Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique
Département de Génie Electrique
Laboratoire de Commande des Processus

THESE

présentée
en vue de l'obtention du titre de

Docteur en Sciences

En

AUTOMATIQUE

par

Mohammed BELKHEIRI
Magister en Génie Electrique de l'école militaire polytechnique

Intitulé :

Commande non linéaire des systèmes incertains

Application aux systèmes électromécaniques

Soutenue publiquement le **25/11 /2008** devant le jury composé de:

Kamal HARICHE	Professeur àl'Université de Boumerdès	Président
Farès BOUDJEMA	Professeur à l'ENP	Rapporteur
Larbi REFFOUFI	Professeur à l'Université de Boumerdès	Examinateur
Mohamed TADJINE	Professeur à l'ENP	Examinateur
Houcine ZERROUG	Maître de Conférences à l'USTHB	Examinateur
Mokhtar ATTARI	Professeur à l' USTHB	Invité

2007/2008

Thèse préparée au **L**aboratoire de **C**ommande des **P**rocessus de l'**E**cole **N**ationale **P**olytechnique

ملخص : نتطرق من خلال هذه الدراسة الى طرق التحكم في الأنظمة اللاخطية المعرضة الى ارتيابات واخطاء النمذجة التي نصادفها عند التطبيقات الحقيقية . في هذا الاطار اقترحنا طريقة تحكم جديدة معتمدين على طريقة التحكم backstepping مدعمة بالشبكات العصبونبة الاصطناعية القائمة على التكيف . وقد بينا ان كل نظام التحكم المقترح مستقر محليا في ظل الارتيابات و اخطاء النمذجة المحدودة. وذلك باستعمال دالة Lyapunov المرفوعة. ومن السمات الرئيسية لهذه الطريقة هو أنه لاحاجة الى نموذج دقيق للنظام.
لاضفاء مصداقية اكثرلهذا الاسلوب المطروح ارتاينا ان نطبقه على اثنين من النظم الكهرميكانيكية. التطبيق الاول يخص استخدام طرق التحكم الشبكات العصبونية المدعومة لنظام الكبح antilock مخبري تجريبا. والثاني هو عبارة عن تطبيق على محرك لاتزامني عبر المحاكاه.

كلمات مفتاحية : *التحكم اللاخطي، الانظمة الريبية، الشبكات العصبونبة الاصطناعية، Backstepping ، الاستقرار، النظم الكهرميكانيكية.*

Résumé: Dans cette thèse, on présente les approches de commande basées sur le feedback des réseaux de neurones adaptatives qui augmentent un contrôleur backstepping applicable à une classe de systèmes non linéaires incertains. La stabilité de l'architecture de contrôle proposée est assurée localement par une fonction de Lyapunov augmentée. Il est robuste aux incertitudes paramétriques et les dynamiques non modélisées. Un élément clé de ces approches est que l'ordre du système n'est pas besoin d'être connus. Cette méthode est validée sur deux systèmes électromécaniques. La première application consiste à utiliser la commande augmentée par les réseaux de neurones à fin de contrôler un système de freinage antiblocage ABS du laboratoire. La seconde est une application de simulation sur une machine asynchrone.

Mot clés : *Commande Non Linéaire, Systèmes Incertains, Réseaux de Neurones Artificiels, Backstepping, Stabilité, Systèmes Electromécaniques.*

Abstract: Neural network-based adaptive feedback approaches that augment a backstepping control design are described in this thesis for a class of nonlinear uncertain systems. The stability of the proposed control architecture is ensured locally by an augmented Lyapunov function. It is robust to parametric uncertainties and unmodelled dynamics. A key feature of these approaches is that the order of the system need not be known. This method is validated on two electromechanical systems. The first application consists of using Neural Network augmented controller for an antilock braking system ABS laboratory test bed. The second is a simulation application on an Induction Machine.

Key words: *Nonlinear Control, Uncertain Systems, Artificial Neural Networks, Backstepping, Stability, Electromechanical Systems.*

To My wife Nour and My son Youcef.

Table of Contents

Acknowledgements

I would like to thank my committee members for suggestions and corrections that greatly improved this dissertation. First I would like to thank my advisor, Pr. Farès Boudjema for his support, advice, and guidance over the years that I have been working with him. While I was doubtful and my idea was superficial, they could concretely envision what would be achieved, and, without him, I could have never come to this stage. His passion for research has had a great influence on me and I have learned from him how to view things from different aspects. I would also like to thank Pr. Kamal Harich to whom I always feel indebted for his excellent teaching and great courses. I thank Dr. Ahmed Hajjaji for providing opportunity to work with Jerom Bosch and Hamid Rabhi on the project in the CREA Lab, without which Application part would have been missing in my thesis. I am greatly indebted to my family for their long-time, patient support and encouragement, especially to my parents. From birth to now, they have shaped what I was born as into who I am. Finally, my special thanks go to my wife, Nour, who has become a joyful company along this journey.

Laghouat, Belkheiri Mohammed
March 7, 2008

Chapter 1

Introduction

During the two last decades there have been significant developments in the control of highly uncertain, nonlinear dynamical systems where advances in real time applications obliged control community to enhance its classical design tools and adapt to the dynamic industry environment that usually opens new problems where many scientific research fields must interact to solve such real world problems gaining from the booming hardware industry along with all types of proposed solutions that help control engineers to implement the developed control strategies. Adaptive nonlinear control has evolved as a promising alternative for systems with parametric uncertainty, leading to global stability and tracking results for this class of nonlinear systems. Advances in geometric nonlinear control theory [37, 82], in conjunction with the development and refinement of new techniques, such as the backstepping procedure and tuning functions [55], have brought about the design of control systems with proven stability properties in a constructive manner for triangular systems or those systems that can be transformed into this form.

In addition, there has been a lot of research activity on robust nonlinear control design methods, such as sliding mode control, Lyapunov redesign method, nonlinear damping, and adaptive bounding control [55, 57]. These techniques are based on the assumption that the uncertainty in the nonlinear functions is within some known, or partially known, bounding functions. In the same pace with such developments based on Lyapunov stability methods in adaptive nonlinear control, there has been a lot of researchers from all over the world that has tried to adopt artificial intelligence methods such as Artificial Neural Networks (ANN) and fuzzy logic approaches [32, 50].

In these studies, neural networks or fuzzy approximators are proven to be the favorite candidates for approximating unknown nonlinearities that are encountered in real applications where no physical models are known. The basic idea behind such approximators is that their input/output response is modified by adjusting the values of certain inherited parameters, usually referred to as weights. From a mathematical control perspective, neural networks and fuzzy approximators represent just two classes of function approximators. Polynomials, splines, radial basis functions, and wavelets are examples of other function approximators that can be used-and have been used-in a similar setting. Such approximation models with adaptivity features are referred to as adaptive approximators, and control methodologies that are based on them as adaptive approximation based control [40].

Adaptive approximation based control encompasses a variety of methods that appear in the literature: intelligent control, neural control, adaptive fuzzy control, memory-based control, knowledge-based control, adaptive nonlinear control, and adaptive linear control.

Researchers in these fields have diverse backgrounds: mathematicians, engineers, and computer scientists. Therefore, the perspective of the various papers in this area is also varied. However, the objective of the various practitioners is typically similar: to design a controller that can be guaranteed to be stable and achieve a high level of control performance for systems that contain poorly modeled nonlinear effects, or the dynamics of the system change during operation (for example, due to system faults). This objective is achieved by adaptively developing an approximating function to compensate the nonlinear effects during the operation of the system. Many of the original papers on neural or adaptive fizzy control were motivated by such concepts as ease of use, universal approximation, and fault tolerance. Often, ease of use meant that researchers without a control or systems background could experiment with and often succeed at controlling certain dynamics systems, at least in simulation [40].

The rise of interest in the neural and adaptive fuzzy control approaches occurred at a time when desktop computers and dynamic simulation tools were becoming sufficiently cheap at reasonable levels of performance to support such research on a wide basis [32]. However, prior to application on systems of high economic value, the control system designer must carefully consider any new approach within a sound analytical framework that allows rigorous analysis of conditions for stability and robustness [31]. This approach opens a variety of questions that have been of interest to various researchers: What properties should the function approximator have? Are certain families of approximators superior to others? How should the parameters of the approximator be estimated? What can be guaranteed about the properties of the signals within the control system? Can the stability of the approximator parameters be guaranteed? Can the convergence of the approximator parameters be guaranteed? Can such control systems be designed to be robust to noise, disturbances, and unmodeled effects. Can this approach handle significant changes in the dynamics due to system failure. What types of nonlinear dynamic systems are amenable to the approach? What are the limitations?

Adaptive approximation based control can be viewed as one of the available tools that a control designer should have in her/his control toolbox to be able to apply these modern techniques to a certain class of systems, and more importantly to gain enough intuition and understanding about adaptive approximation as a useful tool to be used and how to make necessary modifications or how to combine it with other control tools, so that it can be applied to a wider class of systems where we have not accurate models.

1.1 Motivation

Uncertainty being present in all systems to be controlled, the objective of control design is to achieve a satisfactory performance in the presence of such uncertainty, and no longer its elimination. Designing an effective controller for a nonlinear uncertain system is not always evident and it is a challenging task. The difficulties associated with control design in such systems include structural parameter uncertainties and their variations over time, unmodelled dynamics, inherited nonlinearities, and operation under a wide range of conditions and disturbances. In most instances there is a large degree of uncertainty associated with it and only approximation models can be derived for some operating conditions. The presence of a high degree of uncertainty combined with the desire for high performance provides sufficient motivation for developing approaches to Adaptive function approximation augmented Nonlinear control of nonlinear uncertain systems. Adaptive control is a natural strategy to enhance performance of uncertain systems with minimal sacrifice in performance. Early state feedback approaches in adaptive control were developed for linear systems [7], and for nonlinear systems with linearly parameterized uncertainty [79]. Recent progress in state feedback adaptive control include approaches that do not require any parametrization of state-dependent uncertainties [36, 69].

The use of a neural network (NN) in adaptive control greatly broadened the class of systems that can be treated by adaptive control. Whereas in most classical adaptive control approaches [8], uncertainties are restricted to linearly parameterized uncertainties, NN-based adaptive control allows for functional uncertainty as well. It is well established that a NN can approximate any continuous function to any desired accuracy on a compact set [19, 71, 77]. This universal approximator property of NNs has lead to their evolution as a powerful tool for designing adaptive controllers for uncertain nonlinear systems [63]. In the early to mid 1990' s, the feasibility of using NNs in identification and control for uncertain nonlinear systems was illustrated through various simulation studies [20]. Since the mid 1990' s, several researchers have proposed control methods that employ NNs together with stability proofs based on Lyapunov' s direct method in a state feedback setting [60]. Extensions to NN-based adaptive output feedback have been developed either by employing a high-gain observer [71], or by incorporating a second NN in the estimation process [63]. Applying an adaptive technique for control of uncertain systems, in most cases, implies the replacement of an existing control system. Because of the high cost in this process, it is highly desirable to consider an adaptive approach that can be implemented in a form that augments an existing controller. In particular, within various fields of applications, there exists a legacy of experience with an existing control system architecture, and control designers would prefer to augment their controllers with an adaptive process rather than replace them with a totally new control system. This rationale has been a main driving force for applying adaptive control to augment existing control systems in space robotics [6], power systems [89], temperature regulation [18], and flight control systems [13, 41], to name a few.

Incorporating a NN as a tool for augmenting adaptive control was also tried in [74, 91] in a state feedback setting. Since the late 1990' s, NN-based adaptive control in conjunction with a baseline inverting controller has been applied to flight control systems and led to successful implementation results [43, 47]. This success initiated an attempt to employ NN-based adaptive methods for augmenting a baseline linear controller in [2]. These methods were, however, based on state feedback, and assumed that the dimension of the system is known.

The approaches are robust to parametric uncertainties and unmodelled dynamics. Laboratory experiments have illustrated their effectiveness in a real-time environment [4, 62].

The objective of this work is to develop some control strategies that can solve the control problem of uncertain nonlinear systems and it will be applicable to a wide range of systems using controllers based on adaptive function approximations. The second objective is to apply the developed control algorithms to electromechanical systems that are widely used in industry.

1.2 Contributions of Thesis

The research in this thesis is focused on NN-based adaptive control designs for systems operating at highly nonlinear dynamic regimes which have severe parametric uncertainties and disturbances. The contributions of the research can be summarized as:

- An overview of the existing methods based on robust and adaptive nonlinear control theory that can cope with uncertainties are summarized.
- A new adaptive control design methodology that combines the backstepping approach in addition to NN-based adaptive elements is developed for nonlinear uncertain systems, and its stability is proved through Lyapunov theorems. The performance is validated through simulations [54, 10].
- The proposed control methodology is validated experimentally through an application to a nonlinear and uncertain antilock braking laboratory test bed in which the friction forces are poorly modeled.
- As a second application, we have chosen an induction machine that is widely used as an electromechanical device in industry; It is a MIMO, nonlinear system operated at extremely nonlinear dynamic regimes inducing resistance parameter variations due to high temperature and unknown torque loads [11, 12].
- Issues related to stability and NN weights adaptation are discussed and different solutions are presented.
- A thorough comparison study is performed on the performance of a classical control design and two different classes of neural networks: linearly parameterized Radial Basis Functional (RBF) NN and nonlinearly parameterized Single Hidden Layer (SHL) NN for the speed tracking problem for the induction machine.

1.3 Thesis Outline

The thesis is organized as follows:

Chapter 2 presents some mathematical definitions and tools that are used for analysis and stability proofs in uncertain nonlinear systems.

Chapter 3 introduces the idea of adaptive approximation for addressing unknown nonlinear effects. This chapter includes a simple example comparing various control approaches and concludes with a discussion of components of an adaptive approximation based control system with pointers to the locations in the text where each topic is discussed.

Chapter 4 summaries different structures that are used in the literature of adaptive control as Universal Approximators.

Chapter 5 describes some tools that are used for solving nonlinear control problems in the presence of uncertainties and unmodeled nonlinear dynamics in the framework of robust and adaptive control.

Chapter 6 is devoted to the main result of this thesis where a new adaptive nonlinear control method is proposed to overcome uncertainties in a class of nonlinear systems. This method combines backstepping for the known part of the system and neural networks to cope with the unknown uncertain terms. Stability proof is presented thanks to the augmented Lyapunov Function. The validation of such method on experimental ABS laboratory test bed is highlighted.

Chapter 7 presents an application of NN-based control design on an Electromechanical system, the Induction Machine which is widely used as an electromechanical cheap converter for many industrial applications operating under several kinds of nonlinearities and uncertainties. A command augmentation based adaptive control design is developed for the induction machine, and simulation results show the effectiveness of the design.

Chapter 8 summarizes the results of all research efforts, and concludes the thesis along with some future research direction.

Chapter 2

Analysis of Nonlinear Systems and Uncertainties

2.1 Introduction

In this chapter we briefly describe the main mathematical tools used in nonlinear systems and present different definitions and theorems that are useful to proof system stability. All the definition are derived from the energy-like function, the Lyapunov function whose existence for a linear or nonlinear system ensures its stability. Moreover these definitions address the issue of robustness and its relationship to stability. Most of these definitions are extracted from [82, 87], which are excellent books that contain some advanced detailed stability issues.

2.2 Stability and Robustness

Often, when given the challenge of designing a control system for a particular application, one is provided with a model of the plant that contains the dominant dynamic characteristics. The engineer responsible for the design of a control system may then proceed to formulate a control algorithm assuming that when the model is controlled to within specifications, the true plant will also be controlled within specifications. This approach has been successfully applied to numerous systems. More often, however, the controller may need to be adjusted slightly when moving from the design model to the actual implementation due to a mismatch between the model and the true system. There are also cases where a control system performs well for a particular operating region, but when tested outside that region, performance degrades to unacceptable levels [40].

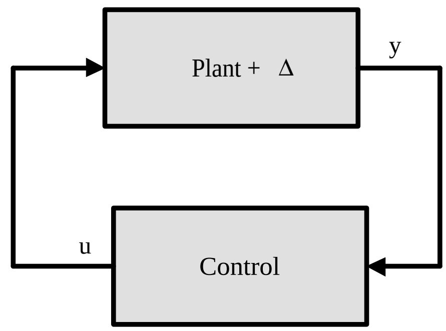

Figure 2.1: Robust control of a plant with unmodeled dynamics.

These issues, among others, are addressed by robust control design. When developing a robust control design, the focus is often on maintaining stability even in the presence of unmodeled dynamics or external disturbances applied to the plant. Figure 2.1 shows the situation in which the controller must be designed to operate satisfactory given any possible plant variation. Unmodeled dynamics are typically associated with every control problem in which a controller is designed based upon a model. This may be due to any one of a number of reasons:

- It may be the case that only a nominal set of parameters are available for the control design. If a controller is to be incorporated into a mass produced product, for example, it may not be practical to measure the exact parameter values for each plant so that a controller can be customized to each particular system.

- It may not be cost effective to produce a model that exactly (or even closely) represents the dynamics. It may be possible to spend fewer resources on a robust control design using an incomplete model than developing a high fidelity model so that traditional non-robust techniques may be used.

Hence, the approach in robust control is to accept a priori that there will be model uncertainty, and try to cope with it.

The issue of robustness has been studied extensively in the control literature. When working with linear systems, one may define phase and gain margins which quantify the range of uncertainty a closed-loop system may withstand before becoming unstable [7]. In the world of nonlinear control design, we often investigate the stability of a closed-loop system by studying the behavior of a

Lyapunov function candidate. The Lyapunov function candidate is a mathematical function designed to provide a simplified measure of the control objectives allowing complex nonlinear systems to be analyzed using a scalar differential equation . When a controller is designed that drives the Lyapunov function to zero, the control objectives are met. If some system uncertainty tends to drive the Lyapunov candidate away from zero, we will often simply add an additional stabilizing term to the control algorithm that dominates the effect of the uncertainty, thereby making the closed-loop system more robust [55].

We will find that by adding a static term in the control law that simply dominates the plant uncertainty, it is often easy to simply stabilize an uncertain plant, however, driving the system error to zero may be difficult if not impossible. Consider the case when the plant is defined by

$$\dot{x} = \theta x + u, \tag{2.2.1}$$

where $x \in \mathbb{R}$ is the plant state that we wish to drive to the point $x = 1, u \in \mathbb{R}$ is the plant input, and θ is an unknown constant. Since θ is unknown, one may not define a static controller that causes $x = 1$ to be a stable equilibrium point. In order for $x = 1$ to be a stable equilibrium point, it is necessary that $\dot{x} = 0$ when $x = 1$, so $u(x) = -\theta$ when $x = 1$. Since θ is unknown, however, we may not define such a controller.

In this case, the best that a static nonlinear controller may do is to keep x bounded in some region around $x = 1$. If dynamics are included in the nonlinear controller, then it turns out that one may define a control system that does drive $x \to 1$ even if θ is unknown. The objective of this work is to use the approach of function approximation based adaptive control to help us define such a nonlinear dynamic controller that will stabilize a certain class of nonlinear uncertain systems.

2.3 Characterizations of Stability and Boundedness

Consider a non autonomous (time-varying) dynamical system described by the following state representation:

$$\dot{x} = f(t, x), \tag{2.3.1}$$

where $x \in \mathbb{R}^n$ is an n dimensional vector and $f : R^+ \times D \to \mathbb{R}^n$ or $D = B_h$ for some $h > 0$, where

$$B_h = \{x \in \mathbb{R}^n : |x| < h\}$$

is a ball centered at the origin with a radius of h. If $D = \mathbb{R}^n$ then we say that the dynamics of the system are defined globally, whereas if $D = B_h$ they are only defined locally. We will not consider systems whose dynamics are defined over disjoint subspaces of $\mathbb{R}^n$. It is assumed that $f(t,x)$ is piecewise continuous in t and Lipschitz in x, for existence and uniqueness of state solutions.

A point x_e is called an equilibrium point of (2.3.1) if $f(t, x_e) = 0$ for all $t \geq 0$. An equilibrium point x_e, is an isolated equilibrium point if there exists an $\rho > 0$ such that the ball around x_e,

$$B_\rho(x_e) = \{x \in \mathbb{R}^n : |x - x_e| < \rho\} \tag{2.3.2}$$

contains no other equilibrium points besides x_e.

2.3.1 Stability Definitions

Stability is a property of systems that we often witness around us. For example, it can refer to the ability of an airplane or ship to maintain its planned flight trajectory or course after displacement by wind or waves. In studies of stability we begin with a model of the dynamics of the system (e.g., airplane or ship) and investigate if the system possesses a stability property. Of course, with this approach we can only ensure that the model possesses (or does not possess) a stability property. In a sense, the conclusions we reach about stability will only be valid about the actual physical system to the extent that the model we use to represent the physical system is valid (i.e., accurate). Having a general intuitive notion of how a stable system behaves, we will show a wide range of precise (and standard) mathematical characterizations of stability and boundedness.

Definition 2.3.1. The equilibrium $x_e = 0$ of (2.3.1) is said to be stable at t_0 (in the sense of Lyapunov) if for every $\epsilon > 0$ and any $t_0 \geq 0$ there exists a $\delta(\epsilon, t_0) > 0$ such that $|x(t, t_0, x_0)| < \epsilon$ for all $t \geq t_0$ whenever $|x_0| < \delta(\epsilon, t_0)$ and $x(t, t_0, x_0) \in B_h(x_e)$ for some $h > 0$.

That is, the equilibrium is stable if when the system (2.3.1) starts close to x_e, then it will stay close to it. Note that stability is a property of an equilibrium, not a system. Often, however, we will refer to a system as being stable if all its equilibrium points are stable. Also, notice that according to this definition, stability in the sense of Lyapunov is a *"local property."* It is a local property since if x_e is stable for some small h, then $x(t, t_0, x_0) \in B_{h'}(x_e)$, for some $h' > h$.

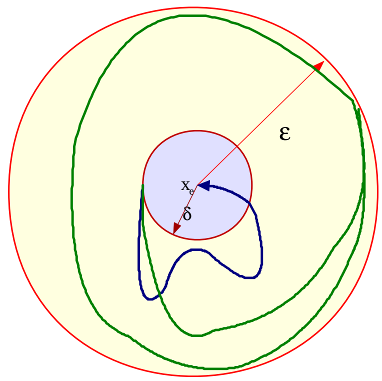

Figure 2.2: Stable and asymptotically stable equilibrium points..

Next, notice that the definition of stability is for a single equilibrium $x_e \in \mathbb{R}^n$ but actually such an equilibrium is a trajectory of points that satisfy the differential equation in (2.3.1). That is, the equilibrium is a solution to the differential $x(t, t_0, x_0) = x_e$ for $t \geq 0$. We call any set such that when the initial condition of (2.3.1) starts in the set and stays in the set for all $t \geq 0$, **an invariant set**.

Definition 2.3.2. An equilibrium that is not stable is called unstable.

Hence, if an equilibrium is unstable, there does not exist an $h > 0$ such that it is stable.

Definition 2.3.3. If in definition 2.3.1, δ is independent of t_0, that is, if $\delta = \delta(\epsilon)$, then the equilibrium x_e is said to be uniformly stable.

Definition 2.3.4. The equilibrium $x_e = 0$ of (2.3.1) is said to be asymptotically stable if it is stable and for every $t_0 \geq 0$ there exists $\eta(t_0) > 0$ such that

$$\lim_{t \to \infty} |x(t, t_0, x_0)| = 0$$

whenever $x_0 < \eta(t_0)$.

That is, it is asymptotically stable if when it starts close to the equilibrium it will converge to it. Asymptotic stability is also a local property. It is a *"stronger"* stability property since it requires that the solutions to the ordinary differential equation converge to zero in addition to what is required for stability in the sense of Lyapunov. See Figure 2.2.

Definition 2.3.5. The equilibrium $x_e = 0$ of (2.3.1) is said to be **uniformly asymptotically stable** if it is uniformly stable and for every $\epsilon > 0$ and $t_0 \geq 0$, there exist a $\delta_0 > 0$ independent of t_0 and ϵ, and a $T(\epsilon) > 0$ independent of t_0, such that $|x(t, t_0, x_0) - x_e| \leq \epsilon$ for all $t \geq t_0 + T(\epsilon)$ whenever $|x_0 - x_e| \leq \delta(\epsilon)$.

Definition 2.3.6. The set $\mathcal{X}_d \subset \mathbb{R}^n$ of all $x_0 \in \mathbb{R}^n$ such that $|x(t, t_0, x_0)| \to 0$ as $t \to \infty$ is called the domain of attraction of the equilibrium $x_e = 0$ of (2.3.1).

Sometimes, if such an $\mathcal{X}_d \subset \mathbb{R}^n$ is known for a system, then it is said to possess a *"regional"* stability property to contrast with the local cases just discussed (and exponential stability below), or the global one to be discussed next.

Definition 2.3.7. The equilibrium $x_e = 0$ is said to be asymptotically stable in the large if $\mathcal{X}_d = \mathbb{R}^n$

That is, an equilibrium is asymptotically stable in the large if no matter where the system starts, its state converges to the equilibrium asymptotically. Notice that this is a global property as opposed to the earlier stability definitions that characterized local properties. This means that for asymptotic stability in the large, the local property of asymptotic stability holds for $B_h(x_e)$ with $h = \infty$ (i.e., on the whole state space).

Definition 2.3.8. The equilibrium $x_e = 0$ is said to be exponentially stable if there exists an $\alpha > 0$ and for every $\epsilon > 0$ there exists a $\delta(\epsilon) > 0$ such that

$$|x(t, t_0, x_0)| \leq \epsilon e^{-\alpha(t-t_0)}, \tag{2.3.3}$$

whenever $|x_0| < \delta(\epsilon)$ and $t \geq t_0 \geq 0$. The constant α is sometimes called - the rate of convergence.

Exponential stability is sometimes said to be a *"stronger"* form of stability since in its presence we know system trajectories decrease exponentially to zero. It is a local property, but we next define its global version.

Definition 2.3.9. The equilibrium $x_e = 0$ is exponentially stable in the large if there exists an $\alpha > 0$ and for every $\beta > 0$ and $\epsilon(\beta) > 0$ such that

$$|x(t, t_0, x_0)| \leq \epsilon(\beta) e^{-\alpha(t-t_0)}, \tag{2.3.4}$$

whenever $|x_0| < \beta$ and $t \geq t_0 \geq 0$.

2.3.2 Boundedness Definitions

Next, we introduce some standard definitions of boundedness. Notice that each of these is a global property of a system in the sense that they apply to trajectories (solutions) of the system that can be defined over all of the state space.

Definition 2.3.10. A solution $x(t, t_0, x_0)$ is bounded if there exists a $\beta > 0$, that may depend on each solution, such that

$$|x(t, t_0, x_0)| \leq \beta, \tag{2.3.5}$$

for all $t \geq t_0 \geq 0$. A system is said to possess **Lagrange stability** if for each $t_0 \geq 0$ and $x_0 \in \mathbb{R}^n$, the solution $x(t, t_0, x_0)$ is bounded.

Definition 2.3.11. The solutions $x(t, t_0, x_0)$ are **uniformly bounded** if for any $\alpha > 0$ and $t_0 \geq 0$, there exists a $\beta(\alpha) > 0$ (independent of t_0) such that if $|x_0| < \alpha$, then $|x(t, t_0, x_0)| \leq \beta(\alpha)$, for all $t \geq t_0 \geq 0$.

Definition 2.3.12. The solutions $x(t, t_0, x_0)$ are said to be **uniformly ultimately bounded** if there exists some $\beta > 0$, and if corresponding to any $\alpha > 0$ and $t_0 > 0$ there exists a $T(\alpha) > 0$ (independent of t_0) such that $|x_0| < \alpha$ implies that $|x(t, t_0, x_0)| \leq \beta$ for all $t \geq t_0 + T(\alpha)$.

Hence, a system is said to be uniformly ultimately bounded if eventually all trajectories end up in a β-neighborhood of the origin.

2.4 Lyapunov s Direct Method

A. M. Lyapunov invented two methods to analyze stability. In his first method (called the indirect method) he showed that if you linearize a system about an equilibrium point, certain conclusions about local stability properties can be made (e.g., if the eigenvalues of the linearized system are strictly in the left half plane then the equilibrium is stable but if at least one is strictly in the right half plane it is unstable).

In his second method (called the direct method) the stability results for an equilibrium $x_e = 0$ of (2.3.1) depend on the existence of an appropriate *"Lyapunov function"* $V : \mathcal{D} \to \mathbb{R}$ where $\mathcal{D} = \mathbb{R}^n$ for global results (e.g., asymptotic stability in the large) and $\mathcal{D} = B_h$ for some $h > 0$, for local results (e.g., stability in the sense of Lyapunov or asymptotic stability). If V is continuously differentiable with respect to its arguments then the derivative of V with respect to t along the solutions of (2.3.1) is

$$\dot{V}(t, x) \mid_{(2.3.1)} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t, x). \tag{2.4.1}$$

The subscript on $\dot{V}$ is sometimes cumbersome, so it will be omitted some times with the understanding that the derivative of V is taken along the solutions of the differential equation at hand.

Finding or construction of a Lyapunov function for a given system is not always evident, which limits the applicability of the method somewhat. However the method does allow us to avoid finding the explicit solution to the nonlinear differential equation in Equation (2.3.1) (which, for some nonlinear ordinary differential equations, can be very difficult or impossible).

2.4.1 Preliminaries: Function Properties

Before we introduce Lyapunov direct method we need the following definitions:

Definition 2.4.1. A function $\gamma : \mathcal{D} \to \mathbb{R}$ is said to be **monotone increasing** on $\mathcal{D} \subseteq \mathbb{R}$ if for every $x, y \in \mathcal{D}$ with $x \leq y$, then $\gamma(x) \leq \gamma(y)$. If for every $x, y \in \mathcal{D}$ with $x < y$, $\gamma(x) < \gamma(y)$, then γ is said to be **strictly increasing**.

Definition 2.4.2. A continuous function $\gamma : \mathcal{D} \to \mathbb{R}^+$ is said to belong to class $\mathcal{K}$ (denoted by $\gamma \in \mathcal{K}$) if it is strictly increasing on $\mathcal{D} = [0, r)$ for some $r \in \mathbb{R}$ or on $\mathcal{D} = [0, \infty)$, and $\gamma(0) = 0$. A continuous function $\gamma : \mathbb{R} \to \mathbb{R}^+$ is said to belong to class $\mathcal{K}_\infty$ if $\gamma \in \mathcal{K}$ with γ defined on $\mathcal{D} = [0, \infty)$ and $\gamma(x) \to \infty$ if $x \to \infty$.

Definition 2.4.3. A continuous function $\beta : \mathcal{D} \times \mathbb{R}^+ \to \mathbb{R}^+$ is said to belong to class- $\mathcal{KL}$ if $\beta(r, s) \in \mathcal{K}$ for each fixed s and $\beta(r, s)$ is decreasing with respect to s for each fixed r with $\beta(r, s) \to \infty$ as $s \to \infty$.

Definition 2.4.4. A continuous function $V(x) : \mathbb{R}^+ \times B_h \to \mathbb{R}(V(t, x) : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R})$ is said to be **positive definite** if $V(t, 0) = 0$ for $t \geq 0$ and there exists a function $\gamma \in \mathcal{K}$ defined on $[0, h)$ such that $V(t, x) \geq \gamma(|x|)$ for all $t \geq 0$ and $x \in B_h$ for some $h > 0$ $(x \in \mathbb{R}^n)$. $V(t, x)$ is said to be **negative definite** if $-V(t, x)$ is positive definite. A continuous function $V(x) : \mathbb{R}^+ \times B_h \to \mathbb{R}(V(t, x) : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R})$ is said to be **positive semidefinite** if $V(t, 0) = 0$ for $t \geq 0$ and $V(t, x) \geq 0$ for all $t \geq 0$ and $x \in B_h$ for some $h > 0$ $(x \in \mathbb{R}^n)$. For **negative semidefinite** replace "$V(t, x) \geq 0$" with "$V(t, x) \leq 0$" in the definition of positive semidefinite.

Definition 2.4.5. A continuous function $V(x) : \mathbb{R}^+ \times B_h \to \mathbb{R}(V(t, x) : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R})$ is said to be **decrescent** if there exists a function $\gamma \in \mathcal{K}$ defined on $[0, r)$ for some $r > 0$ (defined on $[0, \infty)$) such that $V(t, x) \leq \gamma(|x|)$ for all $t \geq 0$ and $x \in B_h$ for some $h > 0$ $(x \in \mathbb{R}^n)$.

Definition 2.4.6. A continuous function $V(x) : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}$ is said to be **radially unbounded** if $V(t, 0) = 0$ for $t \geq 0$ and there exists a function $\gamma \in \mathcal{K}$, such that $V(t, x) \geq \gamma(|x|)$ for all $t \geq 0$ and $x \in \mathbb{R}^n$.

2.4.2 Conditions for Stability

Let $x_e = 0$ be an isolated equilibrium point of (2.3.1). Assume that a unique solution exists to the differential equation in (2.3.1) on $x \in B_h$ for some $h > 0$ for local results, or on $x \in \mathbb{R}^n$ for global results. Below, we let $V : \mathbb{R}^+ \times B_h \to \mathbb{R}^+$ for some $h > 0$ (for local results) or $V : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^+$ for global results be a continuously differentiable function (i.e., it has continuous first order partial derivatives with respect to x and t).

Lyapunov's direct method provides for the following ways to test for stability. The first two are strictly for local properties while the last two local and global versions.

- **Stable:** If $V(t, x)$ is continuously differentiable, positive definite, and $\dot{V}(t, x) \leq 0$, then $x_e = 0$ is stable.

- **Uniformly Stable:** If $V(t,x)$ is continuously differentiable, positive definite, decrescent, and $\dot{V}(t,x) \leq 0$, then $x_e = 0$ is uniformly stable.

- **Uniformly Asymptotically Stable:** If $V(t,x)$ is continuously differentiable, positive definite, and decrescent, with negative definite $\dot{V}(t,x)$, then $x_e = 0$ is uniformly asymptotically stable. or if there exists a continuously differentiable $V(t,x)$ and $\gamma_1, \gamma_2, \gamma_3 \in \mathcal{K}$ defined on $[0,r)$ for some $r > 0$, such that

$$\gamma_1(|x|) \leq V(t,x) \leq \gamma_2(|x|) \tag{2.4.2}$$

$$\dot{V}(t,x) \leq -\gamma_3(|x|) \tag{2.4.3}$$

for all $t \geq 0$ and $x \in B_h$ for some $h > 0$. then $x_e = 0$ is uniformly asymptotically stable.

Theorem 2.4.1. ***LaSalle-Yoshizawa Theorem*** *if there exists a, continuously differentiable $V(t,x)$ and $\gamma_1 \in \mathcal{K}$, such that (2.4.2) holds for all $x \in \mathbb{R}^n$ and $t > 0$, and*

$$\dot{V}(t,x) \leq -W(x) \leq 0$$

for all $t \geq 0$ and $x \in \mathbb{R}^n$, where W is a continuous function (i.e., positive semidefinite), then the solutions of (2.3.1) are uniformly bounded
and

$$\lim_{t\to\infty} W(x(t)) = 0.$$

If, in addition, $W(x)$ is positive definite, then $x_e = 0$ is uniformly asymptotically stable in the large.

- **Exponentially stable:** If there exists a continuously differentiable $V(t,x)$ and $c, c_1, c_2, c_3 > 0$ such that

$$c_1|x|^c \leq V(t,x) \leq c_2|x|^c \tag{2.4.4}$$

$$\dot{V}(t,x) \leq -c_3|x|^c \tag{2.4.5}$$

for all $x \in B_h$ and $t \geq 0$, then $x_e = 0$ is exponentially stable. If there exists a continuously differentiable $V(t,x)$ and Equations (2.4.4) and (2.4.5) hold for some $c, c_l, c_2, c_3 > 0$ for all $x \in \mathbb{R}^n$ and $t > 0$, then $x_e = 0$ is exponentially stable in the large.

Finally, note that in stability analysis it is sometimes convenient to use a, Lyapunov-like function that satisfies all but some properties of a Lyapunov function, then combine the analysis with other properties of the system to conclude convergence.

2.4.3 Conditions for Boundedness

Suppose that there exists a specified function $V(t,x)$ defined on $|x| > R$ (where R may be large) and $t > 0$ that is continuously differentiable. Assume that unique solutions exist to the underlying differential equation over all of $\mathbb{R}^n$.

- **Uniform Boundedness:** if there exists a continuously differentiable $V(t,x)$ and $\gamma_1, \gamma_2, \in \mathcal{K}_\infty$, such that

$$\gamma_1(|x|) \leq V(t,x) \leq \gamma_2(|x|) \tag{2.4.6}$$

$$\dot{V}(t,x) \leq 0 \tag{2.4.7}$$

  for all $|x| \geq R$ and $t \geq 0$ then the solutions to the differential equation are uniformly bounded. Notice that this is less restrictive than the LaSalle-Yoshizawa theorem for uniform boundedness since we only need $\dot{V}(t,x) \leq 0$ for all $|x| \geq R$ for some R, not on all $\mathbb{R}^n$.

- **Uniform Ultimate Boundedness:** if there exists a continuously differentiable $V(t,x), \gamma_1, \gamma_2 \in \mathcal{K}_\infty$, and $\gamma_3, \in \mathcal{K}$ defined on $[0,\infty)$ such that

$$\gamma_1(|x|) \leq V(t,x) \leq \gamma_2(|x|) \tag{2.4.8}$$

$$\dot{V}(t,x) \leq -\gamma_3(|x|), \tag{2.4.9}$$

  for all $|x| \geq R$ and $t \geq 0$ then the solutions to the differential equation are uniformly ultimately bounded.

2.5 Input-to-State Stability

In this section we overview a few concepts from the study of input-to-state stability. We start with definitions, then provide results that will be useful in our laster analysis.

2.5.1 Input-to-State Stability Definitions

In the following we will introduce the basic notions of input-to-state stability and input-to-state practical stability (also referred to as compact input-to-state stability) which are very useful in the study of the stability properties of interconnected systems.

Consider the dynamical system

$$\dot{x} = f(x, u), \tag{2.5.1}$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m, f$ is locally Lipschitz in x and u, representing the input of the system, is a piece-wise continuous and bounded function of t.

Definition 2.5.1. System (2.5.1) is said to be **input-to-state stable** if, for any initial condition $x(t_0)$ and any bounded input $u(t)$ it satisfies

$$|x(t)| \leq \beta(|x(t_0)|, t - t_0) + \gamma\left(\sup_{t_0 \leq \tau \leq t}(|u(\tau)|)\right), \tag{2.5.2}$$

for all $t \geq 0$, where β is class-$\mathcal{KL}$, and γ is class $\mathcal{K}$.

When dealing with uncertainties, it is often useful to define a more general notion of input-to-state stability.

Definition 2.5.2. System (2.5.1) is said to be **input-to-state practically stable** if, for any initial condition $x(t_0)$ and any bounded input $u(t)$, the solution $x(t)$ satisfies

$$|x(t)| \leq \beta(|x(t_0), t - t_0) + \gamma\left(\sup_{t_0 \leq \tau \leq t}(|u(\tau)|)\right) + d, \tag{2.5.3}$$

for all $t \geq 0$, where β is class-$\mathcal{KL}$, and γ is class $\mathcal{K}$ and d is a nonnegative constant.

2.5.2 Conditions for Input-to-State Stability

The following provides useful characterizations of input-to-state stability properties, and a uniform ultimate boundedness property for interconnected systems, in terms of Lyapunov functions.

- **Input-to-State Stability:** System (2.5.2) is input-to-state stable if, and only if, there exists a continuously differentiable function V such that

$$\gamma_1(|x|) \leq V(t, x) \leq \gamma_2(|x|) \tag{2.5.4}$$

$$\frac{\partial V}{\partial x} f(x, u) \leq -\gamma_3(|x|), \qquad \forall |x| \geq \psi(|u|) > 0 \quad , \tag{2.5.5}$$

  where γ_1, γ_2 are class $\mathcal{K}_\infty$ and γ_3, ψ are class $\mathcal{K}$

- **Input-to-State Practical Stability:** System (2.5.2) is input-to-state practical stable if, and only if, there exists a continuously differentiable function V and constants $c > 0, d \geq 0$ such that

$$\gamma_1(|x|) \leq V(t, x) \leq \gamma_2(|x|) \tag{2.5.6}$$

$$\frac{\partial V}{\partial x} f(x,u) \leq -cV + \psi(|u|) + d, \tag{2.5.7}$$

where γ_1, γ_2, ψ are class $\mathcal{K}_\infty$.

- **Uniform Ultimate Boundedness:** Consider $\dot{x} = f(x,y)$ and $\dot{y} = g(x,y)$, where f, g are locally Lipschitz, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. If there exist continuously differentiable functions $V_x, : \mathbb{R}^n \to R$ and $V_y : \mathbb{R}^m \to R$ with $\gamma_{x1}(|x|) \leq V_x \leq \gamma_{x2}(|x|)$ and $\gamma_{y1}(|y|) \leq V_y \leq \gamma_{y2}(|y|)$ such that

$$\dot{V}_x \leq 0 \quad \text{when} V_x \geq V_r \tag{2.5.8}$$

$$\dot{V}_y \leq \gamma_{y3}(|y|), \quad \forall |y| \geq \psi(|x|), \tag{2.5.9}$$

where $\gamma_{x1}, \gamma_{x2}, \gamma_{y1}, \gamma_{y2}$ are class-$\mathcal{K}_\infty, \gamma_{y3}$ and ψ are class-$\mathcal{K}$, and $V_r > 0$, then x and y are uniformly ultimately bounded.

To see why this is the case note the following: From (2.5.8) we find $V_x \leq \max(V_x(0), V_r)$ so

$$\gamma_{x1}(|x|) \leq \max(V(0), V_r). \tag{2.5.10}$$

Thus $|x| \leq d$ for all t, where $d = -\gamma_{x1}^{-1} o \max(V_x(0), V_r)$. If $|y| \geq \psi(d)$, then $V_y \leq -\gamma_{y3} \leq 0$. Thus if $V_y \geq \gamma_{y2} o \psi(d)$ (which implies $|y| \geq \psi(d)$), then $\dot{V}_y \leq 0$ so V_y is bounded. Thus $V_y \leq \max(V(0), \gamma_{y2} o \psi(d))$ so

$$|y| \leq \gamma_{y1}^{-1} o \max(V_y(0), \gamma_{y2} o \psi(d)).$$

for all t.

2.6 Model Uncertainty

Most control designs are based on the use of a design model. The relationship between models and the reality they represent is subtle and complex. A mathematical model provides a mapping from inputs to responses. The quality of a model depends on how closely its responses match those of the true plant. Since no single fixed model can respond exactly like the true plant, we need, at the very least, a set of mappings. However, the modeling problem is much deeper; the universe of mathematical models from which a model set is chosen is distinct from the universe of physical systems. Therefore, a model set which includes the true physical plant can never be constructed. It is necessary for the engineer to make a leap of faith regarding the applicability of a particular design based on a mathematical model. To be practical, a design technique must help make this leap small by accounting for the inevitable inadequacy of models. A good model should be simple enough to facilitate design, yet complex enough to give the engineer confidence that designs based on the model will work on the true plant.

The term uncertainty in control refers to the differences or errors between models and reality, and whatever mechanism is used to express these errors will be called a representation of uncertainty. Representations of uncertainty vary primarily in terms of the amount of structure they contain. This reflects both our knowledge of the physical mechanisms which cause differences between the model and the plant and our ability to represent these mechanisms in a way that facilitates convenient manipulation. For example, consider the problem of bounding the magnitude of the effect of some uncertainty on the output of a nominally fixed linear system. A useful measure of uncertainty in this context is to provide a bound on the spectrum of the output' s deviation from its nominal response. In the simplest case, this spectrum is assumed to be independent of the input. This is equivalent to assuming that the uncertainty is generated by an additive noise signal with a bounded spectrum; the uncertainty is represented as additive noise. Of course, no physical system is linear with additive noise, but some aspects of physical behavior are approximated quite well using this model. This type of uncertainty received a great deal of attention in the literature during the 1960's and 1970's, and the attention is probably due more to the elegant theoretical solutions that are yielded (e.g., white noise propagation in linear systems, Wiener and Kalman filtering, LQG) than to the great practical significance offered [76].

In general, we are forced to use not just a single parameterized model but model sets that allow for plant dynamics which are not explicitly represented in the model structure. Often, when given the challenge of designing a control system for a particular application, one is provided a model of the plant that contains the dominant dynamic characteristics. The engineer responsible for the design of a control system may then proceed to formulate a control algorithm assuming that when the model is controlled to within specifications, the true plant will also be controlled within specifications. This approach has been successfully applied to numerous systems. More often, however, the controller may need to be adjusted slightly when moving from the design model to the actual implementation due to a mismatch between the model and true system. There are also cases when a control system performs well for a particular operating region, but when tested outside that region, performance degrades to unacceptable levels.

2.7 Conclusion

Within this chapter we have presented various mathematical tools that have been found to be useful in the stability analysis of nonlinear systems. In particular, we have covered the following topics:

- Stability definitions: stability in the sense of Lyapunov, uniformly stability, asymptotically stability (in the large), exponentially stability (in the large).
- Boundedness definitions: Lagrange stability, uniform boundedness, uniform ultimate boundedness.
- Direct method for stability and boundedness analysis (including results for all the stability and boundedness definitions)
- The LaSalle-Yoshizawa theorem and a special case of the invariance theorem.
- Input-to-state stability definitions and analysis.
- Uncertainty issues.

Chapter 3

Nonlinear Function Approximators For Control

3.1 Introduction

This chapter treats the background of function approximation, and additionally, it will serve as a connection between the actual problem and the theory in subsequent chapters. In order to make the problem clear, we start with the setting in which the function approximator has to operate. This setting indicates how the training data is generated. Based on this setting, a minimization problem is formulated for the function approximator. Different function approximators use different techniques to approximate the data. Neural networks and fuzzy systems are presented as universal approximators for phenomena where no mathematical model exists to describe it. They have been found to be truly interdisciplinary tools appearing in the fields of economics, business, science, psychology, biology, and engineering to name a few [75]. Based upon the structure of a biological nervous system, artificial neural networks use a number of interconnected simple processing elements ("neurons") to accomplish complicated classification and function approximation tasks [35]. The ability to adjust the network parameters (weights and biases) makes it possible to "learn" information about a process from data, whether it is describing stock trends or the relation between an actuator input and some sensor data. Neural networks typically have the desirable feature that little knowledge about a process is required to successfully apply a network to the problem at hand (although if some domain-specific knowledge is known then it can be beneficial to use it). In other words, they are typically regarded as a "black box" technique. This approach often leads to engineering solutions in a relatively short amount of time since expensive system models required by many conventional approaches are not needed. Of course, however, sufficient data is typically needed for effective solutions. Fuzzy systems are intended to model higher level cognitive functions in a human. They are normally broken into (1) a rule-base that holds a humans knowledge about a specific application domain, (2) an inference mechanism that specifies how to reason over the rule-base, (3) fuzzification which transforms incoming information into a form that can be used by the fuzzy system, and (4) defuzzification which puts the conclusions from the inference mechanism into an appropriate form for the application at hand. Often, fuzzy systems are constructed to model how a human performs a task [48]. They are either constructed manually (i.e., using heuristic domain-specific knowledge in a manner similar to how an expert system is constructed) or in a similar manner to how a neural network is constructed via training with data. While in the past fuzzy systems were exclusively constructed with heuristic approaches we take the view here that they are simply an alternative approximator structure with tunable parameters (e.g., input and output membership function parameters) and hence they can be viewed as a "black box approach" in the same way as neural networks can. Fuzzy systems do, however, sometimes offer the additional beneficial feature of a way to incorporate heuristic information by simply specifying some rules via heuristics and tuning the others using data. In other words, it is sometimes easier to specify a good initial guess for the fuzzy system. In this chapter we define some basic fuzzy systems and neural networks, in fact, ones that are most commonly used in practice. We do not spend time discussing the heuristic construction of fuzzy systems since this is treated in detail elsewhere. In the next chapter we will provide a variety of optimization methods that may be used to help specify the parameters used to define neural networks and fuzzy systems.

3.2 Components of Approximation Based Control

Implementation or analysis of an adaptive approximation-based control system requires the designer to properly specify the problem and solution. This section discusses major aspects of the problem specification.

3.2.1 Control Architecture

Specification of the control architecture is one of the critical steps in the design process. Various nonlinear control methodologies and rigorous tools to analyze their performance have been developed in recent decades [37, 46, 55, 57, 78, 82, 87]. The choices made at this step will affect the complexity of the implementation, the type and level of performance that can be guaranteed, and the properties that the approximated function must satisfy. Major issues influencing the choice of control approach are the form of the system model and the manner in which the nonlinear model error appears in the dynamics.

Consider a dynamic system that can be described as

$$\begin{cases} \dot{x}_i = x_{i+1} \qquad \texttt{for} \quad i = 1, .., n-1 \\ \dot{x}_n = (f_o(x) + f^*(x)) + (g_o(x) + g^*(x))u \\ y = x_1, \end{cases} \tag{3.2.1}$$

where $x(t)$ is the state of the system, $u(t)$ is the control input, f_o and g_o represent the known portions of the dynamics (i.e, the design model), and f^* and g^* are unknown nonlinear functions. Let $\hat{f}$ and $\hat{g}$ represent approximations to the unknown functions f^* and g^*. Then, a feedback linearizing control law can be defined as

$$u(t) = \frac{1}{g_o + \hat{g}}(\nu(t) - f_o(x) - f^*(x)) \tag{3.2.2}$$

where $\hat{g}(t) > -g_o(t)$ and $\nu(t)$ can be specified as a function of the tracking error to meet the performance specification. If the approximations were exact (i.e., $f^* = \hat{f}$ and $g^* = \hat{g}$), then this control law would cancel the plant dynamics resulting in

$$\dot{x}_n = \nu(t).$$

When the approximators are not exact, the tracking error dynamic equations are

$$\dot{x}_n = \nu + (f^*(x) - \hat{f}(x)) + (g^*(x) - \hat{g}(x))u(t) \tag{3.2.3}$$

If adaptive approximation is not used (i,e., $\hat{f}(x) = \hat{g}(x) = 0$), the tracking error will be determined by the n-th integral of the interaction between the control law specified by ν and the model error, as expressed by (3.2.1).

There exist different methods such as Lyapunov redesign, nonlinear damping, and sliding mode to be used by the user to overcome the unknown nonlinear effects. These methods work by adding terms to the control law designed to dominate the worst case modeling error, therefore they may involve either large magnitude or high bandwidth control signals [40].

Alternatively, adaptive approximation methods accumulate model information and attempt to remove the effects of a specific set of nonlinearities that fit the model information. It is not possible to approximate an arbitrary function over the entire $\mathbb{R}^n$. Instead, we must restrict the class of functions, constrain the region over which the approximation is desired, or both. Since the operating envelope is already restricted for physical reasons, we will desire the ability to approximate the functions f^* and g^* only over the compact set denoted by $\mathcal{D}$. Note that $\mathcal{D}$ is a fixed compact set, but its size can be selected as large as need be at the design stage. Therefore, we are seeking to show that initial conditions outside $\mathcal{D}$ converge to $\mathcal{D}$ and that for trajectories in $\mathcal{D}$ the trajectory tracking error converges in a desired sense. Various techniques to achieve this are thoroughly discussed and detailed in the following chapters.

3.2.2 Function Approximator

Having analyzed the control problem and specified a control architecture capable of using an approximated function to improve the system control performance, the designer must specify the form of the approximating function. This specification includes the definition of the inputs and outputs of the function, the domain $\mathcal{D}$ over which the inputs can range, and the structure of the approximating function. This is a key performance limiting step. If the approximation capabilities are not sufficient over $\mathcal{D}$, then the approximator parameters will be adapted as the operating point changes with no long term retention of model accuracy.

For the discussion that follows, the approximating function will be denoted $\hat{f}(x; \theta, \sigma)$ where

$$\hat{f}(x; \theta, \sigma) = \theta^T \phi(x, \sigma) \tag{3.2.4}$$

In this notation x is a dummy variable representing the input vector to the approximation function. The actual function inputs may include elements of the plant state, control input, or outputs. The notation $\hat{f}(x; \theta, \sigma)$ implies that $\hat{f}$ is evaluated as a function of x when θ and σ are considered fixed for the purposes of function evaluation. In applications, the approximator parameters θ and σ will be adapted online to improve the accuracy of the approximating function - this is referred to as training in the neural network literature. The parameters θ are referred to in the (neural network) literature as the output layer parameters. The parameters σ are referred to as the input layer parameters. Note that the approximation of eqn. (3.2.4) is linear-in-the-parameters with respect to θ. The vector of basis functions ϕ will be referred to as the regressor vector. The regressor vector is typically a nonlinear function of x and the parameter vector σ. Specification of the structure of the approximating function includes selection of the basis elements of the regressor ϕ, the dimension of θ, and the dimension of σ. The values of θ and σ are determined through parameter estimation methods based on the online data. Regardless of the choice of the function approximator and its structure, it will normally be the case that perfect approximation is not possible. The approximation error is denoted by $e(x;\theta,\sigma)$ where

$$e(x;\theta,\sigma) = f(x) - \hat{f}(x;\theta,\sigma) \tag{3.2.5}$$

If θ^* and σ^* denote parameters that minimize the ∞-norm of the approximating error over a compact region D, then the Minimum Functional Approximation Error (MFAE) is defined as

$$e_\phi(x) = e(x;\theta^*,\sigma^*) = f(x) - \hat{f}(x;\theta^*,\sigma^*) \tag{3.2.6}$$

In practice, the quantities e_ϕ, θ^* and σ^* are not known, but are useful for the purposes of analysis. Note that $e_\phi(x)$ acts as a disturbance affecting the tracking error and therefore the parameter estimates. Therefore, the specification of the adaptive approximator $\hat{f}(x;\theta,\sigma)$ has a critical affect on the tracking performance that the approximation based control system will be capable of achieving. The approximator structure defined in eqn. (3.2.4) is sufficient to describe the various approximators used in the neural and fuzzy control literature, as well as many other approximators.

3.2.3 Stable Training Algorithm

Given that the control architecture and approximator structure have been selected, the designer must specify the algorithm for adapting the adjustable parameters θ and σ of the approximating function based on the online data and control performance. Parameter estimation can be designed for either a fixed batch of training data or for data that arrives incrementally at each control system sampling instant. The latter situation is typical for control applications; however, the batch situation is the focus for much of the traditional function approximation literature. In addition, much of the literature on function approximation is devoted to applications where the distribution of the training data in D can be specified by the designer. Since a control system is completing a task during the function approximation process, the distribution of training data usually cannot be specified by the control system designer. The portion of the function approximation literature concerned with batches of data where the data distribution is defined by the experiment and not the analyst is referred to as scattered data approximation methods [73]. Adaptive approximation-based control applications are distinct from traditional batch scattered data approximation problems in that:

- the data involved in the parameter estimation will become available incrementally (ad infinitum) while the approximated function is being used in the feedback loop;
- the training data might not be the direct output of the function to be approximated; and,
- the stability of the closed-loop system, which depends on the approximated function, must be ensured.

The main issue to be considered in the development of the parameter estimation algorithm is the overall stability of the closed-loop control system. The stability of the closed-loop system requires guarantees of the convergence of the system state and of (at least) the boundedness of the error in the approximator parameter vector.

This analysis must be completed with caution, as it is possible to design a system for which the system state is asymptotically stable while

1. even when perfect approximation is possible (i.e., $e_\phi = 0$), the error in the estimated approximator parameters is bounded, but not convergent;
2. when perfect approximation is not possible, the error in the estimated approximator parameters may become unbounded.

In the first case, the lack of approximator convergence is due to lack of persistent excitation. This lack of approximator convergence may be acceptable, if the approximator is not needed for any other purpose, since the control performance is still achieved; however, control performance will improve as approximator accuracy increases. Also, the designer of a control system involving adaptive approximation sometimes has interest in the approximated function and is therefore interested in its accuracy. In such cases, the designer must ensure the convergence of the control state and approximator parameters. In the second case (the typical situation), the fact that e_ϕ cannot be forced to zero over $\mathcal{D}$ must be addressed in the design of the parameter estimation algorithm.

3.3 Function Approximation Based Control Motivation

The objective of adaptive approximation-based control methods is to achieve a higher level of control system performance than could be achieved based on the *a priori* model information. Such methods can be significantly more complicated (computationally and theoretically) than non-adaptive or even linear adaptive control methods. This extra complication can result in unexpected behavior (e.g., instability) if the design is not rigorously analyzed under realistic assumptions. Adaptive function approximation has an important role to play in the development of advanced control systems. Adaptive approximation-based control, including neural and fuzzy approaches, have become feasible in recent decades due to the rapid advances that have occurred in computing technologies. Inexpensive desktop computing has inspired many ad hoc approximation-based control approaches. In addition, similar approaches in different fields (e.g., neural, fuzzy) have been derived and presented using different nomenclature yet nearly identical theoretical results. Our objective herein is to present such approaches rigorously within a unifying framework so that the resulting presentation encompasses both the adaptive fuzzy and neural control approaches, thereby allowing the discussion to focus on the underlying technical issues. The three terms, adaptation, learning, and self-organization, are used with different meanings by different authors. In this thesis, we will use adaptation to refer to temporal changes. For example, adaptive control is applicable when the estimated parameters are slowly varying functions of time. We will use learning to refer to methods that retain information as a function of measured variables. Herein, learning is implemented via function approximation. Therefore, learning has a spatial connotation whereas adaptation refers to temporal effects. The process of learning requires adaptation, but the retention of information as a function of other variables in learning implies that learning is a higher level process than is adaptation. Implementation of learning via function approximation requires specification of the function approximation structure. This specification is not straightforward, since the function to be approximated is assumed to be unknown and input-output samples of the function may not be available *a priori*. For the majority of this text, we assume that the designer is able to specify the approximation structure prior to online operation. However, an unsolved problem in the field is the online adaptation of the function approximation structure. Methods that adapt the function approximation structure during online operation are referred to as self-organizing. Since most physical dynamic systems are described in continuous-time, while most advanced control systems are implemented via digital computer in discrete-time, the designer may consider at least two possible approaches. In one approach, the design and analysis would be performed in continuous-time with the resulting controller implemented in discrete-time by numeric integration. The alternative approach would be to transform the continuous-time ordinary differential equation to a discrete-time model that has equivalent state behavior at the sampling instants and then perform the control system design and analysis in discrete-time.

3.4 Approximation Theory

This section formulates the numeric data processing issues of interpolation and function approximation, and then discusses function approximator properties that are relevant to the use of adaptive approximation for estimation and feedback control. Our interest in function approximation is derived from the hypothesis that online control performance could be improved if unknown nonlinear portions of the model are more accurately modeled.

Although the data to improve the model may not be available a priori, additional data can be accumulated while the system is operating. Appropriate use of such data to guarantee performance improvement requires that the designer understand the areas of function approximation, control, stability, and parameter estimation. This chapter focuses on several aspects of approximation theory. The discussion of function approximation is subdivided into off-line and online approximation. Off-line function approximation is concerned with the questions of selecting a family of approximators and parameters of a particular approximator to optimally fit a given set of data. The issue of the design of the set of data is also of interest when the acquisition of the data is under the control of the designer. An understanding of off-line function approximation is necessary before delving into online approximation. The discussion of online approximation builds on the understanding of off-line approximation, and also raises new issues motivated by the need to guarantee stability of the dynamic system and estimation process, the possible need to forget old stored information at a certain rate, and the inability to control the data distribution.

3.4.1 Motivating Example

Consider the following simple example that illustrates some of the issues that arise in approximation based control applications.

$$\begin{aligned} x(k+1) &= f(x(k)) + u(k) \\ y(k) &= x(k), \end{aligned} \tag{3.4.1}$$

where $u(k)$ is the control variable at discrete-time k , $x(k)$ is the state, $y(k)$ is the measured output, the function $f(x)$ is not known to the designer, and the control law is given by

$$u(k) = y_d(k+1) - \beta[y_d(k) - y(k)] - \hat{f}(y(k)). \tag{3.4.2}$$

The above control law assumes that the reference trajectory y_d is known one step in advance. For the purposes of simulation in the example, we will use $f(x) = sin(x)$.

If $\hat{f}(y) = sin(y)$, then the closed-loop tracking error dynamics would be

$$e(k+1) = \beta e(k).$$

where $e(k) = yd(k) - x(k)$, which is stable for $\beta < 1$ (in the following simulation example we use $\beta = 0.5$). If $\hat{f}(y) \neq f(x)$, then the closed-loop tracking error dynamics would be

$$e(k+1) = \beta e(k) - [f(x(k)) - \hat{f}(y(k))]. \tag{3.4.3}$$

Therefore, the tracking performance is directly affected by the accuracy of the design model $f(x)$. The left hand column of Figure 3.1 shows the performance of this closed-loop system when $y_d(k) = \pi sin(0.1k)$ and $f(y) = 0$. When $f(x)$ is not known *a priori*, the designer may attempt to improve the closed loop performance by developing an online (i.e., adaptive) approximation to $f(x)$. In this section a straightforward database function approximation approach is used. At each time step k , the data

$$z(k+1) = [f(y(k-1)), y(k-1)]$$

will be stored. Note that the approach of this example requires that the function value $f(y(k-1))$ must be computable at each step from the measured variables. This assumed approach is referred to as supervised learning. This is a strict assumption that is not always applicable. For this example, at time k , the information in $z(k)$ can be computed from available data according to

$$z(k+1) = [y(k) - u(k-1), y(k-1)]$$

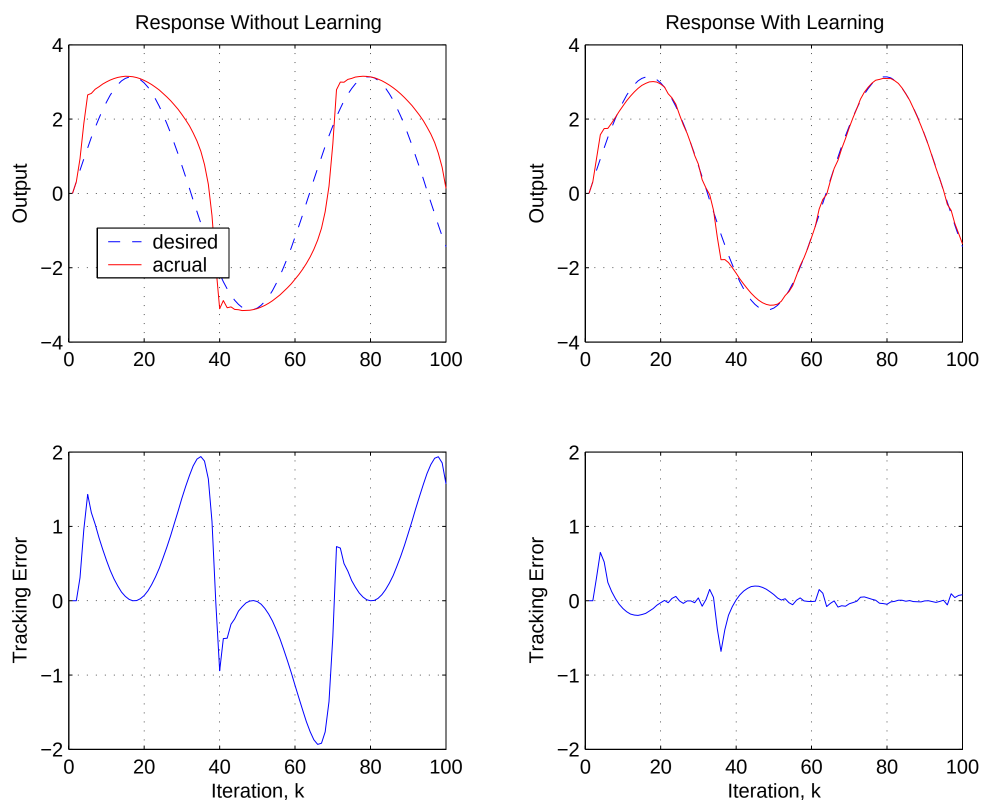

Figure 3.1: Closed-loop control performance for eqn. (3.4.1). Left column corresponds to $\hat{f} = 0$. Right column corresponds to $\hat{f}$ constructed via nearest neighbor matching. For the top row of graphs, the dotted line is the reference trajectory. The solid line is the system response. The tracking error is plotted in the bottom row of graphs.

At time step k with $y(k)$ available, $u(k)$ is calculated using eqn. (3.4.1) as follows: (1) search the second column of z for the row i that most closely matches $y(k)$ (i.e., $i = argmin_{o<j<k}(\|z(j,2) -$

$y(k)\|)$, (2) use $f(y(k)) = z(i, 1)$. The remaining terms in eqn.(3.4.1) can be directly calculated. The right-hand column of Figure 3.2 shows the performance of the closed-loop system using this adaptive approximation based method. Note that as the row dimension of z grows with k (i.e., more data values for $f(x)$ are stored) the tracking performance rapidly improves. However, both the memory required to store z and the computation required to search z increase at each iteration.

The top graph of Figure 3.2 plots as discrete points the first column of z as a function of the second column of z. The approximate function used in the control law is piecewise constant with each piecewise section (of variable width) centered on one of the examples $y(i)$, as shown in the bottom graph of Figure 3.2. With noise-free data, the approximation becomes very good for large k. The approach defined above is referred to as nearest neighbor matching. Various other alternatives are possible such as k-nearest neighbor averaging, which perform better when noise is present in the measurement data.

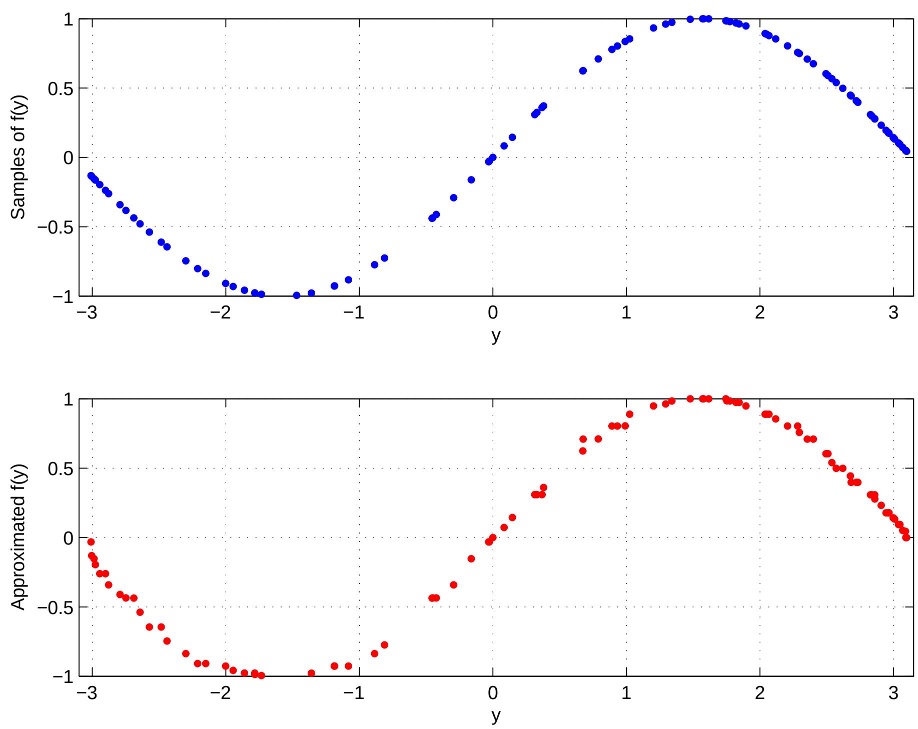

Figure 3.2: Top - Data for approximating $\hat{f}$ using nearest neighbor matching. Bottom - Approximated $\hat{f}$ resulting from nearest neighbor matching. Both graphs correspond to Example 2.1.

Note the following issues related to this example and the broader adaptive function approximation problem:

1. The input-output training data $(f(y(i)), y(i))$ cannot be expected to be distributed according to an analytic distribution. Instead, the training data will be defined by the control task that the system is performing. The distribution of training data over a fixed-duration window will typically be time varying. If control is operating well, then the training samples will cluster in the vicinity of a state trajectory (several may be possible) defined by the reference input. In particular, over short periods of time, the training data will not be uniformly distributed, but will cluster in some small subregion of the domain of approximation. For example, if the control objective is regulation to a certain fixed point (i.e., $y_d(k) =$ constant) then the training data may cluster around a single point.

2. When the raw training data are stored, as in this example, the approach will have growing memory and computational penalties. These can be overcome by the function approximation and recursive parameter estimation techniques to be described.

3. Consider the case of measurement data corrupted by noise. Direct storage of the data does not work as well as shown in Figures 3.1 and 3.2. Figure 3.3 shows performance in the time domain when the measured $y(k)$ is corrupted with Gaussian random noise $n(k$ with standard deviation $\sigma = 0.1$. In this case, $y(k) = x(k) + n(k)$ is stored in the database calculations and used in the control law. The actual tracking error $(x - y_d)$ is plotted. For $k > 100$, the tracking error has standard deviation of 0.16. So the approach has amplified the effects of noise. In this approach, noisy data are stored in the data vector without noise attenuation. It is important to note that, as we will see, methods to attenuate noise through averaging lead directly to function approximation methods.

4. Function approximation problems are not well defined. Consider Figure 3.4, which corresponds to the the data matrix z stored relative to Figure 3.3. If the domain of approximation that is of interest is $D = [-\pi, \pi]$, how should the approximation given the available data be extended to all of D (or should it?). A quick inspection of the data might lead to the conclusion that the function is linear. A more careful inspection, noting the apparent curvature near $\pm\frac{\pi}{2}$; might result in the use of a saturating function. From our knowledge of $f(x)$ neither of these

is of course correct. Extreme care must be exercised in generalizing from available data in given regions to the form of the function in other regions. The manner in which data in one region affects the approximated function in another region is determined primarily by the specification of the function approximator structure. The assumed form of the approximation inserts the designers bias into the approximation problem. The effect of this bias should be well understood.

5. From eqn. (3.4.4) the designer might expect that, as the database accumulates data, then the $(f - \hat{f})$ term and hence e should decrease; however, the control and function approximation approach of this example did not allow a rigorous stability analysis. The parametric function approximation methods that follow will enable a rigorous analysis of the stability properties of the closed-loop system.

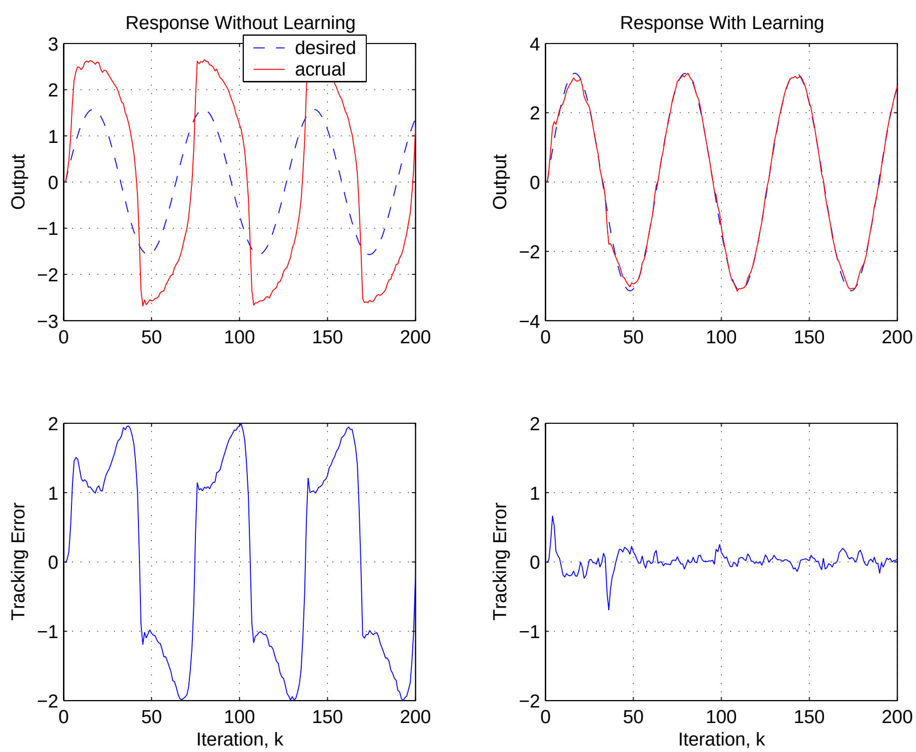

Figure 3.3: Closed-loop control performance for eqn. (3.4.1) with noisy measurement data. Left column corresponds to f = 0. Right column corresponds to f constructed via nearest neighbor matching. In the top row of graphs, the dotted line is the reference trajectory and the solid line is the system response.

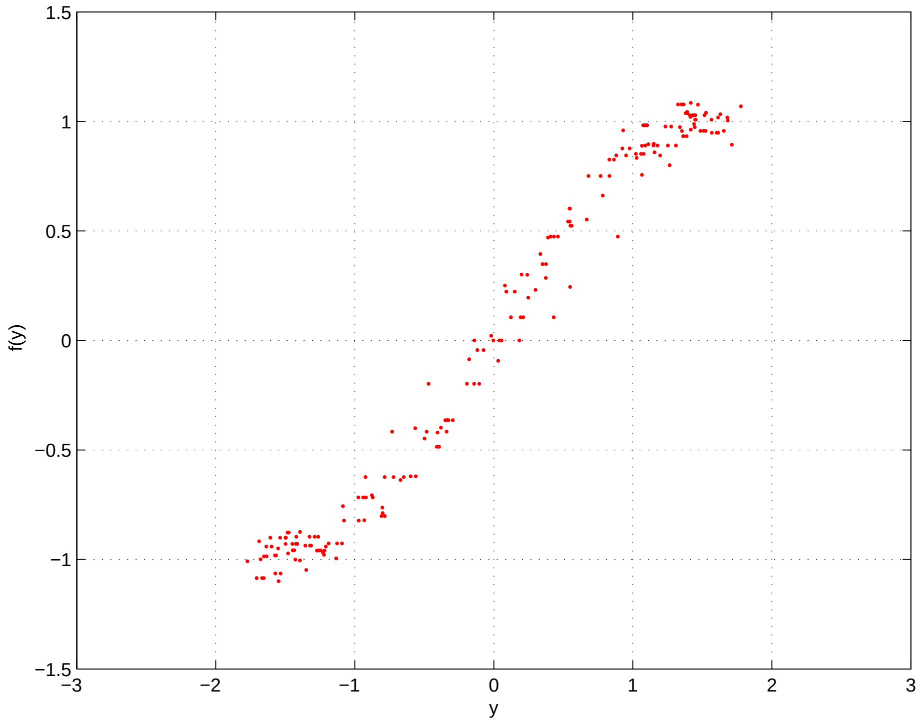

Figure 3.4: Data for approximating $\hat{f}$ corresponding to eqn. (3.4.1) with noisy measurement data.

Items 2 through 4 above naturally direct the attention of the designer to more general function interpolation and approximation issues. The above nearest neighbors approach can be represented as

$$\hat{f}(x : z(k)) = \sum_{i=1}^{k} z(i,1)\phi_i(x : z(k))) \tag{3.4.4}$$

where the notation $\hat{f}(x : z(k))$ means the value of $\hat{f}$ evaluated at x given the data in database matrix z at time k, and

$$\phi_i(x : z(k))) = \begin{cases} 1, & \text{if } |x - z(i,2)| = \min_{1 \le j \le k} |x - z(i,2)| \ ; \\ 0, & \text{otherwise.} \end{cases} \tag{3.4.5}$$

where we have assumed that no two entries (i.e., rows) have the same value for $z(j,2)$. Note that by its definition, this function passes exactly through each piece of measured data (i.e., $f(z(i,2) : z(k)) = z(i,1)$). This is referred to as interpolation. Item 2 above points out the fact that this approximation structure has k basis elements that are redefined at each sampling instant. The computational complexity and memory requirements can be decreased and fixed by using a more fixed number N of basis elements of the form

$$\hat{f}(x:\theta,\sigma,N)=\sum_{i=1}^{N}\theta_i\phi_i(x:\sigma(i))) \tag{3.4.6}$$

where the data matrix z would be used to estimate $\theta=[\theta_1,\cdots,\theta_N]$ and $\sigma=[\sigma_1,\cdots,\sigma_N]$. With such a structure, it will eventually happen that there is more data than parameters, in which case interpolation may no longer be possible. After this instant in time, a well designed parameter estimation algorithm will combine new and previous measurements to attenuate the affects of measurement noise on the approximated function. The choice of basis functions can affect the noise attenuation properties of the approximator. In addition, the choice of approximator will affect the accuracy of the approximation, the degree of approximator continuity and the extent of training generalization, as will be explained in Section §§§.

3.4.2 Interpolation

Given a set of input-output data $\{(x_j,y_j)|j=1,\cdots,m;x_j\in\Re^n,y_j\in\Re^1\}$, function interpolation is the problem of defining a function $\hat{f}(x):\Re^n\rightarrow\Re^1$ such that $\hat{f}(x_j)=y_j$ for all $j=1,\cdots,m$. When $\hat{f}(x)$ is constrained to be an element of a finite dimensional linear space, this is called *Lagrange interpolation*. The interpolating function $\hat{f}(x)$ can then be used to estimate the value of $f(x)$ between the known values of $f(x_j)$.

In Lagrange interpolation with the basis functions $\phi_i(x)_{=1}^{N}$,

$$\hat{f}(x)=\sum_{i=1}^{N}\theta_i\phi_i(x)=\theta^{\mathsf{T}}\phi(x)=\phi(x)^{\mathsf{T}}\theta, \tag{3.4.7}$$

where $\theta=[\theta_1,\cdots,\theta_N]^{\mathsf{T}}\in\Re^N$ and $\phi=[\phi_1,\cdots,\phi_N]^{\mathsf{T}}:\Re^n\rightarrow\Re^N$. The Lagrange interpolation condition can be expressed as the problem of finding θ such that

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}=\begin{bmatrix} \phi(x_1)^{\mathsf{T}} \\ \vdots \\ \phi(x_m)^{\mathsf{T}} \end{bmatrix}\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \end{bmatrix} \tag{3.4.8}$$

$$Y=\Phi^{\mathsf{T}}\theta \tag{3.4.9}$$

Note that $\Phi = [\phi(x_1), \ldots, \phi(x_m)] \in R^{N \times m}$. The matrix Φ^T is referred to as the interpolation or collocation matrix. Much of the function approximation and interpolation literature focuses on the case where $n = 1$. When $n > 1$ and the data points are not defined on a grid, the problem is referred to as scattered data interpolation.

A necessary condition for interpolation to be possible is that $N \geq m$. In online applications, where m is unbounded (i.e., $x_k = x(kT)$), interpolation would eventually lead to both memory and computational problems.

If $N = m$ and Φ is nonsingular, the unique interpolated solution is

$$\theta = (\Phi^\mathsf{T})^{-1} Y \tag{3.4.10}$$

When the basis set $\{\phi_j(x)\}_{j=1}^N$, has the property that the matrix Φ, is nonsingular for any distinct $\{x_j\}_{j=1}^N$, the linear space spanned by $\{\phi_j(x)\}_{j=1}^N$, is referred to as a Chebyshat space or a Haar space [40]. The issue of how to select ϕ to form a Haar space has been widely studied. Even if the theoretical conditions required for Φ to be invertible are satisfied, if x_i is near x_j for $i \neq j$, then Φ, may be nearly singular. In this case, any measurement error in Y may be magnified in the determination of θ. In addition, the solution via eqn. 3.4.10 may be numerically unstable.

For a unique solution to exist, the number of free parameters (i.e., the dimension of θ) must be exactly equal to the number m of sample points x_i. Therefore, the dimension of the approximator parameter vector must increase linearly with the number of training points. Under these conditions, the number of computations involved in solving eqn. 3.4.10) is on the order of m^3 floating point operations (FLOPS) [40]). In addition to this large computational burden, the condition number of Φ often becomes small as m gets large. As the number of data points m increases, there will eventually be more data (and for $m = N$ more degrees of freedom in the approximator) than degrees of freedom in the underlying function. In typical situations, the data y_i will not be measured perfectly, but will include errors from such effects as sensor measurement noise. The described interpolation solution attempts to fit this noisy data perfectly, which is not usually desirable. Approximators with $N < m$ parameters will be over-constrained (i.e., more constraints than degrees of freedom). In this case, the approximated function can be designed (in an appropriate sense) to attenuate the effects of noisy measurement data. An additional benefit of fixing N (independent of m) is that the computational complexity of the approximation and parameter estimation problems is fixed as a function of N and does not change as more data is accumulated.

3.4.3 Function approximation

The linear in the parameters (LIP) function approximation problem can be stated as: Given a basis set $\phi_i(x) : \Re^n \rightarrow \Re$ for $i = 1, \ldots, N$ and a function $f(x) : \Re^n \rightarrow \Re^1$ find a linear combination of the basis elements $\hat{f}(x) = \theta^\mathsf{T}\phi(x) : \Re^n \rightarrow \Re$ that is close to f. Key problems that arise are:

- How to select the basis set?
- How to measure closeness?
- How to determine the optimal parameter vector for the linear combination?

In the function approximation literature there are various broad classes of function approximation problems. The class of problems that will be of interest herein is the development of approximations to functions based on information related to input-output samples of the function.

Off-line or Batch Function Approximation

Given a set of input-output data $(x_i, y_i), i = 1, \ldots, m$ function approximation is the problem of defining a function $f(\hat{x}) : \Re^n \rightarrow \Re^1$ to minimize $\|\hat{Y} - Y\|$ where $Y = [y_1, \ldots, y_m]^\mathsf{T}$ and $\hat{Y} = [\hat{f}(x_1), \ldots, \hat{f}(x_m)]^\mathsf{T}$. The discussion of the following two sections will focus on the over and under constrained cases.

Over-constrained Solution

Consider the approximator structure of eqn. (3.4.8), which can be represented in matrix form as in eqn. (3.4.9). When $N < m$ the problem is over-specified (more constraints than degrees of freedom). In this case, the matrix Φ defined relative to eqn. (3.4.9) is not square and its inverse does not exist. In this case, there may be no solution to the corresponding interpolation problem. Since with the specified approximation structure the data cannot be fit perfectly, the designer may instead select the approximator parameters to minimize some measure of the function approximation error. If a weighted second-order cost function is specified then

$$J(\theta) = \frac{1}{2}(\hat{Y} - Y)^\mathsf{T} W (\hat{Y} - Y) \tag{3.4.11}$$

which corresponds to the norm $\|Y\|_W^2 = \frac{1}{2}Y^\mathsf{T}WY$ where W is symmetric and positive definite. In this case the optimal vector θ^* that minimizes the cost function in eqn. (3.4.11) can be obtained by differentiation :

$$J(\theta) = \frac{1}{2}(\Phi^\mathsf{T}\theta - Y)^\mathsf{T}W(\Phi^\mathsf{T}\theta - Y) \tag{3.4.12}$$

$$\frac{\partial J(\theta)}{\partial \theta}|_{\theta=\theta^*} = (\Phi^\mathsf{T}\theta^* - Y)^\mathsf{T}W\Phi^\mathsf{T} = 0 \tag{3.4.13}$$

$$\theta^* = (\Phi W \Phi^\mathsf{T})^{-1}\Phi W Y \tag{3.4.14}$$

and

$$\frac{\partial J^2(\theta)}{\partial \theta^2} = \Phi W \Phi^\mathsf{T} \tag{3.4.15}$$

where it has been assumed that $rank(\Phi) = N$ so that $\Phi W \Phi^\mathsf{T}$ is nonsingular. When the rank of $rank(\Phi) < N$, then either additional data are required or the under-constrained approach defined below must be used. Since the second derivative of $J(\theta)$ with respect to θ evaluated at θ^* is, at least positive semi-definite, the solution of eqn. (3.4.14) is a minimum of the cost function [40].

Equation (3.4.14) is the weighted least squares solution. If W is a scalar multiple of the identity matrix, then the standard least squares solution results. Note from eqn. (3.4.13) that the weighted least squares approximation error $(\Phi^\mathsf{T}\theta^* - Y)$ has the property that it is orthogonal to all N columns of the weighted regressor $W\Phi^\mathsf{T}$.

Under-constrained Solution

When $N > m$ the problem is under-specified (i.e., fewer constraints than degrees of freedom). This situation is typical at the initiation of an approximation based control implementation. In this case, the matrix Φ defined in eqn. (3.4.9) is not square and its inverse does not exist. Therefore, there will either be no solution (Y is not in the column space of Φ^T) or an infinite number of solutions. In the latter case, Y is in the column space of Φ^T; however, since the number of columns of Φ^T is larger than the number of rows, the solution is not unique. The minimum norm solution can be found by application of Lagrange multipliers. Define the cost function

$$J(\theta, \lambda) = \frac{1}{2}\theta^\mathsf{T}\theta + \lambda^\mathsf{T}(Y - \Phi^\mathsf{T}\theta) \tag{3.4.16}$$

which enforces the constraint of eqn. (3.4.9) and is minimized by the minimum norm solution. Taking derivatives with respect to θ and λ yields

$$\frac{\partial J}{\partial \theta} = \theta^{\mathsf{T}} - \lambda^{\mathsf{T}} \Phi^{\mathsf{T}} = 0 \qquad \text{and} \qquad \frac{\partial J}{\partial \lambda} = Y - \Phi^{\mathsf{T}} \theta = 0 \tag{3.4.17}$$

Combining these two equations and solving yields

$$\lambda = (\Phi^{\mathsf{T}} \Phi)^{-1} Y \qquad \theta = \Phi(\Phi^{\mathsf{T}} \Phi)^{-1} Y \tag{3.4.18}$$

where $(\Phi^{\mathsf{T}}\Phi)$ an $m \times m$ matrix that is assumed to be nonsingular. The matrix $\Phi(\Phi^{\mathsf{T}}\Phi)^{-1}$ is the Moore-Penrose pseudo-inverse of Φ^{T} [16, 90].

This section has discussed the off-line problem of fitting a function to a fixed batch of data. We have introduced the topic of weighted least squares parameter estimation which is applicable when the number of data points exceeds the number of free parameters defined for the approximator. We have also discussed the under-constrained case when there is not sufficient data available to completely specify the parameters of the approximator. Normally in online control applications, the number of data samples m will eventually be much larger that the number of parameters N. This is true since additional training examples are accumulated at each sampling instant. The results for the under-constrained case are therefore mainly applicable during start-up conditions.

Adaptive Function Approximation

Given the first k samples, with $k \geq N$, the weighted least squares (WLS) parameter estimate obtained in the preceding section can be expressed as

$$\theta_k = (\Phi_k W_k \Phi_k^{\mathsf{T}})^{-1} \Phi_k W_k Y_k$$

where $\Phi_k = [\phi(x_1), \ldots, \phi(x_k)] \in \Re^{N \times k}$, $Y_k = [y_1, \ldots, y_k]^{\mathsf{T}}$, and W_k is an appropriately dimensioned positive definite matrix. Solution of this equation requires inversion of an $N \times N$ matrix. When the $(k+1)$st sample becomes available, this expression requires the availability of all previous training samples and again requires inversion of a new $N \times N$ matrix. For a diagonal weighting matrix W_k, direct implementation of the **WLS** algorithm has storage and computational requirements that increase with k . This is not satisfactory, since k is increasing without bound. To overcome such a problem we may use the recursive weighted least squares (**RWLS**) method presented in the next subsection.

Recursive WLS: Derivation

While the batch least squares approach has proven to be very successful for a variety of applications, the fact that by its very nature it is a batch method (i.e., all the data are gathered, then processing is done) may present computational problems.

The WLS parameter estimate can be expressed as

$$\theta_k = P_k^{-1} R_k, \quad \text{where} \quad P_k = \frac{1}{k}\Phi_k W_k \Phi_k^{\mathsf{T}} \quad \text{and} \quad R_k = \frac{1}{k}\Phi_k W_k Y_k. \tag{3.4.19}$$

In the case where $W_k = I$, P_k is the sample regressor autocorrelation matrix and R_k is the sample cross-correlation matrix between the regressor and the function output.

From the definitions of Φ, Y, and W , assuming that W is a diagonal matrix, we have that

$$Y_{k+1} = \begin{bmatrix} Y_k \\ y_{k+1} \end{bmatrix}, \quad \Phi_{k+1} = \begin{bmatrix} \Phi_k & \phi_{k+1} \end{bmatrix}, \quad \text{and} \quad W_{k+1} = \begin{bmatrix} W_k & 0 \\ 0 & w_{k+1} \end{bmatrix}, \tag{3.4.20}$$

therefore

$$\Phi_{k+1} W_{k+1} \Phi_{k+1}^{\mathsf{T}} = \Phi_k W_k \Phi_k^{\mathsf{T}} + \phi_{k+1} w_{k+1} \phi_{k+1}^{\mathsf{T}} \tag{3.4.21}$$

$$\Phi_{k+1} W_{k+1} Y_{k+1}^{\mathsf{T}} = \Phi_k W_k Y_k^{\mathsf{T}} + \phi_{k+1} w_{k+1} y_{k+1}^{\mathsf{T}} \tag{3.4.22}$$

Calculation of the WLS parameter estimate after the $(k + 1)$st sample is available will require inversion of the $\Phi_{k+1} W_{k+1} \Phi_{k+1}^{\mathsf{T}}$. The Matrix Inversion Lemma [52] will enable derivation of the desired recursive algorithm based on eqn. 3.4.21.

Lemma 3.4.1. *The Matrix Inversion Lemma states that if matrices A, C, and $(A + BCD)$ are invertible (and of appropriate dimension [52]), then $(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$.*

Applying the above Lemma, with $A_k = \Phi_k W_k Y_k^{\mathsf{T}}, B = \phi_{k+1}, C = w_{k+1},$ and $D = \phi_{k+1}^{\mathsf{T}}$, yields

$$A_{k+1}^{-1} = \left(\Phi_k W_k Y_k^{\mathsf{T}} + \phi_{k+1} w_{k+1} \phi_{k+1}^{\mathsf{T}}\right)^{-1}$$

$$A_{k+1}^{-1} = A_k^{-1} + A_k^{-1}\phi_{k+1}\left(\phi_{k+1}^{\mathsf{T}} A_k^{-1} \phi_{k+1} + w_{k+1}^{-1}\right)^{-1} \phi_{k+1}^{\mathsf{T}} A_k^{-1} \tag{3.4.23}$$

Note that the **WLS** estimate after samples k and $(k+1)$ can respectively be expressed as

$$\theta_k = A_k^{-1}\Phi_k W_k Y_k \qquad \text{and} \qquad \theta_{k+1} = A_{k+1}^{-1}\Phi_{k+1} W_{k+1} Y_{k+1}$$

Using eqns. (3.4.21) and (3.4.23), we can easily derive the recursive **WLS** parameter estimation equation as follows:

$$\theta_{k+1} = \theta_k + A_k^{-1}\left(\phi_{k+1}^\top A_k \phi_{k+1} + w_{k+1}^{-1}\right)^{-1}\phi_{k+1}\left(y_{k+1} - \phi_{k+1}^\top \theta_k\right) \tag{3.4.24}$$

Eqn. (3.4.24) has a standard predictor-corrector format

$$\theta_{k+1} = \theta_k + \Omega_k \phi_{k+1}\left(y_{k+1} - y_{k+1:k}\right) \tag{3.4.25}$$

where $\Omega_k = A_k^{-1}\left(\phi_{k+1}^\top A_k \phi_{k+1} + w_{k+1}^{-1}\right)^{-1}$ and $y_{k+1:k} = \phi_{k+1}^\top \theta_k$ is the estimate of y_{k+1} based on θ_k.

The standard **WLS** calculation requires inversion of an $N \times N$ matrix, however using the **RWLS** algorithm reduces the task to an inversion of an $n \times n$ matrix where N is the number of basis functions and n is the output dimension of f , which we have assumed to be one. Therefore, the matrix inversion simplifies to a scalar division. Note that A_k is never required. Therefore, A_k^{-1} is propagated, but never inverted.

Due to the equivalence of the **WLS** and **RWLS** solutions, the **RWLS** estimate will not be the unique solution to the **WLS** cost function until the matrix $\Phi_k W \Phi_k^\top$ is not singular. Various alternative parameter estimation algorithms can be derived which require substantially less memory and fewer computations, the tradeoff is that the alternative algorithms converge asymptotically instead of yielding the optimal parameter estimate as soon as k achieves sufficient excitation. In fact, if convergence of the parameter vector is desired for non-WLS algorithms, then the more stringent condition of persistence of excitation will be required.

Approximator Properties

This section discusses properties that families of function approximators may have. In each subsection, the technical meaning of each property is presented and the relevance and tradeoffs of the property in the applications of interest are discussed. Due to the technical nature of and the broad background that would be required for the proofs, in most cases the proofs are not presented. Literature sources for the proofs are cited.

Parameter (Non)Linearity

An initial decision that the designer must make is the form of the function approximator. A large class of function approximators (several are presented in Chapter §§) can be represented as

$$\hat{f}(x:\theta,\sigma) = \theta^{\mathsf{T}}\phi(x,\sigma) \tag{3.4.26}$$

where $x \in \Re^n, \theta \in \Re^N$, and the dimension of σ depends on the approximator of interest. The approximator has a linear dependence on θ, but a nonlinear dependence on σ

For instance, the radial basis function approximator with Gaussian nodes takes the following form:

$$\hat{f}(x:\theta,\sigma) = \sum_{i=1}^{N} \theta_i \exp\left(-\frac{\|x-c_i\|^2}{\gamma_i^2}\right),$$

with $x, c_i \in \Re^n$ and $\theta_i, \gamma_i \in \Re^1$, has the form of eqn. (3.4.26) where

$$\phi(x,\sigma) = \left[\exp\left(-\frac{\|x-c_1\|^2}{\gamma_1^2}\right),\ldots,\exp\left(-\frac{\|x-c_N\|^2}{\gamma_N^2}\right)\right]^{\mathsf{T}},$$

$$\sigma = \left[c_1,\ldots,c_N,\gamma_1,\ldots,\gamma_N,\right]^{\mathsf{T}},$$

and

$$\theta = \left[\theta_1,\ldots,\theta_N\right]^{\mathsf{T}},$$

This radial basis function approximator is only linear in its parameters when all elements of σ are fixed.

In most papers and applications, the parameter N which is the dimension of ϕ is fixed prior to online usage of the approximator. When N is fixed prior to online operation, selection of its value should be carefully considered as N is one of the key parameters that determines the minimum approximation accuracy that can be achieved [40]. Self-organizing approximators that adjust N online while ensuring stability of the closed-loop control system constitute an area of continuing research. A second key design decision is whether a will be fixed apriori (i.e., $\sigma(t) = \sigma(0) and \dot{\sigma} = 0$) or adapted online (i.e., $\sigma(t)$ is a function of the online data and control performance). If σ is fixed during online operation, then the function approximator is linear in the remaining adjustable parameters θ so that the designer has a linear-in-the-parameter (**LIP**) adaptive function approximation problem. Proving theoretical issues, such as closed-loop system stability, is easier in the **LIP** case. Fixing σ is beneficial in terms of simplifying the analysis and online computations, but may limit the functions that can be accurately approximated and may require that $N = \dim(\phi)$ be larger than would be required if σ were estimated online.

Classical Approximation Results

The set of functions $\mathcal{F}(\mathcal{D})$ defined on a compact set $\mathcal{D}$ is a linear space. The ∞-norm of $\mathcal{F}(\mathcal{D})$ is defined as

$$\|f\|_\infty = \sup_{x \in \mathcal{D}} |f(x)|.$$

The set $\mathcal{C}(\mathcal{D})$ of continuous functions defined on $\mathcal{D}$ is also a linear space of functions. Since D is compact, for $f \in \mathcal{C}(\mathcal{D})$,

$$\|f\|_\infty = \max_{x \in \mathcal{D}} |f(x)|.$$

Since $\sup_{x \in \mathcal{D}} |f(x)|$ satisfies the properties of a norm, both $\mathcal{F}(\mathcal{D})$ and $\mathcal{C}(\mathcal{D})$ are normed linear spaces. Given a norm on $\mathcal{F}(\mathcal{D})$, the distance between $f, g \in \mathcal{F}(\mathcal{D})$ can be defined as $d(f,g) = \|f - g\|$. When f, g are elements of a space $\mathcal{S}$, $d(f,g)$ is a metric for $\mathcal{S}$ and the pair $\mathcal{S}, d$ is referred to as a metric space. When $\mathcal{S}(\mathcal{D})$ is a subset of $\mathcal{F}(\mathcal{D})$, the distance from $f \in \mathcal{F}(\mathcal{D})$ to $\mathcal{S}(\mathcal{D})$ is defined to be $d(f, \mathcal{S}) = \inf_{a \in \mathcal{S}} d(f,a)$.

A sequence $f_i \in \mathcal{X}$ is a Cauchy sequence if $\|f_i - f_j\| \to 0$ as $i, j \to \infty$. A space $\mathcal{X}$ is complete if every Cauchy sequence in $\mathcal{X}$ converges to an element of $\mathcal{X}$ (i.e., $\|f_i - f_j\| \to 0$ as $i \to \infty$ for some $f \in \mathcal{X}$) . A Banach space is the name given to a complete normed linear space. Examples of Banach spaces include the $\mathcal{L}_p$ spaces for $p \geq 1$ where

$$\mathcal{L}_p = \left\{ f : \Re^1 \to \Re^1 : \|f\|_p = \left[\int f(x)|^p dx \right]^{\frac{1}{p}} < \infty \right\}$$

or the set $\mathcal{C}(\mathcal{D})$ with norm $\|f\|_\infty$.

Weierstrass Results

Given a Banach space $\mathcal{X}$ with elements f, norm $|f|$, and a sequence $\Phi_N = \{\phi_i\}_{i=1}^N \subset \mathcal{X}$ of basis elements, f is said to be approximable by linear combinations of Φ_N with respect to the norm $\|.\|$ if for each $\epsilon > 0$ there exists N such that $\|f - P_N\| < \epsilon$ where

$$P_N(x) = \sum_{i=1}^{N} \theta_i \phi_i(x), \quad \text{for some} \quad \theta_i \in \mathcal{R}. \tag{3.4.27}$$

The N-th degree of approximation of f by Φ_N is

$$E_N^{\Phi}(f) = d(f, P_N) = \inf_{\theta} \|f - P_N\|. \tag{3.4.28}$$

When the infinum is attained for some $P \in \mathcal{X}$, this P is referred to as the linear combination of best approximation.

Consider the theoretical problem of approximating a given function $f \in \mathcal{C}(\mathcal{D})$ relative to the two norm using P_N. The solution to eqn. (3.4.7) is

$$\left(\int_{\mathcal{D}} \Phi_N(x)\Phi_N^{\mathsf{T}}(x)dx\right)^{-1} \int_{\mathcal{D}} \Phi_N(x)f(x)dx \tag{3.4.29}$$

where the basis elements $\phi_i(x)$ are assumed to be linearly independent so that

$$\int_{\mathcal{D}} \Phi_N(x)\Phi_N^{\mathsf{T}}(x)dx$$

is nonsingular. This solution shows that there is a unique set of coefficients for each N such that the two-norm of the approximation error is minimized by a linear combination of the basis vectors. This solution does not show that $f \in \mathcal{C}(\mathcal{D})$ is approximable by linear combinations of ϕ_N , since eqn. (3.4.29) does not show whether $E_N^{\Phi}(f)$ approaches zero as N increases.

Universal Approximator

Consider the following theorem

Theorem 3.4.2. *Given $f \in \mathcal{L}_2(\mathcal{D})$ and an approximator of the form eqn. (3.4.7), for any N if $(\int_{\mathcal{D}} \Phi_N(x)\Phi_N^{\mathsf{T}}(x)dx)$ is nonsingular; then there exists a unique $\theta^* \in \mathcal{R}^N$ such that $f(x) = (\theta*)^{\mathsf{T}}\phi(x) + e_f^*$ where*

$$\theta^* = \arg\min_{\theta} \int_{\mathcal{D}} \|f(x) - \hat{f}(x:\theta)\|_2^2 dx. \tag{3.4.30}$$

In addition, there are no local minima of the cost function (other than θ^*).

This theorem states the condition necessary that for a given N , there exists a unique parameter vector θ^* that minimizes the $\mathcal{L}_2$ error over $\mathcal{D}$). In spite of this, for $f \in \mathcal{L}_2(\mathcal{D})$, the error e_f^* may be unbounded pointwise. Since $\mathcal{D}$ is compact, if f and $\Phi_N \in \mathcal{C}(\mathcal{D})$, then e_f^* is uniformly bounded on $\mathcal{D}$), but the theorem does not indicate how e_f^* changes as N increases. This is in contrast to results like the Weierstass theorem which showed polynomials could achieve arbitrary ϵ-accuracy approximation to continuous functions uniformly over a compact region, if the order of the polynomial was large enough. Development of results analogous to the Weierstrass theorem for more general classes of functions is the goal of this section.

For approximation based control applications, a fundamental question is whether a particular family of approximators is capable of providing a close approximation to the function $f(x)$. There are at least three interesting aspects of this question:

1. Is there some subset of a family of approximators that is capable of providing an ϵ-accurate approximation to $f(x)$ uniformly over $\mathcal{D}$.

2. If there exists some subset of the family of approximators that is capable of providing an ϵ-accurate approximation, can the designer specify an approximation structure in this subset apriori?

3. Given that an approximation structure can be specified, can appropriate parameter vectors θ and σ be estimated using data obtained during online system operation, while ensuring stable operation?

The N-th degree approximation error of f by $S_{r,N}$ is,

$$E_N^{\Phi}(f) = \inf_{\theta_i, A_i, i=1..N} \left\| f - \left(\sum_{i=1}^{N} \theta_i g(A_i(x)) \right) \right\|. \tag{3.4.31}$$

Uniform Approximation is concerned with the question of whether for a particular family of approximators and f having certain properties (e.g., continuity), is it guaranteed to be true that for any $\epsilon > 0, E_N^{\Phi}(f) < \epsilon$ if N is large enough? Many such universal approximation results have been published (e.g., [34, 35]).

Theorem 3.4.3. ***(Stone-Weierstrass Theorem)*** *Let S be any algebra of real continuous functions on a compact set $\mathcal{D}$. if S separates points on $\mathcal{D}$ and vanishes at no point of $\mathcal{D}$, then for any $f \in \mathcal{C}(\mathcal{D})$ and $\epsilon > 0$ there exists $\hat{f} \in S$ such that $\sup_{\mathcal{D}} |f(x) - \hat{f}(x)| < \epsilon$.*

Approximators that satisfy theorems such as 3.4.3 are referred to as universal approximators. Universal Approximation Theorems such as these state that under reasonable assumptions on the approximator structure and the function to be approximated, if the (single hidden layer) network approximator has enough nodes, then an accurate network approximation can be constructed by selection of θ and σ. Such theorems do not provide constructive methods for determining appropriate values of N, θ, or σ.

Universal aproximators in control

Universal approximation results are one of the most typically cited reasons for applying neural or fuzzy techniques in control applications involving significant unmodeled nonlinear effects. The dynamics involve a function $f(x) = f_o(x) + f^*(x)$ where $f^*(x)$ has a significant effect on the system performance and is known to have properties satisfying a universal approximation theorem, but $f^*(x)$ cannot be accurately modeled a priori. Based on universal approximation results, the designer knows that there exists some subset of S that approximates $f^*(x)$ to an accuracy ϵ for which the control specification can be achieved Therefore, the approximation based control problem reduces to finding $f \in \mathcal{S}$ that satisfies the ϵ accuracy specification. Most articles in the literature address the third question stated at the beginning of this section: selection of θ or (σ, θ) given that the remaining parameters of $\mathcal{S}$ have been specified. However, selection of N for a given choice of(N, σ) for a specified g) is the step in the design process that limits the approximation accuracy that can ultimately be achieved. To cite universal approximation results as a motivation and then select N as some arbitrary, small number are essentially contradictory.

Starting with the motivation stated in the previous paragraph, it is reasonable to derive stable algorithms for adaptive estimation of θ or (σ, θ) if N is specified large enough. Specification of too small of a value for N defeats the purpose of using a universal approximation based technique. When N is selected too small but a provably stable parameter estimation algorithm is used, stable (even satisfactory) control performance is still achievable; however, accurate approximation will not be achievable. Unfortunately, the parameter m is typically unknown, since $f^*(x)$ is not known. Therefore, the selection of N must be made overly large to ensure accurate approximation. The tradeoff for over estimating the value of N is the larger memory and computation time requirements of the implementation. In addition, if N is selected too large, then the approximator will be capable of fitting the measurement noise as well as the function. Fourier analysis based methods for selecting N are discussed in [77]. Online adjustment of N is an interesting area of research which tries to minimize the computational requirements while minimizing ϵ and ensuring stability [25, 33, 39, 86].

Chapter 4

Approximator structures

4.1 Introduction

The objective of this chapter is to present and discuss several neural, fuzzy, and traditional approximation structures in a unifying framework to be used as a solution to model and compensate for the uncertain terms that are neglected. The presentation will make direct references to the approximator properties presented in Chapter 3. Each section of this chapter discusses one type of function approximator, presents the motivation for the development of the approximator, and shows how the approximator can be represented in one of the standard nonlinearly and linearly parameterized forms:

$$NLIP: \qquad \hat{f}(x:\theta,\sigma) = \theta^{\mathrm{T}}\phi(x,\sigma) \tag{4.1.1}$$

$$LIP: \qquad \hat{f}(x:\theta) = \theta^{\mathrm{T}}\phi(x) \tag{4.1.2}$$

where $x \in \mathcal{D} \subset \mathbb{R}^n, \sigma \in \mathbb{R}^p, \hat{f} : \mathcal{D} \to \mathbb{R}^1$, and $\mathcal{D}$ is assumed to be compact. Note that $\hat{f}$ is assumed to map a subset of $\mathbb{R}^n$ onto $\mathbb{R}^1$. This assumption that we are only concerned with scalar functions (i.e., single output) is made only for simplicity of notation. All the results extend to vector functions.

The ultimate objective is to adjust the approximator parameters θ and σ to encode information that will enable better control performance. Proper design requires selection of a family of function approximators, specification of the structure of the approximator, and estimation of appropriate approximator parameters. The latter process is referred to as parameter adaptation or learning.

4.2 Model Types

This section discusses three approaches to adaptive approximation. The first subsection discusses the use of a model structure derived from physical principles. The second subsection discusses the storage and use of the raw data without the intermediate step of function approximation. The third section, which we will focus on, discusses the use of generic function approximators.

4.2.1 Physically Based Models

In some applications, the physics of the problem will provide a well-defined model structure where only parameters with well-defined physical interpretations are unknown. In such cases, the physically defined model may provide a structure appropriate for adaptive parameter identification.

When the physics of the problem provides a well-defined model structure, parameter estimation based on that model is often the most appropriate approach to pursue. However, even these applications must be designed with care to ensure stable operation and meaningful parameter estimates.

Alternatively, the physics of an application will often provide a model structure, but leave certain functions within the structure ill-defined. In these applications, adaptive approximation based approaches may be of interest.

4.2.2 Structure (Model) Free Approximation

In applications where adaptive function approximation is of interest, the data necessary to perform the functions approximation will be supplied by the application itself and could arrive in a variety of formats. The easiest form of data to work with is samples of the input and output of the function to be approximated. Although this is often an unrealistic assumption, for this section we will assume availability of a set of data $\{z_i\}_{i=1}^{m}$, where each vector z_i, can be decomposed as $z_i = [x_i \quad f(x_i)]$ with x_i, being the function inputs and $f(x_i)$ being the function outputs. This set of data can be directly stored without further processing. This is essentially a database approach. If the function value is required at x_j for some $1 \leq j \leq m$, then its value can be retrieved from the database. Note that there is no noise attenuation. However, in control applications, the chance of getting exactly the same evaluation point in the future as one of the sample points from the past is very small. Therefore, the exact input matching requirement would render the database useless.

Many extensions of the database type of approach are available to generate estimates of the functions values at evaluation points $x \notin \{x_i\}_{i=1}^m$ [83]. In such approaches, the sample points $\{x_i\}_{i=1}^m$ affect the estimate of $f(x)$ at points $x \notin \{x_i\}_{i=1}^m$. Therefore, all such approaches cause generalization (appropriately or not) from the training data. If the function samples at several of the x_i are combined to produce the estimate of $f(x)$, then noise on individual samples might be attenuated. When the designer does not have prior knowledge of a parametric description of function, then the basic function approximation problem is nonparametric. A complete description of an arbitrary function could require an infinite number of parameters, which is clearly not physically possible.

4.2.3 Function Approximation Structures

The design philosophy should be to use as much known information as is possible when constructing the dynamic model; however, when portions of a physically based model are either not accurately known or deemed inappropriate for online applications, then it is reasonable to use function approximation structures capable of approximating wide classes of functions. To make this point explicit, we will use the notation $f(z) = f_o(z) + f^*(x)$ to describe a partially known function f . In this notation, f_o is the known information about f and f^* represents its unknown portion. When there is no prior known information, the function f_o is set to zero.

Basic descriptions and properties of specific function approximation structures are discussed in the remaining sections of this chapter. Note that the choice of a family of approximators and the structure of a particular approximator is based on the implicit assumption by the designer that the selected approximator structure is sufficient for the application. Subsequent adaptive function approximation is constrained to the functions that can be implemented only by adjusting the parameters of the (now) fixed approximation structure.

Once the approximation structure and the compact region of approximation $\mathcal{D}$ are fixed, we can define an optimal parameter vector, a parameter error vector, and the residual approximation error for structures defined in (4.1.1,4.1.2) as

$$NLIP: \qquad (\theta^*, \sigma^*) = \arg\min_{\theta} \left(\sup_{x \in \mathcal{D}} \left| f^*(x) - \hat{f}(x : \theta, \sigma) \right| \right) \tag{4.2.1}$$

$$LIP: \qquad \theta^* = \arg\min_{\theta} \left(\sup_{x \in \mathcal{D}} \left| f^*(x) - \hat{f}(x:\theta) \right| \right) \tag{4.2.2}$$

Given these definitions of the optimal parameters, the parameter error vector for LIP approximators is defined by

$$\tilde{\theta} = \theta - \theta^* \tag{4.2.3}$$

For **NLIP** approximators, in addition of $\tilde{theta}$, we also define

$$\tilde{\sigma} = \sigma - \sigma^* \tag{4.2.4}$$

The residual or inherent approximation error (for the specified approximation structure) is defined as

$$NLIP: \qquad e^*(x) = \hat{f}(x:\theta^*, \sigma^*) - f^*(x) \tag{4.2.5}$$

$$LIP: \qquad e^*(x) = \hat{f}(x:\theta^*) - f^*(x) \tag{4.2.6}$$

and will also sometimes be referred to as the Minimum Functional Approximation Error (**MFAE**).

Note that none of $\theta^*, \sigma^*, \tilde{\theta}, \tilde{\sigma}$, or $e^*(x)$ are known. These are theoretical quantities that are necessary for analysis, but they cannot be used in implementation equations. When $f \in \mathcal{C}(\mathcal{D})$ with $\mathcal{D}$ compact, then the quantities θ^* and $\sup_{x\in\mathcal{D}} |e^*(x)|$ are easily shown to be bounded.

4.3 Splines

Polynomials with low order are a good class of approximators when the region of approximation is sufficiently small relative to the rate of change of the function f. This motivates the idea of subdividing a large region and using a low order approximating polynomial on each of the resulting subregions.

Numeric splines implement this idea by connecting in a continuous fashion a set of local, low order, piecewise polynomial functions to fit a function over a region $\mathcal{D}$. For example, given a set of data $\{(x_i, y_i)\}_{i=1}^{K+1}$ with $x_i < x_{i+1}$, if the data are drawn on a standard $x - y$ graph and connected with straight lines, this would be a spline of order two interpolation of the data set. If the data were connected using 2nd order polynomials between the data points in such a way that the graph had a continuous first derivative at these interconnection points, this would be a spline of order three. The name *"spline"* comes from drafting where flexible strips were used to aid the drafter to interpolate smoothly between points on paper [14].

4.3.1 Description

Various types of splines now exist in the literature. The types of splines differ in the properties that they are designed to optimize and in their implementation methods. In the following we will present two types of splines that are used by control designers in the framework of Function Approximation Based Control.

4.3.2 Natural Splines

In one dimension, a spline is constructed by subdividing the interval of approximation $I = (\underline{x}, \bar{x})$ into K subintervals $I_j = (x_j, x_{j+l})$ where the x_j, referred to as knots or break points, are assumed to be ordered such that $\underline{x} = x_0 < x_1 < \cdots < x_K = \bar{x}$. For a spline of order k , a $(k-1)$st order polynomial is defined on each subinterval I_j . Without additional constraints, each $(k-1)$st order polynomial has k free parameters for a total of Kk free spline parameters. The spline functions are, however, usually defined so that the approximation is in $\mathcal{C}^{(k-2)}$ over the interior of I. For example, a 2nd order spline is composed of first order polynomials (i.e., lines) defined so that the approximation is continuous over I including at the knots. With such continuity constraints, the spline has $Kk - (k-1)(K-1) = K + k - 1$ free parameters. With the constraint that the spline be continuous in $(k-2)$ derivatives, splines have the property of being continuous in as many derivatives as is possible without the spline degenerating into a single polynomial. In contrast to polynomial series approximation, the accuracy of a spline approximation can be improved by increasing either k or K . Therefore, splines approximations are more flexible than polynomials series approximators.

4.3.3 Cardinal B-splines

When the B-splines are defined with the knots at

$$\{\cdots, -2, -1, 0, 1, 2, \cdots\}, \tag{4.3.1}$$

they are called Cardinal B-splines. One of the common forms in which B-splines are used in adaptive approximation applications is by translation and dilation of the Cardinal B-splines.

Definition 4.3.1. [28] The functions $g_k : \mathbb{R}^1 \to \mathbb{R}^1$ defined recursively, for $k > 1$, by

$$g_k(x) = \int_0^1 g_{k-1}(x-\lambda)d\lambda = \int_{-\infty}^{\infty} g_{k-1}(x-\lambda)g_1(\lambda)d\lambda \tag{4.3.2}$$

is the Cardinal B-spline of order k (for the knot at 0) where

$$g_1(x) = \begin{cases} 1, & \text{if} \quad 0 \leq x < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{4.3.3}$$

The Cardinal B-splines of orders 2 and 3 are, respectively, for the knot at 0 given by

$$g_2(x) = \begin{cases} x & \text{for} \quad 0 \leq x < 1 \\ 2-x & \text{for} \quad 1 \leq x < 2 \\ 0 & \text{otherwise,} \end{cases} \tag{4.3.4}$$

$$g_3(x) = \begin{cases} \frac{x^2}{2} & \text{for} \quad 0 \leq x < 1 \\ -x^2+3x-\frac{3}{2} & \text{for} \quad 1 \leq x < 2 \\ \frac{(3-x)^2}{2} & \text{for} \quad 2 \leq x < 3 \\ 0 & \text{otherwise.} \end{cases} \tag{4.3.5}$$

Note that the Cardinal B-spline of order k is a piecewise polynomial of degree $k-1$. The piecewise polynomial is in $\mathcal{C}^{(k-2)}$ with points of discontinuity in the $(k-1)$ derivative at $x = 0, 1, 2, \cdots, k$.

The B-spline basis element of order k for the knot at $x = j$ is $g_{kj}(x) = g_k(x-j)$ and has support for $j \in (j, k+j)$. Conversely, for $x \in [0,1]$, the functions $g_k(x-j)$ are nonzero for $j \in [1-k, 0)$. The B-splines basis elements of order $k = 1, 2, 3$, and 4 are shown in Figure 4.1. This figure shows all the B-splines g_{kj} for $j = 1-k, \cdots, 0$ that would be necessary to form a partition of unity for $x \in (0,1)$.

The function $s_k(x) = \sum_{j=1-k}^{N-1} \theta_j g_k(x-j)$ is a spline of order k with $(N+k-1)$ knots at $x = 1-k, 1, 2, \cdots, N-1$. It is also a piecewise polynomial of degree $k-1$ with the same continuity properties as g_k. The function $s_k(x)$ is nonzero on $[l-k, k+N-1]$. For $N > k$, the set of basis elements $\{g_k(x-j)\}_{j=1-k}^{N-1}$ form a partition of unity on $[0, N]$. If instead, the basis elements are selected as

$$\phi_j(x) = g_k\left(N\frac{x-a}{x-b} - j\right), \tag{4.3.6}$$

for $j = 1-k, \cdots, N-1$, then this basis set $\{\phi_j\}_{j=1-k}^{N-1}$, formed by translating and dilating the k-th Cardinal B-spline, is a partition of unity on $[l, b]$. The span of this set of basis functions is a piecewise polynomial of degree $k-1$ that is in $\mathcal{C}^{(k-2)}$. By using an approximator defined as

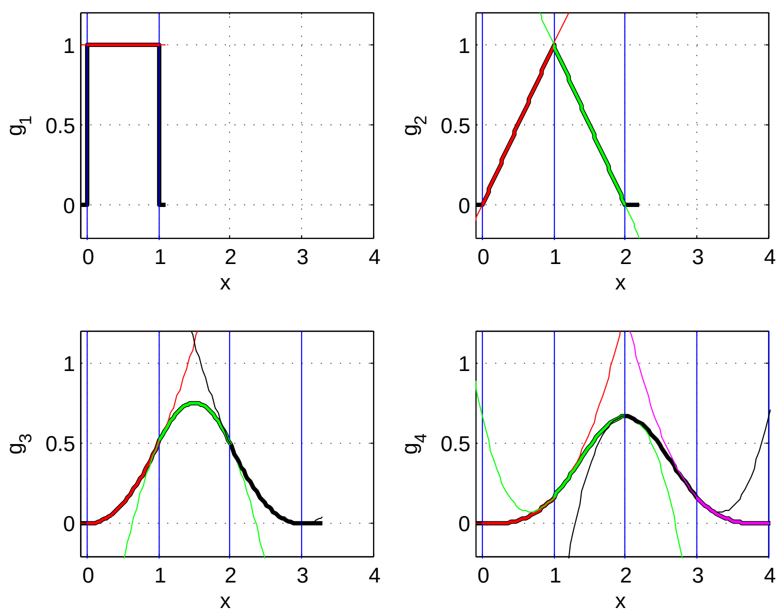

Figure 4.1: B-splines of order 1 thru 4 that are non-zero on $(0, l)$

$$\hat{f}(x,\theta) = \theta^{\mathrm{T}}\phi(x) = \sum_{j=1-k}^{N-1} \theta_j \phi_j(x),$$

with $\phi_j(x)$ as defined in (4.3.6), we are able to adjust the parameters of the approximator without the explicit inclusion of continuity constraints in the parameter adjustment process, such as those that were required for the natural splines. We attain a piecewise polynomial of degree $k-1$ in $\mathcal{C}^{(k-2)}$ because the basis elements have been selected to have these properties.

Splines have the uniform approximation property in the sense that any continuous function on a compact set can be approximated with arbitrary accuracy by decreasing the spacing between knots, which increases the number of basis elements. For nonuniformly spaced knots, if it is desired to add additional knots, there are available methods that can be found by searching for "knot insertion". B-splines are locally supported, positive, normalized, and form a partition of unity [13]. Each basis element is nonzero over the k intervals defined by the knots. Therefore, a change in the parameter

θ affects only the approximation over the k intervals of its support. In addition, at any evaluation point, at most k of the basis elements are nonzero

4.4 Artificial Neural Networks

ANNs model biological neurons in order to do numerical interpolation. Figure 4.2 provides a sketch of a biological neuron and a human-made neuron.

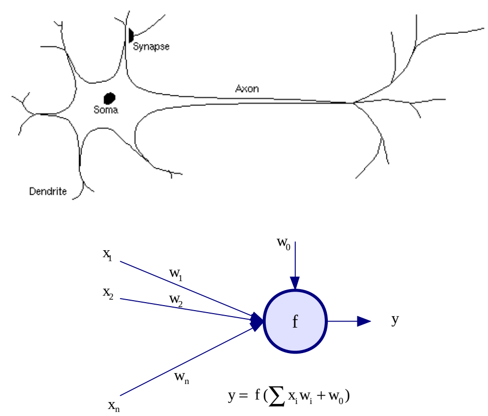

Figure 4.2: Biological Neuron and Artificial Neuron

The biological neuron acts as a processing element that receives many signals. These signals may be modified by a weight at the receiving synapse. Then the processing element sums the weighted inputs. When the input becomes sufficiently large, the neuron transmits a single output that goes off to other neurons.The human-made neuron works by analogy. It takes an input, multiplies it by a weight, adds a bias, and then passes the result through a transfer function. Several neurons in parallel are known as a layer. Adding these layers together produces the neural network. The weights and bias values are optimized to produce the desired output.

4.4.1 Radial basis functions

Radial basis functions (RBFs) were originally introduced as a solution method for batch multi-variable scattered data interpolation problems [40, 45, 63, 67, 68]. Scattered data interpolation problems are the subset of interpolation problems, where the data samples are dictated not by some optimal criteria, but by the application or experimental conditions. Online control applications involve (non-batch) scattered data function approximation.

Description

A radial basis neural network (RBNN) is typically comprised of a layer of radial basis activation functions with an associated Euclidean input mapping (but there are many ways to define this class of neural networks). The output is then taken as a linear activation function with an inner product or weighted average input mapping.

A radial basis function approximator is defined as

$$\hat{f}(x) = \sum_{i=1}^{N} \theta_i g(\|x - c_i\|) + \sum_{i=1}^{\bar{k}} b_i p_i(x) \tag{4.4.1}$$

where $x \in \mathbb{R}^n, \{c_i\}_{i=1}^N$ are a set of center locations, $\|x - c_i\|$ is the distance from the evaluation point to the i-th center, $g(.) : \mathbb{R}^+ \mapsto \mathbb{R}^1$ is a radial function, and $\{p_i(x)\}_{i=1:\bar{k}}$ is a basis for the $\bar{k}$ dimensional linear space of polynomials of degree k in n variables

$$P_k = span\{x_1^{\alpha 1}, \ldots, x_n^{\alpha n} | 0 \le \alpha_1 + \cdots + \alpha_n \le k\}.$$

The polynomial term in (4.4.1) is included so that the RBF approximator will have polynomial precision k [1]. Often in RBF applications, k is specified by the designer to be -1. In that case, the polynomial term does not appear in the approximator structure and the RBF does not have a guaranteed polynomial precision.

Some forms of the radial function that appear in the literature are

$$\text{Gaussian :} \qquad g_1(\rho) = \exp\{-\frac{1}{2}\frac{\rho^2}{\gamma^2}\}$$

$$\text{Multi-Quadraric :} \qquad g_2(\rho) = \left(\rho^2 + \gamma^2\right)^{\beta}, \quad \beta \in (0,1)$$

$$\text{Inverse Multi-Quadraric :} \qquad g_3(\rho) = \left(\rho^2 + \gamma^2\right)^{-\alpha}, \quad \alpha > 0$$

$$\text{Thin Plate Spline :} \qquad g_4(\rho) = \rho^2 \log(\rho + \gamma)$$

[1] An approximator having polynomial precision k means that the approximator is capable of exactly representing a polynomial of that order.

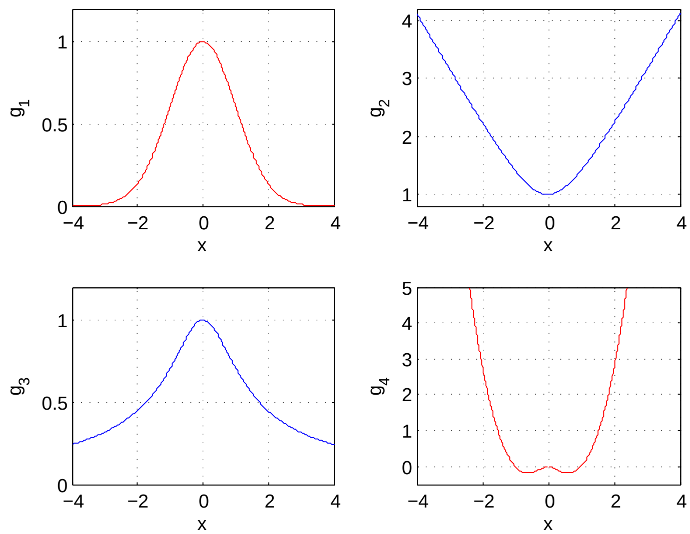

Figure 4.3: RBF Nonlinear functions

where$\rho \in [0, \infty)$ and γ is a constant either defined by the designer prior to online application or a parameter to be estimated online, The multi-quadratic and inverse multi-quadratic are stated for specific ranges of β and α , but the names of the nodal functions relate explicitly to the case where $\alpha = \beta = 0.5$. Multi-quadratics were introduced by Hardy in 1971 [40]. Figure 4.3 displays plots of four radial functions with $\alpha = \beta = 0.5$ and $\gamma = 1$. Radial basis functions (with $\hat{k} = 0$) can be represented in the standard form

$$\hat{f}(x) = \theta^{\mathrm{T}}\phi(x, c, \gamma) = \sum_{i=1}^{N} \theta_i \phi_i(x, c, \gamma) \tag{4.4.2}$$

where the i-th basis element is defined by

$$\phi_i(x, c, \gamma) = g(\|x - c_i\|, \gamma) \quad \text{for} x \in \mathbb{R}^n \quad \text{and} i = 1, \cdots, m. \tag{4.4.3}$$

In the standard RBF, all the elements of ϕ are based on the same radial function $g(.)$. The first argument of g is the radial distance from the input x to the i-th center $c_i, \rho_i(x) = \|c_i - x\|$. When g is selected to be either the Gaussian function or the Inverse multi-quadratic, then the resulting basis function approximator will have localization properties determined by the parameter γ. In a more general approach, different values of γ can be used in different basis elements of the RBF approach.

Properties

Given a constant value for γ, the user can choose among the following three procedures for the selection of the centers c_i:

1. For a fixed batch of sample data $\{(x_j, y_j)\}_{j=1}^N$, when the objective is interpolation of the data set, the centers are equated to the locations of the sample data: $c_i = x_i$ for $i = 1, \cdots, N$. Data interpolation for $j = 1, \cdots, N$ provides a set of N constraints

$$y_j = \sum_{i=1}^{N} \theta_i g(\|x_j - c_i\|) + \sum_{i=1}^{\bar{k}} b_i p_i(x_j) \tag{4.4.4}$$

leaving $\bar{k}$ degrees of freedom. These interpolation constraints can be written as

$$Y = \begin{bmatrix} \Phi^{\mathrm{T}} & P^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \theta \\ b \end{bmatrix} \tag{4.4.5}$$

where Φ and Y are defined in (3.4.9). $P = [p(x_1), \cdots, p(x_N)], p(x_j) = [p_1(x_j), \cdots, p_{\bar{k}}(x_j)]$, and $b = [b_l, \cdots, b_{\bar{k}}]$. Since g is a radial function, $g(\|x_j - x_i\|) = \phi_j(x_j)$; therefore, the matrix Φ is symmetric. The RBF approximator still allows an additional $\bar{k}$ degrees of freedom. The additional constraint that $\sum_{i=1}^{N} \theta_i p_j(x_i) = 0$ for $j = 1, \cdots, \bar{k}$ is typically imposed. The resulting linear set of equations that must be solved for θ and b is

$$\begin{bmatrix} Y \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi^{\mathrm{T}} & P^{\mathrm{T}} \\ P & 0 \end{bmatrix} \begin{bmatrix} \theta \\ b \end{bmatrix} \tag{4.4.6}$$

This is a fully determined set of $N + \bar{k}$ equations with the same number of unknowns. It can be shown that when g is appropriately selected, this set of equations is well-posed [52].

2. The c_i are specified on a lattice covering $\mathcal{D}$. Such specification results in a **LIP** approximation problem with memory requirements that grow exponentially with the dimension of x, but very efficient computation. Function Approximation Theorem presented in the preceding chapter shows that this type of RBF is a universal approximator.

3. The c_i are estimated online as training data are accumulated. This results in a **NLIP** approximation problem. Function Approximation Theorem shows that this type of RBF is also a universal approximator. The resulting approximator may have fewer parameters than case 2, but the approach must address the difficulties inherent in nonlinear parameter estimation.

Although our main interest will be in approximation problems, the use of **RBF**s for data interpolation has an interesting history. The analysis of the interpolation has implications for the choice of g in approximation problems.

4.4.2 Multi-Layer Perceptron

Perceptrons and multilayer perceptron networks [35] have a long history and an extensive literature [68]. Examples of the use of multilayer perceptrons in control application can be found in [5, 11, 19, 20, 21, 22, 23, 29, 43, 54, 58, 61, 63, 92].

Description

The left image in Figure 4.4 illustrates a perceptron. The output of the perceptron denoted by υ_i is

$$\upsilon_i = g\left(b_i + \sum_{j=1}^{n} \omega_{ij} x_j\right). \tag{4.4.7}$$

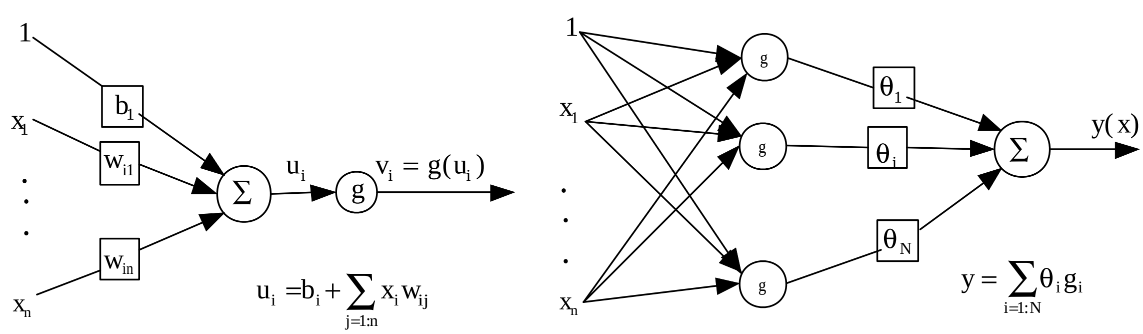

Figure 4.4: Left-Single node perceptron. Right-Single layer perceptron network. The bold lines in the right figure represents the dot product operation (weighting and summing) performed by the connection and nodal processor.

Often for convenience of notation, this will be written as

$$\upsilon_i = g(W_i x) \tag{4.4.8}$$

where $W_i = [b_i, w_{i1}, \cdots, w_{n1}]$ and $x = [1, x_l, \cdots, x_n]$. The function $g : \mathbb{R}^1 \mapsto \mathbb{R}^1$ is a squashing function such as $g(x) = \arctan(x)$ or $g(x) = \frac{1}{1+e^{-x}}$. Note that the perceptron has multiple inputs and a single output. If $g(a)$ is the signum function, then a perceptron divides its input space into two halves using the hyperplane $u_i = W_i x$. If $u_i < 0$, then $v_i = -1$. If $u_i > 0$, then $v_i = 1$. If $u_i = 0$, then $v_i = 0$. This hyperplane is referred to as a linear discriminant function. The image on the right side of Figure 4.4 shows a network that forms a linear combination of perceptron outputs. The network output is

$$y = \theta V \tag{4.4.9}$$

where $V^{\mathrm{T}} = [v_l, \cdots, v_N]$ is the vector of outputs from each perceptron defined in eqn. (4.4.7) and $\theta \in \mathbb{R}^{q \times N}$ is a parameter matrix. This approximator is referred to as a single hidden layer perceptron network. The parameters in W_i, are the hidden layer parameters. The parameters in θ are the output layer parameters. By Theorem 3.4.2, single hidden layer perceptron networks are universal approximators. In the case that y is a scalar (i.e., $q = l$), the function $g(y)$ with g being a signum function defines a general discriminator function that can be used for classification tasks [51].

If desired, networks with multiple hidden layers can be constructed. This is accomplished by defining θ to be a matrix so that y is a vector. If we define $z = \Lambda g(y)$, then the network has two hidden layers defined by the weights in W and θ. The perceptron networks defined above belong to the so-called feedforward networks in witch the information flow through the network is unidirectional (from left to right). There is not feedback of information either from internal variables or from the network output to the network input. In the case where some of the internal network variables or outputs are fed back to serve as a portion of the input, we would have a recurrent network. In this case, the network is a dynamics system with its own state vector. When such recurrent networks are used, the designer must be concerned with the stability of this network state.

Perceptron networks are sometimes referred to as supervised learning or backpropagation networks, but neither of these names are accurate. Supervised learning refer; to the approach of training (i.e., adjusting the parameters) a function approximator $y = \hat{f}(x, \theta, \sigma)$ so that the approximator matches, as closely as possible, a given set of training data, composed of input samples along with their targets, described as $\{(x^i, y^i)\}_{i=1}^{p}$, In this scenario a batch of training samples is available for which the desired output y^i is known for each input x^i. Many early applications of perceptron networks were formulated within the supervised learning approach; however, any function approximator can be trained using such a supervised learning scenario. Therefore, referring to a perceptron network as a supervised learning network is not a clear description.

The backpropagation algorithm was derived for perceptron networks, see e.g. [35], that algorithm is based on the idea of gradient descent. Gradient descent parameter adaptation can be derived for any feed forward network that uses a continuous nodal processor, see e.g. [75]. Therefore, referring to a perceptron network as a backpropagation network is again not a clear description of the network. In addition, the fact that a multilayer perceptron network can be trained using backpropagation is not a motivation for using these networks, since gradient descent training is a general procedure that can be used for many families of approximators.

Properties

The literature on neural networks contains several standard phrases that are often used to motivate the use of perceptron networks. For any particular applications, the applicability of these standard phrases should be carefully evaluated. A typically stated motivation is that perceptron networks are universal approximators. As discussed earlier, numerous families of approximators have this or related properties. Therefore, the fact that perceptron networks are universal approximators is not, by itself, a motivation for using them instead of any other approximator with this property. A perceptron network with adjustable hidden layer parameters is nonlinearly parameterized. Therefore, another stated motivation is that perceptron networks have certain beneficial "order of approximation" properties. On the other hand, there are no engineering procedures available for defining a suitable network structure (i.e., number of hidden layers, number of nodes per layer, etc.) even in situations where the function f to be approximated is known. Also, since the network is nonlinearly parameterized, the initial choice of parameters may not be in the basin of attraction of the optimal parameters. Early in the history of neural networks, it was noticed that perceptron networks "offer the inherent potential for parallel computation". However, any approximation structure that can be written in vector product form is suitable for parallel implementation on suitable hardware. Interesting questions are whether any particular application is worth special hardware, or more generally, is any particular approximation structure worth additional research funding to develop special purpose implementation hardware, when hardware optimized for performing matrix vector products already exists.

Another frequently stated motivation is the idea that perceptron networks are models of the nervous systems of biological entities. Since these biological entities can learn to perform complex control actions (balancing, walking, dancing, etc.), perceptron networks should be similarly trainable. There are several directions from which such statements should be considered. First, is a perceptron network a sufficiently accurate model of a nervous system that such an analogy is justified? Second, is the implemented perceptron network comparable in size to a realistic nervous system? Even if those questions could be answered affirmatively, do we understand and can we accurately replicate the feedback and training process that occur in the biological exemplars? Also, the biological nervous system may be optimized for the biochemical environment in which it operates. The optimal implementation approach on an electronic processing unit could be significantly different.

Another frequent motivation for perceptron networks is by analogy to biological control systems. It is stated that biological control systems are semi-fault tolerant because they rely on large numbers of redundant and highly interconnected nonlinear nodal processors and communication pathways. This is referred to as distributed information processing. To motivate perceptron networks, it is argued that highly interconnected perceptron networks have similar properties to biological control systems, since for perceptron networks the approximator information is stored across a large number of "connection" parameters. The idea being that if a few "connections" were damaged, then some information would be retained via the undamaged parameters and these undamaged parameters could be adapted to reattain the prior level of performance. However, the perceptron networks that are typically implemented are much smaller and simpler than such biological systems, resulting in a weak analogy. In addition, this line of reasoning neglects the fact that perceptron networks are typically implemented with a standard CPU and RAM, where there is no "distributed network implementation" since these standard items fail as a unit. Therefore, the CPU and RAM implementation is not currently analogous to a biochemical network implementation.

4.5 Fuzzy Approximation

This section presents the basic concepts necessary for the designer to be able to construct a fuzzy logic controller. For a detailed presentation of the motivation and theory of fuzzy logic, you may consult, for example, [48, 94, 95]. Detailed discussion of the use of fuzzy logic in fixed and adaptive controllers is presented, for example, in [9, 17, 27, 81].

4.5.1 Description

The four basic components of a fuzzy controller are shown in Figure 4.5. In this figure, over-lined quantities represent fuzzy variables and sets while crisp (real valued) variables and sets have no over-lining.

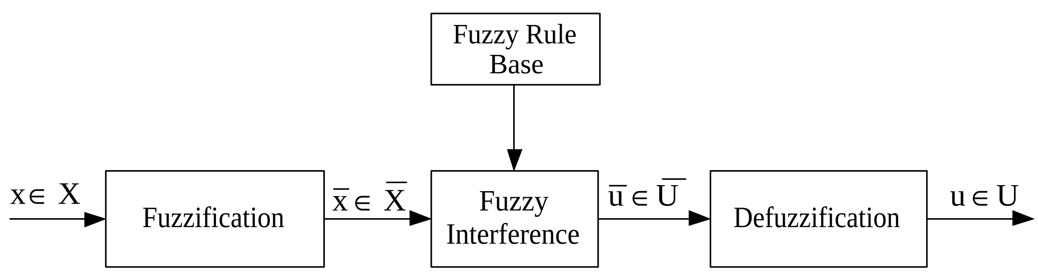

Figure 4.5: Components of a Fuzzy Logic Controller.

4.5.2 Fuzzy Sets and Fuzzy Logic

Given a real valued vector variable $x = [x_l, \cdots, x_n]^{\mathrm{T}}$ that is an element of a domain $X = X_1 \times X_2 \times \cdots \times X_n$, the region X_i is referred to as the *universe of discourse* of x_i and X as the universe of discourse of x. The linguistic variable $\bar{x}_i$ can assume the linguistic values defined by $\bar{X}_i = \left\{ \bar{X}_i^1, \cdots, \bar{X}_i^{N_i} \right\}$. The degree to which the linguistic variable $\bar{x}_i$ is described by the linguistic value $\bar{X}_i^j$ is defined by a membership function $\mu_{\bar{X}_i^j}(x) : X_i \mapsto [0, 1]$. Common membership functions include triangular and Gaussian functions. The fuzzy set $\tilde{X}_i^j$; associated with linguistic variable $\bar{x}_i$, universe of discourse X_i , linguistic value $\bar{X}_i^j$, and membership function $\mu_{\bar{X}_i^j}(x)$

$$\tilde{X}_i^j = \left\{ \left(x_i, \mu_{\bar{X}_i^j}(x_i) \right) | x_i \in X_i \right\} \tag{4.5.1}$$

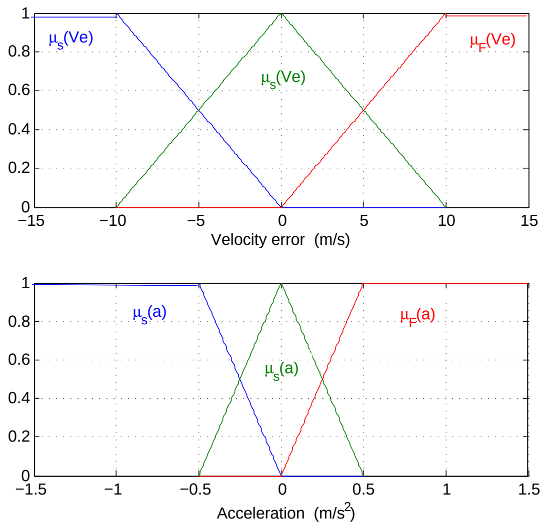

Figure 4.6: Membership functions for speed error and acceleration for the cruise control example

Note that fuzzy sets have members and degrees of membership. The degree of membership is the main feature that distinguishes fuzzy logic from Boolean logic. The support of a fuzzy set $\tilde{F}$ on universe of discourse X is defined as $Supp(\hat{F}) = \{x \in X | \mu_{\hat{F}(x)} \neq 0\}$. If the $Supp(\hat{F})$ is a single point x_s, and $\mu_{\hat{F}(x_s)} = 1$, then x_s is called a fuzzy singleton.

To illustrate the concepts of the previous paragraph, consider a vehicle cruise control application. Let the physical variables be $x = [v_e, a]^{\mathrm{T}}$, where $v_e = v - v_c$, v denotes the actual speed, v_c denotes the commanded speed, and a denotes acceleration. The linguistic variables are defined as $\bar{x} = [speederror, acceleation]^{\mathrm{T}}$.

The linguistic values for each linguistic variable could be defined as

$$\bar{X}_1 = \{Slow, Correct, Fast\}$$

$$\bar{X}_2 = \{Negative, Zero, Positive\}$$

so that $N_1 = N_2 = 3$. Then, the space $\bar{X}$ is defined as

$$\bar{X} = \bar{X}_1 \times \bar{X}_2 = \left\{ \begin{array}{ccc} SN & CN & FN \\ SZ & CZ & FZ \\ SP & CP & FP \end{array} \right\}$$

where each linguistic value has been represented by its first letter. If the universe of discourse is $X = [-15, 15] \times [-2, 2]$, then one possible definition of the membership functions for $\bar{X}_1$ and $\bar{X}_2$ are shown in Figure 4.6.

In fuzzy logic, the $(\bar{A} or \bar{B})$ operation is represented as $(\bar{A} \cup \bar{B})$ and its membership function is calculated by a s-norm operation [95] denoted by $\oplus, \mu_{\bar{A}\cup\bar{B}} = \mu_{\bar{A}} \oplus \mu_{\bar{B}}$. Whereas the $(\bar{A} and \bar{B})$ operation is denoted as $(\bar{A} \cap \bar{B})$ and its corresponding membership function is calculated by a t-norm operation denoted by $\star, \mu_{\bar{A}\cap\bar{B}} = \mu_{\bar{A}} \star \mu_{\bar{B}}$.

Table 4.1 contains several of the possible implementations of the $\star$ and $\oplus$ operations. The membership function for the complement of fuzzy set $\bar{A}$ is $\mu_{\bar{A}^c}(x) = 1 - \mu_{\bar{A}}(x)$. The fuzzy complement is used to implement the "not" operation.

OR	$\mu_{\bar{A}\cup\bar{B}}(x) = \mu_{\bar{A}}(x) \oplus \mu_{\bar{B}}(x)$	AND	$\mu_{\bar{A}\cap\bar{B}}(x) = \mu_{\bar{A}}(x) \star \mu_{\bar{B}}(x)$
Maximum	$\max(\mu_{\bar{A}}(x), \mu_{\bar{B}}(x))$	Minimum	$\max(\mu_{\bar{A}}(x), \mu_{\bar{B}}(x))$
Algebraic Sum	$\mu_{\bar{A}}(x) + \mu_{\bar{B}}(x) - \mu_{\bar{A}}(x)\mu_{\bar{B}}(x)$	Algebraic Product	$\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)$
Bounded Sum	$\min(1, \mu_{\bar{A}}(x) + \mu_{\bar{B}}(x))$	Bounded Product	$\max(0, \mu_{\bar{A}}(x) + \mu_{\bar{B}}(x) - 1)$
Drastic Sum	$\begin{cases} \mu_{\bar{A}}(x) & \text{if} \quad \mu_{\bar{B}}(x) = 0 \\ \mu_{\bar{B}}(x) & \text{if} \quad \mu_{\bar{A}}(x) = 0 \\ 1 & \text{otherwise} \end{cases}$	Drastic Product	$\begin{cases} \mu_{\bar{A}}(x) & \text{if} \quad \mu_{\bar{B}}(x) = 1 \\ \mu_{\bar{B}}(x) & \text{if} \quad \mu_{\bar{A}}(x) = 1 \\ 0 & \text{otherwise} \end{cases}$

Table 4.1: Example implementations of fuzzy logic (left) s-norm operations for $\bar{A} \cup \bar{B}$ and (right) t-norm operations for $\bar{A} \cap \bar{B}$

4.5.3 Fuzzification

Since control systems do not directly involve fuzzy sets, a fuzzification interface is used as a means to convert the crisp plant state or output measurements into fuzzy sets, so that fuzzy reasoning can be applied. Given a measurement x^* of variable x in universe of discourse X, the corresponding fuzzy set is $\bar{X} = \{(x, \mu(x : x^*)\}$. A few common choices are singleton, triangular, and Gaussian fuzzification. For singleton fuzzification,

$$\mu(x : x^*) = \begin{cases} 1 & \text{if} \quad x = x^* \\ 0 & \text{otherwise} \end{cases}$$

For triangular fuzzification,

$$\mu(x:x^*) = \begin{cases} 1 - \frac{|x-x^*|}{\lambda} & \text{if} \quad |x-x^*| < \lambda \\ 0 & \text{otherwise} \end{cases}$$

For Gaussian fuzzification,

$$\mu(x:x^*) = \exp\left(1 - \frac{-(x-x^*)^2}{\lambda^2}\right)$$

In each of the above cases, the parameter λ can either be selected by the designer or adapted online.

The fuzzification process converts each input variable x^* into a fuzzy set $\bar{X}$. Singleton fuzzification is often used as it greatly simplifies subsequent computations. Other forms of fuzzification may be more appropriate for representing uncertainty (or fuzziness) of the control system inputs due, for example, to measurement noise.

4.5.4 Fuzzy implication

The fuzzy rule base will contain a set of rules $\{R^l, l \in [l, \cdots, N]\}$ of the form

$$R^l : \text{IF}\left(\bar{x}_1 \quad \text{is} \quad \bar{X}_1^{l1}\right) \text{and} \ldots \text{and} \left(\bar{x}_n \quad \text{is} \quad \bar{X}_n^{l_n}\right) \text{THEN} \left(\bar{u} \quad \text{is} \quad \bar{U}^{l_n}\right) \tag{4.5.2}$$

where $l_i \in [1, \ldots, Ni]$ and $\bar{U}$ is the set of linguistic values defined for the fuzzy control signal $\bar{u}$. Each term in parenthesis is an *atomic fuzzy proposition*. The antecedent is the compound fuzzy proposition:

$$\bar{A}^l = \left(\bar{x}_1 \quad \text{is} \quad \bar{X}_1^{l1}\right) \text{and} \cdots \text{and} \left(\bar{x}_n \quad \text{is} \quad \bar{X}_n^{l_n}\right). \tag{4.5.3}$$

Each antecedent defines a fuzzy set in $\bar{X} = \bar{X}_1 \times \cdots \times \bar{X}_n$. The antecedent may contain multiple atomic fuzzy propositions using the same variable and need not include all fuzzy variables. The membership function for $\bar{A}^l$ is completely specified once the t-norm and s-norm representation of the "and" "or" operations are selected. Therefore, the applicability or confidence of rule R^l is calculated by the antecedent as

$$\mu_{\bar{A}^l}(x) = \mu_{\bar{X}^{l_l} \cap \ldots \cap \bar{X}^{l_n}}(\bar{x}_1, \cdots, \bar{x}_n) = \mu_{\bar{X}^{l_1}}(\bar{x}_1) \star \ldots \star \mu_{\bar{X}^{l_n}}(\bar{x}_n). \tag{4.5.4}$$

Note that when $\star$ is implemented as the algebraic product, then this membership function can have the form of a tensor product. If $\bar{x}_i$ is not a fuzzy singleton, then evaluation of each atomic fuzzy proposition can become computationally difficult. A rule (implication) of the form

$$R : \text{IF}\left(\bar{x}_1 \quad \text{is} \quad \bar{A}\right)\text{THEN}\left(\bar{u} \quad \text{is} \quad \bar{B}\right) \tag{4.5.5}$$

for $x \in \bar{X}$ and $u \in \bar{U}$ can be interpreted as a relation in $X \times U$. The membership function for this implication may have various forms depending on the interpretation of the implication operation. Four possibilities are displayed in Table 4.2. The first two interpretations are motivated by the fact that $A \Rightarrow B$ has the same truth table as $(\overline{(A)} \quad or \quad B)$. The third row is motivated by the fact that $A \Rightarrow B$ also has the same truth table as $((A \quad and \quad B) \quad or \quad \overline{(A)})$. Such direct truth table equivalence approaches are not always the most appropriate interpretations of the implication [27]. In some situations, a more causal situation is desired where the implication is interpreted as

$$\begin{array}{ll} \text{IF} \quad A & \\ & \text{THEN} \quad B \\ \text{ELSE} & \text{Nothing.} \end{array}$$

Such an interpretation of the implication is equivalent (in the truth table sense) to $(A \quad and \quad B)$. The fourth row indicates the membership function corresponding to this interpretation. Such **Mamdani** implications are widely used in fuzzy control approaches [9].

Name	Interpretation $A \Longrightarrow B$	Membership Function
Dienes-Reschler	$\left(\overline{(A)} \quad or \quad B\right)$	$\mu_R(x,u) = \max(1 - \mu_A(x), \mu_B(u))$
Lukasiewicz	$\left(\overline{(A)} \quad or \quad B\right)$	$\mu_R(x,u) = \min(1, (1 - \mu_A(x)) + \mu_B(u))$
Zadeh	$(A \quad and \quad B) \quad or \quad \overline{(A)}$	$\mu_R(x,u) = \max(\min(\mu_A(x), \mu_B(u)), 1 - \mu_A(x))$
Mamdani	$A \quad and \quad B$	$\mu_R(x,u) = t(\mu_A(x), \mu_B(u))$

Table 4.2: Interpretations of Fuzzy Implication. The — notation denotes logical negation.

4.5.5 Fuzzy Inference

Given the results of the two previous subsections, from a control system point of view, the inputs to the control system have been converted to fuzzy sets and each rule has been translated into a fuzzy relation. Pertaining to the issue of inference there are two related questions. How can the fuzzy set in U that results from a single rule be determined? How can the fuzzy set in U from a set of rules be determined?

According to the compositional rule of inference [95], given a rule of the form of eqn. (4.5.5) and a fuzzy set $\bar{R}$ with membership function μ_{barX}, then the membership function of the resultant fuzzy set in U can be found by the composition

$$\mu_R(u) = \sup_{x \in X} t(\mu_{\bar{X}}(x), \mu_{\bar{R}}(x,u)) \tag{4.5.6}$$

The above text has discussed the method for inferring the fuzzy set output corresponding to a single rule. The remainder of this section will be concerned with the problem of inferring the output fuzzy set that results from a set of rules called the *rule base*. A fuzzy rule base is called complete if for any $x \in X$ there exists at least one rule with a nonzero membership function (i.e.,$\forall x \in X, \exists l \ni \mu_{\bar{A}^l}(x) \neq 0$). Two methods of inferring the output of a rule base are possible: compositional inference and individual rule inference. In compositional inference [95], the relations corresponding to each rule are combined (through appropriately selected logical operations) into one relation representing the entire rule base. Then, composition with the input fuzzy sets is used to define the output fuzzy set. The composition of all the rules into a single relation can become cumbersome.

In individual rule inference, the output fuzzy set $U_i = \{(u, \mu_{\bar{R}^i}(u))\}$ corresponding to each individual rule is determined according to eqn. (4.5.6). The output of the inference engine, based on the entire (l rule) rule base, then has membership function described by either

$$\mu_{RB}(u) = \mu_{\bar{R}^1}(u) \oplus \cdots \oplus \mu_{\bar{R}^l}(u) \tag{4.5.7}$$

$$\textit{or} \quad \mu_{RB}(u) = \mu_{\bar{R}^1}(u) \star \cdots \star \mu_{\bar{R}^l}(u). \tag{4.5.8}$$

Eqn. (4.5.7) is used when the individual rules are interpreted as independent conditional statements intended to cover all possible operational situations. Eqn. (4.5.8) is used when the rule base is interpreted as a strongly coupled set of conditional statements that all should apply to the given situation. For example, given Mamdani product implication, eqn. (4.5.7), with the or operation implemented as max, the output membership function is

$$\mu_{RB}(u) = \max_{i=1}^{L} \left[\sup_{x \in X} (\mu_{\bar{X}}(x : x^*), \mu_{\bar{A}^i}(x), \mu_{\bar{A}^i}(x)) \right]. \tag{4.5.9}$$

Note that the resulting rule base membership function may be multimodal or have disconnected support.

4.5.6 Defuzzification

The purpose of the defuzzifier is to map a fuzzy set, such as $\bar{U} = (u, \mu_{RB}(u))$ for $u \in U$, to a crisp point $u^* \in U$. The point u^* should be in some sense most representative of U . Since there are many interpretations of most representative, there are also many means to implement the defuzzification process.

Table 4.3 summarizes three methods for performing defuzzification. The first method computes an *indexed center of gravity*. This method is often computationally difficult since the rule base membership function is typically not simple to describe. The middle row of the table describes *the center average* defuzzification process. The function "center" could, for example, select the midpoint of the set $\{u \in U | \mu_{\bar{R}^i}(u) > 0\}$. The center average is computationally easier than the indexed center of gravity approach. The final row of the table describes the *maximum defuzzification* process. The set $\mu RB(U)$ contains all values of u that achieve the maximum value of $\mu_{RB}(u)$ over U. The function g processes $\mu_{RB}(U)$ to produce a unique value for u^*. The function g could for example select the minimum, center, or maximum of $\mu_{RB}(U)$.

Defuzzification Method	Calculation
Indexed Center of Gravity	$u^* = \frac{\int_{U_\alpha} \mu_{RB}(u) u du}{\int_{U_\alpha} \mu_{RB}(u) du}$ $U_\alpha = \{u \in U \| \mu_{RB}(u) \geq \alpha\}$
Center Average	$c_i = \text{center}(\{u \in U \| \mu_{\bar{R}^i}(u) > 0\})$ $h_i = \sup_{u \in U}(\mu_{\bar{R}^i}(u))$ $u^* = \frac{\sum_{i=1}^{l} c_i h_i}{\sum_{i=1}^{l} h_i}$
Maximum	$\mu_{RB}(U) = \{u \in U \| \mu_{RB}(u) = \sup_{u \in U}(\mu_{RB}(u))\}$ $u^* = g(\mu_{RB}(U))$

Table 4.3: Example methods of defuzzification.

4.5.7 Takagi-Sugeno Fuzzy Systems

Takagi-Sugeno Fuzzy System is commonly used due to its simplicity. Thanks to its parametric form which enables the designer to analyze and study stability of the systems where such system is integrated. We can highlight the parallels between fuzzy approximators and the other approximators discussed in this chapter. The Takagi-Sugeno fuzzy system uses rules of the form

$$R^l : \text{IF} \left(\bar{x}_1 \quad \text{is} \quad \bar{X}_1^{l1}\right) \quad \text{and} \ldots \text{and} \quad \left(\bar{x}_n \quad \text{is} \quad \bar{X}_n^{l_n}\right) \text{THEN} \left(\bar{u} = f_i(x)\right) \tag{4.5.10}$$

In this thesis we are interested in fuzzy logic structures that represent $f_l(x)$ as a parameterized function (e.g., $f_l(x:\theta_l)$), and the parameters are identified based on experimental data. Typically,

$$f_l(x:\theta_l) = \theta_{l0} + \sum_{i=1}^{N} \theta_{li} x_i, \tag{4.5.11}$$

but nonlinear functions in either x or θ can be used. The membership function for the antecedent is formed as in eqn. (4.5.4). The Takagi-Sugeno approach then calculates the output control action as

$$u = \frac{\sum_i \mu_{\bar{A}^i}(x) f_i(x:\theta_l)}{\sum_j \mu_{\bar{A}^j}(x)} = \sum_i \Gamma_i(x)_i(x:\theta_l) \qquad \text{where} \Gamma_i(x) = \frac{\mu_{\bar{A}^i}(x)}{\sum_j \mu_{\bar{A}^j}(x)} \tag{4.5.12}$$

Note that this approximator has the form of a basis-influence function with basis set $\{f_i(x:\theta_i)\}$ and influence functions $\{\Gamma_i(x)\}$. If the fuzzy rule set is complete and each $\mu_{\bar{A}^i}(x)$ is finite, then this set of influence functions $\{\Gamma_i(x)\}$ will be finite, vanish nowhere, and form a partition of unity.

Equation (4.5.12) has a variety of interesting interpretations. The $f_i(x)$ can be previously existing operating point controllers or local controllers defined by human "experts." Alternatively, this expression can be interpreted as a "gain scheduled" controller. In all these cases, it is of interest to analyze the stability of the nonlinear closed-loop control loop that results.

One of the early motivations for fuzzy systems was there transparency, in the sense that users can (linguistically) read, describe, and understand the rule base. Similarly, a fuzzy system such as the Takagi-Sugeno type is similar to a smoothly interpolated gain scheduled controller, where each control law f_i is applicable over the support of γ_i. Fuzzy systems are capable of universal approximation [88]. Adaptation of fuzzy systems, as with any approximator, must be approached with caution. If for example the antecedents of the rule base are adapted, this is a nonlinear estimation process. Adaptation of the antecedents could lead to loss of completeness of the fuzzy rule base.

4.6 Conclusion

This chapter has briefly introduced various approximation structures. Several of these structures have entire books or journals devoted to their study. Therefore, we have only touched the surface in this chapter. This will provide an overview on the tools that will be used in the Function Approximation based adaptive control that are discussed in chapter 6.

Chapter 5

Robust and Adaptive Control of Uncertain Nonlinear Systems

5.1 Introduction

In the area of *"robust control"* the focus is on the development of controllers that can maintain good performance even if we only have a poor model of the plant or if there are some plant parameter variations. In the area, of adaptive control, to reduce the effects of plant parameter variations, robustness is achieved by adjusting (i.e., adapting) the controller on-line [55, 71].

We will use adaptive mechanisms within the control laws when certain parameters within the plant dynamics are unknown. An adaptive controller will thus be used to improve the closed-loop system robustness while meeting a set of performance objectives. If the plant uncertainty cannot be expressed in terms of unknown parameters, one may be able to reformulate the problem by expressing the uncertainty in terms of a fuzzy system, neural network, or some other parameterized nonlinearity. The uncertainty then becomes recast in terms of a new set of unknown parameters that may be adjusted using adaptive techniques. The purpose of this chapter is to summarize a collection of standard control design techniques for certain classes of nonlinear systems. Later we will use these control techniques to develop adaptive control approaches that are suitable for use when there is additional uncertainty in the plant dynamics in the framework of function approximation based control. We will use Lyapunov-based design techniques where a controller is chosen to help decrease a measure of the system error.

5.2 Robust Nonlinear Control

Nonlinear control methodologies generally, namely small-signal linearization, feedback linearization, and backstepping. are based on the key assumption that the control designer exactly knows the system nonlinearities [78, 79]. In practice, this is not a realistic assumption. Consequently, it is important to consider ways to make these approaches more robust with respect to modeling errors. In this section we introduce a set of nonlinear control design tools that are based on the principle of assuming that the unknown component of the nonlinearities are bounded in some way by a known function. If this assumption is satisfied then it is possible to derive nonlinear control schemes that utilize these known bounding functions instead of the unknown nonlinearities. Although these techniques have been extensively studied in the nonlinear control literature, they tend to yield conservative control laws, especially in cases where the uncertainty is significant [40]. The term conservative is used among control engineers to indicate the fact that due to the uncertainty the control effort applied is more than needed. As a result, the control signal $u(t)$ may be large (high-gain feedback), which may cause several problems, such as saturation of the actuators, large error in the presence of measurement noise, excitation of unmodeled dynamics, and large transient errors. Furthermore, as we will see, these techniques typically involve a switching control function, which may cause chattering. The robust nonlinear control design methods developed in this section provide an important perspective for the adaptive approximation based control described in the next Chapters.

Specifically, adaptive approximation based control can be viewed as a way of reducing uncertainty during operation such that the need for conservative robust control can be eliminated or reduced. Another reason for studying these techniques in the context of adaptive approximation is their utilization, as we will see, to guarantee closed-loop stability outside of the approximation region $\mathcal{D}$ [19, 63].

This section presents five nonlinear control design tools: (i) bounding control, (ii) sliding mode control, (iii) Lyapunov redesign method, (iv) nonlinear damping, and (v) adaptive bounding. As we will see, these techniques are, in fact, quite similar.

5.2.1 Bounding Control

Bounding control is one of the simplest approaches for dealing with unknown nonlinearities. Here, we consider a simple scalar system with one unknown nonlinearity, which lies within certain known bounds [40]. This approach can be extended to more complex systems. Consider the scalar nonlinear system

$$\dot{x} = f(x) + u \tag{5.2.1}$$

where the objective is to design a control law such that $y(t) = x(t)$ tracks a desired signal $y_d(t)$. Let $e(t) = y(t) - y_d(t)$ be the tracking error. We assume that the function f is unknown but belongs to a certain known range as follows:

$$f_L(x) \leq f(x) \leq f_U(x), \qquad \forall x \in \mathbb{R}^1$$

where f_L and f_U are known lower and upper bounds, respectively, on the unknown function f. Consider the following control law:

$$u = \begin{cases} -a_m(x - y_d) + \dot{y}_d - f_U(x) & \text{if} \quad e \geq 0 \\ -a_m(x - y_d) + \dot{y}_d - f_L(x) & \text{if} \quad e < 0 \end{cases} \tag{5.2.2}$$

where $a_m > 0$. Using the above control, it is easy to see that the tracking error dynamics satisfy

$$\begin{cases} \dot{e} = -a_m e + f(x) - f_U(x) & \text{if} \quad e \geq 0 \\ \dot{e} = -a_m e + f(x) - f_L(x) & \text{if} \quad e < 0 \end{cases} \tag{5.2.3}$$

Now, let $V = \frac{1}{2}e^2 \geq 0$ be a Lyapunov function candidate. The time derivative of V along the error dynamics satisfies

$$\dot{V} = \begin{cases} -a_m e^2 + (f(x) - f_U(x))e & \text{if} \quad e \geq 0 \\ -a_m e^2 + (f(x) - -f_L(x))e & \text{if} \quad e < 0 \end{cases}$$

$$\dot{V} \quad \leq -a_m e^2$$

Therefore, the tracking error converges exponentially to zero. It is noted that, in general, the control law (5.2.2) is discontinuous at $e = 0$. This may result in the trajectory $x(t)$ going back and forth between y_d^+, (denoting a value of the trajectory $y(t)$ which is slightly larger than $y_d(t)$); and

y_d^-, (denoting a value of the trajectory $y(t)$ which is slightly smaller than $y_d(t)$) causing the control law to be switching, thus creating chattering problems. The chattering can be remedied by using a smooth approximation to the control law of the form

$$u = \begin{cases} -a_m(x - y_d) + \dot{y}_d - f_U(x) & \text{if} \quad e \geq \delta \\ -a_m(x - y_d) + \dot{y}_d - \frac{1}{2\delta}\left[(\delta - e)f_L(y_d - \delta) = (\delta + e)f_U(y_d + \delta)\right] & \text{if} \quad |e| \geq \delta \\ -a_m(x - y_d) + \dot{y}_d - f_L(x) & \text{if} \quad e < -\delta \end{cases} \tag{5.2.4}$$

where $\delta > 0$ is a small design constant. I t can be proven that the control law of equation (5.2.4) achieves convergence of the set $|x| < \delta$

5.2.2 Sliding Mode Control

Sliding Mode Control is a methodology based on the principle that it is easier to control a first-order system than a n-th order system. Therefore, this approach can be viewed as a way to reduce a higher-order control problem into a simpler one for which there are known feedback control methods. This simplification comes at the expense of using a large control effort, which, as discussed earlier in the chapter, could be the source of other potential problems, especially in the presence of measurement noise or high frequency unmodeled dynamics. The sliding mode control methodology can be applied to several classes of nonlinear systems [15]. Here, we consider its application to a class of feedback linearizable systems.

Consider an n-th order nonlinear system, affine in control of the form

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \vdots \\ \dot{x}_{n-1} = x_n \\ \dot{x}_n = f(x) + g(x)u \end{cases} \tag{5.2.5}$$

where it is assumed that f and g are unknown and $g(x) \geq g_o > 0$ for all $x \in \mathbb{R}^n$. The control objective is for $y(t) = x_1(t)$ to track a desired signal $y_d(t)$.

Let $e = y - y_d$ be the tracking error. The sliding mode surface s is defined as

$$s = e^{(n-1)} + \lambda_{n-1}e^{(n-2)} + \cdots + \lambda_2\dot{e} + \lambda_1 e = 0, \tag{5.2.6}$$

where the coefficients $\{\lambda_1, \lambda_2, \cdots, \lambda_{n-1}\}$ are selected such that the characteristic polynomial (in s)

$$s^{(n-1)} + \lambda_{n-1} s^{(n-2)} + \cdots + \lambda_2 s + \lambda_1 = 0 \tag{5.2.7}$$

is Hurwitz. The manifold described by $s = 0$ is referred to as the *sliding manifold* or *sliding surface* and has dimension $(n-1)$. The objective of sliding mode control is to steer the trajectory onto this sliding manifold. This is achieved by forcing the variable s to zero in finite time. By design of the sliding surface, if x is on the sliding surface defined by s = 0, then

$$e^{(n-1)} - \lambda_{n-1} e^{(n-2)} - \cdots - \lambda_2 \dot{e} - \lambda_1 e.$$

Since the polynomial given by (5.2.7) is Hurwitz, once on the sliding manifold the tracking error will go to zero with a transient behavior characterized by the selected coefficients λ_i (i.e., exponentially fast). The sliding mode control objective can be achieved if the control law u is chosen such that

$$\frac{d}{dt}\frac{1}{2}s^2 \leq -\kappa |s|,$$

where $\kappa > 0$. In this case, the upper right-hand derivative of $|s(t)|$ satisfies the differential inequality

$$\frac{d^+}{dt}|s(t)| \leq -\kappa,$$

which implies that the trajectory reaches the manifold $s = 0$ in finite time. Following (5.2.6), the derivative of $s(t)$ satisfies

$$\dot{s} = f(x) + g(x)u - y_d^{(n)} + \lambda_{n-1} e^{(n-1)} + \cdots + \lambda_2 \ddot{e} + \lambda_1 \dot{e}.$$

If f and g were known function, then we could choose the control law

$$u = \frac{1}{g(x)}\left[-f(x) + y_d^{(n)} - \lambda_{n-1} e^{(n-1)} - \cdots - \lambda_2 \ddot{e} - \lambda_1 \dot{e} - \kappa \,\mathrm{sgn}(s)\right].$$

where κ is a positive design gain and sgn(.) denotes the sign function:

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ -1 & \text{if} \quad x < 0. \end{cases}$$

Based on this control law, the derivative of $s(t)$ satisfies

$$\dot{s} = -\kappa \,\mathrm{sgn}(s),$$

which implies

$$\begin{aligned}\tfrac{d}{dt}\tfrac{1}{2}s^2 &= s\dot{s}\\ &= -s\kappa \,\mathrm{sgn}(s)\\ &= -\kappa|s|\end{aligned}$$

Now consider the case where f and g are unknown but the designer has a known upper bound $\eta(x,t)$ such that

$$\left|\frac{f(x) - y_d^{(n)} + \lambda_{n-1}e^{(n-1)} + \cdots + \lambda_2\ddot{e} + \lambda_1\dot{e}}{g(x)}\right| \leq \eta(x,t).$$

Suppose that the control law is selected as

$$u = -(\eta(x,t) + \eta_0)\,\mathrm{sgn}(s), \tag{5.2.8}$$

where $\eta_0 > 0$ is a design constant. Using the Lyapunov candidate $V = \frac{s^2}{2}$, Its derivative along the tracking error dynamics can be expressed by:

$$\begin{aligned}\dot{V} &= s\dot{s}\\ &= s\left(f(x) - y_d^{(n)} + \lambda_{n-1}e^{(n-1)} + \cdots + \lambda_2\ddot{e} + \lambda_1\dot{e}\right) + sg(x)u\\ &\leq |s|\eta(x,t)g(x) + sg(x)u\\ &= -\eta_0 g_0|s|,\end{aligned}$$

where g_0 is defined in (5.2.5). Therefore, we have achieved the desired objective of forcing the trajectory onto the sliding manifold in finite time. It is interesting to note that this is achieved without specific knowledge of f and g, just the upper bound $\eta(x,t)$. Despite the resulting stability and convergence properties of the sliding mode control approach, it has two key drawbacks in its standard form. The sliding mode control law has two components, the gain $\eta(x,t) + \eta_0$ and the switching function $\mathrm{sgn}(s)$, both of which can create problems:

1. (**High-Gain**) Note that the gain term is the result of taking an upper bound on the uncertainty. In general, this creates a high-gain feedback control, which can create problem in the presence of measurement noise and high-frequency unmodeled dynamics. Moreover, high-gain feedback may require significant control effort, which can be expensive and/or may cause saturation of the actuators. In practice, high-gain feedback control is to be avoided [82].

2. (**Chattering**) The switching function $\mathrm{sgn}(s)$ causes the control gain to switch from $\eta(x,t)+\eta_0$ to $-(\eta(x,t) + \eta_0)$ every time the trajectory crosses the sliding manifold. Although in theory

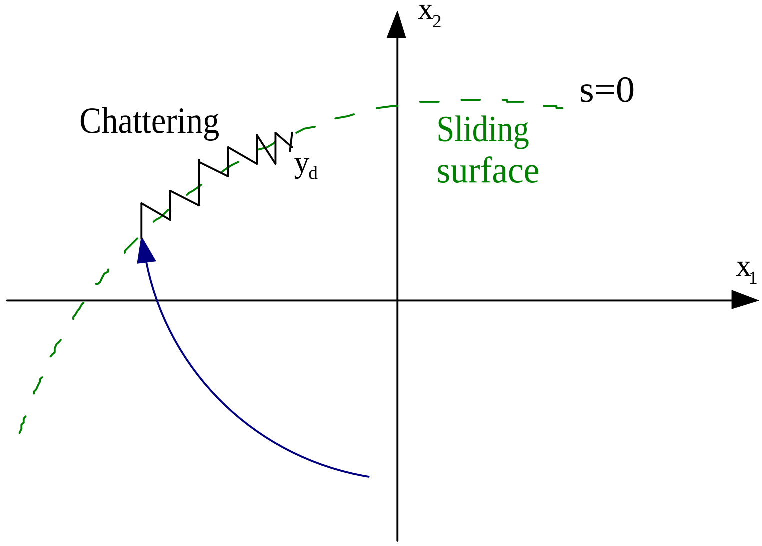

Figure 5.1: Graphical illustration of sliding mode control and chattering as a result of imperfection in the switching.

> the trajectory is suppose to slide on the sliding manifold, in practice there are imperfections and delays in the switching devices, which lead to chattering. This is illustrated in Figure 5.1. Chattering causes significant problems in the feedback control system, especially if it is associated with high gains. For example, chattering may excite high-frequency dynamics which were neglected in the design model, it can cause wear and tear of moving mechanical parts and it can cause high heat losses in electrical power systems [15]

Research in sliding mode control has developed some techniques for addressing the above two issues. The high gain problem can be reduced by using as much a priori information as possible, thus canceling the known nonlinearities and employing an upper bound only for the unknown portions of the nonlinearities. The chattering problem can also be addressed, partially, by employing a continuous approximation of the sign function. The tradeoff in the use of this approximation is that only uniform boundedness of solutions can be proved. Despite these remedies, the sliding mode methodology is based on the principle of bounding the uncertainty by a larger function, and as a result it is a conservative control approach. In this text, we present a methodology for learning or approximating the uncertainty online, instead of using an upper bound for it. However, the approximation will be valid only within a certain compact region $\mathcal{D}$. In order to achieve stability outside this region, we will rely on bounding control techniques such as sliding mode.

5.2.3 Lyapunov Redesign Method

Consider a nonlinear system described by

$$\dot{x} = f(x) + G(x)u, \tag{5.2.9}$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the controlled input. Assume that the vector field $f(x)$ and the matrix $G(z)$ each consist of two components: a known nominal part and an unknown part. Therefore,

$$f(x) = f_0(x) + f^*(x) \tag{5.2.10}$$

$$G(x) = G_0(x) + G^*(x) \tag{5.2.11}$$

where f_0 and G_0 characterize the known nominal plant, and f^*, G^* represent the uncertainty.

Moreover, we assume that the uncertainty satisfies a so-called *matching condition*:

$$f^*(x) = G_0(x)\Delta_f^*(x) \tag{5.2.12}$$

$$G^*(x) = G_0(x)\Delta_G^*(x). \tag{5.2.13}$$

The matching condition implies that the uncertainty terms appear in the same equations as the control inputs u, and as a result they can be handled by the controller [40].

Using equations (5.2.10,5.2.11,5.2.12,5.2.13), the system dynamics in 5.2.9 can be rewritten as:

$$\dot{x} = f_0(x) + G_0(x)\left(u + \eta(x,u)\right), \tag{5.2.14}$$

where $\eta = \Delta_f^* + \Delta_G^* u$ comprises all the uncertainty terms.

The Lyapunov redesign method addresses the following problem: Suppose that the equilibrium of the nominal model $\dot{x} = f_0(x) + G_0(x)u$ can be made uniformly asymptotically stable by using a feedback control law $u = p_0(x)$. The objective is to design a corrective control function $p^*(x)$ such that the augmented control law $u = p_0(x) + p^*(x)$ is able to stabilize the system (5.2.14) subject to the uncertainty $\eta(x,u)$ being bounded by a known function [42].

We assume that there exists a control law $u = p_0(x)$ such that $x = 0$ is a uniformly asymptotically stable equilibrium point of the closed-loop nominal system

$$\dot{x} = f_0(x) + G_0(x)p_0(x).$$

Based on the assumption of the existence of a Lyapunov V_0, and strictly increasing class $\mathcal{K}_\infty$ functions $\alpha_1, \alpha_2, \alpha_3 : \mathbb{R}^+ \mapsto \mathbb{R}^1$, such that:

$$\alpha_1\|x\| \leq V_0(x) \leq \alpha_2\|x\|. \tag{5.2.15}$$

$$\frac{\partial V_0}{\partial x}\left[f_0(x) + G_0(x)p_0(x)\right] \leq \alpha_3\|x\|. \tag{5.2.16}$$

The uncertainty term is assumed to satisfy the bound

$$\|\eta(x,u)\|_\infty \leq \bar{\eta}(t,x) \tag{5.2.17}$$

where the bounding function $\bar{\eta}$ is assumed to be known apriori or available for measurement. Now, we will proceed to the design of the corrective control component $p^*(x)$ such that $u = p_0 + p^*$ stabilizes the class of systems described by (5.2.5) and satisfying (5.2.17). The corrective control term is designed based on a technique following the nominal Lyapunov function V_0, which justifies the name Lyapunov redesign method. Consider the same Lyapunov function V_0 that guarantees the asymptotic stability of the nominal closed-loop system, but now consider the time derivative of V_0 along the solutions of the full system (5.2.5). We have

$$\begin{aligned} \dot{V}_0 &= \tfrac{\partial V_0}{\partial x}\left[f_0(x) + G_0(x)(u + \eta(x,u))\right] \\ &= \tfrac{\partial V_0}{\partial x}\left[f_0(x) + G_0(x)p_0(x)\right] + \tfrac{\partial V_0}{\partial x}G_0(x)\left(p^*(x) + \eta(x,u)\right) \\ &\leq -\alpha_3(\|x\|) + \omega(x)^{\mathrm{T}}\eta(x,u), \end{aligned} \tag{5.2.18}$$

where

$$\omega(x) = \left[\frac{\partial V_0}{\partial x}G_0(x)\right]^{\mathrm{T}} \quad \in \quad \mathbb{R}^m \tag{5.2.19}$$

which is a known function. By taking bounds we obtain

$$\begin{aligned} \dot{V}_0 &\leq -\alpha_3(\|x\|) + \textstyle\sum_{i=1}^m \omega_i(x)p_i^*(x) + \|w(x)\|_1\|\eta(x,u)\|_\infty \\ &= -\alpha_3(\|x\|) + \textstyle\sum_{i=1}^m \left(\omega_i(x)p_i^*(x) + \bar{\eta}(x,t)|w_i(x)|\right) \end{aligned} \tag{5.2.20}$$

The second term of the right-hand side of (5.2.20) can be made zero if $p_i^*(x)$ is selected as

$$p_i^*(x) = -\bar{\eta}(x,t)\mathrm{sgn}(w_i(x)) \tag{5.2.21}$$

Each component of the corrective control vector $p_i^*(x)$ is selected to be of the form $p_i * (x) = \pm\bar{\eta}(x,t)$, where the sign of $p_i^*(x)$ depends on the sign of $\omega_i(x)$ and, in fact, changes as $w_i(x)$ changes sign.

By substituting (5.2.21) in (5.2.20) we obtain the desired stability property

$$\dot{V}_0 = -\alpha_3(\|x\|) \tag{5.2.22}$$

which implies that the closed-loop system is asymptotically stable.

The augmented control law $u = p_0(x) + p^*(x)$ is discontinuous. Moreover, the discontinuity jump $\bar{\eta}(x,t) \mapsto -\bar{\eta}(x,t)$ can be of large magnitude if the uncertainty bound $\bar{\eta}$ is large. As discussed earlier, discontinuities in the control law can cause chattering, therefore it is desirable to smooth the discontinuity and at the same time retain to some degree the nice stability properties of the original discontinuous control law. This can be achieved by replacing the sgn function with a smooth tanh $\left(\frac{w_i(x)}{\epsilon}\right)$, where $\epsilon > 0$ is a small design constant [82].

5.2.4 Nonlinear Damping

One of the key assumptions made in the Lyapunov redesign approach is that the uncertainty term $\eta(x,u)$ is bounded by a known bounding term $\bar{\eta}(t,x)$. The nonlinear damping method developed in this section relaxes somewhat this assumption by not requiring that the bounding term $\bar{\eta}$ is known [40].

Consider the system described by

$$\dot{x} = f_0(x) + G_0(x)\left(u + \eta(x,u)\right), \tag{5.2.23}$$

The uncertainty function $\eta(x,u)$ is assumed to be of the form

$$\eta(x,u) = \Phi(t,x)\eta_0(x,u), \tag{5.2.24}$$

where the $m \times m$ matrix Φ is known, and η_0 is unknown but uniformly bounded (i.e., $\|\eta_0(x,u)\|_\infty < M$ for all (x,u)). In this case the bound M does not need to be known. Again, the objective is to design a corrective control law p*(z) that stabilizes the closed loop system. Following the same procedure as in subsection 5.2.3, we consider a nominal Lyapunov function $V_o(x)$ that satisfies (5.2.15), (5.2.16) for some class $\mathcal{K}_\infty$ functions $\alpha_1, \alpha_2, \alpha_3$. The time derivative of V_0 along the solutions of (5.2.23), (5.2.24) is given by

$$\begin{aligned} \dot{V}_0 &= \tfrac{\partial V_0}{\partial x}\left[f_0(x) + G_0(x)(u + \Phi(t,x)\eta_0(x,t))\right] \\ &\leq -\alpha_3(\|x\|) + \omega(x)^{\mathrm{T}} p^*(x) + \omega(x)^{\mathrm{T}}\Phi(t,x)\eta_0(x,u), \end{aligned} \tag{5.2.25}$$

where $\omega(x)$ is the same as defined in (5.2.19). Now, let us select $p^*(x)$ as

$$p^*(x) = -k\omega(x)\|\Phi(t,x)\|_2^2, \qquad \text{where} \quad k > 0. \tag{5.2.26}$$

Substituting for p^* in equation (5.2.25) yields

$$\dot{V}_0 \leq -\alpha_3(\|x\|) - k\|\omega(x)\|_2^2\|\Phi(t,x)\|_2^2 + \omega(x)^{\mathrm{T}}\Phi(t,x)\eta_0(x,u). \tag{5.2.27}$$

Using the fact that $\eta_0(x,u)$ is uniformly bounded in (x,u), we can obtain

$$\omega(x)^{\mathrm{T}}\Phi(t,x)\eta_0(x,u) \leq \|\omega(x)\|_2\|\Phi(t,x)\|_2 M.$$

The term

$$Q = -k\|\omega(x)\|_2^2\|\Phi(t,x)\|_2^2 + \|\omega(x)\|_2\|\Phi(t,x)\|_2 M$$

is of the form $Q(a) = -ka^2 + aM$, where $a = \|\omega(x)\|_2 \cdot \|\Phi(t,x)\|_2$; therefore, Q attains the maximum value of $M/4k$ at $a = M/2k$. Therefore,

$$\dot{V}_0 \leq -\alpha_3(\|x\|) + \frac{M}{4k}. \tag{5.2.28}$$

Since $\alpha_3(\|x\|)$ is strictly increasing and approaches ∞ as $\|x\| \to \infty$, there exists a ball $\mathcal{B}_\rho$, of radius ρ such that $\dot{V}_0 \leq 0$ for x outside $\mathcal{B}_\rho$. Therefore, the closed-loop system is uniformly bounded and the trajectory $x(t)$ converges to the invariant set

$$\Omega = \left\{ x \,\middle|\, V_0(x) \leq \max_{x=\rho}(V_0(x)) \right\}, \tag{5.2.29}$$

where ρ can be made smaller by increasing the feedback gain k or by decreasing the infinity norm of the model error.

5.2.5 Adaptive Bounding Control

Of the four techniques presented in this section, namely bounding control, sliding mode control, Lyapunov redesign, and nonlinear damping, the first three are based on the key assumption of a known bound on the uncertainty. The nonlinear damping technique does not make this bounding assumption; however, the resulting stability property does not guarantee the convergence of the tracking error to zero, but to an invariant set whose radius is proportional to the m-norm of the uncertainty. Even though the residual error in the nonlinear damping design can be reduced by increasing the feedback gain parameter k , this is not without drawbacks, since increasing the feedback gain may result in high-gain feedback , with all the undesirable consequences. In this subsection, we introduce another technique which also relaxes the assumption of a known bound. Specifically, it is assumed that $\eta(x, u)$ is bounded by

$$\|\eta(x, u)\|_\infty \leq \theta^{\mathrm{T}}\phi(x, t), \tag{5.2.30}$$

where $\theta \in \mathbb{R}^q$ is an unknown parameter vector and ϕ is a known vector function. Since $\theta^{\mathrm{T}}\phi$ represents a bound on the uncertainty, each element of θ and $\phi(x, t)$ is assumed to be non-negative. Typically, the dimension q is simply equal to one. However, the general case where both θ and $\phi(x, t)$ are vectors, allows the control designer to take advantage of any knowledge where the bound changes for different regions of the state-space x. If a known function $\phi(x)$ is not available, it can be simply assumed that $\|\eta(x, u)\|_\infty \leq \theta$, where θ is a scalar unknown bounding constant. The adaptive bounding control method was introduced in [70] and was later used in neural control [69].

It is worth noting that the bounding assumption of the adaptive bounding control method is significantly less restrictive than that of the Lyapunov redesign method where the bound is assumed to be known. Even though one may consider simply increasing the bound of the Lyapunov redesign method until the assumed bound holds, this is not always possible, and quite often it is not an astute way to handle the problem since it will increase the feedback gain of the system. The adaptive bounding control technique is based on the idea of estimating online the unknown parameter vector θ. The feedback controller utilizes the parameter estimate $\hat{\theta}(t)$ instead of the true bounding vector θ. One of the key questions has to do with the design of the adaptive law for generating $\hat{\theta}(t)$. As we will see, this is achieved again by Lyapunov analysis.

Let $\tilde{\theta}(t) = \hat{\theta}(t) - \theta$ denote the parameter estimation error. Consider the augmented Lyapunov function

$$V(x, \tilde{\theta}) = V_0(x) + \frac{1}{2}\tilde{\theta}^{\mathrm{T}}\Gamma^{-1}\tilde{\theta}^{\mathrm{T}}, \tag{5.2.31}$$

where Γ is a positive definite matrix of dimension $q \times q$, which represents the adaptive gain. Differentiating V along the dynamics of (5.2.14), we obtain

$$\begin{array}{rl} \dot{V} & = \frac{\partial V_0}{\partial x}\left[f_0(x) + G_0(x)(u + \eta(xu))\right] + \tilde{\theta}^{\mathrm{T}}\Gamma^{-1}\dot{\tilde{\theta}} \\ & \leq -\alpha_3(\|x\|) + \omega(x)^{\mathrm{T}}p^*(x) + \omega(x)^{\mathrm{T}}\eta(x,u) + \tilde{\theta}^{\mathrm{T}}\Gamma^{-1}\dot{\hat{\theta}} \\ & \leq -\alpha_3(\|x\|) + \sum_{i=1}^{m}\Big(\omega_i(x)p_i^*(x) + \hat{\theta}^{\mathrm{T}}\phi(x,t)|w_i(x)|\Big) \\ & -\tilde{\theta}^{\mathrm{T}}\phi(x,t)\|w(x)\|_1 + \tilde{\theta}^{\mathrm{T}}\Gamma^{-1}\dot{\hat{\theta}} \end{array} \tag{5.2.32}$$

We choose the corrective control term $p_i^*(x)$ and the update law for $\hat{\theta}$ as follows:

$$\begin{cases} p_i^*(x) = -hat\theta^{\mathrm{T}}\phi(x,t)\mathrm{sgn}(\omega_i(x)) \\ \dot{\hat{\theta}} = \Gamma\phi(x,t)\|\omega(x)\|_1, \end{cases} \tag{5.2.33}$$

which implies that $V \leq -\alpha_3(\|x\|)$. Therefore, both $x(t)$ and $\tilde{\theta}$ remain bounded and $x(t)$ converges to zero (using Barbalat's Lemma).

As in subsection (5.2.4), one can use $\tanh(\omega_i/\epsilon)$ instead the discontinuous function sgn(.) to make the control law smooth and reduce chattering. Another issue that arises with adaptive bounding control is the possible parameter drift of the bounding estimate $\theta(\hat{t})$. This may occur as a consequence of using the smooth approximation $\tanh(w_i(z)/\epsilon)$, which may result in a small residual error. Moreover, in the presence of measurement noise or disturbances, again the bounding parameter estimate $\hat{\theta}$ may not converge. Since the right-hand side of (5.106) is nondecreasing, the presence of such residual errors (even if small) may cause the parameter drift of the estimate, which in turn will cause the feedback control signal to become large. This can be prevented by using a robust adaptive law. One of the available techniques is the dead-zone, which requires knowledge of the size of the residual error. Another method is the projection modification, which prevents the parameter estimate from becoming larger than a preselected level. Yet another approach is the σ modification.

5.3 Adaptive Nonlinear Control

Adaptive control deals with systems where some of the parameters are unknown or slowly time-varying. The basic idea behind adaptive control is to estimate the unknown parameters online using parametric function approximators, and then to use the estimated parameters, in place of the unknown ones, in the feedback control law. Most of the research in adaptive control has been developed for linear models, even though in the last decade or so there has been a lot of activity on adaptive nonlinear control as well. Even in the case of adaptive control applied to linear systems, the resulting control law is nonlinear. This is due to the parameter update laws, which render the feedback controller nonlinear.

There are two strategies for combining the control law and the parameter estimation algorithm. In the first strategy, referred to as indirect adaptive control, the parameter estimation algorithm is used to estimate the unknown parameters of the plant. Based on these parameter estimates, the control law is computed by treating the estimates as if they were the true parameters, based on the certainty equivalence principle [7]. In the second strategy, referred to as direct adaptive control, the parameter estimator is used to estimate directly the unknown controller parameters. It is interesting to note the similarities and the differences between the so called the robust control laws and the adaptive control laws. The robust approaches, which were discussed in the previous section, treat the uncertainty as an unknown box where the only information available are some bounds. The robust control law is obtained based on these bounds, and in fact is designed to stabilizes the system for any uncertainty within the assumed bounds. As a result, the robust control law tends to be conservative and it may lead to large control input signals or control saturation. On the other hand, adaptive control assumes a special structure for the uncertainty where the nonlinearities are known but the parameters are unknown. In contrast to robust control, in adaptive control the objective is to try to estimate the uncertain (or time-varying) parameters to reduce the level of uncertainty. In the next chapter, we will start investigating the adaptive approximation control approach where the uncertainty also includes nonlinearities that are estimated online. Hence, adaptive approximation based control can be viewed as an expansion of the adaptive control methodology where instead of having simply unknown parameters we have unknown nonlinearities. Adaptive control is a well-established methodology in the design of feedback control systems. The first practical attempts to design adaptive feedback control systems go back as far as the 1950s, in connection with the design of autopilots [63]. Stability analysis of adaptive control for linear systems started in the mid-1960s [66] and culminated in 1980 with the complete stability proof for linear systems [44]. The first stability results assumed that the only uncertainty in the system was due to unknown parameters; i.e., no disturbances, measurement noise, nor any other form of uncertainty. In the 1980s, adaptive control research focused on robust adaptive control for linear systems, which dealt with modifications to the adaptive algorithms and the control law in order to address some types of uncertainties [36]. In the 1990s, most of the effort in adaptive control focused on adaptive control of nonlinear systems with some elegant results [55]. To illustrate the use of the adaptive control methodology we consider below two examples of adaptive nonlinear control.

5.3.1 Adaptive Feedback Linearization Example

Feedback Linearization is one of the most powerful and commonly found techniques in nonlinear control. It is based on the idea of linearizing the nonlinear dynamics by coordinate transformation. The nonlinearities are canceled by the combined use of feedback and change of coordinates. In such approach, the designer must have an exact model of the system in question. Much research has been conducted to robustify such a nonlinear constructive method and generalize it to wider classes of systems [37, 79].

Consider the n-th order model

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \vdots \\ \dot{x}_{n-1} = x_n \\ \dot{x}_n = \sum_{i=1}^{q-1} \theta_i f_i(x) + \theta_q u \end{cases} \tag{5.3.1}$$

where $(\theta_i; \quad i = 1, \cdots, q)$ are unknown, constant parameters and $(f_i; \quad i = 1, \cdots, q-1)$ are known functions. The objective is to design an adaptive controller such that $y(t) = x_1(t)$ tracks a given desired signal $y_d(t)$. Let $e = y - y_d$ be the tracking error.

If $(\theta_i; \quad i = 1, \cdots, q)$ were known and $\theta_q \neq 0$ then the control law

$$u = \frac{1}{\theta_q}\left[-\sum_{i=1}^{q-1} \theta_i f_i(x) + y_d^{(n)} - \lambda_{n-1} e^{(n-1)} - \cdots - \lambda_2 \ddot{e} - \lambda_1 \dot{e} - \lambda_0 e\right] \tag{5.3.2}$$

would result in the following linear tracking error dynamics:

$$e^{(n)} + \lambda_{n-1}e^{(n-1)} + \cdots + \lambda_2\ddot{e} + \lambda_1\dot{e} + \lambda_0 e = 0 \tag{5.3.3}$$

The coefficients $\{\lambda_0, \lambda_1, \lambda_2, \cdots, \lambda_{n-1}\}$ would be selected such that the characteristic polynomial

$$s^n + \lambda_{n-1}n^{n-1} + \cdots + \lambda_2 s^2 + \lambda_1 s + \lambda_0 = 0 \tag{5.3.4}$$

has all its roots in the left-half complex plane. Since $(\theta_i; \quad i = 1, \cdots, q)$ are unknown, we replace them in the control law by their corresponding estimates $(\hat{\theta}_i(t); \quad i = 1, \cdots, q)$, where it is assumed for the time being that $\hat{\theta}_q(t) \neq 0$ for all $t \geq 0$. The adaptive control law is given by

$$u = \frac{1}{\hat{\theta}_q}\left[-\sum_{i=1}^{q-1}\hat{\theta}_i f_i(x) + y_d^{(n)} - \lambda_{n-1}e^{(n-1)} - \cdots - \lambda_2\ddot{e} - \lambda_1\dot{e} - \lambda_0 e\right] \tag{5.3.5}$$

which yields the following tracking error dynamics

$$e^{(n)} + \lambda_{n-1}e^{(n-1)} + \cdots + \lambda_2\ddot{e} + \lambda_1\dot{e} + \lambda_0 e = -\sum_{i=1}^{q-1}\tilde{\theta}_i f_i(x) - \tilde{\theta}_q u \tag{5.3.6}$$

where $\tilde{\theta}_i = \hat{\theta}_i - \theta_i$ for $i = 1, \cdots, q$. If we let $E = [\ e \quad \dot{e} \quad \ddot{e} \quad \cdots \quad e^{(n-1)}\ ]$ then the tracking error dynamics can be expressed as

$$\dot{E} = A_0 E - B_0\left(\sum_{i=1}^{q-1}\tilde{\theta}_i f_i(x) + \tilde{\theta}_q u\right) \tag{5.3.7}$$

where A_0, B_0 is the controller canonical form pair given by:

$$A_0 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & & & & 1 \\ -\lambda_0 & -\lambda_1 & & \cdots & -\lambda_{n-1} \end{bmatrix}, \qquad B_0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

Since A_0 is a stability matrix, there exists a positive definite matrix P such that

$$A_0^{\mathrm{T}}P + PA_0 = -I.$$

Choosing the Lyapunov function

$$V = E^{\mathrm{T}}PE + \sum_{i=1}^{q}\frac{\tilde{\theta}_i^2}{\gamma_i}$$

whose time derivative along the solution of the tracking error dynamics is given by

$$\dot{V} = -E^{\mathrm{T}}E + 2\sum_{i=1}^{q-1}\frac{1}{\gamma_i}\tilde{\theta}_i\left(\dot{\hat{\theta}}_i - \gamma_i E^{\mathrm{T}}PB_0 f_i(x)\right) + \frac{2}{\gamma_q}\tilde{\theta}_q\left(\dot{\hat{\theta}}_q - \gamma_q E^{\mathrm{T}}PB_0 u\right)$$

Therefore, we select the adaptive laws as follows:

$$\begin{cases} \dot{\hat{\theta}}_i = \gamma_i E^{\mathrm{T}}PB_0 f_i(x), \qquad \text{for} \quad i = 1, \cdots, q-1; \\ \dot{\hat{\theta}}_q = \gamma_q E^{\mathrm{T}}PB_0 u \end{cases} \tag{5.3.8}$$

Clearly, this results in $\dot{V} = -E^{\mathrm{T}}E$ which implies that the tracking error, its derivatives and the parameter estimates are uniformly bounded and the tracking error converges to zero (by Barbalats Lemma). Although it has not been included in the above analysis, projection would be required to maintain $\theta_q > 0$.

5.3.2 Adaptive Backstepping Example

Backstepping is a recursive constructive nonlinear control technique that allows the designer to derive a stabilizing control law along with its corresponding Lyapunov function ensuring closed loop stability for a class of nonlinear systems in the so-called Strict Feedback or Triangular systems by a step wise manner.

In each step an element of the state vector acts as a virtual control signal for the error dynamics defined at the current steps. Then we integrate the next states until we reach the real control signal. For a detailed description about backstepping one may consult the book written by Kristic *et al.*[55] The drawback of backstepping is that the control derivation becomes more complex when the order of the system is increased. Farrell *et al.* proposed a filtered version of Backstepping to solve this problem [38].

In this example we consider the backstepping control procedure for the case where there is an unknown parameter.

Consider the second-order system

$$\begin{cases} \dot{x}_1 = x_2 + \theta_1\varphi_1(x_1) \\ \dot{x}_2 = u \end{cases} \tag{5.3.9}$$

where θ_1 is an unknown parameter and φ_1 is a known function. Defining the parameter estimate as $\hat{\theta}_1(t)$, while $\tilde{\theta}_1 = \hat{\theta}_1(t) - \theta_1$ is the parameter estimation error.

The objective is to design an adaptive nonlinear tracking controller such that $y = x_1$ tracks a desired signal $y_d(t)$.

First Step

Defining the error between the actual and the desired output $z_1 = x_1 - y_d$, its dynamics are given by:

$$\dot{z}_1 = x_2 + \theta_1\varphi_1(x_1) - \dot{y}_d \tag{5.3.10}$$

Using the first Lyapunov function candidate $V_1 = \frac{1}{2}z_1^2$ whose derivative along the solutions of equation (5.3.10) is given by

$$\dot{V}_1 = z_1\dot{z}_1 = z_1(x_2 + \theta_1\varphi_1(x_1) - \dot{y}_d) \tag{5.3.11}$$

and choosing the first fictitious control signal

$$\alpha_1 = -k_1 z_1 - \hat{\theta}_1\varphi_1(x_1) + \dot{y}_d, \qquad \text{with} \quad k_1 > 0 \tag{5.3.12}$$

that the virtual control x_2 must take at this step.

Second Step

Defining the difference between the actual state x_2 and its desired value α_1 as the second error signal $z_2 = x_2 - \alpha_1$.

The augmented error dynamics are then expressed as:

$$\begin{cases} \dot{z}_1 = x_2 + \theta_1\varphi_1(x_1) - \dot{y}_d \\ \dot{z}_2 = u - \frac{\partial\alpha_1}{\partial t} \end{cases} \tag{5.3.13}$$

$$\begin{cases} \dot{z}_1 = (z_2 + \alpha_1) + \theta_1\varphi_1(x_1) - \dot{y}_d \\ \dot{z}_2 = u - \dot{\alpha_1} \end{cases} \tag{5.3.14}$$

$$\begin{cases} \dot{z}_1 = (z_2 - k_1 z_1 - \hat{\theta}_1\varphi_1(x_1) + \dot{y}_d) + \theta_1\varphi_1(x_1) - \dot{y}_d \\ \dot{z}_2 = u - \dot{\alpha_1} \end{cases} \tag{5.3.15}$$

$$\begin{cases} \dot{z}_1 = z_2 - k_1 z_1 - \tilde{\theta}_1\varphi_1(x_1) \\ \dot{z}_2 = u - \dot{\alpha_1} \end{cases} \tag{5.3.16}$$

where

$$\dot{\alpha}_1 = -k_1(\dot{x}_1 - \dot{y}_d) - \dot{\hat{\theta}}_1\varphi_1(x_1) - \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}(x_2 + \theta_1\varphi_1) + \ddot{y}_d \tag{5.3.17}$$

$$\dot{\alpha}_1 = -k_1(x_2 + \theta_1\varphi_1 - \dot{y}_d) - \dot{\hat{\theta}}_1\varphi_1(x_1) - \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}(x_2 + \theta_1\varphi_1) + \ddot{y}_d \tag{5.3.18}$$

Replacing θ_1 by $\hat{\theta}_1 - \tilde{\theta}_1$

$$\dot{\alpha}_1 = -k_1(x_2 + [\hat{\theta}_1 - \tilde{\theta}_1]\varphi_1 - \dot{y}_d) - \dot{\hat{\theta}}_1\varphi_1(x_1) - \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}(x_2 + [\hat{\theta}_1 - \tilde{\theta}_1]\varphi_1) + \ddot{y}_d \tag{5.3.19}$$

Let us define the known term of $\dot{\alpha}_1$ as

$$\alpha_2 = k_1(x_2 + \hat{\theta}_1\varphi_1 - \dot{y}_d) + \dot{\hat{\theta}}_1\varphi_1(x_1) + \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}(x_2 + \hat{\theta}_1\varphi_1) - \ddot{y}_d \tag{5.3.20}$$

So

$$\dot{\alpha}_1 = -\alpha_2 + k_1\tilde{\theta}_1\varphi_1 + \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}\tilde{\theta}_1\varphi_1 \tag{5.3.21}$$

or

$$\dot{\alpha}_1 = -\alpha_2 + \left(k_1 + \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}\right)\tilde{\theta}_1\varphi_1 \tag{5.3.22}$$

$$\begin{cases} \dot{z}_1 = z_2 - k_1 z_1 - \tilde{\theta}_1\varphi_1(x_1) \\ \dot{z}_2 = u + \alpha_2 - \left(k_1 + \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}\right)\tilde{\theta}_1\varphi_1 \end{cases} \tag{5.3.23}$$

Using the second Lyapunov function candidate $V_2 = V_1 + \frac{1}{2\gamma_1}\tilde{\theta}_1^2 + \frac{1}{2}z_2^2$ whose derivative along the solutions of equation (5.3.10) is given by

$$\dot{V}_2 = z_1\dot{z}_1 + \frac{1}{\gamma_1}\tilde{\theta}_1\dot{\tilde{\theta}}_1 + z_2\dot{z}_2 \tag{5.3.24}$$

$$\begin{aligned} \dot{V}_2 &= z_1[z_2 - k_1 z_1 - \tilde{\theta}_1\varphi_1(x_1)] \\ &\quad + \tfrac{1}{\gamma_1}\tilde{\theta}_1\dot{\tilde{\theta}}_1 + z_2\dot{z}_2 \end{aligned} \tag{5.3.25}$$

Using the fact that $\tilde{\theta}_1 = \hat{\theta}_1 - \theta_1$ and $\dot{\tilde{\theta}}_1 = \dot{\hat{\theta}}_1$

$$\begin{aligned} \dot{V}_2 &= z_1\left[z_2 - k_1 z_1 - \tilde{\theta}_1\varphi_1(x_1)\right] \\ &\quad + \tfrac{1}{\gamma_1}\tilde{\theta}_1\dot{\hat{\theta}}_1 + z_2\left[u + \alpha_2 - \left(k_1 + \hat{\theta}_1\tfrac{\partial\varphi_1(x_1)}{\partial x_1}\right)\tilde{\theta}_1\varphi_1\right] \end{aligned} \tag{5.3.26}$$

This suggests the selection of the following control law :

$$u = -\alpha_2 - k_2 z_2 - z_1 = -k_1(x_2 + \hat{\theta}_1\varphi_1 - \dot{y}_d) - \dot{\hat{\theta}}_1\varphi_1(x_1) - \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}(x_2 + \hat{\theta}_1\varphi_1) + \ddot{y}_d - k_2 z_2 - z_1 \quad (5.3.27)$$

Substituting (5.3.27) in (5.3.30) yields :

$$\begin{aligned} \dot{V}_2 \; &= z_1\left[-z_2 - k_1 z_1 - \tilde{\theta}_1\varphi_1(x_1)\right] \\ &\quad + \tfrac{1}{\gamma_1}\tilde{\theta}_1\dot{\hat{\theta}}_1 + z_2\left[-k_2 z_2 + z_1 - \left(k_1 + \hat{\theta}_1\tfrac{\partial\varphi_1(x_1)}{\partial x_1}\right)\tilde{\theta}_1\varphi_1\right] \end{aligned} \quad (5.3.28)$$

$$\dot{V}_2 \; = -k_1 z_1^2 - k_2 z_2^2 + \tfrac{1}{\gamma_1}\tilde{\theta}_1\dot{\hat{\theta}}_1 - \left[z_1 + z_2\left(k_1 + \hat{\theta}_1\tfrac{\partial\varphi_1(x_1)}{\partial x_1}\right)\right]\tilde{\theta}_1\varphi_1(x_1) \quad (5.3.29)$$

$$\dot{V}_2 \; = -k_1 z_1^2 - k_2 z_2^2 + \tfrac{1}{\gamma_1}\tilde{\theta}_1\left[\dot{\hat{\theta}}_1 - \gamma\varphi_1(x_1)\left[z_1 + z_2\left(k_1 + \hat{\theta}_1\tfrac{\partial\varphi_1(x_1)}{\partial x_1}\right)\right]\right] \quad (5.3.30)$$

Based on the above derivative of the Lyapunov function, we select the update law for

$$\dot{\hat{\theta}}_1 = \gamma\varphi_1(x_1)\left[z_1 + z_2\left(k_1 + \hat{\theta}_1\frac{\partial\varphi_1(x_1)}{\partial x_1}\right)\right] \quad (5.3.31)$$

5.4 Conclusion

In addition to introducing a few of the dominant nonlinear control system design methodologies, this chapter has reviewed methods used to achieve robustness to nonlinear model error and discussed situations in which online approximation might be useful for improving such robustness and tracking performance. As discussed earlier, online approximation can be achieved only over a compact set denoted by $\mathcal{D}$. Within $\mathcal{D}$, due to the use of the adaptive approximator, the nonlinear model errors should be small. Outside of $\mathcal{D}$, the nonlinear model errors may still be large. Therefore, $\mathcal{D}$ should be defined to contain the set of desired system trajectories. For this reason, the set $\mathcal{D}$ is often referred to as the operating envelope. An important issue in the design of an adaptive approximation based control system is the design of mechanisms to ensure that, for any initial conditions, the system state converges to and stays within the operating envelope $\mathcal{D}$. In order to prevent the state trajectories from leaving the region $\mathcal{D}$, some bound (possibly state-dependent) on the unknown function will be required. In this chapter, we saw that such bounds were also required for the use of sliding mode control, Lyapunov redesign method and adaptive bounding control.

Chapter 6

Function Approximation Augmented Control of Uncertain Nonlinear Systems

6.1 Introduction

This chapter is devoted to present the main contribution of my research. Starting from the statement that the best performance a control law can achieve is based mainly on the starting model of the system to be regulated. How can we get a better performance with a poor model? This is mainly the subject of my thesis. My objective is to combine nonlinear adaptive and robust control using different approximation structures to get a better performance with minimum knowledge of the system to be controlled.

In this chapter we will introduce the mathematical formulation problem, then the control strategy based on backstepping will be highlighted in section 6.3. Section 6.4 is devoted to the proposed control strategy where the main contribution is mentioned. The application of this method to an ABS system will be detailed in section 6.5. Finally, we end up with some concluding remarks in section 6.6.

6.2 Problem Formulation and Assumptions

Given a Single Input Single Output Nonlinear System described by the following state equation:

$$\begin{cases} \dot{\xi} = f(\xi, u) \\ y = g(\xi) \end{cases} \tag{6.2.1}$$

where ξ is the state vector with dimension n and $u, y \in \Re$ are the system input and output respectively. f, g are unknown smooth functions of appropriate dimension. The objective is to design a controller for the above system so that the output y tracks a given trajectory y_{ref} with bounded tracking error.

6.3 Backstepping Control Design

Assume that there exists a nonlinear nonsingular mapping that transforms system (6.2.1) to the triangular strict feedback form :

$$\begin{cases} \dot{x}_1 = x_2 + \varphi_1(x_1) \\ \dot{x}_2 = x_3 + \varphi_2(x_1, x_2) \\ \vdots \\ \dot{x}_{n-1} = x_n + \varphi_{n-1}(x_1, x_2, \ldots, x_{n-1}) \\ \dot{x}_n = \beta(x)u + \varphi_n(x) \\ y = x_1 \end{cases} \tag{6.3.1}$$

This assumption is always applicable for electromechanical systems.

Backstepping is a nonlinear control synthesis method in which a control Lyapunov function is derived in a stepwise manner ensuring the stability of the nonlinear system [55]. It can be applied to derive a control law that stabilizes the transformed system (6.3.1) given that all the states are available for feedback and the nonlinear terms are known with high precision. At each step we construct a control Lyapunov function using a pseudo control signal from the state vector until we reach the last step where the actual stabilizing control is derived along its associated Lyapunov function ensuring the stability of the whole system.

Our idea to overcome the problem of uncertainty in the nonlinear terms is based on an approximated model of the plant at hand and we proceed in applying the steps of backstepping as follows:

6.3.1 Step 1

Let $z_1 = x_1 - y_{ref}$ be the tracking error, its dynamics is governed by:

$$\dot{z}_1 = x_2 + \varphi_1(x_1) - \dot{y}_{ref} \tag{6.3.2}$$

Define the Lyapunov function for the system z_1 as $V_1 = \frac{1}{2}z_1^2$, the derivative of V_1 along the tracking error dynamics (6.3.2) is given by:

$$\dot{V}_1 = z_1[x_2 + \varphi_1(x_1) - \dot{y}_{ref}] \tag{6.3.3}$$

To render it negative definite, we choose the fictitious control $\alpha_1(x_1) = -c_1 z_1 - \varphi_1(x_1) + \dot{y}_{ref}$, we get $\dot{V}_1 = -c_1 z_1^2$

6.3.2 Step 2

Define z_2 as the difference between the desired value of x_2 and its actual value $z_2 = x_2 - \alpha_1(x_1)$, and $\gamma_2(x_1, x_2) = \frac{\partial \alpha_1}{\partial x_1}[x_2 + \varphi_1]$, the augmented error dynamics will be expressed as:

$$\begin{cases} \dot{z}_1 = -c_1 z_1 + z_2 \\ \dot{z}_2 = x_3 + \varphi_2(x_1, x_2) + c_1[-c_1 z_1 + z_2] \\ \qquad +\gamma_2(x_1, x_2) - \ddot{y}_{ref} \end{cases} \tag{6.3.4}$$

Choosing the candidate Lyapunov function $V_2 = \frac{1}{2}\left[z_1^2 + z_2^2\right]$, its derivative along the augmented error dynamics (6.3.4) is given by

$$\begin{aligned} \dot{V}_2 = -c_1 z_1^2 + z_2\{& z_1 + x_3 + \varphi_2(x_1, x_2) \\ & +c_1[-c_1 z_1 + z_2] + \gamma_1(x_1, x_2) - \ddot{y}_{ref}\} \end{aligned} \tag{6.3.5}$$

To make $\dot{V}_2$ negative definite, the following fictitious control is chosen

$$\begin{aligned} \alpha_2(x_1, x_2) = & -c_2 z_2 - z_1 - c_1\left[-c_1 z_1 + z_2\right] \\ & -\varphi_2(x_1, x_2) - \gamma_1(x_1, x_2) + \ddot{y}_{ref} \end{aligned} \tag{6.3.6}$$

which results in $\dot{V}_2 = -c_1 z_1^2 - c_2 z_2^2$

6.3.3 Step $(n-1)$

Let $z_{n-1} = x_{n-1} - \alpha_{n-2}(x_1, \ldots, x_{n-1})$, the augmented tracking error dynamics become:

$$\begin{cases} \dot{z}_1 = -c_1 z_1 + z_2 \\ \dot{z}_2 = -z_1 - c_2 z_2 + z_3 \\ \vdots \\ \dot{z}_{n-2} = -z_{n-3} - c_{n-2} z_{n-2} + z_{n-1} \\ \dot{z}_{n-1} = x_n + \varphi_{n-1}(x_1, \ldots, x_{n-1}) \\ \quad +C_{n-1} Z_{n-1}^T + \gamma_{n-1}(x_1, \ldots, x_{n-1}) - y_{ref}^{(n-1)} \end{cases} \tag{6.3.7}$$

where $\begin{cases} \gamma_1 = \varphi_1, \\ \gamma_{k-1} = \sum_{i=1}^{k-2} \frac{\partial \gamma_{k-2}}{\partial x_i} [x_{i+1} + \varphi_i] \\ {\scriptstyle k=2,3,\cdots,n} \end{cases}$.

and $C_{n-1}Z_{n-1}$ is used to simplify writing by groupping the linear combination of z_i' s.

Choosing the candidate Lyapunov function:

$V_{n-1} = \frac{1}{2} \sum_{i=1}^{n-1} z_i^2,$

its derivative along the error dynamics in (6.3.7) is

$$\dot{V}_{n-1} = -\sum_{i=1}^{n-2} c_i z_i^2 + z_{n-1}z_{n-2} \\ +z_{n-1} \left[x_n + \varphi_{n-1} + C_{n-1}Z_{n-1}^T + \gamma_{n-1} - y_{ref}^{(n-1)} \right]$$

By choosing

$$\alpha_{n-1} = -c_{n-1}z_{n-1} - \varphi_{n-1}(x_1, \ldots, x_{n-1}) - z_{n-2} \\ -C_{n-1}Z_{n-1}^T - \gamma_{n-1}(x_1, \ldots, x_{n-1}) + y_{ref}^{(n-1)}$$

$\dot{V}_{n-1}$ is forced to be negative and it is given by

$\dot{V}_{n-1} = -\sum_{i=1}^{n-1} c_i z_i^2.$

6.3.4 Step (n)

This is the last step where the control signal appears. By defining $z_n = x_n - \alpha_{n-1}$, the error dynamics take the following form:

$$\begin{cases} \dot{z}_1 = -c_1 z_1 + z_2 \\ \dot{z}_2 = -z_1 - c_2 z_2 + z_3 \\ \vdots \\ \dot{z}_{n-2} = -z_{n-3} - c_{n-2}z_{n-2} + z_{n-1} \\ \dot{z}_{n-1} = -z_{n-2} - c_{n-1}z_{n-1} + z_n \\ \dot{z}_n = \beta(x)u + \varphi_n(x) + C_n Z_n^T \\ \qquad +\gamma_n(x) - y_{ref}^{(n)} \end{cases} \tag{6.3.8}$$

where $\gamma_n(x) = \sum_{i=1}^{n-1} \frac{\partial \gamma_{n-1}}{\partial x_i} [x_{i+1} + \varphi_{i+1}]$ and, $C_n Z_n^T$ is a linear combination of the components of the augmented error vector z.

In this design step we can augment the above Lyapunov function by a quadratic term $\frac{1}{2} z_n^2$ then we get the stabilizing control law to make it negative definite as:

$$u = \frac{\alpha_n(x)}{\beta(x)} \tag{6.3.9}$$

where $\alpha_n = -c_n z_n - \varphi_n(x) - z_{n-1} - C_n Z_n^T - \gamma_n(x) + y_{ref}^{(n)}$
which yields to $\dot{V}_n = -\sum_{i=1}^{n} c_i z_i^2$

The resultant linear system is stable and has the following structure:

$$\dot{z} = \begin{bmatrix} -c_1 & 1 & \cdots & 0 & 0 \\ -1 & -c_2 & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & 1 & 0 \\ \vdots & & -1 & -c_{n-1} & 1 \\ 0 & \cdots & 0 & -1 & -c_n \end{bmatrix} z \tag{6.3.10}$$

and the all the error signals z_i tend exponentially to zero.

6.4 The Proposed Controller

The control law given in equation (6.3.9) can be implemented if these conditions are fulfilled [10]:

- All the state are accessible for feedback,
- All the nonlinear functions are precisely known, and
- $\beta(x) \neq 0, \forall x \in \Re^n$

To relax the first two conditions we can use the same idea as in [59] by introducing neural networks for their approximation ability to compensate for the uncertain terms tgat appear in the last step of backstepping, and continue with another step to derive the adaptation laws for neural network weights.

However the control is not exactly known, and we assume that its best estimate is given by: $\frac{\hat{\alpha}(x_m,\hat{z},y_{ref}^{(n)})}{\hat{\beta}(x_m)}$, which depends only on the measured states x_m, the n^{th} derivative of the output of a reference signal to be tracked, and $\hat{z}$, an estimate of the augmented tracking error dynamics z.

Defining $u = \frac{\alpha(x)}{\beta(x)} + u_{ad}$, an implementable augmented control signal which is composed of the backstepping control signal plus an adaptive term designed to cancel the effect of the unknown terms. Substituting on u in equation (6.3.9), the system dynamics can be expressed as:

$$\dot{z} = C_{bs} z + b[u_{ad} - \triangle] \tag{6.4.1}$$

where $C_{bs} = \begin{bmatrix} -c_1 & 1 & \cdots & 0 & 0 \\ -1 & -c_2 & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & 1 & 0 \\ \vdots & & -1 & -c_{n-1} & 1 \\ 0 & \cdots & 0 & -1 & -c_n \end{bmatrix}$,

$b = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T$, and $\triangle$ represents the nonlinear unknown terms to be canceled by u_{ad}.

This structure looks like the form obtained by a feedback linearization scheme [63, 64], however it is more suitable because we will get the adaptation law of the neural network weights in a constructive way by augmenting the existing Lyapunov function.

6.4.1 Neural Network compensator

Usually in the literature, neural networks are used in function approximation in the modeling phase [93], however we will use them in controller design given some conditions are fulfilled. However here we can use either a simple feed-forward neural network of two layers as shown on figure 6.1 to approximate the error term $\triangle$ or any other type of parameterized function approximator [11].

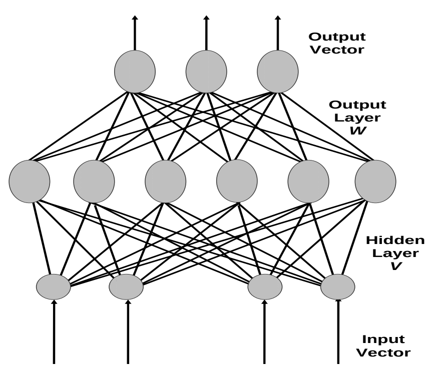

Figure 6.1: A Neural Network with two layers

The approximated error signal can be expressed as:

$$\triangle = W^{\mathrm{T}}\Phi(V, \mu) + \varepsilon(\mu)\ \forall \mu \in D \tag{6.4.2}$$

where

$V \in D_V \subset \Re^{N_1}$ is the hidden layer weight vector

$W \in D_W \subset \Re^{N_2}$ is the output layer weight vector

Φ is a set of basis functions

μ is the network input vector, and

ε is the neural network reconstruction error

Assuming that the approximation reconstruction error is bounded on some domain D by $\|\varepsilon(\mu)\| \leq \epsilon^*, \ \forall \mu \in D$.

The weight vectors are assumed to be bounded $\|W\| \leq W^*$ and their adjustment can be done online. However the whole controller implementation will be difficult as the number of the parameters increases for complex problems. So the first layer parameters can be adjusted offline in such a way $\Phi(\mathbf{V}, \mu)$ is a basis [93] whereas the adaptation law for the second layer is included in the controller design

The adaptive control signal u_{ad} is designed to cancel the unknown nonlinear terms $\triangle$.

$$u_{ad} = \hat{W}^{\mathrm{T}} \Phi(\mathbf{V}, \mu) \tag{6.4.3}$$

where $\hat{W}$ are the estimates of W weights, the initial values of these estimates $\mathbf{W}_0$ can simply be set to zero, while μ is an implementable input vector to the NN defined as:

$\mu = \begin{bmatrix} z^{\mathrm{T}} & \bar{y}_d^{\mathrm{T}} & \bar{u}_d^{\mathrm{T}} & 1 \end{bmatrix}$ with $\bar{y}_d$, and $\bar{u}_d$ are vectors of delayed values of the output and control signals respectively [63]. The tracking error terms z must be available for the implementation of the pseudo-control signal $\hat{\alpha}$.

6.4.2 Neural network weights adaptation

Let $\tilde{W} = \hat{W} - W$, the generalized error vector $E = \begin{bmatrix} z^{\mathbf{T}} & \tilde{W}^{\mathbf{T}} \end{bmatrix}^{\mathrm{T}}$ and defining the augmented Lyapunov function

$$V_a(E) = \frac{1}{2}\left[z^{\mathrm{T}} z + \tilde{W}^{\mathrm{T}} F^{-1} \tilde{W})\right] \tag{6.4.4}$$

where $F > 0$ is an adaptation gain.

Differentiating V_a with respect to E gives

$$\begin{aligned} \dot{V}_a = & -\sum_{i=1}^{n} c_i z_i^2 + \tilde{W}^{\mathrm{T}} F^{-1} \dot{\tilde{W}} \\ & + z_n[\tilde{W}^{\mathrm{T}} \Phi(\mathrm{V}, \mu) - \varepsilon(\mu)] \end{aligned} \tag{6.4.5}$$

Using the fact that $\tilde{W} = \hat{W} - W$, equation (6.4.5) can be written as

$$\begin{aligned}\dot{V}_a = &- \sum_{i=1}^{n} c_i z_i^2 + \tilde{W}^{\mathrm{T}} F^{-1} \dot{\hat{W}} \\ &+ z_n [\tilde{W}^{\mathrm{T}} \Phi(\mathrm{V}, \mu) - \varepsilon(\mu)]\end{aligned} \tag{6.4.6}$$

Hence, we can derive this adaptive law for the network parameters W which will make the derivative augmented Lyapunov function to be negative definite in some domain.

$$\dot{\hat{W}} = -F \left[\Phi(\mathrm{V}, \mu) z_n + 2G(\hat{W} - W_0) \right] \tag{6.4.7}$$

Substituting (6.5.30) in (6.4.6) yields

$$\begin{aligned}\dot{V}_a = &- \sum_{i=1}^{n} c_i z_i^2 - \tilde{W}^{\mathrm{T}} \left[\Phi(\mathrm{V}, \mu) z_n + 2G(\hat{W} - W_0) \right] \\ &+ z_n [\tilde{W}^{\mathrm{T}} \Phi(\mathrm{V}, \mu) - \varepsilon(\mu)]\end{aligned} \tag{6.4.8}$$

or

$$\dot{V}_a = - \sum_{i=1}^{n} c_i z_i^2 - 2\tilde{W}^{\mathrm{T}} G(\hat{W} - W_0) - z_n \varepsilon(\mu) \tag{6.4.9}$$

Using the fact that the neural networks weight vector is bounded, the derivative of the augmented Lyapunov candidate can be upper bounded as

$$\begin{aligned}\dot{V}_a \leq &- \sum_{i=1}^{n} c_i z_i^2 + |z_n| \epsilon^* \\ &- G\|\tilde{W}\|^2 - G\|\hat{W} - W_0\|^2 + G\|W - W_0\|^2\end{aligned} \tag{6.4.10}$$

Putting $\underline{N} = \sum_{i=1}^{n-1} c_i z_i^2 + G\|\hat{W} - W_0\|^2$ yields

$$\begin{aligned}\dot{V}_a \leq &-\underline{N} - c_n z_n^2 + |z_n| \epsilon^* \\ &- G\|\tilde{W}\|^2 + G\|W - W_0\|^2\end{aligned} \tag{6.4.11}$$

Completing the squares using $\epsilon^* |z_n| = -\frac{1}{2} \left(\epsilon^* - |z_n| \right)^2 + \frac{1}{2} {\epsilon^*}^2 + \frac{1}{2} z_n^2$, we get

$$\begin{aligned}\dot{V}_a \leq &-\underline{N} - c_n z_n^2 - G\|\tilde{W}\|^2 + G\|W - W_0\|^2 \\ &- \tfrac{1}{2} \left(\epsilon^* - |z_n| \right)^2 + \tfrac{1}{2} {\epsilon^*}^2 + \tfrac{1}{2} z_n^2\end{aligned} \tag{6.4.12}$$

Further, it can be written as

$$\begin{aligned}\dot{V}_a \leq &-\underline{N} - (c_n - \tfrac{1}{2}) z_n^2 - G\|\tilde{W}\|^2 - \tfrac{1}{2} \left(\epsilon^* - |z_n| \right)^2 \\ &+ G\|W - W_0\|^2 + \tfrac{1}{2} {\epsilon^*}^2\end{aligned} \tag{6.4.13}$$

The following conditions

$$
\begin{array}{l}
|\tilde{z}_n| > \sqrt{\frac{2\epsilon^{*2}+2G\|W-W_0\|^2}{2c_n-1}} \\
\|\tilde{W}\|^2 > \sqrt{\frac{\epsilon^{*2}+G\|W-W_0\|^2}{G}}
\end{array}
\tag{6.4.14}
$$

with $c_n > \frac{1}{2}$, and $G > 0$ ensures that $\dot{V}_a \leq -\underline{N}$. Which is negative definite in some domain defined by conditions in (6.4.14).

The resulting controller looks like the controller proposed in [64] however here it is obtained in a constructive manner for a triangular system without adding an SPR condition. It can be implemented and we have a freedom in selecting the linear controller gains c_i to achieve a desirable performance.

It must be emphasized that Takagi-Segino Fuzzy systems are mathematically almost the same as RBF Neural Networks. They can also be used as adaptive function approximators. The off-line training becomes the selection of the fuzzy rules (membership functions, shapes,..). Whereas the On-line adaptive law is the same as presented for RBF neural networks.

6.5 Application to a Laboratory ABS System

6.5.1 ABS System Modeling

The anti-lock braking system (ABS) in cars were implemented in the late 70s [1]. The main objective of the control system is to prevent wheel-lock while braking. Usually we are interested in the tire slip on each of the four wheels in the car. Only longitudinal motion is considered. The laboratory setup of ABS available at CREA, Centre de Robotique, d'Electrotechnique et d'Automatique d'Amiens) [56], that represents an anti-lock brake system is shown on Figure 6.2, and is schematized on Figure 6.3.

In this simplified ABS system, the lower car-road wheel is animating relative road motion and the upper car wheel permanently remaining in a rolling contact with the lower wheel. The wheel mounted to the balance lever is equipped in a tyre. The car-road wheel has a smooth surface which can be covered by a given material to animate a surface of the road.

There are three torques acting on the upper wheel: the braking torque M_1 , the friction torque in the upper bearing and the friction torque among the wheels. There are two torques acting on the lower wheel: the friction torque in the lower bearing and the friction torque among the wheels. Besides these we have two forces acting on the lower wheel: the gravity force of the upper wheel and

Figure 6.2: ABS Laboratory set-up

the pressing force of the shock absorber.

Denoting the system parameters as:

- $x_1 = \omega_1$, the angular velocity of the upper wheel with radius r_1, and moment of inertia J_1
- $x_2 = \omega_2$, the angular velocity of the lower wheel with radius r_2, and moment of inertia J_2
- λ , the slip which is the relative difference of the wheel velocities,
- M_1 , the brake torque, it is the input signal of the model,
- μ , the friction coefficient between the upper and lower wheels,
- F_n , the normal force - the upper wheel acting on the lower wheel.

The equations of motion of the system can be expressed

$$J_1\dot{x}_1 = F_n r_1 \mu(\lambda) - d_1 x_1 - M_{10} - M_1 \tag{6.5.1}$$

$$J_2\dot{x}_1 = -F_n r_2 \mu(\lambda) - d_2 x_2 - M_{20} \tag{6.5.2}$$

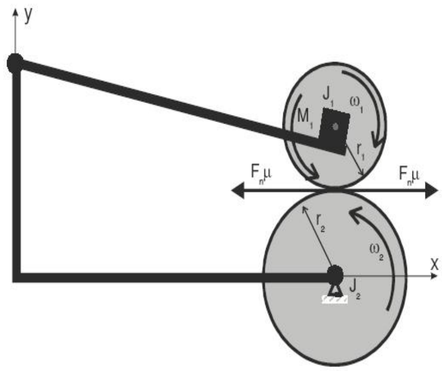

Figure 6.3: Schematic diagram of ABS

or

$$J_1\dot{x}_1 = F_1 - M_1 \tag{6.5.3}$$

$$J_2\dot{x}_1 = F_2 \tag{6.5.4}$$

using the auxiliary variables F_1, F_2

$$F_1 = F_n r_1 \mu(\lambda) - d_1 x_1 - M_{10} \tag{6.5.5}$$

$$F_2 = -F_n r_2 \mu(\lambda) - d_2 x_2 - M_{20} \tag{6.5.6}$$

and assuming that there is a derived model for friction coefficient based on the following model:

$$\mu(\lambda) = \frac{w_4 \lambda^p}{a + \lambda^p} + w_3\lambda^3 + w_2\lambda^2 + w_1\lambda \tag{6.5.7}$$

where w_i are model parameters. The driving system of the brake is governed by the following equation:

$$\dot{M}_1 = c_{31}(b_1 u + b_2 - M_1) \tag{6.5.8}$$

The dynamics of the driving system are very fast compared to those of the mechanical system, in the rest of this chapter we will consider it as a control gain, so we can write:

$$M_1 = K_u u \tag{6.5.9}$$

6.5.2 ABS backstepping control

For ABS laboratory set-up the slip is defined as:

$$\lambda = 1 - \frac{r_1 x_1}{r_2 x_2} \tag{6.5.10}$$

The control objective is essentially to control the value of the slip λ to a given set point λ^* that is either constant or supplied by a higher supervising system. The slip dynamics can be expressed as:

$$\dot{\lambda} = -\frac{r_1}{r_2 x_2}\left[\dot{x}_1 - \frac{x_1}{x_2}\dot{x}_2\right] \tag{6.5.11}$$

Multiplying both sides of equation (6.5.11) by x_2

$$x_2\dot{\lambda} = -\frac{r_1}{r_2}\left[\dot{x}_1 - \frac{x_1}{x_2}\dot{x}_2\right] \tag{6.5.12}$$

Using the fact that $\frac{r_1 x_1}{r_2 x_2} = (1-\lambda)$, we get

$$x_2\dot{\lambda} = -\frac{r_1}{r_2}\dot{x}_1 + (1-\lambda)\dot{x}_2 \tag{6.5.13}$$

Defining the integrated slip error
$z_1 = \int_{h=0}^{t}(\lambda - \lambda^*)dh$, and the slip error
$z_2 = \lambda - \lambda^*$.

Its dynamics take the following form:

$$\begin{cases} \dot{z_1} = z_2 \\ \dot{z_2} = -\frac{r_1}{x_2 r_2}\dot{x}_1 + \frac{1}{x_2}(1-\lambda)\dot{x}_2 - \dot{\lambda}^* \end{cases} \tag{6.5.14}$$

The first step in backstepping considers z_2 as a fictitious control signal that stabilizes the z_1 dynamics. Introducing the first Lyapunov candidate function:

$$V_1 = \frac{1}{2}z_1^2 \tag{6.5.15}$$

its derivative along the error dynamics is expressed as

$$\dot{V}_1 = z_1\dot{z_1} = z_1 z_2 \tag{6.5.16}$$

yields the following fictitious control :

$$z_2^* = -k_1 z_1 \tag{6.5.17}$$

Defining $e_2 = z_2 - z_2^* = z_2 + k_1 z_1$, the error dynamics take the following form

$$\begin{cases} \dot{z_1} = e_2 - k_1 z_1 \\ \dot{e_2} = -\frac{r_1}{x_2 r_2}\dot{x}_1 + \frac{1}{x_2}(1-\lambda)\dot{x_2} - \dot{\lambda}^* \\ \qquad +k_1(e_2 - k_1 z_1) \end{cases} \tag{6.5.18}$$

Substituting for $\dot{x_1}$ and $\dot{x_2}$ in equation (6.5.18)

$$\begin{cases} \dot{z_1} = e_2 - k_1 z_1 \\ \dot{e_2} = -\frac{r_1}{x_2 r_2 J_1}(F_1 - s_1 M_1) \\ \qquad +\frac{1}{x_2}(1-\lambda)\frac{F_2}{J_2} - \dot{\lambda}^* + k_1(e_2 - k_1 z_1) \end{cases} \tag{6.5.19}$$

The second step of backstepping suggests the following Lyapunov function

$$V_2 = \frac{1}{2}(z_1^2 + e_2^2) \tag{6.5.20}$$

Differentiating V_2 along the error dynamics one gets:

$$\begin{aligned} &\dot{V}_2 = -k_1 z_1^2 + e_2\left(-\dot{\lambda}^* + k_1(e_2 - k_1 z_1)\right) \\ &+e_2\left[z_1 - \frac{r_1}{x_2 r_2 J_1}(F_1 - M_1) + \frac{1}{x_2}(1-\lambda)\frac{F_2}{J_2}\right] \end{aligned} \tag{6.5.21}$$

Using the following control signal

$$\begin{aligned} &M_1 = F_1 - \frac{F_2 r_2 J_1}{r_1 J_2}(1-\lambda) \\ &-\frac{x_2 r_2 J_1}{r_1}\left(z_1 - \dot{\lambda}^* + k_1(e_2 - k_1 z_1) + k_2 e_2\right) \end{aligned} \tag{6.5.22}$$

will render $\dot{V}_2$ negative definite that is:

$$\dot{V}_2 = -k_1 z_1^2 - k_2 e_2^2 \tag{6.5.23}$$

6.5.3 ABS Neural Network Augmented Control

The desired braking torque that makes the slip tend to the desired slip can be derived given that all model parameters are precisely known which is not the case due to the estimated friction forces that depend on road conditions so the control signal will be augmented by an adaptive term that compensate for uncertainty.

Let

$$\begin{aligned} &\hat{M}_1 = \hat{F}_1 - \frac{\hat{F}_2 r_2 J_1}{r_1 J_2}(1-\lambda) \\ &-\frac{x_2 r_2 J_1}{r_1}\left(z_1 - \dot{\lambda}^* + k_1(e_2 - k_1 z_1) + k_2 e_2\right) \end{aligned} \tag{6.5.24}$$

The actual control signal u which is the dc voltage applied to the electro-mechanical actuator that will deliver the necessary braking torque is obtained by an inversion of a simplified model that maps the input voltage space to its corresponding delivered braking torque.

$$\begin{aligned} u^* &= K_u^{-1}\left(\hat{F}_1 - \tfrac{\hat{F}_2 r_2 J_1}{r_1 J_2}(1-\lambda)\right) \\ &-\tfrac{K_u^{-1} x_2 r_2 J_1}{r_1}\left(z_1 - \dot{\lambda}^* + k_1(e_2 - k_1 z_1) + k_2 e_2\right) \end{aligned} \tag{6.5.25}$$

This known part of the control signal u^* is augmented by an adaptive element u_a supplied by an artificial neural network to compensate for neglected and unknown terms.

$$u = u^* + u_a \tag{6.5.26}$$

Substituting on u in equation (6.5.19), the system dynamics can be expressed as:

$$\begin{cases} \dot{z_1} = e_2 - k_1 z_1 \\ \dot{e_2} = -z_1 - k_2 e_2 + [u_{ad} - \triangle] \end{cases} \tag{6.5.27}$$

where $\triangle$ represents the nonlinear unknown terms to be canceled by u_{ad}.

The error signal can be expressed as:

$$\triangle = W^{\mathrm{T}}\Phi(V) + \varepsilon \tag{6.5.28}$$

The adaptive control signal u_a is designed to cancel the unknown nonlinear terms $\triangle$.

$$u_a = \hat{W}^{\mathrm{T}}\Phi(\mathbf{V}, e_1, z_2, x_1, x_2) \tag{6.5.29}$$

The adaptive law for the network parameters W is expressed as

$$\dot{\hat{W}} = -F\left[\Phi(\mathbf{V}, e_1, z_2, x_1, x_2) z_2 + 2G(\hat{W} - W_0)\right] \tag{6.5.30}$$

6.5.4 Simulation and Experimental results

Before we proceed for testing the proposed control method for the ABS laboratory set-up, we have conducted simulations on a Simulink model for the system. In this simulation we assumed that the wheels are initially rotating at a given speed then at t_0 a braking action is demanded. The two figures (6.5, 6.5) illustrate the simulation results of simple relay control and the proposed scheme.

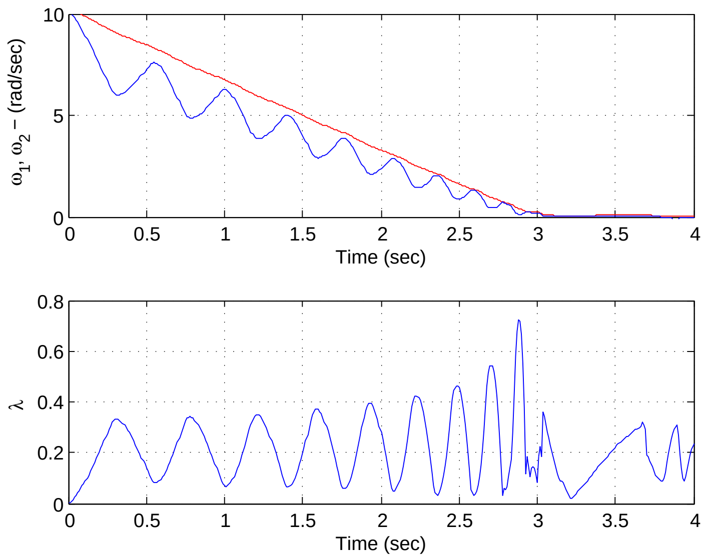

Figure 6.4: Relay Control Simulation Results

It is clearly evident that the proposed controller reached the desired slip (about or smaller 20%) whereas the relay control slip is oscillating (10-40%) which is not desired.

Before starting the implementation of the control algorithm, some experiments are taken to get the appropriate model parameters:

Figure 6.6 shows the angular speeds of both wheels where the control brake is les than 20% of its maximum value showing that the friction forces are more important than that of the braking action.

However when the brake action increased above 20% the braking force becomes more effective and the sleep is becoming more important when the braking force is increased as illustrated by figures 6.7,6.8.

We have applied the proposed control law for the ABS Laboratory system using MATLAB/SIMULINK Real time for windows Target environment. The Neural Network is implemented by a Simulink C-S Function which is complied by Matlab C++ compiler then it is integrated in Simulink as an S-function.

The experiment consists on accelerating the two wheels until we reach a given angular speed,

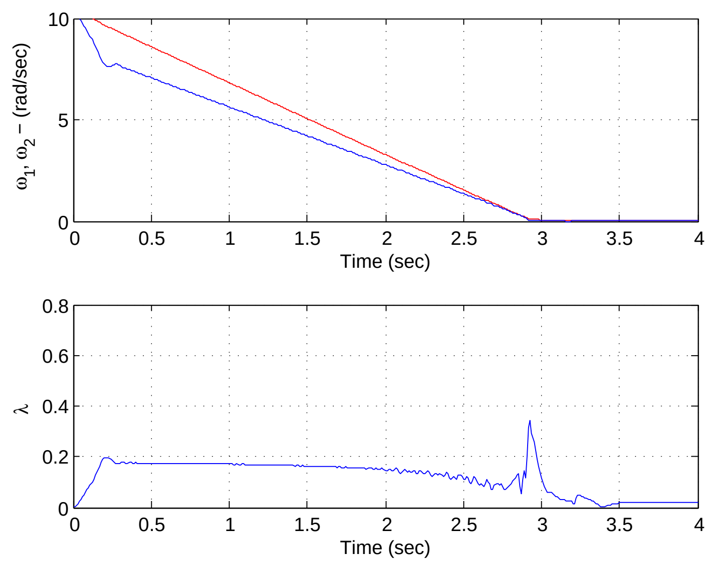

Figure 6.5: Proposed Controller Simulation Results

then a controlled braking action will be taken in such a way to prevent wheel lock.

The same experiment was conducted by achieving different speed levels with different contact conditions.

First, we have applied a relay swishing control as described in the ABS laboratory manual ([1]) to compare it with the results obtained using backstepping augmented controller.

Figure 6.9 presents the results of relay control with high slip ($> 40\%$). Whereas (Figure 6.10 shows the obtained satisfactory results for the normal case where no material is added to change the contact condition between the two wheels simulating the normal road conditions. Figure 6.14 shows the performance of the controller when the wheel contact condition is changed by adding some product like a resin between the rotating wheels to simulate wet road condition where the slip is important.

In all cases the controller achieves an acceptable performance and the adaptive elements adapts for the changing environment.

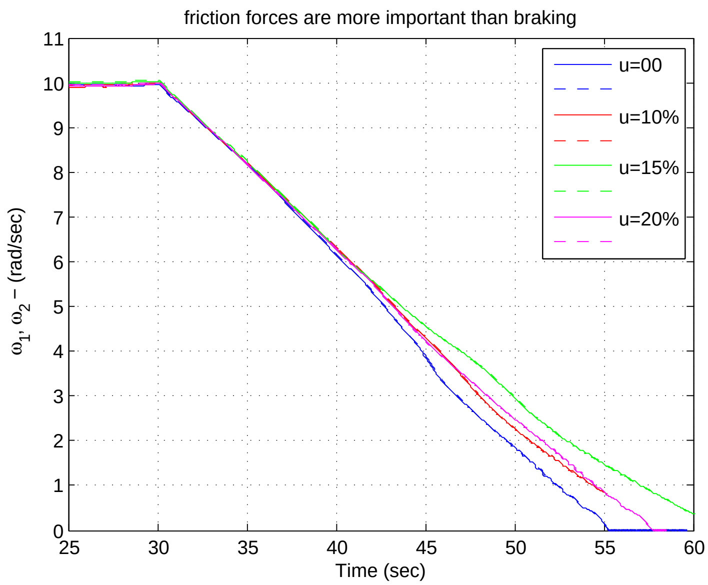

Figure 6.6: Open Loop $(0 - 20\%)$ Braking applied for average speed

6.6 Conclusion

A new adaptive nonlinear controller is presented in this thesis to address a class of uncertain nonlinear systems where the stability is ensured for the closed loop system and all the errors (tracking, observer, and neural network weights) tend asymptotically to zero. The RBF center selection is done using orthogonal least squares structure selection method to minimize the number of parameters of the resulting controller. We have mentioned only Neural Networks in this chapter for function approximation. Finally we have applied this method to an ABS system to show the design choices and the achieved performance although we have assumed a partly known model of the system to be controlled.

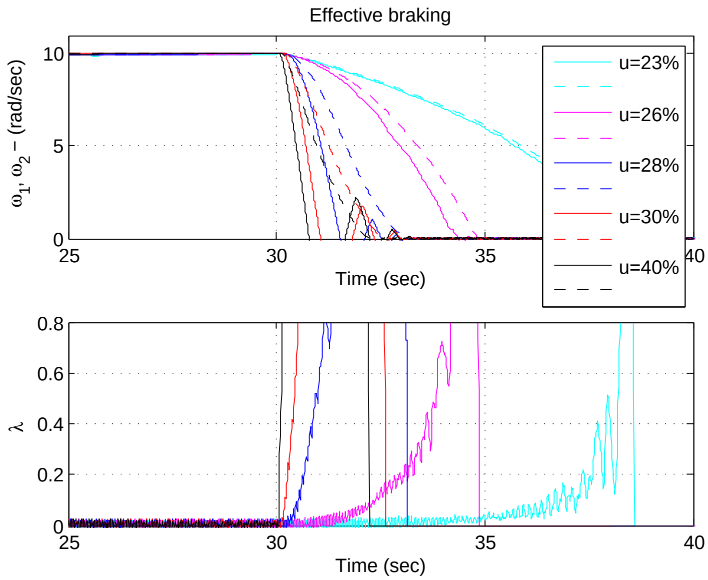

Figure 6.7: Open Loop $(20 - 40\%)$ Braking applied for average speed

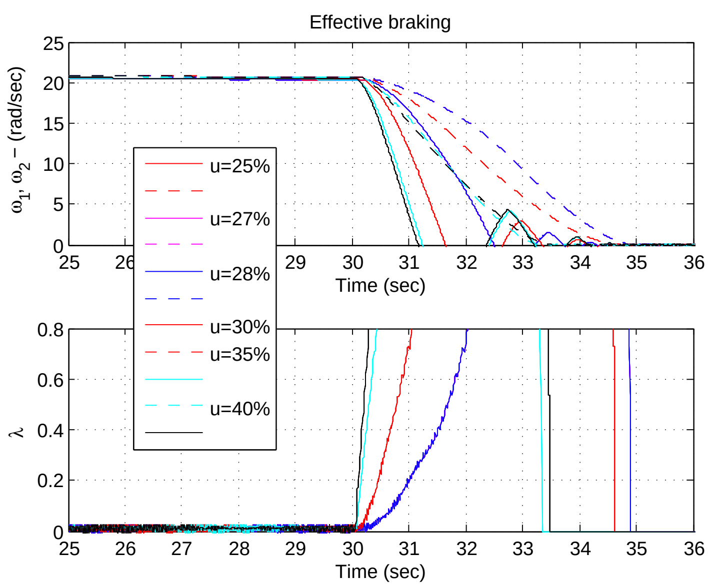

Figure 6.8: Open Loop $(00 - 40\%)$ Braking applied for High speed

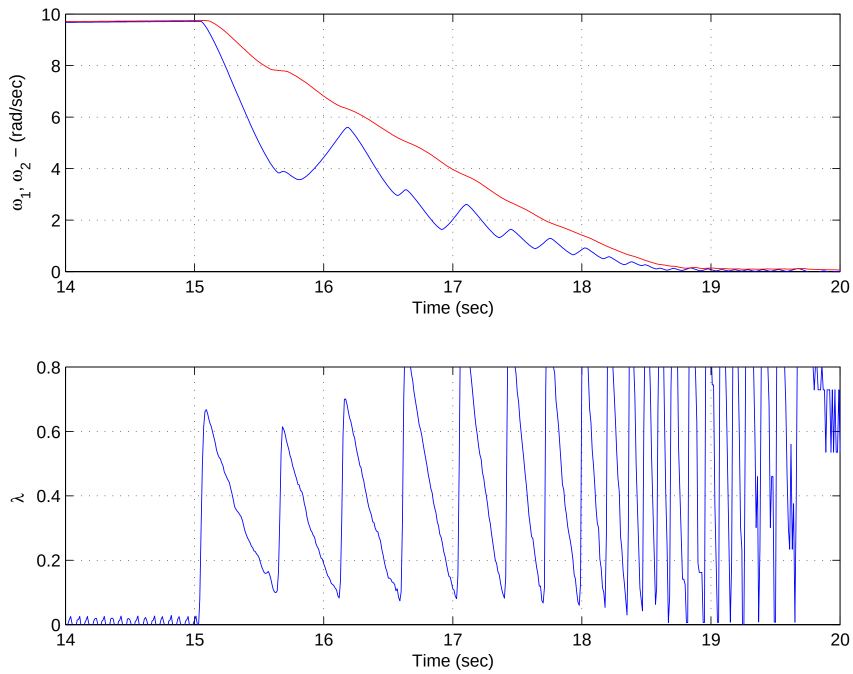

Figure 6.9: Relay Control Experimental Results

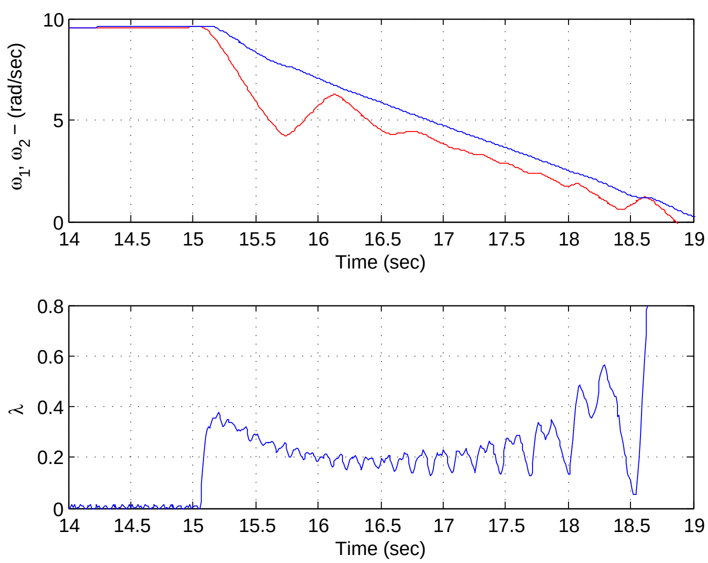

Figure 6.10: Proposed Controller Experimental Results

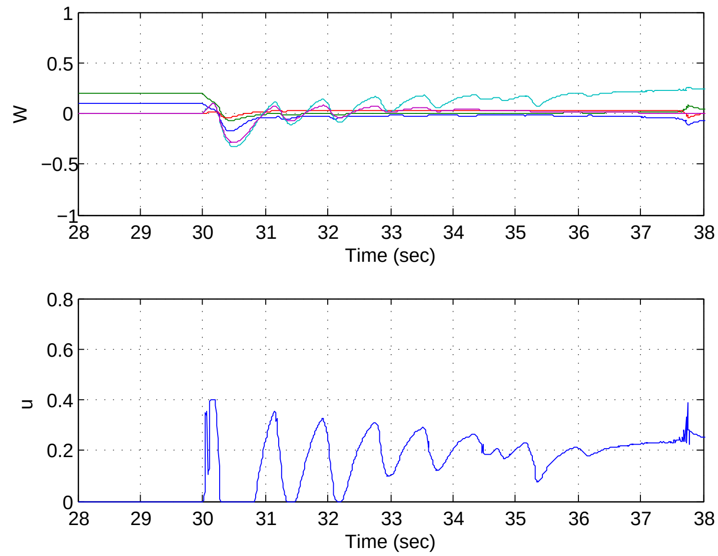

Figure 6.11: Neural Network Weight History and Control Signal

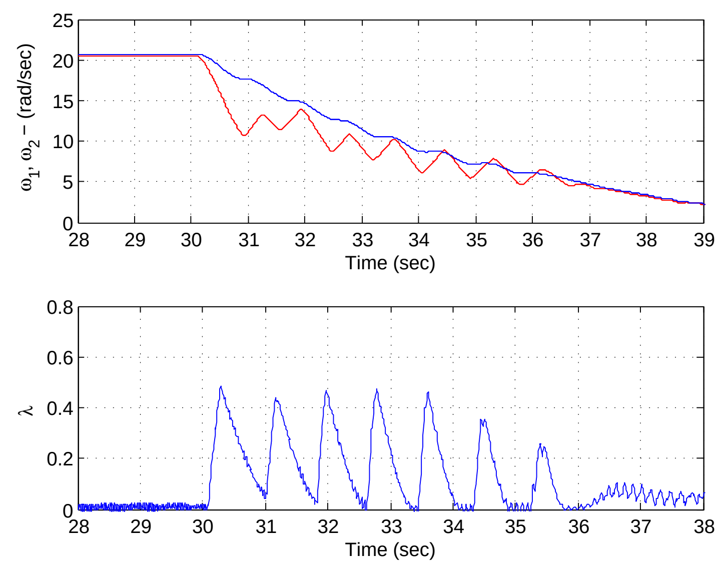

Figure 6.12: Proposed Controller Experimental Results for Highest Speed

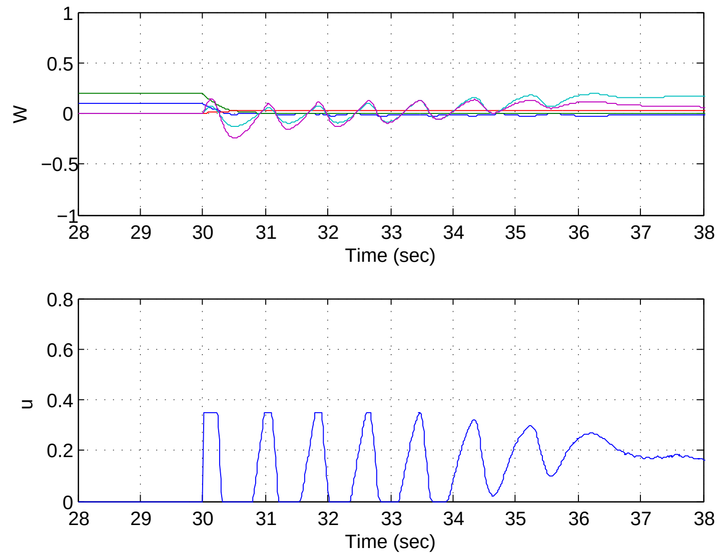

Figure 6.13: Neural Network Weight History and Control Signal

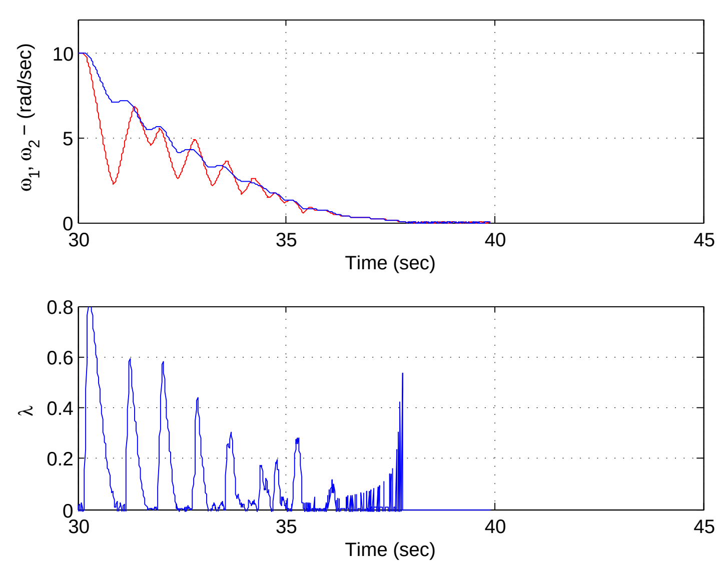

Figure 6.14: Proposed Controller Experimental Results for wet road condition

Chapter 7

Application To Induction Machine

7.1 Introduction

In the first part of my thesis, I have presented some tools that help the control engineer to design control laws that solve the problem with systems that have not a well known model. Development and growth of emerging technologies suggests new actuation and electromechanical devices every year for industry applications in different fields and in the other hand the continuous improvement and invention of new computer and hardware device that ease the implementation of complex control algorithms motivates research to gain from this and enhance the control laws to get more precision and comfort.

In this chapter we will apply function approximation based control for an electromechanical system which is the induction machine which is widely used in industry.

The induction motor is an electromechanical device that converts an electrical energy to a mechanical energy [11, 12] . It is one of the most widely used actuators for industrial applications today mostly because of its low cost, ruggedness and reliability. The important parameters of an induction motor that are susceptible to uncertainties on operating conditions are the load torque and the rotor resistance. The variation of rotor resistance of the induction motor can even be a 100% during operation due to rotor heating. In the case of most loads with a widely varying speed of operation, like in electric vehicle, pumps, blowers the torque varies as a function of time instead of being constant or changing in steps.

The control of induction motors is a difficult task. The system is nonlinear, coupled and multivariable having two control inputs (stator voltages) and two output variables (rotor speed and flux modulus), required to track the desired reference trajectories. To ease difficulties, a Field-Orientation Control [24] is the most popular and widely used approach. Due to the high costs and drawbacks of accurate flux sensors an approach to estimate the flux by flux estimators is more effective. This method is called the Indirect Field-Orientation control.

The advances in nonlinear control theory have caused a notable impact on the control of induction motor. With adaptive backstepping design technique researchers were able to successfully design controllers which achieved global stabilization in the presence of uncertain parameters. Using this design methodology, the construction of both the feedback control laws and associated Lyapunov functions is systematic. Strong properties of global or local stability are built into the nonlinear system with uncertain parameters in a number of steps which is never higher than the system order. Many recent works have focussed on compensation of rotor resistance and load torque effects while also removing the need for rotor flux measurements. In [65] a rotor flux calculation method is used to achieve control under rotor resistance and torque uncertainty. This method requires the motor to be initially at rest. In [26] a rotor flux estimation method was designed to develop an adaptive controller to compensate for rotor resistance uncertainty and uncertainty in mechanical subsystem, which exhibited a singularity when the magnitude of the estimated rotor flux was equal to zero. In [72] a global adaptive controller is designed to achieve speed control in the case of rotor resistance uncertainty and constant unknown load torque. In [30] an adaptive backstepping controller for induction motors which is adaptive with time-varying load torque and uncertain rotor resistance conditions is presented.

In this chapter, we will apply different control algorithms based on function approximation to control the induction machine. Usually the control law is derived by augmenting a backstepping controller by an artificial neural network which can be extended for a larger class of nonlinear systems with partially known models. In section two the problem will be formulated, then field oriented control method is presented in section three . Backstepping will be applied for the known part of the system in section four. Neural network augmentation is detailed in section five. simulation results are presented in section six . Section seven is devoted to some concluding remarks.

7.2 Induction Machine Modeling

The starting point for the control of the induction motor is the system of nonlinear differential equations which characterize its behavior. Under the assumptions of linearity of the magnetic circuit and neglecting iron losses, the dynamics of a n_p pole-pair two phase induction motor are given by the following system of differential equations [24]:

$$\begin{cases} u_{sa} = R_s i_{sa} + L_s \frac{di_{sa}}{dt} + M \frac{d}{dt}\left(i_{ra} cos(n_p\theta) - i_{rb} sin(n_p\theta)\right) \\ u_{sb} = R_s i_{sb} + L_s \frac{di_{sb}}{dt} + M \frac{d}{dt}\left(i_{ra} sin(n_p\theta) - i_{rb} cos(n_p\theta)\right) \\ 0 = R_r i_{ra} + L_r \frac{di_{ra}}{dt} + M \frac{d}{dt}\left(i_{sa} cos(n_p\theta) + i_{sb} sin(n_p\theta)\right) \\ 0 = R_r i_{rb} + L_r \frac{di_{rb}}{dt} + M \frac{d}{dt}\left(-i_{sa} sin(n_p\theta) + i_{sb} cos(n_p\theta)\right) \\ J\frac{d\omega}{dt} = n_p M [i_{sb}\left(i_{ra} cos(n_p\theta) - i_{rb} sin(n_p\theta)\right) \\ \qquad -i_{sa}\left(i_{ra} sin(n_p\theta) - i_{rb} cos(n_p\theta)\right)] \end{cases} \tag{7.2.1}$$

with the flux linkages of the motor phases given by

$$\begin{cases} \lambda_{sa} = L_s i_{sa} + M\left(i_{ra} cos(n_p\theta) - i_{rb} sin(n_p\theta)\right) \\ \lambda_{sb} = L_s i_{sb} + M\left(i_{ra} sin(n_p\theta) - i_{rb} cos(n_p\theta)\right) \\ \lambda_{ra} = L_r i_{ra} + M\left(i_{sa} cos(n_p\theta) + i_{sb} sin(n_p\theta)\right) \\ \lambda_{rb} = L_r i_{rb} + M\left(-i_{sa} sin(n_p\theta) + i_{sb} cos(n_p\theta)\right) \end{cases} \tag{7.2.2}$$

where $L_s \triangleq \frac{\mu_0 \pi l_1 l_2 N_S^2}{8g}, L_r \triangleq \frac{\mu_0 \pi l_1 l_2 N_S N_R}{8g}$.

Here N_s and N_R are the number of windings per pole-pair of the stator and rotor phases, respectively. The retarding torque produced by the friction in the ball bearings of the machine is modeled here by $-f\omega$, where f is the viscous friction coefficient .

The control problem is to choose the input stator voltages u_{sa} and u_{sb} in such a way to make the motor angular velocity ω tracks a given reference trajectory. The stator currents are usually accessible. However, the rotor currents are typically not available for feedback. In fact, the most common type of induction motor is the squirrel cage motor, where rotor currents are distributed on the surface of the rotor making it very impractical to measure the current in each rotor bar. The resulting flux can be measured using Hall effect sensors placed in the air gap, but such sensors are very expensive and reduce the overall reliability of the system. The control problem is still difficult due to the coupled complicated model of the system described in equation (7.2.1). To overcome this problem one could transform it into a simplified form in which $cos(n_p\theta)$ and $sin(n_p\theta)$ expressions are eliminated. Using the following transformation for the flux linkages

$$\begin{bmatrix} \psi_{ra} \\ \psi_{rb} \end{bmatrix} = \begin{bmatrix} cos(n_p\theta) & -sin(n_p\theta) \\ sin(n_p\theta) & cos(n_p\theta) \end{bmatrix} \begin{bmatrix} \lambda_{ra} \\ \lambda_{rb} \end{bmatrix} \tag{7.2.3}$$

Then the dynamic model of the machine in terms of the state variables $\omega, \psi_{ra}, \psi_{rb}, i_{sa}$, and i_{sb} can be written in this form

$$\begin{cases} \frac{d\omega}{dt} = \mu\left(i_{sb}\psi_{ra} - i_{sa}\psi_{rb}\right) - \frac{f}{J}\omega - \frac{1}{J}\tau_L \\ \frac{d\psi_{ra}}{dt} = -\eta\psi_{ra} - n_p\omega\psi_{rb} + \eta M i_{sa} \\ \frac{d\psi_{rb}}{dt} = -\eta\psi_{rb} + n_p\omega\psi_{ra} + \eta M i_{sb} \\ \frac{di_{sa}}{dt} = \beta\left(\eta\psi_{ra} + n_p\omega\psi_{rb}\right) - \gamma i_{sa} + \frac{1}{\sigma L_s}u_{sa} \\ \frac{di_{sb}}{dt} = \beta\left(\eta\psi_{rb} - n_p\omega\psi_{ra}\right) - \gamma i_{sb} + \frac{1}{\sigma L_s}u_{sb} \end{cases} \tag{7.2.4}$$

with $\mu \triangleq \frac{n_p M}{J L_r}, \sigma \triangleq 1 - \frac{M^2}{L_r L_s}, \eta \triangleq \frac{R_r}{L_r}, \beta \triangleq \frac{M}{\sigma L_r L_s}$, and $\gamma \triangleq \frac{M^2 R_r}{\sigma L_r^2 L_s} + \frac{R_r}{\sigma L_s}$.

7.3 Conventional Field-Oriented Control

The control objective is to design a controller in which the stator voltages are selected in order to control the torque, speed and/or position of the motor. The key idea of field-oriented control is to go to another state space representation where the currents regulating the flux and the speed are decoupled [3]. The new coordinate system is a rotating system whose angular position is defined by $\rho \triangleq \arctan(\frac{\psi_{rb}}{\psi_{ra}})$. So, instead of working with (ψ_{rb}, ψ_{ra}), one uses the polar coordinate representation (ρ, ψ_d) given by

$$\rho \triangleq \arctan(\frac{\psi_{rb}}{\psi_{ra}}), \psi_d = \sqrt{\psi_{ra}^2 + \psi_{rb}^2}.$$

The stator phase currents and voltages are then expressed in this new coordinates as follows

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} cos(\rho) & sin(\rho) \\ sin(\rho) & cos(\rho) \end{bmatrix} \begin{bmatrix} i_{sa} \\ i_{sb} \end{bmatrix} \tag{7.3.1}$$

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} cos(\rho) & sin(\rho) \\ sin(\rho) & cos(\rho) \end{bmatrix} \begin{bmatrix} u_{sa} \\ u_{sb} \end{bmatrix} \tag{7.3.2}$$

The quantity ψ_d is referred to as the magnitude of the rotor field flux while ρ is the angle of the rotor field flux. This coordinate system is one that is moving (oriented) with this field flux and thus is called the rotor-flux field oriented coordinate system. The currents i_d and i_q are called the direct and quadrature currents, respectively. Similarly, the voltages u_d and u_q , are called the direct and quadrature voltages, respectively. The rotation matrix used in (7.3.1) and (7.3.2) is called the direct quadrature or (dq) transformation.

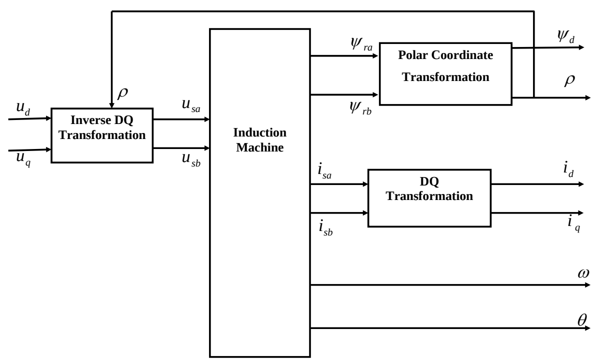

Figure 7.1: Transformation to the field-oriented (dq) coordinate system.

The electromagnetic dynamic model of the induction motor in the fixed stator (direct and quadrant) d-q reference frame can be developed yielding

$$\begin{cases} \frac{d\omega}{dt} = \mu\psi_d i_q - \frac{f}{J}\omega - \frac{1}{J}\tau_L \\ \frac{d\psi_d}{dt} = -\eta\psi_d + \eta M i_d \\ \frac{di_d}{dt} = -\gamma i_d + \beta\eta\psi_d + n_p\omega i_q + \eta M i_q^2/\psi_d + \frac{1}{\sigma L_s}u_d \\ \frac{di_q}{dt} = -\gamma i_q - \beta n_p\omega\psi_d - n_p\omega i_d - \eta M i_d i_q/\psi_d + \frac{1}{\sigma L_s}u_q \\ \frac{d\rho}{dt} = n_p\omega + \eta M i_q/\psi_d \end{cases} \tag{7.3.3}$$

Note that the electromagnetic torque $\tau_e = J\mu\psi_d i_q$ is now just proportional to the product of two state variables ψ_d and i_q. Furthermore, applying the nonlinear state feedback control for system (7.3.3)

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \sigma L_s \begin{bmatrix} -\beta\eta\psi_d - n_p\omega i_q - \eta M i_q^2/\psi_d + \bar{u_d} \\ \beta n_p\omega\psi_d + n_p\omega i_d + \eta M i_d i_q/\psi_d + \bar{u_q} \end{bmatrix} \tag{7.3.4}$$

Then the closed loop system is obtained as follows:

$$\begin{cases} \frac{d\omega}{dt} = \mu\psi_d i_q - \frac{f}{J}\omega - \frac{1}{J}\tau_L \\ \frac{d\psi_d}{dt} = -\eta\psi_d + \eta M i_d \\ \frac{di_d}{dt} = -\gamma i_d + \bar{u_d} \\ \frac{di_q}{dt} = -\gamma i_q + \bar{u_q} \\ \frac{d\rho}{dt} = n_p\omega + \eta M i_q/\psi_d \end{cases} \tag{7.3.5}$$

From (7.3.5), it is clear after field-oriented control and nonlinear state feedback, the final closed-loop system has a simpler structure. Moreover, the flux amplitude dynamics depend only on the direct current i_d and the direct voltage u_d. Thus, it can be regulated to achieve a given flux amplitude that can generate the desired electromagnetic torque τ. However, the robustness to parameter variation of field orientation and nonlinear state feedback control cannot be guaranteed since the designed controller relies entirely on the exact values of the induction motor parameters which are usually estimated so any mismatch could cause system instability and the desired objective is no longer achieved. Much work in the literature of robust control have been devoted to deal with such a problem but there is no a universal solution. To solve the problem of unknown variations in plant parameters and structure, authors in [53] proposed a robust backstepping whereas a combined Neural Network and backstepping controller is presented in [85] to gain from universal approximation property of Neural Networks and ensure the stability of the closed loop system by an Augmented Lyapuonv Function. The main feature of the proposed approach is the application of the novel Lyapunov functions to construct the NN-based backstepping adaptive controller. A simpler strategy is followed here where Neural networks are used for error approximation for nonlinear unknown and neglected terms in the last step of backstepping.

7.4 Bacstepping Control

Let $\tau_e^* = J\mu\psi_d^* i_q$ be the desired torque to be generated with the corresponding reference flux ψ_d^*.

Defining the flux and the speed errors as:

$$\begin{bmatrix} \tilde{\psi_d} \\ \tilde{\omega} \end{bmatrix} = \begin{bmatrix} \psi_d - \psi_d^* \\ \omega - \omega^* \end{bmatrix} \tag{7.4.1}$$

Differentiation of equation (7.4.1) yields to the following error dynamics

$$\begin{cases} \dot{\tilde{\psi_d}} = -\eta(\tilde{\psi_d} + \psi_d^*) + \eta M i_d - \dot{\psi}_d^* \\ \dot{\tilde{\omega}} = \mu\psi_d i_q - \frac{f}{J}(\tilde{\omega} + \omega^*) - \frac{1}{J}\tau_L - \dot{\omega}^* \end{cases} \tag{7.4.2}$$

7.4.1 Step 1

Let us define the first Lyapunov candidate :

$$V_1 = \frac{1}{2}\left[\tilde{\psi_d}^2 + \tilde{\omega}^2\right] \tag{7.4.3}$$

For which the time derivative is expressed as:

$$\dot{V}_1 = \tilde{\psi_d}\dot{\tilde{\psi_d}} + \tilde{\omega}\dot{\tilde{\omega}} \tag{7.4.4}$$

Substituting for the error dynamics in (7.4.4) yields

$$\dot{V}_1 = \tilde{\psi_d}\left[-\eta(\tilde{\psi_d} + \psi_d^*) + \eta M i_d - \dot{\psi}_d^*\right] + \tilde{\omega}\left[\mu\psi_d i_q - \frac{f}{J}(\tilde{\omega} + \omega^*) - \frac{1}{J}\tau_L - \dot{\omega}^*\right] \tag{7.4.5}$$

Assuming that ($\psi_d \neq 0, \quad \forall t \geq t_0$) is ensured, to render 7.4.5 negative definite, we may choose the fictitious control signals, the direct and quadratic currents i_d and i_q as

$$\begin{cases} i_d^* = \frac{1}{\eta M}\left(\eta\psi_d^* + \dot{\psi}_d^*\right) - k_1\tilde{\psi_d} \\ i_q^* = \frac{1}{\mu\psi_d}\left(\frac{f}{J}\omega^* + \frac{1}{J}\tau_L + \dot{\omega}^* - k_2\tilde{\omega}\right) \end{cases} \tag{7.4.6}$$

with k_1 and k_2 are positive design parameters ensuring that the tracking error dynamics will converge exponentially to zero.

7.4.2 Step 2

Let $\tilde{i_d}$ and $\tilde{i_q}$ be the direct and quadratic current errors defined as:

$$\begin{bmatrix} \tilde{i}_d \\ \tilde{i}_q \end{bmatrix} = \begin{bmatrix} i_d - i_d^* \\ i_q - i_q^* \end{bmatrix} \tag{7.4.7}$$

The augmented error dynamics become

$$\begin{cases} \dot{\tilde{\psi}}_d = -\eta(1 + Mk_1)\tilde{\psi_d} + \eta M\tilde{i}_d \\ \dot{\tilde{i}}_d = -\gamma\tilde{i}_d + \bar{u}_d - \gamma i_d^* - i_d'^* \end{cases} \begin{cases} \dot{\tilde{\omega}} = -(\frac{f}{J} + k_2)\tilde{\omega} + \mu\psi_d\tilde{i}_q \\ \dot{\tilde{i}}_q = -\gamma\tilde{i}_q + \bar{u}_q - \gamma i_q^* - i_q'^* \end{cases} \tag{7.4.8}$$

Defining the fictitious control signals

$$\begin{cases} v_d = \bar{u}_d - \gamma i_d^* - i_d'^* \\ v_q = \bar{u}_q - \gamma i_q^* - i_q'^* \end{cases} \tag{7.4.9}$$

The second step of backstepping suggests the following Lyapunov candidate

$$V_2 = V_1 + \frac{1}{2}\left[\tilde{i_d}^2 + \tilde{i_q}^2\right] \tag{7.4.10}$$

The derivative of V_2 along the augmented error dynamics (7.4.8) is given by

$$\begin{aligned}\dot{V}_2 = \dot{V}_1 + \tilde{i_d}\dot{\tilde{i_d}} + \tilde{i_q}\dot{\tilde{i_q}}\dot{V}_2 = -\eta(1+Mk_1)\tilde{\psi_d}^2 + \tilde{i_d}\left(\eta Mp\tilde{s}i_d - \gamma\tilde{i}_d + v_d\right) \\ -(\tfrac{f}{J}+k_2)\tilde{\omega}^2 + \tilde{i_q}\left(\mu\psi_d\tilde{\omega} - \gamma\tilde{i}_q + v_q\right)\end{aligned} \tag{7.4.11}$$

This is the last step of standard backstepping where the control signals appear and could be chosen in a manner to render the derivative of the Lyapunov candidate negative definite as follows:

$$\begin{cases} v_d = -\eta Mp\tilde{s}i_d - k_3\tilde{i}_d \\ v_q = -\mu\psi_d\tilde{\omega} - k_4\tilde{i}_q \end{cases} \tag{7.4.12}$$

with k_3 and k_4 are positive parameters selected to ensure that the current dynamics converge faster than those of the speed and flux.

The obtained control law is implemented given all the state variables are available for feedback and the induction motor parameters are known exactly. In the next section we will propose a control technique to make our designed controller robust and we will use instead of the missed states their estimates.

7.5 Neural Network Augmented Controller

The control law given in equation (7.4.12) can be implemented if these conditions are fulfilled:

- All the state are accessible for feedback, and
- All the nonlinear functions are precisely known.

To relax the first two conditions we can use the same idea as in [61] by introducing neural networks for their approximation ability and continue with backstepping to derive the adaptation laws for neural network weights.

According to equations (7.3.4,7.4.4 and 7.4.11) the control signals depend on all the system parameters, assuming that for each parameter ϱ we have available an estimate $\hat{\varrho}$. The control strategy will be changed by adding an adaptive neural network component in equation (7.4.12) which becomes:

$$\begin{cases} v_d = -\eta M \tilde{psi}_d - k_3\tilde{i}_d + u_d^a \\ v_q = -\mu\psi_d\tilde{\omega} - k_4\tilde{i}_q + u_q^a \end{cases} \tag{7.5.1}$$

Substituting for v_d, and v_q in system equations (7.4.8) and assuming that all neglected terms for each subsystem as an error signal e_i. The extended error dynamics take the form:

$$\begin{cases} \dot{\tilde{\psi}}_d = -c_1\tilde{\psi}_d + \eta M\tilde{i}_d \\ \dot{\tilde{i}}_d = -c_3\tilde{i}_d - \eta M\tilde{\psi}_d + e_d - u_d^a \end{cases} \quad \begin{cases} \dot{\tilde{\omega}} = -c_2\tilde{\omega} + \mu\psi_d\tilde{i}_q \\ \dot{\tilde{i}}_q = -c_4\tilde{i}_q - \mu\psi_d\tilde{\omega} + e_q - u_q^a \end{cases} \tag{7.5.2}$$

The two subsystems are almost identical, we will consider only the dynamics of the flux subsystem to simplify writing:

$$\begin{cases} \dot{\tilde{\psi}}_d = -c_1\tilde{\psi}_d + \eta M\tilde{i}_d \\ \dot{\tilde{i}}_d = -c_3\tilde{i}_d - \eta M\tilde{\psi}_d + e_d - u_d^a \end{cases} \tag{7.5.3}$$

Usually in the literature, neural networks are used in function approximation in the modeling phase [93], however we will use them in controller design to cancel the effect of uncertainty due to neglected dynamics and unknown or varying system parameters, given some conditions are fulfilled.

Assume that there exists an artificial neural network as shown in fig. 7.2 that approximates the neglected dynamics e_d.

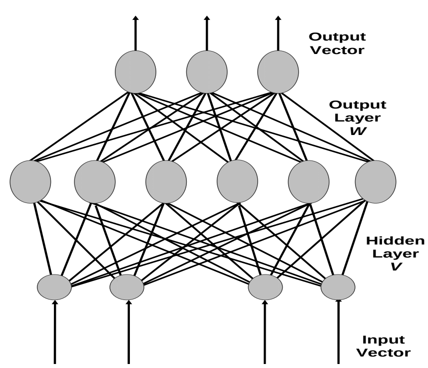

Figure 7.2: A Neural Network with two layers

The approximated error signal can be expressed as:

$$e_d = W^{\mathrm{T}}\Phi(V,\mu) + \varepsilon(\mu)\ \forall\mu \in D \tag{7.5.4}$$

where

$V \in D_V \subset \Re^{N_1}$ is the hidden layer weight vector

$W \in D_W \subset \Re^{N_2}$ is the output layer weight vector

Φ is a set of basis functions

μ is the network input vector, and

ε is the neural network reconstruction error

It has been proven that a neural network that satisfies the conditions of the Stone Weierstrass can approximate any continuous function to any desired accuracy over a compact set [85].

Assuming that the approximation reconstruction error is bounded on the compact set and there exists ideal constant weights W^* and V^* such that $\|\varepsilon(\mu)\| \leq \epsilon^*,\ \forall \mu \in D$ with constant $\epsilon^* > 0$.

The weight vectors are assumed to be bounded $\|W\| \leq W^*$ and their adjustment can be done online. However the whole controller implementation will be difficult as the number of the parameters increases for complex problems. So the first layer parameters can be adjusted off-line in such a way $\Phi(\mathbf{V}, \mu)$ is a basis [93] whereas the adaptation law for the second layer is included in the controller design.

7.5.1 Neural Network Inner Weight Vector Adaptation

Substituting for e_d in equation (7.5.3) yields

$$\begin{cases} \dot{\tilde{\psi}}_d = -c_1\tilde{\psi}_d + \eta M \tilde{i}_d \\ \dot{\tilde{i}}_d = -c_3\tilde{i}_d - \eta M \tilde{\psi}_d + W^{\mathrm{T}}\Phi(V, \mu) + \varepsilon(\mu) - u_d^a \end{cases} \tag{7.5.5}$$

The adaptive control signal u_d^a is designed to cancel the effect of unknown nonlinear terms and is chosen to take the following form.

$$u_d^a = \hat{W}^{\mathrm{T}}\Phi(\mathbf{V}, \mu) \tag{7.5.6}$$

Since the inner layer weights $\mathbf{V}$ are adjusted off-line to fit the real weights V then we can write the mismatch between the adaptive signal and the real neural network as:

$$\begin{aligned} e_d - u_d^a &= W^{\mathrm{T}}\Phi(V, \mu) - \hat{W}^{\mathrm{T}}\Phi(\mathbf{V}, \mu) + \varepsilon(\mu) \\ &= W^{\mathrm{T}}(\Phi(\mathbf{V}, \mu) + e_v) - \hat{W}^{\mathrm{T}}\Phi(\mathbf{V}, \mu) + \varepsilon(\mu) \\ &= \left(W^{\mathrm{T}} - \hat{W}^{\mathrm{T}}\right)\Phi(\mathbf{V}, \mu) + \delta \end{aligned} \tag{7.5.7}$$

where δ is a bounded signal represents the reconstruction error of the neural network plus the error caused by the inner weights mismatch and verifies the following inequality [84].

$\delta \leq \|V^*\|_F \|\mu \hat{W}^T \hat{S}'\|_F + \|W^*\| \, \|\hat{S}'\hat{V}\mu\| + |W^*|$

where

$\hat{S} = \Phi(\mathbf{V}, \mu),\ \hat{S}' = diag\{\hat{s'}_1, \hat{s'}_2, ..., \hat{s'}_{N_1}\}$

with $\hat{s'}_i = d[s(\mu_a)]/d\mu_a|_{\mu_a = \mathrm{v}_i^T\mu}, \quad i = 1, 2, ...N_1$

Let $\tilde{W} = W - \hat{W}$ the error between the real neural network weight and its estimate.

Thus equation (7.5.5) can be rewritten as

$$\begin{cases} \dot{\tilde{\psi}}_d = -c_1\tilde{\psi}_d + \eta M \tilde{i}_d \\ \dot{\tilde{i}}_d = -c_3\tilde{i}_d - \eta M \tilde{\psi}_d + \tilde{W}^{\mathrm{T}}\Phi(\mathbf{V}, \mu) + \delta \end{cases} \tag{7.5.8}$$

Introducing the augmented Lyapunov function

$$V_d^a = \frac{1}{2}\left(\tilde{\psi_d}^2 + \tilde{i}_d^2 + \tilde{W}^{\mathrm{T}} F^{-1} \tilde{W}\right) \tag{7.5.9}$$

where F is an adaptation gain.

The adaptation law for the network weights can be derived to ensure that V_d^a is a Lyapunov function as follows:

$$\dot{\hat{W}} = -F\left[\Phi(\mathbf{V}, \mu)\tilde{i}_d + 2G(\hat{W} - W_0)\right] \tag{7.5.10}$$

where W_0 is an initial estimated value of W and G is a positive design gain.

Proposition 7.5.1. *For system (7.5.3), controller (7.5.6) and adaptive law (7.5.10), there exist a compact set Ξ, and positive constants c_i and given that all assumptions about the neural network weights and reconstruction error are verified then all the signals in the closed-loop system are bounded and the states remain in the compact set Ξ for all time.*

Proof. Differentiating V_d^a with respect to the error dynamics in equation(7.5.8) yields

$$\dot{V}_d^a = -c_1\tilde{\psi_d}^2 - c_3\tilde{i_d}^2 + \tilde{W}^{\mathrm{T}} F^{-1}\dot{\tilde{W}} + \tilde{i_d}\left[\tilde{W}^{\mathrm{T}}\Phi(\mathbf{V}, \mu) + \delta\right] \tag{7.5.11}$$

Using the fact that $\dot{\tilde{W}} = -\dot{\hat{W}}$, equation (7.5.11) can be written as

$$\dot{V}_d^a = -c_1\tilde{\psi_d}^2 - c_3\tilde{i_d}^2 - \tilde{W}^{\mathrm{T}} F^{-1}\dot{\hat{W}} + \tilde{i_d}\left[\tilde{W}^{\mathrm{T}}\Phi(\mathbf{V}, \mu) + \delta\right] \tag{7.5.12}$$

Substituting (7.5.10) in (7.5.12) yields

$$\dot{V}_d^a = -c_1\tilde{\psi_d}^2 - c_3\tilde{i_d}^2 - 2\tilde{W}^{\mathrm{T}} G(\hat{W} - W_0) - \tilde{i_d}\varepsilon(\mu) \tag{7.5.13}$$

Using the fact that the neural networks weight vector is bounded, the derivative of the augmented Lyapunov candidate can be upper bounded as

$$\begin{aligned} \dot{V}_d^a \leq & -c_1\tilde{\psi_d}^2 - c_3\tilde{i_d}^2 + |\tilde{i_d}|\epsilon^* \\ & -G\|\tilde{W}\|^2 - G\|\hat{W} - W_0\|^2 + G\|W - W_0\|^2 \end{aligned} \quad (7.5.14)$$

Putting $\underline{U} = c_1\tilde{\psi_d}^2 + G\|\hat{W} - W_0\|^2$ yields

$$\begin{aligned} \dot{V}_d^a \leq & -\underline{U} - c_3\tilde{i_d}^2 + |\tilde{i_d}|\epsilon^* \\ & -G\|\tilde{W}\|^2 + G\|W - W_0\|^2 \end{aligned} \quad (7.5.15)$$

Completing the squares using $\epsilon^*|\tilde{i_d}| = -\frac{1}{2}\left(\epsilon^* - |\tilde{i_d}|\right)^2 + \frac{1}{2}{\epsilon^*}^2 + \frac{1}{2}\tilde{i_d}^2$, we get

$$\begin{aligned} \dot{V}_d^a \leq & -\underline{U} - c_3\tilde{i_d}^2 - G\|\tilde{W}\|^2 + G\|W - W_0\|^2 \\ & -\tfrac{1}{2}\left(\epsilon^* - |\tilde{i_d}|\right)^2 + \tfrac{1}{2}{\epsilon^*}^2 + \tfrac{1}{2}\tilde{i_d}^2 \end{aligned} \quad (7.5.16)$$

Further, it can be written as

$$\begin{aligned} \dot{V}_d^a \leq & -\underline{U} - (c_3 - \tfrac{1}{2})\tilde{i_d}^2 - G\|\tilde{W}\|^2 - \tfrac{1}{2}\left(\epsilon^* - |\tilde{i_d}|\right)^2 \\ & +G\|W - W_0\|^2 + \tfrac{1}{2}{\epsilon^*}^2 \end{aligned} \quad (7.5.17)$$

The following conditions

$$\begin{aligned} |\tilde{i_d}| &> \sqrt{\tfrac{2{\epsilon^*}^2 + 2G\|W - W_0\|^2}{2c_3 - 1}} \\ \|\tilde{W}\|^2 &> \sqrt{\tfrac{{\epsilon^*}^2 + G\|W - W_0\|^2}{G}} \end{aligned} \quad (7.5.18)$$

with $c_3 > \frac{1}{2}$, and $G > 0$ ensures that $\dot{V}_d^a \leq -\underline{U}$. Which is negative definite in some domain defined by conditions in (7.5.18).

Note that the resulting closed loop system is stable in a given region and the convergence rate of $\tilde{i}_d$ and $\tilde{\psi}_d$ is adjusted by appropriate selection of the design gains in such a manner the $\tilde{i}_d$ dynamics converge faster than those of $\tilde{\psi}_d$. To determine the region of convergence we may use equation (7.5.12), and the boundedness of δ.

Following [85], it can be shown that:

$$\dot{V}_d^a \leq -c_1\tilde{\psi}_d^2 - c_4\tilde{i}_d^2 + \frac{1}{4\lambda_1}\delta^2 \quad (7.5.19)$$

where $c_4 = c_3 - \lambda_1$ and λ_1 is any positive constant such that $c_3 > \lambda_1$.

Using the fact that δ is upper bounded and defining $X = [\tilde{\psi}_d, \tilde{i}_d]^T$, we conclude that $\dot{V}_d^a$ is negative definite whenever $\|X\| \geq \frac{\delta}{2\sqrt{\lambda_1 \min(c_1, c_4)}}$.

It is then straightforward to get that $\dot{V}_d^a$ is negative outside a compact set $\Xi = \{X : \|X\| \leq \frac{\|\delta\|_\infty}{2\sqrt{\lambda_1 \min(c_1, c_4)}}\}$ □

The resulting controller looks like the controller proposed in [61] however here it is obtained in a constructive manner for a triangular system without adding an SPR condition. It can be implemented and we have a freedom in selecting the linear controller gains c_i to achieve a desirable performance.

7.5.2 Hidden layer weight off-line selection

As stated earlier, we have many choices in designing the hidden layer weights depending on the used type of neural network. In this note we will show a design alternative based on Radial Basis Function Neural Networks (RBFNN), which are highly recommended for function approximation due to their simple training.

An RBFNN is composed of two layers; a hidden layer contains a set of neurons with their associated centers, the output of each neuron gives the distance between the input vector μ and its center ν_i. The output ψ of the network is a linear combination of the outputs of the hidden layer as.

$$\psi(\mu_j) = \sum_{i=1}^{N_1} w_i \varphi_{ij} \tag{7.5.20}$$

where $\varphi_{ij} = exp\left(\frac{\|\mu_j - \nu_i\|}{\rho}\right)$ is a Gaussian activation function of the distance between the input μ_j and the i^{th} center ν_i. Defining $W = \begin{bmatrix} w_1 & \cdots & w_{N1} \end{bmatrix}$, $V = \begin{bmatrix} \nu_1 & \cdots & \nu_{N1} \end{bmatrix}$. The design of an RBF network to approximate a given function consists of selecting V in such a way that we construct a set of basis functions [52, 80] and W is adjusted online using an adaptive law to be illustrated in the next subsection. Therefore we have an identification problem based on model (7.5.20) which includes the selection of $N1$ model terms $V = \begin{bmatrix} \nu_1 & \cdots & \nu_{N1} \end{bmatrix}$ from a full model set of $M > N1$ terms $U = \begin{bmatrix} \mu_1 & \cdots & \mu_M \end{bmatrix}$ (typically hundreds or even thousands of terms) while μ_i is defined earlier as a tapped delay line of possible values of the input/output and the estimate of z.

We construct the regression matrix Φ_L corresponding to the set of the input vectors U_L a subset of the starting set of centers U.

$$\Phi_L = \begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1L} \\ \vdots & \ddots & \vdots \\ \varphi_{L1} & \cdots & \varphi_{LL} \end{bmatrix}. \tag{7.5.21}$$

Notice that this matrix is symmetric and all the diagonal elements are ones. It has been shown that the orthogonal algorithm can be employed in selecting the optimal model structure V and to estimate the parameters simultaneously [49]. The orthogonal term selection is formulated using the error reduction ratio vector $ERR_L = \begin{bmatrix} err_i & \cdots & err_L \end{bmatrix}$ defined by:

$$ERR_L = \frac{W^{\mathrm{T}} \Phi_L^2 W}{\Psi^{\mathrm{T}} \Psi} \tag{7.5.22}$$

To find N_1 optimal model terms V a stepwise approach is applied to the full model set U. At each step, the model term with the maximum err_j value from all of the model terms excluding the previously selected terms is chosen. The selection is terminated at the $N1^{th}$ step where a desired tolerated error tol is reached.

$$1 - \sum_{k=1}^{N1} err_k < tol \tag{7.5.23}$$

7.6 Simulation Results

In this section we will investigate the performance of the proposed control strategy to an induction motor. Simulations have been performed with the following induction motor parameters: $1.5kW$ nominal rate power; $1430rpm$ nominal angular speed; two pairs pole; $220V$ nominal voltage; $6.1A$ nominal current.

The other parameters of the machine under investigation are summarized in Table 7.1.

Table 7.1: Induction Machine Simulation Parameters

Parameter	Value
L_r	94 mH
L_s	105 mH
R_s	1.47 Ω
R_r	0.79 Ω
p	2
J	0.0077 $Kg\ m^2$
f	0.0029 $Kg\ m^2/s$

The tracking performance of speed and flux is shown in the following figures: 1- The ideal case (exact model fig. 7.3), 2- Figures (7.4,7.6) show the poor performance in the presence of uncertainty $(+/-50\%)$ of parameter variation without neural network augmentation.

Figures (7.5,7.7) show the tracking performance after neural network augmentation, It is clear that the neural network has compensated for the unknown terms.

Simulations show that this method is robust for induction machine parameter variation.

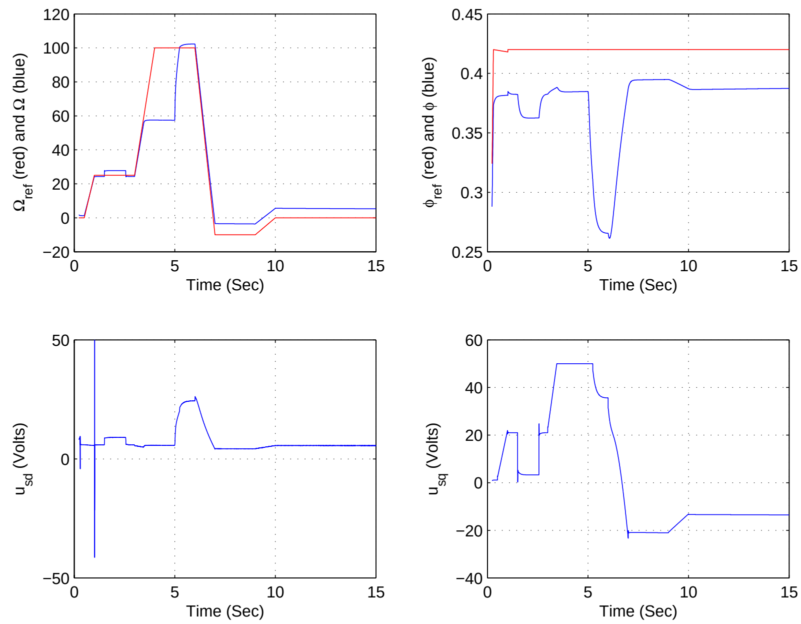

Figure 7.3: Simulation Without Uncertainty but unknown load.

7.7 Conclusion

A new robust adaptive nonlinear controller is proposed to address the tracking problem for a two phase induction machine based on a modified version of FOC. In this scheme, the stability is ensured for the closed loop system and all the errors (tracking and neural network weights) tend asymptotically to zero. The RBF center selection is done using orthogonal least squares structure selection method to minimize the number of parameters of the resulting controller. Finally simulations are presents to highlight the achieved performance although we have assumed a partly known model of the system to be controlled.

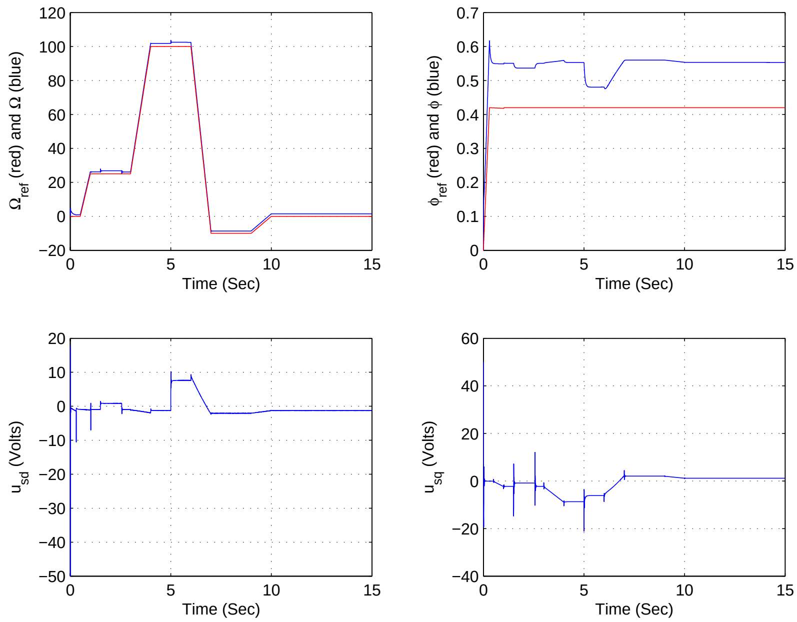

Figure 7.4: Simulation with +50 % uncertainty but without ANN.

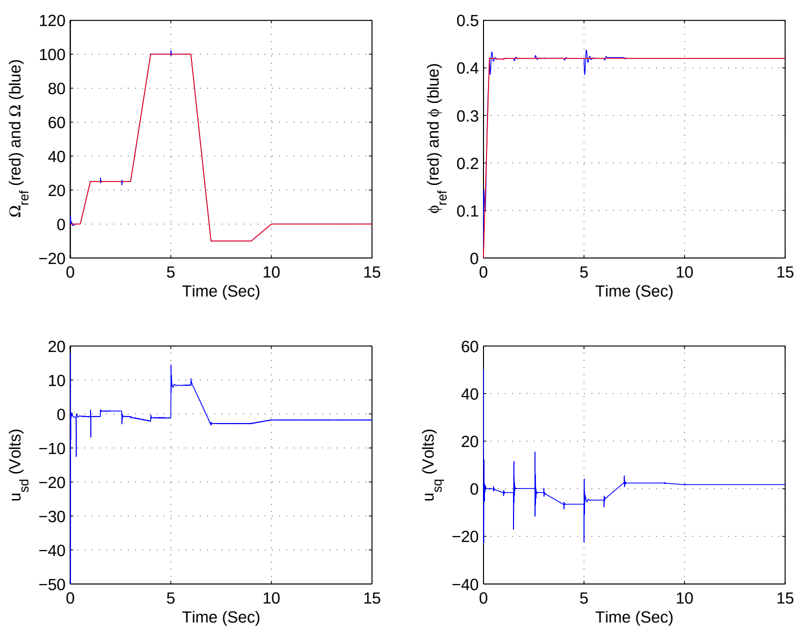

Figure 7.5: Simulation with +50% uncertainty with ANN

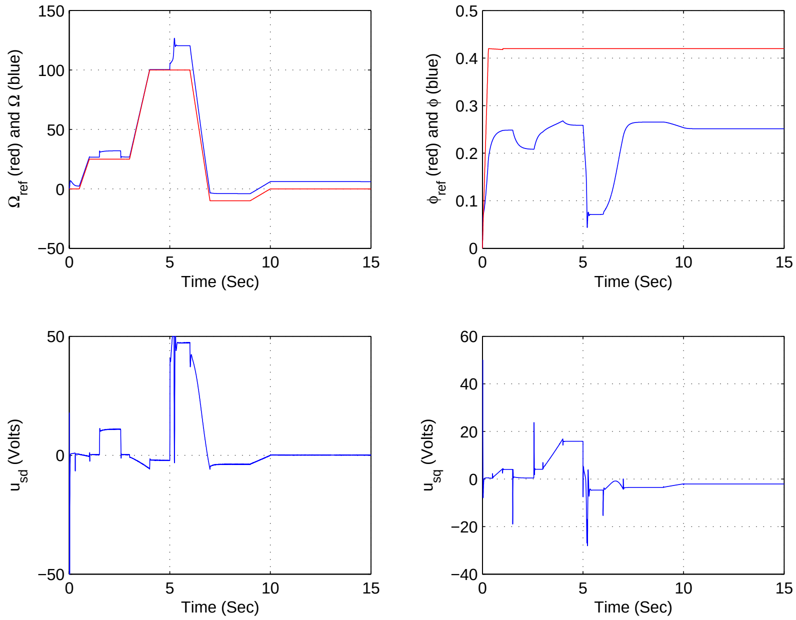

Figure 7.6: Simulation with −50% uncertainty but without ANN

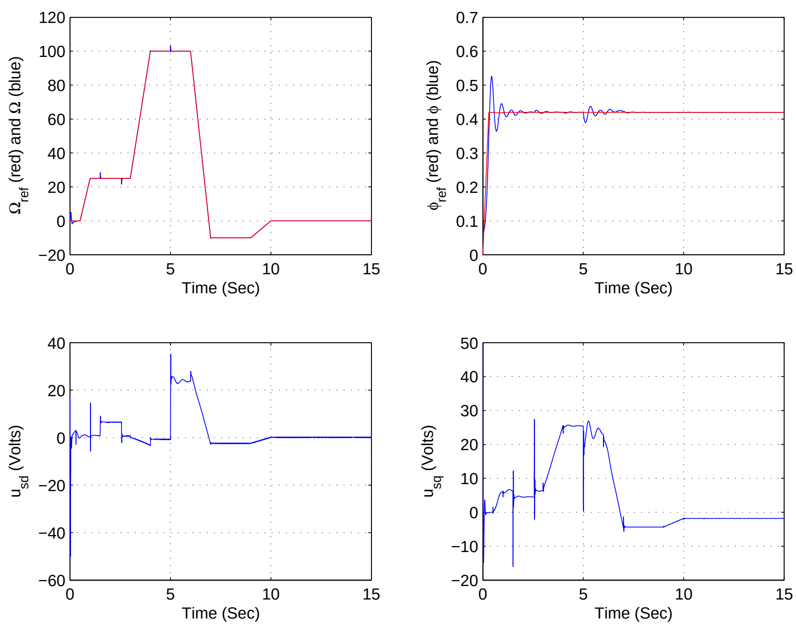

Figure 7.7: Simulation with −50% uncertainty with ANN

Chapter 8

Conclusions and Future Work

In this thesis we have presented a new approach based on combining two existing tools, Artificial Intelligence as Adaptive function approximators and Backstepping feedback control method to solve the problem for a class of nonlinear uncertain systems. First we have highlighted the mathematical definitions and tools used for nonlinear systems analysis and some robustness related issues to address nonlinear uncertain systems. After the presentation of some existing methods based on robust and adaptive nonlinear control theory to overcome systems with uncertainties, we have proposed a new adaptive control design methodology that combines the backstepping approach in addition to NN-based adaptive elements for nonlinear uncertain systems that belong to a class of feedback linearizable systems. Its stability is proved through Augmented Lyapunov Function. The Proposed control methodology is validated experimentally through an application to an antilock braking laboratory test bed which is nonlinear and uncertain in which the friction forces are poorly modeled and vary according to road conditions.

As a second Application we have tested the developed approach on an electromechanical device that is MIMO, nonlinear system, operating on extremely nonlinear dynamic regimes where the uncertainty is present due to the varying load and resistance parametric variations due to high temperature and unknown torque.

A comparison study is performed on the performance of a classical Field oriented control design and two different classes of neural networks: linearly parameterized Radial Basis Functional (RBF) NN and nonlinearly parameterized Single Hidden Layer (SHL) NN for the speed tracking problem for the induction Machine.

As a future study we recommend the integration of neural networks in the design of state observers to achieve output feedback and to reduce the cost of the induction machine control systems. Another issue which will be an interesting topic to be discussed and studied in detail is the optimization of the structure of the artificial intelligent element used for nonlinear function approximation based either on genetic Algorithms or other technics that achieve global optimum.

The validation of the new proposed controller for the induction machine by an experimental study may open other problems that are not highlighted through simulation study. The extension of the developed method to other electromechanical devices is also a good research area.

Bibliography

[1] *The laboratory anti-lock braking system controlled from pc*, Inte Co. Ltd, PL.

[2] N. Hovakimyan A. Calise and M. Idan, *Adaptive output feedback control of nonlinear systems using neural networks*, Automatica **37** (2001), no. 8, 12011211.

[3] J. Olivier A. Razzouk, A. Cheriti and P. Sicard, *Field oriented control for induction motors using neural networks decouplers*, Proceedings of IDS conference (Canada), 1995.

[4] M. Idan A. T. Kutay, A. J. Calise and N. Hovakimyan, *Experimental results on adaptive output feedback control using a laboratory model helicopter*, Proceedings of AIAA Guidance, Navigation, and Control Conference (2002), no. AIAA-2002-4921.

[5] ———, *Experimental results on adaptive output feedback control using a laboratory model helicopter*, IEEE Transactions on Control Systems Technology **13** (2005), no. 2, 196–202.

[6] S. Ananthakrishnan, *Adaptive tachometer feedback augmentation of the shuttle remote manipulator control system*, IEEE International Conference on Robotics and Automation **41** (1995), no. 3, 447–451.

[7] K. J. Astrom and B. Wittenmark, *Adaptive control*, Addison-Wesley Publishing Company, 1995.

[8] M. Corless B. R. Barmish and G. Leitmann, *A new class of stabilizing controllers for uncertain dynamical systems*, SIAM Journal on Control and Optimization **21** (1983), 246–255.

[9] R. Babueska, *Fuzzy modeling for control*, Kluwer Academic Publishers, Boston, 1998.

[10] M. Belkheiri and F. Boudjema, *Adaptive neural network controller for a class of uncertain nonlinear systems*, Transactions on systems, signals and devices: System Analysis & Automatic Control **2** (2007), no. 1.

[11] ———, *Function approximation based augmented backstepping control for an induction machine*, WSEAS Transactions on Systems and Control **2** (2007), no. 9.

[12] ______, *Neural network augmented backstepping control for an induction machine*, International Journal of Modelling, Identification and Control **3** (2008).

[13] M. Bodson and J. Groszkiewicz, *Multivariable adaptive algorithms for reconfigurable fligh control*, IEEE Transactions on Control Systems Technology **5** (1997), no. 2, 217229.

[14] S . A. Bortoff, *Approximate feedback linearization using spline functions*, Auromatica **33** (1997), no. 8, 1449–1458.

[15] F. Boudjema, *Sliding mode control: Application to electrical converters*, Ph.D. thesis, Laboratoire d'Automatique et d'Analyse des Systemes, Toulouse (France)., 1991.

[16] W. Brogan, *Modern control theory*, APrentice-Hall, Englewood Cliffs, NJ, 1991.

[17] M. Brown and C. Harris, *Neurofirzzy adaptive modelling and control*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[18] D. Hittle A. Katz C. Anderson and R. Kretchmar, *Synthesis of reinforcement learning, neural networks and pi control applied to a simulated heating coil*, Artificial Intelligence in Engineering **11** (1997), 421429.

[19] A. J. Calise and R. T. Rysdyk., *Nonlinear adaptive flight control using neural networks*, IEEE Control System Magazine **18, No. 6** (1998).

[20] E. C. Chen and C. C. Liu, *Adaptively controlling nonlinear continuous time systems using multilayer neural networks*, IEEE Transactions on Automatic Control **39** (1994), no. 6, 1306–1310.

[21] F. Chen and H. Khalil, *Adaptive control of nonlinear systems using neural networks*, International Journal of Control **55** (1992), no. 6, 1299–1317.

[22] F. C. Chen and H. K. Khalil, *Adaptive control of a class of nonlinear discrete time systems using neural networks*, IEEE Transactions on Automatic Control **40** (1995), no. 6, 791–801.

[23] S. Chen and S. Billings, *Neural networks for nonlinear dynamic system modelling and identification*, Advances in Intelligent Control **40** (1994), no. 6, 791–801.

[24] John Chiasson, *Modeling and control of electric machines high-performance*, John Wiley & Sons, Hoboken, New Jersey, 2005.

[25] J.Y. Choi and J.A. Farrell, *Nonlinear adaptive control using networks of piecewise linear approximators*, IEEE Transactions on Neural Networks **11** (2000), 390–401.

[26] K.S Yeung C.M Kwan, F.L Lewis, *Adaptive control of induction motors without flux measurements*, Automatica **32** (1996), 903–908.

[27] H. Hellendoom D. Driankov and M. Reinfrank, *An introduction to fuzzy control*, Springer-Verlag, Berlin, 1993.

[28] M. Daehlen and T. Lyche. (eds.), *Box splines and applications.*, In H. Hagen and D. Roller, editors. Geometric Modeling: Methods and Applications. Springer-Verlag, Berlin.

[29] Y. Diao and K. M. Passino, *Stable fault-tolerant adaptive fuzzy/neural control for a turbine engine*, IEEE Transactions on Control Systems Technology **9** (2001), 494–509.

[30] A. Ebrahim and G. Murphy, *Adaptive backstepping control of an induction motor under time-varying load torque and rotor resistance uncertainty*, 38th Southeastern Symposium on System Theory (Tennessee Technological University,Cookeville, TN, USA,), March 2006, pp. 520–530.

[31] S. F. Chowdhury, R. Wahi and Kaminedi, *Aa survey of neural networks applications in automatic control*, Proceedings of the 33rd Southeastern Symposium on System Theory (2001), 349353.

[32] S. Jagannathan F. L. Lewis and A. Yesildirek, *Neural network control of robot manipulators*, Taylor & Francis, London.

[33] S. Fabri and V. Kadirkamanathan, *Dynamic structure neural networks for stable adaptive control of nonlinear systems*, IEEE Transactions on Neural Networks **7** (1996), 1151–1167.

[34] K. Funahashi, *On the approximate realization of continuous mappings by neural networks. neural networks*, Neural Networks **2** (1989), 183–192.

[35] S. Haykin, *Neural networks: A compehencive foundation*, Prentice Hall, NJ, USA.

[36] P. Ioannou and J. Sun, *Robust adaptive control*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[37] A. Isidori, *Nonlinear control systems*, Springer-Verlag, Berlin; New York, 1995.

[38] M. Sharma J. Farrell and M. Polycarpou, *Backstepping-based flight control with adaptive function approximation*, Journal of Guidance, Control, and Dynamics **28** (2005), no. 6, 1089–1102.

[39] J. A. Farrell J. Nakanishi and S. Schaal, *Composite adaptive control with locally weighted statistical learning*, Neural Networks **18** (2005), 71–90.

[40] R. Ordonez J. T. Spooner, M. Maggiore and K. M. Passino, *Stable adaptive control and estimation for nonlinear systems- neural and fuzzy approximator techniques*, John Wiley & Sons, New York, NY, 2002.

[41] A. J. Wohletz, J. Paduano and Annaswamy, *Retrofit systems for reconfiguration in civil aviation*, Proceedings of the AIAA Guidance, Navigation, and Control Conference **AIAA** (1999), no. 99-3964, 217229.

[42] M. Jankovic and I. Kolmanovsky, *Constructive lyapunov control design for turbocharged dieselengines*, IEEE Transactions on Control Systems Technology, **8** (2000), no. 2, 288–299.

[43] E. N. Johnson and A. J. Calise, *Limited authority adaptive flight control for reusable launch vehicles*, AIAA Journal of Guidance, Control and Dynamics **26** (2003), no. 6, 906913.

[44] Y. H. Lin K. S. Narendra and L. S. Valavani, *Stable adaptive controller design, part ii: Proof of stability*, IEEE Transactions on Automatic Control **25** (1980), 440–448.

[45] W. Kaminski and P. Strumillo, *Kernel orthonormalization in radial basis function neural networks*, IEEE Transactions on Neural Networks **8** (1997), no. 5, 1177–1183.

[46] H. Khalil. (ed.), *Nonlinear systems*, Prentice- Hall, Englewood Cliffs, NJ.

[47] B. S. Kim and A. J. Calise, *Nonlinear flight control using neural networks*, AIAA Journal of Guidance, Control and Dynamics **20** (1997), no. 1, 2633.

[48] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: Theory and applications*, Prentice Hall, Upper Saddle River, NJ.

[49] M. J. Korenberg and L. D. Paarmaan, *Orthogonal parameter estimation algorithm for nonlinear stochastic systems*, International Journal of Control **48** (1988).

[50] H. Lewis (ed.), *The foundations of fuzzy control*, Plenum Press, New York.

[51] R. Lipmann, *A critical overview of neural network pattern classifiers*, the IEEE Workshop on Neural Networks for Signal Processing, 1991, pp. 266–275.

[52] L. Ljung (ed.), *System identification : Theory for the user*, Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[53] J. Braslavsky M. Arcak, M. Seron and P. Kokotovic, *Robustification of backstepping against input unmodeled dynamics*, IEEE Transactions on Automatic Control **45** (2000), no. 7, 1358–1363.

[54] F. Boudjema M. Belkheiri, *Adaptive backstepping control for civil structures*, International Conference on Control, Modelling and Diagnosis, (ICCMD06) (Annaba, Algeria), May 22-24 2006.

[55] I. Kanellakopoulos M. Krstic and P. Kokotovic (eds.), *Nonlinear and adaptive control design*, Wiley, New York.

[56] A. El Hajjaji M. Oudghiri, M. Chadli, *Robust fuzzy sliding mode control for antilock braking system*, International Modeling and Simulation Multiconference (Buenos Aires, Argentina), vol. 28, 2007, pp. 1089–1102.

[57] R. Marino and P. Tomei (eds.), *Nonlinear control design: Geometric, adaptive and robust*, Prentice- Hall, Englewood Cliffs, NJ.

[58] M. B. McFarland and A. J. Calise, *Robust adaptive control of uncertain nonlinear systems using neural networks*, Proceedings of the American Control Conference (Albuquerque- New Mexico), no. 0-7803-3832-4, 1997, pp. 1996–2002.

[59] M. B. McFarland and A. J. Calise, *Robust adaptive control of uncertain nonlinear systems using neural networks*, American Control Conference (Albuquerque, New Mexico, USA), June 1997.

[60] F. P. Montero, *Non linear control of uncertain systems: Some application-oriented issues*, Ph.D. thesis, Polytechnic University of Catalunya, Barcelona, Spain, 2004.

[61] A. J. Calise N. Hovakimyan and N. Kim, *Adaptive output feedback control of a class of multi-input multi-output systems using neural networks*, International Journal of Control **77** (2004), no. 15.

[62] N. Hovakimyan J. Prasad N. Kim, A. J. Calise and J. E. Corban, *Adaptive output feedback for high-bandwidth flight control*, AIAA Journal of Guidance, Control & Dynamics **25** (2002), no. 6, 9931002.

[63] F. Nardi, *Neural network based adaptive algorithms for nonlinear control*, Ph.D. thesis, Georgia Institute of Technology, The Academic Faculty of The School of Aerospace Engineering, 2000.

[64] F. Nardi N.Hovakimyan and A. Calise, *A novel observer based adaptive output feedback approach for control of uncertain systems*, American Control Conference, 2001.

[65] D.M. Dawson P. Vedagarbha and T. Burg, *Adaptive control for a class of induction motors via an on-line flux calculation method*, International Control Conference (Dearborn, Michigan), 1996, pp. 520–530.

[66] P. C. Parks, *Lyapunov redesign of model reference adaptive control systems*, IEEE Transacfions on Automatic Control **11** (1966), 362–367.

[67] D. Pham and L. Xing, *Neural networks for identifiication, prediction, and control*, Springer-Verlag, London, UK, 1995.

[68] T. Poggio and F. Girosi, *A theory of networks for approximation and learning*, Technical Report AIM 1140, A1 Laboratory, MIT, Cambridge, MA, 1989.

[69] M. M. Polycarpou, *Stable adaptive neural control scheme for nonlinear systems*, IEEE Transactions on Automatic Control **41** (1996), no. 3, 447–451.

[70] M. M. Polycarpou and P. Ioannou, *A robust adaptive nonlinear control design*, Automatica **32** (1996), no. 3, 423–427.

[71] M. M. Polycarpou and M. J. Mears, *Stable adaptive tracking of uncertain systems using non-linearly parameterized on-line approximators*, International Journal of Control **70** (1998), no. 3, 363–384.

[72] S. Peresada R. Marino and P. Tomei, *Global adaptive output feedback control of induction motors with uncertain rotor resistance*, IEEE Transactions on Automatic Control.

[73] Gonzalo A. Ramos, *Scattered data interpolation using an alternate differential equation interpolant*, Science, University of Toronto, Graduate Department of Computer Science, 2001.

[74] G. Rovithakis, *Robustifying nonlinear systems using high-order neural network controller*, IEEE Transactions on Automatic Control **44** (1999), no. 3, 102108.

[75] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach*, Prentice Hall, Englewood Cliffs, New Jersey 07632 USA.

[76] R. S. Sanchez-Peha and M. Sznaier (eds.), *Robust systems theory and applications*, John Wiley & Sons, INC., USA.

[77] R. M. Sanner and J. E. Slotine, *Gaussian networks for direct adaptive control*, IEEE Transactions on Neural Networks **3** (1992), no. 6, 837–863.

[78] S. Sastry (ed.), *Nonlinear systems: Analysis, stability, and control*, Springer-Verlag, New York, 1999.

[79] S. S. Sastry and A. Isidori, *Adaptive control of linearizable systems*, IEEE Transactions on Automatic Control **34** (1989), no. 11.

[80] R. R. Selmic and F. L. Lewis, *Identification of nonlinear systems using rbf neural networks: application to multimodel failure detection*, XVI Intern. Conf. on Material Flow, Mach. And Dev. In Industry (University of Belgrade), December 2000.

[81] C. De Silva, *Intelligent control: Fuzzy logic*, CRC Press, Boca Raton, FL,, 1995.

[82] J. J. Slotine and W. Li (eds.), *Applied nonlinear control*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[83] D. Specht, *A general regression network*, IEEE Transactions on Neural Networks **2** (1991), 568–576.

[84] S.S. Ge T. Zhang and C.C. Hang, *Design and performance analysis of a direct adaptive controller for nonlinear systems*, Automatica **35** (1999), no. 12, 1809–1817.

[85] ______, *Adaptive neural network control for strict-feedback nonlinear systems using backstepping design*, Automatica **36** (2000), no. 12, 1835–1846.

[86] K. M. Passino V. Gazi and J. A. Farrell, *Adaptive control of discrete time nonlinear systems using dynamic structure approximators*, Proceedings of the American Control Conference (2001), 3091–3096.

[87] M. Vidyasagar, *Nonlinear systems analysis*, 2nd ed., Prentice Hall.

[88] L.-X Wang and J. Mendel, *Fuzzy basis functions, universal approximation, and orthogonalleast-squares learning*, IEEE Transactions on Neural Networks **3** (1992), no. 5, 807–814.

[89] C. Wen and M. Gibbard, *Conventional power systems stabilizer with auxiliary self-tuning/fixed-parameter controller*, Electrical Power & Engergy Systems **17** (1995), no. 1, 3949.

[90] B. Widrow and S. Steams, *Adaptive signal processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[91] N. Sundararajan Y. Li and P. Saratchandran, *Neuro-controller design for nonlinear fighter aircraft maneuver using fully tuned rbf networks*, Automatica **47** (2001), no. 8, 12931301.

[92] B.J. Yang and A. J. Calise, *Adaptive tracking control for a class of nonaffine nonminimum phase systems*, The 2007 American Control Conference (Times Square New York City, USA), no. 0-7803-3832-4, July 2007, pp. 1996–2002.

[93] W. Yu and X. Li, *Some new results on system identification with dynamical neural networks*, IEEE Trans. on Neural Networks **12** (2001), no. 2, 1835–1846.

[94] L. Zadeh, *Fuzzy sets*, Information and Control **8** (1965), 338–353.

[95] ______, *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Transactions on Systems, Man, and Cybernetics **3** (1973), no. 1.

Curriculum Vitae

Mohammed BELKHEIRI, M.Sc. Ingénieur d'Etat (engineer degree) in Electrical Engineering
Address: Pbox1763 Maamoura, Laghouat, 03000, Algeria
Tel. +213- 29931188 Mob. +213-774634957 E-mail: `mbelkhiri@yahoo.com`
Nationality: Algerian
Date & place of birth: 01/11/1977 DJELFA - Algeria.

Education

2003–2008 : Doctorat es Sciences in Electrical Engineering (option: Automatic Control), Department of Electrical Engineering, National Polytechnic School, Algiers, Algeria.
Thesis entitled: "Nonlinear Control of Uncertain Systems : Application to Electro Mechanical Systems"

2000–2002 : Magister degree in Electrical Engineering (Option: Automatic Control and Robotics), Military Polytechnic School, Algiers, Algeria.
Magister Thesis: Nonlinear output feedback control of a helicopter simulator: Backstepping approach.

1995–2000 : Engineering Degree (Ingnieur d'Etat) Department of Electrical Engineering (Formerly INELEC) University of Boumerdès, Algeria. The medium of Instruction: English.
Final year project "Application of neural networks to the prediction of heart rate variability signals".

1995 : Engineering Baccalaureate, Omar Dheina Technical High School. Laghouat, Algeria

Languages

- **Arabic**, mother tongue language.
- **English**: The medium of instruction at INELEC,
  TOEFL SCORE: 573(Date of Test: 14/01/2006 Testing Center: Algiers - Algeria) TWE SCORE (Test of Written English) 4.5
- **French**, Average.

WORKING EXPERIENCE

- **University of Laghouat- Since 2003**: Assistant Teacher : Introduction to computer, Programming Languages, System Identification and nonlinear systems.
- **University of Boumerdès- 2000- 2002**: Part time Teacher : Introduction to Linear systems (Lab Instructor Matlab), Digital Electronics Lab instructor (8086 and 6800-simulation and digital applications.

RESEARCH ACTIVITIES

- **Supervision of Final year projects:**
  1. Modelling and Control of Civil Structures2005/2006
  2. Robust Control of PUMA560 Robot Manipulator2006/2007
- **Research Project Member**: Modeling Identification and Control of Electrical systems using Artificial Intelligence techniques. 2008-2011.

Publications

- **M. Belkheiri, F. Boudjema.***"Backstepping Control Augmented by Neural Networks for Robot Manipulators"* Accepted for presentation at 1st Mediterranean Conference on Intelligent Systems and Automation (CISA'08), June 30- July 2nd, 2008 Annaba -Algeria.
- **M. Belkheiri, F. Boudjema.***"Closed Loop System Identification Based on CCA Subspace Method"*, 2nd International Conference on Electrical and Electronics Engineering 21-23 April 2008 Laghouat, Algeria.
- **M. Belkheiri, F. Boudjema.** *"Adaptive neural network augmented controller for nonlinear uncertain systems"*, Transactions on Systems, Signals and Devices - Systems, Analysis & Automatic Control, Vol. 2, (1) ,ISBN:987-3-8322-6480-2, Shaker-Verlag, Germany, August 2007.
- **M. Belkheiri, F. Boudjema.** *"Function Approximation Based Augmented Backstepping Control for an Induction Machine",* WSEAS Transactions on Systems and Control, Vol. 2, (9), ISSN:1991-8763 September 2007.
- **M. Belkheiri, F. Boudjema.** *"Neural network augmented backstepping control for an induction machine*", International Journal of Modelling, Identification and Control Vol.3 March 2008, INDERSCIENCE Geneva, Switzerland. ISNN:1746-6172 (in press).
- **M. Belkheiri, F. Boudjema.***"Backstepping Control of an Induction machine augmented by a Neural network",* The Fourth IEEE International Conference on Systems, Signals & Devices SSD'07. March 19-22, 2007, Hammamat, Tunisia.
- **M. Belkheiri, F. Boudjema.***"Adaptive backstepping control for civil structures*", International Conference on Control, Modelling and Diagnosis, (ICCMD06), May 22-24, 2006, Annaba, Algeria.
- **M. Belkheiri, F. Boudjema.**"*Adaptive neural network augmented controller for nonlinear uncertain systems*", The Third IEEE International Conference on Systems, Signals & Devices SSD'05.March 21-24, 2005,Sousse,Tunisia.

- **M. Belkheiri, F. Boudjema.**"*Approximate feedback linearization control of a helicopter simulator* ", ECG2 Second National conference on electrical engineering, 17-18 Dec 2002 Bordj Elbahri, Algiers Algeria.