目　录

第1章 绪 论

1.1 智能控制的发展过程

1. 智能控制的提出

传统控制方法包括经典控制和现代控制，是基于被控对象精确模型的控制方式，缺乏灵活性和应变能力，适于解决线性、时不变性等相对简单的控制问题。传统控制方法在实际应用中遇到很多难以解决的问题，主要表现在以下几点：

(1)实际系统由于存在复杂性、非线性、时变性、不确定性和不完全性等，无法获得精确的数学模型；

(2)某些复杂的和包含不确定性的控制过程无法用传统的数学模型来描述，即无法解决建模问题；

(3)针对实际系统往往需要进行一些比较苛刻的线性化假设，而这些假设往往与实际系统不符合；

(4)实际控制任务复杂，而传统的控制任务要求低，对复杂的控制任务如智能机器人控制、CIMS、社会经济管理系统等无能为力。

在生产实践中，复杂控制问题可通过熟练操作人员的经验和控制理论相结合去解决，由此产生了智能控制。智能控制将控制理论的方法和人工智能技术灵活地结合起来，其控制方法适应对象的复杂性和不确定性。

智能控制是控制理论发展的高级阶段，它主要用来解决那些用传统控制方法难以解决的复杂系统的控制问题。智能控制研究对象具备以下一些特点：

(1)不确定性的模型。智能控制适合于不确定性对象的控制，其不确定性包括两层意思：一是模型未知或知之甚少；二是模型的结构和参数可能在很大范围内变化。

(2)高度的非线性。采用智能控制方法可以较好地解决非线性系统的控制问题。

(3)复杂的任务要求。例如，智能机器人要求控制系统对一个复杂的任务具有自行规划和决策的能力，有自动躲避障碍运动到期望目标位置的能力。再如，在复杂的工业过程控制系统中，除了要求对各被控物理量实现定值调节外，还要求能实现整个系统的自动启停、故障的自动诊断及紧急情况下的自动处理等功能。

2. 智能控制的概念

智能控制是一门交叉学科，著名美籍华人傅京逊教授 1971 年首先提出智能控制是人工智能与自动控制的交叉，即二元论。美国学者 G. N. Saridis1977 年在此基础上引入运筹学，提出了三元论的智能控制概念，即

$$IC = AC \cap AI \cap OR$$

式中各子集的含义为：IC 为智能控制(Intelligent Control)；AI 为人工智能(Artificial Intelli-

gence)；AC 为自动控制(Automatic Control)；OR 为运筹学(Operational Research)。基于三元论的智能控制如图 1-1 所示。

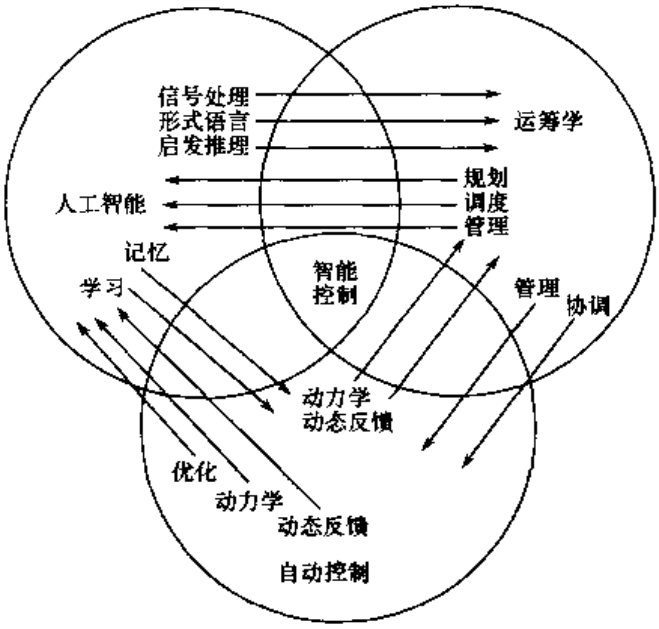


图 1-1 基于三元论的智能控制

人工智能(AI)是一个用来模拟人思维的知识处理系统，具有记忆、学习、信息处理、形式语言、启发推理等功能。

自动控制(AC)描述系统的动力学特性，是一种动态反馈。

运筹学(OR)是一种定量优化方法，如线性规划、网络规划、调度、管理、优化决策和多目标优化方法等。

三元论除了“智能”与“控制”外，还强调了更高层次控制中调度、规划和管理的作用，为递阶智能控制提供了理论依据。

所谓智能控制，即设计一个控制器(或系统)，使之具有学习、抽象、推理、决策等功能，并能根据环境(包括被控对象或被控过程)信息的变化做出适应性反应，从而实现由人来完成的任务。

3. 智能控制的发展

智能控制是自动控制发展的最新阶段，主要用于解决传统控制难以解决的复杂系统的控制问题。控制科学的发展过程如图 1-2 所示。

从 20 世纪 60 年代起，由于空间技术、计算机技术及人工智能技术的发展，控制界学者在研究自组织、自学习控制的基础上，为了提高控制系统的自学习能力，开始注意将人工智能技术与方法应用于控制中。

1966 年，J. M. Mendal 首先提出将人工智能技术应用于飞船控制系统的设计；1971 年，傅京逊首次提出智能控制这一概念，并归纳了 3 种类型的智能控制系统。

(1)人作为控制器的控制系统：人作为控制器的控制系统具有自学习、自适应和自组织的功能。

(2)人机结合作为控制器的控制系统：机器完成需要连续进行的并需快速计算的常规控制

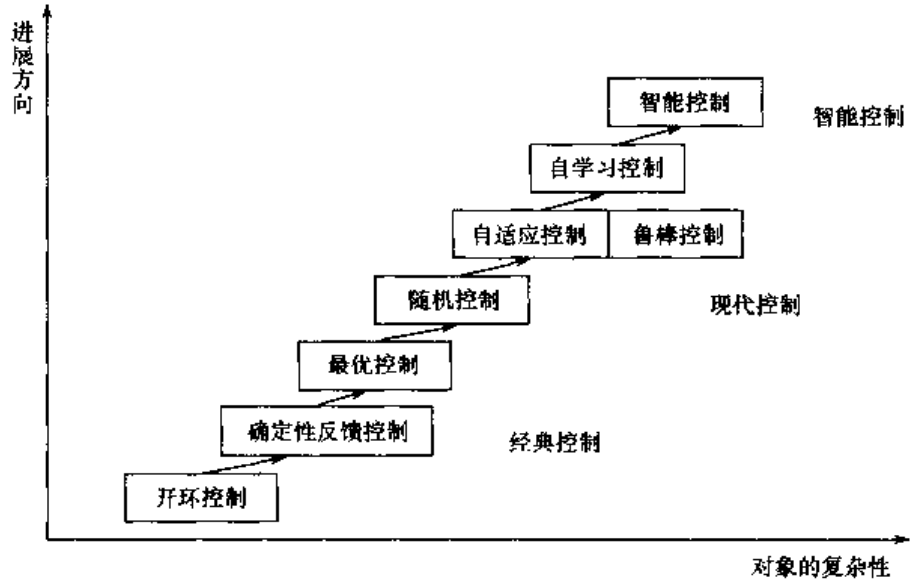


图 1-2　控制科学的发展过程

任务，人则完成任务分配、决策、监控等任务。

(3)无人参与的自主控制系统：为多层的智能控制系统，需要完成问题求解和规划、环境建模、传感器信息分析和低层的反馈控制任务，如自主机器人。

1985 年 8 月，IEEE 在美国纽约召开了第一届智能控制学术讨论会，随后成立了 IEEE 智能控制专业委员会；1987 年 1 月，在美国举行第一次国际智能控制大会，标志着智能控制领域的形成。

近年来，神经网络、模糊数学、专家系统、进化论等各门学科的发展给智能控制注入了巨大的活力，由此产生了各种智能控制方法。智能控制的几个重要分支为专家控制、模糊控制、神经网络控制和遗传算法。

1.2　智能控制的几个重要分支

1. 模糊控制

以往的各种传统控制方法均是建立在被控对象精确数学模型的基础上，然而，随着系统复杂程度的提高，将难以建立系统的精确数学模型。

在工程实践中，人们发现，一个复杂的控制系统可由一个操作人员凭着丰富的实践经验得到满意的控制效果。这说明，如果通过模拟人脑的思维方法设计控制器，可实现复杂系统的控制，由此产生了模糊控制。

1965 年美国加州大学自动控制系 L. A. Zedeh 提出模糊集合理论，奠定了模糊控制的基础；1974 年伦敦大学的 Mamdani 博士利用模糊逻辑，开发了世界上第一台模糊控制的蒸汽机，从而开创了模糊控制的历史；1983 年日本富士电机开创了模糊控制在日本的第一项应用——水净化处理，之后，富士电机致力于模糊逻辑元件的开发与研究，并于 1987 年在仙台地铁线上采用了模糊控制技术，1989 年将模糊控制消费品推向高潮，使日本成为模糊控制技术的主导国家。模糊控制的发展可分为 3 个阶段：

(1)1965—1974 年,为模糊控制发展的第一阶段,即模糊数学发展和形成阶段;
(2)1974—1979 年,为模糊控制发展的第二阶段,产生了简单的模糊控制器;
(3)1979 年至现在,为模糊控制发展的第三阶段,即高性能模糊控制阶段。

2. 神经网络控制

神经网络的研究已经有几十年的历史了。1943 年 McCulloch 和 Pitts 提出了神经元数学模型;1950—1980 年为神经网络的形成期,有少量成果,如 1975 年 Albus 提出了人脑记忆模型 CMAC 网络,1976 年 Grossberg 提出了用于无导师指导下模式分类的自组织网络;1980 年以后为神经网络的发展期,1982 年 Hopfield 提出了 Hopfield 网络,解决了回归网络的学习问题,1986 年美国的 PDP 研究小组提出了 BP 网络,实现了有导师指导下的网络学习,为神经网络的应用开辟了广阔的发展前景。

将神经网络引入控制领域就形成了神经网络控制。神经网络控制是从机理上对人脑生理系统进行简单结构模拟的一种新兴智能控制方法。神经网络具有并行机制、模式识别、记忆和自学习能力的特点,它能充分逼近任意复杂的非线性系统,能够学习与适应不确定系统的动态特性,有很强的鲁棒性和容错性。神经网络控制在控制领域有着广泛的应用。

3. 遗传算法

遗传算法(Genetic Algorithm,GA)是人工智能的一个重要分支,是基于自然选择和基因遗传学原理的搜索算法,是基于达尔文进化论,在计算机上模拟生命进化论机制而发展起来的一门学科。遗传算法由美国的 J. H. Holland 教授在 1975 年提出,20 世纪 80 年代中期开始逐步成熟。从 1985 年起,国际上开始举行遗传算法国际会议。目前遗传算法已经被广泛应用于许多实际问题,成为用来解决高度复杂问题的新思路和新方法。

遗传算法可用于模糊控制规则的优化及神经网络参数及权值的学习,在智能控制领域有广泛的应用。

1.3 智能控制的特点、研究工具及应用

1. 智能控制的特点

(1)学习功能

智能控制器能通过从外界环境所获得的信息进行学习,不断积累知识,使系统的控制性能得到改善。

(2)适应功能

智能控制器具有从输入到输出的映射关系,可实现不依赖于模型的自适应控制,当系统某一部分出现故障时,也能进行控制。

(3)自组织功能

智能控制器对复杂的分布式信息具有自组织和协调的功能,当出现多目标冲突时,它可以在任务要求的范围内自行决策,主动采取行动。

(4)优化能力

智能控制器能够通过不断优化控制参数和寻找控制器的最佳结构形式获得整体最优的控制性能。

2. 智能控制的研究工具

(1)符号推理与数值计算的结合

如专家控制,它的上层是专家系统,采用人工智能中的符号推理方法;下层是传统意义下的控制系统,采用数值计算方法。

(2)模糊集理论

模糊集理论是模糊控制的基础,其核心是采用模糊规则进行逻辑推理,其逻辑取值可在0与1之间连续变化,其处理的方法是基于数值的而不是基于符号的。

(3)神经网络理论

神经网络通过许多简单的关系来实现复杂的函数,其本质是一个非线性动力学系统,但它不依赖数学模型,是一种介于逻辑推理和数值计算之间的工具和方法。

(4)遗传算法

遗传算法(GA)根据适者生存、优胜劣汰等自然进化规则来进行搜索计算和问题求解。对许多传统数学难以解决或明显失效的复杂问题,特别是优化问题,GA提供了一个行之有效的途径。

(5)离散事件与连续时间系统的结合

它主要用于计算机集成制造系统(CIMS)和智能机器人的智能控制。以CIMS为例,上层任务的分配和调度、零件的加工和传输等可用离散事件系统理论进行分析和设计;下层的控制,如机床及机器人的控制,则采用常规的连续时间系统方法。

3. 智能控制的应用

作为智能控制发展的高级阶段,智能控制主要解决那些用传统控制方法难以解决的复杂系统的控制问题,其中包括智能机器人控制、计算机集成制造系统(CIMS)、工业过程控制、航空航天控制、社会经济管理系统、交通运输系统、环保及能源系统等。下面以智能控制在机器人控制和在过程控制中的应用为例进行说明。

(1)在机器人控制中的应用

智能机器人是目前机器人研究中的热门课题。E. H. Mamdan于20世纪80年代初首次将模糊控制应用于一台实际机器人的操作臂控制。J. S. Albus于1975年提出小脑模型关节控制器(Cerebellar Model Arculation Controller,CMAC),它是仿照小脑如何控制肢体运动的原理而建立的神经网络模型,采用CMAC,可实现机器人的关节控制,这是神经网络在机器人控制中的一个典型应用。

目前工业上用的90%以上的机器人都不具有智能。随着机器人技术的迅速发展,需要各种具有不同程度智能的机器人。

(2)在过程控制中的应用

过程控制是指石油、化工、冶金、轻工、纺织、制药、建材等工业生产过程的自动控制,它是自动化技术的一个极其重要的方面。智能控制在过程控制中有着广泛的应用。在石油化工方面,1994年美国的Gensym公司和Neuralware公司联合将神经网络用于炼油厂的非线性工艺过程。在冶金方面,日本的新日铁公司于1990年将专家控制系统应用于轧钢生产过程。在化工方面,日本的三菱化学合成公司研制出用于乙烯工程的模糊控制系统。

将智能控制应用于过程控制领域,是过程控制发展新的方向。

思考题与习题

1-1 简述智能控制的概念。

1-2 智能控制由哪几部分组成？各自的特点是什么？

1-3 比较智能控制和传统控制的特点。

1-4 智能控制有哪些应用领域？试各举出一个应用实例。

第2章 专家控制

在传统控制系统中，系统的运行排斥了人为的干预，人机之间缺乏交互，控制器对被控对象在环境中的参数、结构的变化缺乏应变能力。传统控制理论的不足，在于它必须依赖于被控对象严格的数学模型，试图对精确模型来求取最优的控制效果，而实际的被控对象存在着许多难以建模的因素。

20世纪80年代初，人工智能中专家系统的思想和方法开始被引入控制系统的研究和工程应用中。专家系统主要面临的是各种非结构化的问题，它能处理定性的、启发式或不确定的知识信息，经过各种推理来达到系统的任务目标。专家系统这一特点为解决传统控制理论的局限性提供了重要的启示，两者的结合导致了专家控制这一方法。

2.1 专家系统

2.1.1 专家系统概述

1. 定义

专家系统是一类包含知识和推理的智能计算机程序，其内部包含某领域专家水平的知识和经验，具有解决专门问题的能力。

2. 发展历史

专家系统的发展分为3个时期。

(1)初创期(1965—1971年)

第一代专家系统DENLDRA和MACSMA的出现，标志着专家系统的诞生。其中，DENLDRA为推断化学分子结构的专家系统，由专家系统的奠基人、Stanford大学计算机系的Feigenbaum教授及其研究小组研制。MACSMA为用于数学运算的数学专家系统，由麻省理工学院完成。

(2)成熟期(1972—1977年)

在此期间Stanford大学研究开发了最著名的专家系统——血液感染病诊断专家系统MYCIN，标志着专家系统从理论走向应用。另一个著名的专家系统——语音识别专家系统HEARSAY的出现，标志着专家系统从理论走向成熟。

(3)发展期(1978年至现在)

在此期间，专家系统走向应用领域，专家系统的数量增加，仅1987年研制成功的专家系统就有1000多种。

专家系统可以解决的问题一般包括解释、预测、设计、规划、监视、修理、指导和控制等。目前，专家系统已经广泛地应用于医疗诊断、语音识别、图像处理、金融决策、地质勘探、石油化工、教学、军事、计算机设计等领域。

2.1.2 专家系统的构成

专家系统主要由知识库和推理机构成，专家系统的结构如图 2-1 所示。

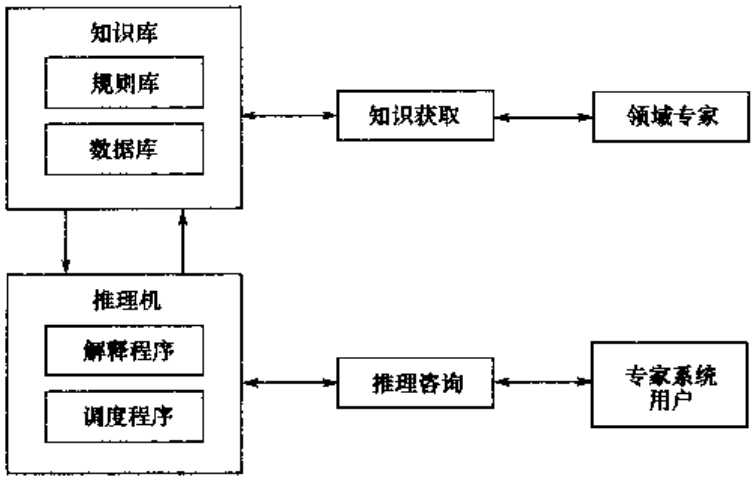


图 2-1 专家系统的结构

2.1.3 专家系统的建立

1. 知识库

知识库包含 3 类知识：

(1)基于专家经验的判断性规则；

(2)用于推理、问题求解的控制性规则；

(3)用于说明问题的状态、事实和概念及当前的条件和常识等的数据。

知识库包含多种功能模块，主要有知识查询、检索、增删、修改和扩充等。知识库通过人机接口与领域专家相沟通，从而实现知识的获取。

2. 推理机

推理机是用于对知识库中的知识进行推理来得到结论的“思维”机构。推理机包括 3 种推理方式：

(1)正向推理：从原始数据和已知条件得到结论；

(2)反向推理：先提出假设的结论，然后寻找支持的证据，若证据存在，则假设成立；

(3)双向推理：运用正向推理提出假设的结论，运用反向推理来证实假设。

3. 知识的表示

常用的知识表示方法为：产生式规则、框架、语义网络、过程。其中，产生式规则是专家系统最流行的表达方法。由产生式规则表示的专家系统又称为基于规则的系统或产生式系统。

产生式规则的表达方式为

IF E THEN H WITH CF(E,H)

式中，E 表示规则的前提条件，即证据，它可以是单独命题，也可以是复合命题；H 表示规则的结论部分，即假设，也是命题；CF(Certainty Factor)为规则的强度，反映当前提为真时，规则对

结论的影响程度，即可信度。

4. 专家系统开发语言

(1)C语言，人工智能语言(如Prolog，Lisp等)。

(2)专家系统开发工具：已经建好的专家系统框架，包括知识表达和推理机。在运用专家系统开发工具开发专家系统时，只需要加入领域知识。

5. 专家系统建立步骤

(1)知识库的设计

①确定知识类型：叙述性知识、过程性知识、控制性知识；

②确定知识表达方法；

③知识库管理系统的设计：实现规则的保存、编辑、删除、增加、搜索等功能。

(2)推理机的设计

①选择推理方式；

②选择推理算法：选择各种搜索算法，如深度优先搜索、广度优先搜索、启发式优先搜索等。

(3)人机接口的设计

①设计"用户-专家系统接口"：用于咨询理解和结论解释；

②设计"专家-专家系统接口"：用于知识库扩充及系统维护。

2.2 专家控制

2.2.1 专家控制概述

瑞典学者K. J. Åstrom于1983年首先把人工智能中的专家系统引入智能控制领域，于1986年提出"专家控制"的概念，构成一种智能控制方法。

专家控制(Expert Control)是智能控制的一个重要分支，又称专家智能控制。所谓专家控制，是将专家系统的理论和技术同控制理论、方法与技术相结合，在未知环境下，仿效专家的经验，实现对系统的控制。

专家控制试图在传统控制的基础上"加入"一个富有经验的控制工程师，实现控制的功能，它由知识库和推理机构构成主体框架，通过对控制领域知识(先验经验、动态信息、目标等)的获取与组织，按某种策略及时地选用恰当的规则进行推理输出，实现对实际对象的控制。

2.2.2 专家控制的基本原理

1. 结构

专家控制的基本结构如图2-2所示。

2. 功能

(1)能够满足任意动态过程的控制需要，尤其适用于带有时变、非线性和强干扰的控制；

(2)控制过程可以利用对象的先验知识；

(3)通过修改、增加控制规则，可不断积累知识，改进控制性能；

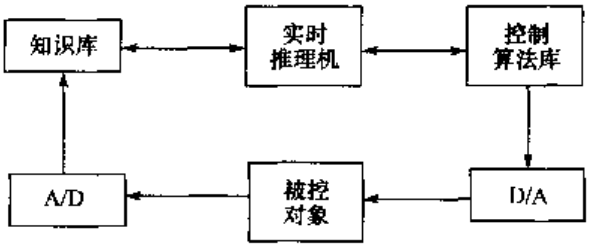


图 2-2 专家控制的基本结构

(4)可以定性地描述控制系统的性能,如超调小、偏差增大等;

(5)对控制性能可进行解释;

(6)可通过对控制闭环中的单元进行故障检测来获取经验规则。

3. 与专家系统的区别

专家控制引入了专家系统的思想,但与专家系统存在以下区别。

(1)专家系统能完成专门领域的功能,辅助用户决策;专家控制能进行独立的、实时的自动决策。专家控制比专家系统对可靠性和抗干扰性有着更高的要求。

(2)专家系统处于离线工作方式,而专家控制要求在线获取反馈信息,即要求在线工作方式。

4. 知识表示

专家控制将系统视为基于知识的系统,控制系统的知识表示如下:

(1)受控过程的知识

①先验知识:包括问题的类型及开环特性;

②动态知识:包括中间状态及特性变化。

(2)控制、辨识、诊断知识

①定量知识:各种算法;

②定性知识:各种经验、逻辑、直观判断。

按照专家系统知识库的结构,有关知识可以分类组织,形成数据库和规则库,从而构成专家控制系统的知识源。

数据库包括:①事实:知识的静态数据,如传感器测量误差、运行阈值、报警阈值、操作序列的约束条件、受控过程的单元组态等;②证据:测量到的动态数据,如传感器的输出值、仪器仪表的测试结果等。证据的类型是各异的,常常带有噪声、延迟,也可能是不完整的,甚至相互之间有冲突;③假设:由事实和证据推导的中间结果,作为当前事实集合的补充,如通过各种参数估计算法推得的状态估计等;④目标:系统的性能指标,如对稳定性的要求、对静态工作点的寻优、对现有控制规律是否需要改进的判断等。目标既可以是预定的,也可以是根据外部命令或内部运行状况在线动态建立的。

专家控制的规则库一般采用产生式规则表示,即

IF 控制局势(事实和数据) THEN 操作结论

由多条产生式规则构成规则库。

5. 分类

按专家控制在控制系统中的作用和功能,可将专家控制器分为以下两种类型。

(1)直接型专家控制器

直接型专家控制器用于取代常规控制器,直接控制生产过程或被控对象。具有模拟(或延伸、扩展)操作工人智能的功能。该控制器的任务和功能相对比较简单,但需要在线、实时控制。因此,其知识表达和知识库也较简单,通常由几十条产生式规则构成,以便于增删和修改。

直接型专家控制器的结构如图 2-3 中的虚线所示。

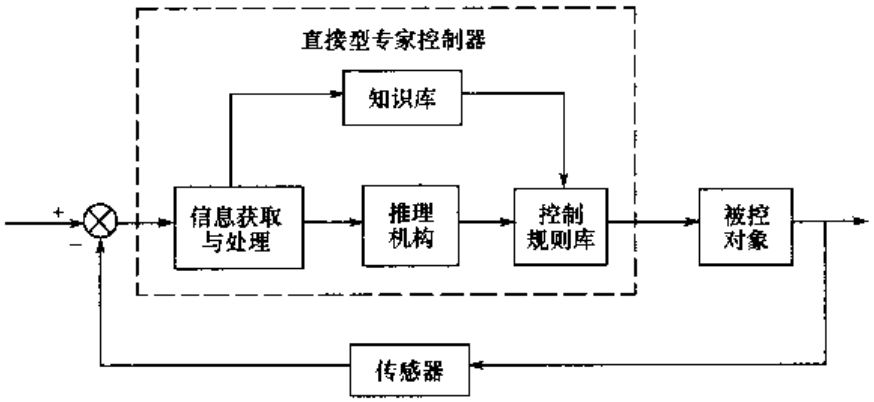


图 2-3 直接型专家控制器的结构

(2)间接型专家控制器

间接型专家控制器用于和常规控制器相结合,组成对生产过程或被控对象进行间接控制的智能控制系统。具有模拟(或延伸、扩展)控制工程师智能的功能。该控制器能够实现优化适应、协调、组织等高层决策的智能控制。按照高层决策功能的性质,间接型专家控制器可分为以下几种类型。

①优化型专家控制器:是基于最优控制专家知识和经验的总结和运用。通过设置整定值、优化控制参数或控制器,实现控制器的静态或动态优化。

②适应型专家控制器:是基于自适应控制专家的知识和经验的总结和运用。根据现场运行状态和测试数据,相应地调整控制律,校正控制参数,修改整定值或控制器,适应生产过程、对象特性或环境条件的漂移和变化。

③协调型专家控制器:是基于协调控制专家和调度工程师的知识和经验的总结和运用。用以协调局部控制器或各子控制系统的运行,实现大系统的全局稳定和优化。

④组织型专家控制器:是基于控制工程组织管理专家或总设计师的知识和经验的总结和运用。用以组织各种常规控制器,根据控制任务的目标和要求,构成所需要的控制系统。

间接型专家控制器可以在线或离线运行。通常,优化型、适应型需要在线、实时、联机运行;协调型、组织型可以离线、非实时运行,作为相应的计算机辅助系统。

间接型专家控制器的结构如图 2-4 所示。

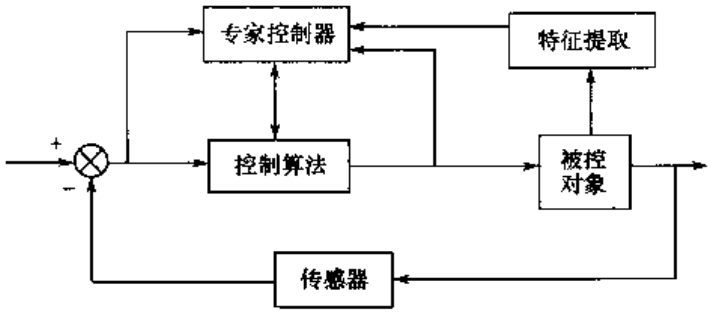


图 2-4 间接型专家控制器的结构

2.2.3 专家控制的关键技术及特点

1. 专家控制的关键技术

(1)知识的表达方法；

(2)从传感器中识别和获取定量的控制信号；

(3)将定性知识转化为定量的控制信号；

(4)控制知识和控制规则的获取。

2. 专家控制的特点

(1)灵活性：根据系统的工作状态及误差情况，可灵活地选取相应的控制律；

(2)适应性：根据专家知识和经验，调整控制器的参数，适应对象特性及环境的变化；

(3)鲁棒性：通过利用专家规则，系统可以在非线性、大偏差下可靠地工作。

2.3 专家 PID 控制

2.3.1 专家 PID 控制原理

专家 PID 控制的实质是：基于受控对象和控制规律的各种知识，无须知道被控对象的精确模型，利用专家经验来设计 PID 参数。专家 PID 控制是一种直接型专家控制器。

典型的二阶系统单位阶跃响应误差曲线如图 2-5 所示。

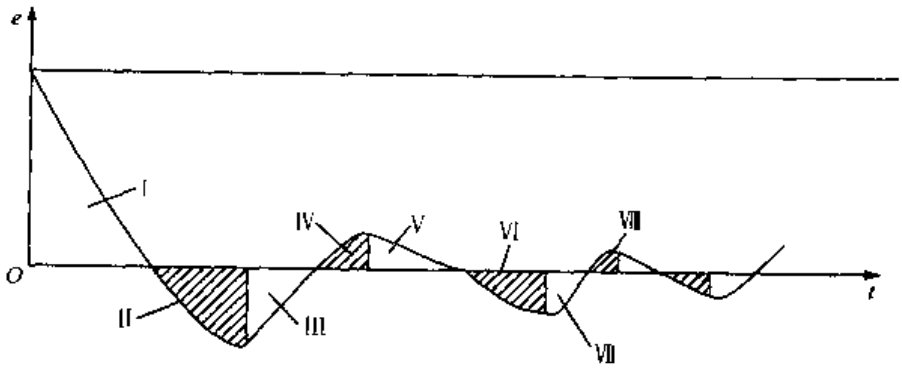


图 2-5 典型二阶系统单位阶跃响应误差曲线

令 $e(k)$ 表示离散化的当前采样时刻的误差值，$e(k-1)$，$e(k-2)$ 分别表示前一个和前两个采样时刻的误差值，则有

$$\Delta e(k) = e(k) - e(k-1)$$

$$\Delta e(k-1) = e(k-1) - e(k-2) \tag{2.1}$$

根据误差及其变化，对图 2-5 所示的二阶系统单位阶跃响应误差曲线进行如下定性分析：

(1) 当 $|e(k)| > M_1$ 时，说明误差的绝对值已经很大。不论误差变化趋势如何，都应考虑控制器的输出按定值输出，以达到迅速调整误差，使误差绝对值以最大速度减小，同时避免超调。此时，它相当于实施开环控制。

(2) 当 $e(k)\Delta e(k) > 0$ 或 $\Delta e(k) = 0$ 时，说明误差在朝误差绝对值增大方向变化，或误差为某一常值，未发生变化。

如果 $|e(k)|\geqslant M_2$，说明误差较大，可考虑由控制器实施较强的控制作用，使误差绝对值朝减小方向变化，迅速减小误差的绝对值，控制器输出为

$$u(k)=u(k-1)+k_1\{k_p[e(k)-e(k-1)]+k_1e(k)+k_d[e(k)-2e(k-1)+e(k-2)]\} \tag{2.2}$$

如果 $|e(k)|<M_2$，说明尽管误差朝绝对值增大方向变化，但误差绝对值本身并不是很大，可考虑实施一般的控制作用，扭转误差的变化趋势，使其朝误差绝对值减小方向变化，控制器输出为

$$u(k)=u(k-1)+k_p[e(k)-e(k-1)]+k_1e(k)+k_d[e(k)-2e(k-1)+e(k-2)] \tag{2.3}$$

(3) 当 $e(k)\Delta e(k)<0$，$\Delta e(k)\Delta e(k-1)>0$ 或者 $e(k)=0$ 时，说明误差的绝对值朝减小的方向变化，或者已经达到平衡状态。此时，可考虑采取保持控制器输出不变。

(4) 当 $e(k)\Delta e(k)<0$，$\Delta e(k)\Delta e(k-1)<0$ 时，说明误差处于极值状态。如果此时误差的绝对值较大，即 $|e(k)|\geqslant M_2$，可考虑实施较强的控制作用，即

$$u(k)=u(k-1)+k_1k_pe_m(k) \tag{2.4}$$

如果此时误差的绝对值较小，即 $|e(k)|<M_2$，可考虑实施较弱的控制作用，即

$$u(k)=u(k-1)+k_2k_pe_m(k) \tag{2.5}$$

(5) 当 $|e(k)|\leqslant\varepsilon$(精度) 时，说明误差的绝对值很小，此时加入积分环节，减少稳态误差。

以上各式中，$e_m(k)$ 为误差 e 的第 k 个极值；$u(k)$ 为第 k 次控制器的输出；$u(k-1)$ 为第 $k-1$ 次控制器的输出；k_1 为增益放大系数，$k_1>1$；k_2 为抑制系数，$0<k_2<1$；M_1，M_2 为设定的误差界限，$M_1>M_2>0$；k 为控制周期的序号(自然数)；ε 为任意小的正实数。

在图 2-5 中，Ⅰ，Ⅲ，Ⅴ，Ⅶ，… 区域，误差朝绝对值减小的方向变化，此时，可采取保持等待措施，相当于实施开环控制；Ⅱ，Ⅳ，Ⅵ，Ⅷ，… 区域，误差绝对值朝增大的方向变化，此时，可根据误差的大小分别实施较强或一般的控制作用，以抑制动态误差。

2.3.2 仿真实例

求 3 阶传递函数的阶跃响应

$$G_p(s)=\frac{523500}{s^3+87.35s^2+10470s}$$

其中，对象采样时间为 1ms。

采用 z 变换进行离散化，经过 z 变换后的离散化对象为

$$y(k)=-\mathrm{den}(2)y(k-1)-\mathrm{den}(3)y(k-2)-\mathrm{den}(4)y(k-3)+\mathrm{num}(2)u(k-1)+\mathrm{num}(3)u(k-2)+\mathrm{num}(4)u(k-3)$$

采用专家 PID 设计控制器。在仿真过程中，ε 取 0.001，程序中的 5 条规则与控制算法的 5 种情况相对应。

专家 PID 控制仿真程序见附录程序 chap2_1. m，仿真结果如图 2-6 和图 2-7 所示。

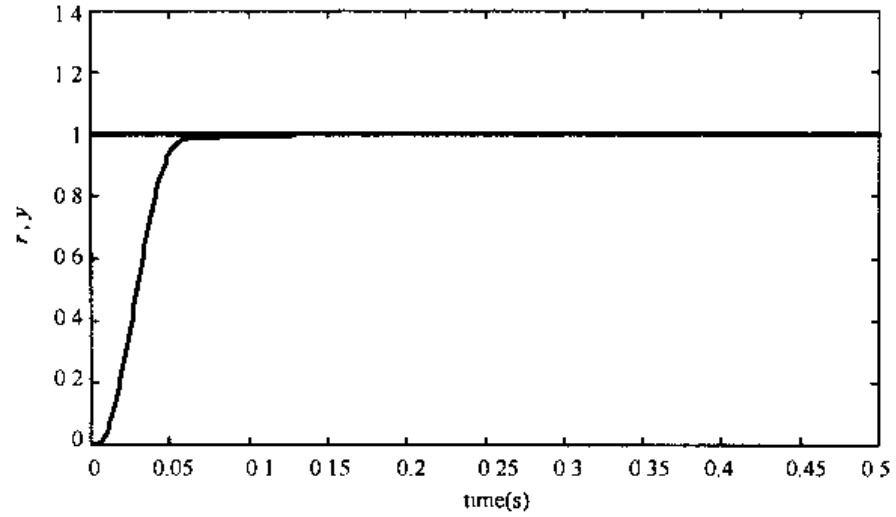


图 2-6　PID 控制阶跃响应曲线

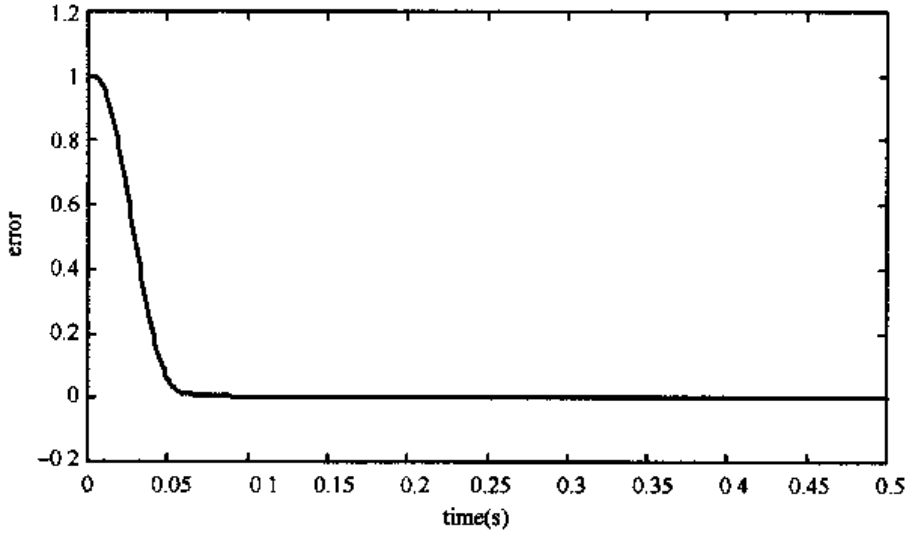


图 2-7　误差响应曲线

思考题与习题

2-1　专家系统由哪几部分组成?各自的特点是什么?

2-2　比较专家系统和专家控制的区别与联系。

2-3　求二阶传递函数的阶跃响应

$$G_p(s) = \frac{133}{s^2 + 25s}$$

取采样时间为 1ms。参照专家 PID 控制仿真程序 chap2_1. m，设计专家 PID 控制器，并进行 Matlab 仿真。

附录（程序代码）

专家 PID 控制仿真程序:chap2_1.m

```
%Expert PID Controller
clear all;
close all;
ts=0.001;

sys=tf(5.235e005,[1,87.35,1.047e004,0]);   %Plant
dsys=c2d(sys,ts,'z');
[num,den]=tfdata(dsys,'v');

u_1=0;u_2=0;u_3=0;
y_1=0;y_2=0;y_3=0;

x=[0,0,0]';
x2_1=0;

kp=0.6;
ki=0.03;
kd=0.01;

error_1=0;
for k=1:1:500
time(k)=k*ts;

r(k)=1.0;                          %Tracing Step Signal

u(k)=kp*x(1)+kd*x(2)+ki*x(3);  %PID Controller

%Expert control rule
if abs(x(1))>0.8          %Rule1:Unclosed control rule
   u(k)=0.45;
elseif abs(x(1))>0.40
   u(k)=0.40;
elseif abs(x(1))>0.20
   u(k)=0.12;
elseif abs(x(1))>0.01
   u(k)=0.10;
end
```

```
if x(1) * x(2)>0|(x(2)==0)          % Rule2
   if abs(x(1))>=0.05
      u(k)=u_1+2 * kp * x(1);
   else
      u(k)=u_1+0.4 * kp * x(1);
   end
end

if (x(1) * x(2)<0&x(2) * x2_1>0)|(x(1)==0)     % Rule3
    u(k)=u(k);
end

if x(1) * x(2)<0&x(2) * x2_1<0      % Rule4
   if abs(x(1))>=0.05
      u(k)=u_1+2 * kp * error_1;
   else
      u(k)=u_1+0.6 * kp * error_1;
   end
end

if abs(x(1))<=0.001     % Rule5:Integration separation PI control
   u(k)=0.5 * x(1)+0.010 * x(3);
end

% Restricting the output of controller
if u(k)>=10
   u(k)=10;
end
if u(k)<=-10
   u(k)=-10;
end

% Linear model
y(k)=-den(2) * y_1-den(3) * y_2-den(4) * y_3+num(1) * u(k)+num(2) * u_1+num(3)
* u_2+num(4) * u_3;
error(k)=r(k)-y(k);

%--------Return of parameters---------- %
u_3=u_2;u_2=u_1;u_1=u(k);
```

```
y_3=y_2;y_2=y_1;y_1=y(k);

x(1)=error(k);                    % Calculating P
x2_1=x(2);
x(2)=(error(k)-error_1)/ts;       % Calculating D
x(3)=x(3)+error(k)*ts;            % Calculating I

error_1=error(k);
end
figure(1);
plot(time,r,'b',time,y,'r');
xlabel('time(s)');ylabel('r,y');
figure(2);
plot(time,r-y,'r');
xlabel('time(s)');ylabel('error');
```

第3章　模糊控制的理论基础

3.1　概述

模糊控制是建立在人工经验基础之上的。对于一个熟练的操作人员，他往往凭借丰富的实践经验，采取适当的对策来巧妙地控制一个复杂过程。若能将这些熟练操作员的实践经验加以总结和描述，并用语言表达出来，就会得到一种定性的、不精确的控制规则。如果用模糊数学将其定量化，就转化为模糊控制算法，从而形成模糊控制理论。

模糊控制尚无统一的定义。从广义上，可将模糊控制定义为："以模糊集合理论、模糊语言变量及模糊推理为基础的一类控制方法"，或定义为"采用模糊集合理论和模糊逻辑，并同传统的控制理论相结合，模拟人的思维方式，对难以建立数学模型的对象实施的一种控制方法"。

模糊控制理论具有一些明显的特点：

(1)模糊控制不需要被控对象的数学模型。模糊控制是以人对被控对象的控制经验为依据而设计的控制器，故无须知道被控对象的数学模型。

(2)模糊控制是一种反映人类智慧的智能控制方法。模糊控制采用人类思维中的模糊量，如"高"、"中"、"低"、"大"、"小"等，控制量由模糊推理导出。这些模糊量和模糊推理是人类智能活动的体现。

(3)模糊控制易于被人们接受。模糊控制的核心是控制规则，模糊规则是用语言来表示的，如"今天气温高，则今天天气暖和"等，易于被一般人所接受。

(4)构造容易。模糊控制规则易于软件实现。

(5)鲁棒性和适应性好。通过专家经验设计的模糊规则可以对复杂的对象进行有效的控制。

3.2　模糊集合

3.2.1　模糊集合的概念

对大多数应用系统而言，其主要且重要的信息来源有两种，即来自传感器的数据信息和来自专家的语言信息。数据信息常用 0.5，2，3，3.5 等数字来表示，而语言信息则用诸如"大"、"小"、"中等"、"非常小"等文字来表示。传统的工程设计方法只能用数据信息而无法使用语言信息，而人类解决问题时所使用的大量知识是经验性的，它们通常是用语言信息来描述的。语言信息通常呈经验性，是模糊的。因此，如何描述模糊语言信息成为解决问题的关键。

模糊集合的概念是由美国加利福尼亚大学著名教授 L. A. Zadeh 于 1965 年首先提出来的。模糊集合的引入，可将人的判断、思维过程用比较简单的数学形式直接表达出来。模糊集合理论为人类提供了能充分利用语言信息的有效工具。模糊集合是模糊控制的数学基础。

1. 特征函数和隶属函数

在数学上经常用到集合的概念。

例如，集合 A 由 4 个离散值 x_1, x_2, x_3, x_4 组成，即

$$A = \{x_1, x_2, x_3, x_4\}$$

例如，集合 A 由 0 到 1 之间的连续实数值组成，即

$$A = \{x, x \in R, 0 \leqslant x \leqslant 1.0\}$$

以上两个集合是完全不模糊的。对任意元素 x，只有两种可能：属于 A，不属于 A。这种特性可以用特征函数 $\mu_A(x)$ 来描述，即

$$\mu_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \tag{3.1}$$

为了表示模糊概念，需要引入模糊集合和隶属函数及隶属度的概念。隶属函数定义为

$$\mu_A(x) = \begin{cases} 1 & x \in A \\ (0,1) & x \in A \text{ 的程度} \\ 0 & x \notin A \end{cases} \tag{3.2}$$

式中，A 称为模糊集合，由 0，1 及 $\mu_A(x)$ 构成，$\mu_A(x)$ 表示元素 x 属于模糊集合 A 的程度，取值范围为 $[0,1]$，称 $\mu_A(x)$ 为 x 属于模糊集合 A 的隶属度。

隶属度将普通集合中特征函数的取值 $\{0,1\}$ 扩展到闭区间 $[0,1]$，即可用 0 到 1 之间的实数来表达某一元素属于模糊集合的程度。

2. 模糊集合的表示

(1) 模糊集合 A 由离散元素构成，表示为

$$A = \mu_1/x_1 + \mu_2/x_2 + \cdots + \mu_i/x_i + \cdots \tag{3.3}$$

或

$$A = \{(x_1, \mu_1), (x_2, \mu_2), \cdots, (x_i, \mu_i), \cdots\} \tag{3.4}$$

(2) 模糊集合 A 由连续函数构成，各元素的隶属度就构成了隶属度函数(Membership Function) $\mu_A(x)$，此时 A 表示为

$$A = \int \mu_A(x)/x \tag{3.5}$$

在模糊集合的表达中，符号“/”，“+”和“$\int$”不代表数学意义上的除号、加号和积分，它们是模糊集合的一种表示方式，表示“构成”或“属于”。

模糊集合是以隶属函数 $\mu_A(x)$ 来描述的，隶属度的概念是模糊集合理论的基石。

例 3.1　设论域 U = {张三，李四，王五}，评语为“学习好”。设 3 个人学习成绩总评分是张三得 95 分，李四得 90 分，王五得 85 分，3 人都学习好，但又有差异。

若采用普通集合的观点，选取特征函数

$$C_A(u) = \begin{cases} 1 & \text{学习好} \in A \\ 0 & \text{学习差} \in A \end{cases}$$

此时特征函数分别为 C_A(张三) = 1，C_A(李四) = 1，C_A(王五) = 1。这样就反映不出三者的差异。假若采用模糊子集的概念，选取 $[0,1]$ 区间上的隶属度来表示它们属于“学习好”模糊子集 A 的程度，就能够反映出 3 人的差异。

采用隶属函数$\frac{x}{100}$，由3人的成绩可知3人“学习好”的隶属度为μ_A(张三) = 0.95，μ_A(李四) = 0.90，μ_A(王五) = 0.85。“学习好”这一模糊子集A可表示为

$$A = \{0.95, 0.90, 0.85\}$$

其含义为张三、李四、王五属于“学习好”的程度分别是0.95,0.90,0.85。

例3.2 以年龄为论域，取$x = [0, 100]$。Zadeh给出了“年轻”的模糊集Y，其隶属函数为

$$Y(x) = \begin{cases} 1.0 & 0 \leqslant x \leqslant 25 \\ \left[1 + \left(\frac{x-25}{5}\right)^2\right]^{-1} & 25 < x < 100 \end{cases}$$

“年轻”的隶属函数仿真程序见附录程序chap3_1.m。隶属函数曲线如图3-1所示。

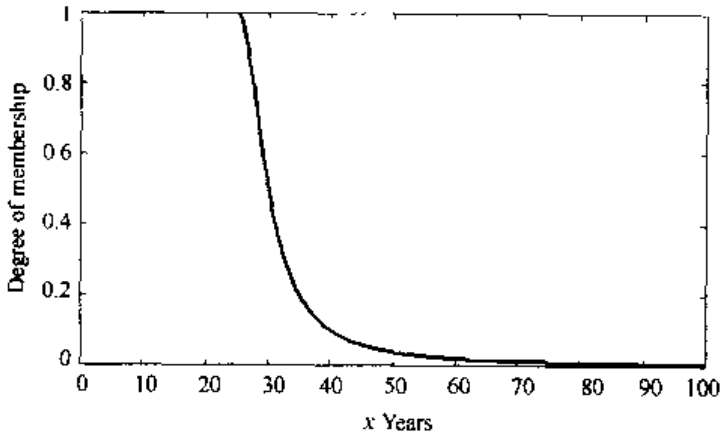


图3-1 “年轻”的隶属函数曲线

3.2.2 模糊集合的运算

1. 模糊集合的基本运算

由于模糊集合是用隶属函数来表征的，因此两个子集之间的运算实际上就是逐点对隶属度进行相应的运算。

(1) 空集

模糊集合A的空集$\varnothing$为普通集，它的隶属度为0，即

$$A = \varnothing \Leftrightarrow \mu_A(u) = 0 \tag{3.6}$$

(2) 全集

模糊集合A的全集E为普通集，它的隶属度为1，即

$$A = E \Leftrightarrow \mu_A(u) = 1 \tag{3.7}$$

(3) 等集

两个模糊集A和B，若对所有元素u，它们的隶属函数相等，则A和B也相等，即

$$A = B \Leftrightarrow \mu_A(u) = \mu_B(u) \tag{3.8}$$

(4) 补集

若$\overline{A}$为A的补集，则

$$\overline{A} \Leftrightarrow \mu_{\overline{A}}(u) = 1 - \mu_A(u) \tag{3.9}$$

例如，设A为“成绩好”的模糊集，某学生u_0属于“成绩好”的隶属度$\mu_A(u_0) = 0.8$，则u_0属于“成绩差”的隶属度$\mu_{\overline{A}}(u_0) = 1 - 0.8 = 0.2$。

(5) 子集

若 B 为 A 的子集，则

$$B \subseteq A \Leftrightarrow \mu_B(u) \leqslant \mu_A(u) \tag{3.10}$$

(6) 并集

若 C 为 A 和 B 的并集，则

$$C = A \cup B$$

一般地，有

$$A \cup B \Leftrightarrow \mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u)) = \mu_A(u) \vee \mu_B(u) \tag{3.11}$$

(7) 交集

若 C 为 A 和 B 的交集，则

$$C = A \cap B$$

一般地，有

$$A \cap B \Leftrightarrow \mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u)) = \mu_A(u) \wedge \mu_B(u) \tag{3.12}$$

(8) 模糊运算的基本性质

模糊集合除具有上述基本运算性质外，还具有如表 3-1 所示的运算性质。

表 3-1　模糊运算的基本性质

名　　称	运 算 法 则
1. 幂等律	$A \cup A = A, A \cap A = A$
2. 交换律	$A \cup B = B \cup A, A \cap B = B \cap A$
3. 结合律	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
4. 吸收律	$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$
5. 分配律	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
6. 复原律	$\overline{\overline{A}} = A$
7. 对偶律	$\overline{A \cup B} = \overline{A} \cap \overline{B}$ $\overline{A \cap B} = \overline{A} \cup \overline{B}$
8. 两极律	$A \cup E = E, A \cap E = A$ $A \cup \varnothing = A, A \cap \varnothing = \varnothing$

例 3.3　设 $A = \dfrac{0.9}{u_1} + \dfrac{0.2}{u_2} + \dfrac{0.8}{u_3} + \dfrac{0.5}{u_4}$，$B = \dfrac{0.3}{u_1} + \dfrac{0.1}{u_2} + \dfrac{0.4}{u_3} + \dfrac{0.6}{u_4}$，求 $A \cup B, A \cap B$。

解　$A \cup B = \dfrac{0.9}{u_1} + \dfrac{0.2}{u_2} + \dfrac{0.8}{u_3} + \dfrac{0.6}{u_4}$，$A \cap B = \dfrac{0.3}{u_1} + \dfrac{0.1}{u_2} + \dfrac{0.4}{u_3} + \dfrac{0.5}{u_4}$

例 3.4　试证明普通集合中的互补律在模糊集合中不成立，即 $\mu_A(u) \vee \mu_{\overline{A}}(u) \neq 1$，$\mu_A(u) \wedge \mu_{\overline{A}}(u) \neq 0$。

证明　设 $\mu_A(u) = 0.4$，则 $\mu_{\overline{A}}(u) = 1 - 0.4 = 0.6$，则

$$\mu_A(u) \vee \mu_{\overline{A}}(u) = 0.4 \vee 0.6 = 0.6 \neq 1$$

$$\mu_A(u) \wedge \mu_{\overline{A}}(u) = 0.4 \wedge 0.6 = 0.4 \neq 0$$

2. 模糊算子

模糊集合的逻辑运算实质上就是隶属函数的运算过程。采用隶属函数的取大(max)-取小(min)进行模糊集合的并、交逻辑运算是目前最常用的方法。但还有其他公式，这些公式统称为“模糊算子”。

设有模糊集合 A、B 和 C，常用的模糊算子如下：

(1) 交运算算子

设 $C = A \cap B$，有 3 种模糊算子。

① 模糊交算子

$$\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{3.13}$$

② 代数积算子

$$\mu_C(x) = \mu_A(x) \cdot \mu_B(x) \tag{3.14}$$

③ 有界积算子

$$\mu_C(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\} \tag{3.15}$$

(2) 并运算算子

设 $C = A \cup B$，有 3 种模糊算子。

① 模糊并算子

$$\mu_C(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{3.16}$$

② 概率或算子

$$\mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \tag{3.17}$$

③ 有界和算子

$$\mu_C(x) = \min\{1, \mu_A(x) + \mu_B(x)\} \tag{3.18}$$

(3) 平衡算子

当隶属函数取大、取小运算时，不可避免地要丢失部分信息，采用一种平衡算子，即“γ 算子”，可起到补偿作用。

设 $C = A \circ B$，则

$$\mu_C(x) = [\mu_A(x) \cdot \mu_B(x)]^{1-\gamma} \cdot [1 - (1 - \mu_A(x)) \cdot (1 - \mu_B(x))]^{\gamma} \tag{3.19}$$

式中，γ 取值为[0,1]。当 $\gamma = 0$ 时，$\mu_C(x) = \mu_A(x) \cdot \mu_B(x)$，相当于 $A \cap B$ 时的代数积算子；当 $\gamma = 1$ 时，$\mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$，相当于 $A \cup B$ 时的概率或算子。

平衡算子目前已经应用于德国 Inform 公司研制的著名模糊控制软件 Fuzzy-Tech 中。

3.3 隶属函数

1. 隶属函数的特点

普通集合用特征函数来表示，模糊集合用隶属函数来描述。隶属函数很好地描述了事物的模糊性。隶属函数有以下两个特点。

(1) 隶属函数的值域为[0,1]，它将普通集合只能取 0,1 两个值，推广到[0,1]闭区间上连续取值。隶属函数的值 $\mu_A(x)$ 越接近于 1，表示元素 x 属于模糊集合 A 的程度越大。反之，$\mu_A(x)$ 越接近于 0，表示元素 x 属于模糊集合 A 的程度越小。

(2) 隶属函数完全刻画了模糊集合，隶属函数是模糊数学的基本概念，不同的隶属函数所

描述的模糊集合也不同。

2. 几种典型的隶属函数及其 Matlab 表示

典型的隶属函数有 11 种，即双 S 形隶属函数、联合高斯型隶属函数、高斯型隶属函数、广义钟形隶属函数、Π 型隶属函数、双 S 形乘积隶属函数、S 状隶属函数、S 形隶属函数、梯形隶属函数、三角形隶属函数、Z 形隶属函数。在模糊控制中应用较多的隶属函数有以下 6 种。

(1) 高斯型隶属函数

高斯型隶属函数由两个参数 σ 和 c 确定，即

$$f(x,\sigma,c)=\mathrm{e}^{-\frac{(x-c)^2}{2\sigma^2}} \tag{3.20}$$

式中，参数 σ 通常为正，参数 c 用于确定曲线的中心。Matlab 表示为 gaussmf(x,[σ,c])。

(2) 广义钟形隶属函数

广义钟形隶属函数由 3 个参数 a,b,c 确定，即

$$f(x,a,b,c)=\frac{1}{1+\left|\dfrac{x-c}{a}\right|^{2b}} \tag{3.21}$$

式中，参数 a 和 b 通常为正，参数 c 用于确定曲线的中心。Matlab 表示为 gbellmf(x,[a,b,c])。

(3)S 形隶属函数

S 形函数由参数 a 和 c 确定，即

$$f(x,a,c)=\frac{1}{1+\mathrm{e}^{-a(x-c)}} \tag{3.22}$$

式中，参数 a 的正负符号决定了 S 形隶属函数的开口朝左或朝右，用来表示“正大”或“负大”的概念。Matlab 表示为 sigmf(x,[a,c])。

(4) 梯形隶属函数

梯形曲线可由 4 个参数 a,b,c,d 确定，即

$$f(x,a,b,c,d)=\begin{cases}0 & x\leqslant a\\ \dfrac{x-a}{b-a} & a\leqslant x\leqslant b\\ 1 & b\leqslant x\leqslant c\\ \dfrac{d-x}{d-c} & c\leqslant x\leqslant d\\ 0 & x\geqslant d\end{cases} \tag{3.23}$$

式中，参数 a 和 d 确定梯形的“脚”，而参数 b 和 c 确定梯形的“肩膀”。Matlab 表示为 trapmf(x,[a,b,c,d])。

(5) 三角形隶属函数

三角形曲线的形状由 3 个参数 a,b,c 确定，即

$$f(x,a,b,c)=\begin{cases}0 & x\leqslant a\\ \dfrac{x-a}{b-a} & a\leqslant x\leqslant b\\ \dfrac{c-x}{c-b} & b\leqslant x\leqslant c\\ 0 & x\geqslant c\end{cases} \tag{3.24}$$

式中，参数 a 和 c 确定三角形的“脚”，而参数 b 确定三角形的“峰”。Matlab 表示为 trimf(x, [a,b,c])。

(6)Z 形隶属函数

这是基于样条函数的曲线，因其呈现 Z 形状而得名。参数 a 和 b 确定了曲线的形状。Matlab 表示为 zmf(x,[a,b])。

在上述隶属函数中，高斯型隶属函数、广义钟形隶属函数、梯形隶属函数和三角形隶属函数可用于描述具有中间模糊状态的模糊概念，如“中等个”、“中年人”等。S 形隶属函数和 Z 形隶属函数可用于描述一个完整的模糊概念，如水箱液位的高低、人的胖瘦等。

例 3.5 隶属函数的仿真：针对上述描述的 6 种隶属函数进行仿真。$x \in [0,10]$，M 为隶属函数的类型，其中 $M=1$ 为高斯型隶属函数，$M=2$ 为广义钟形隶属函数，$M=3$ 为 S 形隶属函数，$M=4$ 为梯形隶属函数，$M=5$ 为三角形隶属函数，$M=6$ 为 Z 形隶属函数。

仿真程序见附录程序 chap3_2.m，仿真结果如图 3-2 至图 3-7 所示。

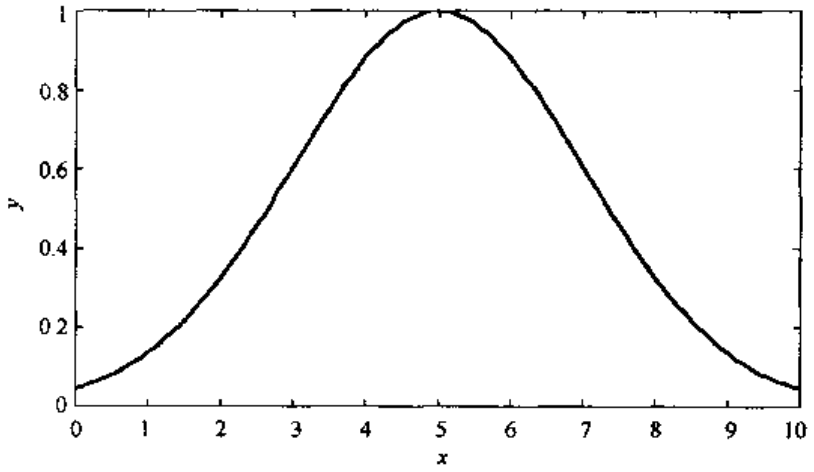


图 3-2 高斯型隶属函数($M=1$)

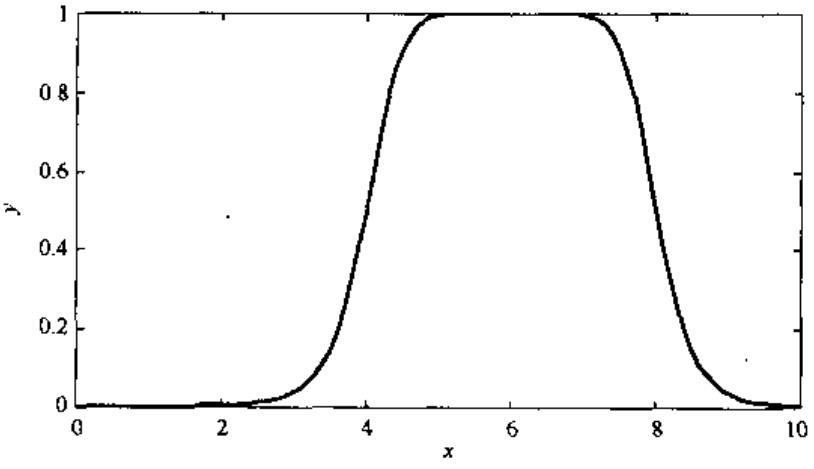


图 3-3 广义钟形隶属函数($M=2$)

3. 模糊系统的设计

采用隶属函数可设计模糊系统。例如，采用三角形隶属函数，按[−3,3]范围 7 个等级，建立一个模糊系统。

模糊系统隶属函数设计程序见附录程序 chap3_3.m，仿真结果如图 3-8 所示。

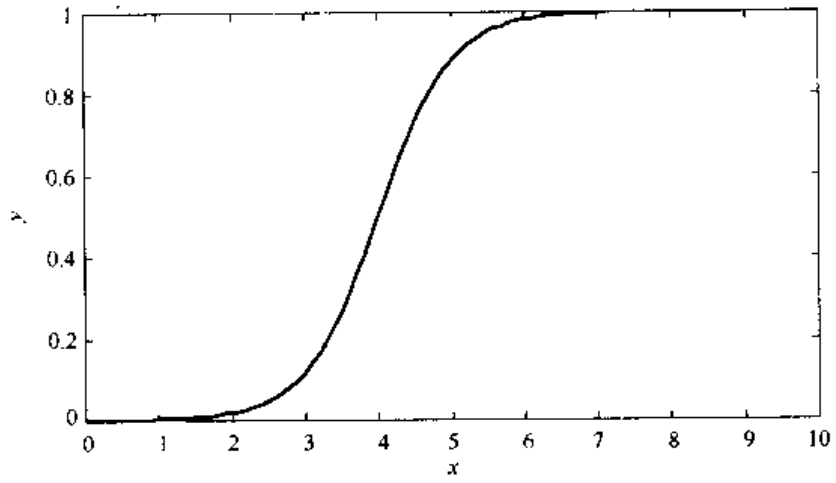


图 3-4　S形隶属函数($M=3$)

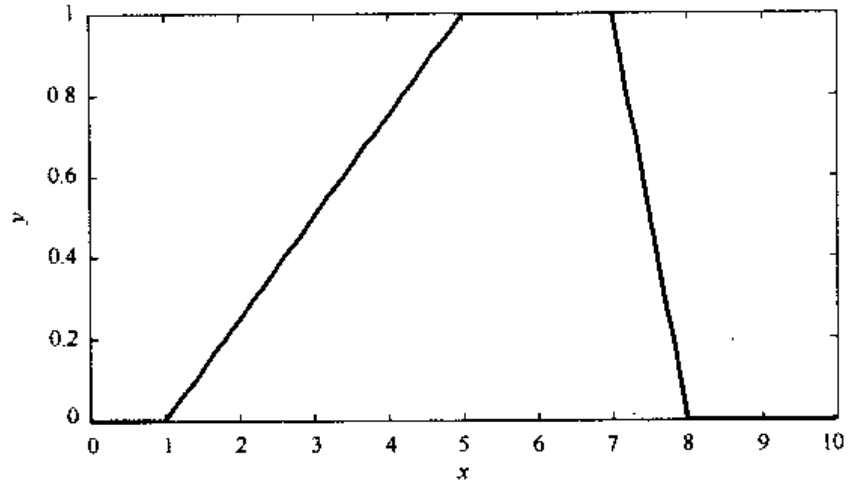


图 3-5　梯形隶属函数($M=4$)

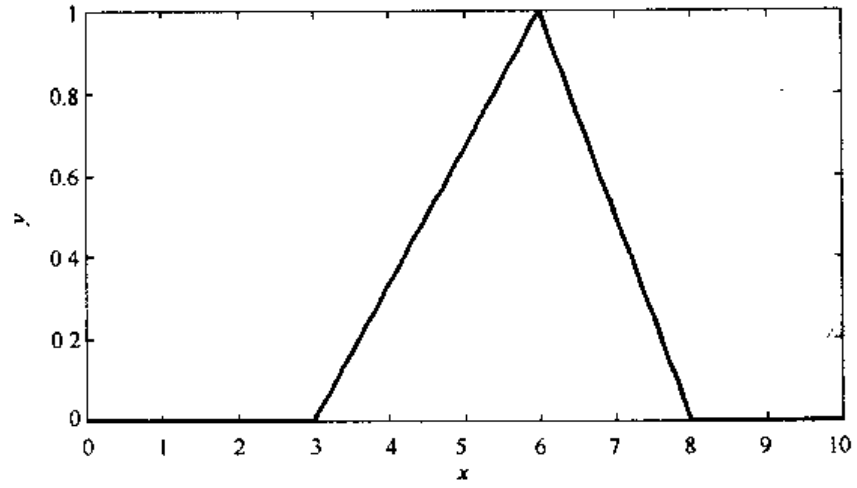


图 3-6　三角形隶属函数($M=5$)

4. 隶属函数的确定方法

隶属函数是模糊控制的应用基础。目前还没有成熟的方法来确定隶属函数，主要还停留在经验和实验的基础上。通常的方法是初步确定粗略的隶属函数，然后通过“学习”和实践来不

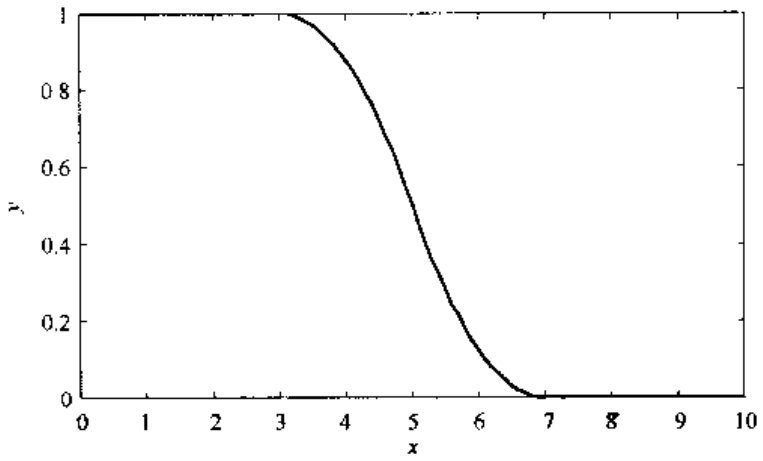


图 3-7 Z形隶属函数($M=6$)

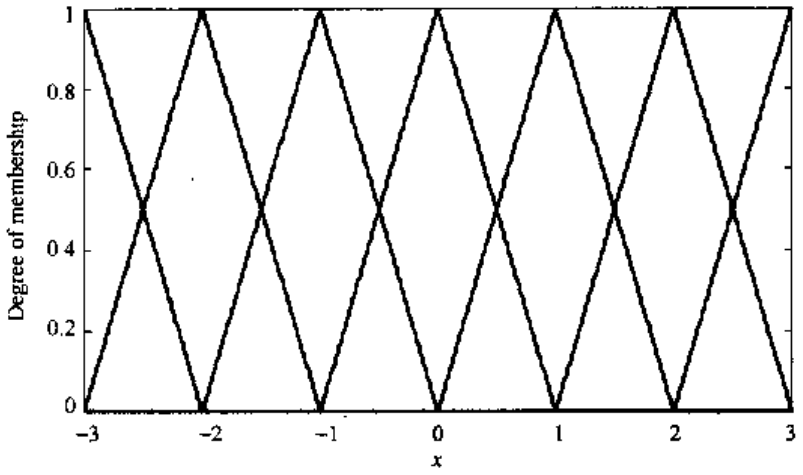


图 3-8 由三角形隶属函数构成的模糊系统

断地调整和完善。遵照这一原则的隶属函数选择方法有以下几种。

(1) 模糊统计法

根据所提出的模糊概念进行调查统计,提出与之对应的模糊集 A,通过统计实验,确定不同元素隶属于 A 的程度,即

$$u_0 \text{ 对模糊集 } A \text{ 的隶属度} = \frac{u_0 \in A \text{ 的次数}}{\text{试验总次数 } N} \tag{3.25}$$

(2) 主观经验法

当论域为离散论域时,可根据主观认识,结合个人经验,经过分析和推理,直接给出隶属度。这种确定隶属函数的方法已经被广泛应用。

(3) 神经网络法

利用神经网络的学习功能,由神经网络自动生成隶属函数,并通过网络的学习自动调整隶属函数的值。

3.4 模糊关系及其运算

描述客观事物间联系的数学模型称为关系。集合论中的关系精确地描述了元素之间是否

相关，而模糊集合论中的模糊关系则描述了元素之间相关的程度。普通二元关系是用简单的“有”或“无”来衡量事物之间的关系，因此无法用来衡量事物之间关系的程度。模糊关系是指多个模糊集合的元素间所具有关系的程度。模糊关系在概念上是普通关系的推广，普通关系则是模糊关系的特例。

3.4.1　模糊矩阵

例 3.6　设有一组同学 X，$X=\{$张三，李四，王五$\}$，他们的功课为 Y，$Y=\{$英语，数学，物理，化学$\}$。他们的考试成绩见表 3-2。

表 3-2　考试成绩表

功课 / 姓名	英语	数学	物理	化学
张三	70	90	80	65
李四	90	85	76	70
王五	50	95	85	80

取隶属函数 $\mu(u)=\dfrac{u}{100}$，其中 u 为成绩。如果将他们的成绩转化为隶属度，则构成一个 $x\times y$ 上的一个模糊关系 $\boldsymbol{R}$，见表 3-3。

表 3-3　考试成绩表的模糊化

功课 / 姓名	英语	数学	物理	化学
张三	0.70	0.90	0.80	0.65
李四	0.90	0.85	0.76	0.70
王五	0.50	0.95	0.85	0.80

将表 3-3 写成矩阵形式，得

$$\boldsymbol{R}=\begin{bmatrix}0.70 & 0.90 & 0.80 & 0.65\\ 0.90 & 0.85 & 0.76 & 0.70\\ 0.50 & 0.95 & 0.85 & 0.80\end{bmatrix}$$

该矩阵称为模糊矩阵，其中各个元素必须在$[0,1]$闭环区间内取值。矩阵 $\boldsymbol{R}$ 也可以用关系图来表示，如图 3-9 所示。

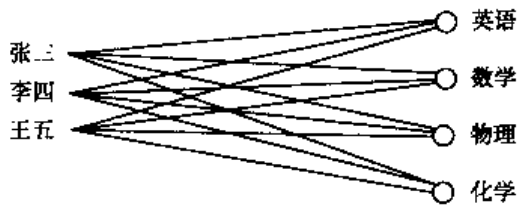


图 3-9　模糊矩阵 $\boldsymbol{R}$ 的关系图

3.4.2　模糊矩阵的运算

设有 n 阶模糊矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，$\boldsymbol{A}=(a_{ij})$，$\boldsymbol{B}=(b_{ij})$，且 $i,j=1,2,\cdots,n$，则定义如下几种模糊矩阵的运算方式。

(1) 相等

若 $a_{ij}=b_{ij}$，则 $\boldsymbol{A}=\boldsymbol{B}$。

(2) 包含

若 $a_{ij}\leqslant b_{ij}$，则 $\boldsymbol{A}\subseteq\boldsymbol{B}$。

(3) 并运算

若 $c_{ij}=a_{ij}\vee b_{ij}$，则 $\boldsymbol{C}=(c_{ij})$ 为 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的并，记为 $\boldsymbol{C}=\boldsymbol{A}\cup\boldsymbol{B}$。

(4) 交运算

若 $c_{ij}=a_{ij}\wedge b_{ij}$，则 $\boldsymbol{C}=(c_{ij})$ 为 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的交，记为 $\boldsymbol{C}=\boldsymbol{A}\cap\boldsymbol{B}$。

(5) 补运算

若 $c_{ij}=1-a_{ij}$，则 $\boldsymbol{C}=(c_{ij})$ 为 $\boldsymbol{A}$ 的补，记为 $\boldsymbol{C}=\overline{\boldsymbol{A}}$。

例 3.7 设 $\boldsymbol{A}=\begin{bmatrix}0.7 & 0.1\\ 0.3 & 0.9\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0.4 & 0.9\\ 0.2 & 0.1\end{bmatrix}$，则

$$\boldsymbol{A}\cup\boldsymbol{B}=\begin{bmatrix}0.7\vee 0.4 & 0.1\vee 0.9\\ 0.3\vee 0.2 & 0.9\vee 0.1\end{bmatrix}=\begin{bmatrix}0.7 & 0.9\\ 0.3 & 0.9\end{bmatrix}$$

$$\boldsymbol{A}\cap\boldsymbol{B}=\begin{bmatrix}0.7\wedge 0.4 & 0.1\wedge 0.9\\ 0.3\wedge 0.2 & 0.9\wedge 0.1\end{bmatrix}=\begin{bmatrix}0.4 & 0.1\\ 0.2 & 0.1\end{bmatrix}$$

$$\overline{\boldsymbol{A}}=\begin{bmatrix}1-0.7 & 1-0.1\\ 1-0.3 & 1-0.9\end{bmatrix}=\begin{bmatrix}0.3 & 0.9\\ 0.7 & 0.1\end{bmatrix}$$

3.4.3 模糊矩阵的合成

模糊矩阵合成是指，根据第一个集合和第二个集合之间的模糊关系与第二个集合和第三个集合之间的模糊关系，进而得到第一个集合和第三个集合之间模糊关系的一种运算形式。模糊矩阵的合成类似于普通矩阵的乘积。将乘积运算换成“取小”，将加运算换成“取大”即可。

设矩阵 $\boldsymbol{A}$ 是 $x\times y$ 上的模糊关系，矩阵 $\boldsymbol{B}$ 是 $y\times z$ 上的模糊关系，则 $\boldsymbol{C}=\boldsymbol{A}\circ\boldsymbol{B}$ 称为 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 矩阵的合成，合成算法为

$$c_{ij}=\bigvee_{k}\{a_{ik}\wedge b_{kj}\} \tag{3.26}$$

例 3.8 设 $\boldsymbol{A}=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{bmatrix}$，则 $\boldsymbol{C}=\boldsymbol{A}\circ\boldsymbol{B}=\begin{bmatrix}c_{11} & c_{12}\\ c_{21} & c_{22}\end{bmatrix}$，其中

$$c_{11}=(a_{11}\wedge b_{11})\vee(a_{12}\wedge b_{21})$$

$$c_{12}=(a_{11}\wedge b_{12})\vee(a_{12}\wedge b_{22})$$

$$c_{21}=(a_{21}\wedge b_{11})\vee(a_{22}\wedge b_{21})$$

$$c_{22}=(a_{21}\wedge b_{12})\vee(a_{22}\wedge b_{22})$$

当 $\boldsymbol{A}=\begin{bmatrix}0.8 & 0.7\\ 0.5 & 0.3\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0.2 & 0.4\\ 0.6 & 0.9\end{bmatrix}$ 时，有

$$\boldsymbol{A}\circ\boldsymbol{B}=\begin{bmatrix}0.6 & 0.7\\ 0.3 & 0.4\end{bmatrix}$$

$$\boldsymbol{B}\circ\boldsymbol{A}=\begin{bmatrix}0.4 & 0.3\\ 0.6 & 0.6\end{bmatrix}$$

可见，$\boldsymbol{A}\circ\boldsymbol{B}\neq\boldsymbol{B}\circ\boldsymbol{A}$。

采用 Matlab 可实现模糊矩阵的合成，仿真程序见附录程序 chap3_4.m。

3.5　模糊推理

3.5.1　模糊语句

将含有模糊概念的语法规则所构成的语句称为模糊语句。根据其语义和构成语法规则的不同，可分为以下几种类型。

(1) 模糊陈述句：语句本身具有模糊性，又称为模糊命题。如“今天天气很热”。

(2) 模糊判断句：是模糊逻辑中最基本的语句。语句形式：“x 是 a”，记为(a)，且 a 所表示的概念是模糊的。如“张三是好学生”。

(3) 模糊推理句：语句形式：若 x 是 a，则 x 是 b，则$(a)\rightarrow(b)$ 为模糊推理语句。如“今天是晴天，则今天暖和”。

3.5.2　模糊推理

常用的有两种模糊推理语句，即

If A then B else C

If A and B then C

下面以第二种推理语句为例进行探讨，该语句可构成一个简单的模糊控制器，如图 3-10 所示。

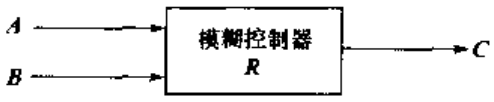


图 3-10　两输入单输出模糊控制器

其中，$\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$ 分别为论域 x,y,z 上的模糊集合，$\boldsymbol{A}$ 为误差信号上的模糊子集，$\boldsymbol{B}$ 为误差变化率上的模糊子集，$\boldsymbol{C}$ 为控制器输出上的模糊子集。

常用的模糊推理有两种方法：Zadeh 法和 Mamdani 法。Mamdani 推理法是一种模糊控制中普遍使用的方法，其本质是一种合成推理方法。

模糊推理语句“If A and B then C”蕴涵的关系为$(\boldsymbol{A}\wedge\boldsymbol{B}\rightarrow\boldsymbol{C})$，根据 Mamdani 模糊推理法，$\boldsymbol{A}\in U$，$\boldsymbol{B}\in U$，$\boldsymbol{C}\in U$ 是三元模糊关系，其关系矩阵 $\boldsymbol{R}$ 为

$$\boldsymbol{R}=(\boldsymbol{A}\times\boldsymbol{B})^{\mathrm{T1}}\times\boldsymbol{C} \tag{3.27}$$

式中，$(\boldsymbol{A}\times\boldsymbol{B})^{\mathrm{T1}}$ 为模糊关系矩阵$(\boldsymbol{A}\times\boldsymbol{B})_{m\times n}$构成的 $m\times n$ 列向量，T1 为列向量转换，n 和 m 分别为 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 论域元素的个数。

基于模糊推理规则，根据模糊关系 $\boldsymbol{R}$，可求得给定输入 $\boldsymbol{A}_1$ 和 $\boldsymbol{B}_1$ 对应的输出 $\boldsymbol{C}_1$，即

$$\boldsymbol{C}_1=(\boldsymbol{A}_1\times\boldsymbol{B}_1)^{\mathrm{T2}}\boldsymbol{R} \tag{3.28}$$

式中，$(\boldsymbol{A}_1,\boldsymbol{B}_1)^{\mathrm{T2}}$ 为模糊关系矩阵$(\boldsymbol{A}_1\times\boldsymbol{B}_1)_{m\times n}$ 构成的 $m\times n$ 行向量，T2 为行向量转换。

例 3.9　设论域 $X=\{a_1,a_2,a_3\}$，$Y=\{b_1,b_2,b_3\}$，$Z=\{c_1,c_2,c_3\}$，已知 $\boldsymbol{A}=\dfrac{0.5}{a_1}+\dfrac{1}{a_2}+\dfrac{0.1}{a_3}$，$\boldsymbol{B}=\dfrac{0.1}{b_1}+\dfrac{1}{b_2}+\dfrac{0.6}{a_3}$，$\boldsymbol{C}=\dfrac{0.4}{c_1}+\dfrac{1}{c_2}$。试确定“If A and B then C”所决定的模糊关系 $\boldsymbol{R}$，以及输入为 $\boldsymbol{A}_1=\dfrac{1.0}{a_1}+\dfrac{0.5}{a_2}+\dfrac{0.1}{a_3}$，$\boldsymbol{B}_1=\dfrac{0.1}{b_1}+\dfrac{1}{b_2}+\dfrac{0.6}{b_3}$ 时的输出 $\boldsymbol{C}_1$。

解

$$\boldsymbol{A}\times\boldsymbol{B}=\begin{bmatrix}0.5\\1\\0.1\end{bmatrix}\circ[0.1\quad 1\quad 0.6]=\begin{bmatrix}0.1&0.5&0.5\\0.1&1.0&0.6\\0.1&0.1&0.1\end{bmatrix}$$

将 $\boldsymbol{A}\times\boldsymbol{B}$ 矩阵扩展成如下列向量

$(\boldsymbol{A}\times\boldsymbol{B})^{T1}=[0.1\quad 0.5\quad 0.5\quad 0.1\quad 1.0\quad 0.6\quad 0.1\quad 0.1\quad 0.1]^{T}$

$\boldsymbol{R}=(\boldsymbol{A}\times\boldsymbol{B})^{T1}\times\boldsymbol{C}=[0.1\quad 0.5\quad 0.5\quad 0.1\quad 1.0\quad 0.6\quad 0.1\quad 0.1\quad 0.1]^{T}\circ$

$$[0.4\quad 1]=\begin{bmatrix}0.1&0.4&0.4&0.1&0.4&0.4&0.1&0.1&0.1\\0.1&0.5&0.5&0.1&1&0.6&0.1&0.1&0.1\end{bmatrix}^{T}$$

当输入为 $\boldsymbol{A}_1$ 和 $\boldsymbol{B}_1$ 时，有

$$(\boldsymbol{A}_1\times\boldsymbol{B}_1)=\begin{bmatrix}1\\0.5\\0.1\end{bmatrix}\circ[0.1\quad 0.5\quad 1]=\begin{bmatrix}0.1&0.5&1\\0.1&0.5&0.5\\0.1&0.1&0.1\end{bmatrix}$$

将 $\boldsymbol{A}_1\times\boldsymbol{B}_1$ 矩阵扩展成如下行向量

$$(\boldsymbol{A}\times\boldsymbol{B})^{T2}=[0.1\quad 0.5\quad 1\quad 0.1\quad 0.5\quad 0.5\quad 0.1\quad 0.1\quad 0.1]$$

最后得 $\boldsymbol{C}_1$ 为

$$\boldsymbol{C}_1=[0.1\quad 0.5\quad 1\quad 0.1\quad 0.5\quad 0.5\quad 0.1\quad 0.1\quad 0.1]\circ$$

$$\begin{bmatrix}0.1&0.4&0.4&0.1&0.4&0.4&0.1&0.1&0.1\\0.1&0.5&0.5&0.1&1&0.6&0.1&0.1&0.1\end{bmatrix}^{T}=[0.4\quad 0.5]$$

即 $\boldsymbol{C}_1=\dfrac{0.4}{c_1}+\dfrac{0.5}{c_2}$。

采用 Matlab 实现上述过程的仿真，模糊推理仿真程序见附录程序 chap3_5.m。

3.5.3 模糊关系方程

1. 模糊关系方程概念

将模糊关系 $\boldsymbol{R}$ 看成一个模糊变换器。当 $\boldsymbol{A}$ 为输入时，$\boldsymbol{B}$ 为输出，如图 3-11 所示。

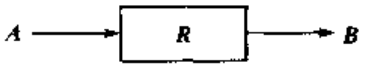


图 3-11 模糊变换器

可分为两种情况进行讨论：

(1) 已知输入 $\boldsymbol{A}$ 和模糊关系 $\boldsymbol{R}$，求输出 $\boldsymbol{B}$，这是综合评判，即模糊变换问题；

(2) 已知输入 $\boldsymbol{A}$ 和输出 $\boldsymbol{B}$，求模糊关系 $\boldsymbol{R}$，或已知模糊关系 $\boldsymbol{R}$ 和输出 $\boldsymbol{B}$，求输入 $\boldsymbol{A}$，这是模糊综合评判的逆问题，需要求解模糊关系方程。

2. 模糊关系方程的解

近似试探法是目前实际应用中较为常用的方法之一。

例 3.10 解方程 $(0.6\quad 0.2\quad 0.4)\circ\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=0.4$。

解　由方程得

$$(0.6 \wedge x_1) \vee (0.2 \wedge x_2) \vee (0.4 \wedge x_3) = 0.4$$

显然3个括弧内的值都不可能超过0.4。由于$(0.2 \wedge x_2) < 0.4$是显然的，因此x_2可以取$[0,1]$的任意值，即$(x_2) = [0,1]$。

现在只考虑

$$(0.6 \wedge x_1) \vee (0.4 \wedge x_3) = 0.4$$

这两个括号内的值可以是：其中一个等于 0.4，另一个不超过 0.4。分两种情况讨论：

(1) 设 $0.6 \wedge x_1 = 0.4, 0.4 \wedge x_3 \leqslant 0.4$，则

$$(x_1) = 0.4, (x_3) = [0,1]$$

即方程的解为$(x_1) = 0.4, (x_2) = [0,1], (x_3) = [0,1]$。

(2) 设 $0.6 \wedge x_1 \leqslant 0.4, 0.4 \wedge x_3 = 0.4$，则

$$(x_1) = [0,0.4], (x_3) = [0.4,1]$$

即方程的解为$(x_1) = [0,0.4], (x_2) = [0,1], (x_3) = [0.4,1]$。

思考题与习题

3-1　已知年龄的论域为$[0,200]$，且设“年老 O”和“年轻 Y”两个模糊集的隶属函数分别为

$$\mu_O(a) = \begin{cases} 0 & 0 \leqslant a \leqslant 50 \\ \left[1 + \left(\dfrac{a-50}{5}\right)^2\right]^{-1} & 50 < a \leqslant 200 \end{cases}$$

$$\mu_Y(a) = \begin{cases} 0 & 0 \leqslant a \leqslant 25 \\ \left[1 + \left(\dfrac{a-50}{5}\right)^2\right]^{-1} & 25 < a \leqslant 200 \end{cases}$$

试设计“很年轻 W”、“不老也不年轻 V”两个模糊集的隶属函数，并采用 Matlab 实现针对上述 4 个隶属函数的仿真。

3-2　已知模糊矩阵 $\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{S}$，$\boldsymbol{P} = \begin{bmatrix} 0.6 & 0.9 \\ 0.2 & 0.7 \end{bmatrix}$，$\boldsymbol{Q} = \begin{bmatrix} 0.5 & 0.7 \\ 0.1 & 0.4 \end{bmatrix}$，$\boldsymbol{R} = \begin{bmatrix} 0.2 & 0.3 \\ 0.7 & 0.7 \end{bmatrix}$，$\boldsymbol{S} = \begin{bmatrix} 0.1 & 0.2 \\ 0.6 & 0.5 \end{bmatrix}$。求：

(1) $(\boldsymbol{P} \circ \boldsymbol{Q}) \circ \boldsymbol{R}$；

(2) $(\boldsymbol{P} \cup \boldsymbol{Q}) \circ \boldsymbol{S}$；

(3) $(\boldsymbol{P} \circ \boldsymbol{S}) \cup (\boldsymbol{Q} \circ \boldsymbol{S})$。

3-3　求解模糊关系方程

$$\begin{bmatrix} 0.8 & 0.5 & 0.6 \\ 0.4 & 0.8 & 0.5 \end{bmatrix} \circ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix}$$

3-4　假设 $\boldsymbol{A} = \dfrac{1}{x_1} + \dfrac{0.5}{x_2}$ 且 $\boldsymbol{B} = \dfrac{0.1}{y_1} + \dfrac{0.5}{y_2} + \dfrac{1}{y_3}$，则 $\boldsymbol{C} = \dfrac{0.2}{z_1} + \dfrac{1}{z_2}$。现已知 $\boldsymbol{A}_1 = \dfrac{0.8}{x_1} + \dfrac{0.1}{x_2}$ 且 $\boldsymbol{B}_1 = \dfrac{0.5}{y_1} + \dfrac{0.2}{y_2} + \dfrac{0}{y_3}$，求 $\boldsymbol{C}_1$，并采用 Matlab 进行仿真。

附录（程序代码）

"年轻"的隶属函数仿真程序:chap3_1.m

```
% Membership function for Young People
clear all;
close all;

for k=1:1:1001
    x(k)=(k-1)*0.10;
if x(k)>=0&x(k)<=25
   y(k)=1.0;
else
   y(k)=1/(1+((x(k)-25)/5)^2);
end
end
plot(x,y,'k');
xlabel('X Years');ylabel('Degree of membership');
```

典型隶属函数仿真程序:chap3_2.m

```
% Membership function
clear all;
close all;

M=6;
if M==1                % Guassian membership function
    x=0:0.1:10;
    y=gaussmf(x,[2 5]);
    plot(x,y,'k');
    xlabel('x');ylabel('y');
elseif M==2            % General Bell membership function
    x=0:0.1:10;
    y=gbellmf(x,[2 4 6]);
    plot(x,y,'k');
    xlabel('x');ylabel('y');
elseif M==3            % S membership function
    x=0:0.1:10;
    y=sigmf(x,[2 4]);
    plot(x,y,'k');
    xlabel('x');ylabel('y');
elseif M==4            % Trapezoid membership function
```

```
    x=0:0.1:10;
    y=trapmf(x,[1 5 7 8]);
    plot(x,y,'k');
    xlabel('x');ylabel('y');
elseif M==5          % Triangle membership function
    x=0:0.1:10;
    y=trimf(x,[3 6 8]);
    plot(x,y,'k');
    xlabel('x');ylabel('y');
elseif M==6          % Z membership function
    x=0:0.1:10;
    y=zmf(x,[3 7]);
    plot(x,y,'k');
    xlabel('x');ylabel('y');
end
```

模糊系统隶属函数设计程序:chap3_3.m

```
% Define N+1 triangle membership function
clear all;
close all;
N=6;

x=-3:0.01:3;
for i=1:N+1
    f(i)=-3+6/N*(i-1);
end
u=trimf(x,[f(1),f(1),f(2)]);

figure(1);
plot(x,u);
for j=2:N
    u=trimf(x,[f(j-1),f(j),f(j+1)]);
    hold on;
    plot(x,u);
end
u=trimf(x,[f(N),f(N+1),f(N+1)]);
hold on;
plot(x,u);
xlabel('x');
ylabel('Degree of membership');
```

模糊矩阵合成仿真程序:chap3_4.m

```
clear all;
close all;
A=[0.8,0.7;
   0.5,0.3];
B=[0.2,0.4;
   0.6,0.9];
% Compound of A and B
for i=1:2
   for j=1:2
      AB(i,j)=max(min(A(i,:),B(:,j)'))
    end
end

% Compound of B and A
for i=1:2
   for j=1:2
      BA(i,j)=max(min(B(i,:),A(:,j)'))
    end
end
```

模糊推理仿真程序:chap3_5.m

```
clear all;
close all;

A=[0.5;1;0.1];
B=[0.1,1,0.6];
C=[0.4,1];

% Compound of A and B
for i=1:3
   for j=1:3
      AB(i,j)=min(A(i),B(j));
   end
end
% Transfer to Column
T1=[];
for i=1:3
   T1=[T1;AB(i,:)'];
```

```
end
%Get fuzzy R
for i=1:9
    for j=1:2
        R(i,j)=min(T1(i),C(j));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A1=[1,0.5,0.1];
B1=[0.1,0.5,1];

for i=1:3
    for j=1:3
        AB1(i,j)=min(A1(i),B1(j));
    end
end
%Transfer to Row
T2=[];
for i=1:3
    T2=[T2,AB1(i,:)];
end
%Get output C1
for i=1:9
    for j=1:2
        D(i,j)=min(T2(i),R(i,j));
        C1(j)=max(D(:,j));
    end
end
```

第4章 模糊控制

4.1 模糊控制的基本原理

4.1.1 模糊控制原理

模糊控制(Fuzzy Control)是以模糊集理论、模糊语言变量和模糊逻辑推理为基础的一种智能控制方法，它从行为上模仿人的模糊推理和决策过程。该方法首先将操作人员或专家经验编成模糊规则，然后将来自传感器的实时信号模糊化，将模糊化后的信号作为模糊规则的输入，完成模糊推理，将推理后得到的输出量加到执行器上。

模糊控制的基本原理框图如图 4-1 所示。它的核心部分为模糊控制器，如图中点画线框中部分所示，模糊控制器的控制律由计算机的程序实现。实现一步模糊控制算法的过程描述如下：微机经中断采样获取被控制量的精确值，然后将此量与给定值比较得到误差信号 E，一般选误差信号 E 作为模糊控制器的一个输入量。把误差信号 E 的精确量进行模糊化变成模糊量。误差 E 的模糊量可用相应的模糊语言表示，得到误差 E 的模糊语言集合的一个子集 $\boldsymbol{e}$（$\boldsymbol{e}$ 是一个模糊矢量），再由 $\boldsymbol{e}$ 和模糊关系 $\boldsymbol{R}$ 根据推理的合成规则进行模糊决策，得到模糊控制量 $\boldsymbol{u}$，即

$$\boldsymbol{u} = \boldsymbol{e} \circ \boldsymbol{R} \tag{4.1}$$

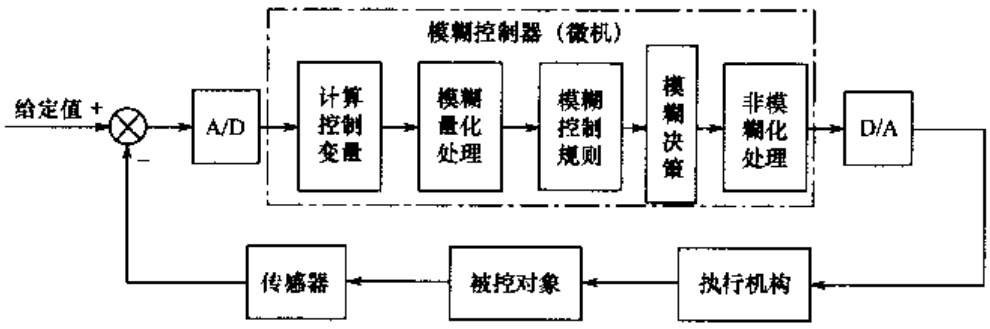


图 4-1 模糊控制原理框图

由图 4-1 可知，模糊控制系统与通常的计算机数字控制系统的主要差别是采用了模糊控制器。模糊控制器是模糊控制系统的核心，一个模糊控制系统的性能优劣，主要取决于模糊控制器的结构、所采用的模糊规则、合成推理算法及模糊决策的方法等因素。

模糊控制器(Fuzzy Controller，FC) 也称为模糊逻辑控制器(Fuzzy Logic Controller，FLC)，由于所采用的模糊控制规则是由模糊理论中模糊条件语句来描述的，因此，模糊控制器是一种语言型控制器，故也称为模糊语言控制器(Fuzzy Language Controller，FLC)。

4.1.2　模糊控制器的组成

模糊控制器的组成框图如图 4-2 所示。

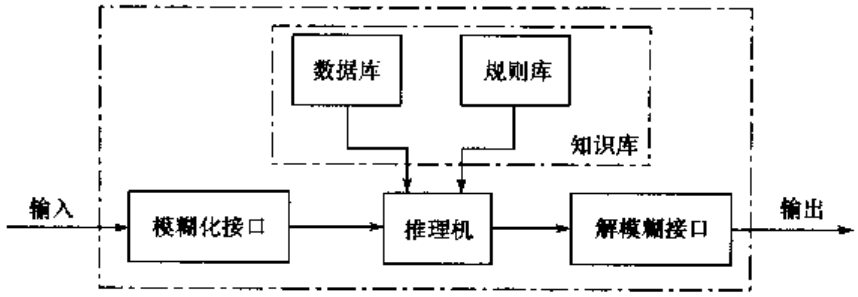


图 4-2　模糊控制器的组成框图

1. 模糊化接口(Fuzzy Interface)

模糊控制器的输入必须通过模糊化才能用于控制输出，因此，它实际上是模糊控制器的输入接口，其主要作用是将真实的确定量输入转换为一个模糊矢量。对于一个模糊输入变量 e，其模糊子集通常可以进行如下方式划分：

(1)e = {负大，负小，零，正小，正大} = {NB, NS, ZO, PS, PB}

(2)e = {负大，负中，负小，零，正小，正中，正大} = {NB, NM, NS, ZO, PS, PM, PB}

(3)e = {大，负中，负小，零负，零正，正小，正中，正大} = {NB, NM, NS, NZ, PZ, PS, PM, PB}

将(3)用三角形隶属度函数表示，如图 4-3 所示。

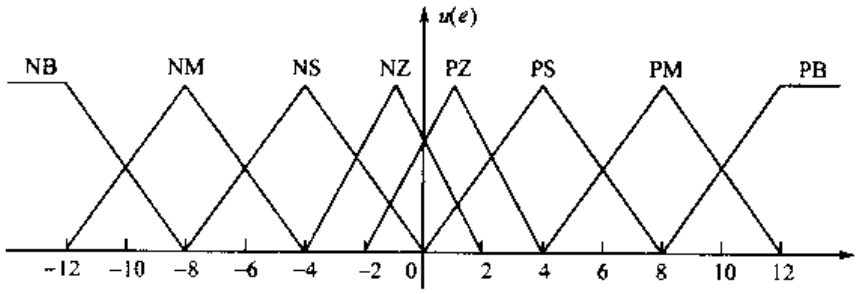


图 4-3　模糊子集和模糊化等级

2. 知识库(Knowledge Base, KB)

知识库由数据库和规则库两部分构成。

(1) 数据库(Data Base, DB)

数据库所存放的是所有输入、输出变量的全部模糊子集的隶属度矢量值(即经过论域等级离散化以后对应值的集合)，若论域为连续域，则为隶属度函数。在规则推理的模糊关系方程求解过程中，向推理机提供数据。

(2) 规则库(Rule Base, RB)

模糊控制器的规则库基于专家知识或手动操作人员长期积累的经验，它是按人的直觉推理的一种语言表示形式。模糊规则通常有一系列的关系词连接而成，如 if-then, else, also, end, or 等，关系词必须经过“翻译”才能将模糊规则数值化。最常用的关系词为 if-then, also，对于多

变量模糊控制系统，还有 and 等。例如，某模糊控制系统输入变量为 e(误差) 和 ec(误差变化)，它们对应的语言变量为 E 和 EC，可给出一组模糊规则为

R_1: IF E is NB and EC is NB then U is PB

R_2: IF E is NB and EC is NS then U is PM

通常把 if… 部分称为“前提部”，而 then… 部分称为“结论部”，其基本结构可归纳为 If A and B then C，其中 A 为论域 U 上的一个模糊子集，B 是论域 V 上的一个模糊子集。根据人工控制经验，可离线组织其控制决策表 R，R 是笛卡儿乘积集 $U\times V$ 上的一个模糊子集，则某一时刻其控制量由下式给出

$$\boldsymbol{C}=(\boldsymbol{A}\times\boldsymbol{B})\circ\boldsymbol{R} \tag{4.2}$$

式中，$\times$ 为模糊直积运算；$\circ$ 为模糊合成运算。

规则库是用来存放全部模糊控制规则的，在推理时为“推理机”提供控制规则。由上述可知，规则的条数与模糊变量的模糊子集划分有关，划分越细，规则条数越多，但并不代表规则库的准确度越高，规则库的“准确性”还与专家知识的准确度有关。

3. 推理与解模糊接口(Inference and Defuzzy-Interface)

推理是模糊控制器中，根据输入模糊量，由模糊控制规则完成模糊推理来求解模糊关系方程，并获得模糊控制量的功能部分。在模糊控制中，考虑到推理时间，通常采用运算较简单的推理方法。最基本的有 Zadeh 近似推理，它包含正向推理和逆向推理两类。正向推理常被用于模糊控制中，而逆向推理一般用于知识工程学领域的专家系统中。

推理结果的获得，表示模糊控制的规则推理功能已经完成。但是，至此所获得的结果仍是一个模糊矢量，不能直接用来作为控制量，还必须进行一次转换，求得清晰的控制量输出，即为解模糊。通常把输出端具有转换功能作用的部分称为解模糊接口。

综上所述，模糊控制器实际上就是依靠微机(或单片机) 来构成的。它的绝大部分功能都是由计算机程序来完成的。随着专用模糊芯片的研究和开发，也可以由硬件逐步取代各组成单元的软件功能。

4.1.3 模糊控制系统的工作原理

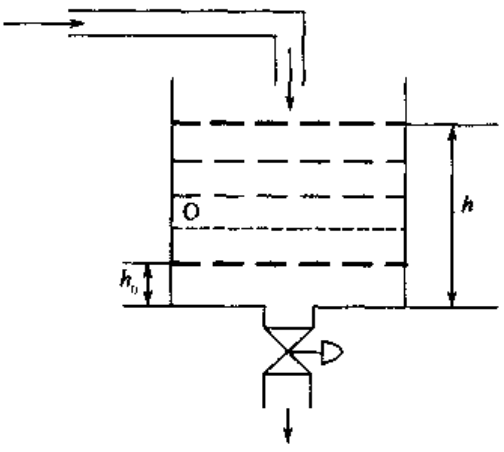


图 4-4 水箱液位控制

以水位的模糊控制为例。如图 4-4 所示，设有一个水箱，通过调节阀可向内注水和向外抽水。设计一个模糊控制器，通过调节阀门将水位稳定在固定点附近。按照日常的操作经验，可以得到基本的控制规则为：

“若水位高于 O 点，则向外排水，差值越大，排水越快”；

“若水位低于 O 点，则向内注水，差值越大，注水越快”。

根据上述经验，可按下列步骤设计一维模糊控制器。

1. 确定观测量和控制量

定义理想液位 O 点的水位为 h_0，实际测得的水位高度为 h，选择液位差为

$$e = \Delta h = h_0 - h$$

将当前水位对于 O 点的偏差 e 作为观测量。

2. 输入量和输出量的模糊化

将偏差 e 分为5个模糊集：负大(NB)，负小(NS)，零(ZO)，正小(PS)，正大(PB)。将偏差 e 的变化分为 7 个等级：$-3, -2, -1, 0, +1, +2, +3$，从而得到水位变化模糊表，见表 4-1。

表 4-1　水位变化 e 划分表

隶属度		变化等级						
		−3	−2	1	0	1	2	3
模糊集	PB	0	0	0	0	0	0.5	1
	PS	0	0	0	0	1	0.5	0
	ZO	0	0	0.5	1	0.5	0	0
	NS	0	0.5	1	0	0	0	0
	NB	1	0.5	0	0	0	0	0

控制量 u 为调节阀门开度的变化。将其分为 5 个模糊集：负大(NB)，负小(NS)，零(ZO)，正小(PS)，正大(PB)。将 u 的变化分为 9 个等级：$-4, -3, -2, -1, 0, +1, +2, +3, +4$。得到控制量模糊划分表，见表 4-2。

表 4-2　控制量 u 变化划分表

隶属度		变化等级								
		−4	−3	−2	−1	0	1	2	3	4
模糊集	PB	0	0	0	0	0	0	0	0.5	1
	PS	0	0	0	0	0	0.5	1	0.5	0
	ZO	0	0	0	0.5	1	0.5	0	0	0
	NS	0	0.5	1	0.5	0	0	0	0	0
	NB	1	0.5	0	0	0	0	0	0	0

3. 模糊规则的描述

根据日常的经验，设计以下模糊规则：

(1)“若 e 负大，则 u 负大”

(2)“若 e 负小，则 u 负小”

(3)“若 e 为零，则 u 为零”

(4)“若 e 正小，则 u 正小”

(5)“若 e 正大，则 u 正大”

其中，排水时 u 为负，注水时 u 为正。

将上述规则采用“IF　A　THEN　B”的形式来描述，则模糊规范表示为

(1)if　e = NB　then　u = NB

(2)if e = NS then u = NS

(3)if e = ZO then u = ZO

(4)if e = PS then u = PS

(5)if e = PB then u = PB

根据上述经验规则，可得模糊控制表，见表 4-3。

表 4-3 模糊控制规则表

若(IF)	NB*e*	NS*e*	ZO*e*	PS*e*	PB*e*
则(THEN)	NB*u*	NS*u*	ZO*u*	PS*u*	PB*u*

4. 求模糊关系

模糊控制规则是一个多条语句，它可以表示为 $U \times V$ 上的模糊子集，即模糊关系 $\boldsymbol{R}$ 为

$$\boldsymbol{R} = (\mathrm{NB}e \times \mathrm{NB}u) \cup (\mathrm{NS}e \times \mathrm{NS}u) \cup (\mathrm{ZO}e \times \mathrm{ZO}u) \cup (\mathrm{PS}e \times \mathrm{PS}u) \cup (\mathrm{PB}e \times \mathrm{PB}u)$$

其中规则内的模糊集运算取交集，规则间的模糊集运算取并集，即

$$\mathrm{NB}e \times \mathrm{NB}u = \begin{bmatrix} 1 \\ 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times [1 \ 0.5 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] = \begin{bmatrix} 1.0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathrm{NS}e \times \mathrm{NS}u = \begin{bmatrix} 0 \\ 0.5 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times [0 \ 0.5 \ 1 \ 0.5 \ 0 \ 0 \ 0 \ 0 \ 0] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathrm{ZO}e \times \mathrm{ZO}u = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 1.0 \\ 0.5 \\ 0 \\ 0 \end{bmatrix} \times [0 \ 0 \ 0 \ 0.5 \ 1 \ 0.5 \ 0 \ 0 \ 0] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$PSe \times PSu = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1.0 \\ 0.5 \\ 0 \end{bmatrix} \times [0\ 0\ 0\ 0\ 0\ 0.5\ 1.0\ 0.5\ 0] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$PBe \times PBu = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \\ 1.0 \end{bmatrix} \times [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0.5\ 1.0] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 1.0 \end{bmatrix}$$

由以上 5 个模糊矩阵求并集，得

$$\boldsymbol{R} = \begin{bmatrix} 1.0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 1.0 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 1.0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 1.0 \end{bmatrix}$$

5. 模糊决策

模糊控制器的输出为误差向量和模糊关系的合成，即

$$\boldsymbol{u} = \boldsymbol{e} \circ \boldsymbol{R}$$

当误差 $\boldsymbol{e}$ 为 NB 时，$\boldsymbol{e} = [1.0\quad 0.5\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0]$，控制器输出为

$$\boldsymbol{u}=\boldsymbol{e}\circ\boldsymbol{R}=[1\quad 0.5\quad 0\quad 0\quad 0\quad 0\quad 0]\circ\begin{bmatrix}1.0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0\\ 0 & 0.5 & 1.0 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0\\ 0 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0\\ 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 1.0 & 0.5 & 0\\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 1.0\end{bmatrix}$$

$$=[1\quad 0.5\quad 0.5\quad 0.5\quad 0\quad 0\quad 0\quad 0\quad 0]$$

6. 控制量的反模糊化

由模糊决策可知，当误差为负大时，实际液位远高于理想液位，e = NB，控制器的输出为一模糊矢量，可表示为

$$\boldsymbol{u}=\frac{1}{-4}+\frac{0.5}{-3}+\frac{0.5}{-2}+\frac{0.5}{-1}+\frac{0}{0}+\frac{0}{+1}+\frac{0}{+2}+\frac{0}{+3}+\frac{0}{+4}$$

如果按照“隶属度最大原则”进行反模糊化，选择控制量为 $u=-4$，即阀门的开度应关大一些，减少进水量，加大排水量。

按照上述步骤，设计水箱液位模糊控制的 Matlab 仿真程序，见附录程序 chap4_1.m。取 flag = 1，可得到模糊系统的规则库并可实现模糊控制的动态仿真。模糊控制响应表见表 4-4。取偏差 $e=-3$，得 $u=-3.1481$。

表 4-4　模糊控制响应表

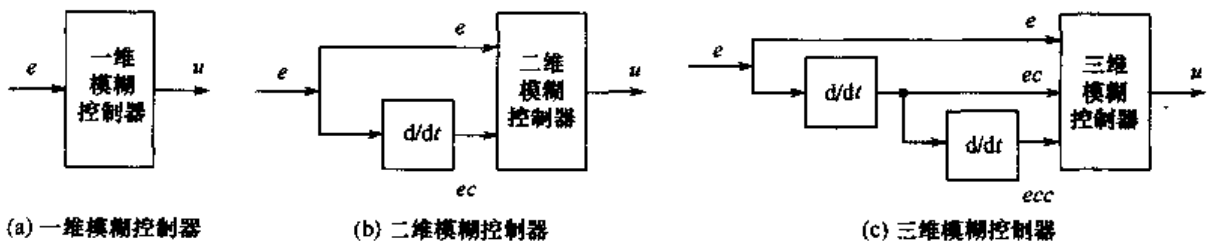
e	−3	−2	−1	0	1	2	3
u	−3	−2	−1	0	1	2	3

4.1.4　模糊控制器的结构

在确定性控制系统中，根据输入变量和输出变量的个数，可分为单变量控制系统和多变量控制系统。在模糊控制系统中，也可类似地划分为单变量模糊控制和多变量模糊控制。

1. 单变量模糊控制器

在单变量模糊控制器(Single Variable Fuzzy Controller，SVFC) 中，将其输入变量的个数定义为模糊控制的维数，如图 4-5 所示。



(a) 一维模糊控制器　(b) 二维模糊控制器　(c) 三维模糊控制器

图 4-5　单变量模糊控制器

(1) 一维模糊控制器

如图 4-4(a) 所示，一维模糊控制器的输入变量往往选择为受控变量和输入给定值的偏差 e。由于仅仅采用偏差值，很难反映过程的动态特性品质，因此，所能获得的系统动态性能是不能令人满意的。这种一维模糊控制器往往被用于一阶被控对象。

(2) 二维模糊控制器

如图 4-4(b) 所示，二维模糊控制器的两个输入变量基本上都选用受控变量值和输入给定值的偏差 e 和偏差变化 ec，由于它们能够较严格地反映受控过程中输出量的动态特性，因此，在控制效果上要比一维控制器好得多，也是目前采用较广泛的一类模糊控制器。

(3) 三维模糊控制器

如图 4-4(c) 所示，三维模糊控制器的 3 个输入变量分别为系统偏差量 e、偏差变化量 ec 和偏差变化的变化率 ecc。由于这种模糊控制器结构较复杂，推理运算时间长，因此，除对动态特性的要求特别高的场合之外，一般较少选用三维模糊控制器。

上述 3 类模糊控制器的输出变量，均选择了受控变量的变化值。从理论上讲，模糊控制系统所选用的模糊控制器维数越高，系统的控制精度也就越高。但是维数选择太高，模糊控制律就过于复杂，基于模糊合成推理的控制算法的计算机实现也就更困难，这是人们在设计模糊控制系统时多数采用二维控制器的原因。在需要时，为了获得较好的上升段特性和改善控制器的动态品质，也可以对模糊控制器的输出量进行分段选择，即在偏差 e“大”时，以控制量的值为输出；而当偏差 e“小”或“中等”时，则以控制量的增量为输出。

2. 多变量模糊控制器

一个多变量模糊控制器 (Multiple Variable Fuzzy Controller, MVFC) 所采用的模糊控制器具有多变量结构，如图 4-6 所示。

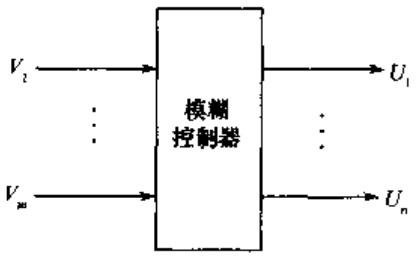


图 4-6　多变量模糊控制器

要直接设计一个多变量模糊控制器是相当困难的，可利用模糊控制器本身的解耦特点，通过模糊关系方程求解，在控制器结构上实现解耦，即将一个多输入、多输出 (MIMO) 的模糊控制器，分解成若干个多输入、单输出 (MISO) 的模糊控制器，这样可采用单变量模糊控制方法进行设计。

4.2　模糊控制系统分类

1. 按信号的时变特性分类

(1) 恒值模糊控制系统

系统的指令信号为恒定值，通过模糊控制器消除外界对系统的扰动作用，使系统的输出跟踪输入的恒定值。也称为“自镇定模糊控制系统”，如温度模糊控制系统。

(2) 随动模糊控制系统

系统的指令信号为时间函数，要求系统的输出高精度、快速地跟踪系统输入。也称为“模糊控制跟踪系统”或“模糊控制伺服系统”。

2. 按模糊控制的线性特性分类

对开环模糊控制系统 S，设输入变量为 u，输出变量为 v。对任意输入偏差 Δu 和输出偏差 Δv，满足 $\frac{\Delta v}{\Delta u}=k, u\in U, v\in V$。

定义线性度 δ，用于衡量模糊控制系统的线性化程度，即

$$\delta=\frac{\Delta v_{\max}}{2\xi\Delta u_{\max}m} \tag{4.3}$$

式中，$\Delta v_{\max}=v_{\max}-v_{\min}$，$\Delta u_{\max}=u_{\max}-u_{\min}$，$\xi$ 为线性化因子，m 为模糊子集 V 的个数。

设 k_0 为一经验值，则定义模糊系统的线性特性为：① 当 $|k-k_0|\leqslant\delta$ 时，系统 S 为线性模糊系统；② 当 $|k-k_0|>\delta$ 时，系统 S 为非线性模糊系统。

3. 按静态误差是否存在分类

(1) 有差模糊控制系统

将偏差的大小及其偏差变化率作为系统的输入，为有差模糊控制系统。

(2) 无差模糊控制系统

在有差模糊控制系统基础上，引入积分作用，使系统的静差降至最小，为无差模糊控制系统。

4. 按系统输入变量的多少分类

控制输入个数为 1 的系统为单变量模糊控制系统，控制输入个数大于 1 的系统为多变量模糊控制系统。

4.3 模糊控制器的设计

4.3.1 模糊控制器的设计步骤

模糊控制器最简单的实现方法是将一系列模糊控制规则离线转化为一个查询表（又称为控制表），存储在计算机中供在线控制时使用。这种模糊控制器结构简单，使用方便，是最基本的一种形式。本节以单变量二维模糊控制器为例，介绍这种形式模糊控制器的设计步骤，其设计思想是设计其他模糊控制器的基础。模糊控制器的设计步骤如下：

1. 模糊控制器的结构

单变量二维模糊控制器是最常见的结构形式。

2. 定义输入、输出模糊集

对误差 e、误差变化 ec 及控制量 u 的模糊集及其论域定义如下：

e，ec 和 u 的模糊集均为：{NB，NM，NS，ZO，PS，PM，PB}

e，ec 的论域均为：{−3，−2，−1，0，1，2，3}

u 的论域为：{−4.5，−3，−1.5，0，1，3，4.5}

3. 定义输入、输出隶属函数

误差e、误差变化ec及控制量u的模糊集和论域确定后，需对模糊变量确定隶属函数，即对模糊变量赋值，确定论域内元素对模糊变量的隶属度。

4. 建立模糊控制规则

根据人的直觉思维推理，由系统输出的误差及误差的变化趋势来设计消除系统误差的模糊控制规则。模糊控制规则语句构成了描述众多被控过程的模糊模型。例如，卫星的姿态与作用的关系、飞机或舰船航向与舵偏角的关系、工业锅炉中的压力与加热的关系等，都可用模糊规则来描述。在条件语句中，误差e、误差变化ec及控制量u对于不同的被控对象有着不同的意义。

5. 建立模糊控制表

上述描写的模糊控制规则可采用模糊规则表4-5来描述，表中共49条模糊规则，各个模糊语句之间是“或”的关系，由第一条语句所确定的控制规则可以计算出u_1。同理，可以由其余各条语句分别求出控制量$u_2,\cdots,u_{49}$，则控制量为模糊集合U，可表示为

$$U = u_1 + u_2 + \cdots + u_{49} \tag{4.4}$$

表 4-5　模糊规则表

U		e						
		NB	NM	NS	ZO	PS	PM	PB
ec	NB	PB	PB	PM	PM	PS	ZO	ZO
	NM	PB	PB	PM	PS	PS	ZO	NS
	NS	PM	PM	PM	PS	ZO	NS	NS
	ZO	PM	PM	PS	ZO	NS	NM	NM
	PS	PS	PS	ZO	NS	NS	NM	NM
	PM	PS	ZO	NS	NM	NM	NM	NB
	PB	ZO	ZO	NM	NM	NM	NB	NB

6. 模糊推理

模糊推理是模糊控制的核心，它利用某种模糊推理算法和模糊规则进行推理，得出最终的控制量。

7. 反模糊化

通过模糊推理得到的结果是一个模糊集合。但在实际模糊控制中，必须要有一个确定值才能控制或驱动执行机构。将模糊推理结果转化为精确值的过程称为反模糊化。常用的反模糊化有3种。

(1) 最大隶属度法

选取推理结果的模糊集合中隶属度最大的元素作为输出值，即$v_0 = \max\mu_v(v), v \in V$。如果在输出论域$V$中，其最大隶属度对应的输出值多于一个，则取所有具有最大隶属度输出的平均值，即

$$v_o = \frac{1}{N}\sum_{i=1}^{N} v_i, v_i = \max_{v \in V}(\mu_v(v)) \tag{4.5}$$

式中，N 为具有相同最大隶属度输出的总数。

最大隶属度法不考虑输出隶属度函数的形状，只考虑最大隶属度处的输出值。因此，难免会丢失许多信息。其突出优点是计算简单。在一些控制要求不高的场合，可采用最大隶属度法。

(2) 重心法

为了获得准确的控制量，就要求模糊方法能够很好地表达输出隶属度函数的计算结果。重心法是取隶属度函数曲线与横坐标围成面积的重心作为模糊推理的最终输出值，即

$$v_o = \frac{\int_V v\mu_v(v)\mathrm{d}v}{\int_V \mu_v(v)\mathrm{d}v} \tag{4.6}$$

对于具有 m 个输出量化级数的离散域情况有

$$v_o = \frac{\sum_{k=1}^{m} v_k\mu_v(v_k)}{\sum_{k=1}^{m} \mu_v(v_k)} \tag{4.7}$$

与最大隶属度法相比较，重心法具有更平滑的输出推理控制。即使对应于输入信号的微小变化，输出也会发生变化。

(3) 加权平均法

工业控制中广泛使用的反模糊方法为加权平均法，输出值由下式决定

$$v_o = \frac{\sum_{i=1}^{m} v_i k_i}{\sum_{i=1}^{m} k_i} \tag{4.8}$$

式中，系数 k_i 的选择根据实际情况而定。不同的系数决定系统具有不同的响应特性。当系数 k_i 取隶属度 $\mu_v(v_i)$ 时，就转化为重心法。

反模糊化方法的选择与隶属度函数形状的选择、推理方法的选择相关。Matlab 提供 5 种反模糊化方法：①centroid，面积重心法；②bisector，面积等分法；③mom，最大隶属度平均法；④som，最大隶属度取小法；⑤lom，最大隶属度取大法。在 Matlab 中，可通过 setfis() 设置反模糊化方法，通过 defuzz() 执行反模糊化运算。

例如，重心法通过下例程序来实现：

```
x = -10:1:10;
mf = trapmf(x,[-10, -8, -4,7]);
xx = defuzz(x,mf,'centroid');
```

在模糊控制中，重心法可通过下例语句来设定：

```
a1 = setfis(a,'DefuzzMethod','centroid')
```

其中，a 为模糊规则库。

4.3.2 模糊控制器的 Matlab 仿真

根据上述步骤，建立两输入、单输出的模糊控制系统，该系统包括两个部分，即模糊控制器

的设计和位置跟踪。

1. 模糊控制器的设计

模糊规则表见表 4-5，控制规则为 49 条。误差 e、误差变化率 ec 和控制输入 u 的范围均为 $[-3.0,3.0]$。通过运行 showrule(a)，可得到用于描述模糊系统的 49 条模糊规则。控制器的响应表见表 4-6。

表 4-6　模糊响应表

ec \ e	−3	−2	−1	0	1	2	3
−3	−3	−2	−2	−1	−1	0	1
−2	−2	−2	−2	1	0	1	1
−1	−2	−2	−1	0	1	1	2
−1	−1	−1	0	1	1	2	2
−1	−1	0	1	1	2	3	3
2	0	1	1	2	3	3	3
3	1	1	2	2	3	3	4

模糊控制器的设计仿真程序见附录程序 chap4_2.m。在仿真时，模糊推理系统可由命令 plotfis(a2) 得到。系统的输入、输出隶属度函数如图 4-7 至图 4-9 所示。

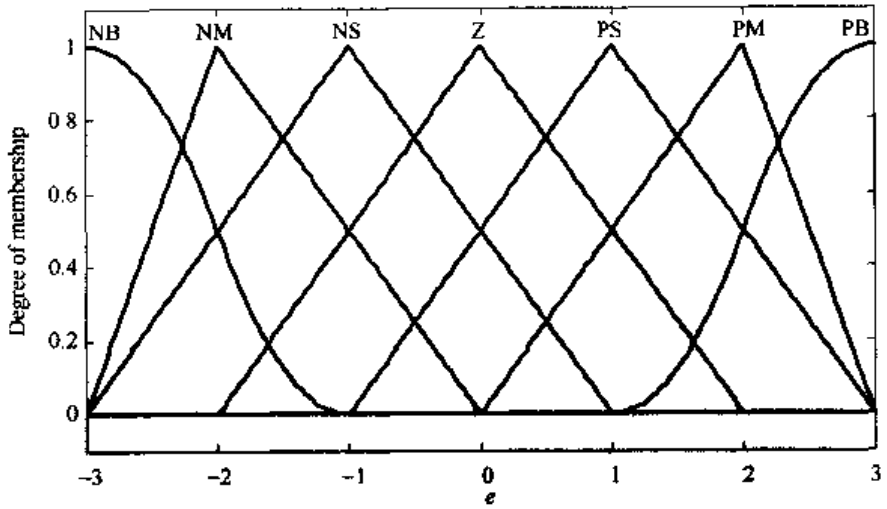


图 4-7　偏差隶属度函数

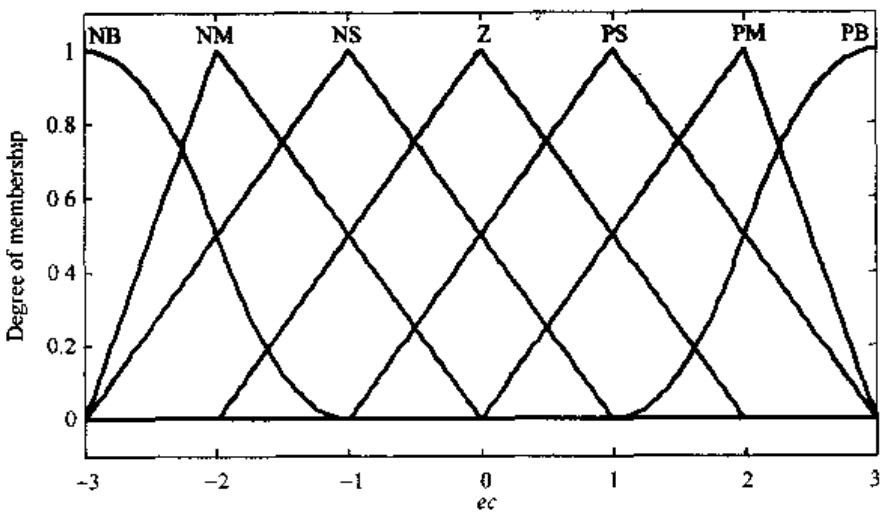


图 4-8　偏差变化率隶属度函数

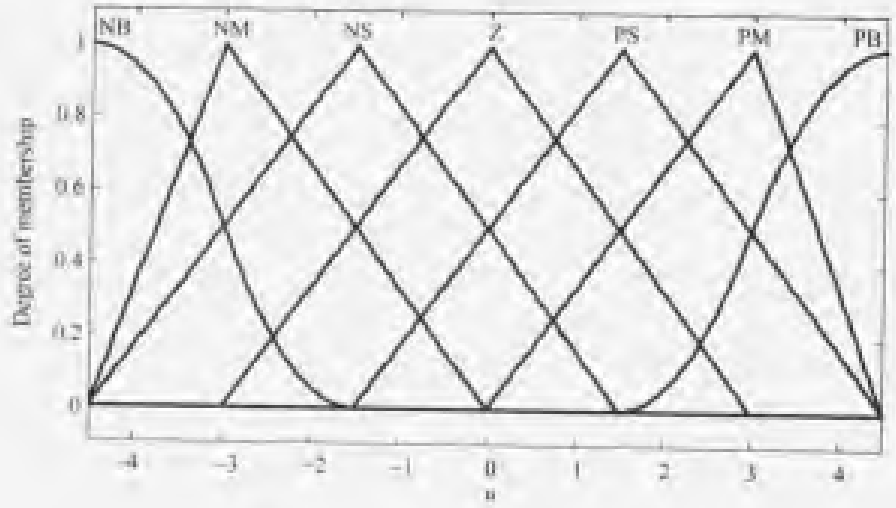


图 4-9　控制器输出隶属度函数

2. 模糊控制位置跟踪

被控对象为

$$G(s)=\frac{400}{s^2+500s}$$

首先运行模糊控制器程序 chap4_2.m(见附录),并将模糊推理系统保存在 a2 之中。然后运行模糊控制的 Simulink 仿真程序,位置指令取正弦信号 $0.5\sin(t)$,仿真结果如图 4-10 所示。模糊控制位置跟踪的 Simulink 仿真程序见附录程序 chap4_3.mdl。

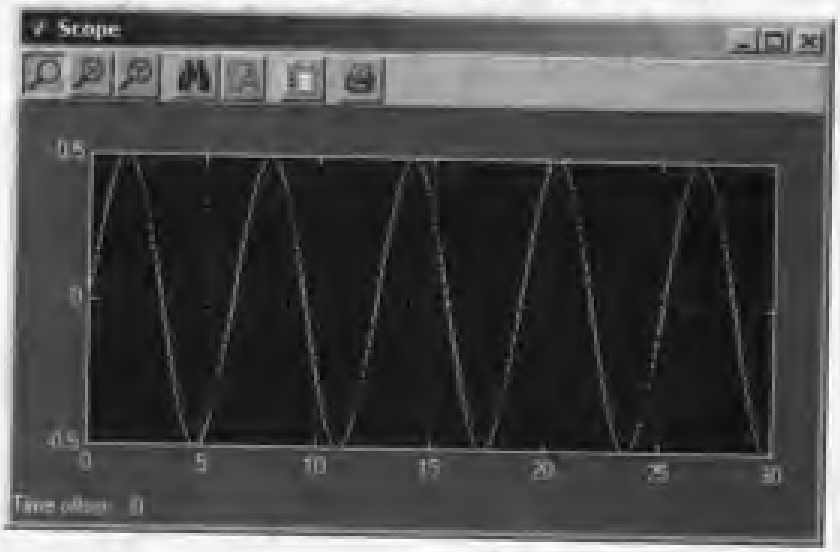


图 4-10　正弦位置跟踪

4.4　模糊控制应用实例——洗衣机的模糊控制

以洗衣机洗涤时间的模糊控制系统设计为例,其控制是一个开环的决策过程,模糊控制按以下步骤进行。

1. 确定模糊控制器的结构

选用两输入单输出模糊控制器。控制器的输入为衣物的污泥和油脂，输出为洗涤时间。

2. 定义输入、输出模糊集

将污泥分为 3 个模糊集：SD(污泥少)，MD(污泥中)，LD(污泥多)；将油脂分为 3 个模糊集：NG(油脂少)，MG(油脂中)，LG(油脂多)；将洗涤时间分为 5 个模糊集：VS(很短)，S(短)，M(中等)，L(长)，VL(很长)。

3. 定义隶属函数

选用如下隶属函数

$$\mu_{\text{污泥}}=\begin{cases}\mu_{\text{SD}}(x)=(50-x)/50 & 0\leqslant x\leqslant 50\\ \mu_{\text{MD}}(x)=\begin{cases}x/50 & 0\leqslant x\leqslant 50\\ (100-x)/50 & 50<x\leqslant 100\end{cases}\\ \mu_{\text{LD}}(x)=(x-50)/50 & 50<x\leqslant 100\end{cases}$$

采用三角形隶属函数可实现污泥的模糊化。采用 Matlab 进行仿真，污泥隶属函数设计仿真程序见附录程序 chap4_4.m，仿真结果如图 4-11 所示。

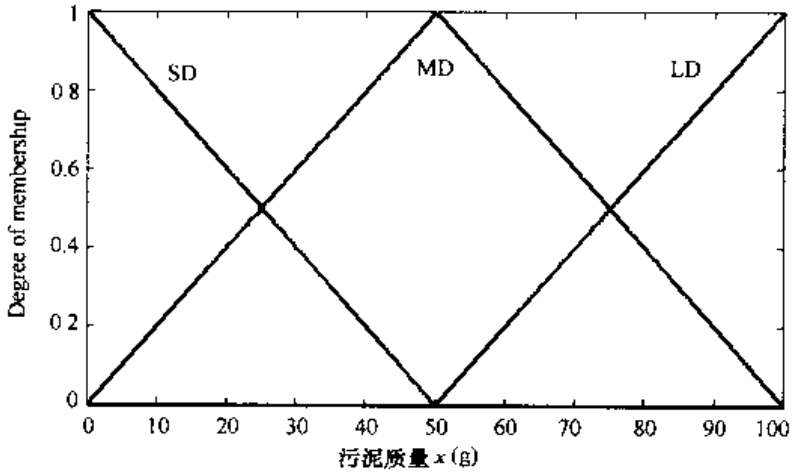


图 4-11　污泥隶属函数

选用如下隶属函数

$$\mu_{\text{油脂}}=\begin{cases}\mu_{\text{NG}}(y)=(50-y)/50 & 0\leqslant y\leqslant 50\\ \mu_{\text{MG}}(y)=\begin{cases}y/50 & 0\leqslant y\leqslant 50\\ (100-y)/50 & 50<y\leqslant 100\end{cases}\\ \mu_{\text{LG}}(y)=(y-50)/50 & 50<y\leqslant 100\end{cases}$$

采用三角形隶属函数实现油脂的模糊化，如图 4-12 所示。仿真程序同 chap4_4.m。

选用如下隶属函数

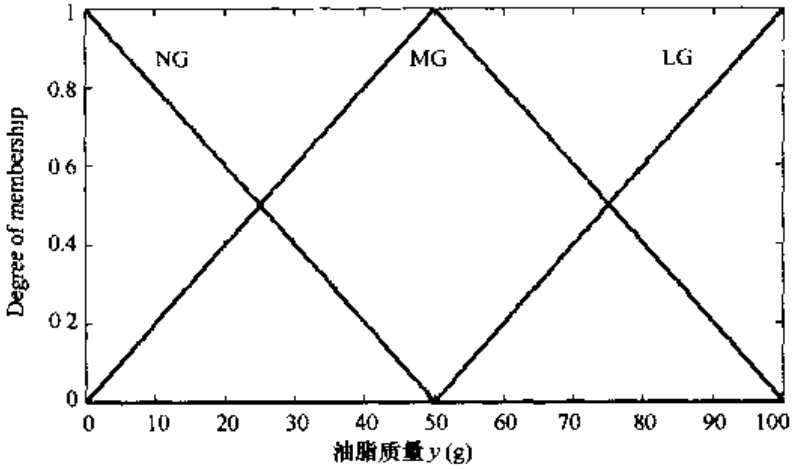


图 4-12　油脂隶属函数

$$\mu_{\text{洗涤时间}}=\begin{cases}\mu_{\text{VS}}(z)=(10-z)/10 & 0\leqslant z\leqslant 10\\ \mu_{\text{S}}(z)=\begin{cases}z/10 & 0\leqslant z\leqslant 10\\ (25-z)/15 & 10<z\leqslant 25\end{cases}\\ \mu_{\text{M}}(z)=\begin{cases}(z-10)/15 & 10\leqslant z\leqslant 25\\ (40-z)/15 & 25<z\leqslant 40\end{cases}\\ \mu_{\text{L}}(z)=\begin{cases}(z-25)/15 & 25\leqslant z\leqslant 40\\ (60-z)/20 & 40<z\leqslant 60\end{cases}\\ \mu_{\text{VL}}(z)=(z-40)/20 & 40\leqslant z\leqslant 60\end{cases}$$

采用三角形隶属函数实现洗涤时间的模糊化，如图 4-13 所示。

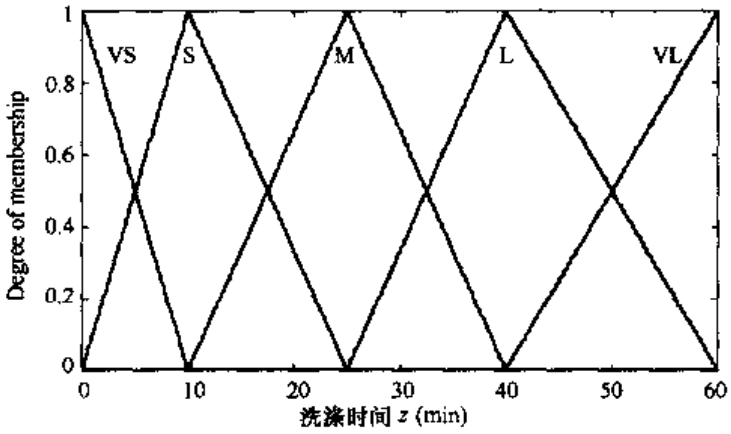


图 4-13　洗涤时间隶属函数

采用 Matlab 仿真，可实现洗涤时间隶属函数的设计，洗涤时间隶属函数的设计仿真程序见附录程序 chap4_5. m。

4. 建立模糊控制规则

根据人的操作经验设计模糊规则，模糊规则设计的标准为："污泥越多，油脂越多，洗涤时间越长"；"污泥适中，油脂适中，洗涤时间适中"；"污泥越少，油脂越少，洗涤时间越短"。

5. 建立模糊控制表

根据模糊规则的设计标准，建立模糊规则表，见表 4-7。

表 4-7　洗衣机的模糊规则

洗涤时间 z		污　泥　x		
		NG	MG	LG
油脂 y	SD	VS*	M	L
	MD	S	M	L
	LD	M	L	VL

第 * 条规则为："IF　衣物污泥少　且　油脂少　THEN　洗涤时间很短"。

6. 模糊推理

分以下几步进行：

(1) 规则匹配

假定当前传感器测得的信息为：x_0(污泥) = 60，y_0(油脂) = 70，分别代入所属的隶属函数中求隶属度为

$$\mu_{MD}(60)=\frac{4}{5},\mu_{LD}(60)=\frac{1}{5}$$

$$\mu_{MG}(70)=\frac{3}{5},\mu_{LG}(70)=\frac{2}{5}$$

通过上述 4 种隶属度，可得到 4 条相匹配的模糊规则，见表 4-8。

表 4-8　模糊推理结果

洗涤时间 z		污　泥　x		
		NG	MG(3/5)	LG(2/5)
油脂 y	SD	0	0	0
	MD(4/5)	0	$\mu_M(z)$	$\mu_L(z)$
	LD(1/5)	0	$\mu_L(z)$	$\mu_{VL}(z)$

(2) 规则触发

由表 4-8 可知，被触发的规则有 4 条，即

Rule 1: IF y is MD and x is MG THEN z is M

Rule 2: IF y is MD and x is LG THEN z is L

Rule 3: IF y is LD and x is MG THEN z is L

Rule 4: IF y is LD and x is LG THEN z is VL

(3) 规则前提推理

在同一条规则内，前提之间通过"与"的关系得到规则结论。前提的可信度之间通过取小运算，得到每一条规则总前提的可信度为

规则 1　前提的可信度为：min(4/5,3/5) = 3/5

规则 2　前提的可信度为：min(4/5,2/5) = 2/5

规则 3　前提的可信度为：min(1/5,3/5) = 1/5

规则 4　前提的可信度为：min(1/5,2/5) = 1/5

由此得到洗衣机规则前提可信度表，即规则强度表，见表 4-9。

表 4-9　规则前提可信度

洗涤时间 z		污泥 x		
		NG	MG(3/5)	LG(2/5)
油脂 y	SD	0	0	0
	MD(4/5)	0	3/5	2/5
	LD(1/5)	0	1/5	1/5

(4) 将上述两个表进行“与”运算

得到每条规则总的可信度输出，见表 4-10。

表 4-10　规则总的可信度

洗涤时间 z		污泥 x		
		NG	MG(3/5)	LG(2/5)
油脂 y	SD	0	0	0
	MD(4/5)	0	$\min\left(\frac{3}{5},\mu_{\mathrm{M}}(z)\right)$	$\min\left(\frac{2}{5},\mu_{\mathrm{L}}(z)\right)$
	LD(4/5)	0	$\min\left(\frac{1}{5},\mu_{\mathrm{L}}(z)\right)$	$\min\left(\frac{1}{5},\mu_{\mathrm{VL}}(z)\right)$

(5) 模糊系统总的输出

模糊系统总的可信度为各条规则可信度推理结果的并集，即

$$\mu_{\text{总}}(z)=\max\left\{\min\left(\frac{3}{5},\mu_{\mathrm{M}}(z)\right),\min\left(\frac{2}{5},\mu_{\mathrm{L}}(z)\right),\min\left(\frac{1}{5},\mu_{\mathrm{L}}(z)\right),\min\left(\frac{1}{5},\mu_{\mathrm{VL}}(z)\right)\right\}$$

$$=\max\left\{\min\left(\frac{3}{5},\mu_{\mathrm{M}}(z)\right),\min\left(\frac{2}{5},\mu_{\mathrm{L}}(z)\right),\min\left(\frac{1}{5},\mu_{\mathrm{VL}}(z)\right)\right\}$$

可见，有 3 条规则被触发。

(6) 反模糊化

模糊系统总的输出 $\mu_{\text{总}}(z)$ 实际上是 3 个规则推理结果的并集，需要进行反模糊化，才能得到精确的推理结果。下面以最大平均法为例，进行反模糊化。

洗衣机的模糊推理过程如图 4-14 和图 4-15 所示。由图可知，洗涤时间隶属度最大值为 $\mu=\frac{3}{5}$。将 $\mu=\frac{3}{5}$ 代入洗涤时间隶属函数中的 $\mu_{\mathrm{M}}(z)$，得到规则前提隶属度 $\mu=\frac{3}{5}$ 与规则结论隶属度 $\mu_{\mathrm{M}}(z)$ 的交点，即

$$\mu_{\mathrm{M}}(z)=\frac{z-10}{15}=\frac{3}{5},\mu_{\mathrm{M}}(z)=\frac{40-z}{15}=\frac{3}{5}$$

得 $z_1=19,z_2=31$。

采用最大平均法，可得精确输出为

$$z^*=\frac{z_1+z_2}{2}=\frac{19+31}{2}=25$$

即所需要的洗涤时间为 25 分钟。

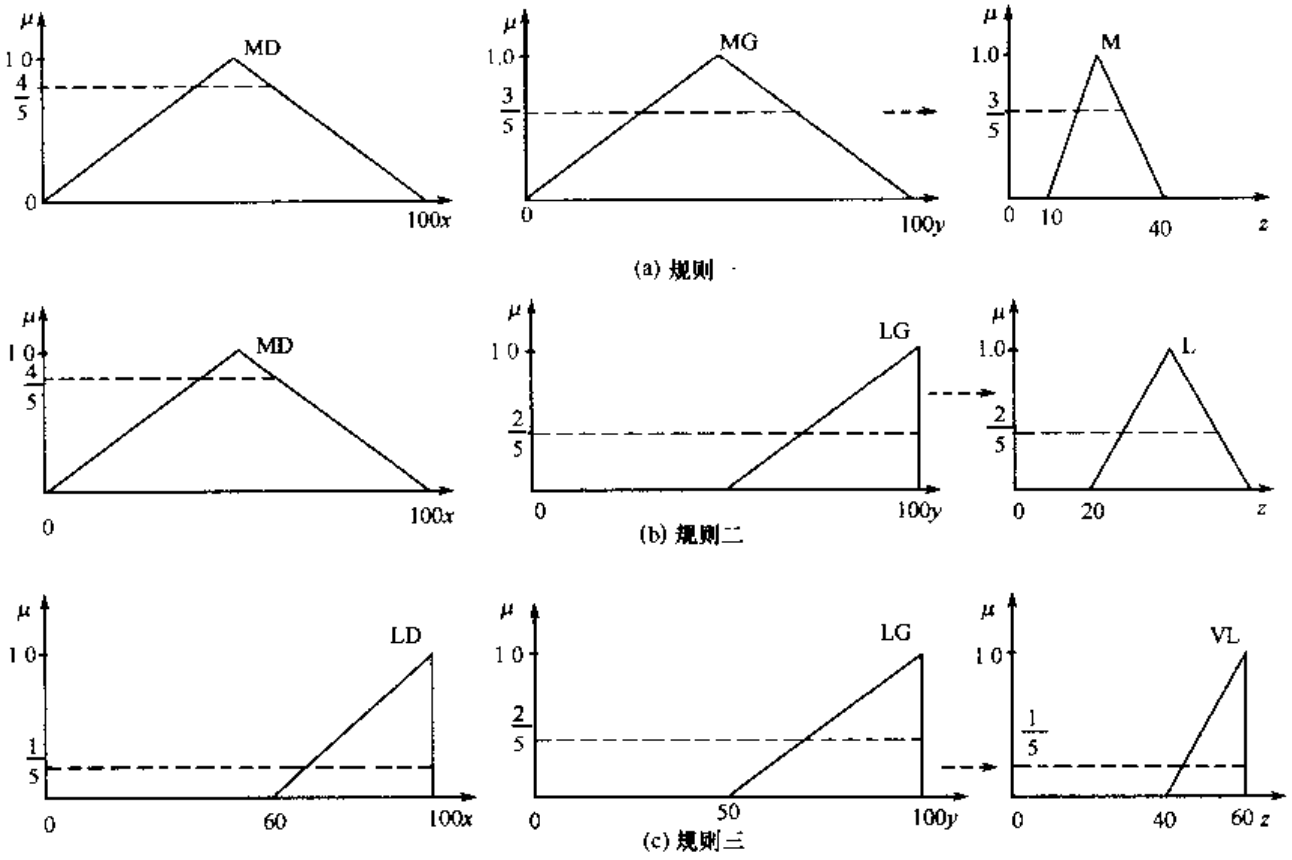


图 4-14　洗衣机的 3 个规则被触发

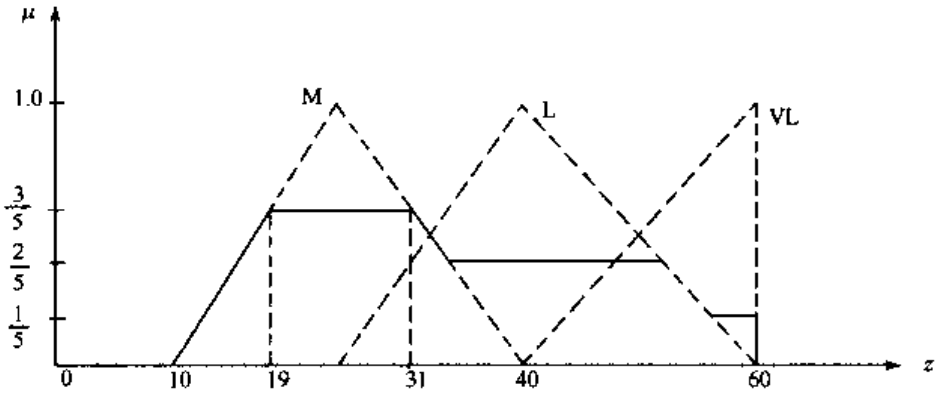


图 4-15　洗衣机的组合输出及反模糊化

7. 仿真实例

采用 Matlab5.3 中模糊控制工具箱可设计洗衣机模糊控制系统。洗衣机模糊控制系统仿真程序见附录程序 chap4_6.m。

取 $x=60, y=70$，反模糊化采用重心法，模糊推理结果为 33.6853。利用命令 showrule 可观察规则库，利用命令 ruleview 可实现模糊控制的动态仿真，动态仿真环境如图 4-16 所示。

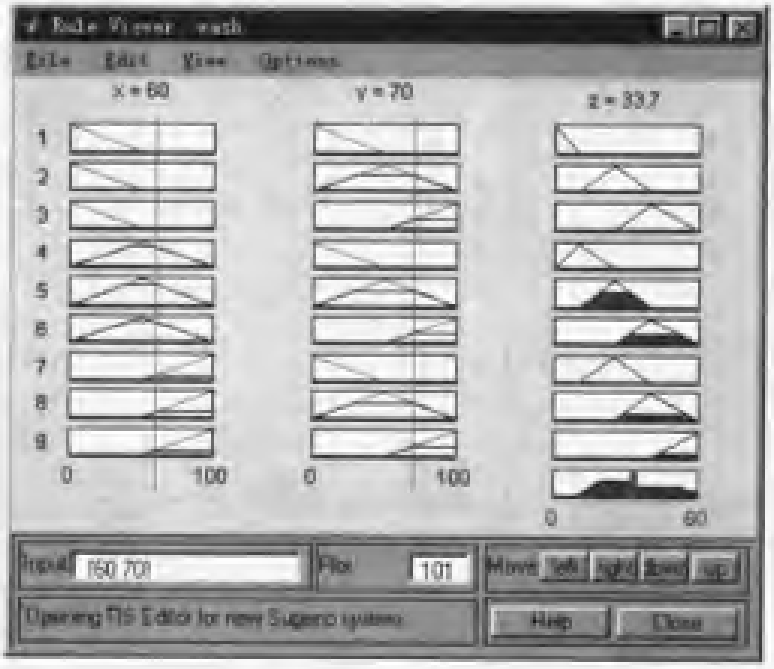


图 4-16 动态仿真模糊系统

4.5 模糊自适应整定 PID 控制

4.5.1 模糊自适应整定 PID 控制原理

在工业生产过程中，许多被控对象受负荷变化或干扰因素影响，其对象特性参数或结构易发生改变。自适应控制运用现代控制理论在线辨识对象特征参数，实时改变其控制策略，使控制系统品质指标保持在最佳范围内，但其控制效果的好坏取决于辨识模型的精确度，这对于复杂系统是非常困难的。因此，在工业生产过程中，大量采用的仍然是 PID 算法。PID 参数的整定方法很多，但大多数都以对象特性为基础。

随着计算机技术的发展，人们利用人工智能的方法将操作人员的调整经验作为知识存入计算机中，根据现场实际情况，计算机能自动调整 PID 参数，这样就出现了专家 PID 控制器。该控制器把古典的 PID 控制与先进的专家系统相结合，实现系统的最佳控制。这种控制方法必须精确地确定对象模型，将操作人员（专家）长期实践积累的经验知识用控制规则模型化，并运用推理对 PID 参数实现最佳调整。

由于操作者经验不易精确描述，控制过程中各种信号量及评价指标不易定量表示，专家 PID 方法受到局限。模糊理论是解决这一问题的有效途径，所以人们运用模糊数学的基本理论和方法，把规则的条件、操作用模糊集表示，并把这些模糊控制规则及有关信息（如评价指标、初始 PID 参数等）作为知识存入计算机知识库中，然后计算机根据控制系统的实际响应情况（即专家系统的输入条件），运用模糊推理，即可自动实现对 PID 参数的最佳调整，这就是模糊自适应 PID 控制。模糊自适应 PID 控制器目前有多种结构形式，但其工作原理基本一致。

自适应模糊 PID 控制器以误差 e 和误差变化 ec 作为输入（利用模糊控制规则在线对 PID

参数进行修改），以满足不同时刻的 e 和 ec 对 PID 参数自整定的要求。自适应模糊 PID 控制器结构如图 4-17 所示。

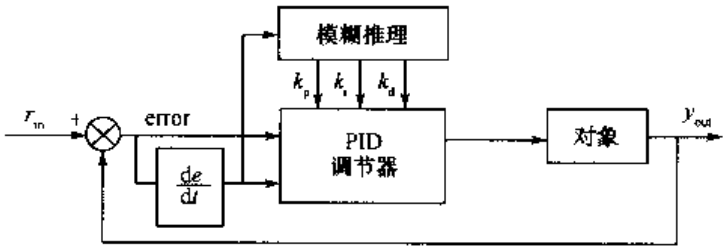


图 4-17　自适应模糊控制器结构

PID参数模糊自整定是找出PID的 3 个参数 k_p，k_i，k_d 与 e 和 ec 之间的模糊关系，在运行中通过不断检测 e 和 ec，根据模糊控制原理来对 3 个参数进行在线修改，以满足不同 e 和 ec 时对控制参数的要求，从而使被控对象有良好的动、静态性能。

从系统的稳定性、响应速度、超调量和稳态精度等各方面来考虑，k_p，k_i，k_d 的作用如下：

（1）比例系数 k_p 的作用是加快系统的响应速度。k_p 越大，系统的响应速度越快，但易产生超调，甚至会导致系统不稳定。k_p 取值过小，则会使响应速度缓慢，从而延长调节时间，使系统静态、动态特性变坏。

（2）积分作用系数 k_i 的作用是消除系统的稳态误差。k_i 越大，系统的静态误差消除越快，但 k_i 过大，在响应过程的初期会产生积分饱和现象，从而引起响应过程的较大超调。若 k_i 过小，将使系统静态误差难以消除，从而影响系统的调节精度。

（3）微分作用系数 k_d 的作用是改善系统的动态特性，其作用主要是在响应过程中抑制偏差向任何方向的变化，对偏差变化进行提前预报。但 k_d 过大，会使响应过程提前制动，从而延长调节时间，而且会降低系统的抗干扰性能。

PID 参数的整定必须考虑在不同时刻 3 个参数的作用及相互之间的互联关系。

模糊控制设计的核心是总结工程设计人员的技术知识和实际操作经验，建立合适的模糊规则表，得到针对 k_p，k_i，k_d 3 个参数分别整定的模糊控制表。

（1）k_p 的模糊规则表（见表 4-11）

表 4-11　k_p 的模糊规则表

k_p / ec / e	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PM	PM	PS	ZO	ZO
NM	PB	PB	PM	PS	PS	ZO	NS
NS	PM	PM	PM	PS	ZO	NS	NS
ZO	PM	PM	PS	ZO	NS	NM	NM
PS	PS	PS	ZO	NS	NS	NM	NM
PM	PS	ZO	NS	NM	NM	NM	NB
PB	ZO	ZO	NM	NM	NM	NB	NB

(2)k_i 的模糊规则表(见表 4-12)

表 4-12 k_i 的模糊规则表

k_i / e \ ec	NB	NM	NS	ZO	PS	PM	PB
NB	NB	NB	NM	NM	NS	ZO	ZO
NM	NB	NB	NM	NS	NS	ZO	ZO
NS	NB	NM	NS	NS	ZO	PS	PS
ZO	NM	NM	NS	ZO	PS	PM	PM
PS	NM	NS	ZO	PS	PS	PM	PB
PM	ZO	ZO	PS	PS	PM	PB	PB
PB	ZO	ZO	PS	PM	PM	PB	PB

(3)k_d 的模糊控制规则表(见表 4-13)

表 4-13 k_d 的模糊控制规则表

k_d / e \ ec	NB	NM	NS	ZO	PS	PM	PB
NB	PS	NS	NB	NB	NB	NM	PS
NM	PS	NS	NB	NM	NM	NS	ZO
NS	ZO	NS	NM	NM	NS	NS	ZO
ZO	ZO	NS	NS	NS	NS	NS	ZO
PS	ZO	ZO	ZO	ZO	ZO	ZO	ZO
PM	PB	NS	PS	PS	PS	PS	PB
PB	PB	PM	PM	PM	PS	PS	PB

k_p,k_i,k_d 的模糊控制规则表建立好后,可根据如下方法进行 k_p,k_i,k_d 的自适应校正。

将系统误差 e 和误差变化 ec 变化范围定义为模糊集上的论域,即

$$e,ec = \{-5,-4,-3,-2,-1,0,1,2,3,4,5\} \tag{4.9}$$

其模糊子集为 $e,ec = \{NB,NM,NS,ZO,PS,PM,PB\}$,子集中元素分别代表负大,负中,负小,零,正小,正中,正大。设 e,ec 和 k_p,k_i,k_d 均服从正态分布,因此可得出各模糊子集的隶属度,根据各模糊子集的隶属度赋值表和各参数模糊控制模型,应用模糊合成推理设计 PID 参数的模糊矩阵表,查出修正参数代入下式计算

$$\begin{aligned} k_p &= k_p' + \{e_i,ec_i\}_p \\ k_i &= k_i' + \{e_i,ec_i\}_i \\ k_d &= k_d' + \{e_i,ec_i\}_d \end{aligned} \tag{4.10}$$

在线运行过程中,控制系统通过对模糊逻辑规则的结果处理、查表和运算,完成对 PID 参数的在线自校正。其工作流程图如图 4-18 所示。

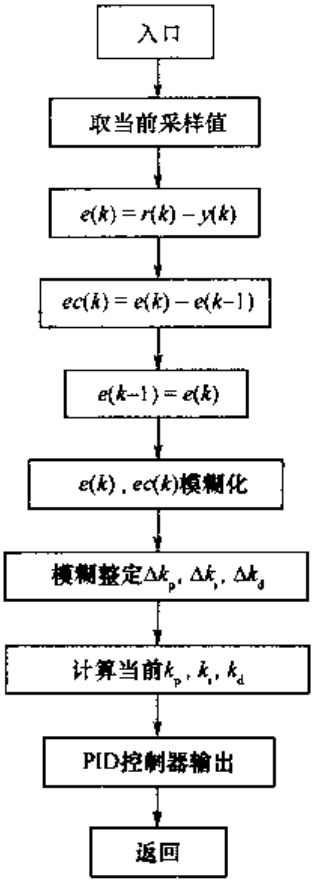


图 4-18　模糊 PID 工作流程图

4.5.2　仿真实例

被控对象为

$$G_p(s) = \frac{523500}{s^3 + 87.35s^2 + 10470s}$$

采样时间为 1ms，采用 z 变换进行离散化，离散化后的被控对象为

$$y_{out}(k) = -\text{den}(2)y_{out}(k-1) - \text{den}(3)y_{out}(k-2) - \text{den}(4)y_{out}(k-3) + \text{num}(2)u(k-1) + \text{num}(3)u(k-2) + \text{num}(4)u(k-3)$$

位置指令为幅值为 1.0 的方波信号，$r(k) = \text{sgn}(\sin(2\pi t))$。仿真时，先运行模糊推理系统设计程序 chap4_7a.m(见附录程序)，实现模糊推理系统 fuzzpid.fis 的设计，并将此模糊推理系统调入内存中，然后运行模糊控制程序 chap4_7b.m(见附录程序)。

在程序 chap4_7a.m 中，根据模糊规则表 4-11 至表 4-13，分别对 e，ec，k_p，k_i，k_d 进行隶属函数的设计。在 Matlab 环境下，对模糊系统 a 运行 plotmf 命令，可得到模糊系统 e，de，k_p，k_i，k_d 的隶属函数，如图 4-19 至图 4-23 所示，运行命令 showrule 可显示模糊规则。另外，针对模糊推理系统 fuzzpid.fis，运行命令 fuzzy 可进行规则库和隶属函数的编辑，如图 4-24 所示，运行命令 ruleview 可实现模糊系统的动态仿真，如图 4-25 所示。

在程序 chap4_7b.m 中，利用所设计的模糊系统 fuzzpid.fis 进行 PID 控制参数的整定，并采用模糊 PID 控制进行阶跃响应，在第 300 个采样时间时控制器输出端加上 1.0 的干扰，响应结果及 PID 控制参数的自适应变化如图 4-26 至图 4-31 所示。

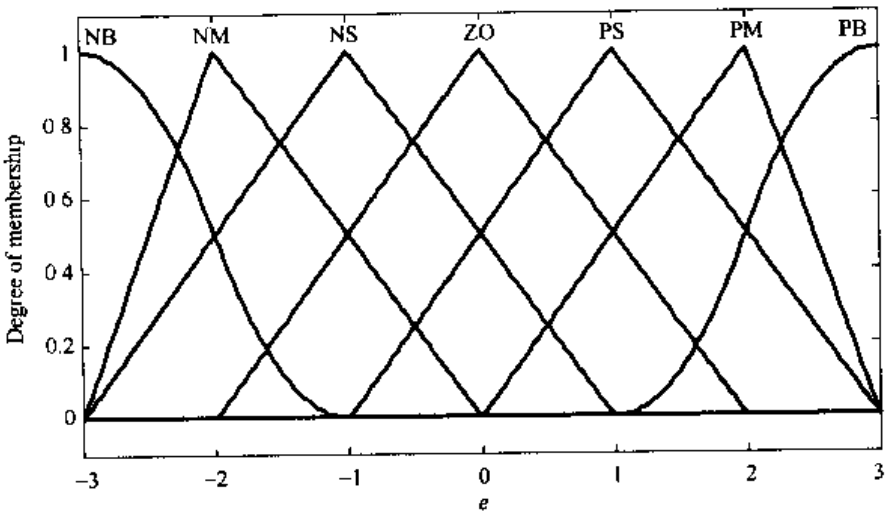


图 4-19　误差的隶属函数

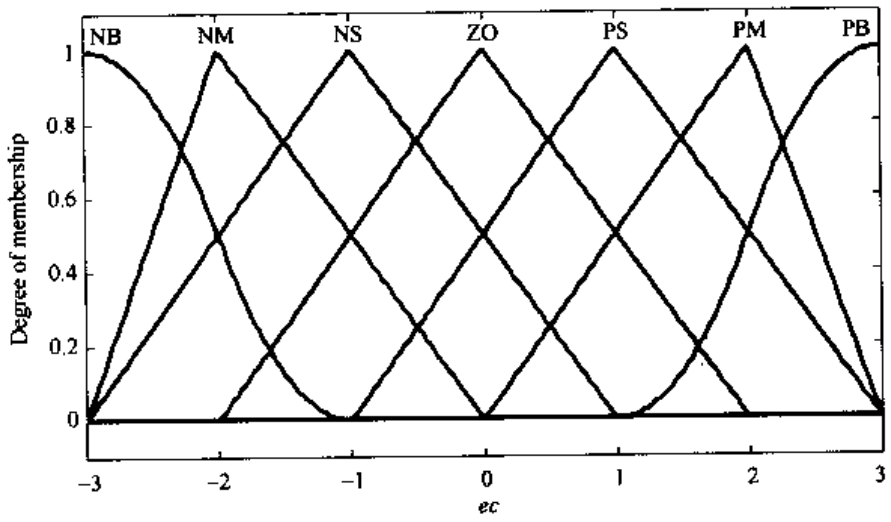


图 4-20　误差变化率的隶属函数

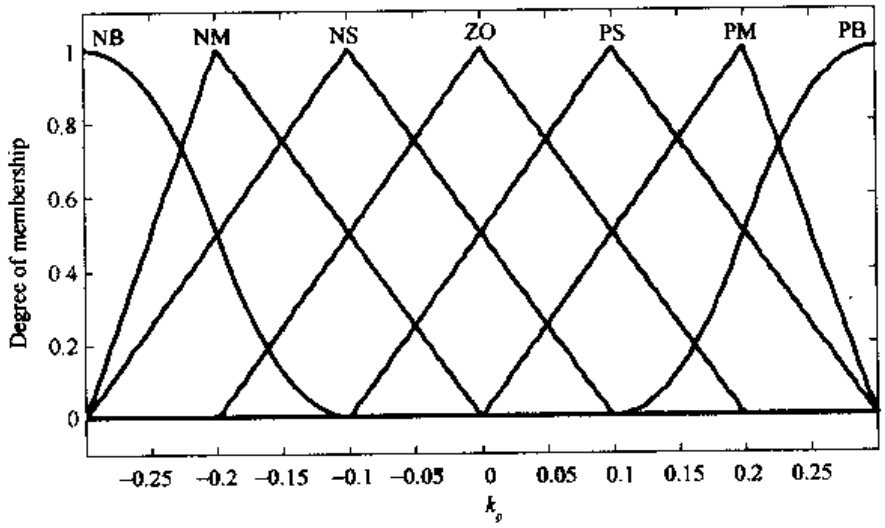


图 4-21　k_p 的隶属函数

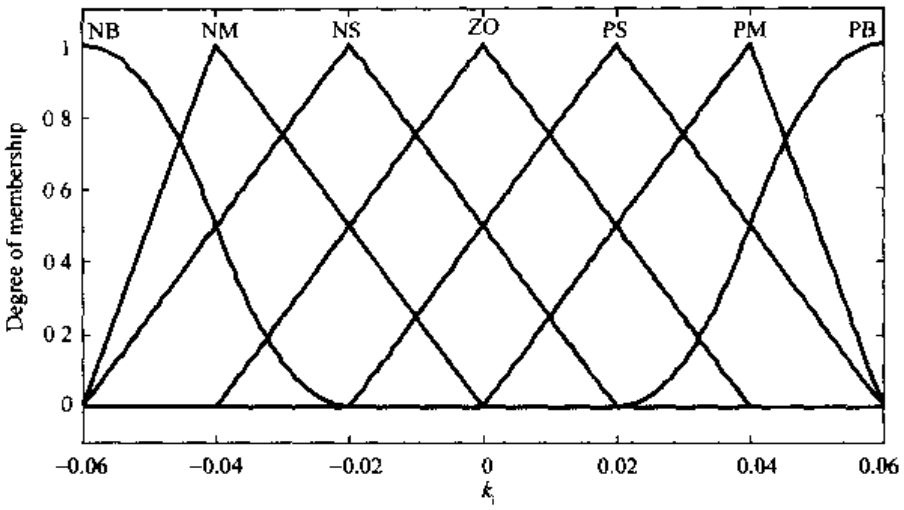


图 4-22　k_i 的隶属函数

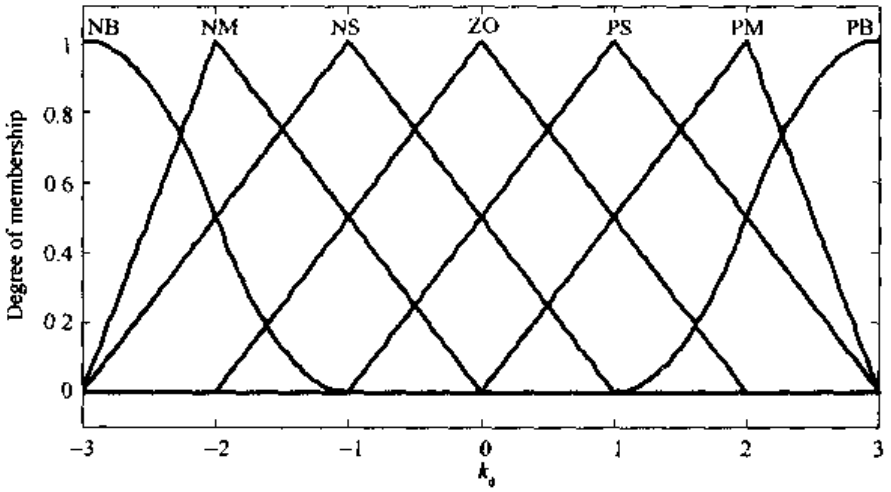


图 4-23　k_d 的隶属函数

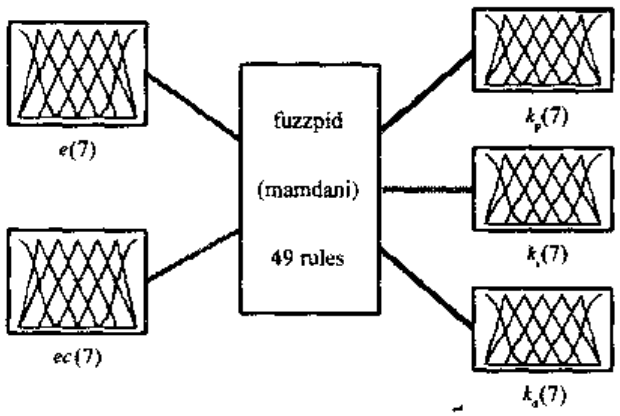


图 4-24　模糊系统 fuzzpid. fis 的结构

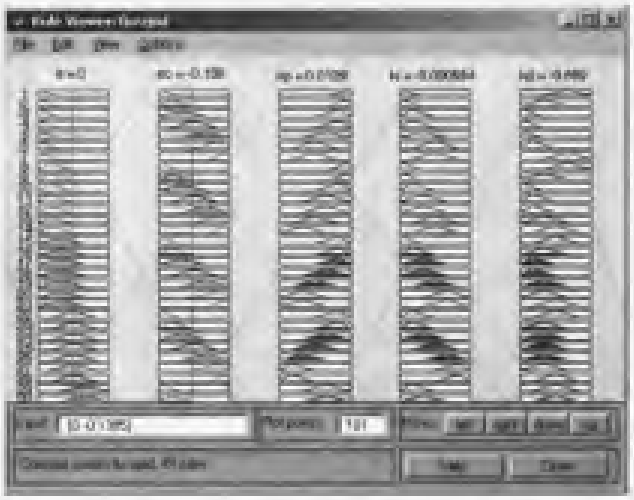


图 4-25　模糊推理系统的动态仿真环境

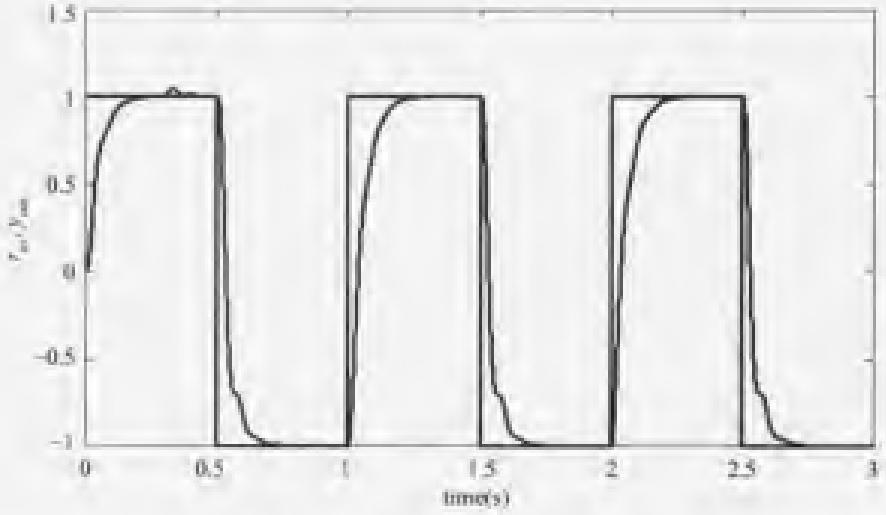


图 4-26　模糊 PID 控制阶跃响应

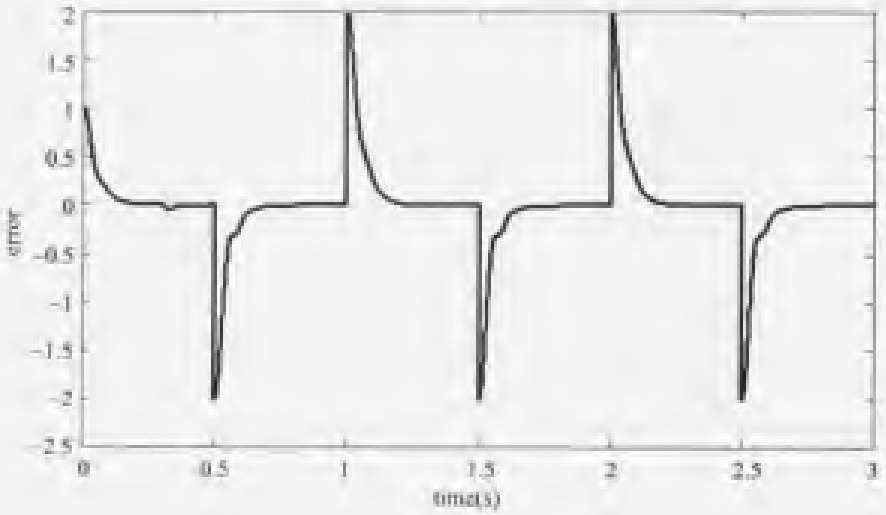


图 4-27　模糊 PID 控制误差响应

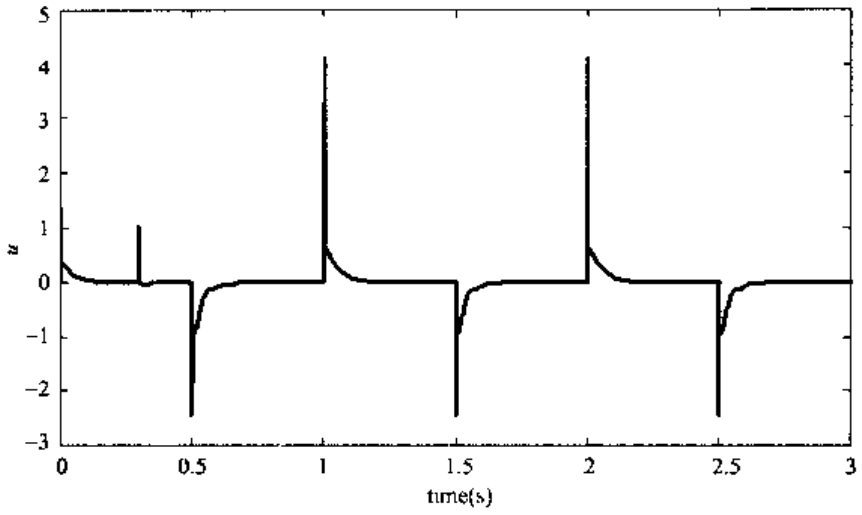


图 4-28　控制器输入 u

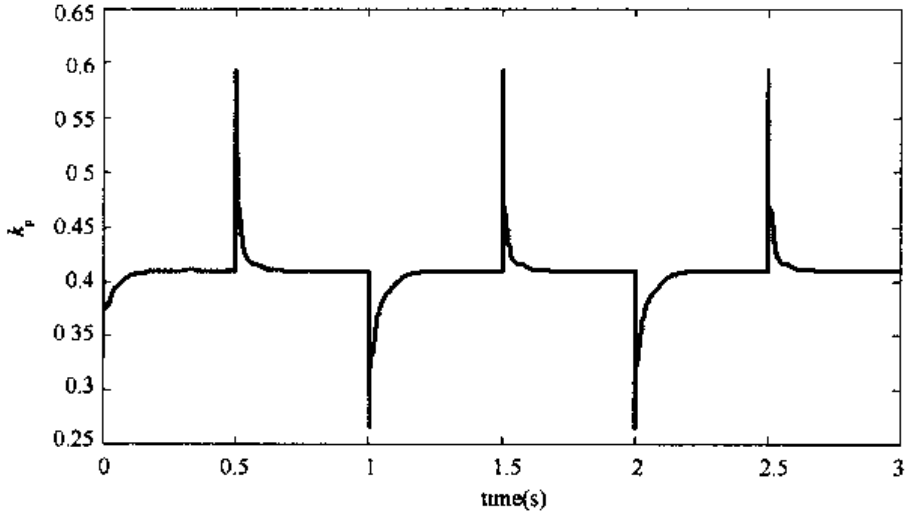


图 4-29　k_p 的自适应调整

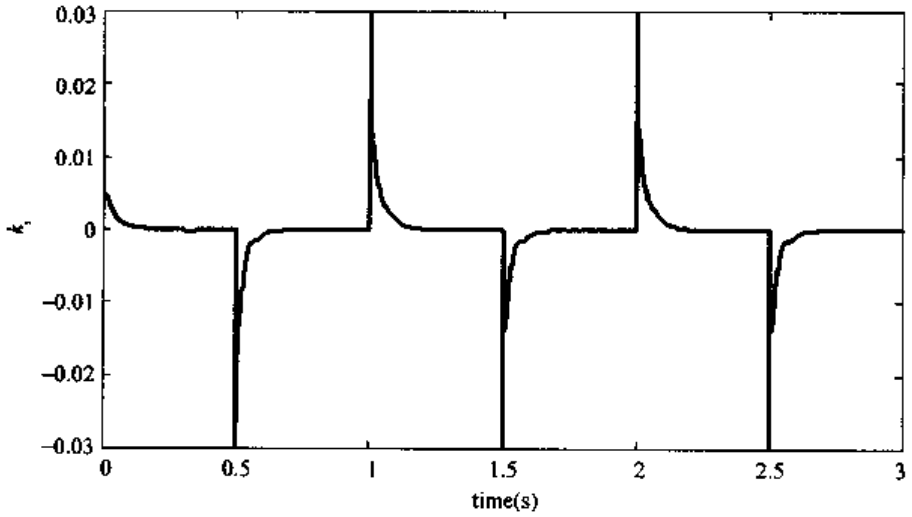


图 4-30　k_i 的自适应调整

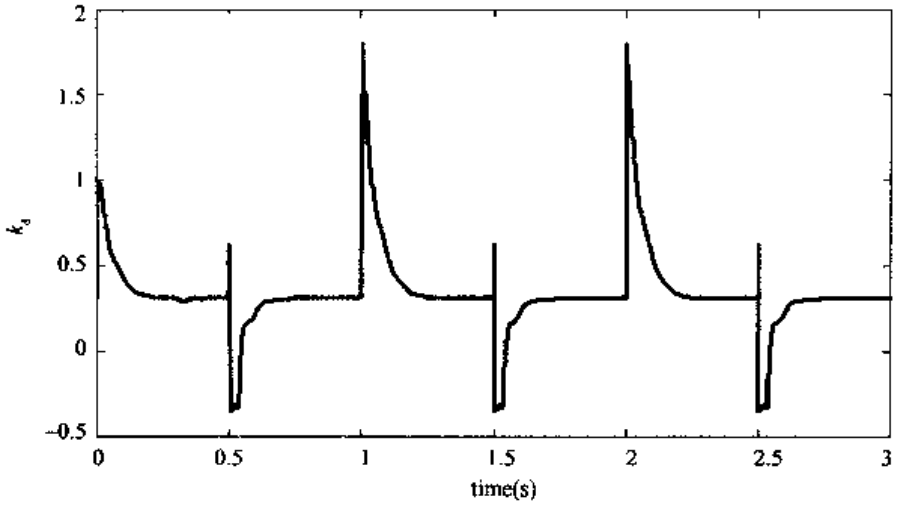


图 4-31　k_d 的自适应调整

4.6　Sugeno 模糊模型

1. Sugeno 模糊模型的形式

前面介绍的是传统的模糊系统，属于 Mamdani 模糊模型，其输出为模糊量。另一种模糊模型为 Sugeno 模糊模型，其输出为常量或线性函数，其函数形式为

$$y = a$$

$$y = ax + b \tag{4.11}$$

Sugeno 模糊模型与 Mamdani 模糊模型的区别在于：①Sugeno 模糊模型输出变量为常量或线性函数；②Sugeno 模糊模型输出为精确量。

Sugeno 型的模糊推理系统非常适合于分段线性控制系统，如在导弹、飞行器的控制中，可根据高度和速度建立 Sugeno 型的模糊推理系统，实现性能良好的线性控制。

2. 仿真实例

设输入 $X \in [0,5]$，$Y \in [0,10]$，将它们模糊化为两个模糊量，即“小”和“大”。输出 Z 为输入 (X,Y) 的线性函数，模糊规则为

If　X 为 small　and　Y 为 small　then　Z = −X + Y − 3

If　X 为 small　and　Y 为 big　then　Z = X + Y − 1

If　X 为 big　and　Y 为 small　then　Z = −2Y + 2

If　X 为 big　and　Y 为 big　then　Z = 2X + Y − 6

仿真程序见附录程序 chap4_8.m。模糊推理系统的输入隶属函数曲线及输入、输出曲线如图 4-32 和图 4-33 所示。

通过命令 showrule(ts2) 可显示模糊控制规则，共有以下 4 条：

(1)If　(X is small)　and　(Y is small)　then　(Z is first area) (1)

(2)If　(X is small)　and　(Y is big)　then　(Z is second area) (1)

(3)If　(X is big)　and　(Y is small)　then　(Z is third area) (1)

(4)If　(X is big)　and　(Y is big)　then　(Z is fourth area) (1)

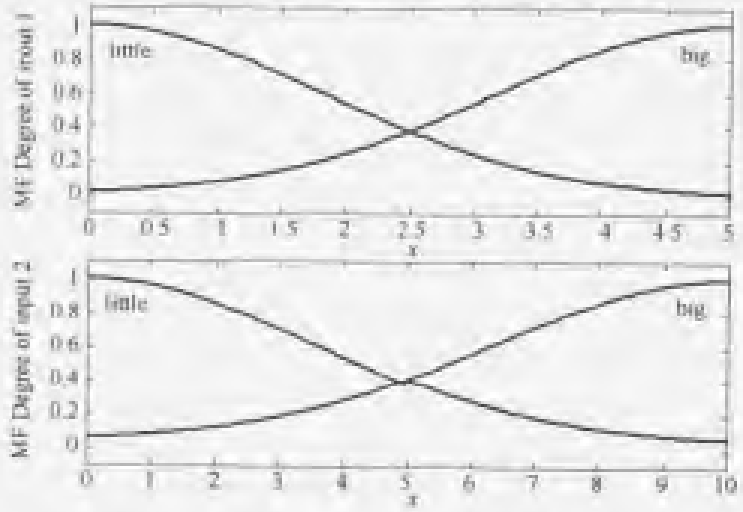


图 4-32　Sugeno 模糊推理系统的输入隶属函数曲线

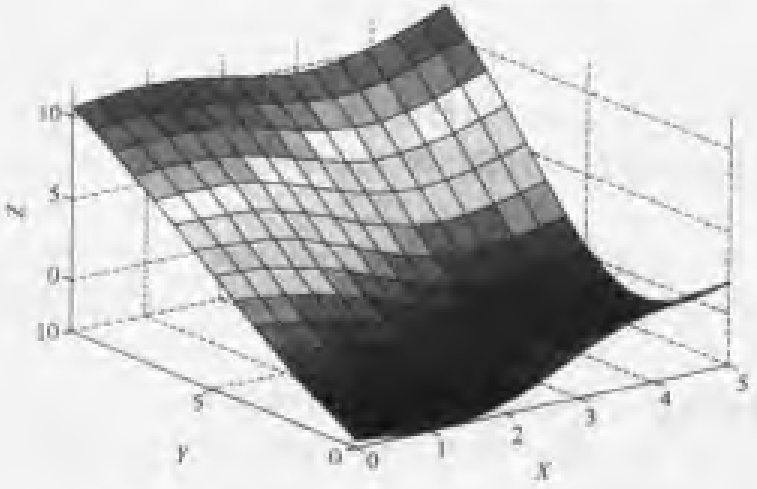


图 4-33　Sugeno 模糊推理系统的输入、输出曲线

4.7　基于 Sugeno 模糊模型的倒立摆模糊控制

4.7.1　倒立摆模型的局部线性化

当倒立摆的摆角和摆速很小时，其模型可进行线性化，从而可实现基于 Sugeno 模糊模型的倒立摆模糊控制。

倒立摆的动力学方程为

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{g\sin(x_1) - aml x_2^2 \sin(2x_1)/2 - a\cos(x_1)u}{4/3l - aml\cos^2 x_1}\end{aligned} \tag{4.12}$$

式中，x_1 表示摆与垂直线的夹角，$x_1 = \theta$，x_2 表示摆的摆动角速度，$x_2 = \omega = \dot{\theta}$，$g = 9.8\text{m/s}^2$ 为重力加速度，m 为倒立摆的质量，$2l$ 为摆长，$a = 1/(m + M)$，M 为小车质量。

由式(4.12)可知，当摆角 θ 和摆速 $\dot{\theta}$ 很小时，$\sin(x_1) \to x_1$，$\cos(x_1) \to 1$，$aml x_2^2 \to 0$。在 $(x_1$，

x_2) 平面上对倒立摆模型进行局部线性化，倒立摆的动力学方程可近似写为

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{g}{4/3l - aml}x_1 - \frac{a}{4/3l - aml}u \tag{4.13}$$

4.7.2 仿真实例

取倒立摆参数 $m = 2\text{kg}, M = 8\text{kg}, l = 0.5\text{m}$。令 $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$，则式(4.13) 可表示为如下状态方程

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.14}$$

式中，$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 15.8919 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 \\ -0.0811 \end{bmatrix}$。

则可得到 Sugeno 型模糊模型规则为

$$\text{if} \quad x_1 \text{ 为 ZR} \quad \text{and} \quad x_2 \text{ 为 ZR} \quad \text{then} \quad \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.15}$$

选择期望的闭环极点 $P(-10-10\text{i}, -10+10\text{i})$，采用 $\boldsymbol{u} = -\boldsymbol{F}\boldsymbol{x}$ 的反馈控制，利用极点配置函数 place(A,B,P)，可以得到系统的反馈增益矩阵 $\boldsymbol{F}$，且 $\boldsymbol{F} = [-2662.7 \quad -246.7]$。

根据倒立摆的模糊建模过程，可以设计 Sugeno 型模糊控制器，其 Sugeno 型模糊控制规则为

$$\text{if} \quad x_1 \text{ 为 ZR} \quad \text{and} \quad x_2 \text{ 为 ZR} \quad \text{then} \quad u = -\boldsymbol{F}\boldsymbol{x} \tag{4.16}$$

利用模糊规则式(4.15) 和式(4.16)，可设计基于 Sugeno 模糊模型的倒立摆模糊控制系统。

设倒立摆的摆角范围为[−15,15]度，摆角角速度范围为[−200,200]度/秒，摆角角加速度范围为[−200,200]度/秒^2。采用三角形隶属函数对摆角和摆角角速度进行模糊化。摆角初始状态为[0.2,0]，运行仿真程序 chap4_9f.m 和 chap4_9.m(见附录)，倒立摆的摆角、角速度、控制输出信号及模糊输入隶属函数曲线的仿真结果如图 4.34 至图 4.37 所示。

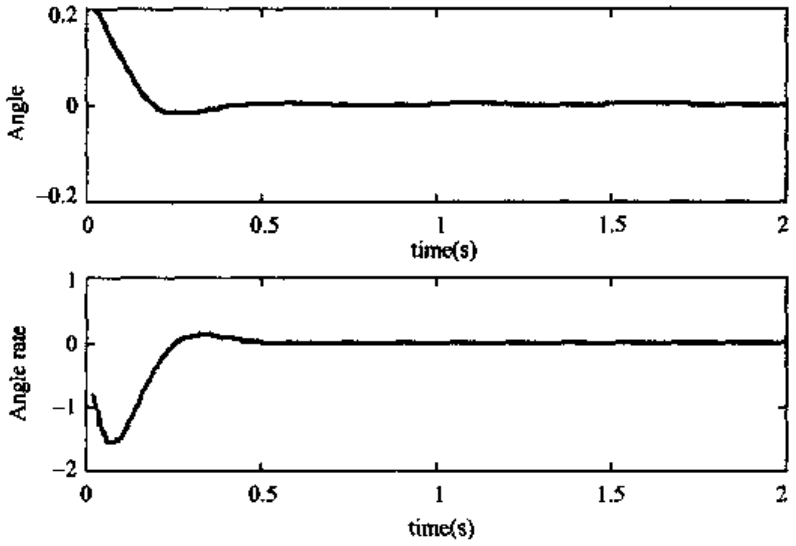


图 4.34 摆角的状态响应

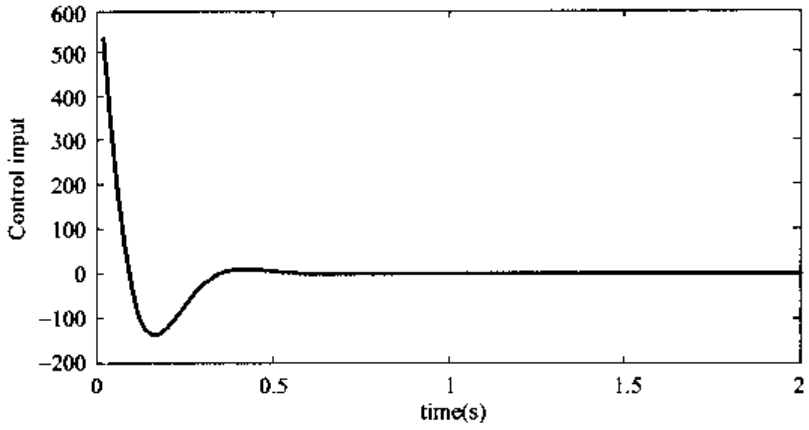


图 4.35　控制器的输出

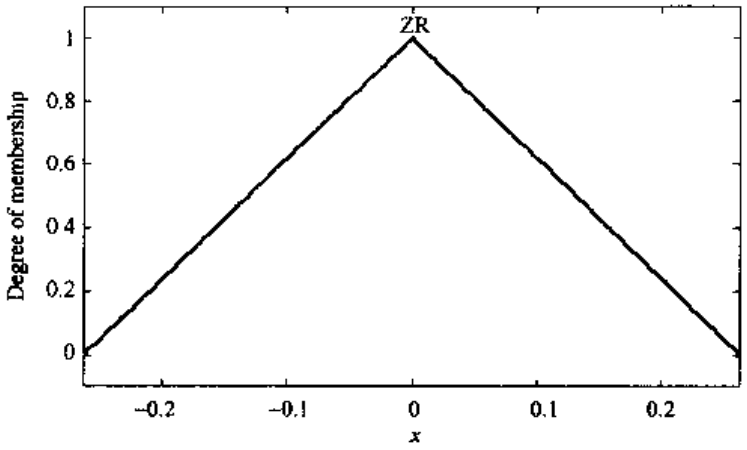


图 4.36　角度 θ 的隶属度曲线

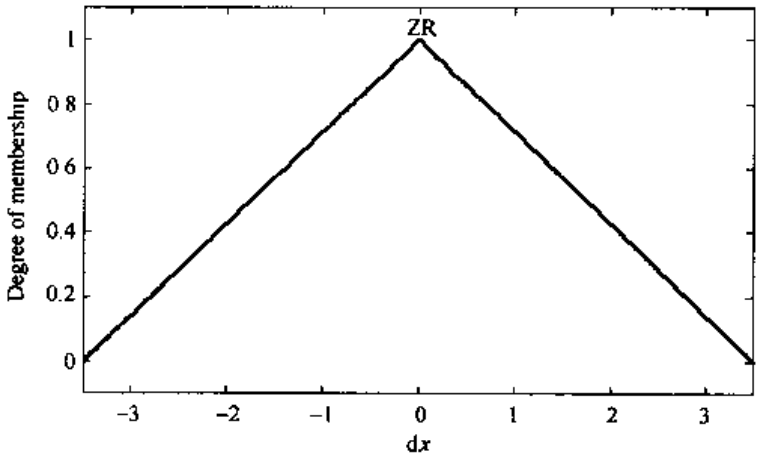


图 4.37　角速度 $\dot{\theta}$ 的隶属度曲线

4.8 模糊控制的应用

1. 模糊控制在家电中的应用

模糊电子技术是21世纪的核心技术，模糊家电是模糊电子技术的最重要应用领域。所谓模糊家电，就是根据人的经验，在电脑或芯片的控制下实现可模仿人的思维进行操作的家用电器。几种典型的模糊家电产品如下：

(1) 模糊电视机

模糊电视机可根据室内光线的强弱自动调整电视机的亮度，根据人与电视机的距离自动调整音量，同时能够自动调节电视机的色度、清晰度和对比度。

1990年3月，日本的三洋公司研制并推出了一种采用模糊技术的彩色电视机，该电视机能够根据室内的亮度和观看距离，对电视机的对比度进行自动调节，保证在各种条件下都能获得最佳的收看效果。

(2) 模糊空调器

模糊空调器可灵敏地控制室内的温度。日本研制了一种模糊空调器，利用红外线传感器识别房间信息(人数、温度、大小、门开关等)，快速调整室内温度，提高了舒适感。

(3) 模糊微波炉

日本夏普公司生产的RE—SEI型微波炉，内部装有12个传感器，这些传感器能对食品的重量、高度、形状和温度等进行测量，并利用这些信息自动选择化霜、再热、烧烤和对流4种工作方式，并自动决定烹制时间。

(4) 模糊洗衣机

以我国生产的小天鹅模糊控制全自动洗衣机为例，它能够自动识别洗衣物的重量、质地、污脏性质和程度，采用模糊控制技术来选择合理的水位、洗涤时间、水流程序等，其性能已达到国外同类产品的水平。

(5) 模糊电动剃刀

日本三洋、松下公司推出了模糊控制电动剃刀，通过利用传感器分析胡须的生长情况和面部轮廓，自动调整刀片，并选择最佳的剃削速度。

2. 模糊控制在过程控制中的应用

(1) 工业炉方面：如退火炉、电弧炉、水泥窑、热风炉、煤粉炉的模糊控制。

(2) 石化方面：如蒸馏塔的模糊控制、废水pH值计算机模糊控制系统、污水处理系统的模糊控制等。

(3) 煤矿行业：如选矿破碎过程的模糊控制、煤矿供水的模糊控制等。

(4) 食品加工行业：如甜菜生产过程的模糊控制、酒精发酵温度的模糊控制等。

3. 模糊控制在机电行业中的应用

如集装箱吊车的模糊控制、空间机器人柔性臂动力学的模糊控制、单片机温度模糊控制、交流随动系统的模糊控制、快速伺服系统定位的模糊控制、电梯群控系统多目标模糊控制、直流无刷电机调速的模糊控制等。

4.9　模糊控制发展概况

4.9.1　模糊控制发展的几个转折点

自从 Zadeh 提出模糊集理论以来，模糊控制开始了它的发展历程。从历史的发展来看，模糊控制发展的几个转折点见表 4-14。

表 4-14　模糊控制发展的转折点

时间	研究人员	研究成果
1965	Zadeh	模糊集理论
1972	Zadeh	模糊控制原理
1973	Zadeh	复杂系统及决策过程的分析
1974	Mamdani et al	蒸汽机的模糊控制
1976	Rutherford et al	模糊算法分析
1977	Ostergaad	热交换器和水泥窑模糊控制
1977	Willaey et al	最优模糊控制
1979	Komolov et al	有限自动机理论
1980	Tong et al	污水处理过程的模糊控制
1980	Fukami et al	模糊条件推理
1983	Hirota et al	概率模糊集理论
1983	Takagi et al	模糊控制规则的获取
1983	Yasunobu et al	预测模糊控制
1984	Sugeno et al	汽车的停车模糊控制
1985	Kiszka et al	模糊系统的稳定性
1985	Togai et al	模糊芯片
1986	Yamakawa	模糊控制的硬件系统
1988	Dubois et al	逼近推理
1988	Czogala	多输入模糊控制系统
1991	De Neyer et al	内模模型的模糊控制
1992	Yager	模糊控制隶属函数的神经网络学习
1992	L. X. Wang	模糊万能逼近器
1992	L. X. Wang	模糊规则的获取
1993	L. X. Wang	自适应模糊控制器

4.9.2　模糊控制的发展方向

1. Fuzzy-PID 复合控制

Fuzzy-PID 复合控制是将模糊控制与常规 PID 控制算法相结合的控制方法，以此达到较高的控制精度。它比单用模糊控制和单用 PID 控制具有更好的控制性能。

2. 自适应模糊控制

自适应模糊控制能自动地对模糊控制规则进行修改和完善，以提高控制系统的性能。它具有自适应、自学习的能力，对于那些具有非线性、大时滞、高阶次的复杂系统有着更好的控制效果。

3. 专家模糊控制

专家模糊控制是将专家系统技术与模糊控制相结合的产物。引入专家系统，可进一步提高模糊控制的智能水平。专家模糊控制保持了基于规则的方法和模糊集处理带来的灵活性，同时

又把专家系统技术的知识表达方法结合进来，能处理更广泛的控制问题。

4. 神经模糊控制

模糊控制规则和隶属函数的获取与确定是模糊控制中的“瓶颈”问题。神经模糊控制是基于神经网络的模糊控制方法。该方法利用神经网络的学习能力，来获取并修正模糊控制规则和隶属函数。

5. 多变量模糊控制

多变量模糊控制有多个输入变量和输出变量，它适用于多变量控制系统。多变量耦合和“维数灾”问题是多变量模糊控制需要解决的关键问题。

4.9.3 模糊控制面临的主要任务

(1) 模糊控制的机理及稳定性分析，新型自适应模糊控制系统、专家模糊控制系统、神经网络模糊控制系统和多变量模糊控制系统的分析与设计。

(2) 模糊集成控制系统的设计方法研究。现代控制理论、神经网络与模糊控制的相互结合及相互渗透，可构成模糊集成控制系统。

(3) 非线性系统应用中的模糊建模、模糊规则的建立和模糊推理算法的深入研究。

(4) 自学习模糊控制策略的研究。

(5) 常规模糊控制系统稳定性的改善。

(6) 模糊控制芯片、模糊控制装置及通用模糊控制系统的开发及工程应用。

思考题与习题

4-1 模糊控制器由哪几部分组成?各完成什么功能?

4-2 模糊控制器设计的步骤怎样?

4-3 已知某一炉温控制系统，要求温度保持在 600℃ 恒定。针对该控制系统有以下控制经验：

(1) 若炉温低于 600℃，则升压；低得越多升压越高。

(2) 若炉温高于 600℃，则降压；高得越多降压越低。

(3) 若炉温等于 600℃，则保持电压不变。

设模糊控制器为一维控制器，输入语言变量为误差，输出为控制电压。输入、输出变量的量化等级为 7 级，取 5 个模糊集。试设计隶属度函数误差变化划分表、控制电压变化划分表和模糊控制规则表。

4-4 已知被控对象为 $G(s)=\dfrac{1}{10s+1}e^{-0.5s}$。假设系统给定为阶跃值 $r=30$，采样时间为 0.5s，系统的初始值 $r(0)=0$。试分别设计：

(1) 常规的 PID 控制器；

(2) 常规的模糊控制器；

(3) 模糊 PID 控制器。

分别对上述 3 种控制器进行 Matlab 仿真，并比较控制效果。

附录（程序代码）

水箱液位模糊控制仿真程序：chap4_1.m

```
%Fuzzy Control for water tank
clear all;
close all;

a = newfis('fuzz tank');

a = addvar(a,'input','e',[-3,3]);          %Parameter e
a = addmf(a,'input',1,'NB','zmf',[-3,-1]);
a = addmf(a,'input',1,'NS','trimf',[-3,-1,1]);
a = addmf(a,'input',1,'Z','trimf',[-2,0,2]);
a = addmf(a,'input',1,'PS','trimf',[-1,1,3]);
a = addmf(a,'input',1,'PB','smf',[1,3]);

a = addvar(a,'output','u',[-4,4]);         %Parameter u
a = addmf(a,'output',1,'NB','zmf',[-4,-1]);
a = addmf(a,'output',1,'NS','trimf',[-4,-2,1]);
a = addmf(a,'output',1,'Z','trimf',[-2,0,2]);
a = addmf(a,'output',1,'PS','trimf',[-1,2,4]);
a = addmf(a,'output',1,'PB','smf',[1,4]);

rulelist = [1 1 1 1;                       %Edit rule base
            2 2 1 1;
            3 3 1 1;
            4 4 1 1;
            5 5 1 1];

a = addrule(a,rulelist);

a1 = setfis(a,'DefuzzMethod','mom');       %Defuzzy
writefis(a1,'tank');                       %Save to fuzzy file "tank.fis"
a2 = readfis('tank');

figure(1);
plotfis(a2);
figure(2);
plotmf(a,'input',1);
figure(3);
```

```
plotmf(a,'output',1);

flag = 0;
if flag = = 1
    showrule(a)                                    % Show fuzzy rule base
    ruleview('tank');                              % Dynamic Simulation
end
disp('------------------------------------------------------------------------');
disp('       fuzzy controller table:e = [-3, +3],u = [-4, +4]          ');
disp('------------------------------------------------------------------------');

for i = 1:1:7
    e(i) = i - 4;
    Ulist(i) = evalfis([e(i)],a2);
end
Ulist = round(Ulist)

e = -3;                                            % Error
u = evalfis([e],a2)                                % Using fuzzy inference
```

模糊控制器的设计仿真程序:chap4_2.m

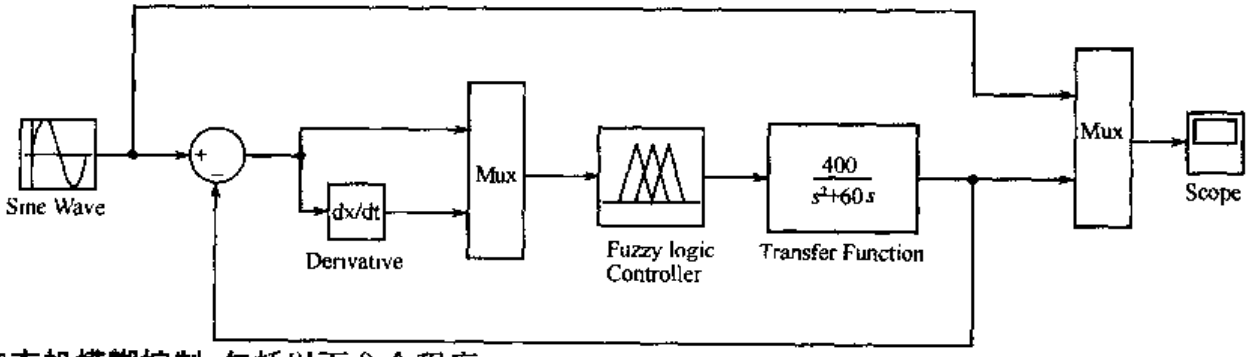
```
% Fuzzy Controller Design
clear all;
close all;

a = newfis('fuzzf');

f1 = 1;
a = addvar(a,'input','e',[-3*f1,3*f1]); % Parameter e
a = addmf(a,'input',1,'NB','zmf',[-3*f1,-1*f1]);
a = addmf(a,'input',1,'NM','trimf',[-3*f1,-2*f1,0]);
a = addmf(a,'input',1,'NS','trimf',[-3*f1,-1*f1,1*f1]);
a = addmf(a,'input',1,'Z','trimf',[-2*f1,0,2*f1]);
a = addmf(a,'input',1,'PS','trimf',[-1*f1,1*f1,3*f1]);
a = addmf(a,'input',1,'PM','trimf',[0,2*f1,3*f1]);
a = addmf(a,'input',1,'PB','smf',[1*f1,3*f1]);

f2 = 1;
a = addvar(a,'input','ec',[-3*f2,3*f2]);% Parameter ec
a = addmf(a,'input',2,'NB','zmf',[-3*f2,-1*f2]);
```

```
a = addmf(a,'input',2,'NM','trimf',[-3*f2, -2*f2,0]);
a = addmf(a,'input',2,'NS','trimf',[-3*f2, -1*f2,1*f2]);
a = addmf(a,'input',2,'Z','trimf',[-2*f2,0,2*f2]);
a = addmf(a,'input',2,'PS','trimf',[-1*f2,1*f2,3*f2]);
a = addmf(a,'input',2,'PM','trimf',[0,2*f2,3*f2]);
a = addmf(a,'input',2,'PB','smf',[1*f2,3*f2]);

f3 = 1.5;
a = addvar(a,'output','u',[-3*f3,3*f3]);% Parameter u
a = addmf(a,'output',1,'NB','zmf',[-3*f3, -1*f3]);
a = addmf(a,'output',1,'NM','trimf',[-3*f3, -2*f3,0]);
a = addmf(a,'output',1,'NS','trimf',[-3*f3, -1*f3,1*f3]);
a = addmf(a,'output',1,'Z','trimf',[-2*f3,0,2*f3]);
a = addmf(a,'output',1,'PS','trimf',[-1*f3,1*f3,3*f3]);
a = addmf(a,'output',1,'PM','trimf',[0,2*f3,3*f3]);
a = addmf(a,'output',1,'PB','smf',[1*f3,3*f3]);

rulelist = [1 1 1 1 1;                          % Edit rule base
            1 2 1 1 1;
            1 3 2 1 1;
            1 4 2 1 1;
            1 5 3 1 1;
            1 6 3 1 1;
            1 7 4 1 1;

            2 1 1 1 1;
            2 2 2 1 1;
            2 3 2 1 1;
            2 4 3 1 1;
            2 5 3 1 1;
            2 6 4 1 1;
            2 7 5 1 1;

            3 1 2 1 1;
            3 2 2 1 1;
            3 3 3 1 1;
            3 4 3 1 1;
            3 5 4 1 1;
            3 6 5 1 1;
            3 7 5 1 1;
```

```
        4 1 2 1 1;
        4 2 3 1 1;
        4 3 3 1 1;
        4 4 4 1 1;
        4 5 5 1 1;
        4 6 5 1 1;
        4 7 6 1 1;

        5 1 3 1 1;
        5 2 3 1 1;
        5 3 4 1 1;
        5 4 5 1 1;
        5 5 5 1 1;
        5 6 6 1 1;
        5 7 6 1 1;

        6 1 3 1 1;
        6 2 4 1 1;
        6 3 5 1 1;
        6 4 5 1 1;
        6 5 6 1 1;
        6 6 6 1 1;
        6 7 7 1 1;

        7 1 4 1 1;
        7 2 5 1 1;
        7 3 5 1 1;
        7 4 6 1 1;
        7 5 6 1 1;
        7 6 7 1 1;
        7 7 7 1 1];

a = addrule(a,rulelist);
 % showrule(a)                                % Show fuzzy rule base

a1 = setfis(a,'DefuzzMethod','mom');          % Defuzzy
writefis(a1,'fuzzf');
a2 = readfis('fuzzf');
disp('.....');
```

```
disp('        fuzzy controller table:e = [-3, +3],ec = [-3, +3]          ');
disp('  ..... ..... ..... ..... ..... ..... ..... ..... ..... ..... ..... ');

Ulist = zeros(7,7);

for i = 1 : 7
    for j = 1 : 7
        e(i) = -4 + i;
        ec(j) = -4 + j;
        Ulist(i,j) = evalfis([e(i),ec(j)],a2);
    end
end

Ulist = ceil(Ulist)
figure(1);
plotfis(a2);
figure(2);
plotmf(a,'input',1);
figure(3);
plotmf(a,'input',2);
figure(4);
plotmf(a,'output',1);
```

模糊控制位置跟踪的 Simulink 仿真程序:chap4_3. mdl



洗衣机模糊控制:包括以下 3 个程序。

(1) **污泥和油脂隶属函数设计仿真程序**:chap4 _4. m

```
% Define N + 1 triangle membership function
clear all;
close all;
N = 2;

x = 0 : 0.1 : 100;
for i = 1 : N + 1
```

```
    f(i) = 100/N*(i - 1);
end

u = trimf(x,[f(1),f(1),f(2)]);
figure(1);
plot(x,u);

for j = 2 : N
    u = trimf(x,[f(j - 1),f(j),f(j + 1)]);
    hold on;
    plot(x,u);
end
u = trimf(x,[f(N),f(N + 1),f(N + 1)]);
hold on;
plot(x,u);
xlabel('x');
ylabel('Degree of membership');
```

(2) **洗涤时间隶属函数设计仿真程序**:chap4_5.m

```
%Define N + 1 triangle membership function
clear all;
close all;
z = 0 : 0.1 : 60;

u = trimf(z,[0,0,10]);
figure(1);
plot(z,u);

u = trimf(z,[0,10,25]);
hold on;
plot(z,u);

u = trimf(z,[10,25,40]);
hold on;
plot(z,u);

u = trimf(z,[25,40,60]);
hold on;
plot(z,u);
```

```
u = trimf(z,[40,60,60]);
hold on;
plot(z,u);

xlabel('z');
ylabel('Degree of membership');
```

(3) **洗衣机模糊控制系统仿真程序**:chap4_6.m

```
%Fuzzy Control for washer
clear all;
close all;

a = newfis('fuzz_wash');

a = addvar(a,'input','x',[0,100]);          %Fuzzy Stain
a = addmf(a,'input',1,'SD','trimf',[0,0,50]);
a = addmf(a,'input',1,'MD','trimf',[0,50,100]);
a = addmf(a,'input',1,'LD','trimf',[50,100,100]);

a = addvar(a,'input','y',[0,100]);          %Fuzzy Axunge
a = addmf(a,'input',2,'NG','trimf',[0,0,50]);
a = addmf(a,'input',2,'MG','trimf',[0,50,100]);
a = addmf(a,'input',2,'LG','trimf',[50,100,100]);

a = addvar(a,'output','z',[0,60]);          %Fuzzy Time
a = addmf(a,'output',1,'VS','trimf',[0,0,10]);
a = addmf(a,'output',1,'S','trimf',[0,10,25]);
a = addmf(a,'output',1,'M','trimf',[10,25,40]);
a = addmf(a,'output',1,'L','trimf',[25,40,60]);
a = addmf(a,'output',1,'VL','trimf',[40,60,60]);

rulelist = [1 1 1 1 1;                      %Edit rule base
            1 2 3 1 1;
            1 3 4 1 1;

            2 1 2 1 1;
            2 2 3 1 1;
            2 3 4 1 1;

            3 1 3 1 1;
```

```
        3 2 4 1 1;
        3 3 5 1 1];

a = addrule(a,rulelist);
showrule(a)                                    % Show fuzzy rule base

a1 = setfis(a,'DefuzzMethod','mom');           % Defuzzy
writefis(a1,'wash');                           % Save to fuzzy file "wash.fis"
a2 = readfis('wash');

figure(1);
plotfis(a2);
figure(2);
plotmf(a,'input',1);
figure(3);
plotmf(a,'input',2);
figure(4);
plotmf(a,'output',1);

ruleview('wash');                              % Dynamic Simulation

x = 60;
y = 70;
z = evalfis([x,y],a2)                          % Using fuzzy inference
```

模糊 PID 仿真程序：包括模糊系统设计程序 chap4_7a.m 及模糊控制程序 chap4_7b.m。

(1) **模糊系统设计程序**：*chap4_7a.m*

```
% Fuzzy Tunning PID Control
clear all;
close all;

a = newfis('fuzzpid');

a = addvar(a,'input','e',[-3,3]);              % Parameter e
a = addmf(a,'input',1,'NB','zmf',[-3,-1]);
a = addmf(a,'input',1,'NM','trimf',[-3,-2,0]);
a = addmf(a,'input',1,'NS','trimf',[-3,-1,1]);
a = addmf(a,'input',1,'Z','trimf',[-2,0,2]);
a = addmf(a,'input',1,'PS','trimf',[-1,1,3]);
```

```
a = addmf(a,'input',1,'PM','trimf',[0,2,3]);
a = addmf(a,'input',1,'PB','smf',[1,3]);

a = addvar(a,'input','ec',[-3,3]);          %Parameter ec
a = addmf(a,'input',2,'NB','zmf',[-3,-1]);
a = addmf(a,'input',2,'NM','trimf',[-3,-2,0]);
a = addmf(a,'input',2,'NS','trimf',[-3,-1,1]);
a = addmf(a,'input',2,'Z','trimf',[-2,0,2]);
a = addmf(a,'input',2,'PS','trimf',[-1,1,3]);
a = addmf(a,'input',2,'PM','trimf',[0,2,3]);
a = addmf(a,'input',2,'PB','smf',[1,3]);

a = addvar(a,'output','kp',[-0.3,0.3]);    %Parameter kp
a = addmf(a,'output',1,'NB','zmf',[-0.3,-0.1]);
a = addmf(a,'output',1,'NM','trimf',[-0.3,-0.2,0]);
a = addmf(a,'output',1,'NS','trimf',[-0.3,-0.1,0.1]);
a = addmf(a,'output',1,'Z','trimf',[-0.2,0,0.2]);
a = addmf(a,'output',1,'PS','trimf',[-0.1,0.1,0.3]);
a = addmf(a,'output',1,'PM','trimf',[0,0.2,0.3]);
a = addmf(a,'output',1,'PB','smf',[0.1,0.3]);

a = addvar(a,'output','ki',[-0.06,0.06]); %Parameter ki
a = addmf(a,'output',2,'NB','zmf',[-0.06,-0.02]);
a = addmf(a,'output',2,'NM','trimf',[-0.06,-0.04,0]);
a = addmf(a,'output',2,'NS','trimf',[-0.06,-0.02,0.02]);
a = addmf(a,'output',2,'Z','trimf',[-0.04,0,0.04]);
a = addmf(a,'output',2,'PS','trimf',[-0.02,0.02,0.06]);
a = addmf(a,'output',2,'PM','trimf',[0,0.04,0.06]);
a = addmf(a,'output',2,'PB','smf',[0.02,0.06]);

a = addvar(a,'output','kd',[-3,3]);          %Parameter kp
a = addmf(a,'output',3,'NB','zmf',[-3,-1]);
a = addmf(a,'output',3,'NM','trimf',[-3,-2,0]);
a = addmf(a,'output',3,'NS','trimf',[-3,-1,1]);
a = addmf(a,'output',3,'Z','trimf',[-2,0,2]);
a = addmf(a,'output',3,'PS','trimf',[-1,1,3]);
a = addmf(a,'output',3,'PM','trimf',[0,2,3]);
a = addmf(a,'output',3,'PB','smf',[1,3]);

rulelist = [1 1 7 1 5 1 1;
```

1 2 7 1 3 1 1;
1 3 6 2 1 1 1;
1 4 6 2 1 1 1;
1 5 5 3 1 1 1;
1 6 4 4 2 1 1;
1 7 4 4 5 1 1;

2 1 7 1 5 1 1;
2 2 7 1 3 1 1;
2 3 6 2 1 1 1;
2 4 5 3 2 1 1;
2 5 5 3 2 1 1;
2 6 4 4 3 1 1;
2 7 3 4 4 1 1;

3 1 6 1 4 1 1;
3 2 6 2 3 1 1;
3 3 6 3 2 1 1;
3 4 5 3 2 1 1;
3 5 4 4 3 1 1;
3 6 3 5 3 1 1;
3 7 3 5 4 1 1;

4 1 6 2 4 1 1;
4 2 6 2 3 1 1;
4 3 5 3 3 1 1;
4 4 4 4 3 1 1;
4 5 3 5 3 1 1;
4 6 2 6 3 1 1;
4 7 2 6 4 1 1;

5 1 5 2 4 1 1;
5 2 5 3 4 1 1;
5 3 4 4 4 1 1;
5 4 3 5 4 1 1;
5 5 3 5 4 1 1;
5 6 2 6 4 1 1;
5 7 2 7 4 1 1;

6 1 5 4 7 1 1;

```
        6 2 4 4 5 1 1;
        6 3 3 5 5 1 1;
        6 4 2 5 5 1 1;
        6 5 2 6 5 1 1;
        6 6 2 7 5 1 1;
        6 7 1 7 7 1 1;

        7 1 4 4 7 1 1;
        7 2 4 4 6 1 1;
        7 3 2 5 6 1 1;
        7 4 2 6 6 1 1;
        7 5 2 6 5 1 1;
        7 6 1 7 5 1 1;
        7 7 1 7 7 1 1];

a = addrule(a,rulelist);
a = setfis(a,'DefuzzMethod','centroid');
writefis(a,'fuzzpid');

a = readfis('fuzzpid');

figure(1);
plotmf(a,'input',1);
figure(2);
plotmf(a,'input',2);
figure(3);
plotmf(a,'output',1);
figure(4);
plotmf(a,'output',2);
figure(5);
plotmf(a,'output',3);
figure(6);
plotfis(a);

fuzzy fuzzpid;
showrule(a);
ruleview fuzzpid;
```

(2) **模糊控制程序**：chap4_7b.m

```
%Fuzzy PID Control
```

```
close all;
clear all;

a = readfis('fuzzpid');                          %Load fuzzpid.fis

ts = 0.001;
sys = tf(5.235e005,[1,87.35,1.047e004,0]);
dsys = c2d(sys,ts,'tustin');
[num,den] = tfdata(dsys,'v');

u_1 = 0.0;u_2 = 0.0;u_3 = 0.0;
y_1 = 0;y_2 = 0;y_3 = 0;

x = [0,0,0]';

e_1 = 0;
ec_1 = 0;

kp0 = 0.40;
kd0 = 1.0;
ki0 = 0.0;

for k = 1:1:3000
time(k) = k*ts;

r(k) = sign(sin(2*pi*k*ts));
%Using fuzzy inference to tuning PID
k_pid = evalfis([e_1,ec_1],a);
kp(k) = kp0 + k_pid(1);
ki(k) = ki0 + k_pid(2);
kd(k) = kd0 + k_pid(3);
u(k) = kp(k)*x(1) + kd(k)*x(2) + ki(k)*x(3);

if k == 300                                      % Adding disturbance(1.0v at time 0.3s)
    u(k) = u(k) + 1.0;
end

y(k) = -den(2)*y_1 - den(3)*y_2 - den(4)*y_3 + num(1)*u(k) + num(2)*u_1 +
num(3)*u_2 + num(4)*u_3;
e(k) = r(k) - y(k);
```

```
%%%%%%%%%%%%%%%%%%Return of PID parameters%%%%%%%%%%%%%%%%%%
    u_3 = u_2;
    u_2 = u_1;
    u_1 = u(k);

    y_3 = y_2;
    y_2 = y_1;
    y_1 = y(k);

    x(1) = e(k);                          % Calculating P
    x(2) = e(k) - e_1;                    % Calculating D
    x(3) = x(3) + e(k) * ts;              % Calculating I

    ec_1 = x(2);
    e_2 = e_1;
    e_1 = e(k);
end

figure(1);
plot(time,r,'b',time,y,'r');
xlabel('time(s)');ylabel('rin,yout');
figure(2);
plot(time,e,'r');
xlabel('time(s)');ylabel('error');
figure(3);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');
figure(4);
plot(time,kp,'r');
xlabel('time(s)');ylabel('kp');
figure(5);
plot(time,ki,'r');
xlabel('time(s)');ylabel('ki');
figure(6);
plot(time,kd,'r');
xlabel('time(s)');ylabel('kd');
```

Sugeno 模糊模型的设计：chap4_8.m

```
%Sugeno type fuzzy model
clear all;
```

```
close all;
ts2 = newfis('ts2','sugeno');

ts2 = addvar(ts2,'input','X',[0 5]);
ts2 = addmf(ts2,'input',1,'little','gaussmf',[1.8 0]);
ts2 = addmf(ts2,'input',1,'big','gaussmf',[1.8 5]);

ts2 = addvar(ts2,'input','Y',[0 10]);
ts2 = addmf(ts2,'input',2,'little','gaussmf',[4.4 0]);
ts2 = addmf(ts2,'input',2,'big','gaussmf',[4.4 10]);

ts2 = addvar(ts2,'output','Z',[-3 15]);
ts2 = addmf(ts2,'output',1,'first area','linear',[-1 1 -3]);
ts2 = addmf(ts2,'output',1,'second area','linear',[1 1 1]);
ts2 = addmf(ts2,'output',1,'third area','linear',[0 -2 2]);
ts2 = addmf(ts2,'output',1,'fourth area','linear',[2 1 -6]);

rulelist = [1 1 1 1 1;
            1 2 2 1 1;
            2 1 3 1 1;
            2 2 4 1 1];

ts2 = addrule(ts2,rulelist);
showrule(ts2);

figure(1);
subplot 211;
plotmf(ts2,'input',1);
xlabel('x'),ylabel('MF Degree of input 1');
subplot 212;
plotmf(ts2,'input',2);
xlabel('x'),ylabel('MF Degree of input 2');

figure(2);
gensurf(ts2);
xlabel('x'),ylabel('y'),zlabel('z');
```

Sugeno 模糊控制仿真程序：包括 Sugeno 模糊模型建模程序和 Sugeno 模糊控制程序。

(1) **Sugeno 模糊模型建模程序**：chap4_9f.m

```
%Local linearization for single inverted pendulum
```

```
clear all;
close all;

g = 9.8;
m = 2;
M = 8;
l = 0.5;
a = 1/(m + M);

%Equation:
A21 = g/(4/3*l - a*m*l);
A = [0 1;
   A21 0];
B2 = - a/(4/3*l - a*m*l);
B = [0;B2];
```

(2) **Sugeno 模糊控制程序**:chap4_9.m

```
%Sugeno type fuzzy control for single inverted pendulum
close all;

P = [-10 - 10i; -10 + 10i];                  %Stable pole point
F = place(A,B,P)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tsu = newfis('tsu','sugeno');
tsu = addvar(tsu,'input','x',[-15,15]*pi/180);
tsu = addmf(tsu,'input',1,'ZR','trimf',[-15,0,15]*pi/180);
tsu = addvar(tsu,'input','dx',[-200,200]*pi/180);
tsu = addmf(tsu,'input',2,'ZR','trimf',[-200,0,200]*pi/180);

tsu = addvar(tsu,'output','u',[-200,600]);
tsu = addmf(tsu,'output',1,'No.1','linear',[-F(1), -F(2)]);

rulelist1 = [1 1 1 1 1];
tsu = addrule(tsu,rulelist1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
model = newfis('model','sugeno');
model = addvar(model,'input','x',[-15,15]*pi/180);
model = addmf(model,'input',1,'ZR','trimf',[-15,0,15]*pi/180);
```

```
model = addvar(model,'input','dx',[-200,200]*pi/180);
model = addmf(model,'input',2,'ZR','trimf',[-200,0,200]*pi/180);

model = addvar(model,'input','u',[-200,600]);

model = addvar(model,'output','dx',[-200,200]*pi/180);
model = addmf(model,'output',1,'No.1','linear',[0 1 0 ]);

model = addvar(model,'output','ddx',[-200,200]*pi/180);
model = addmf(model,'output',2,'No.1','linear',[A(2,1),A(2,2),B(2)]);

rulelist2 = [1 1 0 1 1 1 1];
model = addrule(model,rulelist2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ts = 0.02;
x = [0.2,0]; %Initial state
for k = 1:1:100
    time(k) = k*ts;
    u(k) = evalfis([x(1),x(2)],tsu);          %Using fuzzy T-S model "tsu.fis"
    dx = evalfis([x(1),x(2),u(k)],model);    %Using fuzzy T-S model "model.fis"
    x = x + ts*dx;

    y1(k) = x(1);
    y2(k) = x(2);
end
figure(1);
subplot(211);
plot(time,y1);
xlabel('time(s)'),ylabel('Angle');
subplot(212);
plot(time,y2);
xlabel('time(s)'),ylabel('Angle rate');

figure(2);
plot(time,u);
xlabel('time(s)'),ylabel('Control input');

figure(3);
plotmf(tsu,'input',1);
```

```
figure(4);
plotmf(tsu,'input',2);

showrule(tsu);
showrule(model);
```

第5章　自适应模糊控制

模糊控制器的设计不依靠被控对象的模型，但它却非常依靠控制专家或操作者的经验知识。模糊控制的突出优点是能够比较容易地将人的控制经验融入控制器中，但若缺乏这样的控制经验，很难设计出高水平的模糊控制器。而且，由于模糊控制器采用了 IF-THEN 控制规则，不便于控制参数的学习和调整，使得构造具有自适应的模糊控制器比较困难。

自适应模糊控制是指具有自适应学习算法的模糊逻辑系统，其学习算法是依靠数据信息来调整模糊逻辑系统的参数。一个自适应模糊控制器可以用一个单一的自适应模糊系统构成，也可以用若干个自适应模糊系统构成。与传统的自适应控制相比，自适应模糊控制的优越性在于它可以利用操作人员提供的语言性模糊信息，而传统的自适应控制则不能。这一点对具有高度不确定因素的系统尤其重要。

自适应模糊控制有两种不同的形式：一种是直接自适应模糊控制，即根据实际系统性能与理想性能之间的偏差，通过一定的方法来直接调整控制器的参数；另一种是间接自适应模糊控制，即通过在线辨识获得控制对象的模型，然后根据所得模型在线设计模糊控制器。

5.1　模糊逼近

5.1.1　模糊系统的设计

设二维模糊系统 $g(x)$ 为集合 $U=[\alpha_1,\beta_1]\times[\alpha_2,\beta_2]\subset R^2$ 上的一个函数，其解析式形式未知。假设对任意一个 $x\in U$，都能得到 $g(x)$，则可设计一个逼近 $g(x)$ 的模糊系统。模糊系统的设计步骤为：

步骤 1：在 $[\alpha_i,\beta_i]$ 上定义 $N_i(i=1,2)$ 个标准的、一致的和完备的模糊集 $A_i^1,A_i^2,\cdots,A_i^{N_i}$。

步骤 2：组建 $M=N_1\times N_2$ 条模糊集 IF-THEN 规则，即

$$R_u^{i_1i_2}：\text{如果 } x_1 \text{ 为 } A_1^{i_1} \text{ 且 } x_2 \text{ 为 } A_2^{i_2}，\text{则 } y \text{ 为 } B^{i_1i_2}$$

式中，$i_1=1,2,\cdots,N_1$，$i_2=1,2,\cdots,N_2$，将模糊集 $B^{i_1i_2}$ 的中心（用 $\bar{y}^{i_1i_2}$ 表示）选择为

$$\bar{y}^{i_1i_2}=g(x_1,x_2)=g(e_1^{i_1},e_2^{i_2}) \tag{5.1}$$

步骤 3：采用乘积推理机、单值模糊器和中心平均解模糊器，根据 $M=N_1\times N_2$ 条规则来构造模糊系统 $f(x)$，得

$$f(x)=\frac{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\bar{y}^{i_1i_2}(\mu_{A_1}^{i_1}(x_1)\mu_{A_2}^{i_2}(x_2))}{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\mu_{A_1}^{i_1}(x_1)\mu_{A_2}^{i_2}(x_2)} \tag{5.2}$$

5.1.2　模糊系统的逼近精度

万能逼近定理表明模糊系统是除多项函数逼近器、神经网络之外的一个新的万能逼近器。

模糊系统较之其他逼近器的优势在于它能够有效地利用语言信息的能力。万能逼近定理是模糊逻辑系统用于非线性系统建模的理论基础，同时也从根本上解释了模糊系统在实际中得到成功应用的原因。

万能逼近定理[11] 令 $f(x)$ 为式(5.2)中的二维模糊系统，$g(x)$ 为式(5.1)中的未知函数，如果 $g(x)$ 在 $U=[\alpha_1,\beta_1]\times[\alpha_1,\beta_2]$ 上是连续可微的，则

$$\| g-f \|_\infty \leqslant \left\| \frac{\partial g}{\partial x_1} \right\|_\infty h_1 + \left\| \frac{\partial g}{\partial x_2} \right\|_\infty h_2 \tag{5.3}$$

模糊系统的逼近精度为

$$h_i = \max_{1\leqslant j\leqslant N_i-1} | e_i^{j+1} - e_i^j | \quad (i=1,2) \tag{5.4}$$

式中，无穷维范数 $\| \cdot \|_\infty$ 定义为 $\| d(x) \|_\infty = \sup_{x\in U} | d(x) |$，$e_i^j$ 为 x_i 在第 A_i^j 个模糊集上的中间值或边界值，$j=1$ 和 $j=N_i$ 时为边界值。

由式(5.4)可知：假设 x_i 的模糊集的个数为 N_i，其变化范围的长度为 L_i，则模糊系统的逼近精度满足 $h_i=\dfrac{L_i}{N_i-1}$，即 $N_i=\dfrac{L_i}{h_i}+1$。

由该定理可得到以下结论：

(1)形如式(5.2)的模糊系统是万能逼近器，对任意给定的 $\varepsilon>0$，都可将 h_1 和 h_2 选得足够小，使 $\left\| \dfrac{\partial g}{\partial x_1} \right\|_\infty h_1 + \left\| \dfrac{\partial g}{\partial x_2} \right\|_\infty h_2 < \varepsilon$ 成立，从而保证 $\sup_{x\in U} | g(x)-f(x) | = \| g-f \|_\infty < \varepsilon$。

(2)通过对每个 x_i 定义更多的模糊集可以得到更为准确的逼近器，即规则越多，所产生的模糊系统越有效。

(3)为了设计一个具有预定精度的模糊系统，必须知道 $g(x)$ 关于 x_1 和 x_2 的导数边界，即 $\left\| \dfrac{\partial g}{x_1} \right\|_\infty$ 和 $\left\| \dfrac{\partial g}{x_2} \right\|_\infty$。同时，在设计过程中，还必须知道 $g(x)$ 在 $x=(e_1^{i_1},e_2^{i_2})(i_1=1,2,\cdots,N_1;i_2=1,2,\cdots,N_2)$ 处的值。

5.1.3 仿真实例

例5.1 针对一维函数 $g(x)$，设计一个模糊系统 $f(x)$，使之一致地逼近定义在 $U=[-3,3]$ 上的连续函数 $g(x)=\sin(x)$，所需精度为 $\varepsilon=0.2$，即 $\sup_{x\in U} | g(x)-f(x) | < \varepsilon$。

由 $\| \cdot \|_\infty$ 定义可知，$\left\| \dfrac{\partial g}{\partial x} \right\|_\infty = \| \cos(x) \|_\infty = 1$，由式(5.3)可知，$\| g-f \|_\infty \leqslant \left\| \dfrac{\partial g}{\partial x} \right\|_\infty h = h$，故取 $h\leqslant 0.2$ 满足精度要求。取 $h=0.2$，则模糊集的个数为 $N=\dfrac{L}{h}+1=31$。在 $U=[-3,3]$ 上定义31个具有三角形隶属函数的模糊集 A^j，如图5-1所示。所设计的模糊系统为

$$f(x)=\frac{\sum_{j=1}^{31}\sin(e^j)\mu_A^j(x)}{\sum_{j=1}^{31}\mu_A^j(x)} \tag{5.5}$$

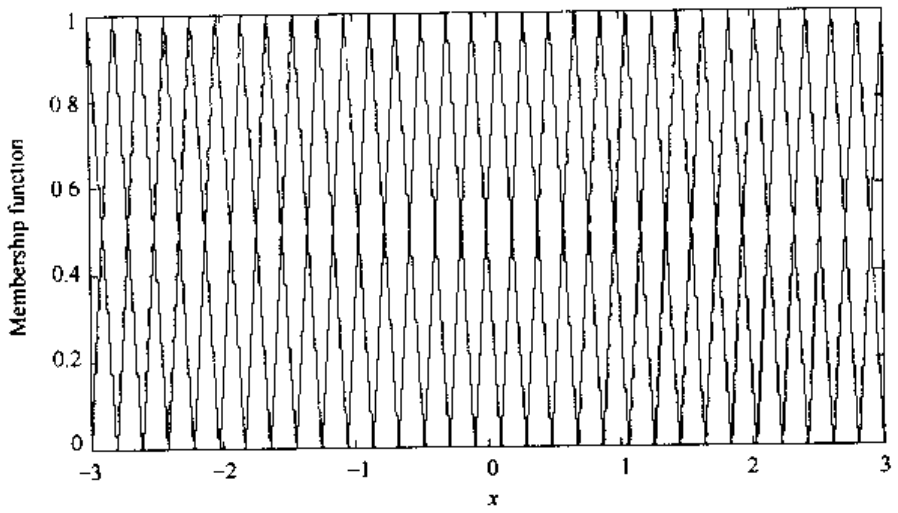


图 5-1　隶属函数

一维函数逼近仿真程序见附录程序 chap5_1.m，逼近效果如图 5-2 和图 5-3 所示。

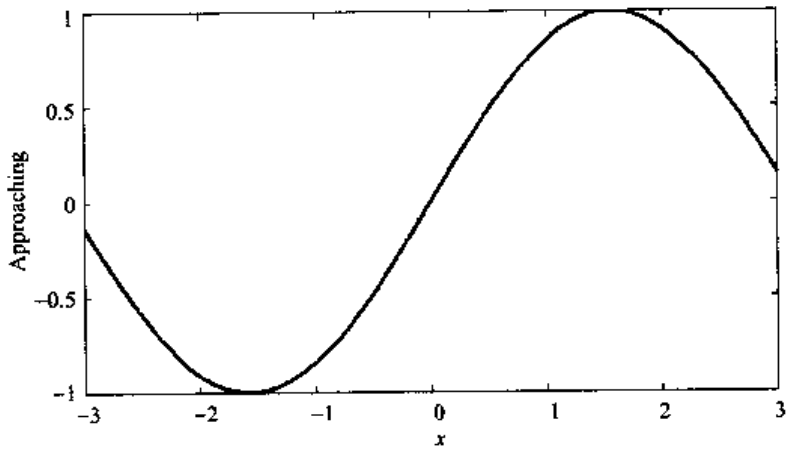


图 5-2　模糊逼近

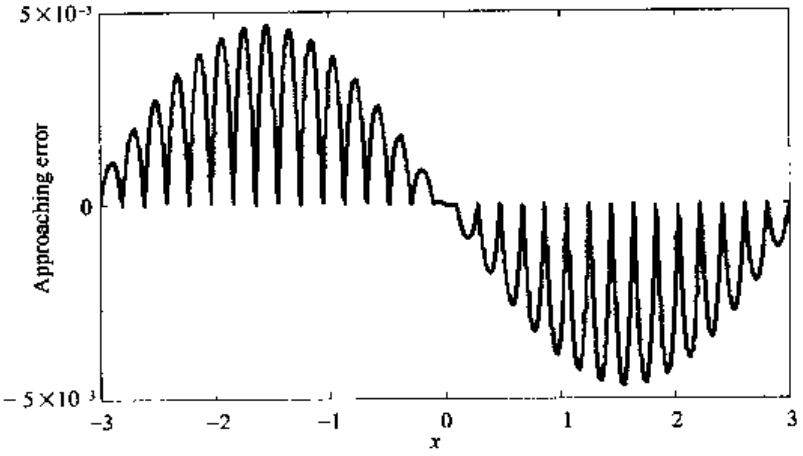


图 5-3　逼近误差

例 5.2　针对二维函数 $g(x)$，设计一个模糊系统 $f(x)$，使之一致地逼近定义在 $U=[-1,1]\times[-1,1]$ 上的连续函数 $g(x)=0.52+0.1x_1+0.28x_2-0.06x_1x_2$，所需精度为 $\varepsilon=0.1$。

由于 $\left\| \dfrac{\partial g}{\partial x_1} \right\|_\infty = \sup_{x \in U} |0.1-0.06x_2| = 0.16$，$\left\| \dfrac{\partial g}{\partial x_2} \right\|_\infty = \sup_{x \in U} |0.28-0.06x_1| = 0.34$，由式(5.3)可知，取 $h_1 = 0.2, h_2 = 0.2$ 时，有 $\| g-f \|_\infty \leqslant 0.16 \times 0.2 + 0.34 \times 0.2 = 0.1$，满足精度要求。由于 $L = 2$，此时模糊集的个数为 $N = \dfrac{L}{h} + 1 = 11$，即 x_1 和 x_2 分别在 $U = [-1,1]$ 上定义 11 个具有三角形隶属函数的模糊集 A^i。

所设计的模糊系统为

$$f(x) = \frac{\sum_{i_1=1}^{11}\sum_{i_2=1}^{11} g(e^{i_1}, e^{i_2})\mu_A^{i_1}(x_1)\mu_A^{i_2}(x_2)}{\sum_{i_1=1}^{11}\sum_{i_2=1}^{11}\mu_A^{i_1}(x_1)\mu_A^{i_2}(x_2)} \tag{5.6}$$

该模糊系统由 $11 \times 11 = 121$ 条规则来逼近函数 $g(x)$。

二维函数逼近仿真程序见附录程序 chap5_2.m。x_1 和 x_2 的隶属函数及 $g(x)$ 的逼近效果如图 5-4 至图 5-7 所示。

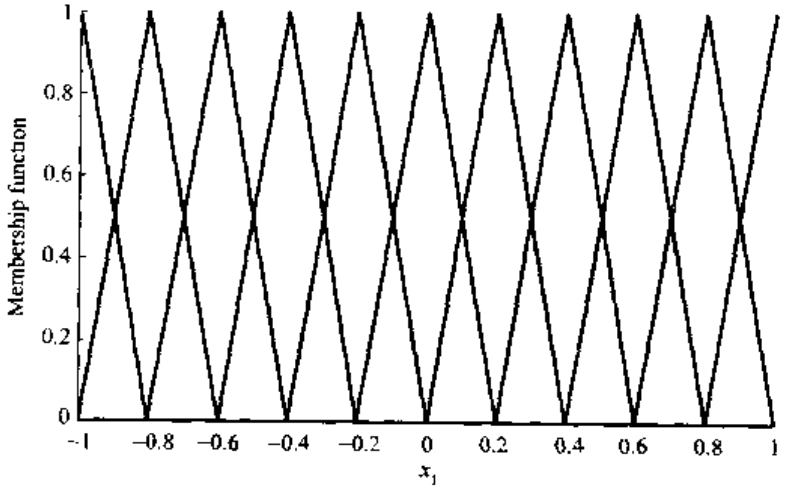


图 5-4　x_1 的隶属函数

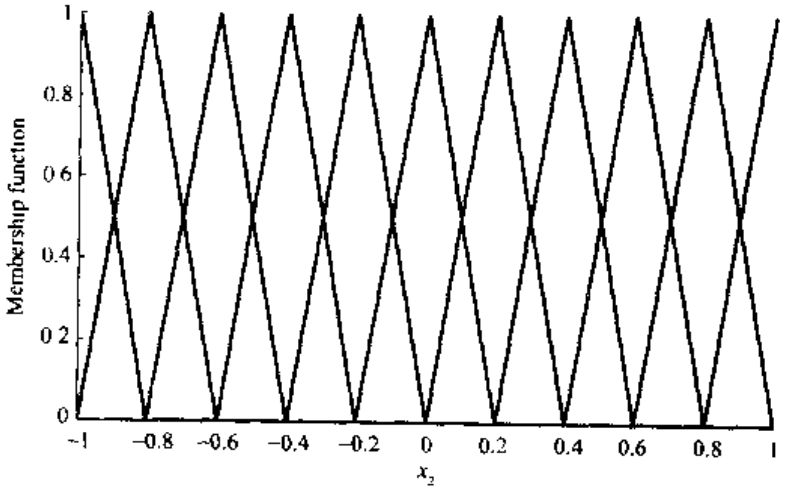


图 5-5　x_2 的隶属函数

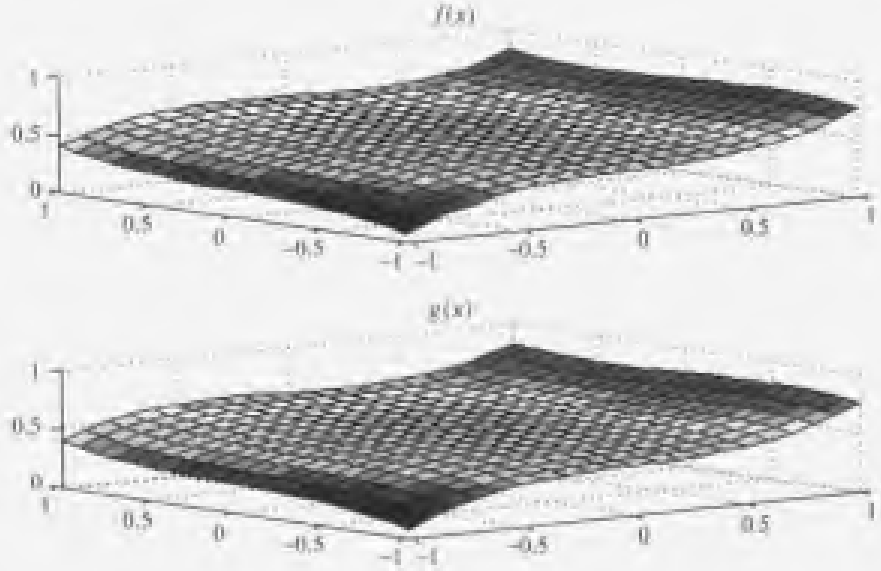


图 5-6 模糊逼近

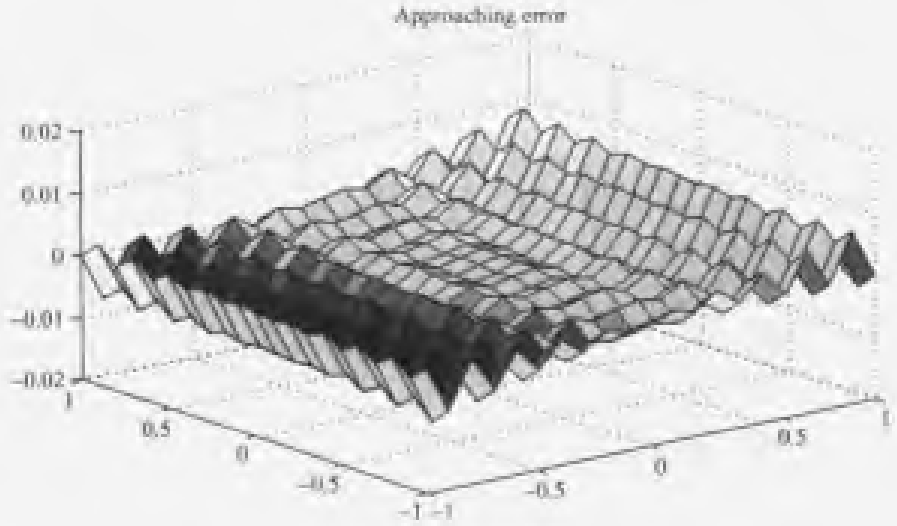


图 5-7 逼近误差

5.2 间接自适应模糊控制

5.2.1 问题描述

考虑如下 n 阶非线性系统

$$\begin{cases} x^{(n)} = f(x,\dot{x},\cdots,x^{(n-1)}) + g(x,\dot{x},\cdots,x^{(n-1)})u \\ y = x \end{cases} \tag{5.7}$$

式中，f 和 g 为未知非线性函数，$u \in R^r$ 和 $y \in R^r$ 分别为系统的输入和输出。

设位置指令为 y_m，令

$$e = y_m - y = y_m - x, \boldsymbol{e} = (e,\dot{e},\cdots,e^{(n-1)})^{\mathrm{T}} \tag{5.8}$$

选择 $\boldsymbol{K} = (k_n,\cdots,k_1)^{\mathrm{T}}$，使多项式 $s^n + k_1 s^{(n-1)} + \cdots + k_n$ 的所有根都在复平面左半平面上。取控制律为

$$u^* = \frac{1}{g(\boldsymbol{x})}[-f(\boldsymbol{x}) + y_m^{(n)} + \boldsymbol{K}^{\mathrm{T}}\boldsymbol{e}] \tag{5.9}$$

将式(5.9)代入式(5.7)，得到闭环控制系统的方程为

$$e^{(n)} + k_1 e^{(n-1)} + \cdots + k_n e = 0 \tag{5.10}$$

由 $\boldsymbol{K}$ 的选取，可得 $t \to \infty$ 时 $e(t) \to 0$，即系统的输出 y 渐近地收敛于理想输出 y_m。

如果非线性函数 $f(\boldsymbol{x})$ 和 $g(\boldsymbol{x})$ 是已知的，则可以选择控制 u 来消除其非线性的性质，然后再根据线性控制理论设计控制器。

5.2.2　控制器的设计

如果 $f(\boldsymbol{x})$ 和 $g(\boldsymbol{x})$ 未知，控制律式(5.9)很难实现。可采用模糊系统 $\hat{f}(\boldsymbol{x})$ 和 $\hat{g}(\boldsymbol{x})$ 代替 $f(\boldsymbol{x})$ 和 $g(\boldsymbol{x})$，实现自适应模糊控制。

1. 基本的模糊系统

以 $\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f)$ 来逼近 $f(\boldsymbol{x})$ 为例，可用以下两步构造模糊系统 $\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f)$。

步骤 1：对变量 $x_i (i = 1,2,\cdots,n)$，定义 p_i 个模糊集合 $A_i^{l_i} (l_i = 1,2,\cdots,p_i)$。

步骤 2：采用以下 $\prod_{i=1}^{n} p_i$ 条模糊规则来构造模糊系统 $\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f)$：

$$R^{(j)}: \text{IF} \quad x_1 \text{ is } A_1^{l_1} \quad \text{and} \cdots \text{and} \quad x_n \text{ is } A_1^{l_n} \quad \text{THEN} \quad \hat{f} \text{ is } E^{l_1 \cdots l_n} \tag{5.11}$$

式中，$l_i = 1,2,\cdots,p_i; i = 1,2,\cdots,n$。

采用乘积推理机、单值模糊器和中心平均解模糊器，则模糊系统的输出为

$$\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f) = \frac{\sum_{l_1=1}^{p_1} \cdots \sum_{l_n=1}^{p_n} \bar{y}_f^{l_1 \cdots l_n} \left(\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right)}{\sum_{l_1=1}^{p_1} \cdots \sum_{l_n=1}^{p_n} \left(\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right)} \tag{5.12}$$

式中，$\mu_{A_i^{l_i}}(x_i)$ 为 x_i 的隶属函数。

令 $\bar{y}_f^{l_1 \cdots l_n}$ 为自由参数，放在集合 $\theta_f \in R^{\prod_{i=1}^{n} p_i}$ 中。引入向量 $\boldsymbol{\xi}(\boldsymbol{x})$，式(5.12)变为

$$\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f) = \boldsymbol{\theta}_f^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) \tag{5.13}$$

式中，$\boldsymbol{\xi}(\boldsymbol{x})$ 为 $\prod_{i=1}^{n} p_i$ 维向量，其第 $l_1 \cdots, l_n$ 个元素为

$$\xi_{l_1 \cdots l_n}(\boldsymbol{x}) = \frac{\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i)}{\sum_{l_1=1}^{p_1} \cdots \sum_{l_n=1}^{p_n} \left(\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right)} \tag{5.14}$$

同理，可构造模糊系统 $\hat{g}(\boldsymbol{x} \mid \boldsymbol{\theta}_g)$ 逼近 $g(\boldsymbol{x})$。

2. 自适应模糊控制器的设计

采用模糊系统逼近 f 和 g，则控制律式(5.9)变为

$$u=\frac{1}{\hat{g}(\boldsymbol{x}\mid\boldsymbol{\theta}_g)}[-\hat{f}(\boldsymbol{x}\mid\boldsymbol{\theta}_f)+y_m^{(n)}+\boldsymbol{K}^{\mathrm{T}}\boldsymbol{e}] \tag{5.15}$$

$$\hat{f}(\boldsymbol{x}\mid\boldsymbol{\theta}_f)=\boldsymbol{\theta}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}),\hat{g}(\boldsymbol{x}\mid\boldsymbol{\theta}_g)=\boldsymbol{\theta}_g^{\mathrm{T}}\boldsymbol{\eta}(\boldsymbol{x}) \tag{5.16}$$

式中，$\boldsymbol{\xi}(\boldsymbol{x})$ 为模糊向量，参数 $\boldsymbol{\theta}_f^{\mathrm{T}}$ 和 $\boldsymbol{\theta}_g^{\mathrm{T}}$ 根据自适应律而变化。

设计自适应律为

$$\dot{\boldsymbol{\theta}}_f=-\gamma_1\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{\xi}(\boldsymbol{x}) \tag{5.17}$$

$$\dot{\boldsymbol{\theta}}_g=-\gamma_2\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{\eta}(\boldsymbol{x})u \tag{5.18}$$

式中，γ_1，γ_2 为正常数。

自适应模糊控制系统如图 5-8 所示。

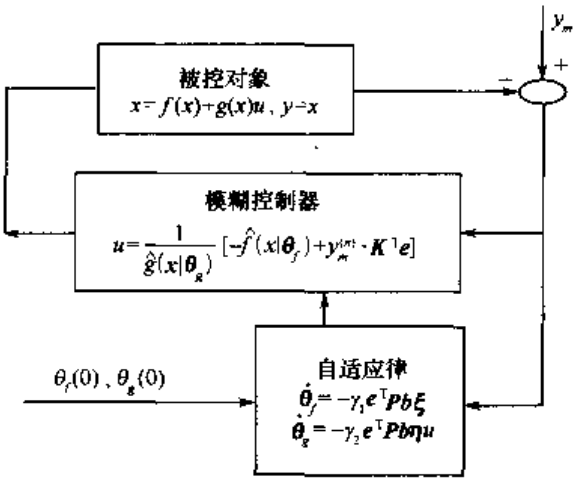


图 5-8　自适应模糊控制系统

3. 稳定性分析

将式(5.15)代入式(5.7)可得如下模糊控制系统的闭环动态方程式

$$\boldsymbol{e}^{(n)}=-\boldsymbol{K}^{\mathrm{T}}\boldsymbol{e}+[\hat{f}(\boldsymbol{x}\mid\boldsymbol{\theta}_f)-f(\boldsymbol{x})]+[\hat{g}(\boldsymbol{x}\mid\boldsymbol{\theta}_g)-g(\boldsymbol{x})]u \tag{5.19}$$

令

$$\boldsymbol{A}=\begin{bmatrix}0&1&0&0&\cdots&0&0\\0&0&1&0&\cdots&0&0\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\0&0&0&0&\cdots&0&1\\-k_n&-k_{n-1}&\cdots&\cdots&\cdots&\cdots&-k_1\end{bmatrix},\boldsymbol{b}=\begin{bmatrix}0\\0\\\cdots\\0\\1\end{bmatrix} \tag{5.20}$$

则动态方程式(5.19)可写为向量形式

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda e} - \boldsymbol{b}\{[\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f) - f(\boldsymbol{x})] + [\hat{g}(\boldsymbol{x} \mid \boldsymbol{\theta}_g) - g(\boldsymbol{x})]u\} \tag{5.21}$$

设最优参数为

$$\boldsymbol{\theta}_f^* = \arg \min_{\boldsymbol{\theta}_f \in \Omega_f} [\sup_{x \in R^n} | \hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f) - f(\boldsymbol{x}) |] \tag{5.22}$$

$$\boldsymbol{\theta}_g^* = \arg \min_{\boldsymbol{\theta}_g \in \Omega_g} [\sup_{x \in R^n} | \hat{g}(\boldsymbol{x} \mid \boldsymbol{\theta}_g) - g(\boldsymbol{x}) |] \tag{5.23}$$

式中，Ω_f 和 Ω_g 分别为 $\boldsymbol{\theta}_f$ 和 $\boldsymbol{\theta}_g$ 的集合。

定义最小逼近误差为

$$\omega = [\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f^*) - f(\boldsymbol{x})] + [\hat{g}(\boldsymbol{x} \mid \boldsymbol{\theta}_g^*) - g(\boldsymbol{x})]u \tag{5.24}$$

式(5.21)可写为

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda e} + \boldsymbol{b}\{[\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f) - \hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f^*)] + [\hat{g}(\boldsymbol{x} \mid \boldsymbol{\theta}_g) - \hat{g}(\boldsymbol{x} \mid \boldsymbol{\theta}_g^*)]u + \omega\} \tag{5.25}$$

将式(5.16)代入式(5.25)，可得闭环动态方程为

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda e} + \boldsymbol{b}[(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) + (\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)^{\mathrm{T}} \boldsymbol{\eta}(\boldsymbol{x}) u + \omega] \tag{5.26}$$

该方程清晰地描述了跟踪误差和控制参数 $\boldsymbol{\theta}_f$，$\boldsymbol{\theta}_g$ 之间的关系。自适应律的任务是为 $\boldsymbol{\theta}_f$，$\boldsymbol{\theta}_g$ 确定一个调节机理，使得跟踪误差 $\boldsymbol{e}$ 和参数误差 $\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*$，$\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*$ 达到最小。

定义 Lyapunov 函数

$$V = \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e} + \frac{1}{2\gamma_1}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*) + \frac{1}{2\gamma_2}(\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)^{\mathrm{T}}(\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*) \tag{5.27}$$

式中，γ_1，γ_2 是正的常数，$\boldsymbol{P}$ 为一个正定矩阵且满足 Lyapunov 方程

$$\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{\Lambda} = -\boldsymbol{Q} \tag{5.28}$$

式中，$\boldsymbol{Q}$ 是一个任意的 $n \times n$ 正定矩阵，$\boldsymbol{\Lambda}$ 由式(5.20)给出。

取 $V_1 = \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}$，$V_2 = \frac{1}{2\gamma_1}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)$，$V_3 = \frac{1}{2\gamma_2}(\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)^{\mathrm{T}}(\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)$。令 $\boldsymbol{M} = \boldsymbol{b}[(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) + (\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}\boldsymbol{\eta}(\boldsymbol{x})u + \omega]$，则式(5.26)变为

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda e} + \boldsymbol{M}$$

$$\dot{V}_1 = \frac{1}{2}\dot{\boldsymbol{e}}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e} + \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\dot{\boldsymbol{e}} = \frac{1}{2}(\boldsymbol{e}^{\mathrm{T}}\boldsymbol{\Lambda}^{\mathrm{T}} + \boldsymbol{M}^{\mathrm{T}})\boldsymbol{P}\boldsymbol{e} + \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{\Lambda e} + \boldsymbol{M})$$

$$= \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}(\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{\Lambda})\boldsymbol{e} + \frac{1}{2}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e} + \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}$$

$$= -\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e} + \frac{1}{2}(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e} + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}) = -\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}$$

即

$$\dot{V}_1 = -\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\omega + (\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{\xi}(\boldsymbol{x}) + (\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)^{\mathrm{T}}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{\eta}(\boldsymbol{x})u$$

$$\dot{V}_2 = \frac{1}{\gamma_1}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}\dot{\boldsymbol{\theta}}_f$$

$$\dot{V}_3 = \frac{1}{\gamma_2}(\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)^{\mathrm{T}}\dot{\boldsymbol{\theta}}_g$$

V 的导数为

$$\begin{aligned}\dot{V} &= \dot{V}_1 + \dot{V}_2 + \dot{V}_3 \\ &= -\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\omega + \frac{1}{\gamma_1}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_f^*)^{\mathrm{T}}[\dot{\boldsymbol{\theta}}_f + \gamma_1\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{\xi}(\boldsymbol{x})] + \\ &\quad \frac{1}{\gamma_2}(\boldsymbol{\theta}_g - \boldsymbol{\theta}_g^*)^{\mathrm{T}}[\dot{\boldsymbol{\theta}}_g + \gamma_2\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\boldsymbol{\eta}(\boldsymbol{x})u]\end{aligned} \tag{5.29}$$

将自适应律式(5.17)和式(5.18)代入式(5.29),得

$$\dot{V} = -\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}\omega \tag{5.30}$$

由于 $-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e} \leqslant 0$,通过选取最小逼近误差 ω 非常小的模糊系统,可实现 $\dot{V} \leqslant 0$。

5.2.3 仿真实例

被控对象取单级倒立摆,如图 5-9 所示,其动态方程为

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{g_0\sin x_1 - mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l(4/3 - m\cos^2 x_1/(m_c+m))} + \frac{\cos x_1/(m_c+m)}{l(4/3 - m\cos^2 x_1/(m_c+m))}u = f(x) + g(x)u$$

式中,x_1 和 x_2 分别为摆角和摆速,$g_0 = 9.8\mathrm{m/s^2}$,$m_c = 1\mathrm{kg}$ 为小车质量,m 为摆杆质量,$m = 0.1\mathrm{kg}$,l 为摆长的一半,$l = 0.5\mathrm{m}$,u 为控制输入。

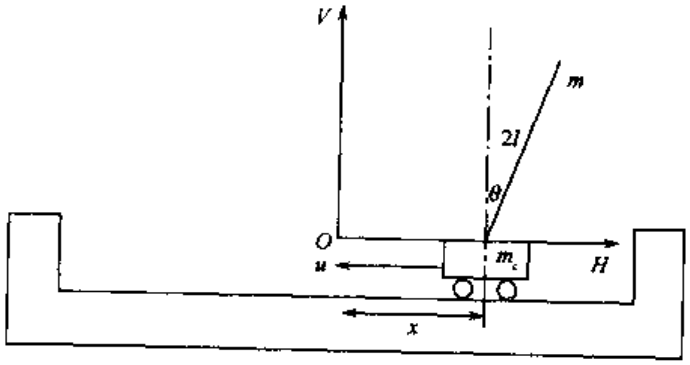


图 5-9 单级倒立摆系统示意图

位置指令为 $x_d(t) = 0.1\sin(\pi t)$。取以下 5 种隶属函数:$\mu_{\mathrm{NM}}(x_i) = \exp[-((x_i + \pi/6)/(\pi/24))^2]$,$\mu_{\mathrm{NS}}(x_i) = \exp[-((x_i+\pi/12)/(\pi/24))^2]$,$\mu_{\mathrm{Z}}(x_i) = \exp[-(x_i/(\pi/24))^2]$,$\mu_{\mathrm{PS}}(x_i) = \exp[-((x_i - \pi/12)/(\pi/24))^2]$,$\mu_{\mathrm{PM}}(x_i) = \exp[-((x_i - \pi/6)/(\pi/24))^2]$。则用于逼近 f 和 g 的模糊规则分别有 25 条。

根据隶属函数设计程序,可得到隶属函数图,如图 5-10 所示。

倒立摆初始状态为[$\pi/60$,0],$\boldsymbol{\theta}_f$ 和 $\boldsymbol{\theta}_g$ 中各元素的初始值取 0.10,采用控制律式(5.15),自适应律采用式(5.17)和式(5.18),取 $\boldsymbol{\eta}(\boldsymbol{x}) = \boldsymbol{\xi}(\boldsymbol{x})$。取 $\boldsymbol{Q} = \begin{bmatrix}10 & 0\\ 0 & 10\end{bmatrix}$,$k_1 = 2$,$k_2 = 1$。考虑到

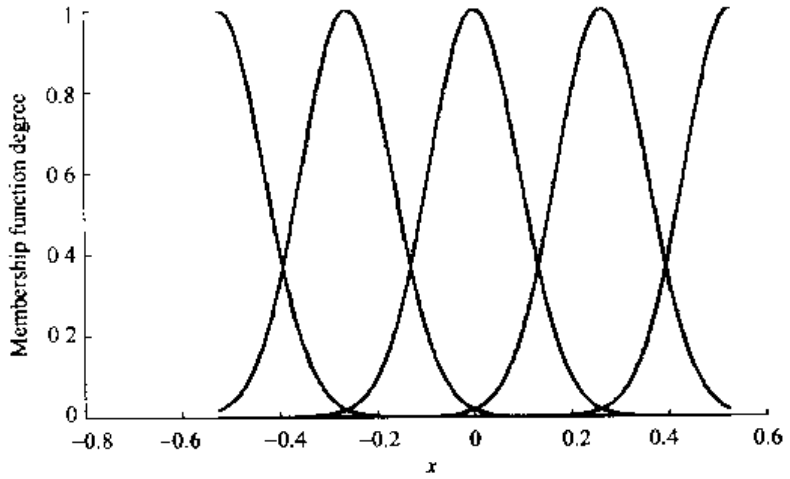


图 5-10　x_i 的隶属函数

$f(x_1,x_2)$ 的取值范围比 $g(x_1,x_2)$ 的取值范围大得多，自适应参数取 $\gamma_1=50,\gamma_2=1$。

在程序中，分别用 FS1，FS2 和 FS 表示模糊系统 $\xi(x)$ 的分子、分母及 $\xi(x)$，仿真结果如图 5-11 至图 5-14 所示。

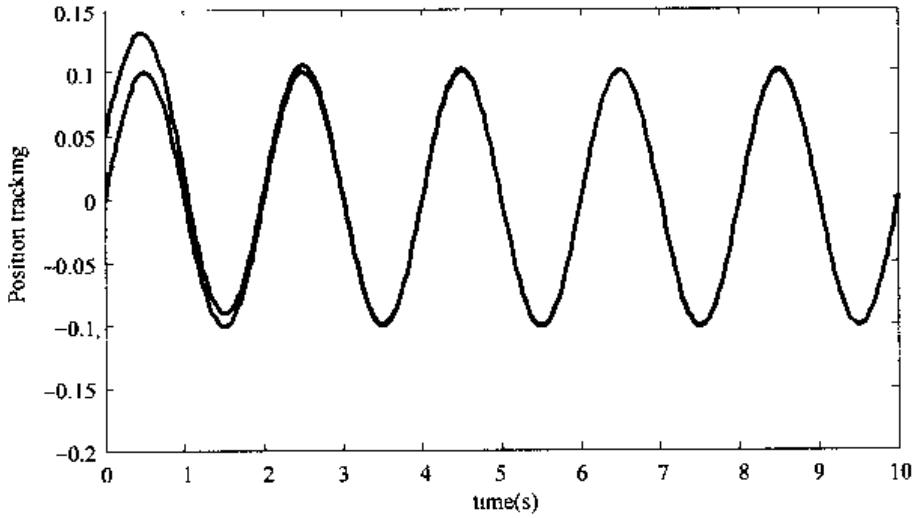


图 5-11　位置跟踪

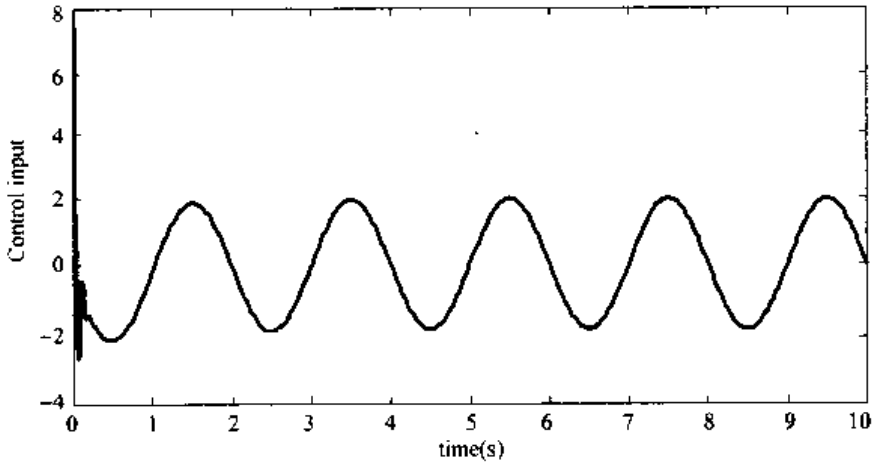


图 5-12　控制输入信号

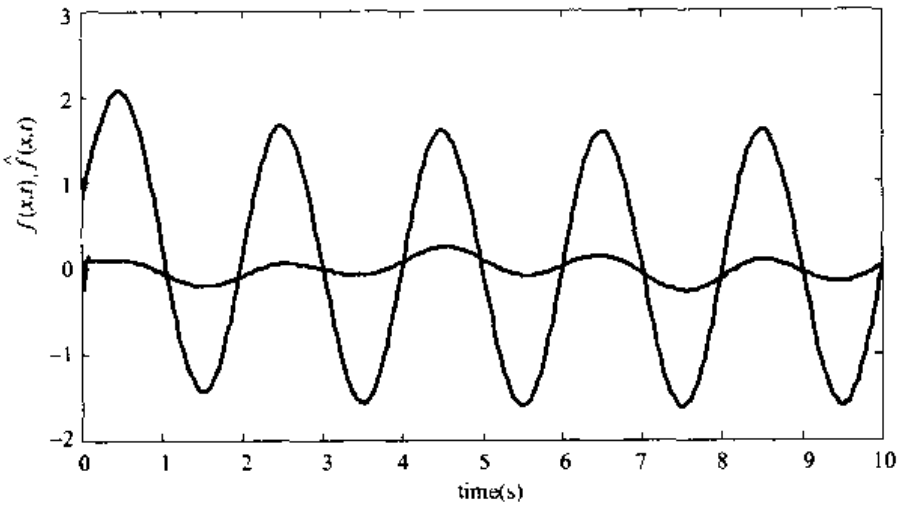


图 5-13　$f(x,t)$ 及 $\hat{f}(x,t)$ 的变化

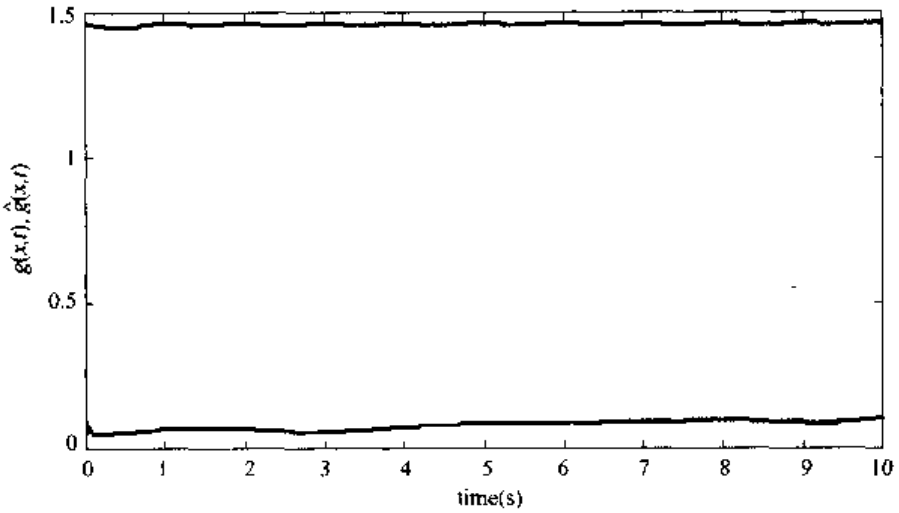


图 5-14　$g(x,t)$ 及 $\hat{g}(x,t)$ 的变化

间接模糊自适应控制仿真程序有5个：① 隶属函数设计程序，chap5_3mf. m；②Simulink主程序，chap5_3sim. mdl；③控制器S函数，chap5_3s. m；④被控对象S函数，chap5_3plant. m；⑤作图程序，chap5_3plot. m。程序见附录。

5.3　直接自适应模糊控制

直接自适应模糊控制和间接自适应模糊控制所采用的规则形式不同。间接自适应模糊控制利用的是被控对象的知识，而直接自适应模糊控制采用的是控制知识。

5.3.1　问题描述

考虑如下方程所描述的研究对象

$$x^{(n)} = f(x,\dot{x},\cdots,x^{(n-1)}) + bu \tag{5.31}$$

$$y = x \tag{5.32}$$

式中，f 为未知函数，b 为未知的正常数。

直接自适应模糊控制采用下面 IF-THEN 模糊规则来描述控制知识，即

$$\text{如果 } x_1 \text{ 是 } P_1^r \text{ 且 } \cdots \text{ 且 } x_n \text{ 是 } P_n^r, \text{则 } u \text{ 是 } Q^r \tag{5.33}$$

式中，P_i^r，Q^r 为 R 中的模糊集合，且 $r = 1,2,\cdots,L_u$。

设位置指令为 y_m，令

$$e = y_m - y = y_m - x, \boldsymbol{e} = (e,\dot{e},\cdots,e^{(n-1)})^{\mathrm{T}} \tag{5.34}$$

选择 $\boldsymbol{K} = (k_n,\cdots,k_1)^{\mathrm{T}}$，使多项式 $s^n + k_1 s^{(n-1)} + \cdots + k_n$ 的所有根都在复平面左半平面上。取控制律为

$$u^* = \frac{1}{b}[-f(\boldsymbol{x}) + y_m^{(n)} + \boldsymbol{K}^{\mathrm{T}}\boldsymbol{e}] \tag{5.35}$$

将式(5.35) 代入式(5.31)，得到闭环控制系统的方程为

$$e^{(n)} + k_1 e^{(n-1)} + \cdots + k_n e = 0 \tag{5.36}$$

由 $\boldsymbol{K}$ 的选取，可得 $t \to \infty$ 时 $e(t) \to 0$，即系统的输出 y 渐近地收敛于理想输出 y_m。

直接自适应模糊控制基于模糊系统设计一个反馈控制器 $u = u(\boldsymbol{x} \mid \theta)$ 和一个调整参数向量 $\boldsymbol{\theta}$ 的自适应律，使得系统输出 y 尽可能地跟踪理想输出 y_m。

5.3.2　控制器的设计

直接自适应模糊控制器为

$$u = u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta}) \tag{5.37}$$

式中，u_{D} 是一个模糊系统，$\boldsymbol{\theta}$ 是可调参数集合。

模糊系统 u_{D} 可由以下两步来构造：

步骤 1：对变量 $x_i(i = 1,2,\cdots,n)$，定义 m_i 个模糊集合 $A_i^{l_i}(l_i = 1,2,\cdots,m_i)$。

步骤 2：用以下 $\prod_{i=1}^{n} m_i$ 条模糊规则来构造模糊系统 $u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta})$，即

$$\text{如果 } x_1 \text{ 是 } A_1^{l_1} \text{ 且 } \cdots \text{ 且 } x_n \text{ 是 } A_n^{l_n}, \text{则 } u_{\mathrm{D}} \text{ 是 } S^{l_1\cdots l_n} \tag{5.38}$$

式中，$l_i = 1,2,\cdots,m_i, i = 1,2,\cdots,n$。

采用乘积推理机、单值模糊器和中心平均解模糊器来设计模糊控制器，即

$$u_{\mathrm{D}} = (\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{\sum_{l_1=1}^{m_1}\cdots\sum_{l_n=1}^{m_n} \bar{y}_u^{l_1\cdots l_n}\left(\prod_{i=1}^{n}\mu_{A_i}^{l_i}(x_i)\right)}{\sum_{l_1=1}^{m_1}\cdots\sum_{l_n=1}^{m_n}\left(\prod_{i=1}^{n}\mu_{A_i}^{l_i}(x_i)\right)} \tag{5.39}$$

令 $y_u^{l_1\cdots l_n}$ 是自由参数，放在集合 $\boldsymbol{\theta} \in R^{\prod_{i=1}^{n} m_i}$ 中，则模糊控制器为

$$u_{\mathrm{D}} = (\boldsymbol{x} \mid \boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) \tag{5.40}$$

式中，$\xi(x)$ 为 $\prod_{i=1}^{n} m_i$ 维向量，其第 $l_1 \cdots, l_n$ 个元素为

$$\xi_{l_1 \cdots l_n}(x) = \frac{\prod_{i=1}^{n} \mu_{A_i}^{l_i}(x_i)}{\sum_{l_1=1}^{m_1} \cdots \sum_{l_n=1}^{m_n} \left(\prod_{i=1}^{n} \mu_{A_i}^{l_i}(x_i) \right)} \tag{5.41}$$

模糊控制规则式(5.33)是通过设置其初始参数而被嵌入到模糊控制器中的。

5.3.3 自适应律的设计

将式(5.35)、式(5.37)代入式(5.31)，并整理得

$$e^{(n)} = -\boldsymbol{K}^{\mathrm{T}}\boldsymbol{e} + \boldsymbol{b}[u^* - u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta})] \tag{5.42}$$

令

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ -k_n & -k_{n-1} & \cdots & \cdots & \cdots & \cdots & -k_1 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ b \end{bmatrix} \tag{5.43}$$

则闭环系统动态方程式(5.42)可写成向量形式

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda}\boldsymbol{e} + \boldsymbol{b}[u^* - u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta})] \tag{5.44}$$

定义最优参数为

$$\theta^* = \arg \min_{\boldsymbol{\theta} \in R^{\prod_{i=1}^{n} m_i}} \left[\sup_{x \in R^n} \left| u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta}) - u^* \right| \right] \tag{5.45}$$

定义最小逼近误差为

$$\omega = u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta}^*) - u^* \tag{5.46}$$

由式(5.44)可得

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda}\boldsymbol{e} + \boldsymbol{b}(u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta}^*) - u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta})) - \boldsymbol{b}(u_{\mathrm{D}}(\boldsymbol{x} \mid \boldsymbol{\theta}^*) - u^*) \tag{5.47}$$

由式(5.40)，可将误差方程式(5.47)改写为

$$\dot{\boldsymbol{e}} = \boldsymbol{\Lambda}\boldsymbol{e} + \boldsymbol{b}(\boldsymbol{\theta}^* - \boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) - \boldsymbol{b}\omega \tag{5.48}$$

定义 Lyapunov 函数为

$$V = \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e} + \frac{\boldsymbol{b}}{2\gamma}(\boldsymbol{\theta}^* - \boldsymbol{\theta})^{\mathrm{T}}(\boldsymbol{\theta}^* - \boldsymbol{\theta}) \tag{5.49}$$

式中，参数 γ 是正的常数。

$\boldsymbol{P}$ 为一个正定矩阵且满足 Lyapunov 方程

$$\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{\Lambda} = -\boldsymbol{Q} \tag{5.50}$$

式中，$\boldsymbol{Q}$ 是一个任意的 $n\times n$ 正定矩阵，$\boldsymbol{\Lambda}$ 由式(5.43)给出。

取 $V_1=\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}$，$V_2=\frac{b}{2\gamma}(\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}(\boldsymbol{\theta}^*-\boldsymbol{\theta})$，令 $\boldsymbol{M}=b(\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})-b\omega$，则式(5.48)变为

$$\dot{\boldsymbol{e}}=\boldsymbol{\Lambda}\boldsymbol{e}+\boldsymbol{M}$$

$$\dot{V}_1=\frac{1}{2}\dot{\boldsymbol{e}}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}+\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\dot{\boldsymbol{e}}=\frac{1}{2}(\boldsymbol{e}^{\mathrm{T}}\boldsymbol{\Lambda}^{\mathrm{T}}+\boldsymbol{M}^{\mathrm{T}})\boldsymbol{P}\boldsymbol{e}+\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{\Lambda}\boldsymbol{e}+\boldsymbol{M})$$

$$=\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}(\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{P}+\boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{e})+\frac{1}{2}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}+\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}$$

$$=-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}+\frac{1}{2}(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}+\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M})=-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}+\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{M}$$

即

$$\dot{V}_1=-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}+\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}((\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})-\omega)$$

$$\dot{V}_2=-\frac{b}{\gamma}(\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}\dot{\boldsymbol{\theta}}$$

V 的导数为

$$\dot{V}=-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}+\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}[(\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})-\omega]-\frac{b}{\gamma}(\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}\dot{\boldsymbol{\theta}}\tag{5.51}$$

令 $\boldsymbol{p}_n$ 为 $\boldsymbol{P}$ 的最后一列，由 $\boldsymbol{b}=[0,\cdots,0,b]^{\mathrm{T}}$ 可知 $\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{b}=\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_n b$。则式(5.51)变为

$$\dot{V}=-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}+\frac{b}{\gamma}(\boldsymbol{\theta}^*-\boldsymbol{\theta})^{\mathrm{T}}[\gamma\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_n\boldsymbol{\xi}(\boldsymbol{x})-\dot{\boldsymbol{\theta}}]-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_n b\omega\tag{5.52}$$

取自适应律

$$\dot{\boldsymbol{\theta}}=\gamma\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_n\boldsymbol{\xi}(\boldsymbol{x})\tag{5.53}$$

则

$$\dot{V}=-\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{e}-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_n b\omega\tag{5.54}$$

由于 $\boldsymbol{Q}>0$，ω 是最小逼近误差，通过设计足够多规则的模糊系统 $u_{\mathrm{D}}(\boldsymbol{x}\mid\theta)$，可使 ω 充分小，并满足 $|\boldsymbol{e}^{\mathrm{T}}\boldsymbol{p}_n b\omega|<\frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Q}$，从而使得 $\dot{V}<0$。

直接自适应模糊控制系统的结构如图 5-15 所示。

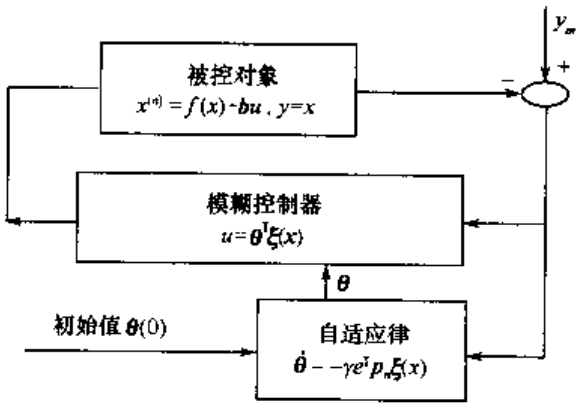


图 5-15　直接自适应模糊控制系统

5.3.4 仿真实例

被控对象为一个二阶系统

$$\ddot{x} = -25\dot{x} + 133u$$

位置指令为 $x_d(t) = \sin(\pi t)$。取以下 6 种隶属函数：$\mu_{N3}(x) = 1/(1+\exp(5(x+2)))$，$\mu_{N2}(x) = \exp(-(x+1.5)^2)$，$\mu_{N1}(x) = \exp(-(x+0.5)^2)$，$\mu_{P1}(x) = \exp(-(x-0.5)^2)$，$\mu_{P2}(x) = \exp(-(x-1.5)^2)$，$\mu_{P3}(x) = 1/(1+\exp(-5(x-2)))$。

系统摆初始状态为[1,0]，θ 中各元素的初始值取 0，采用控制律式(5.39)，自适应率取式(5.53)。取 $Q = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$，$k_1 = 1$，$k_2 = 10$，自适应参数取 $\gamma = 50$。

根据隶属函数设计程序，可得到隶属函数图，如图 5-16 所示。在控制系统仿真程序中，分别用 FS2，FS1 和 FS 表示模糊系统 $\xi(x)$ 的分子、分母及 $\xi(x)$，仿真结果如图 5-17 和图 5-18 所示。

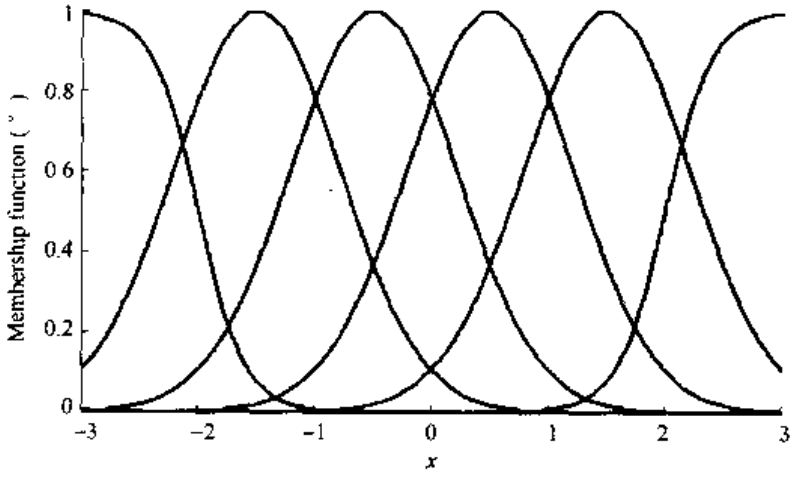


图 5-16 x 的隶属函数

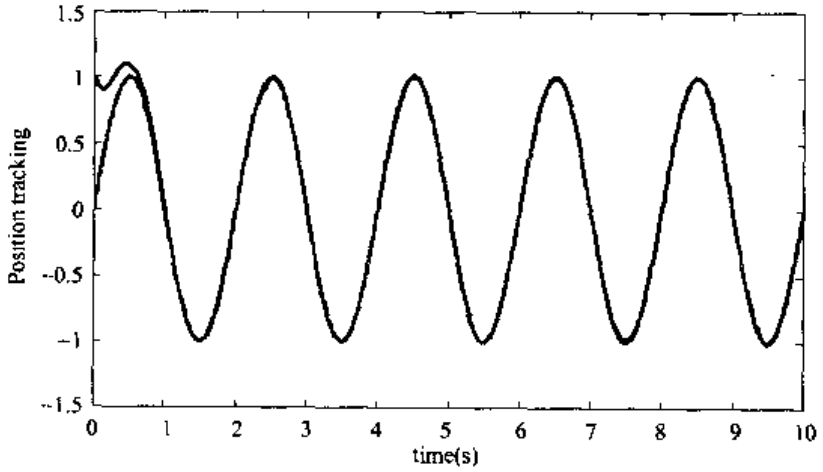


图 5-17 位置跟踪

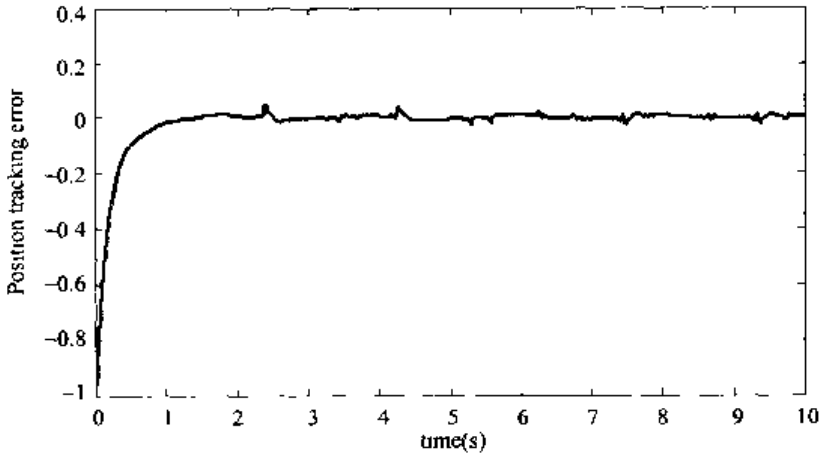


图 5-18 控制输入信号

直接自适应模糊控制程序有 5 个：① 隶属函数设计程序，chap5_4mf. m；②Simulink 主程序，chap5_4sim. mdl；③ 控制器 S 函数程序，chap5_4s. m；④ 被控对象 S 函数程序，chap5 4plant. m；⑤ 作图程序，chap5_4plot. m。程序见附录。

思考题与习题

5-1 设计一个在 $U=[-1,1]$ 上的模糊系统，使其以精度 $\varepsilon=0.1$ 一致地逼近函数 $g(x)=\sin(x\pi)+\cos(x\pi)+\sin(x\pi)\cos(x\pi)$，并进行 Matlab 仿真。

5-2 设计一个在 $U=[-1,1]\times[-1,1]$ 上的模糊系统，使其以精度 $\varepsilon=0.1$ 一致地逼近函数 $g(x)=\sin(x_1\pi)+\cos(x_2\pi)+\sin(x_1\pi)\cos(x_2\pi)$，并进行 Matlab 仿真。

5-3 设被控对象为 $m\ddot{x}=u$，其中 m 为未知。设计模糊自适应控制器 u，使得 $x(t)$ 收敛于参考信号 $y_m(t)$，其中，$\ddot{y}_m+2\dot{y}_m+y_m=r(t)$，并进行 Matlab 仿真。

附录（程序代码）

一维函数逼近仿真程序：chap5_1.m

```
% Fuzzy approaching
clear all;
close all;

Ll = -3;L2 = 3;
L = L2 - Ll;

h = 0.2;
N = L/h + 1;
T = 0.01;

x = Ll : T : L2;
for i = 1 : N
    e(i) = Ll + L/(N - 1) * (i - 1);
end

c = 0;d = 0;
for j = 1 : N
    if j = = 1
        u = trimf(x,[e(1),e(1),e(2)]);          % The first MF
    elseif j = = N
      u = trimf(x,[e(N - 1),e(N),e(N)]);        % The last MF
    else
        u = trimf(x,[e(j - 1),e(j),e(j + 1)]);
    end
    hold on;
    plot(x,u);
    c = c + sin(e(j)) * u;
    d = d + u;
end
xlabel('x');ylabel('Membership function');

for k = 1 : L/T + 1
    f(k) = c(k)/d(k);
end

y = sin(x);
```

```
figure(2);
plot(x,f,'b',x,y,'r');
xlabel('x');ylabel('Approaching');
figure(3);
plot(x,f - y,'r');
xlabel('x');ylabel('Approaching error');
```

二维函数逼近仿真程序:chap5_2.m

```
% Fuzzy approaching
clear all;
close all;

T = 0.1;
x1 = -1:T:1;
x2 = -1:T:1;

L = 2;
h = 0.2;
N = L/h + 1;

for i = 1:1:N                                          % N MF
  for j = 1:1:N
    e1(i) = -1 + L/(N-1)*(i-1);
    e2(j) = -1 + L/(N-1)*(j-1);
    gx(i,j) = 0.52 + 0.1*e1(i)^3 + 0.28*e2(j)^3 - 0.06*e1(i)*e2(j);
    end
end

df = zeros(L/T + 1,L/T + 1);
cf = zeros(L/T + 1,L/T + 1);
for m = 1:1:N                                          % ul change from 1 to N
    if m == 1
        ul = trimf(x1,[-1,-1,-1 + L/(N-1)]); % First ul
    elseif m == N
        ul = trimf(x1,[1 - L/(N-1),1,1]);          % Last ul
    else
    ul = trimf(x1,[e1(m-1),e1(m),e1(m+1)]);
    end
figure(1);
hold on;
```

```
plot(x1,u1);
xlabel('x1');ylabel('Membership function');

for n = 1 : 1 : N                                  %u2 change from 1 to N
    if n == 1
        u2 = trimf(x2,[-1, -1, -1 + L/(N - 1)]);%First u2
    elseif n == N
        u2 = trimf(x2,[1 - L/(N - 1),1,1]);          %Last u2
    else
        u2 = trimf(x2,[e2(n - 1),e2(n),e2(n + 1)]);
    end
figure(2);
hold on;
plot(x2,u2);
xlabel('x2');ylabel('Membership function');

    for i = 1 : 1 : L/T + 1
      for j = 1 : 1 : L/T + 1
        d = df(i,j) + u1(i) * u2(j);
        df(i,j) = d;
        c = cf(i,j) + gx(m,n) * u1(i) * u2(j);
        cf(i,j) = c;
      end
    end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1 : 1 : L/T + 1
    for j = 1 : 1 : L/T + 1
        f(i,j) = cf(i,j)/df(i,j);
        y(i,j) = 0.52 + 0.1 * x1(i)^3 + 0.28 * x2(j)^3 - 0.06 * x1(i) * x2(j);
    end
end
figure(3);
subplot(211);
surf(x1,x2,f);
title('f(x)');
subplot(212);
surf(x1,x2,y);
```

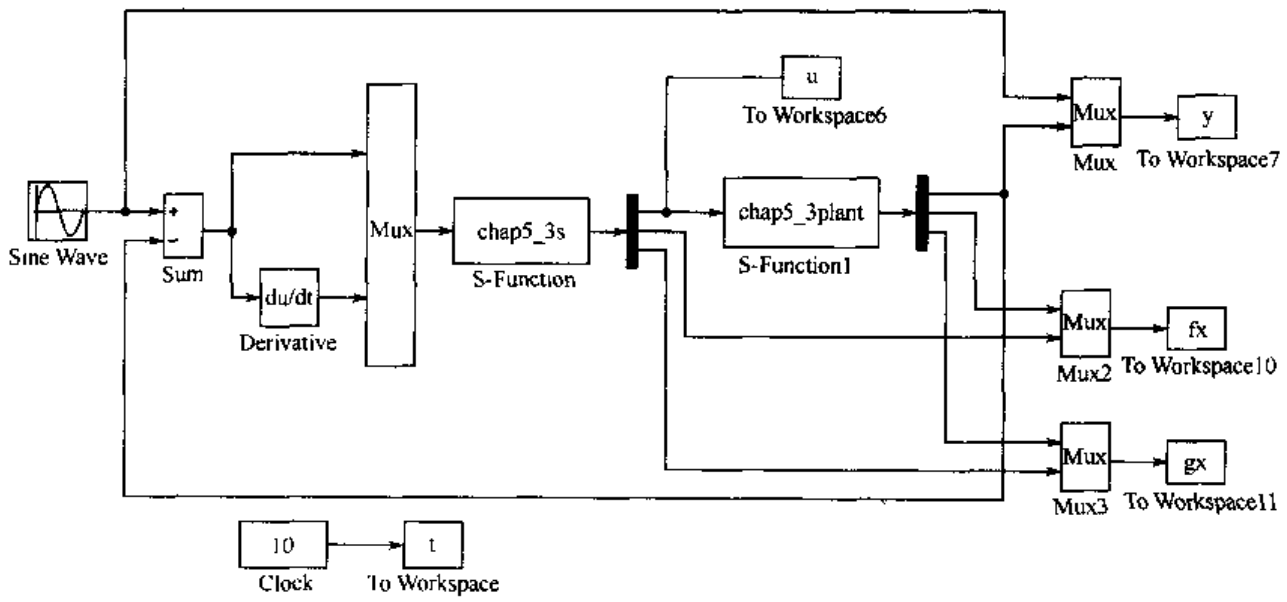
```
title('g(x)');
figure(4);
surf(x1,x2,f - y);
title('Approaching error');
```

间接模糊自适应控制仿真程序

(1) **隶属函数设计程序**:chap5_3mf.m

```
clear all;
close all;

L1 = - pi/6;
L2 = pi/6;
L = L2 - L1;

T = L * 1/1000;

x = L1 : T : L2;
figure(1);
for i = 1 : 1 : 5
    gs = - [(x + pi/6 - (i - 1) * pi/12)/(pi/24)]^2;
    u = exp(gs);
    hold on;
    plot(x,u);
end
xlabel('x');ylabel('Membership function degree');
```

(2) **Simulink 主程序**:chap5_3sim.mdl



(3) **控制器 S 函数**:chap5_3s.m

```
function[sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function[sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates     = 50;
sizes.NumDiscStates     = 0;
sizes.NumOutputs        = 3;
sizes.NumInputs         = 2;
sizes.DirFeedthrough    = 1;
sizes.NumSampleTimes    = 0;
sys = simsizes(sizes);
x0   = [0.1*ones(50,1)];
str  = [];
ts   = [];

function sys = mdlDerivatives(t,x,u)
gama1 = 50;
gama2 = 1;

r = 0.1*sin(t);
dr = 0.1*cos(t);
ddr = -0.1*sin(t);

e = u(1);
de = u(2);
xl = r - e;
```

```
x2 = de - dr;

k1 = 2;
k2 = 1;
k = [k2;k1];
E = [e,de]';

for i = 1 : 1 : 25
    thtaf(i,1) = x(i);
end
for i = 1 : 1 : 25
    thtag(i,1) = x(i + 25);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A = [0   - k2;
     1   - k1];
Q = [10 0;0 10];
P = lyap(A,Q);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FS1 = 0;
for l1 = 1 : 1 : 5
    gs1 = - [(x1 + pi/6 - (l1 - 1) * pi/12)/(pi/24)]^2;
    u1(l1) = exp(gs1);
end

for l2 = 1 : 1 : 5
    gs2 = - [(x2 + pi/6 - (l2 - 1) * pi/12)/(pi/24)]^2;
    u2(l2) = exp(gs2);
end

for l1 = 1 : 1 : 5
    for l2 = 1 : 1 : 5
      FS2(5 * (l1 - 1) + l2) = u1(l1) * u2(l2);
      FS1 = FS1 + u1(l1) * u2(l2);
    end
end

FS = FS2/(FS1 + 0.001);

fx1 = thtaf' * FS';
```

```
gxl = thtag' * FS' + 0.001;

ut = 1/gxl * (- fxl + ddr + k' * E);

b = [0;1];
S1 = - gama1 * E' * P * b * FS;
S2 = - gama2 * E' * P * b * FS * ut;

for i = 1 : 1 : 25
    sys(i) = S1(i);
end
for j = 26 : 1 : 50
    sys(j) = S2(j - 25);
end

function sys = mdlOutputs(t,x,u)
r = 0.1 * sin(t);
dr = 0.1 * cos(t);
ddr = - 0.1 * sin(t);

e = u(1);
de = u(2);
x1 = r - e;
x2 = de - dr;

k1 = 2;
k2 = 1;
k = [k2;k1];
E = [e,de]';

for i = 1 : 1 : 25
    thtaf(i,1) = x(i);
end
for i = 1 : 1 : 25
    thtag(i,1) = x(i + 25);
end

FS1 = 0;
for l1 = 1 : 1 : 5
    gs1 = - [(x1 + pi/6 - (l1 - 1) * pi/12)/(pi/24)]^2;
```

```
    u1(l1) = exp(gs1);
end

for l2 = 1:1:5
    gs2 = -[(x2 + pi/6 - (l2 - 1) * pi/12)/(pi/24)]^2;
    u2(l2) = exp(gs2);
end

for l1 = 1:1:5
    for l2 = 1:1:5
      FS2(5 * (l1 - 1) + l2) = u1(l1) * u2(l2);
      FS1 = FS1 + u1(l1) * u2(l2);
    end
end
FS = FS2/(FS1 + 0.001);

fx1 = thtaf' * FS';
gx1 = thtag' * FS' + 0.001;

ut = 1/gx1 * (- fx1 + ddr + k' * E);
sys(1) = ut;
sys(2) = fx1;
sys(3) = gx1;
```

(4) **被控对象S函数**:chap5_3plant.m

```
%S - function for continuous state equation
function[sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
```

```
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function[sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates     = 2;
sizes.NumDiscStates     = 0;
sizes.NumOutputs        = 3;
sizes.NumInputs         = 1;
sizes.DirFeedthrough    = 1;
sizes.NumSampleTimes    = 0;

sys = simsizes(sizes);
x0 = [pi/60 0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l*(4/3 - m*(cos(x(1)))^2/(mc + m));
fx = g*sin(x(1)) - m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc + m);
fx = fx/S;
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(2);
sys(2) = fx + gx*u;
function sys = mdlOutputs(t,x,u)
g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l*(4/3 - m*(cos(x(1)))^2/(mc + m));
```

```
fx = g * sin(x(1)) - m * l * x(2)^2 * cos(x(1)) * sin(x(1))/(mc + m);
fx = fx/S;
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(1);
sys(2) = fx;
sys(3) = gx;
```

(5) **作图程序**:chap5_3plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,fx(:,1),'r',t,fx(:,2),'b');
xlabel('time(s)');ylabel('fx and estiamted fx');
figure(4);
plot(t,gx(:,1),'r',t,gx(:,2),'b');
xlabel('time(s)');ylabel('gx and estimated gx');
```

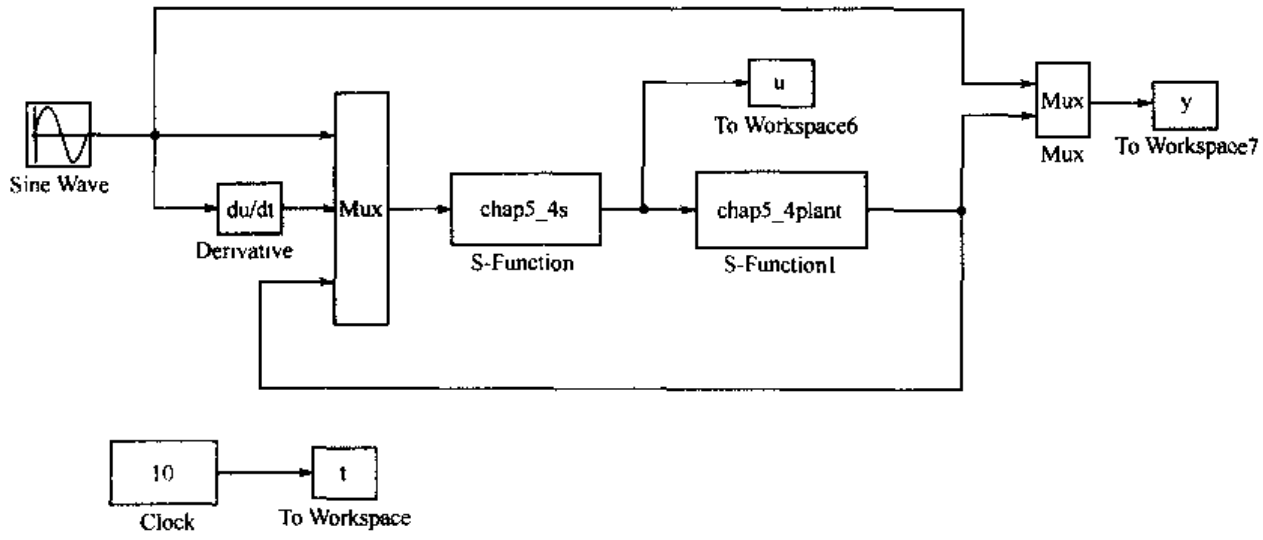
直接自适应模糊控制程序

(1) **隶属函数设计程序**:chap5_4mf.m

```
clear all;
close all;

L1 = -3;
L2 = 3;
L = L2 - L1;

T = 0.001;

x = L1 : T : L2;
```

```
figure(1);
for i = 1 : 1 : 6
    if i = = 1
      u = 1./(1 + exp(5 * (x + 2)));
    elseif i = = 6
      u = 1./(1 + exp(- 5 * (x - 2)));
    else
    u = exp(- (x + 2.5 - (i - 1)).^2);
end
    hold on;
    plot(x,u);
end
xlabel('x');ylabel('Membership function degree');
```

(2) **Simulink 主程序**:chap5_4sim. mdl



(3) **控制器 S 函数**:chap5_4s. m

```
function[sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
```

```
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function[sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 36;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 4;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;
sys = simsizes(sizes);
x0   = [zeros(36,1)];
str  = [];
ts   = [];
function sys = mdlDerivatives(t,x,u)

r = u(1);
dr = u(2);
x(1) = u(3);
x(2) = u(4);

e = r - x(1);
de = dr - x(2);

gama = 50;

k2 = 1;
k1 = 10;
E = [e,de]';
A = [0 -k2;
    1 -k1];
Q = [50 0;0 50];
P = lyap(A,Q);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FS1 = 0;

u1(1) = 1/(1 + exp(5*(x(1) + 2)));
u1(6) = 1/(1 + exp(-5*(x(1) - 2)));
```

```
for i = 2 : 1 : 5
    u1(i) = exp(- (x(1) + 1.5 - (i - 2))^2);
end

u2(1) = 1/(1 + exp(5 * (x(2) + 2)));
u2(6) = 1/(1 + exp(- 5 * (x(2) - 2)));
for i = 2 : 1 : 5
    u2(i) = exp(- (x(2) + 1.5 - (i - 2))^2);
end

for i = 1 : 1 : 6
    for j = 1 : 1 : 6
      FS2(6 * (i - 1) + j) = u1(i) * u2(j);
      FS1 = FS1 + u1(i) * u2(j);
    end
end

FS = FS2/FS1;
b = [0;1];
S = gama * E' * P(:,2) * FS;
for i = 1 : 1 : 36
    sys(i) = S(i);
end

function sys = mdlOutputs(t,x,u)

r = u(1);
dr = u(2);
x(1) = u(3);
x(2) = u(4);

for i = 1 : 1 : 36
    thtau(i,1) = x(i);
end

FS1 = 0;
u1(1) = 1/(1 + exp(5 * (x(1) + 2)));
u1(6) = 1/(1 + exp(- 5 * (x(1) - 2)));
for i = 2 : 1 : 5
    u1(i) = exp(- (x(1) + 1.5 - (i - 2))^2);
```

```
end

u2(1) = 1/(1 + exp(5 * (x(2) + 2)));
u2(6) = 1/(1 + exp(-5 * (x(2) - 2)));
for i = 2 : 1 : 5
    u2(i) = exp(- (x(2) + 1.5 - (i - 2))^2);
end

for i = 1 : 1 : 6
    for j = 1 : 1 : 6
      FS2(6 * (i - 1) + j) = u1(i) * u2(j);
      FS1 = FS1 + u1(i) * u2(j);
    end
end
FS = FS2/FS1;

ut = thtau' * FS';
sys(1) = ut;
```

(4) **被控对象 S 函数**:chap5_4plant.m

```
% S - function for continuous state equation
function[sys,x0,str,ts] = s_function(t,x,u,flag)
switch flag,
% Initialization
    case 0,
      [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
      sys = mdlDerivatives(t,x,u);
% Outputs
    case 3,
      sys = mdlOutputs(t,x,u);
% Unhandled flags
    case {2, 4, 9 }
      sys = [];
% Unexpected flags
    otherwise
      error(['Unhandled flag = ',num2str(flag)]);
end

function[sys,x0,str,ts] = mdlInitializeSizes
```

```
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [1 0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

sys(1) = x(2);
sys(2) = -25*x(2) + 133*u;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(5) **作图程序**:chap5_4plot.m

```
close all;
figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,y(:,1) - y(:,2),'r');
xlabel('time(s)');ylabel('Position tracking error');

figure(3);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

第6章　神经网络的理论基础

模糊控制从人的经验出发，解决了智能控制中人类语言的描述和推理问题，尤其是一些不确定性语言的描述和推理问题，从而在机器模拟人脑的感知、推理等智能行为方面迈出了重大的一步。然而，模糊控制在处理数值数据、自学习能力等方面还远没有达到人脑的境界。人工神经网络从另一个角度出发，即从人脑的生理学和心理学着手，通过人工模拟人脑的工作机理来实现机器的部分智能行为。

人工神经网络（简称神经网络，Neural Network）是模拟人脑思维方式的数学模型。神经网络是在现代生物学研究人脑组织成果的基础上提出的，用来模拟人类大脑神经网络的结构和行为，它从微观结构和功能上对人脑进行抽象和简化，是模拟人类智能的一条重要途径，反映了人脑功能的若干基本特征，如并行信息处理、学习、联想、模式分类、记忆等。

20世纪80年代以来，人工神经网络（Artificial Neural Network，ANN）的研究取得了突破性进展。神经网络控制是将神经网络与控制理论相结合而发展起来的智能控制方法。它已成为智能控制的一个新的分支，为解决复杂的非线性、不确定、不确知系统的控制问题开辟了新途径。

6.1　神经网络发展简史

神经网络的发展历程经历了4个阶段。

1. 启蒙期（1890—1969年）

1890年，W. James发表专著《心理学》，讨论了脑的结构和功能。1943年，心理学家W. S. McCulloch和数学家W. Pitts提出了描述脑神经细胞动作的数学模型，即M-P模型（第一个神经网络模型）。1949年，心理学家Hebb实现了对脑细胞之间相互影响的数学描述，从心理学的角度提出了至今仍对神经网络理论有着重要影响的Hebb学习法则。1958年，E. Rosenblatt提出了描述信息在人脑中存储和记忆的数学模型，即著名的感知机模型（Perceptron）。1962年，Widrow和Hoff提出了自适应线性神经网络，即Adaline网络，并提出了网络学习新知识的方法，即Widrow和Hoff学习规则（δ学习规则），并用电路进行了硬件设计。

2. 低潮期（1969—1982年）

受当时神经网络理论研究水平的限制，加之受到冯·诺依曼式计算机发展的冲击等因素的影响，神经网络的研究陷入低谷。但在美、日等国仍有少数学者继续着网络模型和学习算法的研究，提出了许多有意义的理论和方法。例如，1969年，Grossberg提出了至今为止最复杂的ART神经网络。1972年，Kohonen提出了自组织映射的SOM模型。

3. 复兴期(1982—1986 年)

1982 年,物理学家 Hoppield 提出了 Hoppield 神经网络模型,该模型通过引入能量函数,实现了问题优化求解,1984 年他用此模型成功地解决了旅行商路径优化问题(TSP)。这一成果的取得使神经网络的研究取得了突破性进展。

1986 年,在 Rumelhart 和 McCelland 等出版的《Parallel Distributed Processing》一书中,提出了一种著名的多层神经网络模型,即 BP 网络,该网络是迄今为止应用最普遍的神经网络。

4. 新连接机制时期(1986 年至现在)

神经网络从理论走向应用领域,出现了神经网络芯片和神经计算机。神经网络逐渐在模式识别与图像处理(语音、指纹、故障检测和图像压缩等)、控制与优化、预测与管理(市场预测、风险分析)、通信等领域得到成功的应用。

6.2 神经网络原理

神经生理学和神经解剖学的研究表明,人脑极其复杂,由一千多亿个神经元交织在一起的网状结构构成,其中大脑皮层约 140 亿个神经元,小脑皮层约 1000 亿个神经元。

人脑能完成智能、思维等高级活动,为了能利用数学模型来模拟人脑的活动,导致了神经网络的研究。

单个神经元的解剖图如图 6-1 所示,神经系统的基本构造是神经元(神经细胞),它是处理人体内各部分之间信息传递的基本单元。每个神经元都由一个细胞体、一个连接其他神经元的轴突和一些向外伸出的其他较短分支——树突组成。轴突的功能是将本神经元的输出信号(兴奋)传递给别的神经元,其末端的许多神经末梢使得兴奋可以同时传送给多个神经元。树突的功能是接收来自其他神经元的兴奋。神经元细胞体将接收到的所有信号进行简单的处理后,由轴突输出。神经元的轴突与另外神经元神经末梢相连的部分称为突触。

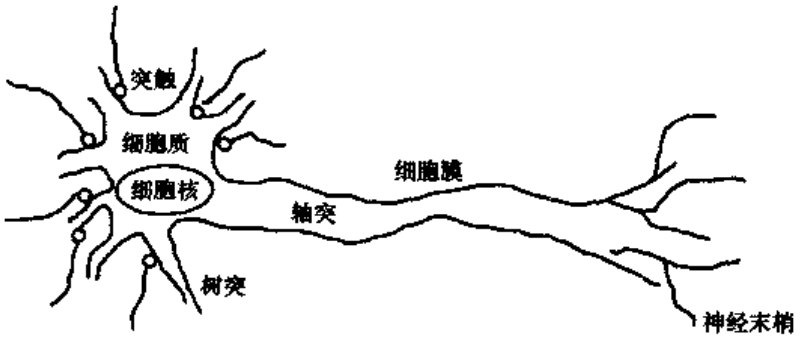


图 6-1 单个神经元的解剖图

神经元由 4 部分构成:

(1) 细胞体(主体部分):包括细胞质、细胞膜和细胞核;

(2) 树突:用于为细胞体传入信息;

(3) 轴突:为细胞体传出信息,其末端是神经末梢,含传递信息的化学物质;

(4) 突触：是神经元之间的接口($10^4 \sim 10^5$ 个每神经元)。一个神经元通过其轴突的神经末梢，经突触与另外一个神经元的树突连接，以实现信息的传递。由于突触的信息传递特性是可变的，随着神经冲动传递方式的变化，传递作用强弱不同，形成了神经元之间连接的柔性，称为结构的可塑性。

神经元具有如下功能：

(1) 兴奋与抑制：如果传入神经元的冲动经整合后使细胞膜电位升高，超过动作电位的阈值时即为兴奋状态，产生神经冲动，由轴突经神经末梢传出。如果传入神经元的冲动经整合后使细胞膜电位降低，低于动作电位的阈值时即为抑制状态，不产生神经冲动。

(2) 学习与遗忘：由于神经元结构的可塑性，突触的传递作用可增强和减弱，因此，神经元具有学习与遗忘的功能。

决定神经网络模型性能的 3 大要素为：

(1) 神经元(信息处理单元)的特性；

(2) 神经元之间相互连接的形式——拓扑结构；

(3) 为适应环境而改善性能的学习规则。

神经网络的研究主要分为 3 个方面的内容，即神经元模型、神经网络结构和神经网络学习算法。

6.3　神经网络的分类

人工神经网络是以数学手段来模拟人脑神经网络的结构和特征的系统。利用人工神经元可以构成各种不同拓扑结构的神经网络，从而实现对生物神经网络的模拟和近似。

目前神经网络模型的种类相当丰富，已有近 40 余种神经网络模型，其中典型的有多层前向传播网络(BP 网络)、Hopfield 网络、CMAC 小脑模型、ART 自适应共振理论、BAM 双向联想记忆、SOM 自组织网络、Blotzman 机网络和 Madaline 网络等。

根据神经网络的连接方式，神经网络可分为 3 种形式。

1. 前向网络

如图 6-2 所示，神经元分层排列，组成输入层、隐含层和输出层。每一层的神经元只接受前一层神经元的输入。输入模式经过各层的顺次变换后，由输出层输出。在各神经元之间不存在反馈。感知器和误差反向传播网络采用前向网络形式。

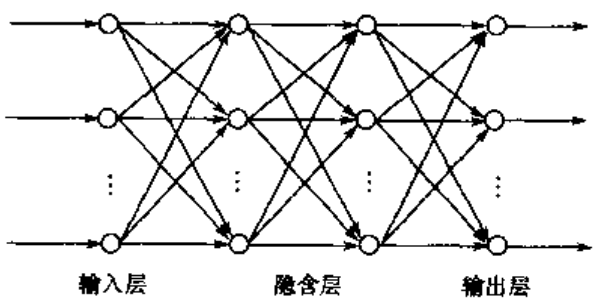


图 6-2　前向型神经网络

2. 反馈网络

网络结构如图 6-3 所示，该网络结构在输出层到输入层存在反馈，即每一个输入节点都有可能接受来自外部的输入和来自输出神经元的反馈。这种神经网络是一种反馈动力学系统，它需要工作一段时间才能达到稳定。Hopfield 神经网络是反馈网络中最简单且应用最广泛的模型，它具有联想记忆的功能，如果将 Lyapunov 函数定义为寻优函数，Hopfield 神经网络还可以解决寻优问题。

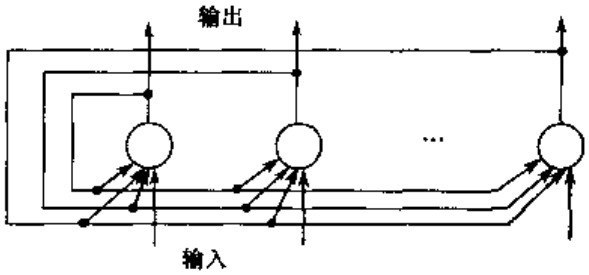


图 6-3　反馈型神经网络

3. 自组织网络

网络结构如图 6-4 所示。Kohonen 网络是最典型的自组织网络。Kohonen 认为，当神经网络在接受外界输入时，网络将会分成不同的区域，不同区域具有不同的响应特征，即不同的神经元以最佳方式响应不同性质的信号激励，从而形成一种拓扑意义上的特征图，该图实际上是一种非线性映射。这种映射是通过无监督的自适应过程完成的，所以也称为自组织特征图。

Kohonen 网络通过无导师的学习方式进行权值的学习，稳定后的网络输出就对输入模式生成自然的特征映射，从而达到自动聚类的目的。

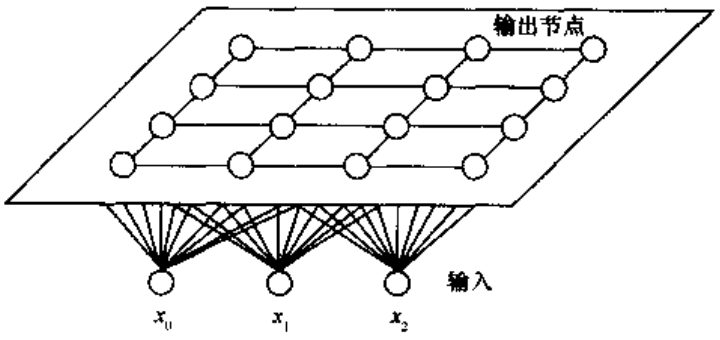


图 6-4　自组织神经网络

6.4　神经网络学习算法

神经网络学习算法是神经网络智能特性的重要标志，神经网络通过学习算法，实现了自适应、自组织和自学习的能力。

目前神经网络的学习算法有多种，按有无导师分类，可分为有导师学习(Supervised Learning)、无导师学习(Unsupervised Learning)和再励学习(Reinforcement Learning)等几大类。在有导师的学习方式中，网络的输出和期望的输出(即导师信号)进行比较，然后根据两

者之间的差异调整网络的权值，最终使差异变小，如图 6-5 所示。在无导师的学习方式中，输入模式进入网络后，网络按照一种预先设定的规则（如竞争规则）自动调整权值，使网络最终具有模式分类等功能，如图 6-6 所示。再励学习是介于上述两者之间的一种学习方式。

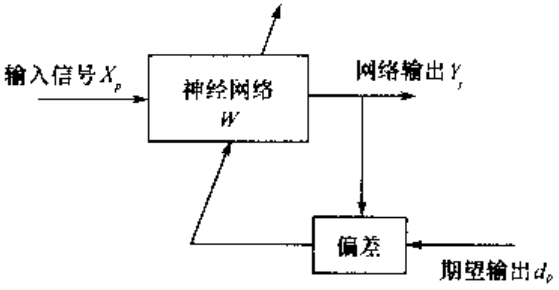


图 6-5　有导师指导的神经网络学习

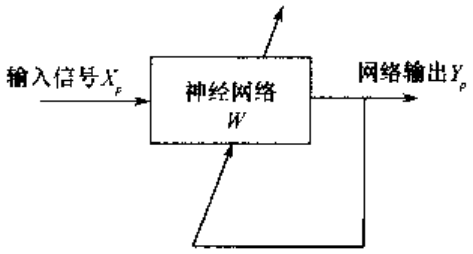


图 6-6　无导师指导的神经网络学习

下面介绍两个基本的神经网络学习算法。

6.4.1　Hebb 学习规则

Hebb 学习规则是一种联想式学习算法。生物学家 D. O. Hebbian 基于对生物学和心理学的研究，认为两个神经元同时处于激发状态时，它们之间的连接强度将得到加强，这一论述的数学描述被称为 Hebb 学习规则，即

$$w_{ij}(k+1) = w_{ij}(k) + I_i I_j \tag{6.1}$$

式中，$w_{ij}(k)$ 为连接从神经元 i 到神经元 j 的当前权值；I_i 和 I_j 分别为神经元 i 和 j 的激活水平。

Hebb 学习规则是一种无导师的学习方法，它只根据神经元连接间的激活水平改变权值，因此，这种方法又称为相关学习或并联学习。

6.4.2　Delta(δ) 学习规则

假设误差准则函数为

$$E = \frac{1}{2}\sum_{p=1}^{P}(d_p - y_p)^2 = \sum_{p=1}^{P} E_p \tag{6.2}$$

式中，d_p 代表期望的输出（导师信号）；y_p 为网络的实际输出，$y_p = f(\boldsymbol{W}\boldsymbol{X}_p)$，$\boldsymbol{W}$ 为网络所有权值组成的向量，即

$$\boldsymbol{W} = (w_0, w_1, \cdots, w_n)^{\mathrm{T}} \tag{6.3}$$

$\boldsymbol{X}_p$ 为输入模式，即

$$\boldsymbol{X}_p = (x_{p0}, x_{p1}, \cdots, x_{pn})^{\mathrm{T}} \tag{6.4}$$

式中，训练样本数为 $p = 1, 2, \cdots, P$。

神经网络学习的目的是通过调整权值 $\boldsymbol{W}$，使误差准则函数最小。可采用梯度下降法来实现权值的调整，其基本思想是沿着 E 的负梯度方向不断修正 $\boldsymbol{W}$ 值，直到 E 达到最小，这种方法的数学表达式为

$$\nabla \boldsymbol{W} = \eta\left(-\frac{\partial E}{\partial W_i}\right) \tag{6.5}$$

$$\frac{\partial E}{\partial W_i} = \sum_{p=1}^{P}\frac{\partial E_p}{\partial W_i} \tag{6.6}$$

其中

$$E_p = \frac{1}{2}(d_p - y_p)^2 \tag{6.7}$$

令 $\theta_p = Wx_p$，则

$$\frac{\partial E_p}{\partial W_i} = \frac{\partial E_p}{\partial \theta_p}\frac{\partial \theta_p}{\partial W_i} = \frac{\partial E_p}{\partial y_p}\frac{\partial y_p}{\partial \theta_p}X_{ip} = -(d_p - y_p)f'(\theta_p)X_{ip} \tag{6.8}$$

W 的修正规则为

$$\Delta w_i = \eta\sum_{p=1}^{P}(d_p - y_p)f'(\theta_p)X_{ip} \tag{6.9}$$

上式称为 δ 学习规则，又称误差修正规则。

6.5 神经网络的特征及要素

1. 神经网络特征

神经网络具有以下几个特征：

(1) 能逼近任意非线性函数；

(2) 信息的并行分布式处理与存储；

(3) 可以多输入、多输出；

(4) 便于用超大规模集成电路(VISI) 或光学集成电路系统实现，或用现有的计算机技术实现；

(5) 能进行学习，以适应环境的变化。

2. 神经网络三要素

神经网络具有以下 3 个要素：

(1) 神经元(信息处理单元) 的特性；

(2) 神经元之间相互连接的拓扑结构；

(3) 为适应环境而改善性能的学习规则。

6.6 神经网络控制的研究领域

1. 基于神经网络的系统辨识

(1) 将神经网络作为被辨识系统的模型，可在已知常规模型结构的情况下，估计模型的参数。

(2) 利用神经网络的线性、非线性特性，可建立线性、非线性系统的静态、动态、逆动态及预测模型，实现系统的建模和辨识。

2. 神经网络控制器

神经网络作为控制器，可对不确定、不确知系统及扰动进行有效的控制，使控制系统达到所要求的动态、静态特性。

3. 神经网络与其他算法相结合

将神经网络与专家系统、模糊逻辑、遗传算法等相结合，可设计新型智能控制系统。

4. 优化计算

在常规的控制系统中，常遇到求解约束优化问题，神经网络为这类问题的解决提供了有效的途径。

目前，神经网络控制已经在多种控制结构中得到应用，如PID控制、模型参考自适应控制、前馈反馈控制、内模控制、预测控制、模糊控制等。

思考题与习题

6-1 神经网络的发展分为哪几个阶段？每个阶段都有哪些特点？

6-2 神经网络按连接方式分有哪几类？每一类有哪些特点？

6-3 分别描述 Hebb 学习规则和 Delta 学习规则。

第 7 章　典型神经网络

7.1　单神经元网络

图 7-1 中 u_i 为神经元的内部状态，θ_i 为阈值，x_j 为输入信号，$j=1,\cdots,n$，w_{ij} 表示从单元 u_j 到单元 u_i 的连接权系数，s_i 为外部输入信号。图 7-1 所示的模型可描述为

$$\mathrm{net}_i = \sum_j w_{ij}x_j + s_j - \theta_i \tag{7.1}$$

$$u_i = f(\mathrm{net}_i) \tag{7.2}$$

$$y_i = g(u_i) = h(\mathrm{net}_i) \tag{7.3}$$

通常情况下，取 $g(u_i) = u_i$，即 $y_i = f(\mathrm{net}_i)$。

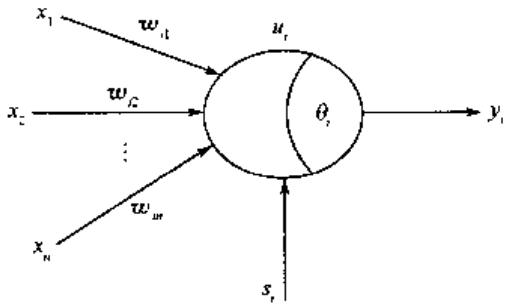


图 7-1　神经元结构模型

常用的神经元非线性特性有以下 3 种。

1. 阈值型

$$f(\mathrm{net}_i) = \begin{cases} 1 & \mathrm{net}_i > 0 \\ 0 & \mathrm{net}_i \leqslant 0 \end{cases} \tag{7.4}$$

阈值型函数如图 7-2 所示。

2. 分段线性型

分段线性型函数表达式为

$$f(\mathrm{net}_i) = \begin{cases} 0 & \mathrm{net}_i \leqslant \mathrm{ner}_{i0} \\ k\mathrm{net}_i & \mathrm{net}_{i0} < \mathrm{net}_i < \mathrm{net}_{i1} \\ f_{\max} & \mathrm{net}_i \geqslant \mathrm{net}_{i1} \end{cases} \tag{7.5}$$

分段线性型函数如图 7-3 所示。

图 7-2　阈值型函数

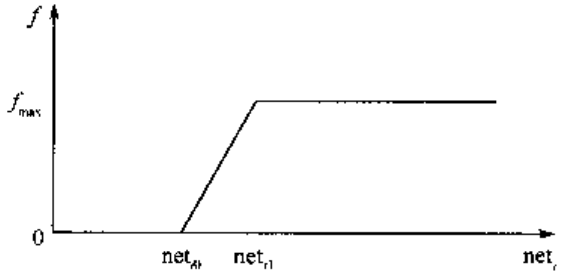


图 7-3　分段线性型函数

3. 函数型

有代表性的有 Sigmoid 型和高斯型函数。Sigmoid 型函数表达式为

$$f(\text{net}_i) = \frac{1}{1+\mathrm{e}^{-\frac{\text{net}_i}{T}}} \tag{7.6}$$

Sigmoid 型函数如图 7-4 所示。

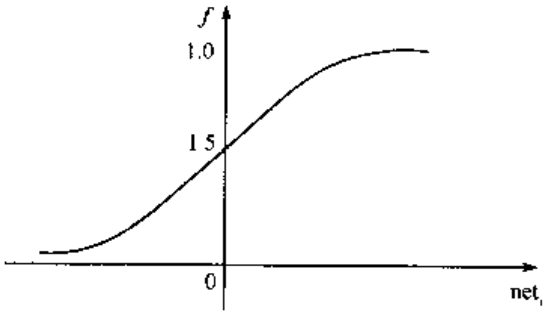


图 7-4　Sigmoid 型函数

7.2　BP 神经网络

1986 年，Rumelhart 等提出了误差反向传播神经网络，简称 BP 网络(Back Propagation)，该网络是一种单向传播的多层前向网络。

误差反向传播的学习算法简称BP算法，其基本思想是梯度下降法。它采用梯度搜索技术，以期使网络的实际输出值与期望输出值的误差均方值为最小。

7.2.1　BP 网络特点

BP 网络具有以下几个特点：

(1) BP 网络是一种多层网络，包括输入层、隐层和输出层；

(2) 层与层之间采用全互连方式，同一层神经元之间不连接；

(3) 权值通过 δ 学习算法进行调节；

(4) 神经元激发函数为 S 函数；

(5) 学习算法由正向传播和反向传播组成；

(6) 层与层的连接是单向的，信息的传播是双向的。

7.2.2 BP 网络结构

含一个隐层的BP网络结构如图7-5所示，图中，i 为输入层神经元，j 为隐层神经元，k 为输出层神经元。

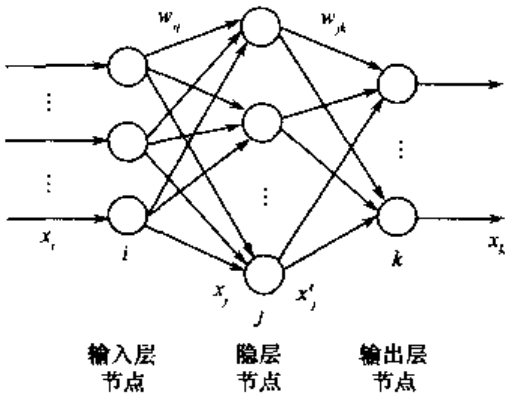


图 7-5 BP 神经网络结构

7.2.3 BP 网络的逼近

BP 网络逼近的结构如图 7-6 所示，图中 k 为网络的迭代步骤，$u(k)$ 和 $y(k)$ 为逼近器的输入。BP 为网络逼近器，$y(k)$ 为被控对象的实际输出，$y_n(k)$ 为 BP 网络的输出。将系统输出 $y(k)$ 及输入 $u(k)$ 的值作为逼近器 BP 的输入，将系统输出与网络输出的误差作为逼近器的调整信号。

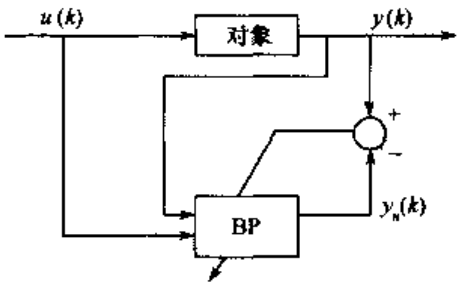


图 7-6 BP 神经网络逼近

用于逼近的 BP 网络如图 7-7 所示。

BP 算法的学习过程由正向传播和反向传播组成。在正向传播过程中，输入信息从输入层经隐层逐层处理，并传向输出层，每层神经元（节点）的状态只影响下一层神经元的状态。如果在输出层不能得到期望的输出，则转至反向传播，将误差信号（理想输出与实际输出之差）按连接通路反向计算，由梯度下降法调整各层神经元的权值，使误差信号减小。

(1) 前向传播：计算网络的输出。

隐层神经元的输入为所有输入的加权之和，即

$$x_j = \sum_i w_{ij} x_i \tag{7.7}$$

隐层神经元的输出 x_j' 采用 S 函数激发 x_j，得

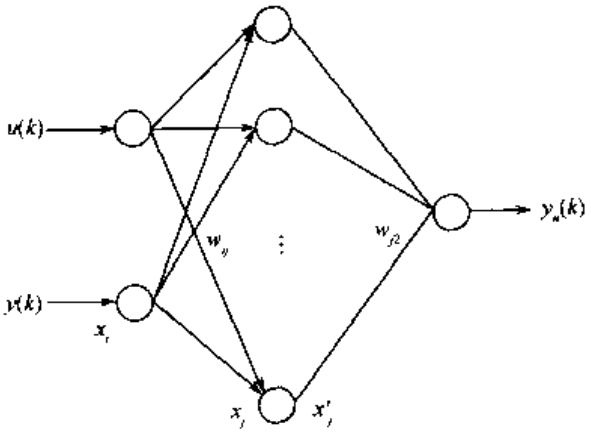


图 7-7　用于逼近的 BP 网络

$$x'_j = f(x_j) = \frac{1}{1+e^{-x_j}} \tag{7.8}$$

则

$$\frac{\partial x'_j}{\partial x_j} = x'_j(1-x'_j)$$

输出层神经元的输出为

$$y_n(k) = \sum_j w_{j2}x'_j \tag{7.9}$$

网络输出与理想输出误差为

$$e(k) = y(k) - y_n(k)$$

误差性能指标函数为

$$E = \frac{1}{2}e(k)^2 \tag{7.10}$$

(2) 反向传播:采用 δ 学习算法,调整各层间的权值。

根据梯度下降法,权值的学习算法如下:

输出层及隐层的连接权值 w_{j2} 学习算法为

$$\Delta w_{j2} = -\eta\frac{\partial E}{\partial w_{j2}} = \eta \cdot e(k) \cdot \frac{\partial y_n}{\partial w_{j2}} = \eta \cdot e(k) \cdot x'_j$$

式中,η 为学习速率,$\eta \in [0,1]$。

$k+1$ 时刻网络的权值为

$$w_{j2}(k+1) = w_{j2}(k) + \Delta w_{j2}$$

隐层及输入层连接权值 w_{ij} 学习算法为

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = \eta \cdot e(k) \cdot \frac{\partial y_n}{\partial w_{ij}}$$

式中,$\frac{\partial y_n}{\partial w_{ij}} = \frac{\partial y_n}{\partial x'_j} \cdot \frac{\partial x'_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = w_{j2} \cdot \frac{\partial x'_j}{\partial x_j} \cdot x_i = w_{j2} \cdot x'_j(1-x'_j) \cdot x_i$。

$k+1$ 时刻网络的权值为

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}$$

为了避免权值的学习过程发生振荡、收敛速度慢，需要考虑上次权值变化对本次权值变化的影响，即加入动量因子 α。此时的权值为

$$w_{j2}(k+1) = w_{j2}(k) + \Delta w_{j2} + \alpha(w_{j2}(k) - w_{j2}(k-1)) \tag{7.11}$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} + \alpha(w_{ij}(k) - w_{ij}(k-1)) \tag{7.12}$$

式中，α 为动量因子，$\alpha \in [0,1]$。

将对象输出对输入的敏感度 $\frac{\partial y(k)}{\partial u(k)}$ 称为 Jacobian 信息，其值可由神经网络辨识而得。辨识算法如下：取 BP 网络的第一个输入为 $u(k)$，即 $x_1 = u(k)$，则

$$\frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_n(k)}{\partial u(k)} = \frac{\partial y_n(k)}{\partial x_j'} \times \frac{\partial x_j'}{\partial x_j} \times \frac{\partial x_j}{\partial x_1} = \sum_j w_{j2} x_j'(1 - x_j') w_{1j} \tag{7.13}$$

7.2.4 BP 网络的优缺点

BP 网络的优点为：

(1) 只要有足够多的隐层和隐层节点，BP 网络可以逼近任意的非线性映射关系；

(2) BP 网络的学习算法属于全局逼近算法，具有较强的泛化能力；

(3) BP 网络输入输出之间的关联信息分布地存储在网络的连接权中，个别神经元的损坏只对输入输出关系有较小的影响，因而 BP 网络具有较好的容错性。

BP 网络的主要缺点为：

(1) 待寻优的参数多，收敛速度慢；

(2) 目标函数存在多个极值点，按梯度下降法进行学习，很容易陷入局部极小值；

(3) 难以确定隐层及隐层节点的数目。目前，如何根据特定的问题来确定具体的网络结构尚无很好的方法，仍需根据经验来试凑。

由于 BP 网络具有很好的逼近非线性映射的能力，该网络在模式识别、图像处理、系统辨识、函数拟合、优化计算、最优预测和自适应控制等领域有着较为广泛的应用。

由于 BP 网络具有很好的逼近特性和泛化能力，可用于神经网络控制器的设计。但由于 BP 网络收敛速度慢，难以适应实时控制的要求。

7.2.5 BP 网络逼近仿真实例

使用 BP 网络逼近对象

$$y(k) = u(k)^3 + \frac{y(k-1)}{1 + y(k-1)^2}$$

采样时间取 1ms。输入信号为 $u(k) = 0.5\sin(6\pi t)$。神经网络为 2-6-1 结构，权值 $\boldsymbol{W}_1$，$\boldsymbol{W}_2$ 的初始值取 $[-1, +1]$ 之间的随机值，取 $\eta = 0.50$，$\alpha = 0.05$。

BP 网络逼近程序见附录程序 chap7_1.m，仿真结果如图 7-8 至图 7-10 所示。

7.2.6 BP 网络模式识别

由于神经网络具有自学习、自组织和并行处理等特征，并具有很强的容错能力和联想能力，因此，神经网络具有模式识别的能力。

在神经网络模式识别中，根据标准的输入输出模式对，采用神经网络学习算法，以标准的

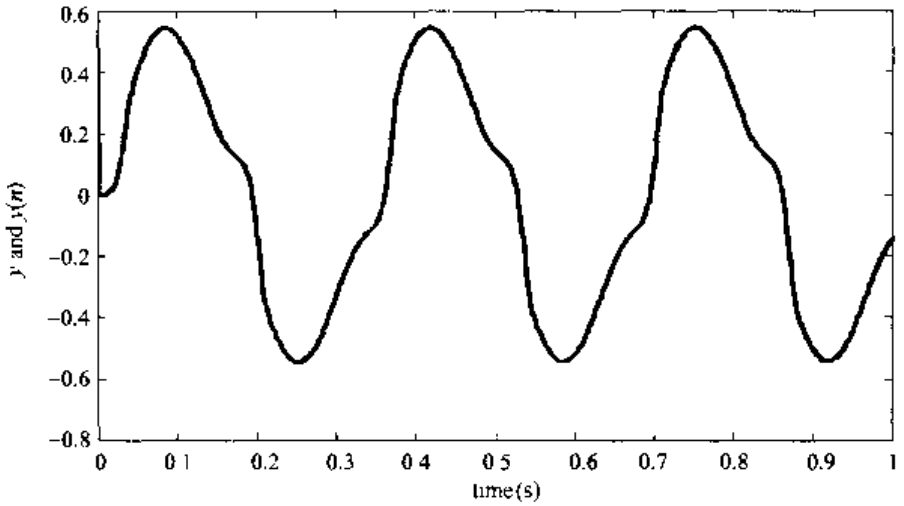


图 7-8　BP 网络逼近效果

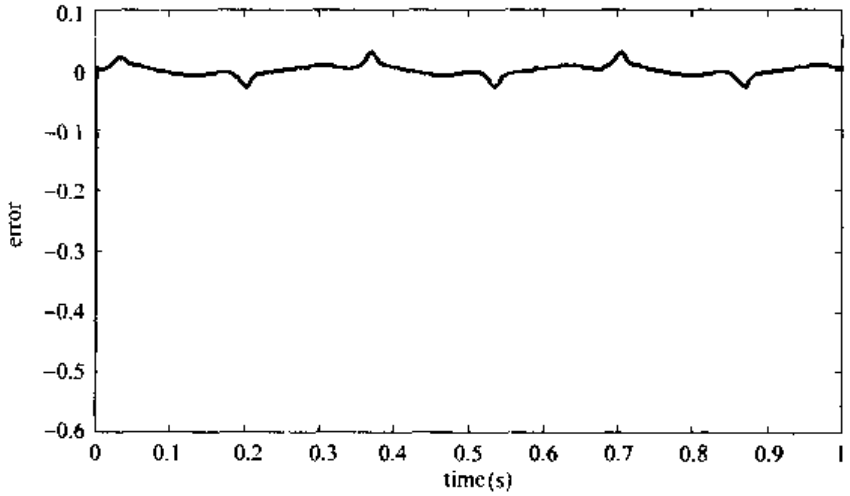


图 7-9　BP 网络逼近误差

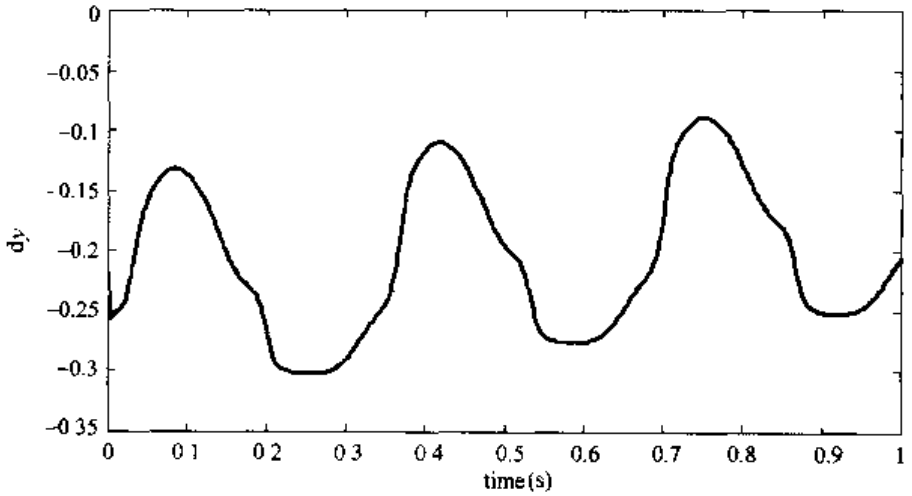


图 7-10　Jacobian 信息的辨识

模式作为学习样本进行训练，通过学习调整神经网络的连接权值。当训练满足要求后，得到的神经网络权值构成了模式识别的知识库，利用神经网络并行推理算法便可对所需要的输入模式进行识别。

神经网络模式识别具有较强的鲁棒性。当待识别的输入模式与训练样本中的某个输入模式相同时，神经网络识别的结果就是与训练样本中相对应的输出模式。当待识别的输入模式与训练样本中所有输入模式都不完全相同时，则可得到与其相近样本相对应的输出模式。当待识别的输入模式与训练样本中所有输入模式相差较远时，就不能得到正确的识别结果，此时可将这一模式作为新的样本进行训练，使神经网络获取新的知识，并存储到网络的权值矩阵中，从而增强网络的识别能力。

BP 网络的训练过程如下：正向传播是输入信号从输入层经隐层传向输出层，若输出层得到了期望的输出，则学习算法结束；否则，转至反向传播。

以第 p 个样本为例，用于训练的 BP 网络结构如图 7-11 所示。

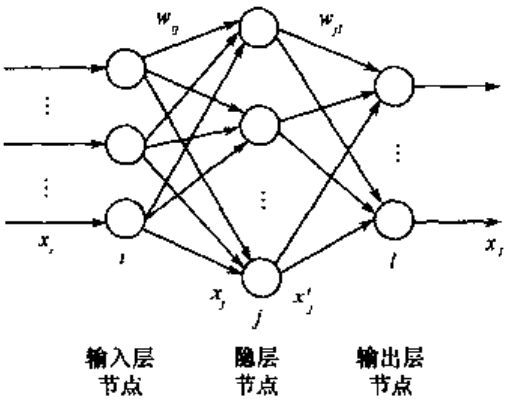


图 7-11　BP 神经网络结构

网络的学习算法如下：

(1) 前向传播：计算网络的输出。

隐层神经元的输入为所有输入的加权之和，即

$$x_j = \sum_i w_{ij} x_i \tag{7.14}$$

隐层神经元的输出 x'_j 采用 S 函数激发 x_j，得

$$x'_j = f(x_j) = \frac{1}{1+e^{-x_j}} \tag{7.15}$$

则

$$\frac{\partial x'_j}{\partial x_j} = x'_j(1-x'_j)$$

输出层神经元的输出为

$$x_l = \sum_j w_{jl} x'_j \tag{7.16}$$

网络第 l 个输出与相应理想输出 x_l^0 的误差为

$$e_l = x_l^0 - x_l$$

第 p 个样本的误差性能指标函数为

$$E_p = \frac{1}{2}\sum_{l=1}^{N} e_l^2 \tag{7.17}$$

式中，N 为网络输出层的个数。

(2) 反向传播：采用梯度下降法，调整各层间的权值。权值的学习算法如下：

输出层及隐层的连接权值 w_{jl} 学习算法为

$$\Delta w_{jl} = -\eta \frac{\partial E_p}{\partial w_{jl}} = \eta e_l \frac{\partial x_l}{\partial w_{jl}} = \eta e_l x_j'$$

式中，η 为学习速率，$\eta \in [0,1]$。

$k+1$ 时刻网络的权值为

$$w_{jl}(k+1) = w_{jl}(k) + \Delta w_{jl}$$

隐层及输入层连接权值 w_{ij} 学习算法为

$$\Delta w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ij}} = \eta \sum_{l=1}^{N} e_l \frac{\partial x_l}{\partial w_{ij}}$$

式中，$\frac{\partial x_l}{\partial w_{ij}} = \frac{\partial x_l}{\partial x_j'} \cdot \frac{\partial x_j'}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = w_{jl} \cdot \frac{\partial x_j'}{\partial x_j} \cdot x_i = w_{jl} \cdot x_j'(1-x_j') \cdot x_i$。

$t+1$ 时刻网络的权值为

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}$$

如果考虑上次权值对本次权值变化的影响，需要加入动量因子 α，此时的权值为

$$w_{jl}(k+1) = w_{jl}(k) + \Delta w_{jl} + \alpha(w_{jl}(k) - w_{jl}(k-1)) \tag{7.18}$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} + \alpha(w_{ij}(k) - w_{ij}(k-1)) \tag{7.19}$$

式中，α 为动量因子，$\alpha \in [0,1]$。

7.2.7　BP 网络模式识别仿真实例

取标准样本为三输入两输出样本，见表 7-1。

表 7-1　训练样本

输　入			输　出	
1	0	0	1	0
0	1	0	0	0.5
0	0	1	0	1

BP 网络为 3-6-2 结构，权值 w_{ij}，w_{jl} 的初始值取[−1，+1]之间的随机值，学习参数取 $\eta = 0.50$，$\alpha = 0.05$。

BP 网络模式识别程序包括网络训练程序 chap7_2a. m 和网络测试程序 chap7_2b. m，程序见附录。运行程序 chap7_2a. m，取网络训练的最终指标为 $E = 10^{-20}$，网络训练指标的变化如图 7-12 所示。将网络训练的最终权值为用于模式识别的知识库，将其保存在文件 wfile. dat 中。

仿真程序中，用 w_1，w_2 代表 w_{ij}，w_{jl}，用 I_{out} 表示 x_j'。运行程序 chap7_2b. m，调用文件 wfile. dat，取一组实际样本进行测试，测试样本及测试结果见表 7-2。由仿真结果可见，BP 网络具有很好的模式识别能力。

表 7-2　测试样本及结果

输　入			输　出	
0.970	0.001	0.001	0.9862	0.0094
0.000	0.980	0.000	0.0080	0.4972
0.002	0.000	1.040	−0.0145	1.0202
0.500	0.500	0.500	0.2395	0.6108
1.000	0.000	0.000	1.0000	−0.0000
0.000	1.000	0.000	0.0000	0.5000
0.000	0.000	1.000	−0.0000	1.0000

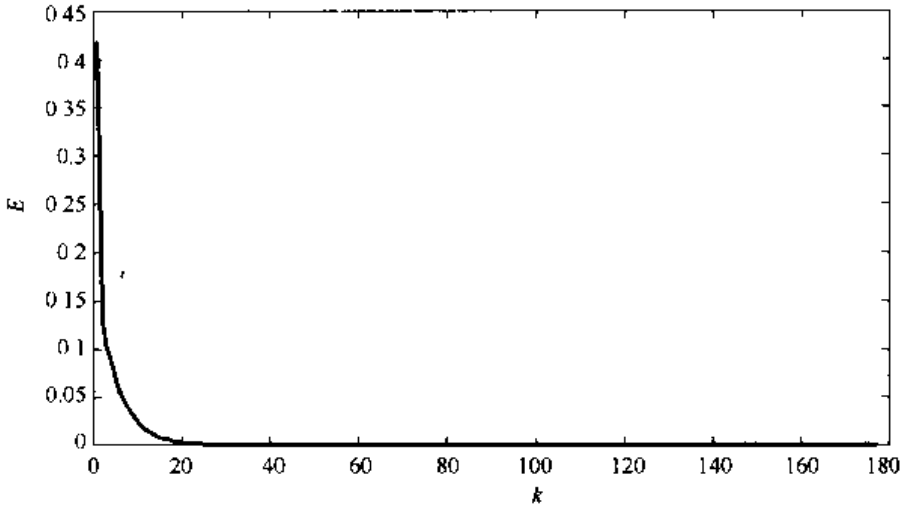


图 7-12　样本训练的收敛过程

7.3　RBF 神经网络

径向基函数(Radial Basis Function,RBF)神经网络是由 J. Moody 和 C. Darken 于 20 世纪 80 年代末提出的一种神经网络,它是具有单隐层的 3 层前馈网络。RBF 网络模拟了人脑中局部调整、相互覆盖接收域(或称感受野,Receptive Field)的神经网络结构,已证明 RBF 网络能任意精度逼近任意连续函数。

RBF 网络的学习过程与 BP 网络的学习过程类似,两者的主要区别在于各使用不同的作用函数。BP 网络中隐层使用的是 Sigmoid 函数,其值在输入空间中无限大的范围内为非零值,因而是一种全局逼近的神经网络;而 RBF 网络中的作用函数是高斯基函数,其值在输入空间中有限范围内为非零值,因而 RBF 网络是局部逼近的神经网络。

理论上,3 层以上的 BP 网络能够逼近任何一个非线性函数,但由于 BP 网络是全局逼近网络,每一次样本学习都要重新调整网络的所有权值,收敛速度慢,易于陷入局部极小,很难满足控制系统的高度实时性要求。RBF 网络是一种 3 层前向网络,由输入到输出的映射是非线性的,而隐层空间到输出空间的映射是线性的,而且 RBF 网络是局部逼近的神经网络,因而采用 RBF 网络可大大加快学习速度并避免局部极小问题,适合于实时控制的要求。采用 RBF 网络构成神经网络控制方案,可有效提高系统的精度、鲁棒性和自适应性。

7.3.1　RBF 网络结构

多输入单输出的 RBF 网络结构如图 7-13 所示。

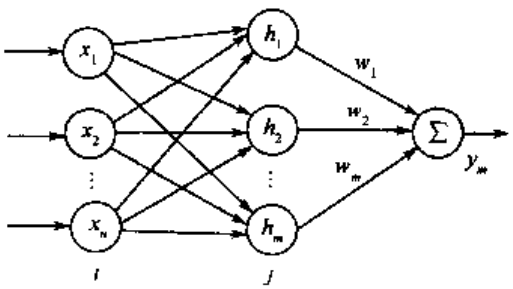


图 7-13　RBF 神经网络结构

7.3.2　RBF 网络的逼近

采用 RBF 网络逼近一对象的结构如图 7-14 所示。

在 RBF 网络结构中，$\boldsymbol{X}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$ 为网络的输入向量。设 RBF 网络的径向基向量 $\boldsymbol{H}=[h_1,h_2,\cdots,h_m]^{\mathrm{T}}$，其中 h_j 为高斯基函数，即

$$h_j=\exp\left(-\frac{\|\boldsymbol{X}-\boldsymbol{C}_j\|^2}{2b_j^2}\right),j=1,2,\cdots,m \tag{7.20}$$

式中，网络第 j 个节点的中心向量为 $\boldsymbol{C}_j=[c_{j1},c_{j2},\cdots,c_{jn}]^{\mathrm{T}}$，$i=1,2,\cdots,n$。

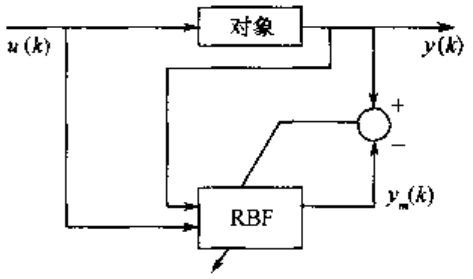


图 7-14　RBF 神经网络逼近

设网络的基宽向量为

$$\boldsymbol{B}=[b_1,b_2,\cdots,b_m]^{\mathrm{T}}$$

b_j 为节点 j 的基宽参数，且为大于零的数。网络的权向量为

$$\boldsymbol{W}=[w_1,w_2,\cdots,w_m]^{\mathrm{T}} \tag{7.21}$$

RBF 网络的输出为

$$y_m(k)=w_1h_1+w_2h_2+\cdots+w_mh_m \tag{7.22}$$

RBF 网络逼近的性能指标函数为

$$E(k)=\frac{1}{2}(y(k)-y_m(k))^2 \tag{7.23}$$

根据梯度下降法，输出权、节点基宽参数及节点中心矢量的迭代算法如下

$$w_j(k)=w_j(k-1)+\eta(y(k)-y_m(k))h_j+\alpha(w_j(k-1)-w_j(k-2)) \tag{7.24}$$

$$\Delta b_j=(y(k)-y_m(k))w_jh_j\frac{\|\boldsymbol{X}-\boldsymbol{C}_j\|^2}{b_j^3} \tag{7.25}$$

$$b_j(k)=b_j(k-1)+\eta\Delta b_j+\alpha(b_j(k-1)-b_j(k-2)) \tag{7.26}$$

$$\Delta c_{ji}=(y(k)-y_m(k))w_j\frac{x_i-c_{ji}}{b_j^2} \tag{7.27}$$

$$c_{ji}(k)=c_{ji}(k-1)+\eta\Delta c_{ji}+\alpha(c_{ji}(k-1)-c_{ji}(k-2)) \tag{7.28}$$

式中，η 为学习速率，α 为动量因子，$\eta\in[0,1]$，$\alpha\in[0,1]$。

将对象输出对输入的敏感度 $\dfrac{\partial y(k)}{\partial u(k)}$ 称为 Jacobian 信息，其值可由 RBF 神经网络辨识而得。辨识算法如下：取 RBF 网络的第一个输入为 $u(k)$，即 $x_1=u(k)$，则

$$\frac{\partial y(k)}{\partial u(k)}\approx\frac{\partial y_m(k)}{\partial u(k)}=\sum_j\frac{\partial w_jh_j}{\partial u(k)}=\sum_j w_j\frac{\partial h_j}{\partial u(k)}=\sum_j w_jh_j\frac{c(1,j)-x_1}{b_j^2} \tag{7.29}$$

7.3.3 RBF 网络逼近仿真实例

使用 RBF 网络逼近下列对象

$$y(k) = u(k)^3 + \frac{y(k-1)}{1+y(k-1)^2}$$

在 RBF 网络中，网络输入信号为两个，即 $u(k)$，$y(k)$，网络初始权值及高斯函数参数初始权值可取随机值，也可通过仿真测试后获得。

输入信号为正弦信号：$u(k) = 0.5\sin(2\pi t)$，采样时间为 0.001s，网络隐层神经元个数取 $m = 4$，网络结构为 2-4-1，网络的初始权值取随机值，高斯函数的初始值取 $\boldsymbol{C}_j = [0.5 \quad 0.5]^{\mathrm{T}}$，$\boldsymbol{B} = [1.5 \quad 1.5 \quad 1.5 \quad 1.5]^{\mathrm{T}}$。网络的学习参数取 $\alpha = 0.05$，$\eta = 0.5$。

RBF 网络逼近程序见附录程序 chap7_3.m。仿真结果如图 7-15 至图 7-17 所示。

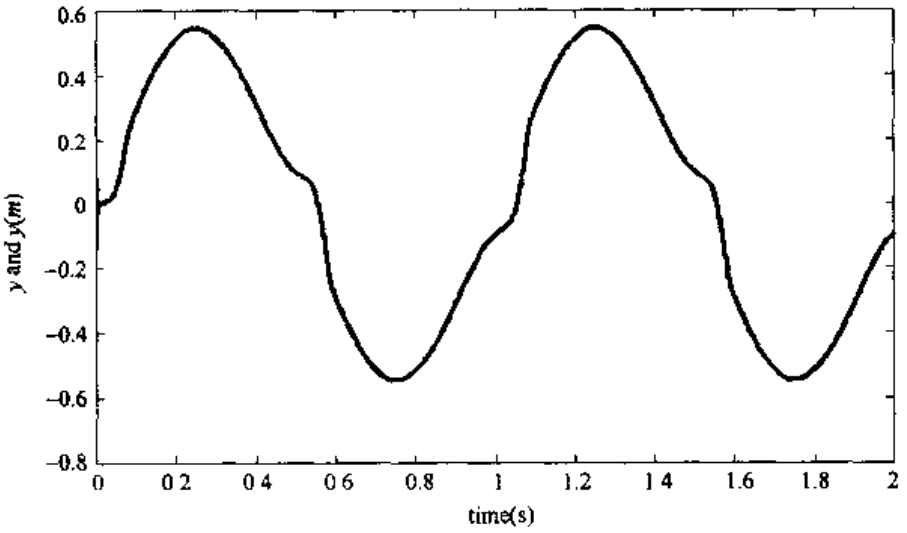


图 7-15 RBF 网络辨识结果

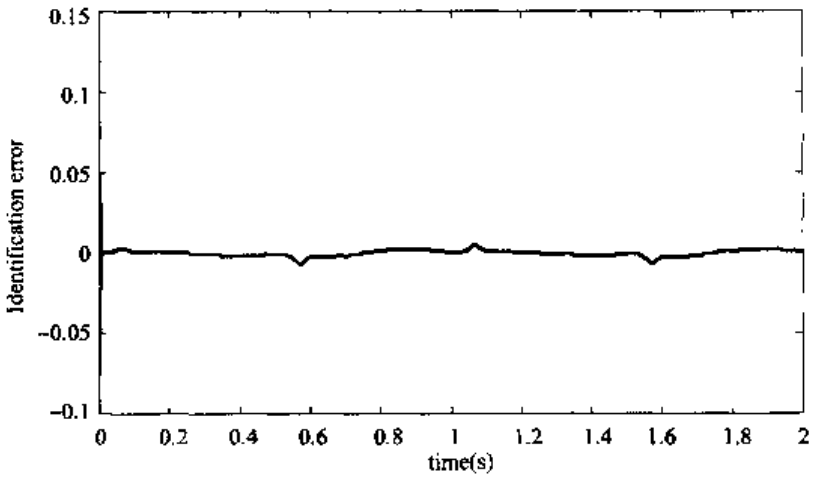


图 7-16 RBF 网络辨识误差

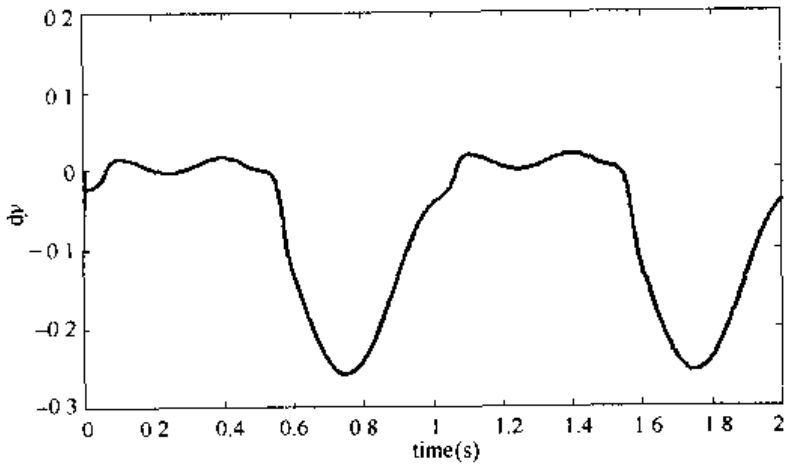


图 7-17　Jacobian 信息的辨识

7.4　回归神经网络

对角回归型神经网络(Diagonal Recurrent Neural Network,DRNN) 是具有反馈的动态网络,该网络能够更直接更生动地反映系统的动态特性,它在BP网络基本结构的基础上,通过存储内部状态使其具备映射动态特征的功能,从而使系统具有适应时变特性的能力,DRNN网络代表了神经网络的发展方向。

7.4.1　DRNN 网络结构

DRNN 网络是一种 3 层前向网络,其隐层为回归层。正向传播是输入信号从输入层经隐层传向输出层,若输出层得到了期望的输出,则学习算法结束;否则,转至反向传播。反向传播就是将误差信号(理想输出与实际输出之差) 按连接通路反向计算,由梯度下降法调整各层神经元的权值和阈值,使误差信号减小。

DRNN 网络结构如图 7-18 所示。

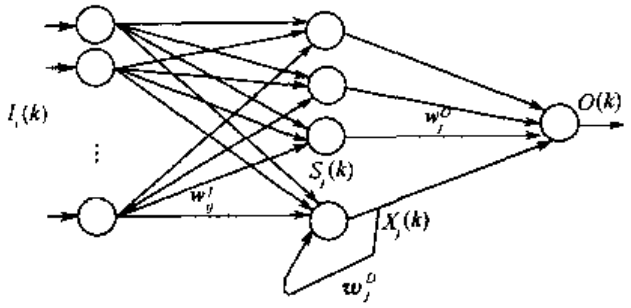


图 7-18　DRNN 神经网络结构

在该网络中,设 $\boldsymbol{I}=[I_1,I_2,\cdots I_n]$ 为网络输入向量,$I_i(k)$ 为输入层第 i 个神经元的输入,网络回归层第 j 个神经元的输出为 $X_j(k)$,$S_j(k)$ 为第 j 个回归神经元输入总和,$f(\cdot)$ 为 S 函数,$O(k)$ 为 DRNN 网络的输出。

DRNN 神经网络的算法为

$$O(k)=\sum_j w_j^O X_j(k),X_j(k)=f(S_j(k)),S_j(k)=w_j^D X_j(k-1)+\sum_i w_{ij}^I I_i(k) \quad (7.30)$$

式中，$\boldsymbol{W}^D$ 和 $\boldsymbol{W}^O$ 为网络回归层和输出层的权值向量，$\boldsymbol{W}^I$ 为网络输入层的权值向量。以 $\boldsymbol{W}^I$ 为例，可表示为

$$\boldsymbol{W}^I = [w_{1j}^I \vdots w_{2j}^I \vdots \cdots \vdots w_{nj}^I] \tag{7.31}$$

7.4.2 DRNN 网络的逼近

DRNN 网络逼近的结构如图 7-19 所示，图中 k 为网络的迭代步骤，$u(k)$ 和 $y(k)$ 为辨识器的输入。DRNN 为网络辨识器。$y(k)$ 为被控对象实际输出，$y_m(k)$ 为 DRNN 的输出。将系统输出 $y(k)$ 及输入 $u(k)$ 的值作为辨识器 DRNN 的输入，将系统输出与网络输出的误差作为辨识器的调整信号。

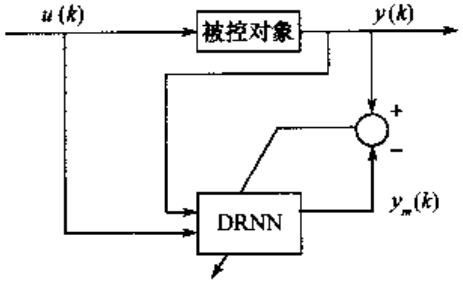


图 7-19　DRNN 神经网络逼近

网络输出层的输出为

$$y_m(k) = O(k) = \sum_j w_j^O X_j(k) \tag{7.32}$$

网络回归层的输出为

$$X_j(k) = f(S_j(k)) \tag{7.33}$$

网络回归层的输入为

$$S_j(k) = w_j^D X_j(k-1) + \sum_i (w_{ij}^I I_i(k)) \tag{7.34}$$

逼近误差为

$$e(k) = y(k) - y_m(k) \tag{7.35}$$

辨识指标取

$$E(k) = \frac{1}{2} e(k)^2 \tag{7.36}$$

学习算法采用梯度下降法

$$\Delta w_j^O(k) = -\frac{\partial E(k)}{\partial w_j^O} = e(k) \frac{\partial y_m}{\partial w_j^O} = e(k) X_j(k) \tag{7.37}$$

$$w_j^O(k) = w_j^O(k-1) + \eta_O \Delta w_j^O(k) + \alpha(w_j^O(k-1) - w_j^O(k-2)) \tag{7.38}$$

$$\Delta w_{ij}^I(k) = -\frac{\partial E(k)}{\partial w_{ij}^I} = e(k) \frac{\partial y_m}{\partial w_{ij}^I} = e(k) \frac{\partial y_m}{\partial X_j} \frac{\partial X_j}{\partial w_{ij}^I} = e(k) w_j^O Q_{ij}(k) \tag{7.39}$$

$$w_{ij}^I(k) = w_{ij}^I(k-1) + \eta_I \Delta_{ij}^I(k) + \alpha(w_{ij}^I(k-1) - w_{ij}^I(k-2)) \tag{7.40}$$

$$\Delta w_j^D(k) = -\frac{\partial E(k)}{\partial w_j^D} = e(k) \frac{\partial y_m}{\partial X_j} \frac{\partial X_j}{\partial w_j^D} = e(k) w_j^O P_j(k) \tag{7.41}$$

$$w_j^D(k) = w_j^D(k-1) + \eta_O \Delta w_j^O(k) + \alpha(w_j^O(k-1) - w_j^D(k-2)) \tag{7.42}$$

其中回归层神经元取双S函数为

$$f(x)=\frac{1-e^{-x}}{1+e^{-x}} \tag{7.43}$$

$$P_j(k)=\frac{\partial X_j}{\partial w_j^D}=f'(S_j)X_j(k-1) \tag{7.44}$$

$$Q_{ij}(k)=\frac{\partial X_j}{\partial w_{ij}^I}=f'(S_j)I_i(k) \tag{7.45}$$

式中，η_O,η_D,η_I 分别为输入层、回归层和输出层的学习速率，α 为惯性系数。

7.4.3 DRNN网络逼近仿真实例

使用DRNN网络逼近对象

$$y(k)=u(k)^3+\frac{y(k-1)}{1+y(k-1)^2}$$

其中对象采样时间为1ms。

神经网络权值 $\boldsymbol{W}^D,\boldsymbol{W}^O$ 和 $\boldsymbol{W}^I$ 的初始值取随机值，取 $\eta_O=0.35,\eta_D=0.35,\eta_I=0.35$。DRNN网络逼近程序见附录程序chap7_4.m。仿真结果如图7-20和图7-21所示。

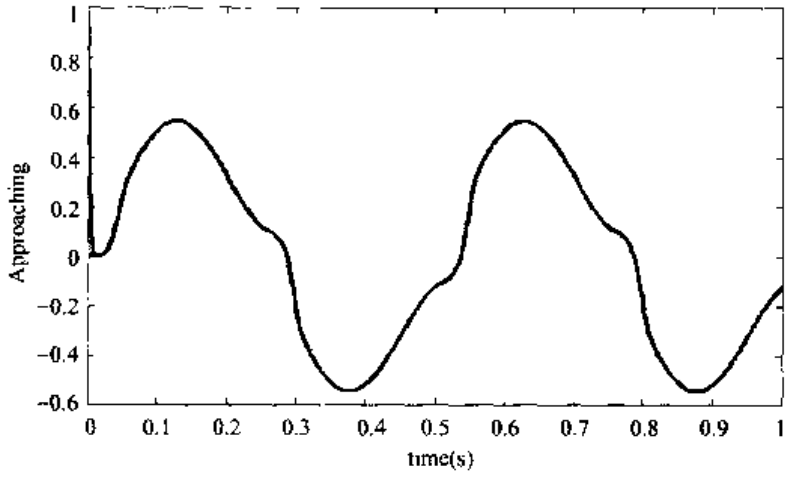


图7-20 DRNN网络逼近效果

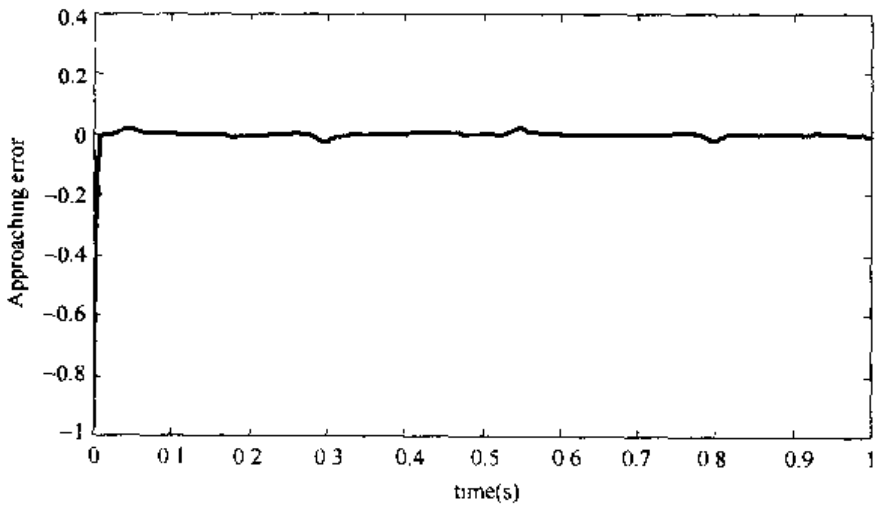


图7-21 DRNN网络逼近误差

思考题与习题

7-1　采用 BP 网络进行模式识别。训练样本为 3 对两输入单输出样本，见表 7-3。

表 7-3　训练样本

输　入		输　出
1	0	1
0	0	0
0	1	−1

试采用 BP 网络对训练样本进行训练，并针对一组实际样本进行测试。用于测试的 3 组样本输入分别为 1，0.1；0.5，0.5 和 0.1，1。

7-2　采用 BP 网络、RBF 网络、DRNN 网络逼近非线性对象 $y(k)=(u(k-1)-0.9y(k-1))/(1+y(k-1)^2)$，分别进行 Matlab 仿真。

附录（程序代码）

BP 网络逼近程序：chap7_1.m

```
% BP identification
clear all;
close all;

xite=0.50;
alfa=0.05;

w2=rands(6,1);
w2_1=w2;w2_2=w2_1;

w1=rands(2,6);
w1_1=w1;w1_2=w1;

dw1=0*w1;

x=[0,0]';

u_1=0;
y_1=0;

I=[0,0,0,0,0,0]';
Iout=[0,0,0,0,0,0]';
FI=[0,0,0,0,0,0]';

ts=0.001;
for  k=1:1:1000

time(k)=k*ts;
u(k)=0.50*sin(3*2*pi*k*ts);
y(k)=u_1^3+y_1/(1+y_1^2);

for  j=1:1:6
     I(j)=x'*w1(:,j);
     Iout(j)=1/(1+exp(-I(j)));
end

yn(k)=w2'*Iout;          % Output of NNI networks
```

```
e(k)=y(k)-yn(k);      % Error calculation

w2=w2_1+(xite*e(k))*Iout+alfa*(w2_1-w2_2);

for j=1:1:6
    FI(j)=exp(-I(j))/(1+exp(-I(j)))^2;
end

for i=1:1:2
    for j=1:1:6
        dw1(i,j)=e(k)*xite*FI(j)*w2(j)*x(i);
    end
end
w1=w1_1+dw1+alfa*(w1_1-w1_2);

%%%%%%%%%%%%%%Jacobian%%%%%%%%%%%%%%%%
yu=0;
for j=1:1:6
    yu=yu+w2(j)*w1(1,j)*FI(j);
end
dyu(k)=yu;

x(1)=u(k);
x(2)=y(k);

w1_2=w1_1;w1_1=w1;
w2_2=w2_1;w2_1=w2;
u_1=u(k);
y_1=y(k);
end
figure(1);
plot(time,y,'r',time,yn,'b');
xlabel('times');ylabel('y and yn');
figure(2);
plot(time,y-yn,'r');
xlabel('times');ylabel('error');
figure(3);
plot(time,dyu);
xlabel('times');ylabel('dyu');
```

BP网络模式识别程序：包括网络训练程序 chap7_2a. m 和网络测试程序 chap7_2b. m。

(1) **网络训练程序**：chap7_2a. m

```
% BP Training for MIMO and Multi—samples
clear all;
close all;

xite=0.50;
alfa=0.05;

w2=rands(6,2);
w2_1=w2;w2_2=w2_1;

w1=rands(3,6);
w1_1=w1;w1_2=w1;
dw1=0 * w1;

I=[0,0,0,0,0,0]';
Iout=[0,0,0,0,0,0]';
FI=[0,0,0,0,0,0]';

OUT=2;
k=0;
E=1.0;
NS=3;

while E>=1e—020
k=k+1;
times(k)=k;

for s=1 : 1 : NS      % MIMO Samples
xs=[1,0,0;
    0,1,0;
    0,0,1];       % Ideal Input
ys=[1,0;
     0.0.5;
     0,1];         % Ideal Output

x=xs(s, : );
for j=1 : 1 : 6
```

```
    I(j)=x*w1(:,j);
    Iout(j)=1/(1+exp(-I(j)));
end

yl=w2'*Iout;
yl=yl';

el=0;
y=ys(s,:);
for l=1:1:OUT
    el=el+0.5*(y(l)-yl(l))^2;    %Output error
end

E=0;
if s==NS
    for s=1:1:NS
        E=E+el;
  end
  end
el=y-yl;

w2=w2_1+xite*Iout*el+alfa*(w2_1-w2_2);

for j=1:1:6
    S=1/(1+exp(-I(j)));
    FI(j)=S*(1-S);
end

for i=1:1:3
    for j=1:1:6
        dw1(i,j)=xite*FI(j)*x(i)*(el(1)*w2(j,1)+el(2)*w2(j,2));
    end
end
w1=w1_1+dw1+alfa*(w1_1-w1_2);

w1_2=w1_1;w1_1=w1;
w2_2=w2_1;w2_1=w2;
end %End of for
Ek(k)=E;
end   %End of while
```

```
figure(1);
plot(times,Ek,'r');
xlabel('k');ylabel('E');

save wfile w1 w2;
```

(2) **网络测试程序**:chap7_2b.m

```
%Test BP
clear all;
load wfile w1 w2;

%N Samples
x=[0.970,0.001,0.001;
   0.000,0.980,0.000;
   0.002,0.000,1.040;
   0.500,0.500,0.500;
   1.000,0.000,0.000;
   0.000,1.000,0.000;
   0.000,0.000,1.000];
for i=1:1:7
   for j=1:1:6
   I(i,j)=x(i,:)*w1(:,j);
   Iout(i,j)=1/(1+exp(-I(i,j)));
   end
end
y=w2'*Iout';
y=y'
```

RBF网络逼近程序:chap7_3.m

```
%RBF identification
clear all;
close all;

alfa=0.05;
xite=0.5;
x=[0,0]';

b=1.5*ones(4,1);
c=0.5*ones(2,4);
w=rands(4,1);
```

```
w_1=w;w_2=w_1;
c_1=c;c_2=c_1;
b_1=b;b_2=b_1;
d_w=0*w;
d_b=0*b;
y_1=0;

ts=0.001;
for k=1:1:2000

time(k)=k*ts;
u(k)=0.50*sin(1*2*pi*k*ts);

y(k)=u(k)^3+y_1/(1+y_1^2);

x(1)=u(k);
x(2)=y(k);

for j=1:1:4
    h(j)=exp(-norm(x-c(:,j))^2/(2*b(j)*b(j)));
end
ym(k)=w'*h';
em(k)=y(k)-ym(k);

for j=1:1:4
    d_w(j)=xite*em(k)*h(j);
    d_b(j)=xite*em(k)*w(j)*h(j)*(b(j)^-3)*norm(x-c(:,j))^2;
    for i=1:1:2
    d_c(i,j)=xite*em(k)*w(j)*h(j)*(x(i)-c(i,j))*(b(j)^-2);
    end
end
    w=w_1+d_w+alfa*(w_1-w_2);
    b=b_1+d_b+alfa*(b_1-b_2);
    c=c_1+d_c+alfa*(c_1-c_2);

%%%%%%%%%%%Jacobian%%%%%%%%%%%%%
yu=0;
for j=1:1:4
yu=yu+w(j)*h(j)*(c(1,j)-x(1))/b(j)^2;
```

```
end
dyu(k)=yu;

y_1=y(k);

w_2=w_1;
w_1=w;

c_2=c_1;
c_1=c;

b_2=b_1;
b_1=b;
end
figure(1);
plot(time,y,'r',time,ym,'b');
xlabel('time(s)');ylabel('y and ym');

figure(2);
plot(time,y-ym,'r');
xlabel('time(s)');ylabel('identification error');

figure(3);
plot(time,dyu,'r');
xlabel('times');ylabel('dyu');
```

DRNN 网络逼近程序：chap7_4.m

```
%Diagonal Recurrent Neural Network
clear all;
close all;

wd=rands(7,1);
wo=rands(7,1);
wi=rands(3,7);

xj=zeros(7,1);
xj_1=xj;

u_1=0;y_1=0;
```

```
xitei=0.35;
xited=0.35;
xiteo=0.35;

ts=0.001;
for k=1:1:1000
time(k)=k*ts;
u(k)=0.5*sin(4*pi*k*ts);
y(k)=u_1^3+y_1/(1+y_1^2);

Ini=[u_1,y_1,1]';
for j=1:1:7
    sj(j)=Ini'*wi(:,j)+wd(j)*xj(j);
end
for j=1:1:7
    xj(j)=(1-exp(-sj(j)))/(1+exp(-sj(j)));
end

Pj=0*xj;
for j=1:1:7
    Pj(j)=wo(j)*(1+xj(j))*(1-xj(j))*xj_1(j);
end

Qij=0*wi;
for j=1:1:7
    for i=1:1:3
        Qij(i,j)=wo(j)*(1+xj(j))*(1-xj(j))*Ini(i);
    end
end

ymk=0;
for j=1:1:7
    ymk=ymk+xj(j)*wo(j);
end
ym(k)=ymk;
e(k)=y(k)-ym(k);

wo=wo+xiteo*e(k)*xj;
wd=wd+xited*e(k)*Pj;
wi=wi+xitei*e(k)*Qij;
```

```
xi_1=xj;
u_1=u(k);
y_1=y(k);
end
figure(1);
plot(time,y,'r',time,ym,'b');
xlabel('time(s)');ylabel('Approaching');
figure(2);
plot(time,y-ym,'r');
xlabel('time(s)');ylabel('Approaching error');
```

第8章　高级神经网络

8.1　模糊 RBF 网络

在模糊系统中，模糊集、隶属函数和模糊规则的设计是建立在经验知识基础上的。这种设计方法存在很大的主观性。将学习机制引入到模糊系统中，使模糊系统能够通过不断学习来修改和完善隶属函数和模糊规则，这是模糊系统的发展方向。

模糊系统与模糊神经网络既有联系又有区别，其联系表现为模糊神经网络在本质上是模糊系统的实现，其区别表现为模糊神经网络又具有神经网络的特性。

神经网络与模糊系统的比较见表 8 1。模糊神经网络充分地利用了神经网络和模糊系统各自的优点，因而受到了重视。

表 8-1　模糊系统与神经网络的比较

	模糊系统	神经网络
获取知识	专家经验	算法实例
推理机制	启发式搜索	并行计算
推理速度	低	高
容错性	低	非常高
学习机制	归纳	调整权值
自然语言实现	明确的	不明显
自然语言灵活性	高	低

将神经网络的学习能力引入到模糊系统中，将模糊系统的模糊化处理、模糊推理、精确化计算通过分布式的神经网络来表示是实现模糊系统自组织、自学习的重要途径。在模糊神经网络中，神经网络的输入、输出节点用来表示模糊系统的输入、输出信号，神经网络的隐含节点用来表示隶属函数和模糊规则，利用神经网络的并行处理能力使得模糊系统的推理能力大大提高。

模糊神经网络是将模糊系统和神经网络相结合而构成的网络。模糊神经网络在本质上是将常规的神经网络赋予模糊输入信号和模糊权值，其学习算法通常是神经网络学习算法或其推广。模糊神经网络技术已经获得了广泛的应用，当前的应用主要集中模糊回归、模糊控制、模糊专家系统、模糊矩阵方程、模糊建模和模糊模式识别等领域。利用 RBF 网络与模糊系统相结合，构成了模糊 RBF 网络。

8.1.1　网络结构

图 8-1 所示为模糊 RBF 神经网络结构，该网络由输入层、模糊化层、模糊推理层和输出层构成。

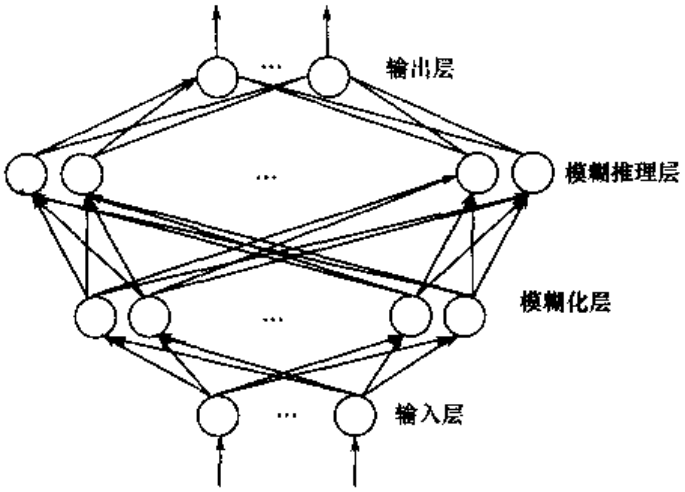


图 8-1　模糊 RBF 神经网络结构

模糊 RBF 网络中信号传播及各层的功能表示如下：

第一层：输入层

该层的各个节点直接与输入量的各个分量连接，将输入量传到下一层。对该层的每个节点 i 的输入输出表示为

$$f_1(i) = x_i \tag{8.1}$$

第二层：模糊化层

采用高斯型函数作为隶属函数，c_{ij} 和 b_j 分别是第 i 个输入变量第 j 个模糊集合的隶属函数的均值和标准差。即

$$f_2(i,j) = \exp(\text{net}_j^2) \tag{8.2}$$

$$\text{net}_j^2 = -\frac{(f_1(i) - c_{ij})^2}{(b_j)^2} \tag{8.3}$$

第三层：模糊推理层

该层通过与模糊化层的连接来完成模糊规则的匹配，各个节点之间实现模糊运算，即通过各个模糊节点的组合得到相应的点火强度。每个节点 j 的输出为该节点所有输入信号的乘积，即

$$f_3(j) = \prod_{j=1}^{N} f_2(i,j) \tag{8.4}$$

式中，$N = \prod_{i=1}^{n} N_i$，N_i 为输入层中第 i 个输入隶属函数的个数，即模糊化层节点数。

第四层：输出层

输出层为 f_4，即

$$f_4(l) = \boldsymbol{W} \cdot f_3 = \sum_{j=1}^{N} w(l,j) \cdot f_3(j) \tag{8.5}$$

式中，l 为输出层节点的个数，$\boldsymbol{W}$ 为输出层节点与第三层各节点的连接权矩阵。

8.1.2 基于模糊 RBF 网络的逼近算法

采用模糊 RBF 网络逼近对象，取网络结构为 2-4-1，如图 8-2 所示。

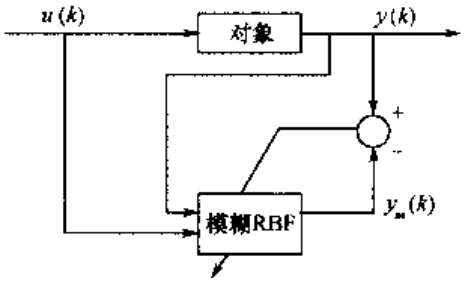


图 8-2 模糊 RBF 神经网络逼近

取 $y_m(k)=f_4$，$y_m(k)$ 和 $y(k)$ 分别表示网络输出和实际输出。网络的输入为 $u(k)$ 和 $y(k)$，网络的输出为 $y_m(k)$，则网络逼近误差为

$$e(k)=y(k)-y_m(k) \tag{8.6}$$

采用梯度下降法来修正可调参数，定义目标函数为

$$E=\frac{1}{2}e(k)^2 \tag{8.7}$$

网络的学习算法如下：

输出层的权值通过如下方式来调整

$$\Delta w(k)=-\eta\frac{\partial E}{\partial w}=-\eta\frac{\partial E}{\partial e}\frac{\partial e}{\partial y_m}\frac{\partial y_m}{\partial w}=\eta e(k)f_3$$

则输出层的权值学习算法为

$$w(k)=w(k-1)+\Delta w(k)+\alpha(w(k-1)-w(k-2)) \tag{8.8}$$

式中，η 为学习速率，α 为动量因子，$\eta\in[0,1]$，$\alpha\in[0,1]$。

隶属函数参数通过如下方式调整

$$\Delta c_{ij}=-\eta\frac{\partial E}{\partial c_{ij}}=-\eta\frac{\partial E}{\partial \mathrm{net}_j^2}\frac{\partial \mathrm{net}_j^2}{\partial c_{ij}}=-\eta\delta_j^2\frac{2(x_i-c_{ij})}{b_{ij}^2}$$

$$\Delta b_j=-\eta\frac{\partial E}{\partial b_j}=-\eta\frac{\partial E}{\partial \mathrm{net}_j^2}\frac{\partial \mathrm{net}_j^2}{\partial b_j}=\eta\delta_j^2\frac{2(x_i-c_{ij})}{b_j^3}$$

式中

$$\delta_j^2=\frac{\partial E}{\partial \mathrm{net}_j^2}=-e(k)\frac{\partial y_m}{\partial \mathrm{net}_j^2}=-e(k)\frac{\partial y_m}{\partial f_3}\frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial \mathrm{net}_j^2}=-e(k)wf_3$$

隶属函数参数学习算法为

$$c_{ij}(k)=c_{ij}(k-1)+\Delta c_{ij}(k)+\alpha(c_{ij}(k-1)-c_{ij}(k-2)) \tag{8.9}$$

$$b_j(k)=b_j(k-1)+\Delta b_j(k)+\alpha(b_j(k-1)-b_j(k-2)) \tag{8.10}$$

8.1.3 仿真实例

使用模糊 RBF 网络逼近对象

$$y(k)=u(k)^3+\frac{y(k-1)}{1+y(k-1)^2}$$

其中采样时间为 1ms。

输入信号为正弦信号：$u(k)=0.5\sin(6\pi t)$。网络结构选 2-25-25-1，神经网络权值 W 的初

始值取[－1，＋1]之间的随机值，中心矢量和高斯基宽向量的初值取 $\boldsymbol{C}=(c_{ij})=\begin{bmatrix}-5 & -2 & 0 & 2 & 5\\ -5 & -2 & 0 & 2 & 5\end{bmatrix}$ 和 $\boldsymbol{B}=(b_j)=[5\quad 5\quad 5\quad 5\quad 5]^{\mathrm{T}}$，网络的学习参数取 $\eta=0.20$，$\alpha=0.05$。

模糊 RBF 网络逼近程序见附录程序 chap8_1.m，仿真结果如图 8-3 和图 8-4 所示。

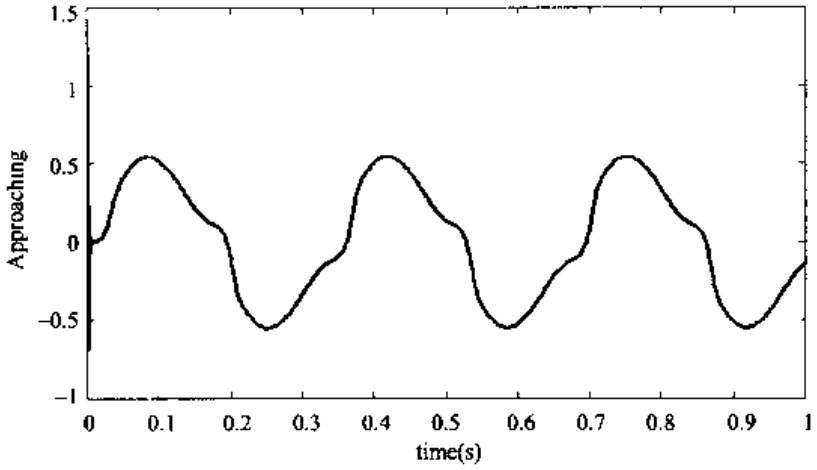


图 8-3　模糊 RBF 网络逼近效果

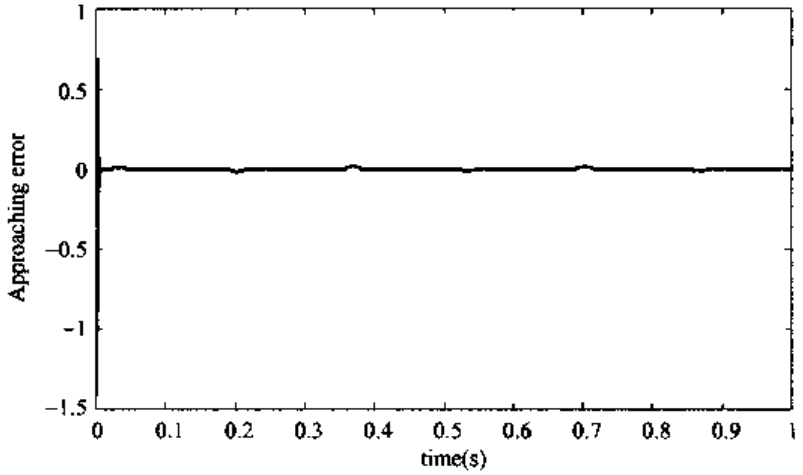


图 8-4　模糊 RBF 网络逼近误差

8.2　pi-sigma 神经网络

神经模糊建模是近年来基于模糊集理论发展起来的一种新的方法。模糊建模技术在 20 世纪 70 年代末就已经开始研究了，其缺点是过分地依赖隶属函数的准确性。采用高木－关野模糊系统，用一种混合型的 pi-sigma 神经网络，可以建立一种自适应能力很强的模糊模型。这种模型不但实现了模糊模型的自动更新，而且能不断修正各模糊子集的隶属函数，使模糊建模更具合理性。

8.2.1　高木-关野模糊系统

在高木－关野模糊系统中，高木和关野用以下“if－then”规则的形式来定义模糊系统的规则

$$R^i: \text{If } x_1 \text{ is } A_1^i,\ x_2 \text{ is } A_2^i, \cdots, x_n \text{ is } A_n^i \text{ then } y^i = p_0^i + p_1^i x_1 + \cdots + p_n^i x_n \tag{8.11}$$

式中，A_j^i 为模糊集，p_r^i 为真值参数，y^i 为系统根据规则 R^i 所得的输出，$i = 1,2,\cdots,m$，$r = 0,1,\cdots,n$，$j = 1,2,\cdots,n$。其中"If"部分是模糊的，"then"部分是确定的，即输出为各输入变量的线性组合。对于输入向量 $\boldsymbol{x} = [x_1, x_2, \cdots, x_k]^{\mathrm{T}}$，高木 - 关野模糊系统的各规则输出 y_n 等于各 y_i 的加权平均，即

$$y_n = \frac{\sum_{i=1}^{m} w^i y^i}{\sum_{i=1}^{m} w^i} \tag{8.12}$$

式中，加权系数 w^i 包括了规则 R^i 作用于输入所取得的值。即

$$w^i = \prod_{j=1}^{k} \mu_{A_j^i}(x_j) \tag{8.13}$$

8.2.2 混合型 pi-sigma 神经网络

常规的前向型神经网络含有求和节点，这给处理某些复杂问题带来了困难。一种基于混合型 pi-sigma 神经网络模型如图 8-5 所示，在该网络中，输入神经元有 4 个，S，P 和 · 分别表示相加、相乘和相乘运算。网络的输出为

$$y_n = \frac{\sum_{i=1}^{m} w^i y^i}{\sum_{i=1}^{m} w^i} = \frac{\sum_{i=1}^{m} \left[\mu_{A_1^i}(x_1) \cdot \mu_{A_2^i}(x_2) \cdot \mu_{A_3^i}(x_3) \cdot \mu_{A_4^i}(x_4)(p_0^i + p_1^i x_1 + p_2^i x_2 + p_3^i x_3 + p_4^i x_4)\right]}{\sum_{i=1}^{m} \left[\mu_{A_1^i}(x_1) \cdot \mu_{A_2^i}(x_2) \cdot \mu_{A_3^i}(x_3) \cdot \mu_{A_4^i}(x_4)\right]} \tag{8.14}$$

这种结构的神经网络属于高木 · 关野模糊系统。采用该网络实现的模糊系统可方便地在线修正隶属函数和参数，适合于复杂系统的模糊预测和控制。

为方便神经网络的学习，各模糊子集的隶属函数均取高斯型，即

$$\mu_{A_j^i} = \exp\left[-(x_j - c_j^i)^2 / b_j^i\right] \tag{8.15}$$

混合型 pi-sigma 神经网络学习算法：假设网络的期望输出为 y_d，定义代价函数为

$$E = \frac{1}{2}(y_d - y_n)^2 \tag{8.16}$$

根据梯度下降法有

$$p_j^i(k) = p_j^i(k-1) - \alpha \frac{\partial E}{\partial p_j^i} \tag{8.17}$$

其中

$$\frac{\partial E}{\partial p_j^i} = \frac{\partial E}{\partial y_n}\frac{\partial y_n}{\partial p_j^i} = -(y_d - y_n)\partial\left[\frac{\sum_{i=1}^{m} w^i y^i}{\sum_{i=1}^{m} w^i}\right] \Big/ \partial p_j^i = -(y_d - y_n)\frac{w^i}{\sum_{i=1}^{m} w^i} \cdot \frac{\partial y^i}{\partial p_j^i} \tag{8.18}$$

$$p_j^i(k) = p_j^i(k-1) - \alpha \frac{\partial E}{\partial p_j^i} = p_j^i(k-1) + \alpha(y_d - y_n)w^i \Big/ \sum_{i=1}^{m} w^i \cdot x_j \tag{8.19}$$

式中，$j = 0,1,2,3,4$。当 $j = 0$ 时，$x_j = 1$。

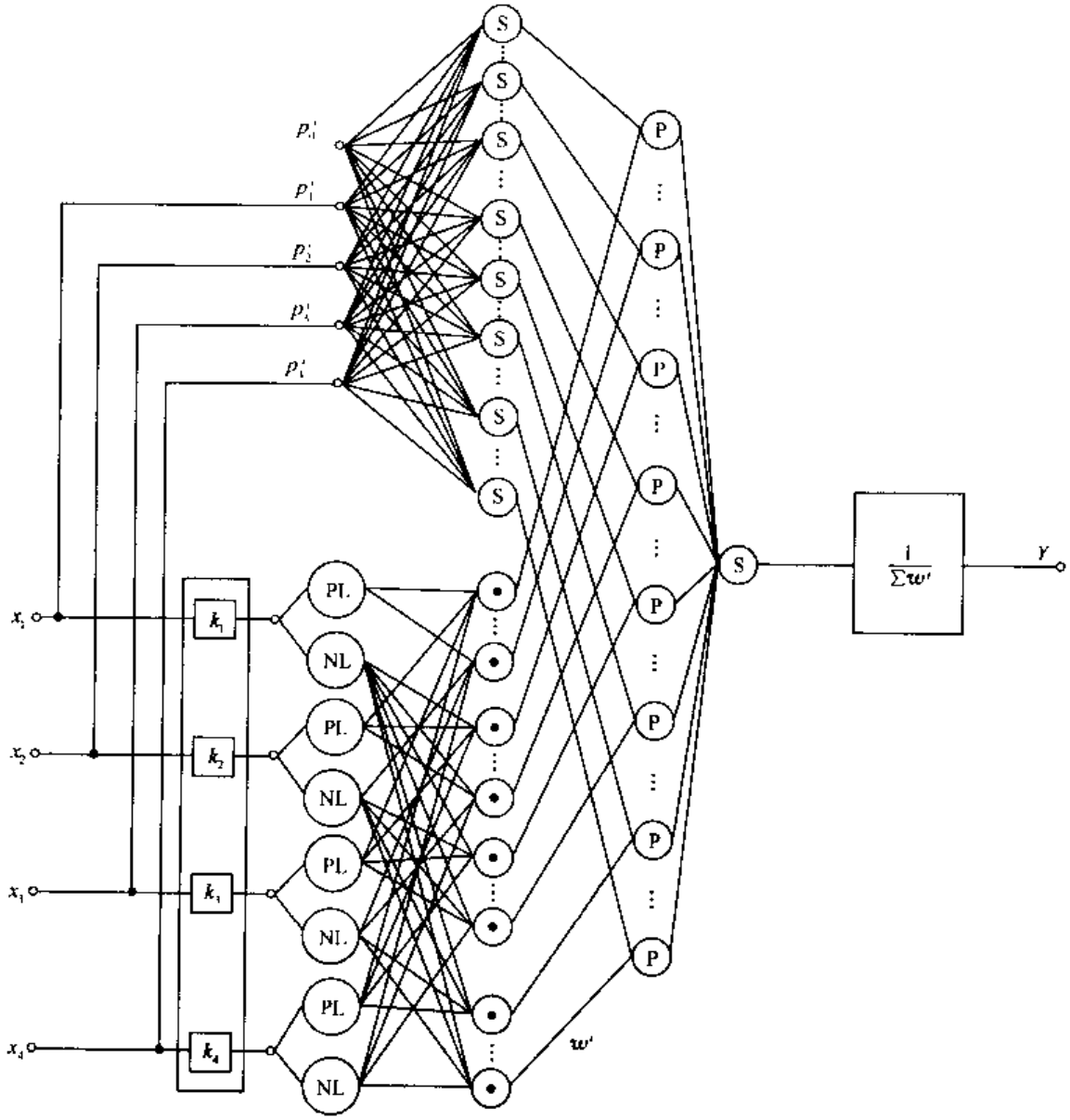


图 8-5　具有 4 个输入神经元的混合型 pi-sigma 神经网络

对 c_j^i, b_j^i 有

$$c_j^i(k) = c_j^i(k-1) - \beta \frac{\partial E}{\partial a_j^i} \tag{8.20}$$

$$b_j^i(k) = b_j^i(k-1) - \beta \frac{\partial E}{\partial b_j^i} \tag{8.21}$$

其中

$$\frac{\partial E}{\partial c_j^i} = \frac{\partial E}{\partial y_n} \cdot \frac{\partial y_n}{\partial c_j^i} = \frac{\partial E}{\partial y_n} \cdot \frac{\partial y_n}{\partial w^i} \cdot \frac{\partial w^i}{\partial c_j^i}$$

$$= -(y_d - y_n) \cdot \frac{y^i \sum_{i=1}^{m} w^i - \sum_{i=1}^{m} (w^i y^i)}{\left(\sum_{i=1}^{m} w^i\right)^2} \cdot 2(x_j - c_j^i) w^i \Big/ b_j^i \tag{8.22}$$

$$\frac{\partial E}{\partial b_j^i}=\frac{\partial E}{\partial y_n}\cdot\frac{\partial y_n}{\partial b_j^i}=\frac{\partial E}{\partial y_n}\cdot\frac{\partial y_n}{\partial w^i}\cdot\frac{\partial w^i}{\partial b_j^i}$$

$$=-(y_d-y_n)\frac{y^i\sum_{i=1}^{m}w^i-\sum_{i=1}^{m}(w^iy^i)}{\left(\sum_{i=1}^{m}w^i\right)^2}\cdot w^i\left[-(x_j-c_j^i)^2\right]\left(-\frac{1}{(b_j^i)^2}\right)$$

$$=-(y_d-y_n)\frac{y^i\sum_{i=1}^{m}w^i-\sum_{i=1}^{m}(w^iy^i)}{\left(\sum_{i=1}^{m}w^i\right)^2}(x_j-c_j^i)^2w^i\Big/(b_j^i)^2 \tag{8.23}$$

$$c_j^i(k)=c_j^i(k-1)+\beta(y_d-y_n)\left[y^i\sum_{i=1}^{m}w^i-\sum_{i=1}^{m}(w^iy^i)\right]\cdot 2(x_j-c_j^i)w^i\Big/\left[b_j^i\left(\sum_{i=1}^{m}w^i\right)^2\right] \tag{8.24}$$

$$b_j^i(k)=b_j^i(k-1)+\beta(y_d-y_n)\left[y^i\sum_{i=1}^{m}w^i-\sum_{i=1}^{m}(w^iy^i)\right]\cdot(x_j-c_j^i)^2w^i\Big/\left[(b_j^i)^2\left(\sum_{i=1}^{m}w^i\right)^2\right] \tag{8.25}$$

式中，α，β 为学习速率。

8.2.3 仿真实例

使用混合型 pi-sigma 神经网络逼近对象

$$y(k)=u(k)^3+\frac{y(k-1)}{1+y(k-1)^2}$$

采样时间取 1ms。对象的输入信号为 $u(k)=0.5\sin(6\pi t)$。神经网络的输入为 2 个，即 $u(k)$ 和 $y(k)$。针对每个输入，采用 9 个模糊集进行模糊化，即 $m=9$。$\boldsymbol{p}_0$，$\boldsymbol{p}_1$，$\boldsymbol{p}_2$ 的初始值取值为0.1的列向量。网络学习参数取 $\eta=0.85$，$\alpha=0.05$。

混合型 pi-sigma 神经网络逼近程序见附录程序 chap8_2.m。仿真结果如图 8-6 和图 8-7 所示。

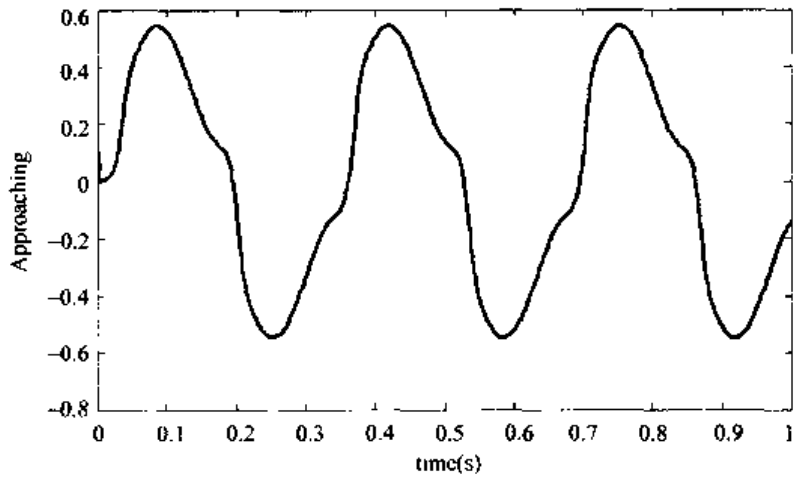


图 8-6 混合型 pi-sigma 神经网络逼近效果

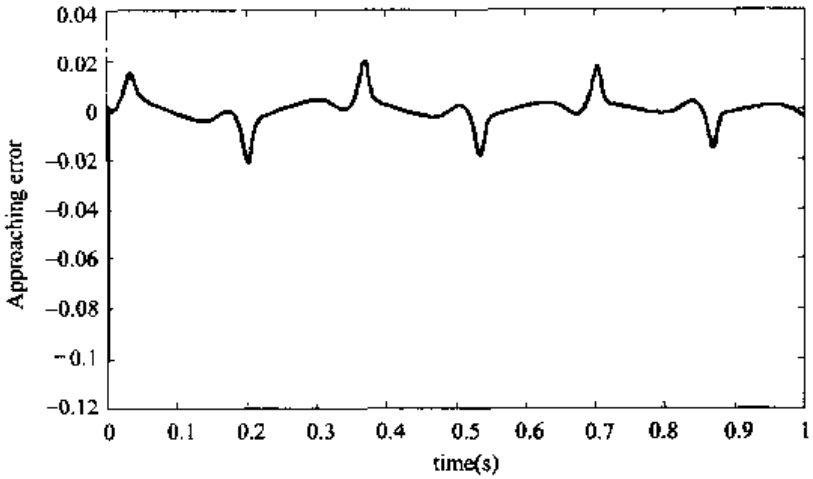


图 8-7　混合型 pi-sigma 神经网络逼近误差

8.3　小脑模型神经网络

8.3.1　CMAC 概述

小脑模型神经网络(Cerebellar Model Articulation Controller,CMAC) 是一种表达复杂非线性函数的表格查询型自适应神经网络,该网络可通过学习算法改变表格的内容,具有信息分类存储的能力。

CMAC 把系统的输入状态作为一个指针,把相关信息分布式地存入一组存储单元。它本质上是一种用于映射复杂非线性函数的查表技术。具体做法是将输入空间分成许多分块,每个分块指定一个实际存储器位置;每个分块学习到的信息分布式地存储到相邻分块的位置上;存储单元数通常比所考虑问题的最大可能输入空间的分块数少得多,故实现的是多对一的映射,即多个分块映射到同样一个存储器地址上。

CMAC 已被公认为是一类联想记忆神经网络的重要组成部分,它能够学习任意多维非线性映射。CMAC 算法可有效地用于非线性函数逼近、动态建模、控制系统设计等。CMAC 较其他神经网络的优越性体现在:

(1) 它是基于局部学习的神经网络,它把信息存储在局部结构上,使每次修正的权很少,在保证函数非线性逼近性能的前提下,学习速度快,适合于实时控制;

(2) 具有一定的泛化能力,即所谓相近输入产生相近输出,不同输入给出不同输出;

(3) 连续(模拟) 输入、输出能力;

(4) 寻址编程方式,在利用串行计算机仿真时,它可使回响速度加快;

(5) 作为非线性逼近器,它对学习数据出现的次序不敏感。

由于 CMAC 所具有的上述优越性能,使它比一般神经网络具有更好的非线性逼近能力,更适合于复杂动态环境下的非线性实时控制。

CMAC 的基本思想在于:在输入空间中给出一个状态,从存储单元中找到对应于该状态的地址,将这些存储单元中的内容求和得到 CMAC 的输出;将此响应值与期望输出值进行比较,并根据学习算法修改这些已激活的存储单元的内容。

CMAC 的结构如图 8-8 所示。

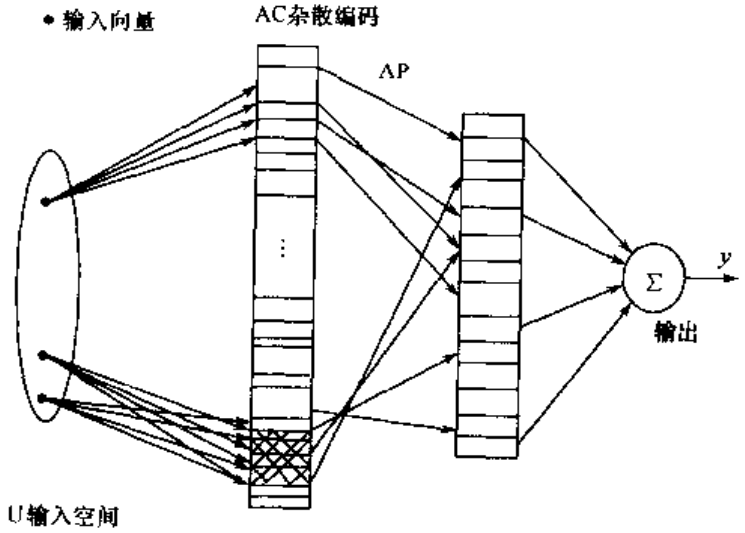


图 8-8 CMAC 结构

8.3.2 一种典型 CMAC 算法

CMAC 网络由输入层、中间层和输出层组成。在输入层与中间层、中间层与输出层之间分别为由设计者预先确定的输入层非线性映射和输出层权值自适应性线性映射。

在输入层，对 n 维输入空间进行划分。中间层由若干个基函数构成，对任意一个输入只有少数几个基函数的输出为非零值，称非零输出的基函数为作用基函数，作用基函数的个数为泛化参数 c，它规定了网络内部影响网络输出的区域大小。

中间层基函数的个数用 p 表示，泛化参数 c 满足 $c \ll p$。在中间层的基函数与输出层的网络输出之间通过连接权进行连接。采用梯度下降法实现权值的调整。

CMAC 神经网络的设计主要包括输入空间的划分、输入层至输出层非线性映射的实现及输出层权值学习算法。CMAC 是前馈网络，输入、输出之间的非线性关系由以下两个基本映射实现。

1. 概念映射(U → AC)

概念映射是从输入空间 U 至概念存储器 AC 的映射。

设输入空间向量为 $\boldsymbol{u}_p = [u_{1p}, u_{2p}, \cdots, u_{np}]^{\mathrm{T}}$，量化编码为 $[u_p]$，输入空间映射至 AC 中 c 个存储单元(c 为二进制非零单元的数目)。

采用下式表示映射后的向量

$$\boldsymbol{R}_p = \mathrm{S}([u_p]) = [s_1(u_p), s_2(u_p), \cdots, s_c(u_p)]^{\mathrm{T}} \tag{8.26}$$

式中，$s_j([u_p]) = 1, j = 1, 2, \cdots, c$。

映射原则为：在输入空间邻近的两个点(一个点表示单输入的 n 维向量)，在 AC 中有部分的重叠单元被激励。距离越近，重叠越多；距离越远，重叠越少。这种映射称为局部泛化，c 为泛化参数。

2. 实际映射(AC → AP)

实际映射是由概念存储器 AC 中的 c 个单元，用编码技术(如杂散编码)映射至实际存储器

AP 的 c 个单元，c 个单元中存放着相应权值。网络的输出为 AP 中 c 个单元的权值的和。

若只考虑单输出，则输出为

$$y = \sum_{j=1}^{c} w_j s_j([u_p]) \tag{8.27}$$

即

$$y = \sum_{j=1}^{c} w_j \tag{8.28}$$

CMAC 采用的学习算法如下：采用 δ 学习规则调整权值，权值调整指标为

$$E = \frac{1}{2c} e(t)^2 \tag{8.29}$$

式中，$e(t) = r(t) - y(t)$。

由梯度下降法，权值按下式调整

$$\Delta w_j(t) = -\eta \frac{\partial E}{\partial w_j} = \eta \frac{(r(t) - y(t))}{c} \frac{\partial y}{\partial w_j} = \eta \frac{e(t)}{c} \tag{8.30}$$

$$w_j(t) = w_j(t-1) + \Delta w_j(t) + \alpha(w_j(t-1) - w_j(t-2)) \tag{8.31}$$

$$w_p = [w_1 \ w_2 \ \cdots \ w_c]^{\mathrm{T}} \tag{8.32}$$

式中，α 为惯性系数。

8.3.3　仿真实例

采用 CMAC 网络逼近非线性对象

$$y(k) = u(k-1)^3 + y(k-1)/(1 + y(k-1)^2)$$

取 $u(k)$ 作为网络的输入，采用线性化函数对输入状态进行量化，实现 CMAC 的概念映射

$$s(k) = \mathrm{round}\left((u(k) - x_{\min}) \frac{M}{x_{\max} - x_{\min}}\right)$$

式中，$x_{\min}$ 和 $x_{\max}$ 为输入的最大、最小值，M 为 $x_{\max}$ 量化后所对应的最大值，round( ) 为四舍五入的 Matlab 函数。

采用杂散编码技术中的除留余数法实现 CMAC 的实际映射。设杂凑表长为 m（m 为正整数），以元素值 $s(k) + i$ 除以某数 N（$N \leqslant m$）后所得余数 $+1$ 作为杂凑地址，实现了实际映射，即

$$ad(i) = (s(k) + i \mathrm{MOD}\ N) + 1$$

式中，$i = 1, 2, \cdots, c$。

在仿真中，取 $M = 100$，$N = 7$，泛化参数 $c = 7$，$\eta = 1.5$，$\alpha = 0.05$。CMAC 网络逼近程序见附录程序 chap8_3.m。仿真结果如图 8-9 所示。

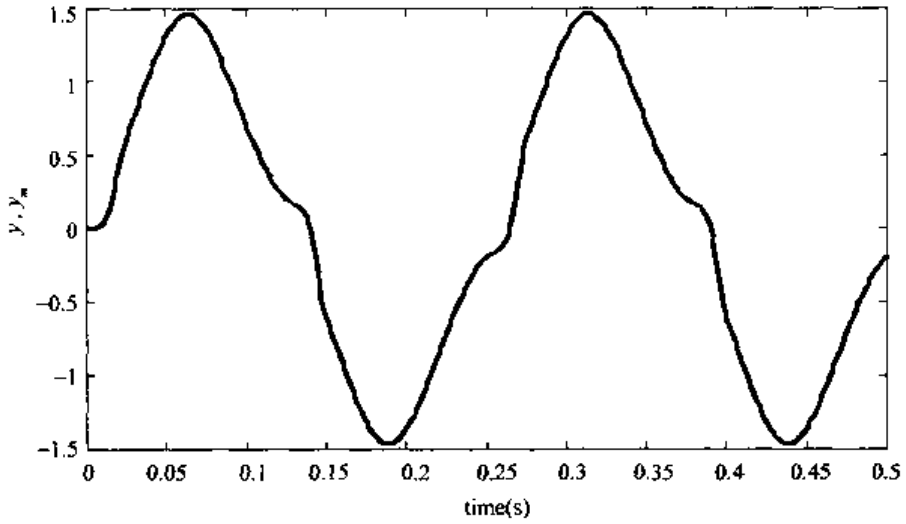


图 8-9 CMAC 的逼近

8.4 Hopfield 网络

8.4.1 Hopfield 网络原理

1986 年美国物理学家 J. J. Hopfield 利用非线性动力学系统理论中的能量函数方法研究反馈人工神经网络的稳定性，提出了 Hopfield 神经网络，并建立了求解优化计算问题的方程。

基本的 Hopfield 神经网络是一个由非线性元件构成的全连接型单层反馈系统，Hopfield 网络中的每一个神经元都将自己的输出通过连接权传送给所有其他神经元，同时又都接收所有其他神经元传递过来的信息。Hopfield 神经网络是一个反馈型神经网络，网络中的神经元在 t 时刻的输出状态实际上间接地与自己的 $t-1$ 时刻的输出状态有关，其状态变化可以用差分方程来描述。反馈型网络的一个重要特点就是它具有稳定状态，当网络达到稳定状态时，也就是它的能量函数达到最小的时候。

Hopfield 网络分离散型和连续型两种，本书介绍的为连续型 Hopfield 网络。

Hopfield 神经网络的能量函数不是物理意义上的能量函数，而是在表达形式上与物理意义上的能量概念一致，表征网络状态的变化趋势，并可以依据 Hopfield 工作运行规则不断进行状态变化，最终能够达到的某个极小值的目标函数。网络收敛就是指能量函数达到极小值。如果把一个最优化问题的目标函数转换成网络的能量函数，把问题的变量对应于网络的状态，那么 Hopfield 神经网络就能够用于解决优化组合问题。

Hopfield 神经网络工作时，各个神经元的连接权值是固定的，更新的只是神经元的输出状态。Hopfield 神经网络的运行规则为：首先从网络中随机选取一个神经元 u_i 进行加权求和，再计算 u_i 的第 $t+1$ 时刻的输出值。除 u_i 以外的所有神经元的输出值保持不变，直至网络进入稳定状态。

Hopfield 神经网络模型是由一系列互连的神经元组成的反馈型网络，如图 8-10 所示，其中虚线框内为一个神经元，u_i 为第 i 个神经元的状态输入，R_i 与 C_i 分别为输入电阻和输入电容，I_i 为输入电流，w_{ij} 为第 j 个神经元到第 i 个神经元的连接权值。v_i 为神经元的输出，是神经元状态变量 u_i 的非线性函数。

对于 Hopfield 神经网络的第 i 个神经元，采用微分方程建立其输入、输出关系，即

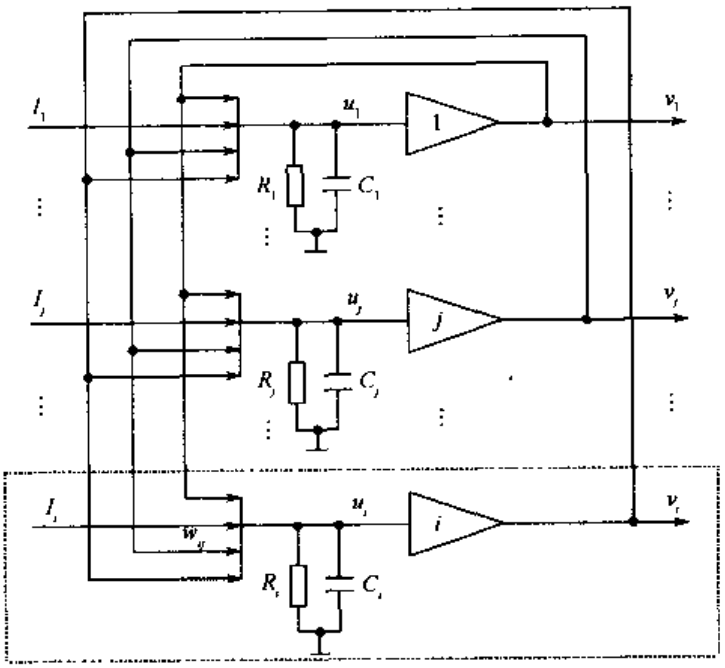


图 8-10　Hopfield 神经网络模型

$$\begin{cases} C_i \dfrac{\mathrm{d}u_i}{\mathrm{d}t} = \sum\limits_{j=1}^{n} w_{ij} v_j - \dfrac{u_i}{R_i} + I_i \\ v_i = g(u_i) \end{cases} \tag{8.33}$$

式中，$i = 1,2,\cdots,n$。

函数 $g(\cdot)$ 为双曲函数，一般取为

$$g(x) = \rho \frac{1-\mathrm{e}^{-x}}{1+\mathrm{e}^{-x}} \tag{8.34}$$

式中，$\rho > 0$。

Hopfield网络的动态特性要在状态空间中考虑，分别令 $\boldsymbol{u} = [u_1, u_2, \cdots, u_n]^{\mathrm{T}}$ 为具有 n 个神经元的 Hopfield 神经网络的状态向量，$\boldsymbol{V} = [v_1, v_2, \cdots, v_n]^{\mathrm{T}}$ 为输出向量，$\boldsymbol{I} = [I_1, I_2, \cdots, I_n]^{\mathrm{T}}$ 为网络的输入向量。

为了描述 Hopfield 网络的动态稳定性，定义能量函数为

$$E = -\frac{1}{2}\sum_{i}\sum_{j} w_{ij} v_i v_j + \sum_{i} \frac{1}{R_i}\int_0^{v_i} g_i^{-1}(v)\mathrm{d}v + \sum_{i} I_i v_i \tag{8.35}$$

若权值矩阵 $\boldsymbol{W}$ 是对称的（$w_{ij} = w_{ji}$），则

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\partial E}{\partial v_i} \cdot \frac{\mathrm{d}v_i}{\mathrm{d}t} = -\sum_{i} \frac{\mathrm{d}v_i}{\mathrm{d}t}\left(\sum_{j} w_{ij} v_j - \frac{u_i}{R_i} + I_i\right) = -\sum_{i} \frac{\mathrm{d}v_i}{\mathrm{d}t}\left(C_i \frac{\mathrm{d}u_i}{\mathrm{d}t}\right) \tag{8.36}$$

由于 $v_i = g(u_i)$，则

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\sum_{i} C_i \frac{\mathrm{d}g^{-1}(v_i)}{\mathrm{d}v_i}\left(\frac{\mathrm{d}v_i}{\mathrm{d}t}\right)^2 \tag{8.37}$$

由于 $C_i > 0$，双曲函数是单调上升函数，显然它的反函数 $g^{-1}(v_i)$ 也为单调上升函数，即有 $\dfrac{\mathrm{d}g^{-1}(v_i)}{\mathrm{d}v_i} > 0$，则可得到 $\dfrac{\mathrm{d}E}{\mathrm{d}t} \leqslant 0$，即能量函数 E 具有负的梯度，当且仅当 $\dfrac{\mathrm{d}v_i}{\mathrm{d}t} = 0$ 时 $\dfrac{\mathrm{d}E}{\mathrm{d}t} = 0 (i = 1,2,\cdots,n)$。由此可见，随着时间的演化，网络的解在状态空间中总是朝着能量 E 减少的方向运动。网络最终输出向量 $\boldsymbol{V}$ 为网络的稳定平衡点，即 E 的极小点。

Hopfield 网络在优化计算中得到了成功应用，有效地解决了著名的旅行推销员问题（TSP

问题)。另外,Hopfield 网络在智能控制和系统辨识中也有广泛的应用。

8.4.2 基于 Hopfield 网络的自适应控制

1. 系统描述

被控对象为一阶系统

$$J\dot{\theta} = u \tag{8.38}$$

即

$$n = \frac{1}{J} \cdot u \tag{8.39}$$

式中,J 为转动惯量,u 为控制输入,实际速度为 $n = \dot{\theta}$。

取速度指令为 $n_d = \dot{\theta}_d$,将控制器设计为"P 控制 + 前馈控制" 的形式,即

$$u = k_p(n_d - n) + k_f n_d \tag{8.40}$$

将式(8.40) 代入式(8.38) 并整理,得

$$n = \frac{k_p}{J}(n_d - n) + \frac{k_f}{J}n_d = -\frac{k_p}{J}n + \left(\frac{k_p}{J} + \frac{k_f}{J}\right)n_d \tag{8.41}$$

令

$$K = \frac{1}{J}, F = k_p, G = k_p + k_f \tag{8.42}$$

则式(8.41) 变为

$$n = -KFn + KGn_d \tag{8.43}$$

式中,F 和 G 为待定的控制器参数,可采用 Hopfield 网络进行辨识。

2. 基于 Hopfield 网络的控制器优化

所采用的 Hopfield 网络结构如图 8-11 所示。

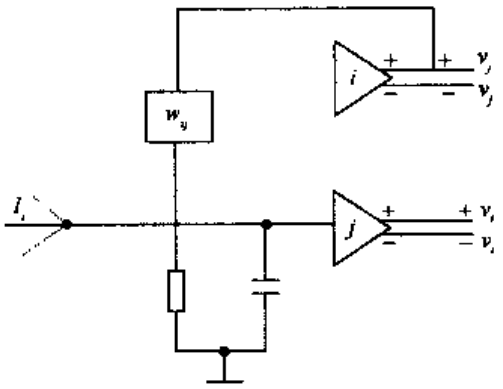


图 8-11 Hopfield 网络结构

Hopfield 网络的能量函数取

$$E = \frac{1}{2}(n_d - n)^2 \tag{8.44}$$

将式(8.43) 代入上式并展开得

$$E = \frac{1}{2}(n_d^2 + K^2F^2n^2 + K^2G^2n_d^2) + \frac{1}{2}(2KFn_dn - 2KGn_d^2 - 2K^2FGnn_d) \tag{8.45}$$

取 Hopfield 网络输出神经元数为 2，假设输入电阻无穷大，此时 Hopfield 网络的标准能量函数为

$$E_N = -\frac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{2} w_{ij} v_i v_j - \sum_{i=1}^{2} v_i I_i \tag{8.46}$$

将 E_N 展开得

$$E_N = -\frac{1}{2}(w_{11}v_1^2 + w_{12}v_1 v_2 + w_{21}v_2 v_1 + w_{22}v_2^2) - v_1 I_1 - v_2 I_2$$

令 $v_1 = F, v_2 = G$，得

$$E_N = -\frac{1}{2}(w_{11}F^2 + w_{12}FG + w_{21}FG + w_{22}G^2) - FI_1 - GI_2$$

当 Hopfield 网络处于平衡状态时，能量函数最小。由网络权值对称得 $w_{12} = w_{21}$。此时

$$\frac{\partial E_N}{\partial F} = \frac{\partial E}{\partial F} = 0$$

$$\frac{\partial E_N}{\partial G} = \frac{\partial E}{\partial G} = 0$$

$$\frac{\partial E_N}{\partial F} = \frac{1}{2}(-2w_{11}F - w_{12}G - w_{21}G) - I_1 = \frac{1}{2}(-2w_{11}F - 2w_{12}G) - I_1 = 0$$

$$\frac{\partial E}{\partial F} = \frac{1}{2}(2K^2Fn^2 + 2Kn_d n - 2K^2Gnn_d) = 0$$

由上面两式得

$$w_{11} = -K^2n^2, w_{12} = w_{21} = K^2nn_d, I_1 = -Kn_d n$$

$$\frac{\partial E_N}{\partial G} = \frac{1}{2}(-w_{12}F - w_{21}F - 2w_{22}G) - I_2 = \frac{1}{2}(-2w_{12}F - 2w_{22}G) - I_2 = 0$$

$$\frac{\partial E}{\partial G} = \frac{1}{2}(2K^2Gn_d^2 - 2Kn_d^2 - 2K^2Fnn_d) = 0$$

由上面两式得

$$I_2 = Kn_d^2, w_{22} = -K^2n_d^2$$

则连接权矩阵 $\boldsymbol{W}$ 和外部输入 $\boldsymbol{I}$ 为

$$\boldsymbol{W} = \begin{bmatrix} -K^2n^2 & K^2nn_d \\ K^2nn_d & -K^2n_d^2 \end{bmatrix}, \boldsymbol{I} = \begin{bmatrix} -Kn_d n & Kn_d^2 \end{bmatrix} \tag{8.47}$$

标准 Hopfield 网络的动态方程为

$$\begin{aligned} C_i \frac{\mathrm{d}u_i}{\mathrm{d}t} &= \sum_{j=1}^{n} w_{ij} v_i + I_i \\ v_i &= g(u_i) \end{aligned} \tag{8.48}$$

取 $C_i = 1.0$，将所求的 $\boldsymbol{W}$ 和 $\boldsymbol{I}$ 代入上式得

$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = w_{11}v_1 + w_{12}v_2 + I_1$$

$$\frac{\mathrm{d}u_2}{\mathrm{d}t} = w_{21}v_1 + w_{22}v_2 + I_2$$

取神经元输出的非线性特性为双曲函数，即

$$g(u_i) = \frac{2S_i}{1 + \mathrm{e}^{-\lambda_i u_i}} - S_i \tag{8.49}$$

式中，$i=1,2;S_i>0$。

网络实际输出为

$$F=g(u_1)$$

$$G=g(u_2)$$

由于 $1+e^{-\lambda_1 u_1}=\frac{2S_1}{F+S_1}$，$1+e^{-\lambda_2 u_2}=\frac{2S_2}{G+S_2}$，则

$$\frac{dF}{du_1}=\frac{-2S_1 e^{-\lambda_1 u_1}(-\lambda_1)}{(1+e^{-\lambda_1 u_1})^2}=2\lambda_1 S_1\frac{S_1-F}{F+S_1}\frac{(F+S_1)^2}{(2S_1)^2}=\frac{\lambda_1(S_1^2-F^2)}{2S_1}$$

$$\frac{dF}{dt}=\frac{dF}{du_1}\frac{du_1}{dt}=\frac{\lambda_1(S_1^2-F^2)}{2S_1}\frac{du_1}{dt} \tag{8.50}$$

同理可得

$$\frac{dG}{du_2}=\frac{\lambda_2(S_2^2-G^2)}{2S_2}$$

$$\frac{dG}{dt}=\frac{dG}{du_2}\frac{du_2}{dt}=\frac{\lambda_2(S_2^2-G^2)}{2S_2}\frac{du_2}{dt} \tag{8.51}$$

求解微分方程式(8.50)和式(8.51)，可得到优化后的 F,G，从而实现 k_p 和 k_I 的整定。

3. 仿真实例

被控对象为一阶系统

$$J\dot{\theta}=u$$

式中，$J=\frac{1}{10}$，$K=\frac{1}{J}=10$。速度指令取 $0.5\sin(2\pi t)$。Hopfield 神经网络中，取 $\lambda_1=\lambda_2=5$，$S_1=S_2=50$。

Hopfield 网络的自适应控制程序包括主程序 chap8_4sim.mdl、控制器 S 函数程序 chap8_4s.m、被控对象 S 函数程序 chap8_4plant.m 和作图程序 chap8_4plot.m，见附录。

仿真结果如图 8-12 至图 8-16 所示，图 8-12 和图 8-13 分别为速度跟踪及跟踪误差，图 8-14 为控制信号的输入，图 8-15 和图 8-16 为控制器参数 k_p 和 k_I 的变化过程。

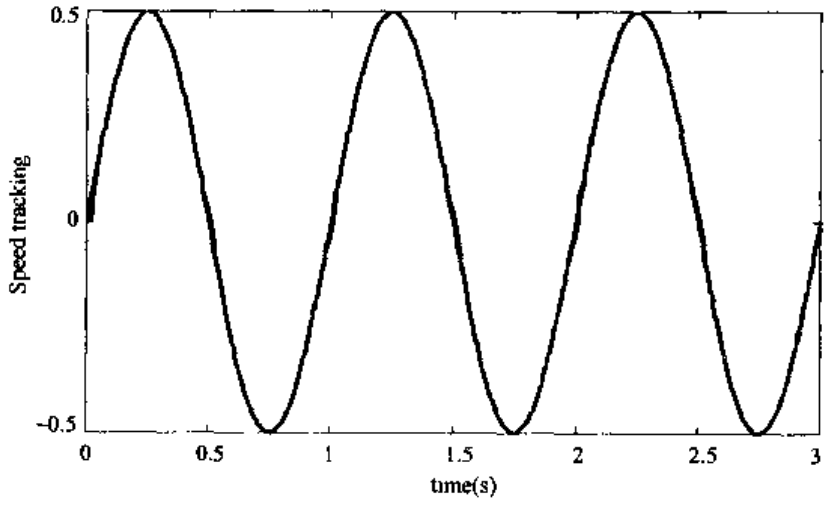


图 8-12　速度跟踪

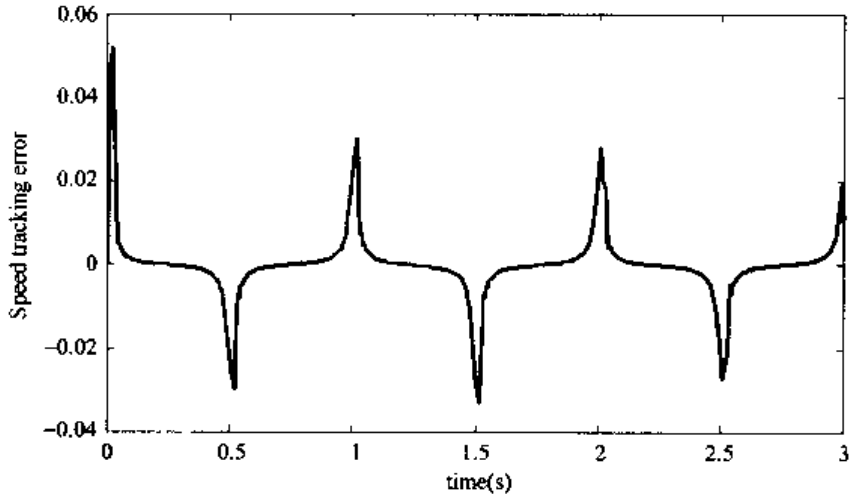


图 8-13　速度跟踪误差

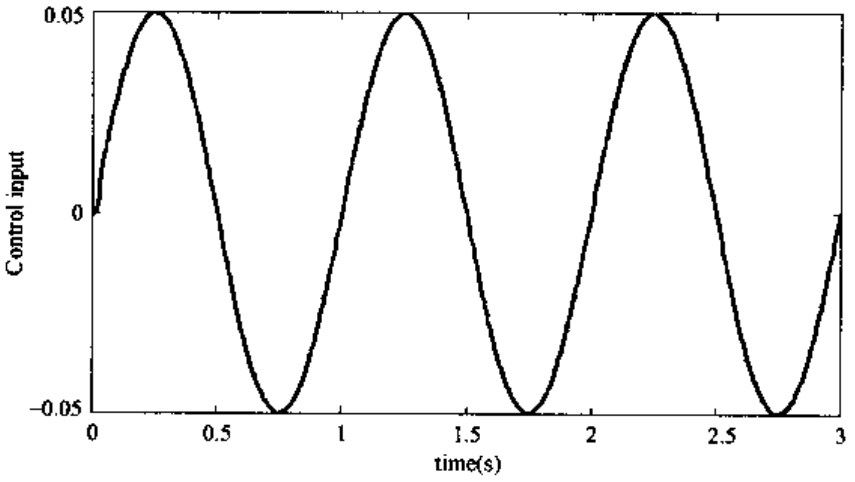


图 8-14　控制信号的输入

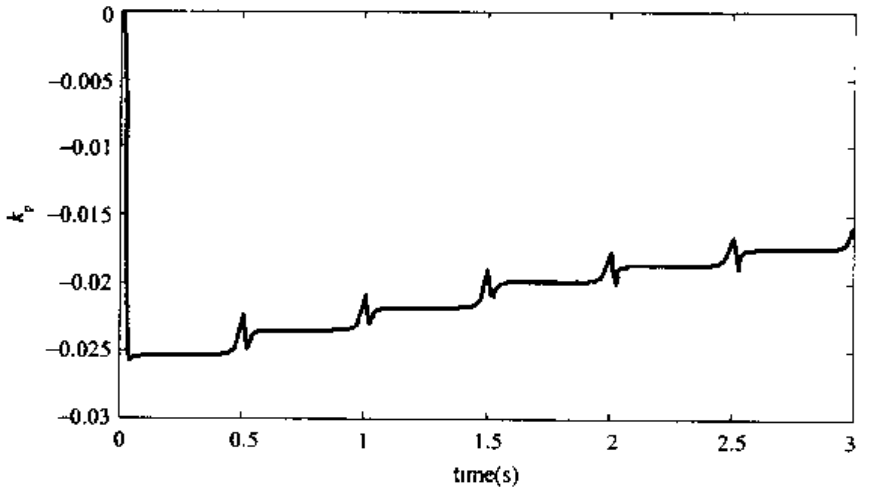


图 8-15　k_p 的变化过程

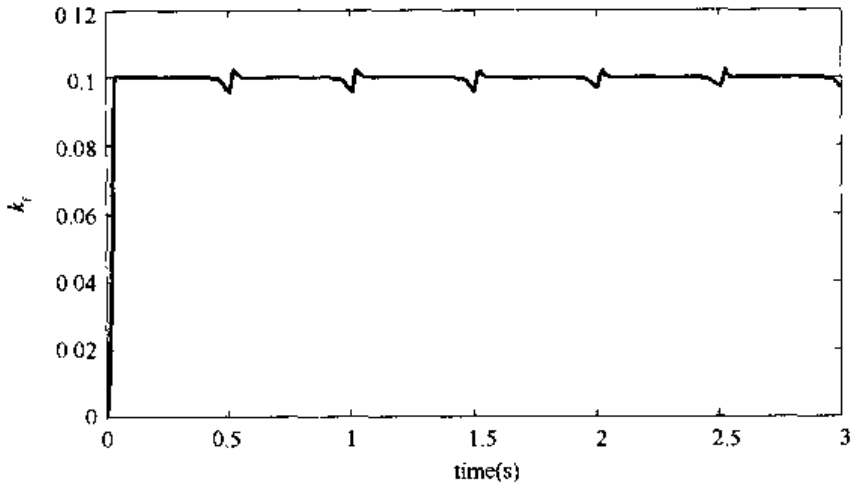


图 8-16 k_I 的变化过程

思考题与习题

8-1 采用模糊 RBF 网络、pi-sigma 神经网络、CMAC 网络逼近非线性对象 $y(k)=(u(k-1)-0.9y(k-1))/(1+y(k-1)^2)$，分别进行 Matlab 仿真。

8-2 在 Hopfield 网络自适应控制中，将被控对象改为一阶系统

$$J\dot{\theta}=au+b$$

其中，$J=\frac{1}{10}$，$a=10$，$b=30$，速度指令取 $0.5\sin(2\pi t)$。控制器采用“P 控制＋前馈控制”的形式，试采用 Hopfield 网络进行控制器的优化设计，并进行 Matlab 仿真。

附录（程序代码）

模糊 RBF 网络逼近程序：chap8_1.m

```
%Fuzzy RBF Approaching
clear all;
close all;

xite=0.20;
alfa=0.05;

b=5*ones(5,1);
c=[-5 -2 0 2 5;
   -5 -2 0 2 5];
w=rands(25,1);

c_1=c;
c_2=c_1;

b_1=b;
b_2=b_1;

w_1=w;
w_2=w_1;

u_1=0.0;
y_1=0.0;

ts=0.001;
for k=1:1:1000
time(k)=k*ts;

u(k)=0.5*sin(6*pi*k*ts);
y(k)=u_1^3+y_1/(1+y_1^2);

x=[u(k),y(k)]';                    % Layer1:input
f1=x;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:1:2                        % Layer2:fuzzation
   for j=1:1:5
      net2(i,j)=-(f1(i)-c(i,j))^2/b(j)^2;
```

```
    end
end
for i=1:1:2
    for j=1:1:5
    f2(i,j)=exp(net2(i,j));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:1:5                              % Layer3:fuzzy inference(49 rules)
     m1(j)=f2(1,j);
      m2(j)=f2(2,j);
end

for i=1:1:5
     for j=1:1:5
     ff3(i,j)=m2(i)*m1(j);
     end
end
f3=[ff3(1,:),ff3(2,:),ff3(3,:),ff3(4,:),ff3(5,:)];
%%%%%%%%%%%%%%%%%%%%%%%%%

f4=w_1'*f3';                            % Layer4:output
ym(k)=f4;

e(k)=y(k)-ym(k);

d_w=0*w_1;
for j=1:1:25
     d_w(j)=xite*e(k)*f3(j);
end
w=w_1+d_w+alfa*(w_1-w_2);

delta2=-e(k)*w'*f3';;

d_b=0*b_1;
for j=1:1:5
     d_b(j)=xite*delta2*2*((x(1)-c(1,j))^2)*(b(j)^-3);
end
b=b_1+d_b+alfa*(b_1-b_2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
d_c=0 * c_1;
for i=1:1:2
    for j=1:1:5
        d_c(i,j)=-xite*delta2*2*(x(i)-c(i,j))*b(j)^-2;
    end
end
c=c_1+d_c+alfa*(c_1-c_2);

   u_1=u(k);
   y_1=y(k);

   w_2=w_1;
   w_1=w;

   c_2=c_1;
   c_1=c;

   b_2=b_1;
   b_1=b;
end
figure(1);
plot(time,y,'r',time,ym,'b');
xlabel('time(s)');ylabel('Approaching');
figure(2);
plot(time,y-ym,'r');
xlabel('time(s)');ylabel('Approaching error');
```

混合型 pi-sigma 神经网络逼近程序：chap8_2.m

```
% Pi-Sigma Type Fuzzy Neural Network
clear all;
close all;

xite=0.85;
alfa=0.05;

p0=0.1*ones(9,1);
p1=0.1*ones(9,1);
p2=0.1*ones(9,1);

p0_1=p0;p0_2=p0_1;
```

```
p1_1=p1;p1_2=p1_1;
p2_1=p2;p2_2=p2_1;

c0=[-5,0,5];
b0=[3,3,3];
for i=1:1:3
for j=1:1:3
m=3*(i-1)+j;
c(m,1)=c0(i);
c(m,2)=c0(j);

b(m,1)=b0(i);
b(m,2)=b0(j);
end
end

c_1=c;
c_2=c_1;
b_1=b;
b_2=b_1;

yd_1=0;
u_1=0;

ts=0.001;
for k=1:1:1000
time(k)=k*ts;

u(k)=0.50*sin(6*pi*k*ts);
yd(k)=u_1^3+yd_1/(1+yd_1^2);

x(1)=u(k);
x(2)=yd(k);

for i=1:1:2
for j=1:1:9
    fz(i,j)=exp(-(x(i)-c(j,i))^2/b(j,i));
end
end
```

```
for i=1:1:9
        w(i)=fz(1,i)*fz(2,i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addw=0;
for i=1:1:9
    addw=addw+w(i);
end

for i=1:1:9
    yi(i)=p0_1(i)+p1_1(i)*x(1)+p2_1(i)*x(2);
end

addyw=0;
for i=1:1:9
    addyw=addyw+yi(i)*w(i);
end

yn(k)=addyw/addw;
e(k)=yd(k)-yn(k);

%Learning
d_p=0*p0;
for i=1:1:9
    d_p(i)=xite*e(k)*w(i)/addw;
end
   p0=p0_1+ d_p+alfa*(p0_1-p0_2);
   p1=p1_1+ d_p*x(1)+alfa*(p1_1-p1_2);
   p2=p2_1+ d_p*x(2)+alfa*(p2_1-p2_2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d_b=0*b_1;
for i=1:1:9
for j=1:1:2
d_b(i,j)=xite*e(k)*(yi(i)*addw-addyw)*(x(j)-c(i,j))^2*w(i)/(b(i,j)^2*ad-
dw^2);
end
end
   b=b_1+d_b+alfa*(b_1-b_2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
d_c=0*c_1;
for i=1:1:9
for j=1:1:2
d_c(i,j)=xite*e(k)*(yi(i)*addw-addyw)*2*(x(j)-c(i,j))*w(i)/(b(i,j)*addw^2);
end
end
c=c_1+d_c+alfa*(c_1-c_2);

p0_2=p0_1;p0_1=p0;
p1_2=p1_1;p1_1=p1;
p2_2=p2_1;p2_1=p2;

c_2=c_1;c_1=c;
b_2=b_1;b_1=b;

u_1=u(k);
yd_1=yd(k);
end
figure(1);
plot(time,yd,'r',time,yn,'b');
xlabel('time(s)');ylabel('Approaching');

figure(2);
plot(time,yd-yn,'r');
xlabel('time(s)');ylabel('Approaching error');
```

CMAC 网络逼近程序:chap8_3.m

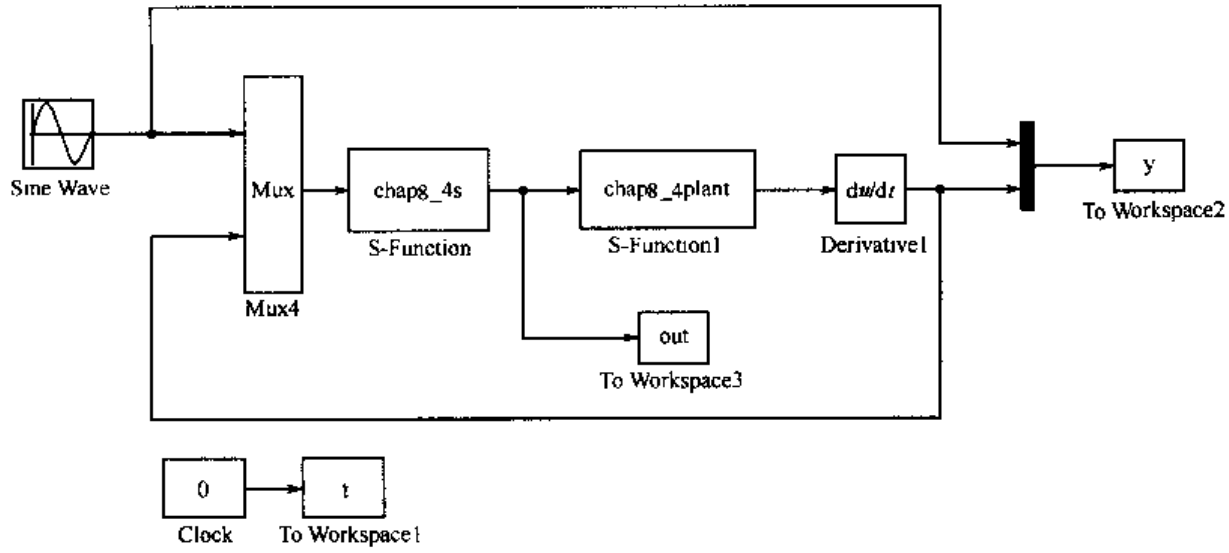
```
%CMAC Identification for nonlinear model
clear all;
close all;

xite=1.5;
alfa=0.05;

M=100;
N=7;
C=7;

w=zeros(N,1);
```

```
% w=rands(N,1);
w_1=w;
w_2=w;
d_w=w;

u_1=0;
y_1=0;
ts=0.001;

for k=1 : 1 : 500
time(k)=k * ts;

u(k)=sin(4 * 2 * pi * k * ts);

xmin=-1.0;
xmax=1.0;

s(k)=round((u(k)-xmin) * M/(xmax-xmin));      % Quantity

sum=0;
for i=1 : 1 : C
   ad(i)=mod(s(k)+i,N)+1;     % Table mapping and Hash transfer:Start address
   sum=sum+w(ad(i));
end

ym(k)=sum;

 % Nonlinear model
y(k)=u_1^3+y_1/(1+y_1^2);

error(k)=y(k)-ym(k);
d_w=xite * error(k)/C;

for i=1 : 1 : C
   ad(i)=mod(s(k)+i,N)+1;
   w(ad(i))=w_1(ad(i))+ d_w+alfa * (w_1(ad(i))-w_2(ad(i)));
end

% % % % Parameters Update % % % %
w_2=w_1;
```

```
w_1=w;

u_1=u(k);
y_1=y(k);
end
figure(1);
plot(time,y,'b',time,ym,'r');
xlabel('time(s)');ylabel('y,ym');
```

Hopfield 网络的自适应控制程序:包括以下 4 个程序。

(1)**主程序**:chap8_4sim. mdl



(2)**控制器 S 函数程序**:chap8_4s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

```
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0;0];
str = [];
ts = [];

function sys=mdlDerivatives(t,x,u)
F=x(1);
G=x(2);

nd=u(1);
n=u(2);

nmn1=5;
nmn2=5;

S1=50;
S2=50;

K=10;

w11=-K^2*n^2;
w12=K^2*n*nd;
w21=w12;
w22=-K^2*nd^2;

I1=-K*nd*n;
I2=K*nd^2;

du1=w11*F+w12*G+I1;
du2=w21*F+w22*G+I2;

sys(1)=nmn1/(2*S1)*(S1^2-F^2)*du1;
```

```
sys(2)=nmn2/(2*S2)*(S2^2-G^2)*du2;
function sys=mdlOutputs(t,x,u)

F=x(1);
G=x(2);

nd=u(1);
n=u(2);
ut=-F*n+G*nd;

kp=F;
kf=G-kp;

sys(1)=ut;
sys(2)=kp;
sys(3)=kf;
```

(3)**被控对象 S 函数程序**:chap8_4plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
```

```
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0];
str = [];
ts = [];

function sys=mdlDerivatives(t,x,u)
K=10;
ut=u(1);

sys(1)=K*ut;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
```

(4)**作图程序**:chap8_4plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('speed tracking');

figure(2);
plot(t,y(:,1)-y(:,2),'r');
xlabel('time(s)');ylabel('speed tracking error');

u=out(:,1);
kp=out(:,2);
kf=out(:,3);

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');

figure(4);
plot(t,kp,'r');
xlabel('time(s)');ylabel('kp');

figure(5);
plot(t,kf,'r');
xlabel('time(s)');ylabel('kf');
```

第9章　神经网络控制

9.1　概述

神经网络是一种具有高度非线性的连续时间动力系统，它有着很强的自学习能力和对非线性系统的强大映射能力，已广泛应用于复杂对象的控制中。神经网络所具有的大规模并行性、冗余性、容错性、本质的非线性及自组织、自学习、自适应能力，给不断面临挑战的控制理论带来生机。

控制理论在经历了经典控制和现代控制以后，随着被控对象变得越来越复杂、控制精度越来越高、对对象和环境的知识知道得越来越少，迫切希望控制系统具有自适应自学习能力、良好的鲁棒性和实时性。

神经网络的智能处理能力及控制系统所面临的愈来愈严重的挑战是神经网络控制的发展动力。由于神经网络本身具备传统的控制手段无法实现的一些优点和特征，使得神经网络控制器的研究迅速发展。从控制角度来看，神经网络用于控制的优越性主要表现为：

(1)神经网络能处理那些难以用模型或规则描述的对象；

(2)神经网络采用并行分布式信息处理方式，具有很强的容错性；

(3)神经网络在本质上是非线性系统，可以实现任意非线性映射，神经网络在非线性控制系统中具有很大的发展前途；

(4)神经网络具有很强的信息综合能力，它能够同时处理大量不同类型的输入，能够很好地解决输入信息之间的互补性和冗余性问题；

(5)神经网络的硬件实现愈趋方便，大规模集成电路技术的发展为神经网络的硬件实现提供了技术手段，为神经网络在控制中的应用开辟了广阔的前景。

神经网络控制所取得的进展为：

(1)基于神经网络的系统辨识：可在已知常规模型结构的情况下，估计模型的参数；或利用神经网络的线性、非线性特性，建立线性、非线性系统的静态、动态、逆动态及预测模型。

(2)神经网络控制器：神经网络作为控制器，可实现对不确定系统或未知系统进行有效的控制，使控制系统达到所要求的动态、静态特性。

(3)神经网络与其他算法相结合：神经网络与专家系统、模糊逻辑、遗传算法等相结合可构成新型控制器。

(4)优化计算：在常规控制系统的设计中，常遇到求解约束优化问题，神经网络为这类问题提供了有效的途径。

(5)控制系统的故障诊断：利用神经网络的逼近特性，可对控制系统的各种故障进行模式识别，从而实现控制系统的故障诊断。

神经网络控制在理论和实践上，以下问题是研究的重点：

(1)神经网络的稳定性与收敛性问题；

(2)神经网络控制系统的稳定性与收敛性问题；

(3)神经网络学习算法的实时性；

(4)神经网络控制器和辨识器的模型和结构；

9.2　神经网络控制的结构

神经网络控制的研究随着神经网络理论研究的不断深入而不断发展起来。根据神经网络在控制器中的作用不同，神经网络控制器可分为两类：一类为神经控制，它是以神经网络为基础而形成的独立智能控制系统；另一类为混合神经网络控制，它是指利用神经网络学习和优化能力来改善传统控制的智能控制方法，如自适应神经网络控制等。

目前神经网络控制器尚无统一的分类方法。综合目前的各种分类方法，可将神经网络控制的结构归结为以下 7 类。

9.2.1　神经网络监督控制

通过对传统控制器进行学习，然后用神经网络控制器逐渐取代传统控制器的方法，称为神经网络监督控制。神经网络监督控制的结构如图 9-1 所示。神经网络控制器实际上是一个前馈控制器，它建立的是被控对象的逆模型。神经网络控制器通过对传统控制器的输出进行学习，在线调整网络的权值，使反馈控制输入 $u_p(t)$ 趋近于零，从而使神经网络控制器逐渐在控制作用中占据主导地位，最终取消反馈控制器的作用。一旦系统出现干扰，反馈控制器重新起作用。因此，这种前馈加反馈的监督控制方法，不仅可以确保控制系统的稳定性和鲁棒性，而且可有效地提高系统的精度和自适应能力。

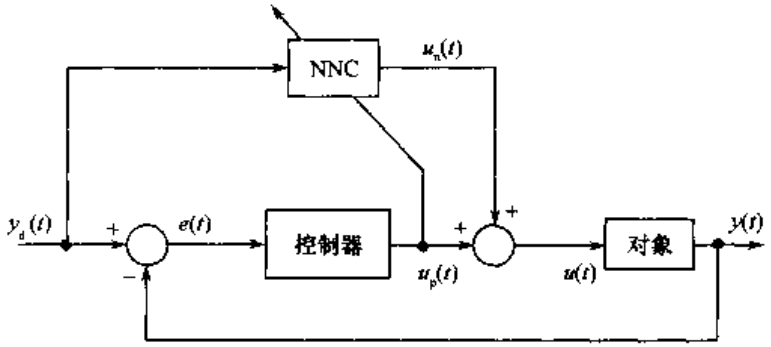


图 9-1　神经网络监督控制

9.2.2　神经网络直接逆控制

神经网络直接逆控制就是将被控对象的神经网络逆模型直接与被控对象串联起来，以便使期望输出与对象实际输出之间的传递函数为 1。则将此网络作为前馈控制器后，被控对象的输出为期望输出。

显然，神经网络直接逆控制的可用性在相当程度上取决于逆模型的准确精度。由于缺乏反馈，简单连接的直接逆控制缺乏鲁棒性。为此，一般应使其具有在线学习能力，即作为逆模型的神经网络连接权能够在线调整。

图 9-2 所示为神经网络直接逆控制的两种结构方案。在图 9-2(a)中，NN1 和 NN2 为具有完全相同的网络结构，并采用相同的学习算法、分别实现对象的逆。在图 9-2(b)中，神经网络 NN 通过评价函数进行学习，实现对象的逆控制。

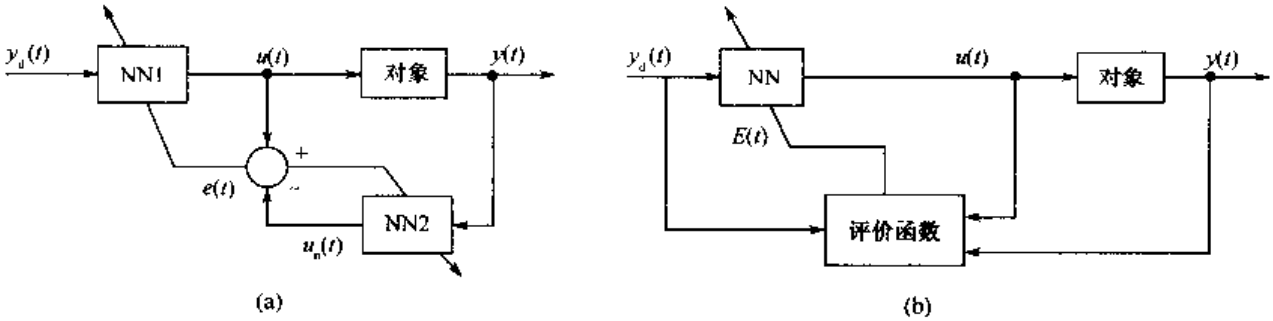


图 9-2　神经网络直接逆控制的两种结构方案

9.2.3　神经网络自适应控制

与传统自适应控制相同，神经网络自适应控制也分为神经网络自校正控制和神经网络模型参考自适应控制两种。自校正控制根据对系统正向或逆模型的结果调节控制器内部参数，使系统满足给定的指标，而在模型参考自适应控制中，闭环控制系统的期望性能由一个稳定的参考模型来描述。

1. 神经网络自校正控制

神经网络自校正控制分为直接自校正控制和间接自校正控制。间接自校正控制使用常规控制器，神经网络估计器需要较高的建模精度。直接自校正控制同时使用神经网络控制器和神经网络估计器。

(1)神经网络直接自校正控制

在本质上同神经网络直接逆控制，其结构如图 9-2 所示。

(2)神经网络间接自校正控制

其结构如图 9-3 所示。假设被控对象为如下单变量仿射非线性系统

$$y(t) = f(y_t) + g(y_t)u(t)$$

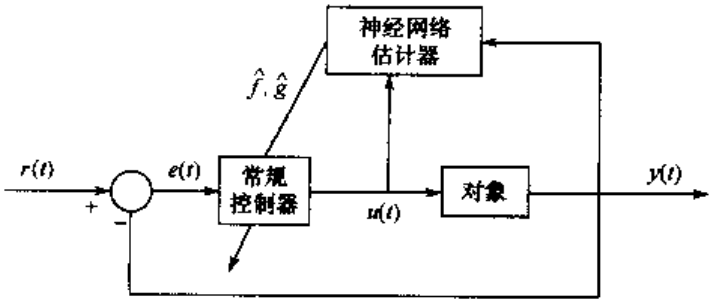


图 9-3　神经网络间接自校正控制

若利用神经网络对非线性函数 $f(y_t)$ 和 $g(y_t)$ 进行逼近，得到 $\hat{f}(y_t)$ 和 $\hat{g}(y_t)$，则常规控制器为

$$u(t) = [r(t) - \hat{f}(y_t)]/\hat{g}(y_t)$$

式中，$r(t)$ 为 t 时刻的期望输出值。

2. 神经网络模型参考自适应控制

分为直接模型参考自适应控制和间接模型参考自适应控制两种。

(1) 直接模型参考自适应控制

如图 9-4 所示。神经网络控制器的作用是使被控对象与参考模型输出之差为最小。但该方法需要知道对象的 Jacobian 信息$\frac{\partial y}{\partial u}$。

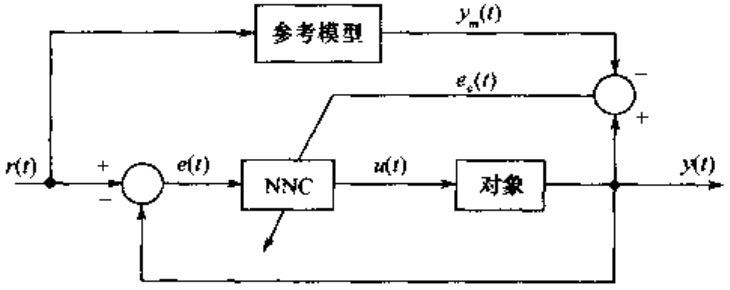


图 9-4　神经网络直接模型参考自适应控制

(2) 间接模型参考自适应控制

如图 9-5 所示。神经网络辨识器 NNI 向神经网络控制器 NNC 提供对象的 Jacobian 信息，用于控制器 NNC 的学习。

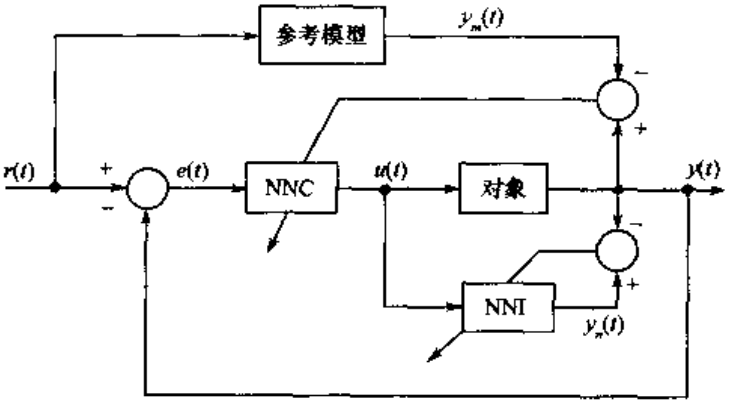


图 9-5　神经网络间接模型参考自适应控制

9.2.4　神经网络内模控制

经典的内模控制将被控系统的正向模型和逆模型直接加入反馈回路，系统的正向模型作为被控对象的近似模型与实际对象并联，两者输出之差被用做反馈信号，该反馈信号又经过前向通道的滤波器及控制器进行处理。控制器直接与系统的逆有关，通过引入滤波器来提高系统的鲁棒性。图 9-6 所示为神经网络内模控制，被控对象的正向模型及控制器均由神经网络来实现。

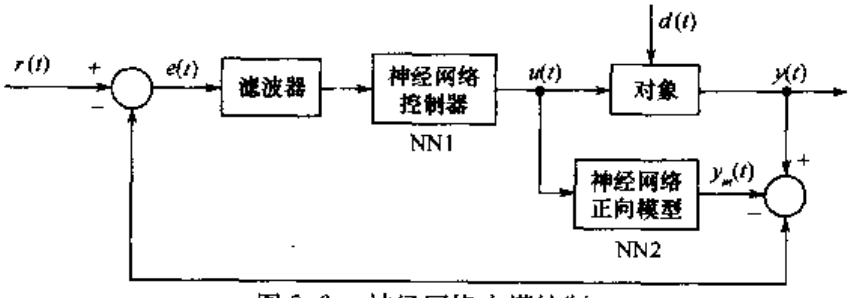


图 9-6　神经网络内模控制

9.2.5 神经网络预测控制

预测控制又称为基于模型的控制，是 20 世纪 70 年代后期发展起来的一类新型计算机控制方法，该方法的特征是预测模型、滚动优化和反馈校正。

神经网络预测控制的结构如图 9-7 所示，神经网络预测器建立了非线性被控对象的预测模型，并可在线进行学习修正。利用此预测模型，可以由当前的系统控制信息预测出在未来一段时间$(t+k)$范围内的输出值$\hat{y}(t+k)$。通过设计优化性能指标，利用非线性优化器可求出优化的控制作用$u(t)$。

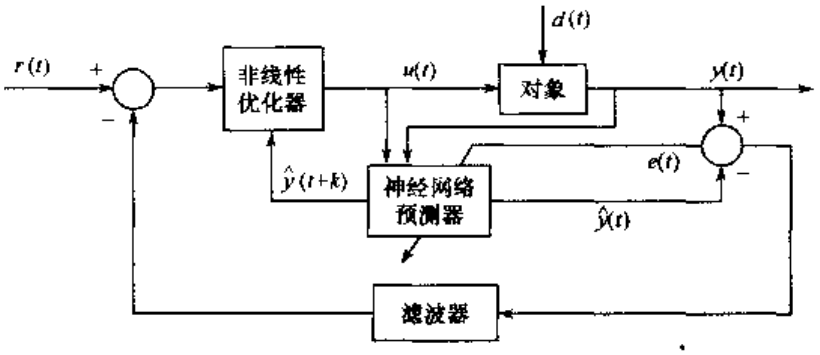


图 9-7 神经网络预测控制

9.2.6 神经网络自适应评判控制

神经网络自适应评判控制通常由两个网络组成，如图 9-8 所示。其中自适应评判网络在控制系统中相当于一个需要进行再励学习的“教师”，它通过不断的奖励、惩罚等再励学习，使自己逐渐成为一个合格的“教师”，学习完成后，根据系统目前的状态和外部再励反馈信号$r(t)$产生一个内部再励信号$\hat{r}(t)$，以对目前的控制效果做出评价。控制选择网络相当于一个在内部再励信号$\hat{r}(t)$指导下进行学习的多层前馈神经网络控制器，该网络进行学习后，根据编码后的系统状态，再允许控制集中选择下一步的控制作用。

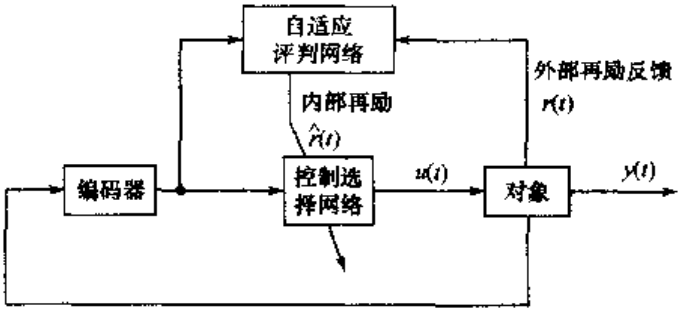


图 9-8 神经网络自适应评判控制

9.2.7 神经网络混合控制

该控制方法是集成人工智能各分支的优点，由神经网络技术与模糊控制、专家系统等相结合而形成的一种具有很强学习能力的智能控制系统。其中，由神经网络和模糊控制相结合构成

模糊神经网络，由神经网络和专家系统相结合构成神经网络专家系统。神经网络混合控制可使控制系统同时具有学习、推理和决策能力。

9.3　单神经元自适应控制

9.3.1　单神经元自适应控制算法

单神经元自适应控制的结构如图 9-9 所示。

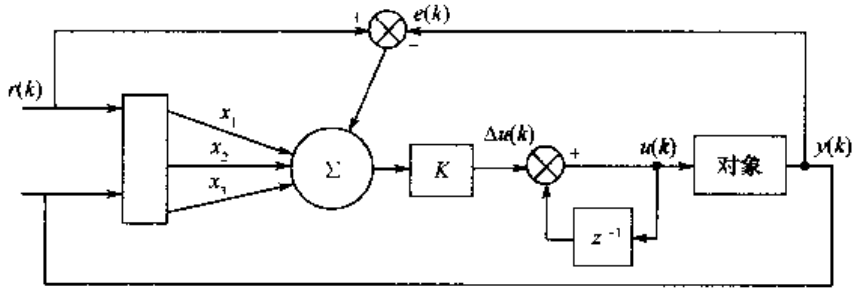


图 9-9　单神经元自适应控制结构

单神经元自适应控制器是通过对加权系数的调整来实现自适应、自组织功能，控制算法为

$$u(k)=u(k-1)+K\sum_{i=1}^{3}w_i(k)x_i(k) \tag{9.1}$$

如果权系数的调整按有监督的 Hebb 学习规则实现，即在学习算法中加入监督项 $z(k)$，则神经网络权值学习算法为

$$\left.\begin{aligned}w_1(k)&=w_1(k-1)+\eta z(k)u(k)x_1(k)\\w_2(k)&=w_2(k-1)+\eta z(k)u(k)x_2(k)\\w_3(k)&=w_3(k-1)+\eta z(k)u(k)x_3(k)\end{aligned}\right\} \tag{9.2}$$

式中，$z(k)=e(k)$，$x_1(k)=e(k)$，$x_2(k)=e(k)-e(k-1)$，$x_3(k)=\Delta^2e(k)=e(k)-2e(k-1)+e(k-2)$，$\eta$ 为学习速率，K 为神经元的比例系数，$K>0$。

K 值的选择非常重要。K 越大，则快速性越好，但超调量大，甚至可能使系统不稳定。当被控对象时延增大时，K 值必须减少，以保证系统稳定。K 值选择过小，会使系统的快速性变差。

9.3.2　仿真实例

被控对象为

$$y(k)=0.368y(k-1)+0.26y(k-2)+0.10u(k-1)+0.632u(k-2)$$

输入指令为一方波信号：$r(k)=0.5\mathrm{sgn}(\sin(4\pi t))$，采样时间为 1ms，采用有监督的 Hebb 学习规则实现权值的学习，初始权值取 $\boldsymbol{W}=[w_1\quad w_2\quad w_3]=[0.1\quad 0.1\quad 0.1]$，$\eta=0.40$，$K=0.12$。单神经元自适应控制程序见附录程序 chap9_1.m，仿真结果如图 9-10 至图 9-13 所示。

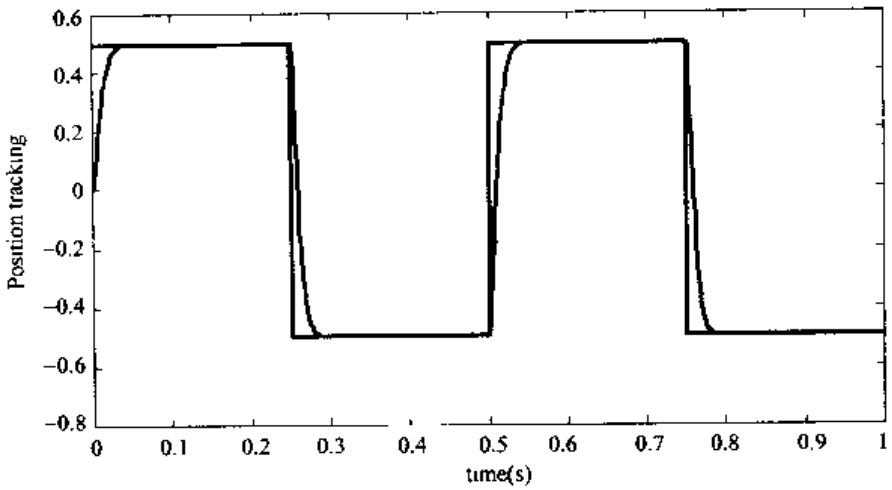


图 9-10　基于 Hebb 学习规则的位置跟踪

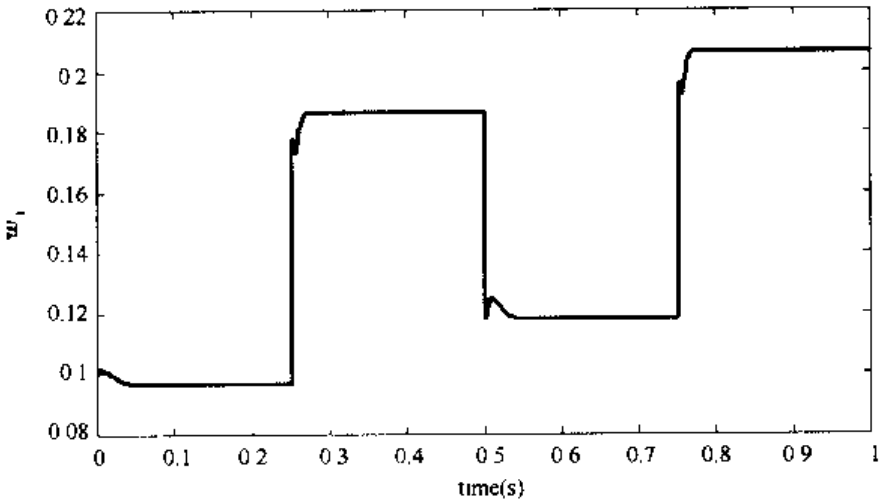


图 9-11　权值 w_1 的变化

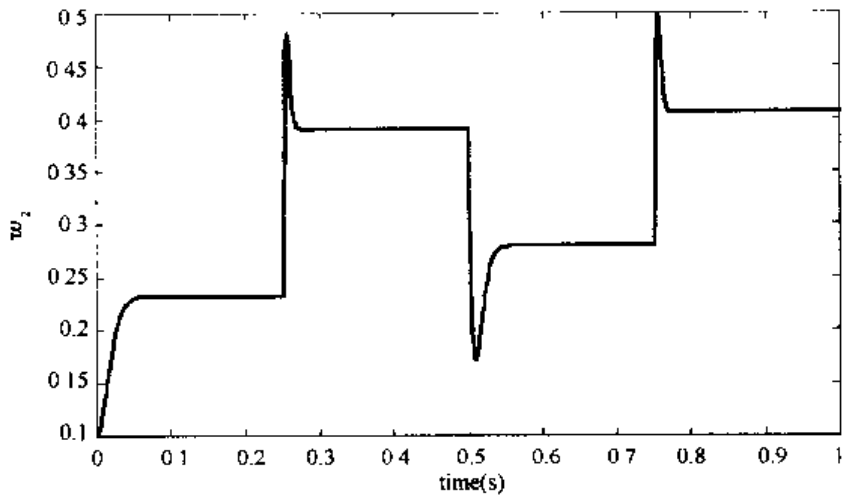


图 9-12　权值 w_2 的变化

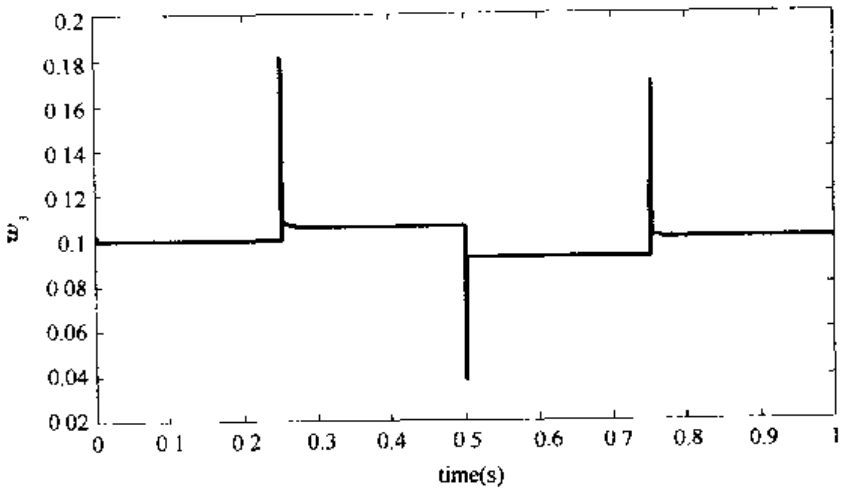


图 9-13　权值 w_3 的变化

9.4　RBF 网络监督控制

9.4.1　RBF 网络监督控制算法

基于 RBF 网络的监督控制系统结构如图 9-14 所示。

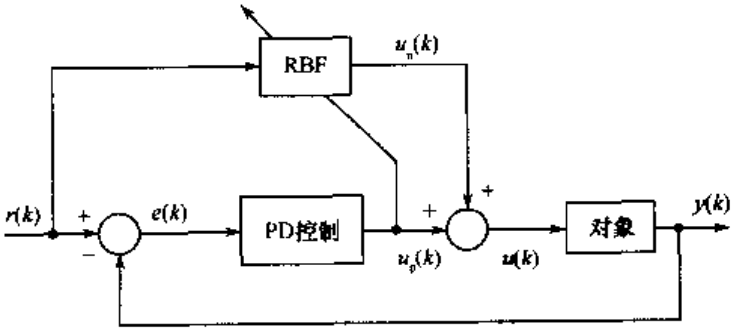


图 9-14　神经网络监督控制

在 RBF 网络结构中，取网络的输入为 $r(k)$，网络的径向基向量为 $H=[h_1,\cdots,h_m]^{\mathrm{T}}$，$h_j$ 为高斯基函数，即

$$h_j=\exp\left(-\frac{\|r(k)-\boldsymbol{C}_j\|^2}{2b_j^2}\right) \tag{9.3}$$

式中，$j=1,\cdots,m$。b_j 为节点 j 的基宽参数，$b_j>0$，$\boldsymbol{C}_j$ 为网络第 j 个节点的中心矢量，$\boldsymbol{C}_j=[c_{1i},\cdots,c_{1m}]^{\mathrm{T}}$，$\boldsymbol{B}=[b_1,\cdots,b_m]^{\mathrm{T}}$。

网络的权向量为

$$\boldsymbol{W}=[w_1,\cdots,w_m]^{\mathrm{T}}$$

RBF 网络的输出为

$$u_n(k)=h_1w_1+\cdots+h_jw_j+\cdots+h_mw_m \tag{9.4}$$

式中，m 为 RBF 网络隐层神经元的个数。

控制律为

$$u(k)=u_p(k)+u_n(k) \tag{9.5}$$

设神经网络调整的性能指标为

$$E(k)=\frac{1}{2}(u_n(k)-u(k))^2 \tag{9.6}$$

近似取 $\dfrac{\partial u(k)}{\partial w_j(k)}=\dfrac{\partial u_n(k)}{\partial w_i(k)}$。由此所产生的不精确通过权值调节来补偿。

采用梯度下降法调整网络的权值为

$$\Delta w_j(k) = -\eta \frac{\partial E(k)}{\partial w_j(k)} = \eta(u_n(k) - u(k))h_j(k)$$

神经网络权值的调整过程为

$$\boldsymbol{W}(k) = \boldsymbol{W}(k-1) + \Delta\boldsymbol{W}(k) + \alpha(\boldsymbol{W}(k-1) - \boldsymbol{W}(k-2)) \tag{9.7}$$

式中，η 为学习速率，α 为动量因子。

9.4.2 仿真实例

被控对象为

$$G(s) = \frac{1000}{s^2 + 50s + 2000}$$

取采样时间为 1ms，采用 z 变换进行离散化，经过 z 变换后的离散化对象为

$$y(k) = -\text{den}(2)y(k-1) - \text{den}(3)y(k-2) + \text{num}(2)u(k-1) + \text{num}(3)u(k-2)$$

指令信号为幅值为 0.5、频率为 2Hz 的方波信号。取指令信号 $r(k)$ 作为网络的输入，网络隐层神经元个数取 $m=4$，网络结构为 1-4-1，网络的初始权值 $\boldsymbol{W}$ 取 0～1 之间的随机值，高斯函数的参数值取 $\boldsymbol{C}_j = [-5 \quad -3 \quad 0 \quad 3]^{\mathrm{T}}$，$\boldsymbol{B} = [0.5 \quad 0.5 \quad 0.5 \quad 0.5]^{\mathrm{T}}$。

网络权值学习参数为 $\eta = 0.30$，$\alpha = 0.05$。RBF 网络监督控制程序见附录程序chap9_2.m，仿真结果如图 9-15 至图 9-16 所示。

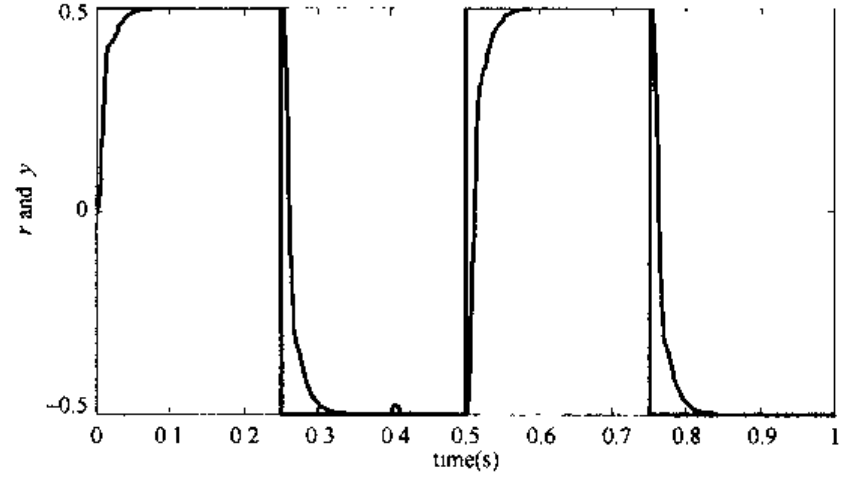


图 9-15 方波位置跟踪

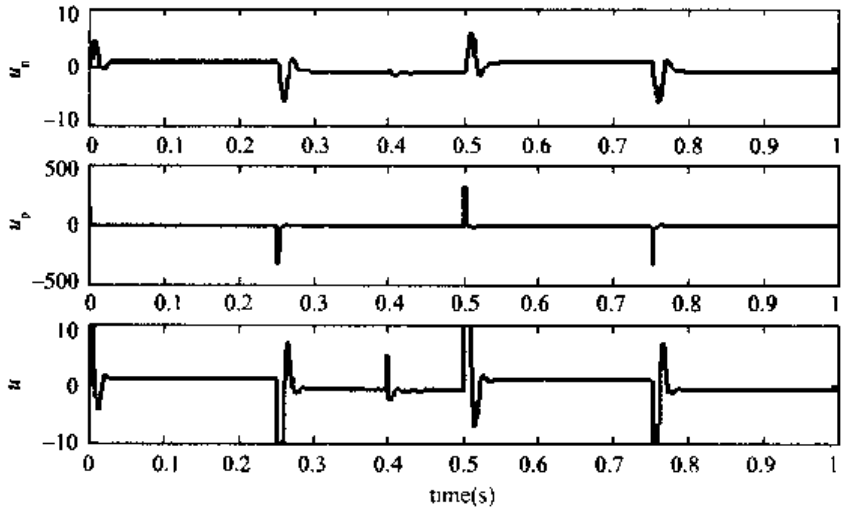


图 9-16 神经网络、PD 及总控制器输出的比较

9.5　RBF 网络自校正控制

9.5.1　神经网络自校正控制原理

自校正控制有两种结构：直接型与间接型。直接型自校正控制也称直接逆动态控制，是前馈控制。间接自校正控制是一种由辨识器将对象参数进行在线估计，用调节器（或控制器）实现参数的自动整定相结合的自适应控制技术，可用于结构已知而参数未知但恒定的随机系统，也可用于结构已知而参数缓慢时变的随机系统。

神经自校正控制结构如图 9 17 所示，它由两个回路组成：

（1）自校正控制器与被控对象构成的反馈回路；

（2）神经网络辨识器与控制器设计，以得到控制器的参数。

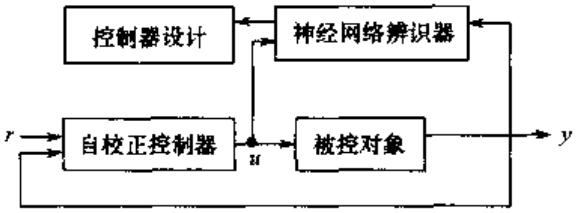


图 9-17　神经网络自校正控制框图

可见，辨识器与自校正控制器的在线设计是自校正控制实现的关键。

9.5.2　自校正控制算法

考虑被控对象

$$y(k+1)=g[y(k)]+\varphi[y(k)]u(k) \tag{9.8}$$

式中，u,y 分别为对象的输入、输出，$\varphi[\cdot]$ 为非零函数。

若 $g[\cdot],\varphi[\cdot]$ 已知，则根据确定性等价原则，自校正控制算法为

$$u(k)=\frac{-g[\cdot]}{\varphi[\cdot]}+\frac{r(k+1)}{\varphi[\cdot]} \tag{9.9}$$

采用控制器式(9.8)，系统的输出 $y(k)$ 能精确地跟踪输入 $r(k)$。

若 $g[\cdot],\varphi[\cdot]$ 未知，则可通过在线训练神经网络辨识器，得到 $g[\cdot],\varphi[\cdot]$ 的估计值 $Ng[\cdot],N\varphi[\cdot]$，此时自校正控制算法为

$$u(k)=\frac{-Ng[\cdot]}{N\varphi[\cdot]}+\frac{r(k+1)}{N\varphi[\cdot]} \tag{9.10}$$

式中，$Ng[\cdot],N\varphi[\cdot]$ 分别为神经网络辨识器的输出。

9.5.3　RBF 网络自校正控制算法

采用两个 RBF 网络分别实现未知项 $g[\cdot],\varphi[\cdot]$ 的辨识。RBF 网络辨识器的结构如图 9-18 所示，$\boldsymbol{W}$ 和 $\boldsymbol{V}$ 分别为两个神经网络的权值向量。

在 RBF 网络结构中，取网络的输入为 $y(k)$，网络的径向基向量为 $\boldsymbol{H}=[h_1,\cdots,h_m]^{\mathrm{T}}$，$h_j$ 为

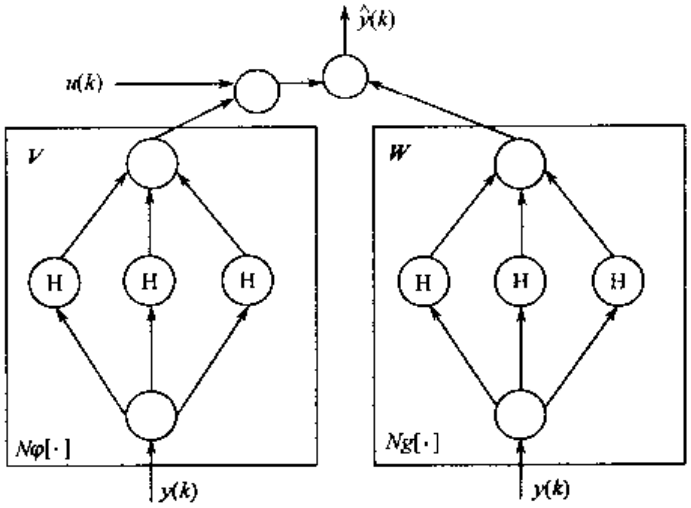


图 9-18　神经网络辨识器

高斯基函数，即

$$h_j = \exp\left(-\frac{\| y(k) - \mathbf{C}_j \|^2}{2b_j^2}\right) \tag{9.11}$$

式中，$j = 1, \cdots, m$。b_j 为节点 j 的基宽参数，$b_j > 0$，$\mathbf{C}_j$ 为网络第 j 个节点的中心矢量，$\mathbf{C}_j = [c_{1j}, \cdots, c_{1m}]^{\mathrm{T}}$，$\boldsymbol{B} = [b_1, \cdots, b_m]^{\mathrm{T}}$。

网络的权向量为

$$\boldsymbol{W} = [w_1, \cdots, w_m]^{\mathrm{T}} \tag{9.12}$$

$$\boldsymbol{V} = [v_1, \cdots, v_m]^{\mathrm{T}} \tag{9.13}$$

两个 RBF 网络的输出分别为

$$Ng(k) = h_1 w_1 + \cdots + h_j w_j + \cdots + h_m w_m \tag{9.14}$$

$$N\varphi(k) = h_1 v_1 + \cdots + h_j v_j + \cdots + h_m v_m \tag{9.15}$$

式中，m 为 RBF 网络隐层神经元的个数。

辨识后，对象的输出为

$$y_m(k) = Ng[y(k-1); \boldsymbol{W}(k)] + N\varphi[y(k-1); \boldsymbol{V}(k)]u(k-1) \tag{9.16}$$

设神经网络调整的性能指标为

$$E(k) = \frac{1}{2}(y(k) - y_m(k))^2 \tag{9.17}$$

采用梯度下降法调整网络的权值为

$$\Delta w_j(k) = -\eta_w \frac{\partial E(k)}{\partial w_j(k)} = \eta_w (y(k) - y_m(k)) h_j(k)$$

$$\Delta v_j(k) = -\eta_v \frac{\partial E(k)}{\partial v_j(k)} = \eta_v (y(k) - y_m(k)) h_j(k)$$

神经网络权值的调整过程为

$$\boldsymbol{W}(k)=\boldsymbol{W}(k-1)+\Delta\boldsymbol{W}(k)+\alpha(\boldsymbol{W}(k-1)-\boldsymbol{W}(k-2)) \tag{9.18}$$

$$\boldsymbol{V}(k)=\boldsymbol{V}(k-1)+\Delta\boldsymbol{V}(k)+\alpha(\boldsymbol{V}(k-1)-\boldsymbol{V}(k-2)) \tag{9.19}$$

式中，η_w 和 η_v 为学习速率，α 为动量因子。

综上所述，神经网络自校正控制系统的结构如图 9-19 所示。

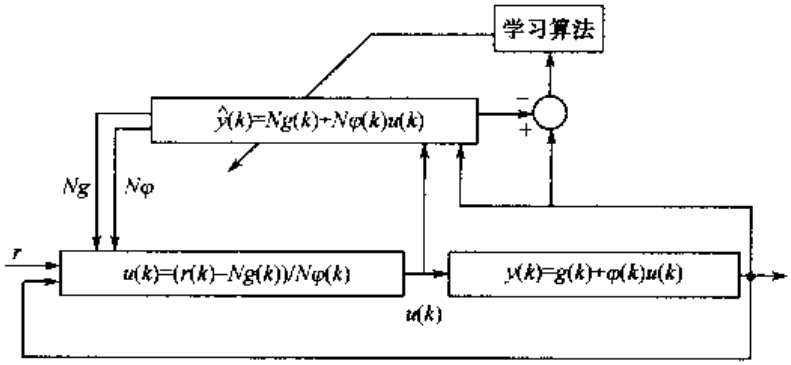


图 9-19　神经网络自校正控制框图

9.5.4　仿真实例

被控对象为

$$y(k)=0.8\sin(y(k-1))+15u(k-1)$$

式中，$g[y(k)]=0.8\sin(y(k-1))$，$\varphi[y(k)]=15$。

输入信号为正弦信号：$r(t)=2.0\sin(0.1\pi t)$。取 $y(k)$ 作为网络的输入，网络隐层神经元个数取 $m=6$，神经网络结构为 1-6-1，网络的初始权值取 $\boldsymbol{W}=[0.5,0.5,0.5,0.5,0.5,0.5]^{\mathrm{T}}$，$\boldsymbol{V}=[0.5,0.5,0.5,0.5,0.5,0.5]^{\mathrm{T}}$，高斯函数的初始值取 $\boldsymbol{C}_j=[0.5,0.5,0.5,0.5,0.5,0.5]^{\mathrm{T}}$，$\boldsymbol{B}=[5,5,5,5,5,5]^{\mathrm{T}}$。

网络权值学习参数为 $\eta_1=0.15$，$\eta_2=0.50$，$\alpha=0.05$。RBF 网络自校正控制程序见附录程序 chap9_3.m。仿真结果如图 9-20 至图 9-22 所示。

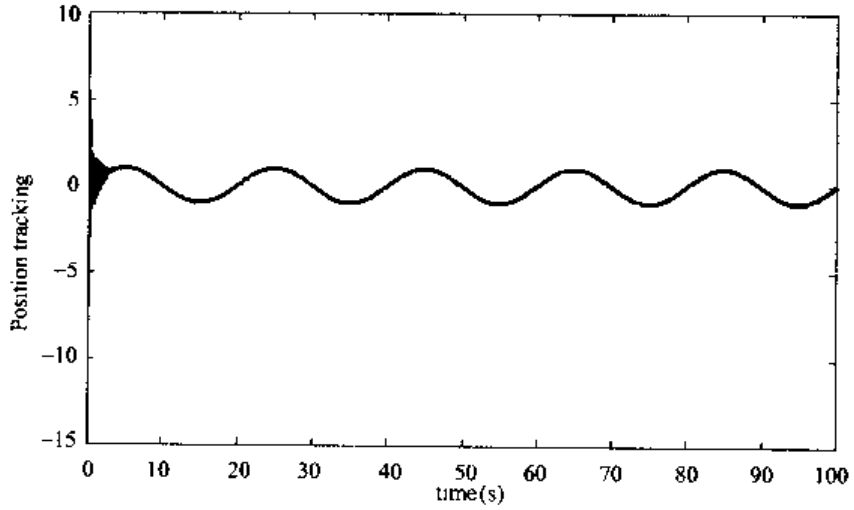


图 9-20　正弦位置跟踪

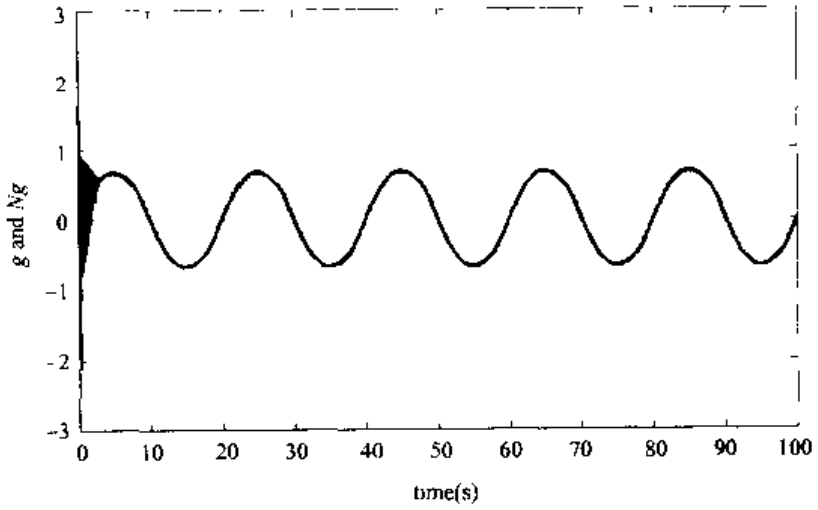


图 9-21　$g(x,t)$ 及其辨识结果 $\hat{g}(x,t)$

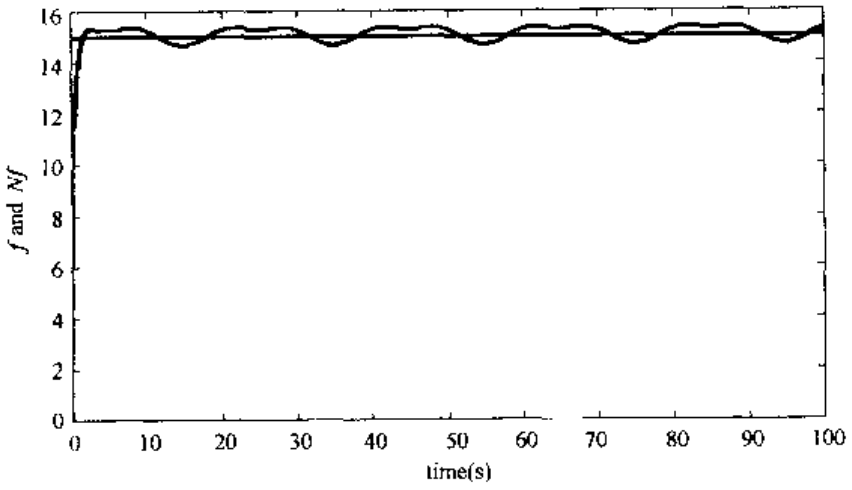


图 9-22　$f(x,t)$ 及其辨识结果 $\hat{f}(x,t)$

9.6　基于 RBF 网络直接模型参考自适应控制

9.6.1　基于 RBF 网络的控制器设计

控制系统的结构如图 9-23 所示。

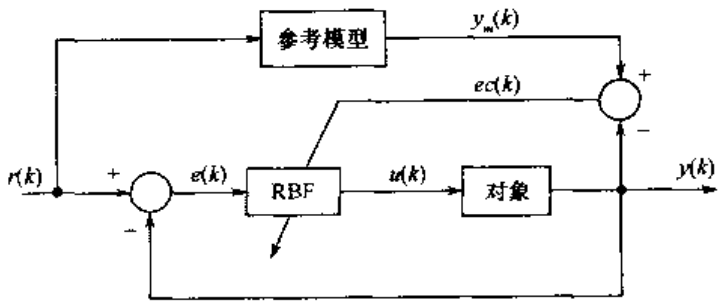


图 9-23　基于 RBF 网络的直接模型参考自适应控制

设参考模型输出为 $y_m(k)$，控制系统要求对象的输出 $y(k)$ 能够跟踪参考模型的输出 $y_m(k)$。则跟踪误差为

$$ec(k) = y_m(k) - y(k) \tag{9.20}$$

指标函数为

$$E(k) = \frac{1}{2}ec(k)^2 \tag{9.21}$$

控制器为 RBF 网络的输出为

$$u(k) = h_1w_1 + \cdots + h_jw_j + \cdots + h_mw_m \tag{9.22}$$

式中，m 为 RBF 网络隐层神经元的个数，w_j 为第 j 个网络隐层神经元与输出层之间的连接权，h_j 为第 j 个隐层神经元的输出。

在 RBF 网络结构中，$\boldsymbol{X} = [x_1, \cdots, x_n]^{\mathrm{T}}$ 为网络的输入向量。RBF 网络的径向基向量为 $\boldsymbol{H} = [h_1, \cdots, h_m]^{\mathrm{T}}$，$h_j$ 为高斯基函数，即

$$h_j = \exp\left(-\frac{\| \boldsymbol{X} - \boldsymbol{C}_j \|^2}{2b_j^2}\right) \tag{9.23}$$

式中，$j = 1, \cdots, m$。b_j 为节点 j 的基宽参数，$b_j > 0$，$\boldsymbol{C}_j$ 为网络第 j 个节点的中心矢量，$\boldsymbol{C}_j = [c_{j1}, \cdots, c_{jn}, \cdots, c_{jm}]^{\mathrm{T}}$，$\boldsymbol{B} = [b_1, \cdots, b_m]^{\mathrm{T}}$。

网络的权向量为

$$\boldsymbol{W} = [w_1, \cdots, w_m]^{\mathrm{T}} \tag{9.24}$$

按梯度下降法及链式法则，可得权值的学习算法如下

$$\Delta w_j(k) = -\eta\frac{\partial E(k)}{\partial w} = \eta ec(k)\frac{\partial y(k)}{\partial u(k)}h_j$$

$$w_j(k) = w_j(k-1) + \Delta w_j(k) + \alpha\Delta w_j(k) \tag{9.25}$$

式中，η 为学习速率，α 为动量因子。

同理，可得 RBF 网络隐层神经元的高斯函数的基宽参数及中心参数的学习算法如下

$$\Delta b_j(k) = -\eta\frac{\partial E(k)}{\partial b_j} = \eta ec(k)\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial b_j} = \eta ec(k)\frac{\partial y(k)}{\partial u(k)}w_jh_j\frac{\| \boldsymbol{x} - c_{ij} \|^2}{b_j^3} \tag{9.26}$$

$$b_j(k) = b_j(k-1) + \eta\Delta b_j(k) + \alpha(b_j(k-1) - b_j(k-2)) \tag{9.27}$$

$$\Delta c_{ij}(k) = -\eta\frac{\partial E(k)}{\partial c_{ij}} = \eta ec(k)\frac{\partial y(k)}{\partial u(k)}\frac{\partial u(k)}{\partial c_{ij}} = \eta ec(k)\frac{\partial y(k)}{\partial u(k)}w_jh_j\frac{x_i - c_{ij}}{b_j^2} \tag{9.28}$$

$$c_{ij}(k) = c_{ij}(k-1) + \eta\Delta c_{ij}(k) + \alpha(c_{ij}(k-1) - c_{ij}(k-2)) \tag{9.29}$$

在学习算法中，$\frac{\partial y(k)}{\partial u(k)}$ 称为 Jacobian 信息，表示系统的输出对控制输入的敏感性，其值可由神经网络辨识而得。在神经网络算法中，对 $\frac{\partial y(k)}{\partial u(k)}$ 值的精确度要求不是很高，不精确部分可通过网络参数及权值的调整来修正，关键是其符号，因此可用 $\frac{\partial y(k)}{\partial u(k)}$ 的正负号来代替，这样可使算法更加简单。

9.6.2　仿真实例

被控对象为一非线性模型

$$y(k) = (-0.10y(k-1) + u(k-1))/(1 + y(k-1)^2)$$

取采样周期为 $ts = 1\mathrm{ms}$，参考模型为 $y_m(k) = 0.6y_m(k-1) + r(k)$，其中 $r(k)$ 为正弦信

号，$r(k)=0.50\sin(2\pi k\times ts)$。

采用 RBF 网络进行控制，取 $r(k)$，$e(k)$ 和 $u(k)$ 作为 RBF 网络的输入，网络采用 3-6-1 结构。取学习速率 $\eta=0.35$，动量因子 $\alpha=0.05$。

高斯函数的初始值取 $\boldsymbol{C}=\begin{bmatrix}-3 & -2 & -1 & 1 & 2 & 3\\ -3 & -2 & -1 & 1 & 2 & 3\\ -3 & -2 & -1 & 1 & 2 & 3\end{bmatrix}$，$\boldsymbol{B}=[2,2,2,2,2,2]^{\mathrm{T}}$。首先，网络的初始权值取随机值，经过多次仿真试凑，最终网络的初始权值取 $\boldsymbol{W}=[-0.0316\quad -0.0421\quad -0.0318\quad 0.0068\quad 0.0454\quad -0.0381]$。

RBF 网络直接模型参考自适应控制程序见附录程序 chap9_4.m，仿真时间为 3s，仿真结果如图 9-24 所示。

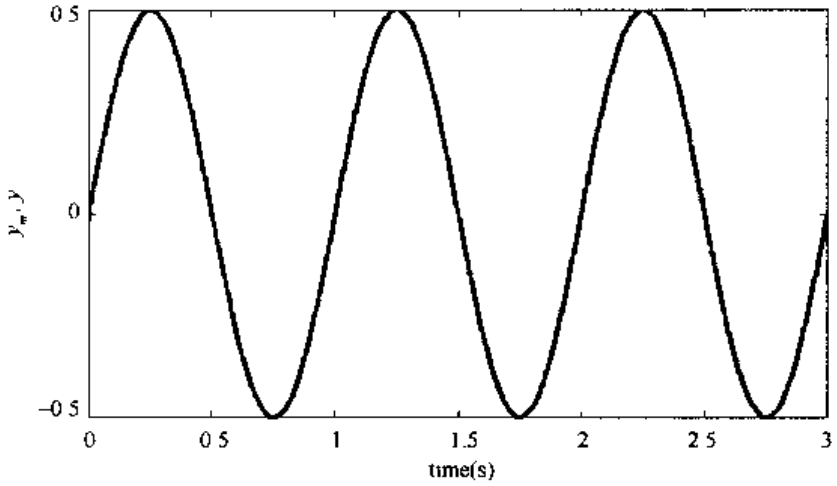


图 9-24　正弦信号跟踪

思考题与习题

9-1　参照 RBF 网络直接模型参考自适应控制算法，试推导 BP 网络直接模型参考自适应控制算法。

9-2　参照 RBF 网络的自校正控制方法，设计基于 RBF 网络的模型参考自校正控制器，并进行 Matlab 仿真。被控对象为 $y(k)=0.8\sin(y(k-1))+15u(k-1)$，采样周期为 $T=0.001$，参考模型为 $y_m(k)=0.6y_m(k-1)+r(k)$，$r(k)$ 为正弦信号，$r(k)=0.50\sin(2\pi kT)$。

9-3　已知一非线性系统

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k)$$

给定的期望轨迹为

$$y_d(k)=\sin\frac{2\pi k}{25}+\sin\frac{2\pi k}{10}$$

试采用 RBF 神经网络进行自适应控制，其中 Jacobian 信息由 RBF 网络辨识，并进行 Matlab 仿真。

附录（程序代码）

单神经元自适应控制程序：chap9_1.m

```
% Single Neural Adaptive Controller
clear all;
close all;

x=[0,0,0]';

xite=0.40;

w1_1=0.10;
w2_1=0.10;
w3_1=0.10;

e_1=0;
e_2=0;
y_1=0;y_2=0;
u_1=0;u_2=0;

ts=0.001;
for k=1:1:1000
    time(k)=k*ts;
    r(k)=0.5*sign(sin(2*2*pi*k*ts));
    y(k)=0.368*y_1+0.26*y_2+0.1*u_1+0.632*u_2;
    e(k)=r(k)-y(k);

% Adjusting Weight Value by supervised Heb learning algorithm
    w1(k)=w1_1+xite*e(k)*u_1*x(1);
    w2(k)=w2_1+xite*e(k)*u_1*x(2);
    w3(k)=w3_1+xite*e(k)*u_1*x(3);
    K=0.12;

    x(1)=e(k)-e_1;
    x(2)=e(k);
    x(3)=e(k)-2*e_1+e_2;

    w=[w1(k),w2(k),w3(k)];
    u(k)=u_1+K*w*x;          % Control law
```

```
e_2=e_1;
e_1=e(k);

u_2=u_1;u_1=u(k);
y_2=y_1;y_1=y(k);

w1_1=w1(k);
w2_1=w2(k);
w3_1=w3(k);
end
figure(1);
plot(time,r,'b',time,y,'r');
xlabel('time(s)');ylabel('Position tracking');
figure(2);
plot(time,e,'r');
xlabel('time(s)');ylabel('error');
figure(3);
plot(time,w1,'r');
xlabel('time(s)');ylabel('w1');
figure(4);
plot(time,w2,'r');
xlabel('time(s)');ylabel('w2');
figure(5);
plot(time,w3,'r');
xlabel('time(s)');ylabel('w3');
```

RBF 网络监督控制程序:chap9_2.m

```
% RBF Supervisory Control
clear all;
close all;

ts=0.001;
sys=tf(1000,[1,50,2000]);
dsys=c2d(sys,ts,'z');
[num,den]=tfdata(dsys,'v');

y_1=0;y_2=0;
u_1=0;u_2=0;
e_1=0;
```

```
xi=[0,0,0]';
x=[0;0]';

b=0.5*ones(4,1);
c=[-5 -3 0 3 5];
w=rands(4,1);
w_1=w;
w_2=w_1;

xite=0.30;
alfa=0.05;

kp=25;
kd=0.3;
for k=1:1:1000
    time(k)=k*ts;
S=1;
if S==1
    r(k)=0.5*sign(sin(2*2*pi*k*ts));    %Square Signal
elseif S==2
    r(k)=0.5*(sin(3*2*pi*k*ts));        %Sine Signal
end

y(k)=-den(2)*y_1-den(3)*y_2+num(2)*u_1+num(3)*u_2;
e(k)=r(k)-y(k);

xi(1)=r(k);

for j=1:1:4
    h(j)=exp(-norm(xi-c(:,j))^2/(2*b(j)*b(j)));
end
un(k)=w'*h';

%PD Controller
up(k)=kp*x(1)+kd*x(2);

M=2;
if M==1          %Only Using PID Control
    u(k)=up(k);
elseif M==2    %Total control output
```

```
    u(k)=up(k)+un(k);
end

if u(k)>=10
    u(k)=10;
end
if u(k)<=-10
    u(k)=-10;
end
if k==400
    u(k)=u(k)+6.0;
end

%Update NN Weight
d_w=-xite*(un(k)-u(k))*h';
w=w_1+ d_w+alfa*(w_1-w_2);

w_2=w_1;
w_1=w;
u_2=u_1;
u_1=u(k);
y_2=y_1;
y_1=y(k);

x(1)=e(k);                    %Calculating P
x(2)=(e(k)-e_1)/ts;           %Calculating D
e_1=e(k);
end
figure(1);
plot(time,r,'r',time,y,'b');
xlabel('time(s)');ylabel('r and y');
figure(2);
subplot(311);
plot(time,un,'b');
xlabel('time(s)');ylabel('un');
subplot(312);
plot(time,up,'k');
xlabel('time(s)');ylabel('up');
subplot(313);
plot(time,u,'r');
```

```
xlabel('time(s)');ylabel('u');
```

RBF 网络自校正控制程序：chap9_3.m

```
% Self-Correct control based RBF Identification
clear all;
close all;

xite1=0.15;
xite2=0.50;
alfa=0.05;

w=0.5*ones(6,1);
v=0.5*ones(6,1);

cij=0.50*ones(1,6);
bj=5*ones(6,1);
h=zeros(6,1);

w_1=w;w_2=w_1;
v_1=v;v_2=v_1;
u_1=0;y_1=0;

ts=0.02;
for k=1:1:5000
time(k)=k*ts;
r(k)=1.0*sin(0.1*pi*k*ts);

% Practical Plant;
g(k)=0.8*sin(y_1);
f(k)=15;
y(k)=g(k)+f(k)*u_1;

for j=1:1:6
    h(j)=exp(-norm(y(k)-cij(:,j))^2/(2*bj(j)*bj(j)));
end

Ng(k)=w'*h;
Nf(k)=v'*h;

ym(k)=Ng(k)+Nf(k)*u_1;
```

```
e(k)=y(k)-ym(k);

d_w=0*w;
for j=1:1:6
    d_w(j)=xite1*e(k)*h(j);
end
w=w_1+d_w+alfa*(w_1-w_2);

d_v=0*v;
for j=1:1:6
    d_v(j)=xite2*e(k)*h(j)*u_1;
end
v=v_1+d_v+alfa*(v_1-v_2);

u(k)=-Ng(k)/Nf(k)+r(k)/Nf(k);

u_1=u(k);
y_1=y(k);

w_2=w_1;
w_1=w;

v_2=v_1;
v_1=v;
end

figure(1);
plot(time,r,'r',time,y,'b');
xlabel('Time(second)');ylabel('Position tracking');
figure(2);
plot(time,g,'r',time,Ng,'b');
xlabel('Time(second)');ylabel('g and Ng');
figure(3);
plot(time,f,'r',time,Nf,'b');
xlabel('Time(second)');ylabel('f and Nf');
```

RBF 网络直接模型参考自适应控制程序:chap9_4.m

```
%Model Reference Aapative RBF Control
clear all;
```

```
close all;

u_1=0;
y_1=0;y_2=0;
ym_1=0;

x=[0,0,0]';
c=[-3 -2 -1 1 2 3;
   -3 -2 -1 1 2 3;
   -3 -2 -1 1 2 3];
b=2*ones(6,1);

%w=rands(6,1)
w=[-0.0316
   -0.0421
   -0.0318
    0.0068
    0.0454
   -0.0381];
xite=0.35;
alfa=0.05;
h=[0,0,0,0,0,0]';

c_1=c;c_2=c;
b_1=b;b_2=b;
w_1=w;w_2=w;

ts=0.001;
for k=1:1:3000
time(k)=k*ts;

r(k)=0.50*sin(2*pi*k*ts);
ym(k)=0.6*ym_1+r(k);

y(k)=(-0.1*y_1+u_1)/(1+y_1^2);    %Nonlinear plant

for j=1:1:6
    h(j)=exp(-norm(x-c(:,j))^2/(2*b(j)*b(j)));
end
u(k)=w'*h;
```

```
ec(k)=ym(k)-y(k);
e(k)=r(k)-y(k);

dyu(k)=sign((y(k)-y_1)/(u(k)-u_1));

    d_w=0*w;
    for j=1:1:6
      d_w(j)=xite*e(k)*h(j)*dyu(k);
    end
    w=w_1+d_w+alfa*(w_1-w_2);

    d_b=0*b;
    for j=1:1:6
      d_b(j)=xite*e(k)*w(j)*h(j)*(b(j)^-3)*norm(x-c(:,j))^2*dyu(k);
    end
    b=b_1+d_b+alfa*(b_1-b_2);

    d_c=0*c;
    for j=1:1:6
      for i=1:1:3
      d_c(i,j)=xite*e(k)*w(j)*h(j)*(x(i)-c(i,j))*(b(j)^-2)*dyu(k);
      end
    end
    c=c_1+d_c+alfa*(c_1-c_2);

% Return of parameters
    u_1=u(k);
    y_2=y_1;y_1=y(k);

    x(1)=r(k);
    x(2)=e(k);
    x(3)=u(k);

    w_2=w_1;w_1=w;
    c_2=c_1;c_1=c;
    b_2=b_1;b_1=b;
end
figure(1);
plot(time,ym,'r',time,y,'b');
```

```
xlabel('time(s)');ylabel('ym,y');
figure(2);
plot(time,u);
xlabel('time(s)');ylabel('Control input');
```

第 10 章　遗传算法及其应用

10.1　遗传算法的基本原理

遗传算法简称GA(Genetic Algorithms)，是1962年由美国Michigan大学Holland教授提出的模拟自然界遗传机制和生物进化论而成的一种并行随机搜索最优化方法。

遗传算法是以达尔文的自然选择学说为基础发展起来的。自然选择学说包括以下3个方面。

(1) 遗传

这是生物的普遍特征，亲代把生物信息交给子代，子代按照所得信息而发育、分化，因而子代总是和亲代具有相同或相似的性状。生物有了这个特征，物种才能稳定存在。

(2) 变异

亲代和子代之间及子代的不同个体之间总是有些差异，这种现象称为变异。变异是随机发生的，变异的选择和积累是生命多样性的根源。

(3) 生存斗争和适者生存

自然选择来自繁殖过剩和生存斗争。由于弱肉强食的生存斗争不断地进行，其结果是适者生存，即具有适应性变异的个体被保留下来，不具有适应性变异的个体被淘汰，通过一代代的生存环境的选择作用，性状逐渐与祖先有所不同，演变为新的物种。这种自然选择过程是一个长期的、缓慢的、连续的过程。

遗传算法将“优胜劣汰，适者生存”的生物进化原理引入优化参数形成的编码串联群体中，按所选择的适配值函数并通过遗传中的复制、交叉及变异对个体进行筛选，使适配值高的个体被保留下来，组成新的群体，新的群体既继承了上一代的信息，又优于上一代。这样周而复始，群体中个体适应度不断提高，直到满足一定的条件。遗传算法的算法简单，可并行处理，并能得到全局最优解。

遗传算法的基本操作为：

(1) 复制(Reproduction Operator)

复制是从一个旧种群中选择生命力强的个体位串产生新种群的过程。根据位串的适配值复制，也就是指具有高适配值的位串更有可能在下一代中产生一个或多个子孙。它模仿了自然现象，应用了达尔文的适者生存理论。复制操作可以通过随机方法来实现。若用计算机程序来实现，可考虑首先产生0～1之间均匀分布的随机数，若某位串的复制概率为40%，则当产生的随机数在0.40～1.0时，该位串被复制，否则被淘汰。此外，还可以通过计算方法实现，其中较典型的几种方法为适应度比例法、期望值法、排位次法等。适应度比例法比较常用。

(2) 交叉(Crossover Operator)

复制操作能从旧种群中选择出优秀者，但不能创造新的染色体。而交叉模拟了生物进化过程中的繁殖现象，通过两个染色体的交换组合，来产生新的优良品种。交叉过程为：在匹配池中任选两个染色体，随机选择一点或多点交换点位置；交换双亲染色体交换点右边的部分，即可

得到两个新的染色体数字串。交换体现了自然界中信息交换的思想。交叉有一点交叉、多点交叉，还有一致交叉、顺序交叉和周期交叉。一点交叉是最基本的方法，应用较广。它是指染色体切断点有一处，例如

$$A:101100\ 1110 \rightarrow 101100\ 0101$$

$$B:001010\ 0101 \rightarrow 001010\ 1110$$

(3) 变异(Mutation Operator)

变异运算用来模拟生物在自然的遗传环境中由于各种偶然因素引起的基因突变，它以很小的概率随机地改变遗传基因(表示染色体的符号串的某一位)的值。在染色体以二进制编码的系统中，它随机地将染色体的某一个基因由 1 变为 0，或由 0 变为 1。若只有选择和交叉，而没有变异，则无法在初始基因组合以外的空间进行搜索，使进化过程在早期就陷入局部解而进入终止过程，从而影响解的质量。为了在尽可能大的空间中获得质量较高的优化解，必须采用变异操作。

10.2　遗传算法的特点

(1) 遗传算法是对参数的编码进行操作，而非对参数本身，这就是使得我们在优化计算过程中可以借鉴生物学中染色体和基因等概念，模仿自然界中生物的遗传和进化等机理。

(2) 遗传算法同时使用多个搜索点的搜索信息。传统的优化方法往往是从解空间的一个初始点开始最优解的迭代搜索过程，单个搜索点所提供的信息不多，搜索效率不高，有时甚至使搜索过程局限于局部最优解而停滞不前。遗传算法从由很多个体组成的一个初始群体开始最优解的搜索过程，而不是从一个单一的个体开始搜索，这是遗传算法所特有的一种隐含并行性，因此遗传算法的搜索效率较高。

(3) 遗传算法直接以目标函数作为搜索信息。传统的优化算法不仅需要利用目标函数值，而且需要目标函数的导数值等辅助信息才能确定搜索方向。而遗传算法仅使用由目标函数值变换来的适应度函数值，就可以确定进一步的搜索方向和搜索范围，无需目标函数的导数值等其他一些辅助信息。因此，遗传算法可应用于目标函数无法求导数或导数不存在的函数的优化问题及组合优化问题等。而且直接利用目标函数值或个体适应度，也可将搜索范围集中到适应度较高的部分搜索空间中，从而提高搜索效率。

(4) 遗传算法使用概率搜索技术。许多传统的优化算法使用的是确定性搜索算法，一个搜索点到另一个搜索点的转移有确定的转移方法和转移关系，这种确定性的搜索方法有可能使得搜索无法达到最优点，因而限制了算法的使用范围。遗传算法的选择、交叉、变异等运算都是以一种概率的方式来进行的，因而遗传算法的搜索过程具有很好的灵活性。随着进化过程的进行，遗传算法新的群体会更多地产生出许多新的优良的个体。理论已经证明，遗传算法在一定条件下以概率 1 收敛于问题的最优解。

(5) 遗传算法在解空间进行高效启发式搜索，而非盲目地穷举或完全随机搜索。

(6) 遗传算法对于待寻优的函数基本无限制，它既不要求函数连续，也不要求函数可微，既可以是数学解析式所表示的显函数，又可以是映射矩阵甚至是神经网络的隐函数，因而应用范围较广。

(7) 遗传算法具有并行计算的特点，因而可通过大规模并行计算来提高计算速度，适合大规模复杂问题的优化。

10.3 遗传算法的发展及应用

10.3.1 遗传算法的发展

遗传算法起源于对生物系统所进行的计算机模拟研究。早在 20 世纪 40 年代，就有学者开始研究如何利用计算机进行生物模拟的技术，他们从生物学的角度进行了生物的进化过程模拟、遗传过程模拟等研究工作。进入 20 世纪 60 年代，美国密执安大学的 Holland 教授及其学生们受到这种生物模拟技术的启发，创造出一种基于生物遗传和进化机制的、适合于复杂系统优化计算的自适应概率优化技术 —— 遗传算法。

以下是在遗传算法发展进程中一些关键人物所做出的主要贡献。

(1)J. H. Holland

20 世纪 70 年代初，Holland 教授提出了遗传算法的基本定理 —— 模式定理，从而奠定了遗传算法的理论基础。模式定理揭示了群体中优良个体(较好的模式) 的样本数将以指数级规律增长，从理论上保证了遗传算法用于寻求最优可行解的优化过程。1975 年，Holland 出版了第一本系统论述遗传算法和人工自适应系统的专著《自然系统和人工系统的自适应性》。20 世纪 80 年代，Holland 教授实现了第一个基于遗传算法的机器学习系统 —— 分类器系统，开创了基于遗传算法的机器学习的新概念。

(2)J. D. Bagley

1967 年，Holland 的学生 Bagley 在其博士论文中首次提出了“遗传算法”一词，并发表了遗传算法应用方面的第一篇论文。他发展了复制、交叉、变异、显性、倒位等遗传算子，在个体编码上使用了双倍体的编码方法。在遗传算法的不同阶段采用了不同的概率，从而创立了自适应遗传算法的概念。

(3)K. A. De Jong

1975 年，De Jong 博士在其博士论文中结合模式定理进行了大量纯数值函数优化计算实验，树立了遗传算法的工作框架。他推荐了在大多数优化问题中都较适用的遗传算法的参数，建立了著名的 De Jong 五函数测试平台，定义了评价遗传算法性能的在线指标和离线指标。

(4)D. J. Goldberg

1989 年，Goldberg 出版了专著《搜索、优化和机器学习中的遗传算法》，该书全面地论述了遗传算法的基本原理及其应用，奠定了现代遗传算法的科学基础。

(5)L. Davis

1991 年，Davis 出版了《遗传算法手册》一书，为推广和普及遗传算法的应用起到了重要的指导作用。

(6)J. R. Koza

1992 年，Koza 将遗传算法应用于计算机程序的优化设计及自动生成，提出了遗传编程的概念，并成功地将遗传编程的方法应用于人工智能、机器学习和符号处理等方面。

10.3.2 遗传算法的应用

1. 函数优化

函数优化是遗传算法的经典应用领域，也是遗传算法进行性能评价的常用算例。尤其是对

非线性、多模型、多目标的函数优化问题，采用其他优化方法较难求解，而遗传算法却可以得到较好的结果。

2. 组合优化

随着问题的增大，组合优化问题的搜索空间也急剧扩大，采用传统的优化方法很难得到最优解。遗传算法是寻求这种满意解的最佳工具。例如，遗传算法已经在求解旅行商问题、背包问题、装箱问题、图形划分问题等方面得到成功的应用。

3. 生产调度问题

在很多情况下，采用建立数学模型的方法难以对生产调度问题进行精确求解。在现实生产中，多采用一些经验进行调度。遗传算法是解决复杂调度问题的有效工具，在单件生产车间调度、流水线生产车间调度、生产规划、任务分配等方面，遗传算法都得到了有效的应用。

4. 自动控制

在自动控制领域中有很多与优化相关的问题需要求解，遗传算法已经在其中得到了初步的应用。例如，利用遗传算法进行控制器参数的优化、基于遗传算法的模糊控制规则的学习、基于遗传算法的参数辨识、基于遗传算法的神经网络结构的优化和权值学习等。

5. 机器人

例如，遗传算法已经在移动机器人路径规划、关节机器人运动轨迹规划、机器人结构优化和行为协调等方面得到研究和应用。

6. 图像处理

遗传算法可用于图像处理过程中的扫描、特征提取、图像分割等的优化计算。目前遗传算法已经在模式识别、图像恢复、图像边缘特征提取等方面得到了应用。

7. 人工生命

人工生命是用计算机、机械等人工媒体模拟或构造出的具有生物系统特有行为的人造系统。人工生命与遗传算法有着密切的联系，基于遗传算法的进化模型是研究人工生命现象的重要基础理论。遗传算法为人工生命的研究提供了一个有效的工具。

8. 遗传编程

遗传算法已成功地应用于人工智能、机器学习等领域的编程。

9. 机器学习

基于遗传算法的机器学习在很多领域都得到了应用。例如，采用遗传算法实现模糊控制规则的优化，可以改进模糊系统的性能；遗传算法可用于神经网络连接权的调整和结构的优化；采用遗传算法设计的分类器系统可用于学习式多机器人路径规划。

10.4 遗传算法的优化设计

10.4.1 遗传算法的构成要素

1. 染色体编码方法

基本遗传算法使用固定长度的二进制符号来表示群体中的个体，其等位基因是由二值符号集{0,1}所组成的。初始个体的基因值可用均匀分布的随机值来生成，如 $x=100111001000101101$ 就可表示一个个体，该个体的染色体长度是 $n=18$。

2. 个体适应度评价

基本遗传算法与个体适应度成正比的概率来决定当前群体中每个个体遗传到下一代群体中的概率多少。为正确计算这个概率，要求所有个体的适应度必须为正数或零。因此，必须先确定由目标函数值到个体适应度之间的转换规则。

3. 遗传算子

基本遗传算法使用下述3种遗传算子。

(1) 选择运算使用比例选择算子；

(2) 交叉运算使用单点交叉算子；

(3) 变异运算使用基本位变异算子或均匀变异算子。

4. 基本遗传算法的运行参数

有下述4个运行参数需要提前设定。

M：群体大小，即群体中所含个体的数量，一般取20～100；

G：遗传算法的终止进化代数，一般取100～500；

P_c：交叉概率，一般取0.4～0.99；

P_m：变异概率，一般取0.0001～0.1。

10.4.2 遗传算法的应用步骤

对于一个需要进行优化的实际问题，一般可按下述步骤构造遗传算法：

第1步：确定决策变量及各种约束条件，即确定出个体的表现型X和问题的解空间；

第2步：建立优化模型，即确定出目标函数的类型及数学描述形式或量化方法；

第3步：确定表示可行解的染色体编码方法，即确定出个体的基因型x及遗传算法的搜索空间；

第4步：确定个体适应度的量化评价方法，即确定出由目标函数值 $J(x)$ 到个体适应度函数 $F(x)$ 的转换规则；

第5步：设计遗传算子，即确定选择运算、交叉运算、变异运算等遗传算子的具体操作方法；

第6步：确定遗传算法的有关运行参数，即 M,G,P_c,P_m 等参数；

第 7 步：确定解码方法，即确定出由个体表现型 X 到个体基因型 x 的对应关系或转换方法。

以上操作过程可以用图 10 1 来表示。

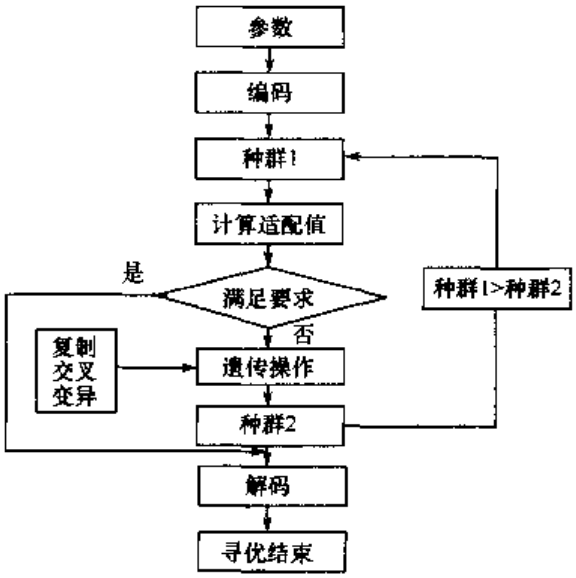


图 10-1　遗传算法流程图

10.5　遗传算法求函数极大值

利用遗传算法求 Rosenbrock 函数的极大值

$$\begin{cases} f(x_1,x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \\ -2.048 \leqslant x_i \leqslant 2.048 (i = 1,2) \end{cases}$$

该函数有两个局部极大点，分别是 $f(2.048, -2.048) = 3897.7342$ 和 $f(-2.048, -2.048) = 3905.9262$，其中后者为全局最大点。

10.5.1　二进制编码遗传算法求函数极大值

求解该问题遗传算法的构造过程：

(1) 确定决策变量和约束条件；

(2) 建立优化模型；

(3) 确定编码方法：用长度为 10 位的二进制编码串来分别表示两个决策变量 x_1, x_2。10 位二进制编码串可以表示从 0 ～ 1023 之间的 1024 个不同的数，故将 x_1, x_2 的定义域离散化为 1023 个均等的区域，包括两个端点在内共有 1024 个不同的离散点。从离散点 −2.048 到离散点 2.048，依次让它们分别对应于从 0000000000(0) ～ 1111111111(1023) 之间的二进制编码。再将分别表示 x_1, x_2 的两个 10 位长的二进制编码串连接在一起，组成一个 20 位长的二进制编码串，它就构成了这个函数优化问题的染色体编码方法。使用这种编码方法，解空间和遗传算法的搜索空间就具有一一对应的关系。例如，x：0000110111 1101110001 就表示一个个体的基因型，其中前 10 位表示 x_1，后 10 位表示 x_2。

(4) 确定解码方法：解码时需要将 20 位长的二进制编码串切断为两个 10 位长的二进制编码串，然后分别将它们转换为对应的十进制整数代码，分别记为 y_1 和 y_2。依据个体编码方法和

对定义域的离散化方法可知，将代码 y_i 转换为变量 x_i 的解码公式为

$$x_i = 4.096 \times \frac{y_i}{1023} - 2.048 \qquad (i = 1,2) \tag{10.1}$$

例如，对个体 x：0000110111 1101110001，它由两个代码组成

$$y_1 = 55, y_2 = 881$$

上述两个代码经过解码后，可得到两个实际的值为

$$x_1 = -1.828, x_2 = 1.476$$

(5) 确定个体评价方法：由于 Rosenbrock 函数的值域总是非负的，并且优化目标是求函数的最大值，故可将个体的适应度直接取为对应的目标函数值，即

$$F(x) = f(x_1, x_2) \tag{10.2}$$

选个体适应度的倒数作为目标函数

$$J(x) = \frac{1}{F(x)} \tag{10.3}$$

(6) 设计遗传算子：选择运算使用比例选择算子，交叉运算使用单点交叉算子，变异运算使用基本位变异算子。

(7) 确定遗传算法的运行参数：群体大小 $M = 80$，终止进化代数 $G = 100$，交叉概率 $P_c = 0.60$，变异概率 $P_m = 0.10$。

上述 7 个步骤构成了用于求 Rosenbrock 函数极大值优化计算的二进制编码遗传算法。

二进制编码求函数极大值仿真程序见附录程序 chap10_1.m。仿真程序经过 100 步迭代，最佳样本为

$$\text{BestS} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

即当 $x_1 = -2.0480, x_2 = -2.0480$ 时，Rosenbrock 函数具有极大值，极大值为 3905.9。

遗传算法的优化过程中目标函数 J 和适应度函数 F 的变化过程如图 10-2 和图 10-3 所示，由仿真结果可知，随着进化过程的进行，群体中适应度较低的一些个体被逐渐淘汰掉，而适应度较高的一些个体会越来越多，并且它们都集中在所求问题的最优点附近，从而搜索到问题的最优解。

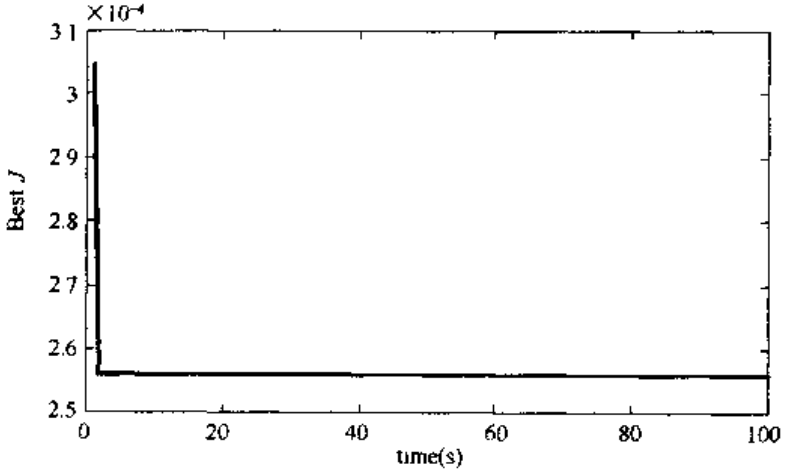


图 10-2　目标函数 J 的优化过程

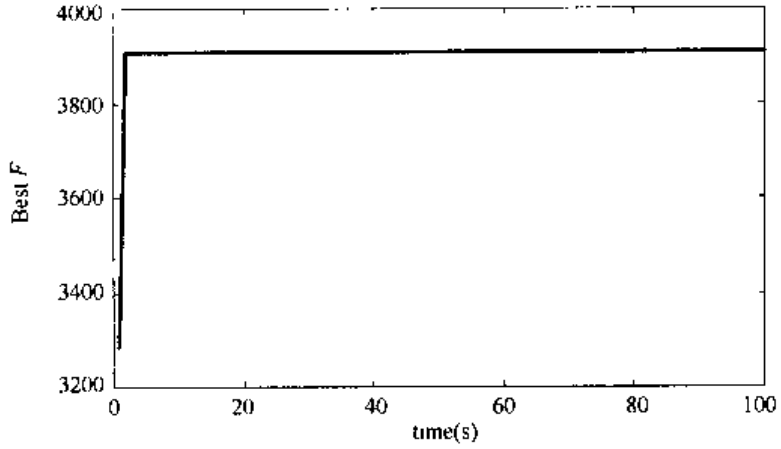


图 10-3　适应度函数 F 的优化过程

10.5.2　实数编码遗传算法求函数极大值

求解该问题遗传算法的构造过程如下：

(1) 确定决策变量和约束条件；

(2) 建立优化模型；

(3) 确定编码方法：用两个实数分别表示两个决策变量 x_1，x_2，分别将 x_1，x_2 的定义域离散化为从离散点 -2.048 到离散点 2.048 的 Size 个实数；

(4) 确定个体评价方法：个体的适应度直接取为对应的目标函数值，即

$$F(x) = f(x_1, x_2) \tag{10.4}$$

选个体适应度的倒数作为目标函数

$$J(x) = \frac{1}{F(x)} \tag{10.5}$$

(5) 设计遗传算子：选择运算使用比例选择算子，交叉运算使用单点交叉算子，变异运算使用基本位变异算子；

(6) 确定遗传算法的运行参数：群体大小 $M = 500$，终止进化代数 $G = 200$，交叉概率 $P_c = 0.90$，采用自适应变异概率 $P_m = 0.10 - [1:1:\text{Size}] \times 0.01/\text{Size}$，即变异概率与适应度有关，适应度越小，变异概率越大。

上述 6 个步骤构成了用于求 Rosenbrock 函数极大值的优化计算的实数编码遗传算法。

十进制编码求函数极大值仿真程序见附录程序 chap10_2.m。仿真程序经过 200 步迭代，最佳样本为

$$\text{Best}S = [-2.0438, -2.044]$$

即当 $x_1 = -2.0438$，$x_2 = -2.044$ 时，Rosenbrock 函数具有极大值，极大值为 3880.3。

遗传算法的优化过程中目标函数 J 和适应度函数 F 的变化过程如图 10-4 和图 10-5 所示，由仿真结果可知，采用实数编码的遗传算法搜索效率低于二进制遗传算法。

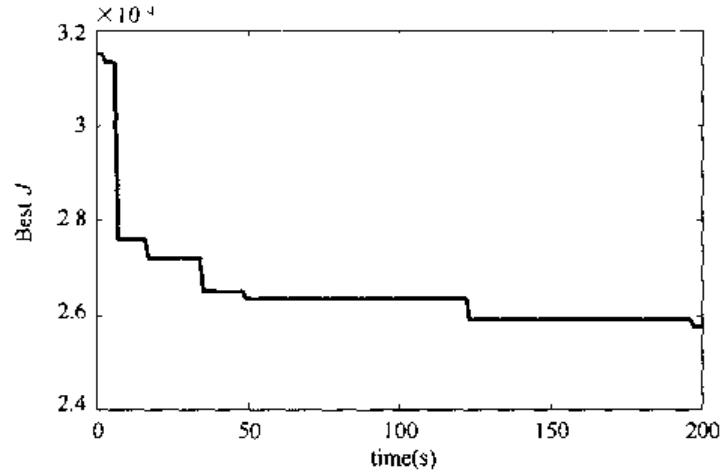


图 10-4　目标函数 J 的优化过程

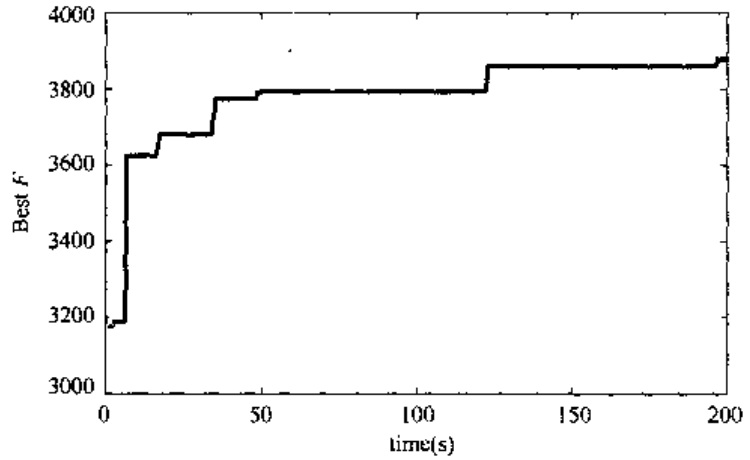


图 10-5　适应度函数 F 的优化过程

10.6　基于遗传算法优化的 RBF 网络逼近

10.6.1　遗传算法优化原理

在 7.3 节的 RBF 网络逼近算法中，网络权值 w、高斯函数的中心矢量 c 和基宽向量 b 的初值难以确定，如果这些参数选择不当，会造成逼近精度的下降甚至 RBF 网络的发散。采用遗传算法可实现 RBF 网络参数的优化。

为获取满意的逼近精度，采用误差绝对值指标作为参数选择的最小目标函数。即

$$J = 100\sum_{i=1}^{N} | e(i) | \tag{10.6}$$

式中，N 为逼近的总步骤，$e(i)$ 为第 i 步 RBF 网络的逼近误差。

在应用遗传算法时，为了避免参数选取范围过大，可以先按经验选取一组参数，然后再在这组参数的周围利用遗传算法进行设计，从而大大减少初始寻优的盲目性，节约计算量。

10.6.2　仿真实例

使用 RBF 网络逼近下列对象

$$y(k)=u^3(k)+\frac{y(k-1)}{1+y^2(k-1)}$$

在 RBF 网络中，网络输入信号为两个，即 $u(k)$，$y(k)$，网络权值及高斯函数参数值通过遗传算法优化而得。

遗传算法优化程序见附录程序 chap10_3a. m，取逼近总步骤为 $N=500$，每一步的逼近误差及目标函数由 chap10_3b. m 求得。采用二进制编码方式，用长度为 10 位的二进制编码串来分别表示向量 $\boldsymbol{b}$、$\boldsymbol{c}$ 和 $\boldsymbol{w}$ 中的每个值。

遗传算法优化中，取样本个数为 Size = 30，交叉概率为 $P_c=0.60$，采用自适应变异概率，即适应度越小，变异概率越大，取变异概率为 $P_m=0.001-[1:1:\text{Size}]\times 0.001/\text{Size}$。取用于优化的 RBF 网络结构为 2-3-1，$i=1,2$，$j=1,2,3$。网络权值 w_j 的取值范围为$[-1,+1]$，高斯函数基宽向量 b_j 的取值范围为$[0.1,+3]$，高斯函数中心矢量 c_{ij} 的取值范围为$[-3,+3]$。取 $\boldsymbol{p}=[b_1\ \ b_2\ \ b_3\ \ c_{11}\ \ c_{12}\ \ c_{13}\ \ c_{21}\ \ c_{22}\ \ c_{23}\ \ w_1\ \ w_2\ \ w_3]$，则共有 12 个参数需要优化。

经过 150 代遗传算法进化，优化后的结果为：

$\boldsymbol{p}=[2.7732,2.6343,2.2630,1.8680,-0.0616,-0.7126,-0.3959,2.2669,-1.4047,-0.3099,0.7478,-0.3353]$。

则 RBF 网络高斯基函数参数的优化值为 $\boldsymbol{B}=[2.7732,2.6343,2.2630]^{\mathrm{T}}$，$\boldsymbol{C}=\begin{bmatrix}1.8680 & -0.0616 & -0.7126\\ -0.3959 & 2.2669 & -1.4047\end{bmatrix}^{\mathrm{T}}$，权的优化值为 $\boldsymbol{W}=[-0.3099,0.7478,-0.3353]^{\mathrm{T}}$。

用于测试的 RBF 逼近算法中，取输入信号为正弦信号：$u(k)=\sin(10\pi t)$，网络隐层神经元个数取 3，网络结构为 2-3-1。网络的学习参数取 $\alpha=0.05$，$\eta=0.85$。采用优化后的网络参数作为网络的初始值，运行 RBF 网络逼近程序见附录程序 chap10_3c. m，RBF 网络逼近结果如图 10-7 所示。

RBF 网络遗传算法优化程序包括 3 部分，即遗传算法优化程序 chap10_3a. m、RBF 网络逼近函数程序 chap10_3b. m 和 RBF 网络逼近测试程序 chap10_3c. m，程序见附录。

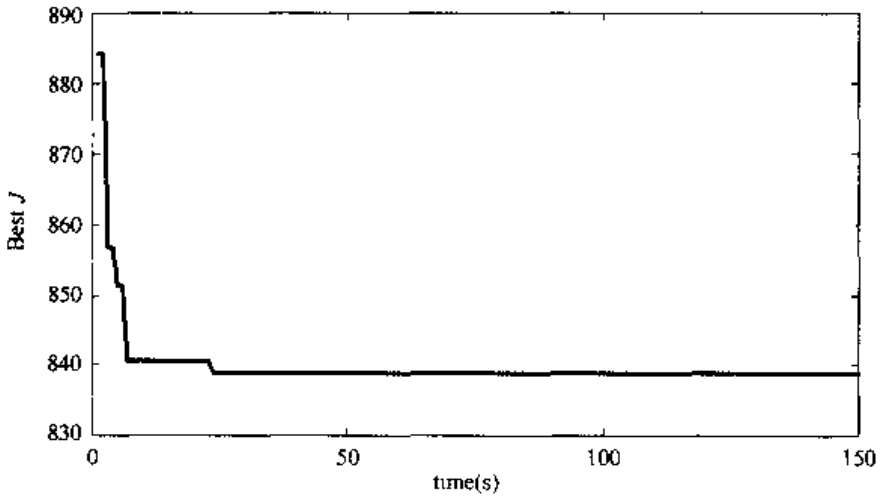


图 10-6　代价函数 J 的优化过程

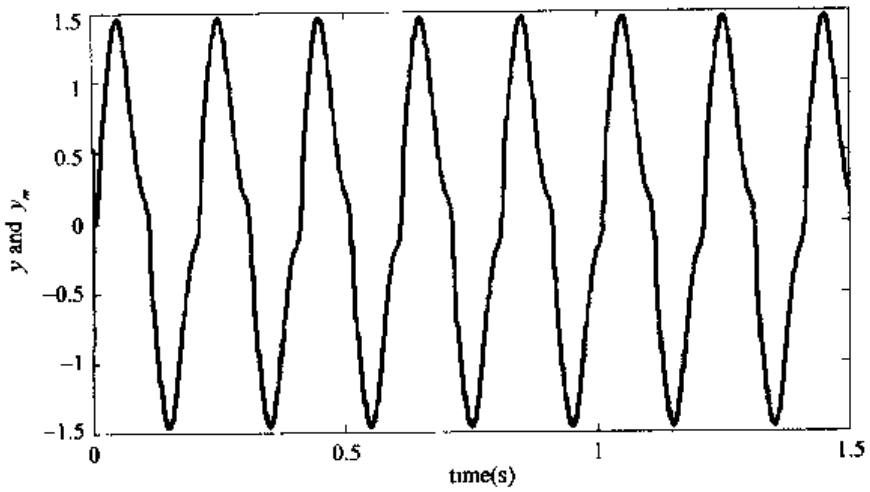


图 10-7　优化后的 RBF 网络逼近

思考题与习题

10-1　在 Rosenbrock 函数极大值遗传算法仿真程序 chap10_1. m 和 chap10_2. m 中，通过改变群体大小 M、终止进化代数 G、交叉概率 P_c 和变异概率 P_m，分析群体大小、终止进化代数、交叉概率和变异概率对优化效果的影响。

10-2　采用二进制编码方法，利用遗传算法求函数 $f(x_1, x_2, x_3)$ 的极小值。

$$\begin{cases} f(x_1, x_2, x_3) = \sum_{i=1}^{3} x_i^2 \\ -5.12 \leqslant x_i \leqslant 5.12 \qquad (i = 1, 2, 3) \end{cases}$$

10-3　在 7.2.4 节的 BP 神经网络逼近算法仿真实例中，试采用遗传算法进行 BP 神经网络学习参数及权值的优化设计，并进行 Matlab 仿真。

附录（程序代码）

遗传算法二进制编码求函数极大值程序：chap10_1.m

```
% Generic Algorithm for function f(x1,x2) optimum
clear all;
close all;

% Parameters
Size=80;
G=100;
CodeL=10;

umax=2.048;
umin=-2.048;

E=round(rand(Size,2 * CodeL));      % Initial Code

% Main Program
for k=1 : 1 : G
time(k)=k;

for s=1 : 1 : Size
m=E(s, : );
y1=0;y2=0;

% Uncoding
m1=m(1 : 1 : CodeL);
for i=1 : 1 : CodeL
    y1=y1+m1(i) * 2^(i-1);
end
x1=(umax-umin) * y1/1023+umin;
m2=m(CodeL+1 : 1 : 2 * CodeL);
for i=1 : 1 : CodeL
    y2=y2+m2(i) * 2^(i-1);
end
x2=(umax-umin) * y2/1023+umin;

F(s)=100 * (x1^2-x2)^2+(1-x1)^2;
end
```

```
Ji=1./F;
% ******Step 1 : Evaluate BestJ ******
BestJ(k)=min(Ji);

fi=F;                                  % Fitness Function
[Oderfi,Indexfi]=sort(fi);             % Arranging fi small to bigger
Bestfi=Oderfi(Size);                   % Let Bestfi=max(fi)
BestS=E(Indexfi(Size),:);              % Let BestS=E(m), m is the Indexfi belong to max(fi)
bfi(k)=Bestfi;

% ******Step 2 : Select and Reproduct Operation ******
  fi_sum=sum(fi);
  fi_Size=(Oderfi/fi_sum) * Size;

  fi_S=floor(fi_Size);                 % Selecting Bigger fi value

  kk=1;
  for i=1 : 1 : Size
      for j=1 : 1 : fi_S(i)            % Select and Reproduce
       TempE(kk, : )=E(Indexfi(i), : );
        kk=kk+1;                       % kk is used to reproduce
    end
  end

% ************Step 3 : Crossover Operation ************
pc=0.60;
n=ceil(20 * rand);
for i=1 : 2 : (Size-1)
    temp=rend;
    if pc>temp                         % Crossover Condition
    for j=n : 1 : 20
        TempE(i,j)=E(i+1,j);
        TempE(i+1,j)=E(i,j);
    end
    end
end
TempE(Size, : )=BestS;
E=TempE;

% ************Step 4: Mutation Operation **************
```

```
%pm=0.001;
%pm=0.001-[1:1:Size]*(0.001)/Size; %Bigger fi, smaller Pm
%pm=0.0;      %No mutation
pm=0.1;       %Big mutation

  for i=1:1:Size
    for j=1:1:2*CodeL
      temp=rand;
      if pm>temp                    %Mutation Condition
        if TempE(i,j)==0
          TempE(i,j)=1;
        else
          TempE(i,j)=0;
        end
      end
    end
  end

%Guarantee TempPop(30,:) is the code belong to the best individual(max(fi))
TempE(Size,:)=BestS;
E=TempE;
end

Max_Value=Bestfi
BestS
x1
x2
figure(1);
plot(time,BestJ);
xlabel('Times');ylabel('Best J');
figure(2);
plot(time,bfi);
xlabel('times');ylabel('Best F');
```

遗传算法十进制编码求函数极大值程序:chap10_2.m

```
%Generic Algorithm for function f(x1,x2) optimum
clear all;
close all;

Size=500;
```

```
CodeL=2;

MinX(1)=-2.048;
MaxX(1)=2.048;
MinX(2)=-2.048;
MaxX(2)=2.048;

E(:,1)=MinX(1)+(MaxX(1)-MinX(1))*rand(Size,1);
E(:,2)=MinX(2)+(MaxX(2)-MinX(2))*rand(Size,1);

G=200;
BsJ=0;

% ***************Start Running ***************
for kg=1:1:G
time(kg)=kg;

% ******Step 1 : Evaluate BestJ ******
for i=1:1:Size
xi=E(i,:);
x1=xi(1);
x2=xi(2);

F(i)=100*(x1^2-x2)^2+(1-x1)^2;

Ji=1./F;
BsJi(i)=min(Ji);

end

[OderJi,IndexJi]=sort(BsJi);
BestJ(kg)=OderJi(1);
BJ=BestJ(kg);
Ji=BsJi+1e-10;      % Avoiding deviding zero

fi=F;

  [Oderfi,Indexfi]=sort(fi);      % Arranging fi small to bigger
  Bestfi=Oderfi(Size);            % Let Bestfi=max(fi)
  BestS=E(Indexfi(Size),:);   % Let BestS=E(m), m is the Indexfi belong to max(fi)
```

```
  bfi(kg)=Bestfi;

  kg
  BestS
% ******Step 2 : Select and Reproduct Operation******
  fi_sum=sum(fi);
  fi_Size=(Oderfi/fi_sum)*Size;

  fi_S=floor(fi_Size);                          % Selecting Bigger fi value
  r=Size-sum(fi_S);

  Rest=fi_Size-fi_S;
  [RestValue,Index]=sort(Rest);

  for i=Size:-1:Size-r+1
    fi_S(Index(i))=fi_S(Index(i))+1;        % Adding rest to equal Size
  end

  k=1;
  for i=Size:-1:1          % Select the Sizeth and Reproduce firstly
    for j=1:1:fi_S(i)
     TempE(k,:)=E(Indexfi(i),:);          % Select and Reproduce
      k=k+1;                             % k is used to reproduce
    end
end

% ************Step 3 : Crossover Operation ************
  Pc=0.90;
  for i=1:2:(Size-1)
      temp=rand;
    if Pc>temp                              %Crossover Condition
      alfa=rand;
      TempE(i,:)=alfa*E(i+1,:)+(1-alfa)*E(i,:);
      TempE(i+1,:)=alfa*E(i,:)+(1-alfa)*E(i+1,:);
  end
end
TempE(Size,:)=BestS;
E=TempE;
```

```
% ************Step 4: Mutation Operation ***************
Pm=0.10-[1:1:Size]*(0.01)/Size;          %Bigger fi,smaller Pm
Pm_rand=rand(Size,CodeL);
Mean=(MaxX + MinX)/2;
Dif=(MaxX-MinX);

  for i=1:1:Size
    for j=1:1:CodeL
      if Pm(i)>Pm_rand(i,j)              %Mutation Condition
        TempE(i,j)=Mean(j)+Dif(j)*(rand-0.5);
      end
    end
  end
 %Guarantee TempE(Size,:) belong to the best individual
  TempE(Size,:)=BestS;
  E=TempE;
end
BestS
Bestfi

figure(1);
plot(time,BestJ,'k');
xlabel('Times');ylabel('Best J');

figure(2);
plot(time,bfi,'k');
xlabel('times');ylabel('Best F');
```

RBF 网络遗传算法优化程序:包括 3 部分,遗传算法优化程序 chap10_3a. m、RBF 网络逼近函数子程序 chap10_3b. m 和 RBF 网络逼近测试程序 chap10_3c. m。

(1)**遗传算法优化程序**:chap10_3a. m

```
%GA(Generic Algorithm) to Optimize Initial Parameters of RBF Approaching
clear all;
close all;

G=150;
Size=30;
CodeL=10;
```

```
for i=1:1:3
    MinX(i)=0.1*ones(1);
    MaxX(i)=3*ones(1);
end
for i=4:1:9
    MinX(i)=-3*ones(1);
    MaxX(i)=3*ones(1);
end
for i=10:1:12
    MinX(i)=-ones(1);
    MaxX(i)=ones(1);
end

E=round(rand(Size,12*CodeL));      % Initial Code!

BsJ=0;

for kg=1:1:G
time(kg)=kg;

for s=1:1:Size
m=E(s,:);

for j=1:1:12
y(j)=0;

mj=m((j-1)*CodeL+1:1:j*CodeL);
  for i=1:1:CodeL
  y(j)=y(j)+mj(i)*2^(i-1);
  end
    f(s,j)=(MaxX(j)-MinX(j))*y(j)/1023+MinX(j);
end

% ******Step 1 : Evaluate BestJ ******
p=f(s,:);

[p,BsJ]=chap10_3b(p,BsJ);

BsJi(s)=BsJ;
end
```

```
[OderJi,IndexJi]=sort(BsJi);
BestJ(kg)=OderJi(1);
BJ=BestJ(kg);
Ji=BsJi+1e-10;

fi=1./Ji;
[Oderfi,Indexfi]=sort(fi);          % Arranging fi small to bigger
Bestfi=Oderfi(Size);              % Let Bestfi=max(fi)
BestS=E(Indexfi(Size),:);         % Let BestS=E(m), m is the Indexfi belong to max(fi)

kg
p
BJ
% ******Step 2 : Select and Reproduct Operation******
  fi_sum=sum(fi);
  fi_Size=(Oderfi/fi_sum)*Size;

  fi_S=floor(fi_Size);          % Selecting Bigger fi value

  kk=1;
  for i=1:1:Size
   for j=1:1:fi_S(i)           % Select and Reproduce
    TempE(kk,:)=E(Indexfi(i),:);
      kk=kk+1;                 % kk is used to reproduce
      end
    end

% ************Step 3 : Crossover Operation ************
pc=0.60;
n=ceil(20*rand);
for i=1:2:(Size-1)
    temp=rand;
    if pc>temp                  % Crossover Condition
    for j=n:1:20
        TempE(i,j)=E(i+1,j);
        TempE(i+1,j)=E(i,j);
    end
    end
end
```

```
TempE(Size, : )=BestS;
E=TempE;

% ************Step 4: Mutation Operation ***************
pm=0.001-[1 : 1 : Size] * (0.001)/Size; % Bigger fi, smaller pm
  for i=1 : 1 : Size
    for j=1 : 1 : 12 * CodeL
      temp=rand;
      if pm>temp                    % Mutation Condition
        if TempE(i,j)==0
          TempE(i,j)=1;
        else
          TempE(i,j)=0;
        end
      end
    end
  end
% Guarantee TempE(Size, : ) belong to the best individual
TempE(Size, : )=BestS;
E=TempE;
% ************************************************************
end

Bestfi
BestS
fi
Best_J=BestJ(G)
figure(1);
plot(time,BestJ);
xlabel('Times');ylabel('Best J');
save pfile p;
```

(2)**RBF 网络逼近函数程序**:chap10_3b.m

```
function [p,BsJ]=rbf_gaf(p,BsJ)
ts=0.001;

alfa=0.05;
xite=0.85;
x=[0,0]';
```