НЕЙРОКОМПЬЮТЕРЫ И ИХ ПРИМЕНЕНИЕ

КНИГА 2

Сигеру Омату, Марзуки Халид,
Рубия Юсоф

НЕЙРОУПРАВЛЕНИЕ И ЕГО ПРИЛОЖЕНИЯ

ИПРЖР

НЕЙРОУПРАВЛЕНИЕ И ЕГО ПРИЛОЖЕНИЯ

Sigeru Omatu, Marzuki Khalid,
Rubiyah Yusof

Neuro – Control and its Applications

Springer

НЕЙРОКОМПЬЮТЕРЫ И ИХ ПРИМЕНЕНИЕ

КНИГА 2

Сигеру Омату, Марзуки Халид,
Рубия Юсоф

НЕЙРОУПРАВЛЕНИЕ И ЕГО ПРИЛОЖЕНИЯ

Перевод с английского Н. В. Батина
Под общей редакцией
докт. техн. наук, проф. А. И. Галушкина
докт. техн. наук, проф. В. А. Птичкина

Издательское предприятие редакции журнала «Радиотехника»

Москва 2000

УДК 621
О 57
ББК 32. 818

Нейрокомпьютеры и их применение

Перевод с английского Н В Батина
Под общей редакцией
докт техн наук, проф А И Галушкина,
докт техн наук, проф В А Птичкина

Сигеру Омату

О 57 Нейроуправление и его приложения Кн 2
/Сигеру Омату, Марзуки Халид, Рубия Юсоф. Пер с англ Н В Батина.
Под ред А И Галушкина. В А Птичкина – М ИПРЖР, 2000 –
272 с ил (Нейрокомпьютеры и их применение)

ISBN 5-93108-006-6

Приводится систематическое изложение методов нейронного управления, основанное на современных разработках в области нечеткой логики и теории нейронных сетей Методология нейронного управления сравнивается с традиционными методами теории управления

Для научных работников, аспирантов и студентов, интересующихся этим новым направлением в информатике и теории управления

УДК 621
ББК 32. 818

Научная серия

НЕЙРОКОМПЬЮТЕРЫ И ИХ ПРИМЕНЕНИЕ

Сигеру Омату, Марзуки Халид, Рубия Юсоф

НЕЙРОУПРАВЛЕНИЕ И ЕГО ПРИЛОЖЕНИЯ. Кн. 2

Зав редакцией И Е Якушина
Корректор Р М Ваничкина

Изд № 61 Сдано в набор 15 02 2000
Подписано в печать 15 07 2000 Формат 60 x 90 1/16
Бумага офсетная Гарнитура Times
Печать офсетная
Печ л 17 Тираж 3000 экз Зак № 674

ЛР № 065229 от 20.06.97.
Издательское предприятие редакции журнала "Радиотехника"
103031, Москва, К-31, Кузнецкий мост, д 20/6, ИПРЖР
Тел /факс 921-48-37, 925-78-72, 925-92-41

Отпечатано в ППП Типография "Наука"
121099, Москва, Шубинский пер , д 6

ISBN 3-540-19965-9

ISBN 5-93108-006-6

Содержание

ПРЕДИСЛОВИЕ РЕДАКТОРОВ ПЕРЕВОДА

Управление динамическими системами является широкой сферой современной науки, связанной с построением систем управления разнообразными объектами. Примерами таких систем могут быть ракетно-космические и авиационные системы, системы управления химическими реакторами и многие другие.

Формальная постановка задачи здесь сводится, как правило, к решению систем обыкновенных дифференциальных уравнений. Теория управления линейными системами, которые описываются линейными дифференциальными уравнениями, была достаточно хорошо разработана еще в 60-е годы с применением аппарата преобразований Лапласа и частотно-фазовых характеристик. Тогда же предпринимались первые попытки применения цифровых вычислительных машин в качестве корректирующих фильтров в контуре управления динамическими системами.

Первые примеры разработки методов управления нелинейными системами относятся также к началу 60-х годов и связаны в основном с методами решения нелинейных дифференциальных уравнений, адекватных однопроцессорным фон-Неймановским вычислительным машинам.

Последовавшее затем развитие вычислительных машин с массовым параллелизмом, к которым относятся и нейрокомпьютеры, привело к созданию принципиально новых алгоритмов и методов управления нелинейными динамическими системами. Они связаны с нейросетевыми алгоритмами решения обыкновенных нелинейных дифференциальных уравнений и, как следствие, с включением нейрокомпьютера в контур управления нелинейной динамической системой.

Данная монография является одной из первых в этой области. В ней сравнительно просто излагаются принципы постро-

ения нейросетевых алгоритмов управления динамическими системами. Достаточно широкое развитие и распространение подобных алгоритмов привело в последние годы к созданию целого раздела науки под названием «нейроуправление». После изложения в достаточно простой форме принципов построения нейронных сетей и традиционных схем управления динамическими системами в монографии рассматриваются различные схемы адаптивного нейросетевого управления. В последней главе приводятся примеры построения нейросетевых алгоритмов управления различными динамическими объектами.

Книга написана простым и ясным языком. Она будет весьма полезна студентам, аспирантам и специалистам более высокой категории в области современной теории управления, а также в области построения конкретных практических систем управления сложными динамическими объектами.

Д.т.н. проф. А.И. Галушкин
Д.т.н. проф. В.А. Птичкин

1. ВВЕДЕНИЕ

1.1. Общие сведения об интеллектуальном управлении

В последнее время интеллектуальное управление становится широко распространенным средством для многих технических и промышленных приложений [1], [2]. Такие системы управления обладают способностью к пониманию и обучению в отношении объекта управления, возмущений, внешней среды, условий работы [3], [4]. Примеры факторов, которым могут обучаться такие системы — характеристики объекта (статические и динамические), некоторые характеристики возмущений и внешней среды, методы управления оборудованием [1], [2]. Рис. 1.1.1 и 1.1.2 иллюстрируют количество статей из INSPEC (Информационная служба физических и инженерных обществ) и патентов из CASSIS (Классификатор для информационно-поисковых систем) [5] соответственно. На них видны тенденции и соотношения в объеме исследований по различным направлениям искусственного интеллекта. Можно убедиться, что исследования по экспертным системам, традиционно считавшиеся основным инструментом интеллектуальных систем, медленно сокращаются, а исследования по нейронным сетям стабильно нарастают.

Благодаря своим способностям к самоорганизации и обучению, искусственные нейронные сети сейчас рассматриваются как перспективные средства для интеллектуальных систем [3], [4]. Архитектура и функции таких сетей строятся на основе биологических структур мозга. В нейронных сетях используется архитектура вычислений, отличная от фон-неймановской. Характерные черты нейронных сетей — параллельность, распределенность, самоорганизация, в то время как обычные компьютеры характеризуются последовательностью, локальностью, работой по алгоритму [3], [4]. «Параллельная архитек-

тура» означает, что обработка информации выполняется несколькими центральными процессорами (ЦП), в то время как в обычных компьютерах, имеющих только один ЦП, информация обрабатывается последовательно, по шагам.

Распределенная память означает, что информация хранится по многим адресам, распределенным образом, так что каждый элемент данных представляется шаблоном активности, распределенным по многим вычислительным элементам, и каждый вычислительный элемент участвует в представлении многих различных элементов данных. В обычных компьютерах реализуется локальная память, или локальное представление, в котором используется один вычислительный элемент для каждого элемента данных. На основе распределенной архитектуры представления информация в нейронных сетях может дробиться и обрабатываться по частям [3].

Последнее из характерных свойств нейронных сетей — самоорганизация — называется также способностью к обучению

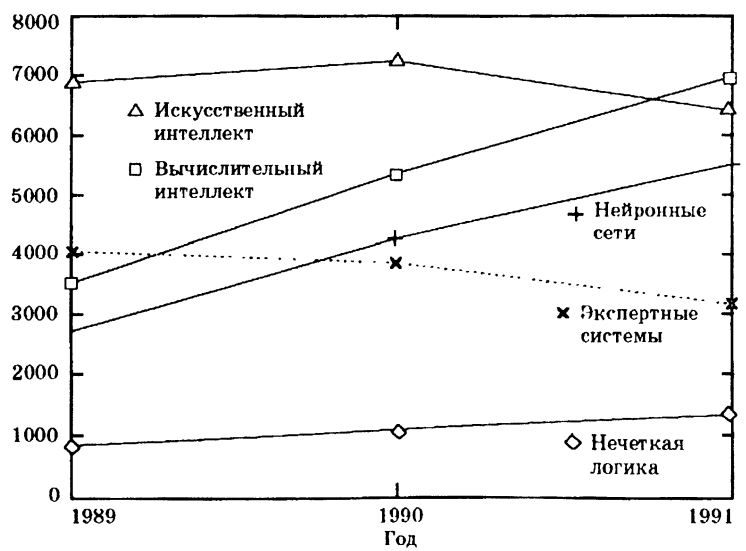


Рис. 1.1.1. Количество статей и других работ по различным областям искусственного интеллекта, зарегистрированных в базе данных INSPEC [5] (©1994 IEEE Press)

[1]. Это означает, что нейронные сети могут автономно «изучать» статические и динамические свойства управляемого объекта на основе результатов измерений, производившихся в прошлом, а затем действовать таким образом, чтобы принять лучшее решение при неизвестном состоянии внешней среды. Обычные компьютеры должны быть предварительно запрограммированы, чтобы иметь возможность обрабатывать данные; они не могут работать за пределами решений, задаваемых программой. Таким образом, инженерия знаний не может быть в полной мере реализована на обычных компьютерах, так как они не могут принимать решения в новых внешних условиях [1].

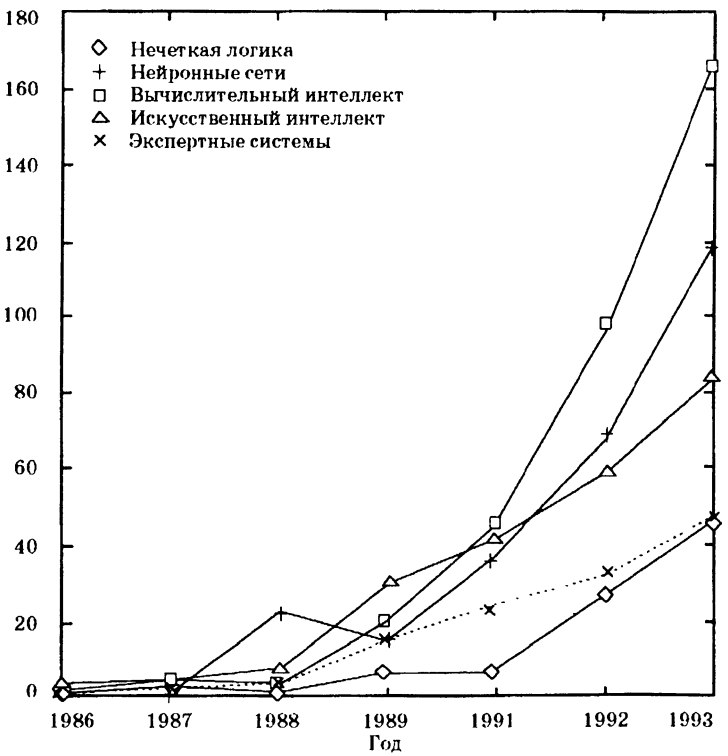


Рис. 1.1.2. Количество патентов, зарегистрированных в базе данных GASSIS [5] (© 1994 IEEE Press)

Нейронные схемы управления — это схемы управления, в которых используется архитектура нейронных сетей и способности к обучению. Нейронная сеть состоит из нейроноподобных вычислительных элементов, которые являются нелинейными преобразователями. Такие свойства нейронных сетей делают возможным нелинейное преобразование данных, что, в свою очередь, позволяет реализовать новые нелинейные схемы управления. На практике могут быть реализованы различные виды схем нейронного управления для решения некоторых задач управления, являющихся пока неразрешимыми. В настоящее время нейронному управлению уделяется большое внимание. Одна из причин такого явления заключается в том, что традиционные методы управления в основном опираются на теорию линейных систем, в то время как реальные объекты управления являются по своей природе нелинейными. Проектировщики оборудования разрабатывают системы не с той или иной точки зрения теории управления, а с позиции осуществимости проекта. Поэтому специалисты по управлению должны вырабатывать стратегию управления так, чтобы достигать максимальной эффективности при многих ограничениях, действующих на реальном объекте управления.

В действительности часто сложно сделать работу оборудования полностью автоматической; управление должно осуществляться людьми-операторами. Даже если будет разработана сложная схема управления, структура и принципы работы которой будут недоступны для понимания оператором оборудования, надежность и безопасность такой схемы окажутся низкими, так как в процессе реальной работы возможны непредсказуемые явления. Это может привести к нежелательным потерям сырья, а также к авариям и несчастным случаям. По этой причине широкое распространение получили пропорциональные-интегрально-дифференциальные регуляторы (PID-контроллеры). Факторами, обусловившими широкое распространение последних, стали простота

их структуры и высокая надежность. По последним данным [6], на 84% японских предприятий все еще используются обычные PID-контроллеры.

В контроллерах этого типа оператор может управлять тремя параметрами, добиваясь улучшения показателей качества. Благодаря своей простоте и надежности PID-контроллеры являются подходящими для промышленного управления. Однако эти контроллеры имеют ряд недостатков. Например, при изменении рабочих точек требуется перенастройка контроллеров. На крупных предприятиях с непрерывным режимом работы при использовании таких контроллеров необходим тщательный контроль, что требует большой численности персонала.

Таким образом, для лучшего решения задач управления требуется разработка новых схем управления. Кроме того, новые схемы управления должны быть достаточно просты по принципам организации и функционированию, чтобы промышленность восприняла их легко и в широком масштабе. По-видимому, может оказаться сложным разработать схемы управления для крупномасштабных систем, обеспечивающие высокую эффективность и при этом использующие только простые принципы управления. Однако желательно, чтобы в новых системах управления манипуляция параметрами была сведена к минимуму. В связи с этим, важным фактором разработки интеллектуальных систем управления является простой, но эффективный человеко-машинный интерфейс.

В инженерном контексте интеллектуальное управление должно обладать следующими свойствами: a) способность к обучению и адаптивность; b) живучесть; c) простой алгоритм управления и «дружественный к пользователю» человеко-машинный интерфейс; d) способность к включению новых компонентов, обеспечивающих лучшие решения в условиях ограничений, накладываемых техническими средствами. Сейчас имеется ряд методов, которые могут использоваться в качестве основы для разработки интеллектуальных систем. Наиболее распространенные из них — экспертные системы (систе-

мы, основанные на знаниях), нечеткая логика, генетические алгоритмы, нейронные сети, искусственная жизнь. Для разработки интеллектуальных систем управления методы искусственного интеллекта (ИИ) должны быть объединены с достижениями современной теории управления.

В этой книге рассматривается концепция нейронных сетей и их применение в системах управления. Рассмотрение нейронных систем управления здесь основывается в основном на алгоритме обратного распространения (backpropagation algorithm). Этот алгоритм широко и успешно применяется во многих инженерных приложениях. Именно поэтому он выбран авторами в качестве алгоритма обучения нейронных сетей. Сначала излагается концепция нейронного управления, затем — различные типы парадигм нейронного управления. Для демонстрации эффективности нейронных систем управления приводятся данные о сравнении их с традиционными подходами к управлению. Традиционные подходы (PID-контроллеры, самонастраивающиеся контроллеры, обобщенные прогнозирующие контроллеры, а также контроллеры на основе нечеткой логики) рассматриваются в следующей главе. Их можно сравнить с нейронными схемами управления, применяемыми к тем же системам управления; такое применение рассматривается в последней главе.

2. НЕЙРОННЫЕ СЕТИ

2.1. Исторические сведения

Нейронные сети — это сети нервных клеток (нейронов) в мозге. Человеческий мозг содержит миллиарды нейронов и триллионы связей между ними. Нейроны непрерывно обрабатывают информацию и обмениваются ею друг с другом. В 1909 г. Кайял (Cajal) [1], [2] обнаружил, что мозг состоит из огромного количества нейронов, сильно связанных друг с другом, способных обмениваться очень простыми возбуждающими и тормозящими сигналами и изменять свое состояние под действием этих сигналов. На рис. 2.1.1 показана клетка Пуркинье (Purkinje) с закрашенным дендритом [2]. Нейрон состоит из трех основных частей: тела клетки (сомы), аксона и дендритов (рис. 2.1.2 [2]).

Сома выполняет опорные функции и создает структуру клетки. Аксон представляет собой разветвленное волокно, передающее сигналы от нейрона. Дендриты также состоят из разветвленных волокон, принимающих сигналы от других нервных клеток. Исходящие сигналы представляют собой импульсы, инициируемые в бугре аксона, там, где аксон соединяется с сомой. На рис. 2.1.3 [2] показана схема синапса, включающая пресинаптическую клетку, ее синаптические пузырьки, синаптическую щель и постсинаптическую клетку.

Конечный выступ находится на конце ветви аксона, как показано на рис. 2.1.2, и имеет сложную структуру. Она выделяет химические трансмиттеры (передатчики), которые, пересекая синаптическую щель, достигают нейрона-приемника. Химические трансмиттеры действуют на мембрану нейрона-приемника, усиливая или ослабляя его тенденцию к возбуждению собственного импульса в зависимости от того, каким был входной сигнал: возбуждающим или тормозящим. Так как «синапс» — основное понятие для искусст-

венных нейронных сетей, рассмотрим его подробнее. На рис. 2.1.3 показана схема синапса, включающая предсинаптическую клетку, синаптические пузырьки, синаптическую щель и постсинаптическую клетку. В предсинаптической клетке входные сигналы действуют на мембрану в предсинаптической области, вызывая выделение трансмиттера из синаптических пузырьков. Трансмиттер пересекает синаптическую щель и достигает постсинаптической клетки, создающей мембранные свойства клетки-приемника. Интенсивность возбуждения постсинаптической клетки усиливается или ослабевает.

На конце аксонного отростка предсинаптической клетки имеется выступ. Поступление действующего потенциала воз-

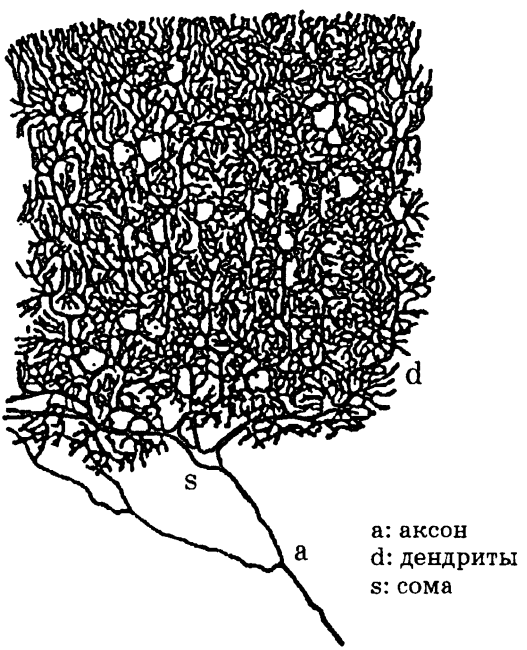


Рис. 2.1.1. Клетка Пуркинье (из мозжечка). Дендриты, образующие плоскую веерную структуру, показаны закрашенными [2]

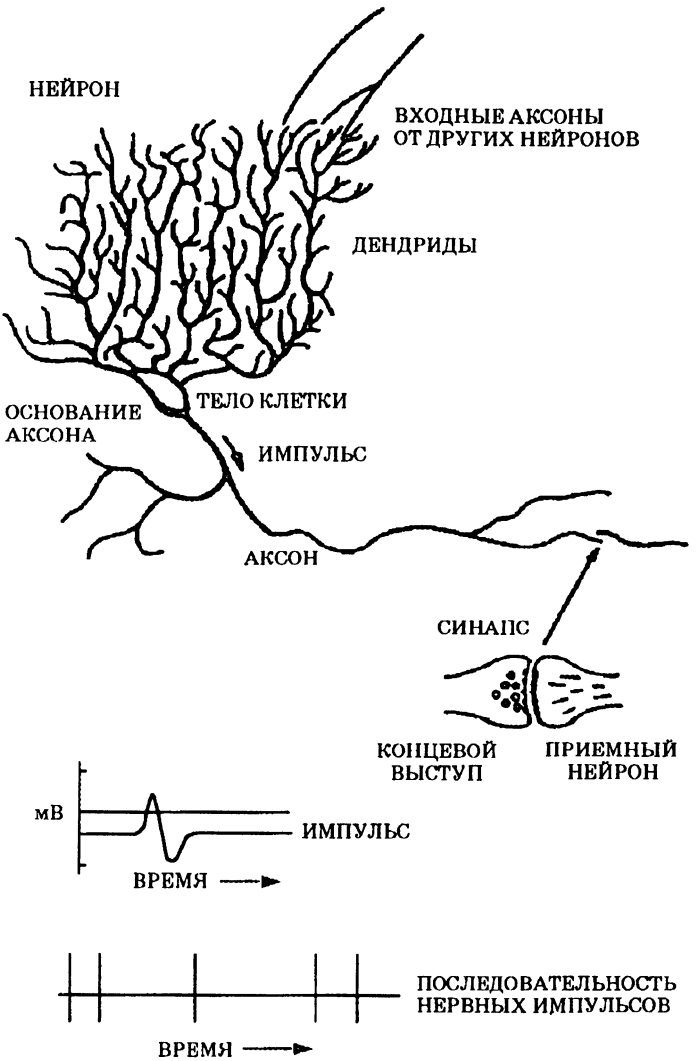


Рис. 2.1.2. Запись последовательности нервных импульсов с помощью микроэлектрода, размещенного вблизи аксона. Показана записанная на осциллографе форма сигнала, полученного от микроэлектрода [2]

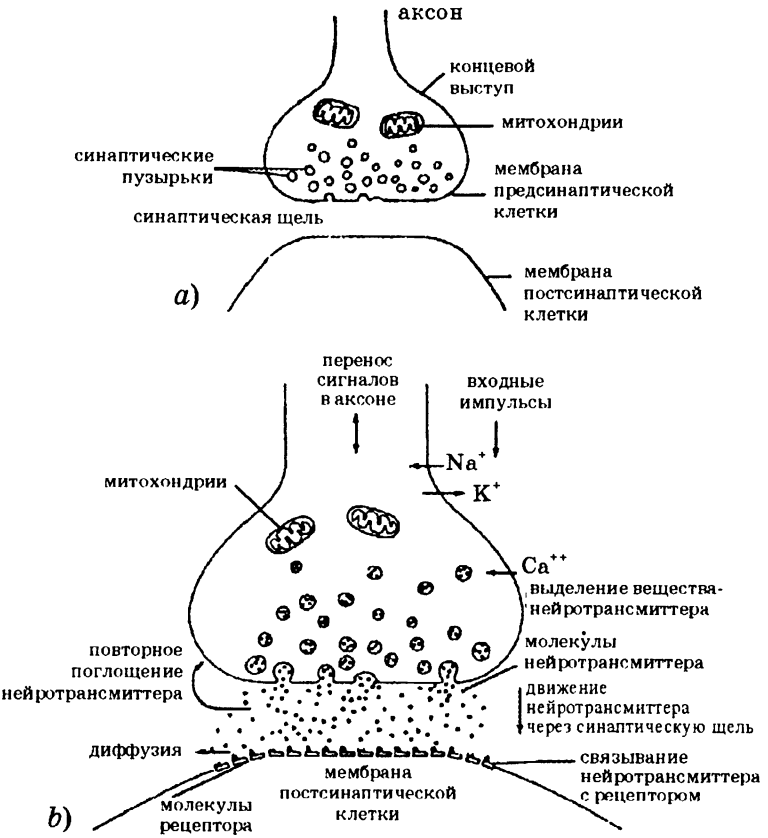


Рис. 2.1.3. (*a*) Анатомические структуры синапса. (*b*) Динамические процессы в синапсе [2].

действует на мембрану предсинаптической клетки в окончании выступа. В ответ на это воздействие мембрана пропускает поток ионов кальция (Ca^{++}), после чего из синаптических пузырьков в щель выделяется трансмиттер. Выделение трансмиттера модулируется ионами кальция. Таким образом, ионы кальция

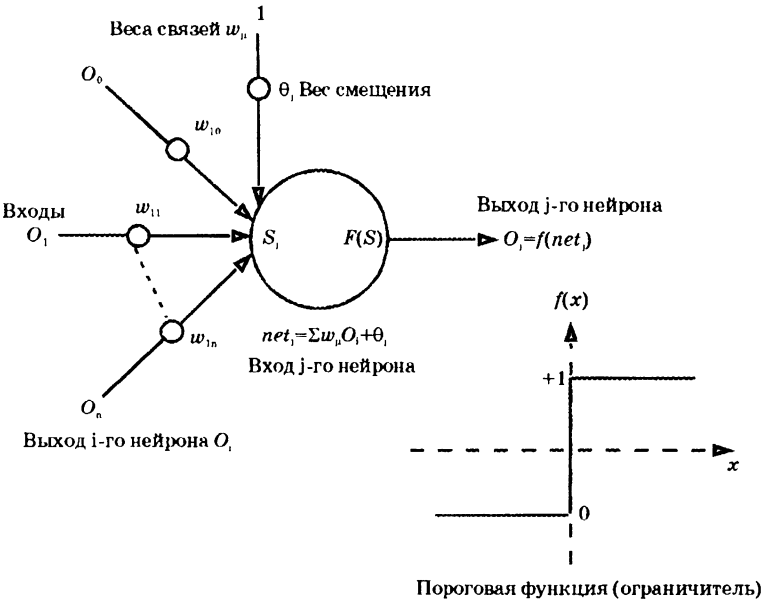


Рис. 2.1.4. Схематическая модель нейрона
(по Маккаллоку и Питтсу)

участвуют в механизме представления кандидатов в процессе обучения, модулируя синаптическую силу. Авторы данной книги не намерены подробно рассматривать биологические нейронные структуры: синаптические пузырьки, вещества — трансмиттеры, постсинаптические клетки и т.д. Подробные сведения об этом приводятся в работе [2].

В 1943 г. Маккаллок и Питтс (McC uloch and Pitts) предложили математическую модель нейронов и показали способы расчета нейроноподобных сетей. Модель нейрона по Маккаллоку и Питтсу приведена на рис. 2.1.4.

Математическая модель нейрона, предложенная Маккаллоком и Питтсом, имеет вид:

$$\mathrm{net}_j = \sum_i w_{ji} O_i + \theta_j, \qquad (2.1.1)$$

$$O_j = f(\mathrm{net}_j), \qquad (2.1.2)$$

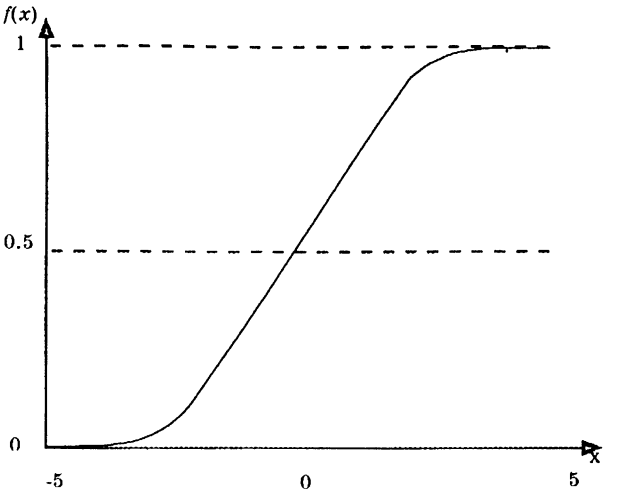


Рис. 2.1.5. Сигмоидная функция активации

где net_j — входной сигнал сети для j-го нейрона, θ_j — пороговая величина, W_{ji} — сила синаптической связи i-го нейрона с j-м, O_i — выходной сигнал i-го нейрона, f(x) — функция выхода, или функция активации нейрона. Для простоты будем использовать следующие типы функции активации:

(i) ограничитель, функция знака или сигнум-функция:

$$f(x) = \mathrm{sign}(x) = \begin{cases} 1 & \text{при } x \geq 0 \\ 0 & \text{при } x < 0 \end{cases},$$

(ii) линейная функция: $f(x) = x$,

(iii) сигмоидная функция:

$$f(x) = \mathrm{sigmoid}(x) = \frac{1}{1 + \exp(-\frac{x}{T})}, \quad T>0$$

Необходимо учитывать, что при $T \to \infty$ сигмоидная функция f(x) переходит в линейную функцию, а при $T \to 0$ — в функцию знака.

Впервые идея обучения в нейронных сетях нашла отражение в книге Хебба (Hebb) [4] «Организация поведения», вышедшей в 1949 г. До этой работы считалось, что для обеспечения обучения в нейронной сети должны происходить какие-то физические изменения, однако было неизвестно, какие именно. Хебб предположил, что существенное биологическое изменение потребуется для укрепления связей между элементами нейронной сети только в том случае, если предсинаптическая и постсинаптическая мембраны активизируются одновременно.

Сущность идей Хебба проявляется в различных парадигмах обучения. Хотя отдельные детали правил изменения весов могут быть разными, основное утверждение Хебба о том, что сила связей между элементами должна изменяться в определенной зависимости от коррелированной активности связанных элементов, была принята во многих моделях обучения.

В 1951 г. Эдмондс (Edmonds) и Минский (Minsky) разработали свою обучающуюся машину на основе идеи Хебба [5]. Хотя Минский первым предложил идею обучающейся машины, фактическим началом разработок в области обучения нейроноподобных сетей стала работа Розенблатта (Rosenblatt), вышедшая в 1962 г. [5]. Розенблатт предложил класс простых нейроноподобных сетей, названный им персептроном. Персептрон, действительно, представлял собой целый класс структур, состоящий из обрабатывающих элементов, способных передавать сигналы и изменять веса своих связей. Исследования Розенблатта были направлены прежде всего на моделирование мозга в попытке понять механизмы работы памяти, обучения, когнитивные процессы. Работы Розенблатта были продолжены и развиты многими учеными и инженерами; был создан ряд машин на основе архитектуры персептрона. Выполнялись работы по математическому анализу вычислительной мощности персептронов (с целью выявления пределов их возможностей). Ослабление интереса к использова-

нию нейроноподобных сетей в качестве вычислительных устройств наступило после выхода книги Минского и Пейперта (Papert) [7].

В 1982 г. Хопфилд (Hopfield) выдвинул идею архитектуры нейронной сети, получившей название сети Хопфилда [8]. Им было показано, как могут строиться вычислительные средства на основе нейронных сетей. Хопфилдом была показана также реализация ассоциативной памяти на основе нейронной сети, а позднее — продемонстрирована возможность использования нейронных сетей для решения задач оптимизации, например, задачи о коммивояжере [9].

Другой подход, широко применявшийся в парадигмах нейронных сетей — обратное распространение ошибок, или обратное распространение (backpropagation). Концептуальные основы этого подхода были предложены Вербосом (Werbos) в 1974 г. [10], а затем, независимо от него — Румельхартом (Rumelhart) и др. в 1986 г. [11]. В книге последнего «Параллельные распределенные процессы» были показаны широкие перспективы для подходов, основанных на нейронных сетях.

Из других разработок, выполненных в последнее время, следует назвать машину Больцмана, конкурентное обучение и карту характеристик Кохонена (Kohonen). Машина Больцмана разработана Хинтоном (Hinton) и Сейновски (Sejnowski) на основе термодинамических моделей [11]. В этих моделях используются бинарные элементы, принимающие значения 0 и 1. Эти значения определяются стохастической функцией входов. При предположениях, применявшихся к сети Хопфилда, можно доказать, что сеть переходит в состояния, в наибольшей степени соответствующие входам и внутренним ограничениям, задаваемым весами.

Парадигма конкурентного обучения использует слой обрабатывающих элементов, конкурирующих друг с другом, в результате чего образуется нейронная сеть, применимая для решения задач распознавания образов. Такой подход был рассмотрен Румельхартом (Rumelhart) и Зипсером (Zipser) [12] и

Гроссбергом (Grossberg) [13]. Самоорганизующаяся матрица характеристик Кохонена представляет собой двухслойную сеть, которая может организовать топологическую карту из случайной начальной точки. Полученная таким образом карта отражает естественные отношения между образами, предъявленными сети. Эта парадигма предложена Кохоненом [14]. Результаты, предложенные в данной книге, основываются главным образом на алгоритме обратного распространения. Рассмотрим его в следующем разделе.

2.2. Алгоритм обратного распространения

Алгоритм обратного распространения (backpropagation algorithm) представляет собой расширение концепции персептрона или АДАЛИНА [15] на многослойные нейронные сети. В алгоритме обратного распространения используется не менее трех слоев обрабатывающих элементов (нейронов). Архитектура типичной трехслойной сети для алгоритма обратного распространения показана на рис. 2.2.1. Левый слой элементов — это входной слой, на который подаются входные данные. За ним следует скрытый слой, в котором обрабатывающие элементы связаны с предшествующим и последующим слоями. Правый слой — выходной. Слои на рис. 2.2.1 являются полностью взаимосвязанными; это значит, что каждый обрабатывающий элемент связан с каждым элементом в предыдущем и следующем слоях. При этом элементы не связаны с другими элементами того же слоя. Следует обратить внимание на то, что сети обратного распространения не обязательно являются полностью связанными, а количество скрытых слоев может быть любым.

2.2.1. Обозначения

Для простоты будем рассматривать трехслойные нейронные сети, аналогичные показанной на рис. 2.2.1, где O_k, O_j и O_i — выходные величины выходного, скрытого и входного слоев соответственно. Вес связи между j-м элементом скрытого слоя и

k-м элементом выходного слоя будем обозначать как W_{kj}. Аналогично вес связи между i-м элементом входного слоя и j-м элементом скрытого слоя будем обозначать как W_{ji}. Таким образом, на выходном слое

$$O_k = f(net_k), \tag{2.2.1}$$

$$net_k = \sum_j w_{kj} O_j + \theta_k, \tag{2.2.2}$$

а на скрытом слое

$$O_j = f(net_j), \tag{2.2.3}$$

$$net_j = \sum_i w_{ji} O_i + \theta_j, \tag{2.2.4}$$

где

$$f(x) = \frac{1}{1 + \exp(-x)}, \tag{2.2.5}$$

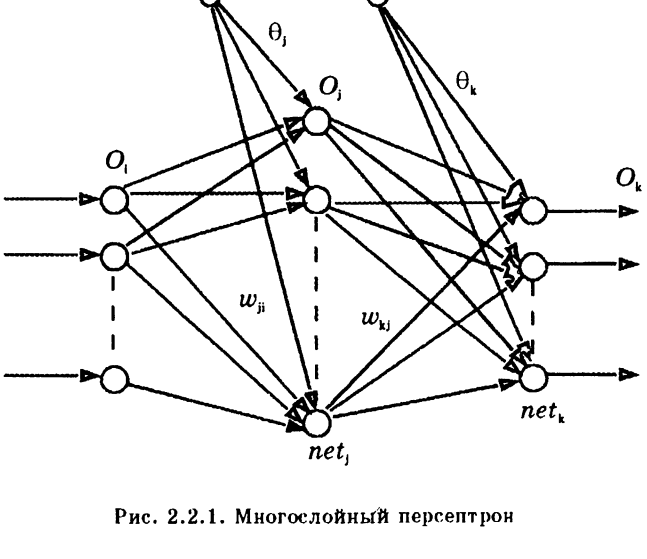


Рис. 2.2.1. Многослойный персептрон

Процедура обучения включает представление набора пар входных и выходных образов. Сначала нейронная сеть на основе входного образа создает собственный выходной образ, а затем сравнивает его с желаемым (целевым) выходным образом. Если различий между фактическим и целевым образом нет, то обучения не происходит. В противном случае веса связей изменяются таким образом, чтобы различие уменьшилось. Мерой различия, используемой в рассматриваемом примере, является функция квадрата ошибки E, задаваемая следующим образом:

$$E = \sum_p E_p, \tag{2.2.6}$$

$$E_p = \frac{1}{2}\sum_k (\tau_{pk} - O_{pk})^2, \tag{2.2.7}$$

где — τ_{pk} желаемый выход для k-й компоненты p-го выходного образа и O_{pk} — соответствующий фактический выход. Таким образом, E_p — мера ошибки вход/выход для p-го образа, а E — общая мера ошибки.

2.2.2. Вывод алгоритма обратного распространения

Вывод строится на основе одного из градиентных методов, например метода наискорейшего спуска. Здесь показан процесс адаптации весов связей между слоями в многослойной нейронной сети. Зададим сначала функцию ошибки и критерий адаптации весов, пропорциональный отношению производной меры ошибки к каждому из весов. Получим сначала алгоритм обратного распространения для конкретного образа p, т.е. найдем минимизирующую последовательность E_p. Для простоты будем опускать индекс p.

Тогда

$$E_p = \frac{1}{2}\sum_k (\tau_k - O_k)^2. \tag{2.2.8}$$

2.2.2.1. Правило для выходного слоя

В соответствии с методом наискорейшего спуска

$$\Delta w_{kj} = -\frac{\partial E_p}{\partial w_{kj}},$$

где Δw_{kj} — необходимое изменение веса связи w_{kj}, т.е.

$$\Delta w_{kj} = w_{kj}(new) - w_{kj}(old). \qquad (2.2.9)$$

Такое изменение показано на рис. 2.2.2. Следует обратить внимание, что метод наискорейшего спуска может сходиться к локальному минимуму, как показано на рис. 2.2.3. Поэтому возьмем

$$\Delta w_{kj} = -\eta \frac{\partial E_p}{\partial w_{kj}}, \eta > 0, \qquad (2.2.10)$$

где η — скорость обучения.

Введем величину обобщенного сигнала ошибки δ_k, задаваемую выражением

$$\delta_k = -\frac{\partial E_p}{\partial net_k}. \qquad (2.2.11)$$

Затем, используя известное цепное правило, получим:

$$\frac{\partial E_p}{\partial w_{kj}} = \frac{\partial E_p}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}}. \qquad (2.2.12)$$

Это означает, что выражение $\frac{\partial E_p}{\partial w_{kj}}$ может быть представлено как произведение двух членов. Один из них представляет собой изменение ошибки в зависимости от изменения входного сигнала сети, а второй — влияние конкретного веса связи на входной сигнал сети. Из выражения (2.2.2) получим:

$$\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial\left(\sum_j w_{kj} O_j + \theta_k\right)}{\partial w_{kj}} = O_j,$$

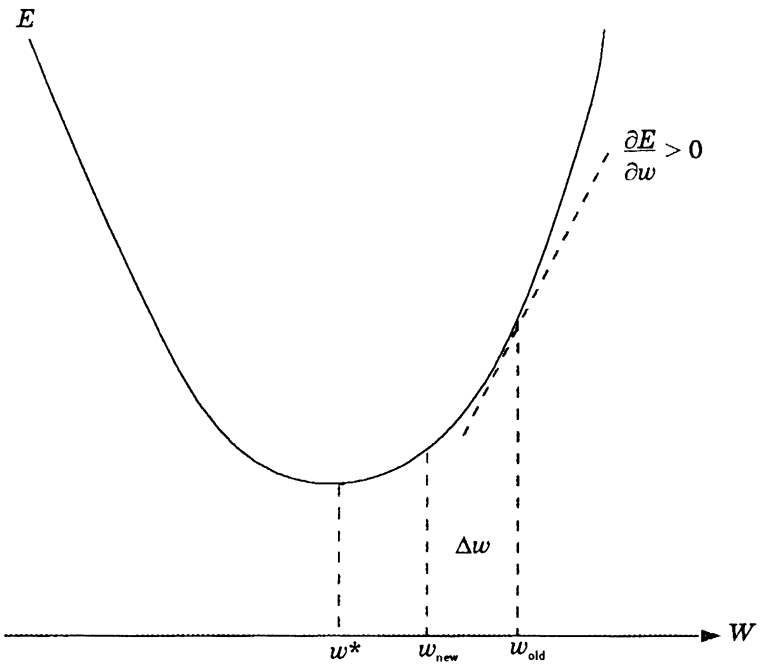


Рис. 2.2.2. Принцип работы метода наискорейшего спуска

откуда

$$\Delta w_{kj} = \eta \delta_k O_j. \qquad (2.2.13)$$

Для вычисления множителя d_k применим цепное правило. Используя его, получим:

$$\delta_k = -\frac{\partial E_p}{\partial net_k} = -\frac{\partial E_p}{\partial O_k}\frac{\partial O_k}{\partial net_k}. \qquad (2.2.14)$$

Таким образом, выражение δ_k представляет собой произведение двух членов. Первый из них отражает изменение ошибки по отношению к выходу элемента, а второй — изменение выхода по отношению к изменению входа сети. Из уравнения (2.2.8) получим:

$$\frac{\partial E_p}{\partial O_k} = -(\tau_k - O_k), \tag{2.2.15}$$

а из выражения (2.2.1) -

$$\frac{\partial O_k}{\partial net_k} = f'(net_k), \tag{2.2.16}$$

где f'(x) — производная от f(x) по x.

Из выражения (2.2.5) получим:

$$f'(x) = \frac{e^{-x}}{(1-e^{-x})^2} = \frac{1}{(1-e^{-x})}\left(1 - \frac{1}{1-e^{-x}}\right) = \\ = f(x)(1-f(x)), \tag{2.2.17}$$

откуда

$$\frac{\partial O_k}{\partial net_k} = O_k(1-O_k). \tag{2.2.18}$$

Таким образом, имеем:

$$\Delta w_{kj} = \eta \delta_k O_j,$$

$$\delta_k = O_k(1-O_k)(\tau_k - O_k). \tag{2.2.19}$$

2.2.2.2. Правило для скрытого слоя

Как показано в предыдущем разделе, правило изменения весов связей для скрытого слоя в соответствии с методом наискорейшего спуска имеет вид:

$$\Delta w_{ji} = -\eta \frac{\partial E_p}{\partial w_{ji}}. \tag{2.2.20}$$

Используя цепное правило, как и раньше, получим:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}. \tag{2.2.21}$$

Зададим множитель δ_j следующей формулой:

$$\delta_j = -\frac{\partial E_p}{\partial net_j}. \qquad (2.2.22)$$

Из уравнения (2.2.4) получим:

$$\frac{\partial net_j}{\partial w_{ji}} = O_i$$

и, таким образом,

$$\Delta w_{ji} = \eta \delta_j O_i. \qquad (2.2.23)$$

Используя цепное правило, получим:

$$\delta_j = -\frac{\partial E_p}{\partial net_j} = -\sum_k \frac{\partial E_p}{\partial net_k} \frac{\partial net_k}{\partial O_j} \frac{\partial O_j}{\partial net_j}. \qquad (2.2.24)$$

Заметим, что изменение величины net_j влияет на все выходы O_k, поэтому в приведенном выше цепном правиле требуется суммирование по k.

Используя выражения (2.2.11), (2.2.2), (2.2.3) и (2.2.5), получим:

$$\delta_j = \sum_k \delta_k w_{kj} f'(net_j) = O_j(1 - O_j)\sum_k \delta_k w_{kj}. \qquad (2.2.25)$$

Таким образом, получен следующий алгоритм изменения весов связей:

$$\Delta w_{kj} = \eta \delta_k O_j, \qquad (2.2.26)$$

$$\Delta w_{ji} = \eta \delta_j O_i, \qquad (2.2.27)$$

$$\delta_k = O_k(1 - O_k)(\tau_k - O_k), \qquad (2.2.28)$$

$$\delta_j = O_j(1 - O_j)\sum_k \delta_k w_{kj}. \qquad (2.2.29)$$

Чтобы повысить скорость обучения, не вызывая при этом колебаний в выходном сигнале, изменим уравнения (2.2.6) и (2.2.7), включив в них момент времени. Это можно сделать по следующему правилу:

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t), \qquad (2.2.30)$$

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t), \qquad (2.2.31)$$

где t — время представления, h — скорость обучения, α — константа, определяющая влияние предыдущего изменения весов на текущее направление движения в пространстве весов связей.

Рассматривая эти выражения с выражениями:

$$\frac{\partial E}{\partial w_{kj}} = \sum_p \frac{\partial E_p}{\partial w_{kj}} \quad \text{или} \quad \frac{\partial E}{\partial w_{ji}} = \sum_p \frac{\partial E_p}{\partial w_{ji}}, \tag{2.2.32}$$

приходим к выводу, что изменение весов связей после одного полного цикла предъявления образов пропорционально вышеприведенной производной. Это означает, что алгоритм обратного распространения, действительно, обеспечивает наискорейшее уменьшение величины E. Конечно, данный вывод является строгим только при условии, что величины весов связей не изменяются во время цикла.

2.2.3. Алгоритм обратного распространения: порядок реализации

Алгоритм обратного распространения может быть представлен в виде следующих шагов.

Шаг 1. Присвоить начальные значения величинам W_{ji}, W_{kj}, θ_k, θ_j, η и α.

Шаг 2. Подать на нейронную сеть входной сигнал. Задать соответствующий ему желаемый выход τ_k, вычислить O_k, O_j и δ_k по формуле:

$$\delta_k = O_k(1 - O_k)(\tau_k - O_k). \tag{2.2.33}$$

Шаг 3. Изменить веса связей на величину:

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t). \tag{2.2.34}$$

Шаг 4. Вычислить δ_j по формуле:

$$\delta_j = O_j(1 - O_j)\sum_k \delta_k w_{kj}. \tag{2.2.35}$$

Шаг 5. Изменить веса связей на величину:

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t). \tag{2.2.36}$$

Шаг 6. Положить $t \to t+1$ и перейти к шагу 2.

2.2.4. Обсуждение алгоритма обратного распространения

Алгоритм обратного распространения, полученный на основе градиентного метода наискорейшего спуска, предназначен для поиска минимума ошибки, что приводит, однако, к поиску локального минимума или точки перегиба (плато), как показано на рис. 2.2.3. В идеальном случае должен отыскиваться глобальный минимум, представляющий собой нижнюю точку на всей области определения. Так как локальный минимум окружен более высокими точками, стандартный алгоритм обратного распространения обычно не пропускает локальных минимумов.

Вид локального минимума не всегда представляет существенный интерес. Можно ожидать, что в ходе обучения в соответствии с алгоритмом обратного распространения ошибка сети будет сходиться к приемлемому значению, если обучающие образцы четко различимы. Чтобы избежать «ловушек» на локальных минимумах, можно изменять параметры обучения, количество скрытых слоев, начальные значения весов связей. Обычно эти методы разрабатываются так, чтобы попытаться изменить сценарий перемещения по участку, содержащему локальные минимумы и максимумы. Для поиска достаточно приемлемых значений параметров алгоритма обратного распространения в последнее время используется генетический алгоритм.

Для вывода алгоритма обратного распространения, представленного в разделе 2.2.2, использована сигмоидная функция (2.2.5). Могут использоваться и другие функции активации. Если, например, использовать функцию вида:

$$f(x) = \frac{1}{1 + \exp(-\frac{x}{T})}, \qquad (2.2.37)$$

то получим:

$$f'(x) = \frac{1}{T} f(x)(1 - f(x)). \qquad (2.2.38)$$

Таким образом, потребовалось только заменить выражения δ_k и δ_j, полученные по формулам (2.2.33) и (2.2.35), на выражения $\frac{\delta_k}{T}$ и $\frac{\delta_j}{T}$ соответственно. Если выбрать меньшее значение T, то δ_k и δ_j увеличатся, и в результате скорость изменения весов связей также увеличится. Однако если выбрать большее значение T, то δ_k и δ_j уменьшатся, и в результате скорость изменения весов связей также уменьшится. Таким образом, изменяя величину T в процессе обучения, можно управлять скоростью сходимости алгоритма обратного распространения.

Следует отметить, что в практической ситуации обучения, когда желаемые выходы являются бинарными (т.е. представляют собой значения из множества {0,1}), система достигает этих значений выхода только при условии, что отношение x/T стремится к бесконечности. Таким образом, если выбрать меньшее значение T, то сходимость выхода сети будет иметь тенден-

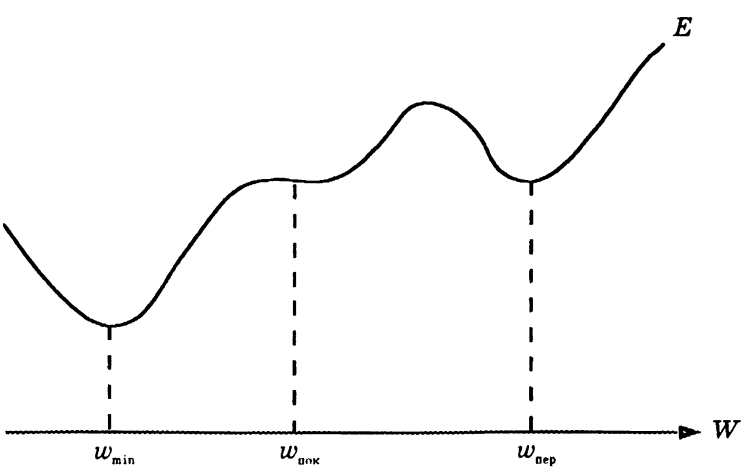


Рис. 2.2.3. Локальный минимум в алгоритме обратного распространения

цию к ускорению. При T=1 производные $O_k(1-O_k)$ и $O_j(1-O_j)$ достигают максимальных значений при $O_k = O_j = 0.5$. В связи с тем, что $0 \le O_k, O_j \le 1$, минимумы производных достигаются при приближении этих величин к нулевому или единичному значениям. Так как величина изменения конкретного веса связи пропорциональна производной, эти веса будут изменяться в основном для этих значений.

Другое свойство функции активации состоит в том, что нейронная сеть не может достигать предельных значений 0 и 1, если веса связей не являются бесконечно большими. Таким образом, в практической ситуации обучения, когда желаемые выходы являются бинарными ({0,1}), нейронная сеть никогда не достигнет этих значений. Поэтому в качестве желаемых величин можно рассматривать 0,1 и 0,9, даже если в действительности желаемыми величинами являются 0 и 1.

Скорость обучения h — это константа, представляющая собой коэффициент пропорциональности между изменением веса связи и градиентом ошибки E относительно веса. Чем больше данная константа, тем больше изменения в весах связей. Обычно скорость обучения выбирается как можно большей, но такой, чтобы не возникало осцилляции. Так как для исключения осцилляции в формулы был введен момент времени, требуется выбрать подходящее значение α. Введение момента времени «отфильтровывает» высокочастотные изменения поверхности ошибки в пространстве весов связей. Это полезно в случаях, когда пространство весов связей содержит длинные овраги с острыми искривлениями. В связи с тем, что такое искривление вызывает резкие расходящиеся колебания в долине, необходимо выбирать небольшой размер шага, для чего требуется небольшая скорость обучения. В книге Румельхарта (Rumelhart) и др. [11] рекомендуется использовать величину α, приблизительно равную 0,9.

Другой проблемой является выбор количества скрытых узлов или скрытых слоев. Из регрессионного анализа известно, что если выбрать достаточно много видов описывающих пере-

менных, то ошибка предсказания окажется небольшой. Пусть, например, для оценки линейной регрессии

$$y_k = a_0 + a_1 x_{1k} + ... + a_p x_{pk} + e_p, \qquad (2.2.39)$$

где k=1,2,...,N, выбрано p видов описывающих переменных $X_1, X_2, ..., X_p$ и одна целевая переменная y.

Если выбрать N=p, то ошибки предсказания e_k (k = 1, 2, ..., p) могут быть равными нулю. Может показаться, что с помощью уравнения регрессии можно достичь точного предсказания. На практике это невозможно, так как N должно быть больше p. Поэтому необходимо отметить, что минимизация квадрата ошибки, достигаемая за счет увеличения количества нейронов, не означает построения хорошей нейронной сети. Если использовать большое количество нейронов, то нейронная сеть может с высокой точностью обучаться на «зашумленных» данных. В этом случае следует выбрать нейронную сеть

Рис. 2.2.4. Явление избыточного подбора

меньшего размера, даже если ошибка при этом оказывается не такой малой, как показано на рис. 2.2.4. На этом рисунке $\hat{E}$ представляет кривую ошибки, полученную при точном обучении на наблюдаемых данных, а E^0 — кривую ошибки, обеспечивающую наименьшую ошибку предсказания для невидимых данных. Это явление называется проблемой избыточного подбора. Для ее решения предложено несколько методов, которые позволяют уменьшить размер нейронной сети.

2.3. Заключение

В данной главе в упрощенном виде рассмотрены структура и функции биологической нейронной сети. Можно видеть, что в терминологии, моделях и функциях, используемых для искусственных нейронных сетей, многое заимствовано из биологических нейронных сетей. В то же время обучение в искусственных нейронных сетях во многом отличается от соответствующих процессов в биологических сетях. Искусственные нейронные сети обучаются на основе упрощенных алгоритмов обучения, адаптирующих синаптические веса; эти алгоритмы могут классифицироваться на алгоритмы управляемого и неуправляемого обучения. В данной главе описан алгоритм обратного распространения — наиболее широко применяемый алгоритм обучения многослойных нейронных сетей. Он составляет основу для схем нейронного управления, рассматриваемых в этой книге. Приведен подробный вывод данного алгоритма.

3. ТРАДИЦИОННЫЕ СХЕМЫ УПРАВЛЕНИЯ

3.1. Введение

В данной главе рассматриваются некоторые другие методы управления: ПИД-управление (пропорционально-интегрально-дифференциальное), управление с самонастройкой, ПИД-управление с самонастройкой, обобщенное прогнозирующее управление, управление на основе нечеткой логики. Одними из первых систем управления были ПИ- и ПИД-контроллеры. Они доказали свою эффективность в управлении разнообразными процессами. Использование ПИ- и ПИД-контроллеров не требует знания точной модели процесса, поэтому они эффективны в управлении промышленными процессами, математические модели которых достаточно сложно определить. ПИ- и ПИД-контроллеры строятся на основе классической теории управления и просты для понимания. Установление связей между параметрами и управление действиями системы могут осуществляться инженерами-практиками и операторами.

Кроме того, за последние десятилетия разработано несколько методов настройки ПИ- и ПИД-контроллеров. Зиглер (Ziegler) и Никольс (Nichols) [1] предложили метод настройки, основанный на данных о реакции на единичное ступенчатое воздействие. Он широко использовался в течение многих лет и оказался достаточно эффективным. На его основе Такахаши (Takahashi) и др. [2] предложили метод настройки для ПИ- и ПИД-контроллеров с дискретным временем. Более современный метод автоматической настройки ПИ- и ПИД-контроллеров был предложен Нишикавой (Nishikawa) и др. [3]. Он требует подачи входного испытательного сигнала для оценки параметров процесса. Оптимальные значения ПИ- и ПИД-

параметров достигаются путем минимизации взвешенного интеграла квадратичной ошибки. Другой метод автоматической настройки ПИД-контроллеров был предложен Хаглундом (Hagglund) и Остремом (Åström) [4]. Он базируется на формуле замкнутого цикла, приведенной в [1].

Однако, наряду с вышеуказанными достоинствами, ПИ- и ПИД- контроллеры имеют и ряд недостатков. Так, если рабочая точка процесса изменяется из-за возмущений, параметры контроллера требуется перенастраивать вручную, чтобы получить новую оптимальную настройку. Настройка должна выполняться опытным оператором. Для систем с взаимодействующими контурами эта процедура может быть сложной и занимать много времени. Кроме того, для процессов с переменными параметрами, временными задержками, существенными нелинейностями и значительными помехами использование ПИ- и ПИД-контроллеров может не обеспечить оптимальных характеристик. Методы настройки ПИ- и ПИД-контроллеров также имеют ряд недостатков. Например, метод Зиглера-Никольса (Ziegler — Nichols) [1] чувствителен к возмущениям, так как разработан на основе экспериментов с разомкнутыми системами. В методе настройки, предложенном Нишикавой (Nishikawa) и др. [3], требуется человеко-машинное взаимодействие, в ходе которого оператор должен генерировать входные сигналы каждый раз, когда требуется изменение параметров с целью адаптации к изменению динамики процесса.

По этим причинам Калман (Kalman) [5] выдвинул идею управления с самонастройкой, в котором параметры контроллера настраивались бы в оперативном режиме. Калман предложил концепцию машины, которая автоматически выполняла бы самонастройку в целях управления произвольным динамическим процессом. Эта идея была важной в общем ходе развития самонастраивающихся средств управления, однако она не получила активной поддержки из-за неадекватности теории и технологии. «Вторым рождением» данной теории,

вызвавшим большой интерес среди инженеров и исследователей в области управления, стало появление работы Острема (Åström) и Виттенмарка (Wittenmark) [6]. Они обнаружили неожиданное явление: в ряде случаев комбинация оценок, полученных по обычному методу наименьших квадратов, и управления по минимальной дисперсии обеспечивает свойство самонастраиваемости. За этой работой последовал ряд практических приложений, которые, в свою очередь, породили новые теоретические разработки, так как проблемы, связанные с реализацией управления с самонастройкой, требовали разъяснения. Быстрое развитие цифровых ЭВМ в 70-е и 80-е годы сыграло важную роль в повышении интереса к управлению с самонастройкой, как в теоретическом, так и в практическом плане.

Исследования и разработки в области самонастраивающихся алгоритмов проводились по двум направлениям: распространение существующих схем управления на новые области применения и выдвижение новых подходов к задачам автоматического регулирования, для которых уже применялись имеющиеся виды контроллеров. Еще одной сферой исследований в этом направлении стали самонастраивающиеся ПИ- и ПИД-контроллеры (СНПИ- и СНПИД-контроллеры). Основная идея разработки контроллеров такого типа состоит в объединении возможностей самонастраивающихся контроллеров, способных в оперативном режиме приспосабливаться к изменениям параметров объекта управления, и простоты структур ПИД-контроллеров.

Предложено несколько схем СНПИД-контроллеров. Так, Виттенмарк (Wittenmark) [7] предложил схему СНПИД-контроллера на основе схемы размещения полюсов. При этом ПИД-алгоритм встраивается в структуру управления, которая рассчитывается на основе схемы размещения полюсов. Этот метод пригоден только для управления процессами ограниченных порядков. Модификация этого метода, позволяющая управлять процессами более общего вида, была предложена

Савелли (Savelli) и др. [8]. Позднее Гаутроп (Gawthrop) [9] предложил схему СНПИ- и СНПИД-контроллера для систем с непрерывным временем. В этой схеме ПИ- или ПИД-алгоритм автоматически выводится из обоснованных предположений о динамике управляемого процесса, а также из результатов моделирования возмущений с ненулевым математическим ожиданием. Иные формы СНПИД-контроллеров приводятся в работах [10—16].

Другая концепция управления, рассматриваемая в данной главе, была предложена Камероном (Cameron) и Себоргом (Seborg) [17]. Предложенный ими алгоритм основан на обобщенном управлении по минимальной дисперсии [18,19]. Закон управления с самонастройкой ориентируется на получение ПИ- или ПИД-структуры. Для этого делаются некоторые обоснованные допущения в отношении полиномиальных функций управления с самонастройкой, задаваемых пользователем. Способ ввода в алгоритм интегрирующего действия, необходимого для устранения ступенчатых возмущений и установившейся ошибки, отличается от используемого в СНПИ- и СНПИД-контроллерах. Параметры контроллеров определяются на основе схемы оценки параметров. Для ПИ-структуры требуются два параметра контроллера, для ПИД-структуры — три. Эти требования налагают некоторые ограничения на порядок полюсов разомкнутых систем управления, однако не на порядок нулей. В то же время можно показать, что для систем высоких порядков может быть осуществлено понижение порядка модели.

Рассматриваются также СНПИД-контроллеры со многими переменными (в разделе 3.3). При построении таких СНПИД-контроллеров учитываются те же проблемы управления, что и для контроллеров с одним входом и одним выходом. К ним относятся нелинейности, переменная динамика объекта управления, детерминированные и стохастические возмущения, временные задержки. Для многомерного случая имеются также некоторые технические сложности, которые не возникают для

контроллеров с одним входом и одним выходом. К ним относятся, в частности, проблемы, вытекающие из хорошо известных разделов теории управления многомерными системами. Например, для контроллеров со многими переменными соотношение «вход-выход» должно представляться в виде левого или правого дробно-матричного описания. Кроме того, полиномиальные матрицы не переставляются; для разрешения этой проблемы требуется вводить некоторые дополнительные матрицы. В результате всего этого усложняются формулировки и вычисления, которые становятся по своему характеру алгебраическими [20]. В разработке СНПИ- и СНПИД-контроллеров со многими переменными имеются и другие проблемы, например, связанные с параметризацией, неравными временными задержками и устойчивостью.

Другой современной разработкой в области адаптивного управления является управление на основе долгосрочного прогнозирования. Принцип этого метода управления показан на рис. 3.1.1. Стратегия метода состоит в том, что в каждый кон-

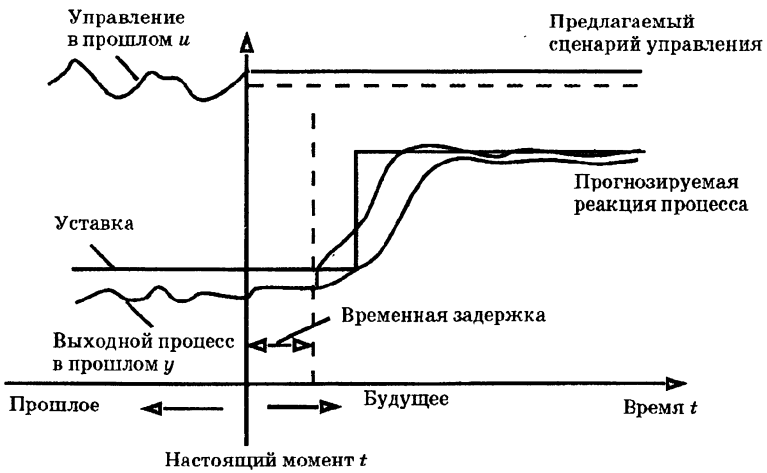


Рис. 3.1.1. Стратегия управления на основе долгосрочного прогнозирования

кретный момент t делается прогноз выхода процесса на длительный временной горизонт. Этот прогноз, основанный на математической модели динамики процесса, делается явно или подразумевается в алгоритме управления.

Кроме всего прочего, прогноз зависит от сценария управления в будущем, которое предполагается постоянным на все время прогнозирования. Из нескольких сценариев управления выбирается стратегия, обеспечивающая сходимость выходного процесса к уставке наилучшим образом. Стратегия, выбранная в качестве наилучшей, применяется затем в качестве управляющего воздействия к реальному процессу, однако только на заданное время. В следующий момент дискретного времени вся процедура повторяется, в результате чего обеспечивается уточненное управляющее действие. Такая стратегия известна под названием стратегии управления по удаляющемуся горизонту.

Предложено несколько методов управления, основывающихся на долгосрочном прогнозировании: модельно-алгоритмическое управление (МАУ) (Model Algorithmic Control) [21], динамическое матричное управление (ДМУ) (Dynamic Matrix Control) [22], расширенное прогнозирующее самоадаптирующееся управление (РПСАУ) (Extended Predictive Self Adaptive Control) [23], адаптивное управление по расширенному горизонту (АУРГ) (Extended Horizon Adaptive Control) [24], система самонастройки на основе прогнозирования [25], обобщенное прогнозирующее управление (ОПУ) (Generalized Predictive Control) [26—28]. В [29] приведен достаточно исчерпывающий сравнительный анализ различных методов управления на основе долгосрочного прогнозирования, с рассмотрением достоинств каждого из них.

В первых методах прогнозирующего управления (МАУ, ДМУ) использовались математические модели объектов управления, получаемые по реакции на единичное ступенчатое или импульсное воздействие. Для этих моделей требуется графическое представление реакции или экспериментальные данные

о ней, что достаточно сложно реализовать. В алгоритме ОПУ используется модель управляемой авторегрессии — проинтегрированного скользящего среднего (УАРПСС) (Controlled Auto-Regressive and Integrated Moving Average (CARIMA) model), параметры которой можно получить, используя методы оценки параметров (например, рекуррентный метод наименьших квадратов). Диапазон параметров настройки для алгоритма ОПУ является более общим, что обеспечивает большую гибкость в разработке разомкнутых систем по сравнению с методами РПСАУ и АУРГ; эти методы могут рассматриваться как частные случаи ОПУ.

Один из новейших и наиболее эффективных методов управления — управление на основе нечеткой логики. Теория нечеткой логики была впервые разработана Заде (Zadeh) [30] в начале 60-х годов и сейчас успешно применяется во многих промышленных системах и бытовых изделиях. Теория нечетких множеств и управление на основе нечеткой логики рассматриваются в последнем разделе данной главы.

3.2. ПИ- и ПИД-контроллеры с дискретным временем

Из всех форм дискретных ПИ- и ПИД-контроллеров для данного алгоритма приняты скоростные контроллеры. Этот тип контроллеров выбран по причине того, что они обладают многими преимуществами перед другими. Среди них можно указать следующие: способность работать в условиях резких и значительных изменений уставки, отсутствие необходимости в инициализации при переходе из ручного режима в автоматический. Другими словами, если резко изменяется величина уставки, или происходят значительные изменения в начале процесса, то данные контроллеры обеспечивают лучшую реакцию, чем ПИ- и ПИД-контроллеры других типов. Эти контроллеры также полезны при насыщении интегратора, которое обычно возникает в случаях, когда выход насыщается и контроллер продолжает интегрировать ошибку. Скоростная форма диск-

ретной ПИД-схемы управления может быть описана следующим уравнением:

$$\Delta u(t) = k_c((e'(t) - e'(t-1)) + \frac{T_s}{2T_i}(e'(t) + e'(t-1)) + \\ + \frac{T_d}{T_s}(e'(t) - 2e'(t-1) + e'(t-2))). \qquad (3.2.1)$$

Уравнение (3.2.1) можно переписать в следующей форме:

$$\Delta u(t) = K_P(e'(t) - e'(t-1)) + K_I e'(t) + K_D(e'(t) - \\ - 2e'(t-1) + e'(t-2)), \qquad (3.2.2)$$

где

$$K_P = k_c - \frac{1}{2}K_I,\ K_I = \frac{k_c T_s}{T_i},\ K_D = \frac{k_c T_d}{T_s},\ \Delta = 1 - z^{-1}.$$

Здесь K_c, T_i, T_d и T_s — коэффициент передачи, время сброса, время дифференцирования и время извлечения выборки соответственно. K_P, K_I, K_D — пропорциональный, интегральный и дифференциальный коэффициенты передачи соответственно. Сигнал ошибки $e'(t)$ может быть записан в виде:

$$e'(t) = r(t) - y(t), \qquad (3.2.3)$$

где $r(t)$ — желаемая уставка. Таким образом, мы имеем:

$$\Delta u(t) = K_P(r(t) - r(t-1) - y(t) + y(t-1)) + K_I(r(t) - y(t)) + \\ + K_D(r(t) - 2r(t-1) + r(t-2) - y(t) + 2y(t-1) - y(t-2)). \qquad (3.2.4)$$

Резкое существенное изменение уставки приведет к пропорциональному и дифференциальному управляющему действию, вызывающему существенное изменение на выходе контроллера. Чтобы подавить это явление, уставка $r(t)$ предполагается постоянной все время до изменения на следующем шаге. Уравнение (3.2.4) принимает вид:

$$\Delta u(t) = K_I r(t) - [K_P + K_I + K_D]y(t) + \\ + [K_P + 2K_D]y(t-1) - K_D y(t-2)). \qquad (3.2.5)$$

3.3. Управление с самонастройкой

Управление с самонастройкой можно рассматривать как вид управления, выполняющий две основные задачи в замкнутом цикле обратной связи. Общая схема такого метода управления приведена на рис. 3.3.1. На ней показаны обе основные задачи системы самонастройки. Первая из них — сбор информации о текущем состоянии управляемого процесса. К данной задаче относится постоянное определение текущего состояния управляемого процесса на основе измеримых данных о входе и выходе процесса, а также сигналов состояния.

Полученная информация используется для идентификации системы, которая включает определение структуры модели, оценку ее параметров, а также оценку параметров неконтролируемых сигналов (например, шумовых сигналов в стохастических системах). Определение структуры модели требует построения вида математического представления системы, соответствующего решаемой задаче. Оценка параметров представляет собой ключевой элемент самонастройки. Она выполняется в оперативном режиме. Для управления с самонастройкой ис-

Рис. 3.3.1. Общая схема самонастраивающегося контроллера

пользуется несколько схем рекурсивной оценки параметров. Наиболее распространенная схема — рекурсивный метод оценки на основе метода наименьших квадратов [31], [32] или его расширение — UD факторизационный метод [33], который является более надежным.

Вторая задача системы самонастройки — задача проектирования (контроллера); ее решение обычно базируется на оптимизации критерия оптимальности управления. Цель управления задается для каждой конкретной системы; при этом требуется принять решения в отношении того, как контроллер должен адаптироваться или настраиваться. На этой основе рассчитывается новый набор параметров контроллера (взамен прежних параметров в цикле управления). Эта часть процесса известна как этап ратификации или утверждения; одно из основных достоинств системы самонастройки состоит в том, что данный процесс выполняется в оперативном режиме и в реальном времени. При традиционном методе процесс утверждения обычно выполняется в автономном режиме; результаты часто оказываются неудовлетворительными, и весь процесс моделирования и проектирования приходится повторять. Расчет закона управления выполняется на основе процедуры, называемой эквивалентом определенности; в ней неопределенность текущих оценок параметров игнорируется.

Подход к оцениванию параметров передаточной функции по процессу и воздействию называется методом косвенной самонастройки, или методом полной определенности. Примером реализации метода такого типа может служить самонастраивающийся регулятор Острема (Åström) и Виттенмарка (Wittenmark) [6], известный также под названием системы управления по минимуму дисперсии. Критерий оптимальности такого регулятора — минимизация дисперсии выходного процесса y(t) по всем моментам дискретного времени t. Это может быть достигнуто следующим образом: сначала предсказывается будущая величина выходного процесса на d шагов вперед — y(t+d), затем текущая величина управляющего воздей-

ствия u(t) устанавливается таким образом, чтобы сделать этот выход нулевым. Однако в этом методе не делается попыток оптимизации уставок, а также возможны чрезмерно большие величины управляющих воздействий. Другим недостатком данного метода является нестабильность замкнутого цикла из-за потери нулей процесса, если процесс не является минимально-фазовым. Для преодоления этих сложностей Кларк (Clarke) и Гаутроп (Gawthrop) [18], [19] изменили критерий: минимизируется не только дисперсия выхода, но и функция, включающая управляющее воздействие, выход, а также изменения уставки. Таким образом обеспечивается возможность использования большего набора целей функционирования. В этом критерии управляющий сигнал подвергается «штрафу» таким образом, что уменьшаются его избыточные пики, и обеспечиваются возможности оптимального отслеживания уставки. Кроме того, модифицированный критерий обеспечивает стабильное управление неминимально-фазовым процессом. Такой тип самонастраивающегося контроллера известен также под названием УОМД (управление по обобщенному минимуму дисперсии). В последнее время в литературе приводятся различные усовершенствования и расширения этого контроллера [34—39].

Для вывода алгоритма работы таких контроллеров рассматривается система в форме модели управляемой авторегрессии — скользящего среднего (УАРСС) (Controlled Auto-Regressive Moving Average (CARMA) model). Систему можно представить в виде следующей математической модели:

$$A(z^{-1})y(t) = B(z^{-1})u(t-d) + C(z^{-1})\xi(t), \qquad (3.3.1)$$

где y(t) — выходной процесс, u(t) — входной процесс (управляющее воздействие), ξ(t) — некоррелированная случайная последовательность с нулевым математическим ожиданием и ковариацией (среднеквадратическим отклонением) σ, d — время задержки, t — целочисленное время. $A(z^{-1})$, $B(z^{-1})$ и $C(z^{-1})$ выражаются через оператор z-преобразования:

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{n_a} z^{-n_a}, \qquad (3.3.2)$$

$$B(z^{-1}) = 1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{n_b} z^{-n_b}, \qquad (3.3.3)$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{n_c} z^{-n_c}. \qquad (3.3.4)$$

Предполагается, что корни полинома $C(z^{-1})$ располагаются внутри круга единичного радиуса. Никаких предположений о корнях полиномов $A(z^{-1})$, и $B(z^{-1})$ не делается, т.е. считается, что за пределами единичного круга могут находиться как корни $A(z^{-1})$ (что означает неустойчивость объекта управления в разомкнутом состоянии), так и корни $B(z^{-1})$ (объект управления неминимально-фазовый).

В этом случае используется критерий качества в виде:

$$J = E[(P(z^{-1})y(t+d) - R(z^{-1})r(t))^2 + (Q(z^{-1})u(t))^2], \qquad (3.3.5)$$

где $E[.]$ — математическое ожидание, $P(z^{-1})$, $R(z^{-1})$ и $Q(z^{-1})$ — задаваемые пользователем полиномы, $r(t)$ — уставка, а $R(z^{-1})$ — дробно-рациональная передаточная функция:

$$P(z^{-1}) = \frac{P_n(z^{-1})}{P_d(z^{-1})}. \qquad (3.3.6)$$

Здесь $P_n(z^{-1})$ и $P_d(z^{-1})$ — полиномы степени n_{pn} и n_{pd} соответственно. Данный критерий качества в некоторых отношениях отличается от критерия качества по минимуму дисперсии, предложенного Остремом. Управление по последнему критерию направлено только на стабилизацию выходного процесса $y(t)$ вблизи нулевого значения при действии случайных возмущений. Однако во многих реальных ситуациях требуется поддерживать при случайных возмущениях выход системы на некотором постоянном уровне. В УОМД-контроллере используется как стабилизация, так и серворегулирование. Кроме того, УОМД-контроллер придает такой вес управляющему воздействию, что он препятствует появлению излишне больших значений этого воздействия при подавлении последствий от нулей, находящихся вне области устойчивости. В рассматриваемом подходе

вводится понятие псевдовыхода, задаваемого следующим $\phi(t)$ выражением:

$$\phi(t) = P(z^{-1})y(t) + Q(z^{-1})t(t-d) - R(z^{-1})r(t-d). \qquad (3.3.7)$$

Таким образом, систему можно рассматривать как обобщенный выход системы с прогнозирующим элементом, фильтрующим действием на входе и с уставкой. Многие стратегии самонастройки основываются на прогнозирующей схеме управления; горизонтом прогнозирования при этом является временная задержка d. Оптимальное прогнозирование выхода в момент t+d может быть получено в момент t, если вход выбран таким образом, что возмущения нейтрализуются. Для получения оптимального прогноза псевдовыхода рассмотрим уравнение тождества (тождественное равенство) в следующей форме:

$$C(z^{-1})P_n(z^{-1}) = E(z^{-1})A(z^{-1})P_d(z^{-1}) + z^{-d}F(z^{-1}), \qquad (3.3.8)$$

где

$$E(z^{-1}) = 1 + e_1 z^{-1} + ... + e_{d-1} z^{-(d-1)}, \qquad (3.3.9)$$

$$F(z^{-1}) = 1 + f_1 z^{-1} + ... + f_{d-1} z^{-(n_f)} \qquad (3.3.10)$$

и

$$n_f = \max(n_a + n_{p_d} - 1, n_c + n_{p_d} - d). \qquad (3.3.11)$$

Умножая уравнения (3.3.1) на $E(z^{-1})$, получим:

$$E(z^{-1})A(z^{-1})y(t+d) = E(z^{-1})B(z^{-1})u(t) + E(z^{-1})C(z^{-1})\xi(t+d). \qquad (3.3.12)$$

Подставляя $E(z^{-1})A(z^{-1})$ из уравнения (3.3.8) в уравнение (3.3.12), получим:

$$C(z^{-1})P(z^{-1})y(t+d) = F(z^{-1})y_f(t) + E(z^{-1})B(z^{-1})u(t) + E(z^{-1})C(z^{-1})\xi(t+d). \qquad (3.3.13)$$

Здесь $y_f = P_d(z^{-1})^{-1}y(t)$. Прибавляя $Q(z^{-1})C(z^{-1})u(t)$-$C(z^{-1})R(z^{-1})r(t)$ к обеим частям уравнения (3.3.13), получим:

$$C(z^{-1})(P(z^{-1})y(t+d)+Q(z^{-1})u(t-d)-R(z^{-1})r(t-d))= \\ =F(z^{-1})y_f(t)+(C(z^{-1})Q(z^{-1})+E(z^{-1})B(z^{-1}))u(t)+ \\ +E(z^{-1})C(z^{-1})\xi(t+d)-C(z^{-1})R(z^{-1})r(t), \qquad (3.3.14)$$

что эквивалентно выражению:

$$C(z^{-1})\phi(t+d)=F(z^{-1})y_f+(C(z^{-1})Q(z^{-1})+ \\ +E(z^{-1})B(z^{-1}))u(t)+ \\ +E(z^{-1})C(z^{-1})\xi(t+d)-C(z^{-1})R(z^{-1})r(t). \qquad (3.3.15)$$

Обозначив через $\phi^*(t+d/t)$ оптимальный прогноз величины $\phi(t+d)$, полученный на основе измерений до момента времени t, а через $\tilde{\phi}(t+d)$ — ошибку прогноза, а также полагая $C(z^{-1})=1$, получим:

$$\phi^*(t+d/t)=F(z^{-1})y_f(t)+(Q(z^{-1})+ \\ +G(z^{-1}))u(t)-R(z^{-1})r(t) \qquad (3.3.16)$$

и

$$\tilde{\phi}(t+d)=\phi(t+d)-\phi^*(t+d/t)=E(z^{-1})\xi(t+d). \qquad (3.3.17)$$

Здесь $G(z^{-1})=E(z^{-1})B(z^{-1})$. Учитывая, что ошибки в последующие моменты времени $\xi(t+1),\ldots,\xi(t+d)$ не коррелированы с текущими и предыдущими измерениями на входе и выходе, управление по минимальной дисперсии $\phi(t+d)$ достигается путем выбора u(t) таким, чтобы $\phi^*(t+d)$ обращалось в ноль. Таким образом, получим

$$u(t)=\frac{R(z^{-1})r(t)-F(z^{-1})y_f}{Q(z^{-1})+G(z^{-1})}. \qquad (3.3.18)$$

Так как в момент времени t величины $Q(z^{-1})u(t)$ и $R(z^{-1})r(t)$ известны, выражение прогноза $\phi(t+d)$ можно уменьшить до

$$\phi_y(t)=P(z^{-1})y(t). \qquad (3.3.19)$$

Таким образом, оптимальный прогноз $\phi_y(t)$ на d шагов вперед задается следующим выражением:

$$\phi_y^*(t+d) = F(z^{-1})\,y_f(t) + G(z^{-1})\,u(t)\cdot \qquad (3.3.20)$$

Параметры контроллера $F(z^{-1})$ и $G(z^{-1})$ можно легко получить из тождества (3.13), если полиномы $A(z^{-1})$ и $B(z^{-1})$ известны. Если же они неизвестны, то для оценки элементов $F(z^{-1})$ и $G(z^{-1})$ можно использовать рекурсивный метод наименьших квадратов, рассматриваемый ниже. Векторы параметров регрессии и данных следующие:

$$\hat{\theta}(t) = [\hat{f}_0(t),\ \hat{f}_1(t),\ \ldots,\ \hat{f}_{nf}(t),\ \hat{g}_0(t),\ \hat{g}_1(t),\ldots,\ \hat{g}_{ng}(t)], \qquad (3.3.21a)$$

$$x^T(t) = [y_f(t-1),\ y_f(t-2),\ \ldots,\ u(t-1),\ u(t-2),\ldots\ ], \qquad (3.3.21b)$$

где $\hat{f}_0,\ \hat{f}_1,\ \ldots,\ \hat{f}_{nf}$ и $\hat{g}_0,\ \hat{g}_1,\ \ldots.\ \hat{g}_{ng}$ — оценки элементов $F(z^{-1})$ и $G(z^{-1})$ соответственно.

Для самонастраивающегося контроллера оценка параметров выполняется рекурсивно, а наблюдения производятся последовательно. Процесс рекурсивной оценки параметров показан на рис. 3.3.2. Оценка текущего выхода $\hat{y}(t)$ вычисляется на основе прошлой информации о модели, полученной из оценки $\hat{\theta}(t-1)$. Ошибка моделирования $\varepsilon(t)$ получается путем сравне-

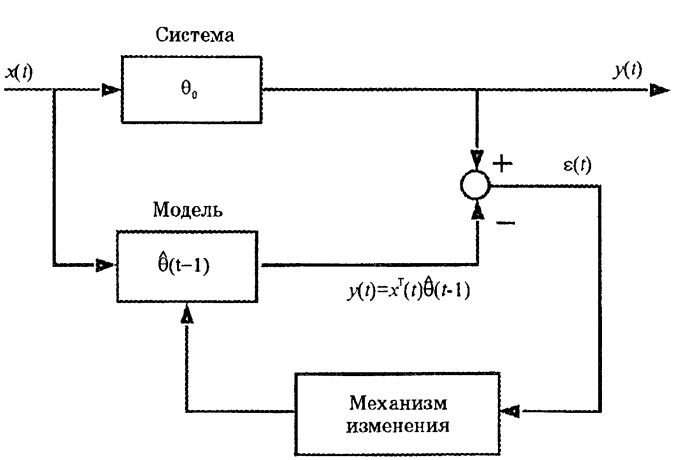


Рис. 3.3.2. Блок-схема рекурсивной оценки параметров

ния оценки выхода с наблюдаемым выходом. Затем ошибка минимизируется в смысле наименьших квадратов, а оценка $\hat{\theta}(t-1)$ корректируется в соответствии с новым значением $\hat{\theta}(t)$. Оценки $\hat{\theta}(t)$ могут быть получены с помощью трех важных рекурсивных уравнений:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)[y(t) - x^T(t)\hat{0}(t-1)], \quad (3.3.22)$$

$$K(t) = P(t-1)x(t)[1+x^T(t)P(t-1)x(t)]^{-1}, \quad (3.3.23)$$

$$P(t) = (1 - K(t)\,x^T(t))P(t-1), \quad (3.3.24)$$

где $P(t)$ — ковариационная матрица, задаваемая выражением:

$$P(t) = [x^T(t)x(t)]^{-1}. \quad (3.3.25)$$

3.4. Самонастраивающиеся ПИ- и ПИД- контроллеры

Доминирование ПИ- и ПИД-контроллеров среди промышленных регуляторов — основная движущая сила в развитии теории и практики самонастраивающихся ПИ- и ПИД-контроллеров (СНПИ- и СНПИД-контроллеров). Хотя самонастраивающиеся контроллеры обладают большей гибкостью и более приспособлены для управления объектами с переменными параметрами, нелинейностями и неопределенностями, их практическое применение пока не выглядит особенно перспективным.

Самонастраивающиеся регуляторы ориентированы на характеристики: характеристика замкнутой системы задается пользователем и алгоритм настраивается таким образом, чтобы достичь заданной характеристики, даже если параметры объекта управления или величины дрейфа неизвестны. Здесь подразумевается, что желаемая характеристика замкнутой системы управления может быть достигнута при определенных характеристиках исполнительного механизма; критически важным при этом является опыт инженерного персонала объекта управления. В этих условиях для упомянутого персонала крайне желательным является обеспечение как

можно большего интуитивного понимания предельной характеристики замкнутой системы в условиях самонастраивающегося управления. В связи с этим теория самонастраивающихся контроллеров быстро продвигается в направлении расширения диапазона возможных целей управления и их интерпретации в терминах классической теории управления, что упрощает понимание, как целей, так и методов их достижения.

Вывод выражений для СНПИ- и СНПИД-контроллеров [17] может быть получен из комбинации выражений (3.2.5) и (3.3.19). Чтобы закон управления (3.3.19) имел ПИД-структуру, аналогичную закону (3.2.5), передаточная функция $F(z^{-1})$ должна быть представлена полином второго порядка. Таким образом, порядок полинома $F(z^{-1})$ определяется равенством:

$$n_f = n_a + n_{pd} - 1, \qquad (3.4.1)$$

считая, что $n_a + n_{pd} - 1 > n_{pn} + n_c$. Для системы первого порядка ПИД-структура получается, если знаменатель $P_d(z^{-1})$ передаточной функции корректирующего фильтра выбирать в виде полинома второго порядка. Так как полином $Q(z^{-1})$ может выбираться пользователем, интегрирующее действие может быть введено следующим выражением:

$$\frac{\Delta}{v} = G(z^{-1}) + Q(z^{-1}), \qquad (3.4.2)$$

где v — константа.

Установившуюся ошибку можно устранить, положив:

$$R(z^{-1}) = H_0 = \sum_{i=0}^{n_f} \frac{F_i(z^{-1})}{P_d(z^{-1})}\Big|_{z^{-1}=1}. \qquad (3.4.3)$$

Используя уравнения (3.4.2) и (3.4.3), выражение (3.3.18) можно записать в форме:

$$\Delta u(t) = v(H_0 r(t) - (\hat{f}_0 + \hat{f}_1 z^{-1} + \hat{f}_2 z^{-2}) y_f(t)), \qquad (3.4.4)$$

что представляет собой закон управления для самонастраивающегося контроллера с ПИД-структурой. Выражения для соответствующих параметров ПИД-контроллера имеют следующий вид:

$$K_I = \frac{-v(\hat{f}_0 + \hat{f}_1 + \hat{f}_2)}{P_d(1)}, \tag{3.4.5}$$

$$K_P = \frac{-v(\hat{f}_1 + 2\hat{f}_2)}{P_d(1)} \tag{3.4.6}$$

и

$$K_P = \frac{-v\hat{f}_2}{P_d(1)}. \tag{3.4.7}$$

Интересно заметить, что, исходя из соответствующих выражений для ПИД-контроллера, можно ожидать, что коэффициент v будет иметь то же влияние для СНПИД-контроллеров, что и коэффициент K_c для ПИД-контроллеров. При небольшой величине v управление будет колебательным (осциллирующим). Целесообразно в качестве фильтра $P_d(z^{-1})$ использовать дифференцирующее звено. Здесь подразумевается, что замкнутая система представляет собой фильтр нижних частот. Величины v, $P_n(z^{-1})$ и $P_d(z^{-1})$ выбираются в основном методом проб и ошибок. Обоснованный подход к выбору этих параметров состоит в следующем: выбрать величину v таким образом, чтобы характеристика замкнутой системы была стабильной и не слишком осциллирующей, а затем варьировать величины $P_n(z^{-1})$ и $P_d(z^{-1})$ так, чтобы добиться достаточной эффективности системы.

Программная реализация СНПИД-контроллера для каждого момента дискретизации может быть представлена следующим образом.

Вычисления СНПИД-контроллера:

Шаг 1. Установить опорный входной сигнал.

Шаг 2. Прочитать входные и выходные данные.

Шаг 3. Задать начальные значения коэффициентов $P_n(z^{-1})$ и $P_d(z^{-1})$.

Шаг 4. Выполнить вычисления по формулам (3.3.22), (3.3.23), (3.3.24).

Шаг 5. Определить управляющее воздействие по формуле (3.4.4).

Шаг 6. Рассчитать ПИД-параметры по выражениям (3.4.5) — (3.4.7).

3.4.1. Замкнутая система

Замкнутая система с СНПИ- или СНПИД-контроллером описывается уравнением

$$(\Delta A(z^{-1}) - z^{-d} B(z^{-1}) \frac{F(z^{-1})}{P_d(z^{-1})}) y(t) = \\ = z^{-d} \nu B(z^{-1}) H_0 r(t) + \Delta\xi(t), \tag{3.4.8}$$

где $\xi(t)$ — возмущающее воздействие на процесс. Очевидно, что полиномы корректирующего фильтра $P_n(z^{-1})$ и $P_d(z^{-1})$ будут оказывать некоторое влияние на замкнутую систему. Можно также видеть, что для постоянного опорного сигнала достигается точное асимптотическое отслеживание, если выполняется условие:

$$H_0 = \frac{F(1)}{P_d(1)}, \tag{3.4.9}$$

что соответствует условию (3.4.3). Отметим также, что в установившемся режиме слагаемое, соответствующее возмущению, равно нулю. Это означает, что замкнутая система может парировать действие возмущения.

3.4.2. Пример моделирования

В большинстве систем управления требуется задавать предварительно некоторые параметры настройки. Для СНПИ- и СНПИД-контроллеров параметрами настройки являются ν, $P_n(z^{-1})$ и $P_d(z^{-1})$. Они играют важную роль в регулировке харак-

теристик замкнутой системы, поскольку относятся к контролируемым параметрам. Для ПИ- и ПИД-контроллеров параметры K_p, K_I и K_D также требуется выбирать, хотя они и остаются неизменными до тех пор, пока на управляемом объекте не произойдут какие-либо существенные изменения, что может потребовать перенастройки параметров.

В данном разделе приводятся результаты имитационного моделирования, которые предназначены помочь в выборе параметров настройки СНПИ- и СНПИД-контроллеров. В этих примерах проводятся некоторые исследования свойств и роли параметров настройки. Для сравнения с данными, полученными для СНПИ- и СНПИД-контроллеров, приводится также несколько примеров моделирования с использованием ПИ- и ПИД-контроллеров с постоянными параметрами. Примеры моделирования получены для 300 дискретных моментов времени. Для оценки параметров использовался метод наименьших квадратов; в качестве ковариационной матрицы была выбрана единичная матрица, умноженная на число 10. Значение коэффициента отсутствия последействия было выбрано равным единице.

В данном примере рассматривается непрерывная система второго порядка с запаздывающим звеном. Передаточная функция системы задается выражением:

$$Y(s) = \frac{e^{-s}}{(s+0.1)(s+1.0)} U(s).$$

Проведя дискретизацию соответствующего дифференциального уравнения с интервалом квантования в 0.5 с. и применив z-преобразование для полученного уравнения, получим z-передаточную функцию в виде:

$$GH(z) = \frac{z^{-3}(0.105 + 0.087\,z^{-1})}{1 - 1.557\,z^{-1} + 0.576\,z^{-2}}.$$

Во временной области данной передаточной функции соответствует уравнение:

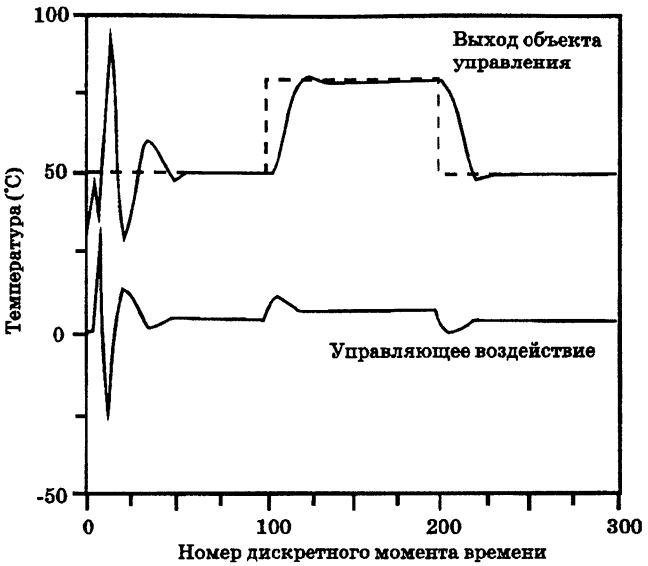


Рис. 3.4.1. Характеристика системы в примере моделирования с использованием СНПИД-контроллера

$$y(t) = 1.557y(t-1) - 0.576y(t-2) + 0.105u(t-3) + 0.087u(t-4).$$

Для данной системы СНПИД-контроллер может быть реализован только при условии, что в качестве полинома $P_d(z^{-1})$ выбран полином первого порядка. Никаких ограничений на порядок полинома $P_n(z^{-1})$ не накладывается; выберем в качестве такового полином второго порядка. Получим:

$$P_d(z^{-1}) = 1 + p_{d1} z^{-1},$$

$$P_n(z^{-1}) = 1 + p_{n1} z^{-1} + p_{n2} z^{-2}.$$

Величина v выбирается таким образом, чтобы получить достаточно стабильную характеристику, хотя возможны и осцилляции. Величины P_{d1}, P_{n1} и P_{n2} выбраны равными -0.75, 0.18 и -0.85 соответственно, чтобы задать форму отклика. При использовании СНПИД-контроллера можно ожидать сильных изменений в начале переходного процесса, возникающих из-за

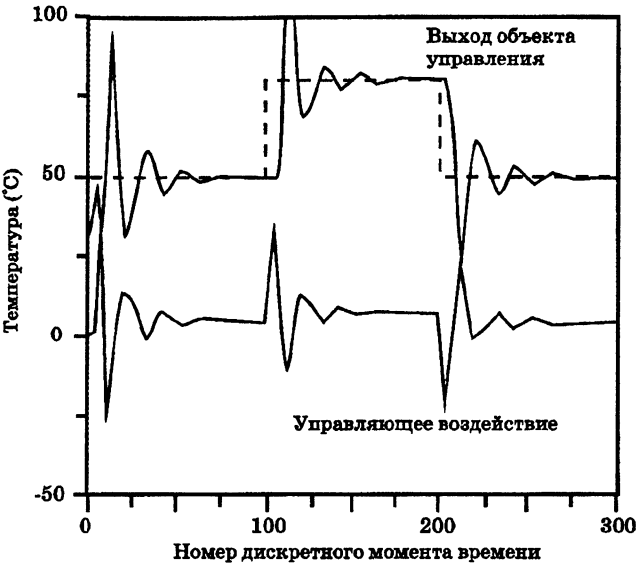


Рис. 3.4.2. Характеристика системы в примере моделирования с использованием ПИД-контроллера

начальных значений оценок параметров. Осцилляцию в начале переходного процесса можно подавить, используя на начальной фазе моделирования (в данном случае — в течение первых 20 моментов дискретного времени) обычное ПИД-управление. На рис. 3.4.1 показана характеристика СНПИД-контроллера для системы второго порядка. Можно убедиться, что при втором и третьем изменении уставки переходной процесс улучшается. Для сравнения приводятся характеристики той же системы с использованием ПИД-контроллера.

На рис. 3.4.2 показана характеристика замкнутой системы при использовании ПИД-контроллера с фиксированными параметрами. Хотя начальный участок переходного процесса не столь резкий по сравнению с предыдущим случаем, характеристики не улучшаются для второго и третьего участков переходного процесса.

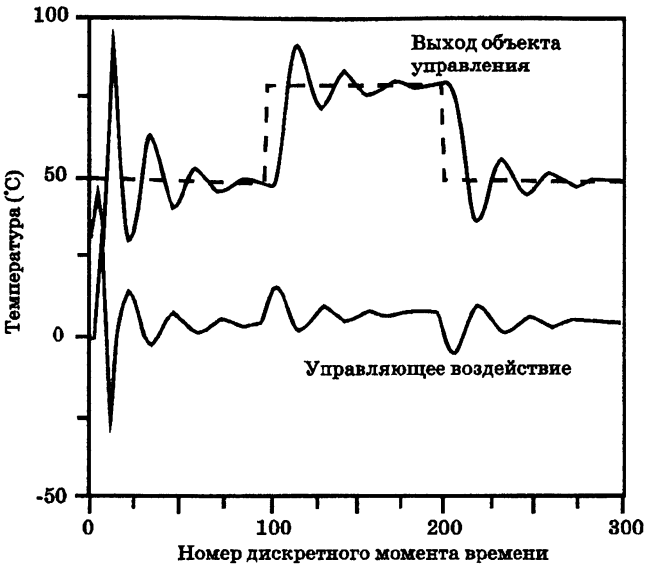


Рис. 3.4.3(*a*). Характеристика СНПИД-контроллера для системы с уменьшенным значением коэффициента p_{n2}

Для исследования роли параметров настройки, уменьшим величину P_{n2} до -0.99; в результате будет получена характеристика с большей осцилляцией, которая показана на рис. 3.4.3, *a*. Как и ожидалось, повышение величины P_{n1} или P_{n2} приводит к недостаточному демпфированию отклика, как показано на рис. 3.4.3, *b*. Для исследования роли полинома $P_d(z^{-1})$ выполнялось также уменьшение P_{d1}. На рис. 3.4.3, *c* видно, что при этом характеристики изменяются медленно.

Роль величин P_{d1}, P_{n1} и P_{n2} можно показать в сравнении с ролью коэффициентов K_p, K_I и K_D обычных ПИД-контроллеров. Из тождества (3.3.8) можно видеть, что для рассматриваемой в данном примере системы величина P_{d1} прямо зависит от величины F_2, которая, в свою очередь, прямо пропорциональна величине K_D, как это следует из выражения (3.4.7).

Использование коэффициента K_D в обычном ПИД-контроллере имеет целью формирование составляющей, которая

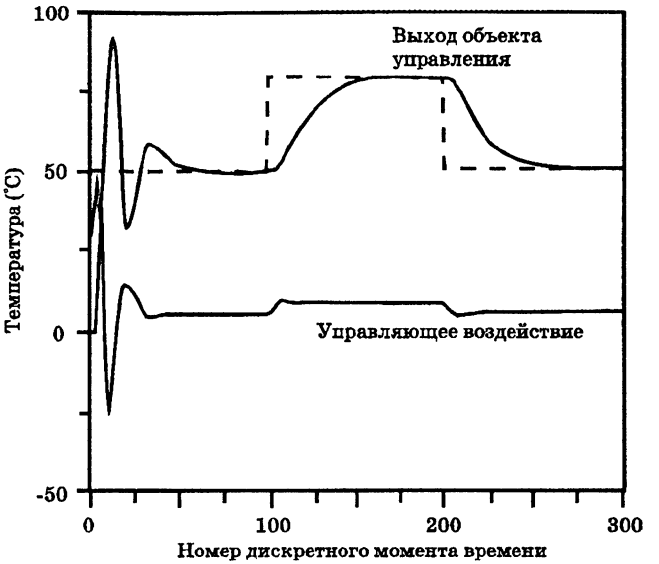


Рис. 3.4.3(*b*). Характеристика СНПИД-контроллера для системы с повышенными значениями коэффициентов p_{n1} и p_{n2}

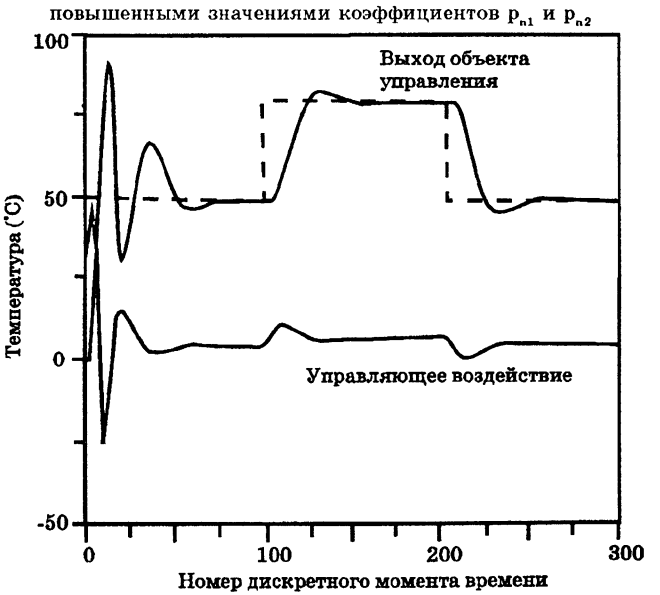


Рис. 3.4.3(*c*). Характеристика СНПИД-контроллера для системы с уменьшенным значением коэффициента p_{d1}

содержит информацию о направлении развития процесса. Из приведенных выше результатов моделирования видно, что действия коэффициентов p_{d1} и K_D аналогичны. Параметры настройки p_{n1} и p_{n2} оказывают некоторое влияние на параметры K_p и K_I, как это следует из выражений (3.3.8), (3.4.5) и (3.4.6).

3.5. ПИД-управление с самонастройкой: многомерный подход

При выводе закона управления для СНПИ- или СНПИД-контроллера в многомерном случае будем использовать следующую модель:

$$A(z^{-1})y(t) = B(z^{-1})u(t-d) + C(z^{-1})\xi(t), \qquad (3.5.1)$$

где $y(t)$ — $(n\times 1)$-вектор выходного процесса, $u(t)$ — $(n\times 1)$-вектор управляющего воздействия, $\xi(t)$ — $(n\times 1)$-вектор некоррелированной последовательности случайных составляющих с нулевым математическим ожиданием и ковариацией $E\{\xi(t)\xi^T(t)\}=P_\xi$. Полиномиальные $(n\times n)$-матрицы $A(z^{-1})$, $B(z^{-1})$ и $C(z^{-1})$ выражаются с помощью оператора обратного сдвига z^{-1} в следующем виде:

$$A(z^{-1}) = I + A_1 z^{-1} + A_2 z^{-2} + \ldots + A_{n_a} z^{-n_a}, \qquad (3.5.2)$$

$$B(z^{-1}) = B_0 + B_1 z^{-1} + B_2 z^{-2} + \ldots + B_{n_b} z^{-n_b}, \qquad (3.5.3)$$

$$C(z^{-1}) = I + C_1 z^{-1} + C_2 z^{-2} + \ldots + C_{n_c} z^{-n_c}. \qquad (3.5.4)$$

Для реализации рассматриваемого алгоритма требуется ряд допущений. Они приводятся ниже:

А1. Шум, нарушающий работу системы, является стабильным в том смысле, что нули полинома $C(z^{-1})$ лежат внутри круга единичного радиуса на z-плоскости.

А2. Полиномы $A(z^{-1})$ и $B(z^{-1})$ являются взаимно простыми.

А3. Структурные индексы системы равны, а верхние границы порядков всех скалярных полиномов, входящих в матрицы $A(z^{-1})$, $B(z^{-1})$ и $C(z^{-1})$, известны.

А4. Связанное с каждым каналом передачи транспортное запаздывание, составляющее не менее d интервалов времени, предполагается известным и одинаковым для всех каналов.

А5. Звенья $A(z^{-1})$ и $B(z^{-1})$ устойчивы, то есть система является минимально-фазовой.

Как и для одномерных систем, вывод закона управления выполняется на основе минимизации дисперсии псевдовыхода $\phi(t)$, заданного в виде:

$$\phi(t) = P(z^{-1})y(t) - R(z^{-1})r(t-d) + Q'(z^{-1})u(t-d), \qquad (3.5.5)$$

где $R(z^{-1})$ и $Q(z^{-1})$ — полиномиальные $(n \times n)$-матрицы, а $r(t)$ — $(n \times 1)$ — вектор уставки. $P(z^{-1})$ — рациональная матрица, которую можно представить следующим прямым матрично-дробным выражением:

$$P(z^{-1}) = P_n(z^{-1})\,P_d(z^{-1})^{-1}, \qquad (3.5.6)$$

где $P_n(z^{-1})$ и $P_d(z^{-1})$ — $(n \times n)$-матрицы числителя и знаменателя соответственно. Полиномы, $P(z^{-1})$, $R(z^{-1})$ и $Q'(z^{-1})$ задаются пользователем, что позволяет получить достаточно общий случай замкнутой системы. Положим $A'(z^{-1})=[A(z^{-1})P_d(z^{-1})]$ и введем (см. [40], [41], [42]) прямое матрично-дробное описание передаточной функции $A'(z^{-1})^{-1}C(z^{-1})$ в виде:

$$[A'(z^{-1})]^{-1}C(z^{-1}) = \overline{C}(z^{-1})[\overline{A}'(z^{-1})]^{-1}. \qquad (3.5.7)$$

Будем считать, что $\det\overline{C}(z^{-1}) = \det C(z^{-1})$ и $\overline{A}'(0) = \overline{C}(0) = I$. Тогда можно ввести тождество:

$$P_n(z^{-1})\overline{C}(z^{-1}) = E'(z^{-1})\overline{A}'(z^{-1}) + z^{-d}F'(z^{-1}), \qquad (3.5.8)$$

где $E'(z^{-1})$ и $F'(z^{-1})$ — полиномиальные $(n \times n)$-матрицы вида:

$$E'(z^{-1}) = I + E'_1 z^{-1} + E'_2 z^{-2} + \ldots + E'_{d-1} z^{-(d-1)}, \qquad (3.5.9)$$

и

$$F'(z^{-1}) = I + F_1 z^{-1} + F_2 z^{-2} + \ldots + F_{(n_a+n_{pd}-1)} z^{-(n_a+n_{pd}-1)}. \qquad (3.5.10)$$

Здесь n_{pd} — степень полинома $P_d(z^{-1})$. Чтобы разрешить проблемы, связанные с некоммутативностью произведения мат-

рицы, введем полиномиальные матрицы $\tilde{C}(z^{-1})$, $\tilde{E}'(z^{-1})$ и $\tilde{F}(z^{-1})$, такие, что

$$E'(z^{-1})C^{-1}(z^{-1}) = \tilde{C}^{-1}(z^{-1})\tilde{E}'(z^{-1}), \tag{3.5.11}$$

$$F(z^{-1})C^{-1}(z^{-1}) = \tilde{C}^{-1}(z^{-1})\tilde{F}(z^{-1}), \tag{3.5.12}$$

где

$$\det C(z^{-1}) = \det \tilde{C}(z^{-1}) \text{ и } \tilde{C}(0) = I.$$

Используя (3.5.5), (3.5.6), (3.5.9) и (3.5.10), получим:

$$\tilde{C}(z^{-1})P_n(z^{-1}) = \tilde{E}'(z^{-1})\overline{A}'(z^{-1}) + z^{-d}\tilde{F}'(z^{-1}). \tag{3.5.13}$$

Умножив последнее уравнение на полином $P_d^{-1}(z^{-1})$ и используя уравнения (3.3.1) и (3.5.11), получим:

$$\begin{aligned} P(z^{-1})y(t+d) = \tilde{C}^{-1}(z^{-1})(\tilde{F}'(z^{-1})P_d^{-1}(z^{-1})y(t) + \\ + \tilde{E}'(z^{-1})B(z^{-1})u(t)) + E'(z^{-1})\xi(t+d). \end{aligned} \tag{3.5.14}$$

Обозначим через $\phi^*(t+d/t)$ оптимальный прогноз значения $\phi(t+d)$ по измерениям до момента времени t, а через $\tilde{\phi}(t+d)$ — ошибку прогноза. Таким образом, получим:

$$\begin{aligned} \phi^*(t+d/t) = \tilde{C}^{-1}(z^{-1})(\tilde{F}'(z^{-1})P_d^{-1}(z^{-1})y(t) + \\ + (\tilde{G}'(z^{-1}) + Q(z^{-1}))u(t)) - R(z^{-1})r(t), \end{aligned} \tag{3.5.15}$$

$$\tilde{\phi}(t+d) = \phi(t+d) - \phi^*(t+d) = E'(z^{-1})\xi(t+d), \tag{3.5.16}$$

где

$$\tilde{C}(z^{-1})Q'(z^{-1}) = Q(z^{-1}) \text{ и } \tilde{E}'(z^{-1})B(z^{-1}) = \tilde{G}'(z^{-1}).$$

Заметим, что будущие ошибки $\xi(t+1), \ldots, \xi(t+d)$ некоррелированы с текущими и прошлыми измерениями входного и выходного процессов. Управление по минимуму дисперсии $\phi(t+d)$ достигается путем выбора управляющего воздействия u(t) таким, чтобы величина $\phi^*(t+d/t)$ обращалась в ноль. На этом основании имеем:

$$(\tilde{G}'(z^{-1}) + Q(z^{-1}))u(t)) = \tilde{C}(z^{-1})R(z^{-1})r(t) - \tilde{F}'(z^{-1})\,y_f(t). \tag{3.5.17}$$

Здесь $y_f(t)=P_d(z^{-1})y(t)$, а $y_f(t)$ — фильтрованный выход. Для

непосредственного получения параметров контроллера $\tilde{F}'(z^{-1})$, $\tilde{G}'(z^{-1})$ и шумовой компоненты $C'(z^{-1})$ рассмотрим величину $\phi_y(t)$, задаваемую выражением:

$$\phi_y(t+d) = P(z^{-1})y(t+d). \qquad (3.5.18)$$

Используя (3.5.13), можно записать выражение в виде:

$$\phi_y(t+d) = \tilde{C}(z^{-1})(\tilde{F}'(z^{-1})\, y_f(t) + \tilde{G}'(z^{-1})u(t)) + E'(z^{-1})\xi(t+d) \qquad (3.5.19)$$

и

$$\phi_y^*(t+d/t) = \tilde{C}(z^{-1})(\tilde{F}'(z^{-1})\, y_f(t) + \tilde{G}'(z^{-1})u(t)), \qquad (3.5.20)$$

где $\phi_y^*(t+d/t)$ — оптимальный прогноз величины $\phi_y(t+d/t)$, полученный на основе данных о выходном процессе до момента времени t. Положим $e(t+d)=E'(z^{-1})\,\xi(t+d)$. Тогда имеем:

$$e(t+d) = \phi_y\ (t+d) - \phi_y^*(t+d/t)\cdot \qquad (3.5.21)$$

Для оценки параметров перепишем (3.5.17) в форме:

$$\begin{aligned}\phi_y(t) = {} & \tilde{F}'(z^{-1})\, y_f(t-d) + \tilde{G}'(z^{-1})u(t-d) + \\ & + (\tilde{C}(z^{-1}) - 1)\,\phi_y^*(t/d-t) + e(t),\end{aligned} \qquad (3.5.22)$$

а вектор данных x(t) зададим следующим образом:

$$\begin{aligned}x(t) = {} & [y_f^T(t), y_f^T(t-1), \ldots, u^T(t), u^T(t-1), \ldots, \\ & \phi_y^{*T}(t+d-1/t-1), \phi_y^{*T}(t+d-2/t-2), \ldots].\end{aligned} \qquad (3.5.23)$$

Соответствующая матрица параметров $\theta(t)$ задается в виде:

$$\begin{aligned}\theta(t) = {} & [\theta_1 \theta_1 \ldots \theta_l]^T] = \\ & = [\tilde{F}'_0(t)\ \tilde{F}'_1(t) \ldots \tilde{G}_0(t)\ \tilde{G}_1(t) \ldots \tilde{C}_1(t)\ \tilde{C}_2(t) \ldots],\end{aligned} \qquad (3.5.24)$$

где вектор-столбец $q_i(t)$ имеет следующий вид:

$$\begin{aligned}\theta_i(t) = [& f_{i0}^0(t), f_{i1}^0(t), \ldots, f_{im}^0(t), f_{i0}^1(t), f_{i1}^1(t), \ldots, f_{im}^1(t), \ldots, \\ & g_{i0}^0(t), g_{i1}^0(t), \ldots, g_{im}^0(t), g_{i0}^1(t), g_{i1}^1(t), \ldots, g_{im}^1(t), \ldots, \\ & c_{i0}^1(t), c_{i1}^1(t), \ldots, c_{im}^1(t), \ldots].\end{aligned}$$

Здесь $f_{i0}^0(t)$, $g_{i1}^0(t)$ и $c_{i0}^0(t)$ — элементы первой строки $\tilde{F}_0(t)$, $\tilde{G}_0(t)$ и $\tilde{C}_1(t)$ соответственно.

Если $\tilde{C}(z^{-1})$ — единичная матрица, то выражение (3.5.22) содержит только величины, известные к моменту времени t, и ошибка прогноза устраняется. Таким образом, параметры контроллера $\tilde{F}'(z^{-1})$, $\tilde{G}'(z^{-1})$ могут быть получены на основе схемы оценивания параметров, например, рекурсивной оценки по методу наименьших квадратов. Если $\tilde{C}(z^{-1})$ — не единичная матрица, то оценка параметра будет смещена, так как величина $\phi_y^*(t+d/t)$ неизвестна. Однако можно заменить $\phi_y^*(t+d/t)$ ее оценкой $\hat{\phi}_y^*(t+d/t)$, которую можно рассчитать, а затем для получения оценок параметров использовать обычный рекурсивный метод наименьших квадратов. Этот метод аналогичен расширенному методу наименьших квадратов [31], [32]. Вектор данных задается следующим образом:

$$\hat{X}(t) = [y_f^T(t), y_f^T(t-1), \ldots, u^T(t), u^T(t-1), \ldots, \hat{\phi}_y^{*T}(t+d-1/t-1), \hat{\phi}_y^{*T}(t+d-2/t-2), \ldots]. \qquad (3.5.25)$$

Чтобы получить структуру многомерного ПИД-контроллера в выражении (3.5.17), необходимо, чтобы степень полинома $\tilde{F}'(z^{-1})$ была равна двум. Поэтому

$$\tilde{F}'(z^{-1}) = \tilde{F}_0 + \tilde{F}_1 z^{-1} + \tilde{F}_2 z^{-2}. \qquad (3.5.26)$$

Следующее требование — введение интегрирующего управляющего воздействия в закон управления многомерного УОМД. Это можно сделать, положив

$$\tilde{G}'(z^{-1}) + Q(z^{-1}) = V'(1 - z^{-1}), \qquad (3.5.27)$$

где

$$V = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}.$$

Устранение установившейся ошибки выполняется несколько по-другому: $\tilde{C}(z^{-1})R(z^{-1}) = H_0$, где

$$H_0 = \sum_{i=0}^{n_a+n_{pd}-1} \tilde{F}_i P_d^{-1}(z^{-1})\Big|_{z^{-1}=1}. \qquad (3.5.28)$$

Закон управления записывается в виде:

$$\Delta u(t) = V H_0 r(t) - V(\tilde{F}_0 + \tilde{F}_1 z^{-1} + \tilde{F}_2 z^{-2}) y_f(t). \qquad (3.5.29)$$

Выражения для параметров многомерного СНПИД-контроллера в терминах параметров многомерного ПИД-контроллера записываются следующим образом:

$$K_P = V(-\tilde{F}_1 - 2\tilde{F}_2)\alpha, \qquad (3.5.30)$$

$$K_I = V(\tilde{F}_0 + \tilde{F}_1 + \tilde{F}_2)\alpha, \qquad (3.5.31)$$

$$K_D = V\tilde{F}_2\alpha, \qquad (3.5.32)$$

где

$$\alpha = p_d^{-1}(z^{-1})\big|_{z=1}.$$

Для реализации структуры ПИ- или ПИД-контроллера степень $\tilde{F}(z^{-1})$ должна быть равна 1 или 2 соответственно. Предполагая взаимную простоту и несингулярность матриц $A(z^{-1})$ и $C(z^{-1})$, для случая, когда $C(z^{-1}) = I$, получим $\tilde{E}'(z^{-1}) = E'(z^{-1})$ и $\tilde{F}(z^{-1}) = F(z^{-1})$. Степень полинома $\tilde{F}(z^{-1})$, таким образом, определяется степенями полиномов $P_d(z^{-1})$ и $A(z^{-1})$. Отсюда следует, что если степень полинома $A(z^{-1})$ равна 1, то эффективным является ПИ- или ПИД-контроллер, в зависимости от того, равна степень полинома $P_d(z^{-1})$ единице или двум. В некоторых случаях подходит управление системой первого порядка с помощью ПИ-контроллера. Преимущество такого варианта — в меньшем количестве параметров, которые требуется оценивать и предварительно выбирать. Однако для некоторых процессов требуются дифференцирующие воздействия с целью устранения осцилляций, возникающих вблизи уставки из-за медленного действия контроллеров. Диапазон процессов, управляемых с использованием данного алгоритма, ограничива-

ется третьим порядком. Однако можно доказать, что для процессов более высоких порядков может быть применена процедура редукции модели. Некоторые примеры редукции модели приводятся в [43—45]. Если $C(z^{-1})$ — не единичная матрица, то для определения $\tilde{E}'(z^{-1})$ и $\tilde{F}(z^{-1})$ требуется сначала вычислить наибольший общий делитель $A'(z^{-1})$ и $C(z^{-1})$, а затем воспользоваться формулами (3.5.7), (3.5.8), (3.5.12), (3.5.13). В этом случае степень полинома $\tilde{F}(z^{-1})$ не обязана совпадать со степенью полинома $F'(z^{-1})$ (см.[46]).

3.5.1. Пример моделирования

Данный пример рассматривается с целью изучения возможности использования СНПИ- и СНПИД-контроллеров для управления процессами в условиях изменения уставки. Объектом управления является водная ванна с двумя входами и двумя выходами, показанная на рис. 3.5.1. В дискретном времени данная система может быть описана уравнением:

$$A(z^{-1})y(t) = B(z^{-1})u(t-1) + \xi(t), \qquad (3.5.33)$$

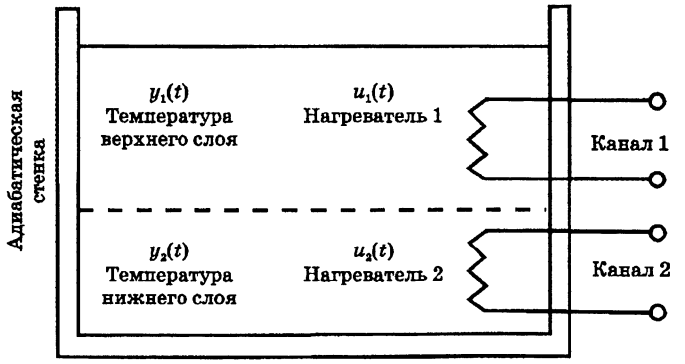


Рис. 3.5.1. Схема двухканальной водной ванны

где

$$A(z^{-1}) = I + A_1 z^{-}$$

и

$$A_1 = \begin{bmatrix} -0.411 & -0.634 \\ -0.103 & -0.885 \end{bmatrix},$$

$$B(z^{-1}) = B_0 = \begin{bmatrix} 0.492 & 0.085 \\ 0.041 & 0.237 \end{bmatrix}.$$

Здесь $\xi(t)$ — некоррелированный гауссов шум с нулевым математическим ожиданием и ковариацией ρ_ξ. Модель строится на основе баланса теплового потока в водной ванне.

При использовании многомерных СНПИ- или СНПИД-контроллеров необходимо заранее выбрать полиномиальные матрицы корректирующих фильтров; этот выбор выполняется методом проб и ошибок. Полиномиальные матрицы можно рассматривать как задающие опережение по фазе для цепи обратной связи в контуре управления. Это свойство оказывает влияние как на переходной процесс по управляющему воздействию (уставкам), так и обеспечивает более надежное подавление возмущений. Следует заметить, что элементы полиномов $P_n(z^{-1})$ и $P_d(z^{-1})$ имеют существенное влияние на параметры контроллера.

Таким образом, эти полиномы могут использоваться для улучшения характеристик замкнутой системы. В случае незначительного взаимодействия каналов можно выбрать диагональную матрицу $P_n(z^{-1})$. Полиномиальные матрицы корректирующих фильтров задаются в этом случае следующим образом:

$$P_n(z^{-1}) = I + P_{n1} z^{-1} \text{ и } P_{n1} = \begin{bmatrix} -0.4 & 0 \\ 0 & -0.7 \end{bmatrix},$$

$$P_d(z^{-1}) = I + P_{d1} z^{-1} \text{ и } P_{d1} = \begin{bmatrix} -0.3 & 0 \\ 0 & -0.4 \end{bmatrix}.$$

Величины V_1 и V_2 выбираются равными 7 и 5 соответствен-

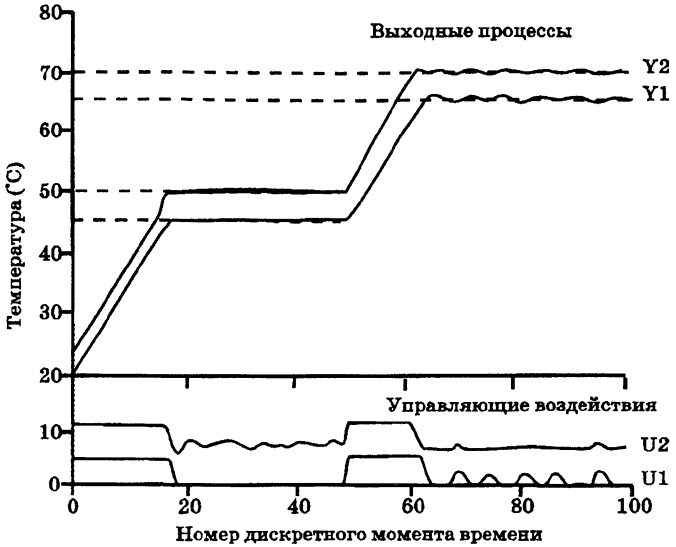


Рис. 3.5.2. Результаты моделирования водной ванны с многомерным СНПИ-контроллером

но. На рис. 3.5.2 показаны характеристики многомерного СНПИ-контроллера при управлении температурой водной ванны в условиях изменяющейся уставки.

3.6. Обобщенное прогнозирующее управление: некоторые теоретические аспекты

3.6.1. Критерий качества

Критерию качества, принятому для алгоритма ОПУ, предшествовал критерий минимума дисперсии [6,47], а также УОМД [18,19], рассмотренный в разделе 3.3. Уравнение контроллера получается для заданной линейной модели «вход — выход» посредством минимизации критерия качества, представленного в следующем виде:

$$J(u, t) = E[(y(t+j)-r(t+j))^2 + \lambda_1(\Delta u(t))^2]. \qquad (3.6.1)$$

Очевидное различие между вышеприведенным критерием качества и критерием для УОМД заключается в постановке задачи оптимизации, которая описывается не через значения процесса u(t), а через конечную разность Δu(t). Она введена в разделе 3.2 и обозначает приращение управляющего воздействия в системе. В момент времени t критерий минимизируется относительно разности Δu(t). Использование инкрементных входных сигналов обеспечивает появление нулевой статической ошибки при ненулевом постоянном опорном сигнале. Однако использование инкрементного управления затрудняет установление точно определенных границ величины u(t). В результате она может достигать очень больших величин. Чтобы предотвратить такое увеличение и при этом сохранить дисперсию как можно меньшей, необходимо при обеспечении устойчивости замкнутой системы поддерживать достаточно малое значение весового коэффициента λ_1. Таким образом, весовой коэффициент приращения управляющего воздействия λ_1 включается в окончательное выражение только для ограничения управляющего сигнала.

Другим свойством ОПУ является способность к управлению неминимально-фазовыми и неустойчивыми в разомкнутом состоянии системами, в частности, системами с не полностью известными величинами транспортного запаздывания. Для удовлетворения этим требованиям, критерий (3.6.1) преобразуется к следующему виду:

$$J1(u, t) = E[\sum_{j=N1}^{N2} (y(t+j)-r(t+j))^2 + \sum_{j=1}^{NU} \lambda_1(j)(\Delta u(t+j-1))^2] \qquad (3.6.2)$$

при условии Du(t+j-1))=0 при j>NU.

Здесь NU — горизонт управления, N1 — минимальный горизонт качества, N2 — максимальный горизонт качества, $\lambda_1(j)$ — последовательность весовых коэффициентов управления. Добавление ограничения на приращение управляющего воздействия можно интерпретировать как назначение беско-

нечных весовых коэффициентов на изменения управляющего воздействия через некоторое время. Это, в свою очередь, не только обеспечивает быстрое достижение контроллером своей цели и стабилизацию неминимально-фазового объекта управления, но и существенно снижает накладные расходы, связанные с вычислениями. Правильный выбор весовых коэффициентов управления и горизонтов стоимости обеспечивает также достижение хороших показателей качества и асимптотической устойчивости замкнутой системы и для систем, устойчивых в разомкнутом состоянии [48].

Процедура минимизации выполняется на нескольких (N2-N1+1) будущих значениях выхода с учетом нескольких (NU) будущих инкрементных управляющих воздействий. Результат алгоритма в некоторой степени зависит от того, входит ли действительное запаздывание в интервал между величинами N1 и N2. Обычно значение величины N1 выбирается равным значению времени запаздывания. В процедуре минимизации используется также концепция управления по удаляющемуся горизонту, в которой управляющие воздействия $\Delta u(t)$, $\Delta u(t+1)$,..., $\Delta u(t+NU-1)$ вырабатываются, но фактически применяется только значение $\Delta u(t)$. В момент $t+1$ решается новая задача минимизации. Управление по такому критерию обеспечивает как хорошие показатели качества, так и асимптотическую устойчивость разнообразных объектов управления.

3.6.2. Модель объекта управления и решение задачи оптимизации

Рассмотрим сначала алгоритм ОПУ для систем с одним входом и одним выходом. При выводе алгоритма ОПУ Кларк и др. ввели модель УАРПСС (управляемая авторегрессия — проинтегрированное скользящее среднее); эта модель является более подходящей для использования в управлении процессами с целью устранения смещения, вызываемого возмущениями. Эти возмущения можно смоделировать как дрейф (броуновское дви-

жение) или как ступенчатую случайную величину, изменяющуюся в случайные моменты времени. Модель УАРПСС с временной задержкой d, равной 1, может быть записана в следующей форме:

$$A(z^{-1})y(t) = B(z^{-1})u(t-1) + C(z^{-1})\frac{\xi(t)}{\Delta}. \qquad (3.6.3)$$

Возмущающее воздействие может быть как детерминированным, так и стохастическим, однако за счет оператора Δ его математическое ожидание принимается равным нулю. В подходе на основе ОПУ процесс $\xi(t)$ предполагается стохастическим, а полином $C(z^{-1})$ — устойчивым и равным 1.

Для решения задачи оптимизации (3.6.2) требуется вычислить прогноз на j шагов вперед для выходного процесса y(t+j) при j=N1,...,N2. Вычисления выполняются на основе информации, известной в момент времени t, а также информации о будущих значениях инкремента управления, которые требуется выбирать таким образом, чтобы оптимизировать критерий (3.6.2). Прогноз выходного процесса на j шагов можно описать в следующей форме:

$$\hat{y}(t+j/t) = G_j(z^{-1})\Delta u(t+j-1) + F_j(z^{-1})y(t) + C(z^{-1})\frac{\xi(t)}{\Delta}, \quad (3.6.4)$$

где $G_j(z^{-1})=E_j(z^{-1})B(z^{-1})$, E_j и F_j получены путем рекурсивного решения диофантова уравнения:

$$1 = E_j(z^{-1})A\Delta + z^{-j}F_j(z^{-1}), \qquad (3.6.5)$$

$$1 = \tilde{R}_j(z^{-1})A\Delta + z^{-(j+1)}\tilde{S}_j(z^{-1}), \qquad (3.6.6)$$

где $\tilde{R}_j(z^{-1}) = E_{j+1}(z^{-1})$ и $\tilde{S}_j(z^{-1}) = F_{j+1}(z^{-1})$, j=1, 2, ..., N.

Вычитая уравнение (3.6.5) из уравнения (3.6.6), получим

$$0 = (\tilde{R}_j(z^{-1}) - E_j(z^{-1}))A(z^{-1})\Delta + z^{-j}(z^{-1}\tilde{S}_j(z^{-1}) - F_j(z^{-1})). \quad (3.6.7)$$

Таким образом,

$$\begin{aligned}\tilde{R}_j(z^{-1}) - E_j(z^{-1}) &= (1 + \tilde{r}_1 z^{-1} + \tilde{r}_2 z^{-2} + \ldots + \tilde{r}_j z^{-j}) - \\ &- (1 + e_1 z^{-1} + e_2 z^{-2} + \ldots + e_j z^{-(j-1)}) = \\ &= (\tilde{r}_1 - e_1) z^{-1} + (\tilde{r}_2 - e_2) z^{-2} + \ldots \\ &+ (\tilde{r}_{j-1} - e_{j-1}) z^{-(j-1)} + \tilde{r}_j z^{-j} = \\ &= \bar{R}_j(z^{-1}) + \tilde{r}_j z^{-j}.\end{aligned} \qquad (3.6.8)$$

Полагая $A(z^{-1})=A(z^{-1})\Delta$, получим

$$\begin{aligned}A\bar{R} &= (1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na+1} z^{-(na+1)})(\bar{r}_1 z^{-1} + \\ &+ \bar{r}_2 z^{-2} + \ldots + \bar{r}_{j-1} z^{-(j-1)}) = (\bar{r}_1 z^{-1} + (\bar{r}_2 + a_1 \bar{r}_1) z^{-2} + \ldots \\ &+ (\bar{r}_{j-1} + a_1 \bar{r}_{j-2} + \ldots) z^{-(j-1)} + \ldots + a_{na+1} \bar{r}_{j-1} z^{-(j+na)}).\end{aligned} \qquad (3.6.9)$$

Из уравнения (3.6.7) получим:

$$A(z^{-1})\bar{R}(z^{-1}) + z^{-j}(z^{-1}\tilde{S}(z^{-1}) - F(z^{-1}) + A(z^{-1})\tilde{r}_j) = 0. \qquad (3.6.10)$$

Так как слагаемое $z^{-j}(z^{-1}\tilde{S}(z^{-1}) - F(z^{-1}) + A(z^{-1})\tilde{r}_j)$ содержит члены z^{-j}, $z^{-(j+1)}$,..., значит, первый (j-1)-й член выражения $A(z^{-1})\bar{R}(z^{-1})$ нулевой. Отсюда следует, что $\bar{R}(z^{-1}) = 0$. Таким образом, имеем

$$z^{-j}(z^{-1}\tilde{S}(z^{-1}) - F(z^{-1}) + A(z^{-1})\tilde{r}_j) = 0$$

и

$$\tilde{S}(z^{-1}) = z(f_0 + f_1 z^{-1} + \ldots + \tilde{r}_j - a_1 \tilde{r}_j z^{-1} - \ldots - a_{na+1} \tilde{r}_j z^{-(na+1)}).$$

Приравнивая члены при одинаковых степенях z, получим:

$$f_0 = \tilde{r}_j \text{ и } \tilde{s}_j = f_{i+1} - a_{i+1}\tilde{r}_j, \qquad (3.6.11)$$

где индекс i изменяется от 0 до степени полинома .

Прогноз выходного значения можно разбить на две компоненты: f(t+j), которая известна в момент t, и $\Gamma\Delta u(t+j-1)$ — компонента, соответствующая управляющему воздействию. Таким образом,

$$Y = \Gamma U + \quad , \qquad (3.6.12)$$

где

$$Y = [y(t+1)\ y(t+2) \quad y(t+N)]^T,$$

$$U = [\Delta u(t)\ \Delta u(t+1) \ldots \Delta u(t+N-1)]^T,$$

$$F = [f(t+1)\ f(t+2) \ldots f(t+N)]^T.$$

Здесь Г — нижняя треугольная N×N-матрица. Ее элементы представляют собой величины реакции объекта управления на соответствующих шагах.

$$\Gamma = \begin{bmatrix} \Gamma_0 & 0\ldots & 0 \\ \Gamma_1 & \Gamma_0 & \\ \vdots & & \vdots \\ \Gamma_{N2-1} & \ldots \Gamma_{N2-NU} & \end{bmatrix}.$$

Отличительная особенность алгоритма ОПУ состоит в том, что выбор параметров N1, N2 и λ_1 является гибким и не влияет на устойчивость управляемой системы. Однако правильный выбор величин N1 и N2 обеспечивает сокращение времени вычислений.

Квадратичной показатель качества (3.6.2) теперь можно представить в виде

$$J1 = E[(Y-r)^T(Y-r) + \lambda_1 U^T U], \qquad (3.6.13)$$

где r — вектор заранее заданных уставок. Получение инкрементного вектора управления Δu(t) для будущих моментов времени достигается посредством дифференцирования показателя качества по U. Таким образом, получим:

$$U = (\Gamma^T\Gamma + \lambda_1^{-1}I)^{-1}\Gamma^T(r-f). \qquad (3.6.14)$$

Так как требуется только первое управляющее воздействие, имеем

$$u(t) = u(t-1) + \bar{\Gamma}^T(r-f). \qquad (3.6.15)$$

где $\bar{\Gamma}^T$ — первая строка матрицы $(\Gamma^T\Gamma + \lambda_1^{-1}I)^{-1}\Gamma^T$.

Чтобы сделать алгоритм самонастраивающимся, выполня-

ется оценка параметров объекта управления на основе рекурсивного метода наименьших квадратов, рассмотренного в разделе 3.2. Векторы данных и параметров для модели объекта управления записываются в следующем виде:

$$x^T = [\Delta u(t-1), \Delta u(t-2), \ldots, \Delta y(t-1), \Delta y(t-2), \ldots], \quad (3.6.16)$$

$$\theta^T = [b_0, b_1, \ldots, a_0, a_1, \ldots]. \quad (3.6.17)$$

3.7. Управление на основе нечеткой логики

В последнее время термин «нечеткий» применяется в отношении широкого набора бытовых изделий и промышленных систем. Большая часть этих разработок стала возможной благодаря лучшему пониманию концепции нечеткой логики, простоте реализации этой концепции и ее удобству для разработки аппаратуры. Первоначально, когда нечеткая логика только разрабатывалась [30,49,50], предполагалось, что она найдет применение в основном в области систем, основанных на знаниях, в которых значительная часть постоянно хранящейся информации является одновременно неточной и неопределенной. Однако в противоположность этим ожиданиям, большинство примеров успешного применения нечеткой логики в настоящее время относится к системам управления, в которых есть неточность, но нет неопределенности.

Идея применения нечеткой логики в системах управления была впервые выдвинута Мамдани (Mamdani) и его коллегами [51,52,53,54]. Основываясь на теории нечетких множеств Заде (Zadeh) [30,49] и на концепции обычных ПИД-контроллерах, Ассилиан (Assilian) и Мамдани [51] разработали то, что сейчас называют базовым нечетко-логическим контроллером. Этот контроллер применялся для регулировки выходов процесса в окрестности заданной уставки; при этом использовалась цифровая ЭВМ. Из-за ограниченности памяти и быстродействия последовательных фон- неймановских ЭВМ нечетко-логические контроллеры сначала не получили широкого распростра-

нения; они в основном применялись для управления процессами с медленными изменениями величин. Некоторые из ранних примеров применения нечеткой логики в управлении процессами приводятся в работах [55,56].

Применение нечеткой логики для управления процессами в промышленности имеет ряд преимуществ по сравнению с использованием традиционных контроллеров. По-видимому, одно из основных преимуществ состоит в том, что нечетко-логический контроллер может разрабатываться по лингвистическим правилам, что тесно связано с искусственным интеллектом. Как описано в книге Винера [57], искусственный интеллект (или теория нейронных систем) и теория управления когда-то рассматривались как одна область знаний. Одна из целей искусственного интеллекта состоит в том, чтобы заменить человека машиной при выполнении точных операций; таким образом, связь между искусственным интеллектом и теорией управления очевидна. Нечеткий контроллер состоит из набора условных лингвистических операторов, или правил (называемых нечеткими ассоциативными матричными правилами, или НАМ-правилами), задающими конкретные ситуации управления. Эти условные лингвистические операторы могут быть легко получены из соображений здравого смысла или из технических сведений о процессе, которым требуется управлять.

Для многих промышленных процессов сложно обеспечить точное управление. Они обычно являются многомерными, нелинейными и изменяющимися во времени. Управление на основе нечеткой логики может успешно применяться для таких процессов. Кроме того, нечеткие контроллеры могут работать с не полностью описанными системами с неизвестной динамикой, так как для них (в отличие от многих традиционных адаптивных контроллеров) не требуется априорная математическая модель объекта управления. Еще одно преимущество нечетких контроллеров состоит в том, что они могут быть легко реализованы на цифровых или аналоговых СБИС [58,59,60],

в которых информация может кодироваться по параллельно-распределенной схеме.

3.7.1. Краткие сведения по теории нечетких множеств и нечетких систем

Теория нечетких множеств — активно развивающаяся сфера исследований. Некоторые ее аспекты рассмотрены в работах [61,62,63,64,65]. В данном разделе приводится краткий обзор теории нечетких множеств с точки зрения Коско (Kosko) [64]. Согласно этой теории [64], нечеткое множество можно определить как точку в кубе, а нечеткую систему — как отображение между такими кубами. S — нечеткая система, отображающая нечеткое множество в нечеткое множество. Таким образом, нечеткая система преобразует нечеткие множества из одного гиперкуба I^n в другой гиперкуб I^p:

$$S: I^n \rightarrow I^p, \qquad (3.7.1)$$

где I^n — n-мерный единичный гиперкуб, содержащий все нечеткие подмножества области определения, а I^p содержит все нечеткие подмножества области значений. Вообще говоря, нечеткая система S отображает семейства нечетких множеств друг на друга, т.е.

$$S: I^{n1} \times \ldots \times I^{nr} \rightarrow I^{p1} \times \ldots \times I^{pl}. \qquad (3.7.2)$$

Такие нечеткие системы действуют как элементы ассоциативной памяти, когда они отображают замкнутые входы на замкнутые выходы; в этом смысле к ним применяется название «нечеткая ассоциативная память» (НАП). Простейшая НАП кодирует НАМ-правило, или ассоциацию (A_i,B_i), связывающую p-мерное нечеткое множество B_i с n-мерным нечетким множеством A_i. Такие минимальные НАП отображают один элемент из гиперкуба I^n в один элемент гиперкуба I^p.

В теории нечетких множеств [30] определены три основные операции: дополнение, объединение и пересечение нечетких множеств. Дополнение нечеткого множества A обозначается, как A' и определяется выражением:

$$m_{A'}(x) = 1 - m_A(x).\ x \in X, \qquad (3.7.3)$$

где X — множество точек вещественной оси R_1, а $m_A(x)$ — функция принадлежности нечеткого множества A, принимающая значения в диапазоне от 0 до 1. Объединение двух нечетких множеств A и B с соответствующими функциями принадлежности $m_A(x)$ и $m_B(x)$ — это нечеткое множество C, обозначаемое как $A \cup B$. Его функция принадлежности определяется по функциям принадлежности множеств A и B в соответствии с выражением:

$$m_C(x) = \mathrm{Max}[m_A(x), m_B(x)],\ x \in X, \qquad (3.7.4)$$

или в сокращенной форме:

$$m_C(x) = m_A(x) \vee m_B(x)]\ . \qquad (3.7.5)$$

Пересечение двух нечетких множеств A и B с функциями принадлежности $m_A(x)$ и $m_B(x)$ — это нечеткое множество C, обозначаемое как $A \cap B$. Его функция принадлежности определяется выражением:

$$m_C(x) = \mathrm{Min}[m_A(x), m_B(x)],\ x \in X, \qquad (3.7.6)$$

или в сокращенной форме:

$$m_C(x) = m_A(x) \wedge m_B(x)]. \qquad (3.7.7)$$

Проще говоря, объединение нечетких множеств A и B — это самое малое нечеткое множество, содержащее как A, так и B. Пересечение нечетких множеств A и B — это нечеткое множество, содержащееся как в A, так и в B.

Вообще говоря, НАП-система $F: I^n \to I^p$ осуществляет кодирование и параллельную обработку набора из m НАМ-правил $(A_1, B_1), \ldots, (A_m, B_m)$, как показано на рис. 3.7.1. Каждый вход НАП-системы A активизирует каждое из хранимых НАМ-правил в различной степени. Минимальное НАМ- правило, i-я ассоциация, или правило (A_i, B_i) отображает вход A в выход B'_i, представляющий собой частично активированный вариант B_i. Чем больше вход A напоминает A_i, тем больше B'_i напоминает B_i. Соответствующее выходное нечеткое множество B объединяет эти частично активированные нечеткие множества $B'_i, \ldots,$

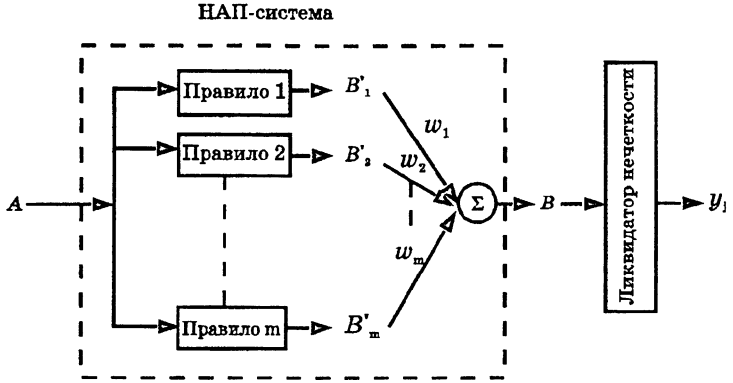


Рис. 3.7.1. Архитектура НАП-системы.
НАМ-ассоциации или правила (от 1 до m) включаются параллельно, создавая выходное нечеткое множество B. Затем выполняется "ликвидация нечеткости": преобразование B в числовую величину y_1

B'_m; оно представляет собой взвешенное среднее частично активированных множеств:

$$B = \sum_{i=1}^{m} w_i B'_i. \qquad (3.7.8)$$

Коэффициент w_1 отражает частоту или силу нечеткой ассоциации (A_1, B_1). На практике выходное нечеткое множество B может быть преобразовано в конкретную числовую величину y_j на действительной оси Y; для этого вычисляется нечеткий центроид множества B относительно выходного универсального множества Y.

3.7.2. Основная концепция нечеткого контроллера

Система управления на основе нечеткой логики показана на рис. 3.7.2. Она состоит из набора НАМ-правил, задающих конкретные ситуации управления. Во многих приложениях

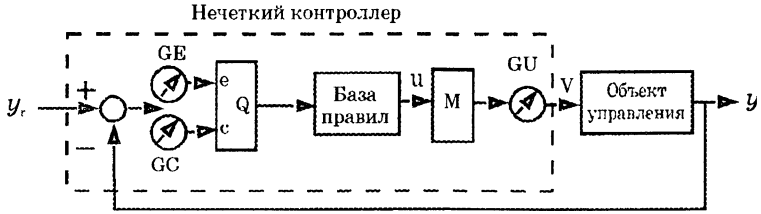


Рис. 3.7.2. Система управления на основе нечеткой логики

достаточно найти соотношение между ошибкой и скоростью изменения ошибки в процессе, чтобы изменить управляющее воздействие на величину, обеспечивающую удовлетворительное управление системой. В этом отношении могут быть сформулированы простые лингвистические правила, основанные на наблюдениях или на простом изучении хода процесса. Приведем пример лингвистического правила на естественном языке, которое эксперт может использовать для описания действия системы управления:

Если ошибка является положительной и большой, а скорость изменения ошибки — отрицательная и малая, то изменение на входе процесса — положительное и большое.

В правиле использованы три переменных: ошибка (E), скорость изменения ошибки (C) и изменение управляющего воздействия (U). Они задаются фиксированными универсальными множествами, определяющими диапазоны измерений величин, возможные в данной системе управления. В некоторых приложениях [66,67] в качестве посылки НАМ-правила, с целью улучшения характеристик системы, может быть добавлена еще какая-либо переменная, например, величина предыдущего управляющего воздействия, однако это приводит к усложнению в разработке системы управления.

Каждая из нечетких переменных может быть преобразована в нечеткие подмножества в соответствии со сложностью процесса. Лингвистические термины (положительное большое, отрицательное малое, нулевое и т.д.), которыми эксперт мо-

жет воспользоваться для описания действий системы управления, используются для обозначения нечетких подмножеств, которые в действительности представляют числовые величины. Каждое из этих нечетких подмножеств содержит элементы вместе со своими степенями принадлежности. Функция принадлежности нечеткого множества ставит в соответствие каждому элементу из универсального множества некоторое вещественное число (обычно — из диапазона от 0 до 1); это число обозначает степень, с которой конкретный объект или элемент данных принадлежит нечеткому множеству.

Например, k-е НАМ-правило может быть записано в виде оператора следующей формы:

Если E есть E_k, и C есть C_k, то U есть U_k. (3.7.9)

Нечеткие подмножества E_k, C_k и U_k задаются набором упорядоченных пар вида:

$$E_k = \{(e, m_{e_k}(e))\} \subset E,$$

$$C_k = \{(c, m_{c_k}(c))\} \subset C,$$

$$U_k = \{(u, m_{u_k}(u))\} \subset U, \qquad (3.7.10)$$

где e, c и u — элементы дискретных универсальных множеств, а m_{ek}, m_{ck} и m_{uk} — соответствующие значения функции принадлежности, указывающие степень, с которой соответствующий элемент является членом нечеткого подмножества.

Графики функций принадлежности нечетких множеств могут иметь различную форму: монотонную, треугольную, трапецеидальную, колоколообразную (форму нормального распределения), в зависимости от предпочтений и опыта разработчика. Примеры основных форм функций принадлежности приведены на рис. 3.7.3. Выбор типа функции принадлежности влияет на используемый тип вывода [68]. На практике широко применяются трапецеидальные и треугольные функции принадлежности, обеспечивающие упрощение вычислений.

Посылки всех НАМ-правил, описывающих управляющее воздействие на процесс, активизируются параллельно (как по-

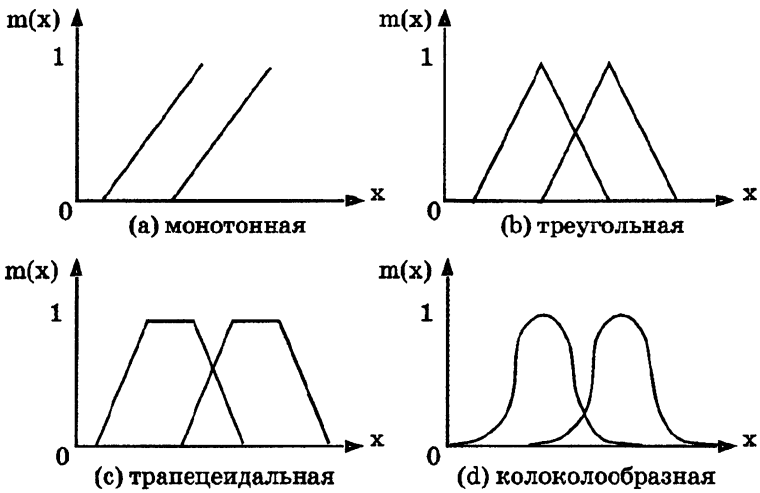


Рис. 3.7.3.Четыре примера функций принадлежности

казано на рис. 3.7.1), когда вид импульса на входах таков, что выбираются единственные соответствующие значения нечетких переменных. Посылка может активизироваться со степенью, равной минимальному или максимальному значению функций принадлежности, в зависимости от того, какая связка использована для нечетких множеств посылки: конъюнктивная (И) или дизъюнктивная (ИЛИ). В приведенном выше примере в посылке использована связка И; в результате нечеткое множество заключения U_k активируется со степенью w_k, вычисляемой, согласно определению операции пересечения (3.7.6), по формуле

$$w_k = m_{e_k}(e) \wedge m_{c_k}(c). \qquad (3.7.11)$$

Форма функции принадлежности выходного нечеткого множества зависит от используемой схемы кодирования НАМ-правила. Широко применяются два метода кодирования: кодирование по минимуму корреляции и по произведению корреляции. При кодировании по минимуму корреляции функция принадлежности нечеткого множества заклю-

чения U_k усекается до степени w_k с минимумом по всем точкам, так, что

$$m_{O_k}(y) = w_k \wedge m_{U_k}(y), \qquad (3.7.12)$$

а при кодировании по произведению корреляции $m_{U_k}(y)$ умножается на величину w_k:

$$m_{O_k}(y) = w_k\, m_{U_k}(y), \qquad (3.7.13)$$

где $m_{0k}(y)$ — степень принадлежности для выходного нечеткого множества в интервале y выходного множества. На рис. 3.7.4 показаны примеры изменения выходного нечеткого множества при использовании кодирования по минимуму корреляции (a) и по произведению корреляции (b).

На практике величина $m_{0k}(y)$ не является единственной; в этом случае для получения функции принадлежности выходного нечеткого множества выполняется объединение всех функций согласно определению нечеткой операции объединения (3.7.4):

$$m_O(y) = m_{O_1}(y) \vee m_{O_2}(y) \vee \ldots \vee m_{O_k}(y). \qquad (3.7.14)$$

Обратите внимание, что здесь k — не количество правил, а индекс квантования выходного нечеткого множества на подмножества.

Выходной ликвидатор нечеткости, т.е. величина изменения выходного сигнала управления u, равен нечеткому центроиду:

$$u = \frac{\int y\, m_O(y)\, dy}{\int m_O(y)\, dy}, \qquad (3.7.15)$$

где пределы интегрирования соответствуют всему универсальному множеству Y.

Для упрощения вычислений Y может подвергаться дискретизации на p значений через постоянные интервалы Dy. Используя дискретное представление $Y=\{y_1, y_2, \ldots, y_p\}$, получим дискретный вариант нечеткого центроида:

$$u = \frac{\sum_{j=1}^{p} y_j m_O(y_j)}{\sum_{j=1}^{p} m_O(y_j)}. \qquad (3.7.16)$$

Нечеткий контроллер имеет ряд параметров, которые должны выбираться разработчиком заранее. В рассматриваемом примере к ним относятся коэффициенты масштабирования GE, GC, GU для ошибки процесса, изменения ошибки и выхода контроллера соответственно. Величины ошибки и изменения ошибки, используемые контроллером, представляют собой элементы соответствующих универсальных множеств дискурса, которые можно представить во времени функциями e(t) и c(t), где t — момент дискретизации. Так как эти величины требуется перевести в количественную величину, соответствующую ближайшему элементу универсального множества дискурса, их сначала требуется масштабировать путем умножения на подходящую величину. Если в момент t выход процесса и уставка имеют значения x(t) и $y_r(t)$ соответственно, то e(t) и c(t) находятся из следующих уравнений:

$$e(kT) = Q[\{y_r(t) - x(t)\}(-GE)], \qquad (3.7.17)$$

$$c(kT) = Q[\{x(t) - x(t-1)\}(-GC)]. \qquad (3.7.18)$$

Аналогично результат преобразования нечеткого центроида в числовую (четкую) форму требуется умножить на соответствующую величину GU, чтобы вычислить действительное изменение входа процесса. Таким образом, если v(t+1) — новая величина входа процесса, то она задается выражением:

$$v(t+1) = v(t) + GUu(t). \qquad (3.7.19)$$

Выбор рассмотренных коэффициентов масштабирования выполняется аналогично выбору параметров ПИД-контроллера или задаваемых пользователем полиномов для некоторых адаптивных контроллеров. Некоторыми исследователями предложены систематические подходы к выбору этих коэффициентов [69,70,71].

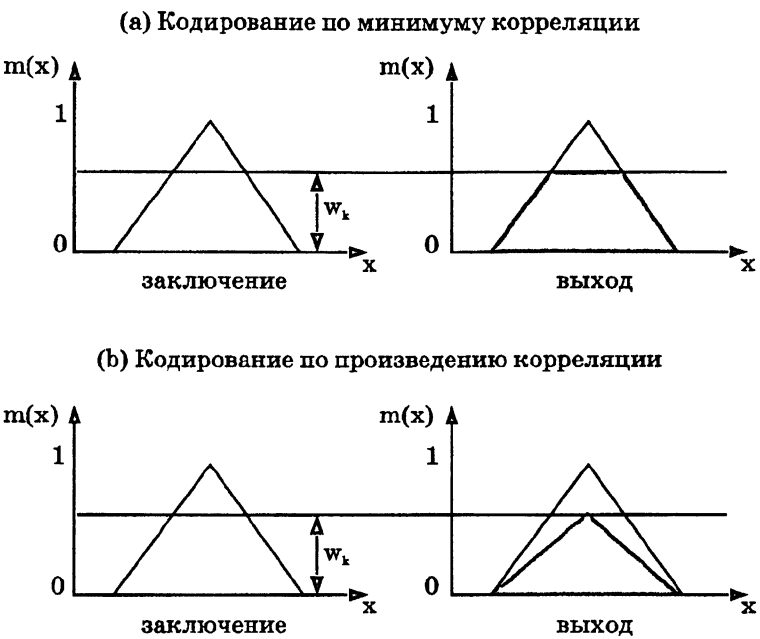


Рис. 3.7.4. Примеры процедур вывода из НАМ-правил, показывающие влияние на выход: (a) кодирование по минимуму корреляции; (b) кодирование по произведению корреляции

3.8. Заключение

В данной главе рассмотрены некоторые традиционные подходы к управлению: обычное ПИД-управление, управление с самонастройкой, ПИД-управление с самонастройкой, обобщенное прогнозирующее управление, управление на основе нечеткой логики. Авторы надеются, что читатель понял некоторые основные принципы этих подходов в достаточной степени, чтобы иметь возможность сравнивать их с подходами на основе нейронного управления, которые рассматриваются в следующей главе, и оценить достоинства и недостатки различных видов управления.

4. ТЕХНОЛОГИИ НЕЙРОННОГО УПРАВЛЕНИЯ

4.1. Введение

Во многих реальных системах имеются нелинейные характеристики, сложные для моделирования динамические элементы, неконтролируемые шумы и помехи, множество обратных связей и другие факторы, затрудняющие реализацию стратегий управления. За последние два десятилетия новые стратегии управления в основном развивались на базе современной и классических теорий управления. Как современная (в частности, технологии адаптивного и оптимального управления), так и классическая теория управления в значительной степени базировались на идее линеаризации систем [1]—[5].

Для практического применения данного подхода необходима прежде всего разработка математических моделей. Однако математическое моделирование, реализуемое на основе предположения о линейности системы, может не отражать ее действительных физических свойств. Даже если удается построить сложные математические модели, точно отражающие физические соотношения между входом и выходом системы, они могут оказаться бесполезными для целей управления. Практически приемлемыми могут быть только модели с низкой чувствительностью по параметрам. Обеспечить это для нелинейных систем достаточно сложно.

Вообще говоря, в качестве «хороших» математических моделей для аппроксимации реальных физических свойств систем показали себя статистические модели [2], например, модель авторегрессии (АР), модель авторегресси — скользящего среднего (АРСС), модель кристаллической решетки и т.д. Эти модели были разработаны как теоретиками, так и инженерами-практиками в области обработки сигналов. Они позволили достичь до-

статочно хороших результатов в решении многих практических задач [3], связанных с распознаванием речи, анализом данных геологических исследований, распознаванием образов и т.д. Достоинство статистических моделей, кроме их простоты, состоит в том, что они допускают проверку на адекватность путем исследования частотных характеристик с использованием спектрального анализа.

Таким образом, при использовании статистических моделей имеется возможность оценить явления, происходящие в определенной полосе частот, например, посредством анализа частотных характеристик методами классической теории управления. Такой статистический подход может показаться очень эффективным, однако используемые при этом приближения часто не обеспечивают достаточной точности для целей управления. Причина этого недостатка заключается в попытке приспособить линейные модели к сложным нелинейным системам. Кроме того, данные методы основаны на моделях «черного ящика», используемые в них переменные и параметры в некоторых случаях не имеют физического смысла. Это только некоторые из причин, по которым такие подходы не находят широкого применения в практике управления [4].

Другой распространенный подход к управлению — адаптивное управление, рассмотренное в предыдущей главе [5]. При этом подходе предполагается наличие математической модели, основанной на физических явлениях, и оцениваются неизвестные параметры, включенные в эту модель. Затем определяется закон управления, направленный на достижение некоторой цели (степень достижения оценивается функцией стоимости); при этом математическая модель рассматривается как реальная система. Такой подход основывается также на теории линейных систем. При каких-либо изменениях в объекте управления или во внешних условиях требуется перестраивать модель и определять для нее новый закон управления. Таким образом, требуется «вручную» проверять, является ли модель адекватной реальной физической системе.

В связи с этим современная теория управления была не слишком популярной, ее было сложно применять для реальных задач, кроме некоторых специфических, например, для задачи управления спутниками. В Японии примерно 84% систем управления до сих пор построены на основе обычных ПИД-контроллеров [4].

Из приведенных фактов можно сделать вывод: для того чтобы алгоритмы управления могли применяться на практике, они должны быть достаточно простыми для реализации и понимания. Кроме того, они должны обладать способностью к обучению, гибкостью, устойчивостью, нелинейностью. Алгоритмы, основанные на нечеткой логике, обладают некоторыми из указанных свойств, благодаря чему они получили в настоящее время достаточно широкое распространение.

В последнее время для целей управления все шире начинают применяться нейронные сети. Они показали свою эффективность для решения задач распознавания образов. Нейронные сети способны обучаться на основе соотношений «вход-выход», поэтому они могут обеспечить более простые решения для сложных задач управления. Кроме того, нейроны — это нелинейные элементы; следовательно, нейронные сети в своей основе являются нелинейными системами, пригодными для решения задач управления, принципиально связанных с наличием нелинейных характеристик. Традиционные методы управления не обеспечивают решения подобных задач. Таким образом, в последнее время интеллектуальное управление стало достаточно подходящим для решения реальных задач [5,6].

4.2. Общие сведения о нейронном управлении

Нейронным управлением называется применение полностью определенных нейронных сетей (искусственных или естественных) для выработки действительных управляющих сигналов. Обычно когда говорят об «управлении», под этим термином подразумевают определенную форму интеллекта.

Например, мы должны управлять собой, чтобы не допускать противозаконных поступков. Во многих религиях от людей требуется соблюдать определенные ограничения; для этого люди также должны управлять собой. Когда мы контролируем свои чувства и эмоции, также имеется элемент мышления, связанный с определенной «формой» интеллекта. В технике, когда разрабатывается контроллер для управления некоторым объектом или процессом в соответствии с заданными параметрами, также можно сказать, что имеется некоторый процесс «мышления», который должен осуществляться контроллером, чтобы обеспечить желаемую работу объекта управления. Одна из характеристик систем управления с обратной связью — их способность к «мышлению» в том смысле, что они могут (в некоторой степени) заменять человека-оператора. Норберт Винер [7] в свое время утверждал, что все системы, как живые, так и неживые, являются информационными системами и системами управления с обратной связью. Он считал, что теория управления, теория информации и теория нейронных систем должны рассматриваться как разделы одной науки — кибернетики. В действительности, однако, наблюдается тенденция самостоятельного развития этих отраслей знания. В последнее десятилетие эффективным средством реализации механизмов обучения оказался алгоритм обратного распространения. Почти во всех областях техники исследователи стали активно применять механизмы, основанные на нейронных сетях, для нахождения лучших решений по сравнению с обычными методами. В области управления нейронные сети оказались подходящим средством для решения сложных нелинейных задач управления, для которых обычные методы управления не дают удовлетворительных с практической точки зрения решений. По мнению авторов, имеется несколько причин, породивших интерес к применению нейронных сетей для задач управления в качестве альтернативы традиционным методам. Основные причины заключаются в следующем.

- Нейронные сети могут обучаться любым функциям [8,9,10]; требуется только, чтобы в процессе обучения был предоставлен достаточно большой объем информации, а также правильный выбор самой нейронной модели. Таким образом, способность нейронных сетей к самообучению избавляет от необходимости использовать сложный математический аппарат в отличие от многих традиционных методов адаптивного и оптимального управления.
- Включение сигмоидных функций активации (или некоторых нелинейных функций общего вида [11]) в скрытые нейроны многослойных нейронных сетей обеспечивает возможность реализации нелинейных отображений. Это важно для решения задач управления с существенными нелинейностями, для которых традиционные подходы пока не дают практически реализуемых решений. Возможно, это достоинство нейронных сетей — наиболее важное с точки зрения теории управления.
- Необходимым условием применения традиционных методов оптимального и адаптивного управления является наличие большого объема априорной информации об объекте управления, например, данных математического моделирования [1],[12]. Благодаря способности нейронных сетей к самообучению, для нейроконтроллеров такой объем информации не требуется. В связи с этим можно полагать, что нейроконтроллеры пригодны для управления в условиях существенных неопределенностей [13],[14].
- Высокая степень параллельности нейронных сетей позволяет реализовывать очень быстрые методы мультипроцессорной обработки на основе использования нейронных кристаллов или параллельных аппаратных средств. Фирмой American Neurologix Inc. разработан нейронный процессор (NLX420), уже поступивший на рынок [15].

- Благодаря реализуемой в нейронной сети архитектуре параллельной обработки, повреждение отдельных элементов технических средств сети не может существенно влиять на работу сети в целом [16], [17].

В последнее время появилось огромное количество научных статей и докладов, посвященных разработкам в области искусственных нейронных сетей в контексте теории управления. Это направление нейронного управления становится все более важным, и его уже нельзя рассматривать как несущественное преувеличение, как это когда-то делалось некоторыми крупными учеными в области теории управления. Одной из первых значительных работ по нейронному управлению стала книга Миллера (Miller) и др. «Нейронные сети в управлении» [18], которая содержит важные ранние результаты по нейронному управлению.

В статье Барто (Barto) [19], упомянутой в первой главе этой книги, рассматривается архитектура нейронных сетей для обучения, идентификации и управления. В книге Вербоса (Werbos) [20] делается попытка (впрочем, достаточно спорная) классификации приложений нейронного управления. Например, выделение в этой классификации «диспетчерского управления» и «инверсно-прямого управления» является в некоторой степени запутывающим, так как диспетчерское управление может представлять собой частный случай инверсно-прямого управления, и наоборот. Кроме того, некоторые специалисты утверждают, что выделяемое в этой классификации «нейронное адаптивное управление» мало отличается от «обратного распространения во времени», что также вносит неясность. Хотя многие считают, что попытка классификации была предпринята слишком рано, данная работа [20] заслуживает определенного доверия.

Выше авторами предпринята попытка классификации некоторых подходов к нейронному управлению на основе собственной точки зрения, которая будет подробно приведена в этой главе. По-видимому, до настоящего времени одним из

самых существенных исследований перспектив нейронного управления является работа Ханта (Hunt) и др. [21]. В этой работе очень уместным является указание на параллели между теориями управления и нейронных сетей. В ней также рассматриваются почти все основанные на нейронных сетях парадигмы, которые применялись исследователями для задач идентификации и управления. Некоторые из основных архитектур нейронного управления, предложенных к настоящему времени и широко обсуждаемых в литературе, будут рассмотрены ниже.

4.2.1. Подходы к нейронному управлению

Возможно, одной из первых значительных работ по нейронному управлению стала работа Видроу (Widrow) и Смита (Smith) [22]. Они показали, что их система ADALINE способна обеспечить стабилизацию неустойчивого маятника на подвижной тележке, копируя работу существующей системы, управляемой человеком. Такой способ управления представляет собой одну из форм «диспетчерского управления». В работе Албуса (Albus) [23],[24] предложен новый подход к управлению манипулятором (рукой робота), который основан на использовании разновидности нейронной сети с просмотровой таблицей. Такая сеть получила название ЦАК (церебральный артикуляционный контроллер). Сотрудники Нью-Гэмпширского университета Миллер и др. [25] применили такую сеть для управления роботом и для решения других практических задач. В 1983 г. Барто и др. [26] выдвинули идею адаптивной обучающейся системы, состоящей (если использовать терминологию, предложенную авторами) из одного ассоциативного поискового элемента (АПЭ) и одного адаптивного критического элемента (АКЭ). Такой прием был применен для стабилизации системы, как показано на рис. 4.2.1. Вербос [19] классифицировал такой подход как «адаптивно-критический». Кроме того, были и другие работы по исследованию и применению данного подхода, основанного на назначении доверия [27—32].

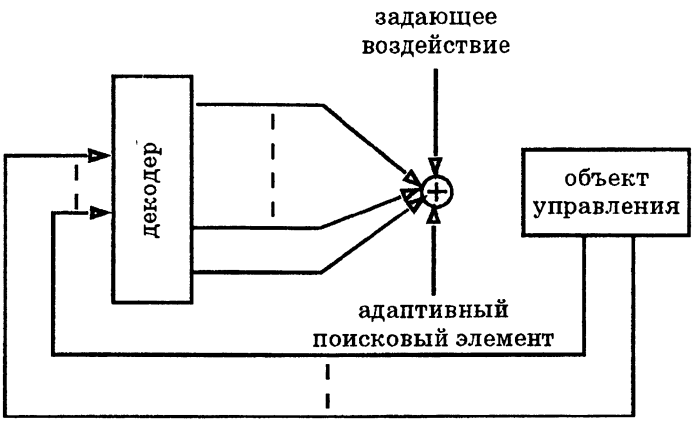


Рис. 4.2.1. Адаптивно-критический метод Барто, Саттона и Андерсена [26]

Успешное применение алгоритма обратного распространения для обучения многослойных сетей дало толчок широкому применению нейронных сетей в управлении. Предлагались различные схемы нейронного управления, основанные на алгоритме обратного распространения. Возможно, одной из наиболее распространенных схем стала схема нейронного управления на основе инверсно-прямой модели. Популярность этого подхода связана с его простотой. Сущность подхода состоит в следующем: сначала нейронная сеть обучается на инверсии объекта управления, а затем она может быть настроена на непосредственное управление этим объектом.

Идея этой схемы была, по-видимому, заимствована из традиционной схемы управления с самонастройкой [1], в которой требуемое управляющее воздействие определяется из инверсной математической модели путем задания желаемого выходного сигнала объекта. В работе [33] показаны некоторые примеры применения схемы управления на основе инверсной модели для линейных систем с использованием нейронных сетей. Другие примеры применения такого подхода приводятся в работах [35]—[37].

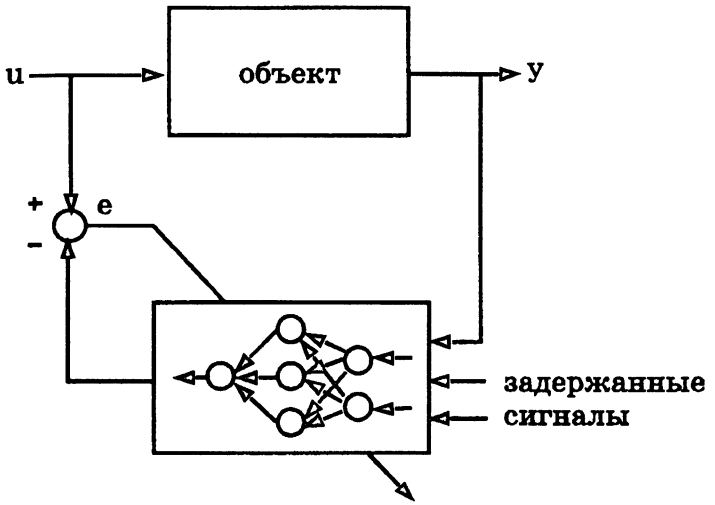


Рис. 4.2.2. Обобщенное обучение, или архитектура прямого инверсного управления. Модель нейронной сети обучается автономно на инверсной модели объекта управления. По окончании обучения она настраивается для непосредственного управления объектом.

Работа Псалтиса (Psaltis) и др. [38], по-видимому, одна из наиболее цитируемых работ в области нейронного управления. В ней предложены два простых, но эффективных метода обучения нейронных сетей или две инверсные модели объекта управления, использующих алгоритм обратного распространения. Они называются архитектурами обобщенного и специализированного обучения. В архитектуре обобщенного обучения, показанной на рис. 4.2.2, сеть обучается автономно, с использованием образцов, полученных по характеристикам разомкнутого или замкнутого объекта управления. Этот метод аналогичен обучению на примерах в задаче распознавания образов. Обученная таким образом сеть настраивается на работу в качестве контроллера для объекта управления, подобно обычной системе управления с обратной связью. Архитектура специализированного обучения (рис. 4.2.3) исполь-

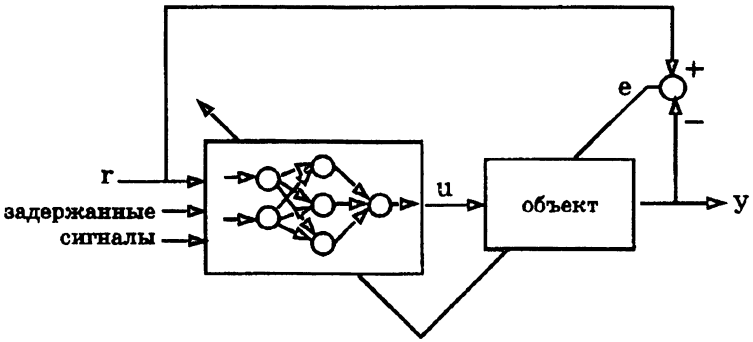


Рис. 4.2.3. Архитектура специализированного обучения. Модель нейронной сети непосредственно обучается на инверсной модели объекта управления (на основе обратного распространения ошибки)

зуется для непосредственного (или «управляемого целью») обучения сети; при этом ошибка выполнения распространяется по сети в обратном направлении при каждой выборке. Однако при этом требуется знать якобиан объекта управления. Чтобы избежать возникающих при этом трудностей, вместо элементов якобиана могут использоваться знаки этих элементов, которые представляют собой направления воздействия параметров управления на выходы объекта управления (как показано в [39]).

Еще одной инверсной (на этот раз, однако, непрямой) моделью непосредственного обучения для нейронного управления, использующей алгоритм обратного распространения, стала схема «обучения с ошибкой обратной связи», предложенная в работах [40],[41]. В этой схеме (показанной на рис. 4.2.4) нейронная сеть настраивается параллельно обычному контроллеру с обратной связью (будем называть такую архитектуру «схемой параллельного нейронного управления»). Сеть обучается непосредственно, путем повторения циклов желаемой траектории; при этом ошибка обратной связи распространяется по сети в обратном направлении. Сходимость достигается, когда нейронная сеть, завершив обучение на инверсии объекта уп-

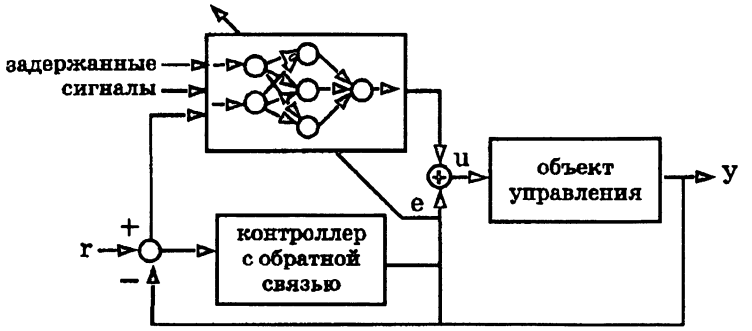


Рис. 4.2.4. Архитектура обучения с ошибкой обратной связи. Модель нейронной сети непосредственно обучается на инверсии модели объекта управления путем обратного распространения выходного сигнала контроллера с обратной связью

равления, принимает на себя управление объектом, устраняя действие контроллера с обратной связью.

Этот метод управления аналогичен, например, обучению игре в теннис. Теннисист-новичок сначала не умеет правильно ударять по мячу и поэтому старается делать это как можно точнее (пока не научится чему-то, подобному обычному управлению). Однако после определенной практики игрок научится более правильно управлять размахом руки при ударе по мячу (высококачественное управление после окончания обучения). Метод обучения на основе ошибки обратной связи состоит в многократном повторении цикла желаемой траектории. Поэтому его практическое применение ограничивается только некоторыми типами систем, в частности, системами управления рукой робота и сервомоторами.

Еще одна важная архитектура нейронного управления, использующая алгоритм обратного распространения — это схема, независимо предложенная в работах [42 — 45] и [20]. Вербос (Werbos) этот метод классифицирует как «обратное распространение во времени» [20]. Хотя Джордан (Jordan) и Румельхарт (Rumelhard) утверждают [43], что они первыми

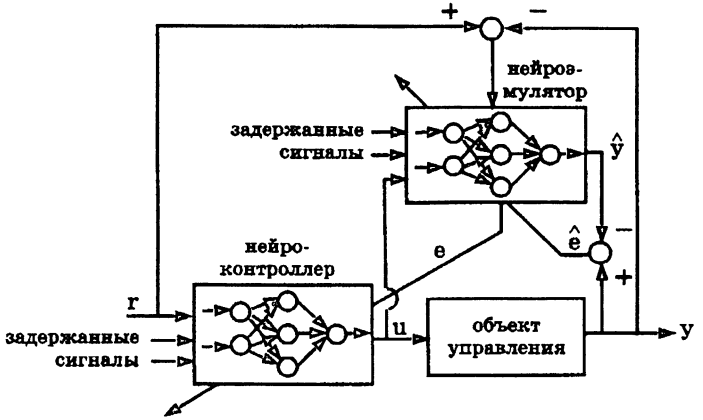


Рис. 4.2.5. Схема нейронного управления с эмулятором и контроллером, или «обратное распространение во времени». Нейроконтроллер обучается на инверсной модели объекта управления, а нейроэмулятор — на обычной модели объекта управления. Нейроконтроллер может обучаться непосредственно, на основе обратного распространения ошибки через нейроэмулятор

разработали этот подход, данная архитектура не представляет собой ничего нового с точки зрения специалистов по теории управления, так как она во многом сходна с одной из традиционных структур адаптивного управления — косвенным адаптивным управлением. В этой схеме для управления объектом используются две нейронные сети, как показано на рис. 4.2.5. Первая сеть используется как эмулятор, вторая — как контроллер. Сеть-эмулятор может обучаться автономно, с использованием архитектуры обобщенного управления [38], или даже непосредственно, путем ввода случайных входных сигналов для обучения динамике объекта управления.

Данная архитектура обеспечивает более точное непосредственное обучение нейроконтроллера, так как ошибка может распространяться в обратном направлении через эмулятор в каждой выборке. В главе 5 будет показано применение такой архитектуры для двух практических задач.

До сих пор при рассмотрении различных технологий нейронного управления речь шла об объекте управления вообще, хотя в последней из рассмотренных схем присутствует определенная идентификация объекта управления за счет использования нейроэмулятора. Однако в данном случае нейроэмулятор используется просто как средство для обратного распространения ошибки, с целью получения эквивалентной ошибки на выходе нейроконтроллера. Благодаря такой способности к обучению нейронные сети могут также использоваться для идентификации системы, что стало одним из основных факторов разработки многих традиционных адаптивных контроллеров. Широкие исследования (не рассматриваемые в данной работе) проведены также в области применения нейронных сетей для идентификации объекта управления. Среди работ, посвященных применению нейронных сетей для идентификации систем, можно упомянуть работы [44], [46 — 51].

С точки зрения авторов данной книги, большинство схем нейронного управления, разработанных до настоящего времени, основаны на следующих подходах.

1. Последовательная схема управления. Нейронная сеть непосредственно обучается отображению желаемых (опорных) сигналов в управляющие воздействия, необходимые для получения таких сигналов.
2. Параллельная схема управления. Нейронная схема используется для компенсации управляющего воздействия, задаваемого обычным контроллером. Компенсация производится таким образом, чтобы выходной сигнал объекта управления поддерживался как можно ближе к желаемому.
3. Схема управления с самонастройкой. Нейронная сеть настраивает параметры управления, задающие работу обычного контроллера, таким образом, чтобы выходной сигнал объекта управления поддерживался как можно ближе к желаемому.
4. Схема управления с эмулятором и контроллером, или схе-

ма обратного распространения во времени. Максимизируется некоторая мера полезности или эффективности во времени, однако при этом эффективный учет действия шумов не обеспечивается, и действительное обучение в реальном времени не достигается (из-за медленной сходимости процесса обучения).

5. Адаптивно-критическая схема. Эта схема приближена к динамическому программированию, т.е. к реализации оптимального управления во времени в условиях шумов и нелинейностей.

Все эти методы нейронного управления, за исключением последнего, будут подробно рассмотрены в последующих разделах на основе алгоритма обратного распространения.

4.2.2. Общая схема управления

Блок-схема общей схемы управления показана на рис. 4.2.6. Для получения желаемых результатов управления, т.е. для поддержания максимального соответствия выходного сигнала объекта управления опорному сигналу, можно настраивать два элемента схемы: контроллер прямой связи (КПС) и контроллер обратной связи (КОС). На схеме они обозначены стрелками.

Контроллер обратной связи (КОС) используется для стабилизации замкнутой системы. Стабилизация достигается на основе сигнала ошибки, представляющего собой разность между опорным сигналом и выходным сигналом объекта управления. Если объект управления имеет существенное временное запаздывание, то воздействие управляющего входного сигнала на выход объекта управления будет проявляться с запаздыванием; получаемый таким образом выходной сигнал будет подаваться обратно (по цепи обратной связи) и сравниваться с опорным сигналом. По разности между сигналами определяется очередной входной управляющий сигнал (управляющее воздействие). Таким образом, временное запаздывание может привести к тому, что замкнутая система потеряет устойчивость.

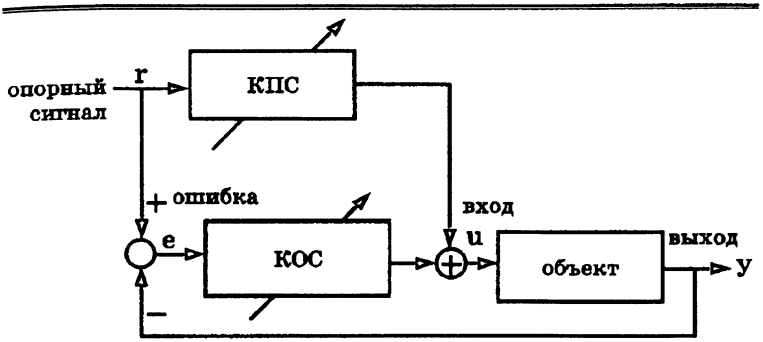


Рис. 4.2.6. Общая схема управления (при отсутствии возмущающих воздействий)

При отсутствии возмущений контроллер прямой связи (КПС) не обеспечивает полной гарантии того, что выходной сигнал объекта управления будет соответствовать опорному сигналу, однако обеспечивает быструю передачу выработанного входного сигнала на объект управления. В результате обеспечивается высокая скорость реакции системы. Проще говоря, КОС обеспечивает стабильность системы, а КПС — быстроту ее реакции. Таким образом, если требуется, чтобы выходной сигнал объекта управления быстро изменялся в соответствии с опорным сигналом, то следует использовать КОС. Ниже будет рассмотрена реализация блоков КОС и КПС на основе нейронных сетей.

4.3. Последовательная схема нейронного управления

Последовательная схема нейронного управления показана на рис. 4.3.1. Для такого типа нейронного управления используется также название «инверсно-прямое управление»; это означает, что нейронная сеть может реализовывать обратную динамику объекта управления.

Если обозначить соответствие между входом и выходом как $f_p(u)$, т.е.

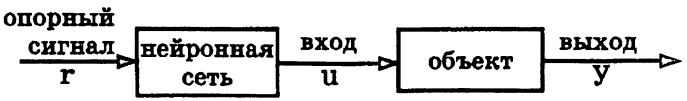


Рис. 4.3.1. Последовательная схема нейронного управления

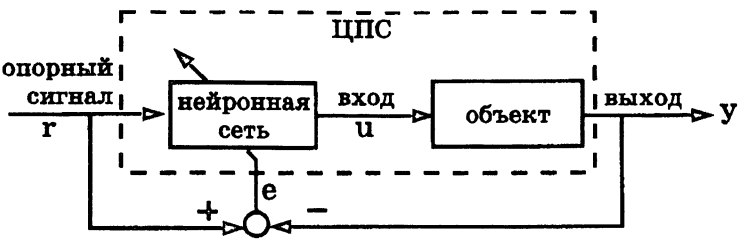


Рис. 4.3.2. Специализированное нейронное управление с обратным отображением; пунктирной линией выделена цепь прямой связи (ЦПС)

$$y = f_p(u), \tag{4.3.1}$$

то нейронная сеть реализует обратное отображение:

$$u = f_p^{-1}(y),$$

где f_p^{-1} — отображение, обратное отображению f_p.

Таким образом, если подать на нейронную сеть опорный сигнал r, то выходной сигнал объекта управления y принимает значение r, так как

$$y = f_p(u) = f_p(f_p^{-1}(\tau)) = \tau, \tag{4.3.2}$$

где u — входной сигнал объекта управления, соответствующий опорному сигналу r, а y — выходной сигнал, соответствующий u.

Можно использовать два способа реализации схемы, показанной на рис. 4.3.1. Первый из них, называемый «специализированным обратным отображением», показан на рис. 4.3.2. Нейронная схема, показанная на ней, обучается таким образом, чтобы минимизировать квадрат ошибки. Если ошибка становится близкой к нулю, можно считать, что нейронная сеть

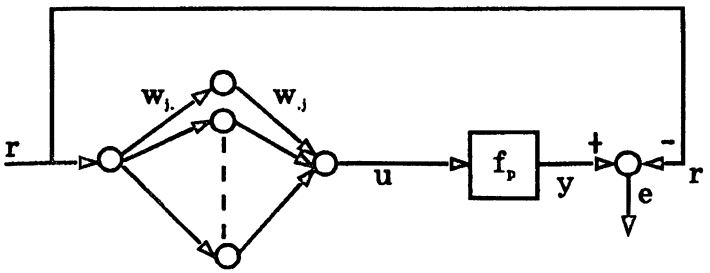


Рис. 4.3.3. Внутренняя структура нейроконтроллера, используемого в схеме, показанной на рис. 4.3.2. Величины $w_{.j}$ и $w_{j.}$ соответствуют коэффициентам w_{kj} и w_{ji} в разделе 2.2.1.

реализует инверсную динамику объекта управления. Заключенная в пунктирную линию часть схемы, называемая цепью прямой связи (ЦПС), может рассматриваться как разновидность нейронной сети, у которой выходной слой жестко задан. Ниже будет показано, что в этом случае для использования алгоритма обратного распространения необходимо знать якобиан системы.

Для схемы, показанной на рис. 4.3.2, простейшая внутренняя структура нейронной сети показана на рис. 4.3.3, где используется один нейрон во входном и один в выходном слоях. На рис. 4.3.3 входным сигналом для нейронной сети является только сигнал r, а выходным — только сигнал u. Для простоты обозначим отношение «вход/выход» для объекта управления как f_p. При этом желательно, чтобы по результатам обучения выходной сигнал имел значение как можно ближе к r. Таким образом, подлежащая минимизации функция ошибки записывается в следующей форме:

$$E = \frac{1}{2}e^2, \tag{4.3.3}$$

$$e = \tau - y, \tag{4.3.4}$$

$$y = f_p(u). \tag{4.3.5}$$

Входной сигнал объекта управления задается следующим образом:

$$u = f(net_{\bullet}), \tag{4.3.6}$$

$$net_{\bullet} = \sum_{j} w_{\bullet j} O_j + \theta_{\bullet}, \tag{4.3.7}$$

$$O_j = f(net_j), \tag{4.3.8}$$

$$net_j = w_{j\bullet} r + \theta_j, \tag{4.3.9}$$

где $w_{\bullet j}$ и $\theta_{\bullet}$ — веса связей и пороговая величина соответственно для выходного слоя, а $w_{j\bullet}$ — вес связи для скрытого слоя. Используя принцип наискорейшего спуска, получим

$$\Delta w_{\bullet j}(t+1) = -\eta \frac{\partial E}{\partial w_{\bullet j}} + \alpha w_{\bullet j}(t). \tag{4.3.10}$$

Используя цепное правило получения частных производных, правило дифференцирования сложных функций, получим:

$$\frac{\partial E}{\partial w_{\bullet j}} = \frac{\partial E}{\partial net_{\bullet}} \frac{\partial net_{\bullet}}{\partial w_{\bullet j}}. \tag{4.3.11}$$

Из (4.3.7) получим

$$\frac{\partial net_{\bullet}}{\partial w_{\bullet j}} = O_j. \tag{4.3.12}$$

Определяя d. как

$$\delta_{\bullet} = \frac{\partial E}{\partial net_{\bullet}} \tag{4.3.13}$$

и снова используя цепное правило для $\delta_{\bullet}$, получим:

$$\begin{aligned} \delta_{\bullet} &= -\frac{\partial E}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial net_{\bullet}} = e\frac{\partial y}{\partial u}\frac{\partial u}{\partial net_{\bullet}} = \\ &= e f'_p(u) f'_p(net_{\bullet}) = e f'_p(u) u(1-u). \end{aligned} \tag{4.3.14}$$

Таким образом, имеем следующее соотношение:

$$\Delta w_{\bullet j}(t+1) = \eta \delta_{\bullet} O_j + \alpha w_{\bullet j}(t), \tag{4.3.15}$$

$$\delta_{\bullet} = e f'_p(u) u(1-u). \qquad (4.3.16)$$

Аналогично для Δw_j. получим:

$$\Delta w_{j\bullet}(t+1) = \eta \delta_j r(t) + \alpha w_{j\bullet}(t) \qquad (4.3.17)$$

и

$$\delta_j = -\frac{\partial E}{\partial net_j} = -\frac{\partial E}{\partial net_{\bullet}} \frac{\partial net_{\bullet}}{\partial O_j} \frac{\partial O_j}{\partial net_j}, \qquad (4.3.18)$$

$$\delta_j = \delta_{\bullet} w_{\bullet j} f'(net_j) = \delta_{\bullet} w_{\bullet j} O_j (1 - O_j). \qquad (4.3.19)$$

Таким образом, получен следующий алгоритм.

Шаг 1. Установить начальные значения $w_{\cdot j}$, $w_{j\cdot}$, θ_j, $\theta_{\cdot}$, η, и α. Положить t=0 и перейти к шагу 2.

Шаг 2. Подать на вход нейронной сети сигнал, равный желаемому выходному сигналу r(t), и вычислить значения сигналов u(t) и y(t). Затем вычислить $\delta_{\cdot}$ по формуле:

$$\delta_{\bullet} = u(t)(1 - u(t)) e(t) f'_p(u(t)),$$
$$e(t) = r(t) - y(t)$$

Шаг 3. Изменить веса связей в соответствии с выражением:

$$\Delta w_{\bullet j}(t+1) = \eta \delta_{\bullet} O_j + \alpha w_{\bullet j}(t).$$

Шаг 4. Вычислить значение δ_j:

$$\delta_j = \delta_{\bullet} w_{\bullet j} O_j (1 - O_j).$$

Шаг 5. Изменить веса связей в соответствии с выражением:

$$\Delta w_{j\bullet}(t+1) = \eta \delta_j r(t) + \alpha w_{j\bullet}(t).$$

Шаг 6. Заменить $t \to t+1$ и перейти к шагу 2.

При практическом применении этого алгоритма возникают, однако, некоторые трудности. Одна из них состоит в том, что соответствие «вход — выход», задаваемое нейронной сетью, показанной на рис. 4.3.3, является не динамическим, а скорее статическим отображением координат u на y. В практических задачах управления динамические отображения явля-

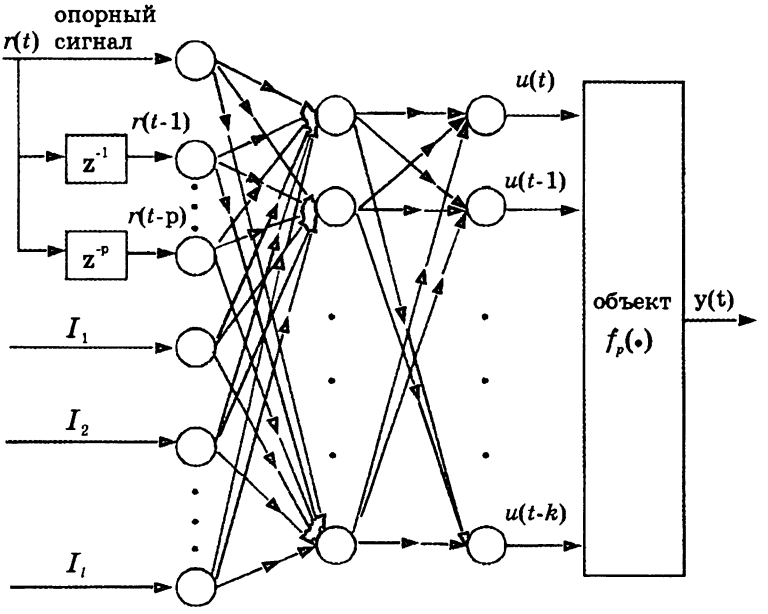


Рис. 4.3.4. Общая схема последовательного нейроконтроллера

ются более привычными. Поэтому нейронная сеть должна непосредственно подавать на вход объекта управления много сигналов, равных прошлым входным сигналам, или эти прошлые входные сигналы должны сохраняться. В первом случае выходами нейронной сети будет не только значение u(t) в текущий момент времени, но и значения того же процесса в предшествующие моменты u(t-m), m=1,2,...,q. Во втором — входами нейронной сети становятся значения r(t-i), i=1,2,...,k.

Кроме того, можно подавать больше информации на вход нейронной сети, чтобы ускорить сходимость обучения. Если известны какие-либо факторы, способные улучшить управляющий сигнал, то следует использовать такую информацию в качестве входного вектора нейронной сети.

Общая схема последовательного нейроконтроллера приведена на рис. 4.3.4. Здесь символами I_1, I_2, ... , I_l обозначена

дополнительная информация, которая может влиять на управляющий сигнал. Однако если величины временной задержки q и p точно не известны, то использовать такую структуру нельзя. В этом случае можно использовать оценки величин q и p.

При использовании схемы управления, показанной на рис. 4.3.4, для обучения нейронной сети можно применить методику, которая аналогична описанной выше. Таким образом, если приравнять соответствующие величины:

$$w_{\cdot j} \rightarrow w_{kj},$$

$$w_{j\cdot} \rightarrow w_{ji},$$

$$\delta_{\cdot} \rightarrow \delta_{k},$$

$$f_p(u(t)) \rightarrow f_p(u(t), u(t-1), \ldots, u(t-q)), \qquad (4.3.20)$$

то для обучения нейроконтроллера можно применить вышеописанный алгоритм (шаги 1 — 6).

Необходимо отметить, что данный алгоритм основан на алгоритме обратного распространения, в котором в качестве функции активации используется сигмоидная функция; причем в данном случае все входные управляющие сигналы могут принимать значения между 0 и 1. Если в качестве управляющих входных сигналов используются только положительные величины, то управляющий сигнал u необходимо умножить на некоторую положительную константу C, а если в качестве управляющих входных сигналов используются только отрицательные величины, то управляющие сигналы требуется умножать на отрицательную константу C. Для определения величины C можно включить дополнительный слой. В более простом случае, если в качестве функции активации для выходного слоя используется функция идентичности, то можно выбрать любую величину (положительную или отрицательную).

В этом случае шаг 2 принимает следующий вид.

Шаг 2. Подать на вход нейронной сети сигнал, равный желаемому выходному сигналу r(t), и вычислить u(t) и y(t). Затем вычислить $\delta_{\cdot}$ по формулам:

$$\delta_* = e(t)\, f'_p(u(t)),$$
$$e(t) = r(t) - y(t).$$

В рассмотренном выше алгоритме единственная сложность состоит в вычислении якобиана системы $f'_p(u(t))$ на шаге 2. Как правило, выражение f'_p неизвестно, и требуется искать его аппроксимацию. Один из способов состоит в использовании численной аппроксимации:

$$f'_p(u(t)) = \frac{\Delta f_p(u(t))}{\Delta u(t)},$$

где

$$\Delta f_p(u(t)) = f_p(u(t-1)) - f_p(u(t-2)),$$
$$\Delta u(t) = u(t-1) - u(t-2).$$

Другой подход состоит в том, чтобы выполнить имитационное моделирование объекта управления, используя другую нейронную сеть, реализующую отображение «вход — выход» для объекта управления. В этом случае данная нейронная сеть используется в качестве идентификатора системы. Для нейронных сетей вместо термина «идентификатор системы» применяется термин «эмулятор». Таким образом, используя эмулятор, можно вычислить якобиан системы, как если бы эмулятор представлял собой действительный объект управления.

Итак, $f'_p(u(t)) = \dfrac{\partial \hat{f}_p(u(t))}{\partial u(t)}$,

где $\hat{f}_p$ — отображение «вход/выход» для эмулятора. Простая процедура обучения для эмулятора показана на рис. 4.3.5.

На рис. 4.3.5 показана схема обучения нейронной сети, при котором минимизируется ошибка предсказания $e = y - \hat{y}$, где $\hat{y}$ — выходной сигнал эмулятора, обозначающий предсказанную оценку величины y. Более подробно эта схема обучения показана на рис. 4.3.6. Если известны порядки временных задержек входных и выходных сигналов, то они используются для определения величин p и q (рис. 4.3.6).

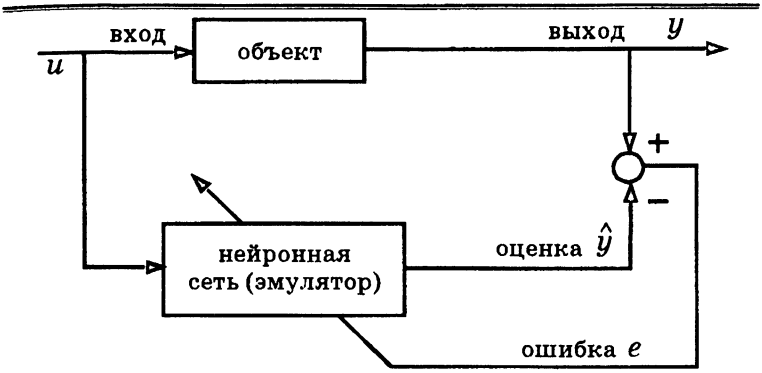


Рис. 4.3.5. Простая схема обучения для получения нейроэмулятора

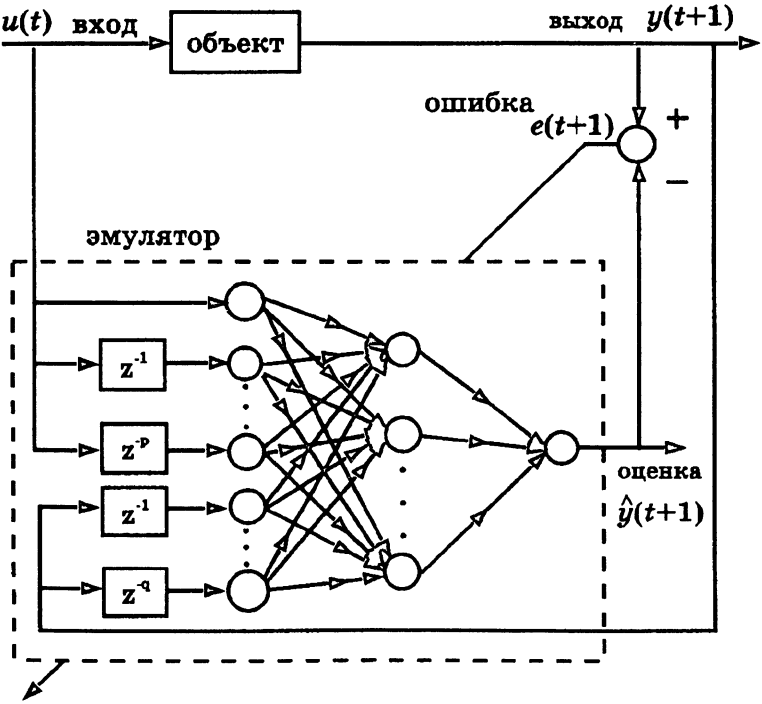


Рис. 4.3.6. Внутренняя структура нейроэмулятора. Величины p и q — порядки временных задержек входных и выходных сигналов соответственно

Для ускорения сходимости процесса обучения можно использовать другую модель эмулятора, показанную на рис. 4.3.7. Здесь в качестве условной модели используется простая математическая модель или физически точная модель, даже если сама система является достаточно сложной и нелинейной. Если между действительным выходным сигналом и выходным сигналом модели имеется несоответствие, то нейронная сеть выполняет самонастройку с целью минимизации ошибки. Выход модели обозначен как $\hat{y}$, а выход нейронной сети — как $\hat{y}_2$. Их сумма представляет собой оценку предсказания действительного выхода $\hat{y}$; расхождение между величинами y и $\hat{y}$ обозначается как ε.

Если величина e не равна нулю, то нейронная сеть обучается с целью минимизации ошибки ε. Таким образом, эмулятор рассматривается здесь как «сумма» условной модели и нейронной сети, но настраиваться (изменяться) может только нейронная сеть. Этот метод моделирования является подходящим для практических задач управления, так как условная модель может не обеспечивать достаточной точности предсказания действительного выходного сигнала.

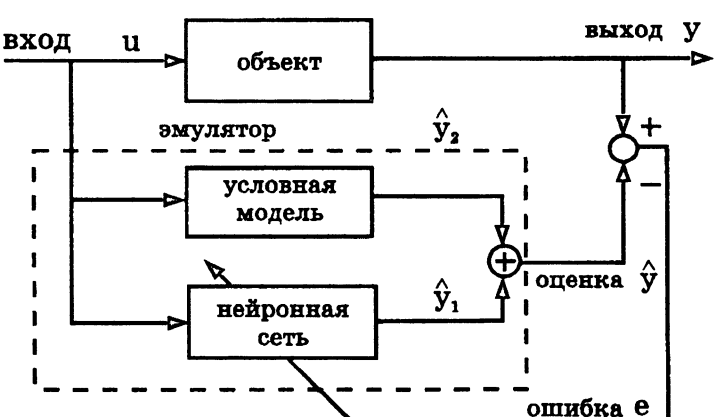


Рис. 4.3.7. Альтернативный метод моделирования с использованием нейронных сетей.

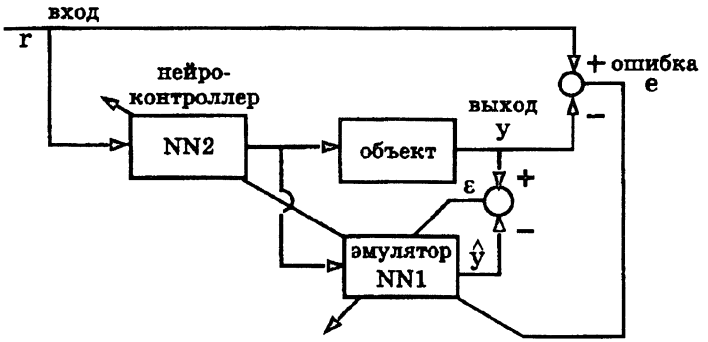


Рис. 4.3.8. Общая схема инверсно-динамического нейроконтроллера. Эмулятор NN1 реализован по схеме, показанной на рис. 4.3.6 и 4.3.7, а нейроконтроллер NN2 — по схеме рис. 4.3.4.

Основываясь на собственном опыте, в качестве условных моделей авторы могут рекомендовать использование статической модели, модели авторегрессии (АР), модель авторегрессии — скользящего среднего (АРСС), линейных векторных уравнений и т.п., так как они позволяют выявить долгосрочные тенденции. В свою очередь, нейронная сеть позволяет реализовать нелинейные соотношения. Таким образом, общая конфигурация схемы управления на основе нейронной сети показана на рис. 4.3.8, где NN1 — нейросетевой эмулятор (или просто — нейроэмулятор), а NN2 — нейроконтроллер, реализующий инверсно-динамическое отображение.

Другая схема реализации инверсной динамики с помощью нейронной сети показана на рис. 4.3.9, где нейронная сеть NN1 используется для обучения инверсному отображению, а математическая модель NN2 копируется с модели NN1 после того, как заканчивается обучение последней. Эта архитектура, предложенная в [38], описана в разделе 4.2.1. Реализация нейронной сети NN1 показана на рис. 4.3.10, где p и q — временные задержки входных и выходных сигналов соответственно.

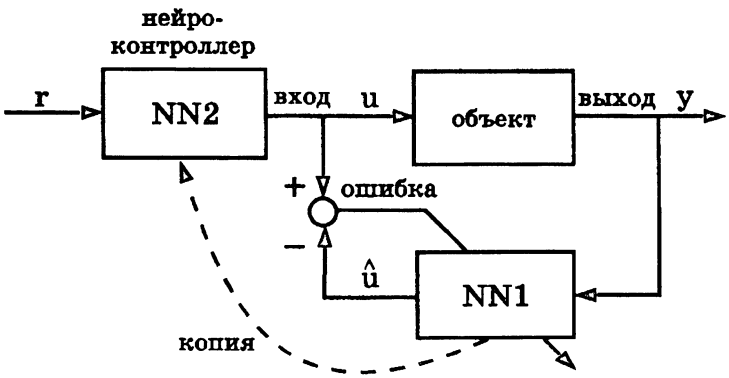


Рис. 4.3.9. Другая схема обратного отображения

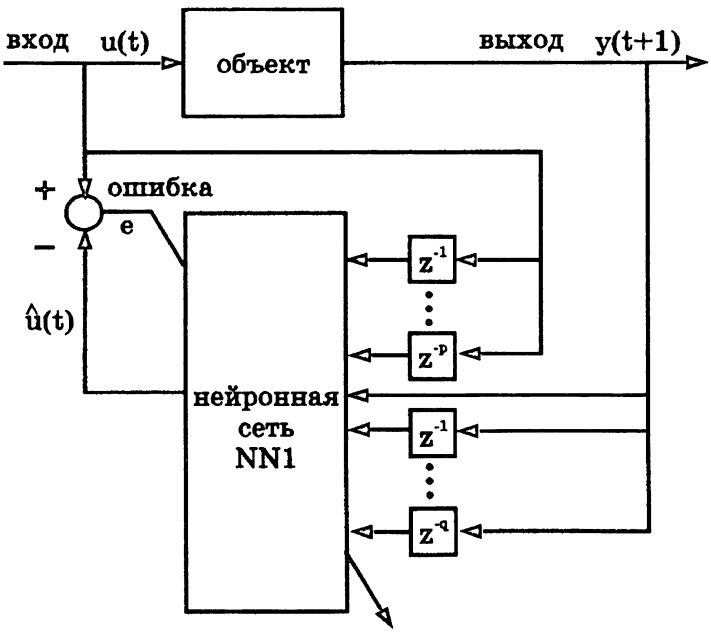


Рис. 4.3.10. Метод инверсно-прямого моделирования для схемы, приведенной на рис. 4.3.9.

4.4. Расширение последовательной схемы нейронного управления

Нейроконтроллеры на основе многослойных нейронных сетей широко применяются для решения нестандартных задач обучения и адаптивного управления. Однако их практическое применение серьезно ограничено из-за необходимого для них длительного срока обучения. В данном разделе рассматривается задача эффективного оперативного обучения многослойных нейроконтроллеров с целью сокращения времени обучения.

Сначала описываются схемы управления, использующие нейроэмуляторы и нейроконтроллеры на основе многослойных нейронных сетей. Нейроконтроллер обучается инверсной динамике объекта управления, используя выбранные случайным образом начальные веса. Для описания нескольких эффективных методов оперативного обучения, основанных на многократном выполнении операций корректировки в течение каждого периода дискретизации, в данном разделе снова рассматриваются основные схемы управления. Один из методов — инверсно-прямое управление — эффективен для небольшой настройки уже обученного контроллера. Другой метод — подход на основе ошибки предсказания выхода — обеспечивает непосредственную минимизацию ошибки управления. Он также существенно ускоряет сходимость процесса обучения контроллера. Результаты моделирования, полученные на несложном объекте управления (водяной ванне), подтверждают эффективность предложенных структур нейронного управления и методов обучения.

4.4.1. Некоторые сведения об оперативном обучении

Искусственные нейронные сети представляют собой математические модели, разработанные для практического использования принципов, на основе которых, как предполагается, действуют биологические системы. Можно ожидать, что ис-

пользование таких принципов позволит в некоторой степени имитировать возможности биологических систем по обработке информации [5]. Результатом современных разработок в области применения нейронных сетей в управлении динамическими процессами стало еще недостаточно развитое, но очень перспективное направление в нейронном управлении. Его можно рассматривать как нетрадиционное (коннекционистское) направление в теории адаптивного управления. Интерес к нейронному управлению, возникший у специалистов по системам управления, можно объяснить тремя основными причинами.

1. Имеются живые образцы интеллектуальных адаптивных контроллеров — биологические нервные системы.
2. Нейронные сети, по существу, представляют собой адаптивные системы, способные обучаться решению сложных задач.
3. Есть основания считать, что нейронные технологии управления позволят преодолеть многие сложности, возникающие при работе с нелинейными объектами или с объектами неизвестной структуры и неразрешимые с помощью обычных методов адаптивного управления.

Вообще говоря, нейронные сети могут обучаться решению задач управления в оперативном или автономном режиме, в зависимости от того, выполняет нейронная сеть полезную работу в процессе обучения или нет. Хотя автономное обучение обычно осуществляется достаточно быстро, обобщение результатов автономного обучения на конкретные условия управления представляет значительные сложности. В связи с этим практически для всех реальных задач управления необходимо оперативное обучение. В идеальном случае обучение должно быть только оперативным; сеть должна быстро обучаться, начиная с некоторого начального набора весов.

Для обучения инверсной динамике объекта управления по результатам наблюдения соотношений между его входными и выходными процессами были предложены нейроконтролле-

ры на основе нейронных сетей. Хотя такие нейроконтроллеры доказали свою эффективность в управлении сложными системами, из-за необходимого для них длительного времени обучения во многих практических ситуациях более эффективными оказываются контроллеры, построенные на основе обычных технологий. Медленная сходимость процесса обучения нейроконтроллеров приводит к плохим характеристикам управления и низкой надежности, особенно на начальных стадиях обучения. Чтобы нейронное управление стало действительной альтернативой традиционным методам, необходимы эффективные алгоритмы оперативного обучения нейроконтроллеров.

В данном разделе приводится некоторая общая структура нейронного управления, основанная на многослойных нейронных сетях. При этом требуются лишь незначительные сведения о количественных характеристиках объекта управления. Кратко рассматриваются основные схемы обучения нейроконтроллеров; отмечается, что хотя эти схемы обучения обеспечивают очень хорошие характеристики нейроконтроллеров, они требуют достаточно длительного времени обучения. В обычных адаптивных системах, где выполняется одна корректирующая операция за период дискретизации, повышение частоты дискретизации кажется естественным способом улучшения характеристик. Однако во многих реальных ситуациях частота дискретизации не может превышать некоторого предельного значения. С учетом этого ограничения в работе [52] предлагаются новые методы оперативного обучения, основанные на эффективном использовании данных о соотношении вход-выход объекта управления. В этих методах различаются понятия частоты дискретизации и частоты обучения. Предлагаемые методы направлены на сокращение времени обучения нейроконтроллеров. Для оценки эффективности предлагаемых нейроморфных структур управления и методов обучения используются результаты имитационного моделирования.

4.4.2. Нейроморфные структуры управления

4.4.2.1. Постановка задачи

Рассмотрим дискретный процесс с одним входом и с одним выходом:

$$y(t+1) = f_p[y(t), y(t-1), \ldots, y(t-\bar{q}+1), u(t), u(t-1), \ldots, u(t-\bar{p})], \quad (4.4.1)$$

где y — выход, u — вход, t — дискретное целочисленное время, $\bar{p}$ и $\bar{q}$ — неотрицательные целые числа, а $f_p(.)$ — некоторая функция. Во многих практических случаях входные координаты объекта управления ограничены по величине, т.е. существуют такие числа u_m и u_M, что для любого момента времени t выполняется условие

$$u_m \le u(t) \le u_M. \quad (4.4.2)$$

Итак, задача состоит в обучении управлению объектом, который описывается уравнением (4.4.1). Управление должно осуществляться таким образом, чтобы выходной сигнал соответствовал некоторому опорному сигналу r(t); при этом должна минимизироваться некоторая норма ошибки e(t) = =r(t) – y(t) во времени. Единственные априорные количественные данные об объекте управления — величины q и p, представляющие собой соответственно оценки величин $\bar{p}$ и $\bar{q}$ выражения (4.4.1) — предполагаются известными. Иными словами, предполагается известной приближенная оценка порядка дифференциального уравнения, описывающего объект управления.

4.4.2.2. Многослойные нейронные сети для управления

Для управления роботами и различными технологическими процессами применяются различные архитектуры нейронных сетей. Однако большая часть литературы по нейронному управлению посвящена именно многослойным нейронным сетям. Этот вид нейронных сетей представляет особый интерес для специалистов по управлению по следующим причинам.

1. Такие сети представляют собой структуры с прямыми связями, в которых информация передается в прямом направлении, от входов к выходам, через скрытые слои. Это свойство особенно удобно для работы с системами, которые можно представить в виде блоков с входами и выходами, но не в виде рекуррентных сетей. В модели Хопфилда [53], например, используются двунаправленные узлы, и фактически отсутствуют входы и выходы.
2. Сети всего с одним скрытым слоем, использующие произвольную сигмоидную функцию активации, могут выполнять любое нелинейное отображение между двумя пространствами конечной размерности с любой заданной точностью; для этого требуется лишь достаточное количество скрытых элементов (нейронов). Другими словами, многослойные нейронные сети обеспечивают самые разнообразные отображения с любой желаемой точностью. Обычно в системах управления многие блоки могут рассматриваться как отображающие элементы, поэтому для их эмуляции можно использовать многослойные нейронные сети с входами и выходами.
3. Основной алгоритм обучения многослойных нейронных сетей — алгоритм обратного распространения — относится к широкому классу градиентных методов наискорейшего спуска, широко применяемых в оптимальном управлении и поэтому известных специалистам по теории управления.

Указанные достоинства многослойных нейронных сетей свидетельствуют о том, что они могут рассматриваться как блоки, пригодные как для обучения, так и для отображения. На основе этой способности к отображению можно предложить две общие структуры нейронного управления: эмулятор объекта управления и контроллер.

4.4.2.3. Нейронный эмулятор объекта управления (НЭОУ)

При заданных оценках величин p и q, многослойная нейронная сеть с m=p+q+1 входами и одним выходом может использоваться для эмуляции зависимости $f_p(.)$ выражения (4.4.1). Обозначив отображение, выполняемое эмулятором объекта управления, как $\varphi_E(.)$, а его выход — как y_1, получим

$$y_1 = \varphi_E(x_E), \tag{4.4.3}$$

где x_E — m-мерный вектор.

Для случая, когда

$$x_E(t) = [y(t), y(t-1),..., y(t-q+1), u(t), u(t-1), ..., u(t-p)]^T,$$

целью обучения эмулятора является минимизация нормы ошибки эмуляции y(t+1)-y1. Схема НЭОУ показана на рис. 4.3.6.

4.4.2.4. Нейроконтроллер

Предположим, что объект управления, описываемый уравнением (4.4.1), является обратимым, т.е. существует функция g(.), такая, что

$$\begin{aligned} u(t) = g[&y(t+1), y(t),..., y(t-q+1),\\ &u(t-1), u(t-2),..., u(t-p)]. \end{aligned} \tag{4.4.4}$$

Рассмотрим многослойную нейронную сеть с m-мерным вектором входов x_c, одним выходом u_1 и соотношением между входом и выходом, кратко описываемым следующим образом:

$$u_1 = \varphi_c(x_c), \tag{4.4.5}$$

где $\varphi_c(.)$ — отображение «вход-выход» многослойной нейронной сети.

Если выход $\varphi_c(.)$ близок к выходу g(.) при соответствующих входах, то многослойная нейронная сеть может рассматриваться как контроллер в прямой цепи управления. В момент времени t значение входа объекта управления можно получить с помощью выражения (4.4.5), подставив в него:

$$x_c(t) = [r(t+1), y(t), \ldots, y(t-q+1), u(t-1), u(t-2), \ldots, u(t-p)]^T, \quad (4.4.6)$$

где вместо неизвестной величины y(t+1) использовано значение опорного сигнала r(t+1). Если ошибка выхода e(t) поддерживается на достаточно низком уровне после окончания процесса обучения, можно использовать выражение

$$x_c(t) = [r(t+1), r(t), \ldots, r(t-q+1), u(t-1), u(t-2), \ldots, u(t-p)]^T. \quad (4.4.7)$$

Отсюда видно, что нейроконтроллер не имеет обратных связей. Эта базовая конфигурация нейроконтроллера показана на рис. 4.3.1 под названием «последовательная схема нейронного управления».

4.4.3. Конфигурации обучения

Обучающий сигнал для контроллера, показанный на рис. 4.3.8, передает информацию, необходимую для обучения нейроконтроллера инверсной динамике объекта управления. Обучение должно обеспечивать минимизацию функции ошибки J, определенной для ошибки выхода объекта управления e(t) = r(t) — y(t). Для обучения на основе метода наискорейшего спуска необходимо вычислить производную функции ошибки J по выходной координате нейроконтроллера, т.е. $\delta = -\frac{\partial J}{\partial u_1}$. Знание величины δ достаточно для корректировки весов нейроконтроллера в соответствии с алгоритмом обратного распространения. Для объяснения этой схемы обучения многослойных нейронных сетей в следующих разделах будут еще раз кратко описаны основные конфигурации нейронного управления.

4.4.3.1. Инверсно-прямое управление

Инверсно-прямая конфигурация управления, или обобщенное обучение (см. рис. 4.3.10) — это схема автономного обуче-

ния. Однако, поскольку эта схема управления строится на основе способности нейроконтроллеров к обобщению, ее не следует применять как единственную схему обучения. С другой стороны, обучающие данные для управляемого обучения могут быть получены достаточно просто. В момент времени t+1 для вектора

$$x'_c(t) = [y(t+1),\ldots,\ y(t-q+1), u(t-1),\ldots, u(t-p)]^T$$

нейроконтроллер можно скорректировать с целью минимизации функции ошибки J^u, определяемой как функция разности $u(t)-u_1(t)$, где $u_1(t)=\varphi_c[x'_c(t)]$; таким образом, выражение $\delta_u = \partial J^u / \partial u_1$ можно легко вычислить, что делает возможным применение алгоритма обратного распространения. Например, если

$$J^u(t) = \frac{1}{2}(u(t)-u_1(t))^2, \qquad (4.4.8)$$

то выражение для производной δ^u принимает следующий вид:

$$\delta^u(t) = u(t) - u_1(t), \qquad (4.4.9)$$

где выражение $\delta^u(t)$ обозначает значение δ^u в момент времени t.

4.4.3.2. Оперативное специализированное обучение

Такая конфигурация показана на рис. 4.2.3. Обучение выполняется в оперативном режиме, а функция ошибки J определяется на основе ошибки выхода объекта управления, как и во многих стандартных задачах оптимизации управления. Сложность заключается в том, что для точного вычисления величины δ (чувствительности показателя J относительно выхода нейроконтроллера) требуется знать якобиан объекта управления. Например, простая функция ошибки и соответствующее выражение δ могут быть заданы следующим образом:

$$J(t) = \frac{1}{2}e(t)^2, \qquad (4.4.10)$$

$$\delta(t) = \xi_t e(t) \frac{dy(t)}{du(t)}, \qquad (4.4.11)$$

где $e(t) = r(t) - y(t)$, а двоичный коэффициент ξ_t вводится для учета ограничений на вход $u(t)$. Введя обозначение $\hat{e}(t) = e(t)\,dy(t)/du(t)$, приведем выражение для коэффициента ξ_t в момент времени t для системы, описанной формулами (4.4.1) и (4.4.2):

$$\xi_t = \begin{cases} 0, & \text{если} \begin{cases} \hat{e}(t) > 0 \quad \text{и} \quad u(t-1) = u_M \\ \text{или} \\ \hat{e}(t) < 0 \quad \text{и} \quad u(t-1) = u_m \end{cases} \\ 1, & \text{иначе.} \end{cases} \qquad (4.4.12)$$

Коэффициент ξ_t введен для того, чтобы исключить неправильное обучение нейроконтроллера по опорным сигналам, которые невозможно отслеживать. Включение коэффициента ξ_t эквивалентно введению ограничителя между выходом нейроконтроллера и входом объекта управления. Когда из-за физических ограничений, заданных выражением (4.4.2), возникает ошибка выхода, ограничитель насыщается, и в выражение (4.4.11) включается нулевая производная ($\xi_t = 0$), запрещающая обучение. В противоположном случае, когда работа в соответствии с опорным сигналом физически возможна, обучение выполняется обычным образом.

4.4.3.3. Схемы эмулятора и контроллера

Во многих практических ситуациях достаточно легко найти оценку производной, используемой в выражении (4.4.11). Иногда даже можно заменить ее значение +1 или -1, т.е. значением сигнум-функции от производной, так как алгебраического знака значения $\delta(t)$ достаточно для определения направления градиента J. В общем случае, однако, все не так просто. В схеме эмулятора и контроллера, приведенной на рис. 4.2.5, для вычисления чувствительности функции ошибки J относительно выхода контроллера используется НЭОУ. Так как НЭОУ представляет собой многослойную нейронную сеть,

желаемую чувствительность легко вычислить, используя обратное распространение. Кроме того, приведенная на рис. 4.2.5 конфигурация особенно полезна в случаях, когда инверсия объекта управления является плохо определенной, т.е. для функции $f_p(.)$ в уравнении (4.4.1) невозможно найти обратную функцию. Для коррекции контроллера, нейроконтроллер и НЭОУ рассматриваются соответственно как переменная и постоянная части многослойной нейронной сети с m входами и одним выходом. Сначала НЭОУ подвергается автономному обучению; используемый при этом набор данных должен быть достаточно представительным, чтобы допускать идентификацию объекта управления. Затем и нейроконтроллер, и НЭОУ подвергаются оперативному обучению. В известном смысле НЭОУ выполняет здесь идентификацию системы; поэтому для быстро изменяющихся систем, по соображениям устойчивости, имеет смысл корректировать НЭОУ чаще, чем нейроконтроллер.

4.4.4. Эффективное оперативное управление

Первоначально многослойные нейронные сети и алгоритм обратного распространения были разработаны для задач распознавания образов, где обучающие образцы являются статическими, процедура обучения и функция ошибки — однозначные, и обучение в реальном времени не требуется. В управлении обучающие образцы для нейронной сети изменяются со временем, возможно несколько алгоритмов обучения и функций ошибки, а обучение в реальном времени необходимо.

Медленная сходимость — основной недостаток многослойных нейронных сетей, серьезно ограничивающий практическое применение нейронного управления. Предложено несколько подходов для ускорения сходимости в нейронном управлении. Перечислим некоторые из них.

1. Разработка эффективных алгоритмов обратного распространения.

2. Встраивание знаний о структуре объекта управления в структуру многослойных нейронных сетей.
3. Применение гибридных сетей, в которых искусственные нейронные сети связываются со структурами управления, полученными на основе других (отличных от нейронных) технологий.
4. Предварительное обучение и эффективные процедуры инициализации.

Ниже рассматриваются новые алгоритмы оперативного обучения, направленные на сокращение времени обучения нейроконтроллеров. Эти алгоритмы основаны на разделении понятий частоты дискретизации и частоты выполнения обучающих итераций (частоты обучения). В системах управления с дискретным временем период дискретизации T обычно выбирается по следующему эмпирическому правилу: величина $2\pi/T$ должна значительно превышать максимальную частоту, имеющуюся в системе с непрерывным временем. Обычно повышение частоты дискретизации улучшает характеристики системы, однако это улучшение быстро прекращается (график изменения характеристик достигает плато). В обычных адаптивных системах управления адаптивные элементы, как правило, корректируются один раз за каждый период дискретизации; таким образом, частоту дискретизации и частоту корректировки (или обучения, или адаптации) можно не различать. Если пренебречь ограничениями на время обработки, то может показаться, что фактическое время обучения можно сократить, повысив частоту дискретизации.

Однако во многих практических случаях превышение некоторого предела частоты дискретизации недопустимо или нежелательно. Например, в обычных промышленных химических установках, как правило, интерес представляют процессы, связанные с большими величинами временных констант. При этом не имеет смысла использовать высокие частоты дискретизации: это может привести к избыточности информации. Использование очень высоких частот дискретизации может привести к

полной перестройке системы управления и усложнить ее; может потребоваться учитывать частные процессы и переходные явления, которые при меньших частотах дискретизации можно было бы игнорировать. Другой пример систем, в которых нельзя использовать произвольно высокую частоту дискретизации — распределенные системы управления, в которых интервалы передачи информации на устройство управления и приема информации с него не зависят от самого устройства управления.

Хотя период дискретизации T задает базовый темп работы системы управления, в системах с итерационным обучением частота обучающих операций может рассматриваться как еще одна основа для отсчета времени. На практике период дискретизации T обычно значительно превышает время T_L, затрачиваемое на одну обучающую итерацию, т.е. на корректировку всех весов сети; по мере появления реализаций многослойных сетей с более высоким быстродействием (за счет улучшения их программной или аппаратной реализации) отношение T/T_L повышается. Таким образом, если имеется соответствующая информация о входе и выходе объекта управления, и интерес представляет только время, то за период дискретизации может выполняться несколько обучающих итераций. Обычный (и самый простой) подход, состоящий в выполнении одной корректировки за период дискретизации, связан с непроизводительными затратами времени обработки. Проблема состоит в том, каким образом выбрать и использовать обучающие данные, как использовать имеющееся время для рационального обучения нейроморфных структур, т.е. для обучения, которое должно повысить эффективность управления. Ниже рассматриваются новые методы обучения, в которых за каждый период дискретизации выполняется несколько обучающих итераций; эти методы предложены в [52].

4.4.4.1. Обучение эмулятора

Предположим, что в момент времени t+1 в памяти хранится текущее значение выхода y(t+1), q+n-1 значений того же про-

цесса в предшествующие моменты времени и p+n значений входного процесса u в предшествующие моменты времени. В этом случае в качестве образцов для обучения НЭОУ в момент времени t+1 могут использоваться n пар (x_E(t-i), y(t+1--i)), i=0,1,...,n-1. Для случая y_1(t+1-i)=φ_E(x_E(t-i)) требуется минимизировать следующую функцию ошибки:

$$J_n(t) = \frac{1}{2}\left(\sum_{i=0}^{n-1} \lambda_i (y(t-i) - y_1(t-i))\right)^2, \qquad (4.4.13)$$

где $\{\lambda_i\}$ $(1 \geq \lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_n\text{-}1 \geq 0)$ — невозрастающая положительная последовательность, предназначенная для определенного «забывания» образцов, полученных ранее.

Пример 4.4.1. Пусть только что получена величина y(10) (таким образом, y(11) еще неизвестна), p=2, q=3, n=3. Предположим также, что величины y(9),y(8),...,y(5) и u(9), u(8),..., u(5) имеются в памяти. В этом случае входные векторы НЭОУ можно представить в виде:

$$x_E(7) = [y(7), y(6), y(5), u(7), u(6), u(5)]^T,$$

$$x_E(8) = [y(8), y(7), y(6), u(8), u(7), u(6)]^T,$$

$$x_E(9) = [y(9), y(8), y(7), u(9), u(8), u(7)]^T.$$

Эти входные векторы и величины y(8),y(9) и y(10) образуют обучающие образцы для обучения в момент времени t+1=10. Данная процедура показана на рис. 4.4.1(а), где через $PE^{t,i}$ обозначено состояние НЭОУ во время r-го интервала дискретизации, после i-й обучающей итерации, i=0,1,...,n-1. Аналогично $\varphi E^{t,i}$(.) обозначает отображение «вход-выход», выполняемое НЭОУ в состоянии $PE^{t,i}$. Очевидно, что

$$\varphi_E{}^{t,i}(.) = \varphi_E{}^{t+1,0}(.).$$

4.4.4.2. Обучение контроллера: подход на основе ошибки инверсно-прямого управления

Рассмотрим инверсно-прямую конфигурацию управления (рис. 4.3.10). Предположим, что в момент времени t+1 в памя-

ти хранится текущее значение выхода y(t+1), q+n-1 предыдущих значений выходного и p+n предыдущих значений входного процессов. В этом случае в качестве образцов для обучения нейроконтроллера в момент времени t+1 могут использоваться n пар

$$(x'_c(t-i),\ u(t-i)),\ i=0,\ldots,n-1,$$

где

$$x'_c(t)=[y(t+1),\ldots,y(t-q+1),u(t-1),\ldots,u(t-q)]^T.$$

При $u_1(t\text{-}i)=\varphi_c[x'_c(t\text{-}i)]$, и функции ошибки

$$J^u(t)=\frac{1}{2}\sum_{i=0}^{n-1}\lambda_i(u(t-i)-u_1(t-i))^2 \qquad (4.4.14)$$

соответствующий δ — член для i-го образца записывается следующим образом:

$$\delta_{t,i}=\lambda_i(u(t-i)-u_1(t-i)). \quad (4.4.15)$$

Следует отметить, что функция ошибки (4.4.14) не включает в себя непосредственно координату ошибки объекта управления. В связи с этим обучение объекта управления непосредственно не повышает эффективность управления, однако такой способ обучения позволяет достичь хорошего обобщения в пространстве параметров управления. На практике обучение нейроконтроллеров, основанное только на инверсно-прямом подходе, дает плохие результаты: выход нейроконтроллера устанавливается на некоторой постоянной величине, в результате чего ошибка обучения оказывается нулевой, однако характеристики управления оказываются явно плохими. Этот недостаток присущ всем методам обучения, основанным на минимизации ошибки инверсного управления. Его можно устранить, используя сочетание таких методов управления с другими, непосредственно минимизирующими ошибку системы регулирования. Это показано на следующем примере.

Пример 4.4.2. Пусть p=3, q=2 и только что получена величина y(9). Предположим также, что в памяти имеются величины y(8),y(7),...,y(5) и u(8),u(7),...,u(3). Обозначим через $\varphi_c^{9,0}(.)$

текущее отображение, выполняемое нейроконтроллером (равенство t+1=9 означает, что обучения еще нет). Входной сигнал для объекта управления u(9) можно вычислить из соотношения $u(9)=\varphi_c^{9,0}(x_c(9))$, где $x_c(9)=[r(10), y(9), y(8), u(8), u(7), u(6),]^T$. Для обучения в соответствии с подходом на основе ошибки инверсно-прямого управления можно использовать следующие векторы:

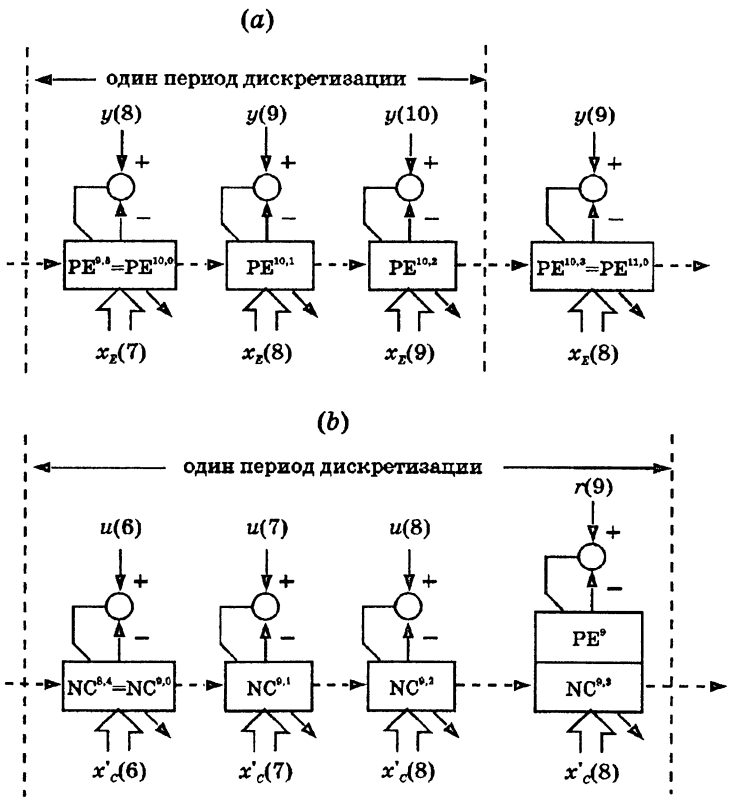


Рис. 4.4.1. Простые схемы множественного обучения: (*a*) НЭОУ (пример 4.4.1); (*b*) нейроконтроллер, объединяющий инверсно-прямое управление и схемы эмулятора и контроллера (пример 4.4.2) [52]

$$x'_c(6) = [y(7), y(6), y(5), u(5), u(4), u(3)]^T,$$

$$x'_c(7) = [y(8), y(7), y(6), u(6), u(5), u(4)]^T,$$

$$x'_c(8) = [y(9), y(8), y(7), u(7), u(6), u(5)]^T.$$

Эти векторы и входные величины u(6), u(7) и u(8) образуют три обучающих образца (входной вектор и желаемый выход) для обучения нейроконтроллера в момент времени t+1=9. Однако такой метод обучения непосредственно не минимизирует ошибку управления; поэтому на практике необходимо сочетание этого подхода с одним из методов, рассмотренных в разделах 4.4.3.2 и 4.4.3.3. На рис. 4.4.1,b показано сочетание множественного обучения (на основе инверсно-прямого управления) и простого обучения на основе обучающей конфигурации; в результате выполняется 4 обучающих итерации за один период дискретизации. Вектор xc(8) задается составляющими [r(9), y(8), y(7), u(7), u(6), u(5),]^T; через $NC^{t,i}$ обозначено состояние нейроконтроллера во время t-го интервала дискретизации, после i-й обучающей итерации (соответствующее отображение задается выражением $\varphi_c^{t,i}(.)$). В результате достигается точное обучение НЭОУ за счет усложнения процесса обучения.

4.4.4.3. Обучение контроллера: подход на основе прогнозируемой ошибки выхода

Более сложный подход к множественному обучению нейроконтроллера можно получить на основе конфигурации из эмулятора и контроллера, рассмотренной в разделе 4.4.3.3. Предположим, что в момент времени t+1, в дополнение к n+p значениям переменной y (включая y(t+1)) и n+q предыдущим значениям переменной u имеется также n значений опорного сигнала r(t+1-i),i=0,1,...,n-1. Из выражений (4.4.6) и (4.4.7) следует, что наличие таких данных эквивалентно наличию n входных векторов x_c(t-i) в памяти. В соответствии с выражением (4.4.5), в момент времени t-i входной управляющий сигнал u(t-i) определяется следующим образом:

$$u(t-i) = \varphi_c^{t-i,0}[x_c(t-i)]. \qquad (4.4.16)$$

Однако с момента времени, когда вектор $x_c(t\text{-}i)$ был сохранен, до текущего момента времени t+1, было выполнено несколько корректировок нейроконтроллера. Поэтому в момент времени t+1 сохраненный входной вектор $x_c(t\text{-}i)$ даст следующий виртуальный входной сигнал:

$$u^*(t-i) = \varphi_c^{t+1,0}[x_c(t-i)]. \qquad (4.4.17)$$

Это означает только, что последняя величина $x_c(t)$ с соответствующими величинами y(t+1) и r(t+1) может быть непосредственно использована для обучения на основе одной из конфигураций адаптивного управления. Однако это не относится в полной мере к предыдущим величинам $x_c(t\text{-}i)$ для i=1,...,n-1. Для таких случаев можно спрогнозировать соответствующий выходной сигнал объекта управления с помощью эмулятора (см. рис. 4.4.2,а) в соответствии с выражением

$$y^*(t+1-i) = \varphi_E^{t+1}[x_E^*(t-i)], \qquad (4.4.18)$$

где второй верхний индекс в обозначении $\varphi_E^{t+1}(.)$ опущен для простоты, а вектор $x_E^*(t-i)$ задается следующим образом:

$$x_E^*(t-i) = [y(t-i),\ldots,y(t-p+1-i),u^*(t-i),\ldots,u^*(t-q-i)]^T. \qquad (4.4.19)$$

Для лучшего понимания процесса обучения нейроконтроллера представим себе нейроконтроллер и НЭОУ как одну многослойную сеть. В момент времени t+1 каждому входному вектору $x_c(t\text{-}i)$, i=1,...,n-1 пусть соответствует прогнозируемая ошибка r(t+1-1)-y*(t+1-i), и обучение направлено на минимизацию некоторой нормы предсказанной ошибки выхода. Функция ошибки может иметь следующий вид:

$$J_n(t) = \frac{1}{2}\left(\sum_{i=0}^{n-1}\lambda_i(y(t-i)-y^*(t-i))\right)^2. \qquad (4.4.20)$$

В обучающей конфигурации, показанной на рис. 4.4.2,а,

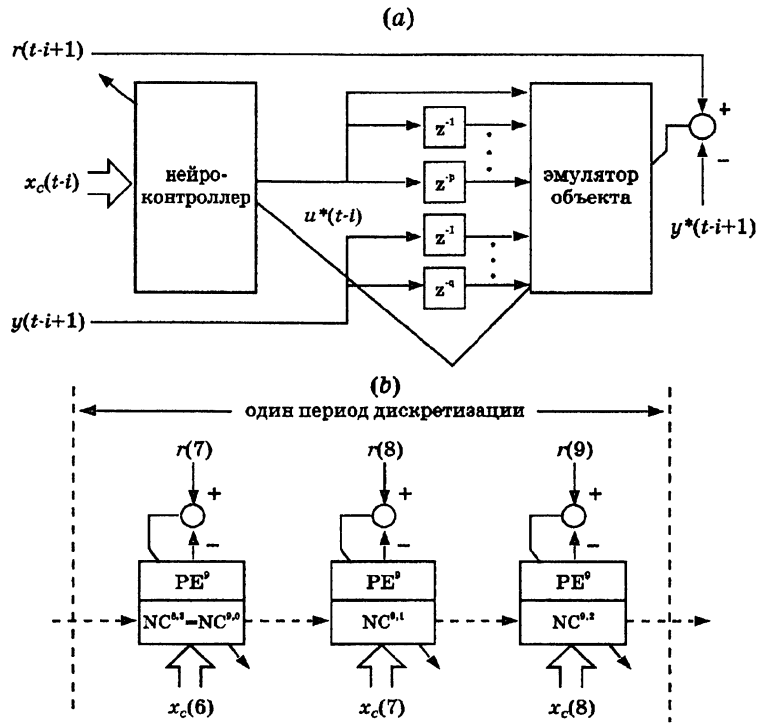


Рис. 4.4.2. Подход на основе прогнозируемой ошибки выхода. (*a*) Схема обучения, использующая расхождение между опорным сигналом и выходом НЭОУ. (*b*) Множественное обучение нейроконтроллера на основе прогнозируемой ошибки выхода (пример 4.4.3) [52]

нейроконтроллер обучается на основе ошибки, представляющей собой расхождение между опорным сигналом и выходом НЭОУ, а не расхождение между опорным сигналом и выходом объекта управления, как в схеме прямого адаптивного управления, приведенной на рис. 4.4.2,b. Предыдущие значения величины выхода объекта управления y(t) необходимы для входных векторов нейроконтроллера и НЭОУ.

Пример 4.4.3. Пусть p=3, q=2 и только что получена величина y(9). Предположим, что в памяти имеются значения y(8),y(7),...,y(5) и u(8),u(7),...,u(3), а также r(9), r(8) и r(7). Входные векторы нейроконтроллера, используемые для вычисления трех последних значений входного управляющего сигнала, можно записать в следующей форме:

$$x_c(6) = [y(7), y(6), y(5), u(5), u(4), u(3)]^T,$$

$$x_c(7) = [y(8), y(7), y(6), u(6), u(5), u(4)]^T,$$

$$x_c(8) = [y(9), y(8), y(7), u(7), u(6), u(5)]^T.$$

Возможная процедура обучения для момента времени t+1=9 показана на рис. 4.4.2,b; за один период дискретизации выполняется три обучающих итерации. Прогнозируемые величины выхода объекта управления вычисляются по схеме, показанной на рис. 4.4.2,a, допускающей использование величин $x_c(6)$ и $x_c(7)$ в момент времени t+1=9; при этом последний вектор $x_c(8)$ используется обычным образом.

4.4.5. Алгоритмы обучения

В данном разделе методы оперативного обучения, предложенные выше, приводятся в виде алгоритмов в псевдокоде (это обеспечивает определенную однозначность реализации таких методов). Предполагается, что читатель знаком с алгоритмом обратного распространения; он указывается в виде вызова процедуры

BP(φ, x, n, o),

где BP — имя процедуры, φ — корректируемая сеть, x — входной вектор, n — желаемый выход, o — фактический выход.

4.4.5.1. Множественное обучение эмулятора

Пусть в момент времени t в памяти хранятся векторы $x_{E,i}$=[y(t-1-i), ..., y(t-q-i), u(t-1-i), ..., u(t-1-p)]^T для i=0, ..., n-1. Очевидно, что новейшие данные соответствуют значению i=0, более старые данные соответствуют большим значениям i. На-

чиная с момента времени t, алгоритм множественного обучения эмулятора можно представить в следующей форме.

Шаг 1. READ y(t).

Шаг 2. {Обучение эмулятора}

$i \leftarrow n - 1$

REPEAT

$y_{1,i} \leftarrow \varphi_E(x_{E,i})$

BP $(\varphi_E, x_{E,i}, \lambda_i y(t\text{-}i), \lambda_i, y_{1,i})$

$i \leftarrow i\text{-}1$

UNTIL $(i<0)$.

Шаг 3. {Генерация управляющего входного сигнала}

$x_c \leftarrow [r(t=1), y(t), \ldots, y(t+1-q), u(t-1), \ldots, u(t-p)]^T$

или

$[r(t+1), r(t), \ldots, r(t+1-q), u(t-1), \ldots, u(t-p)]^T$,

$u(t) \leftarrow \varphi_C(x_c)$.

Шаг 4. Применить u(t) к объекту управления и ждать время T.

Шаг 5. {Сдвиг данных}

$i \leftarrow n\text{-}1$

REPEAT

$x_{E,i} \leftarrow x_{E,i-1}$

$i \leftarrow i\text{-}1$

UNTIL$(i=0)$.

Шаг 6. {Новейший вектор данных}

$x_{E,0} \leftarrow [y(t), y(t\text{-}1), \ldots, y(t+1\text{-}q), u(t), \ldots, u(t\text{-}p)]^T$

Шаг 7.

$t \leftarrow t+1$

Шаг 8. Переход к шагу 1.

4.4.5.2. Множественное обучение контроллера: подход на основе ошибки инверсно-прямого управления

Пусть в момент времени t в памяти хранятся векторы $x'_{c,i} = [y(t-i), \ldots, y(t-q-i), u(t-2-1), \ldots, u(t-1-p-i)]^T$ для i=0, ..., n-1. Начиная с момента времени t, алгоритм множественного обучения контроллера можно представить в следующей форме.

Шаг 1. Прочитать y(t).

Шаг 2. {Новейший вектор данных}

$x'_{c,0} \leftarrow [y(t), \ldots, y(t-q), u(t-2), \ldots, u(t-p-1)]^T$

Шаг 3. {Обучение контроллера}

$i \leftarrow n-1$
REPEAT
$u_{1,i} \leftarrow \varphi_C(x'_{c,i})$
BP$(\varphi_C, x'_{c,i}, \lambda_{iu}(t-1-i), \lambda_i u_{1,i})$
$i \leftarrow i-1$
UNTIL(i<0).

Шаг 4. {Генерация управляющего входного сигнала}

$x_c = [r(t+1), y(t), \ldots, y(t+1-q), u(t-1), \ldots, u(t-p)]^T$

или

$x_c = [r(t+1), r(t), \ldots, r(t+1-q), u(t-1), \ldots, u(t-p)]^T$
$u(t) \leftarrow \varphi_C(x_c)$.

Шаг 5. Применить u(t) к объекту управления и ждать время T.

Шаг 6. {Сдвиг данных}

$i \leftarrow n-1$
REPEAT
$x'_{c,i} \leftarrow x'_{c,i-1}$
$i \leftarrow i-1$
UNTIL(i=0).

Шаг 7. $t \leftarrow t+1$

Шаг 8. Переход к шагу 1.

4.4.5.3. Множественное обучение контроллера: подход на основе прогнозируемой ошибки выхода

Пусть в момент времени t в памяти хранится (n+q+1)- вектор $x_{c,i} = [r(t-i), y(t-1-i), \ldots, y(t-q-i), u(t-2-i), \ldots, u(t-p-1-i)]^T$ или $x_{c,i} = [r(t-i), r(t-1-i), \ldots, r(t-q-i), u(t-2-i), \ldots, u(t-p-1-i)]^T$. Обозначим через $(\varphi_C+\varphi_E)$ нейронные сети, образованные связыванием нейроконтроллера и НЭОУ. Начиная с момента времени t, алгоритм множественного обучения контроллера можно представить в следующей форме.

Шаг 1. Прочитать y(t).

Шаг 2. {Обучение контроллера по прогнозируемым ошибкам}

$i \leftarrow n-1$
REPEAT
$j \leftarrow 0$
REPEAT
$u^*_j \leftarrow \varphi_C(x_{c,i+j})$
$j \neg j+1$
UNTIL(j>p)
{виртуальный входной вектор для эмулятора}
$x_E \leftarrow [y(t-1-i), \ldots, y(t-q-i), u^*_0, u^*_1, \ldots, u^*_p]^T$
{виртуальный выход}
$y^* \leftarrow \varphi_E(x_E)$
BP$(\varphi_C, \varphi_E, x_{c,i}, \lambda_i r(t-i), l_i y^*)$
$i \leftarrow i-1$
UNTIL(i=0).

Шаг 3. {Обычное обучение с использованием новейших данных}

BP$(\varphi_C, \varphi_E, x_{c,i}, \lambda_i r(t-i), \lambda_i y(t))$

Шаг 4. {Сдвиг данных}

```
i ← n+q+1
REPEAT
 xc,i ¬ xc,i-1
 i ← i-1
UNTIL (i=0).
```

Шаг 5. {Генерация управляющего входного сигнала}

$x_{c,0} = [r(t+1), y(t), \ldots, y(t+1-q), u(t-1), \ldots, u(t-p)]^T$

или

$[r(t+1), r(t), \ldots, r(t+1-q), u(t-1), \ldots, u(t-p)]^T$

$u(t) \leftarrow \varphi_C(x_{c,0})$

Шаг 6. Применить u(t) к объекту управления и ждать время T.

Шаг 7. $t \leftarrow t+1$

Шаг 8. Переход к шагу 1.

4.4.6. Оценка алгоритмов обучения на основе моделирования

Чтобы оценить характеристики предлагаемой системы управления и методов обучения, используются результаты моделирования для несложного объекта управления. Компьютерное моделирование особенно необходимо для оперативного обучения нейронных систем управления, так как натурные эксперименты для такого обучения во многих случаях занимают очень много времени. Рассмотрим систему управления температурой объекта, описываемую дифференциальным уравнением

$$\frac{dy(\tau)}{d\tau} = \frac{f(\tau)}{C} + \frac{Y_0 - y(\tau)}{RC}, \qquad (4.4.21)$$

где τ — время; $y(\tau)$ — выходная температура; $f(\tau)$ — тепло, поступающее в систему; Y_0 — комнатная температура (для простоты будем считать ее постоянной); C — теплоемкость системы; R — тепловое сопротивление между границами системы и

окружающей средой. Считая величины R и C постоянными, по реакции на единичное ступенчатое воздействие выведем импульсную переходную функцию для системы, заданной уравнением (4.4.21). Это приводит к системе с дискретным временем, задаваемой уравнением

$$y(\tau+1) = a(T)y(\tau) + b(T)u(\tau), \qquad (4.4.22)$$

где τ — дискретное целочисленное время, T — интервал квантования, $u(\tau)$ и $y(\tau)$ — вход и выход системы соответственно. Обозначив через α и β некоторые постоянные величины, зависящие от констант R и C, остальные параметры можно выразить следующим образом:

$$a(T) = e^{-\alpha T} \text{ и } b(T) = \frac{\beta}{\alpha}(1 - e^{-\alpha T}). \qquad (4.4.23)$$

Приведенные ниже результаты моделирования были получены для системы, несколько отличающейся от заданной уравнениями (4.4.21) — (4.4.23). В системе была задана нелинейная характеристика с насыщением, чтобы исключить превышение выходной температуры некоторого граничного значения. Моделируемый объект управления описывался следующим уравнением:

$$y(\tau+1) = a(T)y(\tau) + \frac{b(T)}{1 + e^{0.5y(t)-\gamma}}u(\tau) + [1 - a(T)]Y_0. \qquad (4.4.24)$$

При моделировании использовались следующие значения параметров: $\alpha=1.00151\times10^{-4}$, $\beta=8.67973\times10^{-3}$, $\gamma=40.0$ и $Y_0=25.0^{\circ}C$. Данные значения параметров были получены на реальном объекте (водяная ванна, рассматриваемая в гл.5). Входной сигнал объекта u(t) мог принимать значения только в диапазоне от 0 до 5; кроме того, предполагалось, что на период дискретизации наложено ограничение

$$T \geq 10 \text{ с.} \qquad (4.4.25)$$

При выбранных значениях параметров моделируемая система эквивалентна системе управления температурой водяной ванны с одним входом и одним выходом. Такая система имеет линейные характеристики при температуре примерно

до 70°С, а при температуре около 80°С она переходит в насыщение. Если сравнить (4.4.24) и (4.4.1), то можно заметить, что p=0, а q=1.

В качестве ЭОУ и нейроконтроллера использовались трехслойные нейронные сети, содержащие от 10 до 20 скрытых нейронов с сигмоидной функцией активации. Сходимость была достигнута для нескольких пар (p,q) при значениях параметра p от 0 до 2, а параметра q — от 1 до 3. Приведенные ниже графики соответствуют значениям p=0 и q=1 (точное согласование). Коррекция сетей производилась по следующему общему правилу:

$$\Delta w(t) = w(t+1) - w(t),$$

$$\Delta w(t) = -\eta \frac{\partial J}{\partial w(t)} + \alpha \Delta w(t-1), \qquad (4.4.26)$$

где t — номер итерации обучения, w — вес связи, J — подлежащая минимизации функция ошибки, $\eta > 0$ — скорость обучения, $\alpha \geq 0$ -мгновенная составляющая.

До начала моделирования ЭОУ подвергался приближенному оперативному обучению, а первоначальные значения весов нейроконтроллера назначались случайным образом. В качестве начального состояния использовалось значение $y(0)=Y_0$. Целью регулирования было поддержание температуры, равной 35 °С для временного интервала $0 \leq tT \leq 30$ мин., 55 °С — для интервала 30 мин.$< tT \leq 60$ мин. и 75 °С для интервала 60 мин. $< tT \leq 90$ мин. Каждый цикл имитации (для значений tT от 0 до 90 мин) будем называть пробой. После каждой пробы веса фиксировались, и предпринималась новая проба. На рис. 4.4.3 показан опорный сигнал, вход и выход объекта после достижения хорошей сходимости (для T= 30 с).

Характеристика нейроконтроллера, выполняющего одну обучающую итерацию за период дискретизации, сравнивается с характеристикой для множественного обучения на рис. 4.4.4—4.4.6. На графиках показана зависимость среднеквадратической ошибки в одном испытании от количества

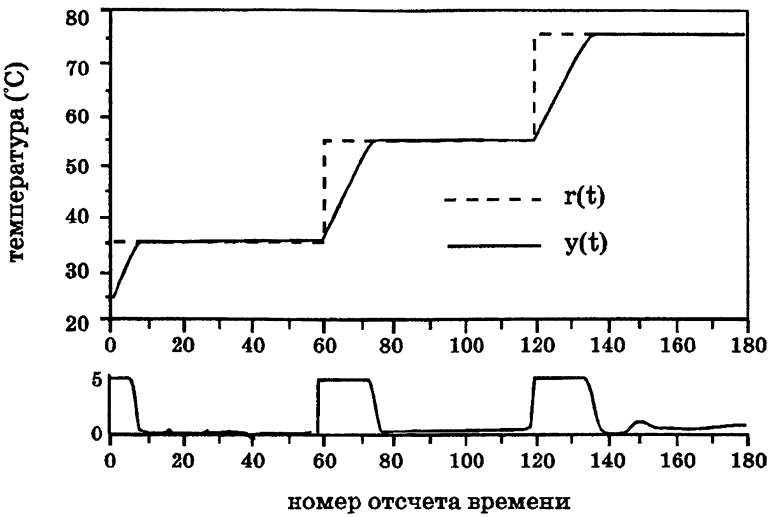


Рис. 4.4.3. Результаты моделирования системы управления температурой водяной ванны после обучения [52]

испытаний. На рис. 4.4.4 показана инициализация контроллера, обеспечивающая небольшую ошибку в начале. Верхние графики соответствуют случаям, когда обучение выполняется один раз за период (с использованием косвенного адаптивного управления). Нижние же графики соответствуют случаям, когда за период выполняется 10 дополнительных корректировок нейроконтроллера (с использованием прямого адаптивного управления по ошибке). Во всех случаях период дискретизации составлял 30 с.

Предложенный метод, как и ожидалось, обеспечивает улучшение характеристик, так как контроллер хорошо обучен с самого начала, и поэтому обобщение является надежным. Из этого можно сделать вывод, что данный метод обучения может использоваться для настройки нейроконтроллера вблизи точки, соответствующей его хорошей работе, т.е. для точной настройки. Это неверно для случая, когда контроллер находится в необученном состоянии. Практический опыт по-

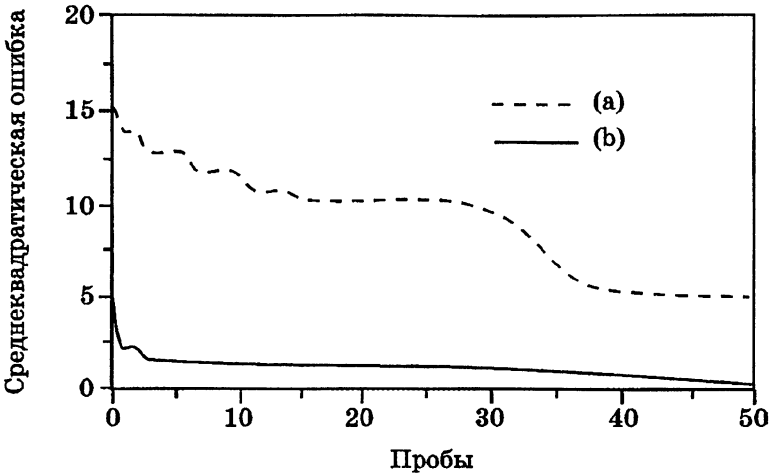


Рис. 4.4.4. Нейроконтроллер со случайной инициализацией, для T=30 с: *a*) только обычное оперативное обучение; *b*) обычное обучение в сочетании с 5-ю обучающими итерациями/период на основе прогнозируемой ошибки выхода; *c*) обычное обучение в сочетании с 10-ю обучающими итерациями/период [52]

казывает: если веса нейроконтроллера таковы, что выходная ошибка объекта управления велика, то обучение, основанное только на инверсном управлении, часто приводит к ситуациям, когда выход нейроконтроллера фиксируется на некоторой величине, что приводит к плохим характеристикам управления.

На рис. 4.4.5, *a* и 4.4.6, *a* показана характеристика случайно инициализируемого нейроконтроллера, для которого выполняется одна обучающая итерация за период на основе косвенного адаптивного управления (период T=30 с). Так как каждое модельное испытание продолжается 90 минут реального времени, то 50 испытаний требуют 75 часов, или более трех дней. По этой причине большинство результатов, приводимых в данной книге, получены на моделях, а не в реальных экспериментах.

Нижние графики на рис. 4.4.5 соответствуют периоду диск-

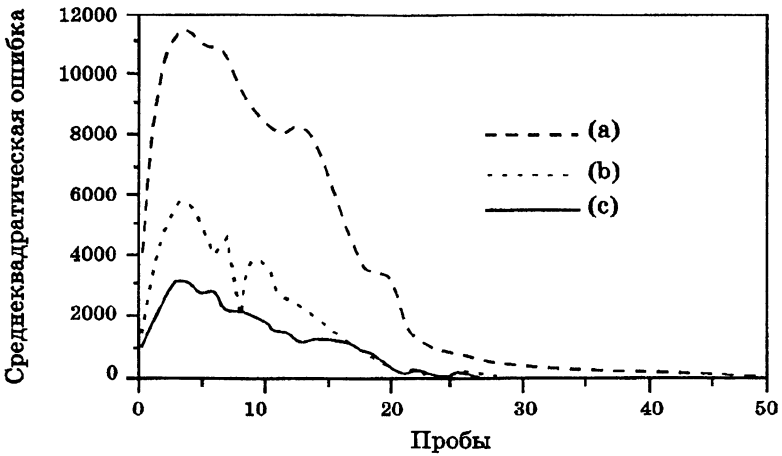


Рис. 4.4.5. Нейроконтроллер со случайной инициализацией, выполняющий одну обучающую итерацию за период дискретизации: *a*) T=30 с; *b*) T=15 с; *c*) T=10 с [52]

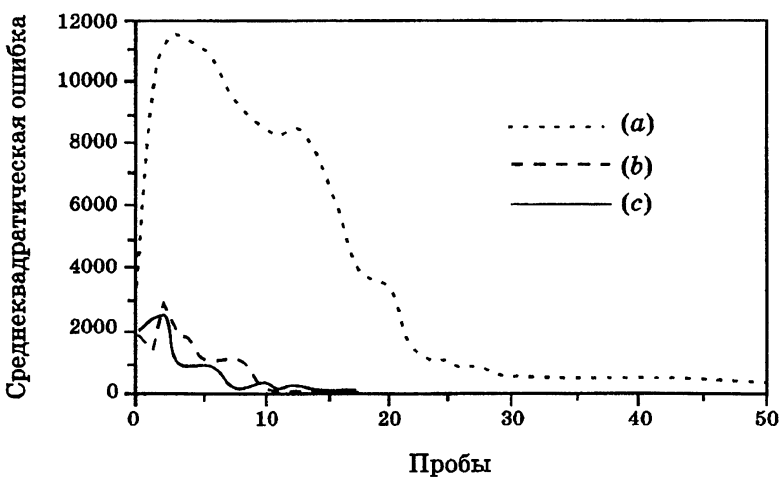


Рис. 4.4.6. Нейроконтроллер, случайным образом инициализируемый с периодом T=30 с. (*a*) Только обычное оперативное обучение. (*b*) Обычное обучение плюс 5 обучающих итераций/период (на основе прогнозируемой ошибки выхода). (*c*) Обычное обучение плюс 10 обучающих итераций/период [52]

ретизации T=15 с (рис. 4.4.5, *b*) и 10 с (рис. 4.4.5, *c*). При этом наблюдается ожидаемое улучшение характеристики. Для остальных графиков на рис. 4.4.6 период дискретизации постоянен и составляет 30 с, однако каждый период включает по 5 или 10 обучающих итераций (рис. 4.4.4, *b* или 4.4.6, *c* соответственно). Включение всего нескольких обучающих итераций за период дискретизации резко ускоряет сходимость и существенно снижает общую ошибку.

Затраты времени на корректировку нейроморфной структуры примерно пропорциональны количеству обучающих итераций; эти затраты зависят в основном от структуры используемых сетей и количества весов в них. В приведенных ниже результатах моделирования 11 обучающих итераций для трехслойного нейроконтроллера с 80 весами заняли примерно $11 \times T_L = 390$ мс (на персональном компьютере), что значительно меньше, чем период дискретизации во многих реальных задачах управления.

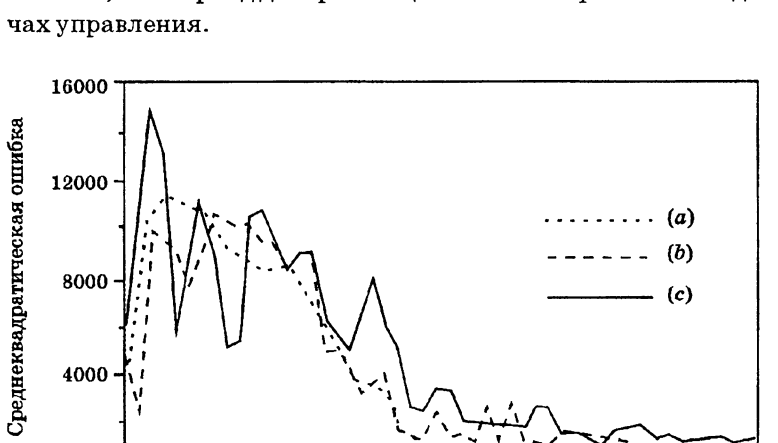


Рис. 4.4.7. Влияние рассогласования между оценками и оптимальным порядком модели объекта управления. (*a*) Точное соответствие для $\overline{p} = p = 0$ и $\overline{q} = q = 1$. (*b*) Рассогласование при сходимости для p=0, q=3. (*c*) Рассогласование при отсутствии сходимости для p=q=2 [52]

Как уже отмечалось, показанные на рис. 4.4.3—4.4.6 результаты получены для случая, когда $\bar{p} = p = 0$ и $\bar{q} = q = 1$. Естественно возникает вопрос о том, что происходит, когда нет точных оценок порядка объекта управления. В проведенных модельных экспериментах обучение выполнялось для нескольких различных сочетаний значений параметров p и q, однако при обобщении полученных результатов требуется соблюдать осторожность. Небольшие изменения в начальных условиях, опорных величинах, структурах нейронных сетей, функциях активации нейронов и т.п. часто приводят к резким изменениям характеристик. Испытания проводились в различных условиях и для разных комбинаций значений параметров p и q; некоторые результаты показаны на рис. 4.4.7. На рис. 4.4.7, *a* показан случай точного соответствия, а на рис. 4.4.7, *b* и 4.4.7, *c* — случаи рассогласования при p=0, q=3 и p=2, q=2 соответственно. Хотя в некоторых случаях (как, например, на рис. 4.4.7, *c*), сходимость не была достигнута, для нескольких пар (p; q) вблизи действительных значений этих величин $\bar{p}$ и $\bar{q}$ были получены хорошие результаты.

4.5. Параллельная схема управления

Параллельная архитектура нейронного управления показана на рис. 4.5.1, где в качестве контроллера используется обычный ПИД-контроллер. Примером такой схемы нейронного управления является метод обучения по ошибке обратной связи, предложенный Кавато (Kawato) и др. в работах [40], [41] и описанный в разделе 4.2.1. Более подробное описание этой схемы приводится в следующем разделе.

Нейроконтроллер параллельного типа используется для настройки управляющего входного сигнала u1, который является выходным сигналом обычного контроллера. Настройка выполняется таким образом, чтобы выходной сигнал объекта управления y как можно точнее соответствовал заданному опорному сигналу r. Задача нейроконтроллера параллельного типа заключается в том, чтобы подкорректировать управляющее

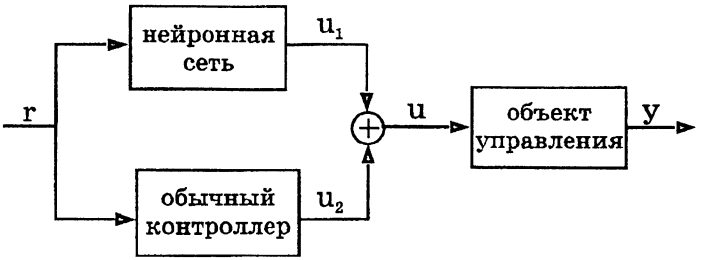


Рис. 4.5.1. Параллельная схема нейроконтроллера.

воздействие u_1, если оно не обеспечивает хорошего результата. Например, хороший повар в ресторане может приготовить отличный обед, однако на каждом столе должны стоять соль и специи, чтобы каждый мог добавить их по вкусу.

Другой пример: инженеры по системам управления могут разработать в лаборатории отличный контроллер, который дает определенную величину u_1, однако в реальных производственных условиях может потребоваться настройка уровня входного управляющего сигнала, например в соответствии с конкретным видом продукции. Из приведенных примеров следует, что даже если удается разработать хорошую общую стратегию управления, может возникнуть необходимость в ее настройке с целью получения лучших практических результатов.

4.5.1. Алгоритмы обучения для параллельной схемы управления

Общая конфигурация управления для нейроконтроллера параллельного типа показана на рис. 4.5.2. На этой схеме блок NN1 представляет собой нейронную сеть для эмулятора объекта управления, создающего оценку $\hat{y}$ выходной координаты объекта управления y. Управляющий сигнал u_2, представляющий собой выходной сигнал нейронной сети NN2, используется для коррекции управляющего сигнала u_1, создаваемого обычным контроллером. Коррекция выполняется таким образом, чтобы обеспечить минимизацию рассогласования между опор-

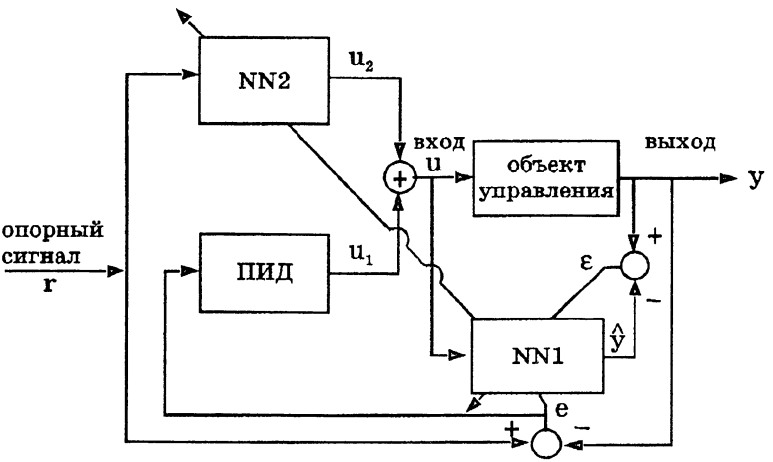


Рис. 4.5.2. Нейроконтроллер параллельного типа с эмулятором.

ным сигналом и выходом объекта управления. Таким образом, блок NN1 используется для эмуляции якобиана системы, необходимого для получения эквивалентной ошибки на выходе блока NN2.

Обозначим через e рассогласование (ошибку) между опорным сигналом r и фактическим выходным сигналом объекта управления y. Необходимо обучить сеть NN2 таким образом, чтобы она могла минимизировать среднеквадратическую ошибку. Обозначим ее через E и определим в виде

$$E = \frac{1}{2}(r-y)^2 \qquad (4.5.1)$$

На рис. 4.5.2. u_2 обозначает выход блока NN2. Предположим, что используется трехслойная нейронная сеть, на вход которой поступает сигнал $r_i(t)$, а на выходе появляется сигнал $O_k(t)$. Для упрощения рассуждений будем считать, что i=1, k=1. Таким образом, представим $r_i(t)$ и $O_k(t)$ как r(t) и O(t) соответственно. Кроме того, предположим, что объект управления имеет единичную временную задержку. Таким образом, получим

$$y(t+1) = f_p(u(t)),\ u(t) = u_1(t) + u_2(t),\ u_2(t) = O(t). \qquad (4.5.2)$$

Чтобы получить алгоритм обучения, применим метод наискорейшего спуска для минимизации ошибки E. Получим

$$\Delta w_j(t+1) = -\eta \frac{\partial E}{\partial w_j} + \alpha \Delta w_j(t), \qquad (4.5.3)$$

$$\Delta w_i(t+1) = -\eta \frac{\partial E}{\partial w_i} + \alpha \Delta w_i(t), \qquad (4.5.4)$$

где η и α — скорость обучения и мгновенный член соответственно. Через w_j обозначен вес связи между j-м нейроном скрытого слоя и выходным нейроном выходного слоя, а через w_i — вес связи между выходным нейроном выходного слоя и i-м нейроном скрытого слоя. Через Δw обозначено приращение веса связи, задаваемое выражением

$$\Delta w = w(new) - w(old). \qquad (4.5.5)$$

Обозначив

$$\delta_\bullet = \frac{\partial E}{\partial net_\bullet}, \qquad (4.5.6)$$

где

$$net_\bullet = \sum_j w_j O_j + \theta_\bullet, \qquad (4.5.7)$$

получим

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial net_\bullet} \frac{\partial net_\bullet}{\partial w_j} = -\delta_\bullet O_j. \qquad (4.5.8)$$

Используя сигмоидную функцию, отношение «вход-выход» для нейрона запишем в следующей форме:

$$O_\bullet(t) = f(net_\bullet), \quad f(x) = \frac{1}{1+e^{-x}},$$

а из отношения «вход-выход» для объекта управления получим

$$y(t+1) = f_p(u(t)),\ u(t) = u_1(t) + u_2(t),\ u_2(t) = O(t). \qquad (4.5.9)$$

Таким образом, получим:

$$\begin{aligned}\delta_{\bullet} &= \frac{\partial E}{\partial y(t+1)}\frac{\partial y(t+1)}{\partial u(t)}\frac{\partial u(t)}{\partial net_{\bullet}} = \\ &= (r(t+1)-y(t+1))\frac{\partial y(t+1)}{\partial u(t)}\left(\frac{\partial u_1(t)}{\partial net_{\bullet}}+\frac{\partial u_2(t)}{\partial net_{\bullet}}\right) = \\ &= (r(t+1)-y(t+1))\frac{\partial y(t+1)}{\partial u(t)}u_2(t)(1-u_2(t)).\end{aligned} \qquad (4.5.10)$$

Здесь использовалось равенство $\partial u_1/\partial net. = 0$.

Если в качестве функции активации на выходном слое использовать отображение тождества, т.е. $f(x)=x$, то получим

$$\delta_{\bullet}^{I} = (r(t+1)-y(t+1))\frac{\partial y(t+1)}{\partial u(t)}, \qquad (4.5.11)$$

где $\delta_{\bullet}^{I}$ обозначает то же самое, что и $\delta_{\bullet}$, но при использовании отображения тождества в качестве функции активации.

Таким образом, имеем

$$\Delta w_j(t+1) = -\eta\delta_{\bullet}O_j + \alpha\Delta w_j(t) \qquad (4.5.12)$$

или

$$\Delta w_j(t+1) = -\eta\delta_{\bullet}^{I}O_j + \alpha\Delta w_j(t). \qquad (4.5.13)$$

Для i-го элемента скрытого слоя имеем

$O_{\bullet}(t) = f(net_{\bullet})$, $net_i = w_i r + \theta_i$.

Определяя δ_i как

$$\delta_i = -\frac{\partial E}{\partial net_i}, \qquad (4.5.14)$$

получим

$$\begin{aligned}\delta_i &= -\frac{\partial E}{\partial net_i} = -\frac{\partial E}{\partial net_{\bullet}}\frac{\partial net_{\bullet}}{\partial O_i}\frac{\partial O_i}{\partial net_i} = \\ &= \delta_{\bullet}w_i f'(net_i) = \delta_{\bullet}w_i O_i(1-O_i).\end{aligned} \qquad (4.5.15)$$

Используя цепное правило, получим

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial net_i}\frac{\partial net_i}{\partial w_i} = -\delta_i r. \qquad (4.5.16)$$

Таким образом, обучающее правило для узлов скрытого слоя запишем в виде

$$\Delta w_i(t+1) = -\eta\delta_i r + \alpha\Delta w_i(t). \qquad (4.5.17)$$

Следовательно, правило нейронного управления для параллельной схемы можно задать в виде следующих шагов.

Шаг 1. Установить начальные значения w_j, w_i, θ_j, θ_i, η и α. Установить t=0 и перейти к шагу 2.

Шаг 2. Подать опорный (желаемый выходной) сигнал r(t) на вход нейронной сети. Вычислить $\delta_\bullet$ или $\delta_\bullet^I$ по формуле

$$\delta_\bullet = u(t)(1-u(t))e(t+1)f_p'(u(t))$$

или

$$\delta_\bullet = c(t+1)f_p'(u(t)),$$

где

$$e(t+1) = r(t+1) - y(t+1).$$

Шаг 3. Изменить веса связей:

$$\Delta w_j(t+1) = -\eta\delta_\bullet O_j + \alpha\Delta w_j(t).$$

или

$$\Delta w_j(t+1) = -\eta\delta_j^I O_j + \alpha\Delta w_j(t).$$

Шаг 4. Вычислить δ_i или δ_i^I по формуле

$$\delta_i = \delta_\bullet w_i O_i(1-O_i),$$

или

$$\delta_i = \delta_\bullet w_i.$$

Шаг 5. Изменить веса связей на следующую величину:

$$\Delta w_i(t+1) = \eta\delta_i r(t) + \alpha\Delta w_i(t)$$

или

$$\Delta w_i(t+1) = \eta \delta_i^J r(t) + \alpha \Delta w_i(t).$$

Шаг 6. $t \to t+1$ и перейти к шагу 2.

Следует отметить, что в случае нейроконтроллера последовательного типа обучающее правило использует якобиан объекта управления $f'_p(u(t)) = \partial y(t+1)/\partial u(t)$. Его можно вычислить приближенно, используя численную разность или применяя сеть-эмулятор, показанную на рис. 4.5.2.

4.6. Алгоритм обучения по ошибке обратной связи

Этот алгоритм предложен в работах [40], [41] и представляет собой особый случай схемы нейронного управления параллельного типа. Первоначальный вариант схемы обучения по ошибке обратной связи показан на рис. 4.2.4. Ошибка обратной связи представляет собой выход контроллера обратной связи, она подается обратно через слои нейронной сети в течение нескольких тысяч циклов обучения, пока не достигается сходимость. После этого нейронная сеть принимает на себя управление в качестве основного контроллера, исключая действие обычного контроллера. Общая схема управления с обучением по ошибке обратной связи, как уже отмечалось, показана на рис. 4.2.4.

В данном разделе рассматривается модифицированный вариант этой схемы обучения, показанной на рис. 4.5.2. В ходе обучения минимизируется функция ошибки E, задаваемая выражением (4.5.1). Предполагается, что NN2 — трехслойная нейронная сеть, а функция активации на выходном слое — отображение тождества, т.е. $f(x) = x$. Используя алгоритм наискорейшего спуска, получим для выходного слоя

$$\Delta w_{kj}(t+1) = -\eta \frac{\partial E}{\partial w_{kj}} + \alpha \Delta w_{kj}(t) \qquad (4.6.1)$$

и для скрытого слоя

$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}(t). \qquad (4.6.2)$$

Используя цепное правило, получим

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}}. \qquad (4.6.3)$$

$$\frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial u} \left(\frac{\partial u}{\partial u_2} \frac{\partial u_2}{\partial net_k} + \frac{\partial u}{\partial u_1} \frac{\partial u_1}{\partial net_k} \right) =$$
$$= -(r-y) \frac{\partial y}{\partial u} \frac{\partial u_2}{\partial net_k} = -(r-y) \frac{\partial y}{\partial u}, \qquad (4.6.4)$$

где u_1 и u_2 — управляющие входные сигналы, задаваемые соответственно обычным и нейронным контроллером. Здесь

$$u_2 = net_k = \sum_j w_{kj} O_j + \theta_k \text{ и } u = u_1 + u_2.$$

Таким образом, если положить

$$\delta_k = -\frac{\partial E}{\partial net_k},$$

$$\delta_j = \frac{\partial E}{\partial net_j}, \qquad (4.6.6)$$

то получим

$$\Delta w_{kj}(t+1) = -\eta \delta_k O_j + \alpha \Delta w_{kj}(t), \qquad (4.6.7)$$

$$\delta_k = e \frac{\partial y}{\partial u} = e \frac{\left|\frac{\partial y}{\partial u}\right|}{\left|\frac{\partial y}{\partial u}\right|} \operatorname{sign}\left(\frac{\partial y}{\partial u}\right), \qquad (4.6.8)$$

$$e = r - y, \qquad (4.6.9)$$

$$\Delta w_{ji}(t+1) = -\eta \delta_j O_i + \alpha \Delta w_{ji}(t), \qquad (4.6.10)$$

$$\delta_j = \delta_k w_{kj} O_j (1 - O_j), \qquad (4.6.11)$$

где sign(x)=1 для $x \geq 0$ и -1 для $x < 0$; $O_j = f(net_j)$ — выход j-го нейрона скрытого слоя. Для этого алгоритма также требуется знать якобиан объекта управления $\frac{\partial y}{\partial u}$. Для его задания имеют-

ся два способа: применить эмулятор, показанный на рис. 4.5.2, или найти знак якобиана и выбрать скорость обучения η, используя в дальнейшем абсолютную величину якобиана $\left|\frac{\partial y}{\partial u}\right|$.

Алгоритм обучения по ошибке обратной связи основан на механизме управления движением, реализуемом в мозге. Пример такой схемы обучения, взятый из биологии, приведен в разделе 4.2.1. То, что именно такой механизм обучения используется в человеческом мозге, доказано в биологии [54]—[56]. Для практического применения можно подобрать лучшую схему для моделирования нейронных сетей NN1 и NN2. Их можно моделировать, используя архитектуры, показанные на рис. 4.3.4 и 4.3.6; кроме того, исходя из своего опыта, авторы рекомендуют для ускорения моделирования эмулятора использовать структуру, показанную на рис. 4.3.7.

4.7. Расширение применения нейроконтроллеров параллельного типа

Еще одно применение таких нейроконтроллеров состоит в решении задач управления нелинейными системами. Такой подход предложен в работе [13] и расширен в работе [57]. Основная его идея состоит в том, чтобы в качестве аппроксимации первого порядка создать линейную математическую модель нелинейной динамической системы, а затем применить нейронную сеть для уменьшения расхождения между выходом модели и фактическим выходом системы. Используя линейную модель, можно определить закон оптимального управления для квадратичной функции затрат. Однако он не должен быть действительным законом управления, так как выведен на основе линеаризованной математической модели. С целью настройки управляющего входного сигнала, обеспечивающей минимизацию квадратичной функции затрат для реального объекта управления, вводится нейронная сеть. Соответствующий алгоритм рассматривается ниже.

4.7.1. Описание системы управления

Рассмотрим обобщенный объект управления, который описывается системой уравнений

$$x(t+1) = Ax(t) + A^*(x(t)) + bu(t), \tag{4.7.1}$$

$$y(t+1) = cx(t+1), \tag{4.7.2}$$

где u(t) и y(t) — скалярные функции, обозначающие соответственно вход и выход системы; $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ — n-мерный вектор состояния; A,b,c — параметры системы; A* — обозначение нелинейной части системы. Эта составляющая необходима в качестве дополнительного члена, когда выполняется подстройка линейной модели в качестве модели объекта управления. Задача управления состоит в том, чтобы найти оптимальное решение, минимизирующее следующую функцию затрат J:

$$J = \sum_{t=0}^{\infty} \left(y^2(t+1) + \gamma\, u^2(t) \right), \tag{4.7.3}$$

где γ — положительная константа.

4.7.2. Линеаризованная система управления

На рис. 4.7.1 приведена оптимальная система управления для аппроксимированной линейной системы, построенная методами линейной теории управления. Используя адаптивный алгоритм наблюдения Люнбергера, запишем алгоритм оценки состояния в виде

$$\begin{aligned} \bar{x}^a(t+1) &= \bar{A}\bar{x}(t) + \bar{b}u(t), \\ \bar{y}^a(t+1) &= C\bar{x}^a(t+1), \\ \bar{x}(t+1) &= \bar{x}^a(t+1) + f(y(t+1) - \bar{y}^a(t+1)), \end{aligned} \tag{4.7.4}$$

где $\bar{x}^a$ и $\bar{x}$ обозначают соответственно априорную и апостериорную оценки x, а f — вес наблюдателя, задаваемый обычно проектным правилом алгоритма наблюдения. В соответствии с теорией оптимального управления входной сигнал u(t) для апп-

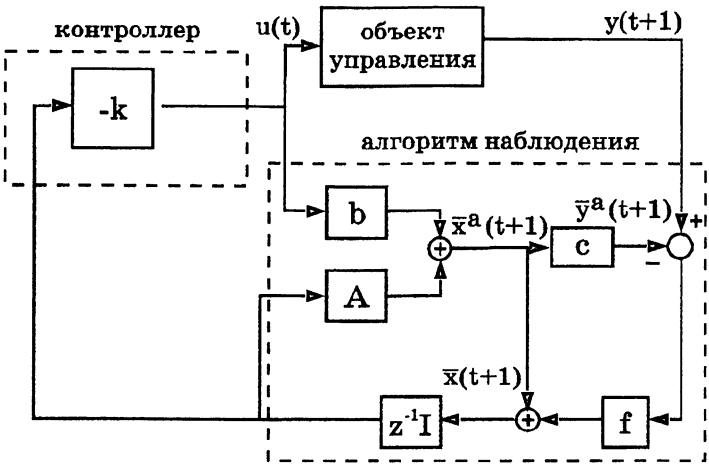


Рис. 4.7.1. Оптимальная система управления для линеаризованной системы, где I - единичная матрица, а z^{-1} - оператор задержки).

роксимированной линейной системы, минимизирующий функцию затрат J, записывается в виде

$$u(t) = -k\bar{x}(t),$$

где k — n-мерный вектор-строка коэффициентов передачи регулятора, который определяется на основе матричного алгоритма Риккати [57].

4.7.3. Система управления с нейронными сетями

На рис. 4.7.2 показана система управления с нейронно-сетевой компенсацией: в такой системе нейронные сети используются для компенсации рассогласования между управляющими входными сигналами и оценками алгоритма наблюдения. Здесь сеть NN1 компенсирует включенные в уравнение системы нелинейности, которые не могут учитываться линейным алгоритмом наблюдения. Сеть NN2 предназначена для настройки управляющего входного сигнала таким образом, чтобы выход системы приближался к нулю.

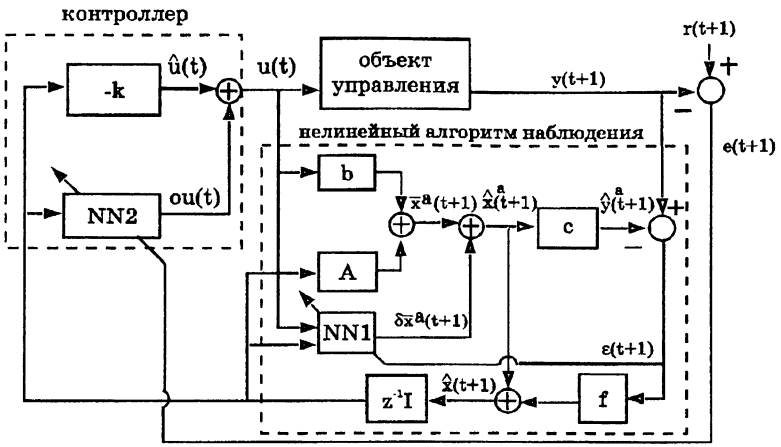


Рис. 4.7.2. Системы управления с использованием нейронных сетей, где r(t+1) - желаемый сигнал; здесь предполагается, что r(t+1)=0.

4.7.4. Нелинейный алгоритм наблюдения для нейронной сети

Как показано на рис. 4.7.2, априорная оценка $\bar{x}^a(t+1)$, полученная с использованием алгоритма наблюдения Люенбергера (Luenberger), задается выражением

$$\bar{x}^a(t+1) = \bar{A}\bar{x}(t) + \bar{b}u(t), \qquad (4.7.6)$$

где $\bar{x}(t)$ — наилучшая оценка x(t). Однако в оценке $\bar{x}^a(t+1)$ не учитываются какие-либо нелинейные характеристики уравнения состояния. Корректировка априорной оценки $\delta\bar{x}^a(t+1)$ выполняется с использованием нейронной сети NN1, показанной на рис. 4.7.2. Определим $\hat{x}^a(t+1)$ следующим выражением:

$$\hat{x}^a(t+1) = \bar{x}^a(t+1) + \delta\bar{x}^a(t+1), \qquad (4.7.7)$$

где $\delta\bar{x}^a(t+1)$ — выход нейронной сети NN1, обученной таким образом, что ошибка выхода ε(t+1) минимизируется.

$$\varepsilon(t+1) = y(t+1) - y^a(t+1),$$
$$y^a(t+1) = c\hat{x}^a(t+1). \qquad (4.7.8)$$

В этом случае оценка $\hat{x}(t+1)$ величины $x(t+1)$ задается следующим образом:

$$\hat{x}(t+1) = \hat{x}^a(t+1) + f(y(t+1) + y^a(t+1)), \qquad (4.7.9)$$

где обучающее правило строится на основе алгоритма обратного распространения, что обеспечивает минимизацию квадратичной ошибки $\hat{}$, задаваемой выражением

$$\hat{E} = \frac{1}{2}\varepsilon^2(t+1). \qquad (4.7.10)$$

Если определить $y^r(t+1) = \frac{1}{2}\varepsilon^2(t+1)$ как

$$y^r(t+1) = y(t+1) - \bar{y}^a(t+1) = c(x(t+1) - \bar{x}^a(t+1)), \qquad (4.7.11)$$

а $\delta y(t+1)$ как $\delta y(t+1) = c\delta\bar{x}^a(t+1)$,

где

$$\bar{y}^a(t+1) = \bar{x}^a(t+1),$$

то величина $\hat{}$, задаваемая выражением (4.7.10), примет следующий вид:

$$\hat{E} = \frac{1}{2}(y^r(t+1) - \delta y(t+1))^2. \qquad (4.7.12)$$

Таким образом, если рассматривать $y^r(t+1)$ как желаемый сигнал, а $\delta y(t+1)$ — как часть выходного сигнала, задаваемую нейронной сетью NN1, то метод обратного распространения может применяться непосредственно для минимизации ошибки $\hat{}$. Здесь $\delta y(t+1) = c\delta\bar{x}^a(t+1)$ — выходная часть нейронной сети NN1, а $\bar{x}(t)$ и $u(t)$ — входы сети NN1.

4.7.5. Нелинейный контроллер для нейронной сети

Как показано на рис. 4.7.2, управляющий входной сигнал $u(t)$ задается выражением

$$u(t) = \hat{u}(t) + \delta u(t), \qquad (4.7.13)$$

$$\hat{u}(t) = -k\hat{x}(t), \qquad (4.7.14)$$

где k — коэффициент передачи цепи обратной связи, полученный в разделе 4.7.2, а $\delta u(t)$ — выход нейронной сети NN2, минимизирующей функцию затрат:

$$E = \frac{1}{2}(e^2(t+1) + w\,u^2(t)). \qquad (4.7.15)$$

Таким образом, если обозначить веса связей выходного и входного слоев w_{kj} и w_{ji} соответственно, то получим

$$\Delta w_{kj}(t+1) = -\eta \frac{\partial E}{\partial w_{kj}} + \alpha \Delta w_{kj}(t),$$

$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}(t), \qquad (4.7.16)$$

где следует положить k=1, так как в выходном слое число нейронов равно единице. Используя цепное правило, получим

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}}, \qquad (4.7.17)$$

где

$$net_k = \sum_j w_{kj} O_j. \qquad (4.7.18)$$

Снова применяя цепное правило, получим

$$\frac{\partial E}{\partial net_k} = \frac{\partial E}{\partial y(t+1)} \frac{\partial y(t+1)}{\partial u(t)} \frac{\partial u(t)}{\partial net_k} + wu(t) \frac{\partial O(t+1)}{\partial net_k}, \qquad (4.7.19)$$

где $O(t+1)=\delta u(t)$ — выходной сигнал нейронной сети NN2.

В качестве функции активации выходного слоя будем использовать отображение тождества. Тогда получим $O(t+1)=net_k$ и

$$\frac{\partial E}{\partial net_k} = -e(t+1) \frac{\partial y(t+1)}{\partial u(t)} + ru(t), \qquad (4.7.20)$$

так как $\partial u(t)/\partial net_k = \partial O(t+1)/\partial net_k = 1$. Далее, определяя δ_k как $\delta_k = \partial E/\partial net_k$, получим следующее обучающее правило:

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \Delta w_{kj}(t), \qquad (4.7.21)$$

$$\delta_k = -e(t+1)\frac{\partial y(t+1)}{\partial u(t)} - ru(t). \qquad (4.7.22)$$

Для скрытого слоя получим

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \Delta w_{ji}(t), \qquad (4.7.23)$$

$$\delta_j = \delta_k w_{kj} O_j (1 - O_j). \qquad (4.7.24)$$

Таким образом, можно построить оптимальную стратегию управления для данного класса нелинейных систем.

4.7.6. Численное моделирование

Рассмотрим объект управления, описываемый следующими уравнениями:

$$\begin{bmatrix} x_{m1}(t+1) \\ x_{m2}(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t),$$

$$x_1(t+1) = x_{m1}(t+1) + \eta(1 - \exp(x_{m2}(t+1)^2),$$

$$x_2(t+1) = x_{m2}(t+1) + \eta(1 - \exp(x_{m1}(t+1)^2),$$

$$y(t+1) = cx(t+1),$$

$$x(t+1) = \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix},$$

$$c = [0, 1], \qquad (4.7.25)$$

где η обозначает степень нелинейности объекта управления. Минимизируемая функция затрат имеет следующий вид:

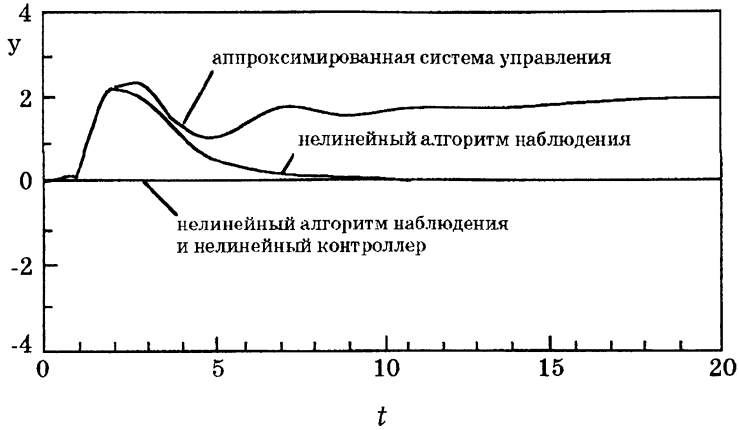
$$J = \sum_{t=0}^{\infty} \left(y^2(t+1) + 2u^2(t) \right). \qquad (4.7.26)$$

Если в качестве грубых оценок для матриц A и b взять следующие значения:

$$A = \begin{bmatrix} 1.1 & -0.1 \\ 0.9 & 1.1 \end{bmatrix}, \quad b = \begin{bmatrix} 1.2 \\ 0.2 \end{bmatrix}, \qquad (4.7.27)$$

то, используя теорию линейных оптимальных систем (LQ), получим

(*a*) Выходной сигнал объекта управления



(*b*) Входной сигнал объекта управления

Рис. 4.7.3. Сравнение результатов для различных алгоритмов управления

$$k=[0.977,\ 0.4000] \tag{4.7.28}$$

для наблюдения с конечным временем, а также

$$f=\begin{bmatrix}1.222\\1.000\end{bmatrix}. \tag{4.7.29}$$

На рис. 4.7.3, *a* и *b* показаны результаты использования различных подходов к управлению. На рис. 4.7.3, *a* показаны выходные сигналы объекта управления, а на рис. 4.7.3, *b* — входные сигналы для этого объекта при использовании аппроксимированного линейного контроллера, нелинейного алгоритма наблюдения, а также нелинейного контроллера в сочетании с нейроконтроллером. Из этих результатов видно, что рассмотренный выше подход на основе нейронного управления является более предпочтительным по сравнению с другими традиционными методами.

4.8. Схема нейронного управления с самонастройкой

Схема нейронного управления с самонастройкой показана на рис. 4.8.1. Здесь нейронная сеть используется для настройки параметров обычного контроллера подобно настройке, выполняемой человеком-оператором.

Человек-оператор накапливает определенный опыт и знания о системе управления. Однако, в отличие от компьютера, человек-оператор, как правило, не может помнить все данные о прошлом системы для всех видов рабочих состояний. Компьютер может легко хранить такую информацию и мгновенно отыскивать ее. Таким образом, если удастся включить знания и опыт оператора в нейронную сеть и обучить ее на основе данных о прошлом системы, то обученную таким образом сеть можно будет использовать как средство оперативной настройки параметров контроллера.

Такой подход может непосредственно применяться для многих традиционных технологий управления, включающих методы адаптивного управления. Многие методы адаптивного управления включают ряд параметров или задаваемых пользователем полиномиальных коэффициентов, которые требуется выбирать или настраивать предварительно. Обычно они выбираются методом проб и ошибок. Встроив в схему управления нейронную сеть, можно затем использовать эту сеть для опера-

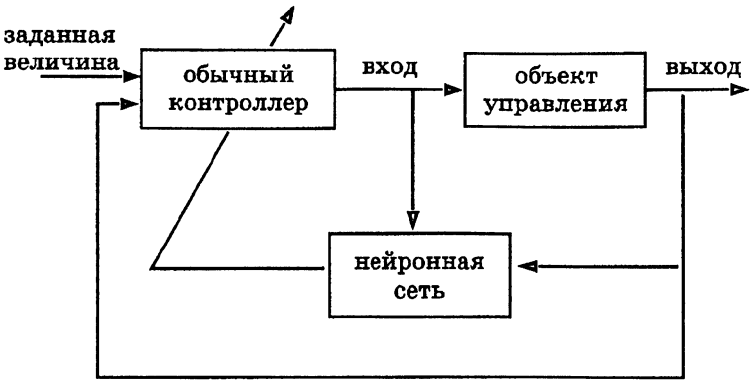


Рис. 4.8.1. Схема нейронного управления с самонастройкой.

тивной настройки указанных параметров. Таким образом, стратегия нейронного управления с самонастройкой может применяться во многих традиционных технологиях управления. Ниже будет рассмотрена схема нейронного управления с самонастройкой, в которой нейронная сеть применяется для настройки параметров ПИД-контроллера (схема ПИД-нейронного управления с самонастройкой).

4.9. ПИД-нейроконтроллер с самонастройкой

ПИД-контроллеры имеют давнюю историю в управлении промышленными объектами, они зарекомендовали себя как надежные, простые и устойчивые средства для многих реальных приложений. В частности, в Японии доля ПИД-контроллеров среди всех средств управления, применяемых на практике, составляет 84% [4]. Это подтверждает, что ПИД-контроллеры пригодны для многих практических задач благодаря простоте их структуры и принципов работы.

Схема ПИД-управления приведена на рис. 4.9.1, где k_c — коэффициент пропорциональности, T_i — постоянная интегри-

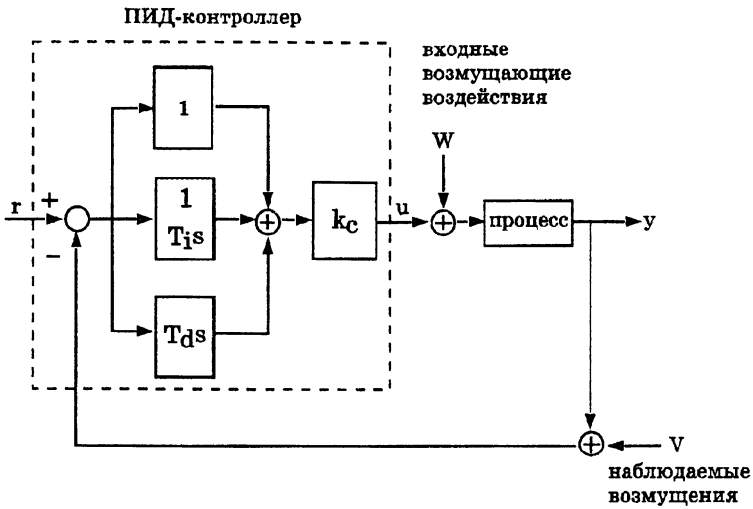


Рис. 4.9.1. Общая схема ПИД-контроллера

рования, а T_d — постоянная дифференцирования. Эти параметры настраиваются в соответствии с желаемым выходным сигналом. Следует обратить внимание, что на этом рисунке для простоты не показаны возмущения.

Управляющий входной сигнал u(τ) ПИД-контроллера задается выражением

$$u(\tau) = k_c\left(e(\tau) + \frac{1}{T_i}\int_0^{\tau} e(\tau')d\tau' + T_d\frac{de(\tau)}{d\tau}\right). \qquad (4.9.1)$$

Упрощенно говоря, пропорциональное действие связано с текущей ошибкой, интегрирующее — с предысторией ошибки, а дифференцирующее — с будущим поведением ошибки. С точки зрения оценивания, пропорциональное, интегрирующее и дифференцирующее действие соответствуют фильтрации, сглаживанию и прогнозированию соответственно.

4.9.1. Вывод уравнений ПИД-нейроконтроллера с самонастройкой

На рис. 4.9.2 показана общая структура управления для обучения ПИД-контроллера с самонастройкой. Нейронная сеть здесь используется вместо человека-оператора так, что обеспечивается минимизация ошибки путем настройки ПИД-коэффициентов [58]. ПИД-контроллер с дискретным временем описывается уравнениями

$$u(t) = u(t-1) + K_P(e(t) - e(t-1)) + \\ + K_I e(t) + K_D(e(t) - 2e(t-1) + e(t-2)), \qquad (4.9.2)$$

$$e(t)=r(t)-y(t),$$

где K_p, K_i и K_D — ПИД-коэффициенты, а $r(t)$ — заданная (желаемая) величина выходного сигнала объекта управления. Чтобы вывести алгоритм самонастройки ПИД-контроллера, зададим функцию стоимости E, подлежащую минимизации, в виде

$$E = \frac{1}{2}e^2(t+1). \qquad (4.9.3)$$

Используя трехслойную нейронную сеть, реализуем обучающее правило для поиска подходящих значений ПИД-коэффициентов. Таким образом, выходными сигналами выходного слоя будут величины K_p, K_i и K_D, обозначенные через O(1), O(2) и O(3) соответственно. На основе алгоритма наискорейшего спуска получим для выходного слоя

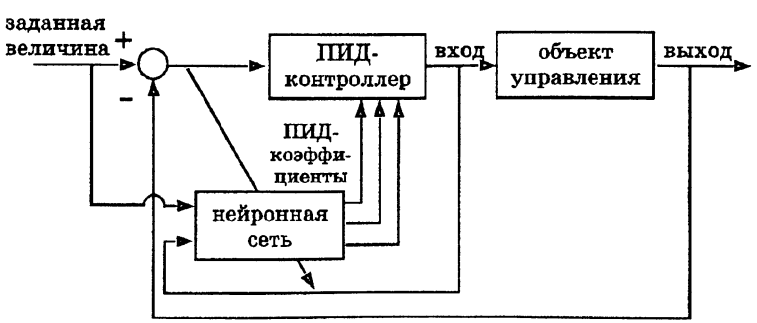


Рис. 4.9.2. Схема обучения ПИД-контроллера с самонастройкой

$$\Delta w_{kj}(t+1) = -\eta \frac{\partial E}{\partial w_{kj}} + \alpha \Delta w_{kj}(t), \qquad (4.9.4)$$

а для скрытого слоя

$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E}{\partial w_{ji}} + \alpha \Delta w_{ji}(t). \qquad (4.9.5)$$

Определим, что

$$\delta_k = \frac{\partial E}{\partial net_k}, \qquad (4.9.6)$$

где

$$net_k = \sum_j w_{kj} O_j + \theta_k.$$

Обозначим выход k-го нейрона выходного слоя через O(k). Тогда имеем

$$O(k) = f(net_k). \qquad (4.9.7)$$

Однако значения ПИД-коэффициентов не ограничены диапазоном от 0 до 1. Поэтому после выхода сети O(k) можно использовать некоторый коэффициент передачи c, такой, что

$$O(k) = cf(net_k), c > 0. \qquad (4.9.8)$$

Параметр c также можно определить путем обучения, увеличивая количество нейронов в скрытом слое. Для простоты положим c = 1. Используя цепное правило, получим

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} O_j \qquad (4.9.9)$$

и

$$\delta_k = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial y(t+1)} \frac{\partial y(t+1)}{\partial u(t)} \frac{\partial u(t)}{\partial O(k)} \frac{\partial O(k)}{\partial net_k}. \qquad (4.9.10)$$

Однако

$$\frac{\partial E}{\partial y(t+1)} = \frac{\partial E}{\partial e(t+1)} \frac{\partial e(t+1)}{\partial y(t+1)} = -(r(t+1) - y(t+1)) = -e(t+1),$$

$$\frac{\partial u(t)}{\partial O(k)} = f'(net_k) = O(k)(1 - 0(k)),$$

$$\frac{\partial u(t)}{\partial O(k)} = \begin{cases} e(t) - e(t-1) & ; k = 1 \\ e(t) & ; k = 2 \\ e(t) - 2e(t-1) + e(t-2) & ; k = 1. \end{cases} \qquad (4.9.11)$$

Последнее соотношение получено из (4.9.2) с учетом того, что $O(1) = K_p$, $O(2) = K_I$ и $O(3) = K_D$.

Таким образом, имеем

$$\Delta w_{kj}(t+1) = -\eta \delta_k O_j + \alpha \Delta w_{kj}(t), \qquad (4.9.12)$$

где

$$\delta_k = e(t+1)\frac{\partial y(t+1)}{\partial u(t)} O(k)(1 - O(k))\frac{\partial u(t)}{\partial O(k)}. \qquad (4.9.13)$$

Для скрытого слоя получим

$$\frac{\partial E}{\partial net_j} = \sum_k \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial O_j}\frac{\partial O_j}{\partial net_j}\frac{\partial net_k}{\partial w_{kj}} = -\sum_k \delta_k w_{kj} f'(net_j) =$$
$$= -\sum_k \delta_k w_{kj} O_j (1 - O_j).$$

Таким образом, имеем

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t), \qquad (4.9.14)$$

$$\delta_j = -\frac{\partial E}{\partial net_j} = \sum_k \delta_k w_{kj} O_j (1 - O_j). \qquad (4.9.15)$$

Чтобы вычислить коэффициенты δ_k по выражению (4.9.13), требуется знать якобиан системы $\partial y(t+1)/\partial u(t)$, и для получения его оценки используется эмулятор, как показано выше. Общая схема показана на рис. 4.9.3. Таким образом, алгоритм обучения ПИД-нейроконтроллера с самонастройкой можно представить в следующей форме.

Шаг 1. Установить начальные значения w_{kj}, w_{ji}, θ_k, θ_j, η и α. Установить t=0 и перейти к шагу 2.

Шаг 2. Вычислить значения e(t+1) и δ_k:

$$e(t+1) = r(t+1) - y(t+1),$$

$$\delta_k = e(t+1)\frac{\partial y(t+1)}{\partial u(t)} O(k)(1 - O(k))\frac{\partial u(t)}{\partial O(k)}, \quad k=1, 2, 3,$$

где

$$\frac{\partial u(t)}{\partial O(k)} = \begin{cases} e(t) - e(t-1) & ; k = 1 \\ e(t) & ; k = 2 \\ e(t) - 2e(t-1) + e(t-2) & ; k = 1. \end{cases}$$

Шаг 3.

$$\Delta w_{kj}(t+1) = -\eta \delta_k O_j + \alpha \Delta w_{kj}(t).$$

Шаг 4. Вычислить δ_j:

$$\delta_j = \sum_k \delta_k w_{kj} O_j (1 - O_j).$$

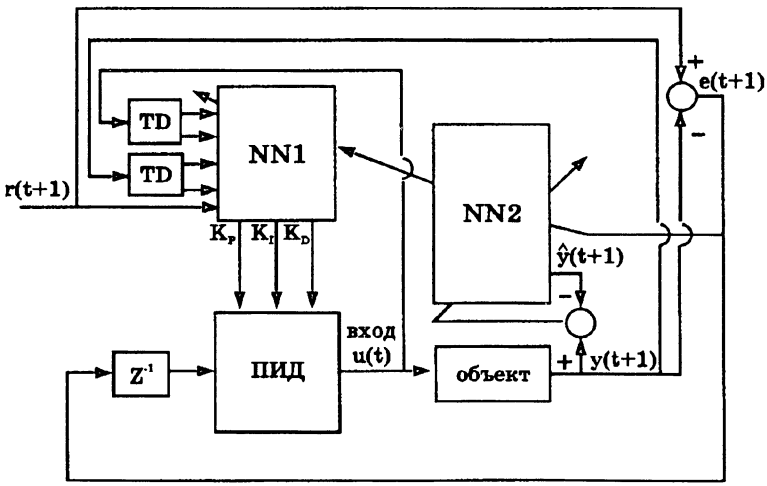


Рис. 4.9.3. Общая схема ПИД-нейроконтроллера с самонастройкой, где нейронная сеть NN1 используется для определения ПИД-коэффициентов, а NN2 — для определения якобиана системы

Шаг 5.

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t).$$

Шаг 6. $t \rightarrow t+1$ и перейти к шагу 2.

Следует отметить, что для вычисления якобиана системы $\partial y(t+1)/\partial u(t)$ используется эмулятор NN2, как показано на рис. 4.9.3.

4.9.2. Примеры эмуляции

В данном примере рассмотрим две модели: линейную (M1) и нелинейную (M2):

$$M1: \quad y(t+1) = 0.988y(t) + 0.232u(t), \tag{4.9.16}$$

$$M2: \quad y(t+1) = 0.9y(t) - 0.001y(t-1)^2 + u(t) + \sin(u(t-1)). \tag{4.9.17}$$

Модель M1 построена для системы управления температурой водяной ванны, рассмотренной в разделе 4.4.6; при построении модели использован алгоритм идентификации на основе метода наименьших квадратов. Модель M2 — нелинейная форма модели M1.

Характеристики двух систем на этапе обучения для моделей M1 и M2 показаны на рис. 4.9.4 и 4.9.5 соответственно. Опорные температуры поддерживались на уровне 35 °C для M1 и 40 °C для M2. Обучение осуществлялось путем многократного (несколько тысяч раз) повторения желаемых циклов траектории. Нейронные сети использовались для минимизации рассогласования между опорным и фактическим выходом объекта управления. Сходимость считалась достигнутой, когда рассогласование оказывалось меньше заданной величины. После обучения наблюдались хорошие характеристики системы даже для тех опорных величин, для которых обучение не проводилось; это показано на рис. 4.9.6 и 4.9.7 для моделей M1 и M2 соответственно. Можно также видеть, что ПИД-нейроконтроллер с самонастройкой может отслеживать опорную величину практически точно, даже если входные сигналы ограничены, т.е. при имитации реальных управляющих входных сиг-

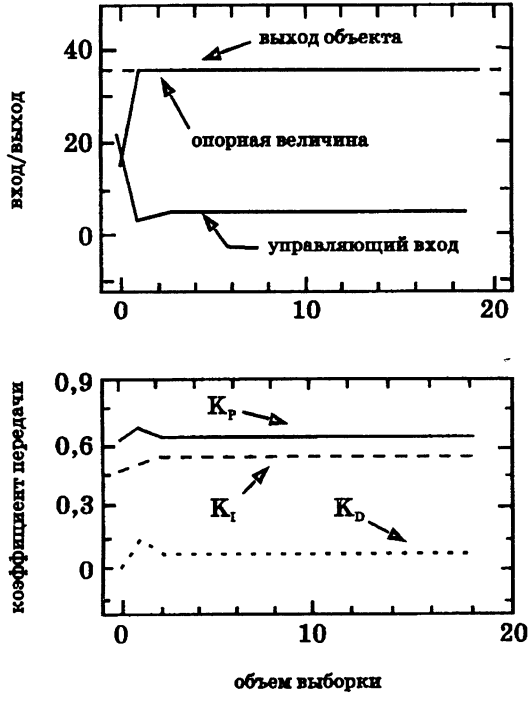


Рис. 4.9.4. Характеристики модели M1 на этапе обучения.

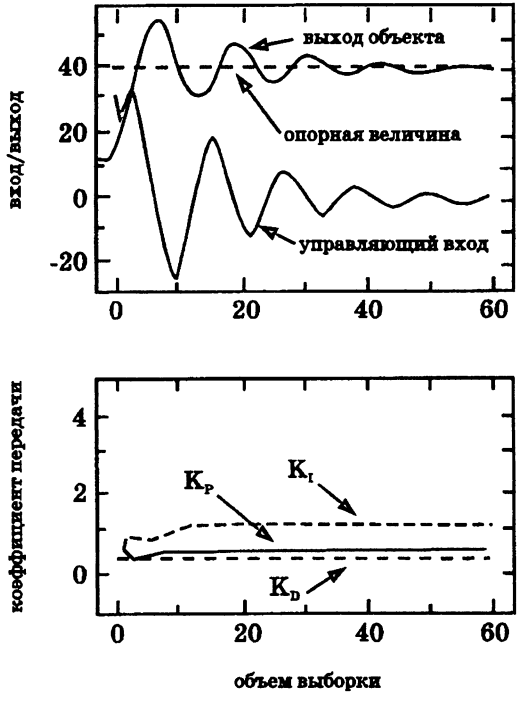


Рис. 4.9.5. Характеристики модели M2 на этапе обучения.

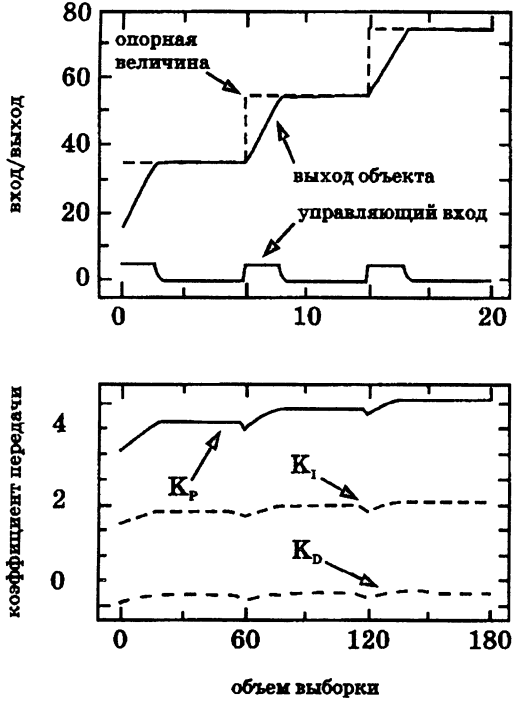


Рис. 4.9.6. Характеристики отслеживания для модели М1 после обучения.

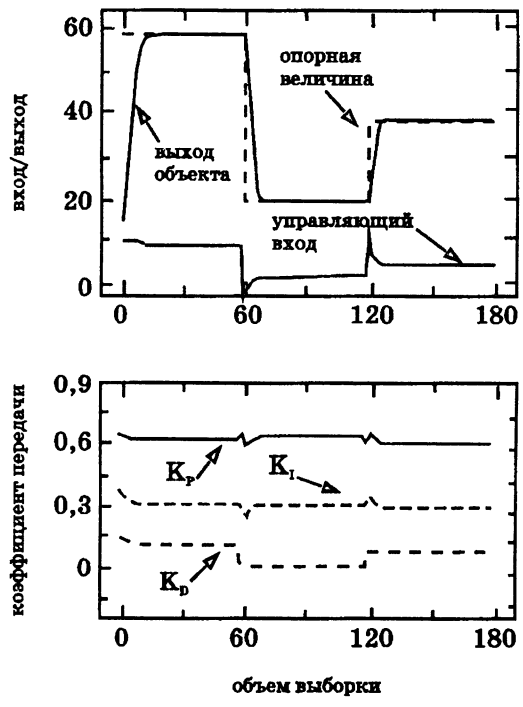


Рис. 4.9.7. Характеристики отслеживания для модели М2 после обучения.

налов (например, при $u(t) \in [0, 5]$ для модели М1 и $u(t) \in [0, 10]$ для модели М2). Реальный пример применения другого метода нейронного управления (для системы управления температурой водяной ванны) будет рассмотрен в следующей главе.

4.10. Схема нейронного управления с эмулятором и контроллером (схема обратного распространения во времени)

Схема нейронного управления, показанная на рис. 4.2.5, представляет собой в основном интегрированный и модифицированный вариант инверсно-прямой схемы управления (рис. 4.2.2) и схемы специализированного обучения (рис. 4.2.3). При таком подходе для приближенного нахождения якобиана объекта, или частных производных $\partial y / \partial u$, используется эмулятор нейронной сети (вместо знаков якобиана). Сначала схема включает автономное обучение нейросетевых моделей с целью изучения инверсной и прямой нейросетевых моделей объекта управления. Нейросетевая модель, обученная таким образом, чтобы обеспечивать обучение инверсии объекта, используется в качестве контроллера. В свою очередь, нейросетевая модель, обученная прямой динамике объекта, используется как средство для обратного распространения ошибки характеристики системы (с целью получения эквивалентной ошибки на выходе нейроконтроллера). Вывод уравнений для сигналов ошибки показан в следующих разделах.

4.10.1. Автономное обучение нейроконтроллера и эмулятора

Аналогично моделям инверсно-прямого нейронного управления, рассмотренным ранее, модели нейроконтроллеров обучаются инверсной динамике объекта управления. Для этого используются выходные сигналы объекта управления вместе с несколькими задержанными значениями (в качестве образцов входных сигналов) и соответствующими входными сигналами объекта управления (в качестве желаемых образцов). Нейро-

эмуляторы обучаются прямой динамике объекта управления, используя для этого входные сигналы вместе с несколькими задержанными значениями (в качестве образцов входных сигналов) и соответствующими выходными сигналами (в качестве желаемых образцов). При автономном обучении нейронной сети с целью изучения инверсной или прямой динамической модели объекта управления сигнал обратного распространения ошибки между выходным и скрытым слоями выражается следующим образом:

$$\delta_k = \tau_k - O_k, \qquad (4.10.1)$$

где τ_k — заданный (желаемый) образец, а O_k — действительный выходной сигнал нейронной сети. Между скрытым и входным слоем модели объекта управления, сигнал обратного распространения ошибки имеет вид

$$\delta_j = f'_k(net_j)\sum_k \delta_k w_{kj}. \qquad (4.10.2)$$

Здесь f' — производная функции f, где $f(x)=1/(1+\exp(-x))$. Веса связей между входным и скрытым слоями корректируются следующим образом:

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t) + \beta \Delta w_{ji}(t-1), \qquad (4.10.3)$$

а между скрытым и выходным слоями — в соответствии с выражением

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t) + \beta \Delta w_{kj}(t-1), \qquad (4.10.4)$$

где O_i и O_j — выходные сигналы входного и скрытого слоев соответственно; η — скорость обучения; α и β — коэффициенты мгновенного значения и ускорения соответственно. Константы η, α и β выбираются эмпирическим путем. Выражения (4.10.3) и (4.10.4) представляют собой модифицированные формы обобщенного дельта-правила [56].

4.10.2. Оперативное обучение

Чтобы проиллюстрировать вывод уравнений сигналов ошибки для нейроконтроллера, рассмотрим рис. 4.10.1. Для этой схемы оперативного обучения корректировка весов связей меж-

ду скрытым и выходным слоями нейронной сети — контроллера может быть выражена следующей формулой:

$$\Delta w_{kj}^{C} \infty \frac{\partial E^{E}}{\partial w_{kj}^{C}} = \eta \frac{\partial E^{E}}{\partial w_{kj}^{C}}, \qquad (4.10.5)$$

где

$$E = \frac{1}{2}(r-y)^2,$$

а r и y — соответственно желаемый и фактический выходной сигнал объекта управления. Верхний индекс C обозначает переменные, относящиеся к нейронной сети — контроллеру, а E — переменные, относящиеся к нейронной сети — эмулятору.

Используя цепное правило, запишем выражение (4.10.5) в виде

$$\Delta w_{kj}^{C} = \eta \frac{\partial E^{E}}{\partial net_{k}^{C}} \frac{\partial net_{k}^{C}}{\partial w_{kj}^{C}} = \eta \delta_{k}^{C} O_{j}^{C}, \qquad (4.10.6)$$

где

$$\frac{\partial E^{E}}{\partial net_{k}^{C}} = \delta_{k}^{C} \text{ и } \frac{\partial net_{k}^{C}}{\partial w_{kj}^{C}} = O_{j}^{C}.$$

Используя эти соотношения, покажем вывод выражений для сигналов ошибки, распространяющихся между слоями нейронной сети. δ_{k}^{C} — сигнал ошибки между скрытым и выход-

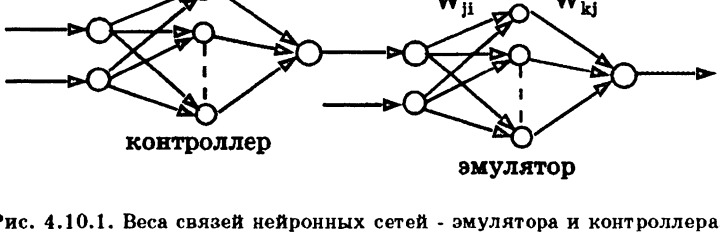


Рис. 4.10.1. Веса связей нейронных сетей - эмулятора и контроллера.

ным слоями нейронной сети — контроллера. На входных и выходных нейронах сетей мы используем линейную функцию, поэтому можно записать следующее уравнение:

$$\delta_k^C = \frac{\partial E^E}{\partial O_k^C} = \frac{\partial E^E}{\partial net_i^E} = \frac{\partial E^E}{\partial O_i^E}, \qquad (4.10.7)$$

где net_i^E и O_i^E — входной и выходной сигналы нейронов входного слоя эмулятора. Снова используя цепное правило, получим

$$\delta_k^C = \frac{\partial E^E}{\partial net_i^E}\frac{\partial net_i^E}{O_i^E}. \qquad (4.10.8)$$

Так как

$$\frac{\partial E^E}{\partial net_j^E} = \delta_j^E \text{ и } \frac{\partial net_j^E}{\partial O_i^P} = w_{ji}^E,$$

то

$$\delta_k^C = \delta_j^E w_{ji}^E. \qquad (4.10.9)$$

Сигнал ошибки между входным и скрытым слоями нейронной сети — контроллера можно записать уравнением

$$\delta_j^C = \frac{\partial E^E}{\partial net_j^C} = \frac{\partial E^E}{\partial O_j^C}\frac{\partial O_j^C}{\partial net_j^C}. \qquad (4.10.10)$$

По цепному правилу

$$\frac{\partial E^E}{\partial O_j^C} = \frac{\partial E^E}{\partial net_k^C}\frac{\partial net_k^C}{\partial O_j^C}$$

и

$$\frac{\partial O_j^C}{\partial net_j^C} = f'\left(net_j^C\right),$$

где

$$f'\left(net_j^C\right) = O_j^C\left(1 - O_j^C\right),$$

значит,

$$\delta_j^C = \delta_k^C w_{kj}^C O_j^C (1 - O_j^C). \qquad (4.10.11)$$

В данной схеме веса связей нейронной сети — эмулятора не поддерживаются постоянными, а оперативно корректируются. Это делается в каждый момент дискретизации с помощью эмулятора посредством обратного распространения ошибки:

$$\varepsilon = \frac{1}{2}(y - \hat{y})^2, \qquad (4.10.12)$$

где y и $\hat{y}$ — выходные сигналы объекта управления и эмулятора соответственно.

Схема нейронного управления с эмулятором и контроллером применяется в системе управления температурой водяной ванны и в системе управления печью со многими входами и выходами (см. главу 5).

4.11. Заключение

В данной главе приведен обзор подходов к нейронному управлению. С точки зрения авторов, можно выделить пять основных схем нейронного управления. Рассмотрены четыре из этих схем, основанные на алгоритме обратного распространения: последовательная схема управления, параллельная схема управления, схема управления с самонастройкой и схема управления с эмулятором и контроллером. Для этих схем приведен вывод уравнений. Представлены также расширения последовательной и параллельной схем нейронного управления. Для некоторых из рассмотренных схем приведены примеры моделирования.

5. ПРИЛОЖЕНИЯ НЕЙРОННОГО УПРАВЛЕНИЯ

5.1. Введение

В этой главе представлены некоторые методы нейронного управления и примеры их практического применения для реальных систем. В качестве таковых здесь рассматриваются: система управления температурой водяной ванны, инвертированный маятник, система управления генератором в электрическом транспортном средстве, печь как многомерный объект управления (со многими входами и выходами). Для системы управления температурой водяной ванны и печи применяется схема нейронного управления с эмулятором и контроллером. Так как эти реальные процессы по своей природе являются медленными, для обучения нейронных сетей сначала применяются методы автономного обучения, а затем для точной настройки характеристик этих сетей применяется оперативное обучение (в соответствии со схемой, показанной на рис. 4.2.5). Для рассматриваемых примеров приводится сравнение предложенных методов с некоторыми традиционными подходами к управлению процессами различной сложности. Так как нейронное управление представляет собой относительно новый подход, сравнение его с традиционными подходами представляет особый интерес.

Для инвертированного маятника в качестве первоначального подхода к управлению используется нечеткая логика с минимальным набором правил. Этот вид управления можно рассматривать как грубое управляющее воздействие, так как нечеткая логика применяется достаточно просто. Однако выбор набора правил и коэффициентов для точной настройки нечеткого контроллера является достаточно сложной задачей. Для ее решения применяется стратегия линейного опти-

мального управления, обеспечивающего стабилизацию маятника в вертикальном положении. Наконец, для точной настройки обоих контроллеров с учетом нелинейных характеристик и неучтенных особенностей динамики системы применяются нейронные сети.

Другой пример — разработка ПИД-нейроконтроллера с самонастройкой для генераторной системы электротранспортного средства. Нейронная сеть здесь используется для оперативной настройки коэффициентов ПИД-контроллера. Эксперименты показывают, что такое применение нейронной сети позволяет уменьшить результирующую ошибку.

5.2. Применение нейронного управления в системе регулирования процессов в водяной ванне и сравнение его с другими схемами управления

5.2.1. Введение

Классическая теория управления с обратной связью послужила основой для разработки простых систем автоматического управления. Основными причинами широкого применения этой теории для разработки промышленных систем управления стали доступность ее принципов и относительная простота реализации. Однако обычные контроллеры с фиксированным коэффициентом передачи являются неэффективными для компенсации вариаций параметров объекта управления. Их адаптация к изменениям внешней среды оставляет желать лучшего. Необходимость разработки контроллера, хорошо настраиваемого не только на одну рабочую точку, но и на весь диапазон рабочих точек, стала основой для выдвижения идеи адаптивного контроллера.

За последние три десятилетия предложено много методов адаптивного управления [1], [2]. Это стало возможным в значительной степени благодаря достижениям современной теории управления и теории устойчивости. Способность автома

тически адаптироваться к изменениям динамики объекта управления и внешней среды сделала адаптивные контроллеры все более привлекательными для различных практических приложений. Однако применение таких контроллеров имеет ряд недостатков. Для их реализации требуется выполнить математическое моделирование объекта управления, что во многих случаях сложно и трудоемко. К тому же неточности модели объекта управления могут привести к ухудшению характеристик контроллера. Они также имеют много параметров, вследствие чего инженеру-практику или оператору сложно связать настройку последних с действиями системы управления. Больше того, адаптивные контроллеры не подходят для нелинейных и сложных систем управления.

Развитие теории искусственных нейронных сетей и нечеткой логики стало источником новых возможностей для реализации более эффективного управления. Нейронные сети представляют собой обучаемые динамические системы, оценивающие характеристики вход-выход. Нечеткие системы преобразуют наборы структурированных данных, связанных с объектом управления, в соответствующие управляющие воздействия. И те, и другие системы кодируют выбранную информацию в виде параллельно-распределенной структуры и не являются сложными для аппаратной реализации [3]—[5]. Нейронные и нечеткие системы имеют принципиальное преимущество перед традиционными системами адаптивного и оптимального управления: для их реализации не требуется априорная математическая модель объекта управления. Используя несколько правил, описывающих поведение переменных управления, нечеткие системы могут разрабатываться по лингвистическим правилам, тесно связанным с областью искусственного интеллекта. В то же время нейронные сети требуется обучать, прежде чем их можно будет использовать для целей управления.

Имеются несколько работ, где системы управления на основе нейронных сетей сравниваются с нечеткими и адаптивными

системами управления [6],[7]. Однако эти исследования полностью основываются на данных моделирования. В данной работе авторами выполнено сравнение характеристик многослойного нейросетевого контроллера (НСК) с контроллером на основе нечеткой логики (НЛК), с обобщенным прогнозирующим контроллером (ОПК) и с обычным ПИ-контроллером (ПИК). Сравнение проводилось для системы регулирования температуры в водяной ванне.

Зачем потребовалось сравнивать разные алгоритмы управления? Они используют разные подходы к управлению. Поэтому, применяя каждый из алгоритмов для решения одной и той же задачи управления, можно понять общие и отличительные черты этих подходов. Сравнивая результаты разных подходов к управлению в идентичных условиях, можно оценить их достоинства и недостатки. Кроме того, нейросетевой подход к управлению находится еще на начальной стадии своего развития в системах управления, работающих в режиме реального времени. Он пока применяется значительно реже, чем более развитые методы, как традиционные, так и новые. Поэтому сравнение его с другими подходами представляет несомненный интерес.

Как выполнялось сравнение? Практически невозможно сравнивать линейные контроллеры (например, ОПК и ПИК) с нелинейными контроллерами (НСК, НЛК) на сложных, слабо структуризованных задачах управления. Поэтому данное исследование проводилось в основном на примере достаточно простой системы управления водяной ванной. Прямое сравнение четырех использованных алгоритмов позволяет проанализировать каждый из рассматриваемых подходов и их характеристики. Для обеспечения точности сравнения эксперименты проводились при одинаковых возмущающих воздействиях и изменениях в объекте управления.

Для каждой системы управления проведено четыре группы экспериментов: регулировка уставки, воздействие неизвестных возмущений по нагрузке, существенные изменения параметров, переменная временная задержка. Для реализации каждого из

алгоритмов требовалась локальная настройка и коррекция объекта управления, выполнявшаяся на основе эвристических соображений. Хотя по результатам этих экспериментов сложно сделать строгие выводы о достоинствах и недостатках каждой из систем, можно надеяться, что данное исследование окажется полезным для будущей разработки эффективной интеллектуальной схемы управления. На наш взгляд, исследования должны быть направлены на выявление положительных сторон каждого класса систем управления.

В нейронной схеме управления сначала выполнялось автономное обучение многослойных нейронных сетей на основе алгоритма обратного распространения с целью изучения моделей инверсной и прямой динамики объекта управления. Затем нейронные сети настраивались в соответствии с архитектурой, показанной на рис. 4.2.5. В такой схеме может выполняться непрерывное оперативное обучение, направленное на дальнейшее улучшение характеристик. Данная архитектура применялась разными исследователями независимо друг от друга [8]—[10]. Обучение НСК начиналось со случайного набора весов связей. Однако для рассматриваемого процесса, медленно изменяющегося во времени, данный подход нецелесообразен. Следует первоначально выполнять автономное обучение как для инверсной, гак и для прямой нейросетевой модели.

Система управления с использованием НЛК была построена на основе простых соображений, характеризующих управление температурой процесса только с позиции здравого смысла. В последнее время были предложены адаптивные нечеткие системы управления, в которых за счет некоторого усложнения системы была обеспечена возможность автоматической композиции и декомпозиции правил [7], [11]—[13]. Однако авторы решили использовать обычный подход [27]. Обобщенная прогнозирующая система управления реализована на основе алгоритма управления с долгосрочным прогнозированием [14], [15]. Он уже обсуждался в разделе 3.6. Так как ПИ- и ПИД-управление все еще остаются наиболее распространенны-

ми методами в промышленности, данная традиционная схема также сравнивалась с тремя вышеуказанными схемами управления. Сравнение производилось на одной и той же системе с одним входом и одним выходом и в идентичных условиях.

Структура данного раздела следующая. Сначала приводится краткое описание системы управления температурой водяной ванны. Затем кратко рассматривается реализация управления на основе нейронных сетей, нечеткой логики и обобщенного прогнозирующего управления для этой системы. После этого приводится сравнение и обсуждение экспериментальных данных.

5.2.2. Система управления температурой водяной ванны

Система управления представляет собой регулятор температуры для лабораторной водяной ванны. Регулятор разработан фирмой Yamato Science Inc. (модель BT-15). Схема экспериментальной установки приведена на рис. 5.2.1. Система состоит из пяти основных частей: (i) водяная ванна; (ii) модуль датчика; (iii) программируемая плата интерфейса ввода-вывода (ПВВ); (iv) микроЭВМ; (v) исполнительное устройство. Ниже приводится краткое описание этих элементов. Емкость водя-

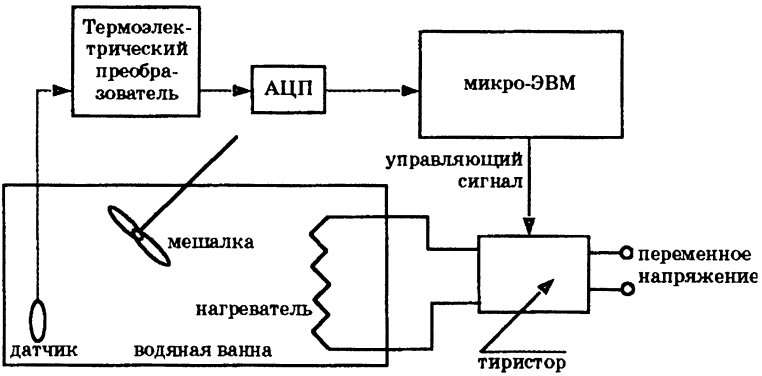


Рис. 5.2.1. Схема системы управления температурой водяной ванны

ной ванны — 8 литров, размеры — 250×290×100 мм. Нагрев ванны осуществляется нагревателем мощностью 600 Вт, подключенным к тиристорной схеме (SI6G12S-12). Для обеспечения равномерного распределения температуры использована мешалка со скоростью вращения 120 об/мин. Модуль датчика построен на основе диодов (1S1588) и усилителей с высоким коэффициентом усиления (A741). Модуль представляет собой двухкаскадную усилительную схему. Он может преобразовывать измеряемую температуру в диапазоне от 0° до 100°C в напряжение от 0 до 10 В. Его разрешающая способность — 0,24°C.

Плата ПВВ состоит из аналого-цифрового преобразователя (АЦП), цифроаналогового преобразователя (ЦАП) и программируемого периферийного интерфейса (PD8255A). Для управления ЦАП и АЦП имеется внешний таймер. Схема таймера построена на кварцевом генераторе и JK-триггерах. В эксперименте использовалась микроЭВМ NEC PC 9801F с 16-разрядным центральным процессором Intel 8086; тактовая частота — 10 МГц. Для передачи управляющих сигналов на исполнительное устройство через ЦАП и для измерения температуры использовалась несложная управляющая программа, разработанная в среде Microsoft C.

В качестве исполнительного устройства для нагревателя использовался тиристор. Его включение/выключение производилось по правилам:

- если $u(tT) \le 0{,}0$, то $V_i = 0{,}0$ В, тиристор выключен;
- если $u(tT) \ge 5{,}0$, то $V_i = 5{,}0$ В, тиристор включен;
- если $0{,}0 < u(tT) < 5{,}0$, то $V_i = u(tT)$ В, тиристор включен.

Здесь $u(tT)$ — выходной сигнал контроллера, T — период дискретизации, t — номер отсчета (0,1,2,3...), V_i — напряжение на входе исполнительного устройства.

5.2.3. Схема нейронного управления

Способность нейронных сетей к обучению и обобщению оказалась интересной для многих приложений. Одним из простейших подходов к реализации нейронного управления стал ин-

версно-прямой подход [16], [17]. В соответствии со схемой обобщенного обучения, рассмотренной в разделе 4.2.1, многослойная нейронная сеть сначала обучается автономно по алгоритму обратного распространения [18] для построения инверсной динамической модели объекта управления.

В исследовании [19], выполненном авторами данной работы, установлено, что количество задержанных сигналов от объекта управления, используемых на этапе обучения в качестве элементов входного вектора нейросетевой модели, влияет на процесс изучения контроллером инверсной модели объекта управления. Это, в свою очередь, влияет на характеристики нейроконтроллера в оперативном режиме. Теоретически эти процессы можно описать следующим образом.

Рассмотрим линейный динамический объект управления с одним входом и одним выходом, без помех. Предположим, что он описывается дискретным уравнением

$$y(t) + a_1 y(t-1) + \ldots + a_q y(t-q) = \\ = z^{-d}[b_0 u(t) + b_1 u(t-1) + \ldots + b_p u(t-p)], \qquad (5.2.1)$$

где y и u — выходной и входной сигналы объекта управления соответственно. Здесь a_i, i=1,2,...,q — коэффициенты знаменателя, b_i, i=1,2,...,p — коэффициенты числителя, d — время запаздывания (или временная задержка), t — номер отсчета. Модель нейронной сети для инверсной динамической модели объекта управления представляет собой нелинейную функцию g, содержащую задержанные входные и выходные сигналы объекта управления в соответствии с оценкой порядка и временной задержкой объекта управления:

$$I(t) = g\{u(t-d)\} = \\ = \{y(t), y(t-1), \ldots, y(t-q), u(t-d-1), \ldots, u(t-d-p)\}, \qquad (5.2.2)$$

где I(t) — выходной сигнал нейронной сети, представляющий собой управляющий входной сигнал для объекта управления. При оперативной реализации системы нейронного управления действительный выходной сигнал объекта управления y(t) за-

меняется желаемым выходным сигналом r(t) в качестве одного из элементов входного вектора нейронной сети.

На данной основе разработаны нейросетевые модели объекта управления (инверсная и прямая); при этом использовалась математическая модель объекта управления пониженного порядка [20]. Хотя нейронные сети могут обучаться модели объекта управления, не используя предварительное математическое моделирование [8]—[10], тем не менее, было бы неправильно при разработке контроллеров игнорировать информацию об объекте управления, если такая информация есть. Более того, на основе такой информации можно избежать ненужных операций, выполняемых методом проб и ошибок [19].

Обучение инверсной нейросетевой модели производилось с использованием выборки обучающих образцов, взятых из данных о реакции разомкнутой системы управления. В некоторых разработках [8]—[10], [21] обучающие образцы выбирались путем «зондирования» объекта управления случайными сигналами. В реальном времени такой метод нецелесообразен, так как изменения температуры происходят медленно. В связи с тем, что разомкнутая система является устойчивой, вместо случайных сигналов выполнялась подача плавно нарастающего сигнала непосредственно на исполнительное устройство (в пределах ограничений, установленных для входа этого устройства). Из данных о реакции системы был выбран набор обучающих образцов «вход-выход», как показано на рис. 5.2.2.

Было обнаружено, что нейросетевая модель, имеющая 8 скрытых нейронов, может изучать инверсную модель объекта управления. При этом используется два элемента входного вектора: текущего выхода объекта управления y(t), и задержанного выхода y(t-1), полученных по математической модели объекта управления пониженного порядка. Прямая нейросетевая модель разрабатывалась аналогично, однако в этом случае входные обучающие образцы представляли собой выборку из входных сигналов объекта управления и задержанных выходных величин, а выходные образцы — из соответствующих

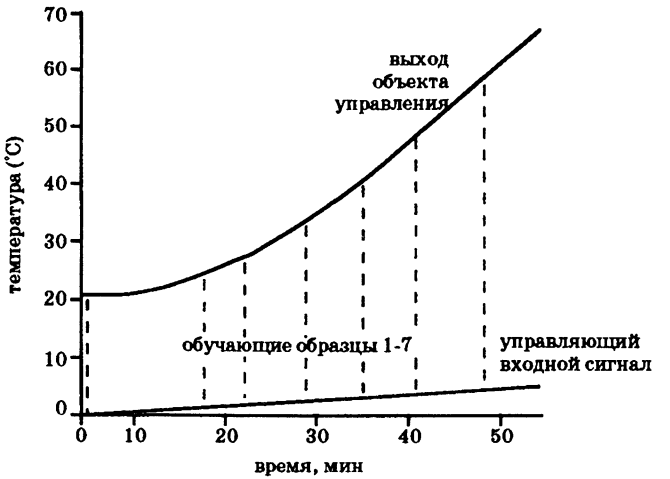


Рис. 5.2.2. Характеристика «вход-выход» разомкнутой системы управления температурой водяной ванны, полученная при подаче плавно нарастающего сигнала непосредственно на исполнительное устройство. Из этой характеристики получена выборка из семи обучающих образцов «вход-выход».

действительных значений выхода объекта управления. Обе модели настраивались, как показано на рис. 4.2.5; такая конфигурация допускает последующее оперативное обучение для дальнейшего улучшения характеристик моделей.

Для схемы обобщенного автономного обучения сигнал обратного распространения между выходным и скрытым слоями можно выразить в виде

$$\delta_k = \tau_k - O_k , \qquad (5.2.3)$$

где τ_k — заданный (целевой) образец, O_k — действительный выход нейронной сети. Сигнал между скрытым и входным слоями выражается следующим образом:

$$\delta_j = f'(net_j)\sum_k \delta_k w_{kj}. \qquad (5.2.4)$$

Здесь $f'(net_j)$ — производная от функции $f(net_j)$, где

$$f'(net_j) = 1/(1 + \exp(-net_j)) .$$

Веса связей между входным и скрытым слоями корректируются в соответствии с выражением

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t) + \beta \Delta w_{ji}(t-1), \qquad (5.2.5)$$

а между скрытым и выходным слоями — по формуле:

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t) + \beta \Delta w_{kj}(t-1), \qquad (5.2.6)$$

где η — скорость обучения, α и β — коэффициент мгновенного значения и ускорения соответственно. Уравнения (5.2.5) и (5.2.6) представляют собой модифицированные формы обобщенного дельта-правила [22].

Для схемы оперативного обучения веса связей нейроконтроллера корректируются на основе ошибки, определяемой как

$$E^E = \frac{1}{2}[r-y]^2, \qquad (5.2.7)$$

где r и y — заданный и фактический выход объекта управления соответственно. Эта ошибка распространяется через эмулятор. Таким образом, сигнал на выходе контроллера можно выразить следующим образом:

$$\delta_k^C = -\frac{\partial E^E}{\partial net_k^C} = -\frac{\partial E^E}{\partial O_k^C}\frac{\partial O_k^C}{\partial net_k^C}, \qquad (5.2.8)$$

$$\delta_k^C = \sum_j \delta_j^E w_{ji}^E, \qquad (5.2.9)$$

где O_k и net_k^C — соответственно входной и выходной сигналы нейронов выходного слоя контроллера. Верхний индекс E обозначает эмулятор, а C — контроллер. Сигнал ошибки между скрытым и входным слоями контроллера выражается следующей формулой:

$$\delta_j^C = \left(\sum_j \delta_k^C w_{kj}^C\right) O_j^C (1 - O_j^C), \qquad (5.2.10)$$

где O_j^C — выходной сигнал нейронов скрытого слоя контроллера.

В данной схеме веса связей нейронного эмулятора не сохра-

няются постоянными, а корректируются в оперативном режиме. Это делается путем обратного распространения ошибки E через эмулятор при каждом отсчете времени. Здесь ошибка E определяется в виде

$$E = \frac{1}{2}[y - \hat{y}]^2, \qquad (5.2.11)$$

где y и $\hat{y}$ — выходы действительного объекта управления и эмулятора, соответственно.

Обратное распространение сигналов ошибки для эмулятора аналогично автономному обучению при использовании только одной сети. Схему оперативного обучения для нейронного управления можно представить в виде следующего алгоритма.

Шаг 1. Прочитать выход процесса y и сравнить его с желаемым опорным значением r.

Шаг 2. Вычислить величину сигнала ошибки δ_k^E между выходным и скрытым слоями нейроэмулятора.

Шаг 3. Вычислить величину сигнала ошибки δ_j^E между скрытым и входным слоями нейроэмулятора.

Шаг 4. Вычислить величину сигнала ошибки $_k$ между выходным и скрытым слоями нейроконтроллера.

Шаг 5. Вычислить величину сигнала ошибки δ_j^C между скрытым и входным слоями нейроконтроллера.

Шаг 6. Скорректировать веса связей для нейроконтроллера (между выходным и скрытым и между скрытым и входным слоями).

Шаг 7. Сравнить выход нейроэмулятора $\hat{y}$ и фактический выход объекта управления y.

Шаг 8. Скорректировать веса связей для нейроэмулятора (между выходным и скрытым и между скрытым и входным слоями).

5.2.4. Схема управления на основе нечеткой логики

Схема управления на основе нечеткой логики (нечеткие системы управления) разрабатывались в соответствии с лингвистическими линиями, тесно связанными с областью искусственного интеллекта. Концепция нечеткой логики, разработанная на основе теории нечетких множеств Заде [23], [24], успешно применялась для управления технологическими процессами, в особенности плохо определенными, а также процессами, допускающими успешное регулирование человеком- оператором [25], [26].

На рис. 3.7.2 приведена схема несложной нечеткой системы управления [27]. Эта система представляет собой набор лингвистических условных операторов, или нечетких ассоциаций, определяющих конкретные ситуации управления. Во многих практических приложениях достаточно установить связь между ошибкой и изменением в ошибке процесса, чтобы осуществить такое изменение управляющего входного сигнала, которое обеспечит удовлетворительное управление системой. Эти простые лингвистические правила можно сформулировать на основе наблюдений или несложных исследований процесса. Базовые концепции управления на основе нечеткой логики рассмотрены в разделе 3.7.

Для системы управления водяной ванной указываются следующие входные переменные: ошибка e(t), представляющая собой рассогласование между желаемой и фактической температурой воды, и скорость изменения ошибки c(t). Выходом, или управляющей нечеткой переменной, является сигнал (напряжение), подаваемый на нагреватель, u(t); диапазон напряжения от 0 до 5В.

В ходе исследования, выполненного авторами, три нечетких переменных были представлены (квантованы) в виде семи нечетких подмножеств: от отрицательного высокого значения (NL) до положительного высокого значения(PL). На практике такое квантование можно расширить в зависимости от сложности задачи.

Для большей ясности нечеткие ассоциации (или правила) можно представить в матричной форме, как показано на рис. 5.2.3. Матрица нечетких ассоциаций для системы управления водяной ванной была разработана на основе здравого смысла и общеинженерных суждений; всего было сформулировано 25 правил.

Каждая группа элементов в матрице задает одну нечеткую ассоциацию, или i-е правило, указывающее, как следует изменять переменную управления u_i для наблюдаемых величин входных нечетких переменных e_i и c_i. В качестве примера приведем интерпретацию правила 7 (PL, ZE, PL) на естественном языке:

Если ошибка (рассогласование) температуры воды положительная и большая и скорость изменения ошибки близка к

ошибка, e

изменение ошибки, Δe = c	NL	NM	NS	ZE	PS	PM	PL
PL				4 PL	3 PL	2 PL	1 PL
PM				10 PM	9 PM	8 PM	5 PL
PS			17 PS	14 PS	13 PS	11 PM	6 PL
ZE	23 NL	21 NM	18 NS	16 ZE	15 PS	12 PM	7 PL
NS			25 NS	20 NS	19 NS		
NM				22 NM			
NL				24 NL			

Рис. 5.2.3. Матрица набора нечетких правил для системы управления водяной ванной

нулю, то нагреватель следует настроить на положительную высокую температуру.

Следует обратить также внимание на то, что заполнение всех элементов матрицы не обязательно. Вообще говоря, некоторые правила могут опускаться или, наоборот, добавляться в зависимости от сложности задачи управления.

Как видно на рис. 5.2.4, для нечетких подмножеств трех нечетких переменных были выбраны треугольные и трапецеидальные функции принадлежности. Видно, что нечеткое под-

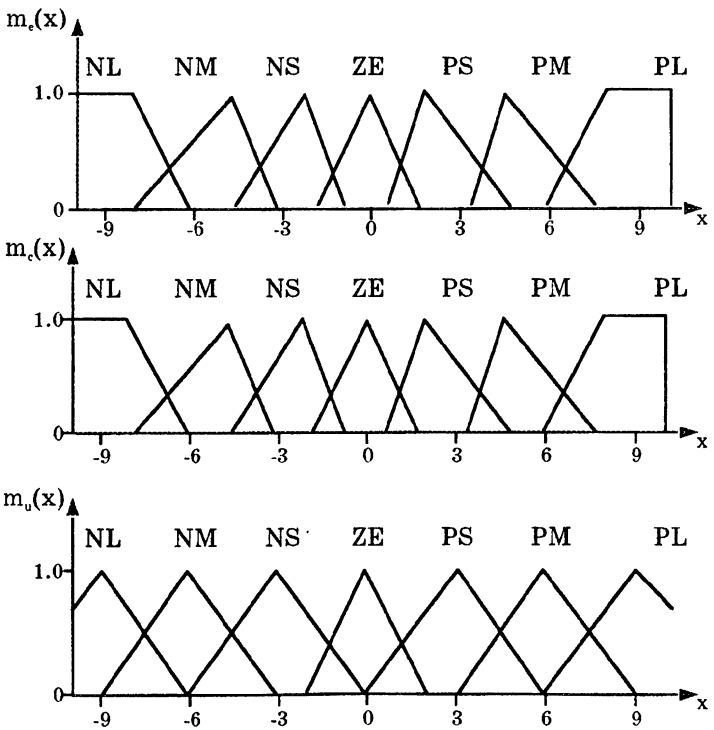


Рис. 5.2.4. Функции принадлежности для нечетких переменных системы управления водяной ванной

множество трех нечетких переменных ZE (значения, близкие к нулевым значениям) более узкие, чем другие; это позволяет повысить точность управления вблизи уставки. Исходя из эвристических соображений, принято, что непрерывные нечеткие подмножества в каждом из наборов перекрываются примерно на 25%. При слишком большом перекрытии теряются различия между величинами, соответствующими разным подмножествам. При слишком же малом перекрытии возникает тенденция к «двузначному» управлению, что приводит к перерегулированию. В реальных условиях перекрытия позволяют сглаживать переход управляющих воздействий друг в друга в ходе работы контроллера.

Посылки правил объединяются операцией И, что соответствует активации нечеткого множества-заключения u_i со степенью w_i, вычисляемой в соответствии с определением нечеткой операции пересечения [23]:

$$w_i = m_{e_i}(e) \wedge m_{c_i}(c), \qquad (5.2.12)$$

где m_{e_i} и m_{c_i} — соответствующие степени принадлежности для ошибки и изменения ошибки.

Форма выходных нечетких множеств зависит от используемой кодирующей схемы частотно-амплитудной модуляции. В данном исследовании использовалась схема кодирования по минимуму корреляции, в которой нечеткое множество-заключение u_i в наборе величин выходных нечетких множеств срезается до уровня w_i и принимает минимальное значение в других точках:

$$m_{o_i}(y) = v_i \wedge m_{u_i}(y). \qquad (5.2.13)$$

Здесь $_{o_i}(y)$ — степень принадлежности активизированного выходного нечеткого множества на интервале значений выходного универсального множества.

На практике величина $w_{o_i}(y)$ не является однозначной; поэтому система комбинирует все эти функции в действительную выходную функцию принадлежности на основе определения нечеткой операции объединения [23]:

$$m_O(y) = w_{O_1}(y) \vee m_{O_1}(y) \vee \ldots \vee m_{O_1}(y). \qquad (5.2.14)$$

Для преобразования нечеткого управляющего выходного сигнала в четкую форму применяется дискретная процедура преобразования нечеткого центроида:

$$u = \frac{\sum_{j=1}^{p} y_i m_O(y_i)}{\sum_{j=1}^{p} m_O(y_i)}, \qquad (5.2.15)$$

где выходное универсальное множество преобразуется в дискретные величины p с шагом Δy:

$$y_p = y_{p-1} + \Delta y.$$

Как и любой другой контроллер, НЛК имеет ряд параметров, которые должны быть выбраны разработчиком заранее. В нечетком контроллере имеется три коэффициента масштабирования: GE, GC и GU — для ошибки процесса, изменения ошибки и выходного сигнала контроллера соответственно. Выбор этих коэффициентов аналогичен выбору параметров ПИД-контроллера или задаваемых пользователем полиномиальных коэффициентов в некоторых адаптивных контроллерах. Некоторыми исследователями предложены систематические подходы к выбору данных коэффициентов [12], [29], [30].

Вычисления, соответствующие действиям нечеткого контроллера в системе управления температурой водяной ванны, можно представить в виде следующего алгоритма.

Шаг 1. Прочитать выходную величину процесса y(t).

Шаг 2. Вычислить ошибку e(t) и скорость изменения ошибки c(t) для процесса в момент времени t.

Шаг 3. Величины e(t) и c(t) квантуются в соответствующие нечеткие подмножества.

Шаг 4. На основе функций принадлежности вычисляются степени принадлежности ошибки и скорости изменения ошибки m_{e_i} и m_{c_i} соответственно.

Шаг 5. Из базы правил активизируется заключение со степенью принадлежности m_{u_i}, вычисляемой по степеням принадлежности посылок m_{e_i} и m_{c_i} с помощью операции И.

Шаг 6. Величина m_{u_i} срезается до величины m_{O_i} методом кодирования по минимуму корреляции.

Шаг 7. Вычисляется действительная выходная степень принадлежности m_0 путем выполнения операции ИЛИ между значениями m_{O_i} всех активизированных заключений.

Шаг 8. Вычисляется действительная выходная величина u путем применения процедуры перевода дискретного нечеткого центроида в четкую форму.

5.2.5. Схема обобщенного прогнозирующего управления

Действия, выполняемые ОПК в системе управления температурой водяной ванны, можно представить следующим образом.

Шаг 1. Прочитать выходную величину процесса y(t).

Шаг 2. Получить оценки процесса по рекурсивному методу наименьших квадратов, используя данные и векторы параметров из выражений (3.6.16) и (3.6.17).

Шаг 3. По оценкам параметров вычислить ступенчатую реакцию процесса для получения матрицы G, используя следующую формулу:

$$g_j = -\sum_{i=1}^{j} a_i g_{j-1} + \sum_{i=0}^{j-1} b_i, \quad j = 1, 2, \ldots, N2. \qquad (5.2.16)$$

Шаг 4. На основе предварительно заданной уставки сформировать вектор r таким образом, чтобы текущая выходная величина y(t) равномерно приближалась к уставке. Вектор r можно получить из следующих выражений:

$$r(t) = y(t), \qquad (5.2.17)$$

$$r(t+j) = \alpha r(t+j-1) + (1-\alpha) r(t+1). \qquad (5.2.18)$$

Шаг 5. Из данных, полученных в момент t, вычисляется вектор f по следующей формуле:

$$f(t+j) = F_j y(t) + \\ + z^{j-1}\left[G_j(z^{-1}) - g_0 - g_1 z^{-1} - \ldots - g_{j-1} z^{-(j-1)}\right]\Delta u(t). \qquad (5.2.19)$$

Шаг 6. По формуле (3.6.15) вычисляется управляющий входной сигнал. К процессу применяется только первый элемент вектора U.

Шаг 7. Выполняется сдвиг данных для следующего момента дискретизации.

5.2.6. Экспериментальные данные и их анализ

Для обычной системы управления температурой водяной ванны с обратной связью использовался следующий ускоренный дискретный пропорционально-интегральный алгоритм регулирования:

$$\Delta u(t) = k_c[e(t) - e(t-1)] + \frac{T}{2T_i}[e(t) - e(t-1)], \qquad (5.2.20)$$

где $\Delta u(t)$ — инкремент входного управляющего воздействия, $e(t)$ — ошибка, k_c — коэффициент передачи контроллера, или коэффициент пропорциональности, T_i — время интегрирования, или сброса (см. раздел 3.2). Для настройки ПИ-контроллера использовался метод Такахаши и др. [31].

Для каждого алгоритма было проведено четыре группы экспериментов с системой управления температурой водяной ванны. Каждый эксперимент проводился для 100 дискретных моментов времени с периодом 30 с; таким образом, общая длительность составляла 50 мин. С целью получения лучших результатов в каждом эксперименте для каждой схемы применялись эвристические правила коррекции и точная настройка. Соответствующие методы и лучшие результаты экспериментов для каж-

дого алгоритма сравниваются и представляются в табличной форме таким образом, чтобы их характеристики были сравнимыми.

В первой серии экспериментов исследовались характеристики четырех контроллеров по отслеживанию при изменении уставки. Для ОПК лучшие характеристики были достигнуты при следующих параметрах: NU=1, N1=1, N2 = 3, $\lambda = 0{,}1$. Для оценки параметров системы во всех экспериментах с использованием ОПК применялся рекурсивный метод наименьших квадратов без коэффициента забывания и с диагональной ковариационной матрицей {1,0,0}. Лучшие характеристики ПИК были достигнуты при следующей точной настройке параметров: $k_c = 2{,}3$ с., $T_i = 80$ с. Для НЛК после нескольких проб и ошибок лучшие характеристики были получены при следующих коэффициентах масштабирования: GE=0,16, GC=0,83, GU=4,20. Характеристики НЛК ухудшались при неправильном выборе заключений в базе правил. Для НСК не задавалось никаких изменений во время оперативного обучения; использовались те же параметры модели и процесса обучения, что и при автономном обучении. Кроме того, было замечено, что характеристики НСК остаются такими же хорошими, даже если оперативное обучение не используется.

Все четыре контроллера показали очень хорошие характеристики в отслеживании трех заданных уставок; результаты экспериментов показаны на рис. 5.2.5, *а*—*d*. Как и ожидалось, характеристики ПИК оказались хорошими для одной рабочей точки (в данном случае для верхней уставки), но ухудшились для другой (для нижней уставки). Лучшие характеристики по отслеживанию продемонстрировал НСК: после того как этот контроллер был обучен, его можно было применять непосредственно, не выбирая каких-либо параметров конструкции или настройки.

Вторая серия экспериментов проводилась с целью изучения возможностей контроллеров в условиях нежелательных возмущений по нагрузке. С целью получения сравнимых резуль-

татов для всех контроллеров использовались импульсные возмущающие воздействия величиной 4,0 °С и 2,0 °С на 53-м и 75-м отсчете соответственно. Во всех экспериментах применялась уставка 50 °С.

Для всех контроллеров использовались те же параметры, что и в первой серии экспериментов. Характеристики контроллеров в условиях возмущений показаны на рис. 5.2.6,*a*—*d*. Для большей ясности характеристики показаны только в соответствующие моменты дискретного времени. Можно видеть, что характеристики НСК и ОПК очень хорошие: они быстро восстанавливают нормальную работу после возмущающего воздействия.

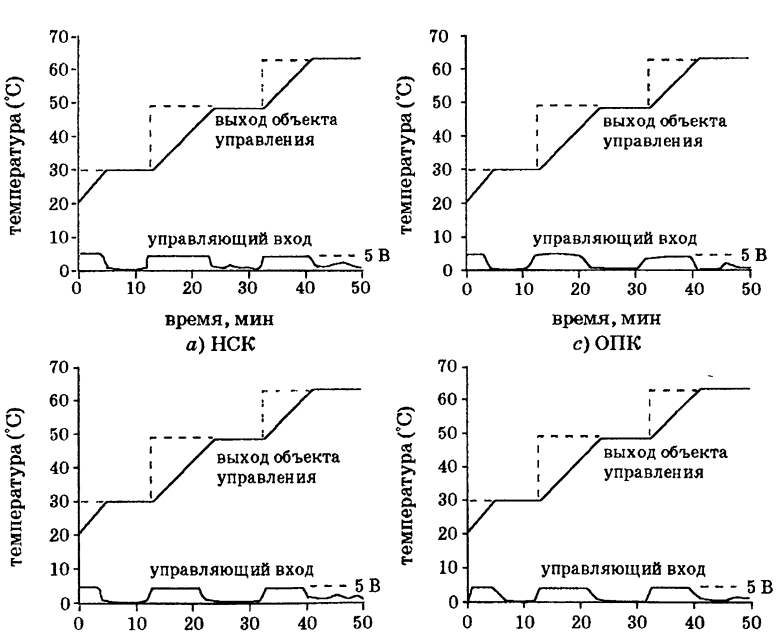


Рис. 5.2.5. Характеристики отслеживания уставки для четырех типов контроллеров в системе управления водяной ванной:
a) НСК; *b*) НЛК; *c*) ОПК; *d*) ПИК

Однако характеристики системы резко ухудшались при использовании НЛК и ПИК. Характеристики НЛК ухудшались после первого возмущающего воздействия в той же степени, что и для ПИК, если это воздействие было резким и непредсказуемым. В этом отношении нечеткая система с адаптивными правилами более приемлема, нежели система с фиксированными нечеткими управляющими правилами, так как в них нельзя учесть непредсказуемые возмущения.

Способность НСК к быстрому восстановлению обеспечивалась возможностями обобщения, которые позволяют ему быстро адаптироваться к неблагоприятным изменениям на входе. ОПК показал такие же хорошие характеристики, как если бы

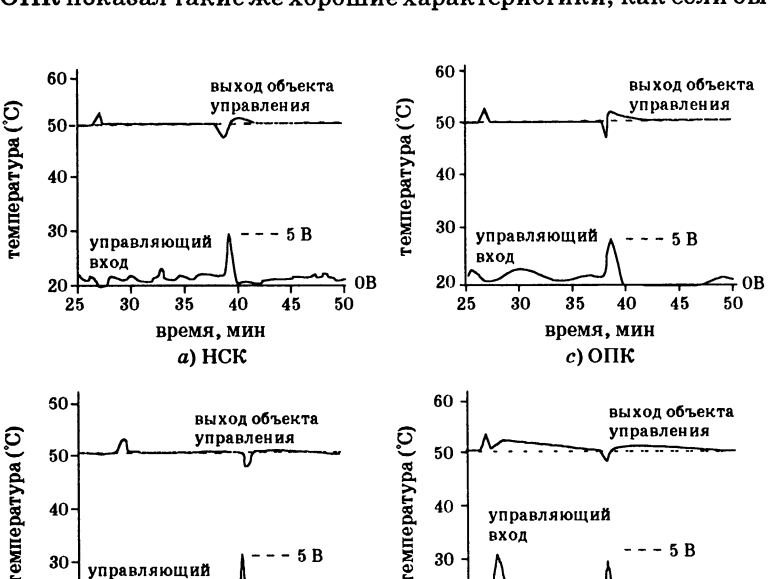


Рис. 5.2.6. Характеристики четырех типов контроллеров в условиях непредсказуемых возмущений на нагрузке:
a) НСК; *b*) НЛК; *c*) ОПК; *d*) ПИК

он был оснащен прогнозирующим устройством на j шагов вперед. Среди всех испытывавшихся контроллеров худшие характеристики показал ПИК, так как он действует сравнительно медленно и не может выполнять быструю автоматическую самонастройку в условиях непредсказуемых возмущений на нагрузке.

Одной из характерных особенностей многих систем управления процессами оказалась тенденция к непредсказуемости изменения их параметров. Проводились испытания контроллеров на устойчивость в подобной ситуации. В каждом из экспериментов после 60-го отсчета дискретного времени к выходной величине процесса была добавлена величина 0,8u(t-2). Таким образом, имитировалось изменение параметров процесса. Наблюдавшиеся изменения в динамике оказались существенными и, хотя трудно представить реальный объект, поведение которого изменялось бы столь резко, тем не менее, были проведены эксперименты для сравнения устойчивости и адаптивных свойств различных алгоритмов управления. Характеристики четырех контроллеров в условиях таких изменений показаны на рис. 5.2.7.

Как и ожидалось, среди четырех исследовавшихся контроллеров худшие характеристики оказались у ПИК. Несмотря на интегрирующее действие, на выходе объекта управления наблюдалось отклонение, возникавшее из-за стойкого насыщения управления. В то же время интегрирующее действие помогало, в конечном счете, устранить это отклонение. Три других контроллера продемонстрировали достаточную устойчивость: они могли отслеживать уставку даже при значительных изменениях динамики объекта управления. НСК оказался очень хорошим в отслеживании уставки: его способность к обобщению и оперативное обучение позволяли улучшить характеристики этого контроллера. Возможности долгосрочного прогнозирования ОПК позволяли стабилизировать характеристики этого контроллера в условиях, когда параметры объекта управления изменяются. Лучшие характеристики из всех четы-

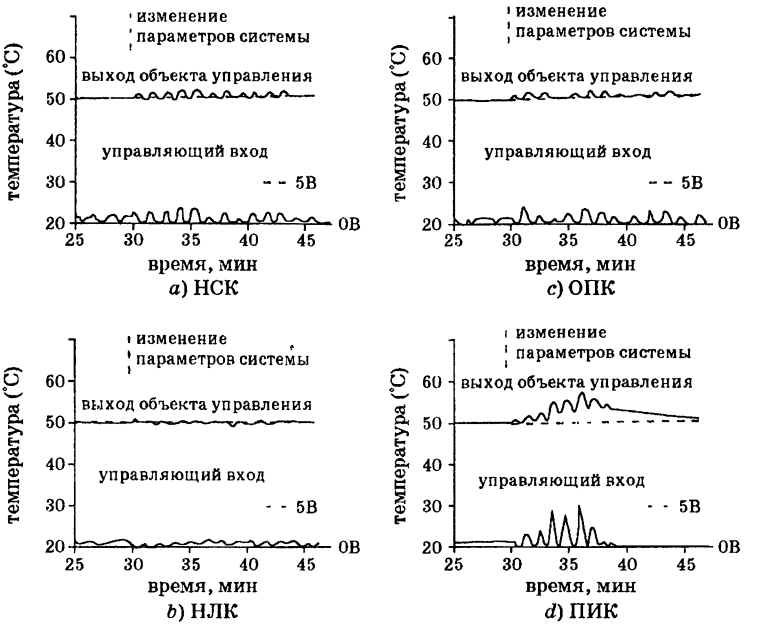


Рис. 5.2.7. Характеристики четырех типов контроллеров в условиях существенных изменений динамики объекта управления:
а) НСК; *b*) НЛК; *с*) ОПК; *d*) ПИК

рех контроллеров продемонстрировал НЛК. В отличие от возмущений на нагрузке, изменения в динамике объекта в этом случае не были столь существенными. НЛК может преобразовывать информацию об ошибке процесса и об изменении ошибки в соответствующие управляющие воздействия, обеспечивающие малые флуктуации на выходе.

Среди проблем управления производственными процессами большое значение имеют проблемы управления процессами с длительными переменными временными задержками. Поэтому имеет смысл сравнить различные алгоритмы управления на предмет их устойчивости в условиях работы с объектами, на которых возникают такие задержки. В четвертой серии экспериментов в цикл управления были введены дополнительные,

искусственные временные задержки на 1 отсчет дискретного времени в диапазоне $0 \le tT \le 30$ и на 2 отсчета — в диапазоне $30 < tT \le 100$. В этих экспериментах были заданы две уставки: 35 °С в диапазоне $0 \le tT \le 30$ и 55 °С — в диапазоне $30 < tT \le 100$. Заранее известной предполагалась только временная задержка в диапазоне $0 \le tT \le 30$.

Нейроконтроллер и эмулятор подвергались переобучению на основе информации «вход-выход» для диапазона $0 \le tT \le 30$. Однако структура нейронной сети и количество обучающих образцов поддерживались неизменными. В ОПК количество оцениваемых параметров полинома $B(z^{-1})$ было увеличено до четырех, а параметры системы были изменены в соответствии с временной задержкой объекта управления следующим образом: NU=2, N1=1, N2=10, λ=0,5. Аналогично выполнялась перенастройка ПИ-контроллера: величины k_c и T_i были увеличены до 3,0 и 100 с. соответственно. Коэффициенты масштабирования НЛК были изменены следующим образом: GE=0,11, GC=0,70, а величина GU осталась без изменений.

Характеристики всех четырех контроллеров на одном и том же объекте управления с переменной временной задержкой показаны на рис. 5.2.8. Как и ожидалось, лучшие характеристики в этих условиях показал ОПК, так как алгоритм его работы в наибольшей степени подходит для объектов управления с переменными временными задержками при условии, что количество оцениваемых параметров полинома $B(z^{-1})$ охватывает диапазон возможных временных задержек. При использовании НСК на второй уставке наблюдалось перерегулирование. Однако оперативное обучение позволило, в конечном счете, улучшить характеристики этого контроллера. НЛК показал себя несколько лучшим, чем НСК. Используемый в НЛК нелинейный алгоритм устранения нечеткости позволил уменьшить перерегулирование в системе. Характеристики ПИК ухудшились на второй уставке: на выходе наблюдались нежелательные колебания. Они возникли из-за того, что использовавшийся метод настройки был основан на временной задержке объек-

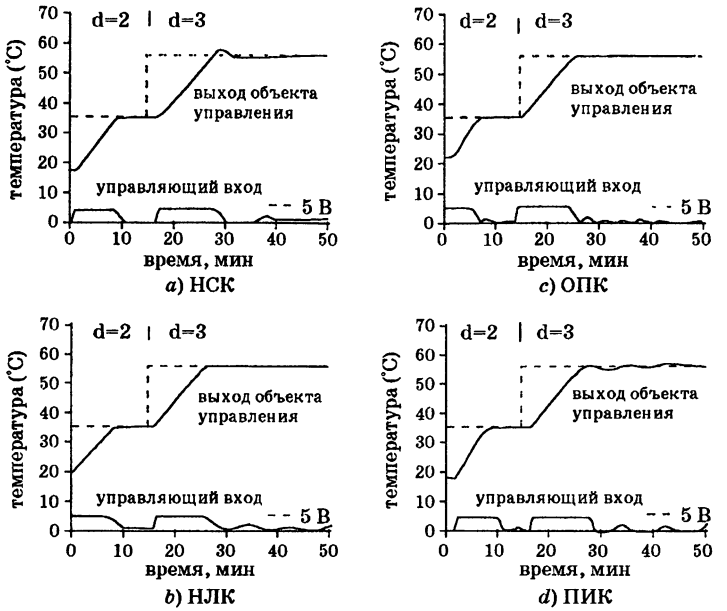


Рис. 5.2.8. Характеристики четырех типов контроллеров в условиях длительных переменных временных задержек на объекте управления: *a*) НСК; *b*) НЛК; *c*) ОПК; *d*) ПИК

та управления и на его переходной характеристике в разомкнутом состоянии.

Во всех экспериментах можно было наблюдать, что выход НСК и НЛК более «гладкий», чем для ОПК и ПИК. На выходе ОПК более четко наблюдались резкие флуктуации, что приводило к меньшим флуктуациям на выходе объекта управления. Нежелательные длительные флуктуации управляющих сигналов могли также причинить повреждения исполнительному устройству. Возможности нелинейного отображения НСК и НЛК обеспечивали более «гладкие» величины выхода контроллера, что позволяет продлить срок службы исполнительного механизма.

5.2.7. Заключение

В данном разделе приведены результаты сравнения четырех схем управления на примере системы регулирования температуры водяной ванны. В подходах, реализованных в этих четырех схемах, имелись как сходства, так и различия. По результатам экспериментов для всех схем управления, кроме ПИК, обнаружились как достоинства, так и недостатки. Из всех рассмотренных схем управления схема, основанная на ПИК, по-видимому, наиболее проста для реализации. Однако было установлено, что этот наиболее распространенный тип контроллера не может работать удовлетворительно в сложных условиях. Таким образом, данный тип контроллера можно рекомендовать только для простых задач управления и только при условии, что к системе не предъявляется высоких требований.

Так как для водяной ванны возможно построение математической модели пониженного порядка, полученная на основе этой модели информация использовалась для разработки всех четырех контроллеров. Сначала в процессе моделирования была сделана грубая настройка параметров НЛК и ПИК, и только потом выполнялась точная оперативная настройка этих величин для улучшения характеристик. Для НСК количество элементов входного вектора нейросетевой модели соответствовало пониженному порядку модели объекта управления. Однако в условиях, когда модели объектов плохо определены [27], НСК, НЛК и ПИК могут использоваться без каких-либо математических моделей. В то же время ОПК и многие традиционные адаптивные контроллеры не могут разрабатываться без априорных математических моделей.

Одним из недостатков нейронного управления является необходимость предварительного обучения. Для успешного применения нейронных сетей важен также обоснованный выбор структуры модели и параметров обучения. Однако после обучения НСК становится самонастраивающимся и не требует ка-

кой-либо настройки. В то же время для ОПК, даже если он является самонастраивающимся, выбор задаваемых пользователем параметров представляет определенную проблему, особенно для сложных объектов управления.

Для НЛК имеется ряд факторов и величин, которые требуется предварительно выбрать, чтобы обеспечить успешную работу контроллера. Вообще говоря, такой выбор является достаточно трудоемким мероприятием. Вместе с тем, некоторые из этих величин (например, ширину функции принадлежности, коэффициент масштабирования и т.д.) можно определить эвристически или методом проб и ошибок. Однако для обеспечения приемлемых характеристик НЛК важна точная настройка данных параметров. В этом отношении более предпочтительными являются нечеткие адаптивные системы [11], [12], [13], так как, кроме всего вышеуказанного, в них возможно автоматическое порождение правил. Однако эти адаптивные нечеткие системы могут оказаться сложными для разработки. Для ПИК время от времени требуется ручная настройка, если рабочая точка процесса изменяется из-за возмущений или по каким-либо иным причинам.

Еще одним преимуществом нейронных сетей и нечетких систем управления (по сравнению с традиционными адаптивными и обычными системами) является возможность их применения для решения задач управления, связанных с существенными нелинейностями. Такому применению нейронных сетей посвящено много работ [8], [9]. Использование сигмоидных функций для скрытых нейронов позволяет применять нейронные сети для решения таких задач управления, для которых традиционные схемы до настоящего времени были непригодны. Установлено также, что для этой цели могут использоваться и НЛК [6], [11], [13]. Источниками нелинейности для НЛК являются алгоритмы формирования и устранения нечеткости, правила амплитудно-частотной модуляции и логический вывод.

Алгоритмы управления, основанные на нейронных сетях и нечеткой логике, могут быть легко реализованы на аппарат-

ных средствах параллельной обработки, что обеспечивает высокое быстродействие данных алгоритмов[3], [4], [5]. В работе [4] показано, что скорость вывода в обычной СБИС- схеме, реализующей операции нечеткой логики, не зависит от количества правил. Кроме того, параллельная работа контроллеров на основе нейронных сетей более устойчива даже в случае отказа некоторых синаптических связей [6],[32]. Затраты времени на вычисления для ОПК очень велики по сравнению с другими контроллерами, так как его работа связана с очень большим объемом вычислений. Затраты времени на вычисления для НЛК минимальны, так как в этих контроллерах используются главным образом логические операции и сравнения. В табл.5.2.1

Таблица 5.2.1. Сравнительные характеристики четырех схем управления (по результатам экспериментов)

КРИТЕРИЙ	НСК	НЛК	ОПК	ПИК
Математическая модель объекта управления	Необязательна	Необязательна	Обязательна	Необязательна
Настройка параметров контроллера	Самонастройка. Никакой настройки параметров не требуется, однако требуется предварительно выбрать и обучить нейронную сеть	Самонастройка. Однако коэффициенты масштабирования, GE, GC и GU требуется выбирать предварительно	Самонастройка. Однако горизонты управления, N1, N2 и NU, а также весовые коэффициенты управления λ требуется выбирать предварительно	Требуется настройка параметров контроллера K_p, K_i и K_D (или k_c, T_i T_d)
Вычислительная сложность	Высокая	Минимальая	Максимальая	Невысокая
Характеристики слежения	Лучшие	Плохие	Средние	Наихудшие
В условиях возмущений	Лучшие	Плохие	Средние	Наихудшие
Изменение параметров объекта	Средние	Лучшие	Средние	Наихудшие
Переменные временные задержки	Плохие	Средние	Лучшие	Наихудшие
Сглаживание управляющего сигнала	Средние	Средние	Наихудшие	Плохие

приведены основные результаты сравнения характеристик различных контроллеров, полученные на основе экспериментальных данных.

Данное исследование показало, что искусственные нейронные сети могут быть легко реализованы в системах управления реального времени. Подход на основе нейронного управления обеспечивает устойчивость. Его характеристики сравнимы с результатами, которые демонстрируют наиболее развитые традиционные методы управления. Хотя нейроконтроллеры обладают хорошими характеристиками и имеют определенные преимущества перед контроллерами трех других типов, можно надеяться на разработку еще более эффективных и надежных схем управления. Этому должно способствовать выявление лучших сторон и характеристик систем различных классов.

5.3. Система стабилизации инвертированного маятника на основе нейронных сетей

5.3.1. Введение

Быстрый прогресс микроэлектронных технологий сделал возможным применение интеллектуальных методов управления. Хотя нейронное управление еще должно проявить себя, многие традиционные алгоритмы управления (например на основе нечеткой логики) хорошо разработаны и демонстрируют достаточно хорошие характеристики в реальных системах управления. В данном разделе рассматривается проблема стабилизации инвертированного маятника. Инвертированный маятник представляет собой модель нестабильной системы управления; поэтому данная проблема представляет интерес для специалистов по управлению. В данной работе предполагается, что инвертированный маятник закреплен сверху на тележке, которая может перемещаться вправо и влево в горизонтальной плоскости, причем это перемещение является управляемым. Задача управления состоит в стабилизации маятника в

вертикальном положении на возможно более продолжительное время.

Сначала для поворота маятника из нижнего положения и его стабилизации в вертикальном положении был разработан нечеткий контроллер. Для поворота и стабилизации маятника использовались два набора нечетких правил. Для поворота маятника использовался первый набор, состоявший из семи правил. Когда характеристики угла маятника (т.е. его угловая скорость и позиция тележки) оказывались в определенных диапазонах, нечеткий контроллер переключался на использование второго набора. Этот набор состоял из 11 нечетких правил, обеспечивающих стабилизацию маятника. Однако выбрать лучший набор правил, обеспечивающий хорошее управление, достаточно сложно.

Для более точного управления в диапазоне, в котором возможно применение линеаризованной модели маятника, использовался метод оптимального управления. Наконец, для регулировки рассогласования между фактической и желаемой позициями маятника применялась нейронная сеть. Характеристики линейного квадратичного оптимального контроллера могут быть улучшены за счет использования нейронной сети, обеспечивающей компенсацию нелинейных характеристик, которые обусловлены скольжением, трением и т.д. Такое улучшение может быть достигнуто путем использования нейронной сети с небольшим количеством скрытых нейронов. На основе указанных принципов управления с инвертированным маятником проводились эксперименты в реальном времени.

5.3.2. Система стабилизации инвертированного маятника

Система стабилизации инвертированного маятника состоит из трех частей: (i) стержень маятника на тележке; (ii) блок управления; (iii) персональный компьютер с АЦП и ЦАП. Структура этой системы показана на рис. 5.3.1, а блок-схема — на рис. 5.3.2. Механическая часть и блок управления

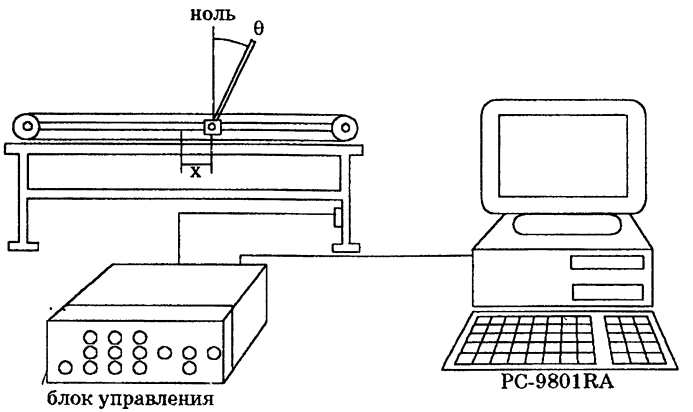


Рис. 5.3.1. Система стабилизации инвертированного маятника

Рис. 5.3.2. Блок-схема системы стабилизации инвертированного маятника

Таблица 5.3.1. Физические параметры инвертированного маятника

Параметры	Значения
Эквивалентная масса тележки	4,4 кг
Коэффициент преобразования "напряжение/момент"	20,09 Н/В
Эквивалентный коэффициент трения тележки	19,2 кг/с
Момент центра тяжести маятника	$2{,}14\times10^{-3}$ кгм^2
Коэффициент вязкого трения оси	0,115 м
Расстояние от центра тяжести маятника	$4{,}9\times10^{-4}$ кгм^2/с
Масса маятника	0,1 кг
Коэффициент преобразования "позиция/напряжение"	1,7 В/м
Коэффициент преобразования "угол/напряжение"	3,34 В/оборот

были разработаны Японской лабораторией современных методов управления (Advanced Control Laboratory of Japan). В экспериментах использовался персональный компьютер NEC PC-9801RA с математическим сопроцессором. Физические параметры инвертированного маятника приведены в табл. 5.3.1.

Выходными величинами системы являются угол маятника θ и позиция тележки x. Угол θ измеряется от вертикальной линии, а позиция x — от центра дорожки тележки, как показано на рис. 5.3.1. Входной величиной является горизонтально направленное усилие, прилагаемое к тележке.

Цель управления состоит в том, чтобы переместить тележку в позицию x=0 таким образом, чтобы маятник оставался в вертикальном положении. Это означает, что при x=0 должно соблюдаться условие $\theta = \dot{\theta} = 0$, где $\dot{}$ — производная от координаты θ по времени. Чтобы перевести маятник в вертикальное положение из первоначального (нижнего) устойчивого положения, его необходимо повернуть; для этого требуется приложить к тележке силу, смещающую ее в горизонтальном направлении. Такое действие будем называть «поворотным управлением». После этого требуется стабилизировать маятник в вертикальном положении; для этого требуется управлять движением тележки в горизонтальном направлении. Это действие будем называть «стабилизационным управлением».

5.3.3. Инициализация поворотного управления с использованием нечеткой логики

Как показано выше, сначала изучалось «поворотное управление», реализуемое с помощью набора из семи нечетких правил. Эти правила показаны в табл. 5.3.2. Чтобы повернуть маятник из устойчивого нижнего положения, необходимо использовать действие кинетической энергии. Знаки координат для движения тележки и для удерживания маятника в вертикальном положении указаны на рис. 5.3.3. Функция принадлежности, выбранная для управления маятником, показана на рис. 5.3.4.

Из устойчивого нижнего положения маятник поворачивается за счет движения тележки вправо или влево по горизонтали. Так как начальное значение величин θ и $\Delta\theta$ равно ZO, из табл. 5.3.2 видно, что управляющее воздействие для тележки равно PM. Здесь $\Delta\theta$ обозначает первую разность координаты θ, которая представляет собой аппроксимацию производной .

Применение нечетких правил обеспечивает движение тележ-

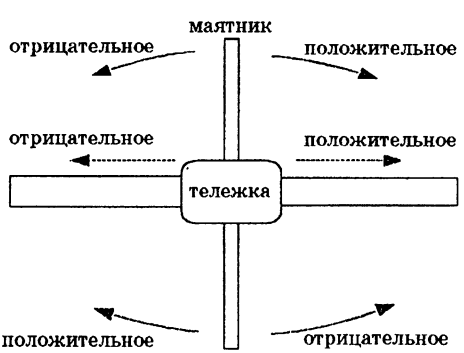


Рис. 5.3.3. Знаки направлений в системе стабилизации инвертированного маятника

ки в положительном направлении. Когда тележка движется в отрицательном направлении, потенциальная энергия маятника нарастает. Затем маятник начинает двигаться вниз, теряя потенциальную энергию и увеличивая кинетическую. Таким образом, маятник снова поворачивается вниз, и в нижнем положении величина θ равна ZO, а приращение $\Delta\theta$ равно NM.

Из табл. 5.3.2 видно, что следующее управляющее действие — NM. Повторение вышеописанных действий обеспечивает повороты маятника вверх и вниз до тех пор, пока величина q при этом увеличивается. Когда величины координаты q и ее производной достигают определенных значений из некоторого заранее заданного диапазона, происходит переход от «поворотного управления» к «стабилизационному». Диапазон смены типа управления задается следующим образом: $\theta \in [-3, 3]$, $\Delta\theta \in [-2, 2]$. Однако если координата $x \in [-5, 5]$, то тележка движется в направлении исходной точки, и действия «поворотного управления» повторяются. Вышеуказанный диапазон действия «стабилизационного управления» называется «диапазоном устойчивости».

Рассмотрим теперь «стабилизационное управление», которое используется при попадании маятника в «диапазон устойчивости». Нечеткие правила для этого вида управления показаны в табл. 5.3.3; их можно интерпретировать следующим образом.

Правило 1: Если θ=PM и $\Delta\theta$=ZO, то v=PM.
Правило 2: Если θ=PS и $\Delta\theta$=PS, то v=PS.
Правило 3: Если θ=PS и $\Delta\theta$=NS, то v=ZO.
Правило 4: Если θ=NM и $\Delta\theta$=ZO, то v=NM.
Правило 5: Если θ=NS и $\Delta\theta$=NS, то v=NS.
Правило 6: Если θ=NS и $\Delta\theta$=PS, то v=ZO.
Правило 7: Если θ=ZO и $\Delta\theta$=ZO, то v=ZO.
Правило 8: Если θ=ZO и $\Delta\theta$=NM, то v=NM.
Правило 9: Если θ=ZO и $\Delta\theta$=NS, то v=NS.
Правило 10: Если θ=ZO и $\Delta\theta$=PS, то v=PS.
Правило 11: Если θ=ZO и $\Delta\theta$=PM, то v=PM.

Таблица 5.3.2. Нечеткие правила для поворотов маятника

θ \ $\Delta\theta$	NM	NS	ZO	PS	PM
NM	ZO				
NS	NS				
ZO	NM		PM		PM
PS					PS
PM					ZO

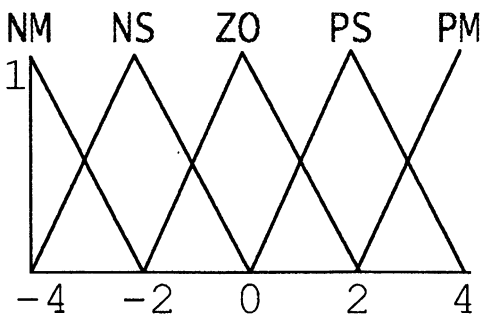


Рис. 5.3.4. Функции принадлежности для нечетких переменных системы стабилизации инвертированного маятника

Таблица 5.3.3. Нечеткие правила для стабилизации маятника

θ \ Δθ	NM	NS	ZO	PS	PM
NM			NM		
NS		NS		ZO	
ZO	NM	NS	ZO	PS	PM
PS		ZO		PS	
PM			PM		

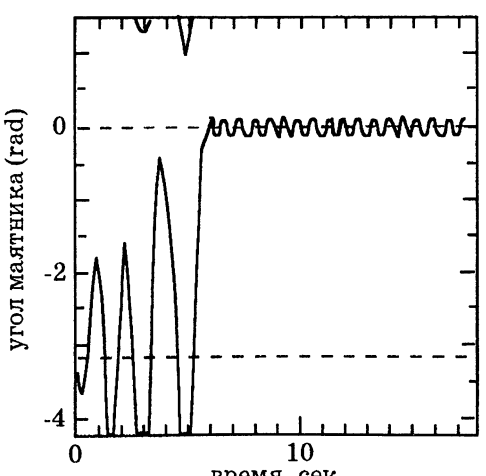


Рис. 5.3.5. Результаты эксперементов по стабилизации маятника с использованием правил, приведенных в табл. 5.3.3

Здесь через v обозначено входное напряжение двигателя тележки.

Нечеткий контроллер работает на основе минимаксного метода центра тяжести [27], который определяется формулой (5.2.15). Применение управляющего воздействия в этом случае позволяет удерживать маятник в вертикальном положении при очень быстром перемещении тележки влево или вправо. Результаты экспериментов показаны на рис. 5.3.5. На нем видно, что при использовании только нечеткого контроллера (с набором правил, заданным в табл. 5.3.3) флуктуации довольно значительны даже тогда, когда маятник стоит в вертикальном положении.

5.3.4. Применение стратегии оптимального управления для стабилизации маятника

Нечеткий контроллер применяется в основном для начальной установки инвертированного маятника. Характеристики системы можно улучшить за счет добавления нечетких правил. Однако из-за ограниченности памяти и быстродействия персонального компьютера увеличение количества нечетких правил практически невозможно для систем реального времени, когда требуется быстрая реакция. Кроме того, достаточно сложно выбрать набор правил, обеспечивающий улучшение характеристик нечеткого контроллера. Обычно для грубой настройки применяется второй набор правил, предназначенный для стабилизационного управления.

Отметим, что в «диапазоне устойчивости» нелинейные уравнения динамики, описывающие инвертированный маятник, могут быть аппроксимированы линейными уравнениями. В этом случае для стабилизации инвертированного маятника можно использовать традиционные методы управления. В данном случае применяется теория линейного квадратичного оптимального управления [35]. Рассмотрим сначала уравнения движения маятника и тележки:

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta - u + F\dot{x} = 0, \qquad (5.3.1)$$

$$(J + ml^2)\ddot{\theta} + ml\ddot{x}\cos\theta - mgl\sin\theta + C\dot{\theta} = 0, \qquad (5.3.2)$$

где M — эквивалентная масса тележки; m — масса инвертированного маятника; F — эквивалентный коэффициент трения тележки; l — расстояние от оси до центра тяжести маятника; J — момент инерции центра тяжести маятника; C — коэффициент вязкого трения оси; g — ускорение силы тяжести.

Если считать, что $\cos\theta \approx 1, 0^2\dot{\theta} \approx 0$ и $\sin q \approx 0$ и использовать приближенные равенства как точные, то нелинейная система уравнений (5.3.1), (5.3.2) становится линейной. В этом случае имеем

$$\dot{x}(t) = Ax(t) + bu(t), \qquad (5.3.3)$$

$$u(t) = cx(t), \qquad (5.3.4)$$

где

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-F(J+ml^2)}{D} & \dfrac{-m^2gl^2}{D} & \dfrac{mlC}{D} \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{mlF}{D} & \dfrac{(M+m)mgl}{D} & \dfrac{(M+m)C}{D} \end{bmatrix},$$

$$b = \begin{bmatrix} 0 & \dfrac{J+ml^2}{D} & 0 & \dfrac{l}{D} \end{bmatrix}^T,$$

$$c = \begin{bmatrix} 1000 \\ 0010 \end{bmatrix},$$

$$D = (M+m)(J+ml^2) - m^2l^2,$$

$$
\begin{aligned}
&x = \left[x_1, x_2, \theta_1, \theta_2\right]^T, \\
&x_1 = x, x_2 = \dot{x}_1, \\
&\theta_1 = \theta, \theta_2 = \dot{\theta}_1.
\end{aligned}
\tag{5.3.5}
$$

В дискретном времени данные уравнения представляются следующим образом:

$$
\begin{aligned}
&x(t+1) = Px(t) + qu(t), \\
&y(t) = cx(t), \\
&P = e^{AT}, \quad q = Q_0 b, \\
&Q_0 = \int_0^T e^{A\tau} d\tau,
\end{aligned}
\tag{5.3.6}
$$

где T — интервал дискретизации. Стратегия оптимального управления состоит в минимизации функции затрат J, задаваемой выражением

$$
J = \sum_{t=0}^{\infty} \left(y(t+1)^T y(t+1) + \Gamma u^2(t) \right), \quad \Gamma > 0. \tag{5.3.7}
$$

Оптимальный управляющий сигнал u(t) задается в этом случае выражением:

$$
\begin{aligned}
&u(t) = -Kx(t), \\
&K = \left(\Gamma + q^T H q\right)^{-1} q^T H P,
\end{aligned}
\tag{5.3.8}
$$

где H — устойчивое решение уравнения Риккати:

$$
\begin{aligned}
H(t+1) = P^T H(t) P - PH(t) q \left(\Gamma + q^T H(t) q\right)^{-1} q^T H(t) P + c^T c, \\
H(0) = \alpha I, \quad \alpha > 0.
\end{aligned}
\tag{5.3.9}
$$

Используя приведенную выше линейную аппроксимацию системы, можно применить линейное квадратичное оптимальное управление для стабилизации маятника. Так как значение x(t) неизвестно, необходимо найти его оценку по данным наблюдения y(n), $n \le t$. Авторами данного исследования разрабо-

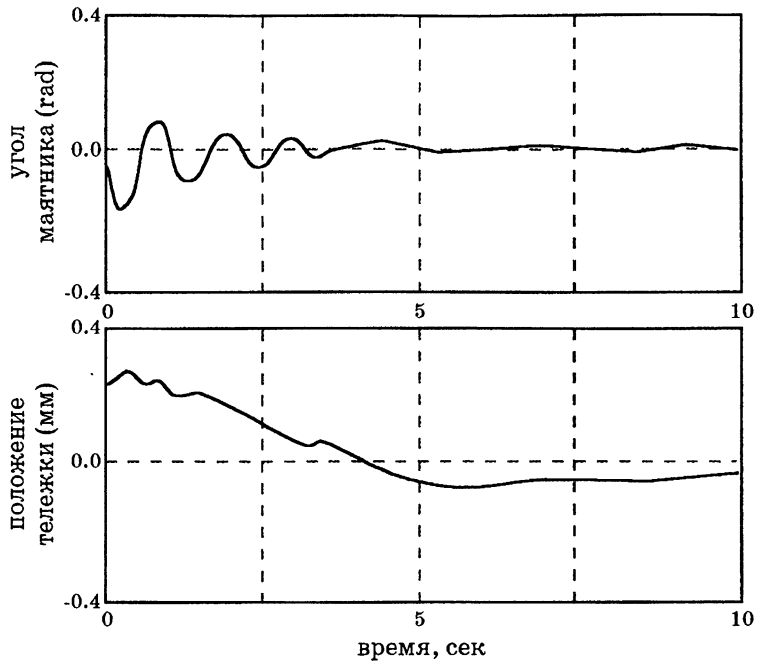


Рис. 5.3.6. Результаты эксперментов при использовании линейного оптимального управления с наблюдателем минимального порядка.

тан алгоритм наблюдения. Оценка x(t) обозначается в нем через $\hat{x}(t)$. Алгоритм наблюдения можно построить таким образом, чтобы $\hat{x}(t)$ сходилось к x(t) при больших значениях t. При этом используется следующий закон обратной связи:

$$u(t) = -K\hat{x}(t). \qquad (5.3.10)$$

Результаты экспериментов при использовании вышеописанной схемы оптимального управления показаны на рис. 5.3.6. Из них видно, что характеристики системы стабилизации маятника улучшаются.

5.3.5. Точное управление на основе нейронных сетей

В предыдущем разделе рассмотрен метод управления, пост-

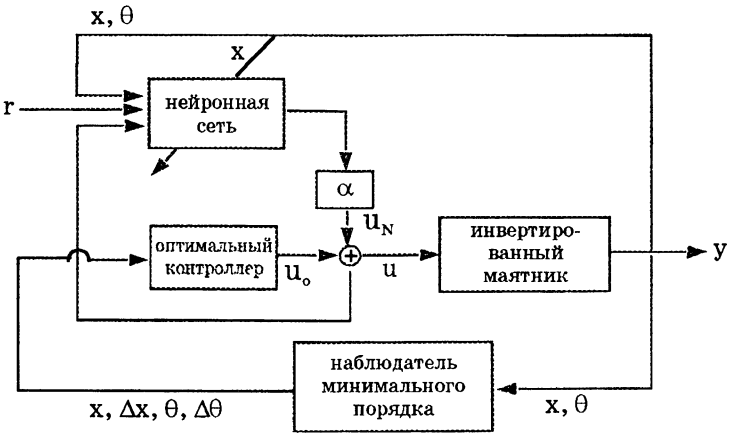


Рис. 5.3.7. Инвертированный маятник с нейросетевым компенсатором

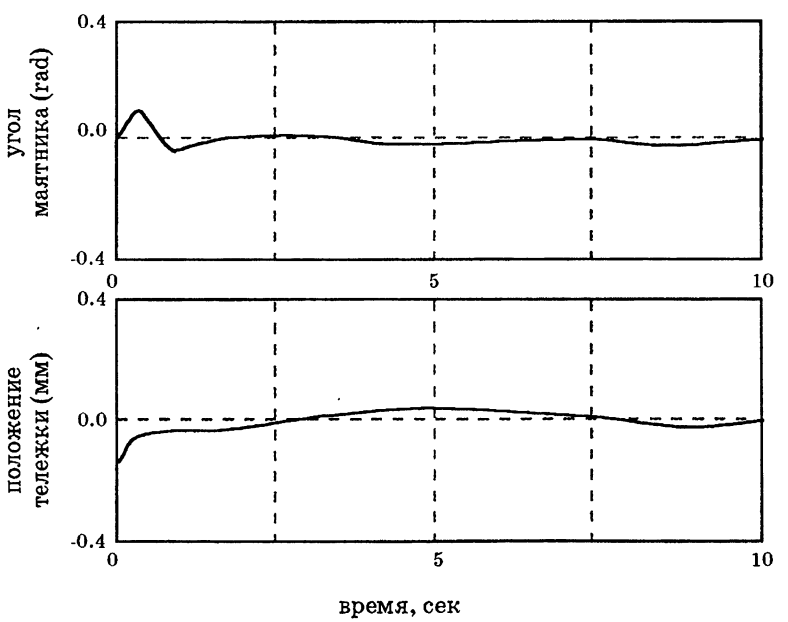


Рис. 5.3.8. Результаты эксперементов с инвертированным маятником с нейросетевым компенсатором

роенный на основе сочетания теорий нечеткого и оптимального управления. При таком методе, даже если весовая функция в функции затрат J изменяется, устранение отклонения позиции x от исходного положения оказывается невозможным. Маломощный двигатель не может устранить смещение из-за нелинейности коэффициента трения системы.

Нейронные сети могут применяться для представления нелинейного отображения «вход-выход» и для реализации нелинейного контроллера. Таким образом, стратегия нейронного управления применяется для улучшения характеристик управления. Конфигурация такого нейроконтроллера показана на рис. 5.3.7. Здесь α — малое положительное число; NN — многослойная нейронная сеть; r(t) — опорная (заданная) величина. На рис. 5.3.7 u_o — оптимальное управление, а u_N — выход нейроконтроллера, минимизирующие величину x^2. Использование этой схемы управления позволяет устранить отклонение, как показано на рис. 5.3.8.

5.3.6. Заключение

В данном исследовании рассмотрено применение гибридного метода, сочетающего традиционные и развитые стратегии управления, для стабилизации инвертированного маятника, представляющего собой пример неустойчивой системы. Сначала маятник поворачивается из устойчивого нижнего положения в вертикальное; для этого применяется набор нечетких правил. Затем маятник удерживается в вертикальном положении с помощью другого набора нечетких правил, обеспечивающих грубое управление. После того как маятник оказывается в вертикальном положении, в действие вступает стратегия линейного оптимального управления; при этом используется линеаризованная модель динамики маятника. Наконец, для компенсации отклонений, возникающих из-за нелинейностей системы, применяется нейронная сеть с небольшим количеством скрытых нейронов.

5.4. Применение ПИД-нейроконтроллера с самонастройкой для управления скоростью электромобиля

5.4.1. Введение

Проблемы охраны окружающей среды вынуждают автомобильную промышленность переходить на выпуск электрических транспортных средств. Японской фирмой Shikoku Electric Power Company разработан новый тип электромобиля, названного PIVOT. Он может поворачиваться вокруг центра и смещаться в боковом направлении. Для эффективного управления электромобилем разработана система имитатора электромобиля, управляемого в реальных условиях дороги. Сопротивление движению может быть реализовано за счет управления моментом двигателя, работающего как генератор.

Авторами данного исследования разработан ПИД-нейроконтроллер с самонастройкой для управления скоростью и моментом нагрузки электромобиля. В этом контроллере настройка ПИД-коэффициентов выполняется нейронной сетью автоматически, в оперативном режиме. Данная система обладает достоинствами самообучения и самоорганизации; кроме того, нейронная сеть может использоваться для решения проблем, связанных с нелинейностями, а также для устранения помех, действующих на систему. С предложенной схемой ПИД-нейронного управления проведено несколько экспериментов. Их результаты сравнивались с данными, полученными при использовании обычных ПИД-контроллеров. Во всех случаях оказывалось, что характеристики ПИД-контроллера с нейросетевой настройкой лучше, а допускаемая им ошибка меньше.

5.4.2. Система управления электромобилем

Схема системы управления электромобилем показана на рис. 5.4.1. Она состоит из двух двигателей и микрокомпьютера. Двигатель постоянного тока M1 работает как генератор и дей-

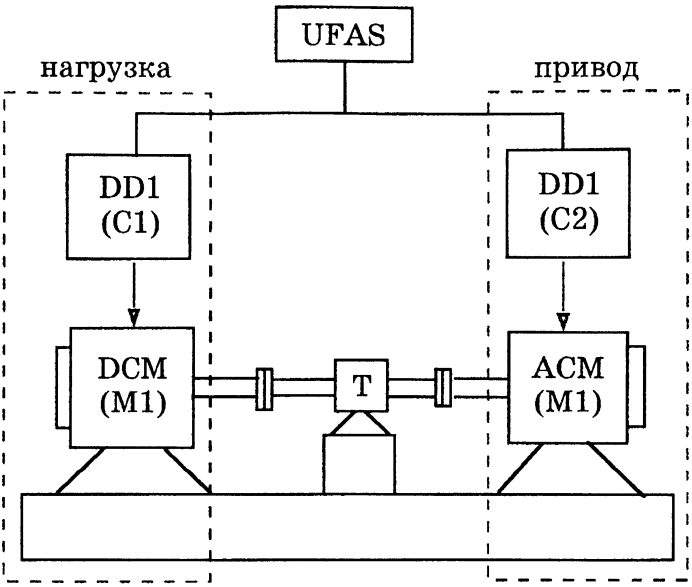


Рис. 5.4.1. Схема генераторной системы электромобиля. UFAS — универсальная суммирующая система; DDC — цифровой контроллер прямого действия; DCM — двигатель постоянного тока; ACM — двигатель переменного тока; T — динамометр для измерения момента.

ствует в качестве движущейся нагрузки электромобиля. Асинхронный двигатель переменного тока M2 используется в качестве приводного двигателя, вращающегося в соответствии с заданной скоростью. Такая конфигурация использована из-за хорошей управляемости двигателя переменного тока, а также из-за легкости и надежности асинхронного двигателя. Управление двумя двигателями (M1 и M2) непосредственно осуществляется двумя цифровыми контроллерами (соответственно C1 и C2). Данная конфигурация системы позволяет имитировать реальные условия движения электромобиля.

Выходная величина (момент) регулируется таким образом, чтобы маховик выходного двигателя вращался в соответствии с опорной скоростью r(t), вычисляемой микрокомпьютером (мо-

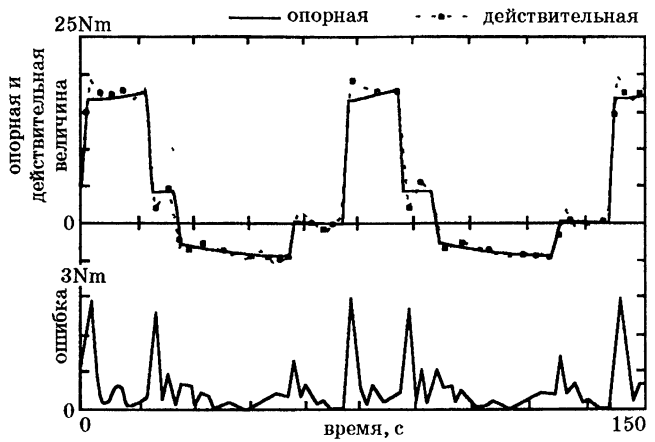


Рис. 5.4.2. Результаты экспериментов для системы управления моментом двигателя с использованием обычного ПИД-контроллера

дель UFAS) на основе рабочего шаблона движения электромобиля. Система управления имеет двойные цепи обратной связи: по скорости и току. Управляющие параметры обычного ПИД-контроллера фиксированы; их значения установлены таким образом, что время интегрирования составляет 0,2 с, полоса пропорциональности (вычисляемая как величина, обратная коэффициенту пропорциональности, умноженная на 100) составляет 5%, а время дифференцирования пренебрежимо мало. Настройка ПИД-контроллера выполняется по методу, изложенному в работе [32].

На рис. 5.4.2 приведены результаты экспериментов по управлению 1000-килограммовым электромобилем с использованием обычного ПИД-контроллера. Скорость автомобиля сначала нарастала с 0 до 60 км/ч, затем поддерживалась постоянной в течение 10 с, после чего снижалась. При увеличении скорости ускорение составляло 4 км/ч/с, при снижении — 2 км/ч/с. Видно, что в диапазоне от 0 до 375 с ошибка составляла от 1,5 до 3,0 Н·м. В следующем разделе рассматривается

использование такого же ПИД-контроллера в аналогичных условиях, однако для настройки ПИД-контроллера применяется нейронная сеть.

5.4.3. ПИД-нейроконтроллер с самонастройкой

На рис. 5.4.3 приведена структурная схема системы управления электромобилем на базе ПИД-нейроконтроллера с самонастройкой. Такой тип нейроконтроллера является практически целесообразным, так как ПИД-контроллеры широко распространены. Передаточная функция ПИД-контроллера Gc(s) задается выражением

$$G_c(t) = k_c\left[1 + (1/T_i)(1/s) + T_d s\right]. \qquad (5.4.1)$$

Здесь k_c — коэффициент пропорциональности, T_i — время интегрирования, T_d — время дифференцирования. В системе с дискретным временем возможна следующая аппроксимация управляющего входного воздействия:

$$u(t) = u(t-1) + K_p(t)[e(t) - e(t-1)] + \\ + K_I(t)e(t) + K_D(t)[e(t) - 2e(t-1) + e(t-2)], \qquad (5.4.2)$$

$$K_p(t) = k_c - K_I(t)/2, \quad K_I(t) = k_c T/T_i,$$

$$K_D(t) = k_c T_d/T, \qquad (5.4.3)$$

где u(t) — управляющее входное воздействие, e(t) — ошибка выхода, K_p, K_I и K_D - пропорциональный, интегральный и дифференциальный коэффициенты соответственно, T — интервал дискретизации.

Нейронная сеть корректирует коэффициенты ПИД-контроллера, которые представляют собой три выходных координаты сети, таким образом, чтобы минимизировать функцию затрат

$$J = \frac{1}{2}e(t)^2, \qquad (5.4.4)$$

где

$$e(t) = r(t) - y(t). \qquad (5.4.5)$$

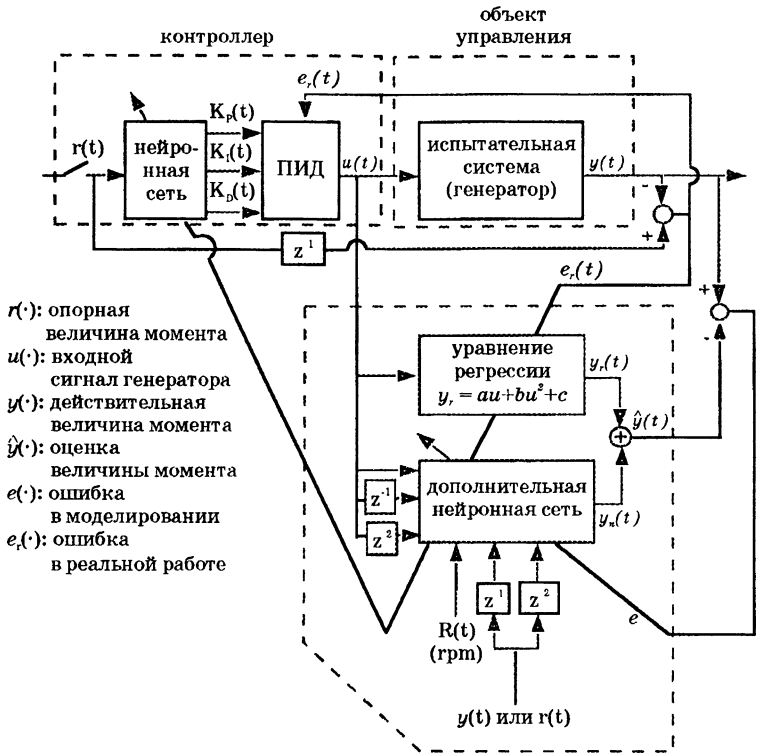


Рис. 5.4.3. Структурная схема системы управления электромобилем на базе ПИД-нейроконтроллера с самонастройкой

Веса связей нейронной сети w_{ij} задаются выражениями

$$\Delta w_{ji}(t) = -\eta \frac{\partial J}{\partial w_{ji}(t-1)} + \alpha \Delta w_{ji}(t-1) + \beta \Delta w_{ji}(t-2), \qquad (5.4.6)$$

$$w_{ji}(t) = w_{ji}(t-1) + \Delta w_{ji}(t), \qquad (5.4.7)$$

где η — скорость обучения, α — мгновенная скорость, β — степень ускорения [6]. Производная $\partial J / \partial w_{ji}(t\text{-}1)$ вычисляется следующим образом:

$$\frac{\partial J}{\partial w_{ji}(t-1)} = -[r(t) - y(t)]\frac{\partial y(t)}{\partial w_{ji}(t-1)} = -e(t)\frac{\partial y(t)}{\partial w_{ji}(t-1)}, \quad (5.4.8)$$

$$\frac{\partial y(t)}{\partial w_{ji}(t-1)} = \frac{\partial y(t)}{\partial u(t-1)}\frac{\partial u(t-1)}{\partial w_{ji}(t-1)}, \quad (5.4.9)$$

$$\frac{\partial u(t-1)}{\partial w_{ji}(t-1)} = \frac{\partial K(t-1)}{\partial w_{ji}(t-1)}\left[\frac{\partial u(t-1)}{\partial K(t-1)}\right]^T, \quad (5.4.10)$$

где $K(t\text{-}1) = [K_p(t\text{-}1), K_I(t\text{-}1), K_D(t\text{-}1)]$. Следует отметить, что $K(t\text{-}1)$ — это выходной вектор нейронной сети, а градиент по весам $w_{ij}(t\text{-}1)$ задается выражением

$$\frac{\partial K(t-1)}{\partial w_{ji}(t-1)} = \left[\frac{\partial K_p(t-1)}{\partial w_{ji}(t-1)}, \frac{\partial K_I(t-1)}{\partial w_{ji}(t-1)}, \frac{\partial K_D(t-1)}{\partial w_{ji}(t-1)},\right]. \quad (5.4.11)$$

Из уравнения (5.4.2) получим

$$\frac{\partial u(t-1)}{\partial K(t-1)} \equiv [u_1(t-1), u_2(t-1), u_3(t-1)], \quad (5.4.12)$$

где

$$u_1(t) = e(t) - e(t-1),$$
$$u_2(t) = e(t),$$
$$u_3(t) = e(t) - 2e(t-1) + e(t-2).$$

Составляющие выражения (5.4.11.) определяются по методу обратного распространения, а производная $\partial J/\partial w_{ji}$ вычисляется следующим образом:

$$\frac{\partial J}{\partial w_{ji}(t-1)} = -e(t)\frac{\partial y(t)}{\partial u(t-1)}[u_1(t-1)\frac{\partial K_p(t-1)}{\partial w_{ji}(t-1)} + u_2(t-1)\frac{\partial K_I(t-1)}{\partial w_{ji}(t-1)} + u_3(t-1)\frac{\partial K_D(t-1)}{\partial w_{ji}(t-1)}]. \quad (5.4.13)$$

Для повышения точности, при оценке якобиана объекта управления $\partial y(t)/\partial u(t\text{-}1)$ используется дополнительная нейронная сеть (обозначенная на схеме как NNM).

5.4.4. Применение нейронной сети для управления скоростью электромобиля

5.4.4.1. Настройка полосы пропорциональности

Рассмотрим применение нейронной сети для настройки полосы пропорциональности ПИД-контроллера, показанного на рис. 5.4.3. Его передаточная функция имеет вид

$$G_S = (100/P_B)[1 + (1/T_I)(1/s)], \qquad (5.4.14)$$

$$P_B = P_{BF} + P_{BN}, \qquad (5.4.15)$$

где P_B — полоса пропорциональности, P_{BF} — ее постоянная составляющая (5%), P_{BN} — параметр, настраиваемый с помощью нейронной сети. T_i — время интегрирования (0,2 с); величина k_c в выражении (5.4.3) есть $100/(P_{BF} + P_{BN})$. Таким образом, корректируя P_{BN}, можно настраивать коэффициенты $K_P(t)$ и $K_I(t)$. В данной задаче коэффициент $K_D(t)$ не используется. Управляющее воздействие u(t), как это следует из выражений (5.4.2) и (5.4.3), рассчитывается по формуле

$$u(t) = u(t-1) + [1/(P_{BF} + P_{BN})] + \\ + [(1 - T/2T_i)(e(t) - e(t-1)) + (T/T_i)e(t)] \qquad (5.4.16)$$

Функция затрат J_R имеет вид

$$J_R = \frac{1}{2}[R_{ref}(t-m) - R(t)]^2, \qquad (5.4.17)$$

где $R_{ref}(t\text{-}m)$ — желаемая (заданная) скорость в момент (t-m)T, а R(t) — действительная величина скорости. Здесь mT и T — интервалы дискретизации для микрокомпьютера и контроллера соответственно; значение mT — порядка 1 с, T — порядка 0,005 с. Причина существенного различия в порядке этих величин заключается в том, что нейронная сеть реализуется с помощью микрокомпьютера с относительно невысоким быстродействием, а для реализации контроллера используется специализированное устройство. Таким образом, веса связей и управляющий сигнал могут корректироваться с интервалом, рав-

ным mT, в то время как действительный интервал дискретизации для систем электромобиля равен T.

В данном случае выходной величиной нейронной сети является только P_{BN}; величина же производной рассчитывается аналогично (5.4.13) в соответствии с выражениями

$$\frac{\partial J_R}{\partial w_{ji}(t-m)} = -[R_{ref}(t-m) - R(t)]\frac{\partial R(t)}{\partial u(t-m)} \times \times \frac{\partial u(t-1)}{\partial P_{BN}} \frac{\partial P_{BN}}{\partial w_{ji}(t-1)}, \quad (5.4.18)$$

$$\frac{\partial u(t-1)}{\partial P_{BN}} = [-1/(P_{BF}+P_{BN})^2] \times \times [(1-T/2T_i)(e(t-1)-e(t-2)) + (T/T_i)e(t-1)]. \quad (5.4.19)$$

5.4.4.2. Настройка времени интегрирования

В данном конкретном случае положим $T_i = T_{if} + T_{iN}$, где T_{if} — постоянная величина (0,2 с), а величина T_{iN} настраивается нейронной сетью. Производная $\partial J_R/\partial w_{ji}(t\text{-}1)$ находится следующим образом:

$$\frac{\partial J_R}{\partial w_{ji}(t-1)} = -[R_{ref}(t-m) - R(t)]\frac{\partial R(t)}{\partial u(t-m)} \times \times \frac{\partial u(t-1)}{\partial T_{iN}} \frac{\partial T_{iN}}{\partial w_{ji}(t-1)}, \quad (5.4.20)$$

$$\frac{\partial u(t-1)}{\partial T_{iN}} = (100/P_B)[T/2T_i^2(e(t-1)-e(t-2)) + + (-T/T_i^2)e(t-1)]. \quad (5.4.21)$$

5.4.5. Заключение

На рис. 5.4.4 показаны характеристики двух систем управления скоростью электромобиля: построенной на основе обыч-

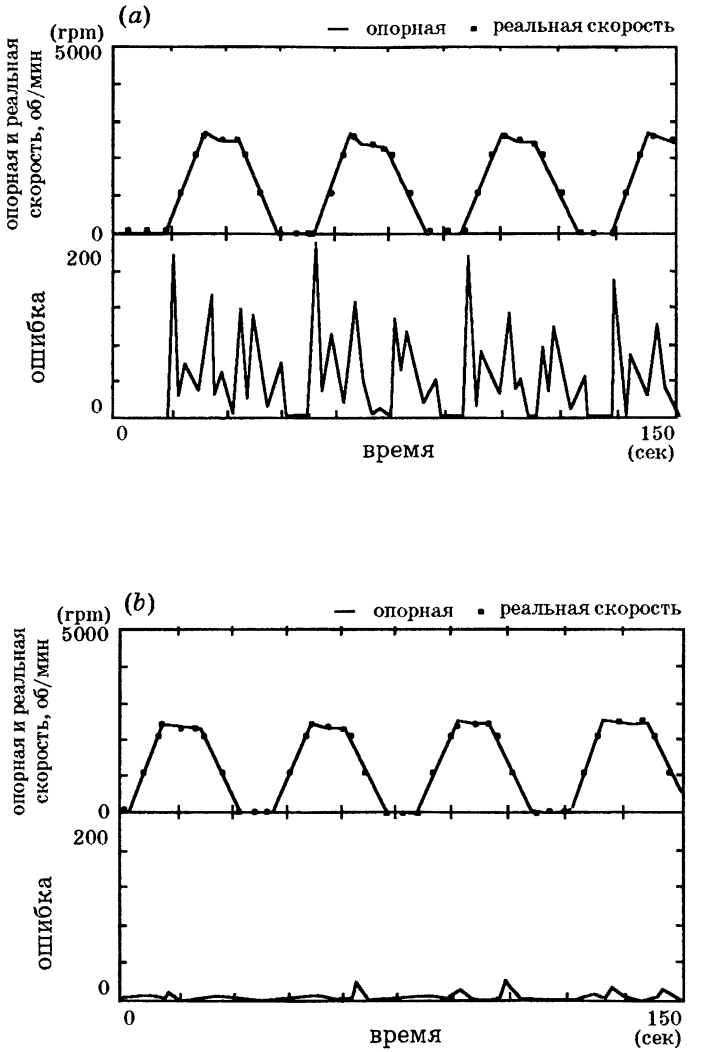


Рис. 5.4.4. Результаты экспериментов по управлению скоростью электромобиля: (*a*) с использованием обычного ПИД-контроллера; (*b*) с использованием ПИД-нейроконтроллера с самонастройкой.

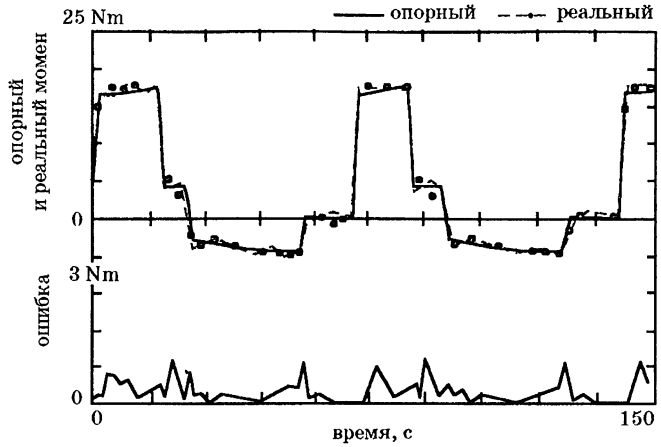


Рис. 5.4.5. Результаты экспериментов по управлению моментом двигателя с использованием ПИД-нейроконтроллера с самонастройкой.

ного ПИД-контроллера (a) и на основе ПИД-нейроконтроллера с самонастройкой (b). Для настройки ПИД-контроллера использовался метод Такахаши (Takahashi) и др. [32]. Сравнивая результаты, можно убедиться, что ПИД-контроллер с нейросетевой настройкой показывает лучшие результаты, обеспечивая меньшую ошибку. На рис. 5.4.5 показаны характеристики системы управления моментом двигателя, построенной на основе ПИД-нейроконтроллера с самонастройкой. Эти характеристики можно сравнить с полученными для обычного ПИД-контроллера (рис. 5.4.2). Как и в предыдущем случае, видно, что при использовании нейронной сети ошибка существенно меньше.

В данном исследовании рассмотрена схема, в которой нейронная сеть используется для настройки ПИД-контроллера; такая схема называется «ПИД-нейроконтроллером с самонастройкой». Данная схема применялась для управления скоростью электромобиля. Из результатов экспериментов видно, что использование предложенной схемы ведет к существенному снижению фактической ошибки (по сравнению с обычным ПИД-контроллером).

5.5. Применение нейронных сетей для управления печью

5.5.1. Введение

В последнее время предложено несколько схем обучения нейронных систем управления [34]. Среди них одной из наиболее ценных оказалась схема с использованием инверсной модели [16], [17]. Одно из достоинств данной схемы — простота ее реализации. В обобщенном подходе к обучению, предложенном в работе [17] и описанном в разделе 4.2, обучение нейронной сети выполняется в автономном режиме. Этот процесс аналогичен пакетному обучению в распознавании образов, когда целевыми образами являются входные сигналы объекта управления, а входные образы состоят из соответствующих выходных сигналов объекта управления и некоторых задержанных сигналов этого объекта. При распознавании образов нейронная сеть обучается запоминать и вспоминать образы, а в системах управлении она обучается обнаруживать отношения между образами. Сходимость достигается путем минимизации ошибки, т.е. рассогласования между целевым образом и выходом нейронной сети. Это рассогласование должно быть ниже некоторой заданной величины. После того как сеть заканчивает обучение на инверсной модели, она перестраивается в контроллер прямого действия.

Хотя схема нейронного управления на основе инверсной модели по своей архитектуре аналогична обычной схеме управления с обратной связью, свойства нейроконтроллера несколько отличаются от свойств обычного контроллера с обратной связью. Нейроконтроллер по своим характеристикам аналогичен традиционному самонастраивающемуся адаптивному контроллеру [1]. В схеме прямого управления с самонастройкой требуется априорное знание структуры математической модели объекта управления. Необходимо также находить оценки параметров объекта управления, например методом наимень-

ших квадратов (см. раздел 3.3). Только после этого регулятор может выработать правильное управляющее воздействие, используя инверсную математическую модель объекта.

Подход к нейронному управлению на основе инверсной модели является в этом отношении более надежным и простым, так как не требует обязательного построения математической модели объекта. Способность нейросетевых моделей к обучению и обобщению на основе характеристик типа «вход-выход процесса» является важным достоинством, так как позволяет разрешать многие проблемы управления, используя при этом менее точные знания об объекте управления. Кроме того, использование полулинейных сигмоидных функций в качестве функций активации для скрытых слоев сети позволяет управлять системами с высокой степенью нелинейности, что невозможно, если использовать только адаптивные и обычные методы. Более того, использование нейросетевых моделей позволяет работать со сложными, плохо определенными объектами управления, для которых сложно разработать математическую модель.

Еще одно достоинство нейросетевых моделей заключается в возможности оперативного обучения, направленного на улучшение характеристик модели. В базовом варианте подхода на основе инверсной модели нейроконтроллер может подвергаться оперативному обучению для минимизации ошибки системы. Однако в такой архитектуре минимизация ошибки характеристики не означает минимизации действительной ошибки выхода объекта управления, связанной с расположением объекта управления. Чтобы минимизировать фактическую ошибку выхода нейроконтроллера, требуется знать якобиан объекта управления (см., например, схему специализированного обучения [17]), что противоречит вышеуказанным достоинствам нейронного управления. В работе [35] предложена схема аппроксимации, где вместо элементов якобиана объекта управления используются только знаки этих элементов. Если известно, в каком направлении параметры управления влияют на

выходные величины объекта управления, то достаточно правильно учитывать только упомянутые направления. Такой подход приемлем для несложных объектов управления, работающих в установившемся режиме.

Более точный подход к оперативному обучению нейроконтроллеров состоит в адаптации архитектуры косвенного адаптивного управления. При этом вместо модели идентификации используется нейронная сеть аналогичной архитектуры, как показано на рис. 4.2.5. В работе [8] такая архитектура использовалась в сочетании с опорной моделью. В работах [36], [9], [10] и ряде других предлагалась аналогичная схема, получившая название «обратное распространение во времени». В этой схеме сначала нейросетевая модель обучалась эмулировать объект управления. Затем нейронный эмулятор использовался в качестве средства обратного распространения ошибки характеристики с целью получения эквивалентной ошибки на выходе нейроконтроллера. Данная схема включает оперативную реализацию нейроконтроллера посредством подачи на вход случайных сигналов и непрерывного повторения желаемого цикла траектории до достижения минимума ошибки рассматриваемой характеристики.

В данной работе архитектура нейроэмулятора и нейроконтроллера применяется для управления миниатюрной промышленной печью со многими входами и выходами. Внешний вид такой печи показан на рис. 5.5.1, а структурная схема — на рис. 5.5.2. Оперативная совместная реализация как нейросетевого эмулятора, так и нейроконтроллера в реальном времени нецелесообразна, так как этот процесс является медленным по своей природе. Нецелесообразно также применять случайные входные сигналы в течение длительного времени или многократно повторять желаемый цикл траектории. Поэтому первоначальное автономное обучение нейросетевой модели является обязательным.

Исходным пунктом в данном подходе является разумный выбор нейросетевых моделей, подвергаемых автономному обучению для построения инверсной и прямой модели каждого из

каналов печи. Обучающие образцы берутся из характеристик «вход-выход» каждого из каналов. Эти модели затем испытываются в оперативном режиме, в результате чего отбираются лучшие модели нейроконтроллера и нейроэмулятора. Далее они настраиваются для управления процессом по схеме, приведенной на рис.4.2.5. Достоинством такой конфигурации является возможность дальнейшего оперативного обучения нейроконтроллера и нейроэмулятора с целью адаптации к окружающим условиям и улучшения характеристик. В настоящем исследовании, однако, не приводится детальный анализ поведения нейронных сетей.

Простейший способ показать эффективность новой схемы управления — сравнить ее с существующими схемами. С этой целью для одной и той же печи была разработана схема адаптивного управления с самонастройкой на основе алгоритма Кларка-Гаутропа [14], [15] и схема обычного ПИД-управления [33]. В экспериментах все три схемы сравнивались по отслеживанию изменений уставки, по характеристикам в условиях неизвестных возмущений на нагрузке и в условиях изменений параметров процесса, по насыщению выходных сигналов контроллера. В последующих разделах приводится краткое описание системы управления печью, рассматривается реализация схемы нейронного управления печью и анализируются результаты экспериментов.

5.5.2. Система управления печью

Система управления печью разработана японской фирмой Omron Inc. Система представляет собой трехканальный медный цилиндр (рис. 5.5.1). Печь представляет собой миниатюрную модель устройств, применяемых в промышленности для формовки пластмассовых изделий. В такой печи требуется контролировать температуры от 100°C до 300°C в трех каналах. В состав системы управления входит модуль датчиков, плата параллельного интерфейса ввода-вывода (ПВВ), микрокомпьютер NEC PC-9801F и исполнительное устройство. Мощность

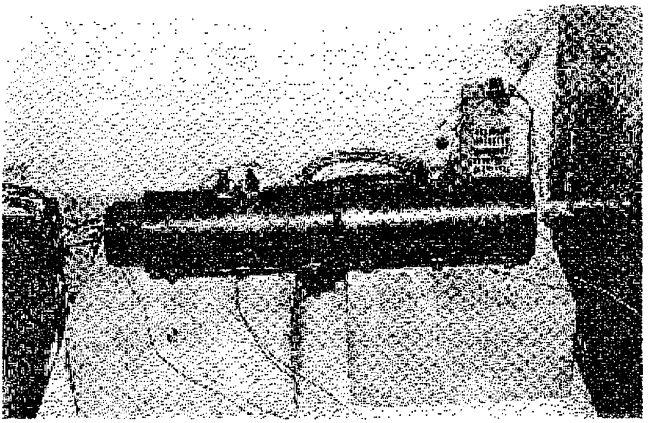


Рис. 5.5.1. Внешний вид печи

нагревателей — 280 Вт для меньшего цилиндра («канал 1») и 1,2 кВт для двух других цилиндров (каналы 2 и 3). Хотя каждым из каналов можно управлять независимо, теплопередача между каналами затрудняет управление.

Модуль датчиков включает термоэлемент типа K (TC12-4K) для изменения температуры на выходе цилиндров. Температуры от 0 до 120°C могут измеряться и преобразовываться в цифровую форму с помощью 12-разрядного АЦП (AD6940) с разрешающей способностью 0,2°C. Плата интерфейса ввода-вывода (PIO-16/16) используется для передачи цифровых сигналов между микрокомпьютером и внешними устройствами; эта плата вставляется в расширенное гнездо ввода-вывода микрокомпьютера. Для включения и выключения нагревателей по управляющим сигналам используются твердотельные тиристоры (Omron G3N-220B). Эти сигналы модулированы по ширине импульса (ШИМ). Когда управляющий сигнал имеет номинальную величину (100%), нагреватель включается в каждом интервале дискретизации на 6 с непрерывной работы. Основная управляющая программа написана на Microsoft C, а подпрограммы управления нагревателями — на Ассемблере (Intel 8086).

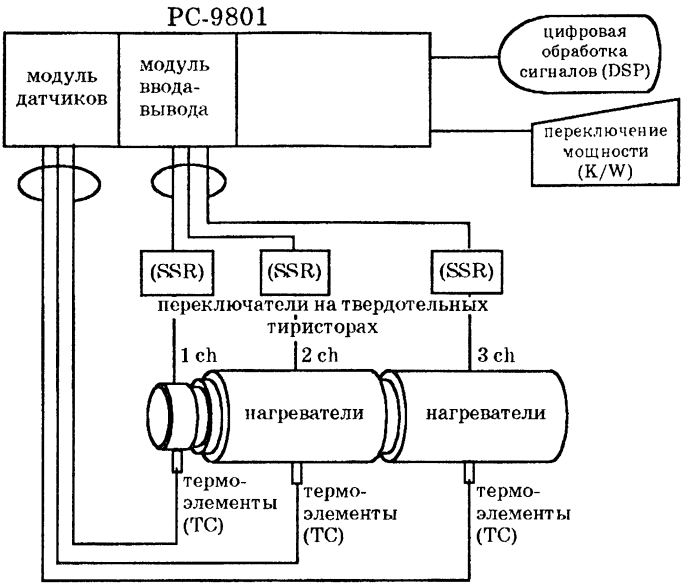


Рис. 5.5.2. Структурная схема системы управления печью.

5.5.3. Схемы нейронного управления печью

5.5.3.1. Схема автономного обучения

Для формирования инверсной и прямой модели каждого из каналов печи на основе архитектуры обобщенного обучения было выбрано несколько нейросетевых моделей, каждая из которых имела свое число элементов входных векторов и скрытых нейронов. Так как обучение нейронных сетей осуществлялось автономно, требовался выбор обучающих образцов «вход-выход».

Во многих разработках [8]—[10] для построения нейросетевых инверсных и прямых моделей объекта управления использовалась случайные входные сигналы. В этом случае сходимость достигалась только после проведения нескольких тысяч циклов оперативного обучения. Процесс, исследовавший-

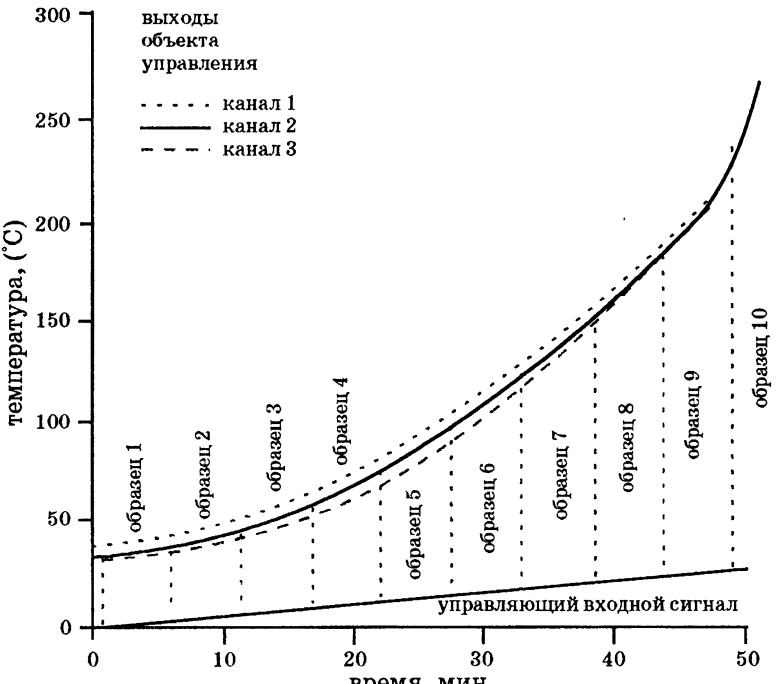


Рис. 5.5.3. Характеристика печи «вход-выход» в разомкнутом состоянии при подаче плавно нарастающего входного сигнала. Для обучения нейросетевых моделей выбрано 10 пар обучающих образцов «вход-выход».

ся в настоящей работе, является по своей природе медленным, устойчивым и разомкнутым. Поэтому использовалась подача плавно нарастающих входных сигналов в диапазоне между предельно допустимыми величинами для данных исполнительных устройств. Характеристики «вход-выход» для каждого канала печи приведены на рис. 5.5.3.

Обучение нейросетевых моделей производилось по обобщенной схеме. Для каждого канала выбирались обучающие образцы «вход-выход» через равные интервалы по всей характеристике объекта управления (см. рис. 5.5.3). Модели нейроконт-

роллеров каждого из каналов печи обучались инверсной динамике с использованием выходных сигналов (и их задержанных значений) в качестве входных образцов и входных сигналов соответствующих каналов в качестве целевых образцов. Модели нейроэмуляторов, в свою очередь, обучались прямой динамике каждого из каналов печи с использованием входных сигналов (и их задержанных значений) в качестве входных образцов и соответствующих выходных сигналов в качестве целевых образцов. В автономном обучении нейронной сети с целью построения модели прямой или инверсной динамики объекта управления, сигнал обратного распространения между выходным и скрытым слоем выражается в виде

$$\delta_k = (\tau_k - O_k), \qquad (5.5.1)$$

где τ_k — целевой, заданный образец, O_k — действительный выход нейронной сети. Сигнал между скрытым и входным слоем описывается выражением

$$\delta_j = f'(S_j)\sum_k \delta_k w_{kj}, \qquad (5.5.2)$$

где f(x)=1/(1+exp(-x)). Веса связей между входным и скрытым слоем корректируются в соответствии с выражением

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t) + \beta \Delta w_{ji}(t-1), \qquad (5.5.3)$$

а между скрытым и выходным слоем — с выражением

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t) + \beta \Delta w_{kj}(t-1), \qquad (5.5.4)$$

где O_i и O_j — выходные сигналы входного и скрытого слоя соответственно, η — скорость обучения, α и β — мгновенный коэффициент и коэффициент ускорения. Константы η, α и β выбираются эмпирически.

Оптимальное количество обучающих образцов, скрытых нейронов и элементов входных векторов для моделей контроллера и нейроэмулятора выбираются экспериментальным путем. К сожалению, надежных алгоритмов определения этих параметров еще не существует. Чтобы избежать необходимости обучать большое количество нейросетевых моделей, для каждого из каналов пер-

воначально было выбрано четыре модели, имеющие 5,10,15 и 20 скрытых нейронов. Затем выполнялось обучение нейросетевых моделей с использованием нескольких компьютеров. После этого сети корректировались в соответствии с лучшим количеством скрытых нейронов. Количество обучающих образцов также выбиралось методом проб и ошибок. В таких случаях сначала обычно устанавливается большая величина скорости обучения η (около 0,01). Если же возникают осцилляции, то скорость снижается до одной десятой от предыдущей величины. Мгновенные величины α и β выбираются равными 0,8 и -0,15 соответственно. Для каждой из моделей обучение прекращается, если сходимость не уменьшается более чем на 0,0001% за 1000 итераций.

Так как нейронные сети имеют высокую степень нелинейности, сложно определить аналитически, какая модель, действительно, обучена инверсной и прямой динамике объекта. Имеется простой и надежный метод оперативного испытания каждой из моделей. Для нейросетевых моделей контроллера испытания проводятся путем настройки этой модели на непосредственное управление процессом. Модели нейроэмулятора испытываются путем применения плавно нарастающего сигнала к каждому из процессов и к соответствующим эмуляторам. Затем производится сравнение выходных процессов.

Было обнаружено, что процесс обучения нейронных сетей зависит от количества скрытых нейронов, элементов входного вектора и обучающих образцов. Лучшим для каждого из каналов оказался нейроконтроллер, содержащий 10 скрытых нейронов и 3 элемента входного вектора: текущая выходная величина процесса y(t) и две задержанных величины y(t-1) и y(t-2). Лучшим нейроэмулятором оказался вариант, содержащий 12 скрытых нейронов и 3 элемента входного вектора: y(t), y(t-1) и y(t-2). Установлено, что схема выбора десяти обучающих образцов, приведенная на рис. 5.5.3, является подходящей для достижения хорошей сходимости при обучении нейросетевых моделей.

5.5.3.2. Оперативное обучение

Настройка нейроконтроллера и нейроэмулятора для каждого процесса выполняется в схеме управления, показанной на рис. 5.5.2. Для этой схемы оперативного обучения коррекция весов нейросетевого контроллера производится на основе характеристики ошибки:

$$E = \frac{1}{2}[r-y]^2, \tag{5.5.5}$$

где r и y — соответственно заданная (желаемая) и фактическая выходные координаты объекта управления. Эта ошибка распространяется в обратном направлении через эмулятор. Так как функция выходных нейронов является линейной, сигнал ошибки между скрытым и выходным слоями эмулятора выражается как

$$\delta_k^E = r - y, \tag{5.5.6}$$

где индекс E обозначает эмулятор, а индекс k — выходной слой. Сигнал ошибки между скрытым и входным слоем эмулятора определяется формулой

$$\delta_k^E = -\frac{\partial E}{\partial net_j^E} = -\frac{\partial E}{\partial O_j^E}\frac{\partial O_j^E}{\partial net_j^E}, \tag{5.5.7}$$

где O_j^E и net_j^E — соответственно выходной и входной сигналы нейронов скрытого слоя эмулятора. Используя цепное правило, получим выражение

$$\delta_j^E = \left(\sum_j \delta_k^E w_{kj}^E\right) O_j^E(1-O_j^E), \tag{5.5.8}$$

где w_{kj}^E — вес связи между скрытым и выходным слоем эмулятора. Сигнал ошибки между скрытым и выходным слоем контроллера:

$$\delta_k^C = \frac{\partial E}{\partial net_k^C} = \frac{\partial E}{\partial O_k^C}\frac{\partial O_k^C}{\partial net_k^C}, \tag{5.5.9}$$

где O_k^C и net_k^C — соответственно выходной и входной сигналы нейронов выходного слоя контроллера. Так как выходные нейроны контроллера описываются линейной функцией с насыщением, то $\frac{\partial O_k^C}{\partial net_k^C} = 1$ и, таким образом,

$$\delta_k^C = \sum_j \delta_j^E w_{ji}^E. \qquad (5.5.10)$$

Для скрытого и входного слоев контроллера сигнал ошибки имеет вид:

$$\delta_j^C = \left(\sum_j \delta_k^C w_{kj}^C \right) O_j^C (1 - O_j^C), \qquad (5.5.11)$$

где O_j^C — выходной сигнал нейронов скрытого слоя контроллера. В данной схеме веса связей нейроэмулятора не являются постоянными, а корректируются в оперативном режиме аналогично тому, как это происходило в системе регулировки температуры водяной ванны.

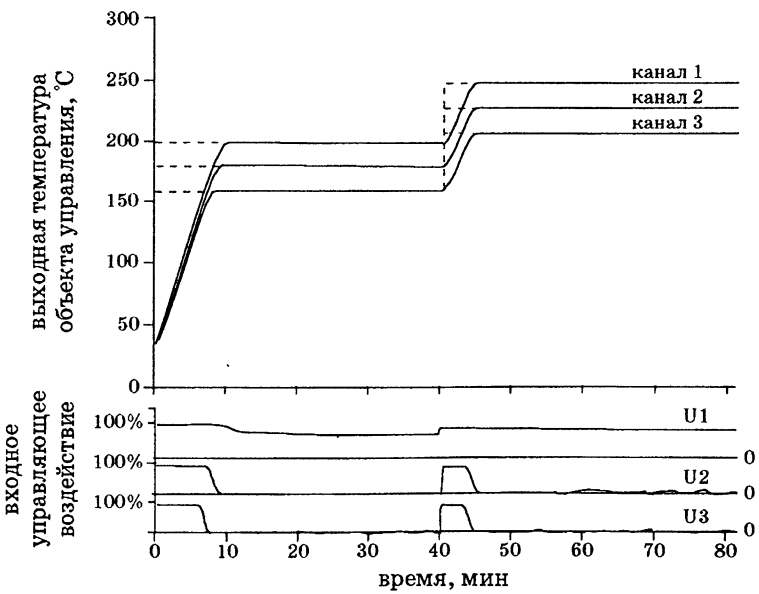
5.5.4. Анализ экспериментальных данных

Во многих промышленных процессах важным фактором является температура. Слишком низкая или слишком высокая температура приводит к снижению качества конечного продукта. В таких случаях целью управления является, возможно, более быстрое и плавное достижение требуемой температуры с последующим удерживанием ее значения в заданных пределах. Для изучения возможности достижения этой цели проводились эксперименты с управлением печью.

Для проверки надежности и адаптивных способностей различных систем управления, построенных на основе нейронных сетей, было проведено несколько экспериментов. Для сравнения нейросетевых систем управления с иными для того же объекта (печи) были построены две другие системы управления: самонастраивающаяся многомерная система управления

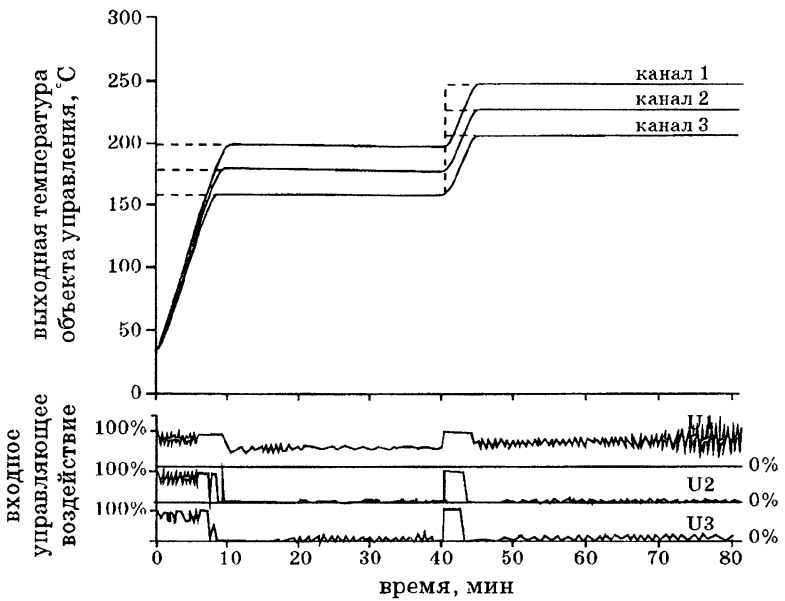
и система на основе обычных ПИД-контроллеров. Принцип работы самонастраивающейся многомерной системы управления основан на алгоритме Кларка (Clarke) и Гаутропа (Gawthrop) [14], [15], а принцип работы системы ПИД-управления — на ускоренном дискретном ПИД-алгоритме [33].

Для реализации самонастраивающейся схемы управления требуется построение априорной математической модели объекта управления. Более подробно эти вопросы рассматриваются в работах [36], [37]. Оценки параметров модели печи рассчитываются оперативно. Для этого используется рекурсивный метод наименьших квадратов с переменным коэффициентом забывания и с диагональной ковариационной матрицей {10, 10, 10}. Для ускорения сходимости оценок парамет-



(*а*) нейронное управление

Рис. 5.5.4, *а*. Характеристики по отслеживанию уставки для различных систем управления печью

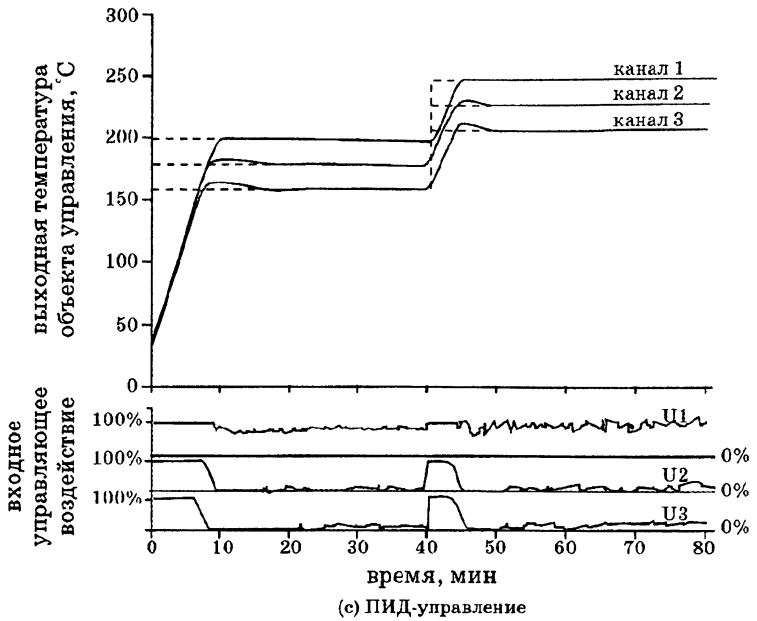


(*b*) самонастраивающееся управление

Рис. 5.5.4, *b*. Характеристики по отслеживанию уставки для различных систем управления печью

ров в начале оценивания используется последовательность, представляющая собой псевдослучайный двоичный сигнал с максимальным периодом. Настройка ПИД-контроллера осуществляется по методу Такахаши (Takahashi) и др. [31], затем для улучшения характеристик выполняется точная настройка. Эксперименты проводились с целью определения способности этих алгоритмов управления к отслеживанию уставки. Каждый из экспериментов проводился для 480 моментов дискретного времени, с интервалом дискретизации в 10 с.

На рис. 5.5.4 показаны характеристики исследуемых схем управления, связанные с изменениями уставки. К характеристикам здесь относятся выходные координаты и соответствующие им входные управляющие сигналы для всех трех каналов печи. Можно видеть, что при использовании ПИД-контролле-



(c) ПИД-управление

Рис. 5.5.4, *с*. Характеристики по отслеживанию уставки для различных систем управления печью

ра заданная уставка достигалась медленнее, чем для других алгоритмов управления, и что при этом допускались выбросы температуры. Видно также, что нейроконтроллеры показали достаточно хорошие характеристики даже без оперативного обучения. Это означает, что нейроконтроллеры каждого канала печи были обучены адекватной инверсной модели соответствующего канала. При оперативном обучении, когда на каждом отсчете выполнялась точная настройка, нейроконтроллер демонстрировал еще лучшие характеристики. Самонастраивающийся контроллер выполнял отслеживание уставки не хуже, чем нейроконтроллер, однако выбор задаваемых пользователем полиномиальных коэффициентов для этого контроллера занимал достаточно много времени. Задаваемые пользователем полиномы P и R определяются как единичные матрицы, а

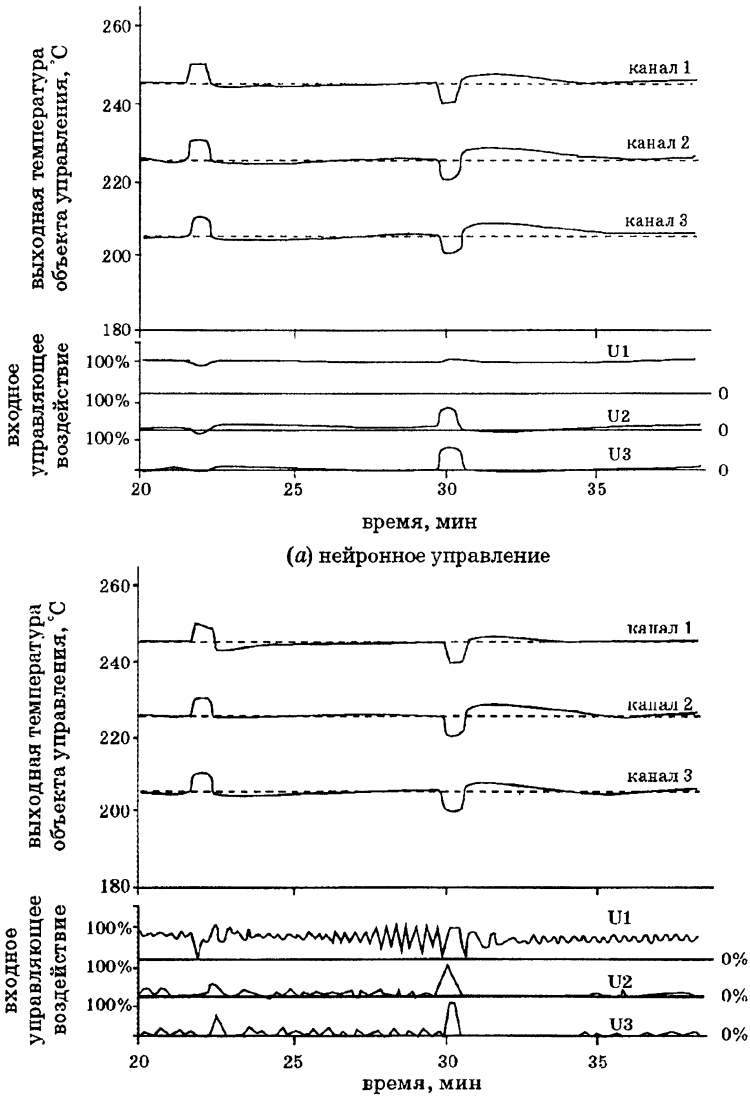
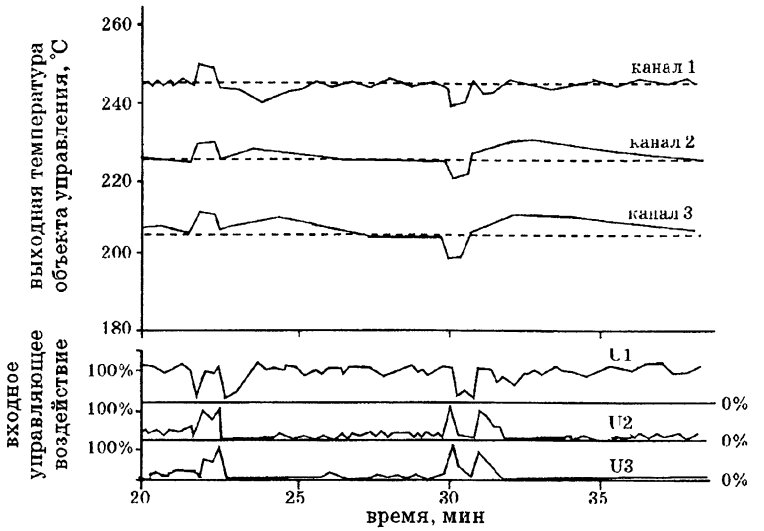


Рис. 5.5.5, *a*, *b*. Характеристики работы в условиях возмущений на нагрузке для различных систем управления печью

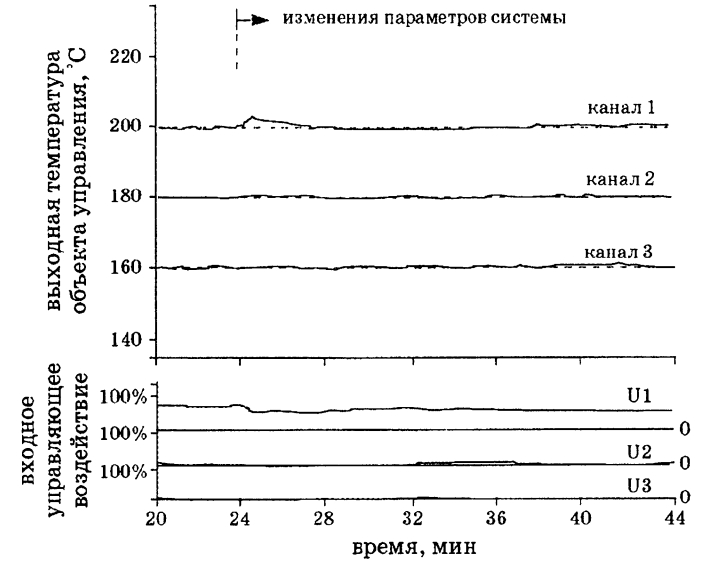


(*c*) ПИД-управление

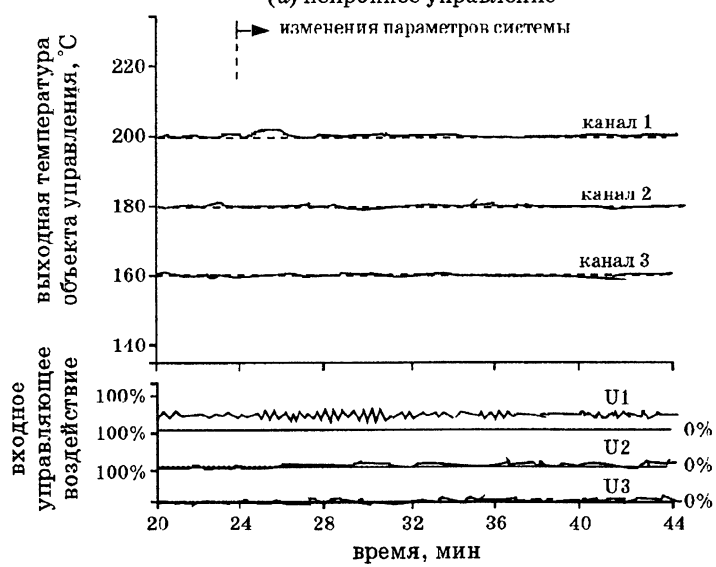
Рис. 5.5.5, *c*. Характеристики работы в условиях возмущений на нагрузке для различных систем управления печью

Q — как диагональная матрица $\{0, 0, 0.02(1-z^{-1})\}$. В этом отношении нейроконтроллер (после того, как он обучен) значительно проще для реализации, так как для него не требуется настройка или выбор задаваемых пользователем параметров.

Вторая серия экспериментов была выполнена с целью исследования работы контроллеров в условиях неизвестных возмущений на нагрузке. Чтобы обеспечить идентичные условия для всех трех контроллеров, на выход каждого канала в интервале времени $150 \le t \le 153$ вносились искусственные возмущения величиной в 5°С, а в диапазоне $180 \le t \le 183$ — величиной в -5°С. Характеристики всех трех систем управления в этих условиях показаны на рис. 5.5.5. Можно заметить, что нейроконтроллер и самонастраивающийся контроллер показали одинаково хорошие характеристики, обеспечивая быстрое устранение последствий положительных возмущений. Хорошие характеристики нейроконтроллера обеспечивались его способ-

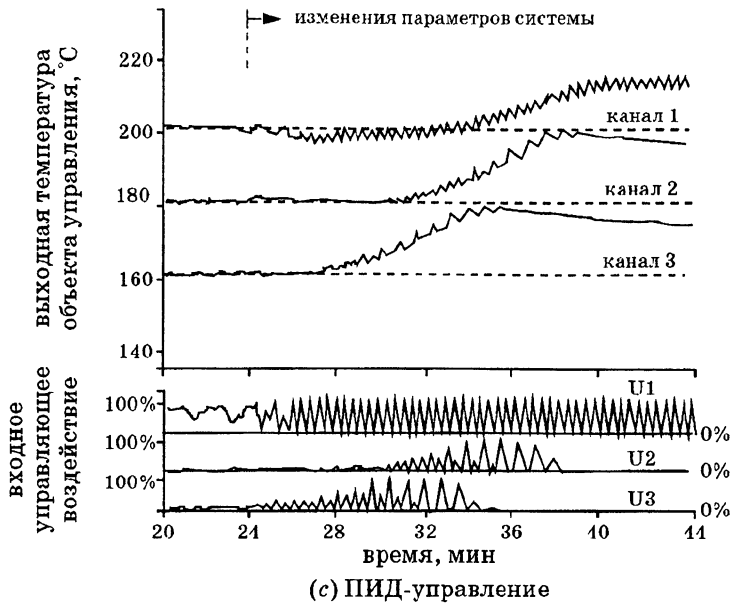


(*a*) нейронное управление



(*b*) самонастраивающееся управление

Рис. 5.5.6, *a*, *b*. Характеристики работы в условиях изменений параметров для различных систем управления печью



(*c*) ПИД-управление

Рис. 5.5.6, *c*. Характеристики работы в условиях изменений параметров для различных систем управления печью

ностью к обобщению, а также возможностями быстрой адаптации к изменениям входных координат. Самонастраивающийся контроллер также успешно работал в этих условиях, так как был оснащен системой прогнозирования на d шагов. В то же время ПИД-контроллер показал худшие характеристики. При отрицательных возмущениях характеристики нейроконтроллера и самонастраивающегося контроллера ухудшались, так как допустимыми были только положительные управляющие входные воздействия; однако можно увидеть, что восстановление для этих контроллеров выполнялось значительно быстрее, чем для ПИД-контроллеров.

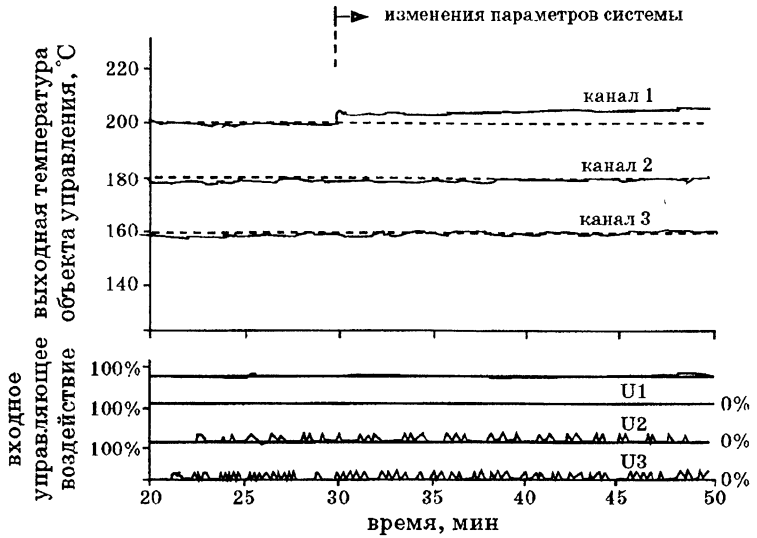
Динамика поведения и параметры многих систем управления нередко имеют тенденцию к непредсказуемому изменению в ходе работы. С целью изучения устойчивости работы рассмат-

риваемых систем управления, были проведены эксперименты, в которых для каждого канала печи после каждого 120-го отсчета дискретного времени выполнялось умножение величины u(t-3) на 0,03. Таким образом имитировалось изменение параметров процесса. Можно убедиться, что такие изменения динамики достаточно существенны. Хотя трудно себе представить реальный объект, параметры которого изменялись бы столь резко; для экспериментального сравнения контроллеров по их устойчивости и способностям к адаптации такое преувеличение допустимо.

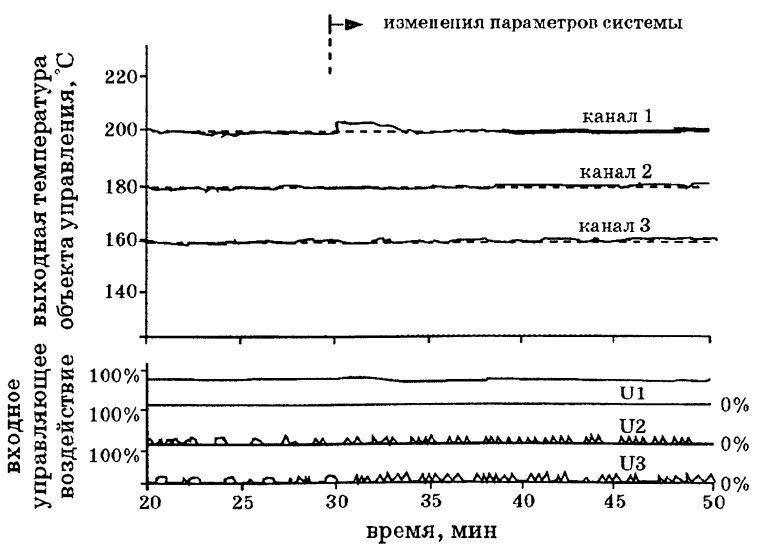
Характеристики контроллеров в данных условиях работы приведены на рис. 5.5.6. Видно, что работа ПИД-контроллеров при изменении параметров системы резко ухудшалась. Несмотря на интегрирующее действие, на всех каналах наблюдалось смещение, возникавшее из-за насыщения в канале управления.

Самонастраивающиеся многомерные контроллеры адаптировались к изменениям параметров объекта управления значительно лучше, чем обычные ПИД-контроллеры, и именно за счет своей способности к самонастройке. Нейросетевые контроллеры также показали в этих условиях высокую устойчивость, благодаря развитым возможностям обобщения, связанным с оперативным адаптивным обучением.

Еще одна серия экспериментов была проведена в условиях наложения строгих ограничений типа насыщения на выходы контроллеров. Для каждого контроллера были установлены нелинейные ограничения: сигнал падал до нуля при достижении уровня ниже 20% и срезался на уровне 80% . Были проведены те же эксперименты, что и в рассмотренной выше третьей серии. Таким образом, исследовалось действие изменений в параметрах процесса. На рис. 5.5.7 (*а*) показаны характеристики нейроконтроллера, не подвергавшегося оперативному обучению. Можно убедиться, что в этом случае на выходах каналов 2 и 3 появляется установившаяся ошибка. При изменении параметров объекта характеристики нейроконтроллера ухудшались. На рис. 5.5.7 (*b*) показаны характеристики нейроконтроллера, под-

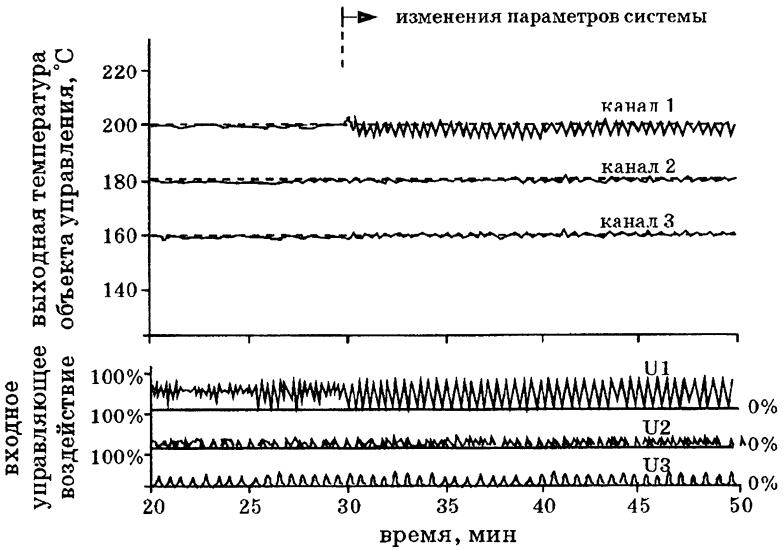


(*a*) нейронное управление без оперативного обучения

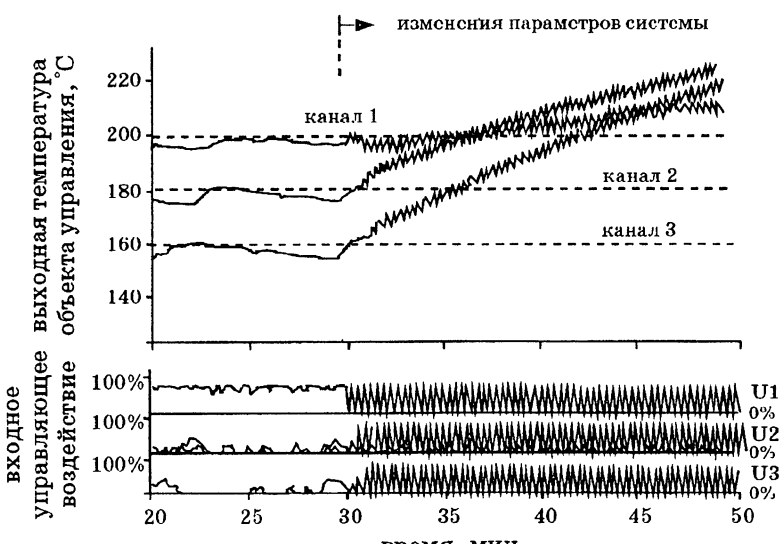


(*b*) нейронное управление с оперативным обучением

Рис. 5.5.7, *a*, *b*. Характеристики работы в условиях изменений параметров для различных систем управления печью



(*c*) самонастраивающееся управление



(*d*) ПИД-управление

Рис. 5.5.7, *c*, *d*. Характеристики работы в условиях изменений параметров для различных систем управления печью

вергавшегося оперативному обучению. Видно, что даже в условиях изменений параметров объекта характеристики нейроконтроллера оставались достаточно хорошими.

На рис. 5.5.7,(*c*) показаны характеристики самонастраивающегося контроллера в тех же условиях. До изменения параметров объекта характеристики контроллера были достаточно хорошими. Однако после изменения параметров характеристики контроллера ухудшились особенно для канала 1. Характеристики ПИД-контроллера в условиях ограничений типа насыщения оказались плохими. На рис. 5.5.7,(*d*) видно, что процесс становится нестабильным при изменении параметров объекта управления и при переходе управляющих сигналов в область насыщения.

Результаты всех экспериментов свидетельствуют, что процессы на выходе нейроконтроллера были очень «гладкими» даже в условиях существенных возмущений на нагрузке. В то же время на выходах самонастраивающегося и ПИД-контроллера наблюдались нежелательные флуктуации, что в реальных условиях может привести к сокращению срока службы исполнительного устройства.

Управляющие сигналы самонастраивающегося контроллера являются сильно осциллирующими. Это особенно относится к первому каналу, так как он больше всех зашумлен и имеет меньшую временную задержку. Колебания сигналов связаны с тем, что самонастраивающееся управление направлено на минимизацию дисперсии выходной величины объекта управления. Если объект управления сильно зашумлен, то дисперсия возрастает, что приводит к колебаниям управляющих сигналов. По этой же причине характеристики самонастраивающегося контроллера ухудшаются, когда на управляющий сигнал налагаются ограничения типа насыщения, как в последней серии экспериментов. В отличие от самонастраивающегося контроллера, нейроконтроллер обучается псевдоинверсной динамике объекта управления. Возможности нелинейного отображения, обеспечиваемые нейронами скрытых слоев сети, позво-

ляют «сгладить» выходные величины нейроконтроллера и тем самым продлить срок службы исполнительной системы.

5.6. Заключительные замечания и выводы

В данном разделе рассмотрены четыре примера применения нейронных сетей в системах управления. В качестве первого примера рассмотрено применение нейронных сетей для управления температурой водяной ванны. Рассмотренная система управления проверялась на устойчивость в условиях различных осложняющих факторов: изменения параметров, переменных временных задержек, возмущений на нагрузке. Схема нейронного управления сравнивалась с тремя другими традиционными схемами: управлением на основе нечеткой логики, обобщенным прогнозирующим управлением и ПИД-управлением. Схема нейронного управления продемонстрировала высокую устойчивость и другие заметные преимущества над остальными схемами.

Результаты сравнения четырех схем управления, полученные в первом примере, подтвердились и при исследовании системы управления печью со многими входами и выходами.

В системе управления инвертированным маятником нейронная сеть используется для улучшения характеристик линейного оптимального контроллера. Очевидно, что линейный оптимальный контроллер не может эффективно работать в условиях проявления нелинейных характеристик, таких, как трение и скольжение. В системе управления инвертированным маятником нейроконтроллер использовался в качестве компенсатора таких нелинейностей.

Еще в одном примере нейронная сеть использовалась для настройки параметров ПИД-контроллера. Такой контроллер был назван «ПИД-нейроконтроллером с самонастройкой».

Литература

Литература к главе 1

1. White, D.A. and D.A Sofge, «Handbook of Intelligent Control», Van Nostrand Reinhold, New York, 1992.

2. Miller, W.T., R.S. Sutton, and PJ. Werbos, «Neural Networks for Control», MIT Press, Cambridge, MA, 1990.

3. Rumelhart, D.E. et al., «Parallel Distributed Processing: Explorations in the Micro Structure of Cognition Vol. I: Foundations», MIT Press, Cambridge. MA, 1986.

4. Dayhoff, J.E., «Neural Network Architectures: An Introduction», Van Nostrand Reinhold, New York, 1990.

5. Zurada, J.M., R.J. Marks II, and C.J. Robinson, «Computational Intelligence Imitating Life», IEEE Press, New York, 1994.

6. Nozaka, Y., «Trend of new control theory application in industrial process control (A survey)», Proc. of 12th IFAC World Congress, Sydney, Vol. VI, pp. 51-56, 1993.

Литература к главе 2

1. Cajal, S.R «Histologie du systeme nerveux de 1-homme et des vertebres», Maloine, Paris, 1909.

2. Dayhoff, J.E «Neural Network Architectures: An Introduction», Van Nostrand Reinhold, New York, 1990.

3. McCulloch, W.S. and W. Pitts, «A logical calculus of the ideas immanent in nervous activity». Bulletin of Mathematical Biophysics, Vol. 9, pp.127-147,1943.

4. Hebb, D.O., «The Organization of Behavior», Wiley, New York, -1949.

5. Bernstein. N., «Profiles: AI, Marvin Minsky», The New Yorker, pp.50 - 126, 1981.

6. Rosenblatt, F., «Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms», Spartan Book, Washinghton D.C., 1961.

7. Minsky, M.L. and S.A. Papert, «Perceptrons', MIT Press, Cambridge, MA, 1969.

8. Hopfield, J.J. «Neural networks and physical systems with emergent computational abilities», Proc. of the National Academy of Sciences», Vol. 79, pp. 2554-2558, 1982.

9. Hopfield, J.J. and D.W. Tank, «Neural computation of decisions in optimization problems». Biological Cybernetics, Vol. 52, pp. 141-152, 1985.

10. Werbos, P.J., «Beyond regression: New tools for prediction and analysis in the behavioral sciences», Ph.D. Thesis, Applied Mathematics, Harvard University, Nov., 1974.

11. Rumelhart, D.E. et al., «Parallel Distributed Processing: Explorations in the Micro Sructure of Cognition Vol. I: Foundations», MIT Press, Cambridge, MA, 1986.

12. Rumelhart, D.E. and D. Zipser, «Feature discovery and competitive learning». Cognitive Science, Vol. 9, pp. 75-112, 1985.

13. Grossberg, S., «Adaptive pattern classification and universal recoding: . 1. Parallel development and coding of neural feature detectors», Biological Cybernetics, Vol. 23, pp. 121-134, 1976.

14. Kohonen, T., «Self-organized formation of topologically correct feature maps». Biological Cybernetics, Vol. 43, pp. 59-69, 1982.

15. Widrow, B., «ADALINE and MADALINE - 1963», Plenary Speech, Vol. I, Proc. of IEEE First Int. Conf. on Neural Networks, San Diego, CA, pp. 143-158,1987.

Литература к главе 3

1. Ziegler, J.G. and N.B. Nichols, «Optimum settings for automatic controllers», Trans. of ASME, Vol. 15. pp. 827-834, 1942.

2. Takahashi, Y. and C.S. Chan, «Parametereinstellung bei

linearen DDC-algorithmen», Regelungstechnik und Prozess-Datenverarbeitung, Vol.19, pp. 237-244, 1971.

3. Nishikawa, Y., N. Sanomiya, T. Ohta, and H. Tanaka, «A method for auto-tuning of PID control parameters», Automatica, Vol. 20, pp. 321-332,1984.

4. Åström, K.J., and T. Hagglung, «Automatic tuning of simple regulators with specifications on phase and amplitude margins», Automatica, Vol. 20, pp. 645-651, 1984.

5. Kalman, R.E., «Design of self-optimizing control system», Trans. of ASME, Vol. 80, pp. 468-478, 1958.

6. Åström, K.J., and B. Wittenmark, «On self-tuning regulators», Automatica, Vol. 9, pp. 185-199, 1973.

7. Wittenmark, B., «Self-tuning PID controllers based on pole placement», Lund Inst. of Technical Report, TFRT-7179, 1979.

8. Savelli, J.C., K. Warwick, and J.H. Wescott, «Implementation of an adaptive PID self-tuning controller», Proc. Of Melcon, Athens, Vol. 2,1983.

9. Gawthrop, P.J. «Self-tuning PID controllers: Algorithm and implementation», IEEE Trans. on Automatic Control, Vol. 31, pp. 201-209,1986.

10. Banyasz, C. and L. Keviczky. «Direct methods for self-tuning PID regulators», Proc. of the 6th IFAC Symposium on Identification & System Parameter Estimation (Bekey, G.A. and G.N. Saridis, Eds.), Washington, D.C., pp. 1395-1400, 1982.

11. Ortega, R. and R. Kelly, «PID self-tuners: Some theoretical and practical aspects», IEEE Trans. on Industrial Electronics, Vol. IE-31, pp. 312-332,1984.

12. Radke, F. and R. Isermann, «A parameter-adaptive PID-controller with stepwise parameter optimization», Automatica, Vol. 23, pp. 449-457,1987.

13. Jones, A.H. and B. Porter, «Design of adaptive digital set-point tracking PID controllers incorporting recursive step-response matrix identifiers for multivariable plants», IEEE Trans. on Automatic Control, Vol. AC-32, pp. 459-463, 1987.

14. Proudfoot, C.G., P.J. Gawthrop, and O.L.R. Jacobs,

«Self-tuning PI control of a pH neutralisation process», Proc. Of IEE, Pt-D, Vol. 130, pp. 267-272,1983.

15. Yamamoto, T., S. Omatu, and T. Hotta, «A construction of self- tuning PID control algorithm and its application», Proc. of SICE, Japan, pp. 1143-1146,1983.

16. Rad, A.B. and PJ. Gawthrop, «Explicit PID self-tuning control for systems with unknown time-delay», Proc. of IFAC Int. Symposium on Intelligent Tuning and Adaptive Control, Singapore, 1991.

17. Cameron, F. and D.E. Seborg, «ASelf-tuning controller with a PID structure». Int. Journal of Control, Vol. 30, pp. 401-417, 1983.

18. Clarke, D.W. and PJ. Gawthrop, «Self-timing controller», Proc. of IEE, Pt-D, Vol. 122, pp. 929-934, 1975.

19. Clarke, D.W. and P.J. Gawthrop, «Self-tuning control», Proc. of IEE, Pt-D, Vol. 126, pp. 633-640, 1979.

20. Johansson, R., «Multivariable adaptive control», Dept. of Automatic Control, Lundinst. ofTech.. Report LUFTD27TRFT-1024/1-207, 1983.

21. Richalet, J.A., J.L.T. Rault, and J. Papon, «Model predictive heuristic control: applications to industrial processes», Automatica, Vol. 14, pp. 413-428, 1976.

22. Cutler, C.R. and B.C. Ramaker, «Dynamic matrix control - A computer control algorithm», JACC, Paper WP5-B, San Francisco, 1986.

23. De Keyser, R.M.C. and A.V. Cauwenberghe, «A self-tuning multi step predictor application», Proc. 6th IFAC Symposium on Identification and System Parameter Estimation, Washington, DC, pp. 1558-1563,1979.

24. Ydstie, B., «Extended horizon adaptive control», Proc. of 9th IFAC World Congress, Budapest, Hungary, 1984.

25. Peterka, V., «Predictor-based self-tuning control», Automatica, Vol. 20, pp. 39-50,1984.

26. Clarke, D.W., C. Mohtadi, and P.S. Tuffs, «Generalized predictive control - Part 1: The basic algorithm», Automatica, Vol. 23, pp. 137- 148,1987.

27. Clarke, D.W., C. Mohtadi, and P.S. Tuffs, «Generalized

predictive control - Part IL Extensions and interpretations», Automatica, Vol. 23, pp. 149-160,1987.

28. Clarke, D.W. and C. Mohtadi, «Properties of generalized predictive control», Automatica, Vol. 25, pp. 859-875, 1989.

29. De Keyser, R.M.C., PH.G.A. Van De Velde, and F.A.O. Dumartier, «A comparative study of self-adaptive long range predictive control methods», Automatica, Vol. 24, pp. 49-163, 1988.

30. Zadeh, L.A «Fuzzy sets». Information and Control. Vol. 8, pp.338-352,1965.

31. Söderström, T. and P. Stöica, «System Identification», Prentice- Hall, London. 1989.

32. Åström, KJ. and B. Wittenmark, «Adaptive Control.», Addison- Wesley, New York, 1989.

33. Bierman, G.J., «Factorization Methods for Discrete Sequential Estimation», Academic Press, New York, 1977.

34. Gawthrop, PJ., «Some interpretations of the self-tuning controller», Proc. Of IEE, Pt-D, Vol. 124, pp. 889-894. 1977.

35. Clarke, D.W., «Introduction to self-tuning controllers». In Harris and Billings (Eds.), Peter Pegrinus, (UK), pp. 36-71, 1981.

36. Clarke, D.W., A.J.F. Hodgson, and P.S. Tuffs, «Offset problem and K-incremental predictors in self-tuning control», Proc. of IEE, Pt-D, Vol. 130, pp. 217-225, 1983.

37. Tuff. P.S. and D.W. Clarke, «Self tuning control of offset-A unified approach». Proc. of IEE, Pt-D , Vol. 132, pp. 100-110, 1985.

38. Clarke, D.W., «Self-tuning control of non minimum-phase systems» Automatica, Vol. 20, pp. 501-517, 1984.

39. Clarke, D.W., P.P. Kanjilal, and C. Mohtadi, «A generalised LQG approach to self-tuning control, part 1: Aspects of design» Int. Journal of Control, Vol. 45, pp. 1509-1523, 1985.

40. Kailath, T. «Linear Systems», Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

41. Gantmacher, F.R., «The Theory of Matrices», Chelsea Publishing Company, New York, 1977.

42. Maciejowski, J.M., «Multivariable Feedback Design», Addison- Wesley, London, 1989.

43. Isermann, R., «Digital Control Systems: Vol. I and II». Springer Verlag, Berlin, 1989.

44. Kokotovic, P.V., R.EJr. O'Malley, and P. Sannuti, «Singular perturbations and order reduction in control theory - An overview», Automatica, Vol 12. pp. 123-132, 1976.

45. Mishra, R.N. and D.A. Wilson, «A New Algorithm for Optimal Reduction of Multivariable Systems», Int. Journal of Control, Vol. 31. pp. 443-466, 1980.

46. Borrison, U., «Self tuning for a class of multivariable systems», Automatica, Vol. 15, pp. 209-215, 1979.

47. Åström, K.J., U. Borrison. L. Ljung, and B. Wittenmark, «Theory and applications of self-tuning regulators», Automatica, Vol. 13, pp 457-476,1977.

48. Bitmead, R.R., M. Covers, and V. Wertz, «Adaptive Optimal Control». Prentice-Hall, Australia Pty Ltd., 1990.

49. Zadeh, L.A.. «Fuzzy sets and systems», Proc. of Symp. on System Theory. Polytechnic Institute of Brooklyn, pp.29-37, 1965.

50. Zadeh, L.A., «Outline to a new approach to the analysis complex systems and decision processes», IEEE Trans. on Systems, Man, and Cybernetics. Vol. 3, pp. 28-44, 1973.

51. Assilian, S. and E.H. Mamdani, «An experiment in linguistic synthesis with a fuzzy logic controller». Int. Journal of Man-Machine Studies, Vol. 7. pp.1-13, 1974.

52. Mamdani, E.H., «Applications of fuzzy algorithms for a simple dynamic plant», Proc. of IEE.. Vol. 121, pp. 1585-1588, 1974.

53. Mamdani, E.H , «Advances in the linguistic synthesis of fuzzy controllers», IEEE Trans. on Computer, Vol. C-26, pp.1182-1191, 1977.

54. King, P.J. and E.H. Mamdani, «The application of fuzzy control systems to industrial processes», Automatica, Vol. 13, pp. 235-242, 1977.

55. Mamdani, E.H., «Application of fuzzy set theory to control systems: a survey». Fuzzy Automata and Decision Proceses (Gupta, M.M., G.N. Sardis, and B.R. Gaines, Eds.), North-Holland. 1977.

56. Tong, R. M. «A control engineering review of fuzzy systems», Automatica, Vol. 13. pp. 559-569.1977.

57. Wiener, N., «Cybernetics: on control and communication in the animal and the machine», MIT Press, Cambridge, MA, 1948.

58. Togai, M. and H. Watanabe, «Expert system on a chip: an engine for real-time approximate reasoning», IEEE Expert, Vol. 1, pp. 55-62,

59. Yamakawa, T. «Fuzzy microprocessors - rule chip and defuzzification chip» Proc. of Int. Workshop on Fuzzy Systems Applications, Iizuka-88, Japan, pp. 51-52, 1988.

60. Kuo, Y.H., C.I. Kao, and JJ. Chen, «A fuzzy neural network model and its hardware implementation», IEEE Trans. on Fuzzy Systems, Vol. 1, pp. 171-183, 1993.

61. Dubois, D. and H. Prade, «Fuzzy Sets and Systems: Theory and Applications», Academic Press, New York, 1980.

62. Klir, G. and T. Folger, «Fuzzy Sets: Uncertainty and Information», Prentice-Hall, Engelwood Cliffs, New Jersey, 1988.

63. Zimmermann, H., «Fuzzy Set Theory and its Applications», 2nd. Ed., Kluwer, Boston, 1990.

64. Kosko, B., «Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence», Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

65. Pedrycz, W. «Fuzzy Control and Fuzzy Systems», 2nd Ed., Wiley, Somerset, 1993.

66. Pacini, P.J. and B. Kosko «Comparison of fuzzy and Kalman-filter target-tracking control systems», in Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, Prentice-Hall, Englewood Cliffs, New Jersey, pp. 379-406, 1991.

67. Kong, S.G. and B. Kosko, «Adaptive fuzzy systems for backing up a truck-and-trailer», IEEE Trans. on Neural Networks, Vol. 3, pp. 211- 223,1992.

68. Langari, R. and H.L Berenji «Fuzzy logic in control engineering», in Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. (White, D.A. and D.A. Sofge, Eds.), Van Nostrand Reinhold, New York, 1992.

69. Procyk, T.J. and E.H. Mamdani, «A linguistic self-organizing process controller», Automatica, Vol. 15, pp. 15-30, 1979.

70. Daley, S. and K.F. Gill, «A design study of a self-organizing fuzzy logic controller», Proc. of 1. Mech. E, Vol. 200, (CI), pp.59-69, 1986.

71. Linkens, D.A. and M.F. Abbod, «Self-organizing fuzzy logic control and the selection of its scaling factors», Trans. on Instrumentation, Measurement and Control, Vol. 14, pp. 114-125, 1992.

Литература к главе 4

1. Åström, K.J. and B. Wittenmark, «Adaptive Control,» Addison- Wesley, New York. 1989.

2. Box, G.E. and G.H. Jenkins, «Time Series Analysis: Forecasting and Control», Holden-Day, San Francisco, 1976.

3. Kosko, B., «Neural Networks for Signal Processing», Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

4. Nozaka, Y., «Trend of new control theory application in industrial process control (A survey)», Proc. of 12th IFAC World Congress, Sydney, Vol. VI, pp. 51-56, 1993.

5. Åström, K.J., «Towards intelligent control», IEEE Control Systems Magazine, Vol. 9, pp.60-69, 1989.

6. White, D.A. and D.A. Sofge, «Handbook of Intelligent Control», Van Nostrand Reinhold, New York, 1992.

7. Wiener, N., «Cybernetics or Control and Communication in the Animal and the Machine», MIT Press, Cambridge, Massachusetts, 1948.

8. Cybenko, G., «Approximation by superpositions of a sigmoidal function». Mathematics Control, Signal & System, Vol. 2, pp.303- 314,1989.

9. Funahashi, K.I., «On the approximate realization of continuous mappings by neural networks». Neural Networks, Vol. 2, pp.183-192, 1989.

10. Hornik, K., M. Stinchcombe, and H. White, «Multilayer feedforward networks are universal approximators». Neural Networks, Vol. 2, pp. 359-366,1989.

11. Saerens, M. and A. Soquet, «A neural controller based on backpropagation algorithm», Proc. of First IEE Int. Conf. on Artificial Neural Networks, London, pp. 211-215, 1989,

12. Ogata, K., «Discre-time Control Systems», Prentice-Hall, Engiswood Cliffs, New Jersey, 1987.

13. Iiguni, Y, H. Sakai, and H. Tokumaru, «A non-linear regulator design in the presence of system uncertainties using multilayered neural networks,» IEEE Trans. on Neural Networks, Vol. 2, pp. 410-417, 1991.

14. Narendna, K.S. and K. Parthasarathy, «Identification and control of dynamical systems using neural networks», IEEE Trans. or. Neural Networks, Vol. 1, 'pp. 4-27, 1990.

15. Neuralogix NLX420 Datasheet, American Neuralogix Inc., 1992.

16. Levin, E., R. Gewirtzman, and G.F. Inbar, «Neural network architecture for adaptive system modelling and control», Proc. of Int. Joint Conf. on Neural Networks. Washington D.C. Vol. 11. pp. 311- 316,1989.

17. Kosko, B., «Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence», Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

18. Miller, W.T., R.S. Sutton, and P.J. Werbos, «Neural Networks for Control», MIT Press, Cambridge, MA, 1990.

19. Barto, A.G. «Connectionist learning for control», in Neural Networks for Control, MIT Press, Cambridge, Massachusetts, pp. 5-58, 1990.

20. Werbos, P.J., «Overview of designs and capabilities», in Neural Networks for Control, MIT Press, Cambridge, MA, pp. 59-65, 1990.

21. Hunt, K.J., D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop, «Neural networks for control systems - a survey», Aulomatica, Vol. 28, pp. 1083-1112. 1992.

22. Widrow, B. and F.W. Smith, «Pattern-recognizing control systems», Proc. of Computer and Information Sciences, Washington D.C» Spartan, Washington, 1964.

23. Albus, J.S., «A new approach in manipulator control: the cerebellar model articulation controller (CMAC)», Journal of Dynamic Systems, Measurement and Control, pp. 220-227, 1975.

24. Albus, J.S., «Data storage in the cerebellar model articulation controller». Journal of Dynamic Systems, Measurement and Control, pp. 228-233,1975.

25. Miller III, W.T., F.H. Glanz, and L.G. Kraft, «CMAC: An associative neural network alternative to backpropagation», Proc. of IEEE, Vol. 78, pp. 1561-1567, 1990.

26. Barto, A.G., R.S. Sutton, and C.W. Anderson, «Neuronlike adaptive elements that can solve difficult learning control problems», IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-13, pp-834-846, 1983.

27. Gaines, B.R. and J.H. Andreae, «A learning machine in the context of the general control problem», Proc. of 3rd IFAC Congress, IME, London, pp. 14B.I-14B.8,1966.

28. Witten, I.H., «An adaptive optimal controller for discrete-time Markov enviroments», Information and Control, Vol. 34, pp. 286-295, 1997.

29. Werbos, P.J., «Advanced forecasting methods for global crisis warning and models of intelligence». General System Yearbook, Vol. 22, pp.25-38,1977.

30. Holland, J.H., «Escaping brittleness: The possiblity of general-purpose learning algorithm algorithms applied to rule-based systems», R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Eds., Machine Learning: An Artificial Intelligence Approach, Vol. II, Morgan Kaufmann, San Mateo, CA, 1986.

31. Anderson, C.W., «Learning and problem solving with multilayer connectionist systems», Ph.D. Thesis, University of Mass., Amherst, 1986.

32. Anderson, C.W., «Strategy learning with multilayer con-

nectionist representations», Tech. Rep. TR87-509.3, GTE Laboratories Inc., Waltham, MA, 1987.

33. Widrow, B. and S. D. Steains, «Adaptive Signal Processing», Prentice- Hall, Englewood Cliffs, New Jersey, 1985.

34. Kuperstein, M., «Neural model of adaptive hand-eye co-ordination for single postures». Science, Vol. 239, pp. 1308-1311, 1988.

35. Kuperstein, M., and J. Rubinstein, «Implementation of an adaptive neural controller for sensory-motor co-ordination», IEEE Control Systems Magazine, Vol. 9, pp. 25-30, 1989.

36. Srinivasan, V., A.C., Barto and B.E. Ydstie «Pattern recognition and feedback via parallel distributed processing». Annual Meeting of the American Institute of Chemical Engineers, Washington, D.C» 1988.

37. Sanner, R.M. and D.L. Akin, «Neuromorphic pitch attitude regulation of an underwater telerobot», IEEE Control Systems Magazine, Vol. 10, pp.62-68.1990.

38. Psaltis D., A. Sideris, and A. Yamamura, «A Multilayered neural network controller», IEEE Control Systems Magazine, Vol. 8, pp.17- 21,1988.

39. Saerens, M. and A. Soquet, «A neural controller based on backpropagation algorithm», Proc. of First IEE Int. Conf. on Artificial Neural Networks, London, pp. 211-215, 1989.

40. Kawato, M., K. Furukawa and R. Suzuki, «A hierarchical neural network model for control and learning of voluntary movement», Biological Cybernetics, Vol. 57, pp. 169-185, 1987.

41. Kawato, M., Y. Uno, M. Isobe, and R. Suzuki, «Hierarchical neural network model for voluntary movement with application to robotics», IEEE Control Systems Magazine. Vol. 8, pp.8-16, 1988.

42. Jordan, M.I., «Generic constraints on underspecified target trajectories», Proc. of Int. Joint Conf. on Neural Networks (IJCNN)' 89, Washington, Vol. 1, pp. 217-225, 1989.

43. Jordan, M.I. and D.E. Rumelhart, «Forward models: Supervised learning with a distal teacher», Cognitive Science, Vol. 16, pp.313- 355,1990.

44. Narendra, K.S. and K Parthasarathy, «Identification and control of dynamical systems using neural networks», IEEE Trans. on Neural Networks, Vol. I, pp. 4-27, 1990.

45. Nguyen, D.H. and B. Widrow, «Neural networks for self-learning control systems», IEEE Control Systems Magazine, Vol. 10, pp. 18-23, 1990.

46. Chen, S., S.A. Billings, and P.M. Grant, «Nonlinear system identification using neural networks». Int. Journal of Control, Vol. 51, pp. 1215-1228,1990.

47. Chen, S., S.A. Billings, C.E. Cowan and P.M. Grant, «Practical identification of NARMAX models using radial basis function». Int. Journal of Control, Vol. 52, pp. 1327-1350, 1990.

48. Bhat, N.V., Jr., P.A. Minderman. T. McAvoy, and N.S., Wang, «Modeling chemical process systems via neural computation», IEEE Control Systems Magazine, Vol. 10, pp. 24-30, 1990.

49. Chu, S.R., R. Shoureshi, and M. Tenorio, «Neural networks for system identification», IEEE Control Systems Magazine, Vol. 10, pp. 31-35, 1990.

50. Billings, S.A., H.B. Jamaludin, and S. Chen, «Properties of neural networks with applications to modeling of nonlinear dynamical systems». Int. Journal of Control, Vol. 55, pp. 193-224, 1992.

51. Sjoberg, J. and L. Ljung, «Overtraining, regularization, and searching for minimum in neural networks», Revised Version Report Lith-ISY-I- 1297, Dept. of Elec. Eng., Linkoping University. Sweden, 1992.

52. Tanomaru, J. and S. Omatu, «Process control by on-line trained neural controllers», IEEE Trans. on Industrial Electronics, Vol. 39, pp. 511- 521,1992.

53. Hopfield, JJ., «Neural networks and physical systems with emergent computational abilities», Proc. of the National Academy of Sciences, Vol. 79, pp. 2554-2558, 1982.

54. Lippmann, R., «An introduction to computing with neural nets» IEEE ASSP Magazine, Vol. 4, pp.4-22, 1987.

55. Gomi, H. and M. Kawato, «Neural network control for a

closed loop system using feedback error learning», Neural Networks, Vol. 6, pp. 933-946,1993.

56. Nagata, S., M. Sekiguchi, and X. Asakawa, «Mobile robot control by a structured hierarchical neural network», IEEE Control Systems Magazine, Vol. 10, pp. 69-76, 1990.

57. Kinjo, H., S. Omatu, T. Yamamoto, and S. Tamaki, «Suboptimal control for a non-linear system using neural networks», Proc. of 1st Asian Control Conference, pp. 551-554, Tokyo, 1994.

58. Saiful A. and S. Omatu, «Neuromorphic self-tuning PID controller», Proc. of 1993 IEEE ICNN. San Francisco, pp. 552-557, 1993.

Литература к главе 5

1. Åström, K.J. and B. Wittenmark, «Adaptive Control», Addison Wesley, USA, 1989.

2. Narendra, K.S., R. Ortega, and P. Dorato, Eds., «Advances in Adaptive Control», IEEE Press, New York, 1991.

3. «Special Issue on Neural Network Hardware», IEEE Trans on Neural Networks , Vol. 3, pp. 347-506, 1992.

4. Togai, M. and H. Watanabe, «Expert system on a chip: An engine for real-time Approximate Reasoning», IEEE Expert, Vol. I, pp. 55-62, 1986.

5. Yamakawa, T., «Fuzzy microprocessors - rule chip and defuzzification chip», Proc. of Int. Workshop on Fuzzy Systems Applications, Iizuka-88, Kyushu, pp. 51-52, 1988.

6. Kong, S.G. and B. Kosko, «Adaptive fuzzy systems for backing up a truck-and-trailer», IEEE Trans. on Neural Networks .Vol. 3, pp. 211- 223,1992.

7. Kraft, L.G., and D.P. Campagna, «A comparison between CMAC neural network control and two traditional adaptive control systems», IEEE Control Systems Magazine. Vol. 10, pp. 36-43, 1990.

8. Narendra, K.S. and K. Parthasarathy, «Identification and control of dynamical systems using neural networks», IEEE Trans. on Neural Networks, Vol. 1, pp. 4-27, 1990.

9. Nguyen, D.H., and B. Widrow, «Neural networks for self-

learning control systems», IEEE Control Systems Magazine. Vol. 10, pp. 18-23.1990.

10. Jordan, M.I. and D.E. Rumelhart, «Forward models: supervised learning with a distal teacher». Cognitive Science, Vol. 16, pp. 313- 355,1992.

11. Moore, C.G. and C.J. Harris, «Indirect adaptive fuzzy control», Int. Journal of Control., Vol. 56, pp. 441-468, 1992.

12. Procyk, TJ. and E.H. Mamdani, «A Linguistic self-organizing process controller», Automatica, Vol. 15, pp. 15-30, 1979.

13. Berenji, H.R. and P. Khedkar, «Learning and tuning fuzzy logic controllers through reinforcements», IEEE Trans. on Neural Networks, Vol. 3, pp. 724-740, 1992.

14. Clarke, D.W., C. Mohtadi and P.S. Tuffs, «Generalized predictive control-part 1: The basic algorithm», Automatica, Vol. 23, pp. 137-148, 1987.

15. Clarke, D.W., C. Mohtadi, and P. S. Tuffs, «Generalized predictive control-part II: Extensions and interpretations», Automatica, Vol. 23, pp.149-160,1987.

16. Widrow, B. and S.D. Steams, «Adaptive Signal Processing», Prentice Hall, Englewood Cliffs, New Jersey, 1985.

17. Psaltis, D., A. Sideris, and A. Yamamura, «A multilayered neural network controller», IEEE Control Systems Magazine, Vol. 10, pp.44- 48,1988.

18. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, «Learning internal representations by error propagation», in Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol.I, MIT Press, Cambridge, MA, pp. 318-362, 1986.

19. Khalid, M. and S. Omatu, «A neural network controller for a temperature control system», IEEE Control Systems Magazine, Vol. 12, pp.58-64,1992.

20. Yusof, R. and S. Omatu, «Application of generalized predictive control to a temperature control process» Proc. of IECON '91, Kobe, Japan, Vol. 3, pp. 1935-1940, 1991.

21. Kuperstein, M., «Neural model of adaptive hand-eye coordination for single postures». Science, Vol. 239, pp. 1308-1311. 1988.

22. Nagata, S., M. Sekiguchi, and K. Asakawa, «Mobile robot control by a structured hierarchical neural network», IEEE Control Systems Magazine, Vol. 10, pp. 69-76. 1990.

23. Zadeh, L.A., «Fuzzy sets». Information and Control, Vol. 8, pp. 338- 352,1965.

24. Zadeh, L.A., «Outline of a new approach to the analysis of complex systems and decision processes», IEEE Trans. on Systems, Man, and Cybernetics, Vol. 3, pp.28-44, 1973.

25. Tong, R.M., «A control engineering review of fuzzy systems», Automatica, Vol. 13, pp. 559-569, 1977.

26. King P.J., and E.H. Mamdani, «The application of fuzzy control systems to industrial processes», Automatica, Vol. 13, pp. 235-242, 1977.

27. Assilian, S. and E.H. Mamdani, «An experiment in linguistic synthesis with a fuzzy logic controller». Int. Journal of Man-Machine Studies, Vol. 7, pp.l-13, 1974.

28. Kosko, B., «Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence», Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

29. Linkens, D.A. and M.F. Abbod, «Self-organizing fuzzy logic control and the selection of its scaling factors», Trans. on Instrumentation, Measurement and Control. VoL 14, pp. 114-125, 1992.

30. Daley, S. and K.F. Gill, «A design study of a self-organizing fuzzy logic controller», Proc. of I Mech E. Vol. 200, (CI), pp.59-69, 1986.

31. Takahashi, Y., C.S. Chan, and D.M. Auslander, «Parametereinstellung bei linearen DDC-algorithmen», Regelungstechnik und Prozeß- Datenverarbeitung, Vol. 19, pp.237-244, 1971.

32. Levin, E., R. Gewirtzman and G.F. Inbar, «Neural network architecture for adaptive system modelling and control», Proc. of Int. Joint Conference on Neural Networks (IJCNN) '89, Washington D.C., Vol. II, pp. 311-316, 1989.

33. Ogata, K., «Modern Control Systems», Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

34. Miller III, W.T., R.S. Sutton, and P.J. Werbos, Eds., «Neural Networks for Control», MIT Press, Cambridge, MA, 1990.

35. Saerens, M. and A. Soquet. «A neural controller based on backpropagation algorithm», Proc. of First IEE Int. Conf. on Artificial Neural Networks, London, pp.211-215, 1989.

36. Yusof, R., S. Omatu, and M. Khalid, «Self-tuning PID control: A multivariable derivation and application», Automatica (to be published).

37. Yusof, R., «Theory and applications of self-tuning PID control and generalized predictive control», Ph.D. Thesis, University of Tokushima, Japan, 1994.

ЛИСТИНГ ПРОГРАММЫ

```
/* Эта программа может использоваться для обучения */
/* нейросетевой модели с одним скрытым слоем обратной */
/* динамики объекта. */
/* Обученная модель может использоваться как прямой */
/* регулятор для объекта. */
/* Характеристики входа-выхода объекта должны быть */
/* сгенерированы и храниться в файле с именем in-out.dat. */
/* Эта программа выполняется в среде Borland C++ 3.1. */

/* Backpropagation Neuro-Control*/
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* Declare the total number of nodes and pattern */
#define No_of_Input_Units    3   /* No. of nodes of input layer i */
#define No_of_Hidden_Units  15   /* No. of nodes of hidden layer j */
#define No_of_Output_Units   1   /* No. of nodes of output layer k */
#define No_of_Pattems        8   /* Total no. of patterns p */

#define Eta          0.0001 /* The learning rate */
#define Alpha        0.8    /* Momentum constant */
#define Beta         -0.15  /* Acceleration constant */
#define ErrorFunc    0.001  /* Maximum error allowed */
#define Wmin         -0.05  /* For use in random function */
#define Wmax         0.05   /* to generate random values */
#define No_of_Data 101      /*Total number of plant input-output data */

/* Functions */
#define f(x)  (1/(1+exp(-(x))))
              /* Sigmoid function where x = net+ bias */
#define rnd( )((float)rand( )0x7fff * (Wmax-Wmin) + Wmin)
              /* This function is used to generate random values */

/* Declare outputs, weights, and biases */
float   o1[No_of_Pattems][No_of_Input_Units];
        /* Output of Input layer i */
float   o2[No_of_Hidden_Units];    /* Output of Hidden laye rj */
float   o3[No_of_Output_Units];    /* Output of Output laye k */
```

```
float   t[No_of_Pattems][No_of_Output_Units];
        /* Desired Output or target value */
float   w21 [No_of_Hidden_Units][No_of_Input_Units];
        /* Weights from i to j layer */
float   dw21[No_of_Hidden_Units][No_of_Input_Units][3];
        /* Change in the above weights */
float   w32[No_of_Output_Units][No_of_Hidden_Units];
        /* Weights from j to k layer */
float   dw32[No_of_Output_Units][No_of_Hidden_Units][3];
        /* Change in the above weights */
float   bias2[No_of_Hidden_Units];
        /* Bias to nodes in layer j (theta j) */
float   dbias2[No_of_Hidden_Units][3];
        /* Change in the above bias j */
float   bias3[No_of_Output_Units];
        /* Bias to nodes in layer k (theta k) */
float   dbias3[No_of_Output_Units][3];
        /* Change in the above bias j */
float   increase, erorfunc, er[3], u[No_of_Data], y(No_of_Data];
float   Errorfunction
int     count, iterations;

/* Main Program */
void main ()
/*begin main program*/
{
  unsigned long              i,j,k;
  static float  errorfunc;
  void          propagation(int);
  void           backpropagation(int);
  void          forward(int);
  void          read_input-output_data( );
  void          initialize( );
  void          save_weight( );
  void          NN_inputs( );
  initialize( );
  read_input-output_data( );
  increase=0.0;
  for (errorfunc = 0.0, i=0; i<No_of_Patterns; i++)
    {
        forward(i);
        for(j=0; j<No_of_Output_Units; j++)
        {
       errorfunc += pow(t[i][j] - o3[j], 2.0);
```

```
        printf("Eiror Function is: %.5f\n", errorfunc);
        /* error is the square of target - output */
    }
  }
errorfunc /=2;
        erorfunc = errorfunc;
        printf("Error Value is: %.3f\n", errorfunc);
        /* print the error value */
        er[0]=errorfunc;
for( i=0; errorfunc>ErrorFunc;)
    {
        for( j=0; j<No_of_Patterns; j++ )
        {
            propagation(j);
            back_propagation(j);
        }
        for( errorfunc=0.0,j=0;j<No_of_Patterns; j++)
            {
                forward(j);
                for(k=0; k<No_of_Output_Units; k++)
                errorfunc += pow(t[j][k] - o3[k], 2.0);
        }
      iterations=1; ++i;
      increase+=(float)iterations;
      errorfunc/= 2; erorfunc=errorfunc;
    {
        printf(" I= %f ", increase);
        printf("E = %.7f\n", errorfunc);
                er[1]=er[0]; er[0]=errorfunc;
        if(er[1]<er[0]) break;
    }
  }
      printf("\n After Learning \n");
      for( i=0; i<No_of_Patterns; i++)
     forward(i);
     save_weight();
};


/* To save the adapted weights in a data file */
void    save_weight()
{
  int      i,j;
  FILE *fp;
```

```
fp = fopen("weight.dat","w");
        /* To save the weights in layer j */
        for( i=0; i<No_of_Hidden_Units; i++)
             for( j=0; j<No_of_Input_Units; j++)
        {
                    fprintf(fp,"%f\n",w21[i][j]);
                    fprintf(fp,"%f\n",bias2[i]);
        }
        /* To save the weights in layer k */
        for( i=0; i<No_of_Output_Units; i++)
        for( j=0; j<No_of_Hidden_Units; j++)
        {
                    fprintf(fp,"%f\n",w32[i][j]);
                    fprintf(fp,"%f\n",bias3[i]);
        }
      fclose(fp);
}


/* Procedure Propagation */
/* To propagate the network forward */
void    propagation(int p)
 {
        int     i,j;
        float   net;
for( i=0; i<No_of_Hidden_Units; /* layer j */ i++)
     {      for( net=0.0, j=0; j<No_of_Input_Units; j++)
              net+=w21[i]0]*o1[p][j];
              o2[i] = f(net+bias2[i]);  /* sigmoid function */
    }
for( i=0; i<No_of_Output_Units; /* layer k */ i++)
    {
       for( net=0.0, j=0; j<No_of_Hidden_Units; j++)
      net+=w32[i][j]*o2[j];
      o3[i] = (net+bias3[i]);
       /* output of NN not bounded by sigmoid function */
  }
}


/* Procedure Back Propagation */
/* To backpropagate error through the network */
void   back_propagation (int p)
{
```

```
int  i,j;
float d2[No_of_Hidden_Units];  /*d2 is delta j*/
float d3[No_of_Output_Units];  /*d3 is delta k*/
float sum;          /* sum is add. of delta k * Wkj */

/* to calculate delta k */
for( i=0; i < No_of_Output_Units; i++ )
  d3[i]=(t[p][i]-o3[i]);
for( i=0; i<No_of_Hidden_Units; i++)
 {
    /* to calculate the weights, w and dw, between layer j and k */
    for( sum=0.0,j=0; j<No_of_Output_Units; j++ )
    {
      dw32[j][i][0] = Eta*d3[j]*o2[i]
              + Alpha*dw32[j][i][1]+Beta*dw32[j][i][2];
      w32[j][i]+=dw32[j][i][0];
      sum+=d3[j]*w32[j][i];
      dw32[j][i][2]=dw32[j][i][1];
      dw32[j][i][1] = dw32[j][i][0];
    }
   /* to calculate delta j */
   d2[i] = o2[i]*(1 - o2[i])*sum;
 }
/* to calculate bias k */
for( i=0; i<No_of_Output_Units; i++)
 {
    dbias3[i][0] = Eta*d3[i]+Alpha*dbias3[i][1]
            + Beta * dbias3[i][2];
    bias3[i] += dbias3[i][0];
    dbias3[i][2] = dbias3[i][1];
    dbias3[i][1] = dbias3[i][0];
 }

/* to calculate the weights, w and dw, between layer j and i */
for( i=0; i<No_of_Input_Units; i++)
  for( j=0; j<No_of_Hidden_Units; j++ )
   {
     dw21[j][i][0] = Eta * d2[j] * o1[p][i]+ Alpha
              * dw21[j][i][1] +Bcta * dw21[j][i][2];
     w21[j][i]+=dw21[j](i][0];
     dw21[j][i][2]=dw21[j][i][1];
     dw21[j][i][1]=dw21[j][i][0];
   }
```

```
  /* to calculate bias j */
  for ( i=0; i<No_of_Hidden_Units; i++ )
   {
     dbias2[i][0] = Eta*d2[i]+Alpha*dbias2[i][1]
              +Beta*dbias2[i][2];
     bias2[i] += dbias2[i][0];
     dbias2[i][2] = dbias2[i][1];
     dbias2[i][1]=dbias2[i][0];
   }
}

/* Procedure to read input-output characteristics of systems */
void read_input-output_data( )
{
  int  j,k;
  FILE *file;
    file=fopen("in-out.dat","r");
     /* this file must be generated which consists
        of plant input-output characteristics */

  for(j=0; j<No_of_Data; j++)
    fscanf(file,"%f %f\n",&u[j],&y[j]);

  for(j=0-, j < No_of_Data; j++)
    printf("u[%d]=%f y[%d]=%f\n",j,u[j],j,y[j]);
  fclose(file);
}

/* Procedure to equate NN inputs */
void NN_inputs()
  /* to select the patterns over entire
     input-output plant characteristics */
{
  int    i,j,k;
  for(i=0,j=0; j < No_of_Pattems; j++)
   {
     for(k=0; k < No_of_Output_Units; k++)
      t[j][k]=u[i+2];
   }
  for(j=0; j < No_of_Patterns; j++)
    for(k=O; k <No_of_Output_Units; k++)
      printf("T[%d][%d]=%f\n",j,k,t[j][k]);

  for(i=0, j=0; j < No_of_Patterns; j++)
```

```
        /*example for 3 NN input vectors */
  {
    o1[j][0]=y[i+2];
    o1[j][1]=y[i+1];
    o1[j][2]=y[i];
   }
 for(j=0; j < No_of_Patterns; j++)
   for(k=0; k <No_of_Input_Units; k++)
     printf("o1[%d][%d]=%f\n",j,k,o1[j][k]);
}


/* Procedure forward */
void forward(int p)
{
 int  i;

 printf("\t%d ->",p+1);
 propagation(p);
 for( i=0; i<No_of_Output_Units; i++)
   printf("o3[%d]=%2.5f\n",p+1,o3[i]);
}


/* Procedure Initialize */
/* To randomly initialize the weights */
void   initialize( )
{
 int   ij;

 /* To initialize the weights between layer j and i */
 for ( i=0; i<No_of_Hidden_Units; i++)
  {
    for(j=0;j<No_of_Input_Units;j++)
     w21[i][j]=rnd();
    bias2[i] = 1.0;
  }
 /* To initialize the weights between layer k and j */
 for ( i=0; i<No_of_Output_Units; i++)
  {
    for( j=0; j<No_of_Hidden_Units; j++)
     w32[i][j]=rnd();
    bias3[i]=1.0;
  }
}
```

ИПРЖР принимает заказы на серию книг, объединенных общим названием «Нейрокомпьютеры и их применение», под редакцией докт. техн. наук, проф. А. И. Галушкина:

Книга 1. А.И. Галушкин. Теория нейронных сетей

Книга 2. Сигеру Омату, Марзуки Халид, Рубия Юсоф. Нейроуправление и его приложения (пер. с англ.)

Книга 3. А.И. Галушкин. Нейрокомпьютеры

Книга 4. История развития нейрокомпьютеров. Сб статей

Книга 5. Теория нейронных сетей. Сб. статей

Книга 6. Нейроматематика. Сб. статей

Книга 7. Нейрокомпьютеры 90-х. Сб. статей

Книга 8. Применение нейрокомпьютеров Сб. статей

Книга 9. В А. Терехов. Нейросетевые системы управления

Книга 10. В. А. Головко. Нейроинтеллект: теория и применение

Книга 11. Нейронные сети. STATISTICA Neural Networks (пер с англ. StatSoft Russia)

Книга 12. Э.Д. Аведьян. Алгоритмы обучения нейронных сетей

Книга 13. А.Ю. Дорогов. Теория модульных нейронных сетей

Книга 14. А. Н. Балухто, А.И. Галушкин, Л.Е. Назаров.
Нейрокомпьютеры в системах обработки изображений

Книга 15. А.И. Галушкин. Применение нейрокомпьютеров

Книга 16. А.А. Фролов, И.П. Муравьев. Ассоциативная память. Нейросетевой поход

Книга 17. Ю.И. Зозуля. Интеллектуальные нейросистемы

Книга 18. Р. Рохас. Нейронные сети: систематизированное введение (пер. с англ.)

Книга 19. А.И. Галушкин, Д.В. Ковальчук, Л.Н. Никифорова.
Нейрокомпьютеры в вертолетах

Книга 20. Н.П. Абовский. Нейроуправляемые конструкции и системы

Книга 21. А.А. Фролов, Е.В. Бирюкова, Р.А. Прокопенко.
Нейросетевое управление антропоморфными роботами и манипуляторами

Книга 22. Э.Ю. Кирсанов. Системотехническое проектирование высокопроизводительных цифровых нейрокомпьютеров с параллельной структурой

Книга 23. А.И. Галушкин, В.А. Шахнов, А.И. Власов. Нейропроцессорные системы
активного гашения промышленных вибраций

Книга 24. А.И. Власов, Г.Л. Яковлева, В.Л. Яковлев. Применение нейросетевых методов в информационных и аналитических системах

Книга 25. А.И. Власов Автоматизированное проектирование нейроадаптивных систем
активного управления волновыми полями

Книга 26. Нейрокомпьютеры в космической технике (колл. авторов)

Книга 27. Б.Г Ильясов. Нейрокомпьютеры в авиации

Книга 28. В.И. Горбаченко. Применение нейрокомпьютеров для решения краевых задач теории поля

Вышла в свет первая книга этой серии «Теория нейронных сетей».