TUP-Springer Project

Jinkun Liu

机械系统RBF神经网络控制

设计、分析及Matlab仿真

Radial Basis Function (RBF) Neural Network Control for Mechanical Systems

Design, Analysis and Matlab Simulation

清华大学出版社

Springer

Jinkun Liu

机械系统RBF神经网络控制

设计、分析及 Matlab 仿真

Radial Basis Function (RBF) Neural Network Control for Mechanical Systems

Design, Analysis and Matlab Simulation

内 容 简 介

本书从 Matlab 仿真角度，结合典型机械系统控制的实例，系统地介绍了神经网络控制的基本理论、基本方法和应用技术，是作者多年来从事控制系统教学和科研工作的结晶，同时融入了国内外同行近年来所取得的新成果。

全书共分 11 章，包括 RBF 网络的设计及分析、基于梯度下降法的 RBF 网络控制、简单的 RBF 网络自适应控制、RBF 网络滑模控制、基于 RBF 网络逼近的自适应控制、基于 RBF 网络的自适应反演控制、RBF 网络数字控制、离散系统的 RBF 网络控制及自适应 RBF 网络观测器的设计。每种控制方法都通过 Matlab 进行了仿真分析。

本书各部分内容既相互联系又相互独立。本书适用于从事生产过程自动化、计算机应用、机械电子和电气自动化领域工作的工程技术人员阅读，也可作为大专院校工业自动化、自动控制、机械电子、自动化仪表、计算机应用等专业的教学参考书。

图书在版编目(CIP)数据

机械系统 RBF 神经网络控制: 设计、分析及 Matlab 仿真: 英文 / 刘金琨著. --北京：清华大学出版社，2013.3

ISBN 978-7-302-30255-1

Ⅰ.①机…　Ⅱ.①刘…　Ⅲ.①神经网络-应用-机械系统-英文　Ⅳ.①TH122

中国版本图书馆 CIP 数据核字(2012)第 233394 号

责任编辑：薛　慧
责任印制：李红英

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市春园印刷有限公司
经　　销：全国新华书店
开　　本：153mm×235mm　　**印张**：24　　**字数**：545 千字
版　　次：2013 年 3 月第 1 版　　**印次**：2013 年 3 月第 1 次印刷
印　　数：1～1000
定　　价：99.00 元

产品编号：047535-01

Preface

Recent years have seen a rapid development of neural network control techniques and their successful applications. Numerous theoretical studies and actual industrial implementations demonstrate that artificial neural network is a good candidate for function approximation and control system design in solving the control problems of complex nonlinear systems in the presence of different kinds of uncertainties. Many control approaches/methods, reporting inventions and control applications within the fields of adaptive control, neural control, and fuzzy systems, have been published in various books, journals, and conference proceedings. In spite of these remarkable advances in neural control field, due to the complexity of nonlinear systems, the present research on adaptive neural control is still focused on the development of fundamental methodologies.

The advantage of neural networks is that a suitable number of neural network functions can model any (sufficiently smooth) continuous nonlinear function in a compact set, and the modeling error is becoming smaller with an increase of neural network functions. It is even possible to model discontinuous nonlinearities assuming the right choice of discontinuous neural network functions. Thus, an adaptive neural network approach is most suitable in an environment where system dynamics are significantly changing, highly nonlinear, and in principle not completely known.

This book is motivated by the need for systematic design approaches for stable adaptive control system design using neural network approximation-based techniques. The main objectives of the book are to introduce the concrete design method and Matlab simulation of stable adaptive RBF (*Radial Basis Function*) neural control strategies.

It is our goal to accomplish these objectives:

- Offer a catalog of implementable neural network control design methods for engineering applications.
- Provide advanced neural network controller design methods and their stability analysis methods.
- For each neural network control algorithm, we offer its simulation example and Matlab program.

This book provides the reader with a thorough grounding in the neural network control system design. Typical neural network controllers' designs are verified using Matlab simulation. In this book, concrete case studies, which present the results of neural network controller implementations, are used to illustrate the successful application of the theory.

The book is structured as follows. The book starts with a brief introduction of adaptive control and neural network control in Chap. 1, RBF neural network algorithm and design remarks are given in Chap. 2, RBF neural network controller design based on gradient descent algorithm is introduced in Chap. 3, since only local optimization can be guaranteed by using the gradient descent method, and several adaptive RBF neural network controller designs are given based on Lyapunov analysis from Chaps. 4, 5, 6, 7 and 8, which include simple adaptive RBF neural network controller, neural network sliding mode controller, adaptive RBF controller based on global approximation, adaptive robust RBF controller based on local approximation, and backstepping adaptive controller with RBF. In Chap. 9, digital RBF neural network controller design is given. Two kinds of discrete neural network controllers are introduced in Chap. 10. At last, a neural network adaptive observer is recommended and a speedless sliding mode controller design is given in Chap. 11.

I would like to thank Prof. S. S. Ge for his fruitful suggestions. I wish to thank my family for their support and encouragement.

In this book, all the control algorithms and their programs are described separately and classified by the chapter name, which can be run successfully in Matlab 7.5.0.342 version or in other more advanced versions. In addition, all the programs can be downloaded via http://ljk.buaa.edu.cn/. If you have questions about algorithms and simulation programs, please E-mail ljk@buaa.edu.cn.

Table of Notation

Notation	Meaning
$\boldsymbol{R}$	The set of real numbers
$\boldsymbol{R}^n$	The set of all n-dimensional real vectors
$\boldsymbol{R}^{n\times m}$	The set of all $n \times m$-dimensional real matrices
$\|a\|$	The absolute value of scalar a
$\det(\boldsymbol{A})$	The determinant of matrix $\boldsymbol{A}$
$\|\boldsymbol{x}\|$	The norm of vector $\boldsymbol{x}$
$\boldsymbol{A}^T$	The transpose of $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	The inverse of $\boldsymbol{A}$
$\boldsymbol{I}$	An identity matrix
$\boldsymbol{I}_n$	The identity matrix of dimension $n \times n$
$\lambda_i(\boldsymbol{A})$	The ith eigenvalue of $\boldsymbol{A}$
$\lambda(\boldsymbol{A})$	The set of eigenvalues of $\boldsymbol{A}$
$\lambda_{\min}(\boldsymbol{A})$	The minimum eigenvalue of $\boldsymbol{A}$ where $\lambda(\mathbf{A})$ are real
$\lambda_{\max}(\boldsymbol{A})$	The maximum eigenvalue of $\boldsymbol{A}$ where $\lambda(\boldsymbol{A})$ are real
x_i	The ith element of vector $\boldsymbol{A}$
a_{ij}	The ijth element of matrix $\boldsymbol{A}$
$(\hat{\bullet})$	The estimate of $(\bullet)$
$(\tilde{\bullet})$	$(\bullet)$-$(\hat{\bullet})$
$\sup\ \alpha(t)$	The smallest number that is larger than or equal to the maximum value of $\alpha(t)$
$\mathrm{diag}[\cdots]$	Diagonal matrix with given diagonal elements
$\boldsymbol{h}$	Output vector of Gaussian function of RBF
$\boldsymbol{c}_j$	Center vector of the jth net of Gaussian function of RBF
b_j	Width of the jth net of Gaussian function of RBF
$\boldsymbol{W}$	Weight vector of RBF

List of Acronyms

BP	back-propagation
GL	Ge–Lee matrix
LIP	linear-in-the-parameters
MFN	multilayer feed-forward network
MIMO	multi-input multi-output
MNN	multilayer neural networks
NN	neural networks
RBF	radial basis function
RKM	Runge–Kutta–Merson
SISO	single-input single-output
SMC	sliding mode control

Contents

Chapter 1
Introduction

Abstract This chapter gives the review of several kinds of neural network control and introduces the concept of RBF neural network and RBF neural network control. To illustrate the attendant features of robustness and performance specification of RBF adaptive control, a typical RBF adaptive controller design for an example system is given. A concrete analysis, simulation examples, and Matlab programs are given too.

Keywords Neural network control • RBF neural network • Adaptive control

1.1 Neural Network Control

1.1.1 Why Neural Network Control?

Since the idea of the computational abilities of networks composed of simple models of neurons was introduced in the 1940s [1], neural network techniques have undergone great developments and have been successfully applied in many fields such as learning, pattern recognition, signal processing, modeling, and system control. Their major advantages of highly parallel structure, learning ability, nonlinear function approximation, fault tolerance, and efficient analog VLSI implementation for real-time applications greatly motivate the usage of neural networks in nonlinear system identification and control [2].

In many real-world applications, there are many nonlinearities, unmodeled dynamics, unmeasurable noise, multi-loop, etc., which pose problems for engineers to implement control strategies.

During the past several decades, development of new control strategies has been largely based on modern and classical control theories. Modern control theory such as adaptive and optimal control techniques and classical control theory have been based mainly on linearization of systems. In the application of such techniques, development of mathematical models is a prior necessity.

J. Liu, *Radial Basis Function (RBF) Neural Network Control for Mechanical Systems: Design, Analysis and Matlab Simulation*, DOI 10.1007/978-3-642-34816-7_1,

There are several reasons that have motivated vast research interests in the application of neural networks for control purposes, as alternatives to traditional control methods, among which the main points are:

- Neural networks can be trained to learn any function. Thus, this self-learning ability of the neural networks eliminates the use of complex and difficult mathematical analysis which is dominant in many traditional adaptive and optimal control methods.
- The inclusions of activation function in the hidden neurons of multilayered neural networks offer nonlinear mapping ability for solving highly nonlinear control problems where to this end traditional control approaches have no practical solution yet.
- The requirement of vast a priori information regarding the plant to be controlled such as mathematical modeling is a prior necessity in traditional adaptive and optimal control techniques before they can be implemented. Due to the self-learning capability of neural networks, such vast information is not required for neural controllers. Thus, neural controllers seem to be able to be applied under a wider range of uncertainty.
- The massive parallelism of neural networks offers very fast multiprocessing technique when implemented using neural chips or parallel hardware.
- Damage to some parts of the neural network hardware may not affect the overall performance badly due to its massive parallel processing architecture.

1.1.2 Review of Neural Network Control

Conventional methods of designing controllers for a MIMO plant like a multi-joint robot generally require, as a minimum, knowledge of the structure and accurate mathematical model of the plant. In many cases, the values of the parameters of the model also need to be precisely known.

Neural networks, which can learn the forward and inverse dynamic behaviors of complex plants online, offer alternative methods of realizing MIMO controllers capable of adapting to environmental changes. In theory, the design of a neural network-based control system is relatively straightforward as it does not require any prior knowledge about the plant.

The approximation abilities of neural networks have been proven in many research works [3–7], and many adaptive neural network controllers based on the approximation abilities are introduced in some books [8–14]; several research groups have involved in the developments of stable adaptive neural network control techniques.

There have been many papers to be published about neural network control. For example, a unified framework for identification and control of nonlinear dynamic systems was proposed in [15], in which the parametric method of both adaptive nonlinear control and adaptive linear control theory can be applied to

perform the stability analysis. Through introducing the Ge–Lee operator for ease of stability analysis and presentation, systematic and coherent treatments of the common problems in robot neural network control are given in [8]. The typical stable neural network approximation control schemes based on Lyapunov training design are given in [16–18].

The popularization of back-propagation (BP) neural network and RBF neural network have greatly boosted the development of neural control [19]. For example, many neural control approaches have been developed with BP neural network [20–28].

1.1.3 Review of RBF Adaptive Control

In recent years, the analytical study of adaptive nonlinear control systems using RBF universal function approximation has received much attention; typically, these methods are given in [29–37].

The RBF network adaptation can effectively improve the control performance against large uncertainty of the system. The adaptation law is derived using the Lyapunov method so that the stability of the entire system and the convergence of the weight adaptation are guaranteed.

Many simulation examples in this book have indicated that by using RBF control, significant improvement has been achieved when the system is subjected to a sudden change with system uncertainty.

1.2 Review of RBF Neural Network

In 1990, artificial neural networks were first proposed for the adaptive control of nonlinear dynamical systems [38]. Since that time, both multilayer neural networks (MNN) and radial basis function (RBF) networks have been used in numerous applications for the identification and control [39].

RBF neural networks were addressed in 1988 [40], which have recently drawn much attention due to their good generalization ability and a simple network structure that avoids unnecessary and lengthy calculation as compared to the multilayer feed-forward network (MFN). Past research of universal approximation theorems on RBF has shown that any nonlinear function over a compact set with arbitrary accuracy can be approximated by RBF neural network [41, 42]. There have been significant research efforts on RBF neural control for nonlinear systems [24, 43].

RBF neural network has three layers: the input layer, the hidden layer, and the output layer. Neurons at the hidden layer are activated by a radial basis function. The hidden layer consists of an array of computing units called hidden nodes. Each hidden node contains a center c vector that is a parameter vector of the same

dimension as the input vector $\boldsymbol{x}$; the Euclidean distance between the center and the network input vector $\boldsymbol{x}$ is defined by $||x(t) - c_j(t)||$.

The output of hidden layer can be produced through a nonlinear activation function $h_j(t)$ as follows:

$$h_j(t) = \exp\left(-\frac{||\mathbf{x}(t) - \mathbf{c}_j(t)||^2}{2b_j^2}\right), \quad j = 1, \ldots, m \tag{1.1}$$

where b_j notes a positive scalar called a width and m notes the number of hidden nodes. The output layer is a linear weighted combination as follows:

$$y_i(t) = \sum_{j=1}^{m} w_{ji} h_j(t), \quad i = 1, \ldots, n \tag{1.2}$$

where w are the output layer weights, n notes the number of outputs, and y notes the network output.

1.3 RBF Adaptive Control for Robot Manipulators

The control of a multi-input multi-output (MIMO) plant is a difficult problem when the plant is nonlinear and time varying and there are dynamic interactions between the plant variables. A good example of such a plant is a robot manipulator with two or more joints [44].

Robot manipulators have become increasingly important in the field of flexible automation. Through the years, considerable research effort has been made in their controller design. In order to achieve accurate trajectory tracking and good control performance, a number of control schemes have been developed. Computed torque control is one of the most intuitive schemes, which relies on the exact cancellation of the nonlinear dynamics of the manipulator system; however, such a scheme has the disadvantage of requiring the exact dynamic model. In practical engineering, the payload of the robot manipulator may vary during its operation, which is unknown in advance. To overcome these problems, adaptive control strategies for robot manipulators have been developed and have attracted the interest of many researchers, as shown in [45–47]. These adaptive control methods have the advantage, in general, of requiring no a priori knowledge of unknown parameters, such as the mass of the payload.

For rigid robotic manipulators, to relax the requirement for exact knowledge of dynamics, control techniques based on neural networks have been developed. Many books and papers have employed neural network-based schemes for stable tracking control of rigid-link manipulators [8–14, 48–52].

For flexible link manipulators, there are many works about neural network adaptive control [53–55]. For example, a neural controller was proposed for joint-position tracking of flexible link manipulators using singular perturbation techniques [53]; the key feature of the approach is that no exact knowledge of the dynamics of the robot arms is assumed for the controller, and no off-line training is required for the neural network. Neural network-based controllers for tip-position tracking of flexible link manipulators were developed by utilizing the modified output redefinition approach [54]; the a priori knowledge of the payload mass is not needed.

1.4 S Function Design for Control System

1.4.1 S Function Introduction

S function provides a powerful mechanism for extending the capabilities of Simulink. An S function is a computer language description of dynamic system. In the control system, S function can be used to describe controller algorithm, adaptive algorithm, and the plant differential dynamic equation.

1.4.2 Basic Parameters in S Function

1. S function routine: include initialization routine, mdlDerivative routine, and mdlOutput routine.
2. NumContStates: to express the number of continuous states.
3. NumDiscStates: to express the number of discrete states.
4. NumOutputs and NumInputs: to express the number of input and output of the system.
5. DirFeedthrough: means that the output or the variable sample time is controlled directly by the value of an input port.
   An example of a system that requires its inputs (i.e., has direct feedthrough) is the operation $\boldsymbol{y} = \boldsymbol{k} \times \boldsymbol{u}$, where $\boldsymbol{u}$ is the input, $\boldsymbol{k}$ is the gain, and $\boldsymbol{y}$ is the output. An example of a system that does not require its inputs (i.e., does not have direct feedthrough) is the equation $\boldsymbol{y} = \boldsymbol{x}, \dot{\boldsymbol{x}} = \boldsymbol{u}$, where $\boldsymbol{x}$ is the state, $\boldsymbol{u}$ is the input, and $\boldsymbol{y}$ is the output.
6. NumSampleTimes: Simulink provides the following options for sample times, such as continuous sample time, discrete sample time, and variable sample time. For continuous sample time, the output changes in minor steps.