаточно прост. В частности, $\mathbf{x}(t)$ можно представить в явном виде

$$\mathbf{x}(t) = \Phi(t, t_0)\,\mathbf{x}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\,\mathbf{G}(\tau)\,\mathbf{u}(\tau)\,d\tau, \tag{2.3}$$

где $\Phi(t,\tau)$ — переходная матрица, удовлетворяющая следующим соотношениям:

$$\frac{d}{dt}\Phi(t,\tau)=\mathbf{F}(t)\Phi(t,\tau), \tag{2.4}$$

$$\frac{d}{dt}\Phi^{\mathrm{T}}(t,\tau)=-\mathbf{F}^{\mathrm{T}}(\tau)\Phi(t,\tau)\ \text{(сопряженное уравнение)}, \tag{2.5}$$

$$\Phi(t_0,\ t_0)=\mathbf{E}\,[\Phi(t,\tau)]^{-1}=\Phi(\tau,t),\ \Phi(t,\tau),\ \Phi(\tau,\theta)=\Phi(t,\theta).$$

Здесь $\mathbf{E}$ — единичная матрица.

Для *стационарной линейной системы* (системы (2.2), у которой матрицы $\mathbf{F}$, $\mathbf{G}$, $\mathbf{C}$ не зависят от времени t) со скалярным выходом (наблюдением) y

$$\begin{aligned}\dot{\mathbf{x}}&=\mathbf{F}\mathbf{x}+\mathbf{G}u,\\ y&=\mathbf{C}^{\mathrm{T}}\mathbf{x},\end{aligned} \tag{2.6}$$

(где $\mathbf{C}$ — некоторый вектор) существует линейное преобразование фазовых координат $\mathbf{z}=\mathbf{A}\mathbf{x}$, переводящее систему (2.6) в систему

$$\begin{aligned}\dot{\mathbf{z}}&=\tilde{\mathbf{F}}\mathbf{z}+\tilde{\mathbf{G}}u,\\ y&=\tilde{\mathbf{C}}^{\mathrm{T}}\mathbf{z},\end{aligned} \tag{2.7}$$

где

$$\tilde{\mathbf{F}}=\begin{bmatrix}0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & \dots & 0\\ \dots & \dots & \dots & \dots & \dots\\ 0 & 0 & 0 & \dots & 1\\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1\end{bmatrix};\quad \tilde{\mathbf{G}}=\mathbf{A}\mathbf{G};$$

$$\tilde{\mathbf{C}}=\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix};\quad \mathbf{A}=\begin{bmatrix}\mathbf{C}^{\mathrm{T}}\\ \mathbf{C}^{\mathrm{T}}\mathbf{F}\\ \vdots\\ \mathbf{C}^{\mathrm{T}}\mathbf{F}^{n-1}\end{bmatrix}. \tag{2.8}$$

В соответствии с формулами (2.8) система (2.6) может быть заменена скалярным дифференциальным уравнением

$$\frac{d^n}{dt^n}y+a_1\frac{d^{n-1}}{dt^{n-1}}y+\dots+a_n y=b_1\frac{d^{n-1}}{dt^{n-1}}u+\dots+b_n u, \tag{2.9}$$

$$b_i=\sum_{k=0}^{i-1}a_{i-k}\tilde{G}_k+\tilde{G}_i.$$

Представление системы в виде (2.8), (2.9) часто более удобно для анализа на ЭВМ.

Для *нестационарной линейной системы* (системы (2.2) с зависящими от времени t матрицами $\mathbf{F}$, $\mathbf{G}$, $\mathbf{C}$) также существует представление в виде дифференциального уравнения (2.9) со следующими коэффициентами:

$$b_i(t)=\sum_{k=0}^{i-1}\sum_{s=0}^{i-k}C_{n+s-1}^{n-1}\,a_{i-k-s}\frac{d}{dt^s}\tilde{G}_k(t)+\tilde{G}_i(t). \tag{2.10}$$

Уравнению (2.9) соответствует передаточная функция системы

$$H(s) = \int_0^\infty \exp(-st)\, h(t)\, dt, \tag{2.11}$$

где $h(t)$ — весовая функция. Кроме того,

$$y(t) = \int_0^\infty h(\tau)\, u(t-\tau)\, d\tau. \tag{2.12}$$

Передаточная функция (2.11) является рациональной функцией и вычисляется по следующей формуле:

$$H(S) = \frac{b_1 s^n + \ldots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n}. \tag{2.13}$$

Современные методы стохастического моделирования основываются на стохастических дифференциальных уравнениях

$$d\mathbf{x} = \mathbf{f}(\mathbf{x},t)dt + \sigma(\mathbf{x},t)d\mathbf{w}, \tag{2.14}$$

где $[\mathbf{w}(t),\ t \in T]$ — n-мерный винеровский процесс с ковариацией приращений $\mathbf{E}dt$.

Скалярный винеровский процесс можно определить как случайный процесс $\{\mathbf{w}(t),\ t \geqslant t_0\}$ со следующими свойствами:

1) $\mathbf{w}(t_0) = 0$;

2) $\mathbf{w}(t)$ для всех $t \geqslant t_0$ имеет нормальное распределение с нулевым математическим ожиданием;

3) процесс $\mathbf{w}(t)$ имеет независимые стационарные приращения, т. е. для любых $t_i > t_0$ $(i=1, \ldots, k)$, таких, что $t_1 < t_2 < \ldots < t_k$, случайные величины $\mathbf{w}(t_1)$, $\mathbf{w}(t_2) - \mathbf{w}(t_1), \ldots, \mathbf{w}(t_k) - \mathbf{w}(t_{k-1})$ взаимно независимы, а распределение разностей $\mathbf{w}(t) - \mathbf{w}(s)$ зависит только от $t-s$.

Аналогично определяется винеровский процесс в многомерной системе. Из определения винеровского процесса следует, что он имеет линейно растущую дисперсию $D = ct$ и ковариационную функцию $\mathrm{K}(s,t) = c\min(s,t)$, где c — некоторый параметр. Из выражения (2.14) следует, что $\mathbf{x} = \mathbf{x}(t)$ — решение стохастического интегрального уравнения

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} f(\mathbf{x}(\tau), \tau)\, d\tau + \sum_{t_0}^{t} \sigma(\mathbf{x}(\tau), \tau)\, d\mathbf{w}(\tau). \tag{2.15}$$

При разработке математического аппарата решения уравнений (2.14); (2.15) основные трудности возникают при попытке корректного определения члена $\int_{t_0}^{t} \sigma(\mathbf{x}(\tau),\tau)d\mathbf{w}(\tau)$ в (2.15). Для этих целей обычно используется интеграл Ито [13, 14]

$$\int_{t_0}^{t} \sigma\{\mathbf{x}(\tau), \tau\}\, d\mathbf{w}(\tau) = \lim \Sigma \sigma\{\mathbf{x}(t_i), t_i\}\, [\mathbf{w}(t_{i+1}) - \mathbf{w}(t_i)],$$

где t_i $(i=1,\ldots, \mathrm{N})$ — разбиение отрезка $[t_0, t_\kappa]$, а предел берется в среднеквадратическом при $\Delta t_i = t_{i+1} - t_i \to 0$. В этом случае математическое ожидание $x(t)$

$$\mathbf{M}\mathbf{x}(t) = \mathbf{M}\mathbf{x}(t_0) + M \int_{t_0}^{t} f(\mathbf{x}(s), s)\, ds,$$

а условное математическое ожидание и ковариация

$$\mathbf{M}[\mathbf{x}(t+h)-\mathbf{x}(t)\,|\,\mathbf{x}(t)]=\mathbf{f}(\mathbf{x},t)h+o(h),$$

$$\mathrm{Cov}[\mathbf{x}(t+h)-\mathbf{x}(t)\,|\,\mathbf{x}(t)]=\sigma(\mathbf{x},t)\sigma^{\mathrm{T}}(\mathbf{x},t)h+o(h)$$

(здесь $o(h)$ — символ бесконечно малой величины высшего порядка по сравнению с h: $o(h)/h\to 0$ при $h\to 0$).

Пусть функция $y(x,t)$ непрерывно дифференцируема по t и дважды непрерывно дифференцируема по x, где x удовлетворяет уравнению (2.14). Тогда $y(x,t)$ удовлетворяет следующему дифференциальному уравнению:

$$d\mathrm{y}=\frac{\partial \mathrm{y}}{\partial t}\,dt+\sum_{i=1}^{n}\frac{\partial \mathrm{y}}{\partial x_i}+\frac{1}{2}\sum_{i,j,k=1}^{n}\frac{\partial^2 \mathrm{y}}{\partial x_i\,\partial x_j}\sigma_{ik}\,\sigma_{jk}\,dt=$$

$$=\left(\frac{\partial \mathrm{y}}{\partial t}+\sum_{i=1}^{n}\frac{\partial \mathrm{y}}{\partial x_i}f_i+\frac{1}{2}\sum_{i,j,k=1}^{n}\frac{\partial^2 \mathrm{y}}{\partial x_i\,\partial x_j}\sigma_{ik}\,\sigma_{jk}\right)dt+$$

$$+\sum_{i=1}^{n}\frac{\partial \mathrm{y}}{\partial x_i}(\sigma d\mathrm{w})_i, \qquad (2.16)$$

где σ_{ij} — элементы матрицы $\sigma(\mathbf{x},t)$; $(\sigma d\mathbf{w})_i$ — i-й элемент вектора $(\sigma d\mathbf{w})$. Соотношение (2.16) позволяет производить операции с функциями от решения стохастического дифференциального уравнения.

Рассмотрим частный случай уравнения (2.14) — линейное векторное стохастическое дифференциальное уравнение

$$d\mathbf{x}=\mathbf{A}(t)\mathbf{x}dt+d\mathbf{w}. \qquad (2.17)$$

Здесь $\mathbf{x}=\mathbf{x}(t)$ — n-мерный вектор; $\mathbf{A}=\mathbf{A}(t)$ — квадратная матрица размера $n\times n$; $\{\mathbf{w}(t),\ t\in T\}$ — n-мерный винеровский процесс с ковариацией приращений $\mathrm{R}_1 dt$.

Предположим, что начальное значение $\mathbf{x}(t_0)$ — гауссовская случайная величина с математическим ожиданием m_0 и ковариацией R_0. Уравнение (2.17) эквивалентно интегральному стохастическому уравнению

$$\mathbf{x}(t)=\Phi(t,t_0)\,\mathbf{x}(t_0)+\int_{t_0}^{t}\Phi(t,s)\,d\mathbf{w}(s), \qquad (2.18)$$

где квадратная матрица Φ удовлетворяет дифференциальному уравнению

$$\frac{d}{dt}\Phi(t,t_0)=\mathbf{A}(t)\Phi(t,t_0)$$

с начальным условием $\Phi(t_0,t_0)=\mathrm{E}$.

Решением стохастического дифференциального уравнения (2.17) является случайный процесс с математическим ожиданием $M\mathbf{x}(t)=\mathrm{m}_x(t)$ и ковариационной функцией $\mathrm{K}(s,t)$, где

$$d\mathrm{m}_x/dt=\mathbf{A}(t)\,\mathrm{m}_x,\quad \mathrm{m}_x(t_0)=\mathrm{m}_0, \qquad (2.19)$$

$$\mathrm{K}(s,t)=\begin{cases}\Phi(s,t)\,\mathbf{P}(t) \text{ при } s\geqslant t,\\ \mathbf{P}(s)\,\Phi(t,s) \text{ при } s<t.\end{cases} \qquad (2.20)$$

Матрица P удовлетворяет следующему дифференциальному уравнению:

$$d\mathbf{P}/dt=\mathbf{AP}+\mathbf{PA}^{\mathrm{T}}+\mathbf{R}_1,\ \mathbf{P}(t_0)=\mathbf{R}_0. \qquad (2.21)$$

Множество значений параметра t представляет собой последовательность целых чисел $T=\{\ldots,-1,0,1,\ldots\}$. *Детерминированная* система с дискретным вре-

менем описывается разностным уравнением

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), t), \\ \mathbf{y}(t) &= \mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t), \end{aligned} \tag{2.22}$$

где $\mathbf{x}(t)$ — n-мерный вектор состояния; $\mathbf{y}(t)$ — l-мерный вектор наблюдений; $\mathbf{u}(t)$ — вектор управления. Анализ детерминированных дискретных систем достаточно прост. В частности, для линейных систем существует представление, аналогичное (2.9), (2.10), что позволяет вести анализ с помощью передаточных функций. При необходимости учета аддитивных случайных факторов модель (2.22) преобразуется в следующую:

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{F}[\mathbf{x}(t), \mathbf{u}(t), t] + \mathbf{U}[\mathbf{x}(t), t], \\ \mathbf{y}(t) &= \mathbf{C}[\mathbf{x}(t), \mathbf{u}(t), t] + \mathbf{V}[\mathbf{x}(t), t], \end{aligned} \tag{2.23}$$

где $\mathbf{U}[\mathbf{x}(t), t]$, $\mathbf{V}[\mathbf{x}(t), t]$ — некоторые случайные величины.

Если предположить далее, что функция $\mathbf{F}$ линейно зависит от $\mathbf{x}(t)$, а управление осуществляется соответствующим выбором элементов $\mathbf{F}$, то получим следующее линейное стохастическое уравнение вектора состояния системы:

$$\mathbf{x}(t+1) = \Phi(t+1, t)\mathbf{x}(t) + \mathbf{e}(t), \tag{2.24}$$

где Φ — некоторая матрица размера $n \times n$.

Допустим, что векторы $\mathbf{e}(t)$ и $\mathbf{e}(s)$ независимы, если $t \neq s$, и нормальны с моментами первого и второго порядка

$$M\mathbf{e}(t) = 0, \quad M\mathbf{e}(t)\mathbf{e}^{\mathrm{T}}(t) = \mathbf{r}(t). \tag{2.25}$$

Пусть начальное состояние $\mathbf{x}(t_0)$ также нормально с математическим ожиданием m_0 и ковариационной матрицей R_0. Тогда решением нормального линейного стохастического разностного уравнения (2.24) является гауссовский процесс с математическим ожиданием, удовлетворяющим соотношению

$$\mathbf{m}(t+1) = \Phi(t+1, t)\mathbf{m}(t) \tag{2.26}$$

с начальным условием $\mathrm{m}(t_0) = m_0$ и ковариационной функцией

$$K(s, t) = \Phi(s, t)\mathbf{P}(t), \quad s \leqslant t, \tag{2.27}$$

где матрица $\mathbf{P}(t)$ находится из разностного уравнения

$$\mathbf{P}(t+1) = \Phi(t+1, t)\,\mathbf{P}(t)\,\Phi^{\mathrm{T}}(t+1, t) + \mathbf{R}_1 \tag{2.28}$$

с начальным условием $\mathbf{P}(t_0) = \mathbf{R}_0$. Если матрицы Φ и $\mathbf{R}_1$ постоянны, то из условий (2.25) — (2.28) следует, что

$$\mathbf{P}(t) = \Phi^{\mathrm{T}} R_0 (\Phi^{\mathrm{T}})^t + \sum_{s=0}^{t-1} \Phi^s R_1 (\Phi^{\mathrm{T}})^s \tag{2.29}$$

(здесь $(\Phi)^t$ — t-я степень матрицы Φ).

В заключение остановимся на переходе от стохастических дифференциальных уравнений к разностным. Данная задача весьма важна при использовании ЭВМ для решения задач моделирования стохастических систем. Это обусловлено тем, что ЭВМ является дискретной системой. При подготовке задачи моделирования непрерывной стохастической системы на ЭВМ необходимо осуществить переход от стохастических дифференциальных уравнений к разностным.

Предположим, что процессы функционирования системы $\mathbf{x}(t)$ и получения наблюдений $\mathbf{y}(t)$ описываются векторными стохастическими дифференциальными уравнениями

$$\begin{aligned} d\mathbf{x} &= \mathbf{A}\mathbf{x}dt + d\mathbf{w}_1, \\ d\mathbf{y} &= \mathbf{C}\mathbf{x}dt + d\mathbf{w}_2, \end{aligned} \tag{2.30}$$

где $\mathbf{x}$ — n-мерный вектор состояния системы; $\mathbf{y}$ — r-мерный вектор наблюдения; $\mathbf{A}$ — матрица размера $n \times n$; $\mathbf{C}$ — матрица размера $r \times n$ ($\mathbf{A}$ и $\mathbf{C}$ зависят в общем случае от t) $\mathbf{w}_1(t)$ и $\mathbf{w}_2(t)$ — n- и r-мерные винеровские процессы с ковариациями приращений $\mathbf{R}_1 dt$ и $\mathbf{R}_2 dt$. Предположим также, что выходные переменные наблюдаются в дискретные моменты времени $t_0, \ldots, t_к$. Значения переменных состояния и наблюдаемых выходных величин стохастических дифференциальных уравнений (2.30) в дискретные моменты времени t_i $(i \geqslant 0)$ связаны стохастическими разностными уравнениями

$$\begin{aligned} \mathbf{x}(t_{i+1}) &= \Phi\mathbf{x}(t_i) + \tilde{\mathbf{w}}_1(t_i), \\ \mathbf{z}(t_{i+1}) &= \mathbf{y}(t_{i+1}) - \mathbf{y}(t_i) = \mathbf{S}\mathbf{x}(t_i) + \tilde{\mathbf{w}}_2(t_i), \end{aligned} \tag{2.31}$$

где матрица $\Phi = \Phi(t_{i+1}, t_i)$ $(i \geqslant 0)$ определяется соотношениями

$$\frac{d}{dt}\Phi(t, t_i) = \mathbf{A}(t)\Phi(t, t_i), \quad t_i \leqslant t \leqslant t_{i+1} \quad \Phi(t_i, t_i) = \mathbf{E};$$

матрица $\mathbf{S} = \mathbf{S}(t_{i+1}, t_i)$ $(i \geqslant 0)$ определяется формулой

$$\mathbf{S}(t_{i+1}, t_i) = \int_{t_i}^{t_{i+1}} \mathbf{C}(s)\,\Phi(s, t_i)\,ds, \tag{2.32}$$

$\{\tilde{\mathbf{w}}_1(t_i),\ i \geqslant 0\}$, $\{\tilde{w}_2(t_i),\ i \geqslant 0\}$ представляют собой последовательности независимых гауссовских случайных величин с нулевым математическим ожиданием и соответствующими ковариациями $\tilde{\mathbf{R}}_1(t_i)$, $\tilde{\mathbf{R}}_2(t_i)$ $(i \geqslant 0)$:

$$\begin{aligned} \tilde{\mathbf{R}}_1(t_i) &= \int_{t_i}^{t_{i+1}} \Phi(t_{i+1}, s)\,\mathbf{R}_1\,\Phi^{\mathrm{T}}(t_{i+1}, s)\,ds, \\ \tilde{\mathbf{R}}_2(t_i) &= \int_{t_i}^{t_{i+1}} \left[\mathbf{S}(t_{i+1}, s)\,\mathbf{R}_1(s)\,\mathbf{S}^{\mathrm{T}}(t_{i+1}, s) + \mathbf{R}_2(s)\right] ds. \end{aligned} \tag{2.33}$$

Взаимная ковариационная функция процессов $\tilde{\mathbf{w}}_1$ и $\tilde{\mathbf{w}}_2$

$$\tilde{\mathbf{K}}_{12}(t_i) = M\tilde{\mathbf{w}}_1(t_i)\,\tilde{\mathbf{w}}_2^{\mathrm{T}}(t_i) = \int_{t_i}^{t_{i+1}} \Phi(t_i, s)\,\mathbf{R}_1(s)\,\mathbf{S}^{\mathrm{T}}(t_{i+1}, s)\,ds, i \geqslant 0.$$

Стохастические разностные уравнения (2.32), (2.33) и стохастические дифференциальные уравнения (2.30) с точки зрения статистических свойств в интервалах выборки совершенно идентичны, что позволяет использовать разностный вариант (2.32), (2.33) для аппроксимации процесса функционирования непрерывной системы (2.30).

2.2. Статистический анализ динамических систем с дискретным временем

Предположим, что динамическая система с дискретным временем описывается линейными уравнениями. Большая часть задач оптимизации параметров динамических систем управления основана на оценке критерия эффективности системы. В инженерной практике наиболее часто эффективность функционирования системы характеризуется некоторым функционалом $F(\mathrm{m}_{\text{вых}}, \mathrm{D}_{\text{вых}})$ от средних значений функций потерь, которые представляют собой линейную и квадратическую функции относительно переменных состояния системы — математического ожидания $\mathrm{m}_{\text{вых}}$ и дисперсии $\mathrm{D}_{\text{вых}}$ выходного сигнала системы. Таким образом, задача заключается в описании стохастических свойств выходного сигнала.

Примем (это не ограничивает общности [11, 14]), что рассматриваемая система стационарная и имеет только один вход и один выход. Предположим также, что входной сигнал $y_{\text{вх}}$ — случайный процесс второго порядка с заданной функцией математического ожидания $\mathrm{m}_{y_{\text{вх}}}$ и ковариационной функцией $\mathrm{K}_{y_{\text{вх}}}(s,t)$. Соотношение между входным $y_{\text{вх}}$ и выходным $y_{\text{вых}}$ сигналами:

$$y_{\text{вых}}(t) = \sum_{i=-\infty}^{t} h(t-i)\,y_{\text{вх}}(i) = \sum_{i=0}^{\infty} h(i)\,y_{\text{вх}}(t-i). \qquad (2.34)$$

Здесь $h(s)$ — весовая функция системы ($h(s)=0$ при $s<0$). Справедлива следующая теорема.

Теорема 2.1. Пусть входной сигнал $y_{\text{вх}}$ асимптотически устойчивой динамической системы с дискретным временем — случайный процесс второго порядка с математическим ожиданием $\mathrm{m}_{y_{\text{вх}}}(t)$ и ковариационной функцией $\mathrm{K}_{y_{\text{вх}}}(s,t)$. Тогда выходной сигнал $y_{\text{вых}}(t)$ — случайный процесс второго порядка с математическим ожиданием и ковариационной функцией:

$$\mathrm{m}_{y_{\text{вых}}}(t) = \sum_{i=0}^{\infty} h(i)\,\mathrm{m}_{y_{\text{вх}}}(t-i); \qquad (2.35)$$

$$\mathrm{K}_{y_{\text{вых}}}(s,t) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} h(i)\,h(j)\,\mathrm{K}_{y_{\text{вх}}}(s-i, t-j). \qquad (2.36)$$

Взаимная ковариационная функция входного и выходного сигнала:

$$K_{y_{вх}y_{вых}}(s,t) = \sum_{i=0}^{\infty} h(i)\, K_{y_{вх}}(s,t-i). \tag{2.37}$$

Результаты данной теоремы можно в значительной степени улучшить, если предположить, что входной процесс $y_{вх}$ стационарный в широком смысле, т. е. что

$$m_{y_{вх}}(t) = \text{const} = m_{y_{вх}}; \quad K_{y_{вх}}(s,t) = K_{y_{вх}}(s-t). \tag{2.38}$$

В случае стационарного в широком смысле входного сигнала динамической системы статистический анализ выходного сигнала существенно упрощается. Выводы могут быть сформулированы в виде следующей теоремы.

Теорема 2.2. Пусть входной сигнал дискретной динамической системы, стационарный в широком смысле случайный процесс.

Если система асимптотически устойчива, то выходной сигнал $y_{вых}(t)$, определяемый соотношением (2.34), есть стационарный случайный процесс с математическим ожиданием

$$m_{y_{вых}} = H(1)\, m_{y_{вх}} \tag{2.39}$$

и спектральной плотностью

$$S_{y_{вых}} = |H[\exp(j\omega)]|^2 S_{y_{вх}}. \tag{2.40}$$

Взаимная спектральная плотность входного и выходного сигналов:

$$S_{y_{вх}y_{вых}}(\omega) = H[\exp(-j\omega)]\, S_{y_{вх}}(\omega). \tag{2.41}$$

Уравнения (2.39) — (2.41) можно использовать для определения передаточной функции динамической системы. Более того, для любой рациональной функции спектральной плотности $S_{y_{вых}}(\omega)$ существует асимптотически устойчивая линейная динамическая система, такая, что при воздействии на ее вход белого шума выходной сигнал представляет собой стационарный процесс со спектральной плотностью $S_{y_{вых}}(\omega)$. Из этого вытекает, что если ограничиться рассмотрением стационарных процессов с рациональными спектральными плотностями, то все они могут быть получены с помощью пропускания белого шума через устойчивую линейную динамическую систему с подходящим образом подобранной передаточной функцией.

Пусть динамическая линейная система имеет передаточную функцию

$$H(z) = B(z)/A(z), \tag{2.42}$$

где

$$\begin{gathered} A(z) = a_0 z^n + a_1 z^{n-1} + \ldots + a_n, \\ B(z) = b_0 z^n + b_1 z^{n-1} + \ldots + b_n, \\ z = \exp(j\omega),\ n \geqslant 1,\ a_0 > 0. \end{gathered} \tag{2.43}$$

Пусть на вход системы действует белый шум с дисперсией $D_{вх}$. Дисперсия выходного сигнала системы

$$D_{вых} = \frac{D_{вх}}{j} \int_{-\pi}^{\pi} H[\exp(j\omega)]\, H[\exp(-j\omega)]\, \exp(-j\omega)\, d[\exp(j\omega)] =$$

$$= \frac{D_{вх}}{j} \int_{|z|=1} H(z)\, H(z^{-1})\, \frac{dz}{z}. \qquad (2.44)$$

Очевидно, что для вычисления интеграла (2.44) достаточно уметь вычислять выражение

$$I = \frac{1}{j2\pi} \int_{|z|=1} \frac{B(z)\, B(z^{-1})}{A(z)\, A(z^{-1})} \frac{dz}{z}. \qquad (2.45)$$

Основной целью последующих рассмотрений будет построение эффективных формул для оценки интеграла (2.45).

Введем следующие обозначения. Пусть

$$A^*(z) = z^n A(z^{-1}),$$

$$A_k(z) = a_0^k z^k + a_1^k z^{k-1} + \ldots + a_k^k,$$

$$B_k(z) = b_0^k z^k + b_1^k z^{k-1} + \ldots + b_k^k,$$

где $A_k(z)$, $B_k(z)$ определяется с помощью следующих рекуррентных соотношений:

$$A_{k-1}(z) = z^{-1}(A_k(z) - \alpha_k A_k(z)),$$

$$B_{k-1}(z) = z^{-1}(B_k(z) - \beta_k A_k^*(z)), \quad k = 1, \ldots, n, \qquad (2.46)$$

$$\alpha_k = a_k^k/a_0^k; \quad \beta_k = b_k^k/a_0^k; \qquad (2.47)$$

$$A_n(z) = A(z); \; B_n(z) = B(z).$$

Из (2.46) видно, что коэффициенты многочленов $A_k(z)$ и $B_k(z)$ задаются следующими соотношениями:

$$a_i^{k-1} = a_i^k - \alpha_k a_{k-i}^k,$$

$$b_i^{k-1} = b_i^k - \beta_k a_{k-i}^k, \qquad (2.48)$$

где $a_i^n = a_i$; $b_i^n = b_i$ $(i = 0,1, \ldots, k-1)$ — начальные условия, а все a_0^k $(k = 0,1,\ldots,n)$ отличны от нуля.

Интеграл (2.45) можно вычислить рекурсивно, используя последовательность интегралов [14]:

$$I_k = \frac{1}{j2\pi} \int_{|z|=1} \frac{B_k(z)\, B_k(z^{-1})}{A_k(z)\, A_k(z^{-1})} \frac{dz}{z}, \quad k = 0,\ldots, n. \qquad (2.49)$$

Легко видеть, что $I_n = I$. Кроме того, справедлива следующая теорема.

Теорема 2.3. Пусть все корни полинома $A(z)$ лежат внутри единичного круга $|z| = 1$. Тогда последовательность интегралов $I_k(k = 0,\ldots,n)$ удовлетворяет следующему соотношению:

$$I_k = \beta_k^2 + I_{k-1}\left(1 - \alpha_k^2\right), \quad k = 1, ..., n, \quad I_0 = \beta_0^2. \tag{2.50}$$

В частности, из теоремы 2.2 следует формула для вычисления интегралов

$$I_k = \frac{1}{a_0^k} \sum_{i=0}^{k} \frac{\left(b_i^i\right)^2}{a_0^i}, \quad k = 1, ..., n. \tag{2.51}$$

Соотношения (2.48) позволяют построить эффективный алгоритм вычисления интеграла (2.45) (табл. 2.1).

Таблица 2.1

	Таблица А		Таблица Б	
*	a_0	$a_1 ... a_{n-1} a_n$	b_0	$b_1 ... b_{n-1} b_n$
	a_n	$a_{n-1} ... a_1 a_0$	a_n	$a_{n-1} ... a_1 a_0$
*	a_0^{n-1}	$a_1^{n-1} ... a_{n-1}^{n-1}$	b_0^{n-1}	$b_1^{n-1} ... b_{n-1}^{n-1}$
...	...	...	...	...
*	a_0^1	a_1^1	b_0^1	b_1^1
	a_1^1	a_0^1	a_1^1	a_0^1
*	a_0^0		b_0^0	

В каждую четную строку таблицы А записывают коэффициенты предыдущей строки в обратном порядке. Четные строки таблиц А и Б совпадают, элементы нечетных строк обеих таблиц получают с помощью соотношений (2.48). Символом * выделены строки, содержащие на первом месте коэффициенты a_0^k $(k=0,...,n)$. Получив из таблиц значения коэффициентов α_k, β_k $(k=0,...,n)$, нетрудно вычислить по формуле (2.51) значение интеграла (2.45).

ПРОГРАММА INTED

Назначение: вычисление математического ожидания и дисперсии выходного сигнала дискретной линейной системы с рациональной передаточной функцией (2.43) $H(z)=B(z)/A(z)$, где $A(z)$, $B(z)$ — полиномы.

Входные параметры:

A — вектор коэффициентов полинома $A(z)=A(1)z^N+A(2)z^{N-1}+...+A(N+1)$;

B — вектор коэффициентов полинома $B(z)=B(1)z^N+B(2)Z^{N-1}+...+B(N+1)$.

N — порядок полиномов A и B (размерности A и B должны быть на единицу больше N);

RM — математическое ожидание входного сигнала системы;

SV — дисперсия входного сигнала системы.

Выходные параметры:

IERR — индикатор итога анализа: если IERR=1, то все нули полинома A лежат внутри единичного круга, т. е. система устойчива, если же IERR=0, то либо

у полинома A существует корень, находящийся вне единичного круга, либо коэффициент A(1) неположителен, т. е. система неустойчива;

VM — математическое ожидание выходного сигнала системы;

V — дисперсия выходного сигнала системы.

Параметры A и B описываются с атрибутами BINARY FLOAT. Атрибуты параметров N, RM, SV, IERR, VM, V определяются по умолчанию.

Обращение: CALL INTED (A, B, N, IERR, SV, RM, V, VM);

Основные этапы работы:

1) вычисление коэффициентов таблиц A и B;

2) проверка устойчивости системы;

3) вычисление математического ожидания выходного сигнала системы.

Пример. Анализ дискретной линейной системы с передаточной функцией (2.43), где

$$A(z)=z^3+0{,}7z^2+0{,}5z-0{,}3;$$
$$B(z)=z^3+0{,}3z^2+0{,}2z+0{,}1.$$

Исходные данные: N=3, RM=0, SV=1.

Результат: IERR=1 (т. е. система устойчива), VM=0, V=18,5228, получен за 0,4 с на ЭВМ ЕС-1050.

```
INTED: PROCEDURE(A,B,N,IERR,SV,RM,V,VM);
       DECLARE
       (A(*),B(*))
       BINARY FLOAT;
       BEGIN;
       DCL AS(N+1)BINARY FLOAT;
       A0=A(1); IERR=1; V=0;
       DO K=1 TO N;
       L=N+1-K; L1=L+1;
       ALFA=A(L1)/A(1);
       BETA=B(L1)/A(1);
       V=V+BETA*B(L1);
       DO I=1 TO L;
       M=L+2-I; AS(I)=A(I)-ALFA*A(M);
       B(I)=B(I)-BETA*A(M);
       END;
       IF AS(1) <= 0 THEN GO TO MET5;
       DO I=1 TO L;
       A(I)=AS(I);
       END;
       END;
       V=V+B(1)**2/A(1);
       V=V/A0;
       B1=0; A1=0;
       DO I=1 TO N+1;
       B1=B1+B(I);
       A1=A1+A(I);
       END;
       VM=B1*RM/A1;
       V=V*SV;
       V=V*2*3.141592654;
       RETURN;
MET5:  IERR=0;
       RETURN;
       END;
       END INTED;
```

```
TEST: PROCEDURE OPTIONS(MAIN);
      DCL A(4),B(4);
      A(1)=1;   A(2)=0.7;
      A(3)=0.5; A(4)=-0.3;
      B(1)=1;   B(2)=0.3;
      B(3)=0.2; B(4)=0.1;
      N=3; RM=0; SV=1;
      CALL INTED(A,B,N,IERR,SV,RM,V,VM);
      END TEST;
```

2.3. Статистический анализ динамических систем с непрерывным временем

Анализ динамических систем с непрерывным временем $t \in T$ (T — некоторый интервал) и при случайных входных воздействиях аналогичен анализу систем с дискретным временем (§ 2.2).

Рассмотрим стационарную динамическую систему с входным $y_{вх}(t)$ и выходным $y_{вых}(t)$ сигналами и весовой функцией $h(t)(t \in T)$. Соотношение между входным и выходным сигналами

$$y_{вых}(t) = \int_{-\infty}^{t} h(t-s)\, y_{вх}(s)\, ds = \int_{0}^{\infty} h(s)\, y_{вх}(t-s)\, ds. \tag{2.52}$$

Справедлива следующая теорема.

Теорема 2.4. Предположим, что линейная динамическая система с непрерывным временем и весовой функцией $h(t)$ асимптотически устойчива, а входной сигнал $y_{вх}(t)$ — случайный процесс второго порядка с математическим ожиданием $m_{y_{вх}}(t)$ и непрерывной ковариационной функцией $K_{y_{вх}}(s,t)$. Тогда интеграл (2.52) существует в смысле Римана, а выходной сигнал $y_{вых}(t)$ есть случайный процесс с математическим ожиданием

$$m_{y_{вых}}(t) = \int_{0}^{\infty} h(s)\, m_{y_{вх}}(t-s)\, ds, \tag{2.53}$$

ковариационной функцией

$$K_{y_{вых}}(s,t) = \int_{0}^{\infty}\int_{0}^{\infty} h(s')\, h(s'')\, K_{y_{вх}}(s-s', t-s'')\, ds'\, ds''. \tag{2.54}$$

Взаимная ковариационная функция входного и выходного сигналов:

$$K_{y_{вх} y_{вых}}(s,t) = \int_{0}^{\infty} h(s')\, K_{y_{вх}}(s, t-s')\, ds'. \tag{2.55}$$

Если ограничиться рассмотрением стационарных в широком смысле входных процессов, то

$$m_{y_{вх}}(t) = m_{y_{вх}} = \text{const}; \quad K_{y_{вх}}(t,s) = K_{y_{вх}}(t-s)$$

и выводы теоремы 2.4 можно существенно усилить. А именно справедлива следующая теорема.

Теорема 2.5. Пусть асимптотически устойчивая линейная динамическая система имеет передаточную функцию

$$G(s) = \int_0^\infty \exp(-st)\, h(t)\, dt$$

и пусть входной сигнал $y_{вх}(t)$ — стационарный в широком смысле случайный процесс с математическим ожиданием $m_{y_{вх}}$ и спектральной плотностью $S_{y_{вх}}(\omega)$. Если выполняется условие

$$K_{y_{вх}}(0) = D_{y_{вх}} = \int_0^\infty S_{y_{вх}}(\omega)\, d\omega < \infty,$$

то выходной сигнал $y_{вых}(t)$ также стационарный в широком смысле процесс с математическим ожиданием

$$m_{y_{вых}} = G(0)\, m_{y_{вх}}, \tag{2.56}$$

спектральной плотностью

$$S_{y_{вых}}(\omega) = G(j\omega)\, G(-j\omega)\, S_{y_{вых}}(\omega) \tag{2.57}$$

и взаимной спектральной плотностью входного и выходного сигналов

$$S_{y_{вх}y_{вых}}(\omega) = G(-j\omega)\, S_{y_{вых}}(\omega). \tag{2.58}$$

Если спектральная плотность $S(\omega)$ — рациональная фукиция, то существует такая асимптотически устойчивая стационарная динамическая система с весовой функцией $h(s)$, что случайный процесс

$$y_{вых}(t) = \int_{-\infty}^{t} h(t-s)\, d\xi(s) \tag{2.59}$$

стационарный со спектральной плотностью $S(\omega)$. Здесь $\xi(t)$ $(t \in T)$ — процесс с ортогональными приращениями.

Оценим дисперсию выходного сигнала $D_{вых}$ для систем с непрерывным временем $t \in T$. Вычисление дисперсии (аналогично § 2.2) сводится к оценке интеграла

$$I = \frac{1}{j2\pi} \int_{-j\infty}^{j\infty} \frac{B(s)\, B(-s)}{A(s)\, A(-s)}\, ds, \tag{2.60}$$

где A и B — полиномы:

$$A(s) = a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n;$$

$$B(s) = b_1 s^{n-1} + b_2 s^{n-2} + \ldots + b_{n-1} s + b_n.$$

Разложим полином $A(s)$ на четные и нечетные члены:

$$A(s)=\bar{A}(s)+\bar{\bar{A}}(s), \tag{2.61}$$

где $\bar{A}(s)=0{,}5\,[A(s)+(-1)^n(-s)]$; $\bar{\bar{A}}(s)=0{,}5[A(s)-(-1)^nA(-s)]$.

Пусть далее $A_k(s)$ и $B_k(s)$ $(k\leqslant n)$ определяются следующими соотношениями:

$$\begin{aligned} A_k(s) &= a_0^k s^k + a_1^k s^{k-1} + \ldots + a_k^k, \\ B_k(s) &= b_1^k s^{k-1} + b_2^k s^{k-1} + \ldots + b_k^k, \end{aligned} \tag{2.62}$$

где

$$A_{k-1}(s) = A_k(s) - \alpha_k \bar{A}_k(s); \tag{2.63}$$

$$B_{k-1}(s) = B_k(s) - \beta_k \bar{A}_k(s);$$

$$\alpha_k = a_0^k/a_1^k; \quad \beta_k = b_1^k a_1^k; \tag{2.64}$$

$$A_n(s) = A(s); \quad B_n(s) = B(s).$$

Введем также

$$I_k = \frac{1}{2\pi} \int_{-j\infty}^{j\infty} \frac{B_k(s)\,B_k(-s)}{A_k(s)\,A_k(-s)}\,ds.$$

Можно заметить, что $I_n=I$.

Как и для систем с дискретным временем, верна следующая теорема.

Теорема 2.6. Предположим, что все корни полинома $A(s)$ лежат в левой полуплоскости $\{s\colon \operatorname{Re} s\leqslant 0\}$. Тогда для последовательности интегралов I_k $(k=0, \ldots, n)$ справедливо следующее рекуррентное соотношение:

$$I_k = I_{k-1} + \beta_k^2/2\alpha_k, \quad k = 1,2,\ldots, n. \tag{2.65}$$

Можно получить выражение для I и в конечном виде

$$I = \sum_{k=1}^{n} \frac{\beta_k^2}{2\alpha_k}.$$

Рекуррентное соотношение (2.65) позволяет построить алгоритм для расчета интеграла (2.60) (табл. 2.2).

Каждая четная строка в таблице А получается сдвигом элементов предшествующей строки влево и соответствующей подстановкой нулей. Четные строки таблицы Б идентичны аналогичным строкам таблицы А. Элементы нечетных строк получаются из двух предыдущих по следующим формулам:

$$a_i^{k-1} = \begin{cases} a_{i+1}^k, & \text{если } i \text{ четное}, \\ a_{i+1}^k - \alpha_k a_{i+2}^k, & \text{если } i \text{ нечетное}; \end{cases}$$

Таблица 2.2

	Таблица А				Таблица Б			
	a_0^n	a_1^n	a_2^n	. . .	b_1^n	b_2^n	b_3^n	. . .
*	a_1^n	0	a_3^n	. . .	a_1^n	0	a_3^n	. . .
	a_0^{n-1}	a_1^{n-1}	a_2^{n-1}	. . .	b_1^{n-1}	b_2^{n-1}	b_3^{n-1}	. . .
	a_0^{n-1}	0	a_3^{n-1}	. . .	a_1^{n-1}	0	a_3^{n-1}	. . .
. . .	. . . . .				. . . . .			
*		a_1^2	0	·		a_1^2	0	
		a_0^1		a_1^1				b_1^1
		a_1^1		0				
				a_0^0 ·				

$$b_i^{k-1} = \begin{cases} b_{i+1}^k, \text{ если } i \text{ четное,} \\ b_{i+1}^k - \beta_k a_{i+1}^k, \text{ если } i \text{ нечетное,} \end{cases}$$

$$\alpha_k = a_0^k/a_1^k, \quad \beta_k = b_1^k/a_1^k, \quad i = 0,..., k-1.$$

Первые элементы выделенных символом * строк таблицы А представляют собой коэффициенты $a_1^k (k=0, ..., n)$. Получив из табл. 2.2 значения коэффициентов α_k, $\beta_k (k=0, ..., n)$, можно с помощью (2.65) вычислить интеграл (2.60).

ПРОГРАММА INTEC

Назначение: вычисление математического ожидания и дисперсии выходного сигнала непрерывной линейной системы с передаточной функцией $H(s)=B(s)/A(s)$, где $A(s)$ и $B(s)$ — полиномы, причем степень $B(s)$ по крайней мере на единицу меньше, чем у $A(s)$.

Параметры: те же, что и параметры программы INTED, только IERR=1, если все нули полинома А лежат в левой полуплоскости, и IERR=0 в противном случае; если коэффициент A(1) неположителен, всегда должно выполняться равенство B(1)=0.

Обращение: CALL INTEC (A, B, N, IERR, SV, RM, V, VM).

Пример. Анализ выходного сигнала непрерывной линейной системы с передаточной функцией (2.43), где

$$A(s)=s^6+3s^5+5s^4+12s^3+6s^2+9s+1,$$
$$B(s)=3s^5+s^4+12s^3+3s^2+9s+1.$$

Исходные данные: N=6, RM=0, SV=1.

Результат: IERR=1, M=0, V=30,368, получен на ЭВМ ЕС=1050 за 0,39 с.

```
INTEC: PROCEDURE(A,B,N,IERR,SV,RM,V,VM);
       DECLARE
       (A(8),B(*))
       BINARY FLOAT;
       IERR=1; V=0;
       IF A(1) <= 0 THEN GO TO MET7;
       DO K=1 TO N;
       IF A(K+1) <= 0 THEN GO TO MET7;
       ALFA=A(K)/A(K+1);
       BETA=B(K)/A(K+1);
       V=V+BETA**2/ALFA;      K1=K+2;
       IF K1-N > 0 THEN GO TO MET2;
       DO I=K1 TO N BY 2;
       A(I)=A(I)-ALFA*A(I+1);  B(I)=B(I)-BETA*A(I+1);
       END;
MET2:  END;
       VM=B(N+1)*RM/A(N+1);
       V=V*SV*3.1415926;      GO TO MET10;
MET7:  IERR=0;
MET10: RETURN;
       END INTEC;

TEST:  PROCEDURE OPTIONS(MAIN)
       DCL A(7),B(7);
       A(1)=1;  A(2)=3;  A(3)=5;
       A(4)=12; A(5)=6;  A(6)=9;
       A(7)=1;
       B(1)=0;  B(2)=3;  B(3)=1;
       B(4)=12; B(5)=3;  B(6)=9;
       B(7)=1;
       N=6; SV=1; RM=0;
       CALL INTEC(A,B,N,IERR,SV,RM,V,VM);
       END TEST;
```

Глава 3.

Синтез динамических систем управления

3.1. Постановка задачи

Корректная постановка математической задачи синтеза оптимального управления динамическими системами предполагает, что у разработчика системы имеются:

переменные, описывающие параметры состояния системы и параметры управления;

математическая модель процесса функционирования динамической системы — объекта управления;

критерий качества управления (одномерная или многомерная функция), с помощью которого выражается цель управления;

математические соотношения, описывающие ограничения, налагаемые на переменные состояния или управления.

В настоящей главе рассмотрены задачи детерминированного оптимального управления и все характеристики задач являются неслучайными функциями.

Состояние объекта управления в общем случае можно описать некоторой векторной переменной x, которая является точкой n-мерного ($n \geqslant 0$) евклидового

пространства $\mathbf{R}^n=\{\mathbf{x}^\tau=(x_1, \ldots, x_n)\}$. Пусть управляющие воздействия описываются m-мерным вектором $\mathbf{u}^\tau=(u_1, \ldots, u_m)\in\mathbf{R}^m$ $(m\geqslant 0)$.

В зависимости от конкретной постановки задачи могут возникать случаи, когда процесс функционирования системы должен рассматриваться либо непрерывно на каком-либо (возможно, неограниченном) отрезке времени, либо в отдельных дискретных точках этого отрезка. В первом случае *детерминированные непрерывные* динамические *системы* управления наиболее часто описываются нелинейными дифференциальными уравнениями

$$\dot{\mathbf{x}}(t)=\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t], \quad t_0\leqslant t\leqslant t_к, \tag{3.1}$$

где t_0, $t_к$ и $\mathbf{x}(t_0)$ заданы, а $\mathbf{f}$ — некоторая детерминированная n-мерная функция.

Процесс функционирования дискретных систем может быть описан с помощью нелинейных разностных уравнений

$$\mathbf{x}(k+1)=\mathbf{f}^k[\mathbf{x}(k), \mathbf{u}(k)],$$

где $k=0, \ldots, N-1$; $\mathbf{x}(0)$ и N заданы, а $\mathbf{f}^k$ — нелинейная n-мерная функция.

Здесь рассматриваются только задачи синтеза оптимального управления по одномерному (скалярному) критерию качества (синтез по векторным критериям).

Общий вид критерия качества может быть задан выражением:

для непрерывных систем

$$J=L^k[\mathbf{x}(t_к), t_к]+\int_{t_0}^{t_к} L[\mathbf{x}(t), \mathbf{u}(t), t]\,dt, \tag{3.2}$$

где L^k и L — некоторые скалярные функции;

для дискретных (многошаговых)

$$J=L^N[\mathbf{x}(N)]+\sum_{k=0}^{N-1} L^k[\mathbf{x}(k), \mathbf{u}(k)], \tag{3.3}$$

где $L^k(k=0, \ldots, N-1)$ — некоторые скалярные функции.

Обычно задача синтеза оптимального управления состоит в нахождении управляющих функций, обеспечивающих максимум или минимум критериев (3.2), (3.3).

Физические ограничения, налагаемые на переменные состояния и управления, наиболее часто выражаются с помощью систем алгебраических равенств и неравенств

$$\mathbf{G}_1[\mathbf{x}(t), \mathbf{u}(t), t]\leqslant 0, \quad \mathbf{G}_2[\mathbf{x}(t), \mathbf{u}(t), t]=0$$

($\mathbf{G}_1$, $\mathbf{G}_2$ — заданные векторные функции, а знаки неравенств и равенств понимаются покомпонентно). Иногда ограничения выражаются в интегральной форме: для непрерывных систем

$$\int_{t_0}^{t_к} \mathbf{G}_3[\mathbf{x}(t), \mathbf{u}(t), t]\,dt\leqslant 0,$$

$$\int_{t_0}^{t_к} \mathbf{G}_4[\mathbf{x}(t), \mathbf{u}(t), t]\, dt = 0;$$

для многошаговых динамических систем

$$\sum_{k=0}^{N} \mathbf{G}_3[\mathbf{x}(k), \mathbf{u}(k), k] \leqslant 0,$$

$$\sum_{k=0}^{N} \mathbf{G}_4[\mathbf{x}(k), \mathbf{u}(k), k] = 0.$$

При решении задач управления на ЭВМ необходимо выполнять дискретизацию моделей и сводить их к дискретным аналогам. Рассмотрим один из наиболее простых вариантов дискретной аппроксимации непрерывных систем управления. Для этого весь временной отрезок функционирования системы $[t_0, t_к]$ разобьем точками $t_0, t_1, \ldots, t_N$ на N равных интервалов $\Delta: t_{i+1} - t_i = \Delta i$ $(i = 0, \ldots, N-1)$. Функция управления $\mathbf{u}(t)$ на каждом из отрезков становится постоянной:

$$\mathbf{u}(t) = \mathbf{u}(t_i),\ t_i \leqslant t \leqslant t_{i+1},\ i = 0,\ \ldots,\ N-1.$$

Система дифференциальных уравнений (3.1) заменяется системой разностных уравнений

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \Delta \mathbf{f}[\mathbf{x}(t_i),\ \mathbf{u}(t_i),\ t_i] + O(\Delta).$$

Пренебрегая погрешностью $O(\Delta)$ и используя выражение для t_i, эту систему представим в виде

$$\mathbf{x}[\Delta(i+1)] = \mathbf{x}[\Delta i] + \Delta \mathbf{f}[\mathbf{x}(\Delta i),\ \mathbf{u}(\Delta i),\ \Delta i].$$

При этом критерий качества управления принимает вид

$$J = L^k[\mathbf{x}(\Delta N), \Delta N] + \Delta \sum_{i=0}^{N-1} L[\mathbf{x}(\Delta i), \mathbf{u}(\Delta i), \Delta i].$$

Аналогично трансформируются к дискретному виду и функции, описывающие ограничения, накладываемые на переменные состояния и управления.

Описанная схема позволяет свести задачу синтеза оптимальной управляющей функции $\mathbf{u}(t)$ $(t_0 \leqslant t \leqslant t_к)$ к задаче нахождения оптимальной последовательности $\mathbf{u}(0), \ldots, \mathbf{u}[(N-1)\Delta]$.

Оптимальное управление — это такое допустимое (т. е. удовлетворяющее заданной системе ограничений) управление, которое обеспечивает выбранному критерию максимальное или минимальное значение.

В качестве критерия выбирается какой-либо функционал, отражающий требуемые свойства системы: максимальные эффективность, экономичность, массу и т. д.

Пусть, например, функционирование системы описывается дифференциальными уравнениями $\mathbf{x}=\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ с критерием

$$J = L^k[\mathbf{x}(t_к), t_к] + \int_{t_0}^{t_к} L(\mathbf{x},\mathbf{u},t)\,dt.$$

Рассмотрим различные типы оптимизационных задач, связанные с различными видами критериев качества.

В задаче оптимизации конечного состояния системы используется критерий $J=L^k[\mathbf{x}(t_к),t_к]$. Например, требуется, чтобы количество продукции, производимой объектом, было максимальным к концу периода $[t_0, t_к]$. Если $L^k=0$, $L=\tilde{L}(\tilde{L}>0)$, то $J=\tilde{L}(t_к-t_0)$ и минимизация критерия дает известную задачу о максимальном быстродействии системы. Минимум функционала

$$J = \int_{t_0}^{t_к} \sum_{j=1}^{m} \beta_j|\mathbf{u}_j|dt,\ \beta_j>0,$$

свидетельствует о наименьшем суммарном расходе энергии в устройствах управления (например, на приводе), причем коэффициенты β_j играют роль коэффициентов значимости (или веса) того или иного управления.

Минимум функционала

$$J = \int_{t_0}^{t_к} \left(\alpha + \sum_{j+1}^{m} \beta_j|\mathbf{u}_j|\right)dt$$

отражает ситуацию, когда мы одновременно стремимся к минимуму времени и расхода энергии.

В качестве примера типичного вида ограничений, налагаемых на управляющие функции, приведем следующий: $|\mathbf{u}_j(t)|\leqslant a_j$ $(j=1, ..., m)$, что означает необходимость принадлежности управляющих переменных m-мерному кубу. Ограничения, заданные на значения переменных состояния в конечный момент времени,

$$\mathbf{G}_1[\mathbf{x}(t_к), t_к]\leqslant 0,\ \mathbf{G}_2[\mathbf{x}(t_к), t_к]=0$$

называют терминальными граничными условиями. Они жестко определяют значения вектора состояния в конечный момент времени.

Рассмотрим условия управляемости системы, при которых динамическую систему с помощью выбранного управления за конечный промежуток времени можно перевести из любого заданного начального состояния в любое требуемое. С методической точки зрения рассмотрение этого вопроса должно предшествовать решению задачи синтеза оптимального управления.

Пусть дискретная динамическая система описывается уравнениями

$$\mathbf{x}(k+1)=\mathbf{F}\mathbf{x}(k)+\mathbf{G}\mathbf{u}(k),\ k=0, ..., N-1,$$

где $\mathbf{x}(0)$ задано. Требуемые условия управляемости легко найти непосредственно из уравнений системы. Действительно,

$$\mathbf{x}(1) = \mathbf{F}\mathbf{x}(0) + \mathbf{G}\mathbf{u}(0),$$
$$\mathbf{x}(2) = \mathbf{F}^2\mathbf{x}(0) + \mathbf{F}\mathbf{G}\mathbf{u}(0) + \mathbf{G}\mathbf{u}(1),$$
$$\mathbf{x}(N) = \mathbf{F}^N\mathbf{x}(0) + \mathbf{F}^{N-1}\mathbf{G}\mathbf{u}(0) + \ldots + \mathbf{G}\mathbf{u}(N-1).$$

Отсюда следует, что

$$\mathbf{x}(N) - \mathbf{F}^N\mathbf{x}(0) = [\mathbf{F}^{N-1}\mathbf{G} \vdots \mathbf{F}^{N-2}\mathbf{G} \vdots \ldots \vdots \mathbf{G}] \quad [\mathbf{u}(0), \ldots, \mathbf{u}(N-1)].$$

Последнее матричное уравнение имеет единственное решение $[\mathbf{u}(0), \ldots, \mathbf{u}(N-1)]^{\mathrm{T}}$ тогда, и только тогда, когда матрица $[\mathbf{F}^{N-1}\mathbf{G} \vdots \ldots \vdots \mathbf{G}]$ имеет ранг N при $N=n$. Это и есть условие управляемости для дискретной динамической системы.

Аналогичны условия управляемости и для непрерывных динамических систем. Рассмотрим, например, линейную систему

$$\dot{\mathbf{x}} = \mathbf{F}(t)\mathbf{x} + \mathbf{G}(t)\mathbf{u},$$

где $\mathbf{F}(t)$ и $\mathbf{G}(t)$ — известные матричные функции времени. Если через $\Phi(t,\tau)$ обозначить соответствующую фундаментальную матрицу, то решение системы можно записать в виде

$$\mathbf{x}(t) = \Phi(t,t_0)\,\mathbf{x}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\,\mathbf{G}(\tau)\,\mathbf{u}(\tau)\,d\tau$$

или

$$\int_{t_0}^{t} \mathbf{H}(t,\tau)\,\mathbf{u}(\tau)\,d\tau = \mathbf{x}(t) - \Phi(t,t_0)\,\mathbf{x}(t_0).$$

Так как вектор справа известный, то условие управляемости можно вывести из разрешимости этого уравнения: для управляемости системы необходимо и достаточно, чтобы строки матрицы $\mathbf{H}(t,\tau)$ были линейно независимы.

3.2. Принцип максимума в теории оптимальных систем

Один из самых эффективных методов синтеза нелинейных систем управления, известный как принцип максимума, предложен акад. Л. С. Понтрягиным. Принцип максимума определяет необходимые условия оптимальности управления в нелинейных системах и является, по выражению акад. Н. Н. Красовского, вычислительным методом, «широким по содержанию, строго обоснованным и удобным по форме для приложений». Остановимся на основных принципах метода.

Пусть динамика управляемого объекта описывается системой дифференциальных уравнений

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \quad t_0 \leqslant t \leqslant t_к, \tag{3.4}$$

причем начальные и конечные значения системы заданы:

$$\mathbf{x}(t_0)=\mathbf{x}^0,\ \mathbf{x}(t_к)=\mathbf{x}^k \tag{3.5}$$

($\mathbf{x}$ и $\mathbf{u}$ — n- и m-мерные векторы). Пусть также задан функционал

$$J(\mathbf{u})=\int_{t_0}^{t_к} L(\mathbf{x},\mathbf{u},t)\,dt.$$

Предположим, что $\mathbf{u}(t)$ — это кусочно-непрерывная вектор-функция, значения которой принадлежат замкнутому ограниченному множеству. Введем дополнительную координату x_0 с помощью соотношения $dx_0/dt=L(\mathbf{x},\mathbf{u},t)$. Теперь система составлена из $n+1$ уравнения и имеет столько же переменных, причем $\mathbf{f}_0(\mathbf{x},\mathbf{u},t)=L(\mathbf{x},\mathbf{u},t)$. Необходимо найти такое решение, чтобы дополнительная координата $x_0(t_к)=J$ имела наименьшее значение, а остальные координаты удовлетворяли условиям (3.4), (3.5).

Рассмотрим новый вектор Ψ, компоненты которого $\psi_0, \psi_1, \ldots, \psi_n$ удовлетворяют сопряженной системе уравнений

$$\frac{d\psi_i}{dt}=\sum_{i=0}^{n}\frac{\partial \mathbf{f}_j(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{x}_i}\psi_j,\quad i=0,\ldots,n. \tag{3.6}$$

Введем нелинейную функцию $H=H(\mathbf{x},\Psi,\mathbf{u})$ — функцию Гамильтона (или гамильтониан) системы — следующим образом:

$$H=\sum_{i=0}^{n}\psi_i\mathbf{f}_i(\mathbf{x},\mathbf{u},t). \tag{3.7}$$

Введение такой функции позволяет получить гамильтонову систему уравнений

$$\frac{dx_i}{dt}=\frac{\partial H}{\partial \psi_i},\ \frac{d\psi_i}{dt}=-\frac{\partial H}{\partial x_i},\ i=0,1,\ldots,n. \tag{3.8}$$

Теорема 3.1. (теорема Понтрягина). Чтобы допустимое управление $\mathbf{u}^*(t)$ было оптимальным, необходимо существование такой ненулевой непрерывной вектор-функции $\psi(t)$, соответствующей функциям $\mathbf{u}(t)$ и $\mathbf{x}(t)$, чтобы:

1) при любом t $(t_0\leqslant t\leqslant t_к)$ функция $H(\mathbf{x},\psi,\mathbf{u})$ достигала в точке $\mathbf{u}^*(t)$ максимума;

2) в конечной точке $t_к$ выполнялись соотношения $\psi_0(t_к)\leqslant 0$, $M[\psi(t_к),\mathbf{x}(t_к)]=\sup_{i\in D} H(\mathbf{x},\psi,\mathbf{u})=0$.

Сформулированная теорема позволяет построить эффективную схему алгоритма, реализующую принцип максимума.

1. Модель динамики системы приводится к форме Коши, т. е. системе дифференциальных уравнений

$$d\mathbf{x}_i/dt=\mathbf{f}_i(\mathbf{x},\mathbf{u},t),\ i=1,\ldots,n,$$
$$\mathbf{x}_i(t_0)=\mathbf{x}_i,\ \mathbf{u}\in D.$$

2. Строится функция Гамильтона (3.7) и записываются канонические уравнения вида (3.8).

3. Анализируется функция H и устанавливается, при каком $\mathbf{u}(t)$ достигается max H. При этом полученная функция $\mathbf{u}(t)$ обычно зависит от неизвестной функции $\psi(t)$.

4. На основании канонических уравнений строится и решается сопряженная система (3.6).

5. Вычислив функции $\psi_i(t)$, а затем и оптимальное управление $\mathbf{u}^{\text{опт}}(t)$, решим исходную систему для функции $\mathbf{x}(t)$.

Рассмотрим примеры решения задач синтеза оптимального управления с помощью принципа максимума Понтрягина.

Пример 1. Пусть дано уравнение системы

$$d^2x/dt^2=u, \text{ где } |u|<u_{\max}.$$

Требуется синтезировать оптимальную по быстродействию систему, т. е. найти управление, минимизирующее критерий $J=t_к-t_0=t_к$, $t_0=0$.

Обозначив $x=x_1$ и $dx/dt=x_2$, получим систему

$$dx_1/dt=x_2,\ dx_2/dt=u.$$

Функция Гамильтона $H=\psi_1x_2+\psi_2u$. Сопряженная система состоит из следующих уравнений:

$$d\psi_1/dt=0,\ d\psi_2/dt=-\psi_1.$$

Находим $\psi_1=c_1$, $\psi_2=c_2-c_1t$ (c_1, c_2—константы). Функция Гамильтона принимает вид

$$H=c_1x_2+(c_2-c_1t)u.$$

Максимум функции H будет достигаться при положительном слагаемом $(c_2-c_1t)u$, т. е. при $u=u_{\max}\operatorname{sgn}(c_2-c_1t)$. Функция c_2-c_1t дает уравнение прямой линии, пересекающей ось t один раз. Следовательно, оптимальная по быстродействию система оказывается релейной, а само управление имеет два предельных значения $u_{\max}$ и $-u_{\max}$.

Пример 2. Пусть объект управления описывается уравнением

$$T(dx/dt)+x=ku,$$

где $|\dot{u}|\leqslant u_{\max}$ и задано начальное значение $x(t_0)=x^0$.

Требуется определить управляющее воздействие таким образом, чтобы получить минимум функционала $J(u)=\int_{t_0}^{t_к}x^2dt$. Обозначив $x=x_1$ и $x_2=\int_{t_0}^{t_к}x_1^2dt$, каноническую форму уравнений запишем в виде

$$\frac{dx_1}{dt}=-\frac{1}{T}x_1+\frac{k}{T}u,\quad \frac{dx_2}{dt}=x_1^2.$$

Составим функцию Гамильтона

$$H=\psi_1\left(-\frac{1}{T}x_1+\frac{k}{T}u\right)+\psi_2x_1^2$$

или

$$H = \psi_2\left(x_1 - \frac{\psi_1}{2\psi_2 T}\right)^2 - \frac{\psi_1^2}{4\psi_2 T^2} + \frac{\psi_1}{T} ku.$$

Отсюда видно, что u обеспечит максимум функции H, если совпадают знаки ψ_1 и u, т. е. если $u = u_{\max} \operatorname{sgn} \psi_1$. Канонические уравнения Гамильтона имеют вид

$$\frac{dx_1}{dt} = -\frac{1}{T}x_1 + \frac{k}{T}u_{\max} \operatorname{sgn} \psi_1, \quad \frac{dx_2}{dt} = x_1^2,$$

$$\frac{d\psi_1}{dt} = \frac{\psi_1}{T} - 2x_1\psi_2, \quad \frac{d\psi_2}{dt} = 0.$$

Начальные условия: $x_1(t_0) = x^0$, $x_2(t_0) = 0$.
Граничные условия: $\psi_1(t_к) = -b_1$, $\psi_2(t_к) = -b_2(b_1, b_2) = \text{const}$.
Учитывая, что $d\psi_2/dt = 0$, получаем:

$$\psi_2 = \text{const}, \quad \frac{d\psi_1}{dt} = \frac{\psi_1}{T} + 2x_1 b_2.$$

Теперь необходимо решить уравнения

$$T\frac{dx_1}{dt} + x_1 = ku_{\max} \operatorname{sgn} \psi_1; \qquad T\frac{d\psi_1}{dt} - \psi_1 = 2Tb_2x_1$$

при $x_1(t_0) = x^0$ и $\psi_1(t_к) = -b_1$. Если $b_1 > 0$, то $\operatorname{sgn} \psi_1(t_к) < 0$, и в конечной точке после решения уравнения $T(dx_1/dt) + x_1 = -ku_{\max}$ получаем

$$x_1 = -ku_{\max}\left(1 - e^{-t_к/T}\right);$$

задавая $x_1 = x^0$, после решения уравнения $T(d\psi_1/dt) - \psi_1 = 2Tb_2x^0$ находим:

$$\psi_1 = 2Tb_2 x^0\left(1 - e^{t/T}\right),$$

$$x(t) = -ku_{\max}\left(1 - e^{-t/T}\right).$$

При $t = 0$ и $\psi_1 = 0$, если $t > 0$, то и $\psi_1 < 0$. Тогда в качестве оптимального процесса можно записать функцию

$$x(t) = -ku_{\max}\left(1 - e^{-t/T}\right)$$

3.3. Метод динамического программирования

Динамическое программирование — специальный вычислительный метод решения задач оптимизации управления динамических систем, позволяющий представить процесс оптимизации в виде последовательности отдельных этапов (шагов). Основу этого метода составляет *принцип оптимальности*, утверждающий, что каков бы ни был путь достижения некоторого состояния системы, последующие решения

должны принадлежать оптимальной траектории для оставшейся части пути, начинающейся с этого состояния. Применение этого принципа позволяет получить все используемые в динамическом программировании функциональные рекуррентные соотношения. Метод динамического программирования пригоден для решения задач синтеза оптимального управления как непрерывных, так и дискретных систем. Особенностями метода являются хорошая алгоритмизуемость и возможность эффективного использования современных средств вычислительной техники.

Пусть движение *непрерывной системы* описывается дифференциальным векторным уравнением

$$\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t),\ t_0\leqslant t\leqslant t_к,$$

с граничным условием $\mathbf{G}[\mathbf{x}(t_к),\ t_к]=0$.

Критерий качества управления системой определяется соотношением

$$J=L^k[\mathbf{x}(t_к),\ t_к]+\int_{t_0}^{t_к}L[\mathbf{x}(\tau),\ \mathbf{u}(\tau),\tau]\,d\tau$$

(J следует минимизировать). Обозначим через $J^{опт}(x,\ t)$ оптимальное значение критерия при условии, что начальная точка есть $(\mathbf{x},\ t)$. Рассмотрим движение из точки $(\mathbf{x},\ t)$ в близкую точку $(\mathbf{x}_1,\ t_1)=[\mathbf{x}+\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t)\Delta t,\ t+\Delta t]$ в течение промежутка времени Δt. Предположим, что управление $\mathbf{u}(t)$ при движении из точки $(\mathbf{x}_1,t_1)$ оптимальное. Если функция $J^{опт}$ дифференцируема по $(\mathbf{x},\ t)$, то с помощью разложения $J^{опт}$ легко получить следующее неравенство:

$$J^{опт}(\mathbf{x},\ t)\leqslant J^{опт}[\mathbf{x}+\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t)\Delta t,\ t+\Delta t]+L(\mathbf{x},\ \mathbf{u},\ t)\Delta t.$$

При этом знак равенства достигается только в том случае, когда на отрезке времени $[t,\ t_1]$ использовалось оптимальное управление. Это условие можно записать с помощью соотношения

$$J^{опт}(\mathbf{x},\ t)=\min_{\mathbf{u}}\{J^{опт}[\mathbf{x}+\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t)\Delta t,\ t+\Delta t]+L(\mathbf{x},\ \mathbf{u},\ t)\Delta t\} \tag{3.9}$$

или (разлагая правую часть (3.9) в ряд и устремляя Δt к нулю) с помощью нелинейного дифференциального уравнения в частных производных первого порядка (уравнения Беллмана)

$$\frac{\partial J^{опт}}{\partial t}=\min_{\mathbf{u}}\left[L(\mathbf{x},\ \mathbf{u},\ t)+\frac{\partial J^{опт}}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t)\right]. \tag{3.10}$$

Общая схема применения этого уравнения для нахождения оптимального управления следующая.

Вначале решается задача минимизации по $\mathbf{u}$ при каждом фиксированном $\mathbf{x}$ и t правой части уравнения (3.10). Таким образом, управление $\mathbf{u}(t)$ определяется как функция от $J^{опт}(\mathbf{x},\ t)$. После подстановки $\mathbf{u}(t)$ в правую часть (3.10) получим уравнение в частных производных для непосредственного нахождения функции $J^{опт}(\mathbf{x},\ t)$. При этом используется естественное граничное условие $J^{опт}(\mathbf{x},\ t_к)=L^k(\mathbf{x},\ t_к)$. После того как определена функция $J^{опт}(\mathbf{x},\ t)$, сразу же вычисляется и оптимальное управление $\mathbf{u}^{опт}(\mathbf{x},\ t)$, минимизирующее правую часть (3.10).

Оптимальное управление системой $\mathbf{u}^{опт}(t)$, которое соответствует ее начальному состоянию $[\mathbf{x}(t_0),\ t_0]$, определяется соотношением $\mathbf{u}^{опт}(t)=\mathbf{u}^{опт}(\tilde{\mathbf{x}}(t),\ t)$, где $\tilde{\mathbf{x}}(t)$ есть решение векторного дифференциального уравнения,

$$\dot{\mathbf{x}}=\mathbf{f}[\mathbf{x},\ \mathbf{u}^{опт}(\mathbf{x},\ t),\ t]\ (\tilde{\mathbf{x}}(t_0)=\mathbf{x}(t_0)).$$

Пусть *дискретная система* описывается с помощью рекуррентных уравнений

$$\mathbf{x}(k+1)=\mathbf{f}^{k+1}[\mathbf{x}(k),\ \mathbf{u}(k)],$$

где $k=0, 1, \ldots, N-1$; $\mathbf{x}(0)$ задано. Целью управления является минимизация функционала

$$J=\sum_{k=0}^{N-1} L[\mathbf{x}(k), \mathbf{u}(k)],$$

где L—скалярная функция. Естественно, что минимальное значение функционала J зависит от начального состояния $\mathbf{x}(0)$ (которое неуправляемо). Обозначим этот минимум через $j_N[\mathbf{x}(0)]$. Нетрудно видеть, что справедливы следующие соотношения:

$$j_N[\mathbf{x}(0)]=\min_{\mathbf{u}(0)}\min_{\mathbf{u}(1)}\ldots\min_{\mathbf{u}(N-1)}\{L[\mathbf{x}(0), \mathbf{u}(0)]+L[\mathbf{x}(1), \mathbf{u}(1)]+\ldots$$

$$\ldots+L[\mathbf{x}(N-1), \mathbf{u}(N-1)]\}=\min_{\mathbf{u}(0)}\{L[\mathbf{x}(0), \mathbf{u}(0)]\}+$$

$$+\min_{\mathbf{u}(1)}\ldots\min_{\mathbf{u}(N-1)}\{L[\mathbf{x}(1)\,\mathbf{u}(1)]+\ldots+L[\mathbf{x}(N-1), \mathbf{u}(N-1)]\}=$$

$$=\min_{\mathbf{u}(0)}\{L[\mathbf{x}(0), \mathbf{u}(0)]+j_{N-1}[\mathbf{x}(1)]\}.$$

Здесь $j_{N-1}[\mathbf{x}(1)]$ — минимальное значение критерия качества для процесса длительностью в $N-1$ шагов и имеющего начальное состояние $\mathbf{x}(1)$:

$$j_{N-1}[\mathbf{x}(1)]=\min_{\mathbf{u}(1)}\{L'[\mathbf{x}'(1), \mathbf{u}(1)]+j_{N-2}[\mathbf{x}(2)]\},$$

Для процесса управления с $N-q$ шагами, имеющего в качестве начального состояния $\mathbf{x}(q+1)(1\leqslant q\leqslant N-1)$

$$j_{N-1}[\mathbf{x}(q)]=\min_{\mathbf{u}(q)}\{L'[\mathbf{x}(q), \mathbf{u}(q)]+j_{N-q-1}[\mathbf{x}(q+1)]\}.$$

Последнее соотношение является дискретным вариантом уравнения Беллмана (3.8).

В качестве примера рассмотрим решение задачи об оптимальном выборе высоты и скорости летательным аппаратом [23, 24].

Пусть в начальный момент времени летательный аппарат находится на высоте H_0 и имеет скорость V_0. Надо перевести его на высоту $H_к > H_0$, получив при этом скорость $V_к > V_0$. Известен расход горючего для подъема аппарата с одной высоты на другую при заданной постоянной скорости и расход горючего для увеличения скорости при постоянной высоте полета. Требуется найти оптимальный режим набора высоты и увеличения скорости, при котором расход горючего J будет наименьшим.

Предположим, что процесс набора высоты и скорости состоит из нескольких шагов, на каждом летательный аппарат увеличивает или только высоту, или только скорость. Таким образом, состояние летательного аппарата определяется высотой H и скоростью V. Фазовая траектория, переводящая точку S, изображающую летательный аппарат, на фазовой плоскости (V, H) из положения $S_0=(V_0, H_0)$ в положение $S_к=(V_к, H_к)$ в данном случае является некоторой ломаной.

Разобьем приращение высоты $(H_к - H_0)$ на n_1 равных частей с шагом $\Delta H = (H_к - H_0)/n_1$, приращение скорости $(V_к - V_0)$ на n_2 равных частей с шагом $\Delta V = (V_к - V_0)/n_2$. Тогда весь процесс набора высоты и скорости будет состоять из $n_0 = n_1 + n_2$ шагов. Точка S из S_0 в $S_к$ может перемещаться по горизонтальным и вертикальным отрезкам. Каждой траектории, переводящей точку S из S_0 в $S_к$, соответствует свой расход горючего J.

Возьмем $n_1 = n_2 = 4$ (частный случай позволит проследить процесс оптимизации целиком от начала до конца) и займемся оптимизацией последнего (8-го) шага (рис. 3.1, *а*). В конце 7-го шага (шага $n_0 - 1$) точка S может оказаться либо в точке S_7^1 (нижний индекс — номер шага, верхний — номер возможного состояния, причем нумерация ведется по диагонали снизу вверх), либо в точке S_7^2, из которых за один последний шаг можно перейти в точку $S_к$ единственным образом (рис. 3.1, *б*). Поэтому выбора условного управления на 8-м шаге нет — оно единственное для каждой точки S_7^1 и S_7^2. Запишем около каждой из этих точек минимальный (и в данном случае неизбежный) расход горючего, необходимый, чтобы из них попасть в точку $S_к$, стрелкой обозначим направление движения точки S из данного состояния.

Чтобы выбрать оптимальное управление на предпоследнем 7-м шаге, рассмотрим все исходы предыдущего 6-го шага, из которого за два шага можно попасть в точку $S_к$. Таких исходов три — точки S_6^1, S_6^2, S_6^3 (рис. 3.1, *б*). Переход из точек S_6^1 и S_6^3 в точку $S_к$ за два шага можно выполнить лишь единственным образом со стоимостью в 17 и 25 единиц горючего (оптимальное движение из этих точек отмечено стрелками). Из точки S_6^2 в точку $S_к$ за два шага можно попасть двумя способами: направо вверх или вверх направо. В первом случае потребуется $15 + 9 = 24$ единицы горючего, во втором $10 + 16 = 26$ единиц. Наименьший расход горючего, таким образом, будет при движении направо вверх (его отмечаем стрелкой). Отметим на схеме оптимальный расход горючего, требуемый для перехода из данного состояния в конечное. Если оптимальных управлений несколько, то возьмем произвольное из них.

Далее аналогично оптимизируем 5-й шаг, рассматривая все точки, из которых за три шага можно перейти в точку $S_к$. Продолжая этот процесс, дойдем до начальной точки S_0. При этом получим схему (рис. 3.1, *в*), где в каждом кружке приведен наименьший расход горючего для перемещения из этой точки в точку $S_к$; стрелкой показано направление, в котором надо перемещаться из этой точки, чтобы расход горючего был наименьшим.

На основе схемы на рис. 3.1, *в* заключаем, что оптимальное управление будет такое: на 1-м шаге увеличиваем скорость на ΔV при постоянной высоте H_0, на 2, 3, 4 и 5-м шагах — набор высоты до $H_к$ при постоянной скорости $V_0 + \Delta V$, на 6, 7 и 8-м шагах увеличиваем скорость до $V_к$ при постоянной высоте $H_к$. При этом управлении будет наименьший расход горючего в 73 условные единицы. Фазовая траектория, соответствующая оптимальному управлению, показана на рис. 3.1, *г*.

Алгоритм метода динамического программирования удобно представить в виде таблицы (табл. 3.1).

Уравнение Беллмана можно записать в следующем виде:

$$j_k(\mathbf{x}) = \min_{\mathbf{u}} I_k(\mathbf{x}, \mathbf{u}),$$

где $I_k(\mathbf{x}, \mathbf{u}) = L(\mathbf{x}, \mathbf{u}) + j_{k-1}(\mathbf{x}')$; $\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u})$.

Для реализации алгоритма на ЭВМ требуется обязательное применение операции дискретизации переменных состояния x и управления u. Поэтому можно пред-

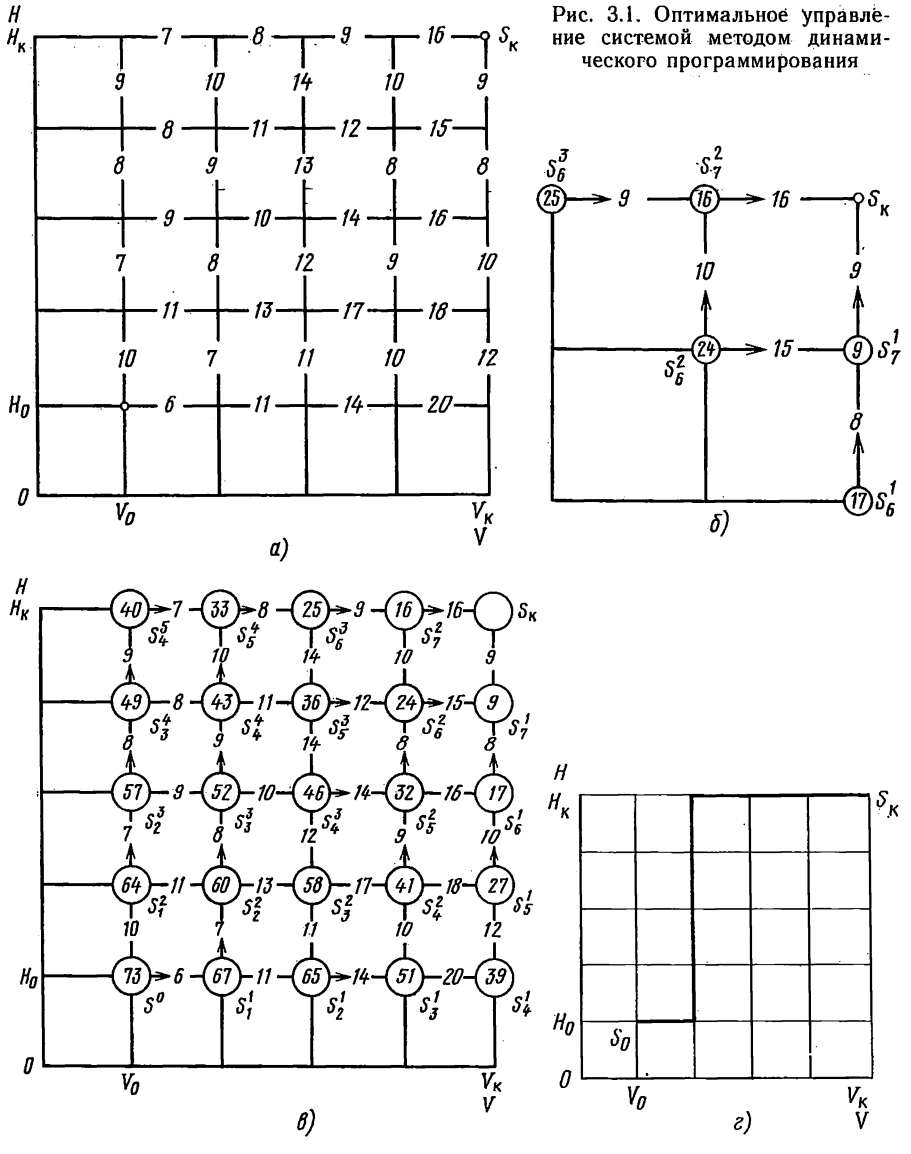


Рис. 3.1. Оптимальное управление системой методом динамического программирования

положить, что множество возможных состояний системы **x** и множество возможных значений управляющих переменных **u** конечны:

$$\mathbf{x}=\{x^{(1)}, \ldots, x^{(s)}\};\ \mathbf{u}=\{u^{(1)}, \ldots, u^{(t)}\},\ s\geqslant 1,\ t\geqslant 1.$$

В столбцы 1 и 2 таблицы записываются все возможные значения переменных **x**

Таблица 3.1

1	2	3	4	5	6	7	8
$x^{(i)}$	$u^{(j)}$	$f(x^{(i)}, u^{(j)})$	$L(x^{(i)}, u^{(j)})$	$j_{k-1}[f(x^{(i)}, u^{(j)})]$	$I_k(x^{(i)}, u^{(j)})$	$j_k(x^{(i)})$	$u^{(i)}$

и $\mathbf{u}$ соответственно (всего st комбинаций), в столбцы 3—6 — соответствующие данным комбинациям $(x^{(i)}, u^{(j)})$ $(i=1, ..., s;\ j=1, ..., t)$ значения функций $\mathbf{f}(x^{(i)}, u^{(j)})$, $L(x^{(i)}, u^{(j)}, j_{k-1}[\mathbf{f}(x^{(i)}, u^{(j)})]$ и $I_k(x^{(i)}, u^{(j)})$. В столбцах 7, 8 приведены значения найденных величин $j_k(x^{(i)})$ и $\tilde{u}^{(i)}$ ($\tilde{u}^{(i)}$ — соответствующее оптимальное управление).

С помощью составленной таблицы проводятся расчеты, в которых определяются величины $j_k(\mathbf{x})$ $(k=1, ..., N)$ и соответствующие значения оптимальных управлений $\mathbf{u}$. Вычисления осуществляются для k, принимающих последовательно значения 1, 2, ..., N, причем для первого этапа расчетов справедливы соотношения

$$j_0(\mathbf{x})=0,\ j_1(\mathbf{x})=\min_{\mathbf{u}} L(\mathbf{x}, \mathbf{u}), \qquad I_1(\mathbf{x}, \mathbf{u})=L(\mathbf{x}, \mathbf{u}).$$

Таким образом, оказываются составлены N вариантов расчетной таблицы, которые соответствуют $j_k(x)$ $(k=1, ..., N)$. Теперь по таблице для $j_N(\mathbf{x})$ и по заданному $\mathbf{x}(0)$ вычисляем $\mathbf{u}^{\text{опт}}(0)$ и $\mathbf{x}(1)=\mathbf{f}[\mathbf{x}(0), \mathbf{u}^{\text{опт}}(0)]$. После этого для $j_{N-1}(\mathbf{x})$ и по найденному $\mathbf{x}(1)$ вычисляем $\mathbf{u}^{\text{опт}}(1)$, $\mathbf{x}(2)=\mathbf{f}[\mathbf{x}(1), \mathbf{u}^{\text{опт}}(1)]$ и т. д. В результате этой последовательности операций и находим оптимальную последовательность управления $(\mathbf{u}^{\text{опт}}, ..., \mathbf{u}^{\text{опт}}_{N-1})$.

ПРОГРАММА DIN*

Назначение: оптимизация управления методом динамического программирования. Разработана на языке Фортран.

Входные параметры:

N — число измерений пространства;

K(N)—K(I) — число точек по i-й координате;

L(N)—L(I) — число точек в грани куба по i-й координате;

NP0 — число начальных точек;

KP0(N, NP0) — координаты начальных точек;

NKAP — число параметров состояния объекта в точке фазового пространства NKAP=1 (критерий)+число дополнительных параметров;

AK0(NP0, NKAP) — начальные значения параметров состояния;

IRS (IRS (I)) — максимальное расстояние просмотра по i-й координате при расчете элементарных переходов;

*Разработана канд. техн. наук В. Г. Байковым.

IGS(I) — направление просмотра по i-й координате: 0 — вперед и назад, +1 — только вперед, —1 — только назад;

IS — тип подготовки пространства: +1 — расширение трубки, 0 — формирование трубки, —1 — подготовка всего пространства;

IR — радиус трубки;

NPSS — число опорных точек трубки;

KRS(N, NPSS) — координаты опорных точек трубки (если хотя бы одна координата точки $\leqslant 0$, то со следующей точки начинается новая траектория);

MS — число точек, зарезервированных под выходную оптимальную траекторию;

KPE(N) — координаты конечной точки;

BP(N, MMK) — координатная сетка;

BP(I, J) — значение координаты реального фазового пространства в j-м узле по i-й координате;

NKZ(NZ)—NKZ(I) — номер куба, размещенного в i-й зоне оперативной памяти ЭВМ (необходимо задать NKZ(I)=I, I=1, ..., NK=NZ);

NP — число точек в пространстве: NP=K(1)*... * K(N);

NK — число кубов;

NPK — число точек в кубе: NPK=L(1)*... * L(N);

NKP — число кубов, необходимых для расчетов элементарных переходов из одного куба: $NKP=3^N$ при IBS(I)$\leqslant$L(I), I=1, ..., N $MMK=\max\limits_{1\leqslant I\leqslant N} K(I)$;

M — число точек пространства, размещающихся в оперативной памяти: M=NP в настоящей версии;

NZ — число кубов пространства, размещающихся в оперативной памяти: NZ=M/NPK=NK в настоящей версии;

KK(N)—KK(I) — число кубов по i-й координате;

EX — тип экстремума: EX=+1—max, EX= —1—min;

EPS — точность сравнения критерия.

Внутренние рабочие массивы (необходимо их задать для организации вычислительного процесса):

IPP(M) — параметр точки;

IGP(M) — история прихода;

AKAP(M, NKAP) — параметр состояния;

IPK(NK) — параметр куба;

NKDK(NKD) — массив номеров кубов, необходимых для расчетов элементарных переходов из одного куба;

BB(NKAP) — начальные значения параметров состояния;

BE(NKAP) — конечные значения параметров состояния.

Выходные параметры:

MSR — действительное число точек в оптимальной траектории;

IST — параметр оптимальной траектории: 2—прилегает к границе трубки, 1—прилегает к границе пространства, 0—не прилегает (кроме начальной и конечной точек);

IPS(MS) — выходной массив (содержит номера точек оптимальной траектории);

FG — процедура определения допустимости точки фазового пространства. Заголовок процедуры FG (N, X, I, F), где N — размерность пространства; X(N)—координаты точки фазового пространства (физические, а не целочисленные); I — выходной индикатор; I=0 — точка допустима; I=1 — точка недопустима;

F — процедура расчета элементарного перехода в фазовом пространстве. Заголовок процедуры F(N, NKAP, XB, AB, XE, AE, IT), где параметры N, NKAP определены выше; XB(N) — координаты начальной точки (физические); AB(NKAP) — параметры состояния в начальной точке; XE(N) — координаты конечной точки (физические); AE(NKAP) — параметры состояния в начальной точке; IT — выходной индикатор, принимающий следующие значения:

IT { =0 — нормальный расчет элементарного перехода,
   { ≠0 — переход невозможен.

Логическое условие вывода выходных параметров IST, MSR, IPS: если расчет происходит в трубке и IST=2, то расчет параметров DFB, DF1, DFA можно повторить и лишь затем вывести на печать параметры IST, MSR, IPS (IPS — массив). Отметим, что, если координаты точек оптимальной траектории в массиве IPS представлены в упакованном виде, для их распаковки можно использовать процедуру D02, обращаясь к ней с помощью следующего оператора CALL D02(IPS(I),N, K, KT, 1); здесь KT(␣ ⩾N) — массив координат i-й точки оптимальной траектории (массив KT необходимо описать).

Основные процедуры:

MAIN — ввод исходных данных, обращение к процедурам проверки, подготовки, поиска и анализа, вывод результатов расчетов;

DFB — подготовка фазового пространства;

DF(DF1 — упрощенная версия) — расчет переходов в пространстве;

DFA — получение и анализ оптимальной траектории.

Вспомогательные процедуры пользователя:

FG — определение допустимости точки фазового пространства;

F — расчет элементарного перехода в фазовом пространстве.

Имена задаются в параметрах процедур и имеют атрибуты EXTERNAL.

Структура главной (вызывающей) процедуры приведена в листинге контрольного примера.

```
COMMON /DDDD/ K(2),L(2),KK(2),KPE(2),IRS(2),IGS(2)
     ,IPP(2500),IGP(2500),AKAP(2500),IPK(64),NKZ(64),BP(2,60)
     ,IPS(100),KPS(2,10),KPO(2,4),AKO(4,1),HIPS(100),KT(2)
EXTERNAL DINA,DINAG
DATA N,NP,NK,NPK,NKD,MMK,M,NZ,NPSS,NPO,IR,MS,NKAP/
    2,100,4,25,4,10,100,4,0,1,0,50,1/
DO 5 I=1,2
K(I)=10
L(I)=5
KK(I)=2
IRS(I)=1
IGS(I)=0.
KPO(I,1)=1.
BP(I,1)=0.
BP(I,2)=10.
BP(I,3)=20.
BP(I,4)=30.
BP(I,5)=40.
BP(I,6)=50.
BP(I,7)=60.
BP(I,8)=70.
BP(I,9)=80.
BP(I,10)=90.
```

```
5       KPE(I)=10
        AKO(1,1)=0.
102     FORMAT(10(I6))
104     FORMAT(10E12.4)
105     FORMAT(8(I10))
        DO 2 J=1,N
2       N1=K(J)
        IS=-1
        DO 4 J=1,NZ
4       NKZ(J)=J
        EX=-1
        EPS=1.E-5
        CALL DFT1 (N,K,NP,L,KK,NK,NPK,NKD,MMK,M,NZ,
     *      IRS,NPO,KPO,IR,NKAP,EX,EPS)
        CALL DFB (IS,N,NP,M,NK,NKAP,NKO,NPK,NZ,NPSS,MMK,NPO,IR,
     *      K,L,KK,IPP,IGP,IPK,NKDK,AKAP,BP,NKZ,KPS,KPO,AKO,DINAG)
        CALL DF1 (N,NP,M,NK,NKAP,NKD,NPK,NZ,MMK,K,L,KK,
     *      IPP,IGP,IPK,BB,BE,NKDK,AKAP,BP,IRS,IGS,NKZ,EX,EPS,DINA)
        CALL DFA (N,NP,M,NKD,NZ,MMK,MS,MSR,IST,NPK,
     *      IPS,KPE,K,L,KK,IPP,IGP,NKZ,BP,NKDK,DINAG)
        WRITE(3, 102)IST,MSR
        WRITE(3, 102)IPS
        DO 33 I=1,100
        CALL DO2(IPS(I),N,K,KT,1)
        HI=I
        WRITE(3, 105)I,KT
33      HIPS(I)=IPS(I)
        WRITE(3, 104)HIPS
        STOP
        END

        SUBROUTINE DINA (N1,N2,AB,BB,AE,BE,I)
        DIMENSION AB(2),AE(2),X1(3),X2(3),Y1(3),Y2(3),Z1(3),Z2(3)
        DATA X1/10.5,23.,29./,X2/83.,45.,31./,Y1/72.,0.,90./,
     *      Y2/0.,77.,67./,Z1/50.,500.,500./,Z2/500.,50.,50./
        WRITE(3,201)
201     FORMAT(6X,'DINA')
        I=0
        BE=BB+SQRT((AE(1)-AB(1))**2+(AE(2)-AB(2))**2)
        DO 1 J=1,3
        A=(Y2(J)-Y1(J))/(X2(J)-X1(J))
        B=(Y1(J)*X2(J)-Y2(J)*X1(J))/(X2(J)-X1(J))
        IF(AB(1).NE.AE(1)) GO TO 2
        XS=AB(1)
        YS=XS*A+B
        GO TO 3
2       A1=(AE(2)-AB(2))/(AE(1)-AB(1))
        B1=(AB(2)*AE(1)-AE(2)*AB(1))/(AE(1)-AB(1))
        YS=(B1*A-B*A1)/(A-A1)
        XS=(YS-B)/A
3       IF(AB(1).EQ.AE(1)) GO TO 4
        IF(SIGN(1.,AB(1)-XS).EQ.SIGN(1.,AE(1)-XS)) GO TO 1
4       IF(AB(2).EQ.AE(2)) GO TO 5
        IF(SIGN(1.,AB(2)-YS).EQ.SIGN(1.,AE(2)-YS)) GO TO 1
5       IF(XS.LT.X1(J).OR.XS.GT.X2(J)) GO TO 1
        BE=BE+500.
1       CONTINUE
        RETURN
        END

        SUBROUTINE DINAG(N,AB,I)
        DIMENSION AB(2)
```

```
      WRITE(3,201)
201   FORMAT(6X,'DINAG')
      I=0
      IF(AB(1).GT.55..AND.AB(1).LT.75..AND.AB(2).GT.35.) I=
      RETURN
      END

      SUBROUTINE DFT1(N,K,NP,L,KK,NK,NPK,NKD,MMK,M,NZ,
     *     IRS,NPO,KPO,IR,NKAP,EX,EPS)
      DIMENSION K(N),L(N),KK(N),IRS(N),KPO(N,NPO)
      WRITE(3,201)
201   FORMAT(4X,'DFT1')
      I=0
      IF(N.LT.1) GO TO 100
      DO 77 J=1,N
      IF(K(J).LT.1) GO TO 100
77    CONTINUE
      I=1
      NPT=1
      DO 1 J=1,N
1     NPT=NPT*K(J)
      IF(NPT.NE.NP) GO TO 100
      I=2
      DO 2 J=1,N
      IF(K(J)/L(J).NE.KK(J)) GO TO 100
      IF(KK(J)*L(J).NE.K(J)) GO TO 100
2     CONTINUE
      I=3
      NKT=1
      DO 3 J=1,N
3     NKT=NKT*KK(J)
      IF(NKT.NE.NK) GO TO 100
      I=4
      NPKT=1
      DO 4 J=1,N
4     NPKT=NPKT*L(J)
      IF(NPKT.NE.NPK) GO TO 100
      I=5
      NKDT=3**N
      IF(NKDT.GT.NK) NKDT=NK
      IF(NKDT.NE.NKD) GO TO 100
      I=6
      MMKT=K(1)
      DO 5 J=1,N
      IF(K(J).GT.MMKT) MMKT=K(J)
5     CONTINUE
      IF(MMKT.NE.MMK) GO TO 100
      I=7
      IF(NPK*NKD.GT.M) GO TO 100
      I=8
      IF(NZ.LT.NKD) GO TO 100
      IF(NZ*NPK.GT.M) GO TO 100
      I=9
      DO 6 J=1,N
      IF(IRS(J).GT.L(J).OR.IRS(J).LT.0) GO TO 100
6     CONTINUE
      I=12
      DO 13 J=1,NPO
      DO 13 J1=1,N
      IF(KPO(J1,J).LT.1.OR.KPO(J1,J).GT.K(J1)) GO TO 100
13    CONTINUE
      I=13
      DO 14 J=1,N
```

```
      IF(IR.GT.L(J)) GO TO 100
14    CONTINUE
      I=14
      IF(NKAP.LT.1) GO TO 100
      I=15
      IF(ABS(EX).GT.2..OR.ABS(EX).LT..5) GO TO 100
      I=16
      IF(ABS(EPS).GT..7) GO TO 100
16    RETURN
100   WRITE(3, 101)I
      STOP
101   FORMAT(6X,'ILLEGAL INPUT',I7)
      END

      SUBROUTINE DF1(N,NP,M,NK,NKAP,NKD,NPK,NZ,MMK,
     *     K,L,KK,IPP,IGP,IPK,BB,BE,NKDK,AKAP,BP,IRS,IGS,NKZ,
     *     EX,EPS,F)
      DIMENSION K(N),L(N),KK(N),IPP(M),IGP(M),IPK(NK),IRS(N),IGS(N),
     *     BB(NKAP),BE(NKAP),NKDK(NKD),NKZ(NZ),AKAP(M,NKAP),BP(N,MMK),
     *     KP1(3),KP2(3),KP2R(3),KK1(3),KK2(3),AB(3),AE(3),IZ(3)
      WRITE(3,201)
201   FORMAT(4X,'DF1')
      NPS=1
      DO 20 J=1,N
20    NPS=NPS*(IRS(J)*2+1)
      NEZ=NPK
1     II=1
      DO 2 I2=1,NK
      NKI=I2
      IF(IPK(NKI))5,2,2
5     CALL DO2(NKI,N,KK1,KK,1)
      IF (NP-M)6,6,3
6     NZ1=NKI
      GO TO 11
3     CONTINUE
11    NZ2=NZ1
      IIK=1
      DO 4 I1=1,NPK
      NPIR=I1
      IF(IPP(NPIR+NEZ*(NZ1-1)))10,4,4
10    CALL DO2(NPIR,N,KP1,L,1)
      DO 7 J=1,N
7     KP1(J)=KP1(J)+L(J)*(KK1(J)-1)
      CALL DO2(NPI,N,KP1,K,0)
      DO 34 J=1,N
34    AB(J)=BP(J,KP1(J))
      DO 35 J=1,NKAP
35    BB(J)=AKAP(NPIR+NEZ*(NZ1-1),J)
      DO 24 J=1,N
24    IZ(J)=-IRS(J)
      DO 29 I=1,NPS
      DO 25 J=1,N
      IF(IZ(J)) 26,25,26
25    CONTINUE
      GO TO 9
26    DO 27 J=1,N
      IF(IZ(J)*IGS(J)) 9,27,27
27    CONTINUE
      DO 45 J=1,N
      IF(IZ(J)/2*2-IZ(J)) 46,45,46
45    CONTINUE
      GO TO 9
```

```
46      DO 47 J=1,N
        IF(IZ(J)/3*3-IZ(J)) 48,47,48
47      CONTINUE
        GO TO 9
48      CONTINUE
        DO 8 J=1,N
        KP2(J)=KP1(J)+IZ(J)
        IF(KP2(J).LT.1.OR.KP2(J).GT.K(J)) GO TO 9
        KK2(J)=(KP2(J)-1)/L(J)+1
8       KP2R(J)=KP2(J)-L(J)*(KK2(J)-1)
        CALL DO2(NKI2,N,KK2,KK,0)
        CALL DO2(NPIR2,N,KP2R,L,0)
        IF(NKZ(NZ2)-NKI2)13,12,13
13      IF(NKI-NKI2) 15,14,15
14      NZ2=NZ1
        GO TO 12
15      IF(NP-M) 16,16,17
16      NZ2=NKI2
        GO TO 12
17      DO 18 J=1,NZ
        NZ2=NZ-J+1
18      CONTINUE
12      IF(IPP(NPIR2+NEZ*(NZ2-1))-2) 19,9,19
19      DO 36 J=1,N
36      AE(J)=BP(J,KP2(J))
        CALL F(N,NKAP,AB,BB,AE,BE,KZ)
        IF(KZ) 9,371,9
371     IF(IPP(NPIR2+NEZ*(NZ2-1))) 37,37,39
37      IF(EX*BE(1)-EX*AKAP(NPIR2+NEZ*(NZ2-1),1)) 9,9,38
38      IF(ABS((AKAP(NPIR2+NEZ*(NZ2-1),1)-BE(1))/BE(1))-EPS) 9,9,39
39      DO 44 J=1,NKAP
44      AKAP(NPIR2+NEZ*(NZ2-1),J)=BE(J)
        IPP(NPIR2+NEZ*(NZ2-1))=-1
        IGP(NPIR2+NEZ*(NZ2-1))=NPI
        IF(NKI-NKI2) 40,42,40
40      IPK(NKI2)=-1
        IF(NKI-NKI2) 9,41,41
41      II=0
        GO TO 9
42      IF(NPIR-NPIR2) 9,43,43
43      IIK=0
9       IZ(1)=IZ(1)+1
        DO 28 J=1,N
        IF(IZ(J).LE.IRS(J)) GO TO 29
        IZ(J)=-IRS(J)
        IZ(J+1)=IZ(J+1)+1
28      CONTINUE
29      CONTINUE
        IPP(NPIR+NEZ*(NZ1-1))=0
4       CONTINUE
100     FORMAT(5(2I4,E16.4))
        WRITE(3, 100)I2
        YYY1=(I2-1)*25+1
        YYY2=YY1+24
        WRITE(3, 100)(IPP(YYY),IGP(YYY),AKAP(YYY,1),YYY=YYY1,YYY2)
        IF(IIK.EQ.0) GO TO 11
        IPK(NKI)=0
2       CONTINUE
        IF(II.EQ.0) GO TO 1
        RETURN
        END
```

```
      SUBROUTINE DFB(IS,N,NP,M,NK,NKAP,NKD,NPK,NZ,NPSS,MMK,NPO,IR,
     *     K,L,KK,IPP,IGP,IPK,NKDK,AKAP,BP,NKZ,KPS,KPO,AKO,
     *     FG)
      DIMENSION K(N),L(N),KK(N),IPP(M),IGP(M),IPK(NK),NKDK(NKD),
     *     AKAP(M,NKAP),BP(N,MMK),NKZ(NZ),KPS(N,NPSS),KPO(M,NPO),
     *     AKO(NPO,NKAP),
     *     AB(7),KK1(7),KP1(7),IZ(7),KPF(7),AP(7),AP1(7),AP2(7)
      WRITE(3,201)
201   FORMAT(4X,'DFB')
      IF(IS) 60,22,21
22    DO 23 I2=1,NK
      IPK(I2)=2
      IF(NP-M) 24,24,25
25    CONTINUE
      GO TO 26
24    NZ1=I2
26    DO 23 I1=1,NPK
23    IPP(I1+NPK*(NZ1-1))=2
21    DO 30 J=1,N
30    KK1(J)=(KPS(J,1)-1)/L(J)+1
      CALL DO2(NKI,N,KK1,KK,0)
      IF(NP-M) 32,32,31
31    CONTINUE
      GO TO 33
32    NZ1=NKI
33    DO 27 I2=1,NPSS
      DO 34 J=1,N
      IF(KPS(J,I2)) 27,27,34
34    KPF(J)=KPS(J,I2)
29    DO 35 J=1,N
35    KK1(J)=(KPS(J,I2)-1)/L(J)+1
      CALL DO2(NKI1,N,KK1,KK,0)
      IF(NKI-NKI1) 36,39,36
36    IF(NP-M) 38,38,37
37    CONTINUE
      GO TO 39
38    NZ1=NKI
39    NZ2=NZ1
      NKIZ=NKI
      N1=(1+2*IR)**N
      DO 40 J=1,N
40    IZ(J)=-IR
      DO 63 I1=1,N1
      DO 41 J=1,N
      KP1(J)=KPF(J)+IZ(J)
      IF(KP1(J).LT.1.OR.KP1(J).GT.K(J)) GO TO 28
      AB(J)=BP(J,KP1(J))
      KP1(J)=(KP1(J)-1)/L(J)+1
41    KP1(J)=KP1(J)-L(J)*(KK1(J)-1)
      CALL DO2(NPI,N,KP1,L,0)
      CALL DO2(NKIZ1,N,KK1,KK,0)
      IF(NKI-NKIZ1) 42,45,42
42    IF(NKIZ-NKIZ1) 43,48,43
43    IF(NP-M) 44,44,46
44    NZ2=NKIZ1
      GO TO 48
45    NZ2=NZ1
      GO TO 48
46    DO 47 J=1,NZ
      NZ2=J
      IF(NKZ(J)-NKIZ1) 47,48,47
47    CONTINUE
```

```
48      CALL FG(N,AB,I)
        IF(IPP(NPI+NPK*(NZ2-1))-2) 28,49,28
49      IF(I) 28,50,28
50      IPP(NPI+NPK*(NZ2-1))=1
        IPK(NKIZ1)=1
28      IZ(1)=IZ(1)+1
        DO 65 J=1,N
        IF(IZ(J).LE.IR) GO TO 63
        IZ(J)=-IR
        IZ(J+1)=IZ(J+1)+1
65      CONTINUE
63      CONTINUE
        IF(I2.EQ.NPSS) GO TO 27
        DO 51 J=1,N
        IF(KPS(J,I2+1)) 27,27,52
52      IF(IABS(KPS(J,I2+1)-KPF(J))-1) 51,51,53
51      CONTINUE
        GO TO 27
53      Z1=0
        DO 54 J=1,N
        AP1(J)=KPF(J)
        AP2(J)=KPS(J,I2+1)
54      Z1=Z1+(AP2(J)-AP1(J))**2
        Z1=SQRT(Z1)
        Z=.5
55      Z=Z*2.
        I=1
        DO 56 J=1,N
        AP(J)=AP1(J)+Z*(AP2(J)-AP1(J))/Z1+.5
        KP1(J)=AP(J)
        IF(KP1(J).NE.KPF(J))I=0
56      CONTINUE
        IF(I.EQ.1) GO TO 55
        DO 57 J=1,N
57      KPF(J)=KP1(J)
        GO TO 29
27      CONTINUE
        GO TO 10
60      II=1
        DO 1 I2=1,NK
        CALL DO2(I2,N,KK1,KK,1)
        IF(NP-M) 3,3,4
4       CONTINUE
3       NZ1=I2
5       IPK(I2)=2
        DO 2 I1=1,NPK
        CALL DO2(I1,N,KP1,L,1)
        DO 6 J=1,N
        KP1(J)=KP1(J)+L(J)*(KK1(J)-1)
6       AB(J)=BP(J,KP1(J))
        CALL FG(N,AB,I)
        IF(I) 8,7,8
7       IPP(I1+NPK*(NZ1-1))=1
        IPK(I2)=1
        GO TO 2
8       IPP(I1+NPK*(NZ1-1))=2
2       CONTINUE
        IF(IPK(I2).EQ.1) II=0
1       CONTINUE
        IF(II) 10,10,9
9       WRITE(3, 101)
        STOP
```

```
101     FORMAT(6X,'NO POINTS FOUND')
10      II=1
        DO 11 I1=1,NPO
        DO 12 J=1,N
        KK1(J)=(KPO(J,I1)-1)/L(J)+1
12      KP1(J)=KPO(J,I1)-L(J)*(KK1(J)-1)
        CALL DO2(NKI,N,KK1,KK,0)
        CALL DO2(NPIR,N,KP1,L,0)
        IF(IPK(NKI)-2) 13,11,13
13      IF(NP-M) 15,15,14
14      CONTINUE
        GO TO 16
15      NZ1=NK1
16      IF(IPP(NPIR+NPK*(NZ1-1))-2) 17,11,17
17      IPP(NPIR+NPK*(NZ1-1))=-1
        IPK(NKI)=-1
        DO 20 J=1,NKAP
        IGP(NPIR+NPK*(NZ1-1))=0
20      AKAP(NPIR+NPK*(NZ1-1),J)=AKO(I1,J)
        IGP(NPIR+NPK*(NZ1-1))=0
        II=0
11      CONTINUE
        IF(II) 18,19,18
18      WRITE(3, 102)
        STOP
102     FORMAT(6X,'ALL STARTING POINTS ARE FORBIDDEN')
19      RETURN
        END

        SUBROUTINE DFA(N,NP,M,NKD,NZ,MMK,MS,MSR,IST,NPK,
     *      IPS,KPE,K,L,KK,IPP,IGP,NKZ,BP,NKDK,FG)
        DIMENSION IPS(MS),KPE(N),K(N),L(N),KK(N),IPP(N),IGP(M),
     *      NKZ(NZ),BP(N,MMK),NKDK(NKD),
     *      KPR(7),KP1(7),KK1(7),KPZ(7),AB(7),IZ(7)
        WRITE(3,201)
201     FORMAT(4X,'DFA')
        DO 32 J=1,N
        IF(KPE(J).GE.1.AND.KPE(J).LE.K(J)) GO TO 32
        WRITE(3, 102)
102     FORMAT(6X,'END-POINT DEFINED NOT CORRECTLY')
        GO TO 31
32      CONTINUE
        IST=0
        CALL DO2(IPS(1),N,KPE,K,0)
        N1=MS-1
        N2=3**N
        DO 1 I=1,N1
        CALL DO2(IPS(I),N,KP1,K,1)
        DO 2 J=1,N
        KK1(J)=(KP1(J)-1)/L(J)+1
2       KPR(J)=KP1(J)-L(J)*(KK1(J)-1)
        CALL DO2(NKI,N,KK1,KK,0)
        CALL DO2(NPI,N,KPR,L,0)
        IF(NP-M) 3,3,4
3       NZ1=NKI
        GO TO 5
4       CONTINUE
5       IPS(I+1)=IGP(NPI+NPK*(NZ1-1))
        IF(IPS(I+1).GT.0) GO TO 30
        MSR=I
        GO TO 31
30      IF(IST-2) 6,1,1
```

```
6       DO 7 J=1,N
7       IZ(J)=-1
        DO 8 I2=1,N2
        DO 9 J=1,N
        KPZ(J)=KP1(J)+IZ(J)
        IF(KPZ(J).GE.1.AND.KPZ(J).LE.K(J)) GO TO 20
        IF(I.EQ.1) GO TO 10
        IST=1
        GO TO 10
20      CONTINUE
        AB(J)=BP(J,KPZ(J))
        KK1(J)=(KPZ(J)+1)/L(J)+1
9       KPR(J)=KPZ(J)-L(J)*(KK1(J)-1)
        CALL DO2(NKIZ,N,KK1,KK,0)
        CALL DO2(NPI,N,KPR,L,0)
        CALL FG(N,AB,II)
        IF(II) 10,17,10
17      IF(NKI-NKI2)12,11,12
11      NZ2=NZ1
        GO TO 16
12      IF(NP-M)13,13,14
13      NZ2=NKI2
        GO TO 16
14      DO 15 J=1,NZ
        NZ2=J
        IF(NKZ(NZ2)-NKI2) 15,16,15
15      CONTINUE
16      IF(IPP(NPI+NPK*(NZ2-1))-2) 10,18,10
18      IST=2
        GO TO 1
10      IZ(1)=IZ(1)+1
        DO 21 J=1,N
        IF(IZ(J)-1) 8,8,23
23      IZ(J)=-1
        IZ(J+1)=IZ(J+1)+1
21      CONTINUE
8       CONTINUE
1       CONTINUE
        WRITE(3, 101)
101     FORMAT(6X,'NOT ENOUGH MEMORY FOR OPTIMISATION')
31      RETURN
        END

        SUBROUTINE DO2(NP,N,K,KM,I)
        DIMENSION K(N),KM(N)
        WRITE(3,201)
201     FORMAT(6X,'DO2')
        IF(I)1,1,2
1       NP=1
        DO 3 I1=1,N
        CALL DO1(KM,I1-1,LP)
3       NP=NP+LP*(K(I1)-1)
        GOTO 4
2       N1=NP
        DO 5 I1=1,N
        I2=N-I1+1
        CALL DO1(KM,I2-1,LP)
        K(I2)=(N1-1)/LP+1
5       N1=N1-LP*(K(I2)-1)
4       RETURN
        END
```

```
      SUBROUTINE DO1(L,N,LP)
      DIMENSION L(N)
      WRITE(3,201)
201   FORMAT(6X,'DO1')
      LP=1
      IF(N)1,1,2
2     DO 3 I=1,N
3     LP=LP*L(I)
1     RETURN
      END
```

3.4. Оптимизация дискретных динамических систем

Пусть детерминированная дискретная система описывается нелинейными разностными уравнениями

$$\mathbf{x}(k+1)=\mathbf{f}^k[\mathbf{x}(k), \mathbf{u}(k)], \tag{3.11}$$

где $k=0, \ldots, N-1$ — дискретные моменты времени изменения состояния системы, а $\mathbf{x}(0)$ и N заданы. Все $\mathbf{x}(k)$ $(k=0, \ldots, N)$ представляют собой n-мерные векторы, вектор управления $\mathbf{u}(i)$ $(i=0, \ldots, N-1)$ m-мерный.

Пусть критерий качества динамической системы (3.11) задан в виде функционала

$$J = L^N[\mathbf{x}(N)] + \sum_{k=0}^{N-1} L^k[\mathbf{x}(k), \mathbf{u}(k)]$$

(где $L^k(k=0, \ldots, N)$ — некоторые скалярные функции), который надо минимизировать при отсутствии ограничений на вектор управления $\mathbf{u}(k)$ $(k=0, \ldots, N-1)$.

Введем последовательность n-мерных векторов $\psi(k)$ (называемых функциями влияния) и скалярных функций H^k:

$$H^k = L^k[\mathbf{x}(k), \quad \mathbf{u}(k)] + \psi^{\mathrm{T}}(k)\,\mathbf{f}^k[\mathbf{x}(k), \mathbf{u}(k)], \quad k = 0, \ldots, N-1.$$

Чтобы определить последовательность оптимальных значений вектора управления $\mathbf{u}(k)$ (см. § 3.3), которая соответствует стационарному значению критерия J, необходимо решить следующую систему разностных уравнений:

$$\mathbf{x}(k+1)=\mathbf{f}^k[\mathbf{x}(k), \mathbf{u}(k)], \tag{3.12}$$

$$\psi(k) = \left[\frac{\partial f^k}{\partial \mathbf{x}(k)}\right]^{\mathrm{T}} \psi(k+1) + \left[\frac{\partial L^k}{\partial \mathbf{x}(k)}\right]^{\mathrm{T}} \tag{3.13}$$

(здесь и в дальнейшем будем предполагать существование всех необходимых производных). Вектор $\mathbf{u}(k)$ находим из условий

$$\frac{\partial H^k}{\partial \mathbf{u}(k)} = \frac{\partial L^k}{\partial \mathbf{u}(k)} + \psi^{\mathrm{T}}(k+1)\frac{\partial f^k}{\partial \mathbf{u}(k)} = 0, \quad k = 0, \ldots, N-1. \tag{3.14}$$

Полученная разностная задача (3.12)—(3.14) называется *двухточечной граничной*. Граничные условия для комплектующих ее уравнений разделены: $\mathbf{x}(0)$ задано в начальный момент $t=0$, а $\psi(N)=[\partial L^N/\partial \mathbf{x}(N)]^{\mathrm{T}}$ — в конечный момент $t=N$.

Чтобы критерий качества J достигал локального минимума на некоторой

последовательности J, дополнительно к условию $\partial H^k/\partial \mathbf{u}(k)$ (3.14) должно еще выполняться и условие неотрицательности дифференциала второго порядка от J

$$d^2J \geqslant 0, \tag{3.15}$$

где

$$d^2J = \frac{1}{2} d\mathbf{x}^{\mathrm{T}}(N) \frac{\partial^2 L^N}{d\mathbf{x}(N)\partial \mathbf{x}(N)} d\mathbf{x}(N) + \frac{1}{2} \sum_{k=0}^{N-1} [d\mathbf{x}^{\mathrm{T}}(k), d\mathbf{u}^{\mathrm{T}}(k)] \times$$

$$\times \begin{bmatrix} \dfrac{\partial^2 H^k}{d\mathbf{x}(k)\,\partial \mathbf{x}(k)} & \dfrac{\partial^2 H^k}{\partial x(k)\,\partial u(k)} \\ \dfrac{\partial^2 H^k}{\partial \mathbf{u}(k)\,\partial \mathbf{x}(k)} & \dfrac{\partial^2 H^k}{\partial \mathbf{u}(k)\,\partial \mathbf{u}(k)} \end{bmatrix} \begin{bmatrix} d\mathbf{x}(k) \\ d\mathbf{u}(k) \end{bmatrix}. \tag{3.16}$$

Здесь

$$d\mathbf{x}(k+1) = \frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)} d\mathbf{x}(k) + \frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)} d\mathbf{u}(k); \quad d\mathbf{x}(0) = 0.$$

Решение задачи (3.12)—(3.14), удовлетворяющее условию (3.15), и будет искомым оптимальным управлением.

Усложним задачу (3.12)—(3.14), предположив, что на фазовые координаты в конечный момент времени N наложено дополнительное ограничение

$$\mathbf{G}[\mathbf{x}(N)] = 0, \tag{3.17}$$

где $\mathbf{G}$ — некоторая q-мерная векторная функция ($q \leqslant n$).

Введем вектор ν дополнительных множителей и составим выражение для расширенного критерия качества:

$$\bar{J} = L^N[\mathbf{x}(N)] + \nu^{\mathrm{T}}\ \mathbf{G}[\mathbf{x}(N)] + \sum_{k=0}^{N-1} \{L^k[\mathbf{x}(k), \mathbf{u}(k)] +$$

$$+ \psi^{\mathrm{T}}(k+1)[\mathbf{f}^k[\mathbf{x}(k), \mathbf{u}(k)] - \mathbf{x}(k+1)]\}. \tag{3.18}$$

Если ввести скалярную последовательность

$$H^k = L^k[\mathbf{x}(k), \mathbf{u}(k)] + \psi^{\mathrm{T}}(k+1)\,\mathbf{f}^k[\mathbf{x}(k), \mathbf{u}(k)], \quad k = 0, \ldots, N-1,$$

и скалярную функцию

$$\Phi = L^N[\mathbf{x}(N)] + \nu^{\mathrm{T}}\,\mathbf{G}[\mathbf{x}(N)],$$

то выражение для расширенного критерия J (3.18) можно представить в виде

$$\bar{J} = \Phi[\mathbf{x}(N)] - \psi^{\mathrm{T}}(N)\,\mathbf{x}(N) + \sum_{k=1}^{N-1} [H^k - \psi^{\mathrm{T}}(k)\,\mathbf{x}(k)] + H^0. \tag{3.19}$$

Оптимальное управление $\mathbf{u}(k)$ ($k=0, \ldots, N-1$) находим из условия стационарности критерия $\bar{J}$, которое можно выразить с помощью следующих соотношений:

$$\frac{\partial H^k}{\partial \mathbf{u}(k)} = \frac{\partial L^k}{\partial \mathbf{u}(k)} + \psi^{\mathrm{T}}(k+1)\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)} = 0, \tag{3.20}$$

а также

$$\psi^{\tau}(k)=\frac{\partial L^k}{\partial \mathbf{x}(k)}+\psi^{\tau}(k+1)\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)},\quad k=0,\ \ldots,\ N-1, \tag{3.21}$$

причем

$$\psi(N)=\frac{\partial L^N}{\partial \mathbf{x}(N)}+\nu^{\tau}\frac{\partial \mathbf{G}}{\partial \mathbf{x}(N)}. \tag{3.22}$$

Следовательно, синтез оптимального управления сводится к решению уравнений *двухточечной краевой задачи* (3.20)—(3.22). Искомыми неизвестными являются: вспомогательный q-мерный вектор ν; последовательности вспомогательных n-мерных векторов (функций влияния) $\psi(0)$, ..., $\psi(N)$, m-мерных векторов управления $\mathbf{u}(0)$, ..., $\mathbf{u}(N-1)$, n-мерных векторов фазовых координат $\mathbf{x}(0)$, ..., $\mathbf{x}(N)$.

Таким образом, система (3.20)—(3.22) содержит $N(2n+m)+n+q$ неизвестных и столько же уравнений, а система (3.12)—(3.14)—$N(2n+m)+n$ неизвестных и столько же уравнений для их определения. Соотношения (3.20)—(3.22) дают необходимые условия оптимальности и требуют дополнительной проверки (3.15).

Задачу (3.20)—(3.22) можно решать последовательно слева направо, если выразить $\psi(k+1)$ через $\psi(k)$ и $\mathbf{x}(k)$:

$$\psi^{\tau}(k+1)=\left[\psi^{\tau}(k)-\frac{\partial L^k}{\partial \mathbf{x}(k)}\right]\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)}\right]^{-1}. \tag{3.23}$$

Однако вычисление обратной матрицы $[\partial \mathbf{f}^k/\partial \mathbf{x}(k)]^{-1}$ занимает значительное время даже при использовании быстродействующих ЭВМ. Алгоритм решения задачи, который не требует вычисления этой матрицы, предложен в [20].

1. Задается начальная последовательность управлений $\mathbf{u}(k)$ ($k=0, \ldots, N-1$) и решается система (3.12) последовательно слева направо. На каждом k-м этапе запоминаются значения $\mathbf{x}(k+1)$, а также финальное значение $\mathbf{G}[\mathbf{x}(N)]$.

2. Задается начальное значение вектора ν и решается система (3.21) последовательно справа налево. При решении используются значения векторов $\mathbf{u}(k)$ ($k=0$, ..., $N-1$) и $\mathbf{x}(l)$ ($l=0$, ..., N), полученные в п. 1.

Параллельно с решением системы (3.21) запоминаются значения $\partial H^k/\partial \mathbf{u}(k)$ и решается система следующих рекуррентных уравнений:

$$\mathbf{S}(k)=\mathbf{Z}_{\mathbf{xx}}(k)-\mathbf{Z}_{\mathbf{xu}}(k)\,\mathbf{Z}_{\mathbf{uu}}^{-1}(k)\,\mathbf{Z}_{\mathbf{ux}}(k),\quad k=N-1,\ldots,0,$$

где

$$\mathbf{S}(N)=\left[\frac{\partial^2 L^N}{\partial \mathbf{x}\partial \mathbf{x}}+\nu^{\tau}\frac{\partial^2 \mathbf{G}}{\partial \mathbf{x}\partial \mathbf{x}}\right]_{\mathbf{x}=\mathbf{x}(N)},$$

$$\mathbf{Z}_{\mathbf{xx}}(k)=\frac{\partial^2 H^k}{\partial \mathbf{x}(k)\,\partial \mathbf{x}(k)}+\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)}\right]^{\tau}\mathbf{S}(k+1)\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)}\right];$$

$$\mathbf{Z}_{\mathbf{ux}}(k)=\frac{\partial^2 H^k}{\partial \mathbf{u}(k)\,\partial \mathbf{x}(k)}+\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\right]^{\tau}\mathbf{S}(k+1)\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)}\right];$$

$$\mathbf{Z}_{\mathrm{xu}}(k)=\left[\mathbf{Z}_{\mathrm{ux}}(k)\right]^{\mathrm{T}};$$

$$\mathbf{Z}_{\mathrm{uu}}(k)=\frac{\partial^2 H^2}{\partial \mathbf{u}(k)\,\partial \mathbf{u}(k)}+\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\right]^{\mathrm{T}}\mathbf{S}(k+1)\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\right];$$

$$\mathbf{R}(k)=\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)}\mathbf{R}(k+1)-\mathbf{Z}_{\mathrm{xu}}(k)\,\mathbf{Z}_{\mathrm{uu}}^{-1}(k)\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\mathbf{R}(k+1),\quad k=N-1,\ldots,0;$$

$$\mathbf{R}(N)=\left[\frac{\partial \mathbf{G}^{\mathrm{T}}}{\partial \mathbf{x}}\right]_{\mathbf{x}=\mathbf{x}(N)};$$

$$\mathbf{Q}(k)=\mathbf{Q}(k+1)-\mathbf{R}^{\mathrm{T}}(k+1)\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\right]^{\mathrm{T}}\mathbf{Z}_{\mathrm{uu}}^{-1}(k)\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}}\mathbf{R}(k+1),\quad k=N-1,\ldots,0;$$
$$\mathbf{Q}(N)=0;$$

$$\mathbf{h}(k)=\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}(k)}\mathbf{h}(k+1)+\mathbf{Z}_{\mathrm{xu}}(k)\,\mathbf{D}(k),\quad k=N-1,\ldots,0;$$

$$\mathbf{D}(k)=\mathbf{Z}_{\mathrm{uu}}^{-1}(k)\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\mathbf{h}(k+1)-d\frac{\partial H^k}{\partial \mathbf{u}(k)};\qquad \mathbf{h}(N)=0,$$

$$d\frac{\partial H^k}{\partial \mathbf{u}(k)}=-\varepsilon\frac{\partial H^k}{\partial \mathbf{u}(k)},\quad 0\leqslant\varepsilon\leqslant 1;$$

$$\mathbf{g}(k)=\mathbf{g}(k+1)-\mathbf{R}^{\mathrm{T}}(k+1)\left[\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\right]^{\mathrm{T}}\mathbf{D}(k),\quad k=N-1,\ldots,0,\quad \mathbf{g}(N)=0.$$

Для каждого k запоминаются $\mathbf{Z}_{\mathrm{uu}}^{-1}(k)$, $\mathbf{Z}_{\mathrm{ux}}(k)$, $\mathbf{Z}_{\mathrm{uu}}^{-1}(k)(\partial \mathbf{f}^k/\partial \mathbf{u}(k))\mathbf{R}(k+1)$, $\mathbf{D}(k)$, $\mathbf{Q}(0)$ и $\mathbf{g}(0)$.

3. Производится попытка выбрать такое значение $d\mathbf{G}$, чтобы приблизиться к требуемому условию $\mathbf{G}[\mathbf{x}(N)]=0$. Обычно $d\mathbf{G}$ выбирается в виде $d\mathbf{G}=\delta\mathbf{G}[\mathbf{x}(N)]$, где δ — некоторая константа: $0<\delta\leqslant 1$. Запоминается $d\nu=[-\mathbf{Q}(0)]^{-1}[d\mathbf{G}-\mathbf{g}(0)]$.

4. Итеративно повторяются пп. 1—3, чтобы улучшить значения последовательности векторов управления $\mathbf{u}(k)$ ($k=0,\ldots,N-1$) и вспомогательного вектора ν по формулам

$$\mathbf{u}_{\text{нов}}(k)=\mathbf{u}_{\text{ст}}+d\mathbf{u}(k),\quad k=0,\ldots,N-1,$$

$$\nu_{\text{нов}}=\nu_{\text{ст}}+d\nu,$$

где
$$d\mathbf{u}(k)=-\mathbf{Z}_{\mathrm{uu}}^{-1}(k)\left[\mathbf{Z}_{\mathrm{ux}}(k)\,d\mathbf{x}(k)+\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\mathbf{R}(k+1)\,d\nu+\frac{\partial \mathbf{f}^k}{\partial \mathbf{u}(k)}\mathbf{h}(k+1)-d\frac{\partial H^k}{\partial \mathbf{u}(k)}\right],$$

$$\delta\mathbf{x}(k)=[\mathbf{x}(k)]_{\text{нов}}-[\mathbf{x}(k)]_{\text{ст}},\quad k=0,\ldots,N-1,$$

а значение приращения dv на каждой итерации определяется в п. 3. Указанный итеративный процесс заканчивается по наступлении событий

$$\|G[\mathbf{x}(N)]\| < \delta_1;\ \max_k \left\| \frac{\partial H^k}{\partial \mathbf{u}(k)} \right\| \leqslant \delta_2,$$

где δ_1, δ_2 — некоторые константы точности, а $\|\cdot\|$ — символ нормы рассматриваемой величины (например, сумма абсолютных значений компонентов вектора $G[\mathbf{x}(N)]$ или матрицы $\partial \mathbf{H}^k/\partial \mathbf{u}(k)$).

В заключение остановимся на рассмотрении задач с более широким классом ограничений. Пусть на траекториях дискретной системы (3.9) заданы скалярные ограничения типа неравенств

$$C_k[\mathbf{x}(k), \mathbf{u}(k)] \leqslant 0,\ k = 0,\ldots, N. \tag{3.24}$$

Одним из самых простых и естественных подходов к решению сформулированной задачи является метод штрафных функций, состоящий в преобразовании критерия (3.3) путем введения дополнительного слагаемого

$$\tilde{J} = J + \mu \sum_{k=0}^{N} \{C_k[\mathbf{x}(k), \mathbf{u}(k)]\}^2 E(C_k),$$

в котором μ — положительная константа, а функция $E(C_k)=0$, если $C_k<0$, и $E(C_k)=1$, если $C_k \geqslant 0$. С помощью соответствующего выбора константы можно добиться приближенного удовлетворения ограничений (3.24). В то же время при использовании метода штрафных функций сходимость к оптимальному решению бывает достаточно медленной [20].

Другие подходы к решению задач синтеза оптимального управления дискретных динамических систем с ограничениями типа (3.24) можно найти в [29, 39].

3.5. Оптимизация непрерывных динамических систем

Предположим, что детерминированная непрерывная динамическая система описывается нелинейными дифференциальными уравнениями

$$\dot{\mathbf{x}} = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t],\ t_0 \leqslant t \leqslant t_к, \tag{3.25}$$

где t_0, $t_к$ и $\mathbf{x}(t_0)$ заданы; $\mathbf{x}(t)$ — n-мерный вектор состояния системы; $\mathbf{u}(t)$ — m-мерный вектор управления.

Предположим далее, что качество функционирования системы (3.25) определяется критерием

$$J = L^k[\mathbf{x}(t_к), t_к] + \int_{t_0}^{t_к} L[\mathbf{x}(t), \mathbf{u}(t), t]\, dt, \tag{3.26}$$

который необходимо минимизировать (L^k и L — некоторые скалярные функции).

Рассмотрим два случая.

1. Ограничения на векторы состояния системы (3.25) и управления отсутствуют. Введем вспомогательный критерий качества

$$\bar{J} = L^k[\mathbf{x}(t_к), t_к] + \int_{t_0}^{t_к} L[\mathbf{x}(t), \mathbf{u}(t), t] + \\ + \psi^т(t)\{\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] - \dot{\mathbf{x}}\}\, dt, \tag{3.27}$$

где $\psi(t)$ — n-мерный вектор вспомогательных переменных (функция влияния). Введем также гамильтониан

$$H[\mathbf{x}(t), \mathbf{u}(t), \psi(t), t] = L[\mathbf{x}(t), \mathbf{u}(t), t] + \psi^т \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]. \tag{3.28}$$

Подставляя выражение (3.28) в (3.27) и интегрируя по частям, получаем:

$$\bar{J} = L^k[\mathbf{x}(t_к), t_к] - [\psi^т(t_к)\,\mathbf{x}(t_к) - \psi^т(t_0)\,\mathbf{x}(t_0)] + \\ + \int_{t_0}^{t_к} \{H[\mathbf{x}(t), \mathbf{u}(t), t] + \psi^т(t)\,\dot{\mathbf{x}}(t)\}\, dt. \tag{3.29}$$

Вектор оптимального управления находим из условия стационарности критерия $\bar{J}$, которое можно записать с помощью следующих уравнений Эйлера—Лагранжа:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \tag{3.30}$$

$$\dot{\psi} = -\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^т \psi - \left[\frac{\partial L}{\partial \mathbf{x}}\right]^т, \tag{3.31}$$

$$\left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right]^т \psi + \left[\frac{\partial L}{\partial \mathbf{u}}\right]^т = 0, \tag{3.32}$$

последнее уравнение, эквивалентное условию $\partial H/\partial \mathbf{u} = 0$, служит для определения $\mathbf{u}(t)$.

Таким образом, задача синтеза оптимального управления непрерывной динамической системой (3.25) сводится к двухточечной краевой задаче (3.30)—(3.32). Граничные условия в ней разделены: $\mathbf{x}(t_0)$ задано в начальной точке t_0, $\psi(t_к) = [dL^k/\partial \mathbf{x}]^т$ — в конечной точке $t_к$.

Чтобы обеспечить локальный минимум критерия качества J, условия $\partial H/\partial \mathbf{u} = 0$ недостаточно; необходимо, чтобы вторая вариация $\delta^2 J$ при выполнении условия $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ была неотрицательной для всех бесконечно малых значений вариации управления:

$$\delta^2 J \geqslant 0, \tag{3.33}$$

где

$$\delta^2 J = \frac{1}{2}\left[\delta \mathbf{x}^т \frac{\partial L^k}{\partial \mathbf{x}^2} \delta \mathbf{x}\right]_{t=t_к} + \frac{1}{2}\int_{t_0}^{t_к} [\delta \mathbf{x}^т, \delta \mathbf{u}^т] \times \begin{bmatrix} \dfrac{\partial^2 H}{\partial \mathbf{x}^2} & \dfrac{\partial^2 H}{\partial \mathbf{x}\,\partial \mathbf{u}} \\ \dfrac{\partial^2 H}{\partial \mathbf{u}\,\partial \mathbf{x}} & \dfrac{\partial^2 H}{\partial \mathbf{u}^2} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix} dt, \tag{3.34}$$

при условии $\delta(\dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}, t)) = 0$, что эквивалентно следующему уравнению:

$$\frac{d}{dt}(\delta \mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \delta \mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \delta \mathbf{u}, \quad \delta \mathbf{x}(t_0) = 0, \tag{3.35}$$

[где $\delta \mathbf{u}$ — вариация управления $\mathbf{u}(t)$].

Усложним решаемую задачу, предположив, что в конечный момент времени $t_к$ задана некоторая q-мерная ($q \leqslant n$) функция $\mathbf{G}$ от фазовых координат $\mathbf{x}(t_к)$:

$$\mathbf{G}[\mathbf{x}(t_к), t_к] = 0 \tag{3.36}$$

Присоединим систему q алгебраических уравнений (3.36) к критерию качества (3.26), введя q-мерный вспомогательный вектор ν. Расширенный критерий качества $\bar{J}$ будет иметь следующий вид:

$$\bar{J} = L^k[\mathbf{x}(t_к), t_к] + \nu^т \mathbf{G}[\mathbf{x}(t_к), t_к] + \int_{t_0}^{t_к} \{L[\mathbf{x}(t), \mathbf{u}(t), t] + \psi^т[\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\mathbf{x}}]\}\, dt. \tag{3.37}$$

Необходимые условия стационарности критерия $\bar{J}$, совпадающего с J на решениях системы (3.30)—(3.32):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \tag{3.38}$$

$$\dot{\psi} = -\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^т \psi - \left[\frac{\partial L}{\partial \mathbf{x}}\right]^т, \tag{3.39}$$

$$\left[\frac{\partial H}{\partial \mathbf{u}}\right]^т = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right]^т \psi + \left[\frac{\partial L}{\partial \mathbf{u}}\right]^т = 0. \tag{3.40}$$

Гамильтониан H определяется по формуле (3.28). Граничные условия разделены: в начальный момент времени t_0 задан вектор $\mathbf{x}(t_0)$, в конечный момент времени $t_к$ заданы следующие условия:

$$\psi^т(t_к) = \left(\frac{\partial L^k}{\partial \mathbf{x}} + \nu^т \frac{\partial G}{\partial \mathbf{x}}\right)_{t = t_к}, \tag{3.41}$$

$$\mathbf{G}[\mathbf{x}(t_к), t_к] = 0.$$

Начальный вектор $\mathbf{x}(t_0)$, а также соотношения (3.41) определяют $2n + q$ граничных условий решения задачи (3.38)—(3.40).

Наиболее часто встречающимся на практике условием является разрешимость уравнений (3.36) относительно переменных $\mathbf{x}_1(t_к), \ldots, \mathbf{x}_q(t_к)$. В этом случае

$$\mathbf{G}[\mathbf{x}(t_к), t_к] = \begin{bmatrix} \mathbf{x}_1(t_к) - \mathbf{x}_1^k \\ \vdots \\ \mathbf{x}_q(t_к) - \mathbf{x}_q^k \end{bmatrix} = 0, \tag{3.42}$$

где $\mathbf{x}_1^k, \ldots, \mathbf{x}_q^k$ — некоторые заданные величины. Необходимые условия для оптимального управления задачи (3.38)—(3.40) с функцией $\mathbf{G}[\mathbf{x}(t_к), t_к]$, удовлетворяющей (3.42), следующие:

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)^{\mathrm{T}} \psi + \left(\frac{\partial L}{\partial \mathbf{u}}\right)^{\mathrm{T}} = 0, \tag{3.43}$$

$$\dot{\psi} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^{\mathrm{T}} \psi - \left(\frac{\partial L}{\partial \mathbf{x}}\right)^{\mathrm{T}}, \tag{3.44}$$

$$\psi_i(t_к) = \left(\frac{\partial L^k}{\partial \mathbf{x}_i}\right)_{t=t_к}, \quad i = q+1, \ldots, n. \tag{3.45}$$

В системе уравнений (3.43)—(3.45) n неизвестных условий $\psi(t_0)$ и n не заданных терминальных условий

$$\Psi = \{\psi_1(t_к), \ldots, \psi_q(t_к);\ \mathbf{x}_{q+1}(t_к), \ldots, \mathbf{x}_n(t_к)\}.$$

Соответственно этому можно разработать алгоритмы решения задачи (3.43)—(3.45), основанные на последовательных изменениях векторов либо $\psi(t_0)$, либо Ψ. Приведенный ниже алгоритм, названный *алгоритмом переходной матрицы*, использует процедуру последовательного выбора $\psi(t_0)$.

1. Выбираются некоторые приближения для начального значения вектора чувствительности $\psi(t_0)$.

2. Из соотношения (3.43) определяется вектор управления $\mathbf{u}(t)$. Интегрируя уравнения (3.38) и (3.39) от t_0 до $t_к$, определяются $\psi(t)$ и $\mathbf{x}(t)$ $(t_0 \leqslant t \leqslant t_к)$. Запоминаются $\mathbf{x}_1(t_к)$, ..., $\mathbf{x}_q(t_к)$, $\psi_{q+1}(t_к)$, ..., $\psi_n(t_к)$.

3. Определяются элементы переходной матрицы $[\partial \mathbf{S}(t_к)/\partial\psi(t_0)]$:

$$\delta \mathbf{S}(t_к) = \begin{bmatrix} \delta \mathbf{x}_1(t_к) \\ \cdots \\ \delta \mathbf{x}_q(t_к) \\ \delta\psi_{q+1}(t_к) \\ \cdots \\ \delta\psi_n(t_к) \end{bmatrix} = \frac{\partial \mathbf{S}(t_к)}{\partial \psi(t_0)} \delta\psi(t_0). \tag{3.46}$$

4. Стараются приблизить значения $\mathbf{S}(t_к)$ к желаемому $\mathbf{S}_k$, выбрав новое значение вектора $\delta\mathbf{S}(t_к) = -\varepsilon[\mathbf{S}(t_к) - \mathbf{S}_k]$, где ε — некоторая константа $(0 \leqslant \varepsilon \leqslant 1)$.

5. Вычисляется обратная матрица $[\partial \mathbf{S}(t_к)/\partial\psi(t_0)]^{-1}$, а также новое приращение $\delta\psi(t_0) = [\partial \mathbf{S}(t_к)/\partial\psi(t_0)]^{-1}\delta\mathbf{S}(t_к)$.

6. Повторяются пп. 1—5, при этом в каждом цикле изменяются значения $\psi(t_0)$ по формуле $\psi(t_0)_{нов} = \psi(t_0)_{ст} + \delta\psi(t_0)$ до тех пор, пока не будет выполнено условие $\|\mathbf{S}(t_к) - \mathbf{S}_к\| \leqslant \gamma$ (γ — заданная точность вычислений; $\|\cdot\|$ — символ некоторой нормы вектора $\mathbf{S}(t_к) - \mathbf{S}_к$).

Приведем два конкретных метода вычисления переходной матрицы $\partial \mathbf{S}(t_к)/\partial\psi(t_0)$. Первый основан на непосредственном решении систем уравнений (3.38), (3.39), которое повторяется n раз. При проведении каждой i-й $(i = 1, \ldots, n)$ процедуры решения i-й компонент вектора влияния $\psi(t_0) - \psi_i(t_0)$ получает приращение $\delta\psi_i(t_0)$ относительно опорного значения $\psi(t_0)$ (вычисленного в п. 1 или 6). Второй метод —

определение единичных решений системы — основан на n-кратном решении системы $2n$ линейных уравнений, описывающих возмущения системы (3.46):

$$\delta\dot{\mathbf{x}} = \mathbf{A}(t)\,\delta\mathbf{x} - \mathbf{B}(t)\,\delta\psi;$$

$$\delta\dot{\psi} = -\mathbf{C}(t)\,\delta\mathbf{x} - \mathbf{A}^{\mathrm{T}}(t)\,\delta\psi,$$

где $\mathbf{A}(t) = \mathbf{f}_{\mathrm{x}} - \mathbf{f}_{\mathrm{u}}\mathbf{H}_{\mathrm{uu}}^{-1}\mathbf{H}_{\mathrm{ux}}$; $B(t) = \mathbf{f}_{\mathrm{u}}\mathbf{H}_{\mathrm{uu}}^{-1}\mathbf{f}_{\mathrm{u}}^{\mathrm{T}}$; $\mathbf{C}(t) = \mathbf{H}_{\mathrm{xx}} - \mathbf{H}_{\mathrm{xu}}\mathbf{H}_{\mathrm{uu}}^{-1}\mathbf{H}_{\mathrm{ux}}$; $\delta\mathbf{x}(t)$ и $\delta\psi(t)$ — возмущения величин $\mathbf{x}(t)$ и $\psi(t)$, вызванные возмущениями начального состояния $\delta\mathbf{x}(t_0)$ и $\delta\psi(t_0)$. При каждом i-м из n решений уравнений (3.47), (3.48) i-й компонент вектора $\delta\psi(t_0)$ устанавливается равным единице, а остальные компоненты $\delta\psi(t_0)$, как и все компоненты $\delta\mathbf{x}(t_0)$,—нулю. Данный метод определения переходной матрицы (3.46) несколько более точный, чем метод непосредственного решения системы (3.38), (3.39), но требует решения дополнительной системы уравнений возмущенного движения, что увеличивает время счета.

2. Ограничения заданы в виде неравенств на фазовые и управляющие переменные на интервале $[t_0, t_{\mathrm{к}}]$:

$$C(\mathbf{x}, \mathbf{u}, t) \leqslant 0. \tag{3.49}$$

Критерий качества системы J может быть модифицирован путем введения дополнительного слагаемого (штрафа)

$$\bar{J} = J + \mu \sum_{t_0}^{t_{\mathrm{к}}} [C(\mathbf{x},\mathbf{u},t)]^2 E(C)\,dt,$$

где μ и E имеют тот же смысл, что и в § 3.2.

Метод интегральных функций позволяет достаточно просто получить приближенное решение задачи синтеза оптимального управления при наличии ограничений (3.49).

Другой подход к данной задаче состоит в учете ограничений (3.51) при составлении гамильтониана системы

$$H = L + \psi^{\mathrm{T}}\mathbf{f} + \lambda C,$$

где $\lambda \geqslant 0$, если $C=0$, и $\lambda=0$, если $C<0$. В этом случае уравнения Эйлера—Лагранжа принимают вид

$$\dot{\psi}^{\mathrm{T}} = -H_{\mathrm{x}} = \begin{cases} -L_{\mathrm{x}} - \psi^{\mathrm{T}}\mathbf{f}_{\mathrm{x}} - \lambda C_{\mathrm{x}}, \text{ если } C = 0, \\ -L_{\mathrm{x}} - \psi^{\mathrm{T}}\mathbf{f}_{\mathrm{x}}, \text{ если } C < 0. \end{cases} \tag{3.50}$$

Оптимальное управление определяется из необходимого условия экстремума H

$$H_{\mathrm{u}} = L_{\mathrm{u}} + \psi^{\mathrm{T}}\mathbf{f}_{\mathrm{u}} + \lambda C_{\mathrm{u}} = 0. \tag{3.51}$$

При решении задач с ограничениями (3.49) может оказаться, что оптимальная траектория системы будет находиться как на границе допустимой области, так и внутри ее, а само оптимальное уравнение $\mathbf{u}(t)$ будет иметь разрывы в отдельных граничных точках [20].

ПРОГРАММА OPTIMA

Назначение: синтез оптимального управления непрерывной динамической системы

$$\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t),\ t_0\leqslant t\leqslant t_к,$$

при начальном условии $\mathbf{x}(t_0)=\mathbf{x}_0$ и ограничениях в конечной точке процесса $h(t_к, \mathbf{x}(t_к))=0$, $g_i(t_к,\mathbf{x}(t_к))=0$ $(i=1, ..., r)$. Первое из этих условий — основное, определяет время окончания процесса управления, остальные — дополнительные. В описанной здесь модели $\mathbf{x}$ — n-мерный вектор фазовых координат, $\mathbf{u}$ — m-мерный вектор управления, принадлежащий некоторому замкнутому подмножеству евклидова пространства $\mathbf{R}^m$, а $0\leqslant r<n$. Требуется найти допустимое управление, минимизирующее функционал качества $J=J(t_к, \mathbf{x}(t_к))$ (к этому виду с помощью введения дополнительных фазовых переменных приводится широкий класс функционалов, например интегральных). Допустимым управлением являются кусочно-непрерывные функций, определенные на $[t_0, t_к]$.

Входные параметры:

N — размерность вектора фазовых координат (вектора состояния системы);

M — размерность вектора управления;

R — число дополнительных краевых условий $(0\leqslant R<N)$;

T0 — начальный момент времени функционирования системы;

X0 — массив, состоящий из компонентов вектора начального состояния системы (размерности X0(N));

HP — постоянный интервал интегрирования систем дифференциальных уравнений;

LL — максимальное число шагов интегрирования (таким образом, длительность процесса управления удовлетворяет соотношению $t_к-t_0\leqslant LL* HP$);

R2 — максимальное число дробления числа α;

E0 — константа, характеризующая требуемую точность достижения минимума оптимизируемого критерия качества (следует задавать не меньше точности интегрирования системы, т. е. $LL* (HP)^3$);

C1 — коэффициент изменения коэффициентов при штрафных функциях $(1<C1\leqslant 2)$;

C2 — максимальное значение коэффициентов штрафов.

Выходные параметры:

A1 — массив, состоящий из значений вспомогательного критерия J' и коэффициентов штрафа a_i $(i=1, ..., r)$: A1(0) $=J'$, A1(I) $=a_i$, размерности A1(0:R). При обращении к программе элементы A1(I) (I=1, ..., R) заполняются начальными приближениями коэффициентов штрафа;

UT — двумерный массив, содержащий значения функций управления процессом, размерности UT(0: LL, M), первый элемент I(I<K) массива — номер шага по времени (I соответствует момент времени T=T0+I* HP), второй — компоненты вектора управляющих функций;

K — номер последнего шага интегрирования, соответствующего текущему сформированному массиву функций управления UT;

HK — последний нестандартный (т. е. соответствующий моменту окончания управления $0\leqslant HK\leqslant HP$) шаг интегрирования;

XT — двумерный массив, состоящий из таблицы значений оптимальной траектории системы, размерности XT(0:LL, N), первый элемент — номер шага во времени, второй — соответствующий компонент вектора состояния;

G — массив, в котором находится значение критерия качества (элемент G(0)), а также невязки в удовлетворении краевых условий (элементы G(I), I=1, ..., R), размерности G(0:R);

R1 — индикатор, указывающий на тип полученного результата: R1=1, если процесс итераций сошелся и оптимальное управление с заданной точностью синтезировано, R1=2, если процесс итераций сошелся, но не все заданные краевые условия удовлетворяются (в этом случае коэффициенты штрафа достигают максимальных значений), R1=3, если процесс итераций расходится.

Атрибуты всех параметров присвоены или по умолчанию, или (если требовалось употребление другого типа) как указано в тексте самой программы.

При обращении к программе всем перечисленным параметрам необходимо присвоить соответствующие значения. Массиву UT присваиваются значения начального приближения, по окончании работы программы в массиве UT находится таблица функций оптимального управления. Элемент A1(0) — оптимальное значение расширенного критерия качества, остальные элементы массива — конечные значения коэффициентов штрафа.

Вспомогательные процедуры:

FIN(T, X) — вычисление функции $h(t, x)$; FINX(T, X, Z1), FG(T, X, Z2), GX(G, X, Z3), FX(T, X, U, Z4), FP (T, X, P, U, Z5) — вычисление значений массивов Z1—Z5 соответственно;

HAM — вычисление функции управления, максимизирующей гамильтониан H системы.

Здесь

T — текущее время;

X — массив (X(N)) вектора состояния системы;

U — массив (U(M)) компонентов вектора управления;

P — массив (P(N)) сопряженных переменных;

Z1 — массив (Z1(N)) значений частных производных $\partial h(t, x)/dx_i$ функции h;

Z2 — массив (Z2(0:R)) значений функций J и g_i $(i=1, ..., R)$;

Z3 — массив (Z3(0:R, N)) частных производных по x_i $(i=1, ..., N)$ функций J и g_j $(j=1, ..., R)$;

Z4 — массив (Z4(N)) компонентов вектор-функции $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$;

Z5 — массив (Z5(N)) значений функций $-\partial H(t, \mathbf{x}, \mathbf{p}, \mathbf{u})/\partial x_j$ $(j=1, ..., N)$, где $H=\sum_{i=1}^{n} p_i f_i(t; \mathbf{x}, \mathbf{u})$.

Программа OPTIMA — модифицированный вариант программы OPTIMAL CONTROL, разработанной на языке Алгол [35]. В частности, в программе OPTIMA использованы средства языка ПЛ/1 по организации операций с массивами, что позволило значительно сократить ее объем по сравнению с программой OPTIMAL CONTROL.

В основу программы положены итерационные алгоритмы последовательных приближений в сочетании с алгоритмом штрафных функций (см. § 3.6). Для улучшения процесса сходимости к оптимальному управлению предусмотрена реализация следующего приема.

Пусть $\mathbf{u}^{(1)}(t)$, ..., $\mathbf{u}^{(l)}(t)$,...—последовательность приближений к функции оптимального управления; Φ — оператор вычисления нового приближения: $\mathbf{u}^{(l+1)}(t)=\Phi[\mathbf{u}^{(l)}(t)]$, причем $\mathbf{u}^{(l)}(t)$ построены на основе указанных алгоритмов.

Модификация итерационного процесса порождения последовательности $\mathbf{u}^{(i)}(t)$ [35]:

$$\mathbf{u}^{(i+1)}(t) = \begin{cases} \mathbf{u}^{(i)}(t) \text{ при } t_0 \leqslant t < t', \\ (1-\alpha)\,\mathbf{u}^{(i)}(t) + \alpha\Phi[\mathbf{u}^{(i)}(t)] \text{ при } t' \leqslant t \leqslant t_к. \end{cases}$$

Величины α, t' подбираются так, чтобы уменьшить значение критерия качества J. Число α изменяется от начального значения 1 до предельного 2^{-R2} $(R2 \geqslant 0)$ с помощью деления пополам. Область изменения переменной t' — весь отрезок $[t_0, t_к]$, причем начальное значение t' есть t_0. Если на очередной итерации при каких-либо α и t' не обеспечивается уменьшение критерия J, то сначала α уменьшается последовательным делением пополам. Если это не приводит к успеху, то значение α восстанавливается в первоначальное $\alpha=1$, интервал $[t', t_к]$ сокращается вдвое и снова подбирается нужное значение α. На последующих итерациях устанавливается $\alpha=1$, а отрезок $[t', t_к]$ либо сохраняется прежним, либо увеличивается по заданному правилу.

Для интегрирования исходной и сопряженной систем дифференциальных уравнений в программе OPTIMA применяется метод Эйлера с постоянным шагом. Момент окончания процесса $t_к$ определяется как приближенный корень уравнения $h(t_к, \mathbf{x}(t_к))=0$. Остальные краевые условия удовлетворяются с помощью применения метода штрафных функций. Коэффициенты штрафа (начальные значения которых задаются при обращении к программе) изменяются в ходе работы самой программы в зависимости от характера нарушения ограничений. Если задача синтеза оптимального управления содержит ограничения на фазовые переменные, то их можно учесть аналогично с помощью добавления функций штрафа к критерию J.

При практическом применении программы OPTIMA в процедуре HAM необходимо обеспечить обращение к процедуре, осуществляющей поиск максимума гамильтониана H системы. В качестве таких процедур можно предположить программы нелинейной оптимизации (см. гл. 6).

Пример. Пусть дана система

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = -x_1 + u_1 + u_2,$$
$$\dot{x}_3 = -u_2^2$$

с начальными условиями $t_0=\pi/2$, $x_{10}=1{,}5$, $x_{20}=1+\pi/4$, $x_{30}=3\pi/4$. Критерий качества принят в виде $J=-x_1^2(t_к)/2=(2+\pi/2)x_3(t_к)$. Задавалось ограничение $x_2(t_к)=0$. Требуется найти оптимальное управление $\mathbf{u}=(u_1, u_2)$, минимизирующее функционал J.

Получено оптимальное значение $J^{opt}=-25{,}49$ на ЭВМ ЕС-1050.

```
OPTIMA: PROCEDURE(M,R,N,K,LL,R1,R2,HP,HK,T0,A1,
                  E0,C2,C1,E,XT,X0,UT,G,FIN,FX,
                  FP,FG,HAM,FINX,GX);
    DECLARE
    (FIN,FX,FP,FINX,HAM,FG,GX) ENTRY,
    E(*),XT(*),X0(*),A1(*),UT(*,*),G(*);
    DECLARE
       (R,R1,R2,B1,B2)
       FIXED BINARY,
```

```
          (JJ,JJ1)
          FLOAT DECIMAL, B FLOAT DECIMAL(9);
        BEGIN;
        DECLARE
        (X,X1,Z,Z1,P) (N),
        U(M),GT1(0:R),
        GRAD(0:R,N),UT1(0:LL,M);
        Q1=2**R2; Q2=Q1-1; Q0=1.5/Q1; R1=0;
        DO I=1 TO R;
         IF A1(I) > C2 THEN A1(I)=C2;
        END;
         L3=0; L1=0; L=0;
         L2=LL+1;  B2=1;
        XT(0.*)=X0(*);
MET2:B1=1;  S,HK,H=HP;
        X(*)=XT(L3,*);
        T=T0+L3*H; A=FIN(T,X);  K=L3+1;
        IF L = 0 THEN
         U(*)=UT(0,*);
        ELSE
         U(*)=UT1(L3,*);
MET3:IF K = L2 THEN
        DO;
         R1=3; GO TO MET5;
        END;
        CALL FX(T,X,U,Z); T=T+H;
         X1=X+H*Z;
         U(*)=UT(K,*);
         CALL FX(T,X1,U,Z1);
         X=X+0.5*H*(Z+Z1);
         B=FIN(T,X);
         IF ABS(B) > E(0) THEN
         DO;
          IF B > 0 THEN
           DO;
            IF B1 = 1 THEN
            DO;
             XT(K,*)=X(*);
            T1=T; A=B; K=K+1;
            GO TO MET3;
            END;
            S=0.5*S; HK=HK+S;
           END;
           ELSE
           DO;
            B1=0; S=0.5*S; HK=HK-S;
           END;
           H=HK; J=K-1; T=T1;
           X(*)=XT(J,*);
           U(*)=UT(J,*);
           GO TO MET3;
           END;
          XT(K,*)=X(*);
          CALL FG(T,X,G);
          IF R1 = 3 THEN
          DO;
           L,L1,L3=0; R1=-1; GO TO MET4;
          END;
MET1:     JJ=G(0);
          DO I=1 TO R;
          JJ=JJ+G(I)*A1(I)*G(I);
          END;
```

```
          IF R1=2 THEN GO TO MET5;
          IF R1=1 THEN
          DO;
           R1=0; GO TO MET6;
          END;
          IF B2 = 1 THEN
          DO;
           B2=0; JJ1=JJ+2*E0;
          END;
          IF JJ < JJ1 THEN
          DO;
           DO I=L TO LL;
            UT1(I,*)=UT(I,*);
           END;
           IF JJ > JJ1-E0 THEN
           DO;
            IF L = 0 & Q > 0.9 THEN
MET6:        DO;
              B1,B2=1;
              DO I=1 TO R;
               IF ABS(G(I)) > E(I) THEN
               DO;
                B=C1*A1(I); B1=0;
                IF B < C2 THEN
                DO;
                 A1(I)=B; B2=0;
                END;
               END;
              END;
              IF B2 = 1 THEN
              DO;
               IF B1 = 1 THEN R1=0; ELSE R1=1;
               GO TO MET5;
              END;
              B2=1; GO TO MET1;
             END;
             IF Q > 0.9 THEN
             DO;
              IF L=L1 THEN L,L1=0;
              ELSE L=(L+L1)*0.5-0.1;
             END;
            END;
            IF L=0 THEN L3=0; ELSE L3=L-1;
            JJ1=JJ; K1=K; R1=0;
MET4:       Q=1;
            CALL GX(T,X,GRAD);
            CALL FINX(T,X,Z);
            S=T-HK; J=K-1;
            X(*)=XT(J,*);
            CALL FG(S,X,GT1);
            S=0;
           DO I=0 TO R;
            GT1(I)=GT1(I)-G(I);
           END;
           DO I=1 TO R;
            S=S+A1(I)*GT1(I)*G(I);
           END;
           S=(S*2+GT1(0))/A;
           DO J=1 TO N;
            B=0;
            DO I=1 TO R;
             B=B+A1(I)*GRAD(I,J)*G(I);
```

```
 END;
 F(J)=S*Z(J)-GRAD(0,J)-2*B;
 X(J)=XT(K,J);
END;
U(*)=UT1(K,*);
T=T0+(K-1)*HP+HK; H=HK;
DO J=K TO 1 BY -1;
 CALL FP(T,X,P,U,Z);
 CALL HAM(T,X,P,U);
 UT(J,*)=U(*);
 X1=P-H*Z;
 T=T-H; K2=J-1;
 X(*)=XT(K2,*);
 U(*)=UT1(K2,*);
 CALL FP(T,X,X1,U,Z1);
 P=P-0.5*H*(Z+Z1);
 H=HP;
END;
CALL HAM(T,X,P,U);
UT(0,*)=U(*);
DO J=1 TO M;
 B=UT(K,J);
 DO I=K+1 TO LL;
  UT(I,J)=B;
 END;
END;
END; ELSE
DO;
 IF JJ> JJ1+E0 ! L  = 0 ! Q < 0.9 THEN
 DO;
  IF Q < Q0 THEN
  DO;
   IF L < K1 THEN
   DO;
    Q=1; L1=L;
    IF L1 = 0 THEN L3=0; ELSE L3=L1-1;
    L=(L+K1)*0.5+0.1;
    DO I=L1 TO L-1;
     UT(I,*)=UT1(I,*);
    END;
    DO I=L TO LL;
     UT(I,*)=Q1*UT(I,*)-Q2*UT1(I,*);
    END;
   END;
   ELSE
   DO;
    UT(K1,*)=UT1(K1,*);
    L3=K1-1; R1=R1+3;
   END;
  END;
 ELSE
 DO;
  Q=0.5*Q;
  IF L = 0 THEN L3=0 ELSE L3=L-1;
  DO I=L TO LL;
   UT(I,*)=(UT(I,*0+UT1(I,*))*0.5;
  END;
 END;
END;
ELSE
DO;
 R1=1;
```

```
          UT=UT1;
          END;
         END;
         GO TO MET2;
   MET5:A1(0)=JJ;
        END;
        END OPTIMA;

FIN:    PROCEDURE(T,X);
        DECLARE X(3);
        FF=6.28319-T;
        RETURN(FF);
        END FIN;
FX:     PROCEDURE(T,X,U,Z);
        DECLARE X(3),U(2),Z(3);
        Z(1)=X(2); Z(2)=-X(1)+U(1)+U(2);
        Z(3)=-U(2)*U(2);
        END FX;
FP:     PROCEDURE(T,X,P,U,Z);
        DECLARE X(3),U(2),Z(3),P(3);
        Z(3)=0; Z(1)=P(2); Z(2)=-P(1);
        END FP;
FINX:   PROCEDURE(T,X,GH);
        DECLARE X(3),GH(3);
        GH=0;
        END FINX;
HAM:    PROCEDURE(T,X,P,U);
        DECLARE X(3),P(3),U(2);
        U(1)=SIGN(P(2)); U(2)=P(2)/(2*P(3));
        END HAM;
FG:     PROCEDURE(T,X,G);
        DECLARE X(3),G(0:1);
        G(0)=-X(1)**2/2-3.5708*X(30; G(1)=X(2);
        END FG;
GX:     PROCEDURE(T,X,GG);
        DECLARE X(3),GG(0:1,3);
        GG(0,1)=-X(1); GG(0,3)=-3.5708;
        GG(0,2)=1;  GG(0,2),GG(1,1),GG(1,3)=0;
        END GX;
TEST :  PROCEDURE OPTIONS(MAIN);
        DECLARE R FIXED BINARY;
        M=2; R=1; N=3; LL=100;
        BEGIN;
         DECLARE XT(0:LL,1:N),UT(0:LL,1:M),
          X0(N),(G,E,A1)(0:R),
          (R1,R2) FIXED BINARY;
         X0(1)=1.5: X0(2)=1.785398; X0(3)=2.356194;
         E(0)=0.001; E(1)=0.05; A1(1)=1;
         DO K=0 TO LL;
         UT(K,1),UT(K,2)=1;
         END;
         R2=5; HP=0.0471239; T0=1.5708;
         E0=0.001; C2=40; C1=2;
         CALL OPTIMA(M,R,N,K,LL,R1,
                     R2,HP,HK,T0,A1,
                     E0,C2,C1,E,XT,
                     X0,UT,G,FIN,
                     FX,FP,FG,HAM,FINX,GX);
        END;
END TEST;
```

3.6. Синтез оптимального управления методами математического программирования

Подавляющее большинство задач синтеза оптимального управления реальных динамических систем решается с помощью быстродействующих ЭВМ. Это приводит к неизбежному сведению задач оптимизации непрерывных динамических систем к дискретным. Таким образом, появляется возможность решить исходную задачу, привлекая хорошо разработанные методы теории математического программирования.

Пусть нелинейная динамическая система описывается разностными уравнениями

$$\mathbf{x}(k+1) = f^k[\mathbf{x}(k)\,\mathbf{u}(k)], \tag{3.52}$$

где $\mathbf{x}(0)$ — начальный вектор состояния. Пусть задана система (вообще говоря, нелинейных) скалярных ограничений $C^k[\mathbf{x}(k), \mathbf{u}(k)] \leqslant 0$ (где $k=0, \ldots, N-1$, а N фиксировано). Как и прежде, вектор $\mathbf{x}$ — n-мерный, а $\mathbf{u}$ — m-мерный. Пусть эффективность функционирования системы (3.52) описывает критерий качества

$$J = L^N[\mathbf{x}(N)] + \sum_{k=0}^{N-1} L^k[\mathbf{x}(k), \mathbf{u}(k)], \tag{3.53}$$

который следует минимизировать.

Введем $\mathbf{y}$ — вектор оптимизируемых параметров размерности $N(n+m)$:

$$\mathbf{y}^{\mathrm{T}} = \{\mathbf{x}(1), \ldots, \mathbf{x}(N); \mathbf{u}(0), \ldots, \mathbf{u}(N-1)\}. \tag{3.54}$$

Пусть также $\mathbf{G}$ — $(Nn+N)$-мерный вектор функций, входящих в систему ограничений

$$\mathbf{G}(\mathbf{y}) = \begin{bmatrix} \mathbf{x}(1) - \mathbf{f}^0[\mathbf{x}(0), \mathbf{u}(0)] \\ \cdots\cdots\cdots\cdots \\ \mathbf{x}(N) - \mathbf{f}^{N-1}[\mathbf{x}(N-1), \mathbf{u}(N-1)] \\ C^0[\mathbf{x}(0), \mathbf{u}(0)] \\ \cdots\cdots\cdots\cdots \\ C^{N-1}[\mathbf{x}(N-1), \mathbf{u}(N-1)] \end{bmatrix}.$$

Таким образом, задача оптимального управления системой (3.52) целиком свелась к задаче минимизации функционала $F(\mathbf{y}) = J$ на множестве допустимых значений $\{\mathbf{G}(y) \leqslant 0\}$:

$$F(\mathbf{y}) \to \min, \quad \mathbf{y} \in \mathbf{D} = \{\mathbf{y} : \mathbf{G}(\mathbf{y}) \leqslant 0\}, \tag{3.55}$$

т. е. к стандартной задаче математического программирования с ограничениями типа неравенств. Аналогично могут быть сведены к задачам математического программирования и другие задачи оптимального управления (с заданными в отдельные моменты времени значениями фазовых координат и т. д.).

Для решения оптимизационной задачи (3.55) можно предложить достаточно большое количество эффективных алгоритмов, в частности алгоритмы и программы,

представленные в гл. 6, 7. При выборе алгоритма оптимизации исследователь должен максимально учитывать специфику задачи. Например, при условии линейности критерия F и ограничений G наиболее эффективно применение симплекс-метода (при отсутствии гладкости у критерия и ограничений алгоритма группы поисковых методов и т. д.).

Стандартизация задачи синтеза оптимального управления в виде задачи математического программирования (3.55) позволяет разработчику системы в процессе исследований формировать в виде программ для ЭВМ лишь процедуры, обеспечивающие расчет функционала качества и функционалов, комплектующих систему ограничений.

Глава 4.

Оптимальное управление с обратной связью

Пусть целью управления является перевод динамической системы

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{4.1}$$

из некоторого начального состояния $\mathbf{x}(t_0)$ в конечное, определяемое терминальными ограничениями

$$\mathbf{G}[\mathbf{x}(t_к), t_к] = 0, \tag{4.2}$$

где $t_к$ — время окончания процесса управления, которое либо фиксировано, либо не задано явно и определяется выполнением некоторых условий. При этом система должна удовлетворять некоторым свойствам оптимальности.

Управление, определяемое как функция данной текущей точки

$$\mathbf{u}(t) = \mathbf{u}[\mathbf{x}(t), t], \tag{4.3}$$

называется *управлением с обратной связью.*

Можно выделить следующий основной класс задач, решаемых в теории управления с обратной связью. Это задачи управления возмущенным движением системы относительно невозмущенного программного; при этом целью управления является удержание системы в достаточной близости от программной траектории (в идеале — точно на ней). Естественно, что реализовать управление возмущенным движением практически можно лишь при наличии информации об отклонениях системы от программной траектории.

В этой главе рассмотрены алгоритмы синтеза оптимальных регуляторов для линейных и нелинейных систем, а также один из возможных алгоритмов решения задачи аналитического конструирования цифровых регуляторов, т. е. синтеза программ работы вычислительных устройств, призванных осуществлять управление возмущенным движением системы. Включение в контур управления системы ЭВМ приводит к необходимости применения преобразователей непрерывных сигналов в дискретные и, наоборот, дискретных управляющих сигналов на выходе ЭВМ в непрерывные. Одна из возможных схем системы управления с цифровым регулятором приведена на рис. 4.1.

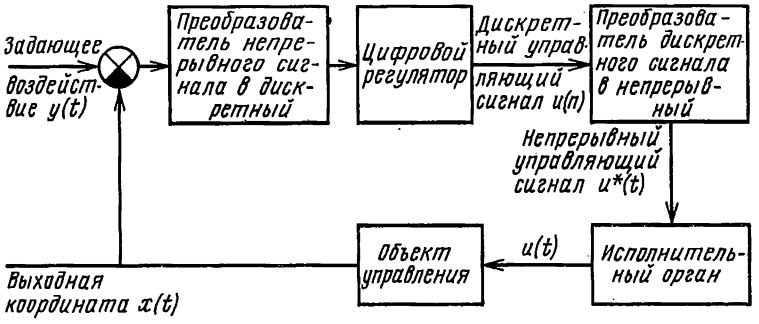


Рис. 4.1. Схема системы управления с цифровым регулятором

Линейные системы с обратной связью занимают одно из центральных мест в современной теории и практике систем управления. Это обусловлено тем, что, во-первых, многие конкретные объекты управления с достаточной точностью описываются линейными динамическими моделями, во-вторых, задачи синтеза оптимального регулятора нелинейных систем могут быть сведены к линейным задачам.

4.1. Синтез оптимального регулятора линейной системы

Пусть динамическая система (4.1) описывается линейными дифференциальными уравнениями

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{B}(t)\,\mathbf{u}(t). \tag{4.4}$$

Здесь $\mathbf{x}$ — n-мерный вектор состояния; $\mathbf{u}$ — m-мерный вектор управления; $\mathbf{A}(t)$ и $\mathbf{B}(t)$ — матрицы размера $n \times n$ и $n \times m$ соответственно. Пусть необходимо перевести систему (4.4) из заданного начального состояния $\mathbf{x}(t_0)$ в некоторое конечное $\mathbf{x}(t_к)$ ($t_к$ — фиксированный момент окончания процесса управления), причем в течение отрезка времени $[t_0, t_к]$ фазовые координаты $\mathbf{x}(t)$ и управление $\mathbf{u}(t)$ не должны выходить за допустимые пределы.

Один из наиболее простых и эффективных подходов приближенного решения этой задачи — использование интегральных штрафных функций. При этом задача сводится к нахождению такого управления $\mathbf{u}(t)$, которое минимизирует критерий качества I, определенный для квадратичных форм от вектора конечного состояния, а также векторов состояния и управления:

$$I = \frac{1}{2}\left[(\mathbf{x} - \mathbf{x}_ж)^{\mathrm{T}}\,\mathbf{S}_к(\mathbf{x} - \mathbf{x}_ж)\right]_{t=t_к} + \frac{1}{2}\int_{t_0}^{t_к}(\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u})\,dt, \tag{4.5}$$

где $\mathbf{S}_к$, $\mathbf{Q}(t)$ и $\mathbf{R}(t)$ — неотрицательно и положительно определенные матрицы; множитель $1/2$ введен для удобства. В дальнейшем без ограничения общности можно считать, что $\mathbf{x}(t_к) = 0$.

Соотношения для оптимального управления системы (4.4) с критерием (4.5) можно вывести аналогично изложенному в гл. 3:

$$\partial H/\partial \mathbf{u} = 0, \tag{4.6}$$

$$\dot{\psi}^{\mathrm{T}} = -\partial H/\partial \mathbf{x}, \tag{4.7}$$

$$\psi(t_{\text{к}}) = \mathbf{S}_{\text{к}}\mathbf{x}(t_{\text{к}}). \tag{4.8}$$

Здесь H — гамильтониан системы:

$$H = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u} + \psi^{\mathrm{T}}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}), \tag{4.9}$$

подставляя выражение (4.9) в соотношения (4.6), (4.7), получаем:

$$\mathbf{u}(t) = -\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\psi(t), \tag{4.10}$$

где ψ определяется из решения линейной двухточечной краевой задачи

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\psi} \end{bmatrix} = \varphi \begin{bmatrix} \mathbf{x} \\ \psi \end{bmatrix}, \tag{4.11}$$

в которой начальное $\mathbf{x}(t_0)$ и конечное $\psi(t_{\text{к}}) = \mathbf{S}_{\text{к}}\mathbf{x}(t_{\text{к}})$ значения заданы;

$$\varphi = \begin{bmatrix} \mathbf{A} & -\mathbf{G}\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}} \\ -\mathbf{Q} & -\mathbf{A}^{\mathrm{T}} \end{bmatrix}. \tag{4.12}$$

Для решения системы (4.11) также можно воспользоваться методами, изложенными в гл. 3, т. е. находить переходные матрицы (матрицы фундаментальных решений) системы (4.11)

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(t_{\text{к}}); \tag{4.13}$$

$$\psi(t) = \Lambda(t)\mathbf{x}(t_{\text{к}}). \tag{4.14}$$

Из соотношений (4.13), (4.14) имеем

$$\mathbf{x}(t) = \Phi(t)\left[\Phi(t_0)^{-1}\right]\mathbf{x}(t_0);$$

$$\psi(t) = \Lambda(t)\left[\Phi(t_0)^{-1}\right]\mathbf{x}(t_0).$$

Поэтому формула для оптимального управления с обратной связью будет

$$\mathbf{u}(t) = -\mathbf{K}(t, t_0)\,\mathbf{x}(t_0),$$
$$\mathbf{K}(t, t_0) = [\mathbf{R}(t)]^{-1}\mathbf{B}^{\mathrm{T}}(t)\Lambda(t)\left[\Phi(t_0)\right]^{-1}, \tag{4.15}$$

где $\mathbf{K}(t, t_0)$ — матрица коэффициентов усиления.

Векторное уравнение (4.15) представляет собой уравнение дискретного управления с обратной связью, так как момент времени t_0 можно трактовать как момент времени предыдущей дискретизации траектории системы (4.4). Если $t_0 \to t$ (что соответствует непрерывному поступлению информации о состоянии системы), то будем иметь следующий закон непрерывного управления с обратной связью:

$$\mathbf{u}(t) = -\mathbf{K}(t)\,\mathbf{x}(t),\ \mathbf{K}(t) = [\mathbf{R}(t)]^{-1}\,\mathbf{B}^{\mathrm{T}}(t)\,\Lambda(t)\,[\Phi(t)]^{-1}. \qquad (4.16)$$

Так как элементы матриц $\Phi(t)$ и $\Lambda(t)$ — величины различных порядков, то при применении на практике методов построения переходных матриц $\Phi(t)$ и $\Lambda(t)$ возможна ощутимая потеря точности решения системы (4.1). В этом случае достаточно эффективно использование метода прогонки, суть которого состоит в следующем.

Вводится матрица $\mathbf{S}(t)=\Lambda(t)\,[\Phi(t)]^{-1}$, удовлетворяющая следующему векторному дифференциальному уравнению:

$$\dot{\mathbf{S}}(t) = -\mathbf{S}(t)\,\mathbf{A}(t) - \mathbf{A}^{\mathrm{T}}(t)\,\mathbf{S}(t) + \mathbf{SBR}^{-1}\,\mathbf{B}^{\mathrm{T}}\mathbf{S} - \mathbf{Q},\ \mathbf{S}(t_{\text{к}}) = \mathbf{S}_{\text{к}} \qquad (4.17)$$

(уравнения вида (4.17) называют *матричными уравнениями Риккати*). Далее уравнение (4.17) решается от терминального момента $t_{\text{к}}$ до начального t_0. Из соотношения

$$\psi(t_0) = \mathbf{S}(t_0)\,\mathbf{x}(t_0)$$

находится вектор $\psi(t_0)$. Так как начальные условия $\mathbf{x}(t_0)$ и $\psi(t_0)$ уже известны, то задача (4.17) превращается в известную задачу Коши, и решение ее достаточно легко находится интегрированием системы (4.13) на отрезке времени $[t_0,\ t_{\text{к}}]$. Само же оптимальное управление с обратной связью вычисляется по известной матрице $\mathbf{S}(t)$ с помощью следующих очевидных модификаций уравнений (4.16):

$$\mathbf{u}(t) = -\mathbf{K}(t)\,\mathbf{x}(t),\ \mathbf{K}(t) = [\mathbf{R}(t)]^{-1}\,\mathbf{B}^{\mathrm{T}}(t)\,\mathbf{S}(t). \qquad (4.18)$$

4.2. Аналитическое конструирование цифровых регуляторов

Предположим, что возмущенное движение системы описывается совокупностью разностных уравнений [31]

$$\sum_{i=0}^{n} a_i x(k+i) = \sum_{j=0}^{n-1} b_j u(k+j),\quad k = 0,1,\ldots, \qquad (4.19)$$

где a_i, b_j — заданные коэффициенты (без ограничения общности можно считать, что $a_n=1$); x — выходная координата системы; u — управляющее воздействие (управляющий сигнал).

Представим уравнение (4.19) с помощью системы разностных уравнений первого порядка:

$$x_i(k+1) = x_i(k) + f_i\,u(k),\quad i = 1,\ldots,n-1,$$

$$x_n(k+1) = -a_{n-1}\,x_n(k) - a_{n-2}\,x_{n-1}(k) - \ldots - a_0\,x_1(k) + f_n\,u(k). \qquad (4.20)$$

Коэффициенты уравнений (4.19) и (4.20) связаны соотношениями

$$f_i = b_{n-i} - \sum_{j=1}^{i-1} a_{n-j} f_{i-j},\quad i = 1,\ldots,n.$$

Переменные $x_i(k)$ ($i=1,\ \ldots,\ n$) в (4.20) — координаты состояния разомкнутой системы.

Считаем заданным на траекториях движения системы функционал в виде обобщенной квадратической оценки переходных процессов

$$J(u) = \sum_{k=0}^{\infty} \left[\sum_{i=1}^{h} c_i x_i^2(k) + cu^2(k) \right],$$

а также координаты состояния $x_i(0) = x_i^0$ $(i = 1, \ldots, n)$ и управляющий сигнал $u(0) = u^0$ в начальный момент времени.

Задача аналитического конструирования состоит в синтезе аналитического выражения для управления с обратной связью $u(k) = u[x_1(k), \ldots, x_n(k)]$, которое обеспечивает:

перевод системы из начального состояния $x_i(0)$ $(i = 1, \ldots, n)$ в конечное нулевое: $x_i(\infty) = 0$ $(i = 1, \ldots, n)$, $u(\infty) = 0$;

достижение минимума функционала $J(u)$.

Методика решения сформулированной таким образом задачи следующая. Введем вспомогательный функционал

$$\bar{J}(u) = \sum_{k=0}^{\infty} L(k),$$

где $L(k)$ $(k = 0, 1, \ldots)$ — функция Лагранжа:

$$L(k) = \sum_{i=1}^{n} \left[c_i x_i^2(k) + cu^2(k) \right] + \sum_{j=1}^{n-1} \psi_j(k) \left[x_j(k+1) - x_{j+1}(k) - f_j u(k) \right] + \psi_n(k) \left[x_n(k+1) + a_{n-1} x_n(k) + \ldots + a_0 x_1(k) - f_n u(k) \right]$$

(функциональные множители $\psi_j(k)$ называют множителями Лагранжа).

Решение исходной задачи находим из системы уравнений

$$2c_1 x_1(k) + a_0 \psi_n(k) + \psi_1(k-1) = 0,$$

$$2c_i x_i(k) - \psi_{i-1}(k) + a_{k-1} \psi_n(k) - \psi_i(k-1) = 0,$$

$$2cu(k) - \sum_{j=1}^{n} f_j \psi_j(k) = 0, \quad i = 2,3,\ldots, n \tag{4.21}$$

к которым нужно присоединить еще и уравнения (4.20).

Оптимальное управление определяется из последнего уравнения системы (4.21):

$$u(k) = \frac{1}{2c} \sum_{j=1}^{n} f_j \psi_j(k).$$

Управляющая функция будет получена в окончательном виде, если выразить множители Лагранжа ψ_j через координаты системы $x_1(k), \ldots, x_n(k)$. Решая однородную самосопряженную систему разностных уравнений (4.20), (4.21) с граничными

условиями $\lim\limits_{k\to\infty} x_j(k)=0$ $(j=1, \ldots, n)$ и $\lim\limits_{k\to\infty} u(k)=0$, получаем общее решение [33]:

$$x_i(k) = \gamma_{i,1} r_1 z_1^k + \ldots + \gamma_{i,n} r_n z_n^k; \tag{4.22}$$

$$\psi_i(k) = \gamma_{n+i,1} r_1 z_1^k + \ldots + \gamma_{n+i,n} r_n z_n^k. \tag{4.23}$$

Здесь $\gamma_{i,j}$ и r_k — постоянные, определяемые по коэффициентам системы и из граничных условий на левом конце экстремалей; z_j $(j=1, \ldots, n)$ — корни характеристического уравнения системы

$$\Delta(z) = \begin{vmatrix} z & -1 & 0 & \ldots & 0 & -\frac{f_1^2}{2c} & \ldots & -\frac{f_1 f_n}{2c} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_0 & a_1 & a_2 & \ldots & z+a_{n-1} & -\frac{f_1 f_n}{2c} & \ldots & -\frac{f_n^2}{2c} \\ 2c_1 & 0 & 0 & \ldots & 0 & z^{-1} & \ldots & a_0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 2c_n & 0 & \ldots & z^{-1}+a_{n-1} \end{vmatrix} = 0.$$

Коэффициенты $\gamma_{i,j}$ $(i, j=1, \ldots, 2n)$ можно определить из матрицы характеристического уравнения — величина $\gamma_{i,j}$ представляет собой значение минора $\Delta(z)$, относящегося к i-му столбцу и j-й строке. Зависимости, отражающие связь множителей ψ_i с координатами системы x_j, можно найти, решив n уравнений (4.22) относительно величин $r_i z_i^k$:

$$r_i z_i^k = (-1)^{i+1} \frac{D_{i1}}{D} x_1(k) + (-1)^{i+2} \frac{D_{i2}}{D} x_2(k) + \ldots + (-1)^{n+k} \frac{D_{in}}{D} x_n(k).$$

Здесь D — определитель системы (4.22) (отличный от нуля, поскольку решения $x_1, \ldots, x_n$ линейно независимы); D_{ij} — относящийся к i-му столбцу и j-й строке минор определителя D. Используя уравнения (4.23), после необходимых преобразований получаем:

$$\psi_i(k) = \rho_{i1} x_1(k) + \ldots + \rho_{in} x_n(k), \quad i = 1, \ldots, n.$$

Величины ρ_{ij} определяются по коэффициентам системы (4.22), (4.23). Следовательно, уравнение для оптимального регулятора

$$u(k) = \sum_{i=1}^{n} S_i x_i(k), \tag{4.24}$$

где коэффициенты s_i $(i=1, \ldots, n)$ вычисляются на основании последовательного применения вышеизложенной схемы.

При выводе (4.22) и (4.23) предполагалось, что корни характеристического уравнения z_i вещественны и различны. Общий случай, характеризующийся наличием комплексных и кратных корней, в целом аналогичен описанному.

Таким образом, уравнение регулятора (4.24), обеспечивающее минимум обобщенной квадратической оценки переходных процессов в замкнутой системе,

представляет собой линейную комбинацию координат $x_1, \ldots, x_n$. Однако в реальной системе можно измерить лишь выходную координату $x(k)$ (и, быть может, некоторое число ее производных). Следовательно, чтобы использовать уравнение оптимального управления (4.24), требуется пересчитать значения неизмеряемых координат $x_2, \ldots, x_n$ через значения выходной координаты $x = x_1$. Чтобы получить соответствующие значения координат $x_2, \ldots, x_n$, необходимо накопление в течение $n-1$ шагов (тактов) работы системы информации о значениях последовательностей $x(0), \ldots, x(n-2)$ и $u(0), \ldots, u(n-2)$.

Предположим, что указанные последовательности получены и занесены в память ЭВМ. Процесс вычисления в момент времени $n-1$ всех координат $x_1(n-1), \ldots, x_n(n-1)$ можно описать с помощью таблицы:

$$
\begin{array}{|ccccccc|}
x_1(0) & x_1(1) & x_1(2) & \ldots & x_1(n-3) & x_1(n-2) & x_1(n-1) \\
x_2(0) & x_2(1) & x_2(2) & \ldots & x_2(n-3) & x_2(n-2) & x_2(n-1) \\
x_3(0) & x_3(1) & x_3(2) & \ldots & x_3(n-3) & x_3(n-2) & x_3(n-1) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_{n-2}(0) & x_{n-2}(1) & x_{n-2}(2) & \ldots & x_{n-2}(n-3) & x_{n-2}(n-2) & x_{n-2}(n-1) \\
x_{n-1}(0) & x_{n-1}(1) & x_{n-1}(2) & \ldots & x_{n-1}(n-3) & x_{n-1}(n-2) & x_{n-1}(n-1) \\
x_n(0) & x_n(1) & x_n(2) & \ldots & x_n(n-3) & x_n(n-2) & x_n(n-1)
\end{array}
$$

где в 1-й строке значения выходной координаты системы. В процессе постепенного заполнения таблицы вычисляются неизвестные элементы последнего столбца, которые и являются искомыми значениями неизмеряемых координат $x_1(n-1), \ldots, x_n(n-1)$. Порядок последовательных вычислений обозначен в таблице стрелками.

Для получения необходимых формул вернемся к системе (4.20) и рассмотрим ее последнее уравнение

$$x_n(k+1) = -a_{n-1}x_n(k) - a_{n-2}x_{n-1}(k) - \ldots - a_0 x_1(k) + f_n u(k). \qquad (4.25)$$

Если в нем положить $k=0$, то имеем

$$x_n(1) = -a_{n-1}x_n(0) - \ldots - a_0 x_1(0) + f_n u(0),$$

причем все элементы правой части известны. Если в предпоследнем уравнении (4.20) предположить $k=1$, то

$$x_{n-1}(2) = x_n(1) + f_{n-1}u(1),$$

где также все элементы правой части уже определены. Следовательно, получены все элементы 2-го столбца таблицы. Продолжим процесс вычислений: подставив в (4.25) значение $k=1$, найдем $x_n(2)$, ... и, наконец, $x_2(n-1), \ldots, x_n(n-1)$, на основании которых и формируется закон управления

$$u(n-1)=\sum_{i=1}^{n} s_i x_i(n-1).$$

При такой организации процесса регулирования система будет неуправляемой в течение $n-1$ тактов (при этом в качестве $u(k)$ обычно используют естественные шумы системы или другие доступные для восприятия чувствительных элементов системы сигналы). Описанный процесс определения неизмеряемых координат $x_2(n-1)$, ..., $x_n(n-1)$ используется только один раз. С момента времени $t=n$ система переходит в стационарный режим, и значения обобщенных координат вычисляются непосредственно с помощью соотношений (4.26).

Процедура вычисления неизмеряемых координат системы аналогична в случае использования наблюдаемой координаты $y(k)$ и ее производной $\dot{y}(k)$ [31].

ПРОГРАММА DRG

Назначение: решение задачи аналитического конструирования цифрового регулятора с заданным критерием оптимальности обеспечивает вычисление управляющего сигнала $u(k)=\sum_{i=1}^{n} S_i x_i(k)$ в каждый момент времени $k=0,1,...$ при использовании только выходной координаты $x(k)=x_1(k)$.

Программа разработана на языке Паскаль применительно к СМ ЭВМ, но может быть легко перенесена и на ЭВМ другого типа, оснащенные соответствующими компиляторами. Это позволяет применять программу DRG для управления различными технологическими процессами.

Параметры:

K — порядок уравнения (4.19);

MAX — максимальное число итераций для определения корней характеристического уравнения;

BET — значение коэффициента c;

EPS1, EPS2, EPS3, ETA — точность определения корней характеристического уравнения (EPS1 — относительный критерий сходимости итерации, EPS2 — абсолютный критерий сходимости по значениям функции, EPS3 — точность определения корней, ETA — критерий точности при наличии кратных корней);

A — массив размера K+1, содержащий коэффициенты a_i уравнения (4.19);

B — массив размера 2K+1, содержащий коэффициенты b_i уравнения (4.19);

AL — коэффициенты целевого функционала (размерность K);

F — коэффициенты перехода от исходного уравнения к системе разностных (размерность K);

X — массив корней характеристического уравнения (размерность K);

XM — характеристическая матрица размера K2*K2, K2=K*2;

GAM, GAM1 — значения миноров определителя, относящихся к любой строке, для которой они не все равны нулю, и к некоторому столбцу (размерность GAM (2K, K), GAM1 (K, K));

T — массив искомых значений коэффициентов S_i (размерности K).

Вспомогательные подпрограммы:

ZEROS — определение вещественных корней произвольной функции методом Мюллера [8];

FUNC — вычисление определителя характеристической матрицы;
ALD — формирование алгебраического дополнения характеристической матрицы;
DET — вычисление определителя произвольной квадратной матрицы (см. § 1.1).

Пример. Решение задачи синтеза дискретного регулятора системы с коэффициентами: $A(1) = -0{,}5$, $A(2) = 0{,}5$, $A(3) = 1$; $B(1) = B(2) = B(4) = B(5) = 0{,}5$, $B(3) = 0$. Остальные параметры задавались следующим образом: $K = 2$, $MAX = 20$, $BET = 1$, $AL(1) = AL(2) = 0{,}5$; $EPS1 = EPS2 = EPS3 = 0{,}01$, $ETA = 0{,}001$; $X(1) = X(2) = 1$, $X(3) = X(4) = 1$.

Расчеты проводились на ЭВМ СМ-4 в течение 5 с. Получено следующее решение: $T(1) = -1{,}37$; $T(2) = -1.36$.

```
PROCEDURE DRG(K,MAX: INTEGER,BET,EPS1,EPS2,EPS3,
ETA:REAL;A,B,AL,F,X:ATIP;
XM,GAM,GAM1:XMTIP;
VAR T:ATIP);
VAR
I,J,K1,M,K2,K21,K211,L,MM,INDEX,MU:INTEGER,
H,AD,RJM,D,R,XMI1:REAL;
LABEL 7;
PROCEDURE ZEROS(KK,K1,K2,M,MAX:INTEGER;
EPS1,EPS2,EPS3,ETA:REAL;A:ATIP;XM:XMTIP;
VAR C:ATIP);EXTERNAL;
PROCEDURE DET(N:INTEGER;A:XMTIP;
VAR AD:REAL);EXTERNAL;
PROCEDURE ALD(K2,I,J:INTEGER;
XM:XMTIP;VAR A:REAL);EXTERNAL;
BEGIN
K1:=K-1;
M:=K+1;
K2:=K*2;
K21:=K2-1;
K211:=K2+1;
FOR I:=1 TO K2 DO
FOR J:=1 TO K2 DO
XM[I,J]:=0.0;
F[1]:=B[K];
FOR I:=2 TO K DO
BEGIN
F[I]:=B[K-I+1];
FOR J:=1 TO I-1 DO
F[I]:=F[I]-A[K-J+1]*F[I-J];
END;
FOR I:=1 TO K1 DO
BEGIN
XM[K,I]:=A[I];
XM[I,I+1]:=-1;
XM[I+K+1,I+K]:=-1;
XM[I+K,K2]:=A[I];
END;
FOR I:=1 TO K DO
BEGIN
XM[I+K,I]:=AL[I]*2.0;
FOR J:=1 TO K DO
XM[I,J+K]:=-F[I]*F[J]/(BET*2.0);
END;
ZEROS(K,K1,K2,M,MAX,EPS1,EPS2,EPS3,ETA,A,XM,X);
FOR J:=1 TO K2-1 DO
```

```
BEGIN
XMI:=ABS(X[J]);
FOR I:=J+1 TO K2 DO
BEGIN
XMI1:=ABS(X[I]);
IF(XMI>XMI1)THEN BEGIN
R:=X[I];
X[I]:=X[J];
X[J]:=R;
XMI:=XMI1;
END;
END;
END;
FOR I:=1 TO K DO
BEGIN
FOR J:=1 TO K-1 DO
BEGIN
XM[J,I]:=X[I];
XM[J+K,J+K]:=1.0/X[I];
END;
XM[K,K]:=X[I]+A[K];
XM[K2,K2]:=1.0/X[I]+A[K];
FOR L:=1 TO K2 DO
BEGIN
H:=0;
FOR J:=1 TO K2 DO
BEGIN
ALD(K2,L,J,XM,AD);
GAM[J,I]:=AD;
H:=H+GAM[J,I]*GAM[J,I];
END;
IF H>0.0 THEN GOTO 7;
END;
7:END;
FOR I:=1 TO K DO
FOR J:=1 TO K DO
GAM1[I,J]:=GAM[I,J];
DET(K,GAM1,D);
FOR MM:=1 TO K DO
BEGIN
T[MM]:=0.0;
FOR J:=1 TO K DO
BEGIN
RJM:=0.;
FOR I:=1 TO K DO
BEGIN
ALD(K,MM,I,GAM1,AD);
RJM:=RJM+AD*GAM[K+J,I];
RJM:=RJM/D;
END;
T[MM]:=T[MM]+F[J]*RJM;
END;
T[MM]:=T[MM]/(BET*2.0);
END;
END(*DRG*);
PROCEDURE ZEROS(KK,K1,K,M,MAX:INTEGER;
EPS1,EPS2,EPS3,ETA:REAL;A:ATIP;XM:XMTIP;
VAR C:ATIP);
VAR P,P1,P2,X0,X1,X2,R,F,F1,D,E,H,U,V,W,T:REAL;
```

```
I,J,K:INTEGER;
LABEL 10,20,30,40,111,222,333,777;
PROCEDURE FUNC(K,K1,K2,M:INTEGER;R:REAL;
A:ATIP;XM:XMTIP;VAR FD:REAL);EXTERNAL;
BEGIN
FOR K:=1 TO K2 DO
BEGIN
J:=0;
P:=0.9*C[K];
P1:=1.1*C[K];
P2:=C[K];
IF C[K]=0.0 THEN  BEGIN P:=-1;
P1:=1;
P2:=0;
END;
R:=P;
GOTO 222;
10:R:=P1;X0:=F1;
GOTO 222;
20:R:=P2;X1:=F1;
GOTO 222;
30:X2:=F1;D:=-0.5;
IF C[K]=0 THEN H:=-1 ELSE H:=-0.1*C[K];
111:E:1+D;
T:=X0*D*D-X1*E*E+X2*(D+E);
U:=T*T-4.0*X2*D*E*(X0*D-X1*E+X2);
IF U<0 THEN U:= ELSE U:=SQRT(U);
V:=T+U;
W:=T-U;
IF (ABS(V)<ABS(W)) THEN U:=W ELSE U:=V;
IF(U=0.0) THEN U:=1;
T:=-2*X2*E/U;
H:=T*H;
R:=R+H;
IF(ABS(H/R)<EPS1) THEN GOTO 333 ELSE GOTO 222;
40:IF(ABS(F1)<ABS(X2*10.)) THEN BEGIN
X0:=X1;X1:=X2;
X2:=F1;D:=T;
GOTO 111;
END;
T:=0.5*T;H:=0.5*H;
R:=R-H;
222:J:=J+1;
333:FUNC(KK,K1,K2,M,R,A,XM,F);
WRITELN(`ZERI=`);
F1:=F;
IF(J>MAX) THEN GOTO 777;
IF (K>1) THEN BEGIN
FOR I:=2 TO K DO
BEGIN
E:=R-C[I-1];
IF(ABS(E)>EPS2) THEN F1:=F1/E
ELSE BEGIN
R:=R+ETA;GOTO 333;END;
END;
END;
IF (ABS(F)<EPS2)&(ABS(F1)<EPS2) THEN GOTO 777
ELSE BEGIN IF (J=1)THEN GOTO 10 ELSE
BEGIN
IF(J=2) THEN GOTO 20 ELSE
BEGIN
IF(J=3) THEN GOTO 30 ELSE
```

```
GOTO 40;
END;
END;
END;
777:C[K]:=R;
WRITELN(`C[K]`,C[K],` J=`,J,` K= ,K);
END(*K*);
END;


PROCEDURE FUNC(K,K1,K2,M:INTEGER;R:REAL;
A:ATIP;XM:XMTIP;VAR FD:REAL);
VAR I,J:INTEGER;
PROCEDURE DET(N:INTEGER;A:XMTIP;
VAR AD:REAL);EXTERNAL;
BEGIN
FOR I:=1 TO M DO
FOR J:=1 TO K1 DO
BEGIN
XM[J,J]:=R;
XM[J+K,J+K]:=1.0/R;
END;
XM[K,K]:=R+A[K];
XM[K2,K2]:=1.0/R+A[K];
DET(K2,XM,FD);
END(*FUNC*);


PROCEDURE ALD(K2,I,J:INTEGER;
XM:XMTIP;VAR AD:REAL);
VAR N:INTEGER;
PROCEDURE DET(N:INTEGER;A:XMTIP;
VAR AD:REAL);EXTERNAL;
BEGIN
FOR N:=1 TO K2 DO
XM[I,N]:=0.0;
XM[I,J]:=1.0;
DET(K2,XM,AD);
END(*ALD*);


CONST MTIP=5;
TYPE ATIP=ARRAY[1..MTIP] OF REAL;
XMTIP=ARRAY[1..MTIP,1..MTIP] OF REAL;
VAR
LP:TEXT;
A,B,AL,T,F,X:ATIP;
XM,GAM,GAM1:XMTIP;
K,M,I,K2,MAX,J:INTEGER;
BET,EPS1,EPS2,EPS3,ETA:REAL;
PROCEDURE DRG(K,MAX:INTEGER;BET,EPS1,EPS2,EPS3,
ETA:REAL;A,B,AL,F,X:ATIP;
XM,GAM,GAM1:XMTIP;
VAR T:ATIP);EXTERNAL;
BEGIN
REWRITE(LP);
K:=2;
MAX:=20;
BET:=1.0;
EPS1:=0.01;
EPS2:=0.01;
EPS3:=0.001;
```

```
ETA:=0.01;
FOR I:=1 TO K DO
X[I]:=1;
X[3]:=-1;
X[4]:=-1;
A[1]:=-0.5;A[2]:=0.5;A[3]:=1.0;
B[1]:=0.5;B[2]:=0.5;B[3]:=0.0;
B[4]:=0.5;B[5]:=0.5;
AL[1]:=B[4];
AL[2]:=B[5];
DRG(K,MAX,BET,EPS1,EPS2,EPS3,
ETA,A,B,AL,F,X,XM,GAM,GAM1,T);
WRITELN(LP,`T[1]`,T[1],` T[2]=`,T[2]);
CLOSE(LP);
END.
```

4.3. Оптимальное управление с обратной связью при возмущенном движении нелинейных систем

Рассмотрим задачу синтеза оптимального управления, поддерживающего в допустимых пределах отклонения системы от заданной номинальной траектории. Предположим, что траектория оптимальна, т. е. удовлетворяет следующей системе уравнений и условий:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t), \tag{4.26}$$

$$\partial H/\partial \mathbf{u} = 0, \tag{4.27}$$

$$\dot{\psi}^{\mathrm{T}} = -\partial H/\partial \mathbf{x}, \tag{4.28}$$

причем заданы величины t_0, $\mathbf{x}(t_0)$, $t_к$, а также

$$\psi^{\mathrm{T}}(t_к) = \left(\frac{\partial L_k}{\partial \mathbf{x}} + \nu^{\mathrm{T}} \frac{\partial \mathbf{G}}{\partial \mathbf{x}}\right)_{t=t_к}, \tag{4.29}$$

$$G[\mathbf{x}(t_ψ)] = 0. \tag{4.30}$$

Гамильтониан имеет вид $H = L + \psi^{\mathrm{T}}\mathbf{f}$.

Предположим, что в процессе функционирования системы имеются малые отклонения от оптимальной траектории, определяемой соотношениями (4.26)—(4.28), которые возникли вследствие малых возмущений начального состояния $\delta\mathbf{x}(t_0)$ и терминальных условий δG. Обусловленные ими возмущения величин $\mathbf{x}(t)$, $\psi(t)$ и ν (т. е. $\delta\mathbf{x}(t)$, $\delta\psi(t)$ и $d\nu$) в окрестности номинальной траектории удовлетворяют уравнениям

$$\delta\dot{\mathbf{x}} = \mathbf{f}_{\mathbf{x}}\,\delta\mathbf{x} + \mathbf{f}_{\mathbf{u}}\,\delta\mathbf{u}; \tag{4.31}$$

$$\delta\dot{\psi} = -H_{\mathbf{xx}}\,\delta\mathbf{x} - \mathbf{f}_{\mathbf{x}}^{\mathrm{T}}\,\delta\psi - H_{\mathbf{xu}}\,\delta\mathbf{u}, \tag{4.32}$$

$$H_{\mathbf{xu}} = \partial(H_{\mathbf{x}})^{\mathrm{T}}/\partial\mathbf{u}, \quad H_{\mathbf{ux}}\,\delta\mathbf{x} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}}\,\delta G + H_{\mathbf{uu}}\,\delta\mathbf{u} = 0, \tag{4.33}$$

в которых заданы следующие начальные и конечные условия:

$$\delta x(t)|_{t=t_0} = \delta\mathbf{x}(t_0), \tag{4.34}$$

$$\delta G = \left[G_x \delta_x \right]_{t=t_к}, \tag{4.35}$$

$$\delta\psi(t_н) = \left[\left((L_к)_{xx} + (\nu^T G_x) \right) \delta x + G_x^T d\nu \right]_{t=t_к}. \tag{4.36}$$

Предположим, что матрица $\mathbf{H_{uu}} = \mathbf{H_{uu}}(t)$ $(t_0 \leqslant t \leqslant t_к)$ не вырождена. Поскольку все коэффициенты системы (4.31)—(4.33) вычисляются на номинальной (и одновременно оптимальной) траектории, то эта система определяет линейную двухточечную краевую задачу, из которой можно найти

$$\delta u(t) = -H_{uu}^{-1}(H_{ux}\delta x + f_u^T \delta\psi). \tag{4.37}$$

Задача (4.31), (4.32) сводится к линейным задачам § 4.2, где

$$\delta\dot{x} = \mathbf{A}(t)\delta x - \mathbf{B}(t)\delta\psi(t), \tag{4.38}$$

$$\delta\dot{\psi} = -\mathbf{C}(t)\delta x - \mathbf{A}^T\delta\psi. \tag{4.39}$$

При этом $\mathbf{A}(t) = f_x - f_u H_{uu}^{-1} H_{ux}$; $\mathbf{B}(t) = f_u H_{uu}^{-1} f_u^T$; $\mathbf{C}(t) = H_{xx} - H_{xu} H_{uu}^{-1} H_{ux}$.

Соотношение (4.37) дает нам искомый закон управления с обратной связью. Однако, чтобы найти неизвестную функцию $\psi(t)$, требуется решить систему (4.38)—(4.39). Единственное ее отличие от рассмотренных в § 4.2 — в ней имеются терминальные условия (4.34)—(4.36).

Эффективный и удобный метод решения этой задачи состоит в следующем [20]. Решение системы (4.38), (4.39) ищется с помощью следующего представления:

$$\delta\psi(t) = \mathbf{S}(t)\,\delta x(t) + \mathbf{R}(t)\,d\nu; \tag{4.40}$$

$$\delta\mathbf{G} = \mathbf{R}^T \delta x(t) + \mathbf{Q}(t)\,d\nu. \tag{4.41}$$

В этих соотношениях $\mathbf{S}(t)$, $\mathbf{R}(t)$ и $\mathbf{Q}(t)$ — некоторые матрицы, удовлетворяющие условиям

$$\mathbf{S}(t_н) = \left[(L_к)_{xx} + (\nu^T G_x) \right]_{t=t_к}; \tag{4.42}$$

$$\mathbf{R}(t_н) = \left[G_x^T \right]_{t=t_к}, \quad \mathbf{Q}(t_н) = 0, \tag{4.43}$$

после дифференцирования которых, используя (4.40), (4.41), находим, что матрицы $\mathbf{S}$, $\mathbf{Q}$, $\mathbf{R}$ удовлетворяют следующим матричным дифференциальным уравнениям:

$$\dot{\mathbf{S}} = -\mathbf{SA} - \mathbf{AS} + \mathbf{SBS} - \mathbf{C}; \tag{4.44}$$

$$\dot{\mathbf{R}} = -(\mathbf{A}^T - \mathbf{SB})\,\mathbf{R}; \tag{4.45}$$

$$\mathbf{Q} = \mathbf{R}^T\mathbf{BR} \tag{4.46}$$

с граничными условиями (4.42), (4.43).

Решив уравнения (4.44)—(4.46) в обратном времени (от $t_к$ до t_0 и запомнив значения матриц $\mathbf{S}$ и $\mathbf{R}$, находим из (4.37) оптимальное управление по следующей формуле:

$$\delta \mathbf{u}(t) = -H_{\mathbf{uu}}^{-1}\left[\left(H_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}}\mathbf{S}\right)\delta\mathbf{x} + f_{\mathbf{u}}^{\mathrm{T}}\mathbf{R}dv\right]. \tag{4.47}$$

Так как приращение dv вычисляется как функция начального момента времени t_0, формула (4.47) определяет закон дискретного управления с обратной связью. Если приращение dv вычисляется непрерывно, то с учетом предположения о невырожденности матрицы $\mathbf{Q}$ закон оптимального непрерывного управления с обратной связью определяется следующей формулой:

$$\delta \mathbf{u}(t) = -\mathbf{K}_1(t)\,\delta\mathbf{x} - \mathbf{K}_2(t)\,\delta\mathbf{G}, \tag{4.48}$$

где

$$\mathbf{K}_1(t) = H_{\mathbf{uu}}^{-1}\left\{\left[H_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathrm{T}}\left(\mathbf{S} - \mathbf{R}\mathbf{Q}^{-1}\mathbf{R}^{\mathrm{T}}\right)\right]\right\};$$
$$\mathbf{K}_2(t) = H_{\mathbf{uu}}^{-1}\mathbf{f}_{\mathbf{u}}^{\mathrm{T}}\mathbf{R}\mathbf{Q}. \tag{4.49}$$

4.4. Субоптимальное управление нелинейными объектами

Моделью движения системы является векторное дифференциальное уравнение

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \mathbf{F}(\mathbf{x}, t)\mathbf{u}, \tag{4.50}$$

где $\mathbf{x}$ — n-мерный вектор состояния; $\mathbf{u}$ — m-мерный вектор управления; $\mathbf{f}$ — n-мерная векторная функция; $\mathbf{F}$—матричная функция размера $n \times m$.

Критерий качества задан в виде функционала обобщенной работы

$$J = \left[\mathbf{x} - \mathbf{x}_3(t_к)\right]^{\mathrm{T}}\mathbf{S}\left[\mathbf{x} - \mathbf{x}_3(t_к)\right] + \int_{t_0}^{t_к}\left\{\left[\mathbf{x} - \right.\right.$$
$$\left.\left. - \mathbf{x}_3(t)\right]^{\mathrm{T}}\mathbf{Q}\left[\mathbf{x} - \mathbf{x}_3(t)\right] + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}_{\mathbf{u}}^{-1}\right\}dt, \tag{4.51}$$

где $\mathbf{S}$, $\mathbf{Q}$ — неотрицательно определенные $(n \times n)$-матрицы; $\mathbf{R}$ — положительно определенная $(m \times m)$-матрица.

Если через $\mathbf{x}(\mathbf{x}_0, t_0, t)$ обозначить общее решение уравнения свободного движения, т. е. решение системы $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ с начальным условием $\mathbf{x}(t_0) = \mathbf{x}_0$, то оптимальным (в смысле минимума критерия J) управлением с обратной связью будет управление

$$\mathbf{u}^{\text{опт}} = -\mathbf{R}\mathbf{F}^{\mathrm{T}}\left\{\frac{\partial}{\partial x}\mathbf{x}(\mathbf{x}, t, t_к)\,\mathbf{S}\left[\mathbf{x}(\mathbf{x}, t, t_к) - \mathbf{x}_3(t_к)\right] + \right.$$
$$\left. + \int_{t}^{t_к}\frac{\partial}{\partial x}\mathbf{x}(\mathbf{x}, t, \tilde{t})\,\mathbf{Q}\left[\mathbf{x}(\mathbf{x}, t, \tilde{t}) - \mathbf{x}_3(\tilde{t})\right]\right\}d\tilde{t}. \tag{4.52}$$

Практическая реализация данного закона управления зависит от возможности получения общего решения $\mathbf{x}(\mathbf{x}_0, t_0, t)$ и, как правило, возможна лишь для линейных систем. При решении уравнения свободного движения предъявляются повышенные требования к ЭВМ — обеспечить периодические процедуры численного интегрирования в реальном времени. Удобные для практического применения модификации алгоритма предложил А. А. Красовский [40].

Методика синтеза модифицированного регулятора создавалась в предположении, что интервал оптимизации $[t,\ t_к]$ не превышает нескольких десятков процентов от периода короткопериодической составляющей движения системы. В этом случае при дополнительном выполнении обычных допущений о достаточной гладкости функции $\mathbf{f}(\mathbf{x},\ \mathbf{u},\ t)$ можно приближенно определить общее решение уравнения свободного движения. Разлагая функцию общего решения в ряд и ограничиваясь тремя его членами, получаем:

$$\mathbf{x}(\mathbf{x},t,t+\Delta t)=\mathbf{x}+\frac{\Delta t}{1!}\mathbf{f}+\frac{\Delta t^2}{2!}O^{(0)}\mathbf{f}+\frac{\Delta t^3}{3!}O^{(1)}\mathbf{f}+\Delta\mathbf{x}.$$

Здесь $\Delta\mathbf{x}$ — остаточный член разложения; $O^{(i)}$ $(i\geqslant 0)$ — символ оператора вычисления производных:

$$O^{(0)}=\frac{\partial\mathbf{f}}{\partial\mathbf{x}},\quad O^{(1)}=\frac{\partial}{\partial\mathbf{x}}\left(\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\mathbf{f}\right),\quad O^{(2)}=\frac{\partial}{\partial\mathbf{x}}\left[\frac{\partial}{\partial\mathbf{x}}\left(\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\mathbf{f}\right)\mathbf{f}\right],\ \ldots$$

Подставив результат разложения функции $\mathbf{x}(\mathbf{x},\ t,\ t+\Delta t)$ в выражение (4.52) и выполнив в нем интегрирование по переменной Δt, получим формулу для субоптимального управления.

Приведем формулы субоптимального управления в случае нетерминальной $(\mathbf{S}=0)$ задачи, скользящего интервала оптимизации $(t_к-t=\tau,\ \tau$ — константа) и стационарной системы $(\mathbf{f}(\mathbf{x},\ t)\equiv\mathbf{f}(\mathbf{x}))$ для различных видов приближения общего решения уравнения свободного движения:

линейного приближения

$$\mathbf{u}^{опт}=-\mathbf{R}\mathbf{F}^{\mathrm{T}}\left\{\tau\mathbf{Q}\mathbf{x}-\int_0^{\tau}\left[\mathbf{Q}\mathbf{x}_э(z)+O^{(0)}\mathbf{Q}\mathbf{x}_э(z)\,z\right]dz+\right.$$
$$\left.+\frac{1}{2}\tau^2\mathbf{Q}\mathbf{f}+\frac{1}{2}\tau O^{(0)}\mathbf{Q}\mathbf{x}+\frac{1}{6}\tau^3\mathbf{Q}O^{(0)}\mathbf{f}\right\};\qquad(4.53)$$

квадратического

$$\mathbf{u}^{опт}=-\mathbf{R}\mathbf{F}\left\{\tau\mathbf{Q}\mathbf{x}-\int_0^{\tau}\left[\mathbf{Q}\mathbf{x}_э(z)+O^{(0)}\mathbf{Q}\mathbf{x}_э(z)\,z+\frac{1}{2}O^{(1)}\mathbf{Q}\mathbf{x}_э(z)\,z^2\right]dz+\right.$$
$$+\frac{1}{2}\tau\mathbf{Q}\mathbf{f}+\frac{1}{2}\tau^2O^{(0)}\mathbf{Q}\mathbf{x}+\frac{1}{6}\tau^3\mathbf{Q}O^{(0)}\mathbf{f}+\frac{1}{8}\tau^4O^{(0)}\mathbf{Q}O^{(0)}\mathbf{f}+\frac{1}{8}\tau^4O^{(1)}\mathbf{Q}\mathbf{f}+$$
$$\left.+\frac{1}{10}\tau^5O^{(1)}\mathbf{Q}O^{(0)}\mathbf{f}\right\};\qquad(4.54)$$

кубического

$$\mathbf{u}^{опт}=-\mathbf{R}\mathbf{F}^{\mathrm{T}}\left\{\tau\mathbf{Q}\mathbf{x}-\int_0^{\tau}\left[\mathbf{Q}\mathbf{x}_э(z)+O^{(0)}\mathbf{Q}\mathbf{x}_э(z)\,z+\frac{1}{2}O^{(1)}\mathbf{Q}\mathbf{x}_э(z)\,z^2+\right.\right.$$
$$\left.+\frac{1}{6}O^{(2)}\mathbf{Q}\mathbf{x}_э(z)\,z^3\right]dz+\frac{1}{2}\tau^2\mathbf{Q}\mathbf{f}+\frac{1}{2}\tau^2O^{(0)}\mathbf{Q}\mathbf{x}+\frac{1}{6}\tau^3\mathbf{Q}O^{(0)}\mathbf{f}+$$
$$+\frac{1}{6}\tau^3O^{(0)}\mathbf{Q}\mathbf{f}+\frac{1}{6}\tau^3O^{(1)}\mathbf{Q}\mathbf{x}+\frac{1}{8}\tau^4O^{(0)}\mathbf{Q}O^{(0)}\mathbf{f}+\frac{1}{8}\tau^4O^{(1)}\mathbf{Q}\mathbf{f}+$$

$$+\frac{\tau^4}{24}O^{(2)}\mathbf{Q}\mathbf{x}+\frac{\tau^4}{24}\mathbf{Q}O^{(1)}\mathbf{f}+\frac{\tau^5}{20}O^{(1)}\mathbf{Q}O^{(0)}\mathbf{f}+\frac{\tau^5}{30}O^{(0)}\mathbf{Q}O^{(1)}\mathbf{f}+$$
$$+\frac{\tau^5}{30}O^{(2)}\mathbf{Q}\mathbf{f}+\frac{\tau^6}{76}O^{(2)}\mathbf{Q}O^{(0)}\mathbf{f}+\frac{\tau^6}{72}O^{(1)}\mathbf{Q}O^{(1)}\mathbf{f}+\frac{\tau^7}{216}D^{(2)}\mathbf{Q}O^{(1)}\mathbf{f}\Big\}. \quad (4.55)$$

Для реализации субоптимального управления в соответствии с формулами (4.53)—(4.55) необходимо иметь заранее функцию задающего воздействия $\mathbf{x}_3(z)$ $(t\leqslant z\leqslant t+\tau)$. Если же эта функция не известна, то для вычисления подынтегральных функций в соотношениях (4.53)—(4.55) можно использовать экстраполяцию (линейную, квадратическую, кубическую и т. д.) для построения задающего воздействия $\mathbf{x}_3(z)$. В общем случае для вычисления функций $\mathbf{x}_3(z)$ и $\mathbf{x}(\mathbf{x}, t, t+z)$ можно использовать разложения с различной степенью точности (т. е. различные степени экстраполяции).

Предложенный алгоритм синтезирует регулятор на основе аналитических зависимостей, которые легко реализуются на ЭВМ. Вычисление функции оптимального управления требует относительно небольшого количества арифметических операций и поэтому может выполняться с достаточно высокой тактовой частотой, что особенно важно при решении задач синтеза регулятора для динамических систем с быстротекущим процессом функционирования (например, летательных аппаратов). Кроме того, алгоритм удобен для сочетания с алгоритмами оценивания, которые позволяют определить вектор текущего состояния системы $\mathbf{x}$, может быть рекомендован для применения в комплексных системах адаптивного управления в реальном времени.

Глава 5.

Оптимальное оценивание параметров стохастических систем

5.1. Постановка задачи

Необходимость повышения эффективности функционирования систем управления в условиях случайных факторов стимулировала развитие специального математического аппарата, который позволял бы решать задачи оптимизации процесса получения и уточнения информации о свойствах проектируемого объекта и режимах его функционирования.

Пусть работа исследуемой системы с удовлетворяющей степенью приближения описывается некоторой моделью, например одной из приведенных в § 2.1. Вектор состояния измеряется с помощью устройства, которое преобразует данный вектор в некоторую совокупность значений, несущих всю полученную информацию о состоянии системы, т. е. в вектор измерения. Естественно, что к процессу измерения предъявляются требования наблюдаемости и идентифицируемости. Формализуем эти понятия.

Пусть объект описывается следующими разностными уравнениями:

$$\mathbf{x}(k+1)=\mathbf{A}\mathbf{x}(k),\ k=0,\ 1,\ \ldots, \quad (5.1)$$

где $\mathbf{x}(k)$ — n-мерный вектор; $\mathbf{A}$ — $(n\times n)$-матрица. Процесс измерения — соотношениями

$$\mathbf{y}(k)=\mathbf{C}\mathbf{x}(k), \tag{5.2}$$

где $\mathbf{y}(k)$ — вектор измерений (размерности r); $\mathbf{C}$ — $(r\times n)$-матрица.

Объект называется *наблюдаемым*, если по измерениям $\mathbf{y}(0)$, ..., $\mathbf{y}(n-1)$ можно определить состояние $\mathbf{x}(0)$ (и тогда, естественно, и все последующие состояния $\mathbf{x}(1)$, $\mathbf{x}(2)$, ...). Записывая последовательно уравнения (5.2) с $k=0, \ldots, n-1$, получаем матричное соотношение

$$[\mathbf{y}^{\mathrm{T}}(0)\vdots\mathbf{y}^{\mathrm{T}}(1)\vdots\ldots\vdots\mathbf{y}^{\mathrm{T}}(n-1)]=\mathbf{x}^{\mathrm{T}}(0)\,\mathbf{H}, \tag{5.3}$$

где $\mathbf{H}=[\mathbf{C}^{\mathrm{T}}\vdots\mathbf{A}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\vdots\ldots\vdots\mathbf{A}^{\mathrm{T}(n-1)}\mathbf{C}^{\mathrm{T}}]$.

Условие наблюдаемости, эквивалентное условию существования и единственности решения $\mathbf{x}(0)$ уравнения (5.3), состоит в том, что ранг матрицы $\mathbf{H}$ равен n. Тогда пару матриц $(\mathbf{A}, \mathbf{C})$ называют наблюдаемой.

Понятие *идентифицируемости* состоит в возможности определения матрицы $\mathbf{A}$ по измеренным значениям вектора состояния. Легко показать, что идентифицируемость эквивалентна условию существования и единственности решения $\mathbf{A}$ матричного уравнения

$$[\mathbf{x}(1)\vdots\ldots\vdots\mathbf{x}(n)]=\mathbf{A}[\mathbf{x}(0)\vdots\ldots\vdots\mathbf{A}^{n-1}\mathbf{x}(0)]=\mathbf{A}\mathbf{S},$$

а это эквивалентно тому, что матрица $\mathbf{S}$ имеет ранг n.

Понятия наблюдаемости и идентифицируемости без труда переносятся на системы с непрерывным временем. Пусть функционирование системы и процесс измерения описываются следующими дифференциальными уравнениями:

$$\begin{aligned}\dot{\mathbf{x}}(t)&=\mathbf{A}(t)\mathbf{x}(t);\\ \mathbf{y}(t)&=\mathbf{C}(t)\mathbf{x}(t),\end{aligned} \tag{5.4}$$

где $t_0\leqslant t\leqslant t_{\text{к}}$.

Обозначим переходную матрицу линейной системы $\dot{\mathbf{x}}(t)=\mathbf{A}(t)\mathbf{x}(t)$ через $\Phi(\tau, t)$. Тогда условие наблюдаемости состоит в выполнении следующего соотношения:

$$\mathbf{M}(t, t_0)=\int_{t_0}^{t}\Phi^{\mathrm{T}}(\tau,t)\,\mathbf{C}^{\mathrm{T}}(\tau)\,\mathbf{C}(\tau)\,\Phi(\tau,t)\,d\tau>0. \tag{5.5}$$

Матрица $\mathbf{M}(t, t_0)$ называется *матрицей наблюдаемости*; с ее помощью можно определить вектор $\mathbf{x}(t)$: $\mathbf{x}(t)=\mathbf{M}^{-1}(t, t_0)\mathbf{y}(t)$. Сама матрица $\mathbf{M}(t, t_0)$ находится из дифференциального уравнения

$$\frac{d}{dt}\mathbf{M}(t,t_0)=-\mathbf{A}^{\mathrm{T}}(t)\,\mathbf{M}(t,t_0)-\mathbf{M}(t,t_0)\,\mathbf{A}(t)+\mathbf{C}^{\mathrm{T}}(t)\,\mathbf{C}(t)$$

с начальным условием $\mathbf{M}(t_0, t_0)=0$.

Условие наблюдаемости для дискретных систем можно получить в виде, аналогичном (5.5). Пусть $\Phi(i, j)$ — переходная матрица системы (5.1):

$$\mathbf{x}(k)=\Phi(k,0)\,\mathbf{x}(0),\quad \Phi(0,0)=\mathbf{E}.$$

Если сформировать матрицу наблюдаемости в виде

$$\mathbf{M}(k,0) = \sum_{t=0}^{k} \Phi^{\tau}(l,k)\, \mathbf{C}^{\tau}(l)\, \mathbf{C}(l)\, \Phi(l,k),$$

то условие наблюдаемости можно записать в виде неравенства $\mathbf{M}(k, 0) > 0$.

На практике процессу измерения сопутствуют случайные погрешности, а процесс функционирования самой системы подвержен воздействию случайных возмущений. Более того, самих измерений может быть или слишком мало, или слишком много. Таким образом, основной проблемой становится получение оптимальных оценок для переменных, описывающих параметры и состояние динамической системы.

Эти задачи можно классифицировать следующим образом. Пусть $\mathbf{y}(t) = \mathbf{x}(t) + \xi(t)$ $(t \in T)$ описывает процесс измерения, где $\{\mathbf{x}(t),\ t \in T\}$ и $\{\xi(t),\ t \in T\}$ $(T = [t_0,\ t_к]$ — действительные случайные процессы. Если на основе реализации $\{\mathbf{y}(\tau),\ t_0 \leqslant \tau \leqslant t\}$ $(t \in T)$ требуется определить оптимальную с точки зрения какого-либо критерия оценку $\mathbf{x}(t_1)$ при $t_1 < t$, то поставленная задача называется *задачей интерполяции* или *сглаживания*, при $t_1 = t$—*задачей фильтрации*, при $t_1 > t$—*задачей экстраполяции* или *прогноза*.

Большое практическое значение имеют примыкающие к перечисленным задачам определения неизвестных параметров — *идентификации*. Методы идентификации неизвестных параметров многомерных систем представлены алгоритмом Качмажа (одним из самых эффективных и в то же время удобных для реализации на ЭВМ), а также программой, позволяющей практически реализовать этот алгоритм. Наиболее полно рассмотрена задача оптимальной фильтрации динамических систем. Приведены алгоритмы для систем с непрерывным и дискретным временем, для линейных и нелинейных систем. Это известные алгоритмы Винера и Калмана—Бьюси. Алгоритмы фильтрации Калмана—Бьюси не всегда достаточно эффективны на практике, так как при их применении требуется точное соответствие между используемыми в них моделями и реальными характеристиками объектов. Поэтому современная инженерная практика выдвинула необходимость усовершенствования классических алгоритмов на случай неадекватности априорных данных, положенных в основу алгоритмов фильтрации. Обычно это реализуется с помощью построения устойчивых процедур оценивания, в основу которых положены принципы адаптации алгоритмов к изменяющимся внешним условиям. В настоящей работе это направление нашло свое отражение в виде адаптивного алгоритма фильтрации для систем в дискретном времени.

5.2. Идентификация параметров линейных систем по методу Качмажа

Метод Качмажа широко применяется в инженерной практике для решения задачи идентификации дискретных линейных систем вида

$$y(k) = \mathbf{C}^{\tau}\mathbf{x}(k), \quad k = 1,2,... \tag{5.6}$$

где $y(k)$ — скалярный выходной сигнал системы; $\mathbf{x}(k)$ — вектор входного сигнала; $\mathbf{C}$ — вектор неизвестных параметров системы ($\mathbf{x}(k)$ и $\mathbf{C}$ имеют размерность m, k —

дискретный параметр времени), позволяя оценить значения вектора параметров $\mathbf{C}$ системы по наблюдаемым последовательностям $\mathbf{x}(k)$ и $\mathbf{y}(k)$ ($k=1, 2, \ldots$). Эффективность применения метода в первую очередь зависит от стохастических свойств входной последовательности $\mathbf{x}(k)$ ($k=1, 2, \ldots$). Например, если векторы $\mathbf{x}(1), \ldots, \mathbf{x}(m)$ взаимно ортогональны, точное значение вектора параметров $\mathbf{C}$ может быть получено за m шагов. Если же векторы $\mathbf{x}(1), \ldots, \mathbf{x}(l)$ — независимые гауссовские случайные процессы с одинаковыми дисперсиями, то для достаточно точного определения вектора параметров $\mathbf{C}$ требуется уже выборка в $l=(4...5)\ m$ членов.

Метод Качмажа получил широкое распространение и для решения задачи идентификации нестационарных линейных систем.

Алгоритм Качмажа имеет простую структуру и состоит из следующих рекуррентных соотношений:

$$\hat{\mathbf{C}}(k) = \hat{\mathbf{C}}(k-1) + \frac{y(k) - \hat{\mathbf{C}}^{\mathrm{T}}(k-1)\,\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\,\mathbf{x}(k)}, \quad k = 2,3,.... \tag{5.7}$$

Здесь $\hat{\mathbf{C}}(k)$ — оценка на k-м шаге неизвестного вектора $\mathbf{C}$. Значение вектора начального приближения $\hat{\mathbf{C}}(1)$ задается априорно.

Геометрическая интерпретация соотношения (5.7) заключается в том, что каждую оценку $\mathbf{C}(k)$ можно рассматривать как проекцию оценки $\hat{\mathbf{C}}(k-1)$ на k-ю гиперповерхность в m-мерном евклидовом пространстве $\mathbf{R}^m$, определяемую соотношением $y(k) = \mathbf{C}^{\mathrm{T}}\mathbf{x}(k)$. При этом последовательность норм векторов $\Delta\mathbf{C}(k) = \hat{\mathbf{C}}(k) - \mathbf{C}$ сходится, монотонно убывая, к нулю. Увеличение временной корреляции между случайными величинами $\mathbf{x}(1), \ldots, \mathbf{x}(k)$ заметно ухудшает точностные характеристики алгоритма Качмажа в первоначальном варианте (5.8). Существенно повысить скорость алгоритма Качмажа позволили его модификации [41—43].

Обобщенный алгоритм рекуррентного оценивания Качмажа может быть описан следующими формулами:

$$\hat{\mathbf{C}}(k) = \hat{\mathbf{C}}(k-1) + \frac{y(k) - \hat{\mathbf{C}}^{\mathrm{T}}(k-1)\,\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\,\mathbf{x}(k)}\,\mathbf{x}(k),$$

$$\hat{\mathbf{C}}(k-1) = \hat{\mathbf{C}}(k-1) + \alpha(k)\,[\mathbf{C}(k-2) - \hat{\mathbf{C}}(k-1)]. \tag{5.8}$$

Здесь $\hat{\mathbf{C}}(k)$ — текущая оценка вектора параметров $\mathbf{C}$;

$$\alpha(k) = \begin{cases} 1, \text{ если } \rho^2[\hat{\mathbf{C}}(k-2), \tilde{\mathbf{C}}(k)] > \{\rho[\tilde{\mathbf{C}}(k-2), \tilde{\mathbf{C}}(k-1)] + \mu\}^2, \\ 0 \text{ в противном случае,} \end{cases}$$

где

$$\rho[\tilde{\mathbf{C}}(i), \tilde{\mathbf{C}}(j)] = \frac{y(j) - \mathbf{C}^{\mathrm{T}}(i)\,\mathbf{x}(j)}{\mathbf{x}^{\mathrm{T}}(j)\,\mathbf{x}(j)};$$

$$\mu = \begin{cases} \dfrac{y(k) - \hat{\mathbf{C}}^{\mathrm{T}}(k-1)\,\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\,\mathbf{x}(k)} \text{ для алгоритма в [43],} \\ 0 \text{ для алгоритма в [42].} \end{cases}$$

Таким образом, алгоритм Качмажа состоит из рекуррентной процедуры с неизвестными начальными значениями $\tilde{\mathbf{C}}(1)$, $\tilde{\mathbf{C}}(2)$ и $\mathbf{C}(2)$, которые задаются априорно. В процессе работы алгоритма значения $\tilde{\mathbf{C}}(i)$ и $\widehat{\mathbf{C}}(i)$ постепенно уточняются. Предельное значение $\widehat{\mathbf{C}}(i)$ при $i \to \infty$ и дает искомую оценку вектора $\mathbf{C}$.

ПРОГРАММА KACHM

Назначение: решение задачи идентификации линейных дискретных систем вида (5.7) с помощью обобщенного алгоритма Качмажа (5.9) в варианте с поправочным коэффициентом $\mu=0$.

Параметры:

C — массив, состоящий из элементов вектора оцениваемых параметров системы;

C1APR — массив априорных оценок вектора $\mathbf{C}(1)$;

CAPR — массив априорных оценок вектора $\mathbf{C}(2)$;

N — размер массивов C, CAPR, C1APR;

EPS — требуемая точность вычислений (с заданным значением EPS сравнивается значение, равное евклидовой норме вектора разности оценок вектора $\mathbf{C}$, полученных на двух последовательных шагах, т. е. $\Delta d_k = \|\widehat{\mathbf{C}}(k+1) - \widehat{\mathbf{C}}(k)\|$).

Атрибуты всех параметров определяются по умолчанию.

Обращение: CALL KACHM (C, CAPR, C1APR, N, EPS);

Перед вызовом программы необходимо описать размеры массивов C, CAPR и C1APR, а также задать значения параметров EPS, CAPR, C1APR.

Подпрограммы:

SKPR — вычисление скалярного произведения векторов;

RAN — вычисление текущих значений векторов входного $\mathbf{x}(k)$ и выходного $\mathbf{y}(k)$ сигналов:

RAN: PROCEDURE (XK, YK, N);

где XK — массив текущего вектора входного сигнала; YK — выходной сигнал системы; N—размерность массива XK; PRY — формирование скалярного произведения для вычисления выходного сигнала.

Пример. Идентификация системы, вход и выход которой описываются уравнениями

$$y = C_1 x_1 + C_2 x_2,$$

где $[x_1, x_2]$ — двумерный массив вектора входных сигналов; y — скалярный выходной сигнал; $(C_1, C_2]$ — двумерный массив вектора параметров системы, подлежащего идентификации: $C_1=2$, $C_2=3$. Величины x_1 и x_2 — независимые гауссовские стационарные процессы с математическим ожиданием и корреляционной функцией каждого процесса $\mathbf{K}(\tau)=\sigma^2\exp[-\alpha|\tau|]$ (здесь σ, $\alpha a>0$ — константы).

Величины $x_i\,(i=1,\ 2)$ моделировались с помощью следующих рекуррентных соотношений [44]:

$$\Delta z_i(j) = 2\sqrt{12}\,\sigma\left[\xi_i(j) - 0{,}5\right]\sqrt{\alpha\Delta t},$$

$$\Delta p_i(j) = -\alpha\left[x_i(j) - \Delta z_i(j)\right],$$

$$x_i(j) = x_i(j-1) + \Delta p_i(j)\,\Delta t,$$

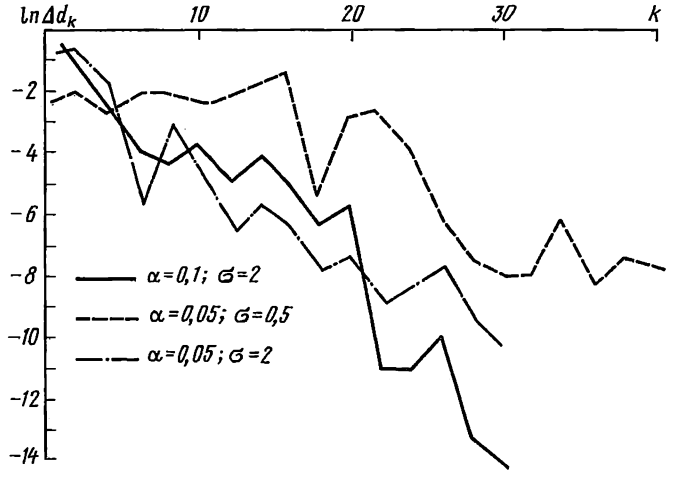


Рис. 5.1. Изменение $\ln \Delta dk$

где $j=1, \ldots, N_1$; N_1 — число шагов для получения одной точки процесса $x_i = x_i(N_1)$; σ, α, Δt — константы (σ^2 — дисперсия процесса; α — ширина спектра; Δt — шаг квантования при порождении очередной точки $x_i(j)$: $\Delta t = N_1^{-1}$); $\xi_i(j)$ — независимые случайные величины, распределенные равномерно на отрезке [0, 1]. На первой итерации начальные значения $x_i(0)$ ($i=1, 2$) полагались нулевыми, на последующих — равными предыдущим сформированным значениям x_i.

Для получения $\xi_i(j)$ использовался генератор псевдослучайных чисел — подпрограмма URAND [2], разработанная на языке Фортран-IV ОС ЕС. Поэтому для ее вызова из программы RAN на языке ПЛ/1 необходимо предусмотреть возможность их стыковки с помощью соответствующих утверждений языка управления заданиями операционной системы (см. приложение 1).

При проведении расчетов предполагалось: $\Delta t=0{,}01$, $\alpha=0{,}05...0{,}5$, $\sigma=0{,}5...1$, CAPR(1) = CAPR(2) = 2,5, C1APR(1) = C1PR(2) = 2,6, EPS = 0,001, N = 2.

Результаты расчетов эволюции $\ln \Delta d_k$ в зависимости от числа шагов идентификации и для различных значений параметров приведены на рис. 5.1 (при построении графиков высокочастотные составляющие Δd_k не учитывались).

```
KACHM:PROCEDURE(C,CAPR,C1APR,N,EPS);
      DECLARE (C,CAPR,C1APR) (*);
      BEGIN;
      DECLARE (CK1,C1K2,CK,C1K1,XK) (N);
      DECLARE XK1(N);
      DCL IY BIN FIXED(31,0)EXT;
      IY=0;  XK=0;
      XK1=1;
      CALL PRY(XK1,N,YK1);
      K=1;
      CALL RAN(XK1,YK1,N);
      DC=EPS+1;
      CK1=CAPR;
```

```
      C1K2=C1APR;
      XK=XK1;
      DO WHILE(ABS(DC) > EPS);
      CALL RAN(XK,YK,N);
      R1=(YK-SKPR(C1K2,XK,N))/SKPR(XK,XK,N)**0.5;
      R2=(YK1-SKPR(C1K2,XK1,N))/SKPR(XK1,XK1,N)**0.5;
      AL=0;
      IF  R1**2 > R2**2 THEN  AL=1;
      C1K1=CK1+AL*(C1K2-CK1);
      CK=C1K1+(YK-SKPR(C1K1,XK,N))/SKPR(XK,XK,N)*XK;
      K=K+1;
      DC=0;
      DO J=1 TO N;
      DC=DC+(CK1(J)-CK(J))**2;
      END;
      DC=DC**0.5;
      CK1=CK;
      C1K2=C1K1;
      YK1=YK;
      XK1=XK;
      END;
      C=CK;
      END;
      END KACHM;


RAN:  PROCEDURE((XK,YK,N);
      DCL IY BIN FIXED(31,0)EXT,
      XK(*);
      ALP=0.05; SIG=2;
      DT=0.01;
      DD=2/ALP; DD=DD/DT;
      DD=SQRT(12*DD);
      DD=SIG*DD;
      BEGIN;
      DECLARE (DZ,PR) (N);
      DO J=1 TO 100;
      DO I=1 TO N;
      CALL URAND(IY,YFL);
      DZ(I)=(YFL-0.5)*DD;
      PR(I)=-ALP*(XK(I)-DZ(I));
      XK(I)=XK(I)+PR(I)*DT;
      END;
      END;
      CALL PRY(XK,N,YK);
      END;
      END RAN;


PRY:  PROCEDURE(X,N,Y);
      DCL X(*);
      Y=2*X(1)+3*X(2);
/END  PRY;

      SUBROUTINE URAND(IY,YFL)
      INTEGER IY
      INTEGER IA,IC,ITWO,M2,M,MIC
      DOUBLE PRECISION HALFM
      REAL S
      DOUBLE PRECISION DATAN,DSQRT
      DATA M2/0/,ITWO/2/
      IF(M2.NE.0) GO TO 20
      M=1
```

```
10    M2=M
      M=ITWO*M2
      IF(M.GT.M2) GO TO 10
      HALFM=M2
      IA=8*IDINT(HALFM*DATAN(1.D0)/8.D0)+5
      IC=2*IDINT(HALFM*(0.5D0-DSQRT(3.D0)/6.D0))+1
      MIC=(M2-IC)+M2
      S=0.5/HALFM
20    IY=IY*IA
      IF(IY.GT.MIC)IY=(IY-M2)-M2
      IY=IY+IC
      UR=URAND
      IF(IY/2.GT.M2)IY=(IY-M2)-M2
      IF(IY.LT.0)IY=(IY+M2)+M2
      URAND=FLOAT(IY)*S
      YFL=URAND
      RETURN
      END

 SKPR: PROCEDURE(A,B,N);
      DECLARE (A,B) (*);
      P=0;
      DO I=1 TO N;
      P=P+A(I)*B(I);
      END;
      RETURN(P);
      END SKPR;


TEST: PROCEDURE OPTIONS(MAIN);
      DECLARE (C,CAPR,C1APR) (2);
      CAPR=2.5;
      C1APR=2.6;
      N=2;
      EPS=0.001;
      CALL KACHM(C,CAPR,C1APR,N,EPS);
      PUT SKIP DATA(C);
  END TEST;
```

5.3. Фильтрация по методу Винера

Рассмотрим задачу синтеза фильтра, обеспечивающего наилучшую точность воспроизведения выходного сигнала многомерной стохастической системы на отрезке времени $[t_0, t_к]$. Пусть $\mathbf{x}(t)$ — n-мерный векторный сигнал системы, который требуется как можно точнее воспроизвести с помощью оценки $\hat{\mathbf{x}}(t)$. Предположим, что $\mathbf{x}(t)$ имеет нулевое математическое ожидание для всех $t \in [t_0, t_к]$.

Будем рассматривать оценки $\hat{\mathbf{x}}(t)$, описываемые системами линейных дифференциальных уравнений с переменными коэффициентами

$$\frac{d\hat{\mathbf{x}}(t)}{dt} = \mathbf{P}(t)\,\hat{\mathbf{x}}(t) + \mathbf{Q}(t)\,\mathbf{z}(t). \tag{5.9}$$

Здесь $\mathbf{z}(t)$ — m-мерный сигнал, поступающий на вход системы (5.9) и представляющий собой нестационарный случайный процесс с нулевым математическим ожиданием; вектор $\hat{\mathbf{x}}(t)$ n-мерный; матрицы $\mathbf{P}(t)$ и $\mathbf{Q}(t)$ размера $n \times n$ и $n \times m$ соответственно.

Сформируем вектор $\mathbf{e}(t)$, представляющий ошибку в воспроизведении желаемого сигнала $\mathbf{x}(t)$:

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t). \tag{5.10}$$

Качество работы фильтра (5.9) может быть оценено с помощью критерия

$$J(t) = \mathbf{M}[\mathbf{e}^{\mathrm{T}}(t)\,\mathbf{e}(t)], \tag{5.11}$$

который следует минимизировать. Правая часть соотношения (5.11) эквивалентна $\mathrm{Sp}\{\mathbf{M}[\mathbf{e}(t)\mathbf{e}^{\mathrm{T}}(t)]\}$, где $\mathrm{Sp}\{\cdot\}$ — след матрицы (т. е. сумма диагональных элементов). Из (5.9) следует, что сигнал $\mathbf{x}(t)$ определяется с помощью соотношения

$$\hat{\mathbf{x}}(t) = \int_{t_0}^{t} \mu(t,\tau)\,\mathbf{z}(\tau)\,d\tau. \tag{5.12}$$

Здесь $\mu(t, \tau)$ — матрица размера $n \times m$:

$$\mu(t, \tau) = \Phi(t, \tau)\,\mathbf{Q}(\tau), \tag{5.13}$$

где $\Phi(t, \tau) = \mathbf{f}(t)\mathbf{f}^{-1}(\tau)$; $\mathbf{f}(t)$ — фундаментальная матрица решений системы однородных дифференциальных уравнений

$$\hat{dx}(t)/dt = \mathbf{P}(t)\hat{\mathbf{x}}(t) \tag{5.14}$$

с начальными нулевыми условиями. С помощью оптимального выбора матрицы $\mu(t, \tau)$ (5.13), являющейся весовой функцией фильтра (5.9), и осуществляется минимизация критерия $J(t)$ (5.11).

Условие оптимальности $J(t)$ эквивалентно матричному интегральному уравнению [18]

$$\mathbf{M}[\mathbf{x}(t)\,\mathbf{z}^{\mathrm{T}}(t_1)] = \int_{t_0}^{t} \mu(t,t_2)\,\mathbf{M}[\mathbf{z}(t_2) \times \mathbf{z}^{\mathrm{T}}(t_1)]\,dt_2,\ t_0 \leqslant t_1 \leqslant t, \tag{5.15}$$

которое называют *интегральным уравнением Винера—Хопфа.* Решая уравнение (5.15) относительно функции $\mu(t, \tau)$, и определяем по формуле (5.12) оценку $\hat{\mathbf{x}}(t)$.

Решение интегрального уравнения (5.15) требует задания априорной информации о сигналах в терминах корреляционных функций и спектральных плотностей, а также больших ресурсов машинного времени для расчетов на ЭВМ.

Достаточно полная подборка работ, посвященных алгоритмам фильтрации по методу Винера, приведена в [13, 18, 20].

5.4. Оптимальная фильтрация линейных систем с дискретным временем (фильтр Калмана)

Оценивание вектора состояния линейной системы на основании наблюдения ее выхода с учетом случайных возмущений системы и погрешности измерения приводит к задаче фильтрации. Одним из наиболее эффективных и распространенных методов решения задачи фильтрации является алгоритм Калмана [20, 45, 46].

Пусть задана математическая модель линейной динамической системы в дискретном времени

$$\mathbf{x}(k+1)=\mathbf{A}(k)\mathbf{x}(k)+\mathbf{w}(k),\ k=0,\ 1,\ \ldots, \tag{5.16}$$

где $\mathbf{x}(k)$ — недоступный непосредственному измерению вектор состояния системы; $\mathbf{w}(k)$ — случайные гауссовские возмущения ($\mathbf{x}(k)$ и $\mathbf{w}(k)$ — n-мерные); $\mathbf{A}(k)$ — детерминированная $(n\times n)$-матрица.

Заданными считаются следующие характеристики:

$$\mathbf{M}[\mathbf{x}(0)]=\bar{\mathbf{x}}(0),\ \mathbf{M}\left[\{\mathbf{x}(0)-\bar{\mathbf{x}}(0)\}\{\mathbf{x}(0)-\bar{\mathbf{x}}(0)\}^{\mathrm{T}}\right]=\mathbf{P}(0),$$

$$\mathbf{M}[\mathbf{w}(k)]=\bar{\mathbf{w}}(k),\ \mathbf{M}\left[\{\mathbf{w}(k)-\bar{\mathbf{w}}(k)\}\{\mathbf{w}(i)-\bar{\mathbf{w}}(i)\}^{\mathrm{T}}\right]=\mathbf{Q}(k)\,\delta_{ki},$$

$$\mathbf{M}\left[\{\mathbf{w}(k)-\bar{\mathbf{w}}(k)\}\{\mathbf{x}(0)-\bar{\mathbf{x}}(0)\}^{\mathrm{T}}\right]=0,$$

$$\delta_{kl}=1 \text{ при } k=i \text{ и } \delta_k=0 \text{ при } k\neq i, \tag{5.17}$$

где $\mathbf{Q}(k)$ — известная матрица.

Соотношения, описывающие процесс измерения вектора состояния, также линейны:

$$\mathbf{y}(k)=\mathbf{C}(k)\mathbf{x}(k)+\mathbf{v}(k),\ k=0,\ 1,\ \ldots, \tag{5.18}$$

где $\mathbf{y}(k)$ — вектор измерения (наблюдения); $\mathbf{v}(k)$ — вектор гауссовских погрешностей измерения ($\mathbf{y}(k)$ и $\mathbf{v}(k)$ — r-мерные); $\mathbf{C}(k)$ — заданная $(r\times n)$-матрица. О векторе $\mathbf{v}(k)$ известно, что

$$\mathbf{M}[\mathbf{v}(k)]=\bar{\mathbf{v}}(k),\qquad \mathbf{M}[\mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(i)]=\mathbf{R}(k)\,\delta_{ki}$$

(матрица $\mathbf{R}(k)$ задана),

$$\mathbf{M}[\{\mathbf{w}(k)-\bar{\mathbf{w}}(k)\}\mathbf{v}^{\mathrm{T}}(i)]=0,\ \mathbf{M}[\{\mathbf{x}(0)-\bar{\mathbf{x}}(0)\}\mathbf{v}^{\mathrm{T}}(i)]=0,$$

т. е. вектор $\mathbf{v}(i)$ не коррелирован ни с возмущениями системы, ни с начальным значением. В дальнейшем без ограничения общности можно считать, что $\mathbf{w}(k)=\mathbf{v}(k)=0$ и $\mathbf{x}(0)=0$.

На основе наблюдений $\mathbf{y}(0)$, ..., $\mathbf{y}(k)$ требуется найти несмещенную (с математическим ожиданием, равным истинному значению оцениваемых параметров) оценку $\hat{\mathbf{x}}(k)$ вектора $\mathbf{x}(k)$. Ошибкой оценивания служит величина $\mathbf{x}(k)-\hat{\mathbf{x}}(k)$, а критерием оптимальности оценки — математическое ожидание ее квадрата $\mathbf{M}[\mathbf{x}(k)-\hat{\mathbf{x}}(k)]^2$. В качестве оптимальной оценки вектора $\mathbf{x}(k)$ служит его условное математическое ожидание [46]

$$\hat{\mathbf{x}}(k)=\mathbf{M}[\mathbf{x}(k)|\mathbf{y}(0),\ldots,\mathbf{y}(k)],$$

которое реализует минимум среднего квадрата ошибки оценивания.

Алгоритм Калмана, который описывается рекуррентными соотношениями, определяющими процесс эволюции оптимальной оценки $\hat{\mathbf{x}}(k)$ во времени:

$$\bar{\mathbf{x}}(k)=\mathbf{A}(k-1)\hat{\mathbf{x}}(k-1),$$

$$\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{C}(k)\,\bar{\mathbf{x}}(k)] \qquad (5.19)$$

($\hat{\mathbf{x}}(0)$ задано; $k = 1, 2, \ldots$). Все необходимые для вычисления матрицы определяются также рекуррентными соотношениями

$$\bar{\mathbf{P}}(k) = \mathbf{A}(k-1)\,\tilde{\mathbf{P}}(k-1)\,\mathbf{A}^{\mathrm{T}}(k-1) + \mathbf{Q}(k-1),$$

$$\mathbf{K}(k) = \bar{\mathbf{P}}(k)\,\mathbf{C}^{\mathrm{T}}(k)\,[\mathbf{C}(k)\,\bar{\mathbf{P}}(k)\,\mathbf{C}^{\mathrm{T}}(k) + \mathbf{R}(k)]^{-1},$$

$$\tilde{\mathbf{P}}(k) = \bar{\mathbf{P}} - \mathbf{K}(k)\,\mathbf{C}(k)\,\bar{\mathbf{P}}(k) = [\mathbf{E} - \mathbf{K}(k)\,\mathbf{C}(k)]\,\bar{\mathbf{P}}(k),\ \bar{\mathbf{P}}(0) = \mathbf{P}(0).$$

Схема оценивания с помощью фильтра Калмана приведена на рис. 5.2. Суть процедуры фильтрации состоит в следующем: оценка $\mathbf{x}(k)$ представляет собой экстраполированную оценку состояния системы после $k-1$ измерения. Оценка $\hat{\mathbf{x}}(k)$ является коррекцией оценки $\bar{\mathbf{x}}(k)$, выполненной с помощью обработки k-го измерения. Матрицы $\mathbf{P}(k)$ и $\tilde{\mathbf{P}}(k)$ представляют собой соответственно экстраполированную и уточненную ковариационные матрицы ошибок фильтрации. Матрица ковариации ошибок оценивания $\tilde{\mathbf{P}}(k)$ не зависит от последовательности измерений $\mathbf{y}(0)$, ..., $\mathbf{y}(k)$. Таким образом, значения матрицы $\mathbf{P}(k)$ можно вычислить заранее и запомнить.

В более сложной математической модели управляемой динамической системы

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{w}(k)$$

последовательность $\mathbf{y}(0)$, ..., $\mathbf{y}(k)$ определяется формулой $\mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{v}(k)$, а $\mathbf{u}(0)$, ..., $\mathbf{u}(k)$ является заданной последовательностью детерминированного управления. В этом случае фильтр Калмана определяется следующими соотношениями:

$$\bar{\mathbf{x}}(k) = \mathbf{A}(k-1)\,\hat{\mathbf{x}}(k-1) + \mathbf{B}(k-1)\,\mathbf{u}(k-1),$$

$$\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{C}(k)\,\bar{\mathbf{x}}(k)]$$

($\mathbf{x}(0)$ задано, а матрица $\mathbf{K}$ вычисляется аналогично вышеизложенному).

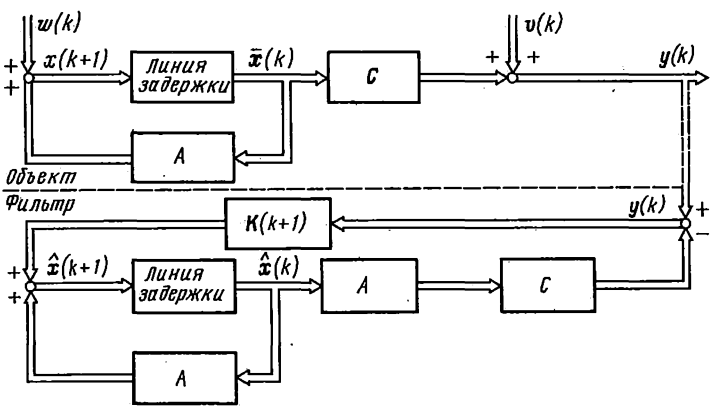


Рис. 5.2. Структурная схема фильтра Калмана

ПРОГРАММА KALMAN

Назначение: рекуррентное гауссовское оценивание дискретной однородной системы (5.16) и фильтрация по Калману для стохастической системы с дискретным временем (5.18). Выбор алгоритма производится изменением значений в соответствующих процедурах FORMQ и FORMR. Чтобы программа работала в режиме гауссовского оценивания, необходимо формировать в процедуре FORMQ нулевую матрицу соответствующего размера, а в процедуре FORMR — единичную.

Параметры:

A — массив размера N*N;

DM — массив размера N*N, соответствующий матрице $\mathbf{M}^{-1}$ ($\mathbf{M}$ — матрица наблюдаемости) при использовании программы для гауссовского оценивания и матрице ковариации $\tilde{\mathbf{P}}(k)=\mathbf{M}[\Delta\hat{\mathbf{x}}(k)\Delta\mathbf{x}^{\mathrm{T}}(k)]$, где $\Delta\hat{\mathbf{x}}(k)=\mathbf{x}(k)-\hat{\mathbf{x}}(k)$ в случае фильтрации по Калману;

X — вектор оценки $\mathbf{x}(k)$;

Y — вектор измерений;

N — размерность фазового вектора X;

M — размерность измеряемого вектора Y;

T — время;

Q — матрица ковариации вектора возмущений;

FORMA — процедура формирования матрицы $\mathbf{A}(k)$;

FORMC — процедура формирования матрицы $\mathbf{C}(k)$;

FORMQ — процедура формирования матрицы ковариации $\mathbf{Q}(k)$;

FORMR — процедура формирования матрицы ковариации $\mathbf{R}(k)$.

Матрицы ковариации $\mathbf{Q}$ и $\mathbf{R}$ вектора возмущений $\mathbf{w}(k)$ размерности N и ошибки измерений размерности M заданы:

$$\mathbf{M}[\mathbf{w}(k)]=0,\ \mathbf{M}[\mathbf{w}(k)\,\mathbf{w}^{\mathrm{T}}(l)]=\mathbf{Q}(k)\,\delta_{kl};$$

$$\mathbf{M}[\mathbf{v}(k)]=0,\ \mathbf{M}[\mathbf{v}(k)\,\mathbf{v}^{\mathrm{T}}(l)]=\mathbf{R}(k)\,\delta_{kl}.$$

При обращении к процедуре KALMAN все параметры должны быть определены на выходе из процедуры $\mathbf{x}=\hat{\mathbf{x}}(k+1)$, остальные параметры исходные для следующего шага.

Процедуры:

GMPRD — перемножение матриц;

TRANSP — транспонирование матриц;

OBRAT — обращение матрицы методом Гаусса-Жордана;

RANDOM — генерация случайных чисел, равномерно распределенных на интервале [0, 1];

NORMRA — генерация случайных чисел, распределенных по нормальному закону.

Пример. Работу процедуры KALMAN иллюстрирует программа ABCKAL. Рассматривается система

$$\mathbf{x}(k+1)=\mathbf{x}(k)+\mathbf{w}(k),\ \mathbf{x}(0)=0,$$
$$\mathbf{y}(k)=\mathbf{x}(k)+\mathbf{v}(k)$$

для двух случаев: 1) $\mathbf{w}(k)\equiv 0$, $\mathbf{Q}\equiv 0$, $\mathbf{R}=1$. Вектор $\mathbf{y}$ при $\mathbf{x}(0)=0$ представляет собой белый шум с нулевым математическим ожиданием; 2) $\mathbf{w}(k)\not\equiv 0$, $\mathbf{Q}=1$, $\mathbf{R}=1$.

Параметры:

RZNQ — признак, если RZNQ≠1, то на систему не действует возмущение **w**;

ZNR — задает значение матрицы **R**(1,1);

ZNQ — задает значение матрицы **Q**(1,1) в процедуре FORMQ;

XOS — задает начальное значение для системы **x**(0);

XOD — задает начальное значение для оценки $\hat{\mathbf{x}}(0)$;

YOD — задает начальное значение для **y**;

QOZ — задает начальное значение **Q**(1,1);

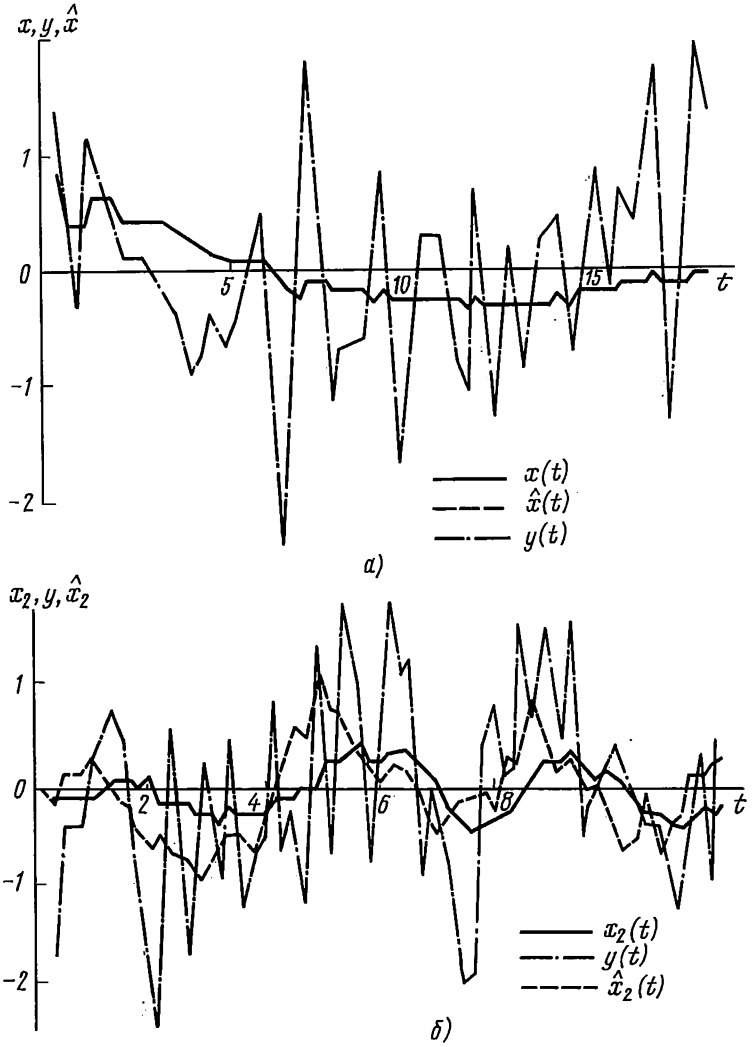


Рис. 5.3. Иллюстрация работы программы фильтра Калмана при $\mathbf{w}(k)=0$(*а*) и $\mathbf{w}(k)\neq 0$(*б*)

DMOZ — задает начальное значение **DM**(1,1);

NRA — число смещений генератора случайных чисел; используется для создания новой последовательности случайных чисел;

AEND — конечное время процесса;

HAGT — шаг по времени между измерениями.

Результаты расчета приведены на рис. 5.3.

```
ABCKAL: PROCEDURE OPTIONS(MAIN);
        DECLARE (ZNDK,ZNQ,ZNR) EXT,
        (I,J,M,N) BIN FIXED(31),
        (AMO,SIGMA) DEC FLOAT EXTERNAL,
        (FORMQ,FORMR) ENTRY,
        X(1), Y(1), E(1,1), Q(1,1),
        A(1,1), AT(1,1), C(1,1), DM(1,1), CT(1,1),
        (GMPRD,FORMA,FORMC,TRANSP) ENTRY,
        (RANDOM,NORMRA) ENTRY,
        KALMAN ENTRY,
        XRANDO FLOAT(53) BIN EXT;
        XRANDO=130000000001;
        PZNQ=1; ZNQ=1; ZNR=1;
        X0S=0; X0D=0; Y0D=0;
        Q0Z=0; DM0Z=1;
        SIGMR=1; SIGQ=1;
        NRA=10; IPW=1;
        AEND=5.2; HAG=0.1;
        DO I=1 TO NRA;
         XN=RANDOM;
        END;
        Q(1,1)=Q0Z;
        X(1)=X0D;     Y(1)=Y0D;     XOB=X0S;
        ZNDK=1;
        NR=1;
        T=0.; N=1; M=1;
        AMO=0.;
        PUT EDIT('SIGMR=',SIGMR)
        (COLUMN(20),A,SKIP(2),COLUMN(25),A,F(8,4));
        PUT EDIT('ZNR=',ZNR)(COLUMN(40),A,E(12,4));
        PUT EDIT('PZNQ=',PZNQ,'ZNQ=',ZNQ,'SIGQ=',SIGQ)
        (SKIP,COLUMN(25),A,F(2,0),2(X(5),A,E(12,4)));
        PUT EDIT('X0S=',X0S,'X0D=',X0D,'Y0D=',Y0D)
        (SKIP(2),COLUMN(25),3(X(3),A,E(12,4)));
        PUT EDIT('DM=',DM0Z,'Q=',Q0Z)(SKIP(2),COLUMN(10),
        2(A,F(5,2),X(5)));
        DM(1,1)=DM0Z;
        CALL FORMA(A,T);
        CALL TRANSP(A,N,N,AT);
        XOB=0;
        IF IPW=1 THEN
        PUT EDIT('XOB','Y','X','ZDK')(SKIP(2),
        COLUMN(17),A,X(15),A,X(10),A,X(10),A);
        DO AZ=0. BY HAG TO AEND;
        NR=NR+1;
        T=AZ;
        CALL NORMRA(AMO,SIGMR,XN);
        Y(1)=XN;
        IF PZNQ=1 THEN DO;
        CALL NORMRA(AMO,SIGQ,XN);
        XOB=XOB+XN;
        Y(1)=XOB+Y(1);
        END;
```

```
    YB=Y(1);
    CALL KALMAN(A,DM,X,Y,N,M,
    T,Q,FORMA,FORMC,FORMQ,FORMR);
    IF IPW=1 THEN
    PUT EDIT(XOB,YB,X(1),ZNDK)(SKIP,COLUMN(5),4E(15,4));
    END;
END ABCKAL;
KALMAN:PROC(A,DM,X,Y,N,M,T,Q,FORMA,FORMC,FORMQ,FORMR);
      DECLARE (N,M,IPZ) BIN FIXED(31),A(*,*),
      DM(*,*),Q(*,*),X(*),Y(*),
      ZNDK EXT, (FORMA,FORMC,FORMQ,FORMR) ENTRY;
      BEGIN;
      DECLARE XP(N),DL(M),(AT,PZ,PR)(N,N),
      (PRNM,CT,DK)(N,M),
      (C,PRM)(M,N),(R,E,PRMM)(M,M),
      (GMPRD,GVPRD,TRANSP) ENTRY;
      DCL OBRAT ENTRY;
      CALL GMPRD(A,DM,PR,N,N,N);
      CALL TRANSP(A,N,N,AT);
      CALL GMPRD(PR,AT,PZ,N,N,N);
      PZ=PZ+Q;
      CALL FORMC(C,T);
      CALL TRANSP(C,M,N,CT);
      CALL GMPRD(C,PZ,PRM,M,N,N);
      CALL GMPRD(PRM,CT,PRMM,M,N,M);
      CALL FORMR(R,T);
      R=PRMM+R;
      CALL OBRAT(R,PRMM,M,IPZ);
      CALL GMPRD(PZ,CT,PRNM,N,N,M);
      CALL GMPRD(PRNM,PRMM,DK,N,M,M);
      ZNDK=DK(1,1);
      CALL GVPRD(A,X,XP,N,N);
      CALL GVPRD(C,XP,DL,M,N);
      Y=Y-DL;
      CALL GVPRD(DK,Y,X,N,M);
      X=X+XP;
      CALL FORMQ(Q,T);
      CALL GMPRD(DK,C,PR,N,M,N);
      CALL GMPRD(PR,PZ,DM,N,N,N);
      DM=PZ-DM;
      CALL FORMA(A,T);
      END;
   END KALMAN;


   OBRAT:PROC(A,E,N,IER);
      DCL A(*,*),E(*,*),(N,IER,I,J,K,KMAX) BIN FIXED(31);
      DO I=1 TO N;
      DO J=1 TO N;
      E(I,J)=0;
      IF I=J THEN E(I,J)=1;
      END;
      END;
      IER=0;
      IF N=1 THEN DO;
      IF A(1,1)=0 THEN GO TO M77;
      E(1,1)=1/A(1,1);
      RETURN;
      END;
      DO J=1 TO N;
      AMAX=ABS(A(J,J));
      IF J=N THEN
```

```
        DO;
        IF A(N,N)=0 THEN GO TO M77;
        GO TO M20;
        END;
        K=J+1;
        KMAX=J;
        DO I=K TO N;
        AM=ABS(A(I,J));
        IF AM > AMAX THEN
        DO;
        AMAX=AM;
        KMAX=I;
        END;
        END;
        IF AMAX=0 THEN GO TO M77;
        IF KMAX=J THEN GO TO M20;
        DO I=1 TO N;
        A1=A(J,I);
        A(J,I)=A(KMAX,I);
        A(KMAX,I)=A1;
        E1=E(J,I);
        E(J,I)=E(KMAX,I);
        E(KMAX,I)=E1;
        END;
   M20:AMAX=A(J,J);
        DO I=1 TO N;
        A(J,I)=A(J,I)/AMAX;
        E(J,I)=E(J,I)/AMAX;
        END;
        DO I=1 TO N;
        IF I=J THEN GO TO M5;
        AMAX=A(I,J);
        DO K=1 TO N;
        E(I,K)=E(I,K)-AMAX*E(J,K);
        IF K>=J THEN
        A(I,K)=A(I,K)-AMAX*A(J,K);
        END;
    M5:END;
       END;RETURN;
  M77:IER=0;
        RETURN;
        END OBRAT;


GVPRD:PROC(A,X,Y,M,N);
        DCL A(*,*),X(*),Y(*),(I,J,M,N) BIN FIXED(31);
        DO I=1 TO M;
        S=0;
        DO J=1 TO N;
        S=S+A(I,J)*X(J);
        END;
        DO J=1 TO N;
        S=S+A(I,J)*X(J);
        END;
        Y(I)=S;
        END;
  END GVPRD;


 GMPRD:PROC(A,B,C,L,M,N);
        DCL A(*,*),B(*,*),C(*,*),
        (I,J,K,L,M,N) BIN FIXED(31);
```

```
     DO I=1 TO L;
     DO J=1 TO N;
     S=0;
     DO K=1 TO M;
     S=S+A(I,K)*B(K,J);
     END;
     C(I,J)=S;
     END;
     END;
 END GMPRD;


 NORMRA:PROC(AM,SIGMA,XNORM);
     DECLARE
     XRANDO FLOAT(53) BIN EXT,
     A STATIC INIT(0.774597),
     RANDOM RETURNS(BIN FLOAT(53));
     SUM=0;
     DO I=1 TO 5;
     XN=RANDOM;
     XN=2*XN*A-A;
     SUM=SUM+XN;
     END;
     XNORM=AM+SIGMA*(SUM+0.01*(SUM**3-3*SUM));
 END NORMRA;


     RANDOM:PROC RETURNS(FLOAT(53)BIN);
     DCL XRANDO FLOAT(53)BIN EXT;
     DCL M37 FLOAT(53)BIN INIT(137438953472E0);
     XRANDO=XRANDO*3125E0;
     XRANDO=XRANDO-FLOOR(XRANDO/M37)*M37;
     RETURN(XRANDO/M37);
     END RANDOM;


     TRANSP:PROC(A,N,M,B);
     DCL (M,N) BIN FIXED(31),
     (I,J) BIN FIXED(31),
     A(*,*),B(*,*);
     DO I=1 TO N;
     DO J=1 TO M;
     B(J,I)=A(I,J);
     END;
     END;
     END TRANSP;


FORMQ:PROC(Q,T);
     DCL Q(1,1),ZNQ EXT;
     Q(1,1)=ZNQ;
END FORMQ;


FORMA:PROC(A,T);
     DCL A(1,1);
     A(1,1)=1;
 END FORMA;


FORMC:PROC(C,T);
     DCL C(1,1);
```

```
    C(1,1)=1;
END FORMC;
FORMR:PROC(R,T);
      DCL R(1,1),ZNR EXT;
      R(1,1)=ZNR;
 END FORMR;
```

5.5. Оптимальная фильтрация линейных систем в непрерывном времени (фильтр Калмана-Бьюси)

При переходе к непрерывному времени разностные и суммарные уравнения должны быть заменены дифференциальными уравнениями и интегралами. Исследования усложняются, так как от конечного числа наблюдений приходится переходить к бесконечному. Моделью исследования служат следующие уравнения объекта и измерителя:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(t)\mathbf{x}(t) + \mathbf{w}(t), \\ \mathbf{y}(t) &= \mathbf{C}(t)\mathbf{x}(t) + \mathbf{v}(t), \end{aligned} \tag{5.20}$$

где $t \geqslant t_0$, подлежащий оценке вектор $\mathbf{x}(t)$ — n-мерный; $\mathbf{x}(t_0)$ — случайный вектор с нулевым математическим ожиданием и матрицей ковариации

$$\mathbf{P}(t_0) = \mathbf{M}\left[(\mathbf{x}(t_0) - \mathbf{M}\mathbf{x}(t_0))(\mathbf{x}(t_0) - \mathbf{M}\mathbf{x}(t_0))^{\mathrm{T}}\right]; \tag{5.21}$$

измеряемый вектор $\mathbf{y}(t)$ — r-мерный; матрицы $\mathbf{A}$ и $\mathbf{C}$ известны. Размерность шумов $\mathbf{w}(t)$ и $\mathbf{v}(t)$ принята n и r соответственно. Возмущения и погрешности гауссовские, причем

$$\begin{aligned} &\mathbf{M}\mathbf{w}(t) = 0, \ \mathbf{M}(\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(t)) = \mathbf{Q}(t)\delta(t-\tau); \\ &\mathbf{M}\mathbf{v}(t) = 0, \ \mathbf{M}(\mathbf{v}(t)\mathbf{v}^{\mathrm{T}}(t)) = \mathbf{R}(t)\delta(t-\tau), \end{aligned} \tag{5.22}$$

где $\delta(t-\tau)$ — дельта-функция. Матрицы $\mathbf{Q}$ и $\mathbf{R}$ симметричные, а их элементы — непрерывные функции времени; $\mathbf{w}(t)$ и $\mathbf{v}(t)$ — взаимно некоррелированные. Искомой является оценка $\hat{\mathbf{x}}(t)$ на основе результатов измерений $\mathbf{y}(\tau)$ $(t_0 \leqslant \tau \leqslant t)$. Линейная несмещенная оценка с минимальной среднеквадратической оценкой описывается уравнением

$$\frac{d}{dt}\hat{\mathbf{x}}(t) = \mathbf{A}(t)\hat{\mathbf{x}}(t) + \mathbf{K}(t)\left[\mathbf{y}(t) - \mathbf{C}(t)\hat{\mathbf{x}}(t)\right], \hat{\mathbf{x}}(t_0) = 0.$$

Матрица Калмана вычисляется по формуле

$$\mathbf{K}(t) = \tilde{\mathbf{P}}(t)\,\mathbf{C}^{\mathrm{T}}(t)\,\mathbf{R}^{-1}(t). \tag{5.23}$$

Матрица ковариации ошибки определяется матричным нелинейным дифференциальным уравнением Риккати

$$\begin{aligned} \frac{d}{dt}\tilde{\mathbf{P}}(t) &= \mathbf{A}(t)\,\tilde{\mathbf{P}}(t) + \tilde{\mathbf{P}}(t)\,\mathbf{A}^{\mathrm{T}}(t) - \\ &- \tilde{\mathbf{P}}(t)\,\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}(t)\,\mathbf{C}(t)\,\tilde{\mathbf{P}}(t) + \mathbf{Q}(t), \ \tilde{\mathbf{P}}(t_0) = \mathbf{P}(t_0). \end{aligned} \tag{5.24}$$

Если математическая модель задана в виде

$$\dot{\mathbf{x}}(t) + \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{B}(t)\,\mathbf{u}(t) + \mathbf{w}(t),\ \mathbf{M}\{\mathbf{x}(t_0)\} = \xi,$$

$$\mathbf{y}(t) = \mathbf{C}(t)\,\mathbf{x}(t) + \mathbf{v}(t),$$

то линейная несмещенная оценка вектора состояния $\mathbf{x}(t)$ с минимальной среднеквадратической ошибкой определяется как решение следующего дифференциального уравнения:

$$\frac{d}{dt}\hat{\mathbf{x}}(t) = \mathbf{A}(t)\,\hat{\mathbf{x}}(t) + \mathbf{B}(t)\,\mathbf{u}(t) + \mathbf{K}(t)\{\mathbf{y}(t) - \mathbf{C}(t)\,\hat{\mathbf{x}}(t)\},\ \mathbf{x}(t_0) = \xi.$$

Матрица $\mathbf{K}(t)$ определяется по формуле (5.23). Матрица $\mathbf{P}(t)$ является решением уравнения Риккати (5.24). Уравнение (5.24) аналитически решается редко, поэтому его следует решать численными методами (см. [1, 2, 62]). Однако в некоторых случаях это уравнение может быть решено и аналитически. Пусть однородное дифференциальное уравнение ошибки фильтрации имеет вид

$$\dot{\mathbf{x}}(t) = [\mathbf{A}(t) - \tilde{\mathbf{P}}(t)\,\mathbf{C}^{\tau}(t)\,\mathbf{R}^{-1}(t)\,\mathbf{C}(t)]\,\tilde{\mathbf{x}}(t),$$

сопряженное ему

$$\dot{\xi}(t) = [-\mathbf{A}^{\tau}(t) + \mathbf{C}^{\tau}(t)\,\mathbf{R}^{-1}(t)\,\mathbf{C}(t)\,\tilde{\mathbf{P}}(t)]\,\xi(t).$$

Правила построения сопряженной системы даются следующей формулой: если $\dot{\mathbf{x}} = \mathbf{F}(t)\mathbf{x}$, то сопряженная матрица $\dot{\xi} = -\mathbf{F}^{\tau}(t)\xi$, т. е. транспонируется и берется с противоположным знаком. Уравнение Риккати (5.24) умножается на $\xi(t)$ справа:

$$\frac{d}{dt}\tilde{\mathbf{P}}(t)\,\xi(t) = [\mathbf{A}(t)\,\tilde{\mathbf{P}}(t) + \tilde{\mathbf{P}}(t)\,\mathbf{A}^{\tau}(t) - \\ - \tilde{\mathbf{P}}(t)\,\mathbf{C}^{\tau}(t)\,\mathbf{R}^{-1}(t)\,\mathbf{C}(t)\,\tilde{\mathbf{P}}(t) + \mathbf{Q}(t)]\,\xi(t).$$

Умножая сопряженное уравнение на $\tilde{\mathbf{P}}(t)$ слева и складывая с предыдущим, получаем:

$$\tilde{\mathbf{P}}(t)\,\dot{\xi}(t) + \tilde{\mathbf{P}}(t)\,\xi(t) = (\mathbf{A}(t)\,\tilde{\mathbf{P}}(t) + \mathbf{Q}(t)\,\xi(t)).$$

Нелинейный член в этом уравнении отсутствует. Введя $\tilde{\mathbf{P}}(t)\xi(t) = \eta(t)$, получим однородную линейную систему

$$\frac{d}{dt}\begin{bmatrix}\xi\\ \eta\end{bmatrix} = \begin{bmatrix}-\mathbf{A}^{\tau}(t) & \mathbf{C}^{\tau}(t)\,\mathbf{R}^{-1}(t)\,\mathbf{C}(t)\\ \mathbf{Q}(t) & \mathbf{A}(t)\end{bmatrix}$$

($\xi(t_0)$ и $\eta(t_0)$ задаются через $\mathbf{P}(t_0)$). Если найдены $\xi(t)$ и $\eta(t)$, то при существовании $\xi^{-1}(t)$ получим $\tilde{\mathbf{P}}(t) = \eta(t)\xi^{-1}(t)$.

Понятия наблюдаемости и управляемости тесно между собой связаны, могут друг друга дополнять благодаря принципу двойственности Калмана: наблюдение и фильтрация исходной системы

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{B}(t)\,\mathbf{u}(t);\ \mathbf{y}(t) = \mathbf{C}(t)\,\mathbf{x}(t),\ t_0 \leqslant t,$$

соответствуют управлению и регулированию двойственной системы, и наоборот.

Математическая модель двойственной системы

$$-\dot{\xi}(t) = \mathbf{A}^{\tau}(t)\,\xi(t) + \mathbf{C}^{\tau}(t)\,\nu(t),\ \eta(t) = \mathbf{B}^{\tau}(t)\,\xi(t).$$

Правила получения модели двойственной системы следующие:

1) транспонирование матриц **A**, **B** и **C**;
2) перестановка входной $\mathbf{B}(t)$ и выходной $\mathbf{C}(t)$ матриц;
3) обращение времени.

Основное назначение модели двойственности состоит в том, что закономерности оптимального управления могут переноситься (в программном аспекте) на синтез оптимальных фильтров.

ПРОГРАММА KALBUS

Назначение: оценивание состояния $\mathbf{x}(t)$ системы с непрерывным временем и процессом наблюдения $\mathbf{y}(\tau)$ $(t_0 \leqslant \tau \leqslant t)$ (фильтр Калмана—Бьюси), описываемых соотношениями (5.20). Априорная информация определяется соотношениями

$$\mathbf{M}[\mathbf{x}(t_0)] = 0,\ \mathbf{M}[\mathbf{x}(t_0)\,\mathbf{x}^{\tau}(t_0)] = \mathbf{P}_0,\ \mathbf{M}[\mathbf{w}(t)] = 0,$$

$$\mathbf{M}[\mathbf{w}(t)\,\mathbf{w}^{\tau}(\tau)] = \mathbf{Q}(t)\,\delta(t-\tau),\quad \mathbf{M}[\mathbf{v}(t)] = 0,$$

$$\mathbf{M}[\mathbf{v}(t)\,\mathbf{v}^{\tau}(\tau)] = \mathbf{R}(t)\,\delta(t-\tau),$$

$$\mathbf{M}[\mathbf{x}(t_0)\,\mathbf{w}^{\tau}(t)] = 0,\ \mathbf{M}[\mathbf{x}(t_0)\,\mathbf{v}^{\tau}(t)] = 0,\ \mathbf{M}[\mathbf{w}(t)\,\mathbf{v}^{\tau}(t)] = 0.$$

Параметры:

TO — начальное время оценивания состояния $\mathbf{x}(t)$;
TK — конечный момент времени оценивания;
XV — вектор состояния $\mathbf{x}(t)$, при обращении к процедуре содержит начальное значение вектора состояния, при выходе — конечное;
PV1 — матрица ковариации $\mathbf{P}(t)$, аналогично предыдущему определяются начальное и конечное значения;
M — размерность вектора измерений $\mathbf{y}(t)$;
N — размерность вектора состояния $\mathbf{x}(t)$;
H — интервал интегрирования системы дифференциальных уравнений (интервал измерения);
FA — формирование матрицы **A**;
FC — формирование матрицы **C**;
FR1 — формирование матрицы **R**;
FQB — формирование матрицы **Q**;
FY — формирование промежуточных значений вектора измерения $\mathbf{y}(t)$ или моделирование $\mathbf{x}(t)$ и формирование $\mathbf{y}(t)$ в соответствии с уравнениями процесса измерений;
OUTP — вывод значений $\hat{\mathbf{x}}(t)$, а также $\mathbf{x}(t)$ и $\mathbf{y}(t)$ при моделировании процесса фильтрации;
ZAMER — вывод измеренных значений $\mathbf{y}(t)$ или генерация случайных чисел для моделирования возмущений $\mathbf{w}(t)$, $\mathbf{v}(t)$.

Дополнительные процедуры:

GMPRD — перемножение матриц;

•TPANSP — транспонирование матриц;

RANDOM — генерация случайных чисел, равномерно распределенных на интервале (0,1);

NORMRA — генерация случайных чисел, распределенных по нормальному закону;

FCTP — вычисление правых частей дифференциальных уравнений, описывающих линейную несмещенную оценку и матрицу ковариации ошибки оценивания.

Пример. Работу программы KALBUS иллюстрирует программа ABC. Рассматривается следующая система:

$$\begin{Vmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Vmatrix} = \begin{Vmatrix} 0 & 1 \\ -4 & -0{,}8 \end{Vmatrix} \begin{Vmatrix} x_1 \\ x_2 \end{Vmatrix} + \begin{Vmatrix} 0 \\ w \end{Vmatrix}, \quad Q = \begin{Vmatrix} 0 & 0 \\ 0 & 4 \end{Vmatrix};$$

$$y = \begin{Vmatrix} 0 & 1 \end{Vmatrix} \begin{Vmatrix} x_1 \\ x_2 \end{Vmatrix} + v, \quad R = 1.$$

Выходные параметры:

TO — начальный момент времени;

TK — конечный момент времени;

H — шаг по времени при работе программы KALBUS;

PV — начальное значение матрицы ковариации;

XV — начальное значение вектора $\hat{\mathbf{x}}(t)$;

SIGMW — среднеквадратическая ошибка возмущений $\mathbf{w}(t)$;

ZNQ — значение матрицы **Q**, передаваемое в процедуру FORMQB;

ZNR — значение матрицы **P**, передаваемое в процедуру FORMR1;

SIGMV — среднеквадратическая погрешность измерений $\mathbf{v}(t)$.

Результаты расчета представлены на рис. 5.4, 5.5.

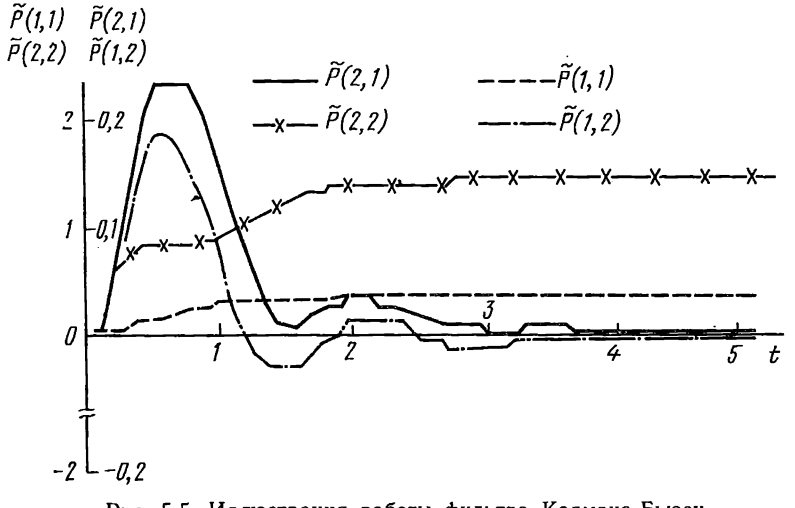


Рис. 5.5. Иллюстрация работы фильтра Калмана-Бьюси

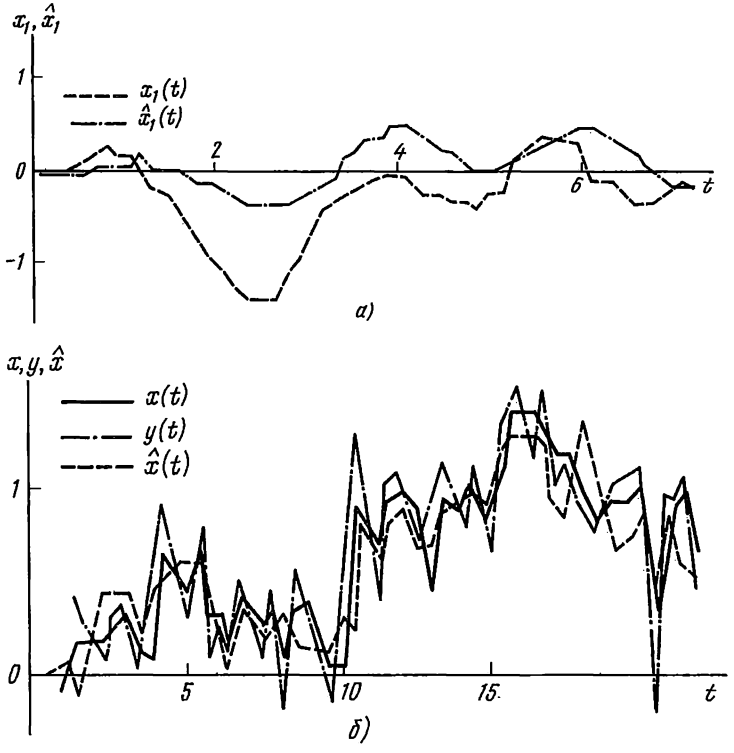


Рис. 5.4. Иллюстрация работы программы фильтра Калмана-Бьюси

```
ABC:PROCEDURE OPTIONS(MAIN)
DECLARE KALBUS ENTRY
(FORMA,FORMC,FORMR1,FORMQB,OUTP,FORMY,ZAMER) ENTRY,
(M,N,NN,IPZ) BIN FIXED(31),
XV(2),PV1(2,2),RANDOM ENTRY,
(X1Z,H,X2Z,WZY,VZY)EXT,
(XRANDO FLOAT(53) BIN,ZNQ,ZNR,SIGMY,SIGMW) EXT;
XRANDO=130000000001;
M=1;
N=2;
NN=6;
NRG=0; WZY=0; VZY=0;
AMO=0.;
GET LIST(TO,TK,H,PV1,NRA);
DO I=1 TO NRA;XSL=RANDOM;END;
PUT EDIT('PV1',PV1(1,1),PV1(1,2),PV1(2,1),PV1(2,2))
(SKIP(2),COL(10),A,4F(10,3));
GET LIST(XV(1),XV(2));
X1Z=XV(1);X2Z=XV(2);
```

```
PUT EDIT('X1=',X1Z,'X2=',X2Z)
(SKIP,COL(10),2(A,F(7,3),X(3)));
GET LIST(SIGMV,ZNQ,SIGMW,ZNR);
PUT EDIT('SIGMV=',SIGMV,
'ZNQ=',ZNQ,'SIGMW=',SIGMW,'ZNR=',ZNR)
(SKIP(3),COLUMN(30),A,SKIP(2),
COLUMN(5),4(A,F(6,3),X(5)));
PUT EDIT('T','X1','X01','X2','Y','X02','P11',
'P12','P21','P22')(SKIP(2),COLUMN(10),A,
2(X(6),A),X(8),3(X(6),A),X(5),4(X(6),A));
CALL KALBUS(TO,TK,XV,PV1,M,N,H,IPZ,
FORMA,FORMC,FORMR1,FORMQB,FORMY,OUTP,ZAMER);
END ABC;
KALBUS:PROC(TO,TK,XV,PV1,M,N,H,IHLF,
FA,FC,FR1,FQB,FY,OUTP,ZAMER);
DECLARE (FA,FC,FR1,FQB,FY,ZAMER,OUTP,
FCTP) ENTRY,
(I,J,K,M,N,,NN,IHLF) BIN FIXED(31),
TZ EXTERNAL,
XV(*),PV1(*,*);
NN=N*N+N;
BEGIN;
DECLARE Z1 BIN FIXED(31) INIT(1B),
(Y,Y1,YP,(K1,K2,K3,K4)DEC FLOAT)(NN),
YZ(M);
DO I=1 TO N;
K=N*N+1; Y(K)=XV(I);
DO J=1 TO N;
K=J+N*(I-Z1);
Y(K)=PV1(I,J);
END;
END;
IHLF=0; YZ(1)=0; T=TO;
P=0.;
CALL OUTP(T,Y,YP,NN,P,YZ);
METKA:;
IF T+H>TK THEN H=TK-T; TZ=T;
CALL ZAMER;
CALL FCTP(T,Y,YP,M,N,YZ,FA,FC,FR1,FQB,FY);
K1=YP*H;Y1=Y+K1*0.5;
T1=T+H/2;
CALL FCTP(T1,Y1,YP,M,N,YZ,FA,FC,FR1,FQB,FY);
K2=YP*H;Y1=Y+K2*0.5;
CALL FCTP(T1,Y1,YP,M,N,YZ,FA,FC,FR1,FQB,FY);
K3=YP*H;Y1=Y+K3;
T1=T+H;
CALL FCTP(T1,Y1,YP,M,N,YZ,FA,FC,FR1,FQB,FY);
K4=YP*H;
Y1=Y+(K1+2*K2+2*K3+K4)/6;
T=T1;
Y=Y1;
CALL OUTP(T,Y,YP,NN,P,YZ);
IF P=1 THEN
DO;
IHLF=-1; GOTO MK;
END;
IF T>=TK THEN GOTO MK;
GOTO METKA; MK:;
DO I=1 TO N;
K=N*N+I;XV(I)=Y(K);
DO J=1 TO N;
K=J+N*(I-Z1); PV1(I,J)=Y(K);
```

```
END;
END;
END;
END KALBUS;
ZAMER:PROCEDURE;
DCL (SIGMW,SIGMV,TZ,THACH,X1Z,X2Z,WZY,
VZY,XZNM(2))EXT;
DCL (WZYB,VZYB) EXTERNAL, NORMRA ENTRY;
WZYB=WZY;VZYB=VZY;AMO=0;
CALL NORMRA(AMO,SIGMW,WZY);
CALL NORMRA(AMO,SIGMV,VZY);
THACH=TZ;
XZNM(1)=X1Z;XZNM(2)=X2Z;
END ZAMER;
FCTP:PROC(T,PRK,P1RK,M,N,YZ,FORMA,FORMC,
FORMR1,FORMQB,FORMY);
DECLARE PRK(*),P1RK(*),YZ(*),
(M,N,NM,I,J,K,)BIN FIXED(32),
(FORMA,FORMC,FORMR1,FORMQB,FORMY)ENTRY;
NN=N*N+N;
BEGIN;
DECLARE
P4(N,M),DK(N,M),CT(N,M),C(N,M),R1(M,M),
Y1(M),X1(N),X2(N),(A,AT,P1,P2,P3,QB)
(N,N),PV(N,N),X(N),
(GMPRD,TRANSP,GVPRD) ENTRY;
DO I=1 TO N;
K=N*N+1; X(I)=PRK(K);
END;
END;
CALL FORMQB(QB,T);
CALL FORMA(A,T); CALL TRANSP(A,N,M,AT);
CALL FORMC(C,T);
CALL TRANSP(C,M,N,CT);
CALL FORMR1(R1,T);
CALL GMPRD(PV,CT,P4,N,N,M);
CALL GMPRD(P4,R1,DK,N,M,M);
CALL GMPRD(DK,C,P3,N,M,N);
CALL GMPRD(A,PV,P1,N,N,M);
CALL GMPRD(PV,AT,P2,N,N,N);
DO I=1 TO N;
DO J=1 TO N;
K=J+N*(I-1);
P1RK(K)=P1(I,J)+P2(I,J)-P3(I,J)+QB(I,J);
END;
END;
CALL GVPRD(C,X,Y1,M,N);
CALL FORMY(YZ,M,T);
Y1=YZ-Y1;
CALL GVPRD(DK,Y1,X2,N,M);
CALL GVPRD(A,X,X1,N,N);
DO I=1 TO N;
K=N*N+I; P1RK(K)=X1(I)+X2(I);
END;
END;
END FCTP;
OUTP:PROC(T,PRK,P1RK,NN,PR,YZ);
DCL (M,N,NN) BIN FIXED(31);
DCL (VZY,WZY,X1Z,X2Z) EXT;
DCL PRK(*),P1RK(*),YZ(*);
PP=0;
PUT EDIT(T,X1Z,PRK(5),X2Z,YZ(1),PRK(6),
```

```
PRK(1),PRK(2),PRK(3),PRK(4))
(COLUMN(3),F(5,2),X(3),2E(10,2),X(3),
3E(10,2),X(3),4E(10,2));
END OUTP;
FORMA:PROC(A,T);
DCL A(2,2);/*A(N,N)*/
A(1,1)=0.;A(1,2)=1.;A(2,1)=-4.;A(2,2)=-0.8;
END FORMA;
FORMC:PROC(C,T);
DCL C(1,2);/* C(M,N) */
C(1,1)=0.;C(1,2)=1.;
END FORMC;
FORMQB:PROC(QB,T);
DCL ZNQ EXT;
DCL QB(2,2);
QB(1,2)=0;
QB(2,1)=0;
QB(1,1)=0;
QB(2,2)=ZNQ;
END FORMQB;
FORMR1:PROC(R1,T);
DCL R1(1,1);
DCL ZNR EXT;
DCL M BIN FIXED(31);
R1(1,1)=ZNR;
END FORMR1;
FORMY:PROC(Y,M,T);
DCL M BIN FIXED(31),
Y(1),
(VZY,WZY,WZYB,VZYB,
X1Z,X2Z,
H,THACH,XZNM(2)) EXTERNAL;
X1Z=XZNM(1)+XZNM(2)*(T-THACH);
VZ=VZYB+(VZY-VZYB)/H*(T-THACH);
X2Z=XZNM(2)+(-4*XZNM(1)-0.8*XZNM(2)+VS)*
(T-THACH);
Y(1)=X2Z+(WZYB-WZY)/H*(T-THACH)+WZYB;
END FORMY;
```

5.6. Фильтрация нелинейных систем

Необходимость оптимизации процессов измерения в сложных системах (большинство из которых нелинейные) стимулировала разработку методов нелинейной фильтрации. Основой для их развития послужила теория стохастических дифференциальных уравнений [46—48], некоторые понятия которой изложены в § 2.1.

Стохастическое дифференциальное уравнение диффузионного типа

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x},t)\,dt + \mathbf{v}(\mathbf{x},t)\,d\beta(t), \quad t \geqslant t_0. \tag{5.25}$$

Здесь $\beta(t)$ — стандартный винеровский процесс, т. е. процесс, принимающий вещественные значения с независимыми и нормально распределенными приращениями, математическим ожиданием $\mathbf{M}[d\beta]=0$ и дисперсией $\mathbf{M}[d\beta^2]=dt$; функции $\mathbf{f}(\mathbf{x}, t)$ и $\mathbf{v}(\mathbf{x}, t)$ — коэффициенты сноса и диффузии. Часто (особенно в инженерных приложениях) используется стохастическое дифференциальное уравнение вида

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x},t) + \mathbf{v}(\mathbf{x},t)\,\dot{\beta}(t). \tag{5.26}$$

Пусть *нелинейная система с непрерывным временем* задана векторным стохастическим уравнением диффузионного типа

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x},t)\,dt + \mathbf{v}(\mathbf{x},t)\,d\beta. \tag{5.27}$$

Вектор состояния системы $\mathbf{x}(t)$ недоступен непосредственному наблюдению, однако наблюдается другой вектор $\mathbf{y}(t)$, так что процесс измерений также описывается стохастическим дифференциальным уравнением

$$d\mathbf{y}(t) = h(\mathbf{x},t)\,dt + d\gamma(t). \tag{5.28}$$

Векторы $\mathbf{x}$ и $\mathbf{f}$ — n-мерные, $\mathbf{y}$ и $\mathbf{h}$ — r-мерные. Винеровские процессы $\beta(t)$ и $\gamma(t)$ принимаются независимыми.

Для оценки вектора состояния применяется линеаризация уравнений (5.27), (5.28) с последующим использованием методов линейной фильтрации, что приводит к следующим уравнениям:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}},t) + \mathbf{v}(\hat{\mathbf{x}},t)\,\dot{\bar{\beta}}(t) + \mathbf{P}\left(\frac{\partial h}{\partial \mathbf{x}}\right)^{\mathrm{T}} \mathbf{R}^{-1}\left[\mathbf{y}(t) - \mathbf{h}(\hat{\mathbf{x}},t)\right],$$

$$\hat{\mathbf{x}}(t_0) = \bar{\mathbf{x}}(t_0) = \mathbf{M}\left[\mathbf{x}(t_0)\right], \tag{5.29}$$

$$\dot{\mathbf{P}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{P} + \mathbf{P}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^{\mathrm{T}} + \mathbf{v}\mathbf{Q}\mathbf{v}^{\mathrm{T}} - \mathbf{P}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)^{\mathrm{T}} \mathbf{R}^{-1}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{P},$$

$$\mathbf{P}(t_0) = \mathbf{P}_0 = \mathbf{M}\left[\left\{\mathbf{x}(t_0) - \bar{\mathbf{x}}(t_0)\right\}\left\{\mathbf{x}(t_0) - \mathbf{x}(t_0)\right\}^{\mathrm{T}}\right].$$

$$\mathbf{M}\left[\left\{\dot{\beta}(t) - \dot{\bar{\beta}}(t)\right\}\left\{\dot{\beta}(\tau) - \dot{\bar{\beta}}(\tau)\right\}^{\mathrm{T}}\right] = \mathbf{Q}(t)\,\delta(t-\tau);$$
$$\mathbf{M}\left[\left\{\dot{\gamma}(t) - \dot{\bar{\gamma}}(t)\right\}\left\{\dot{\gamma}(\tau) - \dot{\bar{\gamma}}(\tau)\right\}^{\mathrm{T}}\right] = \mathbf{R}(t)\,\delta(t-\tau);$$
$$\mathbf{M}\left[\left\{\mathbf{x}(t_0) - \bar{\mathbf{x}}(t_0)\right\}\left\{\dot{\beta}(t) - \dot{\bar{\beta}}(t)\right\}^{\mathrm{T}}\right] = 0,$$

$\dot{\bar{\beta}}(t)$, $\dot{\bar{\gamma}}(t)$, $\dot{\bar{\mathbf{x}}}(t)$ — функции математических ожиданий соответствующих процессов. Частные производные $\partial \mathbf{f}/\partial \mathbf{x}$ и $\partial \mathbf{h}/\partial \mathbf{x}$ могут вычисляться как вдоль опорной (номинальной) траектории, описываемой дифференциальным уравнением $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$, так и в точке, представляющей оптимальную оценку $\mathbf{x}$. В последнем случае матрицу $\mathbf{P}(t)$ можно вычислять лишь в режиме реального времени, так как она будет зависеть от текущей оценки $\hat{\mathbf{x}}$ (через $\partial \mathbf{f}/\partial \mathbf{x}$, $\partial \mathbf{h}/\partial \mathbf{x}$).

Пусть в *динамической системе с дискретным временем* задан нелинейный многомерный процесс

$$\mathbf{x}(k+1) = \mathbf{f}^k[\mathbf{x}(k)] + \mathbf{w}(k),\ k = 0,1,\ldots. \tag{5.30}$$

Измерения производятся с помощью устройства, описываемого нелинейным уравнением

$$\mathbf{y}(k) = \mathbf{h}^k[\mathbf{x}(k)] + \mathbf{v}(k),\ k = 0,1,\ldots. \tag{5.31}$$

Предположения о статистических свойствах всех случайных величин остаются такими же, как и в § 5.5.

Алгоритм нелинейной фильтрации определяется следующими рекуррентными соотношениями:

$$\hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{h}^k\{\bar{\mathbf{x}}(k)\}], \quad \bar{\mathbf{x}}(k) = \mathbf{f}^k[\hat{\mathbf{x}}(k-1)];$$

$$\bar{\mathbf{P}}(k) = \frac{\partial \mathbf{f}^{k-1}}{\partial \mathbf{x}(k-1)} \bar{\mathbf{P}}(k-1) \left[\frac{\partial \mathbf{f}^{k-1}}{\partial \mathbf{x}(k-1)}\right]^{\mathrm{T}} + \mathbf{Q}(k-1);$$

$$\mathbf{K}(k) = \bar{\mathbf{P}}(k) \left[\frac{\partial \mathbf{h}^k}{\partial \mathbf{x}(k)}\right]^{\mathrm{T}} \left\{\frac{\partial \mathbf{h}^k}{\partial \mathbf{x}(k)} \bar{\mathbf{P}}(k) \left[\frac{\partial \mathbf{h}^k}{\partial \mathbf{x}(k)}\right]^{\mathrm{T}} + \mathbf{R}(k)\right\}^{-1};$$

$$\tilde{\mathbf{P}}(k) = \bar{\mathbf{P}}(k) - \mathbf{K}(k) \frac{\partial \mathbf{h}}{\partial \mathbf{x}(k)} \bar{\mathbf{P}}(k). \tag{5.32}$$

Здесь $\bar{\mathbf{P}}(0) = \mathbf{P}(0)$, а начальное значение оценки $\hat{\mathbf{x}}(0)$ задано.

Частные производные $\partial \mathbf{f}^k/\partial \mathbf{x}(k)$ и $\partial \mathbf{h}^k/\partial \mathbf{x}(k)$ можно вычислять либо на номинальной траектории, описываемой уравнением $\mathbf{x}(k+1) = \mathbf{f}^k[\mathbf{x}(k)]$, либо $\partial \mathbf{f}^k/\partial \mathbf{x}(k)$ в точке $\hat{\mathbf{x}}(k)$, а $\partial \mathbf{h}^k/\partial \mathbf{x}(k)$ в точке $\bar{\mathbf{x}}(k)$. В последнем случае матрица $\bar{\mathbf{P}}(k)$ может быть вычислена лишь в режиме реального времени.

Метод линеаризации (и, следовательно, сведение задачи нелинейной фильтрации к хорошо разработанным методам линейной фильтрации) зарекомендовал себя как достаточно эффективный. Однако точность фильтрации может быть значительно ухудшена при наличии в системе случайных возмущений большой интенсивности, так как в этом случае процесс линеаризации связан с большими погрешностями. Кроме того, заметное влияние на точность работы алгоритма нелинейной фильтрации имеет выбор начальной оценки $\hat{\mathbf{x}}(t_0)$ ($\hat{\mathbf{x}}(0)$).

Для проведения практических расчетов на ЭВМ предложенные алгоритмы нелинейной фильтрации могут быть реализованы с помощью приведенных выше программ KALMAN и KALBUS. При этом необходимо обеспечить вычисление частных производных $\partial \mathbf{f}^k/\partial \mathbf{x}(k)$, $\partial \mathbf{h}^k/\partial \mathbf{x}(k)$ в подпрограммах, формирующих модели системы, а также модели процесса измерений (см. § 5.5).

5.7. Адаптивная фильтрация линейных систем с дискретным временем

Алгоритм Калмана при решении практических задач бывает недостаточно эффективным (т. е. имеет недостаточную точность) или, более того, в процессе вычисления оценок может разойтись. Это вызвано следующими причинами: при функционировании реальной системы ее параметры могут подчас существенно отличаться от параметров математической модели этой системы, на основе которой построен алгоритм фильтрации; реально действующие на систему и измерительные устройства шумы являются негауссовскими или имеют параметры распределения, отличные от тех, на которые настроен алгоритм. Таким образом, потребности практической реализации алгоритмов фильтрации обусловливают необходимость усовершенствования классического фильтра Калмана так, чтобы он эффективно работал и при несоот-

ветствии априорных моделей системы и шумов их конкретным реализациям. Современные методы решения данной задачи базируются на двух основных подходах. Первый связан с оцениванием (идентификацией) параметров модели в процессе функционирования и использованием полученных оценок в традиционном фильтре Калмана. Второй подход основан на анализе «обновляющегося» процесса и адаптации фильтра с помощью перестройки его структуры на реальные процессы, действующие в канале измерений.

Рассмотрим алгоритм адаптивной фильтрации, построенный на основе второго подхода. Пусть линейная система с дискретным временем вида

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}(k)\,\mathbf{x}(k) + \mathbf{G}(k)\,\mathbf{w}(k), \\ \mathbf{y}(k) &= \mathbf{C}(k)\,\mathbf{x}(k) + \gamma(k)\,\alpha(k) + [1-\gamma(k)]\,\mathbf{p}(k), \end{aligned} \tag{5.33}$$

где $\mathbf{A}(k)$ и $\mathbf{G}(k)$ — детерминированные вещественные матрицы размера $n\times n$ и $n\times s$ соответственно; $\mathbf{x}(k)$ — n-мерный вектор состояния системы; $\mathbf{w}(k)$ — случайные независимые в совокупности гауссовские возмущения (s-мерные) с нулевым математическим ожиданием и ковариационной матрицей $\mathbf{K}_{\mathbf{w}(k)}$; $\mathbf{y}(k)$ — r-мерный вектор измерений; $\mathbf{C}(k)$ — $(r\times n)$-матрица, $k=0, 1, \ldots$.

Последовательность $\gamma(k)$ образуют независимые между собой случайные величины, принимающие значения 0 и 1 с вероятностями g_k и p_k: $\mathrm{P}\{\gamma(k)=1\}=p_k=1-q_k$. Случайные r-мерные величины $\alpha(k)$ и $\beta(k)$ независимы в совокупности и являются гауссовскими с нулевым математическим ожиданием и ковариационными матрицами $\mathbf{K}_{\alpha(k)}$ и $\mathbf{K}_{\beta(k)}$ соответственно. Обычно предполагают, что $p_k \ll q_k$, и тогда приведенная модель описывает ситуацию, когда в канале измерения имеются обычные по характеру шумы $\beta(k)$, чередующиеся с редко появляющимися, но мощными помехами $\alpha(k)$, соответствующими случаям аномальных измерений.

Начальное значение $\mathbf{x}(0)$ вектора состояния системы предполагается гауссовским с математическим ожиданием $\mathbf{m}_{\mathbf{x}(0)}$ и ковариационной матрицей $\mathbf{K}_{\mathbf{x}(0)}$. Предположим далее, что последовательности $\{\mathbf{w}(k)\}$, $\{\gamma(k)\}$, $\{\alpha, (k)\}$ и $\{\beta(k)\}$ взаимно независимы и независимы от $\mathbf{x}(0)$. В дальнейшем тот факт, что случайная величина ξ с математическим ожиданием $\mathbf{m}_\xi$ и ковариационной матрицей $\mathbf{K}(\xi)$ гауссовская, будем обозначать $\xi \sim \mathbf{N}(\mathbf{m}_\xi \mathbf{K}_\xi)$.

В основе алгоритма адаптивной фильтрации лежит идея классификации результатов наблюдений $\mathbf{y}(k)$ на две группы, соответствующие шумам $\alpha(k)$ и $\beta(k)$, с последующей модификацией алгоритмов фильтрации, учитывающей реальную модель помех. Для этого привлекается методология байесовских решений [49, 50]. В этом случае оценка $\gamma(k)$ определяется следующим образом:

$$\hat{\gamma}(k) = \begin{cases} 0, \text{ если } \Lambda_k[\Delta\mathbf{y}(k)] \geqslant c_k, \\ 1, \text{ если } \Lambda_k[\Delta\mathbf{y}(k)] < c_k, \end{cases}$$

где отношение правдоподобия $\Lambda_k[\Delta\mathbf{y}(k)]$ задается как

$$\Lambda_k[\Delta\mathbf{y}(k)] = \frac{f[\Delta\mathbf{y}(k)|\gamma(k)=1]}{f[\Delta\mathbf{y}(k)|\gamma(k)=0]},$$

а константа c_k определяет заданный порог. В приведенных соотношениях $\Delta\mathbf{y}(k) = \mathbf{y}(k) - \mathbf{C}(k)\mathbf{m}_{\mathbf{x}(k)}$, а математическое ожидание процесса $\mathbf{x}(k)$ определяется рекуррентно: $\mathbf{m}_{\mathbf{x}(k+1)} = \mathbf{A}(k)\mathbf{m}_{\mathbf{x}(k)}$ $(k \geqslant 0)$.

Введем условную плотность случайной величины $\Delta y(k)$ при данной $\gamma(k)$:

$$f[\Delta y(k)|\gamma(k)] = \begin{cases} \sim N(0, \tilde{K}_{x(k)} + K_{\alpha(k)}) & \text{при } \gamma(k) = 1, \\ \sim N(0, \tilde{K}_{x(k)} + K_{\beta(k)}) & \text{при } \gamma(k) = 0, \end{cases}$$

где $\tilde{K}_{x(k)} = C(k) K_{x(k)} C^{\mathrm{T}}(k)$.

Определим функцию отношения правдоподобия

$$\Lambda[\Delta y(k)] = \frac{\left|K_{x(k)+\beta(k)}\right|^{1/2}}{\left|K_{x(k)+\alpha(k)}\right|^{1/2}} \exp\left\{ \frac{1}{2} \Delta y^{\mathrm{T}}(k) \times \right.$$

$$\left. \times \left[K^{-1}_{x(k)+\beta(k)} + K^{-1}_{x(k)+\alpha(k)} \right] \Delta y(k) \right\}.$$

Здесь $K_{x(k)+\beta(k)} = \tilde{K}_{x(k)} + K_{\beta(k)}$; $K_{x(k)+\alpha(k)} = \tilde{K}_{x(k)} + K_{\alpha(k)}$,
где ковариационная матрица $K_{x(k)}$ определяется с помощью следующего рекуррентного соотношения:

$$\tilde{K}_{x(k+1)} = A(k) K_{x(k)} A^{\mathrm{T}}(k) + G(k) K_{w(k+1)} G^{\mathrm{T}}(k).$$

Введем вектор $\hat{\Gamma}^k = \{\hat{\gamma}(0), \ldots, \hat{\gamma}(k)\}$ оценок случайных величин $\gamma(0), \ldots, \gamma(k)$.

Адаптивный алгоритм получения стабильного решения $\hat{x}(k)$ задачи фильтрации имеет следующий вид:

$$\bar{x}(k) = A(k-1)\hat{x}(k-1);$$

$$\bar{P}(k) = A(k-1)\bar{P}(k-1)A^{\mathrm{T}}(k-1) + G(k-1) K_{w(k-1)} G^{\mathrm{T}}(k-1),$$

$$\hat{x}(k) = \bar{x}(k) + K(\hat{\Gamma}^k, k)[y(k) - C(k)\bar{x}(k)],$$

$$K(\Gamma^k, k) = \bar{P}(k) C^{\mathrm{T}}(k) [C(k)\bar{P}(k) C^{\mathrm{T}}(k) +$$

$$+ \hat{\gamma}(k) K_{\alpha(k)} + (1 - \hat{\gamma}(k)) K_{P(k)}]^{-1}, \quad k = 1, 2, \ldots.$$

Начальные приближения задаются: $\hat{x}(0) = \bar{x}(0) = m_{x(0)}$, $\hat{P}(0) = K_{x(0)}$.

Как видно из приведенных соотношений, адаптивный фильтр Калмана состоит из двух фильтров, которые подключаются в соответствии с тем, какая из моделей шумов измерения классифицируется как истинная. Указанные фильтры формируются на основе однотипных алгоритмов, различающихся лишь способом выбора коэффициентов усиления $K(\hat{\Gamma}_k, k)$, которые меняются в зависимости от вида помехи.

Коэффициент c_k, который определяет значение порога в решающем правиле $\Lambda_k[\Delta y(k)] \geqslant c_k$, целесообразно определять для конкретной задачи с помощью имитационного моделирования процесса фильтрации. Можно предложить также другой подход [52], связанный с применением оптимальных (байесовских) решающих правил. В этом случае произвол, имеющийся в выборе c_k, переносится на выбор соответствующей матрицы потерь, реализовать который не менее (а подчас и более) сложно, чем определить непосредственно коэффициенты c_k.

Рассмотрим задачу построения адаптивного варианта фильтра Калмана в случае, когда ковариационные функции шумов системы и погрешности измерения

частично известны. Предположим, что дискретная линейная система описывается рекуррентными соотношениями

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{G}\mathbf{w}(k); \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{v}(k), \end{aligned} \tag{5.34}$$

где $\mathbf{x}(k)$ — n-мерный вектор состояния системы; $\mathbf{A}$ — переходная $(n \times n)$-матрица; $\mathbf{C}$ и $\mathbf{G}$ — матрицы размера $r \times n$ и $n \times s$; $k = 0, 1, \ldots$.

Предположим, что векторы $\mathbf{w}(k)$ и $\mathbf{v}(k)$ (s и r-мерные соответственно) представляют собой некоррелированные гауссовские шумы со следующими характеристиками:

$$\mathbf{M}[\mathbf{w}(k)] = 0, \ \mathbf{M}[\mathbf{w}(k)\,\mathbf{w}^{\mathrm{T}}(m)] = \mathbf{Q}\delta_{km};$$

$$\mathbf{M}[\mathbf{v}(k)] = 0, \ \mathbf{M}[\mathbf{v}(k)\,\mathbf{v}^{\mathrm{T}}(m)] = \mathbf{R}\delta_{km};$$

$$\mathbf{M}[\mathbf{w}(k)\,\mathbf{v}^{\mathrm{T}}(m)] = 0.$$

Здесь $\mathbf{Q}$ и $\mathbf{R}$ — положительно определенные (ковариационные) матрицы. Предположим также, что вектор начального состояния системы также гауссовский с нулевым математическим ожиданием и матрицей ковариации $\mathbf{P}_0$. Будем считать систему, описываемую соотношениями (5.34), наблюдаемой.

Алгоритм фильтрации Калмана (§ 5.4) существенным образом базируется на точном знании ковариационных матриц $\mathbf{Q}$ и $\mathbf{R}$. В то же время существует большое количество практических задач оценки состояния динамических систем, в которых указанные матрицы известны лишь приближенно или вообще не известны. Адаптивная модификация классического фильтра Калмана, предложенная Р. Мехрой [53], эффективно работает и при отсутствии информации о матрицах $\mathbf{Q}$ и $\mathbf{R}$. Как и в предыдущем алгоритме адаптивной фильтрации, процесс оценивания основывается на анализе свойств обновляющего процесса фильтрации, т. е. последовательности

$$\Delta\mathbf{y}(k) = \mathbf{y}(k) - \mathbf{C}\bar{\mathbf{x}}(k). \tag{5.35}$$

Обновляющий процесс интересен тем, что он содержит всю новую статистическую информацию, получаемую с помощью наблюдений $\mathbf{y}(k)$. С помощью обновляющего процесса можно определить оценки ковариационных матриц $\mathbf{Q}$ и $\mathbf{R}$ [53].

В случае достижения системой установившегося состояния коэффициент усиления фильтра $\mathbf{K}$, а также экстраполированная матрица ковариаций ошибки $\tilde{\mathbf{P}}$ не изменяются со временем:

$$\begin{aligned} \mathbf{K} &= \tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}}(\mathbf{C}\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}} + \mathbf{R})^{-1}, \\ \tilde{\mathbf{P}} &= \mathbf{A}(\mathbf{E} - \mathbf{K}\mathbf{C})\,\tilde{\mathbf{P}}\,(\mathbf{E} - \mathbf{K}\mathbf{C})^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} + \mathbf{A}\mathbf{K}\mathbf{R}\mathbf{K}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} + \mathbf{G}\mathbf{Q}\mathbf{G}. \end{aligned} \tag{5.36}$$

Выражения (5.36) можно использовать для расчета матрицы $\mathbf{P}$ как при оптимальном коэффициенте усиления, так и при субоптимальном коэффициенте, рассчитанном на основе приближений $\mathbf{Q}_0$, $\mathbf{R}_0$ ковариационных матриц $\mathbf{Q}$, $\mathbf{R}$.

Введем последовательность

$$\mathbf{C}_k = \mathbf{M}[\Delta\mathbf{y}(i)\,\Delta\mathbf{y}^{\mathrm{T}}(i-k)].$$

Используя определение обновляющего процесса, нетрудно получить следующие соотношения:

$$\mathbf{C}_0 = \mathbf{C}\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}} + \mathbf{R},$$

$$\mathbf{C}_k = \mathbf{C}[\mathbf{A}(\mathbf{E} - \mathbf{K}\mathbf{C})]^{k-1}\mathbf{A}[\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}} - \mathbf{K}\mathbf{C}_0], \quad k \geqslant 1.$$

Эти формулы в дальнейшем будут играть важную роль для получения оценок неизвестных ковариационных матриц $\mathbf{Q}$ и $\mathbf{R}$. Суть же самого алгоритма адаптивной фильтрации состоит в том, чтобы, используя на первом шаге априорные оценки $\mathbf{Q}_0$, $\mathbf{R}_0$ неизвестных ковариационных матриц, уточнить их в процессе фильтрации и в дальнейшем использовать уже уточненные оценки. Таким образом, весь процесс существенно зависит от возможности получить статистические оценки матриц $\mathbf{Q}$, $\mathbf{R}$ с требуемыми статистическими свойствами (например, состоятельные и асимптотические несмещенные).

Приведем вычислительную схему получения оценок ковариационных матриц $\mathbf{Q}$, $\mathbf{R}$:

1. Вычисление оценки величин $\mathbf{C}_k$ по формуле

$$\hat{\mathbf{C}}_k = \frac{1}{N}\sum_{i=k}^{N} \Delta\mathbf{y}(i)\,\Delta\mathbf{y}^{\mathrm{T}}(i-k),$$

где $N-k$ — число наблюдений.

2. Вычисление оценки произведения по следующей схеме. Имеем

$$\mathbf{C}_1 = \mathbf{C}\mathbf{A}\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}} - \mathbf{C}\mathbf{A}\mathbf{K}\mathbf{C}_0$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\mathbf{C}_n = \mathbf{C}\mathbf{A}^n\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}} - \mathbf{C}\mathbf{A}\mathbf{K}\mathbf{C}_{n-1} - \ldots - \mathbf{C}\mathbf{A}^n\mathbf{K}\mathbf{C}_0.$$

Следовательно,

$$\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}} = \mathbf{B}^{+}\left\|\begin{array}{c} \mathbf{C}_1 + \mathbf{C}\mathbf{A}\mathbf{K}\mathbf{C}_0 \\ \mathbf{C}_2 + \mathbf{C}\mathbf{A}\mathbf{K}\mathbf{C}_1 + \mathbf{C}\mathbf{A}^2\mathbf{K}\mathbf{C}_0 \\ \cdots\cdots\cdots\cdots\cdots \\ \mathbf{C}_n + \mathbf{C}\mathbf{A}\mathbf{K}\mathbf{C}_{n-1} + \ldots + \mathbf{C}\mathbf{A}^n\mathbf{K}\mathbf{C}_0 \end{array}\right\|.$$

Здесь $\mathbf{B}^{+}$ — псевдообращение матрицы

$$\mathbf{B} = \left\|\begin{array}{c} \mathbf{C} \\ \mathbf{C}\mathbf{A} \\ \vdots \\ \mathbf{C}\mathbf{A}^{n-1} \end{array}\right\|\mathbf{A}$$

(матрица $\mathbf{B}$ является произведением матрицы наблюдаемости и переходной матрицы $\mathbf{A}$, имеет ранг n). Тогда для оценки матрицы $\mathbf{P}\mathbf{C}^{\mathrm{T}}$ можно записать

$$\hat{\tilde{\mathbf{P}}}\mathbf{C}^{\mathrm{T}} = \mathbf{B} + \left\|\begin{array}{c} \hat{\mathbf{C}}_1 + \mathbf{C}\mathbf{A}\mathbf{K}\mathbf{C}_0 \\ \cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{C}}_n + \mathbf{C}\mathbf{A}\mathbf{K}\hat{\mathbf{C}}_{n-1} + \ldots + \mathbf{C}\mathbf{A}^n\mathbf{K}\hat{\mathbf{C}}_0 \end{array}\right\|.$$

3. Вычисление оценки ковариационной матрицы $\mathbf{R}$ по формуле $\widehat{\mathbf{R}}=\widehat{\mathbf{C}}_0-\mathbf{C}(\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}})$.

4. Вычисление оценки ковариационной матрицы $\mathbf{Q}$ путем решения следующей системы:

$$\sum_{j=0}^{k-1}\mathbf{C}\mathbf{A}^j\mathbf{G}\widehat{\mathbf{Q}}\mathbf{G}^{\mathrm{T}}(\mathbf{A}^{j-k})^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}=(\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{A}^{-k})\,\mathbf{C}^{\mathrm{T}}-$$

$$-\mathbf{C}\mathbf{A}^k\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}}-\sum_{j=0}^{k-1}\mathbf{C}\mathbf{A}^j\widehat{\mathbf{S}}(\mathbf{A}^{j-k})^{\mathrm{T}}\mathbf{C}^{\mathrm{T}},\quad k=1,\ldots,n;$$

$$\widehat{\mathbf{S}}=\mathbf{A}\left[-\mathbf{K}(\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}})^{\mathrm{T}}-\tilde{\mathbf{P}}\mathbf{C}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}+\mathbf{K}\mathbf{C}_0\mathbf{K}^{\mathrm{T}}\right]\mathbf{A}^{\mathrm{T}}.$$

Сделаем следующие замечания по поводу изложенного метода вычисления оценок $\widehat{\mathbf{Q}}$, $\widehat{\mathbf{R}}$: 1) величины $\widehat{\mathbf{C}}_k$ можно определять и рекурсивно по мере накопления новых точек с помощью формулы

$$\widehat{\mathbf{C}}_k^{N+1}=\widehat{\mathbf{C}}_k^{N}+\frac{1}{N}\left[\Delta\mathbf{y}(N+1)\,\Delta\mathbf{y}^{\mathrm{T}}(N+1-k)-\widehat{\mathbf{C}}_k^{N}\right],$$

где $\widehat{\mathbf{C}}_k^{l}$ — оценка величины $\mathbf{C}_k$ по l наблюдениям; 2) изложенный метод имеет ограничение — для его применения необходимо, чтобы число неизвестных элементов ковариационной матрицы $\mathbf{Q}$ было не больше $n\times r$, напомним, что $\mathbf{Q}$—$(q\times q)$-матрица.

Результаты по построению адаптивных алгоритмов фильтрации можно обобщить и на случай коррелированности шумов системы и погрешностей измерений [52, 53].

Глава 6.

Оптимизация непрерывных параметров систем управления

6.1. Постановка задач

В самом широком смысле общая задача оптимизации непрерывных параметров систем автоматического управления заключается в отыскании экстремума критерия (целевой функции) при заданных ограничениях в виде равенств и (или) неравенств, т. е. в решении задачи математического программирования [16, 18, 20, 29, 55].

Пусть некоторая действительная функция $F(\mathbf{x})$ представляет собой целевую функцию (критерий); функции $h_1(\mathbf{x})$, ..., $h_n(\mathbf{x})$ задают ограничения в виде равенств, а $g_{n+1}(\mathbf{x})$, ..., $g_p(\mathbf{x})$ — в виде неравенств, где $\mathbf{x}=[x_1, \ldots, x_m]^{\mathrm{T}}$ — вектор-столбец компонентов x_1, ..., x_m в m-мерном евклидовом пространстве $\mathbf{R}^m$.

Общая задача нелинейного программирования может быть сформулирована следующим образом: минимизировать функцию $F(\mathbf{x})$ $(\mathbf{x}\in\mathbf{R}^m)$

$$F(\mathbf{x})\to\min \tag{6.1}$$

при n ограничениях в виде равенств

$$h_j(\mathbf{x})=0,\ j=1,\ldots,n, \tag{6.2}$$

и $p-n$ ограничениях в виде неравенств

$$g_j(\mathbf{x}) \geqslant 0,\ j = n+1, \ldots, p. \tag{6.3}$$

Вектор $\mathbf{x}^* = [x_1^*, \ldots, x_m^*]^T$, удовлетворяющий соотношениям (6.2), (6.3) и минимизирующий функцию $F(\mathbf{x})$, называется *оптимальной точкой*, а соответствующее значение $F(\mathbf{x}^*)$ — *оптимальным значением целевой функции*. Пара $\mathbf{x}^*$ и $F(\mathbf{x}^*)$ составляет *оптимальное решение*. В дальнейшем без ограничения общности будем считать, что целью разработчика системы управления является минимизация некоторого критерия $F(\mathbf{x})$ (задача максимизации $F(\mathbf{x})$ сводится к минимизации критерия — $F(\mathbf{x})$).

Традиционно в математическом программировании выделяются следующие основные разделы:

линейное программирование — целевая функция линейная, а множество, на котором осуществляется поиск экстремума функции (критерия), задается системой линейных равенств и неравенств;

нелинейное программирование — нелинейные функции критерия и ограничения.

Нелинейное программирование принято подразделять на выпуклое — когда выпукла целевая функция и выпукло множество, на котором решается экстремальная задача, и квадратическое — когда целевая функция квадратична, а ограничения — линейные, равенства и неравенства.

В настоящей главе приведены алгоритмы и программы решения задач нелинейного программирования. Приводятся следующие алгоритмы:

1) оптимизации одномерного унимодального критерия;
2) оптимизации методом последовательного изменения переменных;
3) оптимизации методом прямого поиска (методом конфигураций);
4) оптимизации методом скользящего допуска;
5) оптимизации методом сопряженных направлений;
6) оптимизации градиентным методом;
7) штрафных и барьерных функций.

Алгоритмы 1—3, 5, 6 предназначены для решения задач оптимизации без ограничений на область существования оптимальных параметров (т. е. при $h_j(\mathbf{x}) \equiv 0$ и $q_i(\mathbf{x}) \geqslant 0$ для всех $\mathbf{x} \in \mathbf{R}^m$, $j = 1, \ldots, n$; $i = n+1, \ldots, p$). Алгоритмы 4, 6 решают общую задачу оптимизации с ограничениями. Приведенные в главе алгоритмы 1—4 относятся к так называемым поисковым, в которых направление минимизации определяется на основании последовательных вычислений функции критерия $F(\mathbf{x})$. Основным преимуществом методов поиска является то, что они не требуют регулярности и непрерывности целевой функции, а также существования производных. На практике поисковые алгоритмы часто оказываются достаточно эффективными и удобными для пользователя. Алгоритм 5 принадлежит к семейству алгоритмов, использующих факт существования у функции критерия $F(\mathbf{x})$ двух производных. Как правило, при решении задач нелинейного программирования методы, использующие производные, сходятся быстрее, чем прямые методы поиска. С другой стороны, проблемой использования методов оптимизации, основанных на вычислении первых и при необходимости вторых производных, является довольно большое время (по сравнению с методами поиска) на подготовку задачи к решению на ЭВМ.

При решении задач математического программирования необходимо уметь оценить эффективность работы того или иного алгоритма. Один из наиболее часто

используемых на практике критериев позволяет оценить скорость сходимости алгоритма к оптимальному решению.

Пусть $\mathbf{x}_0$ — начальная точка поиска, $\mathbf{x}$ — искомая точка экстремума. Пусть далее в результате работы некоторого алгоритма порождается последовательность точек $\alpha_1(\mathbf{x}_0), \ldots, \alpha_k(\mathbf{x}_0), \ldots$, сходящаяся к оптимальной точке $\mathbf{x}$. Пусть порядок сходимости p определяется отношением

$$p = \sup \left\{ q : \beta = \overline{\lim_{k \to \infty}} \frac{|\alpha_{k+1}(\mathbf{x}_0) - \mathbf{x}|}{|\alpha_k(\mathbf{x}_0) - \mathbf{x}|^q} < \infty \right\}.$$

Здесь $\lim\limits_{k \to \infty} y_k$ — верхний предел последовательности y_k; $\lim\limits_{k \to \infty} y_k = \lim\limits_{k \to \infty} \sup\{y_k, y_{k+1}, \ldots\}$;

β — коэффициент сходимости. Если $p=1$ и $\beta<1$, то алгоритм имеет линейную сходимость, если $p>1$ или $p=1$ и $\beta=0$, то сверхлинейную. В частности, если $p=2$, то имеет место квадратичная сходимость.

Скорость сходимости не может служить единственным критерием эффективности алгоритма. Другими важными факторами, влияющими на процесс оптимизации параметров систем, являются устойчивость, универсальность алгоритма, а также затраты, связанные с подготовкой задачи и ее решением на ЭВМ.

6.2. Минимизация одномерного унимодального критерия

Предположим, что подлежащая минимизации на отрезке $[a, b]$ $(a \leqslant b)$ одномерная действительная функция $F(x)$ унимодальная, т. е. для любых двух точек $x_1, x_2 \in [a, b]$

$$F(x_1) < F(x_2), \text{ если } \tilde{x} < x_1 < x_2,$$

$$F(x_1) > F(x_2), \text{ если } x_1 < x_2 < \tilde{x}.$$

Здесь $\tilde{x}$ — единственная точка положения минимума $F(x)$ на отрезке $[a, b]$. Из определения унимодальной функции следует, что справа и слева от положения ее минимума унимодальная функция только возрастает (рис. 6.1).

Унимодальность является достаточно распространенным свойством систем автоматического управления. При наличии априорной информации об унимодальности

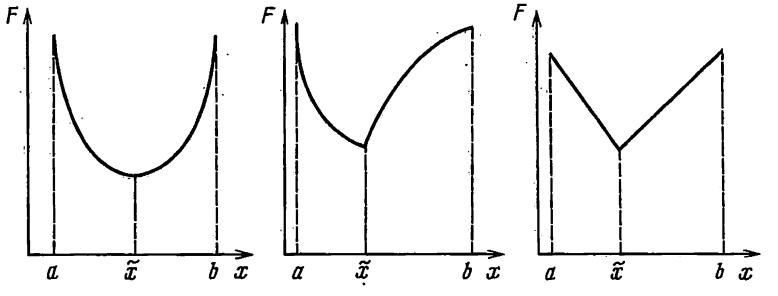


Рис. 6.1. Примеры унимодальных функций

оптимизируемого критерия появляется возможность построения эффективных методов поиска его экстремума. Одним из таких методов и является предлагаемый ниже, представляющий собой комбинацию методов золотого сечения и парабол.

В основе *метода золотого сечения* лежит схема Кифера [2, 58, 59]. Предположим, что в процессе работы алгоритма функция $F(x)$ вычисляется k раз ($k \geqslant 1$). Естественно стремление использовать информацию, полученную в процессе этих вычислений таким образом, чтобы максимально сузить интервал неопределенности положения точки экстремума.

Введем следующие обозначения: x_k — координата на отрезке $[a, b]$, вычисленная на k-м шаге работы алгоритма; x_k^* — положение минимального значения $F(x)$ к k-му шагу поиска; l_k — соответствующее значение из интервала неопределенности.

Разобьем пространство возможных исходов $(k+1)$-го шага на события Ω_1 и Ω_2:

$$\Omega_1 : F(x_{k+1}) > F(x_k^*), \quad \Omega_2 : F(x_{k+1}) \leqslant F(x_k^*), \quad \Omega = \Omega_1 \cup \Omega_2.$$

Теперь задачу построения траектории поиска $x_1, \ldots, x_k$ такой, чтобы максимально сузить интервал неопределенности l_{k+1}, можно сформулировать в виде следующей минимаксной задачи:

$$l_{k+1} = l_{k+1}(x_k, \Omega) \to \min_{x_k} \max_{\Omega = \{\Omega_1, \Omega_2\}}$$

Введем $v_k = h_{k+1} - a_k$, где a_k — координата левого конца интервала неопределенности l_k. Построим функцию $l_{k+1}(x_k, \Omega)$.

1. Если $x_k < z_k = x_k^* - a_k$, то

$$l_{k+1} = \begin{cases} l_k - v_k & \text{при} \quad \Omega = \Omega_1, \\ z_k & \text{при} \quad \Omega = \Omega_2. \end{cases}$$

2. Если $v_k > z_k$, то

$$l_{k+1} = \begin{cases} v_k & \text{при} \quad \Omega = \Omega_1, \\ l_k - z_k & \text{при} \quad \Omega = \Omega_2. \end{cases}$$

Теперь нетрудно определить как саму зависимость функции $\max\limits_{\Omega} l_{k+1}(v_k, \Omega)$, так и гарантированное значение интервала неопределенности $l_{k+1} = l_k - z_k$. Очередную точку поиска x_k следует располагать в пределах $z_k < x_k \leqslant l_k - z_k$. Обычно выбирают точку x_k, максимально удаленную от наилучшей уже отобранной точки x_k^*, полагая $x_k = l_k - z_k$. Последнее соотношение и определяет минимаксное правило Кифера: новую точку поиска следует располагать симметрично экстремальному относительно середины интервала неопределенности. Сам процесс поиска целиком зависит от выбора начальной точки поиска z_1.

При построении плана последовательности пробных точек $x_1, \ldots, x_k$ используем следующее рекуррентное соотношение:

$$l_k = l_{k+1} + l_{k+2}. \tag{6.4}$$

Теперь потребуем, чтобы отношение интервалов неопределенности на двух последующих шагах поиска было постоянным: $l_{k+1}/l_k=\tau=\text{const}(k=1, 2, \ldots)$. Из (6.4) имеем $\tau=1+l_{k+1}/l_k=1+1/\tau$, откуда находим $\tau=(1+\sqrt{5})/2\approx 1{,}618$. Точки экспериментов располагаются на интервалах неопределенности, соответствующих соотношению (6.4). Точка первого эксперимента определяется из соотношения $z_1=l_2=l_1/\tau$. Интервал неопределенности $l_k=l_1/\tau^{k-1}=l_1\tau^{1-R}(k\geqslant 2)$.

Таким образом, метод золотого сечения характеризуется экспоненциальной скоростью сходимости и состоит в построении последовательности интервалов $l_k=[a_k, b_k]$, стягивающихся к точке x^* минимума функции $F(\mathbf{x})$ на исходном отрезке $[a, b]$.

Приведем схему алгоритма золотого сечения, удобную для реализации на ЭВМ.

1. Выбираем отрезок $[a, b]$, содержащий точку минимума x^* функции $F(\mathbf{x})$, задаем ε — точность определения x^*, $l_1=b-a$, $\mu=1/\tau=(\sqrt{5}-1)/2$, $l_2=\mu l_1$, $l_3=l_1-l_2$, $r_1=a+l_3$, $p_1=b-l_3$, $k=2$.

Вычисляем величины $F(r_1)$, $F(p_1)$.

2. *а*) Если $F(r_{k-1})\leqslant F(p_{k-1})$, то

$$a_k=a_{k-1},\ b_k=p_{k-1};$$

$$l_{k+2}=l_k-l_{k+1};\ r_k=a_k+l_{k+2}.$$

Вычисляем $F(r_k)$. Переходим к п. 3.

б) Если $F(r_{k-1})>F(p_{k-1})$, то

$$a_k=r_{k-1},\ b_k=b_{k-1},$$

$$l_{k+2}=l_k-l_{k-1},\ p_k=b_k-l_{k+2},\ r_k=p_k-1.$$

Вычисляем $F(p_k)$, переходим к п. 3.

3. Если $\Delta_k\leqslant\varepsilon$, то $x^*=x_k=\operatorname{argmin}\{F(y_k), F(z_k)\}$. Если $\Delta_k\geqslant\varepsilon$, то полагаем $k=k+1$ и переходим к п. 2.

Отметим, что на каждом шаге алгоритма, за исключением первого, производится только одно вычисление функции $F(\mathbf{x})$.

Одним из недостатков алгоритма золотого сечения является то, что он не использует возможной дифференцируемости функции $F(x)$, встречающейся достаточно часто в приложениях. При наличии непрерывной производной $F(x)$ возможна модернизация алгоритма — его усиление с помощью процедуры последовательной параболической интерполяции.

Метод парабол целесообразно применять после того, как найдены опорные решения x_k, x_{k+1}, x_{k+2} (например, в результате $k+2$ шагов алгоритма золотого сечения). Без ограничения общности можно предположить, что $x_k\leqslant x_{k+1}\leqslant x_{k+2}$. Метод парабол основывается на следующем факте теории интерполяции: многочлен второго порядка, график которого проходит через точки $(x_k, F(x_k))$, $(x_{k+1}, F(x_{k+1}))$, $(x_{k+2}, F(x_{k+2}))$, достигает минимального значения в точке

$$\bar{x}=x_{k+1}-\frac{1}{2}\,\frac{(x_{k+1}-x_k)^2[F(x_{k+1})-F(x_{k+2})]-(x_{k-2}-x_{k+1})^2[F(x_{k+1})-F(x_k)]}{(x_{k+1}-x_k)[F(x_{k+1})-F(x_{k+2})]-(x_{k+2}-x_{k+1})[F(x_{k+1})-F(x_k)]}.$$

При этом $|\bar{x}-x_{k+1}|\leqslant 0{,}5\max\{x_{k+1}-x,\ x_{k+2}-x_{k+1}\}$, и для достаточно хороших начальных приближений итерации метода парабол сходятся со скоростью порядка $p=1{,}324$, если в точке минимума $F''(x^*)>0$ (т. е. $F'(x)$ имеет простой нуль в точке x^*).

ПРОГРАММА FMINIM

Назначение: оптимизация по методу золотого сечения. Эта программа эквивалентна алгоритму Брента, записанному на языке Алгол-60 [2].

Параметры:

AX, BX — левый и правый концы исходного интервала поиска, описываются с атрибутами BINARY FLOAT;

F — процедура-функция, вычисляющая значения функции критерия;

TOL — интервал неопределенности конечного результата, описывается с атрибутом BINARY FLOAT.

FMINIM — значение процедур-функций, стандартное обращение: Z=FMINIM(AX, BX, F, TOL);

Основные этапы работы:

1) вычисление коэффициента сужения зоны поиска экстремума;

2) определение EPS — наименьшего положительного числа, представимого на данной ЭВМ (определение машинной точности);

3) задание начальных значений интервала неопределенности;

4) задание характеристик точности — начало основного цикла процедуры поиска;

5) проверка условия окончания работы;

6) проверка необходимости применения метода золотого сечения;

7) построение параболы и проверка ее приемлемости;

8) параболическая интерполяция;

9) проверка близости к концам отрезка поиска;

10) осуществление шага золотого сечения;

11) проверка близости к точке x;

12) задание новых значений параметров.

Пример. Минимизация функции Валлиса $F(x)=x(x^2-2)-5$.

Исходные данные: A=AX=0, B=BX=1, TOL=10^{-5}.

Точное значение минимума z=0,8165 получено за 0,9 с на ЭВМ ЕС-1050.

```
     FMINIM:  PROCEDURE(AX,BX,F,TOL);
     DECLARE
          (AX,BX,TOL,A,B,C,D,E,EPS,XM,P,Q,R,
           TOL1,TOL2,U,V,W,FU,FV,FW,FX,X)
                 BINARY FLOAT;
     DECLARE
          F ENTRY;
/* 1 */
          C=0.5*(3.-SQRT(5.0));

/* 2 */
          EPS=1.00;
  MET:    EPS=EPS/2.00; TOL1=1.0+EPS;
          IF  TOL1> 1.00  THEN GO TO MET1;
          EPS=SQRT(EPS);
```

```
/* 3 */
        A=AX;       B=BX;    V=A+C*(B-A);
        W=V;        X=V;     E=0.0;
        FX=F(X);    FV=FX;   FW=FX;
/* 4 */
  MET2: XM=0.5*(A+B);
        TOL1=EPS*ABS(X)+TOL/3.0;
        TOL2=2.0*TOL1;
/* 5 */
        IF  ABS(X-XM) <= (TOL2-0.5*(B-A))
        THEN  GO TO MET9;
/* 6 */
        IF  ABS(E)  <=  TOL1  GO TO MET4;
/* 7 */
        R=(X-W)*(FX-FV);     Q=(X-V)*(FX-FW);
        P=(X-V)*Q-(X-W0*R;   Q=2.00*(Q-R);
        IF  Q >  0.0 THEN    P=-P;
        Q=ABS(Q);
        R=E;  E=D;
  MET3: IF  ABS(P)  >=  ABS(0.5*Q*R)
        THEN  GO TO MET4;
        IF  P  <=  Q*(A-X)
        THEN  GO TO MET4;
        IF  P  >=  Q*(B-X)
        THEN  GO TO MET4;
/* 8 */
        D=P/Q;  U=X+D;
/* 9 */
        IF  U-A <  TOL2  THEN
        D=SIGN(XM-X)*ABS(TOL1);
        IF  B-U <  TOL2  THEN
        D=SIGN(XM-X)*ABS(TOL1);
        GO TO MET5;
/* 10 */
  MET4: IF  X  >=  XM  THEN  E=A-X;
         IF  X  <   XM  THEN  E=B-X;
        D=C*E;
  MET5: IF  ABS(D)  >= TOL1  THEN  U=X+D;
        IF  ABS(D)  <  TOL1  THEN
        U=X+SIGN(D)*ABS(TOL1);
        FU=F(U);
        IF  FU  >  FX  THEN  GO TO MET6;
        IF  U  >=  X   THEN  A=X;
        IF  U  <   X   THEN  B=X;
        V=W;  FV=FW;
        W=X;  FW=FX;
        X=U;  FX=FU;
        GO TO MET2;
  MET6: IF  U <  X  THEN A=U;
        IF  U >= X  THEN B=U;
        IF  FU <= FW  THEN GO TO MET7;
        IF  W   = X   THEN GO TO MET7;
        IF  FU <= FV  THEN GO TO MET8;
        IF  V   = X   THEN GO TO MET8;
        IF  V   = W   THEN GO TO MET8;
        GO TO MET2;
  MET7: V=W;  FV=FW;
        W=U;  FW=FU;
        GO TO MET2;
  MET8: V=U;  FV=FU;
        GO TO ME2;
```

```
MET9: RETURN(X);
  END   FMINIM;

F:      PROCEDURE(X);
        F1=X*(X**2-2.)-5.;
        RETURN(F1);
        END F;

TEST:   PROCEDURE OPTIONS(MAIN);
        DECLARE (FMINIM,F) ENTRY;
        DECLARE (A,B,TOL) BINARY FLOAT;
        A=0;  B=1;
        TOL=1.E-05;
        Z=FMINIM(A,B,F,TOL);
        PUT SKIP DATA(Z);
        END TEST;
```

6.3. Метод последовательного изменения переменных

При оптимизации по методу последовательного изменения переменных (покоординатного спуска) избирается траектория поиска экстремума в виде ломаной линии, отдельные отрезки которой параллельны координатным осям пространства оптимизируемых параметров R^m. Суть его состоит в следующем.

После выбора некоторого начального приближения $x^0=\{x_1^0, \ldots, x_m^0\}^T$ изменяется каким-либо способом компонент x_1 при постоянных значениях остальных компонентов $x_2=x_2^0, \ldots, x_m=x_m^0$. Таким образом, движение из точки x_0 осуществляется по прямой, параллельной оси Ox_1, в сторону убывания функции $F(x)$. Аргумент x_1 изменяется до тех пор, пока функция одной переменной $F(x_1, x_2^0, \ldots, x_m^0)$ не достигнет минимума в некоторой точке x_1^1. После этого поиск продолжается из точки $x^1=\{x_1^1, x_2^0, \ldots, x_m^0\}$ вдоль оси Ox_2 аналогично вышеизложенному. Описанная процедура последовательно повторяется для всех x_i $(i=3, \ldots, m)$. Таким образом, траектория поиска состоит из последовательности отрезков, параллельных координатным осям. По завершении поиска по всем m переменным цикл повторяется, т. е. вновь изменяется аргумент x_1^1 и т. д. Итерационный процесс поиска заканчивается, когда изменение аргумента перестает приводить к изменению целевой функции.

Как видно из описания, метод поочередного изменения переменных является естественным применением одномерного поиска к решению задачи многомерной оптимизации и эффективность его существенно зависит от того, какой метод используется при проведении одномерных оптимизаций каждого компонента вектора параметров функции F. В принципе для этих целей можно использовать любой достаточно эффективный метод однопараметрической параметризации. Например, при условии существования производных довольно часто применяется метод наискорейшего спуска [58, 60]. Соответствующий вариант метода последовательного изменения переменных обычно называют алгоритмом Гаусса—Зейделя.

Метод последовательного изменения координат имеет ряд достоинств — он достаточно прост для практического использования, а кроме того, инвариантен относительно выбора масштабов по каждой координатной оси. К недостатку метода следует отнести зависимость эффективности поиска от расположения поверхностей уровня $F(x)$ относительно осей системы координат $Ox_1, \ldots, x_m$. В частности, метод последовательного изменения координат следует признать неэффектив-

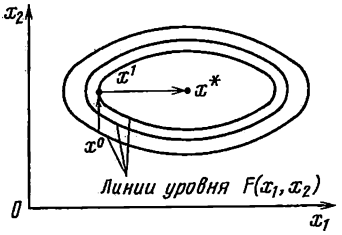


Рис. 6.2. Траектория поиска при минимизации двумерного эллиптического объекта

ным для решения задач оптимизации, в которых множество значений целевой функции имеет структуру типа поверхности с гребнем.

Наиболее целесообразно применение метода последовательного изменения координат в задачах, когда функция F имеет вид (сепарабельная функция)

$$F(x) = F^* + \sum_{i=1}^{m} F_i(x_i - x_i^*), \qquad (6.5)$$

где F_i $(i=1, \ldots, m)$ — унимодальные функции с минимумом в начале координат $F_i(0) \leqslant F_i(y)$ $(i=1, \ldots, m)$. В этом случае отсутствие перекрестных связей между параметрами $x_1, \ldots, x_m$ позволяет в процессе оптимизации лишь однажды обращаться к каждому параметру. Действительно, для сепарабельной функции (6.5) положение экстремума по каждому параметру x_i $(i=1, \ldots, m)$ не зависит от значений остальных параметров. Поэтому достаточно сделать только m спусков, причем окончательный результат поиска не зависит от порядка проведения покоординатных оптимизаций.

Траектория поиска с помощью метода последовательного изменения координат при решении задачи минимизации функции эллиптического типа

$$F(x_1, x_2) = F^* + c_1(x_1 - x_1^*)^2 + c_2(x_2 - x_2^*)^2,$$

где F^*, c_1, c_2 — некоторые константы, показана на рис. 6.2. Из рисунка видно, что минимизация функции $F(x_1, x_2)$ производится за две итерации.

6.4. Минимизация методом прямого поиска (конфигураций)

Методы последовательного изменения переменных неэффективны, если имеет место связь между отдельными переменными. Для устранения этого недостатка в [58] предложена логически простая стратегия поиска экстремума, использующая априорные сведения в процессе поиска и в то же время отвергающая устаревшую информацию относительно структуры оптимизируемого критерия. Алгоритм состоит из следующих основных двух этапов: этапа исследующего поиска вокруг базисной точки и этапа поиска в выбранном направлении.

На первом этапе сначала задаются начальные значения оптимизируемого вектора $\mathbf{x}^0$ и вектора приращений $\Delta\mathbf{x}^0$, вычисляется $F(\mathbf{x}^0)$. Затем в циклическом порядке изменяется каждая переменная (каждый раз только одна) на определенное значение, пока все параметры не будут таким образом изменены. Например, x_i изменяется на $x_i + \Delta x_i^0$, и если приращение не улучшает значение критерия, то x_i заменяется на $x_i - \Delta x_i$. Если и в этом случае $F(\mathbf{x})$ не улучшается, то x_i оставляют без изменений. На каждом шаге циклического изменения переменных значение критерия сравнивается с его значением в предыдущей точке. Если целевая функция улучшается на данном шаге, то ее старое значение заменяется новым при последующих сравнениях. В результате на первом этапе вычисляется новая точка $\mathbf{x}$, являющаяся базисной для дальнейшего поиска.

На этапе поиска в выбранном направлении вектор удачных изменений переменных $\Delta\bar{x}=\bar{x}-x^0$ определяет направление минимизации, которое может привести к успеху. Серия ускоряющихся шагов проводится вдоль этого вектора до тех пор, пока $F(x)$ уменьшается. Размер шага при поиске вдоль выбранного направления по данной координате определяется пропорционально числу удачных шагов, совершенных в этом координатном направлении за время предыдущих циклов поиска. Для ускорения процесса оптимизации приращение Δx изменяется путем введения некоторого множителя μ при Δx:

$$x_i^{k+1}=x_i^k+\mu(x_i^k-x_i^0),\ i=1,\ldots,m,$$

где x_i^{k+1}, x^k — точки траектории поиска при k—й и $(k+1)$-й последовательных итерациях.

ПРОГРАММА DIRECT

Назначение: оптимизация по методу прямого поиска. Программа получена из процедуры DIRECT SEARCH [59], разработанной на языке Алгол-60, с помощью модификаций и следующего уточнения. В программу DIRECT введена процедура вычисления параметра MIN — наименьшего числа в данной вычислительной системе (MIN — машинная точность), который в процедуре DIRECT SEARCH задается самим пользователем.

Входные параметры:

S — процедура-функция, вычисляющая значения минимизируемого критерия; заголовок процедуры: PROCEDURE(PHI), где PHI — вектор аргументов;

K — размерность вектора оптимизируемых параметров;

D — доля начальных значений аргументов, используемая как начальный размер шага, на выходе процедуры D — конечный размер шага;

RHO — множитель, уменьшающий размер шага;

DELTA — минимально допустимый размер шага (процедура заканчивается, если размер шага становится меньше DELTA);

MAX — максимально допустимое число вычислений критерия, на выходе из процедуры MAX это фактически затраченное число вычислений критерия;

PSI — одномерный массив из K элементов; на входе в процедуру это начальная точка поиска, на выходе — это найденная точка минимума; PSI описывается с атрибутами BINARY FLOAT;

SPSI — значение критерия в точке минимума.

Всем переменным (кроме PSI) атрибуты присваиваются по умолчанию.

Обращение: CALL DIRECT(K, PHO, DELTA, S, PSI, D, MAX, SPSI);

Основные этапы работы:

1) вычисление MIN;

2) вычисление S1 — вектора начальных приращений исходной точки поиска;

3) проведение поиска по образцу;

4) проверка основного условия окончания поиска минимума процедурой DIRECT: превышает ли текущий шаг изменения переменных заданный уровень DELTA?

5) проведение исследующего поиска.

Пример. Задача поиска минимума функции $F(x_1, x_2, x_3)=(x_1-0{,}01)^2+(x_2-1)^2+(x_3-100)^2$.

Исходные данные: K=3, RHO=0.1, DELTA=0,001, D=0,2, PSI=(0,05, 5, 500), MAX=100.

Точное значение минимума PSI=(0,01, 1, 100) получено на ЭВМ ЕС-1050 за 4,2 с.

```
DIRECT: PROCEDURE(K,RHO,DELTA,S,PSI,D,MAX,SPSI);
        DECLARE PSI(*), BINARY FLOAT,
        (MIN,TOL1) BINARY FLOAT,
        S  ENTRY;
        BEGIN;
        DECLARE (S1,PHI) (K) BINARY FLOAT;
/* 1 */
        MIN=1.00;
ME:     MIN=MIN/2.00;  TOL1=1.0+MIN;
        IF  TOL1 >  1.00  THEN  GO TO ME;
        MIN=SQRT(MIN);
/* 2 */
START:  DO I=1 TO K;
        S1(I)=D*ABS(PSI(I));
        IF S1(I)  <  MIN  THEN  S1(I)=D;
        END;
        SPSI=S(PSI);  EVAL=1;
/* 3 */
MET1:   SS=SPSI;
        DO  I=1  TO K;
        PHI(I)=PSI(I);
        END;
        LIND=1;  GO TO EE;
MET2:   IF  SS <  SPSI  THEN
        DO;
MET3:   DO  I=1  TO  K;
        IF (PHI(I)  <  PSI(I))  THEN IND=1;
        ELSE  IND=0;
        IF  (S1(I)  <  0)  THEN  IND1=1;
        ELSE  IND1=0;
        IF  IND=IND1  THEN  S1(I)=-S1(I);
        THETA=PSI(I);  PSI(I)=PHI(I);
        PHI(I)=2.0*PHI(I)-THETA;
        END;
        SPSI=SS;  IF  <  MAX  THEN  EVAL=EVAL+1;
        ELSE  GO  TO  EXIT;
        SPSI=S(PHI);  SS=SPHI;
        LIND=2;  GO TO EE;
MET4:   IF  SS  >=  SPSI THEN GO TO MET1;
        DO  I=1  TO K;
        IF  ABS(PHI(I)-PSI(I)) > 0.5*ABS(S1(I))
        THEN GO TO MET3;
        END;
/* 4 */
        IF  D  >=  DELTA  THEN
        DO;
        IF EVAL  >  MAX  THEN  GO TO EXIT;
        D=RHO*D;
        DO I=1  TO  K;
        S1(I)=RHO*S1;  GO  TO  MET1;
        END;
        END;
/* 5 */
EE:     DO  I=1  TO  K;
        PHI(I)=PHI(I)+S1(I);
        SPHI=S(PHI);  EVAL=EVAL+1;
```

```
            IF  SPHI  <  SS  THEN  SS=SPHI;
            ELSE  DO;
            S1(I)=-S1(I);  PHI(I)=PHI(I)+2.0*S1(I);
            IF  EVAL  <  MAX  THEN  EVAL=EVAL+1;
            ELSE  GO TO  EXIT;  SHSI=S(PHI);
            IF  SPHI  <  SS  THEN SS=SPHI;
            ELSE  PHI(I)=PHI(I)-S1(I);
            END;
            END;
            IF  LIND=1  THEN  GO TO  MET2;
            IF  LIND=2  THEN  GO TO  MET4;
            END;
EXIT:       MAX=EVAL;
            END DIRECT;
/* 6 */
```

6.5. Метод скользящего допуска

Метод скользящего допуска предназначен для решения общей задачи нелинейного программирования — задачи минимизации нелинейного критерия (целевой функции) $F(\mathbf{x})$ на области допустимых значений аргумента. Последняя задается системой ограничений (6.2), (6.3), в которой функции $h_i(\mathbf{x})$, $g_j(\mathbf{x})$ ($i=1, \ldots, n$; $j=n+1, \ldots, p$) могут быть как линейными, так и нелинейными. Метод скользящего допуска позволяет улучшить значения целевой функции за счет информации, полученной как в допустимых точках $\mathbf{R}^m$, так и при просмотре точек, лежащих вне области допустимости, но близких в определенном смысле к допустимым. Множество почти допустимых решений в ходе решения постепенно сужается и стягивается к области строго допустимых решений, удовлетворяющих соотношениям (6.2), (6.3).

Метод скользящего допуска основан на замене системы ограничений (6.2), (6.3) единственным ограничением

$$S^{(k)} - T(\mathbf{x}) \geqslant 0. \tag{6.6}$$

Здесь $S^{(k)}$ — значение критерия скользящего допуска на k-м этапе поиска:

$$S^{(k)} = \min \left\{ S^{(k-1)}, \frac{n+1}{r+1} \sum_{i=1}^{r+1} \left\| x_i^{(k)} - x_{r+2}^{(k)} \right\|_m \right\}, \quad k \geqslant 1, \qquad S^{(0)} = 2t(n+1),$$

где n — число ограничений в виде равенств; t — величина, характеризующая размеры исходной зоны (многогранника) поиска; $r=m-n$ — число степеней свободы критерия; $x_i^{(k)}$ — точка, задающая положение i-й вершины многогранника, процедуры поиска; $x_{r+2}^{(k)}$ — точка, задающая положение $(r+2)$-й вершины многогранника, соответствующей его центру тяжести; $\|\cdot\|$ — символ нормы в евклидовом пространстве $\mathbf{R}^m$. Оценить степень удовлетворения заданной системе ограничений (6.2), (6.3) позволяет функционал

$$T(\mathbf{x}) = \left[\sum_{i=1}^{n} h_i^2(\mathbf{x}) + \sum_{i=m+1}^{p} u_i g_i^2(\mathbf{x}) \right]^{1/2}$$

где u_i — оператор Хэвисайда: $u_i=0$ при $g_i(\mathbf{x}) \geqslant 0$ и $u_i=1$ при $g_i(\mathbf{x}) < 0$. Из определения функции $T(\mathbf{x})$ легко видеть, что если $T(\mathbf{x})=0$, то точка $\mathbf{x}$ является допустимой. Чтобы учесть случай, когда значения $T(\mathbf{x})$ малы (что соответствует случаю близости

x к области допустимости), вводятся следующие определения: точка x является допустимой, если $T(\mathbf{x})=0$, почти допустимой (квазидопустимой), если $T(\mathbf{x}) \leqslant S^{(k)}$, недопустимой, если $T(\mathbf{x}) > S^{(k)}$.

Общая схема поиска с помощью метода скользящего допуска строится на основе следующего принципа: по мере проведения поиска уменьшается значение $S^{(k)}$, что приводит к сужению области квазидопустимости. При этом процедуры минимизации $F(\mathbf{x})$ и поиска квазидопустимых точек отделены друг от друга. При заданном $S^{(k)}$ в точке $\mathbf{x}^{k+1}$ имеет место один из следующих вариантов:

$T(\mathbf{x}^{(k+1)}) \leqslant S^{(k)}$ — точка $\mathbf{x}^{(k+1)}$ может служить очередной точкой процедуры минимизации $F(\mathbf{x})$;

$T(\mathbf{x}^{(k+1)}) > S^{(k)}$ — точка $\mathbf{x}^{(k+1)}$ недопустима и значения $T(\mathbf{x}^{(k+1)})$ уменьшают до тех пор, пока не будет выполнено условие $T(\mathbf{x}^{(k+1)}) \leqslant S^{(k)}$.

Таким образом в процессе оптимизации для улучшения значений $F(\mathbf{x})$ используются только допустимые или квазидопустимые точки x. Отыскание допустимых или квазидопустимых точек осуществляется при минимизации функции $T(\mathbf{x})$ на множестве всех точек множества поиска S_n до наступления события $T(\mathbf{x}) \leqslant S^{(k)}$ (k — номер последнего шага алгоритма, на котором произведено улучшение x).

Задачи минимизации $F(\mathbf{x})$ при наличии ограничений (6.2), (6.3), (6.5) эквивалентны. Действительно, так как $S^{(k)}=0$ тогда, и только тогда, когда дальнейшее улучшение значений $F(\mathbf{x})$ при наличии ограничения (6.5) невозможно, то в пределе $S^*=\lim\limits_{k\to\infty} S^{(k)}=0$ и условию $S^* \geqslant T(\mathbf{x}) \geqslant 0$ могут удовлетворять только допустимые точки x: $\{\mathbf{x}: h_i(\mathbf{x})=0,\ g_i(\mathbf{x}) \geqslant 0,\ i=1, \ldots, n;\ j=n+1, \ldots, p\}$, т. е. те точки, для которых $T=0$.

Одним из преимуществ метода скользящего допуска является то, что степень нарушения ограничений (6.2), (6.3) по мере приближения к искомому решению постепенно уменьшается. Другое преимущество заключается в том, что в процессе поиска функцию $S^{(k)}$ удобно использовать в качестве критерия окончания поиска — поиск продолжается до тех пор, пока $S^{(k)}$ не станет меньше некоторого наперед заданного числа ε.

В методе скользящего допуска в процессе поиска квазидопустимых точек и улучшения значений критерия $F(\mathbf{x})$ для допустимых использовался алгоритм Нелдера—Мида — алгоритм безусловной оптимизации методом деформируемого многогранника [58], который оказывается исключительно эффективным при решении поисковых задач безусловной оптимизации и легко реализуемым на ЭВМ. Здесь функция $F(\mathbf{x})$ m-мерного вектора x минимизируется с помощью деформации специально подобранного (имеющего $m+1$ вершину) многогранника в $\mathbf{R}^m$. Вершина многогранника, в которой значение $F(\mathbf{x})$ максимально, проецируется через центр тяжести оставшихся вершин. Улучшение значений $F(\mathbf{x})$ производится последовательной заменой точки с максимальным значением $F(\mathbf{x})$ на лучшие с помощью поиска вдоль проецирующей линии. Основные этапы работы алгоритма Нелдера—Мида следующие.

Пусть $x_i^{(k)}$ ($i=1, \ldots, m+1$) — вершины деформируемого многогранника на k-м этапе поиска ($k=0, 1, \ldots$). Введем

$$\mathbf{x}_h^{(k)}: F(\mathbf{x}_h^{(k)}) = \max\{F(\mathbf{x}_1^{(k)}), \ldots, F(\mathbf{x}_{m+1}^{(k)})\},$$

$$x_l^{(k)}: F\left(\mathbf{x}_l^{(k)}\right) = \min\left\{F\left(\mathbf{x}_1^{(k)}\right), \ldots, F\left(\mathbf{x}_{m+1}^{(k)}\right)\right\}.$$

Пусть $\mathbf{x}_{m+2}$ — точка центра тяжести всех вершин многогранника, исключая $\mathbf{x}_h^{(k)}$. Тогда

$$\mathbf{x}_{m+2}^{(k)} = \frac{1}{m}\left[\left(\sum_{i=1}^{m+1} x_i^{(k)}\right) - x_h^{(k)}\right].$$

Процедура улучшения значений $F(\mathbf{x})$ состоит из следующих основных операций.

1. Отражение — проектирование «наихудшей» точки $\mathbf{x}_h^{(k)}$ через центр тяжести $\mathbf{x}_{m+2}^{(k)}$, что позволяет получить точку

$$\mathbf{x}_{m+3}^{(k)} = \mathbf{x}_{m+2}^{(k)} + \alpha\left(\mathbf{x}_{m+2}^{(k)} - \mathbf{x}_h^{(k)}\right),$$

где $\alpha(\alpha > 0)$ — коэффициент отражения.

2. Растяжение — дальнейшее продвижение вдоль направления $\mathbf{x}_{m+3}^{(k)} - \mathbf{x}_{m+2}^{(k)}$, если оно является перспективным, т. е. если $F(\mathbf{x}_{m+3}^{(k)}) \leqslant F(\mathbf{x}_l^{(k)})$:

$$\mathbf{x}_{m+4}^{(k)} = \mathbf{x}_{m+2}^{(k)} + \gamma\left(\mathbf{x}_{m+3}^{(k)} - \mathbf{x}_{m+2}^{(k)}\right),$$

где γ $(\gamma > 1)$ — коэффициент растяжения. Если $F(\mathbf{x}_{m+4}^{(k)}) < F(x_l^{(k)})$, то $\mathbf{x}_l^{(k)}$ заменяется на $\mathbf{x}_{m+4}^{(k)}$, k на $k+1$, производится переход к операции 1 (отражения). Если $F(\mathbf{x}_{m+4}^{(k)}) \geqslant F(\mathbf{x}_l^{(k)})$, то $\mathbf{x}_l^{(k)}$ заменяется на $\mathbf{x}_{m+3}^{(k)}$, k — на $k+1$ и также осуществляется переход к операции 1.

3. Сжатие — если $F(\mathbf{x}_{m+3}^{(k)}) > F(\mathbf{x}_l^{(k)})$ для всех $i \neq h$, то производится сжатие вектора $\mathbf{x}_h^{(k)} - \mathbf{x}_{m+2}^{(k)}$ в соответствии со следующей формулой:

$$\mathbf{x}_{m+5}^{(k)} = \mathbf{x}_{m+2}^{(k)} + \beta\left(\mathbf{x}_h^{(k)} - \mathbf{x}_{m+2}^{(k)}\right),$$

где β $(0 < \beta < 1)$ — коэффициент сжатия. После этого $\mathbf{x}_h^{(k)}$ заменяем на $\mathbf{x}_{m+5,k}^{(k)}$ — на $k+1$ и возвращаемся к операции 1.

4. Редукция — в оставшемся из всех логически возможных случаев $F(\mathbf{x}_{m+3}^{(k)}) > F(\mathbf{x}_h^{(k)})$ все векторы $\mathbf{x}_i^{(k)} - \mathbf{x}_l^{(k)}$ $(i = 1, \ldots, m+1)$ уменьшаются:

$$\mathbf{x}_i^{(k)} = \mathbf{x}_l^{(k)} + \xi\left(\mathbf{x}_i^{(k)} - \mathbf{x}_l^{(k)}\right),$$

где ξ — коэффициент редукции (можно принять его равным $1/2$), затем k заменяется на $k+1$ и производится возврат к операции 1.

Деформируемый от итерации к итерации многогранник $\{\mathbf{x}_i^{(k)}\}$ $(i = 1, \ldots, m+1)$ в процессе поиска адаптируется к поведению целевой функции, вытягиваясь вдоль длинных склонов, изменяя направление в изогнутых впадинах и сжимаясь в окрестности минимума $P(\mathbf{x})$. Критерием завершения поиска служит условие

$$\left\{\frac{1}{m+1}\sum_{i=1}^{m+1}\left[F\left(\mathbf{x}_i^{(k)}\right) - F\left(\mathbf{x}_{m+2}^{(k)}\right)\right]^2\right\} \leqslant \varepsilon,$$

где ε — наперед заданная точность.

В описанном выше алгоритме Нелдера — Мида фигурируют определенные точно коэффициенты отражения α, растяжения ν и сжатия β. Назначение этих коэффи-

циентов — масштабирование размера, формы деформируемого многогранника не изменяются в процессе поиска. Исследование влияния на процедуру поиска выбора коэффициентов α, ν, β, проведенное с помощью решения группы тестовых задач при различных комбинациях значений коэффициентов [58], позволили рекомендовать следующие значения коэффициентов: $\alpha=1$, $\nu=2$, $\beta=0{,}5$.

В алгоритме скользящего допуска можно применить и любой другой метод определения безусловного минимума, если при этом гарантируется его эффективность.

ПРОГРАММА FLEXI

Назначение: вычисление минимума функций многих переменных при наличии ограничений методом скользящего допуска. Программа получена из программы ФЛЕКСИПЛЕКС [58], разработанной на языке Фортран-IV с помощью модификаций и следующего уточнения: отсутствие условия (|*1*|) и оператора № 230 (|*2*|) в процедуре FEASBL вызвало осцилляцию исходной программы в окрестности искомой точки и тем самым сделало невозможным нахождение достаточно точного решения программой ФЛЕКСИПЛЕКС.

Параметры:

NX — общее число аргументов минимизируемой функции;

NC — общее число ограничений в виде неравенств;

NIC — общее число ограничений в виде равенств;

SIZE — размер деформируемого многогранника (подробнее о выборе параметра SIZE см. ниже);

CONVER — точность, с которой производятся вычисления;

X — вектор аргументов минимизируемой функции размерности NX.

Если верхний и нижний пределы вектора X известны, то значение параметра SIZE выбирается в соответствии со следующими принципами:

SIZE$\approx$20 % — разности между верхним и нижним пределами возможных значений X, если ожидаемые интервалы изменения x_i вдоль каждой оси координат приблизительно равны:

если ожидаемые интервалы различны, то параметру SIZE присваивается значение, равное наименьшей разности между соответствующими верхним и нижним пределами любого x_i. Например,

$$-11 \leqslant x_1 \leqslant 98{,}7 \qquad 0 \leqslant x_1 \leqslant 400$$
$$-10 \leqslant x_2 \leqslant 100{,}1 \qquad 0 \leqslant x_2 \leqslant 1000$$
$$-9{,}5 \leqslant x_3 \leqslant 101 \qquad 340 \leqslant x_3 \leqslant 420$$
$$-10{,}2 \leqslant x_4 \leqslant 99{,}5 \qquad 340 \leqslant x_4 \leqslant 420$$
$$-9{,}8 \leqslant x_5 \leqslant 100 \qquad -1000 \leqslant x_5 \leqslant 1000$$
$$\text{SIZE} \approx 0{,}2*100 = 22 \qquad 0 \leqslant x_6 \leqslant 0{,}5236$$
$$\text{SIZE} \approx 1.$$

Атрибуты всех параметров определяются по умолчанию. Входные параметры вводятся с помощью оператора GET LIST.

Пример. Минимизация функции $F(\mathbf{x}) = 4x_1 - x_2^2 - 12$ при ограничениях

$$h_1(\mathbf{x}) = 25 - x_1^2 - x_2^2 = 0,$$

$$g_2(\mathbf{x}) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geqslant 0,$$

$$g_3(\mathbf{x}) = x_1 \geqslant 0,\ g_4(\mathbf{x}) = x_2 \geqslant 0.$$

В качестве начальной точки выбиралась недопустимая точка $\mathbf{x}=(1,1)^T$, остальные параметры следующие: SIZE=0.3; CONVER=1.OE−5; NX=2; NC=1; NIC=3.

Функции $F(\mathbf{x})$ и функции ограничений $h_i(\mathbf{x})$ $(i=1, ..., m)$ и $g_i(\mathbf{x})$ $(j=m+1, ..., p)$ в каждой точке $\mathbf{x}$ вычисляются по программе PROBLM. Для вычисления ограничений в виде равенств используются строки 1, ..., m:

$$\mathrm{R}(1) = h_1(\mathbf{x}), ...,\ \mathrm{R}(m) = h_m(\mathbf{x}), \qquad (/*3*/)$$

в виде неравенств — строки $m+1$, ..., p:

$$\mathrm{R}(m+1) = g_{m+1}(\mathbf{x}), ...,\ \mathrm{R}(p) = g_p(\mathbf{x}) \qquad (/*4*/)$$

В строке $p+1$ вычисляется функция

$$\mathrm{R}(p+1) = F(\mathbf{x}), \qquad (/*5*/)$$

В данном примере

$$m = 1,\ p = 4;$$

$$\mathrm{R}(1) = 25 - \mathrm{X}(1)**2 - \mathrm{X}(2)**2,$$

$$\mathrm{R}(2) = 10*\mathrm{X}(1) - \mathrm{X}(1)**2 + 10*\mathrm{X}(2) - 34 - \mathrm{X}(2)**2,$$

$$\mathrm{R}(3) = \mathrm{X}(1),$$

$$\mathrm{R}(4) = \mathrm{X}(2),$$

$$\mathrm{R}(5) = 4*\mathrm{X}(1) - \mathrm{X}(2)**2 - 12.$$

Заметим, что если ограничения у задачи отсутствуют, то $\mathrm{R}(p+1)=\mathrm{R}(1)$, т. е. сразу надо вводить оператор, вычисляющий значения целевой функции.

После подготовки описанным выше способом исходных данных проведены расчеты на ЭВМ ЕС-1050. Точное (с уровнем CONVER=1.0 E−5) решение задачи $x_1=1{,}00128$, $X_2=4{,}989872$, $F(\mathbf{x}) = -3{,}19923 \cdot 10^1$ было получено за 11,6 с.

Программа FLEXI является достаточно сложной и объемной. Поэтому было проведено ее тестирование по нескольким контрольным примерам из приведенных в [58]. Чтобы пользователю было удобно следить за ходом выполнения программы FLEXI, организован поэтапный вывод информации о траектории поиска экстремума. Назначение отдельных частей программы FLEXI указано в виде комментариев в самой программе.

```
FLEXI: PROCEDURE OPTIONS(MAIN);
  ON FINISH PUT PAGE DATA;
  DECLARE
  (NX,NC,NIC) EXTERNAL,
  (STEP,ALFA,BETA,GAMMA,FDIFER,SEQL) EXTERNAL,
  (IN,INF,K1,K2,K3,K4,K5,
  K6,K7,K8,K9)  EXTERNAL,
  (SIZE,CONVER,R1A,R2A,R3A) EXTERNAL,
```

```
(LFEAS,L5,L6,L7,L8,L9)  EXTERNAL,
(SCALE,FOLD)  EXTERNAL,
LXN  EXTERNAL;
CALL VVOD;
K1=NX+1;  K2=NX+2;  K3=NX+3;
K4=NX+4;  K5=NX+5;  K6=NX+6;
K7=NX+7;  K8=NX+8;  K9=NX+9;
N=NX-NC;
IF  N<3  THEN N=2;
N1=N+1;  N2=N+2;  N3=N+3;
N4=N+4;  N5=N+5;  N6=N+6;  N7=N+7;  N8=N+8;
XN=N;    XNX=NX;  XN1=N1;
R1A=0.5*(SQRT(5.)-1); R2A=R1A*R1A;
R3A=R2A*R1A;
L5=NX+5; L6=NX+6; L7=NX+7;
L8=NX+8; L9=NX+9;
LXN=MAX(K9,L9);
BEGIN;
ON FINISH PUT PAGE DATA;
DECLARE
X(NX),X1(LXN,NX),X2(LXN,NX),F(LXN),R(LXN),
ZUM(LXN),SR(LXN),ROLD(LXN),
VRM CHAR(9);
ON  ENDFILE(SYSIN)
BEGIN;
 PRZNK=10;  GO TO OS;
END;
X=0; X1,X2=0; F,ZUM,SR,R,ROLD=0;
STEP=SIZE; PRZNK=0;
 DO WHILE(PRZNK=0);
  VRM=TIME; PUT SKIP DATA(VRM); PUT SKIP;
  GET LIST(X);
  PUT LIST(` START VECTOR `); PUT SKIP;
  PUT DATA(X); PUT SKIP;
 ICONT=1; NCONT=1;
 FDIFER=2.*(NC+1)*STEP; FOLD=FDIFER;
 IN=N1; CALL SUMR(X,ZUM); SR(N1)=SQRT(SEQL);
 PUT LIST(  START CRITERIUM `,FDIFER);
 PUT LIST(  SUM VIOLATION `,SR(N1));
 PUT SKIP;
 IF SR(N1)>FDIFER THEN
 DO;
  CALL WRITEX(R,X);
  PUT LIST(` CRITERIUM VIOLATED FOR X0 `);
  PUT SKIP;
  INF=N1; STEP=0.05*FDIFER;
  CALL FEASBL(X,X2,SR);
  PUT LIST(` X0 SATISFIES CRITERIUM  );
  PUT LIST(X2(INF,*)); PUT SKIP;
  PUT LIST(` SUM VIOLATION =
  SR(INF));  PUT SKIP;
  IF FOLD<1.0E-09  THEN
  DO;
```

```
     PRZNK=1;  GOTO  CS;
    END;
   END;
   DO I=1 TO 40;
    PUT EDIT(` * `)(A);
   END;
   PUT SKIP;
   PUT LIST(` CALCULATION NUMBER =
   ICONT);  PUT SKIP;
   PUT LIST(` CRITERIUM =
   FDIFER); PUT SKIP;
   CALL WRITEX(R,X); FTER=R(K9);
/* 1 */
   STEP1=STEP*(SQRT(XNX+1.)+XNX-1.)/(XNX*SQRT(2.));
   STEP2=STEP*(SQRT(XNX+1.)-1.)/(XNX+1.);
   ETA=(STEP1+(XNX-1.)*STEP2)/(XNX+1.);
   X=X-ETA;  CALL START(X,X1);  X2=X1;
   DO I=1 TO N1;
    IN=I;  X=X2(I,*);  CALL SUMR(X,ZUM);  SR(I)=SQRT(SEQL);
    IF FOLD >= FDIFER THEN
    DO;
     CALL FEASBL(X,X2,SR);
     IF FOLD < 1.0E-09 THEN
     DO;
      PRZNK=2; GOTO CS;
     END;
    END;
    CALL PROBLM(3,X,R);  F(I)=R(K9);
   END;
   DO WHILE(FDIFER >= CONVER);
    STEP=0.05*FDIFER;  ICONT=ICONT+1;
/* 2 */
    FH=F(1);  LHIGH=1;
    DO I=2 TO N1;
     IF F(I) >= FH THEN
     DO;
      FH=F(I); LHIGH=I;
     END;
    END;
/* 3 */
    PRZ=0;
    DO WHILE(PRZ=0);
     FL=F(1); LOW=1;
     DO I=2 TO N1;
      IF FL >= F(I) THEN
      DO;
       FL=F(I); LOW=I;
      END;
     END;
    X=X2(LOW,*); IN=LOW;
    CALL SUMR(X,ZUM); SR(LOW)=SQRT(SEQL);
    IF SR(LOW)<FDIFER THEN PRZ=1; ELSE
    DO;
     CALL FEASBL(X,X2,SR);
     IF FOLD<1.0E THEN
     DO;
      PRZNK=3; GOTO CS;
     END;
     CALL PROBLM(3,X,R); F(LOW)=R(K9);
    END;
   END;
```

```
/* 4 */
   DO J=1 TO NX;
    SUM2=0.;
    DO I=1 TO N1;
     SUM2=SUM2+X2(I,J);
    END;
    X2(N2,J)=1./XN*(SUM2-X2(LHIGH,J));
   END;
   SUM2=0;
   DO I=1 TO N1;
    DO J=1 TO NX;
     SUM2+SUM2+(X2(I,J)-X2(N2,J))**2;
    END;
   END;
    FDIFER=(NC+1)/XN1*SQRT(SUM2);
    IF FDIFER < FOLD THEN FOLD=FDIFER;
    ELSE FDIFER=FOLD;
    FTER=F(LOW); NCONT=NCONT+1;
    IF NCONT >= N1*4 THEN IF ICONT >= 1500
    THEN FOLD =0.5*FOLD; ELSE
    DO;
    NCONT=0;
    DO I=1 TO 40;
     PUT EDIT(' * ')(A);
    END;
    PUT SKIP;
    PUT LIST(' CALCULATION NUMBER =
    ICONT); FDIFER=FDIFER;
    PUT LIST(' CRITERIUM =
    FDIFER); PUT SKIP; CALL WRITEX(R,X);
    END;
    IF FDIFER < CONVER THEN
    DO;
     PRZNK=11; GOTO CS;
    END;
/* 5 */
    IF LHIGH > 1 THEN
    DO:
     FS=F(1); LSEC=1;
    END;
    ELSE
    DO;
     FS=F(2); LSEC=2;
    END;
     DO I=1 TO N1;
    IF LHIGH  = I THEN IF F(I) >= FS
    THEN
    DO;
     FS=F(I); LSEC=I;
    END;
   END;
/* 6 */
   DO J=1 TO NX;
    X2(N3,J)=X2(N2,J)+ALFA*(X2(N2,J)-X2(LHIGH,J));
    X(J)=X2(N3,J);
   END;
   IN=N3; CALL SUMR(X,ZUM); SR(N3)=SQRT(SEQL);
   IF SR(N3) >= FDIFER THEN
   DO;
    INF=N3; CALL FEASBL(X,X2,SR);
    IF FOLD < 1.0E-09 THEN
```

```
 DO;
  PRZNK=1; GOTO CS;
 END;
END;
CALL PROBLM(3,X,R); F(N3)=R(K9);
IF F(N3) >= F(LOW) THEN
IF F(N3) <  F(LSEC) THEN
DO;
 X2(LHIGH,*)=X2(N3,*); SR(LHIGH)=SR(N3);
 F(LHIGH)=F(N3);
END;
ELSE
DO;
 IF F(N3) <= F(LHIGH) THEN X2(LHIGH,*)=X2(N3,*);
 X2(N4,*)=BETA*X2(LHIGH,*)+(1.-BETA)*X2(N2,*);
 X=-X2(N4,*); IN=N4; CALL SUMR(X,ZUM);
 SR(N4)=SQRT(SEQL);
 IF SR(N4) >= FDIFER THEN
 DO;
  INF=N4; CALL FEASBL(X,X2,SR);
  IF FOLD < 1.0E-09 THEN
  DO;
   PRZNK=5; GOTO CS;
  END;
 END;
 CALL PROBLM(3,X,R); F(N4)=R(K9);
 IF F(LHIGH) > F(N4) THEN
 DO;
 X2(LHIGH,*)=X2(N4,*);
 SR(LHIGH)=SR(N4); F(LHIGH)=F(N4);
END;
ELSE
DO;
 DO I=1 TO N1;
 X2(I,*)=0.5*(X2(I,*)+X2(LOW,*));
END;
DO I=1 TO N1;
 X=X2(I,*); IN=I; CALL SUMR(X,ZUM):
 SR(I)=SQRT(SEQL);
 IF SR(I) < FDIFER THEN
 DO;
  CALL PROBLM(3,X,R);  F(I)=R(K9);
 END;
 ELSE
 DO;
  INF=I; CALL FEASBL(X,X2,SR);
  IF FOLD < 1.0E-09 THEN
  DO;
   PRZNK=6;  GOTO CS;
  END;
 END;
END;
END;
END;
ELSE
DO;
X2(N4,*)=X2(N3,*)+GAMMA*(X2(N3,*)-X2(N2,*));
IN=N4; X=X2(N4,*); CALL SUMR(X,ZUM); SR(N4)=SQRT(SEQL):
IF SR(N4) >= FDIFER THEN
DO;
 INF=N4; CALL FEASBL(X,X2,SR);
```

```
    IF FOLD < 1.0E-09 THEN
    DO;
     PRZNK=7; GOTO CS;
    END;
   END;
   CALL PROBLM(3,X,R); F(N4)=R(K9);
   IF F(LOW) < F(N4) THEN
   DO;
    X2(LHIGH,*)=X2(N3,*); SR(LHIGH)=SR(N3);
    F(LHIGH)=F(N3);
   END;
   ELSE
   DO;
    X2(LHIGH,*)=X2(N4,*); SR(LHIGH)=SR(N4);
    F(LHIGH)=F(N4);
   END;
  END;
 END;
 CS:
   PUT LIST('TOTAL NUMBER OF CALCULATIONS = ',ICONT);
   PUT SKIP; PUT DATA(PRZNK);
   FDIFEP=FDIFER;
   PUT LIST(' VIOLATION OF BOUNDARIES NOT MORE = ',FDIFEP);
   PUT SKIP; CALL WRITEX(R,X);
   IF PRZNK > 9 THEN
   DO;
    VRM=TIME; PUT DATA(VRM);
    PUT LIST('  FINAL SOLUTION ');
   END;
   IF PRZNK < 9 THEN
   DO;
    PRZNK=0;
    PUT LIST('  NOT A SOLUTION ');
   END;
  END;
     OS:
     END;
   END FLEXI;
FEASBL:PROCEDURE(XS,XS2,SSR);
       DECLARE
       (IQZNK,JQZNK,
       NX,NC,NIC,
       STEP,ALFA,BETA,GAMMA,FDIFER,SEQL,
       IN,INF,K1,K2,K3,K4,K5,K6,K7,K8,K9,
       SIZE,CONVER,SCALE,FOLD,R1A,R2A,R3A,
       LFEAS,L5,L6,L7,L8,L9,LXN)
       EXTERNAL,
       XS(*),XS2(*,*),SSR(*);
       XNX=NX;
       BEGIN;
       DECLARE
       X(NX),X2(LXN,NX),SR(LXN),
       X1(LXN,NX),F(LXN),H(NX),R1(LXN),
       R3(LXN),R(LXN),ZUM(LXN);
       X1=0; F,R,R1,R3,ZUM=0; H=0;
       X=XS; X2=XS2; SR=SSR;
       ICONT=0; LCHEK=0; ICHEK=0;
       CALL START(X,X1);
       DO WHILE(ICHEK < 3 & SR(INF) > FDIFER);
        DO I=1 TO K1;
         X=X1(I,*); IN=I;
```

```
 CALL SUMR(X,ZUM);
END;
DO WHILE(ICONT < 2*K1 & SR(INF) > FDIFER);
 SUMH=ZUM(1); INDEX=1;
 DO I=2 TO K1;
  IF ZUM(I) > SUMH THEN
 DO;
  SUMH=ZUM(I); INDEX=I;
 END;
END;
SUML=ZUM(1); KOUNT=1;
DO I=2 TO K1;
 IF SUML < ZUM(I) THEN
 DO;
  SUML=ZUM(I); KOUNT=I;
 END;
END;
DO J=1 TO NX;
 SUM2=0;
 DO I=1 TO K1;
  SUM2=SUM2+X1(I,J);
 END;
 X1(K2,J)=1./XNX*(SUM2-X1(INDEX,J));
   X1(K3,J)=2.*X1(K2,J)-X1(INDEX,J); X(J)=X1(K3,J);
 END;
 IN=K3; CALL SUMR(X,ZUM);
 IF ZUM(K3) < SUML THEN
 DO;
  X1(K4,*)=X1(K2,*)+2.*(X1(K3,*)-X1(K2,*));
  X=X1(K4,*); IN=K4; CALL SUMR(X,ZUM);
  IF ZUM(K4) < SUML THEN
  DO;
   X1(INDEX,*)=X1(K4,*); X=X1(INDEX,*);
   ZUM(INDEX)=ZUM(K4); SR(INF)=SQRT(ZUM(K4));
  END;
  ELSE
  DO;
   X1(INDEX,*)=X1(K3,*);
   ZUM(INDEX)=ZUM(K3); SR(INF)=SQRT(ZUM(K3));
  END;
END;
ELSE
DO;
 IF INDEX=1 THEN SUMS=ZUM(2);
 ELSE SUMS=ZUM(10;
 DO I=1 TO K1;
  IF INDEX  = I THEN
  IF ZUM(I)>SUMS THEN SUMS=ZUM(I);
 END;
 IF ZUM(K3) <= THEN
 DO;
  X1(INDEX,*)=X1(K3,*); ZUM(INDEX)=ZUM(K3);
  SR(INF)=SQRT(ZUM(K3));
 END;
 ELSE
 DO;
  IF ZUM(K3) <= SUMH THEN
  X1(INDEX,*)=X1(K3,*);
  X1(K4,*)=0.5*X1(INDEX,*)+0.5*X1(K2,*);
  X=X1(K4,*); IN=K4; CALL SUMR(X,ZUM);
  IF SUMH <= ZUM(K4) THEN
  DO;
```

```
            DO I=1 TO K1;
             X1(I,*)=0.5*(X1(I,*)+X1(KOUNT,*));
            END;
            DO I=1 TO K1;
             X=X1(I,*); IN=I; CALL SUMR(X,ZUM);
            END;
           END;
           ELSE
           DO;
            X1(INDEX,*)=X1(K4,*); ZUM(INDEX)=ZUM(K4);
           END;
          END;
          SUML=ZUM(I); KOUNT=1;
          DO I=2 TO K1;
           IF SUML >= ZUM(I) THEN
           DO;
            SUML=ZUM(I); KOUNT=1;
           END;
           SR(INF)=SQRT(ZUM(KOUNT)); X=X1(KOUNT,*);
          END;
         END;
         ICONT=ICONT+1; X2(INF,*)=X;
        END;
        IF ICONT >= 2*K1 THEN
        DO;
         ICONT=0; IN=K2; X=X1(K2,*); CALL SUMR(X,ZUM);
         DIFER=0;
         DO I=1 TO K1;
          DIFER=DIFER+(ZUM(I)-ZUM(K2)**2;
         END;
         DIFER=1./(K7*XNX)*SQRT(DIFER);
        END;
/* 1 */
        IF DIFER <= 1.0E-07/(K7*XNX) THEN
        DO;
         IN=K1; STEP=20.*FDIFER;
         CALL SUMR(X,ZUM); X1(K1,*)=X; J=0;
        DO WHILE(SR(INF) > FDIFER & J < NX);
         J=J+1; FACTOR=1; X(J)=X1(K1,J)+FACTOR*STEP;
         X1(L9,J)=X(J); IN=L9; CALL SUMR(X,ZUM);
         X(J)=X1(K1,J)-FACTOR*STEP; X1(L5,J)=X(J); IN=L5;
         CALL SUMR(X,ZUM);
         DO WHILE(ZUM(L9) < ZUM(K1) ! ZUM(L5) < ZUM(K1));
          IF  ZUM(L9) < ZUM(K1) THEN
          DO;
           LZ=L9; KZ=L5; ZK=1.;
          END;
          ELSE IF ZUM(L5) < ZUM(K1) THEN
          DO;
           LZ=L5; KZ=L9; ZK=-1;
          END;
          X1(KZ,J)=X1(K1,J); ZUM(KZ)=ZUM(K1);
          X1(K1,J)=X1(LZ,J); ZUM(K1)=ZUM(LZ); FACTOR=FACTOR+1.;
          X(J)=X1(K1,J)+ZK*FACTOR*STEP; IN=LZ; CALL SUMR(X,ZUM);
         END;
         FD=0.01*FDIFER; ASX=ABS(X1(L9,J)-X1(L5,J));
         DO WHILE(ABS > FD & ASX> 1.E-06);
          ASX=ABS(X1(L9,J)-X1(L5,J)); H(J)=X1(L9,J)-X1(L5,J);
          X1(L6,J)=X1(L5,J)+H(J)*R1A; X(J)=X1(L6,J); IN=L6;
          CALL SUMR(X,ZUM); X1(L7,J)=X1(L5,J)+H(J)*R2A;
          X(J)=X1(L7,J); IN=L7; CALL SUMR(X,ZUM);
          IF ZUM(L6) > ZUM(L7) THEN
```

```
            DO;
             X1(L9,J)=X1(L6,J); X1(L8,J)=X1(L5,J)+R3A*H(J);
             X(J)=X1(L8,J); IN=L8; CALL SUMR(X,ZUM);
             ZUM(L9)=ZUM(L6); IF ZUM(L7) > ZUM(L8) THEN
             DO;
              X1(L9,J)=X1(L7,J); ZUM(L9)=ZUM(L7);
             END;
             ELSE
             DO;
              X1(L5,J)=X1(L8,J); ZUM(L5)=ZUM(L8);
             END;
            END;
            ELSE
            DO;
             X1(L8,J)=X1(L5,J)+(1.-R3A)*H(J);
             X1(L5,J)=X1(L7,J);
             X(J)=X1(L8,J); IN=L8; CALL SUMR(X,ZUM);
             IF ZUM(L8) > ZUM(L6) THEN
             DO;
              X1(L9,J)=X1(L8,J); ZUM(L9)=ZUM(L8);
             END;
             ELSE
             DO;
              X1(L5,J)=X1(L6,J); ZUM(L5)=ZUM(L6);
             END;
            END;
           END;
           X1(K1,J)=X1(L7,J); X(J)=X1(L7,J);
           ZUM(K1)=ZUM(L5); SR(INF)=SQRT(ZUM(K1));
          END;
/* 2 */
          ICHEK=ICHEK+1; STEP=0.005*FDIFER;
          IF SR(INF) < FDIFER THEN
          DO;
           X2(INF,*)=X1(K1,*); X=X1(K1,*);
          END;
           END;
          END;
          IF SR(INF) <= 0. THEN
          DO;
           CALL PROBLM(3,X,R); FINT=R(K9); X=X2(INF,*);
           CALL PROBLM(2,X,R);
           DO I=K7 TO K8;
            R1(I)=R(I);
           END;
           X=X1(KOUNT,*); CALL PROBLM(2,X,R);
           DO I=K7 TO K8;
            R3(I)=R(I);
           END;
           H=X1(KOUNT,*)-X2(INF,*); X=X2(INF,*)+0.5*H;
           CALL PROBLM(2,X,R); FLG1,FLG2,FLG3=0.;
           DO I=K7 TO K8;
            IF R3(I) <= 0.THEN
            DO;
             FLG1=FLG1+R1(I)*R1(I);
             FLG2=FLG2+R(I)*R(I);  FLG3=FLG3+R3(I)*R3(I);
            END;
           END;
           SR(INF)=SQRT(FLG1); IF SR(INF) < FDIFER THEN
           DO;
            ZNC=1; GOTO CZ;
           END;
```

```
    ALFA1=FLG1-2.*FLG2-FLG3;  BETA1=3.*FLG1-4.*FLG2+FLG3;
    RATIO=BETA1/(4.*ALFA1); X=X2(INF,*)+H*RATIO;
    IN=INF; CALL SUMR(X,ZUM); SR(INF)=SQRT(SEQL); I=0;
    DO WHILE(SR(INF) >= FDIFER & I < 20 );
     I=I+1; X=X-0.05*H;
CZ:
     CALL SUMR(X,ZUM); SR(INF)=SQRT(SEQL);
    END;
    CALL PROBLM(3,X,R); IF FINT > R(K9) THEN ZNC=0;
    ELSE
    DO;
     ZNC=1; SR(INF)=0.;
    END;
   END;
   ELSE ZNC=1;
   IF ICHEK >= 3 THEN
   DO;
    FOLD=1.0E-12; ZNC=0;
    PUT SKIP LIST(`*****PROGRAM FEASBL TRIED ,
     TO FIND POINT*****`);
     PUT SKIP LIST(`THIS IS NOT A SOLUTION, ANOTHER ,
      START VECTOR IS NEEDED`);
    PUT SKIP LIST(`CURRENT START VECTOR`);
    PUT LIST(X);
    PUT LIST(`CRITERIUM =  ,FDIFER);
    PUT LIST(`SUM VIOLATION = `,SR(INF));
    PUT SKIP;
   END;
   IF ZNC=0 THEN X2(INF,*)=X; ELSE X=X2(INF,*);
   XS=X; XS2=X2; SSR=SR;
   END;
 END FEASBL;
```

6.6. Градиентный метод

Одним из наиболее распространенных в инженерной практике является градиентный метод [64, 65]. Данный метод относится к группе локальных методов поиска экстремума, которые предполагают, что целевая функция $F(\mathbf{x})$ имеет только один локальный максимум в задачах максимизации или один локальный минимум в задачах минимизации и обязательно во внутренней точке области D. Наиболее существенной особенностью также является существование у $F(\mathbf{x})$ вектора-градиента

$$\nabla F(\mathbf{x}) = \left\{ \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}_1}, \ldots, \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}_m} \right\}^{\tau}.$$

Производная функции $F(\mathbf{x})$ по любому направлению d вычисляется как проекция градиента $\Delta F(\mathbf{x})$ на это направление:

$$\frac{\partial F(\mathbf{x})}{\partial d} = \| \nabla F(\mathbf{x}) \| \cos(\widehat{\nabla F, d}).$$

Следовательно, функция $F(\mathbf{x})$ в любой точке пространства $\mathbf{x}$ быстрее всего возрастает по направлению градиента $\Delta F(\mathbf{x})$, а убывает — по направлению антиградиента $-\Delta F(\mathbf{x})$.

Рассмотрим существо градиентного метода на примере задачи максимизации $F(\mathbf{x})$ (в случае минимизации $F(\mathbf{x})$ построения аналогичные). В соответствии с принципом максимального возрастания $F(\mathbf{x})$ вдоль $\Delta F(\mathbf{x})$ согласно градиентному методу избирается такая траектория поиска (от исходного приближения $\mathbf{x}_0$), что в каждой точке направления градиента $\Delta F(\mathbf{x})$ является касательным к этой траектории. Данная траектория $\mathbf{x}_n(t)$ является ортогональной поверхностям уровня $F(\mathbf{x})$ (т. е. кривым $F(\mathbf{x}) = F_\Phi$, где F_Φ — фиксированное значение $F(\mathbf{x})$) и определяется из решения дифференциального уравнения

$$\frac{d\mathbf{x}_n(t)}{dt} = \nabla F(\mathbf{x}),\ \mathbf{x}_n(0) = \mathbf{x}_0,$$

где t — параметр кривой поиска (t определено в интервале $t \in [0, t_к]$, $t_к$ — конечное значение параметра поиска, определенное попаданием $\mathbf{x}_n(t)$ в точку максимума $F(\mathbf{x})$). Таким образом, при поиске экстремума методом градиента каждый раз выбирается оптимальное направление поиска. При практической реализации вышеприведенная система дифференциальных уравнений решается приближенными методами. При этом ортогональная траектория $\mathbf{x}_n(t)$ заменяется ломаной с вершинами в последовательных точках x_0, x_1, ..., которые определяются, например, с помощью метода Эйлера по формулам

$$x_{i+1} = x_i + \Delta x_i,$$

где

$$\Delta x_i = \nabla F(x_i)\,\Delta t_i;\ \Delta t_i = t_{i+1} - t_i;$$

$$\Delta x_i = x_n(t_{i+1}) - x_n(t_i),\ i \geqslant 0.$$

Окончательно итерационная формула алгоритма:

$$x_{k+1} = x_k + \alpha \nabla F(x_k),$$

где α — задаваемый коэффициент. Последнее соотношение реализуется с помощью последовательности шагов h_0, h_1, ..., которые определяют геометрическое продвижение из вершин x_0, x_1, ... в направлении градиента (антиградиента в случае задачи минимизации). Шаги h_0, h_1, ... вычисляются по формулам $h_i = \alpha \|\Delta F(x_i)\|$ $(i \geqslant 0)$.

При практическом применении данного метода основным сдерживающим фактором является необходимость многократного вычисления вектора градиента $F(\mathbf{x})$, что и обусловливает дополнительные требования к точности вычисления $F(\mathbf{x})$.

ПРОГРАММА FPFP

Назначение: решение задачи математического программирования градиентным методом (см. также [1]).

Параметры:

N — размерность оптимизируемого вектора параметров;

X — вектор оптимизируемых параметров. Перед обращением к программе FPFP X присваивается начальное значение, по окончании работы X принимает оптимальное значение.

EPS — точность определения оптимальных параметров;
EST — значение оптимизируемого функционала в начальной точке;
FUNK — процедура расчета значений оптимизируемого функционала;
FX — процедура расчета частных производных оптимизируемого функционала.

Обращение: CALL FPFP(X, N, EPS, EST, FUNK, FX);

Пример. Задача сепарабельного непрерывного программирования $F(x_1, x_2)= =(x_1-100)^2+(x_2-10)^2$.

Точное решение (EPS=0,0001) получено за 0,1 с на ЭВМ ЕС-1040.

```
FMFP:PROCEDURE (FUNCT,N,X,F,G,EST,EPS,LIMIT);
     DECLARE
          (I,J,KOUNT,K,L,LIMIT,N,NS,N2,N3)
          BINARY  FIXED,
          (X(*),G(*),H(N*(N+7)/2),ALFA,AMBDA,DALFA,DX,DY,GS,
          GNRM,FS,EPS,EST,F,FX,FY,H1,H2,HNRM,DLDF,T,W,Z)
          BINARY FLOAT,
           FUNCT
      ENTRY,
      ERROR EXTERNAL
      CHARACTER(1);
     NS=N;
     N2=NS+NS;
     N3=N2+NS;
     CALL FUNCT(X,FS,G);
     ERROR='0';
     KOUNT=0;
    CONT:
     I=N3;
         DO J=NS-1  TO  0  BY  -1;
        K=I+1;
         H(K)=1;
         I=K+J;
              DO  L=K+1 TO I;
              H(L)=0;
              END;
         END;
    LOOP:
     KOUNT=KOUNT+1;
     DDDF=FS;
     DY,HNRM,GNRM=0;
         DO J=1  TO  NS;
         H(NS*J),GS=G(J);
         H(N2*J)=X(J);
         T=0;
         K=N3+J;
              DO L=1 TO NS;T=T-G(L)*H(K);
              IF L<J
              THEN K=K+NS-L;
              ELSE K=K+1;
              END;
         H(J)=T;
         HNRM=HNRM+ABS(T)
         GNRM=GNRM+ABS(JS);
         DY=DY+T*GS;
         END;
      IF DY<0
      THEN IF HNRM/GNRM > EPS
      THEN GO TO LAB1;
      GO TO REST;
```

```
LAB1:
 FY=FS;
 AMBDA=MIN(1,2*(EST-FS)/DY);
 IF AMBDA <= 0
 THEN AMBDA=1;
 ALFA=0;
SAVE:
 FX=FY;
 DX=DY;
    DO I=1 TO NS;
    X(I)=X(I)+AMBDA*H(I);
   END;
CALL FUNCT(X,FS,G);
FY=FS;
DY=0;
     DO I=1 TO NS;
     DY=DY+G(I)*H(I);
     END;
IF FY<FX
THEN DO;
    IF DY=0
    THEN GO TO COMP;
    IF  DY<0
    THEN DO;
           ALFA,AMBDA=AMBDA+ALFA;
           IF  HNRM*AMBDA <= 1E10
           THEN GO TO SAVE;
           ERROR='2';
           GO TO RETURN;
           END;
    END;
T=0;
LAB2:
 IF  AMBDA=0
 THEN GO TO COMP;
 Z=3*(FX-FY)/AMBDA*DX+DY;
 ALFA=MAX(ABS(Z),ABS(DX),ABS(DY));
 DALFA=Z/ALFA;
 DALFA=DALFA*DALFA-DX/ALFA*DY/ALFA;
 IF  DALFA  <  0
 THEN GO TO REST;
 W=ALFA*SQRT(DALFA);
 ALFA=DY-DX+W+W;
 IF  ALFA  =  0
 THEN ALFA=(Z+DY-W)/(Z+DX+Z+DY);
 ELSE ALFA=(DY-Z+W)/ALFA;
 ALFA=ALFA*AMBDA;
 DALFA=T-ALFA;
    DO I=1 TO NS;
    X(I)=X(I)+DALFA*H(I);
    END;
 CALL  FUNCT(X,FS,G);
 IF  FS  <=  FX
 THEN  IF  FS <= FY
 THEN GO TO COMP;
 DALFA=0;
    DO I=1 TO NS;
    DALFA=DALFA+G(I)*H(I);
    END;
 IF DALFA < 0
 THEN  IF  FS <= FX
 THEN DO;
```

```
   FX=FS;
   DX=DALFA;
   T,AMBDA=ALFA;
   GO TO LAB2;
   END;
FY=FS;
DY=DALFA;
AMBDA=AMBDA-ALFA;
T=0;
GO TO LAB2;
COMP;
    DO J=1 TO NS;
    K=NS+J;
    H(K)=G(J)-H(K);
    K=NS+K;
    H(K)=X(J)-H(K);
    END;
 IF DLDF+EPS < FS
 THEN GO TO REST;
 ERROR='0';
 IF KOUNT >= NS
 THEN DO;
    T,Z=0;
             DO J=1 TO NS;
             W=H(N2+J);
             T=T+ABS(W);
             Z=Z+H(NS+J)*W;
             END;
    IF  HNRM <= EPS
    THEN  IF  T <= EPS
    THEN  GO  TO  RETURN;
    END;
 IF  KOUNT >=LIMIT
 THEN GO TO NCON;
 ALFA=0;
    DO J=1 TO NS;
    W=0;
    K=N3+J;
             DO L=1 TO NS;
             W=W+H(NS+L)*H(K);
             IF  L < J
             THEN  K=K+NS-L;
             ELSE  K=K+1;
             END;
    ALFA=ALFA+W*H(NS+J);
    H(J)=W;
    END;
 IF  Z*ALFA=0
 THEN  GO TO  CONT;
 K=N3+1;
    DO L=1  TO  NS;
    H1=H(N2+L)/Z;
    H2=H(L)/ALFA;
             DO J=L  TO  NS;
             H(K)=H(K)+H1*H(N2+J)
                  -H2*H(J);
             K=K+1;
             END;
    END;
 GO TO LOOP;
NCON:
 ERROR='1';
```

```
      GO TO RETURN;
     REST:
         DO J=1 TO NS;
         X(J)=H(N2+J);
         END;
      CALL FUNCT(X,FS,G);
      IF  GNRM  >  EPS
      THEN  IF  ERROR='3'
      THEN GO TO RETURN;
      ELSE DO;
         ERROR='3';
         GO TO CONT;
         END;
      ERROR='0';
     RETURN:
      F=FS;
      END FMFP;


    FUNCT: PROC(X,FS,G);
    DCL (X(*),G(*),FS) BINARY FLOAT,
  N BINARY FIXED;
  FS=100*(X(2)-X(1)**2)**2+
  (1-X(1))**2;
  G(1)=-400*(X(2)-X(1)**2)*X(1)
  -2*(1-X(1));
  G(2)=200*(X(2)-X(1)**2);
 END;


 TEST:PROCEDURE OPTIONS(MAIN);
 DCL (X(2),G(2),F,EST,EPS) BINARY FLOAT,
 (N,LIMIT) BINARY FIXED;
  N=2;  X(1)=-1.2;  X(2)=1;
   EST=24.2;  EPS=0.001;  LIMIT=250;
 CALL FMFP(FUNCT,N,X,F,G,EST,EPS,LIMIT);
 PUT SKIP DATA(F,X(1),X(2));
 END TEST;
```

6.7. Метод Дэвидона—Флетчера—Пауэлла

Метод Дэвидона — Флетчера — Пауэлла [61, 58] относится к группе методов оптимизации, называемых методами переменной метрики. Методы этого типа называются еще и квазиньютоновскими или сопряженных направлений. Название, как будет ясно из дальнейшего изложения, подсказано самой схемой алгоритма. Метод Дэвидона—Флетчера—Пауэлла предназначен для безусловной минимизации дифференцируемой функции нескольких переменных, т. е. для решения задачи (6.1)—(6.3), если ограничения на область изменения оптимизируемых параметров отсутствуют. Особенностью метода Дэвидона—Флетчера—Пауэлла является то, что направление поиска минимума функции $F(\mathbf{x})$ осуществляется вдоль направления, задаваемого вектором

$$-\mathbf{B}\nabla F(x), \quad \nabla F(\mathbf{x}) = \left(\frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_m}\right)^{\mathrm{T}}$$

— градиент функции F в точке $\mathbf{x}$; $\mathbf{B}$ — некоторая положительно определенная

квадратная $(m\times m)$—матрица, которая является аппроксимацией матрицы $\mathbf{D}$, обратной матрице $\mathbf{H}=\left\{\dfrac{\partial^2 F}{\partial x_i \partial x_j}\right\}(i, j=1, \ldots, m)$.

Основной принцип алгоритма Дэвидона—Флетчера—Пауэлла базируется на простом факте: если представить функцию F в окрестности точки $\mathbf{x}_{\min}$ в виде разложения в ряд Тейлора

$$F(\mathbf{x}) \approx F(\mathbf{x}_{\min}) + 0{,}5(\mathbf{x}-\mathbf{x}_{\min})^{\mathrm{T}}\mathbf{H}(\mathbf{x}-\mathbf{x}_{\min}),$$

то ее градиент можно приближенно вычислить по формуле $\nabla F(\mathbf{x})=\mathbf{H}(\mathbf{x}-\mathbf{x}_{\min})$. Следовательно, $\mathbf{x}_{\min}=\mathbf{x}-\mathbf{H}^{-1}\nabla F(\mathbf{x})\approx\mathbf{x}-\mathbf{B}\nabla F(\mathbf{x})$. Это подсказывает возможный подход к решению задачи нахождения минимума F — построение итерационного процесса вида

$$\mathbf{x}_{k+1}=\mathbf{x}_k-\alpha_k\mathbf{B}_k\nabla F(\mathbf{x}_k),\quad k=1,2,\ldots, \tag{6.7}$$

где $\mathbf{B}_k$ — матрица аппроксимации $\mathbf{H}^{-1}$ на k=м 'шаге; α_k — положительное число, выбираемое так, чтобы в точке $\mathbf{x}_{k+1}$ достигался локальный минимум вдоль направления $-\mathbf{B}_k\nabla F(\mathbf{x}_k)$, т. е. чтобы α_k являлось оптимальным решением задачи минимизации $F(\mathbf{x}_k-\alpha_k\mathbf{B}_k\nabla F(\mathbf{x}_k))$ по α $(\alpha\geqslant 0)$. Рекуррентное соотношение (6.7) поясняет название алгоритма «переменной метрики»— матрица $\mathbf{B}_k$ изменяет метрику в пространстве векторов $\nabla F(\mathbf{x}_k)$. В результате умножения на матрицу $\mathbf{B}_k$ направление градиента $\nabla F(\mathbf{x}_k)$ отклоняется и преобразуется в $\mathbf{B}_k\nabla F(\mathbf{x}_k)$.

Существенным в алгоритме Дэвидона—Флетчера—Пауэлла является то, что матрица $\mathbf{B}_k$ вычисляется с использованием только первых производных F и так, чтобы после m шагов первой же итерации (п: 1 основного этапа алгоритма) она совпала с $\mathbf{H}^{-1}$, если бы F была квадратичной функцией. Таким образом, отпадает необходимость в непосредственном вычислении и обращении матрицы Гессе в процессе работы алгоритма Дэвидона—Флетчера—Пауэлла минимизации дифференцируемой функции нескольких переменных (см. также [58, 61]).

Начальный этап. Выбор начальной точки поиска $\mathbf{x}_0$ и начального приближения матрицы $\mathbf{B}_0$ ($\mathbf{B}_0$ — симметричная и положительно определенная), а также параметров точности вычисления минимума $\varepsilon>0$. Принимаем $\mathbf{z}_0=\mathbf{x}_0$, $k=1$, $i=1$. Переходим к основному этапу минимизации.

Основной этап. 1. Если $\|\nabla F(z_i)\|<\varepsilon$, то $\mathbf{z}_i$ — искомая точка минимума F и алгоритм заканчивает свою работу. В противном случае вычисляем $\Delta_i=-\mathbf{B}_i\nabla F(\mathbf{z}_i)$ и решаем задачу минимизации (по скаляру $\alpha\geqslant 0$) функции $F(\mathbf{z}_i+\alpha\Delta_i)$. Полагаем $\mathbf{z}_{i+1}=\mathbf{z}_i+\alpha_i\Delta_i$. Если $i<m$, то осуществляем переход к п. 2. Если $i=m$, то полагаем $\mathbf{z}_1=\mathbf{x}_{k+1}=\mathbf{z}_{m+1}$, $i=1$, заменяем k на $k+1$ и переходим к началу п. 1.

2. Вычисляем матрицу $\mathbf{B}_{i+1}$ по следующим формулам:

$$\mathbf{v}_i=\alpha_i\Delta_i;\quad \mathbf{w}_i=\nabla F(z_{i+1})-\nabla F(\mathbf{z}_i); \tag{6.8}$$

$$\mathbf{B}_{i+1}=\mathbf{B}_i+\frac{\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}}{\mathbf{v}_i^{\mathrm{T}}\mathbf{w}_i}-\frac{\mathbf{B}_i\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\mathbf{B}_i}{\mathbf{w}_i^{\mathrm{T}}\mathbf{B}_i\mathbf{w}_i}.$$

Заменяем i на $i+1$ и возвращаемся к п. 1.

При реализации алгоритма в виде программы обычно в качестве начальной

матрицы $\mathbf{B}_0$ выбирается единичная матрица $\mathbf{E}$, можно также использовать любую другую симметричную положительно определенную квадратную $(m\times m)$-матрицу. Весьма важен правильный выбор алгоритма одномерного поиска, осуществляющего минимизацию функции $F(\mathbf{z}_l+\alpha\Delta_l)$, так как на эту часть алгоритма приходится большая часть общего объема вычислений. Желательно, чтобы точность одномерного поиска была выше (или по крайней мере эквивалентна) точности ε самого алгоритма Дэвидона—Флетчера—Пауэлла. Отметим также, что алгоритм Дэвидона—Флетчера—Пауэлла достаточно чувствителен к ошибкам, получаемым при вычислении градиента $\nabla F(\mathbf{x})$. Поэтому надо осторожно подходить к использованию вместо градиента его оценок, получаемых с помощью отношений разностей. Использование грубых оценок градиента приводит либо к значительному росту количества вычислений минимизируемой функции по сравнению со случаем, когда используются аналитические выражения, либо вообще к аварийному завершению процесса оптимизации [58, 64,].

В частном случае, когда функция F квадратичная, в ходе поиска, вырабатываются $\mathbf{H}$-сопряженные направления поиска Δ_1, ..., Δ_m, т. е. имеют место соотношения $\Delta_i^T\mathbf{H}\Delta_j=0$ при $i\neq j$. После одной итерации алгоритм останавливается в точке минимума (т. е. z_{m+1} является оптимальным решением задачи). Кроме того, имеет место соотношение $\mathbf{B}_{m+1}=\mathbf{H}^{-1}$.

Отметим, что алгоритм Дэвидона—Флетчера—Пауэлла включается в более общий класс квазиньютоновских процедур, в которых направление поиска задается в виде $-\mathbf{M}\nabla F(\mathbf{x})$, где $\mathbf{M}$ — некоторая положительно определенная симметричная матрица. Различные принципы выбора матрицы $\mathbf{M}$ порождают серию алгоритмов оптимизации (см. [1, 58, 60, 64, 65]). Программные аспекты реализации алгоритмов переменной метрики достаточно полно освещены в современной научно-технической литературе. С наиболее эффективными программами читатель может познакомиться по обзорам, помещенным в [1, 58].

6.8. Методы штрафных и барьерных функций

Широко распространенными в инженерных приложениях методами условной оптимизации являются методы штрафных и барьерных функций, которые основаны на сведении общей задачи (6.1)—(6.3) (ограничения (6.3) заданы в виде $g_i(\mathbf{x})\leqslant 0$) к одной или последовательности задач безусловной оптимизации.

В *методе штрафных функций* это реализуется прибавлением к функции критерия $F(\mathbf{x})$ функции $A(\mathbf{x})$, которая определяет штраф за выход из области допустимых значений аргумента и не должна штрафовать допустимые точки. Таким образом, наличие штрафной функции $A(\mathbf{x})$ стимулирует обязательный вход траектории поиска в область допустимости и дальнейший поиск в этой области. Этот метод (называемый также методом внешних штрафных функций) предложен Р. Курантом в 1943 г., а затем развит многими авторами (см. [63, 67, 68]).

Для ограничений (6.2), (6.3) обычно используют штрафные функции вида

$$A(\mathbf{x})=\sum_{i=1}^{n}H[h_i(\mathbf{x})]+\sum_{i=n+1}^{p}G[g_i(\mathbf{x})],$$

где H, G — непрерывные скалярные функции, удовлетворяющие соотношениям

$$H(z) = 0, \text{ если } z = 0;\ H(z) > 0, \text{ если } z \neq 0;$$

$$G(z) = 0, \text{ если } z \leqslant 0;\ G(z) > 0, \text{ если } z > 0.$$

Наиболее часто на практике используют функции следующего вида:

$$H(z) = [\max\{0,z\}]^{\alpha_1},\ H(z) = [\max\{0,z\} + 1]^{\alpha_2} - 1;$$

$$G(z) = |z|^{\alpha_3},$$

где α_1, α_2, α_3 — положительные константы.

Введение штрафных функций $A(x)$ позволяет сформировать вспомогательную критериальную функцию $\bar{F}_\beta(x) = F(x) + \beta A(x)$, где β — параметр штрафа (положительная константа).

При соответствующем выборе β решение задачи минимизации функции $\bar{F}_\beta(x)$ будет приближаться к решению задачи (6.1)—(6.3). Можно строго доказать [63], что если $A(x)$ — непрерывная функция на $\mathbf{R}^m$, то

$$\lim_{\beta \to \infty} \inf_{x \in \mathbf{R}^m} \bar{F}_\beta(x) = \inf_{x \in D} F(x),$$

где D — область допустимости, задаваемая соотношениями (6.2), (6.3). Следовательно, решение задачи минимизации $\bar{F}_\beta(x)$, вообще говоря, приближается к решению задачи (6.1)—(6.3) при возрастании значений параметра штрафа β. Соответственно этому большинство алгоритмов штрафных функций основано на последовательном решении задачи минимизации $\bar{F}_{\beta_k}(x)$ при некоторой совокупности возрастающих параметров штрафа $\beta_k \to \infty$ $(k \to \infty)$.

Предположим, что условие непрерывности функций F, h_i $(i = 1, \ldots, n)$, g_j $(j = n+1, \ldots, p)$ выполнено. Алгоритм штрафных функций состоит из следующих трех основных этапов.

1. Задаются начальная точка поиска x, начальное значение параметра β_0, коэффициент увеличения штрафного параметра $k_\beta > 1$, а также параметр точности решения задачи $\varepsilon > 0$. Полагаем $k = 0$.

2. На $(k+1)$-м шаге и при начальной точке x_k решаем задачу минимизации

$$\bar{F}_{\beta_k}(x) = F(x) + \beta_k A(x),\ x \in \mathbf{R}^m.$$

Полагаем x_{k+1} равным вектору решения этой задачи. Переходим к п. 3.

3. Проверяем выполнение условия $\beta_k A(x_{k+1}) < \varepsilon$: если исход проверки положительный, оптимальное решение основной задачи найдено, в противном случае полагаем $\beta_{k+1} = k_\beta \beta_k$, заменяем k на $k+1$ и переходим к п. 2.

Несмотря на имеющиеся доказательства сходимости метода штрафных функций к оптимальному решению, на практике этот подход сопряжен с определенными вычислительными трудностями. Они вызваны тем, что при больших значениях β_k будет в основном осуществляться поиск допустимого решения и конечная точка поиска x может отстоять достаточно далеко от оптимальной, если функция $F(x)$ слишком мала по сравнению с $\beta_k A(x)$ $(k \geqslant 1)$. Следовательно, при практических расчетах целесообразно начинать с небольших значений коэффициентов β_0,

например таких, чтобы функция $\beta_0 A(\mathbf{x})$ была примерно одного порядка со средним значением функции $F(x)$ на множестве допустимых значений.

Метод барьерных функций аналогичен методу штрафных функций и также основан на преобразовании исходной задачи минимизации с системой ограничений в задачу (или последовательность задач) безусловной минимизации с помощью введения барьерных функций, препятствующих выходу из области допустимости в течение процедуры поиска. Поэтому этот метод часто называют методом внутренних штрафных функций.

Метод барьерных функций работоспособен в задачах с ограничениями типа (6.3). Будем предполагать, что $h_i(\mathbf{x}) = 0 (\mathbf{x} \in \mathbf{R}^m\ i = 1, \ldots, n)$. Аналогично вышеизложенному формируется вспомогательная (барьерная) задача минимизации

$$\tilde{F}_\alpha(\mathbf{x}) = F(\mathbf{x}) + \alpha B(\mathbf{x}),$$

где $B(\mathbf{x})$ — барьерная функция. По своему назначению функция $B(\mathbf{x})$ должна стремиться к бесконечности по мере приближения к границе области допустимости изнутри. Функция $B(\mathbf{x})$ выбирается непрерывной и неотрицательной в области допустимости. Наиболее часто используется функция вида

$$B(\mathbf{x}) = \sum_{i=n+1}^{p} G[g_i(\mathbf{x})],$$

где $G(\mathbf{z}) \geqslant 0$ при $\mathbf{z} < 0$ и $G(\mathbf{z}) \to \infty$ при $\mathbf{z} \uparrow 0$ (последнее выражение означает стремление к нулю слева).

В качестве примера типичных функций G, формирующих барьерную функцию, приведем следующую: $G[g_i(\mathbf{x})] = -1/g_i(\mathbf{x})$, $i = n+1, \ldots, p$, где функции $g_i(\mathbf{x})$ непрерывны в области D. Легко видеть, что соответствующая этим функциям барьерная функция $B(\mathbf{x}) \to \infty$ при приближении к границе области допустимости из ее внутренней точки. Можно показать, что аналогично методу штрафных функций при самых широких предположениях алгоритм барьерных функций сходится к решению задачи.

Алгоритм барьерных функций состоит из следующих этапов:

1. Задаются начальная точка поиска $\mathbf{x}_0$, такая, что $g_i(\mathbf{x}_0) < 0$ $(i = n+1, \ldots, p)$, начальное значение параметра штрафа $\alpha_0 > 0$, коэффициент увеличения параметра штрафа $k_\alpha > 1$ и параметр точности решения задачи $\varepsilon > 0$.

2. На $(k+1)$=м шаге и при начальной точке $\mathbf{x}_k$ решается задача минимизации

$$\tilde{F}_{\alpha_k}(\mathbf{x}) = F(\mathbf{x}) + \alpha_k B(\mathbf{x}), \quad \mathbf{x} \in \mathbf{R}^m,$$

при наличии строгих ограничений $g_i(\mathbf{x}) < 0$ $(i = n+1, \ldots, p)$. Полагаем $\mathbf{x}_{k+1}$ равным решению этой задачи. Переходим к п. 3.

3. Проверяем выполнение условия $\alpha_k \mathbf{B}(\mathbf{x}_{k+1}) < \varepsilon$: если оно выполняется, алгоритм заканчивает процесс поиска, в противном случае вычисляем $\alpha_{k+1} = k_\alpha \alpha_k$, заменяем k на $k+1$ и переходим к п. 2.

Формально в алгоритме барьерных функций формируется система ограничений $g_{n+1}(\mathbf{x}) < 0, \ldots, g_p(\mathbf{x}) < 0$. Эти ограничения неэффективны, если оптимальное решение на каждой итерации принадлежит области допустимости, В то же время при

применении численных методов минимизации $\bar{F}_{\alpha_k}(\mathbf{x})$ выход из области допустимых значений не исключен, особенно при относительно небольших значениях α_k. Таким образом, процедура безусловной минимизации функции $F_{\alpha_k}(\mathbf{x})$ на каждой итерации должна подкрепляться процедурой проверки допустимости текущего решения.

Поиск начальной точки поиска $\mathbf{x}_0$, такой, что $g_i(\mathbf{x}_0)<0$ $(i=n+1, \ldots, p)$, может быть осуществлен, например, с помощью следующего алгоритма.

1. Выбирается $\mathbf{y}_0 \in \mathbf{R}^m$, $k=0$.

2. Определяется множество индексов $\mathbf{I}=\{i: g_i(\mathbf{y}_k)<0\}$. Если $\mathbf{I}=\mathbf{P}=\{1, \ldots, m\}$, то искомая точка $\mathbf{x}_0$ найдена: $\mathbf{x}_0=\mathbf{y}_k$, в противном случае выбирается j, такое, что $j \in \mathbf{I}$ и $j \in \mathbf{P}$, и переходим к п. 3.

3. Решается задача минимизации $g_j(\mathbf{y})$ при наличии ограничений $g_i(\mathbf{y})<0$ $(i \in \mathbf{I})$. Пусть $\mathbf{y}_{k+1}$ — решения этой задачи. Если $g_i(\mathbf{y}_{k+1}) \geqslant 0$, то множество допустимых значений имеет пустую внутренность $\{\mathbf{y}: \mathbf{y} \in \mathbf{R}^m, g_i(\mathbf{y})<0, i=1, \ldots, m\}$, в противном случае заменяем k на $k+1$ и переходим к п. 2.

Данный алгоритм самое большее за m итераций либо находит точку, принадлежащую внутренности множества $\mathbf{D}$, либо устанавливает факт отсутствия таких точек.

На практике иногда используются комбинации методов штрафных и барьерных функций, которые построены на основе минимизации смешанной вспомогательной функции вида

$$F(\mathbf{x}) = \beta A(\mathbf{x}) + \alpha B(\mathbf{x}),$$

где $A(\mathbf{x})$ — штрафная функция для ограничений-равенств; $B(\mathbf{x})$ — барьерная для ограничений-неравенств; α, β — параметры соответствующих штрафов. В ряде задач такой подход позволяет улучшить характеристики сходимости алгоритма (подробнее см. [60, 64]).

В качестве примера использования метода штрафных функций рассмотрим задачу минимизации функции трех переменных

$$F(x_1, x_2, x_3) = (x_1 - 0{,}04)^2 + (x_2 - 4)^2 + (x_3 - 400)^2$$

при наличии линейных ограничений $x_1 \leqslant 0{,}03$, $x_2 \leqslant 3$, $x_3 \leqslant 300$. Образуем вспомогательную функцию в соответствии с алгоритмом

$$\bar{F}(x_1, x_2, x_3) = F(x_1, x_2, x_3) + \alpha A(x_1, x_2, x_3).$$

Здесь

$$A(x_1, x_2, x_3) = (0{,}03 - x_1)^+ + (3 - x_2)^+ + (300 - x_3)^+,$$

где $y^+=y$, если $y \geqslant 0$, и $y^+=0$, если $y<0$.

Для решения задачи безусловной минимизации функции $\bar{F}(x_1, x_2, x_3)$ применим алгоритм метода прямого поиска (§ 6.4) и программу DIRECT. Для этого достаточно лишь внести соответствующие изменения в процедуру-функцию **S**, которая в этом случае будет иметь следующий вид:

```
S:    PROCEDURE(PHI);
      DECLARE PHI(3) BINARY FLOAT;
      RM=10;
      P1,P2,P3=0;
      IF PHI(1) > 0.03 THEN P1=RM*(PHI(1))=0.03)**2;
      IF PHI(2) > 3 THEN P2=RM*(PHI(2))=3)**2;
      IF PHI(3) > 300 THEN P3=RM*(PHI(1))=300)**2;
      F=F*P1*P2*P3;
      RETURN;
END   S;
```

Здесь RM — идентификатор параметра штрафа. Результаты расчетов оптимальных значений x_1, x_2, x_3 в зависимости от параметра штрафа PM приведены на рис. 6.3. Каждая из переменных требует существенно различных коэффициентов штрафа, достаточных для решения задачи условной минимизации. При этом для выхода на оптимальное значение по переменной x_3 достаточно, чтобы RM=3...5.

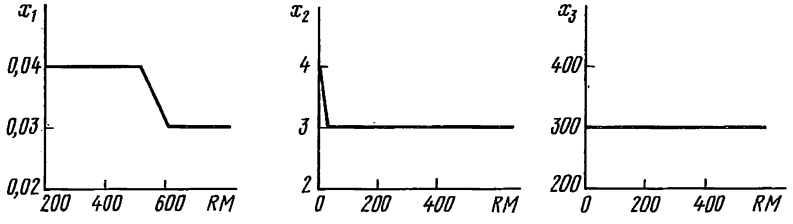


Рис. 6.3. Оптимальные значения x_1, x_2, x_3

При проведении практических расчетов по методу штрафных функций необходимо тщательно следить, чтобы алгоритм, используемый для безусловной минимизации, соответствовал свойствам вспомогательной критериальной функции, формируемой с помощью функций основного критерия и функций ограничений. Например, при использовании в алгоритме на этапе безусловной оптимизации процедуры Дэвидона—Флетчера—Пауэлла требуется существование у $\bar{F}_\beta(\mathbf{x})$ непрерывной производной, при использовании метода золотого сечения — непрерывности.

6.9. Оценка погрешности решения задач оптимизации

Рассмотрим метод оценки погрешности решения задачи оптимизации в зависимости от погрешностей исходной информации и используемой математической модели*. Под исследуемой задачей оптимизации понимается обычная задача нелинейного математического программирования. Особенность предлагаемого метода состоит в том, что он позволяет оценить значение некорректируемой погрешности критерия качества, возникающей из-за смещения точки экстремума в пространстве оптимизируемых параметров. Оценка погрешностей может производиться для прикладных задач оптимизации, имеющих алгоритмическое описание функции критерия качества.

*Разработан канд. техн. наук. В. Г. Байковым.

Рассматривается погрешность решения задачи оптимизации

$$\bar{x}^* = \arg\max_{x \in R^n} f(\bar{x}, \bar{\xi}^*), \tag{6.9}$$

где $\bar{x}$ — n-мерный вектор оптимизируемых параметров; $f(\bar{x}, \bar{\xi}^*)$ — функция качества; R^n — евклидово пространство размерности n; $\bar{\xi}^*$ — ожидаемое значение вектора неопределенных параметров, характеризующих погрешности используемой расчетной модели.

Точное решение задачи

$$\bar{x}^R = \arg\max_{\bar{x} \in R^n} f(\bar{x}, \bar{\xi}^R), \tag{6.10}$$

где $\bar{\xi}^R$ — действительное значение вектора $\bar{\xi}$.

Погрешность вектора $\Delta\bar{\xi} = \bar{\xi}^R - \bar{\xi}^*$ обусловливает погрешность решения задачи оптимизации (6.9), которую можно охарактеризовать погрешностью экстремального значения критерия качества

$$\Delta_\Sigma(\Delta\bar{\xi}) = f(\bar{x}^R, \bar{\xi}^R) - f(\bar{x}^*, \bar{\xi}^*). \tag{6.11}$$

Представим эту погрешность в виде суммы корректируемой $\Delta_к$ и некорректируемой Δ_0 погрешностей значения функции качества при решении задачи (6.10):

$$\Delta_\Sigma = \Delta_к + \Delta_0,$$

$$\Delta_к(\Delta\bar{\xi}) = f(\bar{x}^*, \bar{\xi}^*) - f(\bar{x}^*, \bar{\xi}); \tag{6.12}$$

$$\Delta_0(\Delta\bar{\xi}) = f(\bar{x}^R, \bar{\xi}^R) - f(\bar{x}^*, \bar{\xi}^R). \tag{6.13}$$

(рис. 6.4). Погрешность $\Delta_к$ отсутствует при точном расчете без последующей оптимизации с использованием точной модели. Если процесс определения параметров проектируемого объекта рассматривать как совокупность оптимизации с использованием приближенной модели и последующего точного расчета, то корректируемая погрешность оптимизации может быть сколь угодно большой — ее значение не оказы-

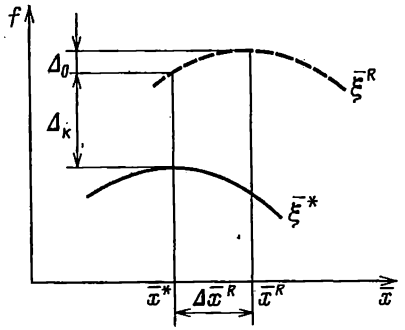


Рис. 6.4. Погрешности функции качества при решении задачи оптимизации

вает влияния на качество решения проектной задачи. Корректируемая погрешность может быть интерпретирована как погрешность расчета значения функции качества в точке $\mathbf{x}$ по модели, используемой при оптимизации.

Погрешность Δ_0 обусловливается сдвигом точки экстремума функции качества в пространстве оптимизируемых параметров. Может быть интерпретирована и как потеря значения функции качества из-за неоптимальности полученного $\bar{\mathbf{x}}^*$ для реального объекта, что объясняется несоответствием расчетной модели объекта, используемой при оптимизации, реальному объекту. Погрешность $\Delta_0 \geqslant 0$ является показателем качества проводимой оптимизации. Некорректируемая погрешность может быть устранена только в результате оптимизации с использованием точной модели расчета.

Математическую модель объекта проектирования, используемую в процессе оптимизации, можно представить в виде следующей вектор-функции:

$$\mathbf{y} = \bar{\mathbf{y}}(\bar{\mathbf{x}}, \bar{\xi}),$$

где $\bar{\mathbf{y}}$ — расчетные параметры объекта проектирования. В рассматриваемом комплексе программ для оценки погрешностей оптимизации производится также оценка погрешностей расчетных параметров $\bar{\mathbf{y}}$.

Неопределенные параметры могут быть двух типов: обычные и производные. Обычные неопределенные параметры представляют собой погрешности значений расчетных параметров объекта проектирования $\bar{\mathbf{y}}$ из-за неточности используемой математической модели или погрешности значений исходных параметров. Эти параметры влияют как на корректируемую $\Delta_к$, так и на некорректируемую Δ_0 погрешность оптимизации. Производные неопределенные параметры представляют собой погрешности первых производных обычных неопределенных параметров по оптимизируемым или расчетным параметрам. Эти параметры влияют только на некорректируемую Δ_0 погрешность оптимизации.

Оптимизируемые параметры, определяющие облик объекта проектирования по возможности их изменения в процессе создания объекта, можно разделить на два типа: жесткие и гибкие. К жестким будем относить параметры, изменение которых связано с существенной переделкой первоначально принятого проекта, и поэтому в рамках данного проекта в процессе его уточнения они практически не могут быть изменены. К гибким будем относить параметры, которые могут быть изменены в процессе создания проектируемого объекта. К таким параметрам относятся обычно регулируемые параметры настройки, если таковые имеются в проектируемом объекте.

Будем считать, что в векторе оптимизируемых параметров $\mathbf{x}=(x_1, \ldots, x_n)^T$ параметры $\bar{\mathbf{x}}^Y=(x_1, \ldots, x_{n1})^T$ — гибкие, а параметры $\mathbf{x}^F=(x_{n1+1}, \ldots, x_n)^T$ — жесткие. С точки зрения достоверности принимаемых значений оптимизируемых параметров на начальных стадиях проектирования объекта важна только достоверность значений жестких оптимизируемых параметров, поскольку гибкие параметры можно изменить в процессе дальнейшей проработки проекта или при эксплуатации объекта. При оценке достоверности получаемых значений жестких оптимизируемых параметров некорректируемую погрешность оптимизации необходимо определять, учитывая последующую дополнительную оптимизацию гибких параметров:

$$\Delta_0(\Delta\bar{\xi}) = f\left(\bar{\mathbf{x}}^{vR}, \bar{\mathbf{x}}^{FR}, \bar{\xi}^R\right) - f\left(\bar{\mathbf{x}}^{vR^*}, \bar{\mathbf{x}}^{F^*}, \bar{\xi}^R\right), \qquad (6.14)$$

где $\bar{x}^v \cup \bar{x}^F = \bar{x}$; $\bar{x}^v \cap \bar{x}^F = 0$; $\bar{x}^{v*} \cup \bar{x}^{F*} = \bar{x}^*$ — решение задачи (6.9); $\bar{x}^{vR} \cup \bar{x}^{FR} = \bar{x}^R$ — решение задачи (6.10); $\bar{x}^{vR*}$ — оптимальные значения гибких параметров при $\bar{x}^F = \bar{x}^{F*}$. Можно записать некорректируемую погрешность оптимизации и в виде разности двух погрешностей:

$$\Delta_0(\Delta\bar{\xi}) = \Delta_0^{Fv}(\Delta\bar{\xi}) - \Delta_0^{v}(\Delta\bar{\xi}), \tag{6.15}$$

где

$$\Delta_0^{Fv}(\Delta\bar{\xi}) = f\left(\bar{x}^{vR}, \bar{x}^{FR}, \bar{\xi}^R\right) - f\left(\bar{x}^{v*}, \bar{x}^{F*}, \bar{\xi}^R\right); \tag{6.16}$$

$$\Delta_0^{v}(\Delta\bar{\xi}) = f\left(\bar{x}^{vR*}, \bar{x}^{F*}, \bar{\xi}^R\right) - f\left(\bar{x}^{v*}, \bar{x}^{F*}, \bar{\xi}^R\right). \tag{6.17}$$

Погрешности Δ_0^{Fv} — обычно некорректируемые погрешности оптимизации параметров $\bar{x} = \bar{x}^v \cup \bar{x}^F$, а Δ_0^v — некорректируемая погрешность оптимизации гибких параметров $\bar{x}^v$ при фиксируемых значениях жестких параметров $\bar{x}^{F*}$, определяется аналогично Δ_0^{Fv}, но размерность вектора оптимизируемых параметров меньше.

Без учета членов третьего порядка малости некорректируемая погрешность оптимизации (6.13) определяется значением квадратичной формы следующего вида:

$$\Delta_0(\Delta\bar{\xi}) = \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij} \Delta\bar{\xi}_i \Delta\bar{\xi}_j, \tag{6.18}$$

где i_{ij}, i, $j = 1, \ldots, k$, — постоянные коэффициенты, значения которых определяются в программе; k — число неопределенных параметров.

Погрешности вектора неопределенных параметров $\Delta\bar{\xi}_i$ ($i = 1, \ldots, k$) считаются независимыми случайными величинами. Некорректируемая погрешность оптимизации зависит от $\Delta\bar{\xi}$ и, следовательно, сама является случайной величиной. Для оценки некорректируемой погрешности оптимизации используются следующие параметры: математическое ожидание и верхний доверительный предел некорректируемой погрешности оптимизации.

Математическое ожидание некорректируемой погрешности оптимизации

$$M(\Delta_0) = \sum_{i=1}^{k} c_{ii} \sigma^2(\Delta\bar{\xi}_i), \tag{6.19}$$

где $\sigma^2(\Delta\bar{\xi}_i)$ — дисперсия $\Delta\bar{\xi}_i$; c_{ii} — коэффициенты, определяемые в программе.

Верхний доверительный предел некорректируемой погрешности оптимизации $\bar{\Delta}_0(p)$, где p — доверительная вероятность, равен значению p-й квантили распределения $\Delta_0(\Delta\bar{\xi})$. Для оценки $\bar{\Delta}_0(p)$ используется метод статистического моделирования некорректируемой погрешности оптимизации. Обозначим $\Delta_i (i = 1, \ldots, m)$ элементы случайной выборки некорректируемой погрешности оптимизации размером m, упорядоченные по возрастанию. В качестве точечной оценки $\bar{\Delta}_0(p)$ используется

$$\bar{\Delta}_0(p) = \Delta r,$$

где $r = \mathrm{int}(mp)$ — ближайшее целое к числу mp, в качестве интервальной оценки $\bar{\Delta}_0(p)$ — интервал $[\Delta_{r-s}, \Delta_{r+s}]$. Доверительная вероятность этого интервала

$$p_I = I_p(r - s, m - r + s + 1) - I_p(r + s, m - r - s + 1),$$

где $I_p(a, b)$ — функция бета-распределения с параметрами a и b. Доверительные вероятности интервалов p_I для некоторых значений m, r, s приведены в табл. 6.1.

Таблица 6.1

p	$m(r)$	s	p_I	p	$m(r)$	s	p_I
0,9	100 (90)	5	0,9028	0,95	100 (95)	3	0,8200
		6	0,9555			4	0,9349
		7	0,9821			5	0,9824
		8	0,9933				
	200 (180)	7	0,9005		200 (190)	6	0,9500
		8	0,9404			8	0,9917
		9	0,9668			10	0,9988
		10	0,9822				
		11	0,9911				
	500 (450)	11	0,8987		500 (475)	8	0,8986
		12	0,9264			9	0,9356
		14	0,9633			10	0,9606
		16	0,9831			12	0,9867
		18	0,9929			14	0,9961

Предложенный метод может использоваться для оценки некорректируемой погрешности решений задачи оптимизации с ограничениями вида

$$\bar{x}^* = \arg\max_{\bar{x}\in X} f(\bar{x}, \bar{\xi}^*), \tag{6.20}$$

$$X = \{\bar{x} \in R^n : g_i(\bar{x}, \bar{\xi}^*) = 0,\ i = 1, ..., m\},$$

где $g_i(\bar{x}, \bar{\xi})$ $(i=1, ..., m)$ — функция ограничений.

В качестве точного решения задачи (6.20) используется решение следующей задачи:

$$\bar{x}^R = \arg\max_{\bar{x}\in R^n} f(\bar{x}, \bar{\xi}^R), \tag{6.21}$$

$$X^R = \{\bar{x} \in R^n : g_i(\bar{x}, \bar{\xi}^R) = g_i(\bar{x}^*, \bar{\xi}),\ i = 1, ..., m\}.$$

Использование в задаче (6.21) допустимого множества $X^R(\bar{\xi}^*, \bar{\xi}^R)$ вместо множества $X(\bar{\xi}^*)$, используемого в задаче (6.20), позволяет поставить сравниваемые значения функции качества $f(\bar{x}^R, \bar{\xi}^R)$ и $f(\bar{x}^*, \bar{\xi}^R)$ в одинаковые условия по значениям ограничений. Это необходимо для корректного сравнения, поскольку значения ограничений влияют на экстремальное значение функции качества.

Некорректируемая погрешность оптимизации решения задачи с ограничениями может быть интерпретирована так же, как и в задаче без ограничений, как неточное определение значений функции качества из-за неоптимальности $\bar{x}^*$ для реального объекта. Параметры оптимального объекта $\bar{x}^R$ при этом определяются для тех же условий (ограничения (6.21)), в которых функционирует реальный объект с $\bar{\xi}=\bar{\xi}^R$ и которые могут отличаться от условий, принятых при проектировании.

Назначение: оценка погрешностей решения задачи оптимизации.

Параметры (объединяются в группы):

1) II, NX, NX1, NTO, NTP, NY:

II — код печати выходной информации;

NX — общее число оптимизируемых параметров;

NX1 — число гибких оптимизируемых параметров (гибкие оптимизируемые параметры располагаются обязательно в начале общего вектора оптимизируемых параметров);

NTO — число обычных неопределенных параметров;

NTP — число производных неопределенных параметров;

NY — число исследуемых расчетных параметров.

2) Массивы JZX(i), ZX(i), SRX(i), XO(i) размера NX:

JZX(i) — признак определения относительных отклонений i-го оптимизируемого параметра: 0 — относительно значения номинала, 1 — относительно заданного значения;

ZX(i) — заданное значение для определения относительных отклонений i-го оптимизируемого параметра (если отклонения определяются относительно номинала, то ZX(i)=0);

SRX(i) — относительный шаг по i-му оптимизируемому параметру для расчета производных;

XO(i) — координата точки экстремума.

3) Массивы JZY(i), ZY(i) размера NY.

4) JZF, ZF.

5) Массивы JZTO(i), ZTO(i), SRTO(i), DRTO(i) размера NTO:

SRTO(i) — относительный шаг по i-му обычному неопределенному параметру для расчета производных;

DRTO(i) — относительное стандартное отклонение i-го обычного неопределенного параметра.

6) Массивы JTPI(i), JTP(i, 1), JTP(i, 2), DRTP(i), размерности NTP:

JTPI(i) — идентификация типа i-го производного неопределенного параметра: 1 — производная JTP(i, 1)-го обычного неопределенного параметра по JTP(i, 2)-му оптимизируемому параметру, 2 — производная JTP(i, 1)-го обычного неопределенного параметра по JTP(i, 2)-му исследуемому расчетному параметру;

DRTP(i) — относительное стандартное отклонение i-го производного неопределенного параметра (относительно $\frac{\text{ZTO}}{\text{ZX}}$ или $\frac{\text{ZTO}}{\text{ZY}}$).

Вся выходная информация обозначена идентификаторами:

DX/DT — производная $\partial x_{opi}/\partial \xi$;

MDRF — математическое ожидание общей относительной некорректируемой погрешности оптимизации;

MDRF1 — математическое ожидание относительной некорректируемой погрешности оптимизации гибких оптимизируемых параметров;

MDRFS — математическое ожидание относительной некорректируемой погрешности

оптимизации с учетом дополнительной оптимизации гибких оптимизируемых параметров. В соответствии с (6.16) MDRFS = MDRF — MDRF1.

Везде индекс D обозначает производную или стандартное отклонение параметра.

Дополнительные подпрограммы:

PROG — ввод исходной информации и обращение к основной процедуре EAGLE;

EAGLE — основная процедура оценки погрешностей оптимизации, обращение к ней может производиться из других программ, описания всех входных переменных приводятся в комментариях в тексте процедуры;

MFG, MDLG — процедуры решения систем линейных уравнений;

WBRN — процедура статистической выборки некорректируемой погрешности оптимизации при нормальном распределении неопределенных параметров;

WBRC — процедура статистической выборки некорректируемой погрешности оптимизации при равномерном распределении неопределенных параметров;

GAUSS — датчик случайной величины, имеющей нормальное распределение;

RANDU — датчик случайной величины, имеющей равномерное распределение на интервале [0, 1];

PROCF — расчет критерия качества, имя задается в качестве параметра процедуры EAGLE.

Подключение процедуры расчета значения критерия качества производится через вспомогательную процедуру PROCPL. Описание параметров процедуры PROGF приведено в комментариях в тексте процедуры EAGLE.

В программе статистическое моделирование некорректируемой погрешности оптимизации производится для нормального и для равномерного распределений всех $\Delta\xi_i (i = 1, \ldots, k)$, стандартные отклонения неопределенных параметров задаются в исходных данных.

Программа может использоваться для оценки погрешностей решения задачи оптимизации с ограничениями, но при этом необходимо учесть следующее:

1. Вместо функции качества $f(\bar{x}, \bar{\xi})$ нужно использовать следующую функцию Лагранжа:

$$L(\bar{x}, \bar{\xi}) = f(\bar{x}, \bar{\xi}) + \sum_{i=1}^{m} \lambda_i \left(g_i(\bar{x}, \bar{\xi}) - g_i(\bar{x}^*, \bar{\xi}) \right),$$

где λ_i $(i = 1, \ldots, m)$ — множители Лагранжа.

2. Вместо вектора оптимизируемых параметров $\bar{x}$ нужно использовать следующий вектор:

$$\bar{y} = \bar{x} \cup \lambda = (x_1, \ldots, x_n; \quad \lambda_1, \ldots, \lambda_m)^T.$$

```
EAGLE: PRUC(II,IT,NX,NX1,NTO,NTP,NY,XO,JTPI,JTP,JZX,JZTO,
           JZY,JZF,ZX,ZTO,ZY,ZF,SRX,SRTO,DRTO,DRTP,PROCF);
DCL (II,IT,NX,NX1,NTO,NTP,NY,JZX(*),JZTO(*),JZY(*),JZF,
    JTPI(*),JTP(*,*),NT)          BIN FIXED (15,0),
   (XO(*),ZX(*),ZTO(*),ZY(*),ZF,SRX(*),SRTO(*),DRTO(*),DRTP(*))
                                  DEC FLOAT (6),
   PROCF ENTRY;
BLOC: BEGIN;
```

```
  DEC (I,J,K,NS,
       IS,IPER(NX),IS1,IPER1(NX1)) BIN FIXED (15,0),
  (YO(NY),FO,X(NX),TJ(NT),T(NT),T1(NT),Y(NT),F,F1,
   ZTR(NTO),DRX(NT),DRY(NT),DRF,DRT(NT),
   FDX(NT),FDT(NT),DFDX(NT),DFDT(NT),
   D2F1DTDX(NX1,NT),DX1DT(NX1,NT),C(NT,NT),
   D2FDXDX(NX,NX),D2FDTDX(NX,NT),CORX(NX,NX),
   DYDX(NY,NX),DYDT(NY,NT),DXDT(NX,NT))  * DEC FLOAT (6),
  (A(NX,NX) ,BX(NX,NT))                  DEC FLOAT (12) CTL,
  (A1(NX1,NX1),BX1(NX1,NT))              DEC FLOAT (12) CTL,
  OPT CHAR (1);
DO I=1 TO NTO; DRT(I)=DRTO(I); END;
DO I=1 TO NTP; DRT(NTO+I)=DRTP(I); END;
ZTR=0;
CALL PROCF(NX,NT,NY,XO,ZTR,JZTO,ZTO,YO,FO,IT);
IF IT =0 THEN DO;
 PUT SKIP LIST(`   ***EAGLE***   NOT FO`); RETURN; END;
IF JZF=0 THEN ZF=FO;
DO I=1 TO NY;
IF JZY(I)=0 THEN ZY(I)=TO(I);
END;
DO I=1 TO NX;
 IF JZX(I)=0 THEN ZX(I)=XO(I);
END;
DO I=1 TO NTO;
 IF JZTO(I)=0 THEN ZTO(I)=0;
END;
PUT SKIP(3) LIST(`        FO`);
PUT SKIP    EDIT   (FO)  (SKIP,    E(12,6));
PUT SKIP(3) LIST(`        ZF`);
PUT SKIP    EDIT   (ZF)  (SKIP,    E(12,6));
PUT SKIP(3) LIST(`        YO`);
PUT SKIP    EDIT   (YO)  (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`        ZY`);
PUT SKIP    EDIT   (ZY)  (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`        XO`);
PUT SKIP    EDIT   (XO)  (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`        ZX`);
PUT SKIP    EDIT   (ZX)  (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`        ZTO`);
PUT SKIP    EDIT   (ZTO)  (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`       DRT  (REL.)`);`);
PUT SKIP    EDIT   (DRT)  (SKIP,(10)F(12,6));
DO I=1 TO NTO;
 ZTR=0; ZTR(I)=SRTO(I);
 CALL PROCF(NX,NT,NY,XO,ZTR,JZTO,ZTO,Y,FDT(I),IT);
 IF IT=0 THEN DO;
  DFDT(I)=(FDT(I)-FO)/ZF/SRTO(I);
  DYDT(*,I)=(Y-YO)/ZY/SRTO(I);  END;
 ELSE DO; DFDT(I)=0; DYDT(*,I)=0;
  PUT SKIP LIST(`   ***EAGLE***   NOT DF/DT`,I);
 END;
END;
DO I=1 TO NTP; DFDT(NTO+I)=0; DYDT(*,NTO+I)=0; END;
PUT SKIP(3) LIST(`        DF/DT`);
PUT SKIP    EDIT(DFDT)(SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`        DY/DT`);
DO I=1 TO NY;
 PUT SKIP    EDIT(DYDT(I,*))(SKIP,(10)F(12,6));
END;
PUT SKIP(3) LIST(`        DF/DT*DT*ZF`);
```

```
PUT SKIP    EDIT(DFDT*DRT*ZF)(SKIP,(10)F(12,4));
PUT SKIP(3) LIST(`        DY/DT*DT*ZY`);
DO I=1 TO NY;
PUT SKIP    EDIT(DFDT(I,*)*DRT*ZY(I))(SKIP,(10)F(12,4));
END;
DRF=SQRT(SUM(DFDT*DFDT*DRT*DRT));
DO I=1 TO NY;
 DRY(I)=SQRT(SUM(DYDT(I,*)*DYDT(I,*)*DRT*DRT));
END;
PUT SKIP(3) LIST(        DRF(REL.) `);
PUT SKIP    EDIT(DRF)(SKIP,    F(12,6));
PUT SKIP(3) LIST(`        DRY(REL.)`);
PUT SKIP    EDIT(DRY)(SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`        DRF*ZF`);
PUT SKIP    EDIT(DRF*ZF)(SKIP,    F(12,4));
PUT SKIP(3) LIST(`        DRY*ZY`);
PUT SKIP    EDIT(DRY*ZY)(SKIP,(10)F(12,4));
ZTR=0;
DO I=1 TO NX;
 X=XO; X(I)=X(I)+ZX(I)*SRX(I);
 CALL PROCF(NX,NT,NY,X,ZTR,JZTO,ZTO,Y,FDX(I),IT);
 IF IT=0 THEN DO;
  DFDX(I)=(FDX(I)-FD)/ZF/SRX(I);
  DYDX(*,I)=(Y-YO)/ZY/SRX(I);  END;
 ELSE DO; DFDX(I)=0; DYDX(*,I)=0;
PUT SKIP LIST(`    ***EAGLE***    NOT DF/DX`,I); END;
END;
PUT SKIP(3) LIST(        DF/DX`);
PUT SKIP    EDIT(DFDX)(SKIP,(10)F(12,6));
PUT SKIP(3) LIST(        DY/DX`);
DO I=1 TO NY;
PUT SKIP    EDIT(DYDX(*,I))(SKIP,(10)F(12,6));
ZTR=0;
DO I=1 TO NX; DO J=1 TO I;
 X=XO; X(I)=X(I)+ZX(I)*SRX(I); X(J)=X(J)+ZX(J)*SRX(J);
 CALL PROCF(NX,NT,NY,X,ZTR,JZTO,ZTO,Y,F,IT);
 IF IT=0 THEN
  D2FDXDX(I,J),D2FDXDX(J,I)=
      ((F-FDX(J))/ZF/SRX(I)-DFDX(I))/SRX(J);
  ELSE DO; D2FDXDX(I,J),D2FDXDX(J,I)=0;
PUT SKIP LIST(`    ***EAGLE***    NOT D2F/DXDX`,I,J);
END;
IF  I  =J THEN D2FDXDX(I,J),D2FDXDX(J,I)=D2FDXDX(I,J)*1.;
END;  END;
PUT SKIP    EDIT(DFDX)(SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`        D2F/DXDX`);
DO I=1 TO NX;
PUT SKIP    EDIT(D2FDXDX(I,*))(SKIP,(10)F(12,6));
END;
DO I=1 TO NX;
 DO J=1 TO NTO;
  X=XO; X(I)=X(I)+ZX(I)*SRX(I);
  ZTR=0; ZTR(J)=SRTO(J);
  CALL PROCF(NX,NT,NY,X,ZTR,JZTO,ZTO,Y,F,IT);
  IF IT=0 THEN
  D2FDTDX(I,J)=((F-FDT(J))/ZF/SRX(I)-DFDX(I))/SRTO(J);
  ELSE DO; D2FDTDX(I,J)=0;
  PUT SKIP LIST(`    ***EAGLE***    NOT D2F/DTDX`J,I);
 END;
END;
DO J=1 TO NTP;
```

```
      IF JTPI(I)=1 THEN DO;
       IF JTP(J,2)=I THEN D2FDTDX(I,NTO+J)=DFDT(JTP(J,1));
       ELSE D2FDTDX(I,NTO+J)=0;
       GOTO LXTE; END;
       IF JTPI(J)=2 THEN DO;
        D2FDTDX(I,NTO+J)=DFDT(JTP(J,1))*DYDX(JTP(J,2),I);
        GO TO LXTE; END;
       D2FDTDX(I,NTO+J)=0;
LXTE: END;
      END;
     PUT SKIP(3) LIST(         D2F/DTDX`);
     DO I=1 TO NX;
     PUT SKIP    EDIT(D2FDTDX(I,*))(SKIP,(10)F(12,6));
     END;
     ALLOCATE A,BX;
     A=D2FDXDX;
     CALL MFG(A,IPER,NX,IS);
     OPT=`0`;
     BX=-D2FDTDX;
     CALL MDLG(A,BX,IPER,NX,NT,OPT);
     DXDT=BX;
     FREE A,BX;
     PUT SKIP(3) LIST(`        DX/DT`);
     DO I=1 TO NX;
     PUT SKIP    EDIT(DXDT(I,*))(SKIP,(10)F(12,6));
     END;
     PUT SKIP(3) LIST(`        DX/DT*DT`);
     DO I=1 TO NX;
     PUT SKIP    EDIT(DXDT(I,*)*DRT)(SKIP,(10)F(12,6));
     END;
     PUT SKIP(3) LIST(`        DX/DT*DT*ZX`);
     DO I=1 TO NX;
     PUT SKIP    EDIT(DXDT(I,*)*DRT*ZX(I))(SKIP,(10)F(12,4));
     END;
     DO I=1 TO NX;
      DRX(I)=SQRT(SUM(DXDT(I,*)*DXDT(I,*)*DRT*DRT));
     END;
     PUT SKIP(3) LIST(`        DRX (REL.)`);
     PUT SKIP    EDIT(DRX)(SKIP,(10)F(12,6));
     PUT SKIP(3) LIST(`        DRX*ZX`);
     PUT SKIP    EDIT(DRX*ZX)(SKIP,(10)F(12,6));
     DO I=1 TO NY;
      DO J=1 TO NT;
       TJ(J)=SUM(DYDX(I,*)*DXDT(*,J))+DYDT(I,J);
      END;
      DRY(I)=SQRT(SUM(TJ*TJ*DRT*DRT));
     END;
     PUT SKIP(3) LIST(`        DRY (REL.)`);
     PUT SKIP    EDIT(DRY)(SKIP,(10)F(12,6));
     PUT SKIP(3) LIST(`        DRY*ZY`);
     PUT SKIP    EDIT(DRY*ZY)(SKIP,(10)F(12,4));
     DO I=1 TO NT;
      T(I)=.5*SUM(D2FDTDX(*,I)*DXDT(*,I))*DRT(I)**2;
     END;
     F=SUM(I);
     PUT SKIP(3) LIST(`        MDRFT (REL.)`);
     PUT SKIP    EDIT (T) (SKIP,(10)F(12,6));
     PUT SKIP(3) LIST(`        MDRF (REL.)    =SUM(MDRFT)`);
     PUT SKIP    EDIT (F) (SKIP,(10)F(12,6));
     PUT SKIP(3) LIST(`        MDRFT * ZF`);
     PUT SKIP    EDIT (T*ZF) (SKIP,(10)E(12,4));
```

```
PUT SKIP(3) LIST(`       MDRF * ZF `);
PUT SKIP    EDIT (F*ZF) (SKIP,(10)E(12,4));
DO I=1 TO NX; X(I)=.5*D2FDXDX(I,I); END;
PUT SKIP(3) LIST(`       DRF (REL.)`);
PUT SKIP    EDIT (X*DRX*DRX)(SKIP,(10)F(12,6));
DO I=1 TO NX; DO J=1 TO I;
 CORX(I,J),CORX(J,I)=
  SUM(DXDT(I,*)*DXDT(J,*)*DRT*DRT)/DRX(I)/DRX(J);
 END; END;
DO I=1 TO NX;
PUT SKIP    EDIT (CORX(I,*))(SKIP,(10)F(12,6));
END;
ALLOCATE A1,BX1;
DO I=1 TO NX1; DO J=1 TO NX1;
 A1(I,J)=D2FDXDX(I,J); END; END;
CALL MFG(A1,IPER1,NX1,IS1);
OPT=`0`;
DO I=1 TO NX1; DO J=1 TO NT;
 D2F1DTDX(I,J)=D2FDTDX(I,J); END; END;
BX1=-D2F1DTDX;
CALL MDLG(A1,BX1,IPER1,NX1,NT,OPT);
DX1DT=BX1;
FREE A1,BX1;
PUT SKIP(3) LIST(`       DX1/DT`);
DO I=1 TO NX1;
PUT SKIP    EDIT (DX1DT(I,*))(SKIP,(10)F(12,6));
END;
PUT SKIP(3) LIST(`       DX1/DT * DT  `);
DO I=1 TO NX1;
PUT SKIP    EDIT(DX1DT(I,*)*DRT)(SKIP,(10)F(12,6));
END;
PUT SKIP(3) LIST(`       DX1/DT * DT * ZX`);
DO I=1 TO NX1;
PUT SKIP    EDIT(DX1DT(I,*)*DRT*ZX(I))(SKIP,(10)E(12,4));
END;
DO I=1 TO NT;
 T1(I)=.5*SUM(D2F1DTDX(*,I)*DX1DT(*,I))*DRT(I)**2;
END;
F1=SUM(T1);
PUT SKIP(3) LIST(       MDRF1T (REL.) );
PUT SKIP    EDIT (T1) (SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`       MDRF1 (REL.) =SUM(MDRF1T)`);
PUT SKIP    EDIT (F1) (SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`       MDRF1T * ZF`);
PUT SKIP    EDIT (T1*ZF) (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`       MDRF1 * ZF`);
PUT SKIP    EDIT (F1*ZF) (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`       MDRFST (REL.)`);
PUT SKIP    EDIT (T-T1) (SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`       MDRFS (REL.)  =SUM(MDRFST)`);
PUT SKIP    EDIT (F-F1) (SKIP,(10)F(12,6));
PUT SKIP(3) LIST(`       MDRFST * ZF`);
PUT SKIP    EDIT ((T-T1)*ZF) (SKIP,(10)E(12,4));
PUT SKIP(3) LIST(`       MDRFS * ZF`);
PUT SKIP    EDIT ((F-F1)*ZF) (SKIP,(10)E(12,4));
NS=500;
C=0.;
DO I=1 TO NT; DO J=1 TO NT;
 DO K=1 TO NX;
  C(I,J)=C(I,J)+.5*(D2FDTDX(K,I)*DXDT(K,J)); END;
 END; END;
PUT SKIP(3) LIST(` STAT. SAMPLING DRF N`);
```

```
     CALL WBRN(NT,NS,C,DRT);
     PUT SKIP(3) LIST(' STAT. SAMPLING DRF C');
     CALL WBRC(NT,NS,C,DRT);
     C=0.
     DO I=1 TO NT; DO J=1 TO NT;
      DO K=1 TO NX;
      C(I,J)=C(I,J)+.5*(D2FDTDX(K,I)*DXDT(K,J)); END;
      DO K=1 TO NX1;
       C(I,J)=C(I,J)-.5*(D2F1DTDX(K,I)*DX1DT(K,J)); END;
      END; END;
     PUT SKIP(3) LIST(' æÆÇÆ. éçüÄÉèÇ DRFS N');
     CALL WBRN(NT,NS,C,DRT);
     PUT SKIP(3) LIST(' æÆÇÆ. éçüÄÉèÇ DRFS C');
     CALL WBRC(NT,NS,C,DRT);
     END BLOC;
     RETURN;
     END EAGLE;


WBRN: PROC(NT,NS,C,DRT);
      DCL (NT,NS,I,J,K,L,N)          BIN FIXED (15,0),
          (IX,IY)                    BIN FIXED (31,0),
          (C(*,*),DRT(*),D,DF(NS),XT(NT),X,AM,AS) DEC FLOAT (6);
      IX=1234567;
      AS=1,;  AM=0.;
      DO I=1 TO NS;
       DO J=1 TO NT;
        CALL GAUSS(IX,AS,AM,X);
        XT(J)=DRT(J)*X; END;
       D=0.;
       DO K=1 TO NT; DO L=1 TO NT;
        D=D+C(K,L)*XT(K)*XT(L); END; END;
       DO J=1 TO I-1;
        N=J; IF D<DF(J) THEN GO TO LE; END; N=I; LE:
       DO J=I-1 TO N BY -1 ; DF(J+1)=DF(J); END;
       DF(N)=D;
      END;
      PUT SKIP    EDIT (DF) (SKIP,(10)E(12,4));
      RETURN;
  END WBRN;

WBRC: PROC(NT,NS,C,DRT);
      DCL (NT,NS,I,J,K,L,N),         BIN FIXED (15,0),
          (IX,IY)                    BIN FIXED (31,0),
          (C(*,*),DRT(*),D,DF(NS),XT(NT),X) DEC FLOAT (6);
      IX=1234567;
      DO I=1 TO NS;
       DO J=1 TO NT;
        CALL RANDU(IX,IY,X); IX=IY;
        XT(J)=DRT(J)*(X-.5)*3.464; END;
       D=0.;
       DO K=1 TO NT; DO L=1 TO NT;
       D=D+C(K,L)*XT(K)*XT(L); END; END;
      DO J=1 TO I-1;
       N=J; IF D<DF(J) THEN GOTO LE; END; N=I; LE:
      DO J=I-1 TO N BY -1; DF(I+1)=DF(J); END;
      DF(N)=D;
     END;
    PUT SKIP    EDIT (DF) (SKIP,(10)E(12,4));
    RETURN;
END WBRC;
```

```
PROCPL: PROC(NX,NTO,NY,X,DTOR,JZTO,ZTO,Y,F,IT);
        DCL (NX,NTO,NY,JZTO(*),IT)        BIN FIXED (15,0),
            (X(*),DTOR(*),ZTO(*),Y(*),F)  DEC FLOAT (6);
        CALL FORTF(NX,NTO,NY,X(1),DTOR(1),JZTO(1),ZTO(1),Y(1),F,IT);
        RETURN;
  END   PROCPL;


       SUBROUTINE FORTF(NX,NTO,NY,X,DTOR,JZTO,ZTO,Y,F,IT)
       IMPLICIT INTEGER *2 (I-N)
       DIMENSION X(1),DTOR(1),JZTO(1),ZTO(1),Y(1),T(3)
       T(1)=0.
       T(2)=0.
       T(3)=10.
       T(1)=T(1)+T(1)*DTOR(1)*(1-JZTO(1))+ZTO(1)*DTOR(1)*JZTO(1)
       T(2)=T(2)+T(2)*DTOR(2)*(1-JZTO(2))+ZTO(2)*DTOR(2)*JZTO(2)
       T(3)=T(3)+T(3)*DTOR(3)*(1-JZTO(3))+ZTO(3)*DTOR(3)*JZTO(3)
       Y(1)=(X(1)+T(1))**2
       Y(2)=2*(X(2)+T(2))**2
       F=-Y(1)-Y(2)+T(3)+.5*X(1)*X(2)
       IT=0
       RETURN
       END


TEST:PROCEDURE OPTIONS(MAIN);
     DCL (II,IT,NX,NX1,NTO,NTP,NY)   BIN FIXED (15,0);
     GET EDIT (II,NX,NX1,NTO,NTP,NY) ((6)F(10));
     PUT SKIP(3)   DATA(II,NX,NX1,NTO,NTP,NY);
     BLOC: BEGIN;
     DCL (JTPI(NTP),JTP(NTP,2),JZX(NX),JZTO(NTO),JZY(NY),JZY(NY),JZF,I)
                                BIN FIXED (15,0),
         (XO(NX),ZX(NX),ZTO(NTO),ZY(NY),ZF,
          SRX(NX),SRTO(NTO),DRTO(NTO),DRTP(NTP)) DEC FLOAT (6),
         PROCPL ENTRY;
     GET EDIT ((JZX(I),ZX(I),SRX(I),XO(I) DO I=1 TO NX))
                                (SKIP,(4)F(10));
     PUT SKIP(3)   DATA (JZX,ZX,SRX,XO);
     GET EDIT (JZY(I),ZY(I)) DO I=1 TO NY))(SKIP,(2)F(10));
     PUT SKIP(3)   DATA (JZY,ZY);
     GET EDIT (JZF,ZF) (SKIP,(2)F(10));
     PUT SKIP(3)   DATA (JZF,ZF);
     GET EDIT ((JZTO(I),ZTO(I),SRTO(I),DRTO(I) DO I=1 TO NTO))
                                (SKIP,(4)F(10));
     PUT SKIP(3)   DATA (JZTO,ZTO,SRTO,DRTO);
     GET EDIT (JTPI(I),JTP(I,*),DRTP(I) DO I=1 TO NTP))
                                (SKIP,(4)F(10));
     CALL EAGLE(II,IT,NX,NX1,NTO,NTP,NY,XO,JTPI,JTP,JZX,JZTO,
               JZY,JZF,ZX,ZTO,ZY,ZF,SRX,SRTO,DRTO,DRTP,PROCPL);
         END BLOC;
         STOP;
     END TEST;
```

Глава 7.

Оптимизация дискретных параметров системы управления

7.1. Постановка задачи

Современная инженерная практика выдвигает огромное количество задач оптимизации дискретных параметров систем [67—73]. Все эти задачи объединяются общей постановкой и могут быть сведены к задачам целочисленного программирования, общая формулировка которых следующая.

Пусть $\mathbf{R}^m$ — m-мерное ($m \geqslant 1$) евклидово пространство, $\mathbf{Z}^m$ — его целочисленная решетка, т. е. наибольшее подмножество $\mathbf{R}^m$, состоящее из векторов $\mathbf{y}=\{y_1, ..., y_m\}^T$, все компоненты $\{y_i, i=1, ..., m\}$ которых целочисленные. Заданы функционал $F(\mathbf{y})$ и некоторое подмножество $\mathbf{D}$ решетки $\mathbf{Z}^m$. Ставится задача нахождения такого $\mathbf{y}_{opt}$, что

$$\mathbf{y}_{opt} \in \mathbf{D} \subseteq \mathbf{Z}^m, \tag{7.1}$$

$$F(\mathbf{y}_{opt}) = \operatorname*{opt}_{\mathbf{y} \in \mathbf{D}} F(\mathbf{y}). \tag{7.2}$$

Здесь opt — операция определения минимума или максимума. В дальнейшем без ограничения общности будем считать, что наша задача — минимизация функционала $F(\mathbf{y})$. Множество $\mathbf{D}$ — множество допустимых значений аргумента, задается, как правило, с помощью совокупности ограничений вида

$$F_1(\mathbf{y}), ..., F_l(\mathbf{y}) \leqslant 0, \; l \geqslant 0. \tag{7.3}$$

Наиболее распространенные классы задач *целочисленного программирования* (7.1)—(7.3) следующие:

1. Задачи линейного программирования — задачи (7.1)—(7.3), у которых

$$F(\mathbf{y}) = \sum_{i=1}^{m} c_i y_i \text{ и } F_j(\mathbf{y}) = \sum_{i=1}^{m} a_{ji} y_i,$$

где a_{ji} и $c_i (j=1, ..., l;\ i=1, ..., m)$ заданы; $\mathbf{y}=\{y_1, ..., y_m\}^T$. В случае нелинейности хотя бы одной из функций $F_j (j=1, ..., l)$ задача (7.1)—(7.3) становится задачей нелинейного целочисленного программирования.

2. Задачи с булевыми переменными $m=2$ и $y_i=0,1\,(i=1,2)$.

3. Комбинаторные задачи целочисленного программирования — решение задачи (7.1)—(7.3) ищется на конечном множестве задания функционала $F(\cdot)$.

4. Выпуклые, вогнутые задачи целочисленного программирования, задачи стохастического целочисленного программирования, определяемые соответствующей структурой критерия $F(\cdot)$ и ограничений F_i $(i=1, ..., l)$.

Вплотную к задачам целочисленного программирования примыкают задачи *частично целочисленного программирования* — задачи, у которых условие целочис-

ленности относится лишь к части компонентов оптимизируемого вектора $\mathbf{y}=\{y_1, \ldots, y_m\}^T$. Для решения этих задач можно выделить два эффективных подхода. Первый заключается в разбиении множества всех оптимизируемых параметров $\{y_1, \ldots, y_m\}$ на два непересекающихся подмножества — целочисленных параметров $\{\tilde{y}_1, \ldots, \tilde{y}_{m_1}\}$ и непрерывных $\{\hat{y}_1, \ldots, \hat{y}_{m_2}\}$:

$$\{\tilde{y}_1, \ldots, \tilde{y}_{m_1}\} \cup \{\hat{y}_1, \ldots, \hat{y}_{m_2}\} :=: \{y_1, \ldots, y_n\}, \; m_1 + m_2 = m.$$

Такое разбиение позволяет в дальнейшем производить оптимизацию попеременно непрерывных и целочисленных параметров, осуществляя итерационную процедуру нахождения решения смешанной задачи. Второй подход основан на преобразовании исходной задачи частично целочисленного программирования в задачу с целочисленными переменными путем дискретизации параметров $\{\hat{y}_1, \ldots, \hat{y}_{m_2}\}$ с некоторым интервалом $\{d_1, \ldots, d_{m_2}\}$ $(d_i>0, \; i=1, \ldots, m_2)$. При достаточно широких предположениях о структуре функции F решение этой новой задачи сходится к решению исходной при условии $\max\limits_i d_i \to 0$. Таким образом, второй подход заключается в сведении задач частично целочисленного программирования к задачам чисто целочисленного программирования, он удобен для реализации на ЭВМ и широко применяется при проведении практических расчетов.

В настоящее время в теории целочисленной оптимизации можно выделить следующие три основные группы методов: метод отсечений, комбинаторные и приближенные (эвристические).

Метод отсечений предназначен для решения задач линейного целочисленного программирования. Общая схема метода отсечения следующая. Вначале задача (7.1)—(7.3) решается без учета целочисленности переменных. Если полученное решение целочисленно, то оно является решением задачи (7.1)—(7.3). В противном случае (нецелочисленности решения) к ограничениям (7.3) добавляется новое ограничение, обладающее тем свойством, что множество допустимых решений новой задачи содержит любое допустимое решение целочисленной задачи и не содержит оптимального решения предыдущей задачи без ограничений целочисленности. Затем указанный процесс повторяется, но добавляются новые ограничения, названные по эффекту их действия отсечениями. В настоящее время наиболее известны метод Гомори, предназначенный для решения задач полностью и частично целочисленного линейного программирования, алгоритмы Бендерса и др. [67, 72].

Комбинаторные методы предназначены для решения задач целочисленного программирования с конечным множеством допустимых значений **D**. В основе комбинаторных методов лежит идея использования частичного перебора такого множества. Наиболее популярные комбинаторные методы и алгоритмы: методы ветвей и границ, аддитивный алгоритм Балаша, метод последовательного анализа вариантов, метод последовательных расчетов [72, 76].

В современной инженерной практике, особенно в практике автоматизированного выбора наилучших решений, возрос интерес к *приближенным методам дискретной оптимизации*. Приближенные методы позволяют получить приемлемые для практики результаты в широком диапазоне решаемых задач, в то время как точные методы ориентированы на узкоспециализированные задачи. Среди приближенных методов можно выделить методы локальной оптимизации, случайного

поиска, методы, являющиеся модификацией точных, а также различные их комбинации. (Приближенным методом посвящено много работ, среди них отметим [72], а также серию работ школы В. С. Михалевича и И. В. Сергиенко [69, 73].)

7.2. Оптимизация дискретных параметров методом Гомори

Во многих практических приложениях представляет большой интерес задача линейного целочисленного программирования, в которой дана система линейных неравенств и линейная форма

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{im}x_m \leqslant b_i,\ i = 1, \ldots, n;\ x_j \geqslant 0,\ j = 1, \ldots, m; \tag{7.4}$$

$$F = d_0 + d_1x_1 + \ldots + d_mx_m. \tag{7.5}$$

Требуется найти целочисленное решение системы неравенств (7.4), которое минимизирует целевую функцию F, причем все коэффициенты a_{ij}, b_i и c_l целые ($i=1, \ldots, n$; $j=1, \ldots, m$; $l=0, \ldots, m$).

Один из методов решения задачи целочисленного программирования предложен Р. Е. Гомори [67, 72]. Идея метода заключается в использовании методов непрерывного линейного программирования, в частности симплекс-метода. Если найденное с помощью симплекс-метода решение задачи (7.4), (7.5), у которой снято требование целочисленности решения, оказывается все-таки целочисленным, то искомое решение целочисленной задачи также будет найдено. В противном случае вводится дополнительное линейное ограничение

$$a_1x_1 + \ldots + a_mx_m \leqslant b, \tag{7.6}$$

которое отсекает от допустимого многогранника часть, бесперспективную для поиска решения задачи целочисленного программирования. Затем снова с помощью симплекс-метода решается задача линейного программирования при дополнительном ограничении (7.6), и так до тех пор, пока за конечное число шагов не будет получено целочисленное решение задачи линейного программирования, а следовательно, искомое.

Приведем систему (7.4), (7.5), к следующему эквивалентному виду:

$$\begin{gathered} x_1, \ldots, x_m \geqslant 0, \\ y_1 = b_1 - a_1 - a_1x_1 - \ldots - a_{1m}x_m \geqslant 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_m = b_m - a_{m1}x_1 - \ldots - a_{mn}x_m \geqslant 0, \end{gathered} \tag{7.7}$$

$$F = c_0 - c_1x_1 - \ldots - c_mx_m, \tag{7.8}$$

где требуется уже максимизировать функцию F. Все условия (7.7), (7.8) записываются с помощью следующей симплекс-таблицы (табл. 7.1), где первая строка $*$ — строка независимых переменных, а столбец $**$ — столбец зависимых переменных.

Симплекс-метод [56, 67] состоит из последовательного перебора вершин допустимого многогранника, определяемого неравенствами (7.7) в пространстве $\mathbf{R}^m$. Это производится с помощью перемен ролями подходящих независимой

Таблица 7.1 **Исходная симплекс-таблица**

** / *	1	$-x_1$	$-x_2$	...	$-x_p$	...	$-x_m$
z_1	b_1	a_{11}	a_{12}	...	a_{1p}	...	a_{1m}
z_2	b_2	b_{21}	b_{22}	...	b_{2p}	...	b_{2m}
...	...	...	...	...	...	...	...
z_q	b_q	b_{q1}	b_{q2}	...	b_{qp}	...	b_{qm}
...	...	...	...	...	...	...	...
z_n	b_n	b_{n1}	b_{n2}	...	b_{np}	...	b_{nm}
F	c_0	c_1	c_2	...	c_p	...	c_m

Таблица 7.2 **Преобразованная симплекс-таблица**

** / *	1	$-\tilde{x}_1$	$-\tilde{x}_2$	...	$-\tilde{x}_p$	...	$-\tilde{x}_m$
$\tilde{z}_1$	$\tilde{b}_1$	$\tilde{a}_{11}$	$\tilde{a}_{12}$	...	$\tilde{a}_{1p}$	...	$\hat{a}_{1m}$
$\tilde{z}_2$	$\tilde{b}_2$	$\tilde{b}_{21}$	$\tilde{b}_{22}$	...	$\tilde{b}_{2p}$	...	$\tilde{b}_{2m}$
...	...	...	...	...	...	...	...
$\tilde{z}_q$	$\tilde{b}_q$	$\tilde{b}_{q1}$	$\tilde{b}_{q2}$	...	$\tilde{b}_{qp}$	...	$\tilde{b}_{qm}$
...	...	...	...	...	...	...	...
$\tilde{z}_n$	$\tilde{b}_n$	$\tilde{b}_{n1}$	$\tilde{b}_{n2}$	...	$\tilde{b}_{np}$	...	$\tilde{b}_{nm}$
F	$\tilde{c}_0$	$\tilde{c}_1$	$\tilde{c}_2$	...	$\tilde{c}_p$	...	$\tilde{c}_m$

x_p и зависимой z_q переменных и соответствующей этой операции трансформации табл. 7.1 в табл. 7.2. Элементы табл. 7.2 рассчитывают по формулам

$$\tilde{b}_q = \frac{b_q}{b_{qp}}, \quad \tilde{b}_{qp} = \frac{1}{b_{qp}}, \quad \tilde{b}_{qi} = \frac{b_{qi}}{b_{qp}}, \quad i \neq p,$$

$$\tilde{b}_{jp} = -\frac{b_{jp}}{b_{qp}}, \quad j \neq q; \qquad \tilde{c}_p = -\frac{c_p}{b_{qp}};$$

$$b_{ij} = \begin{vmatrix} b_{ij} & b_{ip} \\ b_{qj} & b_{qp} \end{vmatrix} b_{qp}^{-1}, \quad i \neq q, \ j \neq p;$$

$$\tilde{b}_i = \begin{vmatrix} b_i & b_{ip} \\ b_q & b_{qp} \end{vmatrix} b_{qp}^{-1}, \quad i \neq q;$$

$$\tilde{c}_j = \begin{vmatrix} c_j & c_p \\ b_q & b_{qp} \end{vmatrix} b_{qp}^{-1}, \quad j \neq p,$$

с последующей заменой местами переменных x_p и z_q.

Пусть в результате последовательных итераций по симплекс-методу найдена оптимальная точка и получена симплекс-таблица типа табл. 7.2, но с коэффициентами, соответствующими оптимальному решению. Оптимальное решение находится из столбца свободных членов b_1, ..., b_n либо равно нулю, смотря по тому, зависим ли компонент или нет.

Далее проверяем найденную оптимальную точку на целочисленность. С этой целью просматриваем все b_i ($i=1$, ..., n). Если все b_i целые, то задача решена. Если же среди чисел b_i есть дробное, например b_l, то у оптимального решения имеется хотя бы одна нецелая координата. В этом случае проверяется наличие целых точек в допустимом многограннике. Если в какой-либо строке с дробным членом b_l и все остальные коэффициенты b_{li} окажутся целыми, то в допустимом многограннике нет целых точек, и задача целочисленного программирования не имеет решения. В противном случае составляем дополнительное ограничение. Для этого берем l-ю строку с дробным членом b_l и записываем дополнительное ограничение

$$u_1 = \tilde{b}_l - [\tilde{b}_l] + g_{l1}\tilde{x}_1 + \ldots + g_{lm}\tilde{x}_m > 0,$$

где $g_{li} = b_{li} - [b_{li}]$ ($i=1$, ..., m); $[x]$ — символ целой части числа x. Полупространство $\{u_l \geqslant 0\}$ может содержать каждую целую точку допустимого многогранника и не содержать ранее найденной оптимальной точки. Добавив (в виде строки) новое ограничение, заполним новую таблицу условий. Описанный процесс повторяется до тех пор, пока за конечное число шагов или находится оптимальное целочисленное решение, или же обнаруживается, что такого решения у задачи (7.7), (7.8) нет.

ПРОГРАММА GOMORY

Назначение: решение задачи целочисленного линейного программирования. Предназначена для определения (N—1)-мерного целочисленного вектора [X(1), ..., X(N—1)]^T, минимизирующего целевую функцию $F=$A(1,1)X(1) $+\ldots+$A(1,N—1) $*$ $*$X(N—1), при условии X(1) $\geqslant 0$, ..., X(N—1) $\geqslant 0$, A(I, 1)X(1) $+\ldots+$A(I,N—1) $\leqslant$ $\leqslant$A(I,N). Написана на языке ПЛ/1, прототипом программы послужила Алгол-процедура GOMORY [62].

Параметры:

N — размерность, на единицу большая размерности оптимизируемого вектора параметров;

M — число строк матрицы (симплекс-таблицы);

A — матрица (симплекс-таблица) размера N*N с атрибутами BINARY FIXED;

F — оптимальное значение целевой функции;

ERROR — индикатор, позволяющий судить об успешности решения задачи: ERROR = =0 при успешном выполнении поиска оптимума, ERROR=1 в противном случае.

Переменная F описывается с атрибутами BINARY FIXED, а ERROR — с атрибутами CHARACTER(1), атрибуты параметров N и M задаются по умолчанию. Первые ненулевые элементы столбцов матрицы A должны быть положительными,

часть входных значений A задается при постановке задачи, остальные присваиваются самой программой GOMORY.

Обращение: CALL GOMORY (A, M, N, ERROR, X, F);

Основные этапы работы:

1) присвоение недостающих начальных значений элементов матрицы A;

2) проверка условия неотрицательности всех, кроме самого верхнего, элементов последнего столбца матрицы A; в случае положительного ответа текущие значения вектора X и функции F объявляются оптимальными;

3) проверка наличия в матрице A строки, в которой все элементы, кроме, быть может, последнего, неотрицательны, что означает отсутствие решения задачи;

4) организация сечений по методу Гомори;

5) формирование выходных параметров программы.

Пример. Решение двух задач минимизации функций:

1) $F=x_1+x_2$ при наличии ограничений $-x_1-x_2\leqslant -1$, $2x_1-x_2\leqslant 1$;

2) $F=14x_1+25x_2$ при наличии ограничений $3x_1-5x_2\leqslant -11$, $7x_1+2x_2\leqslant -13$.

Для получения верного решения первой задачи $F=1$, $x_1=0$, $x_2=1$ потребовалось 0,3 с, второй задачи $F=142$, $x_1=3$, $x_2=4$—0,35 с на ЭВМ ЕС-1050.

```
GOMORY:  PROCEDURE(A,M,N,ERROR,X,F):
         DECLARE
         (A(*,*),X(*)) BINARY FIXED,
         (M,N,F,I,J,K,Q,R,C,T,S,LBD1,LBD2)
         BINARY FIXED,
         ERROR
         CHARACTER(1);
         A(1,N)=0;
/* 1 */
         DO I=M-N+3 TO M+1;
          DO J=1 TO N;
           IF I=J+M-N+2
           THEN A(I,J)=-1; ELSE a9I,J)=0;
          END;
         END;
/* 2 */
    MET1:
         DO I=2 TO M+1;
         IF A(I,N) < 0 THEN
          DO;
           R=I; GO TO MET2;
          END;
         END;
         GO TO FIN;
/* 3 */
    MET2:
         DO K=1 TO N-1;
          KIND=K;
          IF A(R,K) < 0 THEN
          GO TO MET4;
         END;
         ERROR='1'; GO TO FIN1;
/* 4 */
    MET4:
         Q=KIND; K=KIND;
         DO J=K+1 TO N-1;
          IF A(R,J) < 0 THEN
           DO;  I=1;
```

```
   MET3:  IF A(I,J) < A(I,Q) THEN Q=J;
          ELSE
          IF A(I,J) = A(I,Q) THEN
           DO;
            I=I+1; GO TO MET3:
           END;
          END;
         END;
        S=1;
   MET5:IF A(S,Q)=0 THEN
         DO;
          S=S+1; GO TO MET5;
         END;
       LBD1=-A(R,Q); LBD2=1;
       DO J=1 TO Q-1,
          Q+1 TO N-1;
        IF A(R,J) < 0 THEN
         DO;
          DO I=1 TO S;
          IF A(I,J)   = 0 THEN GO TO MET7;
          END;
          T=EUCLID(A(S,J),A(S,Q));
          IF (T*A(S,Q) = A(S,J)) & (T > 1)
          THEN DO;
   MET6:          I=S;
                  I=I+1;
                  IF T*A(I,Q) = A(I,J) THEN
                  GO TO MET6;
                  IF T*A(I,Q) > A(I,J) THEN
                  T=T-1;
                 END;
          IF  -A(R,J)*LBD2 > T*LBD1 THEN
          DO;
           LBD1=-A(R,J); LBD2=T;
          END;
    MET7:END;
         END;
        DO J=1 TO Q-1,
           Q+1 TO N;
         C=EUCLID(A(R,J)*LBD2,LBD1);
         IF C   = 0 THEN
         DO I=1 TO M+1;
          A(I,J)A(I,J)+C*A(I,Q);
         END;
        END;
        GO TO MET1;
    FIN: F=-A(1,N);
        DO I=1 TO N-1;
        X(I)=A(M-N+I+2,N);
        END;
        ERROR='0';
    FIN1:
    END  GOMORY;

EUCLID: PROCEDURE(U,V);
        DECLARE
        (U,V,W) BINARY FIXED;
        W=FLOOR(U/V);
MET6:   IF W*V > U THEN
        DO;
         W=W-1; GO TO MET8;
        END;
```

```
MET9:   IF (W+1)*V <= U THEN
        DO;
         W=W+1; GO TO MET9;
        END;
        RETURN(W);
END     EUCLID;

TEST:   PROCEDURE OPTIONS(MAIN);
        DCL (A(5,3),X(2),M,N,F) BINARY FIXED;
        DCL ERROR CHAR(1);
        M=4; N=3;
        A(1,1)=1;   A(1,2)=1;   A(1,3)=0;
        A(2,1)=-1;  A(2,2)=-1;  A(2,3)=-1;
        A(3,1)=2;   A(3,2)=-1;  A(3,3)=1;
        CALL GOMORY(A,M,N,ERROR,X,F);
        PUT SKIP DATA(ERROR,X,F);
        PUT SKIP DATA(A);
END:    TEST;
```

7.3. Оптимизация дискретных параметров методом вектора спада

Для решения разнообразных задач целочисленного программирования (линейного, комбинаторного, булевого), а также частично целочисленного широко применяется метод вектора спада [69]. Метод вектора спада представляет собой метод локальной оптимизации функции F, заданной на целочисленной решетке $\mathbf{Z}^m$. В основу метода положен принцип перебора элементов подходящим образом определенной окрестности данного начального решения, чтобы выявить наилучшее, которое, в свою очередь, становится центром новой окрестности локальной оптимизации. Весь процесс многократно повторяется до тех пор, пока станет невозможным улучшить решение. Естественно, что для использования этого подхода необходимо иметь метрику на $\mathbf{Z}^m$, которая позволяет определить окрестность данной точки, а также эффективную процедуру перебора всех элементов этой окрестности. Одним из преимуществ этого метода является его способность эффективно решать задачи целочисленного программирования (7.1)—(7.3) с нелинейными функциями F.

Пусть $\mathbf{Z}^m$ — метрическое пространство, т. е. на множестве задана метрика (функция расстояния) $g(\mathbf{x}, \mathbf{y})$, например

$$g(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|,$$

$$\mathbf{x} = \{x_i\}, \quad \mathbf{y} = \{y_i\}, \quad i = 1, \ldots, m,$$

ее называют *метрикой Хэмминга* и обозначают $\mathbf{H}(\mathbf{x}, \mathbf{y})$.

Множество $U_g(\mathbf{x},r) = \{z : g(\mathbf{x},\mathbf{z}) \leqslant r\}$ назовем *окрестностью* с центром в точке $\mathbf{x} \in \mathbf{Z}^m$ и радиусом $r \geqslant 0$. В случае задания на $\mathbf{Z}^m$ метрики Хэмминга $\mathbf{H}(\mathbf{x},\mathbf{y})$ соответствующая окрестность также называется окрестностью Хэмминга.

Ограничимся рассмотрением окрестностей с целочисленными радиусами r. Отметим ряд очевидных свойств окрестности $\mathbf{U}_g(\mathbf{x},r)$:

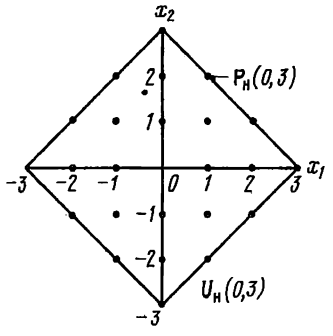


Рис. 7.1. Двумерная окрестность Хэмминга

$$\mathbf{U}_g(\mathbf{x},r) = \bigcup_{j=1}^{r} \{\mathbf{z} : g(\mathbf{x},\mathbf{z}) = j\}, \qquad (7.9)$$

$$\mathbf{U}_g(\mathbf{x}, r) = \mathbf{U}_g(\mathbf{x}, r-1) \cup \{\mathbf{z}: g(\mathbf{x}, z) = r\}. \qquad (7.10)$$

В дальнейшем множества $\mathbf{P}_g(\mathbf{x},j)=\{\mathbf{z}:g(\mathbf{x},\mathbf{z})=j\}$ будем называть *поверхностями* радиуса j.

В качестве примера на рис. 7.1 изображены точки двумерной окрестности Хэмминга $\mathbf{U}_H(0,3)$, непрерывной линией соединены точки, входящие в поверхность $\mathbf{P}_H(0,3)$.

Точку $\mathbf{x}\in\mathbf{Z}^m$ назовем *точкой локального минимума функции* $F(\mathbf{x})$ относительно окрестности $\mathbf{U}_g(\mathbf{x},r)$, если для всех точек $\mathbf{y}\in\mathbf{U}_g(\mathbf{x},r)$ выполняется условие $F(\mathbf{x})\leqslant F(\mathbf{y})$.

Вектором спада функции $F(\mathbf{x})$ относительно окрестности $\mathbf{U}_g(\mathbf{x},r)$ радиуса r назовем многомерную функцию $\Delta^r(\mathbf{x})$, определенную на $\mathbf{Z}^m$, если выполняются следующие условия:

1) для каждой точки $\mathbf{x}\in\mathbf{Z}^m$ значения функции $\Delta^r(\mathbf{x})$ являются $n(\mathbf{x},r)$-мерным действительным вектором с компонентами $\Delta_1(\mathbf{x})$, ..., $\Delta_{n(\mathbf{x},r)}(\mathbf{x})$;

2) точка $\mathbf{x}$ является точкой минимума F относительно окрестности $\mathbf{U}_g(\mathbf{x},r)$ радиуса r тогда, и только тогда, когда $\Delta_i(\mathbf{x})\geqslant 0$ при $i=1$, ..., $n(\mathbf{x},r)$.

Примером вектора спада может служить функция $\Delta^r(\mathbf{x})=F(\mathbf{y})-F(\mathbf{x})$ ($\mathbf{x}\in\mathbf{Z}^m$, $y\in\mathbf{U}_g(\mathbf{x},r)$, которая удовлетворяет условиям 1 и 2.

Приведем схему эффективного порождения всех точек окрестности Хэмминга $\mathbf{U}_H(\mathbf{y},M)$ ($\mathbf{y}\in\mathbf{Z}^m$, $M\geqslant 0$). Отметим, что для достижения этой цели достаточно (в силу (7.10)), во-первых, уметь находить все точки $\mathbf{Z}\in\mathbf{Z}^m$, расстояние $\mathbf{H}(\mathbf{y},\mathbf{z})$ от которых в точности равно j для всех натуральных j, меньших M. Во-вторых, достаточно порождать все точки окрестности $\mathbf{U}_H(0,M)$, так как любую точку окрестности $\mathbf{U}_H(\mathbf{y},M)$ можно линейным переносом преобразовать в единственную точку $\mathbf{U}_H(0,M)$ и наоборот. В-третьих, можно ограничиться перебором точек окрестности $\mathbf{U}_H(0,M)$, лежащих в первом октанте $\mathbf{R}_1=\{x_1, ..., x_m: x_1\geqslant 0, ..., x_m\geqslant 0\}$, так как все остальные ее элементы можно получить, вращая каждую точку $\mathbf{y}\in\mathbf{U}_H(0,M)\cap\mathbf{R}_1$ относительно начала координат или, что эквивалентно, последовательно отображая элементы $\mathbf{U}_H(0,M)\cap\mathbf{R}_1$ относительно координатных гиперплоскостей. Следовательно, достаточно уметь порождать все точки $\mathbf{x}$, лежащие в первом октанте $\mathbf{R}_1$, целочисленные и отстоящие от начала координат на расстоянии $\mathbf{H}(0,\mathbf{x})=j(j=1, ..., M)$. Эта задача есть по своей сути задача решения в неотрицательных целых числах совокупности уравнений вида

$$x_1 + ... + x_m = j,\ j = 1, ..., M. \qquad (7.11)$$

Для решения уравнений (7.11) построим специальную модель. Представим себе j ячеек, между которыми надо проставить $m-1$ перегородку. Тогда каждому размещению перегородок между ячейками соответствует решение $(x_1, ..., x_m)$ уравнения (7.11), где x_1 — число ячеек, предшествующих первой перегородке; $x_l(l=2,..., m-1)$ — число ячеек между l-й и $(l+1)$-й перегородками; x_m — число ячеек, следую-

щих за $(m-1)$-й перегородкой. Верно и обратное: каждому решению $(x_1, ..., x_m)$ уравнения (7.11) можно поставить в соответствие (описанным выше способом) единственное распределение перегородок в модели.

Теперь можно определить количество неотрицательных целых решений уравнения (7.11):

$$K(m,M) = C^{M}_{M+m-1} = C^{m-1}_{M+m-1} \tag{7.12}$$

(здесь C^n_l — число сочетаний из l элементов по n: $C^n_l = l!/n!(l-n)!$).

Процедура порождения всех решений уравнений (7.11) состоит из следующих операций: вводится совокупность $m-1$ индексов $j_1, ..., j_{m-1}$, принимающих значения из совокупности $\{1, 2, ..., j+m-1\}$ и удовлетворяющих условию $j_1 < j_2 < ... < j_{m-1}$. Передвигая эти индексы в совокупности возможных значений с сохранением их относительного порядка, последовательно получаем все решения уравнения (7.11). Каждое из решений преобразуется в точки окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{y},M)$ с помощью вращения в пространстве $\mathbf{R}^m$ относительно начала координат и линейным переносом.

Используя свойства окрестностей (7.9), (7.10), можно составить соотношение для расчета $\bar{K}(m,M)$ — количества элементов окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{y},M)$ $(\mathbf{y} \in \mathbf{Z}^m,\ M \geqslant 0)$:

$$\bar{K}(m,M) = \begin{cases} 2^m \sum\limits_{i=0}^{M-2} K(m,i) + 2^{m-1} m \sum\limits_{i=0}^{M} K(m-1,i) - \\ - 2^{m-2}(m-1) \sum\limits_{i=0}^{M} K(m-2,i) + 2^{m-3}(m- \\ - 2) \sum\limits_{i=0}^{M} K(m-3,i) + ... + (-1)^{m+1} \sum\limits_{i=1}^{M} K(0,i),\ M \geqslant 2, \\ 2m+1,\ M=1, \\ 1,\ M=0 \end{cases} \tag{7.13}$$

Перебор элементов окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{y},M)$ производится путем последовательного порождения элементов поверхностей Хэмминга $\mathbf{P}_{\mathbf{H}}(\mathbf{y},j) = \{\mathbf{z}: \mathbf{H}(\mathbf{y},\mathbf{z}) = j\}$ $(j = 1, ..., M)$.

Метод вектора спада основывается на вычислении координат вектора спада $\Delta^r(\mathbf{x})$, которые позволяют определить направление, по которому значения функции $F(\mathbf{x})$ в окрестности $U_g(\mathbf{x},r)$ для каждой точки $\mathbf{x} \in \mathbf{Z}^m$ уменьшаются. Различные формы вектора спада приведены в [69]. Следует отметить, что часто координаты вектора спада вычисляются проще, чем значения функции $F(\cdot)$, а для локальной оптимизации можно ограничиться лишь частью из всех координат вектора $\Delta^r(\mathbf{x})$. Тем не менее одной из основных и наиболее часто встречающихся при решении практических задач форм вектора спада является следующая:

$$\Delta^r(\mathbf{x}) = F(\mathbf{y}) - F(\mathbf{x}),\ \mathbf{x} \in \mathbf{Z}^m,\ y \in \mathbf{U}_g(\mathbf{x},\mathbf{r}). \tag{7.14}$$

Именно эту форму вектора спада будем использовать для целочисленной оптимизации. Будем также считать, что на $\mathbf{Z}^m$ задана метрика Хэмминга $\mathbf{H}(\mathbf{x},\mathbf{y}) = \sum\limits_{i=1}^{M} |x_i - y_i|$. Заметим, что при решении задач целочисленной оптимизации методом

вектора спада требуется, чтобы начальная точка поиска (начальное приближение) была допустимой, т. е. принадлежала множеству $\mathbf{D}$ (о поиске таких допустимых начальных приближений см. ниже, § 7.4).

Таким образом, алгоритм метода вектора спада состоит из следующих основных этапов:

1. Задаем некоторое начальное приближение $\mathbf{x}_0 \in \mathbf{D}$, а также $r_{\min}$ и $r_{\max}$ — минимальный и максимальный радиусы окрестности Хэмминга локальной минимизации.

2. Полагаем $n=0$, $r=r_{\min}$.

3. На $(n+1)$-м шаге выполняем следующие действия:

3.1. Производим перебор всех у-элементов в окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r)$.

3.2. Находим $\mathbf{y} \in \mathbf{A} = \mathbf{D} \cap \mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r) / \{\mathbf{x}_{\mathbf{H}}\}$, такое, что $F(\mathbf{y}) = \min\limits_{\mathbf{y} \in \mathbf{A}} F(\mathbf{y})$.

3.3. Если множество $\mathbf{A}$ пусто или если $F(\tilde{\mathbf{Y}}) - F(\mathbf{x}_n) \leqslant 0$ (т. е. если данный компонент вектора спада отрицательный), переходим к п. 4, в противном случае, если $r < r_{\max}$, значение радиуса r заменяется на $r+1$ и переходим к п. 3.1, в противном случае к п.5.

4. Заменяем n на $n+1$, задаем $\mathbf{x}_n = \mathbf{y}$, $r = r_{\min}$ и переходим к п.3.

5. Значение $\mathbf{x}_n$ объявляем локальным (по отношению к окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r_{\max})$) минимумом функций F.

При переходе к перебору элементов (п.3) окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r)$ с $r > r_{\min}$ достаточно рассматривать точки, принадлежащие множеству $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r) / \mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r-1)$, т. е. поверхности Хэмминга $\mathbf{P}_{\mathbf{H}}(\mathbf{x}_n, r)$.

Алгоритм, реализующий метод вектора спада, сходится к точному решению задачи целочисленного программирования при достаточно широких предположениях о структуре оптимизируемой целевой функции. В частности, сходимость алгоритма (при соответствующем выборе радиуса $r_{\max}$) имеет место при конечном множестве определения целевых функций.

Если решается оптимизационная задача с функцией, не удовлетворяющей полученным достаточным условиям, в алгоритм вводятся дополнительные критерии окончания поиска. Таким критерием может быть количество шагов поиска или продолжительность его по времени, а также оценка $\min\limits_{\mathbf{x} \in \mathbf{D}} F(\mathbf{x})$, полученная с помощью алгоритма статистической оценки экстремальных значений (см. § 7.6).

Результаты машинного эксперимента [69] по решению задач целочисленного программирования показали высокую эффективность применения метода вектора спада, в частности в среднем затрачивается меньше операций, чем при применении циклического алгоритма Гомори [72]. К несомненным достоинствам метода спада можно отнести также достаточную простоту реализации на ЭВМ. Эти обстоятельства и предопределили широкое распространение этого метода для решения задач синтеза систем управления.

ПРОГРАММА VEK

Назначение: решение задач нелинейного выпуклого целочисленного программирования с помощью метода вектора спада.

Параметры:

X0 — вектор начального положения точки поиска размерности X, описывается с помощью атрибутов FIXED BINARV;

F0 — значение функции цели в точке X0;

N — размерность вектора оптимизируемых переменных;

MAXR — максимальное значение радиуса окрестности Хэмминга, в пределах которой производится локальная оптимизация текущей точки траектории поиска;

FUNCT — вычисление целевой функции цели в данной точке, процедура, вызываемая оператором CALL FUNCT (X1, F1), X1 — вектор оптимизируемых параметров; F1 — возвращаемое значение функции цели в точке X1;

OGR — определение допустимости данной точки X1; заголовок процедуры OGR: PROCEDURE (X1, KIN), где KIN=0 для допустимого вектора X1 и KIN=1 для недопустимого;

X0, F0 — оптимальный вектор аргументов и оптимальное значение функции цели соответственно;

MAXN — фактическое количество шагов, затраченных на поиск оптимального решения.

Атрибуты параметров F0, N, MAXN, MAXR устанавливаются по умолчанию.

Обращение: CALL VEK (X0, F0, FUNCT, OGR, N, MAXR, MAX);

Основные этапы работы:

1) перебор элементов окрестности Хэмминга данной точки;

2) проверка, исчерпан ли заданный лимит шагов;

3) проверка допустимости данного решения;

4) проверка изменения (улучшения) целевой функции при данном допустимом решении.

Основное значение в работе программы имеет реализация процедуры перебора элементов произвольной окрестности Хэмминга с данным центром. Для этого используется процедура генерации разбиения произвольного положительного целого числа на произвольное число частей GENER, а также процедура всех последовательных перестановок этих разбиений PERM. Эти процедуры являются модифицированными вариантами соответствующих процедур, разработанных на языке Алгол [62].

Пример. Целочисленная минимизация функции $F(x_1,x_2)=x_1^2+x_2^2$ при наличии ограничения $x_1^2+x_2^2>1$.

Исходные данные: X0(1)=4, X0(2)=2; N=2, MAXR=3, MAXN=200.

Точное значение допустимого минимума X0(1)=X0(2)=1, F0=2 получено на ЭВМ ЕС-1050 за 0,5 с.

```
   VEK:PROCEDURE(X0,F0,N,MAXR,MAXN);
     DCL X0(*) FIXED BIN;
     N0=0;
     BEGIN;
     DCL NX(N);
     DCL X(N) FIXED BIN;
MET:DO I=1 TO MAXR;
      FD=1; Z=1; NW=N;
MET1:CALL GENER(I,N,NW,Z,X,FD);
      PRI=1;
      NN=N;
```

```
MET2:CALL PERM(X,NN,PRI);
     BEGIN;
     DCL X1(N) FIXED BIN;
     DO I1=2*(N+1)-2 TO 0 BY -1;
     R1=I1;
      DO J1=1 TO N;
      R1=R1/2;
      I11=FLOOR(R1);
      IF R1*2=I11*2 THEN X1(J1)=X(J1);
      ELSE X1(J1)=-X(J1);
      R1=I11;
      END;
      X1=X1+X0;
      N0=N0+1;
      IF N0 > MAXN THEN GO TO MET3;
      CALL OGR(X1,KIN);
      IF KIN=0 THEN
      DO;
      CALL FUNCT(X1,F1);
      IF F1 < F0 THEN
      DO;
      X0=X1;
      F0=F1;
      END;
      END;
      END;
     END;
     IF PRI=0 THEN GO TO MET2;
     IF FD=0 THEN GO TO MET1;
    END;
    END;
MET3:MAXN=N0;
     END VEK;

PERM:PROCEDURE(X,N,PRI);
     DCL X(*) BIN FIXED;
     DCL Q BIN FIXED;
     DCL (P,D) (2:100) BIN FIXED INTERNAL;
     IF PRI=1 THEN DO; PRI=0;
     DO K=2 TO N; P(K)=0; D(K)=1; END; END; K=0;
INDEX:P(N)=P(N)+D(N); Q=P(N);
     IF Q=N THEN DO; D(N)=-1;
     GO TO ITER; END;
     IF Q =0 THEN GO TO TRANS;
     D(N)=1; K=K+1;
ITER:IF N>2 THEN DO;
     N=N-1; GO TO INDEX; END;
     Q=1; PRI=1;
TRANS:Q=Q+K; T=X(Q); X(Q)=X(Q+1);
      X(Q+1)=T;
END PERM;

GENER:PROCEDURE(N,K,H,Z,P,F);
      DCL P(*) BIN FIXED;
      DCL (A,Q,B,H) BIN FIXED;
      IF F=0 THEN DO;
      A=P(1)-P(2)-2; J=2;
 TES: IF P(1)-P(J)>=2 THEN GO TO DIV;
      A=A-1+J*(P(J)-P(J+1)); J=J+1;
      GO TO TES; END;
      IF Z=1 THEN
```

```
        A=N; ELSE A=N-K;
        IF Z=1 THEN P(K)=-1;
        ELSE P(K)=0;
        F=0; J=K;
  DIV:  B=H-1-P(J);  Q=FLOOR(A/B);
        DO I=1 TO Q; P(I)=H; END;
        IF Q=K THEN GO TO FIN;
        DO I=Q+1 TO J; P(I)=1+P(J); END;
        P(Q+1)=A-B*Q+P(Q+1);
        IF P(1)-P(K) < 2 THEN
  FIN:  F=1; RETURN;
        END GENER;


  OGR:  PROCEDURE(X1,KIN);
        DCL X1(*) FIXED BIN;
        IF X1(1)**2+X1(2)** <= 1 THEN KIN=1; ELSE KIN=0;
        END OGR;


FUNCT:  PROCEDURE(X1,F1);
        DCL X1(*) FIXED BIN;
        F1=X1(1)**2+X1(2)**2;
        END FUNCT;


 TEST:  PROCEDURE OPTIONS(MAIN);
        DCL X0(2)FIXED BIN;
        X0(1)=4; X0(2)=2; F0=20;
        N=2;
        MAXR=3; MAXN=200;
        CALL VEK(X0,F0,N,MAXR,MAXN);
    END TEST;
```

7.4. Оптимизация дискретных параметров методом направляющих окрестностей

Для приближенного решения задачи целочисленного программирования (7.1)—(7.3) с помощью локальной оптимизации исходной начальной точки предназначен метод направляющих окрестностей. В основу метода положены идеи метода возможных направлений Г. Зойтендейка [75], разработанного для решения задач выпуклого программирования. Трансформация метода на дискретные системы достигается с помощью введения характеристик убывания целевой функции в заданном направлении и выбора из всех возможных подходящего направления для оптимизации. В методе направляющих существенно используются основные идеи и понятия метода вектора спада (см. § 7.3).

Пусть множество допустимых решений $\mathbf{D}$ ($\mathbf{D}\subseteq\mathbf{Z}^m$) является *пересечением* $\mathbf{A}$ — некоторого выпуклого подмножества $\mathbf{Z}^m\subset\mathbf{R}^m$, а F — выпуклая функция на $\mathbf{A}$. (Множество называется выпуклым, если для любых точек $\mathbf{x}$, $\mathbf{y}\in\mathbf{B}$ и произвольных чисел α, $\beta>0$, таких, что $\alpha+\beta=1$, $\mathbf{z}=\alpha\mathbf{x}+\beta\mathbf{y}\in\mathbf{B}$. Функция $f(\mathbf{x})$ ($\mathbf{x}\in\mathbf{B}$) называется выпуклой, если для любых точек $\mathbf{x}$, $\mathbf{y}\in\mathbf{B}$ и произвольных чисел α, $\beta>0$, таких, что $\alpha+\beta=1$, имеем $f(\mathbf{z})\leqslant\alpha f(\mathbf{x})+\beta f(\mathbf{y})$.)

Пусть $\mathbf{x}_0$ — произвольная точка $\mathbf{D}$, $\mathbf{U}_g(\mathbf{x}_0,r)$ — некоторая (соответствующая метрике $g(\mathbf{x},\mathbf{y})$ на $\mathbf{z}^m$) окрестность радиуса r с центром в точке $\mathbf{x}_0$. Если $\mathbf{x}\in\mathbf{D}$ и $F(\mathbf{x})<F(\mathbf{x}_0)$, то элемент $\mathbf{x}\in\mathbf{U}_g(\mathbf{x}_0,r)$ будем считать подходящим. Векторную функцию

назовем Δ-вектором функции F в окрестности $\mathbf{U}_g(\mathbf{x}_0, r)$ произвольной точки $\mathbf{x}_0 \in \mathbf{Z}^m$, если она имеет компоненты

$$\Delta_i(\mathbf{x}_0) = F(\mathbf{x}_i) - F(\mathbf{x}_0),$$

$$\mathbf{x}_i \in \mathbf{U}_g(\mathbf{x}_0, r), \quad i = 1, \ldots, n(\mathbf{x}_0, r) \tag{7.15}$$

(здесь $n(\mathbf{x}_0, r)$ — число элементов окрестности $\mathbf{U}_g(\mathbf{x}_0, r)$). Очевидно, что Δ-вектор функции F есть один из вариантов вектора спада. Отрицательные компоненты Δ-вектора позволяют выделить перспективные направления дальнейшего поиска, среди которых и выбирают подходящие для реализации вдоль них процедуры дальнейшего поиска. С другой стороны, неотрицательность всех компонентов Δ-вектора свидетельствует о том, что $\mathbf{x}_0$ — точка локального минимума (в пределах окрестности $\mathbf{U}_g(\mathbf{x}_0, r)$).

В дальнейшем будем считать, что на $\mathbf{Z}^m$ задана метрика Хэмминга $\mathbf{H}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|$. При другой метрике на $\mathbf{Z}^m$ алгоритм аналогичен.

Алгоритм метода направляющих окрестностей состоит из следующих основных этапов:

1. Задаем начальное приближение $\mathbf{x}_0 \in \mathbf{D}$ и максимальный радиус перебора $r_{\max} > 0$. Полагаем $n = 0$.

2. Полагаем $r = 1$.

3. На $(n+1)$-м шаге производим перебор всех $\mathbf{y}$-элементов окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r)$. На каждом шаге процедуры перебора определяем, является ли текущий сгенерированный на данном шаге элемент $\mathbf{y}$ подходящим. Если подходящих элементов в $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r)$ нет, переходим к п.4, если есть, производим следующие действия:

3.1. Определяем размер шага поиска λ_n из точки $\mathbf{x}_n$ в соответствии со следующей формулой:

$$\lambda_n = \max\{\lambda : \mathbf{x}_n + \lambda \mathbf{s}_n \in \mathbf{D}, \ F(\mathbf{x}_n + \mu \mathbf{s}_n) \geqslant F(\mathbf{x}_n + \lambda \mathbf{s}_n)\},$$

где $\mathbf{s}_n = \mathbf{y} - \mathbf{x}_n$, $0 \leqslant \mu \leqslant \lambda$.

3.2. Если $\lambda_n = \infty$, то делается вывод, что решение задачи (7.1) — (7.3) не ограничено снизу, и алгоритм заканчивает свою работу.

3.3. Если $\lambda_n = 0$, то производится возвращение к п. 3.1, для продолжения процедуры последовательного перебора окрестности $\mathbf{U}_{\mathbf{H}}(\mathbf{x}, r)$.

3.4. Если $0 < \lambda_n < \infty$, то вычисляем $\mathbf{x}_{n+1}$ по формуле $\mathbf{x}_{n+1} = \mathbf{x}_n + \lambda_n \mathbf{s}_n$. Заменяем n на $n+1$ и переходим к п.2.

4. Если $r < r_{\max}$, то заменяем r на $r+1$ и переходим к п.3. Если $r = r_{\max}$, то $\mathbf{x}_n$ объявляется приближенным решением задачи (7.1) — (7.3).

В данном алгоритме (как и в алгоритме метода вектора спада) при переходе к процедуре перебора элементов окрестности Хэмминга $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r)$ с увеличенным на единицу радиусом достаточно рассматривать точки, принадлежащие поверхности Хэмминга $\mathbf{P}_{\mathbf{H}}(\mathbf{x}_n, r) = \mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r) / \mathbf{U}_{\mathbf{H}}(\mathbf{x}_n, r-1)$. Достаточные условия сходимости алгоритма метода направляющих окрестностей при решении задачи выпуклого целочис-

ленного программирования сформулированы в [69]. Там же рассмотрены примеры его применения для решения задач линейного целочисленного программирования, частично целочисленного программирования, а также для поиска локальных решений параметрических задач линейного целочисленного программирования (т. е. задач, целевая функция и (или) коэффициенты системы ограничений которых зависят от некоторого параметра). В качестве критерия окончания поиска (или качества решения) по алгоритму метода направляющих окрестностей аналогично алгоритму метода вектора спада может служить оценка величины $\min_{x \in D} F(x)$, полученная с помощью алгоритма статистической оценки экстремальных значений функции (см. § 7.6).

При описании алгоритма метода направляющих окрестностей осталась нерешенной задача получения хотя бы одного допустимого решения, т. е. элемента множества **D**. В общем случае для нахождения допустимых решений можно воспользоваться алгоритмами безусловной минимизации, предназначенными для решения задачи поиска такого y_{opt}, что

$$T(y_{opt}) = \min_{y \in Z^m} T(y),$$

где $T(y) = \sum_{i=1}^{l} u_i(y) F_i^2(y)$; $u_i(y)$ — оператор Хэвисайда; $u_i = 0$ при $F_i(y) \leqslant 0$ и $u_i = 1$ при $F_i(y) > 0$; F_i $(i = 1, \ldots, l)$ — функции, формирующие систему ограничений (7.3). Функционал $T(y)$ характеризует степень нарушения данной точкой **y** системы ограничений. Легко видеть, что $T(y) \geqslant 0$ для всех $y \in Z^m$ и $T(y) = 0$ при $y \in D$. В данной постановке задача поиска допустимого решения трансформируется в задачу целочисленного программирования без ограничений, для решения которой можно применить методы направляющих окрестностей и вектора спада.

Алгоритм направляющих окрестностей реализован в виде программы NAPRO (на языке ПЛ/1), которая в целом аналогична программе VEK, но в отличие от последней содержит дополнительный фрагмент, размещаемый после строки 31 программы VEK:

```
DO    WHILE   (N)<=MAXN);
      X1=X1+NX;
      N0=N0+1;
      CALL  OGR(X1,KIN);
      IF  KIN = 0 THEN
      DO;
      CALL  FUNCT(X1,F1);
      IF  F1 < F0  THEN
      DO;
      X0=X1; F0=F1;
      END;
      ELSE  GO TO  MET;
      END;
      ELSE  GO TO  MET;
END;
```

7.5. Оптимизация дискретных параметров имитационных моделей

Дискретный характер управления функционированием современных систем управления, обусловленный иерархической структурой и формированием управления в виде выбора альтернативных вариантов из конечного числа заданных, приводит к следующей постановке задачи оптимизации.

Предположим, что в результате отдельной реализации имитационного эксперимента вычисляется некоторая скалярная функция (критерий) $f(\mathbf{x})$, зависящая от m-мерного параметра $\mathbf{x}$, принадлежащего некоторому дискретному множеству $\mathbf{x} \in \mathbf{R}^m$ ($\mathbf{R}^m$—m-мерное евклидово пространство). Так как отдельные реализации эксперимента определяются с помощью методов моделирования случайных процессов, то зависимость $f(\mathbf{x})$ имеет неоднозначный, вероятностный характер. Все это приводит к необходимости постановки задачи оптимизации имитационной модели в рамках теории стохастического программирования.

В качестве критерия обычно выбирается функция математического ожидания случайной величины $f(\mathbf{x}): F(\mathbf{x}) = Mf(\mathbf{x})$. В этом случае $f(\mathbf{x})$ ($\mathbf{x} \in \mathbf{X}$) — совокупность случайных величин, заданных на некотором вероятностном пространстве $(\Omega, \mathbf{A}, \mathbf{P})$, где Ω — достоверное событие; $\mathbf{A}$ — σ-алгебра измеримых подмножеств Ω (событий); $\mathbf{P}$ — вероятностная мера на $(\Omega, \mathbf{A})$. На основании наблюдения отдельных реализаций $f(\mathbf{x})$ требуется найти экстремальные точки функционала $F(\mathbf{x})$.

Пусть множество оптимизируемых параметров дискретно и конечно. Без ограничения общности можно считать, что $\mathbf{x}$ имеет вид декартова произведения

$$\{n_1, \ldots, N_1\}^{\mathrm{T}} \oplus \ldots \oplus \{n_m, \ldots, N_m\}^{\mathrm{T}},$$

где n_i, $N_i \geqslant 0$ — целочисленные нижняя и верхняя границы изменения i-го параметра ($i = 1, \ldots, m$); m — количество оптимизируемых параметров. На $\mathbf{X}$ считаем заданными некоторый детерминированный функционал качества (критерий) и некоторое множество $\mathbf{D}$ ($\mathbf{D} \subseteq \mathbf{X}$). Критерий $F(\mathbf{x})$ вычисляется, а принадлежность множеству идентифицируется в ходе имитационного эксперимента. Ставится задача найти

$$\operatorname{Arg} \min_{\mathbf{x} \in \mathbf{D}} F(\mathbf{x}). \tag{7.16}$$

Таким образом, исходная задача формализована в терминах целочисленного программирования. Множество допустимых значений аргумента $\mathbf{D}$ задается, как правило, с помощью совокупности ограничений вида

$$F_1(\mathbf{x}) \leqslant 0, \ldots, F_l(\mathbf{x}) \leqslant 0, \quad l \geqslant 0, \mathbf{x} \in \mathbf{X}. \tag{7.17}$$

Решение задачи оптимизации дискретных параметров имитационных моделей затрудняют следующие факторы:

1. Большая мощность (количество элементов) множества $\mathbf{X}$, что не позволяет реализовать процедуру его полного перебора, особенно учитывая ограниченный объем имитационного моделирования.

2. Определенный порядок просмотра переменных при расчете критерия и ограничений, поскольку исходные данные для некоторых подсистем из числа комплектую-

щих всю оптимизируемую систему являются выходными характеристиками других подсистем.

3. Нерегулярность задачи, в данном контексте это означает, что у функционалов критерия и ограничений отсутствует хорошо выраженная структурность (типа выпуклости, линейности и т. д.).

4. Незначительная априорная информация о структуре функционалов критерия и ограничений.

Эти факторы затрудняют применение традиционных методов целочисленного программирования, ориентированных на решение задач с функционалами, принадлежащими строго заданному классу (линейных функционалов, выпуклых и т. д.). С другой стороны, при решении большинства практических задач синтеза систем требование достижения точного экстремума целевой функции не только невыполнимо, но и излишне. Так как процессу реального функционирования исследуемой системы сопутствует большое количество неформализуемых факторов, не учитываемых в моделировании, то в реальных разработках предпочтительнее иметь вместо одного точного оптимального решения совокупность субоптимальных решений, из которых ответственное лицо (или группа таких лиц) и выберет окончательное на основе интуиции, опыта аналогичных разработок и т. д. Таким образом, с практической точки зрения необходимо получение устойчивых (хотя и приближенных) результатов приближенной оптимизации в широком диапазоне решаемых задач.

Решать общую задачу и при наличии перечисленных выше факторов 1—4 позволяет алгоритм оптимизации дискретных оптимизационных моделей (АОДИМ).

Приведем некоторые сведения, относящиеся к заданию подходящей топологии на пространстве $\mathbf{X}$. Введем на $\mathbf{X}$ метрику $\mathbf{H}$:

$$\mathbf{H}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|, \quad \mathbf{x} = \{x_i\}^{\mathrm{T}}, \quad \mathbf{y} = \{y_i\}^{\mathrm{T}}, \quad i = 1, \ldots, m. \tag{7.18}$$

Через $\mathbf{U}_{\mathbf{H}}(\mathbf{y},M)$ обозначим соответствующую метрике $\mathbf{H}$ окрестность точки $\mathbf{y}$ радиуса M:

$$\mathbf{U}_{\mathbf{H}}(\mathbf{y},M) = \{\mathbf{z} : \mathbf{H}(\mathbf{y},\mathbf{z}) \leqslant M\}. \tag{7.19}$$

Отметим очевидные свойства введенных окрестностей $\mathbf{U}_{\mathbf{H}}$:

$$\mathbf{U}_{\mathbf{H}}(\mathbf{y},M) = \bigcup_{j=1}^{M} \{\mathbf{z} : \mathbf{H}(\mathbf{y},\mathbf{z}) = j\}, \tag{7.20}$$

$$\mathbf{U}_{\mathbf{H}}(\mathbf{y}, M) = \mathbf{U}_{\mathbf{H}}(\mathbf{y}, M-1) \cup \{\mathbf{z} : \mathbf{H}(\mathbf{y},\mathbf{z}) = M\}.$$

Поисковые операции в процессе работы АОДИМ базируются на процедуре эффективного порождения всех точек окрестности $\mathbf{U}_{\mathbf{H}}(\mathbf{y},M)$, основанной на многократном решении диофантовых уравнений вида

$$x_1 + \ldots + x_m = j, \quad j = 1, \ldots, M, \tag{7.21}$$

с последующим вращением каждого решения в пространстве $\mathbf{R}^m$ вокруг точки $\mathbf{y}$.

Пусть окрестность $\mathbf{U}_{\mathbf{H}}(\mathbf{y},M)$ содержит $K(m,M)$ элементов, удовлетворяющих рекуррентному соотношению (7.13).

Заменим исходную задачу оптимизации (7.16) другой, в которой поиск минимума функционала F осуществляется на $\mathbf{D}'$, задаваемом системой равенств:

$$\begin{aligned} F_1(\mathbf{y}) + G_1(\mathbf{y}) &= 0, \\ \cdots\cdots\cdots\cdots & \\ F_l(\mathbf{y}) + G_l(\mathbf{y}) &= 0, \end{aligned} \tag{7.22}$$

где функции $G_1(\mathbf{y})$, ..., $G_l(\mathbf{y})$ все неотрицательные. Так как, с одной стороны, на множестве $\mathbf{D}$ функции $G_i(\mathbf{y}) = -F_i(\mathbf{y})$ неотрицательные, а с другой — каждый элемент $\mathbf{y}\in\mathbf{D}'$ (т.е. для которого все $G_i(\mathbf{y})\geqslant 0$ $(i=1, \ldots, l)$) принадлежат также и $\mathbf{D}$, то $\mathbf{D}=\mathbf{D}'$, и исходная задача эквивалентна задаче, задаваемой соотношениями (7.16), (7.22).

Обозначим через $u(\mathbf{y})$ характеристику недопустимости данного решения $\mathbf{y}$:

$$u(\mathbf{y}) = \sum_{i=1}^{l} A_i \chi_i G_i(\mathbf{y}), \tag{7.23}$$

где $\{A_i,\ i=1, \ldots, l\}$ — набор отрицательных весовых коэффициентов, χ_i — оператор Хэвисайда функции $G_i(\mathbf{y})$: $\chi_i=1$, если $G_i(\mathbf{y})<0$, и $\chi_i=0$, если $G_i(\mathbf{y})\geqslant 0$.

Пусть дана исходная точка (начальное приближение) поиска $\mathbf{x}_0$. Без ограничения общности можно считать, что $\mathbf{x}_0$ совпадает с началом координат. Вычисляется совокупность величин

$$S_{i,r} = \frac{1}{M_{i,r}} \sum_{j=1}^{M_{i,r}} F\left[\mathbf{x}_{ij} - dF(\mathbf{x}_0)(\mathbf{x}_{ij} - \mathbf{x}_0) / \mathbf{H}(\mathbf{x}_0, \mathbf{x}_{ij}) + \mathbf{x}\right], \tag{7.24}$$

$$i=1, \ldots, 2^m,\ d\geqslant 0,$$

где суммирование ведется по всем элементам множества $\mathbf{A}_{i,r}$ — находящейся в i-м октанте $\mathbf{R}^m$ части окрестности $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_0,r)$ $(r=1, \ldots, R)$, где R задается априори. Количество элементов $M_{i,r}$ множества $\mathbf{A}_{i,r}$, как следует из соотношения (7.13), вычисляется по формуле

$$M_{i,r} = \sum_{j=0}^{r} \tilde{K}(m,j). \tag{7.25}$$

Ограничения (7.22) учитываются следующим образом: вместо каждого слагаемого, соответствующего недопустимому значению $\mathbf{x}$, в сумму (7.24) войдет $\Theta(\mathbf{x})$ — функция степени недопустимости данного $\mathbf{x}$, учитывающая информацию о значениях функционала F на множестве $\mathbf{X}$, например, вида

$$\begin{aligned} \theta_1(\mathbf{x}) &= \max_{\mathbf{y}\in\mathbf{X}} \max\{F(\mathbf{y}),\ -F(\mathbf{y})\}, \\ \theta_2(\mathbf{x}) &= u(\mathbf{x}), \\ \theta_3(\mathbf{x}) &= Q(\theta_1(\mathbf{x}), F(\mathbf{x}), u(\mathbf{x})), \end{aligned} \tag{7.26}$$

где $Q(u,v,t)$ — некоторая монотонно возрастающая по всем своим аргументам действительная функция.

Далее АОДИМ выбирает для последующего поиска октант с номером $N_п$ ($N_п=1, ..., 2^m$) в соответствии со следующей схемой:

1) если $S_{N,r}=\min\limits_{i\in A} S_{i,r}<F(\mathbf{x}_0)$ ($A=\{1, ..., 2^m\}$, то $N_п=N$;

2) если октантов с номерами $N_1, ..., N_s$, удовлетворяющими условиям п.1, несколько, то когда условия п.1 выполняются для какого-либо, но наибольшего, если их несколько, $r=(R-1), ..., 1$ и $A=\{N_1, ..., N_s\}$;

3) если условия пп. 1 и 2 не дают возможности отдать предпочтение ни одному из номеров $N_1, ..., N_s$, то выбирается произвольный из них (например, с наименьшим номером);

4) если $\min\limits_{i\in A} S_{i,r}\geqslant F(\mathbf{x}_0)$ при $r>r_0$ $(1\leqslant r_0\leqslant \tilde{R})$, то условие п.1 проверяется с каким-нибудь (но максимальным) значением r из совокупности $\{r_0-1, ..., 1\}$;

5) если условия пп. 1—4 не позволяют выбрать $N_п$, то $N_п$ — номер того октанта, в котором отобрано наилучшее решение $\mathbf{x}_{нл}$, причем $F(\mathbf{x}_{нл})<F(\mathbf{x}_0)$ (если и таких октантов несколько, то любой из них).

Если условия пп. 1—5 не обеспечивают выбора нужного направления, решение $\mathbf{x}_0$ объявляется приближенным локально оптимальным, в противном случае дальнейший поиск осуществляется с помощью перебора всех целочисленных точек, принадлежащих октанту с номером $N_п$ и лежащих внутри цилиндра радиуса $R_ц$ ($R_ц$ задается априори) с осью симметрии, проходящей через точки $\mathbf{x}_0$ и $\mathbf{x}_{нл}$. Поиск прекращается по наступлении ситуации невозможности улучшения значений F, после чего процедура выбора направления поиска в соответствии с условиями пп. 1—5 повторяется. Центральным в схеме алгоритма является поиск той части окрестности $\mathbf{U}_H(\mathbf{x}_0,r)$, в которой достигается минимум среднего значения обобщенного градиента функционала F. Это позволяет прогнозировать тенденции в изменениях поведения функционала F, что, в свою очередь, обеспечивает продвижение в направлении уменьшения значений F с помощью процедуры линейного поиска (аналогично алгоритму метода направляющих окрестностей). Таким образом, непосредственный перебор элементов окрестности $\mathbf{U}_H(\mathbf{x}_0,r)$ точки $\mathbf{x}_0$ используется лишь как вспомогательное средство для выбора направления, перспективного для дальнейшего поиска. В случае невозможности выделения такого направления в АОДИМ осуществляется процесс улучшения решения с помощью схемы перебора, аналогичной алгоритму метода вектора спада. Существенным является то, что в процессе работы АОДИМ используется несколько альтернативных режимов, соответствующих различным способам улучшения данного решения. Это обеспечивается схемой алгоритма, отражающей последовательность применения указанных способов улучшения.

Проведем анализ работоспособности АОДИМ для детерминированных систем. Множество $\mathbf{D}$ целочисленно выпуклое, если для любых $\mathbf{x},\mathbf{y}\in\mathbf{D}$ всякая целочисленная точка $\mathbf{z}$, такая, что $\mathbf{z}=\alpha\mathbf{x}+\beta\mathbf{y}$ $(\alpha,\beta>\alpha+\beta=1)$, также принадлежит $\mathbf{D}$. Справедлива следующая теорема

Теорема 7.1. Пусть $\mathbf{D}$ — целочисленно выпуклое подмножество пространства $\mathbf{R}^m$, F — выпуклый функционал, заданный на $\mathbf{D}$ и удовлетворяющий следующим условиям:

1) $|F(\mathbf{x})|\leqslant c<\infty$ (ограниченность F);

2) существует такое R_0, что

$$\min_{\substack{x,y,\in U_H(z,R_0)\cap D \\ x\neq y}} |F(x)-F(y)| \geqslant d>0 \text{ для всех } z\in D.$$

Тогда АОДИМ с максимальным радиусом поиска $\bar{R}\leqslant R_0$, начальной точкой x_0, такой, что $\min_{y\in D} H(x_0,y)\leqslant R$ (т. е. отстоящей от D не далее чем на R), использующий в качестве штрафной функции $\theta_1(x)=u(x)$, сходится к решению задачи (7.16), (7.22).

Доказательство. Для произвольной целочисленной начальной точки $x_0\in X$ рассмотрим множество S_{x_0} — совокупность допустимых значений аргумента, значения F в которых не больше $F(x_0)$: $S_{x_0}=\{y: F(y)\leqslant F(x_0)\}$. Пусть $S_{x_0}=\{y_1, \ldots, y_n, \ldots\}$. Можно показать, что S_{x_0} целочисленно выпукло. Действительно, пусть y_{i1}, y_{i2}, y_{i3} таковы, что $y_{ij}\in D$ $(j=1,2,3)$ и $y_{i2}=\alpha y_{i1}+\beta y_{i3}$ $(\alpha,\beta>0,\ \alpha+\beta=1)$. Тогда из определения выпуклой функции получим

$$F(y_{i2}) = F(\alpha y_{i1}+\alpha y_{i3}) \leqslant \alpha F(y_{i1}) + \beta F(y_{i3}) \leqslant F_0.$$

Отсюда следует, что $U_H(x_0, R)$ обязательно имеет непустое пересечение с S_{x_0}. Следовательно, АОДИМ улучшает значение $F(x_0)$ (если оно не оптимально) с помощью одного из условий пп. 1—5 (по крайней мере с помощью п.5), что, принимая во внимание условия 1,2 теоремы, и доказывает утверждение.

Заметим, что при доказательстве теоремы 7.1 мы не использовали предположение о конечности множества X.

Для исследования эффективности функционирования АОДИМ в условиях нерегулярной структуры критерия F представим F с помощью следующей модели стохастического программирования. Допустим, что нам доступно наблюдение реализаций случайного критерия $f(x)=F(x)+\xi(x)$, где $F(x)$ — детерминированная целочисленно выпуклая составляющая, которую следует минимизировать; $\xi(x)$ — случайная величина, заданная на измеримом пространстве (Ω, A, P).

Теорема 7.2. Предположим, что случайные величины $\xi(x)$ $(x\in X)$ независимые центрированные и имеют конечные моменты второго порядка $m_2(x)$. Пусть АОДИМ, осуществляющий поиск минимума $F(x)$, выносит решение о выборе перспективного дальнейшего поиска под номером $N_п$ в соответствии с п.1 алгоритма, применяемого с радиусом поиска r $(r>0)$.

Тогда АОДИМ, производящий минимизацию реализаций $f(x)$, выбирает для дальнейшего поиска также октант под номером $N_п$ с вероятностью P_1, удовлетворяющей следующему неравенству:

$$P_1 \geqslant 1-\delta. \tag{7.27}$$

Здесь $\delta=\tilde{m}_2/(\varepsilon^2 M_{N_{п,r}})$; $\tilde{m}_2=\max_{x\in X} m_2(x)$; ε — требуемый уровень уклонения среднего значения реализаций $f(x)$ на множестве $A_{l,r}$ от своего математического ожидания.

Доказательство: Пусть $x_{N_{п,j}}$ $(j=1, \ldots, M_{N'_{п,r}})$ — точки, принадлежащие октанту с номером $N_п$ и отстоящие от x_0 не далее чем на расстоянии $H(x_0, x_{N_{п,j}})=r$.

Используя неравенство Чебышева, имеем следующую оценку вероятности уклонения:

$$P\left\{\left|\sum_{j=1}^{M_{N_{\text{п}},r}}\frac{\left(F\left(\mathbf{x}_{N_{\text{п}},j}\right)-f\left(\mathbf{x}_{N_{\text{п}},j}\right)\right)}{M_{N_{\text{п}},r}}\right|>\dot{\varepsilon}\right\}=$$

$$=P\left\{\left|\sum_{j=1}^{M_{N_{\text{п}}}}\frac{\xi\left(\mathbf{x}_{N_{\text{п}},j}\right)}{M_{N_{\text{п}},l}}\right|>\varepsilon\right\}\leqslant$$

$$\leqslant\frac{\sum_{j=1}^{M_{N_{\text{п}}}}m_2\left(\mathbf{x}_{N_{\text{п}},j}\right)}{\varepsilon^2M_{N_{\text{п}},r}^2}\leqslant\frac{\tilde{m}_2}{\varepsilon^2M_{N_{\text{п}},r}}.$$

Отсюда ясен выбор оценки для P_1.

Соотношение (7.27) дает универсальную, не зависящую от конкретного распределения шумов $\xi(\mathbf{x})$ оценку вероятности правильного выбора алгоритмом направления дальнейшего поиска. В конкретных примерах соотношение может дать заниженную оценку указанной вероятности, которая может быть существенно уточнена при решении задачи с конкретной моделью шумов (например, что чаще всего бывает на практике, гауссовских).

При применении АОДИМ вероятность P_1 может быть существенно повышена путем выведения из суммы нескольких слагаемых, являющихся большими выбросами случайного поля $f(\mathbf{x})$, т. е. нескольких максимальных и минимальных слагаемых. В этом случае эффективность АОДИМ может быть заметно улучшена именно на этапе определения перспективного направления для поиска [79]. Кроме того (как уже указано в § 2.1), представляется целесообразным сочетание настоящего подхода с широко применяемыми методами случайного поиска для локализации зоны абсолютного экстремума (см. также [71]).

Опишем модификацию АОДИМ, которая более полно учитывает информацию о топологических свойствах оптимизируемого функционала $F(\cdot)$ в процессе определения направления, перспективного для поиска оптимального решения.

Идея улучшения процесса сходимости заключается в использовании имеющихся элементов окрестности $\mathbf{U}_{\mathbf{H}}(\mathbf{x}_0,R)$ начальной точки поиска $\mathbf{x}_0$. Для этого производится попытка интерполяции точек опорной поверхности $F(\cdot)$ в непрерывной окрестности $\mathbf{x}_0$, ограниченной поверхностью уровня $\mathbf{P}_{\mathbf{H}}(\mathbf{x}_0,R)$ $(R>0)$. С помощью интерполирующей зависимости можно вычислить аппроксимирующий градиент $F(\cdot)$ в $\mathbf{x}_0$ и принять решение о перспективном направлении поиска. В рамках всей процедуры поиска, включающей, как в АОДИМ, этапы выбора перспективного направления и линейного поиска вдоль выбранного направления, осуществляется последовательный перебор точек в коридоре, окружающем указанное направление. Основная идея, таким образом, состоит в попытке применения хорошо разработанных в настоящее время методов интерполяций функций многих переменных для вычисления $F'(\mathbf{x}_0)$ — функции некоторой непрерывной окрестности $\mathbf{x}_0$. Эта задача является классической, и ее решению посвящено большое количество работ [78, 80, 81].

Например, когда область аппроксимации $\mathbf{D}$ замкнута в $\mathbf{R}^m$ и известно, что функция $F(\cdot)$ непрерывная $(F\in\mathbf{C}(\mathbf{D}))$, то для любого $\varepsilon>0$ существует многочлен $P(\mathbf{x})$ переменных $\mathbf{x}^{\mathrm{T}}=(x_1,\ldots,x_m)$, что $|P(\cdot)-F(\cdot)|_\infty\leqslant\varepsilon$ в метрике, определяемой соотноше-

нием $|f-g|_\infty = \sup\limits_{\mathbf{x}\in\mathbf{R}^m} |f(\mathbf{x})-g(\mathbf{x})|$. Указанный многочлен $P(\mathbf{x})$ представляет собой алгебраическую форму

$$P(\mathbf{x}) = \sum_{\bar{0}\leqslant\bar{k}\leqslant\bar{n}-1} a_{\bar{k}}\bar{x}^{\bar{k}}, \tag{7.28}$$

где $\bar{x}^{\bar{k}} = x_1^{k_1}, \ldots, x_m^{k_m}$, $\bar{k}$, $\bar{n}$ — мультииндексы; $\bar{k}=(k_1, \ldots, k_m)$; $\bar{n}=(n_1, \ldots, n_m)$. Соответственно $\bar{n}-1=(n_1-1, \ldots, n_m-1)$, а неравенство $\bar{0}\leqslant\bar{k}\leqslant\bar{n}-1$ понимается покомпонентно: $0\leqslant k_i\leqslant n_i-1$, где $i=1, \ldots, m$. Легко видеть, что полином $P(\mathbf{x})$ содержит $N[P(\mathbf{x})]=n_1\times\ldots\times n_m$ коэффициентов.

Сформулированный результат представляет собой хорошо известную в классическом анализе теорему Вейерштрасса. Аналогичный результат имеет место и для m-мерного тора $s\times\ldots\times s$ при аппроксимации тригонометрическими полиномами от m переменных:

$$\bar{P}(x) = \sum_{\substack{|k_i|\leqslant n_i-1,\\ 1\leqslant i\leqslant m}} C_{\bar{k}} \exp\{j\bar{k}\bar{x}\},$$

где $\bar{k}\bar{x}=k_1x_1+\ldots+k_mx_m$ и $C_{-\bar{k}}=C_{\bar{k}}$. Отметим, что полином $P(x)$ имеет $[P(x)]=(2n_1-1)\times\ldots\times(2n_m-1)$ коэффициентов.

Рассмотрим некоторую область интерполяции $\mathbf{D}\subseteq\mathbf{R}^m(\mathbf{x}_0\in\mathbf{D})$ функционала $F(\cdot)$, заданного с помощью N точек $x^{(1)}, \ldots, x^{(N)}$ $(x^{(i)}=(x_1^{(i)}, \ldots, x_m^{(i)})$, $i = 1, \ldots, N)$ — узлов интерполяции. Найдем многочлен $P(x)=\Sigma a_kx^k$ $(|k|=k_1+\ldots+k_m)$, совпадающий с $|k|\leqslant m$ в заданных узлах интерполяций:

$$P(x^{(i)}) = F(x^{(i)}),\ i = 1, \ldots, N.$$

Для этого сформируем линейную (относительно неизвестных коэффициентов a_k) систему уравнений

$$F(x^{(i)}) = \sum_k a_k [x^{(i)}]^k,\ i = 1, \ldots, N.$$

Для разрешимости системы необходимо, чтобы матрица известных коэффициентов была неособая, т. е. чтобы $\det\mathbf{A}=\det\{[(x^{(i)})^k]_{i,k}\}\neq 0$. Так как этот детерминант обращается в нуль на многообразиях размерности N_m-1, то выбор узлов интерполяции возможен в принципе.

Фундаментальную систему многочленов $P_1(x), \ldots, P_N(x)$ можно получить из системы, в которой коэффициенты a_k получены из условия $F(x^{(i)})=\delta_{ji}$, где δ_{st} — символ Кронекера: $\delta_{st}=1$, если $s=t$, и $\delta_{st}=0$, если $s\neq t$. При этом явный вид многочлена $P_j(x)$ следующий:

$$P_j(x) = \sum_k a_k^{(j)}\bar{x}^{\bar{k}},$$

а сам интерполяционный многочлен записывается в виде

$$P(\bar{x}) = \sum_{i=1}^{N} F(\bar{x}^{(i)})\, P_i(\bar{x}).$$

Точность аппроксимации в многомерном случае определяется следующим образом:

$$|F(\cdot) - P(\cdot)| \leqslant (1 + \Lambda_N)\, E(F),$$

где $E(F) = \inf\limits_{p \in \mathbf{L}} \|F - P\|_\infty$; $\mathbf{L}$ — множество интерполяционных степенных многочленов $\mathbf{L} \subset \mathbf{C}[\mathbf{D}]$ ($\mathbf{C}[\mathbf{D}]$ — множество непрерывных на $\mathbf{D}$ функций); Λ_N — некоторая константа (константа Лебега): $\Lambda_N = \max\limits_{\mathbf{x} \in \mathbf{D}} \sum\limits_{i=1}^{N} |P_i(\mathbf{x})|$. В этом случае, когда множество $\mathbf{D}$, на котором производится аппроксимация функционала $F(\cdot)$,— многомерный интервал,

$$\mathbf{D}\{\mathbf{x} \in \mathbf{R}^m : a_i \leqslant x_i \leqslant b\}, \tag{7.29}$$

где $\mathbf{x}_0 \in \mathbf{D}$. Обычно множество $\mathbf{D}$ является многомерным кубом с центральной точкой $\mathbf{x}_0$:

$$\mathbf{D} = \{\mathbf{x} \in \mathbf{R}^m : -c \leqslant x_i - x_{0i} \leqslant +c\},\ 1 \leqslant i \leqslant m, \tag{7.30}$$

при этом c — натуральное число.

Если $x_i^{(1)}, \ldots, x_i^{(n_i)}$ ($i = 1, \ldots, m$) — i-е координаты узлов интерполяции, то в $\mathbf{D}$ узлов интерполяций $N = n_1 \times \ldots \times n_m$ (если $\mathbf{D}$ — куб, то $n_1 = \ldots = n_m = n$ и $N = n^m$). На основании множества узлов интерполяции

$$\mathbf{x}^p = \left(x_1^{(p_1)}, \ldots, x_m^{(p_m)}\right), \quad p = (p_1, \ldots, p_m),$$

$$p_i = 1, \ldots, n_i, \quad i = 1, \ldots, m,$$

определяется вид многочлена.

Соотношения, определяющие фундаментальный многочлен, связанный с узлом $\mathbf{x}^{(p)}$, следующие:

$$P_p(\mathbf{x}) = \prod_{i=1}^{m} \prod_{\substack{k=1 \\ k \neq p_i}}^{n_i} \frac{x_i - x_i^{(k)}}{x_i^{(p_i)} - x_i^{(k)}},$$

$$P(\mathbf{x}) = \sum_p F(x^{(p)})\, P_p(x)$$

— интерполяционный многочлен имеет константу Лебега $\Lambda_N = \prod\limits_{i=1}^{m} n_i$ (если множество $\mathbf{D}$ — симметричный куб, то $(\ln n)^m$).

С помощью интерполяционного многочлена $P(x)$ вычисляется вектор-градиент

$$\nabla P(\mathbf{x}) = \left(\frac{\partial p(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial p(\mathbf{x})}{\partial x_m}\right)^\tau,$$

на основании которого вычисляется перспективное направление в точке x_0:

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} \approx \Delta P(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0}.$$

Аппроксимация с помощью приведенных формул достаточно эффективна и обеспечивает построение гладкой поверхности интерполяции.

Кусочно-полиномиальная интерполяция может быть реализуема с помощью сплайн-функций [80, 81]. Сплайны позволяют осуществить гладкую интерполяцию многочленами заданной степени и получить достаточно качественную оценку $\nabla F(\cdot)$ по небольшому количеству опорных точек на поверхности $F(\cdot)$.

При использовании системы точек (7.29) для определения (оценки) вектор-градиента $\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}$ требуется подключение процедуры полного перебора целочисленных точек из многомерного параллелепипеда

$$\Pi = \{ \mathbf{x} \in \mathbf{R}^m : \mathbf{x}^\tau = (x_1, ..., x_m),\ a_i \leqslant x_i \leqslant b_i,\quad i = 1, ..., m,$$

где (a_i, b_i) $(i=1, ..., m)$ — целочисленные граничные точки параллелепипеда. (Эффективный алгоритм, а также реализующая его программа, основанные на процедуре лексикографического перебора в заданных границах, приведены в [59].)

Проведенный анализ позволяет оценить возможность построения аппроксимации градиента $\nabla F(\cdot)$, а также алгоритм вычисления указанной оценки. В соответствии с сформулированной выше общей схемой оптимизации дискретных параметров на основе $\nabla F(\mathbf{x}_0)$ (или ее оценки) формируется последовательность точек $x_1, x_2, ...$, полученная с помощью процедуры линейного поиска $x_{i+1} = x_i + \Delta$ $(i=0, 1, ...)$, в которой $\Delta = t[-\nabla F(\mathbf{x}_0)]$, и суммирование осуществляется до увеличения функционала $F(\cdot)$, т. е. до тех пор, пока не начнет выполняться неравенство $F(x_i) \leqslant F(x_{i+1})$, где $t[-\nabla F(\mathbf{x}_0)]$ — функционал (оператор), обеспечивающий целочисленность генерируемых решений и максимальное сопряжение с направлением, определяемым антиградиентом — $\nabla F(\mathbf{x}_0)$ (более подробно о строении оператора $t(\cdot)$ см. ниже). Учет необходимости получения допустимых решений достигается введением штрафных функций, подобно тому как это реализовано в основном варианте АОДИМ.

Идея построения оператора $t(\cdot)$ заключается в последовательном переборе точек некоторого исходящего из точки начала поиска круглого конуса с осью симметрии, который определяется направляющим вектором градиента (точнее, его оценкой) $\nabla F(\mathbf{x}_0)$. Граничная поверхность конуса определяется из условия превышения объема просматриваемых целочисленных точек или невозможности улучшения значений функционала $F(\cdot)$ в пределах данного конуса, соответствующего направлению $\nabla F(\mathbf{x}_0)$.

Назначение оператора $t(\cdot)$ — в генерации целочисленных точек, находящихся в некотором пространственном объеме, заключающем выбранное направление (определяемое, например, оценкой градиента $\nabla F(\mathbf{x}_0)$). При изложении используем некоторые понятия выпуклого анализа [57].

Множество $\mathbf{K}$, принадлежащее m-мерному евклидову пространству $\mathbf{R}^m$ $(m \geqslant 1)$,— выпуклый конус с вершиной в нуле $\bar{0}$ $(\bar{0}^\tau = (0, ...,))$, если оно удовлетворяет следующим двум условиям:

1) множество $\mathbf{K}$ выпукло;

2) если $\mathbf{x} \in \mathbf{K}$, то при любом скалярном параметре $s > 0$ и $s\mathbf{x} \in \mathbf{K}$.

Целочисленное сечение конуса является пересечением поверхности конуса плоскостью, определяемого целочисленным значением $x^{(k)} = l$ (k-й координаты $\mathbf{R}^m$). Все точки, принадлежащие внутренности конуса, обязательно лежат на одном из

целочисленных сечений. Это, в свою очередь, позволяет генерировать целочисленные точки конуса, продвигаясь по последовательно формируемым сечениям.

При оценке эффективности предлагаемого алгоритма, основанного на вычислении функционала аппроксимации поверхности $F(\cdot)$ и соответствующего приближения градиента $\nabla F(\mathbf{x}_0)$ с последующим линейным поиском улучшения $F(\mathbf{x})$ на целочисленных точках конуса $\mathbf{K}(\mathbf{x}_0)$ с вершиной в $\mathbf{x}_0$, предлагается подход, связанный с использованием аппроксимирующей задачи непрерывного программирования для анализа скорости сходимости $F(\mathbf{x}) \to \min\limits_{\mathbf{x}\in\mathbf{R}^m}$ (задача называется сопровождающей).

При этом генерируется некоторая последовательность точек y_i $(i \geqslant 0)$, удовлетворяющая соотношениям

$$y_{i+1} = y_i + \alpha_i h_i, \tag{7.31}$$

$$-\langle \nabla F(y_i), h\rangle \geqslant \gamma \|\nabla F(y_i)\| \|h_i\|, \tag{7.32}$$

где $i \geqslant 0$; $h_i = -\nabla F(y_i)$; $\gamma > 0$, а α_i — наименьшее положительное λ, для которого

$$F(y_i + \alpha_i h_i) = \min_{\alpha \geqslant 0} F(y_i + \alpha h_i). \tag{7.33}$$

Предположение 1. Функция $F(\cdot)$ как функция непрерывного аргумента $\mathbf{x}\in\mathbf{R}^m$ дважды непрерывно дифференцируема и в окрестности точки оптимума $\mathbf{x}^{\text{opt}}$, реализующей локальный минимум, строго выпукла, т. е. если для любых $\mathbf{y}, \mathbf{z}\in\mathbf{R}^m: \mathbf{y}\neq\mathbf{z}$ и любого $\alpha\in(0,1)$:

$$F(\alpha\mathbf{z} + (1-\alpha)\mathbf{y}) < \alpha F(\mathbf{z}) + (1-\alpha) F(\mathbf{y}).$$

При данном предположении для всех $\mathbf{x}\in\mathbf{R}^m$, для которых $\|\mathbf{x}-\mathbf{x}^{\text{opt}}\| < \varepsilon$ $(\varepsilon > 0)$, существуют m, M: $0 < m < M$, для которых

$$m\|\mathbf{y}\|^2 \leqslant \langle \mathbf{y}, \partial^2 \frac{F(\mathbf{x})}{\partial x^2}\mathbf{y}\rangle \leqslant M\|\mathbf{y}\|^2. \tag{7.34}$$

Здесь $\langle\cdot,\cdot\rangle$ — символ скалярного произведения элементов пространства $\mathbf{R}^m$. Потребуем выполнения соотношения (7.33) в достаточно большой области, содержащей точку оптимума $\mathbf{x}^{\text{opt}}$.

При использовании алгоритмов минимизации $F(\cdot)$ имеют место оценки [16]

$$\|\mathbf{y}_i - \mathbf{x}^{\text{opt}}\| \leqslant \frac{2}{m}[F(\mathbf{y}_i) - F(\mathbf{x}^{\text{opt}})] \tag{7.35}$$

или

$$\|y_i - \mathbf{x}^{\text{opt}}\| \leqslant \frac{2}{m}[F(\mathbf{x}_0) - F(\mathbf{x}^{\text{opt}})]\,\beta^i, \tag{7.36}$$

где коэффициент β $(\beta > 0)$ подсчитывается по формуле $\beta = 1 - (\gamma^m/M)^2$; $\gamma\in(0,1)$ — показатель роста скалярного произведения.

Из соотношений (7.34)—(7.36) можно получить

$$F(\mathbf{y}_i) - F(\mathbf{x}^{\text{opt}}) \leqslant [F(\mathbf{x}_0) - F(\mathbf{x}^{\text{opt}})]\,\beta^i;$$

$$\|\mathbf{y}_i - \mathbf{x}^{\text{opt}}\| \leqslant \sqrt{\frac{2[F(\mathbf{x}_0) - F(\mathbf{x}^{\text{opt}})]}{m}} (\sqrt{\beta})^i, \qquad (7.37)$$

где $i \geqslant 0$.

При наличии системы функциональных ограничений, требующих найти оптимальное решение в некотором выпуклом множестве D путем введения специальных штрафных функций, исходная задача сводится к задаче безусловной оптимизации на всем целочисленном подмножестве (решетке) пространства $\mathbf{R}^m$. В этом случае обращение к сопровождающей задаче непрерывного программирования позволяет получить оценку эффективности сходимости процедуры поиска АОДИМ. Соотношение (7.37) позволяет оценить расстояния от решения $\mathbf{y}_i$, сформированного на i-й итерации непрерывного поиска от $\mathbf{x}^{\text{opt}}$ — оптимального решения непрерывной задачи.

Предположение 2. Решение дискретной задачи $\mathbf{y}^{\text{opt}}$ содержится в некоторой окрестности $U_H(\mathbf{x}^{\text{opt}}, r)$ оптимального решения $\mathbf{x}^{\text{opt}}$ непрерывной задачи с заданным радиусом r ($r > 0$). Это означает по сути возможность использования приближенного решения, достаточно близко (на расстоянии, не превышающем r) отстоящего от оптимального. В этом случае с помощью неравенства треугольника оценка расстояния между оптимальным решением дискретной задачи $\mathbf{y}^{\text{opt}}$ и i-й итерацией АОДИМ $\mathbf{x}_i$ оценивается (в евклидовой метрике, но могут быть получены аналогичные результаты и для любой другой метрики, заданной на $\mathbf{R}^m$) величиной

$$\|\mathbf{y}^{\text{opt}} - \mathbf{x}_i\| \leqslant \|\mathbf{x}^{\text{opt}} - \mathbf{y}^{\text{opt}}\| + \|\mathbf{y}_i - \mathbf{x}^{\text{opt}}\| + \|\mathbf{y}_1 - \mathbf{x}_i\| \leqslant r + \\ + \frac{2}{m}[F(\mathbf{x}_0) - F(\mathbf{x}^{\text{opt}})]\,\beta^i + \|\mathbf{y}_i - \mathbf{x}_i\|. \qquad (7.38)$$

Оценим разность между непрерывным (принадлежащим сопровождающей задаче) и дискретным решениями $\delta_i = \|\mathbf{y}_i - \mathbf{x}_i\|$ на i-й интерации. Рассмотрим многомерное тело, образованное конусом и его граничной сферической поверхностью $\Gamma(\mathbf{y}_i)$, проходящей через точку $\mathbf{y}_i$. Тело соответствует решению сопровождающей непрерывной задачи на i-й итерации, т. е. радиусом $R = \|y_i - x_{i-1}\|$.

Введем множество $\mathbf{A}_i$ со следующими определяющими его свойствами:

$$\mathbf{A}_i \in \Gamma(\mathbf{y}_i), \quad \mathbf{z} \in \mathbf{A}_i \to F(\mathbf{z}) \leqslant F(\mathbf{x}_{i-1}).$$

Если $F(\cdot)$ — выпуклая функция, то множество $\mathbf{A}_i$ также выпуклое в $\Gamma(\mathbf{y}_i)$ (т. е. рассматриваемое как подмножество пространства $\Gamma(\mathbf{y}_i)$).

Обозначим через $\mathbf{B}_i$ наибольшее (с наибольшим диаметром, или, что эквивалентно, с наибольшим коэффициентом S, входящим в определение конуса) сечение конуса, содержащее целочисленные точки z, удовлетворяющие условию $F(\mathbf{z}) < F(\mathbf{x}_{i-1})$. Множество $\mathbf{B}_i$ (являющееся проекцией множества $\mathbf{A}_i$ с центром проецирования $\mathbf{x}_i$) с требуемыми свойствами существует по крайней мере для множеств $\mathbf{A}_i$, содержащих сферический элемент достаточно большого радиуса d, проекция которого на целочисленные сечения конуса содержит целочисленные точки. (Критериям существования целочисленных точек в выпуклых множествах евклидовых пространств посвящена [57].)

Таким образом, требуемая оценка $\delta_i = \|\mathbf{x}_i - \mathbf{y}_i\|$ определяется максимальным расстоянием от точки $\mathbf{y}_i$ до наиболее удаленного целочисленного элемента сечения $\mathbf{B}_i$, что может быть, в свою очередь, оценено расстоянием от $\mathbf{y}_i$ (центра симметрии $\Gamma(\mathbf{y}_i)$) до границы $\mathbf{B}_i$: $\delta_i^2 = r_i^2 + (\Delta h_i)^2$, где r_i—максимальный радиус эллиптического сечения $\mathbf{B}_i$; Δh_i—расстояние от $\mathbf{y}_i$ до $\mathbf{B}_i$ (Δh_i определяется из условия пропорциональности отношения объемов $\mathbf{B}_i$ и $\mathbf{A}_i$ расстоянием последних от вершин конуса).

Проведенный анализ можно сформулировать в виде следующей теоремы.

Теорема 7.3. Пусть $F(\cdot)$ — выпуклая функция на $\mathbf{R}^m$ и выполняются предположения 1, 2. Тогда имеет место следующая оценка расстояния между решением $\mathbf{x}_i$, генерируемым АОДИМ на i-й итерации, от оптимального $\mathbf{y}^{\text{opt}}$:

$$\left\|\mathbf{y}^{\text{opt}} - \mathbf{x}_i\right\| \leqslant r_1 + r_2 + r_3,$$

где r_1 — минимальный радиус окрестности $\mathbf{U}_H(\mathbf{x}^{\text{opt}}, r_1)$ оптимального решения сопровождающей непрерывной задачи; $r_2^2 = (2/m)[F(\mathbf{x}_0) - F(\mathbf{x}^{\text{opt}})]\beta^i$; $r_3 = (r_i^2 + \Delta h_i)^{1/2}$.

Указанная оценка позволяет оценить требуемое число итераций алгоритма (и, следовательно, требуемое количество замеров значений функционала $F(\cdot)$ для достижения требуемой точности вычисления дискретного оптимума $F(\cdot)$. Отметим, что фигурирующие в оценке для значения $F(\mathbf{x}^{\text{opt}})$ могут быть вычислены с помощью алгоритма статистической оценки экстремальных значений критерия.

ПРОГРАММА AODIM

Назначение: оптимизация целевой функции, как стохастической, так и детерминированной, на целочисленной решетке, допускающей разрывы (язык Фортран).

Входные параметры:

N — количество векторов;

IRM — радиус перебора точек;

IRMB — ширина трубки просмотра при движении по направлению;

X0 — начало координат (начальная точка поиска);

NJUMP — число прыжков в случае неудач;

IPR1 — признак структуры функционала оптимизации: 0 — стохастический, 1—детерминированный;

IPR4 — признак выбора перспективного направления при IPR1 = 1.

Выходные параметры:

XO — координаты оптимальной точки;

FOP — значение функционала в оптимальной точке XO.

Внутренние параметры:

XS — абсолютный вектор координаты текущей точки оптимизации;

IRX — признаки допустимости;

IX, IX1, IX2, IX3, IZ — вспомогательные массивы;

IXMAXT — координаты перспективной точки;

IZOP — координаты оптимальной точки;

IRP — радиус перебора.

Используемые программы:

ISMEAN — вызывающая (тестовая);

FORTUB — вычисление значения функционала;

SMEAN — получение среднего значения функционала в ортанте;

JUMPT — движение по направлению;
BLOKRS — осмотр ближайшей окрестности;
SRED — получение среднего значения функционала в текущей точке;
CHEINO — изменение индекса ортанта;
IAT — присвоение значения текущему массиву;
COIMA1 — копирование текущего массива;
OKRS — генератор точек в ортанте;
GENPL — генератор точек, ближайших к биссектрисе;
BESTP — генератор точек на осях координат.

Пример. Решение задачи оптимизации вариантов бортового оборудования летательных аппаратов с дистанционным управлением — найти минимум функционала (критерия качества оборудования)

$$F(\mathbf{x}) = \sum_{i=1}^{m} Q_i \left\{ \left[\sum_{j=1}^{i} R_{ij} x_j \right] + \left[\prod_{j=1}^{i} P_{ij} x_i^{-1} \right] \right\}$$

при наличии ограничений

$$G_i = \left[\sum_{j=1}^{m} c_i x_i \right] \leqslant \tilde{G}_i \left[\prod_{j=1}^{m} \tilde{c}_j x_j \right], \quad i = 1, \ldots, l,$$

где $\mathbf{x}^{\mathrm{T}} = (x_1, \ldots, x_m)$ — m-мерный вектор, описывающий возможные альтернативные варианты реализации модулей; Q_i, R_{ij}, P_{ij}, c_j, $\tilde{c}_j$ — неотрицательные константы; G_j, $\tilde{G}_j$ — заданные нелинейные функции ограничений (m — количество отдельных модулей бортового оборудования). Область изменения параметров $\mathbf{x}_i$ ($i=1, \ldots, m$) — конечное множество натуральных чисел: $\mathbf{x}_i \in \{1, \ldots, N_i\}$ ($N_i \geqslant 1$) ($\mathbf{x}_i$ индексирует возможные состояния модуля).

При достаточно больших m, N_i (на практике $m=5...20$, $N_i=5...50$) полный перебор всего множества возможных значений параметров нереализуем даже с помощью современных быстродействующих ЭВМ. Сформулированная задача может служить одновременно и тестом для анализа эффективности алгоритмов нелинейного дискретного программирования. Действительно, можно показать, что при $Q_i=R_{ij}=P_{ij}=1$ минимум F достигается в точке $x_1=\ldots=x_m=1$ (ограничения не учитывались). Это позволяет проанализировать работу программы АОДИМ при различных начальных точках поиска, которые задавались с помощью генератора псевдослучайных чисел.

Во всех реализациях (проводилось 30 опытов) был достигнут точный минимум $F^{\mathrm{opt}}=35{,}0$ ($m=N_i=7$). Время поиска минимума 0,5...4 с. Машинные эксперименты позволили проанализировать поведение F^{opt}, x^{opt} при изменениях параметров Q_i, R_{ij}, P_{ij}. Например, при увеличении значений P_{ij} ($Q_i=R_{ij}=1$) наблюдается перемещение вектора оптимального решения x^{opt} в сторону увеличения значений компонентов (в частности, при $P_{ij}=3$ $x^{\mathrm{opt}}=(2, 2, 3, 3, 3, 3, 3)$, $P^{\mathrm{opt}}=86{,}0$). Одновременно наблюдается и увеличение времени поиска оптимального решения — 4...10 с при $P_{ij}>2$ (при тех же значениях параметров $m=N_i=7$). Максимальная по сложности задача, решаемая с помощью программы АОДИМ, характеризовалась параметрами $m=15$, $N_i=50$ ($i=1, \ldots, m$).

Точное решение задачи дискретного квадратичного программирования с целевой функцией

$$F(x_1, ..., x_5) = \sum_{i=1}^{5} x_i^2,$$

с начальной точкой поиска $x_1, ..., x_5 = 0$ и областью допустимости $x_i \geqslant 1$ $(i = 1, ..., 5)$ получено за 0,4 с на ЭВМ СМ-1420.

```
      PROGRAM TSMEAN
      DOUBLE PRECISION XRAN
      COMMON /FIKSN/NFKR,NFY,IPRF,IRAZR,NNMAX
      COMMON /OKRGEN/IMAXOK,IMINOK,IOSTOK,IKMOK
      COMMON /OUTOPT/IOUTST
      COMMON /OTLOPT/IPECH,IOTL
      INTEGER XO(5),XS(5)
      DIMENSION IX(5),IZ(5),IZOP(5),IXMAXT(5),IXMOR(5),IXMO(5),
     *    IRX(5),FOPT(5),DFI(5),DFXI(5),IX3(5)

      COMMON /BRAN/ XRAN
      DATA XO/3,3,5,1,-2/
      XRAN=0.130000000001D+12
      IOUTST=0
      NFY=0
      N=5
      IRM=3
      IBI=1
      KOT=0
      IPECH=1
      IOTL=0
      NNMAX=1
      IPRF=1
      NJUMP=1
      IBI=1
      IRM=1
      IRAZR=0
      IPR1=IPRF
      IPR2=0
      IPR3=1
      IPR4=0
      FOP=0
      NFY=0
      NFKR=0
      CALL AODIM(N,IRM,IBI,XO,XS,IXMAXT,IXMOR,IXMO,IZ,IZOP,IX,IX3,
     *          NJUMP,IPR1,IPR2,IPR3,IPR4,FOP)
 9900 STOP
      END
      SUBROUTINE AODIM(N,IRM,IRMB,XO,XS,IXMAXT,IXMOR,IXMO,
     *          IZ,IZOP,IX,IX3,NJUMP,IPR1,IPR2,IPR3,IPR4,FOP)
      COMMON /ZNMAX/ FOPT,SMAXT,SMAXO
      COMMON /OKRGEN/IMAXOK,IMINOK,IOSTOK,IKMOK
      COMMON /FIKSN/NFKR,NFY,IPRF,IRAZR,NNMAX
      COMMON /OUTOPT/IOUTST
      INTEGER XO(N),XS(N),IX3(N)
      DIMENSION IX(N),IZ(N),IXMAXT(N),IXMOR(N),IXMO(N),IZOP(N)
      ND=2**N
      NPP=10
 50   CALL COIMA1(XO,N,IXMAXT)
      NP=1
      IF(IPR1.EQ.0) NP=NPP
```

```
      CALL SRED(NP,N,XO,KIN,SMO)
      IF(KIN.NE.0)RETURN
500   SMAXT=SMO*2
      NT0=0
      FOP=SMO
      FOPT=FOP
      SMAXT=FOP
      CALL COIMA1(XO,N,IXMAXT)
      SMAXO=FOP
      CALL IAT(IZ,N,1)
      IREZ=0
      IRO=0
      IR=0
       IF(IPR3.EQ.0)GO TO 40
        IR=N-1
        IF(IRM.LT.N)IRM=N
40     IR=IR+1
        IF(IR.GT.IRM) GO TO 30
      NT=ND
6     CALL SMEAN(N,NP,XO,XS,IZ,IX,IR,IXMAXT,IXMO,SUN,KOT,
     *                    IPR1,IPR2,IPR3,IPR4,IREZ)
      IF(IOUTST.EQ.1)RETURN
      IF(KOT.EQ.1)GO TO 16
      IF(SUN.GE.SMO)GO TO 16
      NT0=NT
      SMO=SUN
      SMAXOR=SMAXO
      SMAXO=FOP
      IRO=IR
      CALL COIMA1(IXMO,N,IXMOR)
      CALL COIMA1(IZ,N,IZOP)
 16   CONTINUE
      NT=NT-1
      IF(NT.EQ.0)GO TO 20
      CALL CHEINO(N,N,IZAZV,IZ)
      GO TO 6
 20   NT=ND
      CALL IAT(IZ,N,1)
      GO TO 40
 30   CONTINUE
      IF(NT0.NE.0)GO TO 2000
      IF(SMAXT.GE.FOP) GO TO 32
 31   SMO=FOP
      IPR=0
      CALL JUMPT(N,NP,XO,IZ,IXMAXT,NJUMP,IRMB,IPR,
     *                    XS,IX,IXMOR,IXMO,IX3,IZOP,SMO,NOP)
      IF(IOUTST.EQ.1)RETURN
      IF(NOP.NE.0)GO TO 33
 32   IRKOOR=1
      SMO=FOP
      CALL BLOKRS(N,NP,IRM,XO,XS,IX,IZ,IXMAXT,SMO,KOT,
     *                    IPR1,IPR2,IPR3)
      IF(IOUTST.EQ.1)RETURN
      IF(SMO.GE.FOP)RETURN
      GO TO 31
 33   CALL COIMA1(IZOP,N,XO)
      CALL IAT(IZ,N,1)
      GOTO 500
2000  CONTINUE
      CALL COIMA1(IZOP,N,IZ)
      IF(IRO.LT.IRM) GO TO 205
      SMO=FOP
      IPR=1
```

```
      CALL JUMPT(N,NP,XO,IZ,IXMAXT,NJUMP,IRMB,IPR,
     *                  XS,IX,IXMOR,IXMO,IX3,IZOP,SMO,NOP)
      IF(IOUTST.EQ.1)RETURN
      IF(NOP.EQ.0)GO TO 205
        CALL COIMA1(IZOP,N,XO)
      GOTO 500
205   CALL IAT(IX,N,0)
      IX(N)=IRO
      SMAXO=FOP**2
      IREZ=1
      CALL SMEAN(N,NP,XO,XS,IZ,IX,IR,IXMAXT,IXMO,
     *          SUN,KOT,IPR1,IPR2,IPR3,IPR4,IREZ)
      IF(IOUTST.EQ.1)RETURN
      IF(KOT.EQ.1)GO TO 208
      CALL COIMA1(IXMO,N,IXMAXT)
      IF(SMAXO.LT.FOP)GOTO 208
      IRKOOR=1
      SMO=FOP
      CALL BLOKRS(N,NP,IRM,XO,XS,IX,IZ,IXMAXT,SMO,
     *                  KOT,IPR1,IPR2,IPR3)
      IF(IOUTST.EQ.1)RETURN
      IF(SMO.GE.FOP)RETURN
208   SMO=FOP
      IPR=0
      CALL JUMPT(N,NP,XO,IZ,IXMAXT,NJUMP,IRMB,IPR,
     *          XS,IX,IXMOR,IXMO,IX3,IZOP,SMO,NOP)
      IF(IOUTST.EQ.1)RETURN
      CALL COIMA1(IZOP,N,XO)
      GO TO 500
      RETURN
      END
      SUBROUTINE COIMA1(X0,N,X1)
      INTEGER X0(N),X1(N)
      DO 1 I=1,N
1     X1(I)=X0(I)
      RETURN
      END
      SUBROUTINE JUMPT(N,NP,XO,IZ,IXMAXT,NJUMP,IRMB,IPR,
     *                  XS,IX,IX1,IX2,IX3,IZOP,SMO,NOP)
      INTEGER XS(N),XO(N)
      DIMENSION IXMAXT(N),IZ(N),IZOP(N),IX(N),
     *          IX1(N),IX2(N),IX3(N)
      NS=0
      NOP=0
      NKIN=0
      NEYD=0
100   NS=NS+1
       IF(NKIN.EQ.NJUMP)RETURN
       IF(NEYD.EQ.NJUMP)RETURN
        DO 1 I=1,N
         IF(IPR.EQ.0)XS(I)=XO(I)+NS*(IXMAXT(I)-XO(I))
         IF(IPR.EQ.1)XS(I)=XO(I)+NS*IZ(I)
1        CONTINUE
      CALL SRED(NP,N,XS,KIN,SMOS)
      IF(KIN.EQ.0)GO TO 2
      IF(IRMB.LE.0)GO TO 20
      SMOS=SMO
      CALL BLOKRS(N,NP,IRMB,XS,IX1,IX2,IX3,IX,SMOS,
     *                  KOT,0,0,1)
      IF(KOT.EQ.0)GO TO 2
20          NKIN=NKIN+1
            GO TO 100
```

```
 2      IF(SMOS.LT.SMO)GO TO 4
        IF(IRMB.LE.0)GOTO 3
        IF(KIN.NE.0)GO TO 3
          KIN=1
          SMOS=SMO
        CALL BLOKRS(N,NP,IRMB,XS,IX1,IX2,IX3,IX,SMOS,
     *          KOT,0,0,1)
        IF(KOT.NE.0)GO TO 3
         GO TO 2
 3      NEYD=NEYD+1
        GO TO 100
 4      SMO=SMOS
        IF(KIN.EQ.0)CALL COIMA1(XS,N,IZOP)
        IF(KIN.EQ.1)CALL COIMA1(IX,N,IZOP)
        NOP=NOP+1
        NKIN=0
        NEYD=0
        GO TO 100
        RETURN
        END
        SUBROUTINE IAT(IT,N,IA)
        DIMENSION IT(N)
        DO 1 I=1,N
 1      IT(I)=IA
        RETURN
        END
        SUBROUTINE SRED(NP,N,XS,KIN,SMO)
        INTEGER XS(N)
        SMO=0
        DO 1 I=1,NP
        CALL FOPTVB(N,XS,F0,KIN)
        IF(KIN.EQ.1)RETURN
        SMO=SMO+F0
        SMO=SMO/NP
        RETURN
        END
        SUBROUTINE SMEAN(N,NP,XO,XS,IZ,IX,IR,IXMAXT,IXMO,
     *          SU,KOT,IPR1,IPR2,IPR3,IPR4,IREZ)
        INTEGER XO(N),XS(N)
        DIMENSION IX(N),IZ(N),IXMAXT(N),IXMO(N)
        COMMON /ZNMAX/ FOP,SMAXT,SMAXO
        COMMON /OKRGEN/IMAXOK,IMINOK,IOSTOK,IKMOK
                IZAV=0
                IMINOK=0
                IF(IPR3.EQ.1)IMINOK=1
                CALL IAT(IX,N,IMINOK)
                IMAXOK=IR
                IOSTOK=IR
                IF(IPR3.NE.1)GO TO 11
                IF(IR.LT.N)IR=N
                IOSTOK=IR-N
                IMAXOK=1+IOSTOK
11              IX(N)=IX(N)+IOSTOK
        KZ=0
        KOT=0
        SU=0.
        NRAZ=N
        SMAXO=FOP*2.0
        J=0
 5      J=J+1
         IF(IZAV.NE.0)GO TO 3
         IF(J.NE.1)CALL OKRS(NRAZ,N,IMINOK,IMAXOK,IR,IPR3,IX,IZAV)
```

```
      IF(IR.EQ.N)IZAV=1
      DO 4 I=1,N
4      XS(I)=XO(I)+IX(I)*IZ(I)
      IF(IREZ.EQ.0)CALL FOPTVB(N,XS,FS,KIN)
      IF(IREZ.EQ.1)CALL SRED(NP,N,XS,KIN,FS)
      IF(KIN.EQ.0) GOTO 9
      IF(IPR2.EQ.1)SU=SU+FOP**2
      IF(IPR2.EQ.0)GO TO 5
      IF(IPR1.EQ.1.AND.IPR4.EQ.1)GO TO 5
      IF(IREZ.EQ.1)GOTO 5
      KZ=KZ+1
      GOTO 5
9     IF(IPR1.EQ.1.AND.IPR4.EQ.1)GOTO 10
      IF(IREZ.EQ.1)GOTO 10
      SU=SU+FS
      KZ=KZ+1
10    IF(SMAXO.LE.FS)GO TO 12
      CALL COIMA1(XS,N,IXMO)
      SMAXO=FS
 12   IF(IREZ.EQ.1)GOTO 5
      IF(SMAXT.LE.FS)GOTO 5
      CALL COIMA1(XS,N,IXMAXT)
      SMAXT=FS
      GO TO 5
3     IF(KZ.GT.0) GOTO 6
      KOT=1
      SU=FOP**2
      RETURN
6     SU=SU/KZ
      RETURN
      END
      SUBROUTINE OKRS(NRAZ,N,IMINOK,IMAXOK,IR,IPR,IX,IZAV)
      DIMENSION IX(NRAZ)
1000  I=N+1
100   I=I-1
      IF(I.GT.1)GO TO 10
            IZAV=2
            RETURN
10    IF(IX(I).EQ.IMINOK)GO TO 100
      IX(I-1)=IX(I-1)+1
      IX(I)=IX(I)-1
      IF(IX(I-1).NE.IMAXOK)GO TO 2
       IF((I-1).EQ.1)GO TO 1
      IF(IPR.EQ.2)GO TO 1000
      RETURN
1     IZAV=1
      IF(IPR.EQ.2)IZAV=2
      RETURN
2     IF(I.EQ.N) RETURN
      IF(IX(I).EQ.IMINOK) RETURN
              IX(N)=IX(I)
              IX(I)=IMINOK
      RETURN
      END
      SUBROUTINE BLOKRS(N,NP,IRM,XO,XS,IX,IZ,IXMAXT,SMO,
     *                  KOT,IPR1,IPR2,IPR3)
      INTEGER XO(N),XS(N)
      DIMENSION IX(N),IZ(N),IXMAXT(N)
      COMMON /OTLOPT/ IPECH,IOTL
      SU=0
      KOT=1
      SREDF=0
```

```
      NTF=0
      SMOF=SMO
      IRP=0
      IF(IPR1.EQ.1)IRP=IRM
      IF(IPR3.NE.0)IRP=0
      IPRS=IPR
      IRPM=N-1
3000  IRP=IRP+1
      IF(IRP.GT.IRPM)GO TO 4
      CALL BESTP(N,NP,IRP,XO,XS,IX,IXMAXT,SMOS,SUM,
     *           KTF,KIN,IPROP)
      IF(KIN.EQ.1)GO TO 3000
       KOT=0
      IF(SMOS.GE.SMO)GO TO 3000
       SMO=SMOS
       RETURN
4     IPR=IPRS
4000  IPR=IPR+1
      IF(IRP.GT.IRPM)GOTO 500
      CALL IAT(IX,N,0)
      N1=N-IRP+1
      DO 1 I=N1,N
1     IX(I)=1
      IZAV=0
      J=0
 2000 J=J+1
      IF(IZAV.EQ.1)GO TO 4000
      IF(J.NE.1)CALL GENPL(N,IRP,IX,IZAV)
      CALL IAT(IZ,N,1)
      IZAVC=0
      K=0
1000  K=K+1
      IF(K.NE.1)CALL CHEINO(N,IRP,IZAVC,IZ)
      IF(IZAVC.EQ.1) GOTO 2000
      JR=1
       DO 2 JK=1,N
        XS(JK)=XO(JK)
        IF(IX(JK).NE.1)GO TO 2
        XS(JK)=XS(JK)+IZ(JR)*IX(JK)
        JR=JR+1
2      CONTINUE
      CALL SRED(NP,N,XS,KIN,SMOS)
      IF(KIN.EQ.1) GO TO 1000
      KOT=0
      IF(SMOS.GE.SMO) GO TO 1000
       SMO=SMOS
      CALL COIMA1(XS,N,IXMAXT)
       RETURN
500   RETURN
      END
      SUBROUTINE GENPL(N,IR,IX,IZAV)
      DIMENSION IX(N)
      IZAV=0
      IOST=0
      I=N+1
100   I=I-1
      IF(I.GT.1)GO TO 1
      IZAV=2
      RETURN
1     IF(IX(I).EQ.0)GO TO 2
```

```
                IOST=IOST+1
                IX(I)=0
          IF(IOST.EQ.0)GO TO 100
2         IF(IX(I-1).EQ.1.OR.IOST.EQ.0)GO TO 100
             IX(I-1)=1
             IOST=IOST-1
          IF(IOST.GT.0)GO TO 3
             IF((I-1).EQ.IR)IZAV=1
             RETURN
3         N1=N+1
          DO 4 J=1,IOST
4         IX(N1-J)=1
          RETURN
          END
          SUBROUTINE BESTP(N,NP,IRKOOR,XO,XS,IX,IXMO,
      *                    SMO,SRE,KTF,KOT,IPROP)
          INTEGER XO(N),XS(N)
          DIMENSION IX(N),IXMO(N)
          SRE=SMO
          SUM=0
          KTF=0
          IPROP=0
          KOT=1
          DO 3 K=1,2
           KJ=1-2*(K-1)
           J=0
5          J=J+1
           IF(J.GT.N) GO TO 3
              DO 4 I=1,N
               XS(I)=XO(I)
               IF(I.EQ.J)XS(I)=XO(I)+KJ*IRKOOR
4             CONTINUE
           CALL SRED(NP,N,XS,KIN,SMOS)
           IF(KIN.EQ.0) GOTO 5
           KOT=0
           SUM=SUM+SMOS
           KTF=KTF+1
           IF(SMOS.GE.SMO) GOTO 5
           SMO=SMOS
           CALL COIMA1(XS,N,IXMO)
           IPROP=1
           GO TO 5
3         CONTINUE
          IF(KOT.EQ.1)RETURN
          SRE=SUM/KTF
          RETURN
          END
          SUBROUTINE CHEINO(NM,N,IZAV,IZ)
          DIMENSION IZ(NM)
          IZAV=0
          NSUM=0
          J=0
          DO 6 K=NM,1,-1
            IF(IZ(K).EQ.0)GO TO 6
            IF(J.EQ.1)GO TO 5
            IF(IZ(K).EQ.-1)GO TO 4
              IZ(K)=-1
              J=1
            GO TO 5
4           IZ(K)=1
```

```
5          IF(IZ(K).EQ.1)NSUM=NSUM+1
6          CONTINUE
        IF(NSUM.EQ.N)IZAV=1
        RETURN
        END
        SUBROUTINE FOPTVB(N,X,F0,KIN)
        INTEGER X(N)
        COMMON /FIKSN/NFKR,NFY,IPRF,IRAZR,NNMAX
        NFY=NFY+1
        KIN=0
        FS=0
        F0=0
        DO 1 I=1,N
        F=X(I)*X(I)
        F0=F0+F
        IF(IPRF.EQ.0)GO TO 1
        CALL URAN(FS)
        FS=FS*2-1
        F0=F0+FS
1       CONTINUE
        RETURN
        END
        SUBROUTINE URAN(FS)
        DOUBLE PRECISION XRAN,DM
        COMMON /BRAN/ XRAN
        DM=0.137438953472D+12
        XRAN=XRAN*3125.
        XRAN=XRAN-IDINT(XRAN/DM)*DM
        FS=XRAN/DM
        RETURN
        END
```

7.6. Вычисление статистической оценки экстремальных значений критерия

В инженерной практике автоматизированного выбора наилучших проектных решений все более широкое распространение находят методы многокритериальной оптимизации. При этом большинство применяемых при практических расчетах методов носит эвристический характер. Более того, при интенсивно развивающемся и популярном ныне интерактивном подходе к построению методов оптимизации, главным действующим лицом в диалоге с ЭВМ является лицо, принимающее решение (ЛПР). ЛПР ответственно за выход в область оптимальных решений, а ЭВМ играет роль инструмента для автоматического представления информации о путях выхода в эту область.

С точки зрения реализации поисковых методов оптимизации практически важным представляется построение методов оценки экстремальных значений векторного критерия эффективности $F(\cdot)$, заданного на некотором подмножестве m-мерного евклидова пространства $\mathbf{R}^m$. Информация об экстремальных значениях критерия может быть весьма эффективно использована при формировании тестов на оптимальность алгоритмов, построенных с помощью приближенных методов, а также в процессе работы самого алгоритма в качестве критерия окончания поиска, для обучения в ходе поиска, выделения перспективных зон в пространстве аргументов и т. д.

Конкретизируем задачу. Пусть $\mathbf{D}$ — некоторое непустое подмножество целочисленной решетки $\mathbf{Z}^m$ евклидова пространства $\mathbf{R}^m$, $F(\mathbf{x}) = (F_1(\mathbf{x}), ..., F_k(\mathbf{x}))^T$ — критерий—

функция цели, каждый компонент которой желательно минимизировать. Будем считать

$$F(\mathbf{x}) \leqslant F(\mathbf{y}), \text{ если } F_i(\mathbf{x}) \leqslant F_i(\mathbf{y});$$

$$F(\mathbf{x}) = F(\mathbf{y}), \text{ если } F_i(\mathbf{x}) = F_i(\mathbf{y});$$

$$F(\mathbf{x}) < F(\mathbf{y}), \text{ если } F_i(\mathbf{x}) \leqslant F_i(\mathbf{y}),$$

и существует j такое, что $F_j(\mathbf{x}) < F_j(\mathbf{y})$ (j, $i = 1, \ldots, k$).

Обозначим через $\Pi_{\mathbf{x}}$ множество Парето в пространстве аргументов:

$$\Pi_{\mathbf{x}} = \{\mathbf{x} : \mathbf{x} \in \mathbf{D}, \ \bar{\exists}\mathbf{y} \in \mathbf{D} : F(\mathbf{y}) < F(\mathbf{x})\},$$

через Π_F — множество Парето в пространстве значений $F(\mathbf{x})$:

$$\Pi_F : \{F : F = F(\mathbf{x}), \ \mathbf{x} \in \Pi_{\mathbf{x}}\},$$

Аналогично вводится и множество недоминируемости критериев. Множество недоминируемости в пространстве аргументов определяется соотношением

$$N_{\mathbf{x}} = \{\mathbf{x} : \mathbf{x} \in \mathbf{D} : \bar{\exists}\mathbf{y} \in \mathbf{D} : F(\mathbf{y}) < F(\mathbf{x})\},$$

в пространстве значений $F(\mathbf{x})$

$$N_F = \{F : F = F(\mathbf{x}), \ \mathbf{x} \in N_{\mathbf{x}}\},$$

В случае скалярной функции F алгоритм основывается на факте возможности аппроксимации функции распределения группового (случайного) минимума F некоторым предельным распределением $\Phi(y;\ \varepsilon,\ \sigma,\ \eta)$, зависящим от параметров ε, σ, η:

$$\Phi(y;\varepsilon,\sigma,\eta) = \begin{cases} 1 - \exp\left\{-\left(\dfrac{y-\varepsilon}{\sigma}\right)\eta\right\} & \text{при } y \geqslant \varepsilon, \\ 0 \text{ при } y < \varepsilon. \end{cases} \tag{7.39}$$

Здесь $-\infty < \varepsilon < \infty$, $\sigma > 0$, $\eta > 0$ — параметры минимума, масштаба и формы распределения $\Phi(y;\ \varepsilon,\ \sigma,\ \eta)$. Само распределение $\Phi(y;\ \varepsilon,\ \sigma,\ \eta)$ в теории вероятностей называют третьим предельным распределением или законом Вейбулла-Гнеденко. Распределение $\Phi(y;\ \varepsilon,\ \sigma,\ \eta)$ [79] достаточно хорошо аппроксимирует функцию распределения минимумов реализаций случайных величин, удовлетворяющих требованию ограниченности слева с вероятностью 1. Параметр ε и дает искомое значение $\min\limits_{\mathbf{x} \in \mathbf{D}} F(\mathbf{x})$.

Рассмотрим n выборок, каждая из которых размером в N членов, взятых из генеральной совокупности $\{F(\xi)\}$, где ξ — случайная величина, распределенная на $\mathbf{D}$ по некоторому закону (при отсутствии априорной информации можно считать ξ распределенной равновероятно на $\mathbf{D}$). Пусть F_i^* — случайный минимум F в i-й группе ($i = 1, \ldots, n$). Так как в большинстве практических задач мощность (т. е. количество элементов) множества $\mathbf{D}$ достаточно велика, то с вероятностью, близкой к 1, можно принять гипотезу о независимости величин $F_1^*, \ldots, F_n^*$.

Теперь задача определения параметра $\varepsilon = \min\limits_{x \in D} F(x)$ решается с помощью статистических методов оценки параметров распределения (7.39) по выборке $F_1^*, \ldots, F_n^*$. Для вычисления оценок векторного параметра $\theta = (\varepsilon, \sigma, \eta)$ используются следующие методы: метод моментов, метод максимального правдоподобия и метод байесовских оценок.

Метод моментов основан на вычислении первых трех выборочных моментов

$$\mu_1 = \frac{1}{n}\sum_{i=1}^{n} F_i^*, \quad \mu_2 = \frac{1}{n}\sum_{i=1}^{n} F_i^{*2}, \quad \mu_3 = \frac{1}{n}\sum_{i=1}^{n} F_i^{*3}.$$

Если теперь через $a_i(\varepsilon, \sigma, \eta)$ $(i = 1, 2, 3)$ обозначим соответствующие моменты распределения:

$$a_i(\varepsilon,\sigma,\eta) = \int_{-\infty}^{\infty} x^i d\Phi(x,\varepsilon,\sigma,\eta), \quad i = 1, 2, 3,$$

то оценку $\hat{\theta}_n = (\hat{\varepsilon}_n, \hat{\sigma}_n, \hat{\eta}_n)$ вектора параметров $\theta = (\varepsilon, \sigma, \eta)$ находим как решение системы уравнений $\mu_i = a_i(\hat{\varepsilon}_n, \hat{\sigma}_n, \hat{\eta}_n)$ $(i = 1, 2, 3)$, которая, в свою очередь, преобразуется в следующую систему:

$$\hat{\sigma}_n = b(\eta_n)(\mu_2 - \mu_1^2)^{1/2},$$

$$\hat{\varepsilon}_n = \mu_1 + [A(\hat{\eta}_n) - B(\hat{\eta}_n)](\mu_2 - \mu_1^2)^{1/2},$$

$$\sqrt{B_1} = \left[\Gamma\left(1 + \frac{3}{\hat{\eta}_n}\right) - 3\Gamma\left(1 + \frac{2}{\hat{\eta}_n}\right)\Gamma\left(1 + \frac{1}{\hat{\eta}_n}\right) + 2\Gamma^3\left(1 + \frac{1}{\hat{\eta}_n}\right)\right] B^3(\hat{\eta}_n), \tag{7.40}$$

где $\sqrt{B_1} = (\mu_3 - 3\mu_1\mu_2 + 4\mu_1^3)(\mu_2 - \mu_1^2)^{-3/2}$ — выборочная асимметрия;

$\Gamma(x) = \int_0^{\infty} y^{x-1} \exp(-y)\, dy \; (x > 0)$ — гамма-функция;

$$A(x) = \left[1 - \Gamma\left(1 + \frac{1}{x}\right)\right] B(x);$$

$$B(x) = \left[\Gamma\left(1 + \frac{2}{x}\right) - \Gamma^2\left(1 + \frac{1}{x}\right)\right]^{-1/2}$$

Оценки, полученные с помощью метода моментов, состоятельны, хотя и не имеют наименьшую из всех возможных дисперсию. Так как функции $a_i(\varepsilon, \sigma, \eta)$ $(i = 1, 2, 3)$ непрерывно зависят от своих аргументов, то с практической точки зрения для достижения требуемой точности при оцениваний по методу моментов достаточное число членов в выборке $F_1^*, \ldots, F_n^*$ определяется установлением величины

$\mu_i(\varepsilon, \sigma, \eta)$ в стационарные значения. В этом случае, используя неравенства Чебышева, получаем:

$$\mathbf{P}_{\theta_0}\{|\mu_j - a_j(\theta_0)| > \delta\} \leqslant a_{2j}(\theta_0)/(n\delta^2), \; j = 1, 2, 3, \tag{7.41}$$

где $\mathbf{P}_{\theta_0}\{\cdot\}$ — вероятностная мера, соответствующая θ_0 — «истинному» значению параметра θ. Таким образом, верхняя граница требуемого количества наблюдений определяется как

$$\max[a_{2j}(\theta_0)/(\delta_1 \delta^2)] + 1, \; j = 1, 2, 3,$$

где $[\cdot]$ — символ операции взятия целой части; δ_1 — заданный уровень значимости вероятности события $\{|\mu_j - a_j(\theta_0)| > \delta\}$.

Метод максимального правдоподобия приводит к решению уравнения

$$Y_n(\hat{\theta}_n) = \max_{\theta \in \bar{\Theta}} Y_n(\theta),$$

где $\bar{\Theta}$ — замыкание множества изменения параметров; $Y_n(\theta)$ — функция правдоподобия: $Y_n(\theta) = \prod_{i=1}^{n} f(F_i^*; \theta)$;

$$f(x, \theta) = \begin{cases} f(x, \varepsilon, \sigma, \eta) = \dfrac{n}{\sigma}\left(\dfrac{x-\varepsilon}{\sigma}\right)^{\eta-1} \exp\left\{-\left(\dfrac{x-\varepsilon}{\sigma}\right)^{\eta}\right\} \text{ при } x \geqslant \varepsilon, \\ 0 \text{ при } x < \varepsilon. \end{cases}$$

Так как $Y_n(\theta)$ — гладкая функция по θ, $\hat{\theta}_n \in \Theta$, то $\hat{\theta}_n$ есть также и решение относительно θ системы уравнений правдоподобия

$$\begin{aligned} &\frac{\partial}{\partial \varepsilon} \ln Y_n(\varepsilon, \sigma, \eta) = 0; \\ &\frac{\partial}{\partial \sigma} \ln Y_n(\varepsilon, \sigma, \eta) = 0; \\ &\frac{\partial}{\partial \eta} \ln Y_n(\varepsilon, \sigma, \eta) = 0 \end{aligned} \tag{7.42}$$

или, раскрывая смысл $Y_n(\theta)$, системы уравнений

$$\begin{aligned} &\sum_{i=1}^{n}\left(\frac{F_i^* - \varepsilon}{\sigma}\right)^{\eta} \ln \eta + 2n \ln \sigma - \frac{n}{\eta} = 0; \\ &\sum_{i=1}^{n} \eta (F_i^* - \varepsilon)^{\eta} \sigma^{-\eta-1} - \left(\frac{2n\eta + n}{\sigma}\right) = 0; \\ &\sum_{i=1}^{n}\left[\frac{n}{\sigma}\left(\frac{F_i^* - \varepsilon}{\sigma}\right)^{\eta-1} - \frac{1}{F_i^* - \varepsilon}\right] = 0. \end{aligned} \tag{7.43}$$

Для решения системы (7.43) можно использовать какой-нибудь стандартный метод нахождения корней нелинейной системы уравнений. Если система (7.43)

имеет несколько решений, то в качестве оценки для θ можно взять любое из них, максимизирующее $Y_n(\theta)$.

Оценки максимального правдоподобия состоятельны при самых общих предположениях [82], а их сходимость носит экспоненциальный характер:

$$\mathbf{P}_{\theta_0}\left\{\left\|\hat{\theta}_n-\theta_0\right\|_3>\gamma\right\}\leqslant\sum_{i=1}^{L}\exp\{-c_i n\},$$

где $\|\cdot\|$ — символ евклидовой метрики в пространстве $\mathbf{R}^3$, а величины $c_i>0$ $(i=1,\ldots,L)$ вычисляются заранее для определения требуемого числа наблюдений.

Метод обобщенных байесовских (относительно квадратичной функции потерь) *оценок* требует вычисления обобщенного апостериорного среднего для параметра ε:

$$\hat{\varepsilon}_n=\frac{\int\limits_{\Theta}\varepsilon\prod\limits_{i=1}^{n}f(F_i^*,\theta)\,w(\theta)\,d\theta}{\int\limits_{\Theta}\prod\limits_{i=1}^{n}f(F_i^*,\theta)\,w(\theta)\,d\theta}=$$

$$=\frac{\int\limits_{a_\sigma}^{b_\sigma}d\sigma\int\limits_{a_\eta}^{b_\eta}dn\int\limits_{a_\varepsilon}^{b_\varepsilon}\varepsilon\prod\limits_{i=1}^{n}f(F_i^*;\varepsilon,\sigma,\eta)\,w(\varepsilon,\sigma,\eta)\,d\varepsilon}{\int\limits_{a_\sigma}^{b_\sigma}d\sigma\int\limits_{a_\eta}^{b_\eta}d\eta\int\limits_{a_\varepsilon}^{b_\varepsilon}\prod\limits_{i=1}^{n}f(F_i^*;\varepsilon,\sigma,\eta)\,w(\varepsilon,\sigma,\eta)\,d\varepsilon},$$

где $[a_\varepsilon, b_\varepsilon]\times[a_\sigma, b_\sigma]\times[a_\eta, b_\eta]=\Theta$; $w(\theta)$ — обобщенная априорная плотность распределения θ; т. е. функция $w(\theta)$ непрерывна, положительна и удовлетворяет условию $\int\limits_{\Theta}w(\theta)\,d\theta<\infty$.

При решении конкретных задач чаще всего нет достоверной информации об априорном распределении параметров ε, σ, η. В этом случае оправдано использование обобщенных байесовских оценок $w(\theta)=1$ $(\theta\in\Theta)$. Обобщенные байесовские оценки асимптотически эффективны, состоятельны [82] и удобны при практическом применении, так как оценивание параметра ε не требует предварительной оценки параметров σ и η. (Скорость сходимости к истинным значениям дается в работе [83].)

После вычисления оценки по одному из методов следует проверить гипотезу о принадлежности выборки $F_1^*, \ldots, F_n^*$ трехпараметрическому семейству с помощью критерия соответствия Пирсона. Корректное применение этого критерия требует употребления асимптотически эффективных оценок параметров ε, σ, η, например оценок максимального правдоподобия. (Алгоритм и программа процедуры проверки гипотезы соответствия с помощью критерия Пирсона приведены в [83].)

Алгоритмы статистической оценки экстремальных значений целевой функции следующий:

1. Задаем значения n, N.

2. Многократная (в nN членов) групповая случайная выборка из множества **D**: $\xi_1, \ldots, \xi_{nN}$. Вычисление $F(\xi_1), \ldots, F(\xi_{nN})$.

3. Вычисление групповых минимумов $F_1^*, \ldots, F_n^*$.

4. Оценивание параметров третьего предельного распределения (Вейбулла—Гнеденко) по выборке $F_1^*, \ldots, F_n^*$ одним из методов: моментов максимального правдоподобия, обобщенных байесовских оценок.

5. Проверка гипотезы принадлежности функции распределения элементов выборки, сгенерированной в п. 3, трехпараметрическому семейству третьего предельного распределения с помощью критерия Пирсона. Если гипотеза с заданным уровнем значимости отвергается, увеличение n и N и переход к п. 2. Если гипотеза принимается, оценка ε_n объясняется оценкой $\min\limits_{\mathbf{x}\in\mathbf{D}} F(\mathbf{x})$.

Построим алгоритм оценки множества Парето Π_F. В основу предлагаемого алгоритма положены операция свертывания векторного критерия $F(\cdot)$ в семейство скалярных для получения паретовских решений и применения методов статистики экстремальных значений для оценки экстремума очередной сформированной свертки исходного векторного критерия. Наиболее часто в практических расчетах используются следующие формы сверток:

$$S(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i F_i(\mathbf{x}),\ \alpha_i \geqslant 0,\ i = 1, \ldots, k;\ \sum_{i=1}^{k} \alpha_i = 1,$$

$$S(\mathbf{x}) = F_j(\mathbf{x}),\ j \in \{1, \ldots, k\},$$

$$F_i(\mathbf{x}) \leqslant c_i,\ i \neq j;\ i = 1, \ldots, k.$$

При этом осуществляется преобразование многомерного пространства значений функции цели $F(\cdot)$, в общем случае нелинейный образ $L[F(\cdot)]$. Минимизируя (скалярную) функцию $S(\mathbf{x})$ при различных значениях параметров α_i (или c_i), можно получить все требуемые точки границы Парето.

Алгоритм вычисления минимума скалярного значения свертки состоит из процедуры групповой случайной выборки из множества $\mathbf{D}$ и статистической оценки по последовательности минимумов свертки $S(\mathbf{x})$ в каждой группе $\min\limits_{\mathbf{x}\in\mathbf{D}} S(\mathbf{x})$ абсолютного значения минимума свертки $S(\mathbf{x})$.

Основными требованиями, предъявляемыми к семейству сверток, являются требования неизбыточности и достаточности, состоящие в том, что операция свертывания не приводит ни к потере оптимальных, ни к появлению лишних решений. Из семейства сверток, которые удовлетворяют этим требованиям, выбрано линейное семейство

$$S(F; \alpha_1, \ldots, \alpha_k; \mathbf{x}) = \sum_{i=1}^{k} \alpha_i F_i(\mathbf{x}),\ i = 1, \ldots, k,$$

$$\sum_{i=1}^{k} \alpha_i = 1,\ \alpha_i \geqslant 0,\ i = 1, \ldots, k.$$

Задачу оценивания конечной аппроксимирующей сети множества Π_F сведем к

совокупности N задач статистической оценки величин $\min\limits_{x\in D} S(F; \alpha_1, \ldots, \alpha_k, x)$ для всех возможных наборов чисел $\alpha_1^j, \ldots, \alpha_k^j$ $(j=1, \ldots, N)$, удовлетворяющих соотношениям

$$\alpha_i^j \geqslant 0,\ i = 1, \ldots, k;\ j = 1, \ldots, N;$$

$$\sum_{i=1}^{n} \alpha_i^j = 1,\ j = 1, \ldots, N;$$

$$\left|\alpha_i^j - \alpha_i^{j+1}\right| \leqslant \Delta_i,\ i = 1, \ldots, k.$$

Информацию о необходимом уровне приращений Δ_i $(i=1, \ldots, k)$, обеспечивающем заданную точность аппроксимации $S(\Pi_F)$ — линейного образа множества Парето Π_F, дает следующая теорема.

Теорема 7.4. Пусть существуют и конечны величины

$$c_i = \max_{x\in D} |F_i(x)|,\ i = 1, \ldots, k.$$

Тогда разница между последовательными точками линейного образа

$$S(\Pi_F) : S_j = \min_{x\in D} S(F; \alpha_1^j, \ldots, \alpha_k^j; x),$$

$$S_{j+1} = \min_{x\in D} S(F; \alpha_1^{j+1}, \ldots, \alpha_k^{j+1}, x)$$

оценивается соотношениями $|S_j - S_{j+1}| \leqslant \mathbf{R}\times\mathbf{G}$, где $\mathbf{R} = \sum\limits_{i=1}^{n} \Delta_i^2$; $\mathbf{G} = \sum\limits_{i=1}^{n} c_i^2$.

Доказательство. Пусть

$$F(x) = \sum_{i=1}^{n} \alpha_i F_i(x),\ c_i = \max_{x\in D} |F_i(x)|.$$

Соседний функционал $\tilde{F}(x)$ определяется соотношением

$$\tilde{F}(x) = \sum_{i=1}^{n} (\alpha_i + \Delta_i) F_i(x).$$

Используя неравенство Коши—Буняковского, имеем

$$\tilde{F}(x) = \sum_{i=1}^{n} \alpha_i F_i(x) + \sum_{i=1}^{n} \Delta_i F_i(x) \leqslant$$
$$\leqslant \sum_{i=1}^{n} \alpha_i F_i(x) + \left[\sum_{i=1}^{n} \Delta_i^2\right]\left[\sum_{i=1}^{n} F_i^2(x)\right] \leqslant$$
$$\leqslant \sum_{i=1}^{n} \alpha_i F_i(x) + \left[\sum_{i=1}^{n} \Delta_i^2\right]\left[\sum_{i=1}^{n} c_i^2\right].$$

Вычислив минимум от обеих частей полученного неравенства, получим:

$$\min_{x \in D} \left\{ \sum_{i=1}^{n} (\alpha_i + \Delta_i) F_i(x) \right\} \leqslant \min_{x \in D} \left\{ \sum \alpha_i F_i(x) \right\} + \left[\sum_{i=1}^{n} \Delta_i^2 \right] \left[\sum_{i=1}^{n} c_i^2 \right].$$

Таким образом доказано неравенство

$$S_{j+1} - S_j \leqslant \left[\sum_{i=1}^{n} \Delta_i^2 \right] \left[\sum_{i=1}^{n} c_i^2 \right]. \tag{7.44}$$

Аналогично из равенства

$$\tilde{F}(x) - \sum_{i=1}^{n} \Delta_i F_i(x) = \sum_{i=1}^{n} (\alpha_i + \Delta_i) F_i(x) - \sum_{i=1}^{n} \Delta_i F_i(x) = \sum_{i=1}^{n} \alpha_i F_i(x) = F(x)$$

получим

$$\sum_{i=1}^{n} \alpha_i F_i(x) = F(x) - \sum_{i=1}^{n} \Delta_i F_i(x) \leqslant \tilde{F}(x) + \\ + \sum_{i=1}^{n} |-\Delta_i| |F_i(x)| \leqslant \tilde{F}(x) + \left[\sum_{i=1}^{n} \Delta_i^2 \right] \left[\sum_{i=1}^{n} c_i^2 \right]$$

или, переходя к вычислению минимума от обеих частей неравенства,

$$\min_{x \in D} \left\{ \sum_{i=1}^{n} (\alpha_i + \Delta_i) F_i(x) \right\} + \left[\sum_{i=1}^{n} \Delta_i^2 \right] \left[\sum_{i=1}^{n} c_i^2 \right] \geqslant \min_{x \in D} \left\{ \sum_{i=1}^{n} \alpha_i F_i(x) \right\},$$

т. е. неравенство

$$S_j - S_{j+1} \leqslant \left[\sum_{i=1}^{n} \Delta_i^2 \right] \left[\sum_{i=1}^{n} c_i^2 \right], \tag{7.45}$$

которое вместе с (7.44) и дает окончательный результат

$$|S_{j+1} - S_j| = \left| \min_{x \in D} \left\{ \sum_{i=1}^{n} (\alpha_i + \Delta_i) F_i(x) \right\} - \right. \\ \left. - \min_{x \in D} \left\{ \sum_{i=1}^{n} \alpha_i F_i(x) \right\} \right| \leqslant \left[\sum_{i=1}^{n} \Delta_i^2 \right] \left[\sum_{i=1}^{n} c_i^2 \right] = \mathbf{RG}. \tag{7.46}$$

Таким образом, при заданном уровне точности $\delta_3 = |S_{j+1} - S_j|$ порождения последовательности S_j ($j = 1, \ldots, N$) получаем следующую оценку частоты квантования $\mathbf{R}$:

$$|S_{i+1} - S_j| \leqslant \mathbf{RG} \leqslant \delta_3.$$

откуда

$$\mathbf{R} \leqslant \delta_3 / \mathbf{G} = \delta_3 / \left[\sum_{i=1}^{n} c_i^2 \right].$$

Оценим линейный образ границы Парето $S(\mathbf{\Pi}_F)$ с помощью свертки $S(F; \alpha_2, ..., \alpha_k; \mathbf{x}) = F_1(\mathbf{x})$ при наличии совокупности ограничений $F_i(\mathbf{x}) \leqslant c_i$ $(i=2, ..., k)$. В этом случае процедура оценивания конечной сети множества $S(\Pi_F)$ состоит из двух основных этапов:

1. Оценка $\max\limits_{\mathbf{x} \in \mathbf{D}} F_i(\mathbf{x})$, $\min\limits_{\mathbf{x} \in \mathbf{D}} F_i(\mathbf{x})$ $(i=2, ..., k)$ с помощью алгоритма статистической оценки скалярного критерия, что позволяет в дальнейшем задавать правые части ограничений $F_i(\mathbf{x}) \leqslant c_i$ лишь в пределах отрезков $[\min F_i(\mathbf{x}), \max F_i(\mathbf{x})]$ $(i=2, ..., k)$. Разумеется, при этом должно действовать условие абсолютной ограниченности функций $F_i(\mathbf{x})$ $(i=2, ..., k)$.

2. Приняв достаточно малый уровень приращений Δ_i $(i=2, ..., k)$, задачу оценивания конечной сети множества $S(\mathbf{\Pi}_F)$ сведем к совокупности N $(N = M_2 \times ... \times M_k)$ задач статистической оценки величин

$$\min_{\mathbf{x} \in \mathbf{D}_{l_2, ..., l_k}} F_1(\mathbf{x}),$$

где

$$\mathbf{D}_{l_2, ..., l_k} = \mathbf{D} \cap \left[\bigcap_{i=2}^{k} \left\{ F_i(\mathbf{x}) \leqslant \min_{\mathbf{x} \in \mathbf{D}} F_i(\mathbf{x}) + l_i d_i \right\} \right], \quad l_i = 1, ..., M_i;$$

$$M_i = \left[\left\{ \max_{\mathbf{x} \in \mathbf{D}} F_i(\mathbf{x}) - \min_{\mathbf{x} \in \mathbf{D}} F_i(\mathbf{x}) \right\} / d_i \right] + 1, \quad i = 2, ..., k.$$

Объединение соотношений позволяет определить необходимый объем вычислений для оценивания множества Парето $\mathbf{\Pi}_F$ с помощью методов статистики экстремальных значений.

ПРОГРАММА OPTI

Назначение: вычисление оценки минимального значения функции $\min\limits_{\mathbf{x} \in \mathbf{D}} F(\mathbf{x})$.

Область изменения допустимых значений параметров представляет собой декартово произведение $(1, ..., l_1)^T \otimes ... \otimes (1, ..., l_m)^T$, где l_i — предел изменения i-го $(i=1, ..., m)$ параметра.

Параметры;

FM — массив верхних пределов изменения оптимизируемых переменных (аргументов функции F), описывается с атрибутами BINARY FIXED;

K — размер массива FM;

M — число групп в случайной выборке;

F — процедура-функция, вычисляющая значения критериальной функции;

EST — минимальное значение критериальной функции.

Атрибуты параметров K, M, N, EST определяются по умолчанию.

Обращение: CALL OPTI(FM, K, M, N, EST, F);

Внутренние процедуры:

BISEK — нахождение корней нелинейной функции методом деления отрезка пополам;

FI — процедура-функция вычисления правой части уравнения (7.40);

GA — вычисления значений гамма-функции

$$\log\Gamma(x) \approx \left(x-\frac{1}{2}\right)\log x+\frac{1}{2}\log 2\pi - x+\frac{1}{12x}-\frac{1}{360x^3}+\frac{1}{1260x^5}-\frac{1}{1680x^7}$$

(разложение Эйлера-Маклорена). Это представление достаточно точно аппроксимирует значение $\Gamma(x)$ при $x>18$. При $x\leqslant 18$ по этой формуле производится вычисление $\log\Gamma(y)$, где $y=n+x$ — наименьшее значение, такое, что $y>18$. Наличие поправки $n(n$ — целое) компенсируется с помощью n-кратного применения формулы $\Gamma(x+1)=x\Gamma(x)$.

Входные и выходные параметры процедур BISEK, FI и GA задаются автоматически в самой программе OPTI. Кроме того, в программе OPTI используется процедура-генератор независимой выборки из равномерного распределения на отрезке [0,1]—RANDU. В качестве генератора RANDU можно использовать стандартный генератор, входящий в состав математического обеспечения ЕС ЭВМ, или генераторы, описанные в § 5.2, 5.3.

Основные этапы работы:

1. Организация случайной групповой выборки из множества поиска. Вычисление минимального члена в каждой группе. Накопление выборки групповых минимумов функции цели.

2. Нахождение первых трех центрированных выборочных моментов. Вычисление выборочной асимметрии.

3. Вызов процедуры отыскания корней нелинейного уравнения BISEK и нахождение оценки параметра EST с помощью решения уравнения (7.40).

4. Вычисление параметра EST — оценки минимального значения функции цели.

Пример. Решение задачи оценки с минимума функционала

$$F(x_1, x_2, x_3) = 0{,}1\left[(x_1-50)^2+(x_2-50)^2+(x_3-50)^2\right].$$

Исходные данные: K=3, N=M=200, FM(1)=FM(2)=FM(3)=100.

Получена оценка EST$=1{,}2\cdot 10^{-2}$ за 5,5 с на ЭВМ ЕС-1050. Точность полученного результата следует признать вполне удовлетворительной, особенно если учесть, что среднее значение функционала $F(\cdot)$ на рассматриваемой области его определения $F_{ср}\approx 750$. Таким образом, относительная (к величине $F_{ср}$) погрешность определения минимального значения F примерно $1{,}3\cdot 10^{-3}$ %.

```
OPTI: PROCEDURE(FM,K,M,N,EST,F);
      DECLARE FM(*) BINARY FIXED,
      P DEC, AS EXTERNAL, F ENTRY,
      XRANDO FLOAT(53)BIN EXT,
      RANDU ENTRY RETURNS(DEC);
      BEGIN;
      DECLARE VIB(M),ER(K) BINARY FIXED;
/* 1 */
      DO I=1 TO M;
      DO J=1 TO N;
      DO L=1 TO K;
      P=RANDU;
      FR(L)=P*FM(L);
      FR(L)=FR(L)+1;
      END;
      IF J=1 THEN
      DO;
      G1=F(FR); GO TO MET1;
      END;
      G2=E(FR);
      IF G1 > G2 THEN G1=G2;
MET1: END;
      VIB(I)=G1;
      END;
/* 2 */
      SR1=0;
      DO I=1 TO M;
      SR1=SR1+VIB(I)/M;
      END;
      SR2=0; SR3=0;
      DO I=1 TO M;
      SR2=SR2+(VIB(I)-SRI)**2/M;
      SR3=SR3+(VIB(I)-SRI)**3/M;
      END;
      SIG=SQRT(SR2);
      AS=SR3/SIG**3;
/* 3 */
      A=0.5; B=20;
      EP=0.001; EP1=0.001;
      ZZ=0;
      CALL BISEK(A,B,EP,EP1,ZZ);
      IF ZZ=0 THEN ZZ=2;
/* 4 */
      Z1=1+2/ZZ; Z2=1+1/ZZ;
      CALL GA(Z1,GA1);  CALL GA(Z2,GA2);
      BB=(GA1-GA2**2)**-0.5;
      AA=(1-GA2)*BB;
      EST=SR1+(AA-BB)*SIG;
BISEK: PROCEDURE(A,B,EPS,EPS1,X);
      DECLARE AS EXTERNAL;
      X=A; Y=FI(X); IF ABS(Y) <= EPS THEN GO TO FIN;
      X=B; Z=FI(X); IF ABS(Y) <= EPS THEN GO TO FIN;
      IF SIGN(Y)=SIGN(Z)
      THEN GO TO FIN;
MET:  X=(A+B)/2;
      Y=FI(X);  IF ABS(Y) <= EPS THEN GO TO FIN1;
      IF SIGN(Y)=SIGN(Z)
      THEN B=X; ELSE A=X;
      IF ABS(A-B) >= EPS1
      THEN GO TO MET;
FI:   PROCEDURE(Y1);
```

```
      DECLARE AS EXTERNAL;
      P1=1+3/Y1; P2=1+2/Y1;
      P3=1+1/Y1;
      CALL GA(P1,Q1);
      CALL GA(P2,Q2);
      CALL GA(P3,Q3);
      F2=(Q1-3*Q2*Q3+2*Q3**3)/(Q2-Q3**2)**1.5;
      F2=F2-AS;
      RETURN(F2);
      END;
FIN:  END BISEK;
GA:   PROCEDURE(X,G);
      T=1; XX=X;
      DO WHILE(X-18 <= 0) ;
      T=T*X; X=X+1;
      END;
      G=EXP((X-0.5)*LOG(X)+0.5*LOG(6.283)-
        LOG(T)-X+1/(12*X)-1/(360*X**3)+
        1/(1260*X**5)-1/(1680*X**7));
      X=XX;
      END;
END OPTI;

F:    PROG(FR);
      DCL FR(*) FIXED BINARY;
      P=(FR(1)-50)**2+(FR(2)-50)**2;
      P=P+(FR(3)-50)**2;
      P1=P*0.1;
      RETURN(P1);
      END F;

TEST: PROCEDURE OPTIONS(MAIN);
      DECLARE
      FM(3) BINARY FIXED,
      F ENTRY,
      XRANDO FLOAT(53)BIN EXT,
      RANDU ENTRY RETURNS(DEC);
      FM=100;
      K=3;
      N,M=200;
      CALL OPTI(FM,K,M,N,EST,F);
      PUT SKIP DATA(EST);
      END TEST;
```

7.7. Адаптивный выбор вариантов управления

При синтезе оптимального управления процессами функционирования одной из наиболее важных с практической точки зрения является задача оптимального выбора варианта управления. В качестве возможных вариантов управления могут фигурировать очередность выполнения целевых операций, режимы функционирования объекта и т. д., которые на практике наиболее часто встречаются в виде заданного конечного множества. Задачи такого типа вплотную примыкают к задачам оптимизации параметров систем и являются приложением дискретного программирования в управлении динамическими системами.

Пусть в дискретные моменты времени $t_i\,(i=0,\ \ldots,\ k)$ осуществляется выбор одного из заданных N возможных вариантов $\mathbf{u}(1),\ \ldots,\ \mathbf{u}(N)$ управления работой дина-

мической системы с вектором состояния $\mathbf{x}(t_i)$ $(i=0, \ldots, k)$ (вектор $\mathbf{x}(\cdot)$ размерности n, вектор $\mathbf{u}(\cdot)$ размерности m, n, $m \geqslant 1$). Эволюция вектора состояния $\mathbf{x}(\cdot)$ определяется оператором перехода Φ:

$$\mathbf{x}(t_{i+1}) = \Phi[\mathbf{x}(t_i), \mathbf{u}(t_i), \xi(t_i)], \tag{7.47}$$

где $\xi(t_i)$ — l-мерный вектор шумов, действующий на систему; $\mathbf{u}(t_i)$ — вариант управления динамической системой, выбранной на момент t_i.

На траекториях системы (7.47) задана некоторая (обычно скалярная) функция потерь $L[\mathbf{x}(t_i), \mathbf{u}(t_{l_i}), \xi(t_i), i=0, \ldots, k]$. Задача выбора вариантов управления состоит в определении вариантов управления $\mathbf{u}(t_{l_i})$ $(i=0, \ldots, k)$, обеспечивающих решение задачи оптимизации

$$\mathbf{M}\left\{ L\left[\mathbf{x}(t_i), \mathbf{u}\left(t_{l_i} \right), \xi(t_i), \quad i = 0, \ldots, k \right] \right\} \to \underset{\to \max}{}$$

на множестве допустимых значений

$$\left\{ \mathbf{x}(t_i), \mathbf{u}\left(t_{l_i} \right), \quad i = 0, \ldots, k \right\} \in D. \tag{7.48}$$

Данная постановка задачи достаточно общая и включает различные частные случаи задач выбора оптимальных вариантов управления. Рассмотрим математические модели адаптивного выбора вариантов управления динамической системой.

Пусть динамическая система описывается n-мерным $(n>0)$ вектором $\mathbf{x}(t, \mathbf{p})$, где $\mathbf{p}$ — m-мерный $(m>0)$ вектор параметров, описывающих структуру, состав и вариант функционирования исследуемой системы. Процесс функционирования протекает в интервале времени $[t_0, t_к]$ (возможно бесконечном); $-\infty < t_0 < t_к \leqslant +\infty$. На траекториях динамической системы $\mathbf{x}(t, \mathbf{p})$ $(t_0 \leqslant t \leqslant t_к)$ задан критерий эффективности системы $\mathrm{K}(\mathbf{p})$ (векторный, вообще говоря) размерности l $(l>0)$. Семейство $\mathbf{x}(t, \mathbf{p})$ является многомерным случайным процессом, заданным на некотором вероятностном пространстве $(\Omega, \mathbf{F}, \mathbf{P})$ с полным множеством событий $\{\omega \in \Omega\}$ (ω — элементарное событие), $\mathbf{F}$ — σ-алгебра, а $\mathbf{P}$ — вероятностная мера (вероятность), определенная на множествах из $\mathbf{F}$.

Булевой σ-алгеброй подмножеств множества Ω называется класс $\mathbf{A}$ подмножеств множества Ω, содержащий пустое множество и полное, замкнутый относительно операций счетного дополнения, объединения и пересечения. Пара $(\Omega, \mathbf{A})$ называется измеримым пространством. Следует отметить, что если обеспечена замкнутость относительно счетного дополнения, то требование замкнутости относительно счетного времени пересечения излишне. Чтобы класс $\mathbf{A}$ подмножеств множества Ω был σ-алгеброй, достаточно, чтобы он содержал пустое множество и полное и был замкнут относительно операций дополнения и объединения.

Определенный на отрезке $[t_0, t_к]$ случайный процесс $\mathbf{x}(t, \mathbf{p})$ и возрастающая последовательность σ-алгебр $\{\mathbf{A}_t, t \in [t_0, t_к]\}$ $(A_{t_1} \subset A_{t_2}$ при $t_1 < t_2$, $t_1, t_2 \in [t_0, t_к])$ называются адаптированными, если при каждом $t \in [t_0, t_к]$ процесс $\mathbf{x}(t, \mathbf{p})$ является $\mathbf{A}_t$-измеримым. При этом события из $\mathbf{A}$ являются предыдущими по отношению к моменту t.

Пусть $\{\mathbf{A}_t, t \in [t_0, t_к)\}$ — возрастающее семейство σ-подалгебр σ-алгебры $\mathbf{A}$. Отображение τ нулевого подмножества Ω_τ множества Ω в интервале $[t_0, t_к]$, если удовлетворяет условию $\{\tau \leqslant t\} \in \mathbf{A}_t$ при $t \in [t_0, t_к]$, — момент остановки. Каждому мо-

менту остановки сопоставляется σ-алгебра подмножеств $\mathbf{A}_\tau$ множества Ω_τ, удовлетворяющих условию $A \cap \{\tau \leqslant t\} \in \mathbf{A}_t$ при всех $t \in [t_0, t_к)$.

События из $\mathbf{A}_\tau$ являются предыдущими по отношению к τ. В дискретном по времени случае процесса $\mathbf{x}(t, \mathbf{p})_\tau$ (принимает лишь счетное или конечное множество значений) τ есть момент остановки относительно семейства $\{\mathbf{A}_t, t \in [t_0, t_к]\}$ тогда, и только тогда, когда $\{\tau = t\} \in \mathbf{A}_t$ при всех t, являющихся значениями τ. Если τ — некоторый момент остановки относительно семейства $\{\mathbf{A}_t, t \in [t_0, t_к]\}$, то моментом остановки будет и любое измеримое отображение $\xi(\tau)\colon [t_0, t_к] \to [t_0, t_к]$, удовлетворяющее условию $\xi(t) \geqslant t$ для всех $t \in [t_0, t_к]$.

Введем отношение порядка в множество возможных моментов, определенных относительно фиксированного возрастающего семейства σ-алгебр $\{\mathbf{A}_t\}$, а именно будем говорить, что $\tau_1 \leqslant \tau_2$ (τ_1 предшествует τ_2), если $\Omega_{\tau_2} \subset \Omega_{\tau_1}$, $\tau_1(\omega) \geqslant \tau_2(\omega)$, если $\omega \in \Omega_{\tau_2}$. С помощью отношения порядка моментов определяется верхняя и нижняя грани двух произвольных моментов остановки τ_1 и τ_2:

$$\bar{\tau} = \tau_1 \vee \tau_2 = \max[\tau_1(\omega), \tau_2(\omega)];$$

$$\underline{\tau} = \tau_1 \wedge \tau_2 = \begin{cases} \tau_1(\omega), \text{ если } \omega \in \Omega_{\tau_1} \cap \Omega^c_{\tau_2} = \Omega_{\tau_1} \setminus \Omega_{\tau_2}, \\ \min[\tau_1(\omega), \tau_2(\omega)], \text{ если } \omega \in \Omega_{\tau_1} \cap \Omega_{\tau_2}, \\ \tau_2(\omega), \text{ если } \omega \in \Omega^c_{\tau_1} \cap \Omega_{\tau_2} = \Omega_{\tau_2} \setminus \Omega_{\tau_1}, \end{cases}$$

причем область определения $\bar{\tau}$ есть $\Omega_{\bar{\tau}} = \Omega_{\tau_1} \cap \Omega_{\tau_2}$. Таким образом определенная функция $\bar{\tau}$ есть функция множества $\Omega_{\tau_1} \cup \Omega_{\tau_2}$.

Если $(\Omega, \mathbf{A}, \mathbf{P})$ — некоторое вероятностное пространство, τ — момент остановки, определенный на Ω_τ относительно возрастающего семейства $\{\mathbf{A}_t, t \in [t_0, t_к]\}$, и $\mathbf{x}(t, \mathbf{p})$ ($t \in [t_0, t_к]$) — случайный (многомерный) процесс, адаптированный к $\{\mathbf{A}_t, t \in [t_0, t_к)\}$, то отображение $\mathbf{x}(\tau(\omega), \mathbf{p})$ множества Ω_τ в множество $\mathbf{R}^n$ является $\mathbf{A}_\tau$-измеримым, если момент остановки τ принимает лишь счетное (и тем более конечное) множество различных значений [13].

Достаточно эффективно описать поведение многоэтапного функционирования динамических систем позволяет сформированная в теории случайных процессов методика марковских моментов. Рассмотрим задание многоэтапного функционирования динамической системы $\mathbf{x}(t, \mathbf{p})$ с помощью некоторой последовательности марковских моментов (моментов остановки) $\tau_1, \ldots, \tau_N$, где N — число отдельных этапов функционирования системы. Этапы определяются выполнением различных целевых операций, вследствие которых меняются условия функционирования системы. Чаще всего это происходит за счет того, что рассматриваемая система функционирует в различных областях $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_N$ фазового пространства. Марковские случайные моменты $\tau_1, \ldots, \tau_N$ определяются моментами времени первого попадания в множества $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_N$ соответственно. Множества $\mathbf{B}_i$ ($i = 1, \ldots, N$) попарно не пересекаются: $\mathbf{B}_j \cap \mathbf{B}_k = 0$ ($j, k = 1, \ldots, N$).

Предположим, что вначале семейство множеств $\mathbf{B}_1, \ldots, \mathbf{B}_N$ удовлетворяет свойству невозвратности по времени функционирования:

$$\mathbf{P}\{\mathbf{x}(t_1, \mathbf{p}) \in \mathbf{B}_l \,|\, \mathbf{x}(t_2, \mathbf{p}) \in \mathbf{B}_k\} > 0, \qquad l < k\ (l, k = 1, \ldots, N);\ t_1, t_2 \in [t_0, t_к],\ t_1 < t_2.$$

Здесь $\mathbf{P}\{\cdot\}$ — вероятность, заданная на фазовом пространстве траекторий рассмат-

риваемой динамической системы. Сформулированное условие означает, что совокупность множеств $\mathbf{B}_1$, ..., $\mathbf{B}_N$ определяет «поступательное» функционирование системы. При этом возвращение на предыдущие этапы функционирования невозможно, возможны переходы на этапы функционирования, имеющие несоседние индексы (отличающиеся более чем на единицу):

$$\mathbf{P}\{\mathbf{x}(t_1, \mathbf{p}) \in \mathbf{B}_l \,|\, \mathbf{x}(t_2, \mathbf{p}) \in \mathbf{B}_k\} > 0,$$
$$|l-k| > 1,\ l \geqslant k;\ t_1, t_2 \in [t_0, t_к],\ t_1 < t_2. \tag{7.49}$$

Последнее соотношение означает наличие нескольких альтернатив в ходе функционирования системы, возможность невыполнения некоторых этапов функционирования, частичное решение целевой задачи и т. д.

На множестве $\bar{\mathbf{X}}$ возможных траекторий системы $\mathbf{x}(t, \mathbf{p})$ рассматриваются события $\mathbf{A}_l = \bigcup_l \{\mathbf{x}(t, \mathbf{p}) \in \mathbf{B}_l\}$. В свою очередь, на множестве $\mathbf{A}_l$ определена функция $\tau_l = \tau_l(\omega) = \inf\{t : t \in [t_0, t_к] \cap \mathbf{x}(t, \mathbf{p}) \in \mathbf{B}_l\}$ $(l = 1, ..., N)$. Таким образом, τ_l есть первый момент, для которого $\mathbf{x}(t, \mathbf{p}) \in \mathbf{B}_l$, т. е. первый момент времени попадания в $\mathbf{B}_l$ (момент начала функционирования системы в области $\mathbf{B}_l$).

Определим последовательность марковских моментов τ_1, ..., τ_N начала выполнения l-х $(l = 1, ..., N)$ этапов функционирования системы. Если обозначить через τ_1 детермированный момент начала функционирования системы $(\tau = t_0)$, а через τ_{N+1} — детерминированный конечный $(\tau_{N+1} = t_к)$, то длительность функционирования задается с помощью соотношений $\Delta\tau_l = \tau_{l+1} - \tau_l$ $(l = 1, ..., N)$, которые определяют время нахождения системы в соответствующей области $\mathbf{B}_l$.

Рассмотрим более общий случай, когда система может возвратиться из состояния $\mathbf{B}_l$ в состояние (область) $\mathbf{B}_k$ пространства $\bar{\mathbf{X}}$, где $k < l$ $(k, l = 1, ..., N)$. Для этого введем расширение множества траекторий (реализаций) рассматриваемой динамической системы в виде $\bar{\bar{\mathbf{X}}} = [t_0, t_к] \oplus \bar{\mathbf{X}}$, где $\oplus$ — символ декартова произведения двух множеств. В этом случае условие (7.49) будет выполняться из-за непересекаемости множеств $[\underline{t}, \overline{t}] \otimes \mathbf{B}_l$ $(l = 1, ..., N)$, где $[\underline{t}, \overline{t}]$ — некоторый временной интервал функционирования системы: $[t, \bar{t}] \subseteq [t_0, t_к]$.

Будем считать, что на каждом этапе $\mathbf{B}_1$, ..., $\mathbf{B}_N$ $(N > 0)$ функционирования заданная совокупность критериев $\mathbf{K}_l(\mathbf{p})$ $(l = 1, ..., N)$ отражает цели функционирования каждого из M модулей $(M > 0)$, входящих в систему. Пусть критерии $\mathbf{K}_l(\mathbf{p}) = \{\mathbf{K}_{i_1}(\mathbf{p}), ..., \mathbf{K}_{i_m}(\mathbf{p})\}$ определяют цель функционирования на l-м этапе. В реальных задачах управления число целей функционирования для каждого модуля конечное. Например, для l-го этапа:

для 1-го модуля $\{1, ..., m_1\}$,

для 2-го модуля $\{m_1 + 1, ..., m_2\}$,

. . . . .

для M-го модуля $\{m_{M-1} + 1, ..., m_M\}$,

где $m_1, m_2, ..., m_M > 0$. В этом случае критерий $\mathbf{K}_{ij}(\mathbf{p})$ определяет качество (например, вероятность) выполнения j-м $(j = 1, ..., M)$ модулем на i-м этапе $(i = 1, ..., N)$ выбранной цели функционирования из набора $\{m_{l-1} + 1, ..., m_j\}$.

Таким образом, с учетом существования общего критерия эффективности $\bar{\mathbf{K}}_l(\mathbf{p})$, отражающего качество выполнения целевого задания всем комплексом модулей на данном l-м этапе, цель функционирования отдельного модуля может быть выбрана (оптимизирована), а все ее альтернативы включены в параметрическое

множество $\{\mathbf{p} \in \mathbf{P}\}$. С учетом этого структура параметрического множества имеет следующий иерархический вид:

$$\mathbf{p} \in \mathbf{P} \Rightarrow \mathbf{p} = (\bar{\mathbf{p}}, \bar{\bar{\mathbf{p}}}),$$

где $\bar{\mathbf{p}}$ — первый компонент параметра $\mathbf{p}$, определяющий выбор цели функционирования для j-го ($j = 1, \ldots, M$) модуля:

$$\mathbf{p} = \{j, \mathbf{K}_j\}, \quad i = 1, \ldots, \mathbf{M};$$

$$\mathbf{K}_j \in \{m_{j-1} + 1, \ldots, m_j\}, \quad j = 2, \ldots, \mathbf{M};$$

$$\mathbf{K}_j \in \{1, \ldots, m_j\}, \quad j = 1.$$

Для объединения последних двух включений в одно с индексом $j = 1, \ldots, M$ удобно положить, что $m_0 = 1$. Возможные значения компонента $\bar{\bar{\mathbf{p}}}$ вектора $\mathbf{p} = (\bar{\mathbf{p}}, \bar{\bar{\mathbf{p}}})$ определяют альтернативный выбор параметров реализации набора целей функционирования отдельных модулей. Если первый компонент (цели функционирования) выбран, то вторая является единственно варьируемой. Предположим, что $\bar{\bar{\mathbf{p}}}$ — m-мерный вектор, принадлежащий соответствующему евклидову пространству $\mathbf{R}^{m'}$. Наиболее важен (в приложениях в первую очередь) с учетом реализации алгоритмов управления на ЭВМ случай, когда $\bar{\bar{\mathbf{p}}}$ — дискретное множество.

При решении задачи выбора вариантов управления сложными системами необходимо учитывать следующие особенности:

сложность структуры и режимов функционирования;

сложность и изменение целевых задач в процессе функционирования;

многофакторный и заранее непредсказуемый характер условий функционирования.

Все это приводит к необходимости привлечения адаптивных систем управления. Внедрение адаптивных систем управления связано в первую очередь с развитием современных средств вычислительной техники (микроЭВМ и микропроцессоров), позволяющих реализовать управление в реальном масштабе времени. В современных системах управления стремятся рационально сочетать принципы программного (автономного) управления и управления с обратной связью. Адаптация понимается как способность системы гибко реагировать на факторы, сопутствующие реальному процессу функционирования. Такая трактовка потребовала формулирования обобщенного принципа адаптивного управления — принципа адаптации по прогнозирующей оценке эффективности, который лежит в основе методики выбора вариантов управления. Суть этого принципа состоит в использовании модели μ, прогнозирующей оценку эффективности функционирования системы:

$$\mu = \bigcup_{l=1}^{N} \mu_l \tag{7.50}$$

где μ_l — частная (этапная) модель эффективности функционирования на отдельном l-м этапе целевого применения системы. На различных траекториях модели μ считаем заданным некоторый функционал—критерий эффективности $\mathbf{F}$ (обычно это, например, вероятность выполнения целевой задачи). Основным требованием к

структуре модели является возможность вычисления для каждого момента времени t математического ожидания критерия эффективности окончания процесса функционирования $\bar{F}_t$ при условии имеющейся к моменту времени реализации этого процесса $\mathbf{x}[\tau,\ t_0 \leqslant \tau \leqslant t_к]$:

$$\bar{F}_t[\,\mathbf{x}(\tau, t_0 \leqslant \tau \leqslant t)] = \mathbf{M}\{F[\,\mathbf{x}(\tau_1, t \leqslant \tau_1 \leqslant t_к)]\,|\mathbf{x}(\tau_2, t_0 \leqslant \tau_2 \leqslant t)\}, \qquad (7.51)$$

где $\mathbf{x}(t)$ — вектор состояния системы.

Если обозначить обобщенный вектор измерения состояния системы на момент времени t через $\mathbf{z}(t)$, то адаптация может быть реализована с помощью следующей схемы (рис. 7.2). Предлагаемая схема синтеза адаптации процесса функционирования системы содержит три основных элемента:

модель, прогнозирующую оценку эффективности функционирования;

процедуру оценки неизвестных параметров функционирования;

процедуру оптимизации параметров и режимов функционирования по имеющейся прогнозирующей модели.

Как видно из (7.51), для проведения процедур оптимизации в ходе реального функционирования системы требуется применение методов, реализуемых в виде программного обеспечения на используемых микроЭВМ и эффективно работающих в условиях жесткого дефицита времени, отведенного для решения целевой задачи. Таким образом, для гибкого управления (с адаптацией по последовательности выполняемых этапов целевых операций или к непредсказуемым факторам функционирования) требуется проведение исследований по построению эффективных методов и алгоритмов:

идентификации неизвестных параметров с одновременным определением необходимых объемов статистической информации, обеспечивающих нужную точность оценки параметров;

выбора требуемого (оптимального) варианта продолжения процесса функционирования, адаптированного к структуре оптимизируемого критерия и условиям дефицита времени на принятие решения.

Совокупность моделей μ, μ_i $(i=1, \ldots, N)$ может быть сформирована с помощью различных принципов — аналитического и имитационного моделирования [71].

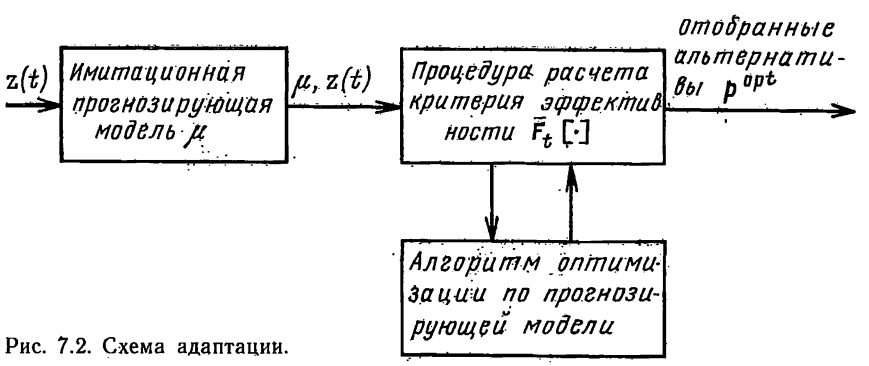


Рис. 7.2. Схема адаптации.

Для выбора вариантов управления предлагается программа АОДИМ (см. § 7.6), ориентированная на оптимизацию параметров моделей оценки эффективности. При этом пользователь должен сформировать процедуру расчета критерия эффективности $\bar{F}_l$ (7.51).

Приложение 1.

Некоторые элементы программирования

Сведения об операционных системах ЕС ЭВМ

Пакет программ, разработанных в настоящей книге, предназначен для использования на ЕС ЭВМ. Здесь приводится необходимый минимум сведений об операционных системах, которыми оснащены ЕС ЭВМ (см. [85, 87]).

Под *операционной системой* (или монитором) понимается набор программ, которые организуют непрерывную работу ЭВМ по обработке потока программ. Наиболее распространенными операционными системами являются ОС ЕС и ДОС ЕС. Операционные системы предназначены для планирования работы вычислительной системы, повышения ее пропускной способности, обеспечения ее универсальности, а также для работы программиста и оператора с вычислительной системой.

Операционные системы состоят из управляющих и обрабатывающих программ. *Управляющие программы* определяют последовательность выполнения обрабатывающих программ и обеспечивают необходимый для этого сервис. *Обрабатывающие программы* состоят из системных программ, которые предназначены для сокращения усилий и времени, затрачиваемых на разработку, подготовку и выполнение программ пользователя. В число системных обрабатывающих программ входят трансляторы, сервисные программы, утилиты и программы сортировки.

В *сервисные программы* входят редактор связей, предназначенный для формирования одной программы из нескольких оттранслированных, загрузчик, обеспечивающий перемещение (загрузку) отредактированной программы в оперативную память ЭВМ, а также средства отладки программ.

Утилиты предназначены для работы с библиотеками программ и для передачи информации между отдельными устройствами ЭВМ.

Программы сортировки служат для работы с наборами данных: упорядочивания, перебора и др.

При работе с операционной системой разработанная программа должна пройти на ЭВМ следующие стадии: трансляцию (или ассемблирование, если она написана на Ассемблере), редактирование и непосредственно выполнение.

Программа, написанная на каком-либо языке программирования, называется *исходным модулем*. Обычно в процессе разработки программы последняя разбивается на отдельные части (причем они могут быть написаны на различных языках). С помощью процедуры трансляции исходные модули преобразуются в *объектные модули*. Чтобы можно было загрузить их в оперативную память ЭВМ и выполнить, объектные модули должны пройти редактирование связей. Модуль, полученный в результате редактирования, называется *загрузочным*. Загрузочный модуль уже пригоден для размещения в оперативную память ЭВМ.

Исходный модуль может быть размещен на разных носителях: магнитной ленте, магнитном диске, может быть введен с дисплея. Для его прохождения в среде операционной системы ЭВМ требуется информация о том, какие ресурсы необходимы транслятору, редактору и самой программе для ее успешного выполнения, а также где и в каком виде расположены исходная информация и исходные данные программы. Вся эта информация поставляется системе с помощью языка управления заданиями. Следовательно, необходимо подготовить программу, написанную на одном из языков, а также исходные данные для нее и составить операторы управления заданиями.

Управление заданиями в операционной системе

На практике операторы языка управления заданиями чаще всего вводятся с экрана дисплея. Существует 10 типов управляющих операторов ОС, которые позволяют регулировать выполнение задания, описывать необходимые для выполнения данные, осуществлять связь между ЭВМ и человеком-оператором: JOB — оператор задания; EXEC — исполнительный оператор; ДД — оператор описания данных; /*—разделительный оператор; // — оператор конца задания; PROC — оператор процедуры; PEND — оператор конца процедуры; командный оператор, оператор комментариев.

Описание каждой выполняемой программы, а также необходимых для ее выполнения данных составляет пункт задания. Несколько пунктов задания образуют само задание, а несколько заданий — пакет заданий. Схема построения пакета заданий приведена на рис. П.1.

Оператор задания JOB имеет следующий формат:

// имя задания JOB, учетная информация, имя программиста, MSGLEVEL= =..., COND=..., TIME=...

Оператор JOB является первым оператором задания. В поле, отведенном для имени задания, содержится некоторое имя (обязательно начинающееся с буквы). Все параметры, следующие за словом JOB, являются необязательными. Параметры

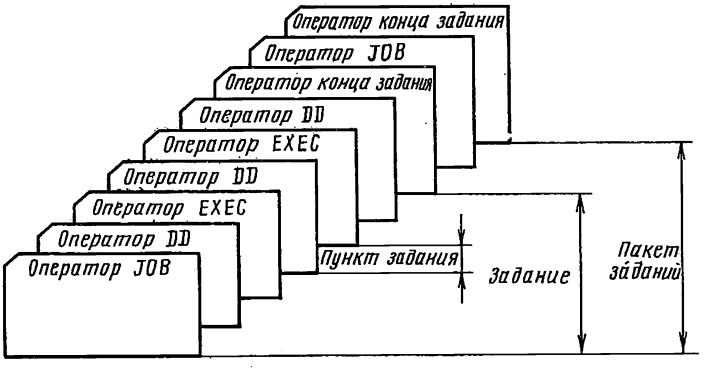


Рис. П. 1. Схема построения пакета заданий

«учетная информация» и «имя программиста» используются для автоматического сбора информации о работах, ведущихся на ЭВМ.

Параметр MSGLEVEL дает указание, какую информацию о задании желательно получить на выходном устройстве. Формат параметра MSGLEVEL следующий:

MSGLEVEL=(операторы, сообщения)

Параметр «операторы» принимает три значения: 0, если выводится только оператор JOB; 1, если выводятся все входные управляющие операторы задания, операторы каталогизированной процедуры и внутреннее представление операторов, и, наконец, 2, если выводятся только входные управляющие операторы задания. Параметр «сообщения» принимает следующие значения: 0, если сообщения о распределении (завершении) не выводятся при успешном выполнении задания (при аварийном завершении все сообщения появятся в качестве вывода), и 1, если выводятся все сообщения о распределении (окончании) независимо от успешности выполнения задания.

Параметр COND определяет условие, при котором прекращается выполнение задания. По мере завершения каждого пункта задания регистрируется некоторое число в качестве кода возврата. Производится серия проверок имеющихся кодов, и, если хотя бы одна из проверок дает положительный результат, выполнение задания прекращается. Формат параметра COND следующий:

COND=((код, оператор), ...)

Здесь «код» — некоторое число от 0 до 4095, с которым сравнивается код возврата; «оператор», представляющий собой операцию сравнения, может принимать следующие значения:

GT (больше),	LT (меньше),
GE (больше или равно),	LE (меньше или равно),
EQ (равно),	NE (не равно).

Параметр COND позволяет поставить выполнение очередного шага задания в зависимость от итогов выполнения предыдущих. Например, если задание включает трансляцию, редактирование и выполнение некоторой программы, то в случае обнаружения грубых ошибок при трансляции операции редактирования и выполнения становятся невозможными. В этом случае параметр COND выглядит следующим образом:

COND=(9, LE)

Так как при обнаружении грубых ошибок транслятор с языка ПЛ/1 выдает код возврата, больший 8, то с помощью логического условия, записанного с помощью такого оператора COND, операционная система завершит обработку данной программы.

Параметр TIME задает максимальное время использования центрального процессора заданием. Если задание использует центральный процессор дольше, чем указано в TIME, то это приводит к аварийному окончанию всего задания. Параметр имеет следующий формат:

TIME=(минуты, секунды)

Оператор идентификации пункта задания EXEC служит для указания программы или процедуры, которые должны выполняться в данном пункте задания. Программа и процедура указываются с помощью взаимоисключающих друг друга

параметров PGM и PROC соответственно. Исполнительный оператор EXEC имеет следующий формат:

// имя шага задания EXEC PGM=
имя программы, ...

Здесь параметр «имя шага задания» идентифицирует данный шаг задания, а параметр «имя программы» указывает имя раздела библиотеки, в котором хранится выполняемая на данном шаге программа. Если на шаге задания используется процедура, то вместо имени программы указывается имя этой процедуры (процедура представляет собой один или несколько шагов задания, идентифицированных операторами EXEC) путем задания после оператора EXEC

PROC=имя процедуры

либо самого имени. Многоточием обозначены остальные возможные параметры оператора, которые аналогичны параметрам оператора JOB. При работе с пакетом программ необходимо употреблять следующий формат оператора EXEC:

// имя шага задания EXEC PL1FCLG

Здесь PL1LFCLG — каталогизированная процедура трансляции, редактирования и выполнения исходного модуля, записанного на языке ПЛ/1. Если достаточно провести только трансляцию или трансляцию и редактирование, необходимо вызывать процедуры PL1LFC или PL1LFCL соответственно.

Оператор описания данных DD предназначен для идентификации наборов данных, используемых или создаваемых программами пункта задания, следует непосредственно за оператором EXEC. Оператор DD имеет следующий формат:

// имя DD DD — операнды

Здесь параметр «имя DD» — это имя из блока управления данными в программе. Если в проекте задания содержится обращение к каталогизированной процедуре, то операторы DD этого пункта задания используются для замещения или добавления к параметрам каталогизированной процедуры новых операторов DD. В этом случае в поле имени добавляемого оператора DD требуется записать «имя шага процедуры, имя DD», где «имя шага процедуры» указывает, куда нужно добавить оператор DD, «имя DD» — имя добавляемого оператора. Первым операндом обычно является символ «*» или слово DATA. Это означает, что набор данных непосредственно следует за оператором DD во входном потоке данных.

Для шага задания, состоящего в вызове процедуры трансляции, редактирования и выполнения исходного модуля, записанного на языке ПЛ/1 (т. е. процедуры PL1LCLG), этот модуль идентифицируется оператором//PL1L.SYSINDD*, за которым следует исходная программа. Если на шаге счета по отредактированной и загруженной программе необходимы исходные данные, то употребляется оператор DD следующего вида://CO.SYSINDD*, а за ним помещаются исходные данные. Набор данных должен заканчиваться оператором ограничения /*. Эти символы помещаются в первые две колонки оператора, в остальных колонках — пробелы (либо комментарии).

Оператор//служит для указания операционной системе конца задания. После этого оператора система ищет во входном потоке следующий оператор идентификации задания. Оператор конца задания содержит символ//в первых двух колонках, а во всех остальных — пробелы.

В целом, суммируя вышеизложенное, можно составить следующий набор управляющих карт ОС, который предназначен для трансляции, редактирования и выполнения некоторой программы, разработанной на языке ПЛ/1:

```
//RAB JOB
//ST EXEC PL1LFCLG
/PL1L. SYSIN DD*
  исходный модуль
/*
//GO. SYSIN DD*
  исходные данные для счета
//
```

Элементы программирования на ПЛ/1

Язык программирования ПЛ/1 в отличие от языков Фортран, Кобол, РПГ, ориентированных на определенный класс задач,—язык универсальный, в одинаковой степени удобный для решения как научных и инженерных задач, так и экономических. Одним из самых ценных его свойств является то, что язык можно использовать для решения задач в реальном времени, а также для создания систем программирования.

Программа на ПЛ/1 состоит из одного или нескольких блоков, называемых процедурными блоками или просто процедурами. *Процедура* — это или основная программа, или подпрограмма, или функция. Процедура может транслироваться отдельно от вызывающей процедуры или может быть вложена в вызывающую процедуру и транслироваться вместе. Каждая процедура может содержать объявление имен (т. е. некоторые описания переменных, наборов данных и т. д.), при этом определяется область их действия и происходит распределение памяти для них. Кроме процедурных блоков в ПЛ/1 существуют обычные блоки (блоки BEGIN), обязательно вкладываемые в некоторый процедурный блок, который для них является внешним.

В языке программирования ПЛ/1 имеются разнообразные средства для описания и обработки различных типов данных. В программах на ПЛ/1 могут быть использованы арифметические данные, строковые и данные управления программой. *Арифметические данные* могут быть представлены несколькими способами: они могут быть десятичными, двоичными, с фиксированной или плавающей точкой. *Строковые данные* — это последовательность знаков или бит (строки знаков или строки бит). *Данные управления* программой — это данные типа метки или указателя. Обрабатываемые в программе ПЛ/1 данные могут участвовать в арифметических, логических операциях, а также в операциях сцепления и сравнения, допускаются и смешанные типы операций. Для определения порядка выполнения операций вводится иерархия их приоритетов.

Переменные могут объединяться в массивы или структуры. *Массив* — это совокупность элементов с одинаковыми свойствами (типом, разрядностью). *Структура* — это специальным образом организованная совокупность переменных разного свойства. Память переменным в ЭВМ может отводиться двумя основными способами: статически перед началом выполнения программы или динамически во время выполнения.

Язык ПЛ/1 имеет весьма совершенную систему управления операциями ввода-вывода данных. Средства ввода-вывода обеспечивают два типа передачи данных: потоком и записями.

В языке имеются разнообразные средства, которые позволяют управлять прерываниями программы, устанавливать нужную программисту реакцию на прерывание, а также средства, позволяющие имитировать возникновение различных ситуаций, которые приводят к прерыванию программы.

Для выполнения часто используемых вычислений, различных операций по обработке данных в ПЛ/1 имеется набор специальных (встроенных) функций. Для вызова функции программисту достаточно лишь указать ее имя со списком аргументов, которые она требует. Поскольку при работе с пакетом программ читателю потребуются некоторые навыки по описанию и вводу-выводу данных, остановимся на способах работы с проблемными данными в языке ПЛ/1.

Характеристики объектов, которые представлены в программе некоторыми именами (такие, как основание счисления, способ представления данных и др.), и характеристики самих имен (такие, как область действия имени, класс памяти и т. д.) вместе составляют набор атрибутов, который связывают со всяким изменением объекта в программе. Все атрибуты данных объявляются в операторе DECLARE, или контекстуально, или неявно.

Для спецификации арифметических данных в ПЛ/1 применяют следующие описания: DECIMAL (десятичный) или BINARY (двоичный), при отсутствии указаний предполагается DECIMAL; FIXED (фиксированный) или FLOAT (плавающий) при отсутствии указаний предполагается FLOAT; REAL (вещественный) или COMPLEX (комплексный) при отсутствии указания предполагается REAL.

Если атрибуты не указаны, то предполагается, что имена переменных, начинающихся с букв I, J, K, L, M, N, соответствуют типу FIXED BINARY, а имена переменных, начинающихся с любой буквы,— типу DECIMAL FLOAT.

Для спецификации строковых данных применяются такие описания, как BIT (битовый) или CHARACTER (символьный).

Разрядность указывается в виде (w, d) для данных с фиксированной точкой и в виде (w) для данных с плавающей точкой. Здесь w — общее число цифр (максимальное), допустимое для переменной, т. е. число значащих цифр в случае числа с плавающей точкой, а d — число дробных разрядов в десятичном или двоичном числе. Если значение не задано, то предполагается, что d=0. Комплексное число представляется как пара вещественных чисел, имеющих одинаковые описания.

В языке ПЛ/1 имеются следующие ограничения максимального размера чисел разного типа:

	По умолчанию	Максимальные
DECIMAL FIXED	(5,0)	(15, m)
BINARY FIXED	(15,0)	(31, m)
DECIMAL FLOAT	(6)	(16)
BINARY FLOAT	(21)	(53)
CHARACTER	(1)	(21767)
BIT	(1)	(32767)

Здесь параметр «имя» представляет собой имя описываемой переменной, используемой в программе. Оператор имеет следующий вид:

```
DECLARE
        имя описатель;
        имя описатель;
        . . . . . .
        имя описатель;
```

где параметр «имя» представляет собой имя описываемой переменной, а параметр «описатель» идентифицирует ее свойства с помощью атрибутов, описанных выше.

К одним из самых важных в языке ПЛ/1 и используемых в данном пакете программ относятся атрибуты INTERNAL и EXTERNAL. Первый указывает, что соответствующее ему имя известно только в блоке, содержащем объявление, и во вложенных в него блоках подразумевается по умолчанию. Если задан атрибут EXTERNAL, то это означает, что имя может быть известно в других блоках, содержащих объявление того же имени с атрибутом EXTERNAL. Максимальная длина любого внешнего имени равна шести знакам.

Очень удобным в языке ПЛ/1 является возможность одним оператором DECLARE специфицировать любое количество переменных. Например, оператором

```
DECLARE
        CHIN(2,3) FIXED,
        BOL(5,6) FLOAT,
        X CHARACTER(4);
```

специфицируются одновременно массивы CHIN и BOL, а также строка знаков X.

При объявлении имени переменной можно задать ее значение, которое будет присвоено в момент загрузки объектной программы в память. Для этого в операторе DECLARE записывается оператор INITIAL в следующем виде:

INITIAL (значение) для скалярной величины,

INITIAL (1-е значение, ..., h-е значение) для массива.

Например, если массиву A(2) требуется присвоить значения $A(1) = \varnothing$, $A(2) = 2.5$, то это можно сделать следующим образом:

```
DECLARE
A(2) INITIAL(2.5)
```

В языке ПЛ/1 широко используются процедуры, которые позволяют многократно выполнять одни и те же группы вычислений или преобразований данных с различными входными параметрами. Для процедур используется оператор PROCEDURE, который имеет следующие форматы:

```
для главной процедуры
    имя входа: PROCEDURE OPTIONS(MAIN);
для процедуры-подпрограммы
    имя входа: PROCEDURE(1-й параметр, ..., h-й параметр)
для процедуры-функции
    имя входа: PROCEDURE(1-й параметр, ..., h-й параметр)
                RETURNS (атрибуты функции);
```

Оператор PROCEDURE определяет начало процедуры и основную точку входа в процедуру, задает параметры (если они есть) и атрибуты возвращаемого значения, если процедура вызывается как функция. Процедура заканчивается оператором END, который определяет границы процедурного блока. Вызов процедуры производится с помощью оператора CALL, который имеет следующий формат:

CALL имя входа (1-й аргумент, ..., h-й аргумент);

Оператор CALL вызывает процедуру и организует передачу управления к указанной точке входа этой процедуры. Программисту необходимо тщательно следить за соответствием набора и типа параметров-аргументов в операторах CALL и PROCEDURE.

В приведенных выше программах анализа и синтеза систем управления несколько раз (например, в программе КАСНМ) осуществлялось обращение к вспомогательным программам, разработанным на языке Фортран. Таким образом, становится необходимым организовать с помощью средств языка управления заданиями связь модулей, написанных на разных языках.

Связь модулей ПЛ/1 с модулями других языков имеет существенные особенности. В частности, многомерные массивы в Фортране хранятся последовательно по столбцам, в ПЛ/1 — по строкам. Кроме того, язык ПЛ/1 имеет больше типов данных, чем Фортран. Разнятся и способы организации данных. Необходимо тщательно следить за соответствием атрибутов информации, обмениваемой между программами, написанными на различных языках. (Более подробно о способах организации передачи данных см. [86].) Ниже помещена таблица соответствия данных языков ПЛ/1 и Фортран (табл. П.1).

Таблица П.1. **Таблица соответствия данных ПЛ/1 и Фортрана**

Тип	Длина, байт	Выравнивание	
		ALIGNED	UNALIGNED
	ПЛ/1		
REAL FIXED BIN	2	Полуслово	Байт
REAL FIXED BIN	4	Слово	»
REAL FLOAT BIN	4	»	»
REAL FLOAT BIN	8	Двойное слово	»
REAL FIXED DEC	~	Байт	»
REAL FLOAT DEC	4	Слово	»
REAL FLOAT DEC	8	Двойное слово	»
CPLX FLOAT DEC	8	Слово	»
CPLX FLOAT DEC	16	Двойное слово	»
BIT	~	Байт	Бит
BIT	1	»	»
BIT	4	»	»
CHAR	~	Байт	Байт
	Фортран		
INTEGER	2	Полуслово	
	4	Слово	
—	—	—	
—	—	—	
—	—	—	
REAL	4	Слово	
	8	Двойное слово	
COMPLEX	8	Слово	
	16	Двойное слово	
LOGICAL	1	Байт	
	4		
—	—	—	

Примечание. ~ означает произвольную допустимую длину.

При разработке программ, приведенных в настоящей книге, использовалась возможность вызова Фортран-программы из программы на языке ПЛ/1, причем передавались значения лишь арифметических переменных и массивов. Осуществить указанную операцию позволяет следующий набор управляющих операторов:

```
//PLFORT JOB
//PL   EXEC PL1FC
//PL1L.SYSIN DD *
    ПРОГРАММА НА ЯЗЫКЕ
         ПЛ/1
/*
//FOR   EXEC FORTGCLG
//FORT.SYSIN DD *
    ПРОГРАММА НА ЯЗЫКЕ
         ФОРТРАН
/*
//LKED.SYSLIB DD DSN=SYS1.PL1LIB,DISP=SHR
//            DD DSN=SYS1.FORTLIB,DISP=SHR
//LKED.SYSIN DD *
  ENTRY IHENTRY
/*
//GO.FT05F001 DD DDNAME=S
//GO.SYSPRINT DD SYSOUT=A
//GO.SYSIN    DD *
    ИСХОДНЫЕ ДАННЫЕ
/*
//
```

Приложение 2.

Элементы линейной алгебры

Матрицей называется прямоугольная таблица, составленная из чисел (функций) — элементов матрицы. Горизонтальные ряды элементов называются строками, вертикальные ряды — столбцами. Матрицы принято обозначать прописной буквой, соответствующей строчным буквам, которыми обозначены ее элементы, может быть указан также размер матрицы. Например,

$$\mathbf{A} = \begin{Vmatrix} a_{11} & & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{Vmatrix} = [\,a_{ij}\,]_{m\times n} = [\,a_{ij}\,].$$

Элементы матрицы A можно указывать и следующим способом: $(\mathbf{A})_{ij}$. Наиболее часто встречающимися являются следующие типы матриц.

Квадратная матрица — матрица, число строк которой равно числу столбцов:

$$\mathbf{A} = [\,a_{ij}\,]_{n\times n}.$$

Матрица-строка размера $1\times n$:

$$\mathbf{A} = \|a_1 \cdots a_n\|.$$

матрица-столбец размера $n\times 1$:

$$\mathbf{A} = \left\| \begin{matrix} a_1 \\ \vdots \\ a_n \end{matrix} \right\|,$$

матрицы-столбцы носят названия векторов.

Диагональная матрица — квадратная матрица, все элементы которой, возможно, кроме стоящих на главной диагонали (т. е. кроме элементов вида a_{ij}, $i=1, \ldots, n$), равны нулю:

$$\mathbf{A} = \left\| \begin{matrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ \ldots & \ldots & \ldots \\ 0 & 0 & a_{nn} \end{matrix} \right\| = \operatorname{diag}[a_{ij}],$$

в частном случае, когда все $a_{ii}=1$ ($i=1, \ldots, n$), матрица называется *единичной* и обозначается буквой E.

Транспонирование матрицы — это замена строк матрицы ее столбцами без изменения их последовательности:

$$\mathbf{A} = \left\| \begin{matrix} a_{11} & & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{matrix} \right\|, \quad \mathbf{A}^{\mathrm{T}} = \left\| \begin{matrix} a_{11} & & a_{m1} \\ \vdots & & \vdots \\ a_{1n} & \cdots & a_{mn} \end{matrix} \right\|,$$

транспонированная матрица обозначается $^{\mathrm{T}}$. Матрица A, для которой $\mathbf{A}^{\mathrm{T}}=\mathbf{A}$, называется *симметрической матрицей*.

Определитель квадратной матрицы $\mathbf{A}=[a_{ij}]_{n\times n}$ обозначается $\det(\mathbf{A})$ или $[\mathbf{A}]$ и вводится с помощью следующего рекуррентного соотношения:

$$\det(\mathbf{A}) = \sum_{i=1}^{n} a_{1i} \mathbf{D}_{1i},$$

где D_{1i} — алгебраическое дополнение элемента a_{1i}, т. е. определитель, полученный из A вычеркиванием 1-й строки и i-го столбца и умноженный на коэффициент $(-1)^{i+1}$. Отметим, что $\det(\mathbf{A})=\det(\mathbf{A}^{\mathrm{T}})$.

Обратной матрицей по отношению к исходной квадратной матрице A называется матрица, определяемая следующим образом:

$$(\mathbf{A}^{-1})_{ij} = D_{ji}/\det(\mathbf{A}), \quad \det(\mathbf{A}) \neq 0.$$

Если $\mathbf{A}^{-1}=\mathbf{A}^{\mathrm{T}}$, матрица называется *ортогональной*.

Среди операций, составляющих алгебру матриц, важнейшими являются операции умножения на скаляр, сложения и умножения.

Умножение матрицы на скаляр равносильно умножению на это число каждого элемента матрицы:

$$\lambda\mathbf{A} = \lambda \left\| \begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & a_{nn} \end{matrix} \right\| = \left\| \begin{matrix} \lambda a_{11} & \cdots & \lambda a_{1n} \\ \vdots & & \vdots \\ \lambda a_{n1} & & \lambda a_{nn} \end{matrix} \right\|.$$

Суммой двух матриц называется такая матрица, каждый элемент которой равен сумме соответствующих элементов слагаемых матриц:

$$\mathbf{C} = \mathbf{A} + \mathbf{B},$$

где $\mathbf{A}=[a_{ij}]_{m\times n}$; $\mathbf{B}=[b_{ij}]_{m\times n}$; $\mathbf{C}=[a_{ij}+b_{ij}]_{m\times n}$. Операция сложения матриц обладает следующими основными свойствами:

1) коммутативностью: $\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}$;

2) ассоциативностью: $(\mathbf{A}+\mathbf{B})+\mathbf{C}=\mathbf{A}+(\mathbf{B}+\mathbf{C})$;

3) для $\mathbf{A}$, $\mathbf{B}$ — любых двух матриц одинакового размера и произвольного скаляра λ справедливо $\lambda(\mathbf{A}+\mathbf{B})=\lambda\mathbf{A}+\lambda\mathbf{B}$;

4) транспонированная матрица суммы двух матриц равна сумме транспонированных слагаемых: $(\mathbf{A}+\mathbf{B})^{\mathrm{T}}=\mathbf{A}^{\mathrm{T}}+\mathbf{B}^{\mathrm{T}}$.

Умножение двух матриц $\mathbf{A}$ и $\mathbf{B}$ возможно, если число столбцов первого сомножителя равно числу строк второго. Само произведение определяется следующей формулой:

$$\mathbf{C} = \mathbf{AB},$$

где $(\mathbf{C})_{ik}=C_{ik}\sum\limits_{j=1}^{n} a_{ij}b_{jk}$; n — число столбцов матрицы $\mathbf{A}$. В общем случае умножение матриц некоммутативно, т. е. $\mathbf{AB}\neq\mathbf{BA}$. Операция умножения имеет следующие свойства:

1) ассоциативностью: $(\mathbf{AB})\mathbf{C}=\mathbf{A}(\mathbf{BC})$;

2) дистрибутивностью: $\mathbf{A}(\mathbf{B}+\mathbf{C})=\mathbf{AB}+\mathbf{AC}$;

3) транспонированная матрица произведения равна произведению транспонированных сомножителей, взятых в обратном порядке: $(\mathbf{AB})^{\mathrm{T}}=\mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$;

4) определитель произведения матриц равен произведению определителей сомножителей: $|\mathbf{AB}|=|\mathbf{A}||\mathbf{B}|$;

5) если $\det(\mathbf{A})\neq 0$, то $\mathbf{A}^{-1}\mathbf{A}=\mathbf{A}\mathbf{A}^{-1}=\mathbf{E}$;

6) если $\det(\mathbf{A})\neq 0$ и $\det(\mathbf{B})\neq 0$, то справедливо следующее правило обращения произведения матриц: $(\mathbf{AB})^{-1}=\mathbf{B}^{-1}\mathbf{A}^{-1}$.

Совокупность k векторов $x_1, \ldots, x_k$ называется линейно независимой, если уравнение

$$a_1\mathbf{x}_1+\ldots+a_k\mathbf{x}_k=0,$$

где $a_1, \ldots, a_k$ — скаляры, не имеет нетривиального (т. е. такого, у которого хотя бы один коэффициент a_i отличен от нуля) решения.

Рангом матрицы называется максимальное число линейно независимых столбцов (или строк). Квадратная n-мерная матрица $\mathbf{A}$ имеет ранг n, если $\det(\mathbf{A})\neq 0$. Это условие эквивалентно условию существования матрицы $\mathbf{A}^{-1}$ или условию линейной независимости строк (столбцов) матрицы $\mathbf{A}$.

Если матрица $\mathbf{A}$ имеет размер $m\times n$ и ее ранг равен $\min(m, n)$, то говорят, что $\mathbf{A}$ — *матрица полного ранга.*

Собственными значениями и собственными векторами квадратной матрицы $\mathbf{A}$ называются скалярные величины λ и вектора $\mathbf{x}$, удовлетворяющие уравнению $\lambda\mathbf{x}=\mathbf{A}\mathbf{x}$. Собственные значения находятся из решения уравнения $\det(\mathbf{A}-\lambda\mathbf{E})=0$, корнями которого они и являются.

Если квадратная матрица $\mathbf{A}$ размера $n\times n$ удовлетворяет соотношению $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}>0$ для всех ненулевых n-мерных векторов $\mathbf{x}$, то матрица $\mathbf{A}$ положительно

определена. Если выполняется условие $\mathbf{x}^T\mathbf{A}\mathbf{x} \geqslant 0$ для всех n-мерных векторов $\mathbf{x}$, то матрица $\mathbf{A}$ положительно определена (полуопределена) тогда, и только тогда, когда ее собственные значения λ положительны (неотрицательны).

Часто бывает очень удобным использовать разбиение произвольной матрицы $\mathbf{A}$ размера $m \times n$ на совокупность подматриц $\mathbf{A}_{ij}$ $(i=1, \ldots, k;\ j=1, \ldots, l)$ размера $m_i \times n_i$ $(m_1 + \ldots + m_k = m;\ n_1 + \ldots + n_l = n)$:

$$\mathbf{A} = \begin{Vmatrix} \mathbf{A}_{11} & \cdots & \mathbf{A}_{1l} \\ \cdots & \cdots & \cdots \\ \mathbf{A}_{k1} & \cdots & \mathbf{A}_{kl} \end{Vmatrix}.$$

Если некоторая матрица $\mathbf{A}$ — квадратная и неособенная ($\det \mathbf{A} \neq 0$), то для нее существует обратная матрица $\mathbf{A}^{-1}$. Если же $\mathbf{A}$ — прямоугольная матрица размера $m \times n (m \neq n)$ или квадратная, но особенная, то она не имеет обратной. В этом случае можно ввести операцию псевдообращения матрицы так, что псевдообратная матрица $\mathbf{A}^+$ будет обладать некоторыми свойствами обратной.

Матрица $\mathbf{A}^+$ размера $n \times m$ называется *псевдообратной* для матрицы $\mathbf{A}$ размера $m \times n$, если выполнены равенства

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A},\ \mathbf{A}^+ = \mathbf{U}\mathbf{A}^* = \mathbf{A}^*\mathbf{V},$$

где $\mathbf{U}$, $\mathbf{V}$ — некоторые матрицы, а матрица $\mathbf{A}^*$ — сопряженная матрица по отношению к $\mathbf{A}$: $\mathbf{A}^* = \|a_{ik}^*\|_{n \times m}$, $\mathbf{A} = \|a_{ik}\|_{m \times n}$ и $a^*_{kl} = \hat{a}_{ik}$ $(i=1, \ldots, m;\ k=1, \ldots, n;\ \hat{a}_{ik}$ — комплексносопряженная величина к $a_{ik})$.

Если $\mathbf{A}$ — квадратная неособенная матрица, псевдообратная матрица $\mathbf{A}^+$ совпадает с обратной: $\mathbf{A}^+ = \mathbf{A}^{-1}$. Кроме этого, матрица $\mathbf{A}^+$ обладает следующими свойствами:

$$1)\ (\mathbf{A}^*)^+ = (\mathbf{A}^+)^*, \quad 2)\ (\mathbf{A}^+)^+ = \mathbf{A}, \quad 3)\ (\mathbf{A}\mathbf{A}^+)^* = \mathbf{A}\mathbf{A}^+,$$

$$4)\ (\mathbf{A}^+\mathbf{A})^* = \mathbf{A}^+\mathbf{A}, \quad 5)\ (\mathbf{A}\mathbf{A}^+)^2 = \mathbf{A}\mathbf{A}^+, \quad 6)\ (\mathbf{A}^+\mathbf{A})^2 = \mathbf{A}^+\mathbf{A}.$$

Операция псевдообращения матриц широко используется при решении систем линейных уравнений.

Приложение 3.

Некоторые понятия теории случайных процессов

Случайным процессом называется семейство случайных величин $\xi(t) = \xi(t, \omega)$, зависящих от скалярного параметра t, принадлежащего некоторому множеству $\mathbf{T} \subset \mathbf{R}$ ($\mathbf{R}$ — действительная ось), и элементарного случайного события $\omega \in \Omega$ (Ω— вероятностное пространство). Область значений случайного процесса может быть произвольным измеримым пространством. На практике обычно $\xi(t)$ принимает значения из n-мерного евклидова пространства $\mathbf{R}^n$. Параметр t может принимать значения как их непрерывного множества, так и из дискретного.

Функция $\xi(t, \omega_0)$ при фиксированном значении $\omega_0 \in \Omega$ называется *реализацией процесса (траекторией, выборочной функцией).*

Для описания поведения случайного процесса $\xi(t)$ достаточно иметь функции конечномерных распределений вероятностей

$$F\{\xi_1, ..., \xi_m; t_1, ..., t_m\} = P\{\xi(t_1) \leqslant \xi_1, ..., \xi(t_m) \leqslant \xi_m\}, m \geqslant 1; t_i \in T, i = 1, ..., m.$$

Функции $F(\xi_1, ..., \xi_m; t_1, \quad t_m\}$ должны удовлетворять условию симметричности и согласованности

$$F\{\xi_1, ..., \xi_m; t_1, ..., t_m\} = F\{\xi_{t_1}, ..., \xi_{t_m}; t_{i_1}, ..., t_{i_m}\},$$

где $(i_1, ..., i_m)$ — некоторая перестановка множества $(1, ..., m)$;

$$F\{\xi_1, ..., \xi_{m-1}; t_1, ..., t_{m-1}\} = \lim_{\xi_m \to \infty} F\{\xi_1, ..., \xi_m; t_1, ..., t_m\}.$$

Математическое ожидание $m(t)$ и ковариационная функция $K(s, t)$ процесса $\xi(t)$ являются функциями времени и определяются с помощью следующих соотношений:

$$m(t) = M\xi(t) = \int_{-\infty}^{\infty} x dF(x,t),$$

$$K(s,t) = M[\xi(s) - m(s)][\xi(t) - m(t)]^T$$

Пусть $F\{\xi(t) \leqslant \xi(t) | \xi(t_1), ..., \xi(t_m)\}$ — условная функция распределения процесса $\xi(t)$ при фиксированных значениях $\xi(t_1), ..., \xi(t_m) (t_1 < t_m < t)$.

Процесс $\xi(t)$ называется *марковским*, если выполняется соотношение

$$F\{\xi(t) \leqslant \xi_t | \xi(t_1), ..., \xi(t_m)\} = F\{\xi(t) \leqslant \xi_t | \xi(t_m)\},$$

т. е. условное распределение $\xi(t)$ зависит лишь от последнего значения $\xi(t_m)$. Марковский процесс целиком определяется функцией начального распределения $F\{\xi_0, t_0\}$ и функцией распределения вероятности перехода

$$F\{\xi_t, t | \xi_s, s\} = P\{\xi(t) \leqslant \xi_t | \xi(s) = \xi_s\}$$

С помощью начального распределения и распределения перехода конечномерные распределения марковского процесса записываются в следующем виде:

$$F\{\xi_0, ..., \xi_m; t_0, ..., t_m\} = \int_{-\infty}^{\xi_0} dF\{x_0, t_0\} \int_{-\infty}^{\xi_1} \int_{-\infty}^{\xi_{m-1}} F\{\xi_m, t_m | x_{m-1}, t_{m-1}\} dF\{x_{m-1}, t_{m-1} | x_{m-2}, t_{m-2}\}.$$

Случайный процесс $\xi(t)$ называется *процессом с независимыми приращениями*, если для любых $t_i \in T (i = 1, \quad n)$, таких, что $t_1 < t_2 < ... < t_m$, случайные величины

$\eta_1 = \xi(t_1)$, $\eta_2 = \xi(t_2) - \xi(t_1)$, ..., $\eta_m = \xi(t_m) - \xi(t_{m-1})$ взаимно независимы. Если η_i ($i = 1, ..., m$) только не коррелированы, то процесс $\xi(t)$ называется *процессом с ортогональными приращениями*.

Случайный процесс $\xi(t)$ называется *стационарным*, если распределение любой совокупности $\xi(t_1)$, ..., $\xi(t_m)$ ($t_i \in T$) инвариантно относительно сдвига на величину τ:

$$F(\xi_1, ..., \xi_m; t_1, ..., t_m) = F\{\xi_1, ..., \xi_m; t_1 + \tau, ..., t_m + \tau\}, \quad t_i + \tau \in T, \; i = 1, ..., m.$$

Если инварианты только первые два момента процесса $\xi(t)$, то он называется *слабостационарным*.

Важное место в теории случайных процессов и ее приложениях занимают гауссовские процессы. Процесс $\xi(t)$ называется *гауссовским*, если плотность распределения величин $\xi(t_1)$, ..., $\xi(t_m)$ определяется следующим соотношением:

$$f(\mathbf{x}) = (2\pi)^{-k/2} [\det(\mathbf{K})]^{-1/2} \exp[0{,}5\,(\mathbf{x} - \mathbf{m})^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{x} - \mathbf{m})],$$

где $\mathbf{x} = (x_1, ..., x_k)^{\mathrm{T}}$; $\mathbf{m} = (m_1, ..., m_k)^{\mathrm{T}}$, $m_i = \mathrm{M}\xi(t_i)$, $i = 1, ..., k$;

$$\mathbf{K} = [k_{ij}]_{k \times k}, \; k_{ij} = \mathrm{M}[\xi(t_i) - m_i][\xi(t_j) - m_j]^{\mathrm{T}}, \quad i, j = 1, ..., k.$$

Таким образом, гауссовский процесс целиком определяется математическим ожиданием и ковариационной функцией.

Пусть $\xi(t)$ — слабостационарный процесс с ковариационной функцией $\mathrm{K}(\tau)$. *Спектральной плотностью* процесса $\xi(t)$ называется функция $G(\omega)$, определяемая соотношениями:

для процесса с непрерывным временем

$$G(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-j\omega\tau} \mathrm{K}(\tau)\, d\tau,$$

для процесса с дискретным временем

$$G(\omega) = \frac{1}{2\pi} \sum_{l=-\infty}^{\infty} \mathrm{K}(l)\, e^{-jl\omega}.$$

Спектральная плотность есть функция, равная преобразованию Фурье ковариационной функции $\mathrm{K}(\tau)$. С помощью обратного преобразования по известной функции $G(\omega)$ можно восстановить $\mathrm{K}(\tau)$:

для процесса с непрерывным временем

$$\mathrm{K}(\tau) = \int_{-\infty}^{\infty} e^{j\omega\tau} G(\omega)\, d\omega,$$

для процесса с дискретным временем

$$\mathrm{K}(l) = \int_{-\pi}^{\pi} e^{jl\omega} G(\omega)\, d\omega.$$

Приведем алгоритмы и программу вычисления определителя матрицы и ее обращения, которые находят практическое применение в большинстве алгоритмов анализа и синтеза систем автоматического управления. Проблемам синтеза эффективных методов вычисления определителей и обращения матриц посвящено много работ (например, [88, 89]). Большую известность получил излагаемый здесь метод Гаусса, а также его модификации.

Пусть надо вычислить определитель квадратной матрицы $\mathbf{A}=(a_{ij}^1)$ $(i, j=1, ..., n)$. Здесь верхний индекс у элементов a_{ij}^1 — номер итерации вычислений. При условии $a_{11}^1 \neq 0$, разделив первую строку на a_{11}^1, получим:

$$\Delta_1 = \det(a_{ij}^1) = a_{11}^{11} \begin{vmatrix} 1 & b_{12} & \dots & b_{1n} \\ a_{21}^1 & a_{22}^1 & \dots & a_{2n}^1 \\ \dots & \dots & \dots & \dots \\ a_{n1}^1 & a_{n2}^1 & \dots & a_{nn}^1 \end{vmatrix} = a_{11}^1 \Delta_2;$$

после линейных преобразований последовательно получаем

$$\Delta_2 = a_{22}^2 \Delta_3, \dots, \Delta_{n-1} = a_{n-1,\,n-1}^{n-1} \Delta_n.$$

$$\Delta_i = \begin{vmatrix} a_{ij}^i, \dots, a_{in}^i \\ a_{ni}^i, \dots, a_{nn}^i \end{vmatrix}, \quad i = 1, \dots, n-1; \quad \Delta_n = a_{nn}^n,$$

где $a_{ij}^{k+1} = a_{ij}^k - a_{ik}^k b_{kj}$, $b_{kj} = a_{kj}^k / a_{kk}^k$, $k=1, \dots, n$; $i=k+1, \dots, n$; $j=k+1, \dots, n$.

Для ведущих элементов a_{kk}^k $(k=1, \dots, n)$ должно выполняться условие $a_{kk}^k \neq 0$. Таким образом, $\Delta_1 = \det\mathbf{A} = \prod_{i=1}^{n} a_{ii}^i$. Общее количество арифметических операций умножения и деления, необходимых для вычисления определителя, $(n^3+2n)/3-1$.

Аналогичная схема (по методу исключения Гаусса) используется для решения систем линейных уравнений с квадратной матрицей n-го порядка:

$$\mathbf{A}_x = a_{n+1}, \quad \mathbf{A} = (a_{ij}^1);$$

$$\mathbf{x}^T = (x_1, \dots, x_n);$$

$$\mathbf{a}_{n+1}^T = (a_{1,n+1}^1, \dots, a_{n,n+1}^1).$$

Определив a_{ij}^{k+1}, b_{kj}, находим неизвестные $x_1, \dots, x_n$:

$$x_i = b_{i,n+1} - \sum_{j=i+1}^{n} b_{ij} x_j, \quad i = n, \dots, 1.$$

Обращение матрицы $\mathbf{A}$ находим с помощью решения n систем линейных уравнений вида

$$\mathbf{A}\mathbf{x}^1 = \mathbf{e}^1, \ \mathbf{A}\mathbf{x}^2 = \mathbf{e}^2, \dots, \mathbf{A}\mathbf{x}^n = \mathbf{e}^n,$$

где $\mathbf{e}^i$ $(i=1, \dots, n)$ — вектор со всеми нулевыми координатами и $i=$й, равной единице. Обратная по отношению к $\mathbf{A}$ матрица $\mathbf{A}^{-1}$ вычисляется с помощью $\mathbf{x}^1, \dots, \mathbf{x}^n$: $\mathbf{A}^{-1} = (\mathbf{x}^1, \dots, \mathbf{x}^n)$.

В настоящее время имеется достаточно большое количество программ линейной алгебры. В первую очередь следует отметить программы, описанные в [88]. Здесь приведена программа обращения матрицы — программа INV, разработанная на языке Фортран. Программа вызывается с помощью оператора

CALL INV (A, B, C, Z, N, EPS).

Входные параметры:

A(N,N) — исходная матрица размера N*N;

B(N), C(N) — вспомогательные массивы размера N;

N — параметр размерности;

EPS — точность вычисления;

Z(N) — рабочий массив.

Тестовой пример состоит в обращении матрицы

$$A(1,1) = 5,\ A(2,1) = 7,\ A(3,1) = 6,\ A(4,1) = 5;$$
$$A(1,2) = 7,\ A(2,2) = 10,\ A(3,2) = 8,\ A(4,2) = 7;$$
$$A(1,3) = 6,\ A(2,3) = 8,\ A(3,3) = 10,\ A(4,3) = 9;$$
$$A(1,4) = 5,\ A(2,4) = 7,\ A(3,4) = 9,\ A(4,4) = 10.$$

Результаты вычислений:

$$A^{-1}(1,1) = 67{,}999;$$
$$A^{-1}(2,1) = -40{,}999,\ A^{-1}(2,2) = 24{,}999;$$
$$A^{-1}(3,1) = -16{,}999,\ A^{-1}(3,2) = 9{,}999,\ A^{-1}(3,3) = 4{,}999;$$
$$A^{-1}(4,1) = 9{,}999,\ A^{-1}(4,2) = -5{,}999,\ A^{-1}(4,3) = -2{,}999,\ A^{-1}(4,4) = 1{,}999.$$

(остальные элементы обратной матрицы A^{-1} нулевые).

```
      PROGRAM INV
      REAL DET
      DIMENSION A(2,2),A1(2,2),C(2,2),B(2)
      A(1,1)=5
      A(1,2)=7
      A(2,1)=7
      A(2,2)=5
      PRINT 1000
1000  FORMAT(1X,'INITIAL DATA'//1X,'MATRICA A'/)
      PRINT 1001,((A(I,J),J=1,2),I=1,2)
1001  FORMAT(1X,2(F6.2)/,1X,2(F6.2)/,1X,2(F6,2))
      CALL OBR(A,A1,C,B,DET,2)
      PRINT 1100
1100  FORMAT(1X,'INVERT MATRIX'/)
      PRINT 1110,((A(J,I),I=1,2),J=1,2)
1110  FORMAT(1X,2(F10.5)/,1X,2(F10.5)/,1X,2(F10.5))
      PRINT 1002,DET
1002  FORMAT(1X,'DET=',F10.5)
      STOP
      END
      SUBROUTINE OBR(A,A1,C,B,DET,N)
      DIMENSION A(N,N),A1(N,N),C(N,N),B(N)
      DO 100 K=1,N
      DO 17 I=1,N
      B(I)=0
      DO 17 J=1,N
17    C(I,J)=A(I,J)
      B(K)=1
```

```
      CALL GAUS(C,B,DET,N)
      DO 27 I=1,N
27    A1(I,K)=B(I)
100   CONTINUE
      RETURN
      END
      SUBROUTINE GAUS(A,B,DET,N)
      DIMENSION A(N,N),B(N)
      DET=1
      II=1
      DO 100 I=1,N-1
      K=I
      T=ABS(A(I,I))
      DO 70 L=I+1,N
      F=ABS(A(L,I))
      IF(T.GE.F)GOTO 70
      T=F
      K=L
70    CONTINUE
      IF(T.EQ.0)GOTO 50
      IF(K.EQ.I)GOTO 80
      DO 60 J=I,N
      D=A(I,J)
      A(I,J)=A(K,J)
60    A(K,J)=D
      D=B(I)
      B(I)=B(K)
      B(K)=D
      II=-II
80    DO 110 K=I+1,N
110   A(I,K)=A(I,K)/A(I,I)
      B(I)=B(I)/A(I,I)
      DO 140 L=I+1,N
      DO 130 K=I+1,N
130   A(L,K)=A(L,K)-A(I,K)*A(L,I)
140   B(L)=B(L)-B(I)*A(L,I)
100   CONTINUE
      IF(A(N,N).EQ.0)GOTO 50
      B(N)=B(N)/A(N,N)
      DO 170 I=N-1,1,-1
      DO 170 J=I+1,N
170   B(I)=B(I)-A(I,J)*B(J)
      DO 117 I=1,N
117   DET=DET*A(I,I)
      DET=DET*II
      RETURN
50    PRINT 55
55    FORMAT(10X,'GAUS: MATRIX A HAS ZERO DETERMINANT')
      RETURN
      END
```

Список литературы

1. **Шуп Т.** Решение инженерных задач на ЭВМ.— М.: Мир, 1982.—240 с.
2. **Форсайт Д. Ж., Малькольм М., Моулер К.** Машинные методы математических вычислений.— М.: Мир, 1980.— 280 с.
3. **Мелса Дж., Джонс Ст.** Программы в помощь изучающим теорию линейных систем управления.— М.: Машиностроение, 1981.— 200 с.
4. **Пакет** прикладных программ по автоматизированному проектированию цифровых оптимальных и адаптивных систем управления МИФИ.— М.: 1980.— 60 с.
5. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов.— М.: Мир, 1979.—536 с.
6. **Трауб Дж., Вожьняковский Х.** Общая теория оптимальных алгоритмов.— М.: Мир, 1983.— 384 с.
7. **Растригин Л. А.** Вычислительные машины, системы, сети.— М.: Наука, 1982.— 224 с.
8. **Библиотека** алгоритмов 1б — 50б.— М.: Сов. радио, 1975.— 176 с.
9. **Цыпкин Я. З.** Основы теории автоматических систем.— М.: Наука, 1977.—560 с.
10. **Попов Е. П.** Теория нелинейных систем автоматического регулирования и управления. М.: Наука, 1979.—256 с.
11. **Справочник** по теории автоматического управления/Под ред. А. А. Красовского.— М.: Наука, 1987.—712 с.
12. **Крутько П. Д.** Статистическая динамика импульсных систем.—М.: Сов. радио, 1963.—560 с.
13. **Гихман И. И., Скороход А. В.** Теория случайных процессов.— В 3-х т.—М.: Наука,—Т.1.—1971; Т.2.—1973; Т.3.—1975.
14. **Острем К.** Введение в стохастическую теорию управления.— М.: Мир, 1973.— 324 с.
15. **Колмановский В. Б. Носов Р. В.** Устойчивость и периодические режимы регулируемых систем с последействием.— М.: Наука, 1981.—448 с.
16. **Полак Э.** Численные методы оптимизации.— М.: Мир, 1974.— 376 с.
17. **Ройтенберг Я. Н.** Автоматическое управление.— М.: Наука, 1978.—552 с.
18 **Эйкхофф П.** Основы идентификации систем управления.— М.: Мир, 1975.— 688 с.
19. **Основы** кибернетики/Под ред. К. А. Пупкова.— М.: Высшая школа, 1976.— 408 с.
20. **Брайсон А., Хо Ю-Ши.** Прикладная теория оптимального управления.— М.: Мир, 1972.—544 с.
21. **Курант Р., Гильберт Д.** Методы математической физики.— Т.2.—М.: Гостехиздат, 1951.
22. **Курант Р.** Дифференциальные уравнения в частных производных.— М.: Мир, 1968.
23. **Вентцель Е. С.** Элементы динамического программирования.— М.: Наука, 1964.— 174 с.
24. **Высшая** математика. Специальные главы/Под ред. П. И. Чинаева. Киев: Вища школа, 1977.—368 с.
25. **Коршунов Ю. М.** Математические основы кибернетики.— М.: Энергия, 1980.— 424 с.
26. **Пропой А. И.** Элементы теории оптимальных дискретных систем.— М.: Наука, 1973.—210 с.
27. **Черноусько Ф. Л., Баничук Н. В.** Вариационные задачи механики и управления.— М.: Наука, 1973.—252 с.
28. **Моисеев Н. Н.** Элементы теории оптимальных систем.— М.: Наука, 1975.—318 с.
29. **Математическая** теория оптимальных процессоров/Понтрягин Л. С., Болтянский В. Г., Гамкрелидзе Р. В., Мищенко Е. Ф.— М.: Наука, 1969.—392 с.

30. **Красовский Н. Н.** Теория управления движением.— М.: Наука, 1968.—476 с.
31. **Крутько П. Д.** Вариационные методы синтеза систем с цифровыми регуляторами.— М.: Сов. радио, 1967.—440 с.
32. **Крутько П. Д., Максимов А. И., Скворцов Л. М.** Алгоритмы и программы проектирования автоматических систем.— М.: Радио и связь, 1988.—304 с.
33. **Летов А. М.** Динамика полета и управления.— М.: Наука, 1969.—360 с.
34. **Зубов В. И.** Лекции по теории управления.— М.: Наука, 1975.—496 с.
35. **Красовский А. А., Поспелов Г. С.** Основы автоматики и технической кибернетики.— М.: Энергоиздат, 1962 с.—600 с.
36. **Емельянов С. В., Коровин С. К., Сизиков В. И.** Управление свободным движением динамических систем с использованием координатно-параметрических и параметрических обратных связей//Изв. АН СССР. Техн. кибернетика.— 1982.—№ 4.— С. 136—143.
37. **Чинаев П. И.** Методы анализа и синтеза многомерных автоматических систем.— Киев: Техника, 1969.—378 с.
38. **Чинаев П. И., Сильвестров А. Н.** Идентификация и оптимизация автоматических систем.— М.: Энергоатомиздат, 1987.—198 с.
39. **Калман Р., Фалб П., Арбиб М.** Очерки по математической теории систем.— М.: Мир, 1971.—400 с.
40. **Красовский А. А.** Аналитическая форма субоптимального адаптивного управления нелинейными объектами//Изв. АН СССР. Техн. кибернетика.—1983.—№ 2.— С. 137—146.
41. **Аведьян Э. Д.** Модифицированный алгоритм Качмажа для оценки параметров линейных объектов//Автоматика и телемеханика.—1978.—№ 5.— С. 64—72.
42. **Аведьян Э. Д., Цыпкин Я. З.** Обобщенный алгоритм Качмажа//Автоматика и телемеханика.—1979.—№ 1.— С. 72—78.
43. **Маркл Я.** Ускорение сходимости алгоритма Качмажа в случае временной корреляции входного процесса//Автоматика и телемеханика.—1980.—№ 8.— С. 70—73.
44. **Быков В. В.** Цифровое моделирование в статистической радиотехнике.— М.: Сов. радио, 1971.—328 с.
45. **Брамер К., Зиффлинг Г.** Фильтр Калмана-Бьюси.— М.: Наука, 1982.—200 с.
46. **Липцер Р. Ш., Ширяев А. Н.** Статистика случайных процессов.—М.: Наука, 1974.—696 с.
47. **Гихман И. И., Скороход А. В.** Стохастические дифференциальные уравнения.— Киев: Наукова думка, 1968.—328 с.
48. **Скороход А. В.** Стохастические уравнения для сложных систем.—М.: Наука, 1983.—192 с.
49. **Ершов А. А., Липцер Р. Ш.** Робастный фильтр Калмана в дискретном времени// Автоматика и телемеханика.—1978.—№ 3.—С. 60—69.
50. **Шапиро Е. Н.** Стабильное решение задачи нелинейной фильтрации в дискретном времени//Автоматика и телемеханика.—1980.—№5.—С. 99—105.
51. **Демин Н. С., Жадан Л. И.** Некоторые адаптивные варианты фильтра Калмана-Бьюси для дискретных систем//Адаптация и обучение в системах управления и принятия решений.— Новосибирск: Наука, 1982.— С. 61—70.
52. **Ершов А. А.** Стабильные методы оценки параметров//Автоматика и телемеханика.—1978.—№ 8.— С. 66—100.
53. **Гриценко Н. С.** и др. Адаптивное оценивание//Зарубежная радиоэлектроника.— 1983.—№ 7.— С. 3—27.
54. **Руководство** по проектированию систем автоматического управления/Под ред. В. А. Бесекерского.— М.: Высшая школа, 1983.—296 с.
55. **Растригин Л. А.** Системы экстремального управления.— М.: Наука, 1974.—632 с.
56. **Карманов В. Г.** Математическое программирование.— М.: Наука, 1980.—256 с.
57. **Белоусов Е. Г.** Введение в выпуклый анализ и целочисленное программирование.— М.: Изд-во МГУ, 1977.—196 с.
58. **Химмельблау Д.** Прикладное нелинейное программирование.— М.: Мир, 1975.— 534 с.
59. **Библиотека** алгоритмов 151б—200б.— М.: Радио и связь.—1981.—184 с.

60. **Базара М., Шетти К.** Нелинейное программирование: Теория и алгоритмы.— М.: Мир, 1982.—584 с.
61. **Известия** АН СССР. Техн. кибернетика.—1983.—№ 1.—208 с.
62. **Библиотека** алгоритмов 101б—150б.— М.: Сов. радио, 1978.—128 с.
63. **Кудрявцев Е. М.** Исследование операций в задачах, алгоритмах и программах.— М.: Радио и связь, 1984.—184 с.
64. **Фиакко А., Мак-Кормин Г.** Нелиненое программирование: Методы последовательной безусловной минимизации.— М.: Мир, 1972.—240 с.
65. **Гроссман К., Каплан А. А.** Нелинейное программирование на основе безусловной минимизации.— Новосибирск: Наука, 1981.
66. **Сборник** задач по теории автоматического регулирования и управления/ Под ред. В. А. Бесекерского.—3-е изд.—М.: Наука, 1969.
67. **Корбут А. А., Финкельштейн Ю. Ю.** Дискретное программирование.—М.: Наука, 1969.—368 с.
68. **Мишин В. П., Осин М. И.** Введение в машинное проектирование летательных аппаратов.— М.: Машиностроение, 1978.—174 с.
69. **Сергиенко И. В., Лебедева Т. Т., Рощин В. А.** Приближенные методы решения дискретных задач оптимизации.— Киев: Наукова думка, 1980.—276 с.
70. **Кудрицкий В. Д., Синица М. А., Чинаев П. И.** Автоматизация контроля радиоэлектронной аппаратуры.— М.: Сов. радио, 1977.—244 с.
71. **Новоселов А. С., Болнокин В. Е., Чинаев П. И., Юрьев А. Н.** Системы адаптивного управления летательными аппаратами.— М.: Машиностроение, 1987.—280 с.
72. **Финкельштейн Ю. Ю.** Приближенные методы и прикладные задачи дискретного программирования.— М.: Наука, 1976.
73. **Михалевич В. С., Волкович В. Л.** Вычислительные методы исследования и проектирования сложных систем.— М.: Наука, 1982.—288 с.
74. **Юдин Д. Б., Гольштейн Е. Г.** Линейное программирование: Теория, методы и приложения.— М.: Наука, 1969.—424 с.
75. **Зойнтендейк Г.** Методы возможных направлений.— М.: ИЛ, 1963.—198 с.
76. **Баженов В. И., Болнокин В. Е., Качанов А. И., Осин М. И.** Инженерные методы поиска рациональных решений при автоматизированном проектировании ЛА// Труды 14-х чтений, посвященных разработке научного наследия и развитию идей К. Э. Циолковского.— М.: Наука, 1980.—С. 92—97.
77. **Корнейчук Н. П.** Сплайны в теории приближений.—М.: Наука, 1984.—352 с.
78. **Бабенко К. И.** Основы численного анализа.— М.: Наука, 1986.—744 с.
79. **Гумбель Э.** Статистика экстремальных значений.— М.: Мир, 1965.—450 с.
80. **Березин И. С., Жидков Н. П.** Методы вычислений.— М.: Наука, 1966.—632 с.
81. **Демидович Б. П., Марон И. А.** Основы вычислительной математики.—М.: Наука, 1970.—664 с.
82. **Ибрагимов И. А., Хасьминский Р. З.** Асимптотическая теория оценивания.— М.: Наука, 1979.—528 с.
83. **Сборник** научных программ на Фортране. Вып. 1.— М.: Статистика, 1974.—316 с.
84. **Цвиркун А. Д.** Основы синтеза структуры сложных систем.— М.: Наука, 1982.— 200 с.
85. **Лебедев В. Н., Соколов А. П.** Введение в систему программирования ОС ЕС.— М.: Статистика, 1978.—144 с.
86. **Джермейн К.** Программирование на IBM/360.—М.: Мир, 1971.—870 с.
87. **Переход** от ДОС ЕС к ОС ЕС: Справочное пособие.—М.: Статистика, 1980.— 232 с.
88. **Носов В. Р.** Алгоритмы решения задач линейной алгебры.— М.: МИЭМ, 1983.— 80 с.
89. **Хорн Р., Джонсон Ч.** Матричный анализ.— М.: Мир, 1989.—656 с.

Оглавление