基于超限学习机的半参数自适应控制研究

***

2017 年 12 月

中图分类号 TQ028.1

UDC分类号： 540

基于超限学习机的半参数自适应控制研究

作 者 姓 名 ***

学 院 名 称 自动化学院

指 导 教 师 ** 教授

答辩委员会主席 ** 教授

申 请 学 位 工学硕士

学 科 专 业 控制科学与工程

学 位 授 予 单 位 北京理工大学

论 文 答 辩 日 期 2017 年 12 月

Semi-parametric Adaptive Control Based on Extreme Learning Machine

Candidate Name:	***
School or Department:	School of Automation
Faculty Mentor:	Prof. **
Chair, Thesis Committee:	Prof. **
Degree Applied:	Master
Major:	Control Science and Enginerring
Degree by:	Beijing Insititute of Technology
The Data of Defence:	12，2017

基于超限学习机的半参数自适应控制研究

北京理工大学

研究成果声明

本人郑重声明：所提交的学位论文是我本人在指导教师的指导下进行的研究工作获得的研究成果。尽我所知，文中除特别标注和致谢的地方外，学位论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京理工大学或其它教育机构的学位或证书所使用过的材料。与我一同工作的合作者对此研究工作所做的任何贡献均已在学位论文中作了明确的说明并表示了谢意。

特此申明。

作者签名：_____　　签字日期：_____

关于学位论文使用权的说明

本人完全了解北京理工大学有关保管、使用学位论文的规定，其中包括：① 学校有权保管、并向有关部门送交学位论文的原件与复印件；② 学校可以采用影印、缩印或其它复制手段复制并保存学位论文；③ 学校可允许学位论文被查阅或借阅；④ 学校可以学术交流为目的, 复制赠送和交换学位论文；⑤ 学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

作者签名：_____　　导师签名：_____

签字日期：_____　　签字日期：_____

摘要

本文……。（摘要是一篇具有独立性和完整性的短文，应概括而扼要地反映出本论文的主要内容。包括研究目的、研究方法、研究结果和结论等，特别要突出研究结果和结论。中文摘要力求语言精炼准确，硕士学位论文摘要建议 500~800 字，博士学位论文建议 1000~1200 字。摘要中不可出现参考文献、图、表、化学结构式、非公知公用的符号和术语。英文摘要与中文摘要的内容应一致。）

关键词： 超限学习机；半参数；估计；自适应控制

Abstract

In order to exploit ·····.

Key Words: Extreme learning machine; Semi-parametric; Estimation; Adaptive control

目录

摘要 I

Abstract II

第 1 章　绪论 1

1.1　本论文研究的目的和意义 1

1.2　国内外研究现状及发展趋势 3

1.2.1　自适应控制 3

1.2.2　非线性估计与神经网络 6

1.2.3　超限学习机与控制 8

1.3　本文主要研究内容 8

第 2 章　半参数系统与信息浓缩估计 10

2.1　反馈机制能力极限的探索 10

2.2　半参数系统 12

2.2.1　模型描述 12

2.2.2　先验知识举例 13

2.3　信息浓缩估计 15

2.3.1　提出背景 15

2.3.2　主要思想 16

2.3.3　典型实现 18

2.4　本章总结 22

第 3 章　二维参数的信息浓缩估计算法 23

3.1　问题描述 23

3.2　几何关系分析 24

3.3　算法实现 27

3.4 仿真实例 29
3.5 优化问题 30
3.6 主要特点 31
3.7 本章总结 32
第 4 章 离散时间半参数自适应控制 36
4.1 问题描述 36
4.2 非参数部分的估计 37
4.2.1 最近邻估计 37
4.2.2 神经网络 38
4.2.3 ELM 及其变体 42
4.2.4 ELM 估计 44
4.3 自适应控制器设计 47
4.4 仿真实例 50
4.5 本章总结 50
第 5 章 半参数自适应运动控制 51
5.1 问题描述 51
5.2 仿真实例 51
5.3 本章总结 51
结论 52
参考文献 53
附录 A *** 56
攻读学位期间发表论文与研究成果清单 57
致谢 58
作者简介 59

第 1 章　绪论

1.1　本论文研究的目的和意义

自动控制理论的进步和发展意味着动态系统的性能更优，意味着生产力的提高，或者意味着自动化程度的提高等，因此自动控制在工程和科学领域一直都起着至关重要的作用。自动控制的基本思想甚至可以追溯到公元前 300 年左右[1]，其理论的初步形成源于在二十世纪中叶奈奎斯特、伯德、维纳等人建立的经典控制理论，其建立在频率法和根轨迹法等方法的基础上；1960 年左右，Kalman 等人提出的状态空间方法与概念标示着现代控制理论与方法的萌芽和诞生。而后，自动控制系统的运行环境越来越复杂，人们对系统控制的目标越来越多，要求也越来越苛刻。随着科学技术的发展和应用需求的不断催生，加上众多学者的不断努力，基于状态空间、传递函数等反馈理论基础和实践过程，现代控制理论迅速兴起。近年来，不论是在现代工业领域，还是在国防科技领域，现代控制理论都发挥着举足轻重的作用。线性系统理论、系统辨识理论、最优控制理论、自适应控制理论和鲁棒控制理论等重要分支的蓬勃发展，促进了自动控制在机器人伺服系统、工业过程、航空航天、人口与经济控制等领域的应用。

在经典控制理论和现代控制理论中，通常假设被控对象是线性系统并且模型的参数是精确已知的，然后以此为前提设计控制律。典型的控制方法[2, 3] 就是比例-积分-微分（PID）控制、极点配置[4]、线性二次型调节器等。其中针对单变量的 PID 反馈控制算法应用十分广泛，到目前为止，工业过程大部分也都采用 PID 控制或者其改进算法。实际系统从本质上来说都是非线性的，而非线性系统的表现形式丰富多彩，比如饱和、时滞、间隙、死区等；对这些的复杂非线性系统，目前没有很好的系统辨识方法和实际测量工具能够给出系统精确的建模结果，对系统认知的有限性和对系统模型不精确描述，造成系统普遍存在一定的不确定性；许多系统例如机器人、飞行器等运动控制系统都是强耦合的，增加了系统的不确定性。系统的非线性（nonlinearity）和不确定性（uncertainty）给分析和设计控制律带来了极大的挑战。一方面，由于系统工作环境和时间累积的影响，系统实际模型的参数和标称模型的参数总会存在不确定的误差，系统的负载和惯量也会在长时间运行之后发生变化，这些因素都导致了系统参数的不确定性，不恰当的处理会导致系统性能的降低，甚至引起系统的不稳定。另一

方面，系统中可能存在的未建模动态、执行器的非线性特性等，也会影响系统的性能，如摩擦非线性会引起伺服系统在低速运转下的不平稳，出现时滞现象；而输入间隙、死区则容易导致系统出现抖震和极限环。因此作为影响系统控制性能的重要因素，与系统不确定性特别是非线性系统的不确定性相关理论研究和实际运用一直备受关注。

随着计算机技术、网络技术的飞速发展，越来越多的控制系统采用数字微处理器作为控制器，特别是机器人伺服系统、航空航天控制系统等运动控制领域。数字化和网络化是现代伺服电机和飞行控制器等的发展趋势。机器人和飞行器等本体作为被控对象，从本质上来说都是连续时间系统，而给这些被控对象施加的都是离散控制信号，只是采样时间很短，通常在毫秒级别。为了简化，在给这些系统设计控制律的时候，常常借助于连续时间控制理论去设计，然后直接施加离散时间控制信号。这样的手段在使用过程中一般不会出现问题，但是存在着隐患，因为系统离散化后特性常常发生变化，也就是输入输出关系发生了改变，特别是非线性和不确定性等特性经过离散化变得更加不可预测。例如，摩擦、死区等非光滑非线性经过离散化之后，更加难以用数学模型精确描述或近似，增加了系统的不确定性。如果不加考虑，可能会使得系统不稳定。因此，从离散时间角度研究实际系统的控制问题更加符合真实条件。

本文将上述的非线性、不确定性和离散化导致的未知因素统一归纳为参数不确定性（parametric uncertainty）和非参数不确定性（non-parametric uncertainty）随着控制理论的不断发展，对非线性系统的不确定性的处理方法也越来越多，如鲁棒控制、自适应控制等。在这些方法中，自适应控制基于其完善的理论基础和良好的鲁棒性能，在系统不确定性的相关研究中起着极其重要的作用。自适应控制在线性系统中的应用相对来说已经比较成熟，基于连续时间对象的自适应控制算法也层出不穷。然而针对同时具有参数不确定性和非参数不确定性的离散时间自适应控制相关研究还处于探索阶段。本文将借鉴一些其他领域的思想，如机器学习、数据驱动、先验知识关联、计算几何等，同时解决存在这两种不确定性的系统控制问题，这一方向的研究具有相当大的理论意义和实用价值。

1.2 国内外研究现状及发展趋势

1.2.1 自适应控制

一般认为，自适应控制的相关理论和技术起源于二十世纪五十年代解决不同飞行高度飞行器的控制器设计问题。自适应控制是在指在被控对象的参数未知或者时变的情形下，在常规反馈控制器中引入自适应算法，动态地对控制方案进行调节，以抵消被控对象的模型时变或受到较大干扰的影响。线性系统的自适应控制方法已经比较成熟，在解决线性被控对象在参数未知和时变情形下取得了比较满意的控制效果。自适应控制的核心思想在于对系统不确定性进行估计、辨识和学习，同时设计出稳定的控制律，这使得控制器能满足系统的实际情况。辨识与控制的具体设计是实现自适应估计与控制哲学的关键步骤。模型参考自适应控制（Model Reference Adaptive Control，MRAC）和自校正控制（Self-tuning Control，STC）是解决线性系统的两类典型自适应控制方法。除了这两种写入教科书的经典自适应控制方法外，经过数十年的发展，还涌现了自适应 PID 整定、特征模型全系数自适应控制、集值系统自适应控制、L1-自适应控制、无模型自适应控制、模糊自适应控制等[5]。

除了自适应控制，在解决系统的不确定性问题时，鲁棒控制也是一种比较常用的方法。鲁棒控制方法指的是设计一种控制器, 使得当系统存在一定程度的参数不确定性和未建模动态时, 闭环系统仍能保持稳定, 并保持一定的动态性能[6]。一般来说，鲁棒控制器的设计和最终效果都依赖于对系统不确定性程度的先验假设。为了使系统稳定，常常假设系统的不确定性在较小的范围内，并且鲁棒控制器设计过程中某些项的对消或者抑制等也严格利用了系统的假设结构和精确的先验知识，这样导致鲁棒控制不能解决系统存在较大不确定性的情形。相对于鲁棒控制，自适应控制由于在反馈回路中嵌入了在线学习机制，因此能够对付较大的系统不确定性。

针对非线性特性或者非参数不确定性的辨识方法常常是设计出性能满意的自适应控制律的难点，特别是对于现代数字化控制器，在被控对象的数据经过离散采样化之后系统对应的结构和参数都存在较大不确定性。虽然经典的控制理论是从连续时间系统的研究出发，并且现代控制理论很多也是基于连续时间对象考虑的，并且在许多实际应用中取得了比较好的控制效果。但是，模拟量连续控制器由于扩展性和可重构性差，且设计复杂的控制规律时十分复杂，随着计算机技术的发展逐渐被历史淘汰。相反，数字化和离散化控制器越来越被广泛应用。实际的被控对象是离散时间系统，

这样从连续时间角度设计的控制器嵌入到实际过程中形成闭环系统后，与理论分析有较大偏差，这必然会限制实际的控制性能的提高。

经典的模型参考自适应控制主要针对连续时间系统建立。自校正控制从一开始就是针对离散时间系统而建立的自适应控制规律，即可考虑如下被控对象

$$y_{k+1} = \boldsymbol{\theta}^T \cdot \phi_k + \omega_{k+1} \tag{1.1}$$

其中，u_k，y_k 在 k 时刻的系统输入和输出；$\phi_k = [y_k, y_{k-1}, \ldots, _{k-p+1}, u_k, u_{k-1}, \ldots, u_{k-q}]^T$；$p$ 和 q 分别是系统输出和输入的阶数；$\boldsymbol{\theta}$ 是待估计的参数；ω_k 是干扰。对于(1.1)这样的被控对象，使用最小二乘算法（Recursive Least Squares, LS）去辨识未知参数 θ，或者递推最小二乘算法（Recursive Least Squares, RLS）[7] 及其诸多变体，如带遗忘因子的最小二乘算法[8]。在解决实际问题时，自校正控制采用参数辨识和必然等价原理设计自适应控制律，这是一种比较容易理解的思路。最小二乘算法主要针对线性系统设计，因而导致经典的基于 LS 的自校正调节器[9] 难以对付强非线性的被控对象，相关对比实验也表明了从线性系统角度出发设计的自适应控制在碰到存在具有强非线性的不确定性系统时的捉襟见肘[10]。这常常表现为系统的跟随性能满足不了要求，系统输出与期望存在较大偏差，或者系统难以对付较大的非线性干扰。不过，传统自校正控制算法设计中的必然等价原理对后续的自适应控制器的设计有很好的指导意义。

上述讨论的模型(1.1)常常被称为带外部输入的自回归模型（auto-regression model with extra input, ARX）[11]，其中的未知参数向量 $\boldsymbol{\theta}$ 刻画了系统的不确定性，但只是考虑到了参数不确定性。另一种模型被称为 Nonlinear ARX（NARX）模型，其单输入单输出情形的方程为

$$y_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k+1-p}, u_k, u_{k-1}, \ldots, u_{k-q}) + \omega_{k+1} \tag{1.2}$$

其中，$f(\cdot)$ 代表某个未知的非线性函数。方程(1.2)从非线性的角度刻画了系统的不确定性，将系统当作一个几乎可以说是黑箱的模型，完全不受限与系统的具体机构，直接建立系统输入输出的对应关系。除了随机干扰之外，(1.2)整个系统的不确定性主要体现在未知函数关系 $f(\cdot)$ 上，显然其囊括了线性过程(1.1)，似乎可以当作一种通用的模型回答之前的问题。然而，要解决好系统(1.2)的自适应控制问题十分困难，有赖于较好的非线性辨识方法和基于辨识结果设计出合适的控制输入 u_k 的表达式，这就涉

及到非线性离散时间自适应控制。现代非线性离散时间自适应控制方法[12–14] 也都大多针对满足一定简化条件的非线性模型或者某一特定的被控对象进行研究，针对不同的结构采用不同的方法，设计和分析都比较复杂，且通用性较差。

解决具有较大不确定性的非线性系统时，难以找到合适的参考模型与之对应以及难以满足匹配条件，就难以实施模型参考自适应控制方法。从必然等价原理出发实施自适应控制需要解决非线性系统的辨识（identification）与估计（estimate）问题。辨识高阶复杂的受控对象模型依赖于很好的辨识算法，并且，高阶复杂的系统辨识容易导致高阶复杂的控制器，而高阶复杂的控制器会给控制技术的具体实施、诊断维护、成本控制等带来一系列困难。有一些基于数据驱动的自适应控制避开了这个问题，例如，无模型自适应控制[15] 依据系统的输入输出数据设计控制律，充分利用了数据驱动的思想，不过其算法依赖于动态线性化数据模型的准确性，其假设模型具有局限性。综上所述，在伺服电机、机器人、飞行器等具有强非线性、强耦合的离散时间被控系统中，未知参数、非线性干扰、控制离散化带来的不确定性等这些问题并得到没有很好的解决。

设计上述被控对象的自适应控制器的难点在于如何解决同时存在被控系统的参数不确定性和非参数不确定性[16]。运动控制系统在高速或者超低速等情况下往往具有很大的不确定性，需要自适应反馈，诸如此类。近年来，郭雷院士等学者[17] 指出自适应反馈机制是存在极限的。受自适应反馈机制的最大能力和局限这一研究方向的启发，马宏宾等学者[10] 引入基于集合的辨识方法和数据驱动的思想，归纳出了一种等价模型，即半参数系统（semi-parametric system）用来刻画大部分非线性离散时间对象，由此提出了半参数自适应控制这一研究方向。相比于前面提到的常见自适应控制算法，半参数自适应控制出发点具有独特性，和常见的自适应控制算法的详细比较见表(1.1)。半参数自适应控制这一研究方向结合了先验知识和数据驱动的思想，设计思路不依赖于具体的被控对象，是一种比较新颖的思路。目前半参数自适应控制中的非线性部分辨识主要考虑采用的是最近邻估计（nearest-neighbor estimate），这种方法未必最优，可以引入其他非线性辨识方法作进一步的优化和扩展。半参数自适应控制目前正处于探索期，还有待于深入和推广。

表 1.1 常见自适应控制算法比较

算法名称	时间特性	适用对象	系统不确定性类别
模型参考自适应控制	大部分为连续时间	线性系统	参数
自校正调节器	大部分为离散时间	线性系统	参数
无模型自适应控制	大部分为离散时间	非线性系统	非参数
半参数自适应控制	离散时间	非线性系统	参数和非参数

1.2.2 非线性估计与神经网络

通常的自适应控制主要处理在控制科学中的参数不确定性，尤其是线性参数不确定性，然而实际系统经常是参数不确定性和非参数不确定性共存，这就给实际的应用带来了很大的困难。为了实现对含有较大非线性特性的不确定被控对象的控制，常用的做法是利用系统的输入输出数据去估计系统的非参数不确定性，为设计出满足性能要求的控制律提供依据。从数学上看，自适应控制中的辨识与估计类似于机器学习（machine learning, ML）[18]，两者的交叉点在于都含有根据数据寻找输入输出映射关系这一过程。基于数据驱动的机器学习方法以客观存在的事物为对象，研究数据的客观规律，实现数据的分类和预测。机器学习是人工智能的核心[19]，是使计算机具有智能的根本途径。自从上世纪中叶以来，机器学习在理论研究、算法设计和工程应用等方面都取得了飞速发展。

人工神经网络 (Artifical Neural Network，ANN) 是机器学习的一个庞大的分支，通常用于解决分类和回归问题。自 McCuNoch 和 Pitts 于 1943 年提出了“似脑机器”和神经网络概念以来，神经网络（(Neural Network，NN)）研究走过的是一条波浪式推进的发展道路，其在模式识别、不同的领域应用已经十分广泛 [23]，陆陆续续存在了几百种不同的算法。神经网络的基本结构示意图如图1.1所示，主要包括 1 个输入层、0 个或多个隐含层和一个输出层。

一般认为，网络结构、激活函数（activation function）和学习算法是神经网络的三大要素，神经网络的分类主要依据这三个要素。神经网络最显著的特点是具有学习能力[20]。学习过程是根据训练数据，通过不断修正神经元之间的连接权值及每个神经元的阈值来实现的。它对非线性系统和难以建模的系统的控制具有良好效果。人工神经网络的模型通常可以划分为前馈神经网络和反馈神经网络等。在实际的控制系统应用中，由于反馈神经网络运算十分耗时间，因此对于在线动力学辨识和实时控制系统应用较少。前馈神经网络在解决高度非线性和严重不确定性系统的控制方面具有很大

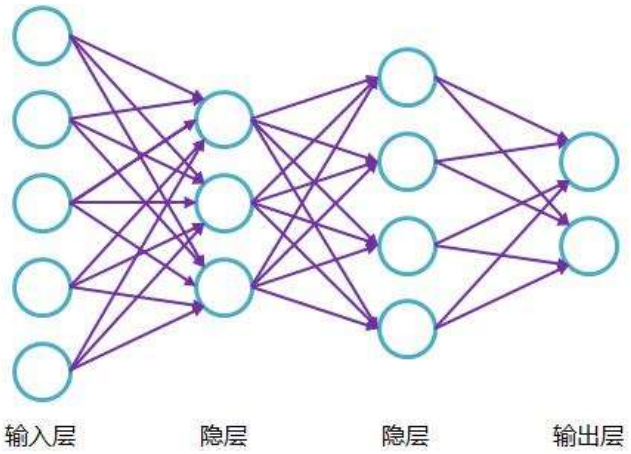


图 1.1 多层前馈神经网络示意图（含 2 个隐含层）

潜力。将神经网络引入控制系统是控制学科发展的必然趋势[21]，出现了神经网络逆控制、神经广义预测控制、自适应神经输出反馈控制和模糊神经网络控制等方法。

对于控制界，神经网络的吸引力主要在于其强大的非线性建模与辨识能力，具体体现在以下几个方面：

1. 能够充分逼近任意复杂的非线性系统；
2. 能够学习和适应严重不确定系统的动态特性；
3. 由于大量神经元之间广泛连接，即使少量神经元或连接损坏，也不影响系统的整体功能，表现出很强的鲁棒性和容错性；
4. 采用并行分布处理方法，使得快速进行大量运算成为可能。

神经网络的引入不仅给这一控制领域的突破带来了生机，也带来了许多亟待解决的问题。例如，尽管现在有很多神经控制算法，但绝大多数没有闭环系统稳定性分析；被控对象特性的不断变化和控制效果要求的不断提高使得一些传统神经控制难以达到理想的效果；快速在线学习算法是实时控制的关键，而目前大部分神经网络采用的误差反向传播（Back Propagation，BP）学习算法计算量大。因此，开发新型神经网络和快速在线学习算法并将其用于控制系统将是基于神经网络的自适应控制研究的重要方向。

1.2.3 超限学习机与控制

如前所述，目前的神经网络结构和在线学习算法难以满足控制系统的要求。传统的神经网络需要人为设置大量的网络参数；在网络训练时主要采用 BP 算法，需要耗费很长的时间多次迭代才能修正权值和隐元的偏置[22]；由于使用梯度下降有陷入局部极小的缺陷，而未到达全局最小。一些研究学者也开始研究基于神经网络的实时辨识与控制算法[23]，但针对的是特定的问题，难以做到通用性。

2004 年，黄广斌等人提出了一种新型的单隐层前馈神经网络[24, 25]，即超限学习机（Extreme Learning Machine, ELM）。基本的超限学习机仅需设置网络隐层节点个数，在算法执行过程中不需要调整网络的输入权值以及隐层神经元的偏置，就能产生唯一的最优解，因此具有学习速度快且泛化性能好的优点。ELM 的实现过程[26] 主要分为三个阶段，首先初始化输入权重和隐元的偏置，然后根据输入计算出隐含层的输出，最后由期望的范数解确定输出权重，一般采用二范数，即采用最小二乘法确定。相对传统网络在保证更好的泛化性的同时具有极快的学习速度，且可以克服传统梯度算法常有的局部极小、学习时间长、性能指标及学习率等问题，超限学习机表现出了对解决非线性问题极好的应用性能。

超限学习机是一种简单易用、有效的前馈神经网络学习算法，在手写识别、目标跟踪等等领域[27] 取得了很大成就。它具有很好的系统辨识能力，在控制领域也出现了不少应用[28]，特别是在线序列超限学习机（Online-sequential ELM，OS-ELM）[29] 及其相关变体[30, 31]，很适合解决具有强非线性的不确定性系统的辨识与控制问题[32, 33]。

1.3 本文主要研究内容

本文的主要研究对象是同时具有参数不确定性和非参数不确定性的离散时间系统，结合超限学习机对现有的半参数自适应控制进行扩展和改进。主要体现在：

本文的章节安排如下：

第一章是绪论部分，主要阐述了非线性自适应控制的研究背景和意义，总结了经典自适应控制方法的发展历程、特点与方法和存在的问题等，分析了神经网络应用于自适应估计与控制的前景和挑战，并介绍了一些新型有效的非线性辨识方法。

第二章，半参数系统与信息浓缩估计。对半参数自适应思想的来源作进一步阐述，正式描述了半参数系统的数学模型，详细介绍了信息浓缩估计的思想和具体实施

方法，对原有的一阶半参数自适应估计进行了扩展。

第三章，二维参数的信息浓缩估计算法。

第四章，离散时间半参数自适应控制。

第五章，半参数自适应运动控制。

第 2 章　半参数系统与信息浓缩估计

传统鲁棒控制主要考虑小范围的结构不确定性，传统自适应控制主要考虑系统的参数不确定性，而一般的非线性自适应控制直接考虑非线性模型从而设计复杂的控制器。实际问题中参数不确定性和较大的结构不确定性可同时出现，因此这些方法对于解决非线性不确定性被控对象的离散时间控制问题都存在一定的局限性。所有的被控对象本质上都是非线性的，但是线性控制方法在反馈控制前期发展中起到了很大作用。这不由得让人思考，是否很多非线性对象都存在一个起主要作用的线性模型？按照这个思路，虽然无法精确的已知非线性不确定性系统的离散化模型，但是这些非线性系统可以用一个线性模型（如 ARX 模型）加上一个非线性部分（关于输入输出数据的未知函数）近似，也就是把系统的不确定性分为参数和非参数不确定性。为了解决控制问题，需要先辨识参数部分。这就是本章要介绍的半参数系统和所谓的信息浓缩（information concentration estimate, IC）估计。

2.1　反馈机制能力极限的探索

不管是采用连续时间模型还是离散时间角度设计控制律，实际系统都会存在无法预测的不确定性，如干扰、未建模动态等，这会导致输出就会产生偏差。反馈从一开始就是为了解决这种偏差和不确定性而引入的。传统针对线性时不变系统设计的控制方法属于线性反馈。然而，即便是针对 ARX 模型设计的自校正调节器，自适应控制从本质上说是一种非线性反馈[34]。当系统存在不确定性时，在设计自适应辨识与控制算法时，是否一定存在一种自适应控制律能够使得系统闭环稳定？这是在设计自适应控制之前首先要回答的问题。然而直到 20 世纪 90 年代末，郭雷院士等学者开始探索反馈机制[35]，创造性地提出反馈机制的最大能力和局限（the limit to the capability of feedback）这一重要命题，并针对一些基本控制系统的自适应反馈极限进行定量研究，才引起来广泛的关注。

最开始关于反馈机制的研究始于一类基本的不确定非线性系统，即下面的模型

$$y_{k+1} = \theta\phi_k + u_k + \omega_{k+1},\ \phi_k = O(y_k^b),\ b > 0 \tag{2.1}$$

其中，u_k 和 y_k 分别是系统在 k 时刻的输入和输出，一般假定 ω_k 为高斯白噪声干扰。θ

是未知参数，b 刻画了系统非线性的增长速率。除了参数不确定性 θ 外，系统(2.1)的不确定性主要表现为 ϕ_k 的未知性，即非参数不确定性。经过严格的数学证明，当 $b \geq 4$ 时，不存在任何的自适应反馈控制律使得系统(2.1)满足全局稳定。进一步考虑如下的系统：

$$y_{k+1} = f(y_k) + u_k + \omega_{k+1},\ \dot{f()} \in F(L) \tag{2.2}$$

其中，$f(\cdot)$ 是未知函数，$F(L)$ 代表一类满足 Lipschitz 条件的非线性函数。对于这种非参数不确定性系统，有如下结论[36]：如果 $L < \frac{3}{2} + \sqrt{2}$，则存在反馈控制律使得系统(2.2)全局稳定；相反，若 $L \geq \frac{3}{2} + \sqrt{2}$，则不存在反馈控制律使得系统(2.2)全局稳定。$f(\cdot)$ 的绝对值随自变量 y_k 绝对值的增长速率决定了反馈机制能力的上限。

上述这些现象在连续时间非线性系统中至今没有表现出来，这说明离散时间控制研究的必要性。这些“不可能性”（impossibility）定理表明了，在离散时间非线性系统的自适应控制中反馈机制存在极限。在随后的研究[37, 38] 中将这“不可能性”定理一推广到了高阶情形，其中 $f(\cdot)$ 就推广到了多元函数，同样是满足 Lipschitz 条件。可以这样理解，如果 $f(\cdot)$ 增长的比较快，那么就不存在使得系统(2.1)趋于稳定的自适应反馈控制律。分析出反馈机制的上下界并给出数学证明，确实不是一件容易的事情，需要极强的数学技巧。这一理论很直观地说明了系统的不确定性的大小常常表现为系统未知函数或者参数的变化剧烈程度，这些结论从控制稳定性和可行性的角度对系统的不确定性给出了一定的定量描述。

目前关于反馈机制及能力的研究主要针对离散时间控制系统[39]，这与离散时间控制器的广泛应用这一大趋势不谋而合，并且一开始就考虑到了非参数不确定性，因而具有较好的理论高度和普适性。反馈机制及能力给出了对不确定性系统的认识，从理论上看，不是所有的未知系统都可以设计出稳定的自适应控制律[40]；用但是同时也留下了一个问题：如果系统属于自适应反馈控制可稳定的范围内，那么如何设计出合适的自适应反馈控制律使系统达到满意的性能？一般来说，不同的被控对象自然结构和参数都不同，但是否存在一种较为通用的模型可以刻画大部分面对的被控对象？可以类比经典控制理论，毕竟线性系统和微分方程这些通用工具在几十年的控制理论中起到了关键的作用。类似这样的问题促进了半参数模型的提出。

2.2 半参数系统

2.2.1 模型描述

虽然数字化控制器输出到被控对象的信号是离散时间，但是实际的过程是连续时间变化的。一般的离散时间非线性控制往往忽略了这一点，就导致这些研究侧重于理论和离散时间仿真分析。再次考虑下面的 NARX 系统

$$y_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k+1-p}, u_k, u_{k-1}, \ldots, u_{k-q}) + \omega_{k+1} \tag{2.3}$$

在系统(2.3)中，把被控对象当成完全非线性模型，忽略了他们的线性特性，也就是说很多被控对象虽然具有较大的不确定性，但是在一定程度上是可以当作线性对象处理，只不过同时具有非线性部分，导致实际的线性部分参数与理论值有较大的偏差；对于不确定性系统，也就常常表现为同时具有参数不确定性和非参数不确定性。例如电机伺服系统简化处理就是一个二阶线性模型，一般的 PID 控制也可以获得还算不错的效果；只不过是在负载未知、快时变，或者一些难以建模的非线性动态特性如摩擦等严重的情况下，应用传统的 PID 控制电机伺服系统时，其效果才会大打折扣。

上述事实说明了实际系统表现出的线性特性有助于解决非线性自适应控制问题，不可忽视。这样看来，一般的非线性系统都可以看作是由线性部分和非线性部分组成。也就说是，在分析具有强非线性的不确定离散时间被控系统时，要同时考虑系统存在的参数不确定性和非参数不确定性。受反馈机制能力与极限等相关研究的启发，用半参数系统刻画非线性离散时间被控对象，然后基于半参数系统建立自适应控制律，是一种很自然的思路。半参数系统的模型并不唯一。

首先考虑如下一类半参数模型

$$y_{k+1} = \boldsymbol{\theta}^T \boldsymbol{\phi_k} + u_k + f(\boldsymbol{\psi_k}) + \omega_{k+1} \tag{2.4}$$

其中，y_k，u_k 分别表示系统在 k 时刻的输出和输入，由它们的历史数据组成的回归向量 $\boldsymbol{\phi_k}$ 和 $\boldsymbol{\psi_k}$ 具体定义如下:

$$\boldsymbol{\phi_k} = [y_k, y_{k-1}, \ldots, y_{k-p_1}, u_{k-1}, \ldots, u_{k-q_1+1}]^T \tag{2.5}$$

$$\boldsymbol{\psi_k} = [y_k, y_{k-1}, \ldots, y_{k-p_2}, u_{k-1}, \ldots, u_{k-q_2+1}]^T \tag{2.6}$$

参数部分和非参数部分涉及的回归向量不一定相同，因此分别定义更加具有概括性。再记

$$d_1 = p_1 + q_1 \tag{2.7}$$

$$d_2 = p_2 + q_2 \tag{2.8}$$

分别代表了回归向量 $\boldsymbol{\phi_k}$ 和 $\boldsymbol{\psi_k}$ 的维数。

系统的输出 y_k 对应到实际的伺服电机系统中可能就是需要控制的电机转速或者位置等物理量，而 u_k 可能就是控制电流（或者控制力矩）等。$\boldsymbol{\theta} \in \Theta$ 是未知参数向量，$f(\cdot) \in \mathcal{F}$ 是未知函数，$\omega_k \in \Delta_k$ 是外部随机干扰；这里 Θ，$\mathcal{F}$，Δ_k 分别代表了参数不确定性 $\boldsymbol{\theta}$，非参数不确定性 $f(\cdot)$ 和随机干扰 ω_k 的先验知识。他们都是集合的形式，即 F 是函数集合、Θ 是向量集合、Δ_k 是普通的数值集合。

上述的模型(2.4)直观地反映了被控系统的参数不确定性和非参数不确定性，综合性和普适性较强。关于不确定性的先验知识也是半参数系统中的组成部分。下面分别对这三个量从数学关系上描述一些先验知识的例子。

2.2.2 先验知识举例

2.2.2.1 未知参数

记

$$\boldsymbol{\theta} = [\theta_1, \ldots, \theta_p]^T \tag{2.9}$$

未知参数向量 $\boldsymbol{\theta}$ 具有很强的物理意义，例如对应到伺服电机系统中，$\boldsymbol{\theta}$ 中的某些分量可能就表示电机的力矩常数、电枢回路常数等。$\boldsymbol{\theta}$ 可能存在如下的某种先验知识：

- 除了向量的元素个数外，没有其他任何关于未知参数的信息如上下界等；
- $\boldsymbol{\theta}$ 各分量的上下界已知，即 $\underline{\theta}_n \leq \theta_n \leq \overline{\theta}_n$，其中 $\underline{\theta}_n$ 和 $\overline{\theta}_n$ 为已知的常数；
- 未知参数向量 $\boldsymbol{\theta}$ 和标称向量 $\boldsymbol{\theta}_0$ 之间的距离取值在一个范围内，即 $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| \leq r_0$，这里 r_0 是一个已知的常数。
- 未知参数向量的取值有限可数,即 $\boldsymbol{\theta}$ 属于某一个已知的集合,$\boldsymbol{\theta} \in \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \ldots, \}$;
- ...

2.2.2.2 未知函数

未知函数 $f(\cdot)$ 在实际系统如电机物理系统中，对应可能就是难以建模的非线性摩擦力（力矩）、高频振动特性等。$f(\cdot)$ 可能存在如下的某种先验知识：

- $f(x)=0$ 对于任意的自变量 x，这意味着系统不存在任何未建模动态特性；
- $f(x)$ 的值受限于其他已知函数，即 $\underline{f}(x) \leq f(x) \leq \overline{f}(x)$，其中 $\underline{f}(x)$ 和 $\overline{f}(x)$ 为已知函数；
- 未知函数 $f(x)$ 和标称函数 $f_0(x)$ 之间的距离在某一个已知范围内内，即 $\|f(x)-f_0(x)\| \leq r_f$，这里 $r_f \geq 0$ 是一个已知的常数，这里函数 $f_0(x)$ 可以认为是定义在赋范可测函数空间集合 { 的中心；
- 未知函数的表达式有限可数，即 $f(x)$ 属于某一个已知的函数集合，$f(x) \in \{f_1(x), f_2(x), f_3(x), \ldots,\}$。
- 未知函数 $f(x)$ 是满足 Lipschitz 定义的，即存在某个常数 $L_f \geq 0$，使得 $\|f(x_1)-f(x_2)\| \leq L\|x_1-x_2\|$ 成立；
- 未知函数 $f(x)$ 是相对于自变量来说是单调递增（或者递减）的；
- 未知函数 $f(x)$ 是凸函数（或者凹函数）；
- ...

2.2.2.3 随机干扰

随机干扰一般来源于测量误差、未知干扰等，ω_k 可能存在如下的先验知识：

- $\omega_k=0$ 对于任意的 $k \geq 0$，这意味着系统不存在随机干扰；
- ω_k 的取值在一个已知的范围内，即上下界已知，$\underline{\omega} \leq \omega_k \leq \overline{\omega}$；
- ω_k 的取值受限于其他已知的干扰序列，即 $\underline{\omega}_k \leq \omega_k \leq \overline{\omega}_k$；
- ω_k 的取值都来自于一个已知的集合，即 $\omega_k \in \{e_1, e_2, \ldots, e_N\}$；

- ω_k 的统计特性符合某种概率分布；例如 ω_k 是高斯随机过程，则 $E(\omega_k) = 0$，$E(\omega_k^2) = \sigma^2$；又比如 ω_k 的取值满足均匀分布，即 $\omega_k \sim U(0,1)$；
- ω_k 是一个衰减的序列，如 $\omega_k = \frac{1}{k^2}$；
- ...

2.3 信息浓缩估计

2.3.1 提出背景

按照必然等价原理，为了控制系统(2.4)，需要辨识其中的未知参数向量 $\boldsymbol{\theta}$。文献中已经存在很多经典的参数估计算法，例如递推最小二乘算法（Recursive Least Squares, RLS）、卡尔曼滤波（Kalman filter, KF）算法、最小均方算法（Least Mean Squares, LMS）等。不过这些算法大部分都是基于线性系统建立的估计算法。由于系统存在非参数部分 $f(\cdot)$，因此不能直接使用。实验数据[10] 表明，在非线性对象中盲目使用基于线性系统的参数估计算法会导致估计结果偏差较大，特别是在强非线性系统中。

被研究的系统(2.4)经常存在随机干扰。在系统辨识、信号处理、随机逼近、状态估计以及传统的自适应控制等领域中[8]，基于噪声或者干扰统计特性的参数估计算法应用十分广泛。随机干扰是系统的一种不确定性，为了描述和设计相应的参数估计算法，总是假设被研究的系统中的随机干扰符合某种统计特性。这是利用先验知识转化为可描述的数学关系的方法。本文在2.2.2.3中也把统计特性作为先验知识的例子说明。事实上，随机干扰可能存在若干种来源，导致这些干扰或误差的概率分布（如果存在）并不符合模型假定的特性，最终使得这些估计算法在实际应用过程中失效。例如，如果对随机干扰的先验知识没有深入了解或测量，一种普遍的做法是假设其干扰服从某种高斯分布。这种做法在很多情况下确实有效，结果较为准确，但是对于有些情况可能就会偏离很远。为了使结果准确，可能需要调节高斯分布的方差等参数。

在实际工程中，随机干扰的数值往往较小（如果较大，就得考虑把他当作非线性部分了），且一般可以不会超过某个已知范围，对应2.2.2.3中的第 2 个例子。这不是一种基于概率统计的先验知识，能够比较准确地反映真实的情况。比如在某个伺服电机系统中的电磁干扰导致的脉动力矩，一般不会超过某个幅值。和随机干扰类似，系统的非参数部分 $f(\cdot)$ 部分的先验知识也常常是一个范围，即区间的形式。比如经过理论

分析，电机的摩擦力不会超过某个数值，多关节机器人运动控制中重力这一非线性项是一个有限值等情形。

因此由于上面种种因素的存在，需要另辟蹊径解决系统(2.4)的参数估计问题，传统的基于数值的估计算法难以利用基于集合的先验知识。

2.3.2 主要思想

为了实现参数估计，需要综合充分利用系统的先验知识。从2.2.2.1到2.2.2.3介绍的先验知识可以看出，对参数向量 $\boldsymbol{\theta}$、非参数部分 $f(\cdot)$、随机干扰 ω 等不确定性的等价数学描述一般都是表现为集合的形式，因此设计基于集合的参数估计是一种自然的思路。文献 [10] 中提出的信息浓缩估计就是这样的一类估计算法。

信息浓缩估计是一种针对离散时间系统的基于集合的参数估计器，其主要思想是从初始的关于参数的先验集合出发，将每一时刻的先验知识都看作是对未知函数和参数的数学约束，然后结合采集到的真实的输入输出数据剔除掉不符合真实数据的取值，逐渐缩小这个先验集合的大小，也就减少了系统的参数不确定性；这样经过足够长的时间和数据筛选之后，理论上可以从参数集合中选择任意值或者根据一定的规则选择最优值作为最终的参数估计。

下面基于数学语言详细解释息浓缩估计器的具体思路。这里再次考虑系统

$$y_{k+1} = \theta^T \boldsymbol{\phi_k} + u_k + f(\boldsymbol{\psi_k}) + \omega_{k+1}, \tag{2.10}$$

并记

$$\begin{aligned} z_k &= f(\psi_k) + \omega_{k+1} \\ &= y_k - \psi_{k-1}^T \theta - u_{k-1} \end{aligned} \tag{2.11}$$

这个被控对象具有先验知识 $\theta \in \Theta \subseteq \mathcal{R}^d$，$z_k \in V_k$。第一项是关于参数的先验集合，可以作为参数在初始状态的估计，记作

$$I_0 = \Theta \tag{2.12}$$

这样，在时刻 k，随着 ϕ_k、y_k 和 u_k 等数据的获得（经过简单计算也就获得了 z_k），可

以定义如下所谓的信息集合 I_k 的表达式为

$$I_k \triangleq \{\theta \in \Theta \colon y_k - \phi_{k-1}^T \theta - u_{k-1} \in V_k\} \tag{2.13}$$

然后定义如下所谓的信息浓缩集合

$$C_k = \bigcap_{j=0}^{k} I_k \tag{2.14}$$

写成递推的形式为

$$C_k = C_{k-1} \bigcap I_k \tag{2.15}$$

可以定义初始的集合为

$$C_0 = I_0 = \Theta \tag{2.16}$$

其中，C_k 也就是每个时刻 k 的参数集，可以从中选择任意值作为参数估计参与后续计算，不过对于这种区域形式，一般选择中心点作为每个时刻的参数估计值相对比较准确。

命题 2.1. 经过分析，容易得到信息浓缩估计器有如下数学性质：

1. 信息集合 I_k 的定义跟具体的模型有关，形式不唯一；
2. 单调性，即相邻时刻的参数估计集合之间满足如下关系：

$$C_0 \supseteq C_1 \supseteq C_2 \supseteq \ldots; \tag{2.17}$$

3. 收敛性，经过足够长的时间后，参数估计集合序列 C_k 满足：

$$C_\infty = \bigcap_{k=0}^{\infty} C_k; \tag{2.18}$$

4. 集合 C_∞ 非空的充分必要条件是，系统的模型和相应的先验知识以及参与计算的输入输出数据是完全匹配的。

命题2.1第三条的具体含义是：如果模型和相应的先验知识以及参与计算的数据是完全正确的，那么 C_∞ 必定是一个非空集合，并且满足真实的参数向量 $\boldsymbol{\theta} \in C_\infty$，集

合 C_∞ 的任意一个元素都可以匹配模型和数据；反过来，也成立。但是，如果 $C_\infty = \varnothing$ 说明参与计算的数据、模型设置或者先验知识中某个部分存在问题，不是相对应的，那么就会导致结果出现偏差。这个充要条件一般来说比较容易满足的，可以把先验知识设定足够宽泛，并保证参与计算的数据都来源于真实的系统，至于野值的出现属于数据和信息预处理的问题，就不是本文要考虑的范畴了。

2.3.3 典型实现

从上面一小节2.3.2关于信息浓缩估计器的描述和举例中，可以看出信息浓缩估计是一种在线实时估计器，充分利用了系统的先验知识和实时数据，因此也可以看作是一种数据驱动型的估计算法。随着采集到的数据不断增多，未知参数的不确定性就越来越小，我们对参数的估计就越来越准确。系统关于参数的先验知识不同，会导致信息集合 I_k 和信息浓缩之后的参数估计集合 C_k 不同，因而关于其中的递归运算(2.15)也就不同。

有一种十分典型的情形，如同在上述的提出背景2.3.1中描述的 $z_k \in V_k$ 是区间的形式。具体数学描述为

$$\underline{\theta}_n \leq \theta_n \leq \overline{\theta}_n \tag{2.19}$$

$$\underline{z} \leq z_k \geq \overline{z} \tag{2.20}$$

这里，z_k 是(2.11)定义的非参数部分 $f(\cdot)$ 和随机干扰 ωk 的和，z_k 的上下界也就是这两个量的上下界的组合。为了便于后面的计算，这里直接考虑他们的组合形式 z_k。

本节首先考虑这种先验知识的系统。文献 [10] 设计了一维情形下 $d_1 = 1$ 的参数估计算法，即 $\boldsymbol{\theta}$ 是标量的情形。信息浓缩估计器主要就是求出每一步的参数估计集合 C_k。

2.3.3.1 一维参数算法

这里首先给出 d_1 为 1 时，C_k 的推导过程。当 $d_1 = 1$ 时，θ 和 ϕ_k 都是标量。由方程(2.13)可知

$$I_k \triangleq \{\theta \in \Theta \colon \underline{z} \leq y_k - \phi_{k-1}\theta - u_{k-1} \leq \overline{z}\} \tag{2.21}$$

解上面的不等式得到

$$y_k - u_{k-1} - \overline{z} \leq \theta\phi_{k-1} \leq y_k - u_{k-1} - \underline{z}. \tag{2.22}$$

如果 $\phi_{k-1} \neq 0$，则

$$\theta \in [\underline{s}_k, \overline{s}_k] \tag{2.23}$$

其中定义

$$\begin{aligned} \underline{s}_k &= \frac{\min(\text{sign}(\phi_{k-1})(y_k - u_{k-1} - \overline{z}), \text{sign}(\phi_{k-1})(y_k - u_{k-1} - \underline{z}))}{\phi_{k-1}} \\ \overline{s}_k &= \frac{\max(\text{sign}(\phi_{k-1})(y_k - u_{k-1} - \overline{z}), \text{sign}(\phi_{k-1})(y_k - u_{k-1} - \underline{z}))}{\phi_{k-1}} \end{aligned} \tag{2.24}$$

sign(·) 是常见的符号函数：

$$\text{sign}(x) = \begin{cases} 1, \ x > 0 \\ 0, \ x = 0 \\ -1, \ x < 0 \end{cases} \tag{2.25}$$

一般来说，C_0 直接由先验知识 Θ 给出一个区间范围。在时刻 k，由方程(2.14)得到信息浓缩集合（参数集）为

$$C_k = [\underline{h}_k, \overline{h}_k] \tag{2.26}$$

其中 $\underline{h}_k$ 和 $\overline{h}_k$ 的递推表示形式为

$$\begin{aligned} \underline{h}_k &= \max(\underline{h}_{k-1}, \underline{s}_k) \\ \overline{h}_k &= \min(\overline{h}_{k-1}, \overline{s}_k) \end{aligned} \tag{2.27}$$

经过上面的推导，(2.24)，(2.26)和(2.27)就是 $d = 1$ 情形下的信息浓缩参数算法的具体公式，上面的过程可以表示为图2.1。

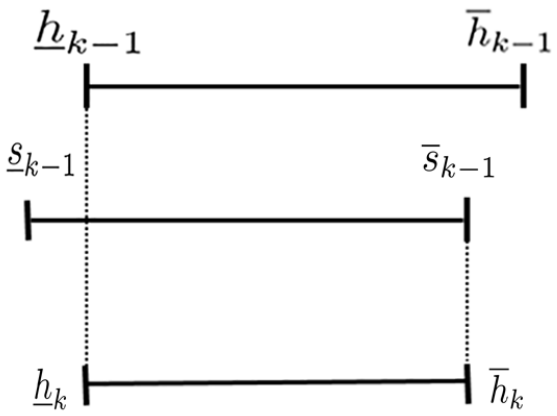


图 2.1 $d_1 = 1$ 时，信息浓缩过程的几何解释

2.3.3.2 多维参数情形

当 $d_1 > 1$ 时，$\boldsymbol{\theta}$ 和 $\boldsymbol{\phi_k}$ 都是向量，和方程(2.21)类似可以得到

$$\boldsymbol{\phi}_{k-1}^T \boldsymbol{\theta} \in [y_k - u_{k-1} - \overline{z}, y_k - u_{k-1} - \underline{z}]. \tag{2.28}$$

上面的区间形式可以写成下面关于参数向量 $\boldsymbol{\theta}$ 的两个不等式

$$\begin{aligned} \boldsymbol{\phi}_{k-1}^T \cdot \boldsymbol{\theta} &\leq y_k - u_{k-1} - \overline{z} \\ (-\boldsymbol{\phi}_{k-1}^T) \cdot \boldsymbol{\theta} &\leq -(y_k - u_{k-1} - \underline{z}) \end{aligned} \tag{2.29}$$

从几何上看，方程组(2.29)中的任意一个线性不等式的解，都代表着一个定义在域 $\mathcal{R}^d$ 上的广义半平面。因此，通常来说，关于同一向量 $\boldsymbol{\theta}$ 的多个线性不等式的解在在几何上表示为一个广义多边形或者多面体，仅需要找出这个多面体的顶点（向量描述）就可确定这个多面体。特别的，$d = 2$ 时，多个方程组(2.29)的解就是平面上的一个普通多边形，确定了这个多边形的顶点就等价于确定了参数满足的集合 C_k。

记

$$\Gamma_k = \{P_n, n = 1, 2, \ldots, N_k\} \tag{2.30}$$

为定义在区域 C_k 上的多面体顶点集合，其中 P_n 代表某个顶点，第 k 时刻集合 Γ_k 包含 N_k 个顶点。这里 Γ_k 记为一个多面体。这里的顶点集情况是对于 $d_1 > 2$ 时有意义。

下面先给出多维情形的一般公式。

由方程(2.14)可知，从约束条件来看，集合 C_k 比 C_{k-1} 多了一组不等式

$$\begin{aligned} \boldsymbol{\phi}_{\boldsymbol{k-1}}^T \cdot \boldsymbol{\theta} &\leq e_{k,1} \\ (-\boldsymbol{\phi}_{\boldsymbol{k-1}}^T) \cdot \boldsymbol{\theta} &\leq e_{k,2} \end{aligned} \tag{2.31}$$

其中

$$\begin{aligned} e_{k,1} &= y_k - u_{k-1} - \underline{z} \\ e_{k,2} &= -(y_k - u_{k-1} - \overline{z}) \end{aligned} \tag{2.32}$$

本文称(2.31)为线性约束。和 $d_1 = 1$ 情形类似，可以得到信息约束的递推形式为

$$\begin{aligned} \Gamma'_{k-1} &= G(\Gamma_{k-1}, \boldsymbol{\phi_{k-1}}, e_{k,1}) \\ \Gamma_k &= G(\Gamma'_{k-1}, \boldsymbol{\phi_{k-1}}, e_{k,2}) \end{aligned} \tag{2.33}$$

这里 $G(\Gamma, \boldsymbol{\phi}, f)$ 代表一种变换（算法），它表示在初始的多面体 Γ 中加入一个线性约束条件 $\boldsymbol{\phi} \cdot \boldsymbol{\theta} \leq e$。

图2.2展示了 $d_1 = 2$ 时 G 变换的过程。图中，原有的参数区域是由顶点集

$$\{P_1, P_2, P_3, P_4, P_5\} \tag{2.34}$$

围成的多边形，加入线性约束条件 $\boldsymbol{\phi} \cdot \boldsymbol{\theta} \leq e$ 之后，变成了顶点集

$$\{P'_1, P'_2, P_3, P_4, P_5\} \tag{2.35}$$

围成的多边形（深色部分）。

可以看出，多边形的面积在原来的基础上缩小了，即参数的可行域减少了，参数不确定性随着也就减少了。不过图2.2只是线性约束和多边形位置的一种情况，因为很可能直线 L 和多边形没有交点，那么经过这个时刻的信息浓缩后，参数集合不变。因此在设计算法实现时需要考虑周全，否则可能导致 $C_\infty = \varnothing$ 的出现。

$d_1 > 2$ 时高维情形的情况类似，只是多边形变成了多面体，直线变成了广义半平面，由于多维情形不容易形象描述，此处就不再一一画出图形说明，留给读者自己想

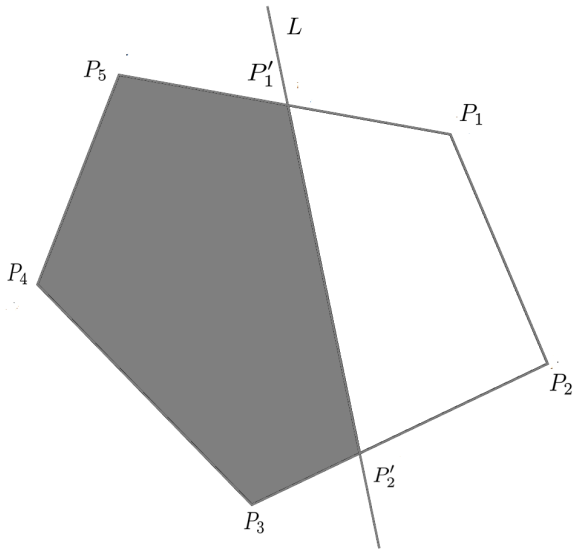


图 2.2 $d_1 = 2$ 时，信息浓缩 G 变换过程的几何解释

象。

2.4 本章总结

本章的主要内容是半参数系统和信息浓缩估计的基本思想。首先由关于已有的反馈机制能力极限的探索引出存在的不确定性系统的自适应控制问题；然后从实际问题抽象出半参数系统的基本模型，介绍了相关的先验知识举例；接着介绍了一种基于集合的参数估计算法即所谓的信息浓缩估计的主要思想；最后针对典型系统介绍了信息浓缩估计的具体实施框架。

第 3 章　二维参数的信息浓缩估计算法

设计高阶情形下基于信息浓缩思想的参数估计算法是实现半参数自适应控制的前提。现有的半参数自适应控制实现了一维情形下的参数估计算法，但二阶及其以上的参数估计并未具体实现，这涉及计算几何的知识。上一章介绍了半参数系统中的信息浓缩估计方法，从相关的定理2.1可知不同的先验知识会导致信息集合 I_k 的形式不同。一维情形由于只有一个参数不涉及顶点集的组合，因而具体算法比较容易实现，上面第2.3.3.1已经给出具体计算步骤。但是实际情况往往不只有一个参数，因而需要设计多维情形的估计算法，第2.3.3.2小节给出了实施框架。多维情形的算法如果直接通过代数推导，不容易解决，需要借助计算几何的思想。在线性控制理论和单变量系统中，两个参数的情况十分常见。因此本章针对两个参数即 $d_1 = 2$ 的情形，设计信息浓缩估计器的具体算法。

3.1　问题描述

在第二章的基础上，本章继续考虑上下界这种典型的先验知识。如前所述，需要针对关联这种先验知识的系统设计信息浓缩估计算法，完整的过程主要以下分为两步：

1. 信息浓缩。在上一个时刻的基础上，根据这个时刻的输入输出数据得到的约束，确定这个时刻的信息浓缩之后的参数集合。

2. 选定合适值。根据一定的法则，从上述的参数集合中选择合适的估计值作为这个时刻的参数估计值参与控制律等后续的计算。

第一步的具体操作是在某个确定时刻，得到一组输入输出数据，可以确定两个线性约束不等式，将这个两个约束先后加入到之前时刻的顶点集合中，得到这一时刻的顶点集合，也就确定了当前时刻的参数集合。从理论上说，第一步得到的参数集合中的每一个值都可以作为这个时刻的估计值，但是控制律等后续计算往往需要一个具体的值，所以需要按照一定的法则从集合中确定一个最优值，一般选择多边形的中心作为这个时刻的估计值。上面的两步中，第一步信息浓缩是难点。因此，本章的重点在于第一步的算法设计。一个时刻的信息浓缩过程包含两组约束关系的先后加入，也就

是两次 G 变换。这两次的变换过程一样，只是每次的数据不一样，故这里只需要考虑一次变换。另外，为了叙述方便，本章关于时刻 k 的下标省略。

$d_1 = 2$ 时，用 Γ 和 Γ' 分别表示两个不同的多边形，其顶点集合表示（按顺时针排列，下同）为

$$\Gamma = \{P_1, P_2, \ldots, P_N\}$$
$$\Gamma' = \{P'_1, P'_2, \ldots, P'_{N'}\}$$

$d_1 = 2$ 时，关于未知参数 $\boldsymbol{\theta}$ 的约束不等式为

$$\boldsymbol{\phi}^T \cdot \boldsymbol{\theta} \leq e \tag{3.1}$$

这里 $\boldsymbol{\phi}$ 和 $\boldsymbol{\theta}$ 都是二维向量，$\boldsymbol{\phi}$ 和 e 都可以由输入输出组合得到，$\boldsymbol{\theta}$ 是这个不等式的未知变量。

因此，变换 $G(\Gamma, \boldsymbol{\phi}, e)$ 可以记为映射

$$\text{AddLinear2D}\colon \Gamma \to \Gamma' \tag{3.2}$$

这就是二维参数的信息浓缩估计算法需要解决的核心部分。

3.2 几何关系分析

二维情形下主要是考虑同一个平面内，直线和多边形的位置关系。图2.2展示了直线和多边形位置关系的一种可能的例子，从这个图中可以知道求解这个顶点集涉及到计算几何的知识，需要全面考虑各种情形。具体来看，分析一条直线和多边形的位置关系，就是考察多边形中每个顶点和直线的位置关系。单个顶点和直线的位置关系主要涉及平面解析几何的知识。

$d_1 = 2$ 时，$\boldsymbol{\phi}^T \cdot \boldsymbol{\theta} = e$ 表示一条直线 L，它将整个平面分成两个半平面。用二元组

$$L_c = (\boldsymbol{\phi}, \boldsymbol{\theta})$$

表示不等式约束关系

$$\boldsymbol{\phi}^T \cdot \boldsymbol{\theta} \leq e \tag{3.3}$$

在几何上，L_c 也可以被认为是一个半平面。

记

$$\boldsymbol{\phi}^T = [\phi_1, \phi_2]$$

顶点 P_n 的坐标为

$$P_n : (X_{P_n}, Y_{P_n})$$

分别对应于待估计参数向量 $\boldsymbol{\theta}$ 的第一个分量和第二个分量。

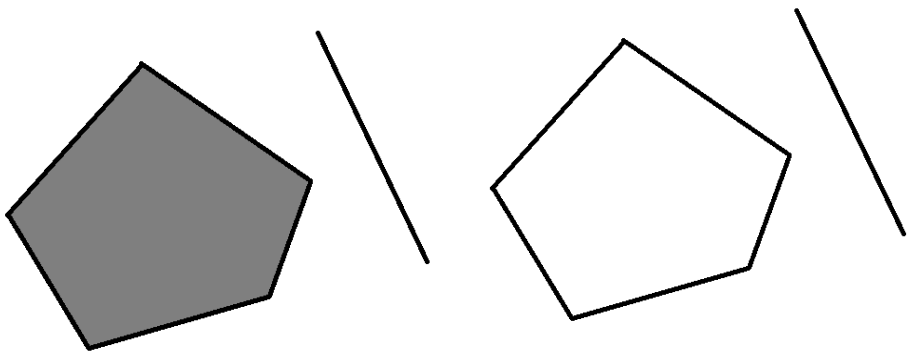
顶点 P_n 和直线 L 的位置关系可以由下面的不等式确定

$$\phi_1 X_{P_n} + \phi_2 Y_{P_n} \leq e \tag{3.4}$$

如果不等式(3.4)成立，则顶点 P_n 满足约束关系(3.3)，即在半平面 L_c 内；否则，顶点 P_n 不在半平面 L_c 内。根据不等式(3.4)可以确定多边形 Γ 的所有顶点和直线 L 的位置关系，从而判断出这个多边形和直线的位置关系。

经过分析，根据各个顶点和直线的分布情况，可以将多边形和直线的位置关系分为三类。

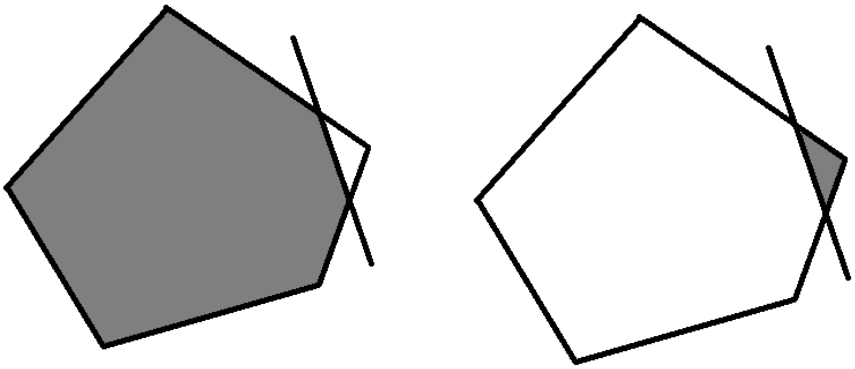
1. 多边形顶点在直线的同一侧，则多边形和直线没有交点，如图3.1所示又可具体分为两种情况。第一种情况3.1(a)，多边形不变，即 $\Gamma' = \Gamma$；第二种情况3.1(b)，一般是在数据出错的情况下才会发生（在定理(2.1)已经指出），此时 $\Gamma' = \varnothing$，算法停止。



(a) 情形 1.1　　(b) 情形 1.2

图 3.1 直线和多边形位置关系的第一类情形

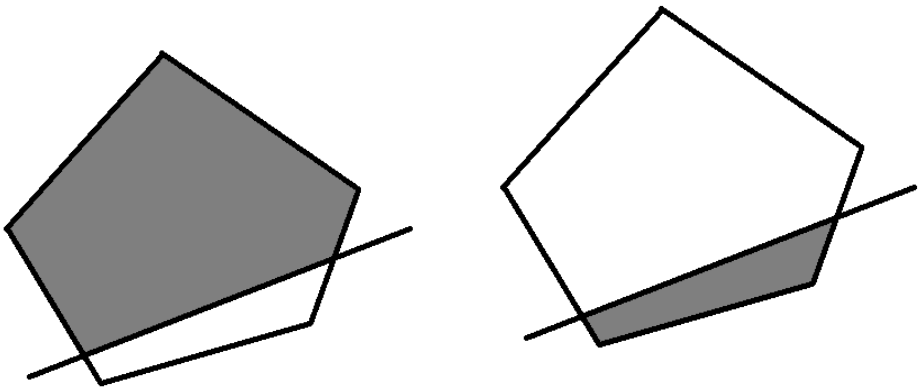
2. 多边形只有一个顶点和其他顶点不在直线的同一侧，如图3.2所示又可具体分为两种情况。第一种情况3.2(a)，只有一个顶点不满足约束，多边形顶点个数加 1；第二种情况3.2(b)，只有一个顶点满足约束，多边形变成一个三角形。



(a) 情形 2.1　　(b) 情形 2.2

图 3.2 直线和多边形位置关系的第二类情形

3. 多边形有至少两个顶点和其他顶点不在直线的同一侧，如图3.2所示又可具体分为两种情况。图3.3(a)和3.3(b)是两种对立的情形，事先都无法准确知道剩余的顶点个数，只能根据实际情况确定。



(a) 情形 3.1　　(b) 情形 3.2

图 3.3 直线和多边形位置关系的第三类情形

3.3 算法实现

根据上节关于多边形和直线的几何位置关系的分析和总结，本节设计了 $d_1 = 2$ 时多边形顶点集变换 AddLinear2D 的计算几何算法。

需要遍历多边形的所有顶点与多边形的位置关系。可以用一个数组记录顶点 P_n 和约束 L_c 的位置关系，即

$$\mathrm{Flag}(n) = \begin{cases} 1,\ P_n \in L_c \\ -1,\ P_n \in L_c \end{cases} \tag{3.5}$$

这里，$n = 1, 2, \ldots, N$。

根据上节的位置关系结果，可以设计算法的流程图如图3.4所示。

最后写出具体过程伪代码实现见算法3.1。

算法 3.1. AddLinear2D

输入: $\Gamma = \{P_1, P_2, \ldots, P_N\}$

输出: $\Gamma' = \{P'_1, P'_2, \ldots, P'_{N'}\}$

Denote the number of vertices in Γ *by* N

for $j = 1$ *to* N ***do***

Denote the j*th vertex by* P_j

Let $F_j = \mathrm{sgn}(c - \phi^T P_j)$

end for

if $F_j \geq 0\ \forall 1 \leq j \leq n$ ***then***

$\Gamma' \leftarrow \Gamma$ *and return*

else if $F_j < 0\ \forall 1 \leq j \leq n$ ***then***

$\Gamma \leftarrow \emptyset$ *and return*

end if

for $j = 2$ *to* $N - 1$ ***do***

if *(*P_j *and* P_{j-1} *are in different sides)* $\bigwedge$ *(*P_j *and* P_{j+1} *are in different sides)* ***then***

$P_1^i \leftarrow$ *the intersection of* l_i *and* $\overline{P_{j-1}P_j}$

$P_2^i \leftarrow$ *the intersection of* l_i *and* $\overline{P_{j+1}P_j}$.

if *(*$F_j > 0$*)* ***then***

$\Gamma' = \{P_1^i, P_j, P_2^i\}$

else

$\Gamma' = \{P_1, \ldots, P_{j-1}, P_1^i, P_2^i, P_{j+1}, \ldots, P_n\}$

end if

break

else if *(P_j, P_{j-1} are in different sides)* $\bigwedge$ *($P_j, P_{j+1}, \ldots, P_{j+a-1}$ are on the same side)* $\bigwedge$ *(P_{j+a-1}, P_{j+a} are on different sides)* ***then***

$P_1^i \leftarrow$ *the intersection of* l_i *and* $\overline{P_{j-1}P_j}$

$P_2^i \leftarrow$ *the intersection of* l_i *and* $\overline{P_{j+a-1}P_{j+a}}$.

if *($F_j > 0$)* ***then***

$\Gamma' = \{P_1^i, P_j, \ldots, P_{j+a-1}, P_2^i\}$

else

$\Gamma' = \{P_1, \ldots, P_{j-1}, P_1^i, P_2^i, P_{j+a}, \ldots, P_n\}$

end if

break

else if *(P_j, P_{j+1} are in different sides)* $\bigwedge$ *($P_j, P_{j-1}, \ldots, P_{j-b+1}$ are on the same side)* $\bigwedge$ *(P_{j-b+1}, P_{j-b} are on different sides)* ***then***

$P_1^i \leftarrow$ *the intersection of* l_i *and* $\overline{P_{j-b+1}P_{j-b}}$

$P_2^i \leftarrow$ *the intersection of* l_i *and* $\overline{P_{j+1}P_j}$.

if *($F_j > 0$)* ***then***

$\Gamma' = \{P_1^i, P_{j-b+1}, \ldots, P_j, P_2^i\}$

else

$\Gamma' = \{P_1, \ldots, P_{j-b}, P_1^i, P_2^i, P_{j+1}, \ldots, P_n\}$

end if

break

end if

$j \leftarrow j + 1$

end for

return Γ'

算法3.1实现了第3.1中提到的信息浓缩估计的第一步。对于第二步，一般选择多

边形 Γ' 的中心作为参数估计值 $\hat{\boldsymbol{\theta}}$，即

$$\begin{aligned}\hat{\theta}_1 &= \frac{1}{N}\sum_{n=1}^{N} X_{P'_n} \\ \hat{\theta}_2 &= \frac{1}{N}\sum_{n=1}^{N} Y_{P'_n}\end{aligned} \tag{3.6}$$

本节设计的算法3.1和方程(3.6)就是 $d_1 = 2$ 时完整的信息浓缩估计算法，可以用于后续的控制器设计。

3.4 仿真实例

为了验证本章设计算法的有效性，本节在实际的被控对象中仿真上面设计的信息浓缩估计算法。选择如下的被控对象

$$y_{k+1} = \theta_1 \cdot y_k + \theta_2 \cdot y_{k-1} + u_k + \omega_{k+1} + \sin(\frac{k}{2}) \tag{3.7}$$

其中 $\theta_1 = -0.5$，$\theta_2 = 0.5$。随机干扰取为均匀分布，即

$$\omega_k \in U(0,1)$$

则随机干扰的上下确界为

$$0 \leq \omega_k \leq 1$$

而非线性部分是三角函数，其上下界为

$$-1 \leq \sin(\frac{k}{2}) \leq 1$$

为了便于计算，令 $u_k = 0$。在本节中，为了更加符合实际情况，在计算时，将随机干扰和非线性的上下界范围扩大，分别取为

$$-1 \leq \omega_k \leq 2$$

和

$$-2 \leq \sin(\frac{k}{2}) \leq 2$$

因此，合并得到

$$\underline{z} = -3$$

$$\overline{z} = 4$$

初始的参数集合取为

$$\Theta = \{-1 \leq \theta_1 \leq 1,\ -1 \leq \theta_2 \leq 1\}$$

仿真到第 $k = 500$ 时，得到的多边形如图3.5所示，其中红色圈点代表真实值 $\boldsymbol{\theta}$ 在几何上的表示，星形点代表了估计值 $\hat{\boldsymbol{\theta}}$ 在几何上的表示。画出整个仿真过程的参数估计曲线如图3.6所示。从这个图中可以看出，参数估计值接近真实值。

3.5 优化问题

在信息浓缩估计方法中，关键的问题在于计算每一步（时刻 k）的信息集 I_k 和信息浓缩集 C_k。从上面的讨论中，可以看出 $d_1 = 1$ 时很容易解决这两个集合的计算问题。然而，当 $d_1 > 1$ 时，通常情况下集合 C_k 中的顶点个数 N_k 随着 $k \to \infty$ 会不断增长，特别是高维情形 $d_1 \geq 3$ 时。因此，有可能在实际应用中，随着 $k \to \infty$，信息浓缩估计过程需要的内存会无限增长，这是一个十分致命的问题，需要优化。

其实，随着 $k \to \infty$，C_k 的范围是在不断缩小的，即使顶点个数可能会增加。因此，为了克服顶点个数无限增长的问题，在实际应用中，可以不需要如此多的顶点来精确的表示信息浓缩集 C_k。也就是说，在多维情形中，在每个时刻 k 的 G 变换过程中，可以用具有足够少顶点个数 N_L 的集合 $\hat{C}_k$ 来近似 C_k。

一般有两种优化策略，这里主要介绍被称为信息浓缩估计算法的“松实现”（Loose IC estimator, Loose-IC），另外一种是“紧实现”（Tight IC estimator, Tight-IC）。Loose-IC 的具体要求是，对于任意的 k，都满足 $\hat{C}_k \supseteq C_k$.

记

$$\hat{C}_\infty = \bigcap_{k=1}^{\infty} \hat{C}_k,$$

则

$$C_\infty = \bigcap_{k=1}^{\infty} C_k \subseteq \hat{C}_\infty \tag{3.8}$$

上面方程(3.8)意味着，Loose-IC 在实施过程中不会丢弃任何可能的参数值。对于 $d_1 = 2$ 这种二维情形，为了便于计算，一般选择顶点个数 $N_L = 3$ 或者 $N_L = 4$，分别如图3.7和3.8所示。从图中可以看出，顶点个数的减少和多边形的简化，意味者计算复杂度的降低，实现较好的优化效果。

3.6 主要特点

从上面的讨论和仿真实例中，可以总结出信息浓缩估计算法有如下优点：

1. 充分利用先验信息和后验数据产生的信息。理想情况下没有任何的信息被浪费掉，而一般传统的辨识算法仅仅利用了部分先验信息和某些随机先验假设。

2. 这是基于集合的估计，实际上每次给出的是一个参数可行的集合，不同于最小二乘算法等给出的点估计，即单个值。这样一方面可以获得比较全面的参数信息，另一方面可以衡量估计值的准确性。

3. 随着时间的增长，逐步找出最优（最有可能的）参数估计值。在参数估计过程中，同时可以用于检查实际采集的数据和系统模型的先验信息的一致性。

4. 可以针对随机系统，也可以处理非统计语言描述的先验信息，不受限于概率模型。

5. 具体实施比较灵活，主要取决于先验信息的具体表现形式，这也给算法的优化带来了极大的提升空间。例如为了降低计算复杂度，可以用上节讨论的“松实现”。

6. 不像传统辨识算法一样，在某些时候表现出随着时间的推移，估计值漂移或者精度下降的现象。相反，数据越多，估计越准确。

一般事物总存在两面性，经过分析，信息浓缩估计算法也存在如下问题：

1. 难以利用系统干扰项的先验信息，特别是无界随机干扰。也许在这种没有非参数不确定性且随机干扰是无界噪声序列时，传统基于值估计的算法表现得更加适合。

2. 实际的算法效率依赖于先验信息的利用程度和具体实现。一般情况下，由于信息浓缩估计算法涉及到计算几何过程（也许有其他的实施策略），因而其计算复杂度和设计难度要稍大于 RLS 等传统辨识算法。毕竟后者一般只会涉及到矩阵的加减乘除运算。

3. 目前不存在针对信息浓缩估计算法的显示表达式或者固定的公式，同时，基于集合的辨识思路给具体控制系统的应用和闭环稳定性分析带来了一定的困难。

虽然存在诸如此类的缺点，但信息浓缩估计算法仍然提供了很好的辨识思路和较好的收敛性和精确度，值得深入研究和进一步优化。

3.7 本章总结

本章主要讨论了信息浓缩估计算法的具体实现。首先描述了二维参数情形下，信息浓缩估计算法的主要步骤和基本问题；然后对信息浓缩变换中直线和多边形的几何关系进行了详细地分析，同时设计了信息浓缩变换的算法流程图，并给出了具体的伪代码实现；接着在实际的被控对象中仿真前面设计的信息浓缩估计算法，给出了最终估计结果和过程曲线；最后讨论了信息浓缩估计的计算复杂度优化问题，总结了信息浓缩估计的主要优缺点。

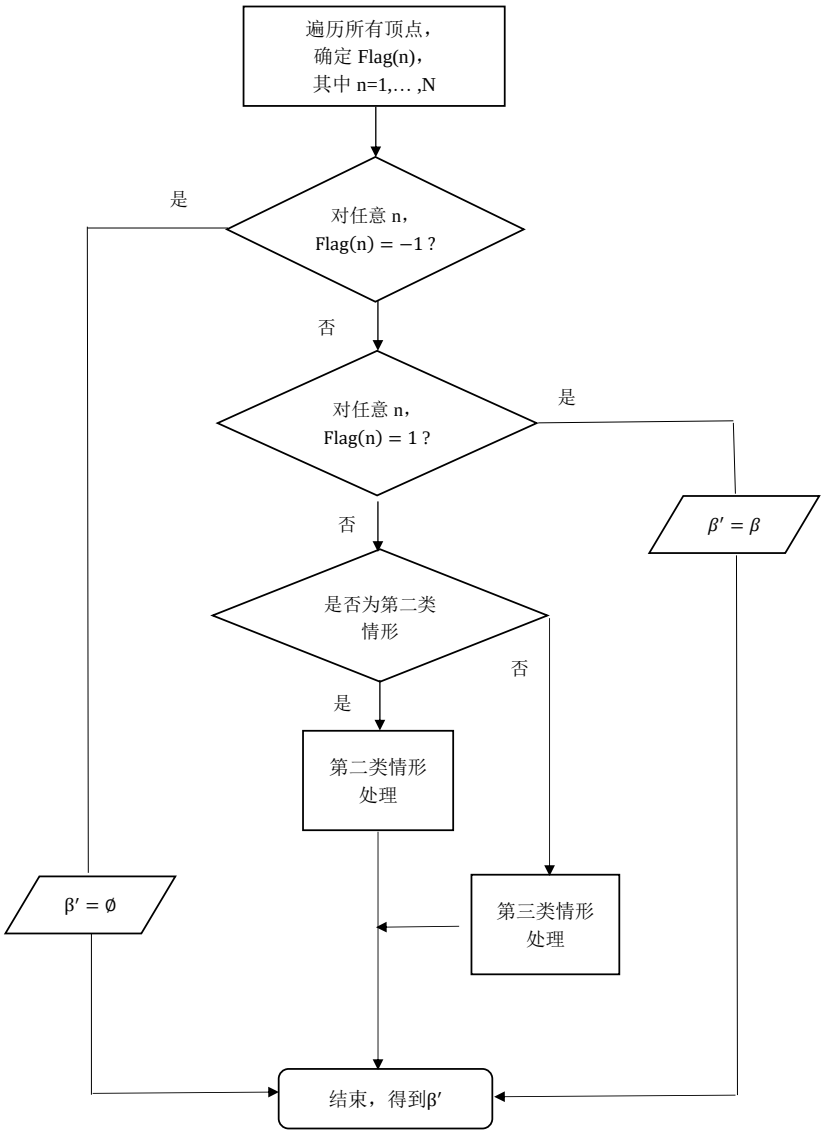


图 3.4 AddLinear2D 的算法流程图

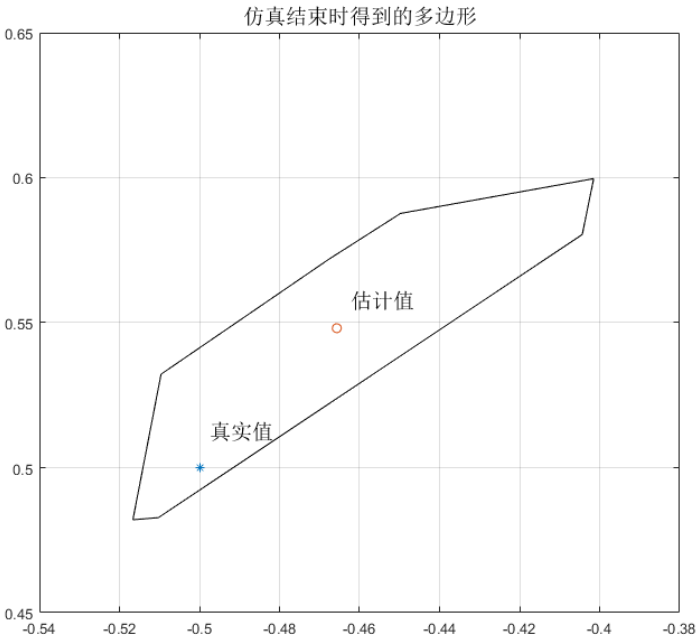


图 3.5 当 $d_1 = 2$ 时仿真实例中 $k = 500$ 时的多边形及参数估计

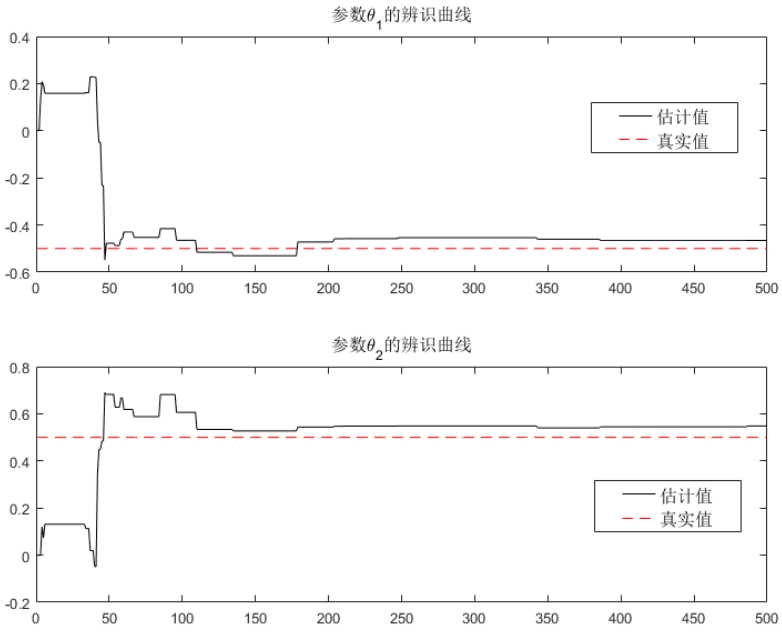


图 3.6 当 $d_1 = 2$ 时仿真实例中的参数估计曲线（信息浓缩估计算法）

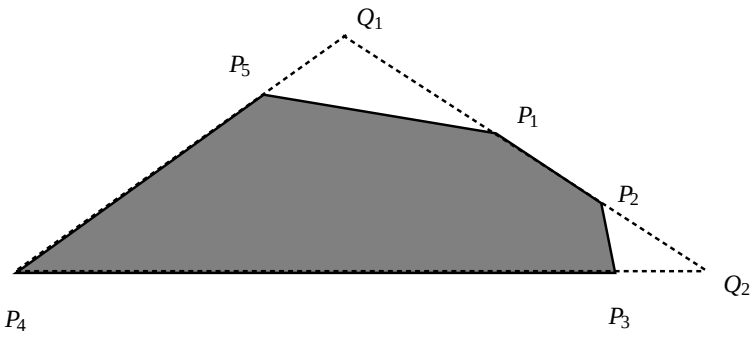


图 3.7 $d_1 = 2$ 且 $N_L = 3$ 时的 Loose-IC 实现，用三角形近似多边形

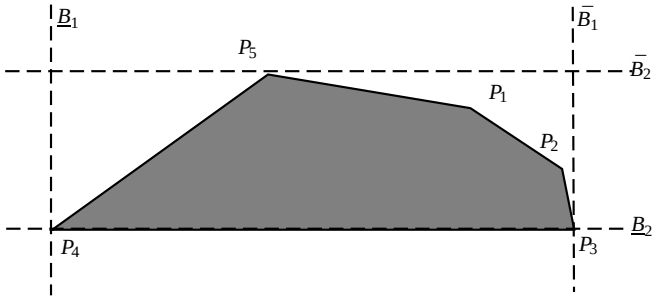


图 3.8 $d_1 = 2$ 且 $N_L = 4$ 时的 Loose-IC 实现，用矩形近似多边形

第 4 章　离散时间半参数自适应控制

自适应控制中的的辨识是为了控制问题，上一章设计了二维情形下的信息浓缩估计算法去辨识线性部分的未知参数。本章将在第三章的基础上设计非参数部分的估计算法，主要借助于一种机器学习算法，然后根据两部分的估计结果去设计自适应控制律在去解决随动控制问题。

4.1　问题描述

再次考虑第二章给出的半参数模型

$$y_{k+1} = \boldsymbol{\theta}^T \boldsymbol{\phi_k} + u_k + f(\boldsymbol{psi_k}) + \omega_{k+1} \tag{4.1}$$

其中，$\boldsymbol{\theta}$ 是未知参数向量，$f(\cdot)$ 是未知函数。同理，令

$$z_k = f(\boldsymbol{\psi}_k) + \omega_{k+1} \tag{4.2}$$

针对系统(4.1)，自适应估计与控制问题的一般表述是：给定关于 $\boldsymbol{\theta}$，$f(\cdot)$ 和 ω_{k+1} 的先验信息，如何根据实时产生的一系列输入输出数据 $\{y_k, u_k; k = 1, 2, \ldots\}$，去估计未知参数 $\boldsymbol{\theta}$ 和 z_k？然后根据 $\boldsymbol{\theta}$ 和 z_k 的估计（预测）值去设计合适的自适应控制输入 u_k 使得 $k+1$ 时刻的输出 y_{k+1} 能跟踪上期望的输出 y^*_{k+1}。

第三章已经给出了 $\boldsymbol{\theta}$ 的估计算法，因此剩下的部分可以分为两个核心问题，一方面是设计 $f(\cdot)$ 的估计算法，另一方面是设计控制输入 u_k 的表达式。假设第二章设计出的未知参数的估计值为 $\hat{\boldsymbol{\theta}}_k$，下面需要设计出 $\breve{z}_k$ 的表达式。因此，根据必然等价原理，设计出的控制输入的一般表达式为：

$$u^*_k = y^*_{k+1} - \boldsymbol{\phi}^T_{\boldsymbol{k}} \cdot \hat{\boldsymbol{\theta}}_k - \breve{z}_k \tag{4.3}$$

非线性部分的辨识与控制器的设计是本章重点解决的问题。

4.2 非参数部分的估计

4.2.1 最近邻估计

前面的章节用信息浓缩估计解决了参数不确定性的辨识问题，半参数自适应控制在提出来的时候，针对非参数部分使用的是最近邻估计。最近邻估计主要针对的是一维参数情形（$d_1 = 1$ 且 $d_2 = 1$），即下面的一维半参数系统

$$y_{k+1} = \theta y_k + u_k + f(y_k) + \omega_{k+1} \tag{4.4}$$

这里，标量 θ 是未知的，为参数不确定性部分，同样 $f(\cdot)$ 是非参数不确定性部分，满足 Lipschitz 条件。最近邻估计不显示地估计 $f(\cdot)$ 部分，而是直接估计

$$\eta_k = \theta y_k + f(y_k) \tag{4.5}$$

这个整体。这样，在用信息浓缩算法估计出参数 θ 之后，移除这个估计值 $\hat{\theta}$，就可以获得非参数部分的估计。

在系统(4.4)中，由于 Lipschitz 条件的存在，不管是非参数部分还是参数部分都主要和输出值 y_k 相关，这样输出值 y_k 相近的时刻，那么 η_k 的值也应该相差不大。因此最近邻估计就主要利用到了这一点，其主要思想在于寻找和当前时刻输出值 y_k 最相近的时刻，然后进行比较计算。

首先定义最近邻时刻为

$$i_k = \arg\min_{i<k} |y_k - y_i| \tag{4.6}$$

i_k 是历史输出值中和当前时刻最接近的时刻。那么有如下等式成立

$$\begin{aligned} \eta_k &= \eta_k - z_{i_k} + z_{i_k} \\ &= [\theta y_k + f(y_k)] - [\theta y_{i_k} + f(y_{i_k}) + w_{i_k+1}] + z_{i_k} \\ &= [\theta(y_k - y_{i_k}) + z_{i_k}] + [f(y_k) - f(y_{i_k}) - w_{i_k+1}] \end{aligned}$$

一般来说，上面等式中的最后一行中 $f(y_k) - f(y_{i_k})$ 的值接近零，并且干扰项比较小。

因此，可以取整体表达式(4.5)的估计为

$$\hat{\eta}_k \triangleq \hat{\theta}_k(y_t - y_{i_k}) + z_{i_k} = \hat{\theta}_k(y_k - y_{i_k}) + (y_{i_k+1} - u_{i_k}) \tag{4.7}$$

接着，令

$$\begin{aligned} \bar{b}_k &\triangleq \max_{i \le k} y_i = \max(\bar{b}_{k-1}, y_k) \\ \underline{b}_k &\triangleq \min_{i \le k} y_i = \min(\underline{b}_{k-1}, y_k). \end{aligned} \tag{4.8}$$

最后基于设计出下面的控制律

$$u_t = \begin{cases} -\hat{\eta}_k + y^*_{k+1} & \text{if } |y_k - y_{i_k}| \le D \\ -\hat{\eta}_k + \frac{1}{2}(\bar{b}_k + \underline{b}_k) & \text{if } |y_k - y_{i_k}| > D \end{cases} \tag{4.9}$$

这里，D 是一个合适的常数值。

由此看出，最近邻估计可以比较准确地给出一维情形下非参数部分的估计，从而解决半参数系统的自适应控制问题。但是有一定的局限性，主要体现在：

1. 由于最近邻时刻 i_k 的计算需要遍历所有的历史值，因此，随着时间的增长，理论上需要无限的内存用来存储历史数据。这给实施带来了困难。

2. 由于在计算 $\hat{g}_k$ 时没有对干扰值作特别处理，因此适用于干扰值的有界范围比较小且变化比较慢的系统。

3. 没有充分利用历史数据和关联先验信息，只是利用最近邻时刻的数据，计算结果不一定是最优的。

以上的缺点存在，导致现有的半参数自适应控制存在很大的局限性。本章借助于其他的非线性辨识算法去设计新的非参数部分的估计算法。

4.2.2 神经网络

机器学习解决的问题从数学上主要分为分类和回归。其中分类处理的数据主要是离散型数据，回归解决的对象和思路跟控制中的估计、预测十分类似。而神经网络作为一种十分有效的回归与预测算法，在非线性辨识中应用十分广泛。神经网络主要由

输入层、隐含层和输出层组成（最简单的神经网络可能没有隐含层），依据不同的激活函数或者不同的学习算法，种类十分丰富。

本章以及本文主要讨论一种十分常见的单隐层前馈神经网络（Single hidden layer feedforwar network, SLFN），即具有一个输入层、一个隐含层以及一个输出层的前馈型神经网络。这是一种多层网络，其中输入层神经元接受外界输入，隐含层与输出层的神经元对信号进行加工，最终结果由输出层神经元输出。

下面首先阐述神经网络的基本数学模型。对于一个具有 N_i 个输入神经元、N_o 个输出神经元以及 N_h 个隐含层神经元的单隐层前馈神经网络来说，如果其激活函数为 $g(\cdot)$，并存在 N_s 个输入输出样本对

$$\{\boldsymbol{x}_j, \boldsymbol{t}_j\},\ j = 1, 2, \ldots, N_s$$

这里，输入向量为

$$\boldsymbol{x}_j = [x_{j,1}, x_{j,2}, \ldots, x_{j,N_i}]^T \in \mathcal{R}^{N_i}$$

期望的输出向量为

$$\boldsymbol{t}_j = [t_{j,1}, t_{j,2}, \ldots, t_{j,N_o}]^T \in \mathcal{R}^{N_o}$$

则对于任意输入向量 $\boldsymbol{x}_j$，第 l 个（$l = 1, 2, \ldots, N_o$）输出层神经元的输出为

$$\begin{aligned} o_{j,l} &= \sum_{i=1}^{N_h} \beta_{i,l} g(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j) \\ &= \sum_{i=1}^{N_h} \beta_{i,l} h(\boldsymbol{x}_i) \\ &= \boldsymbol{h}(\boldsymbol{x}_j)^T \cdot \boldsymbol{\beta}_l \end{aligned} \tag{4.10}$$

其中，$\boldsymbol{w}_i$ 和 b_i 分别是连接第 i 个隐层神经元与输入层的权重向量和阈值，$\boldsymbol{\beta}_l$ 是连接 l 个输出层神经元和隐含层的权重，$\boldsymbol{h}(\boldsymbol{x}_j)$ 为隐含层的输出向量。

一般来说，同一层的激活函数是一致的。如果第 i 个隐含神经元的激活函数 $g(\cdot)$ 是加性的（addictive），比如 Sigmoid 型或者正弦型，则其输出为

$$g(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j) = g(\boldsymbol{w}_i^T \cdot \boldsymbol{x}_j + b_i) \tag{4.11}$$

如果是径向基（Radias base function, RBF）网络，则

$$g(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j) = g(b_i \|\boldsymbol{x}_j - \boldsymbol{w}_i\|) \tag{4.12}$$

此时，g 是某种径向对称的标量函数，通常定义为关于样本到数据中心 $\boldsymbol{w}_i$ 之间的欧氏距离的单调函数，比如高斯径向基函数等。这样，在命名上，$\boldsymbol{w}_i$ 和 b_i 一般不称为隐层的权重和阈值，而是激活函数的中心（center）和影响因子（impact factor）。

理论上，对于给定的样本数据集，上述定义的具有 N_h 个隐含层神经元和激活函数为 $g(\cdot)$ 的 SLFN，具有零误差逼近 N_s 个样本的性质，即

$$\sum_{j=1}^{N_h} \|o_j - t_j\| = 0,\ j = 1, 2, \ldots, N_s$$

这意味着，存在一组 $\boldsymbol{w}_i$、b_i 和 $\boldsymbol{\beta}$，使得

$$\boldsymbol{h}(\boldsymbol{x}_j)\boldsymbol{\beta} = t_j,\ j = 1, 2, \ldots, N_s \tag{4.13}$$

将上面 N_s 个方程堆积起来，写成紧凑的矩阵形式就是

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T} \tag{4.14}$$

其中，$\boldsymbol{H}$ 是隐含层的输出矩阵，具体为

$$\begin{aligned}
&\boldsymbol{H}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{N_h}; b_1, \ldots, b_{N_h}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_s}) = \\
&\begin{bmatrix}
h(\boldsymbol{w}_1, b_1, \boldsymbol{x}_1) & \cdots & h(\boldsymbol{w}_{N_s}, b_{N_s}, \boldsymbol{x}_{N_s}) \\
\vdots & \cdots & \vdots \\
h(\boldsymbol{w}_1, b_1, \boldsymbol{x}_{N_s}) & \cdots & h(\boldsymbol{w}_{N_s}, b_{N_s}, \boldsymbol{x}_{N_s})
\end{bmatrix}_{N_s \times N_h}
\end{aligned} \tag{4.15}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{N_h}^T \end{bmatrix}_{N_h \times N_o} \tag{4.16}$$

且

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_{N_h}^T \end{bmatrix}_{N_s \times N_o} \tag{4.17}$$

如果隐含层神经元个数等于样本的个数，即 $N_s = N_h$，那么矩阵 $\boldsymbol{H}$ 是方阵且可逆的，则 SLFN 可以实现零误差逼近的结果。然而，在实际的大多数情况下，隐含层神经元个数远小于样本个数，即 $N_h \ll N_s$。这样，$\boldsymbol{H}$ 不是一个方阵，就不存在准确的一组 $\boldsymbol{w}_i$、b_i 和 $\boldsymbol{\beta}$，使得 $\boldsymbol{H\beta} = \boldsymbol{T}$ 精确成立。这种情况下，神经网络学习的目标就是寻找合适的一组 $\hat{\boldsymbol{w}}_i$、$\hat{b}_i$ 和 $\hat{\beta}$，使得

$$\begin{aligned} &\|\boldsymbol{H}(\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_{N_h}; \hat{b}_1, \ldots, \hat{b}_{N_h})\hat{\boldsymbol{\beta}} - \boldsymbol{T}\| = \\ &\min_{\hat{\boldsymbol{w}}_i, \hat{b}_i, \hat{\boldsymbol{\beta}}} \|\boldsymbol{H}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{N_h}; b_1, \ldots, b_{N_h}) - \boldsymbol{T}\| \end{aligned} \tag{4.18}$$

成立，等价于最小化基于累计误差的损失函数

$$E = \sum_{j=1}^{N_s} (\boldsymbol{h}(x_j)\boldsymbol{\beta} - \boldsymbol{t}_j)^2 \tag{4.19}$$

多层网络的学习能力依赖于强大的学习算法，误差反向传播（error BackPropagation, BP）是一种十分经典的代表。迄今为止，现实中的神经网络都是使用 BP 算法进行训练，不仅仅局限于多层前馈神经网络，还适用于递归神经网络等其它类型。BP 算法是一种基于梯度下降（gradient-descent-based）的学习算法。将需要调整的权重 $\boldsymbol{w}_j$ 和 $\boldsymbol{\beta}$ 以及阈值 b_j 等网络参数，整体记作向量 $\boldsymbol{W}$，则 BP 算法按照下面的式子调整 $\boldsymbol{W}$:

$$\boldsymbol{W}_k = \boldsymbol{W}_{k-1} - \eta \frac{\partial E(\boldsymbol{W})}{\boldsymbol{W}}, \tag{4.20}$$

其中，η 是学习速率。

算法 4.1 (htp). BP 学习算法的主要流程

输入: 训练样本集 $\{\boldsymbol{x}_j, \boldsymbol{t}_j\}$, $j = 1, 2, \ldots, N_s$;

学习速率 η;

迭代次数上限 N_{max}

初始化所有网络参数（随机选择，或者根据经验指定）

for $m = 1$ *to* N_{max} ***do***

根据输入样本和当前参数计算所有样本的输出结果；

计算输出层的累计误差；

将误差逆向传递至隐层神经元，根据隐层神经元的误差来对连接权重和阈值进行调整；

根据(4.20)更新 $\boldsymbol{W}$；

判断累积误差是否达到停止条件，如果达到，则退出当前循环；否则，继续当前迭代过程；

end for

输出: 网络参数向量 $\boldsymbol{W}$ 确定的前馈型神经网络

在前馈型神经网络中，BP 算法的主要工作流程见算法(4.1)。从学习过程可以看出，BP 算法存在如下问题：

1. 学习速率 η 的大小会影响算法的迭代过程，主要表现在，如果 η 太大，则算法会变得不稳定且可能会发散；如果 η 太小，则收敛速度非常慢。
2. 本质上是基于梯度的搜索寻优方法，会陷入局部最优的困境，即网络参数的更新会收敛到某个误差局部最小的情况，这是梯度下降所导致的。
3. 神经网络可能存在过度训练、过度拟合的情形，以致于网络的泛化能力很差。
4. 从初始解出发，迭代寻找最优参数值，其过程通常非常耗时，难以做到实时在线计算。

以上这些因素对于神经网络在控制系统中的应用带来了很多问题，前面三条包括学习速率、局部最优、过度训练的问题有一些方法去改进，但是第四条迭代过程耗时这个问题是由梯度算法本身的特性所决定，难以从根本上解决。这就导致了超限学习机的提出。

4.2.3 ELM 及其变体

传统上对于神经网络的理解，都认为需要网络的学习过程需要调整所有的参数。但实际上，如果输入层到隐含层的权重和阈值不变，那么隐含层的输出矩阵 $\boldsymbol{H}$ 就保

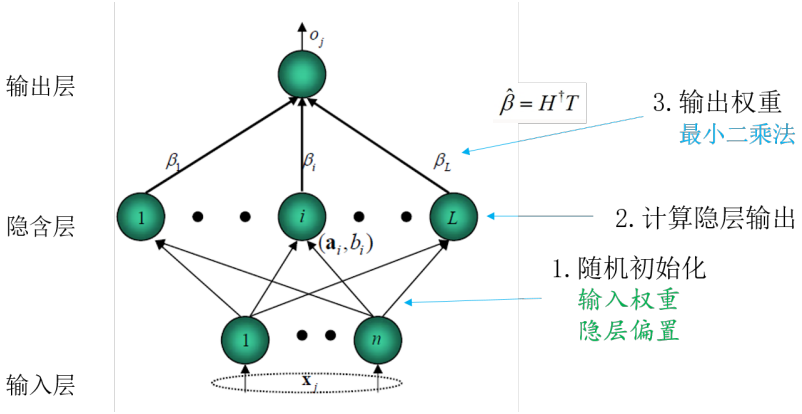


图 4.1 超限学习机的网络结构（单隐层前馈型）和算法流程

持不变，仅调整隐含层到输出层的权值就可以改变整个网络的最终输出。超限学习机就是利用这个思路建立起来的，用来解决传统基于梯度学习算法的耗时训练的缺点。

超限学习机的基本算法主要包括三个步骤，针对单隐层前馈神经网络的学习过程见算法4.2。

算法 4.2. ELM 算法

输入: 训练样本集 $\{\boldsymbol{x}_j, \boldsymbol{t}_j\}$, $j = 1, 2, \ldots, N_s$

Step1: 随机初始化输入权重 $\boldsymbol{w}_i$ 和阈值 b_i 参数

Step2: 计算隐含层输出矩阵 $\boldsymbol{H}$，见表达式(**??**)

Step3: 计算输出层权重向量 $\boldsymbol{\beta}$，根据如下表达式

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{4.21}$$

其中 $\boldsymbol{H}$、$\boldsymbol{T}$ 和 $\boldsymbol{\beta}$ 的定义分别见方程(4.15)、(4.17)和(4.16)，而 $\boldsymbol{H}^{\dagger}$ 代表了矩阵 $\boldsymbol{H}$ 的 *Moore-Penrose* 广义逆。

输出: 网络参数向量 $\boldsymbol{W}$ 确定的前馈型神经网络

图(4.1)表示的是 ELM 在单隐层神经网络学习过程中的主要计算步骤。实际上，(4.21)计算的是方程(4.14)的最小二乘解，即

$$\boldsymbol{\beta} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{T} \tag{4.22}$$

由于没有迭代过程，因此 ELM 在解决分类或者回归等问题时学习时间很短，主要耗时在矩阵的加减乘除和求逆运算的时间。从学习过程可以看出，ELM 的求解思

表 4.1 超限学习机与常见深度学习算法的性能比较

算法名称	测试精度 (%)	训练时间
H-ELM	99.14	281.37s
Multi-layer ELM	99.03 ± 0.04	281.37s
Deep Belief Networks(DBN)	98.87	20580s(5.7 hours)
Deep Boltzmann Machine(DBM)	99.05	68246s(19 hours)
Stacked Auto Encoders(SAE)	98.6	>17 hours
Stacked Denoising AutoEncodes(SDAE)	98.72	>17 hours

路和最优化方法以及最小二乘辨识算法有些相似，因此基本算法存在的问题和改进思路也类似。上述算法4.2介绍的是原始 ELM 的批处理形式，由于实际中样本可能不是一次性获得的，因此和最小二乘算法一样，得到 ELM 的递推形式，即在线序列 ELM 算法（OS-ELM）[29]。

OS-ELM 的推导过程不再详细介绍，和 RLS 类似。OS-ELM 对于控制系统的辨识十分有用，因为控制系统的数据大多是实时产生的，需要在线辨识。然而 OS-ELM 和批处理形式最后的效果是一致的，只是中间过程不一样，并没有对网络结构和目标函数进行优化。基本 ELM 优化的仅仅是经验损失风险，目标函数见(4.19)，因此 ELM 和 OS-ELM 都可能存在泛化能力不足和过拟合的现象。为了解决这个问题，相关学者[41, 42] 提出了正则化 ELM（简称 Re-ELM）和正则化 OS-ELM（简称 ReOS-ELM）。

人工智能的发展一直推动着神经网络的研究，特别是 2012 年之后，深度学习（Deep Learning, DL）在图像分类、语音识别、自然语言处理等应用领域中获得了巨大成功[43]。目前大部分的深度学习算法大部分是基于神经网络建立。对于深度学习来说，隐含层的数量级为十几层到几百层不等，甚至上千层。深层网络结构和特征学习思想是深度学习的两大主要特点。针对大数据的应用，超限学习机也有相关变体，如 H-ELM（High-Performance Extreme Learning Machine）[44] 与深度学习相比，超限学习机在不丢失精度的条件下具有快速的优势，比较见表4.1所示。

4.2.4 ELM 估计

基于机器学习的辨识与建模是数据驱动的，可以适用于线性和非线性系统，主要建立系统的输入输出模型。快速、高精度的特点使得 ELM 迅速称为系统与控制领域很好的建模、估计与预测算法。特别是经过优化，提高泛化等方面的性能的 ELM 变

体算法，如 Re-OSELM。不同于(4.19)和(4.18)，ReOS-ELM 最小化目标的损失函数是

$$\|\boldsymbol{H}\boldsymbol{\beta}-T\|^2+\lambda\|\boldsymbol{\beta}\|^2 \tag{4.23}$$

正则化因子 λ 的引入提高了网络的泛化能力。图(4.2)展示了用 Re-OSELM 建立非线性模型的框图。

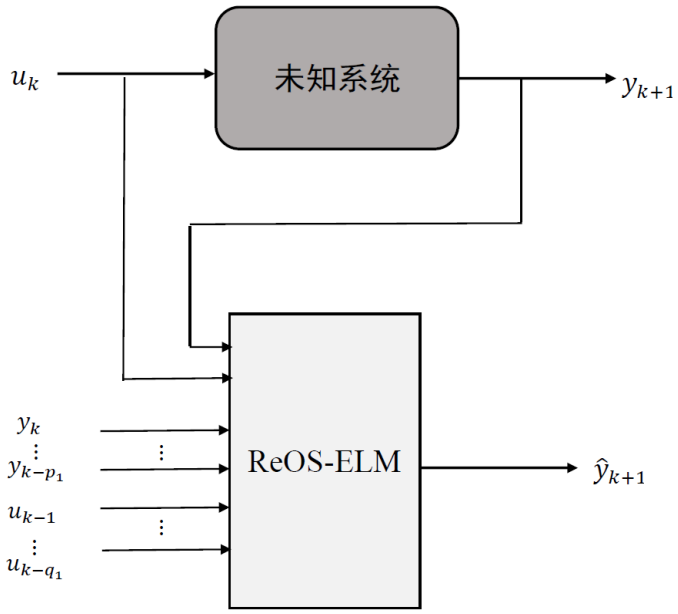


图 4.2 Re-OSELM 算法的建模框图

类似于 RLS，OS-ELM 和 Re-OSELM 等算法在系统建模与估计过程一般都是分为两个阶段。第一个阶段是初始化网络参数和输出权重 $\boldsymbol{\beta}$ 估计值。对于 Re-OSELM 来说，需要一部分的离线输入输出数据用来完成初始化阶段，通过网络得出 $\boldsymbol{H}_0$ 和 $\boldsymbol{T}_0$ 之后，需要计算

$$\begin{aligned}\boldsymbol{P}_0&=(\boldsymbol{H}_0^T\boldsymbol{H}_0+\lambda\boldsymbol{I}_{N_h})^{-1}\\ \boldsymbol{\beta}^0&=\boldsymbol{P}_0\boldsymbol{H}_0\boldsymbol{T}_0\end{aligned} \tag{4.24}$$

这里，$\boldsymbol{I}_{N_h}$ 是 $N_h\times N_h$ 的单位矩阵

在 Re-OSELM 算法更新权重的过称中引入恰当的或者自适应的遗忘因子 可以使得算法能够应用到时变参数系统中。由于正则化因子 λ 的存在，即使不需要大于隐含层个数的样本数据用来初始化 $\boldsymbol{\beta}^0$，初始小规模的训练集也是比不可少的。然而，在控制系统中，由于输出值跟实际输入密切相关，因此一定数量的离线输入输出数据一般

情况下是难以获得的，这就限制了 Re-OSELM 在实际控制系统中的应用。

ITF-OELM 算法就是用来解决缺少离线初始化数问题而提出的。ITF-OELM 算法主要针对下面的单输入单输出离散时间非线性系统

$$y_{k+1} = g(\boldsymbol{x}_k) + \phi(\boldsymbol{x}_k)u_k \tag{4.25}$$

这里 $\boldsymbol{x}_k$ 是由 $\{u_k, y_k\}$ 组成的回归向量，$g(\cdot)$ 和 $\phi(\cdot)$ 是具有未知参数的未知非线性函数。用 ITF-OELM 算法分别估计 $g(\cdot)$ 和 $\phi(\cdot)$，得到 $\hat{g}(\cdot)$ 和 $\hat{\phi}(\cdot)$，那么基于 ITF-OELM 估计的自适应控制律为

$$u_k = \frac{y^*_{k+1} - \hat{g}(\boldsymbol{x}_k)}{\hat{\phi}(\boldsymbol{x}_k)} \tag{4.26}$$

这里，y^*_{k+1} 是期望的输出值。

基于 ITF-OELM 的自适应控制框图如图4.3所示。ITF-OELM 算法与 Re-OSELM 的不同点主要在于初始化输出权重的过程。前者不需要一定数量的离线数据，仅仅用第一组到达的数据初始化估计。另外，ITF-OELM 还引入了再学习机制，即如果发现估计偏差较大，就重新初始化网络的所有参数，重新学习。这样，ITF-OELM 可以应用于快时变系统，且不需要过多的离线数据。

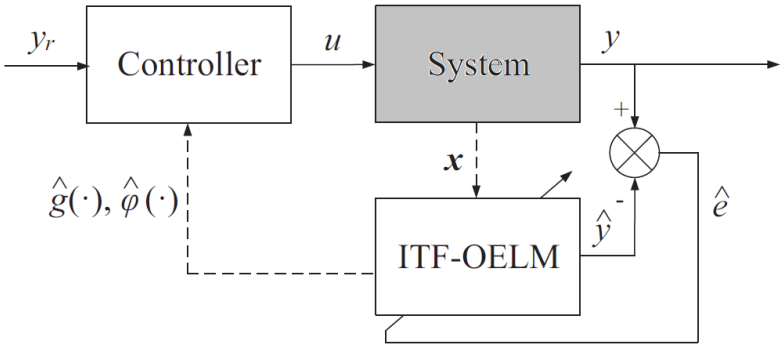


图 4.3 基于 ITF-OELM 的自适应控制框图

下面借助 ITF-OELM 来解决本课题的非参数部分的估计问题。首先要确定的是网络的输入输出样本数据。本文要估计的非线性部分为

$$z_k = f(\boldsymbol{\psi}_k) + \omega_{k+1} \tag{4.27}$$

因此，神经网络的输入是 $d_2 = p_2 + q_2$ 维向量

$$\boldsymbol{\psi_k} = [y_k, y_{k-1}, \ldots, y_{k-p_2}, u_{k-1}, \ldots, u_{k-q_2+1}]^T$$

d_2 的具体数值视实际被控对象而定。输出是标量 z_k。因此网络的输出层只有一个神经元，输入层含 d_2 个神经元。

本课题针对半参数系统在原始的 ITF-OELM 算法基础上做了几点变化。首先是估计的非线性函数只有一个，其次是在学习机制的不同。训练网络采用的输入数据 $\boldsymbol{\psi}_k$ 直接来源于输入和观测到的历史输出数据，然而输出样本数据为

$$\hat{z}_{k-1} = y_k - \phi_k^T \hat{\theta}_{k-1} - u_{k-1} \tag{4.28}$$

这个非线性估计部分是除去参数估计之后的剩余部分。用单隐层网络加上 ELM 学习算法得到的估计值是

$$\breve{z}_k = H_k \beta^k \tag{4.29}$$

和其他监督学习算法一样，网络的训练结果依赖于样本数据的准确性。$\hat{z}_{k-1}$ 的值依赖于参数估计值 $\hat{\theta}$，因此当参数估计值趋于稳定或者参数集合收敛到一定范围之后，需要重新初始化网络参数，继续辨识。也就是说，本课题在实施 ITF-OELM 算法时，不直接采用原本的再学习机制，而是引入另外一种情况下的再学习机制。这样就可以做到非参数估计更加准确。

本课题基于 ELM 算法估计非参数部分的实施过程依赖和控制密切相关，故具体算法公式将和自适应控制器设计一起在下一节介绍。

4.3 自适应控制器设计

用信息浓缩估计参数部分，用 ELM 估计非参数部分，如图4.4。在前面两种算法的基础上，可以设计出半参数自适应控制器，主要包含两个过程，即初始化阶段和自适应控制阶段。在初始化阶段，需要完成信息浓缩估计器关联先验信息设置相关参数集的任务，以及 ELM 学习算法的网络参数初始化任务；自适应控制阶段前半段主要是参数估计，后半段主要是神经网络的学习更新非参数部分的估计，从而产生合适的控制输入，完成轨迹跟踪任务。下面介绍具体实施流程，其中信息浓缩估计部分直接

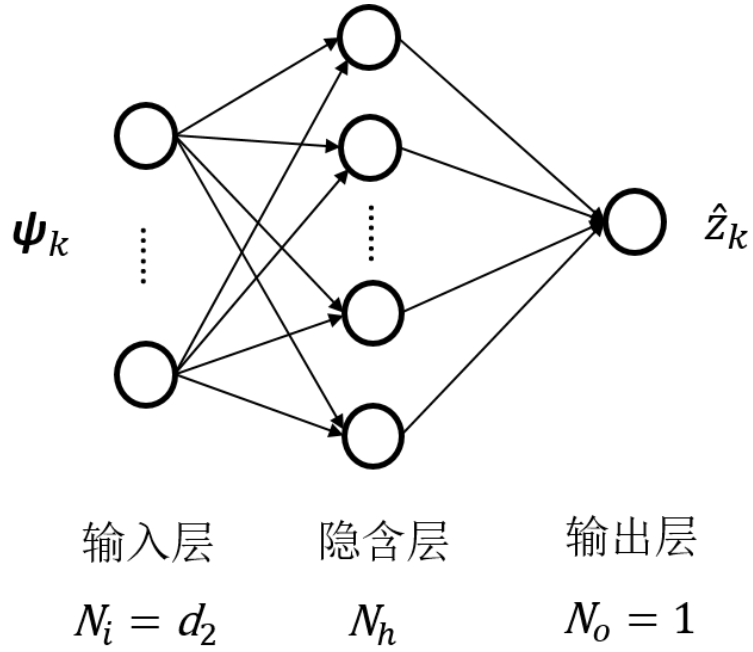


图 4.4 半参数自适应控制系统中采用的单隐层前馈网络结构

针对二维参数情形设计。

1. 初始化阶段

(a) 设置 $k = 0$，系统的初始状态为

$$\begin{aligned} \phi_0 &= [y_0,\ y_{-1}]^T \\ \psi_0 &= [y_0, \ldots, , y_{-p_2}, u_{-1}, \ldots, u_{-q_2+1}] \end{aligned}$$

(b) 根据关于 $\boldsymbol{\theta}$ 的先验信息可以得到初始的多边形为 Γ_0（通常是四边形），以及初始的参数估计 $\hat{\boldsymbol{\theta}}_0$.

(c) 定义 SLFN 网络的隐含层神经元个数为 N_h，正则化因子为 λ 和遗忘因子为 ρ.

(d) 对于第 i 个隐层神经元，随机初始化网络参数 (a_i, b_i)，$i = 1, 2, \ldots, N_h$.

(e) 设置 β^0 和 P_0 为

$$\begin{aligned} P_0 &= (\lambda \boldsymbol{I}_{\boldsymbol{N_h}})^{-1} \\ \beta^0 &= \mathbf{0}_{N_h} \end{aligned} \tag{4.30}$$

这里，$\boldsymbol{I_{N_h}}$ 和 $\mathbf{0}_{N_h}$ 分别记作 $N_h \times N_h$ 的单位矩阵和 $N_h \times 1$ 的零向量。

(f) 设置再学习机制参数。让 $\sigma_k = var(C_k)$ 代表在多边形区域 C_k 的所有顶点坐标的方差，即

$$\begin{aligned} \sigma_{n_k}(1) &= var(X_{P_1}, X_{P_2}, \ldots, X_{P_{n_k}}) \\ \sigma_{n_k}(2) &= var(Y_{P_1}, Y_{P_2}, \ldots, Y_{P_{n_k}}) \end{aligned} \tag{4.31}$$

(g) 设置 $Flag = 0$，意味着尚未激活再学习过程，并记 δ 为学习方差的上限。

(h) 随机产生一个有界的控制输入 u_0，并发送给控制系统(4.1)；设置 $k = 1$，然后转到 Step 2(a).

2. **自适应控制阶段**

(a) 获得系统在第 k 个时刻的输出值 y_k，并计算 $e_{k,1}$ 和 $e_{k,2}$.

(b) 用第三章的公式得到当前的多边形区域 C_k，并将 C_k 的中心作为当前参数估计值 $\hat{\boldsymbol{\theta}}_k$.

(c) 计算 σ_k，如果 $Flag = 0 \bigwedge \sigma_k(1) \leq \delta \bigwedge \sigma_{n_k}(2) \leq \delta$ 那么设置 $\gamma = 1$ 与 $Flag = 1$。

(d) 设置 $\hat{z}_{k-1} = y_k - \phi_{k-1}^T \hat{\boldsymbol{\theta}}_k - u_{k-1}$.

(e) 如果 $\gamma = 1$ 那么执行 (i); 否则，执行 (ii).

i. *Updating:* 将 $(\boldsymbol{\psi}_{k-1}, \hat{z}_{k-1})$ 作为 SLFN 的输入输出样本数据，计算网络的隐含层输出 H_{k-1}，并使用下面的等式计算隐含层到输出层的权重

$$\begin{aligned} P_k =& \frac{1}{\rho}(P_{k-1} - P_{k-1}H_{k-1}^T(\rho \\ &+ H_{k-1}P_{k-1}H_{k-1}^T)^{-1}H_{k-1}P_{k-1}) \\ \beta^k =& \beta^{k-1} + P_{k-1}H_{k-1}^T(\hat{z}_{k-1} - H_{k-1}\beta^{k-1}) \end{aligned} \tag{4.32}$$

ii. *Relearning:* 执行在初始化阶段的步骤 1-(c),1-(d),1-(e).

(f) 将 $\boldsymbol{\psi}_k$ 作为 SLFN 的输入向量，并计算隐含层的输出矩阵 H_k，从而得到非参数的估计

$$\breve{z}_k = H_k\beta^k \tag{4.33}$$

(g) 结合参数和非参数两部分的估计，根据必然等价原理设计控制输入为

$$u_k = y_{k+1}^* - \phi_k^T \hat{\boldsymbol{\theta}}_k - \breve{z}_k \tag{4.34}$$

(h) 设置 $k = k + 1$，然后转到 Step 2(a).

4.4 仿真实例

4.5 本章总结

第 5 章　半参数自适应运动控制

5.1　问题描述

5.2　仿真实例

5.3　本章总结

结论

本文采用……。（结论作为学位论文正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。）本文采用……。（结论作为学位论文正文的最后部分单独排写，但不加章号。结论是对整个论文主要结果的总结。在结论中应明确指出本研究的创新点，对其应用前景和社会、经济价值等加以预测和评价，并指出今后进一步在本研究方向进行研究工作的展望与设想。结论部分的撰写应简明扼要，突出创新性。）

参考文献

[1] 李阳, 朱家仪, 李树民. 非线性控制理论的回顾与展望 [J]. 飞航导弹, 2004 (11): 55–58.

[2] Gata K. 现代控制工程 [M]. 4th ed. 北京: 清华大学出版社, 2005.

[3] 潘胜强. 扰动下线性二次型反馈调节器的鲁棒性与灵敏度 [J]. 控制理论与应用, 1990 (01): 98–104.

[4] Brasch J, Pearson J. Pole placement using dynamic compensators [J]. IEEE Transactions on Automatic Control, 1970 (15): 34–43.

[5] 马宏宾, 张星红, 周浩. 例说自适应控制：从倒立摆谈起 [J]. 系统与控制纵横, 2016, 3 (1): 64–75.

[6] Ball J A, N C. The sensitivity minimization in an Hnorm: parameterization of all optimal solutions [J]. In. J. Contr, 1987, 46: 785–816.

[7] Chen H, Guo L. Identification and Stochastic Adaptive Control [M]. 1991.

[8] 郭雷. 时变随机系统 [M]. 长春: 吉林科学技术出版社, 1992.

[9] Goodwin G C, Ramadge P J, Caines P E. Discrete-Time Multivariable Adaptive Control [J]. IEEE Transactions on Automatic Control, 1980, 25 (3): 449–456.

[10] Ma H, Lum K Y. Adaptive Control. InTech [M]. 2009.

[11] Chen H F, Zhao W X. Recursive identification and parameter estimation [J]. Taylor Francis Usa, 2014.

[12] 李晓理. 一类离散时间非线性系统的多模型自适应控制 [J]. 控制与决策, 2010 (06): 841–846.

[13] 侯忠生. 无模型自适应控制的现状与展望 [J]. 控制理论与应用, 2006 (04): 586–592.

[14] Yang C G, Dai S S, S Ge, Lee T H. Adaptive Asymptotic Tracking Control of a Class of Discrete-Time Nonlinear Systems with Parametric and Nonparametric Uncertainties [C]. In Proceedings of 2009 American Control Conference, St. Louis, USA, June 2009: 580–585.

[15] 侯忠生. 再论无模型自适应控制 [J]. 系统科学与数学, 2014, 34 (10): 1182–1191.

[16] Yang C G, Chai T Y, Zhai L F, et al. Semi-parametric Adaptive Control of Discrete-time Nonlinear Systems [C]. In Proceedings of the IEEE International Conference on Automation and Logistics, Shenyang, 2009.

[17] Xie L L, Guo L. Fundamental Limitations of Discrete-Time Adaptive Nonlinear Control [J]. IEEE Transactions on Automatic Control, 1999, 44 (9): 1777–1782.

[18] Anonymous. Machine Learning [J]. Technology Review, 2014, 117 (6): 9–9.

[19] 周志华. 机器学习 [M]. 清华大学出版社, 2016.

[20] Han X, Xie W F, Fu Z. Nonlinear systems identification using dynamic multi-time scale neural networks [J]. Neurocomputing, 2011, 74 (17): 3428–3439.

[21] Narendra K S, Kannan P. Identification and Control of Dynamical Systems Using Neural Networks [J]. IEEE TransactLons on Neural Networks, 1990, 1 (1).

[22] 朱娟萍, 侯忠生, 熊丹. 神经网络控制、无模型控制 PID 控制仿真比较 [J]. 系统仿真学报, 2005 (03): 751–754+766.

[23] Korjani M M, Bazzaz O, Menhaj M B. Real time identification and control of dynamic systems using recurrent neural networks [J]. Artificial Intelligence Review, 2008, 30 (1-4): 1–17.

[24] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: a new learning scheme of feedforward neural networks [J]. Proc.int.joint Conf.neural Netw, 2004 (2): 985–990.

[25] Huang G B, Siew C K. Extreme learning machine: RBF network case [C]. In Proceedings of Control, Automation, Robotics and Vision Conference, 2004, 2005: 1029–1036.

[26] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: Theory and applications [J]. Neurocomputing, 2006 (70): 489–501.

[27] Huang G, Huang G B, Song S J, et al. Trends in extreme learning machines: A review [J]. Neural Networks the Official Journal of the International Neural Network Society, 2015, 61: 32–48.

[28] Rong H, Zhao G. Direct adaptive neural control of nonlinear systems with extreme learning machine [J]. Neural Computation Application, 2013, 22: 577–586.

[29] Liang N Y, B H G. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks [J]. IEEE TransactLons on Neural Networks, 2006, 17 (6): 1411–1423.

[30] 杨乐, 张瑞. 在线序列 ELM 算法及其发展 [J]. 西北大学学报 (自然科学版), 2012, 42 (6).

[31] Shao Z F, Meng J E. An online sequential learning algorithm for regularized Extreme Learning Machine [J]. Neurocomputing.

[32] 李军, 乃永强. 基于 ELM 的机器人自适应跟踪控制 [J]. 电机与控制学报, 2015, 19 (4): 106–116.

[33] Wang N, Sun J C, Meng J E, et al. A Novel Extreme Learning Control Framework of Unmanned Surface Vehicles [J]. IEEE Transactions on Cybernetics, 2016, 46 (5): 1106–1117.

[34] 郭雷. 关于反馈的作用及能力的认识 [J]. 自动化博览, 2003 (1): 5–7.

[35] Guo1997. On critical stability of discrete-time adaptive nonlinear control [J]. IEEE Transactions on Automatic Control, 1997, 42 (11): 1488–1499.

[36] Xie L L, Guo L. How much uncertainty can be dealt with by feedback [J]. IEEE Transactions on Automatic Control, 2000, 45 (12): 2203–2217.

[37] Guo L. Exploring the maximum capability of adaptive feedback [J]. INTERNATIONAL JOURNAL

OF ADAPTIVE CONTROL AND SIGNAL PROCESSING, 2002, 16: 341–354.

[38] Zhang Y X, Guo L. A Limit to the Capability of Feedback [J]. IEEE Transactions on Automatic Control, 2002, 47 (4): 687–692.

[39] Ma H. Further results on limitations to the capability of feedback [J]. International Journal of Control, 2008, 81 (1): 21–42.

[40] Ma H B. An “impossibility” theorem on a class of high-order discrete-time nonlinear control systems [J]. Systems Control Letters, 2008 (57): 497–504.

[41] Escandell-Montero M, Soria-Olivas E, Magdalena-Benedito R. Regularized extreme learning machine for regression problems [J]. Neurocomputing, 2011, 74 (17): 3716–3721.

[42] Huynh H, Won Y. Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks [J]. Pattern Recognition Letters, 2011, 32 (14): 1930–1935.

[43] 郑胤, 陈权崎, 章毓晋. 深度学习及其在目标和行为识别中的新进展 [J]. 中国图象图形学报, 2014 (02): 175–184.

[44] Akusok A, Bjork K M, Miche Y. High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications [J]. Access IEEE, 2015 (3): 1011–1025.

附录 A　***

附录相关内容…

攻读学位期间发表论文与研究成果清单

[1] HAO ZHOU AND HONGBIN MA* AND NANNAN LI AND CHENGUANG YANG. Semi-parametric Adaptive Control of Discrete-time Systems Using Extreme Learning Machine[C]. The 9th International Conference on Modelling, Identification and Control, 2017, 532 － 535.（EI 检索会议）

致谢

本论文的工作是在导师 *** 教授的指导下独立完成的。

作者简介

本人…。