Stable Adaptive Control and
Recursive Identification of Nonlinear Systems
Using Radial Gaussian Networks

by
Robert M. Sanner

Submitted to the Department of Aeronautics and Astronautics
on April 15, 1993, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract
Massively parallel arrays of simple processing elements, the so-called feedforward “neural” network models, can be used to greatly enhance and extend techniques for identification and control of complex, nonlinear dynamical systems. However, the design of practical algorithms capable of ensuring prespecified performance levels requires a comprehensive, cross-disciplinary treatment, drawing techniques and insights from such diverse fields as machine learning, constructive approximation, nonlinear dynamical stability, and robust systems analysis. This thesis presents a methodology for assembling all of these elements into an integrated and systematic framework, developing both constructive neural network analysis and synthesis methods, as well as algorithms for adaptive nonlinear prediction and control with guaranteed levels of performance.

The possible instability mechanisms underlying neural network applications are first diagnosed, and the methods for avoiding them found to depend upon an ability to quantify the uniform approximation capability of the chosen network architecture. To provide this characterization, sampling theory is used to develop a constructive theory of radial Gaussian networks, precisely quantifying the impact of each component on the quality of the approximation. The approximation error bounds provided by these constructions are then used with robust adaptation mechanisms to produce stable recursive identifiers whose asymptotic prediction capabilities are limited only by the approximating capability of the networks used. Moreover, use of radial basis function networks for these applications allows specification of conditions which must be satisfied by the signals input to the network to ensure that the adaptation mechanism is persistently excited.

Robust adaptive techniques are insufficient in a control application to accommodate the inexact parameterization provided by a network; the control law itself must also be made robust. By smoothly combining sliding control with an adaptive feedback linearization algorithm, in which Gaussian networks adaptively cancel the natural nonlinear dynamics of the plant, a class of stable, convergent direct adaptive controllers is developed. This blended control strategy is then combined with a robust version of classical adaptive robot control algorithms to create stable direct adaptive control laws for a class of multivariable nonlinear systems controlled by neural networks.

Thesis Committee:

Prof. Wallace E. Vander Velde, Chairman
Professor of Aeronautics and Astronautics

Prof. Jean-Jacques E. Slotine, Thesis Supervisor
Associate Professor of Mechanical Engineering and Information Sciences

Prof. Tomaso A. Poggio
Uncas and Helen Whitaker Professor of Vision Sciences and Biophysics
## Contents

### 1 Introduction

1.1 Historical Overview .......................................................... 3
1.2 Thesis Overview ............................................................... 6
1.3 The March of Progress ....................................................... 11

### 2 Neural Networks and Function Approximation

2.1 Feedforward Nets as Function Approximators ....................... 14
2.2 Parallel Architectures for Approximation ............................... 16
2.3 Constructive Network Approximations .................................. 22

### 3 Prior Designs and Instability Mechanisms

3.1 Indirect Adaptive Control Algorithms .................................. 31
  3.1.1 Identification Methods and Neural Networks .................... 32
  3.1.2 Gradient Network Training ........................................ 37
  3.1.3 Indirect Controller Structures and Training .................... 39
3.2 Direct Adaptive Control Methods ....................................... 45
3.3 Instability in Network Adaptive Controllers .......................... 48
  3.3.1 Stability of Gradient Training .................................... 48
  3.3.2 Generalization, Excitation, and Network Design ............... 53
  3.3.3 Stability of Control Laws Using Neural Networks ............. 55

### 4 Gaussian Network Analysis and Synthesis

4.1 The Smoothness Assumption .............................................. 68
## CONTENTS

4.2 Classical Sampling Theory .................................................. 71
4.3 Sampling Theory and Feedforward Networks .......................... 72
4.4 Generalized Sampling Theory .............................................. 75
  4.4.1 Oversampling and General Low Pass Filters ...................... 75
  4.4.2 Approximation on Compact Sets ................................... 77
4.5 Generalized Sampling and Gaussian Networks ....................... 83
4.6 Polynomial and Trigonometric Approximation ....................... 86
  4.6.1 Polynomial Approximation ..................................... 88
  4.6.2 Trigonometric Approximation .................................. 92
4.7 Choice of Gaussian Network Parameters ............................... 93
  4.7.1 Review of Error Sources ........................................ 94
  4.7.2 Design for bandlimited functions .............................. 95
  4.7.3 Design for trigonometric polynomials ......................... 97
  4.7.4 Design for polynomials and constants ......................... 98
  4.7.5 Choice of the output weights ................................. 100
4.8 Construction Examples .................................................. 100
  4.8.1 Bandlimited function: \( f_1(x) = \sin(2\pi x)/(\pi x) \) ........ 100
  4.8.2 Polynomial function: \( f_2(x) = ax(1-x) \) .................. 103
  4.8.3 Polynomial term: \( f_3(x) = f_2(f_2(x)) \) .................. 101
  4.8.4 Constant and Trigonometric function: \( f_4(x) = 2 + \cos(4\pi x) \) . 105
  4.8.5 Polynomial term in \( \mathbb{R}^2 \): \( f_5(x_1,x_2) = ax_1 + bx_2 + c \) ... 106
4.9 Remarks ................................................................. 106
4.10 Appendix: Aliasing and truncation bounds ......................... 115

5 Recursive Identification .................................................. 118
5.1 Stable Identification Methods ......................................... 119
  5.1.1 Analysis of Classical Identification Models .................. 119
  5.1.2 Gaussian Networks for Identification ......................... 121
5.2 Discrete Time Identification .......................................... 126
### CONTENTS

5.2.1 Stability and Convergence Analysis .......................................................... 127
5.2.2 Remarks and Comments .............................................................................. 128
5.3 Continuous Time ........................................................................................... 130
5.4 Examples ......................................................................................................... 135
  5.4.1 Supervised learning .................................................................................. 135
  5.4.2 Time series prediction ............................................................................. 136
  5.4.3 Multistep Prediction ................................................................................ 137
  5.4.4 Continuous Time Identification .............................................................. 138
5.5 Persistency of Excitation ................................................................................ 147
  5.5.1 Persistency and Gaussian Networks ......................................................... 147
  5.5.2 Output Weight Convergence and Generalization ..................................... 150
  5.5.3 Examples ................................................................................................. 156

6 Direct Adaptive Control ....................................................................................... 164
  6.1 Stable Adaptive Feedback Linearization ...................................................... 166
      6.1.1 Adaptive Feedback Linearization ......................................................... 166
      6.1.2 Gaussian Networks and Feedback Linearization .................................. 171
  6.2 Controller Designs and Stability Analysis ..................................................... 176
      6.2.1 Unity Control Gain ............................................................................... 176
      6.2.2 Non-Unity Control Gains ..................................................................... 179
  6.3 Remarks ......................................................................................................... 184
  6.4 Examples ....................................................................................................... 187
      6.4.1 Unity Control Gain: Linear Plant ......................................................... 188
      6.4.2 Unity Control Gain: Nonlinear Plant .................................................... 189
      6.4.3 Non-Unity Control Gains ..................................................................... 191
  6.5 Robots and Spacecraft ................................................................................... 193
  6.6 Robotic Example ........................................................................................... 200
7 Enhanced Adaptive Techniques

7.1 Laterally inhibitive adaptation laws ......................... 220
7.2 Dynamic Network Structure .................................. 222
7.3 Moving Centers and Changing Variances ..................... 225
  7.3.1 Effects of bandwidth mistuning ............................. 228
  7.3.2 Stable bandwidth adaptation ................................ 232
  7.3.3 Comments .................................................. 235
  7.3.4 Examples ................................................... 236
7.4 Incorporating Prior Plant Knowledge ............................ 238

8 Directions for Future Research .................................. 248

8.1 Control Extensions ............................................. 249
  8.1.1 Adaptive input-output linearization ....................... 249
  8.1.2 Backstepping designs ..................................... 252
8.2 Network Extensions ............................................. 254
  8.2.1 Alleviating the "curse of dimensionality" .................. 255
  8.2.2 Scattered-center networks and irregular sampling ........ 257
  8.2.3 Local bandwidth: Gabor models and wavelets .............. 260

9 Summary and Conclusions ......................................... 268

Bibliography ....................................................... 273
Acknowledgments

It almost seems unfair to claim this thesis as mine. Now, don't get me wrong: that was me skulking through the mathematics section of Hayden library; haunting a variety of coffee houses in Cambridge and Boston, o'erspreading their tables with discarded foolscap; and working until the early morning in my darkened office, illuminated only by the blue-white glow from my NeXT machine, editing papers or debugging simulators and Mathematica scripts. No, the actual research described herein is mine, warts and all. The existence of the thesis itself, however, owes as much to the support and/or prodding I received from those around me as it does to the research it describes.

My committee was a constant source of inspiration and encouragement: Prof. Slotine continually challenged me to go one step further, never letting me fall back on past success; Prof. Poggio would, in the whirlwind course of a ten minute meeting, give me new ideas to think about for months; and Prof. Vander Velde's balanced perspective kept me anchored to the control mainstream. More than brilliant individuals, they have been true mentors, and I can only hope this thesis reflects well on the guidance they have given me in the past two years.

Nearly qualifying as a committee member himself in the amount of advice and support he has given me, is Dr. Federico Girosi, who not only unstintingly made time to explain to me a fraction of his copious knowledge of approximation theory, but also in the process became a good friend.

As they have been throughout my life, my parents were always just a phone call away, with an infinite capacity for listening to my troubles and for giving solid, practical advice. The love and encouragement they have provided through the years gave me the determination to keep going. I can only hope I return to them some small fraction of what they have given to me.
Almost as if they knew their help was needed, I found old friends re-appearing and new ones emerging as I struggled with this thesis during the past year. Foremost among them have been Mark Edeburn and Jennie ("I really hate losing volume in my agar") Heymont. Feeding me, dragging me spontaneously out of the office "for my own good", and providing a series of much needed diversions, they truly kept me sane through times when the stress seemed intolerable. Then of course at Jennie's instigation there was Infamous Trip (#1) to Brooklyn, during which I met the rest of the amazing Heymont clan, especially Joan, Paul, Sara, and Andrea. The (very pleasant) repercussions of this trip are being felt to this day, and have helped immeasurably during the final stages of preparation of the thesis and its defense. Thanks Andy; I'll think of some way to repay you...

Of course, Infamous Trip (#2) to Brooklyn was actually the Infamous Trip to Yale-New Haven Hospital, whose (*ouch*) repercussions are also still being felt as I write this. Two of my oldest and dearest friends, Bob and Gayle Koch, were instrumental in helping me back to my feet (while still scolding me for being so scarce the past year or so), and enabling me to finish the thesis. John Hinsdale came all the way up from New York to visit me in the hospital, and Joe Shea, Lisa Russell, Stuart Broson, and the rest of the old Senior House crowd chimed in with encouragement, as they always have during our infrequent reunions. I wish I could see you all more often.

And speaking of Senior House, several of the current residents have become good friends in the past year, and have proceeded to follow in Mark and Jennie's mischievous footsteps. Birgit, Thad, T, and Tavenner, all deserve thanks for feeding me wonderful things, and for cheering me up when I most needed it.

The old MIT SSL, birthplace of several of the ideas pursued in this thesis, is happily still alive in spirit, though its members are scattered now about the country. Sam and Anna here in Boston, Dave and Russ down in Maryland, and Ella and Karl out in Southern California have all checked in regularly to see how I'm doing and to boost my spirits. At times they have had more faith in me than I've had in myself,
and I'm glad to have (finally) lived up to their expectations.

A special thanks is due to Carl Much at Lincoln Labs, who agreed provide the funding which allowed me to continue the research I began in the SSL, even after it became clear that I was drifting far away from the practical circuits we had originally envisioned. Although I eventually had to find other funding sources, his support during a critical time was vital in accomplishing the research described below.

And last but never least, my comrades in misery in 3-441, Günter and Hyun Seok, have always been ready to help, either by answering theoretical or practical questions about their research, or by swapping horror stories about graduate student life over nachos and beer. Dr. Yang and I await your company at the finish line, G-man!
Chapter 1

Introduction

Sometimes, connecting the scattered points of tangency between otherwise unrelated fields, a new discipline springs into being, fusing in an intimate fashion techniques and insights from each of the subjects which gave it birth. Usually this nascence is driven by an application or observation which refuses to neatly fit into pre-existing categories, and out of the attempts to build a formal framework for analysis arises the foundations of the new field.

This thesis will argue that just such a synthesis is required for successful treatment of control and identification algorithms which employ the so-called "neural" network model of parallel, distributed computation. Indeed, the analysis herein involves a merger of neural network theory, approximation theory, nonlinear and adaptive control system design, and robust stability theory. Each of these separate strands will be woven into a comprehensive fabric in which to clothe the new discipline, providing both a firm theoretical underpinning as well practical, constructive algorithms.

A key intention of this thesis is to present an analysis of certain classes of neural networks which provides a precise engineering interpretation for each component. In so doing we hope to de-mystify the behavior and properties evinced by these architectures, and to firmly establish them as yet another element of the standard toolbox of techniques which can be brought to bear on practical problems.

While the initial inspiration for these networks comes from the structure of biological nervous systems, the specific forms employed are dictated by practical engineering
considerations. Thus, no attempt is made herein to conform to accurate models of neural function, but rather certain aspects of neural computation will be exploited for engineering purposes. By extracting from the biological models those features which are most attractive to our ends, and using techniques from approximation theory to guide the choices of components in our models, a constructive theory of networks can be developed, capable of accurately quantifying the learning and generalization capability of the resulting architectures.

This analysis also reveals that a finite sized neural network can rarely learn to exactly produce the signals necessary for perfect identification and control, and the resulting perturbations can create serious stability problems in both the learning mechanisms and the closed loop dynamics of the controlled system. However, by combining the constructive network designs with robust adaptation techniques, stable and convergent recursive neural network identifiers can be constructed, and their asymptotic accuracy predicted. By then combining these insights with techniques of robust nonlinear control, the same stability and convergence results can be extended to important classes of direct adaptive nonlinear tracking controllers which use neural networks.

Before embarking on this program, the remainder of this chapter serves as an orientation to the history and rapid development of approaches to using neural network models for adaptive control and recursive identification. Section 1.1 reviews the historical development of the field, while Section 1.2 gives a specific, chapter-by-chapter overview of the ideas developed in this thesis. Finally Section 1.3 briefly reviews the impact the work described in this thesis has had to date, and describes developments in the field which have paralleled the work described herein.
CHAPTER 1. INTRODUCTION

1.1 Historical Overview

The period from the early 1950s to approximately the mid-1960s was an era of explosive growth in learning systems research: tremendous resources were directed toward imbuing computing machinery with the speed, flexibility, and adaptability of biological computing structures. The ability to learn from mistakes, and to use newly acquired knowledge to anticipate and react correctly to new situations, was (and still is) seen as the next major threshold in the evolution of “artificial intelligence” [182].

Among the many groups exploring these methods, two stand out as precursors to the research undertaken herein: the so-called “neural network” research groups, and the “adaptive control” research groups. While neural network researchers used biologically inspired computing and learning models in an attempt to create machines with these adaptive features [59, 99, 138], the control system community concentrated on the task of designing control laws whose adjustable parameters could be continually tuned so as to elicit an “optimal” performance from the dynamic system being controlled [19, 80, 168].

Both fields developed mathematical models for their training algorithms, based partly upon heuristics, and partly upon mathematical optimization theory [38, 97, 138, 168, 178]. While proofs of the stability and convergence of these various approaches to the learning problem were scarce or lacking altogether, the great proliferation of successful heuristic applications of these models drove the research. The most celebrated proof from this era was the perceptron convergence theorem (PCT) [138] which demonstrated that one (biologically inspired) training algorithm for the perceptron model of neural computation could in fact be guaranteed to produce successful solutions, and in finite time, for a class of pattern recognition problems. In most cases however, successful applications of the learning procedures in each field required substantial amounts of trial-and-error tuning of various free parameters in the designs. The importance of a convergence theorem such as PCT should thus not be underestimated, and much of this thesis is devoted to developing similar guarantees
for the new neural network models and training procedures.

The two research groups were not long kept separate, of course. Control theorists realized early on that the learning and pattern recognition capabilities of neural networks could also be very useful in solving difficult control tasks. By learning to associate measurable properties of a dynamic system, presented to the network as a "pattern" of signals, with the control actions required to keep these measurements at specified levels, neural networks could be used to develop control systems for devices too complicated to model and control by conventional techniques. Many of the early methods used to train networks for control tasks also drew inspiration from biology, often not from the low level, intercellular models of learning, but rather from observation of higher cognitive functions. Learning techniques based upon Pavlovian stimulus/response and Skinner's operant conditioning and positive reinforcement ideas were used [50, 100], as were methods of entraining precomputed "optimal" control strategies into network representations [179]. Unfortunately, as with the learning models used by adaptive control theorists, it was not possible to develop prior conditions which would ensure the success of these methods; as a result, "neural" control system design was also very much an ad hoc, trial and error process.

Retrenchment in both neural network and adaptive control research occurred in the mid- and late 1960s, as field experiments and more rigorous analysis showed the potential instability mechanisms underlying the adaptive methods developed by the control community [9, 119], while [106] rigorously demonstrated that there was no theoretical basis for the universal computing properties implicitly assumed to underlie the perceptron learning model.

The adaptive control community responded to this challenge by shifting their aims to more gradual goals. Starting first with single-input, single-output, linear, time invariant dynamic systems, and carefully progressing to more complex multivariable and nonlinear systems, control researchers developed over the next two decades the analysis techniques and algorithms required to ensure the stability and convergence
of their recursive identification and adaptive tracking control designs [46, 81, 89, 107, 108, 112, 113, 117, 150, 155].

Neural network researchers, meanwhile, developed progressively more complex models which finally appeared to be capable of overcoming at least some of the previous criticisms [57, 58, 64, 65, 66, 85, 126, 139]. Driven especially by the apparent ease with which biological computing structures seem to be capable of solving computationally "hard" problems in speech and visual processing, the new multilayer, sigmoidal networks in particular have been extensively applied to a wide range of such problems in recent years, usually using backpropagation or similar gradient descent training techniques. The local nature of signal processing performed in these models, which facilitates a high degree of parallelism in the computation, is thought to embody a key reason for the success of biological solutions to these problems, and the extremely simple input-output behavior of the individual processing elements makes the model particularly amenable to implementation in analog or digital hardware, which further increases their popularity.

The new, multilayer network models have met with a frustrating pattern of successes and failures, however; there seems often to be no clear indication how complex a network is required for a given task, nor how well a trained network will provide correct responses for inputs not encountered during training. In order to analyze these problems rigorously, several researchers began to re-evaluate from first principles the learning problems being posed to their networks, seeking a mathematically formal framework in which to address the matter. Making a complete break from biologically inspired neural models, the regularization approach [126] views the common supervised learning application of neural networks as an ill-posed problem in interpolatory function theory. Perhaps surprisingly, when the necessary constraints are added to the problem statement, by including information about the nature of the functions assumed to underlie the example data, the formal solutions often have a very biological "feel" to them. The layered, parallel arrangements of neuron-like
elements, and the local nature of the required signal processing both naturally emerge from this analysis, although the input-output behavior of the individual elements is somewhat different from sigmoidal or perceptron models [126].

Often, however, applications demand more than simple interpolation of the training data; one is willing to sacrifice an exact fit to the training set for the more desirable feature of ensuring a prespecified worst case error over range of possible network inputs. Even in this extended framework the principle underlying the regularization formulation, that of using assumed constraints on the function to be learned as a guide for developing a network architecture, is still quite sound. By replacing the idea of function interpolation with that of function approximation, one can draw upon the formidable, and still rapidly developing, field of approximation theory to develop architectures suitable for representing entire classes of functions to a specified level of uniform accuracy. Provided then that the function generating the training set belongs to the assumed function class, the network can be guaranteed to have sufficient flexibility to correctly learn and generalize the mapping implicit in the data.

1.2 Thesis Overview

At this point, however, the march away from biology in search of a rigorous formulation has perhaps come too far; in abstracting the learning problem to this level we may have sacrificed the features which make neural network designs so attractive as a computing model. And yet as in the regularization analysis, it is possible to identify within formal approximation theory certain representations which still seem to capture many of the essential features of the biological model. Chapter 2 attempts to strike a balance between these approximation theoretic and engineering requirements, and proposes a more general definition of feedforward "neural" network than often found in the literature. This definition encompasses not only the more popular feedforward neural network models, but a great number of additional models as well
including several powerful representations provided by approximation theory.

Inevitably, these new learning models have again attracted the attention of control engineers, and the past several years have seen development of a great variety of adaptive control and system identification algorithms which employ the newer network architectures and gradient training methods. Section 3.1 briefly reviews the salient features of these designs, and discusses some of their strengths and weaknesses. Unfortunately, despite the great number of reported successful applications, these methods suffer from the same drawback as their predecessors: there is no method of guaranteeing when the training will be successful, or what the quality of the resulting control will be. In fact, Section 3.3.1 demonstrates that, viewed as nonlinear dynamic systems, gradient network training algorithms such as backpropagation actually contain potential instability mechanisms. At least one manifestation of this instability is the well known overtraining phenomenon, which has long plagued neural network applications. Section 3.3.3 then shows how similar instability mechanisms may be present when a dynamic system is controlled by a trained neural network, even when no additional learning occurs.

A careful analysis of the nature of these problems shows that the measures necessary to circumvent them require a prior quantitative characterization of the ability of the chosen network architecture to approximate the mappings necessary for successful identification or control. This discussion reveals the intimate connection between the stability and the approximation theoretic aspects of the overall analysis; a connection which will arise often in the sequel as more sophisticated identification and control applications are considered. It also demonstrates that stability considerations are not merely theoretical technicalities, but are rather vitally important for understanding the circumstances under which a training procedure will be successful, and for explaining observed pathologies of existing network training procedures.

It is only quite recently, however, that the kind of rigorous stability analyses which have marked the past 25 years of development in adaptive control have begun to
appear for identification and control applications of neural networks. Thus, starting with Chapter 4, the remainder of this thesis is devoted to accomplishing a formal merger of the current state of the art in both neural network and stable, adaptive nonlinear control theory. The resulting network training procedures are as far removed from the Pavlovian conditioning ideas of the original learning control methods as the approximation theoretic networks are from their biological counterparts, but, as in these latter, there are surprising similarities between the formal mathematical solutions and the biologically inspired heuristics.

To provide the quantitative characterization of network approximation capabilities required in the stability analysis, Chapter 4 develops a number of constructive analysis techniques for single hidden layer networks. The analysis begins with consideration of the celebrated sampling theorem, which itself provides an excellent example of an approximation theoretic expansion that can be mapped onto a neural network architecture, providing a parallel arrangement of neuron-like elements each of whose input-output behavior is that of an ideal low-pass filter. Moreover, the construction inspired by sampling theory permits precise relations to be established between the components of this network and the properties of the function it approximates, providing a powerful synthesis tool.

The classical sampling theorem, however, has some undesirable numerical properties which imply that quite large networks would be required in order to ensure good approximations over a given range of network inputs. However, by modifying the processing elements in our networks to have better spatial localization properties, while preserving to the extent possible the frequency localization (low-pass) properties of the elements in the classical model, representations can be devised which employ much smaller networks to achieve the same level of accuracy. A formal analysis shows that the Gaussian radial basis function is the optimum choice in this tradeoff, and that by arranging the centers of these elements to lie on a regular grid the components of the resulting networks admit the same interpretation as those of the classical
CHAPTER 1. INTRODUCTION

sampling model, allowing the specification of explicit design procedures. Moreover, adopting a distributional view of this sampling procedure, it is possible to establish a correspondence between radial Gaussian approximation and multivariate spline techniques, which permits specification of constructive Gaussian network designs for approximations to polynomial and trigonometric functions as well.

Armed with a constructive theory of network computation, Chapter 5 re-examines applications of these devices in the framework of formal stability theory, starting with recursive identification. Algorithms for stable, convergent identification of continuous and discrete time nonlinear systems are developed, and include several popular supervised learning techniques as a special case. This analysis demonstrates the importance of using robust adaptive techniques for the training algorithms, and also again highlights the interplay between stability and approximation theory, as the quantitative measures of the degree and extent of network approximation capabilities are vital components of the robust designs which ensure stable and convergent learning.

Many of the tracking control models discussed in Chapter 3 require use of an identification model of the process to be controlled. Satisfactory control of this type thus demands a high fidelity identifier, and it is hence necessary that certain excitation conditions be satisfied by the signals used to train the networks during the identification phase. An extremely attractive feature of the Gaussian network identification models developed in Chapter 5 is that one can exactly determine the conditions a sequence of network inputs must satisfy to guarantee that the signals used in the learning process will satisfy these conditions. Section 5.5 formally derives these conditions, and discusses the enhanced stability properties and asymptotic fidelity bounds obtained when they are satisfied.

Unfortunately, as Section 3.3.3 shows, regardless of how accurate a network identification model is, it is still possible for a control law developed from this model to drive the actual process unstable unless careful attention is given to the interaction between the structure of the control law, the approximation capabilities of the net-
work, and the actual process dynamics. Moreover, the assumption of a sufficiently exciting identification phase is somewhat constraining and may be difficult to ensure in practice. These considerations lead to the detailed exploration, in Chapter 6, of direct adaptive tracking control algorithms which permit learning and control to proceed simultaneously, and which do not require the development of accurate models of the process. By combining the robust adaptation methods developed in Chapter 5 with techniques of robust nonlinear control, it is possible to specify direct adaptive tracking control algorithms which ensure the global stability of the adaptive system and the asymptotic convergence of the model tracking errors. While current state of the art in nonlinear control system design and the complexity of the stability analysis somewhat limits the class of processes for which these structures can be reliably developed, very satisfying convergence results can be given for important classes of nonlinear systems.

Chapter 7 then extends the fundamental ideas of the preceding three chapters, exploiting the interaction between approximation theory and the stability analysis to specify stable algorithms for both on-line construction of the required networks, as well as for on-line adjustment of the centers of radial Gaussian networks. This chapter also demonstrates the ease with which prior knowledge can be incorporated into the designs of the previous chapters, and thus suggests that neural network architectures be used to augment, not replace, known techniques of adaptive nonlinear control.

Use of a regular grid for the centers of a radial Gaussian network, as required by the constructive designs of Chapter 4, may be quite restrictive especially as the number of network inputs increases. Chapter 8 thus explores the extent to which arbitrary collections of centers may be used, by outlining how some new ideas in irregular approximation theory may be used to develop other constructive theories of Gaussian networks. This chapter also discusses the possibilities of extending the control algorithms developed in Chapter 6 to larger classes of nonlinear dynamic systems, by using several new techniques in adaptive nonlinear control theory.
Finally, Chapter 9 summarizes the main points illustrated by the research described in this thesis, and offers some final comments on the algorithms developed below.

1.3 The March of Progress

Development of stable algorithms for using neural networks in control and identification problems has been a "hot" research topic for at least 6 years now. The key results reported in this thesis, the sampling theoretic Gaussian network constructions of Section 4.5 and the stable direct adaptive controller design in Section 6.2.1 combining robust control and robust adaptation, were initially presented in the Spring of 1991 [143], and have been modified and extended substantially in the two years since. In the chapters which follow, no mention is made of developments in the field which paralleled, or were indeed inspired by, the results described in this thesis. This section thus serves to bridge the gap and place the work described in this thesis in the larger, dynamic context of this rapidly developing field.

Simultaneous with the release of [143], Tzirkel-Hancock and Fallside [169] released a report which utilized several similar ideas (feedback linearizing control, linearly parameterized networks, and robust adaptation). Their stability and convergence results were, however, fundamentally flawed, principally because of a failure to accurately characterize the impact of the approximation capabilities of the network on the closed loop performance. When an expanded version of [143] was widely released as [144], Tzirkel-Hancock and Fallside used the analysis in this report to fix the difficulties with their previous algorithm, resulting in [170], which treats a slightly larger class of systems, but uses almost exactly the same algorithm as [144]. They subsequently extended this new algorithm to a class of multivariable nonlinear systems [171].

The report [145] extended the Gaussian network construction and robust adapta-
tion ideas to problems in time series prediction and discrete-time recursive identification. At the same time, Polycarpou and Ioannou released a report [129] which again used linearly parameterized networks with a slightly different form of robust adaptation, to create stable, continuous-time, neural network identifiers. Their extension of these techniques to control applications also used the analysis and techniques of [143] to ensure globally stable closed loop operation.

Similarly, Chen and Khalil [30] at this time incorporated some of the analysis and robustness adaptive techniques utilized in [144] to develop a class of locally stable, discrete-time, adaptive tracking controllers based upon their previous designs [28, 29]. Unlike the algorithms in [144] however, their designs used neither linearly parameterized networks nor robust control augmentation, limiting the basin of attraction of the closed-loop adaptive system.

The end of 1991 saw the publication of Sadegh's paper [140] in which he proposed linearly parameterized network designs for adaptive feedforward linearization of discretized, nonlinear dynamic systems. His analysis relies upon persistency of excitation to provide the necessary robustness in the adaptation mechanism, and the feedforward nature of the controller removes the need for additional robustness mechanisms in the control law itself.

The input weight adaptation and on-line construction methods of Chapter 7 were initially proposed in the Spring of 1992 [146], as were the persistency of excitation results for Gaussian networks [147]. The scattered-center network constructions briefly considered in Chapter 8 were presented in the Fall of that year [148]. At this latter time Polycarpou and Ioannou presented a paper in which they extended their neural network identification designs to time series prediction applications, using a robust, recursive least square technique, as opposed to the recursive gradient technique proposed in [145].
Chapter 2

Neural Networks and Function Approximation

This thesis will describe how to use "neural" networks for identification and control applications. Neurobiology, however, has inspired many different models of distributed, analog processing loosely grouped under the heading "neural networks". Before proceeding, it is necessary to sift through the many available models and decide which of these are suitable for the proposed applications, and indeed to identify what features of this computational paradigm are most attractive from an engineering viewpoint.

This chapter thus serves to introduce the kinds of networks which will be used throughout this thesis, as well as to introduce the notations and assumptions which will be used regarding their architecture and constituent components. Section 2.1 discusses the use of static, feedforward networks as general purpose function approximation modules, and defines and justifies the specific types of network architectures which will be used in the sequel. Section 2.2 examines in detail the components of these architectures and attempts to distill from the different popular and/or biologically inspired models their essential features. A very general description of a class of devices which conforms to these constraints is then proposed as a definition of a static, feedforward "neural network", at least for the engineering tasks for which these devices are used in this thesis. Finally, Section 2.3 provides an introduction
to the problem of actually constructing such networks to accurately approximate a given function, in preparation for the more detailed constructive analysis undertaken in Chapter 4.

2.1 Feedforward Nets as Function Approximators

Consider a typical problem in time series prediction, of the kind which will be examined in detail in Chapter 5. Suppose that the next term in the series can be expressed as a nonlinear function of the previous value, i.e. \( y[t] = f(y[t-1]) \). To attempt to predict the next value of the series, one could employ a parameterized mapping, \( \hat{y}[t] = \hat{f}(y[t-1], \hat{c}[t-1]) \), where \( \hat{y}[t] \) is the predicted value, and \( \hat{c}[t-1] \) are adjustable parameters. If a parametric adjustment algorithm can be found such that \( \epsilon[t] = \hat{y}[t] - y[t] \) asymptotically tends towards zero, while ensuring that \( ||\hat{c}[t-1]|| \) remains bounded, a satisfactory prediction algorithm has been obtained. Clearly this requires the chosen mapping \( \hat{f} \) to be capable of recreating, or at least closely approximating, the actual function \( f \), for some (possibly non-unique) values, \( c \in \mathbb{R}^p \), of the adjustable parameters.

In fact, many proposed applications of neural networks can be expressed in terms of these function approximation capabilities. Problems in object recognition, pattern classification, time series prediction, and others can all be expressed in terms of a network’s ability to approximate the nonlinear function which underlies the examples in its training set, as illustrated in Figure 2-1. In the problem considered above, for example, a network, \( N \), could be used to implement the parameterized mapping \( \hat{f} \). As will be shown in the following chapters, a great number of practical problems in identification and control can also be solved using this function approximation framework.

For this paradigm to be useful, it is necessary to have some guarantees that this is an appropriate use for networks, i.e. that one can reasonably expect a suitably
designed network to be capable of approximating a particular function. Major advances in this direction occurred in the late 1980's, [39, 51, 54, 67] when it was shown that single hidden layer, feedforward networks, of the type depicted in Figure 2-2, are capable of representing continuous, multivariate functions to a chosen degree of uniform accuracy on any compact subset of \( \mathbb{R}^n \), provided the hidden layer is sufficiently large. These results hold for a large number of different "neural" models, several of which will be examined in detail below. Networks of this type implement simple nonlinear transforms of the input signals, and their hidden nodes have no internal state, in contrast both to the models proposed by [57, 58, 65, 66, 87], in which the outputs of the "neurons" are described by coupled, nonlinear differential equations driven by the network inputs, and to the so-called "recurrent" networks [2, 64, 180] which implement similar dynamic behavior in discrete time.

A static model is incapable of capturing some of the more interesting behaviors of the dynamic models, such as limit cycles or strange attractors [63], which are themselves potentially useful forms of computation. However, in many practical applications the "output" of a dynamic network is "read out" only after the internal dynamics have converged to a steady state for a given (constant) input; indeed, an important area of research for these models is understanding the conditions under which the internal dynamics will converge to an equilibrium set [36, 87].

Assuming that the network is such that an equilibrium is obtained in response to a constant exogenous input, the static and dynamic models can be made equivalent by equating the nonlinear transforms implemented by the nodes of a static feedforward network with the flow maps carrying the initial states of a dynamic network to their equilibrium values as a function of the applied network inputs. Note that the required feedforward network would actually have more inputs than the dynamic network, since the initial states of the dynamic nodes would also have to be included as inputs to the corresponding static network. The approximation capabilities of a dynamic network used in this fashion can hence in principle be analyzed simply in terms of the static
properties of its nonlinear flow maps.

Thus, if the task to be performed by the network can be reduced to function approximation, the added complexity and stability concerns introduced by permitting internal network dynamics do not seem to be warranted.\footnote{Admittedly, very complicated flows may arise from very simple recurrent networks, blurring the complexity issue, since very small dynamic networks may require very large static networks to accurately model the resulting flow maps, given constraints (such as those discussed below) on the components of the static network.} There are, however, situations in identification and control where the nonlinear signal processing implemented by dynamic networks are quite useful, and Chapter 3 will briefly explore several of these applications. As discussed in [114], however, these dynamic networks are most easily analyzed by viewing them as static feedforward mappings placed in a feedback configuration with linear dynamic systems, as illustrated in Figure 2-3. Tools for analyzing such feedback configurations are well developed in systems theory, and again shifts the attention to the properties of the nonlinear functions implemented by the static feedforward component. This thesis will thus restrict its focus to the analysis and synthesis of static feedforward network architectures and their uses in identification and control.
Figure 2-2: Illustration of the structure of a single hidden layer, feedforward network.
2.2 Parallel Architectures for Approximation

There are many classical methods available for function approximation, such as Chebyshev polynomials, Fourier series expansion, and splines interpolation, to name only a few [92, 132]. What then makes the approximations implemented by a feedforward network so interesting? At least part of the answer lies with the architectural simplicity of its construction: a neural network is particularly well suited to carrying out these approximations in an efficient manner. A single prototype building block, henceforth called a node, is utilized in a massively parallel fashion. Every node performs exactly the same kind of computation, simultaneous with and independent of the other nodes in the network.

A node is usually composed of two structures, both loosely modeled after concepts from neurobiology. The input function, $\varphi : \mathbb{R}^n \times \mathbb{R}^{p_1} \mapsto \mathbb{R}$, is inspired by the dendritic computation of a membrane potential from the signals carried by incident axons, while the activation function, $g : \mathbb{R} \mapsto \mathbb{R}$, is analogous to the rate at which a neuron generates action potentials in its own axons as a function of its current membrane potential [139, 162]. Figure 2-4 provides a more detailed view of a one hidden layer network illustrating these ideas.

More generally, a fixed number of different kinds of nodes, $g_i \circ \varphi_i$, can also be used without compromising the architectural simplicity of this model. Similarly, more complex architectures might use multiple hidden layers, in which the node outputs of one layer form the inputs to the nodes of the next layer. The above cited approximation theorems indicate that this additional complexity is theoretically unnecessary in order to accurately represent continuous functions, however, it is possible that use of additional layers might allow a reduction in the size of the required representation [34]. This thesis, however, will restrict itself almost exclusively to the single hidden layer architecture depicted in Figure 2-4.

The input function, $\varphi$, is parameterized by a set of adjustable local weights, $\xi_k \in \mathbb{R}^{p_1}$, and serves to reduce the signals incident on node $k$ to a scalar "activation energy".
\( r_k = \varphi(x, \xi_k) \). The activation function, \( g \), is usually a nonlinear function of \( r_k \), but this is by no means necessary. For example, the simple summing junction which forms the output of the network in Figure 2-4 fits easily into this model by taking \( \varphi \) as the inner product of its arguments, and \( g(r) = r \). The approximation implemented by the single hidden layer architecture pictured can be represented mathematically as:

\[
\hat{f}(x, c) = \mathcal{N}(x, c) = \sum_{k=1}^{N} c_k g(\varphi(x, \xi_k)),
\]

where \( c^* = [c_1, \ldots, c_N, \xi_1^T, \ldots, \xi_N^T] \in \mathbb{R}^p \) with \( p = N(p_1 + 1) \). It is usual practice to refer to the parameters in this expansion relative to their position in the hidden layer, thus the \( \xi_k \) are called the input weights, while the \( c_k \) are the output weights (these latter are also the input weights to the summing junction node).

It is the simplicity and potential power of this structure that holds such promise for engineering applications. The ability to compute approximations to nonlinear functions in a massively parallel fashion, using arrays of extremely simple processing elements, is very attractive for real time nonlinear control and identification applications. It must be stressed that these particular features are independent of the actual models \( \varphi \) and \( g \) used for each of the building blocks. Indeed, the above structure seeks to capture what is, possibly, an important component of the information processing mechanism underlying biological nervous systems. To the extent it succeeds at this goal, the actual neural structures which are observed by neurophysiology can be viewed as just specific instances of this paradigm, constrained by the capabilities of the biological building blocks available for their construction. In human engineered implementations the construction constraints are quite different, and practical applications are free to, and in fact should, exploit this flexibility and choose those models which are easiest to analyze and construct.

Thus, while this computing structure has been inspired by neurobiology, the connection with biological nervous systems ends at the abstract level depicted in Figure 2-4. In the following, the chosen forms for \( \varphi \) and \( g \) will have nothing to do with
the input-output behavior of actual neurons, but will rather be driven by engineering convenience and the demands of the specific applications. As stated, however, the model is perhaps too general: virtually any parameterized nonlinear mapping from $\mathbb{R}^n$ to $\mathbb{R}$ could be expressed by taking $g(r) = r$, and using an arbitrary $\varphi$. To keep closer to the kind of local, weighted signal processing which has provided the inspirations for these models, and which indeed is one of their principal attractions, some manner of restriction is needed.

That is, to be practical in an engineering sense, there is a fundamental tradeoff which must be effected in these constructions. On the one hand the device should be easy to implement, especially in hardware, while on the other it should be capable of recreating a great variety of complex functions. These are often conflicting requirements. For example, restricting $g$ and $\varphi$ to simple multiplication and addition results in an easily constructed device, but one which can accomplish only linear transformations of the input. Conversely, if the activation function $g$ is an appropriately chosen continuous function, and if the $\varphi$ are chosen to implement a weighted sum of Kolmogorov's "hashing functions", that is

$$\varphi_k(x, \xi) = \sum_{i=1}^{n} \xi_i h_k(x_i)$$

where the $\xi_i$ are rationally independent and the $h_k$ are appropriately chosen continuous, non-decreasing functions, the resulting network can represent any continuous function defined on the unit $n$-cube, using only $2n + 1$ nodes $g \circ \varphi_k$ [55, 161, 91]. Note that each of the hidden layer nodes in this design has the same input weights. Unfortunately, the functions $h_k$ required for this representation, while continuous, are highly nonsmooth, and would thus be extremely difficult to implement with analog, or even finite precision digital, electronics [55].

Of course, "easy to implement" depends upon the available technology and the constriction media. To attempt to balance these requirements, for the purpose of the feedforward network architectures developed in this thesis, the following constraints are adopted. The input function $\varphi$ is assumed to be such that, for each $n$ and any
\( \xi \in \mathbb{R}^n, \varphi(x, \xi) \) requires only a fixed, finite number of multiplications and summations to compute. The "interesting" processing is confined to the activation function \( g \), which is assumed to be a smooth function except possibly for a finite number of discontinuities. Furthermore, any changes in the number of independent variables in the mapping to be computed should require nothing more than minimal changes to the input function, such as adding an extra afferent signal and suitably augmenting the activation; in particular, the structure of \( g \) should not change with the dimension of the network input.

These constraints preclude most kinds of tensor product expansions, such as tensor product splines; they similarly exclude input functions such as

\[
\varphi(x, \xi) = \prod_{i=1}^{n} x_i^{\xi_i},
\]

which thus precludes even univariate polynomial approximation. In general, \( \varphi \) will be taken as an inner product of the types shown in Figure 2-5, although it will be occasionally useful to extend this structure to permit more general quadratic forms, such as

\[
\varphi(x, \xi, W) = (x - \xi)^T W (x - \xi).
\]

Of course, if more than one hidden layer is allowed, tensor product expansions using nodes satisfying the above constraints can be implemented using "product nodes" in the second hidden layer [126].

The most commonly employed network model, especially with "back propagation" training techniques, employs a biased dot product for \( \varphi \), together with a sigmoidal (smoothly saturating) function for \( g \). However, in the generalized form presented above the model easily encompasses many other techniques of approximation theory, including "flexible" Fourier series [128], radial basis functions [26, 126], and even radial wavelet [42] expansions. Figure 2-5 displays how \( \varphi \) and \( g \) can be chosen to map these and other methods onto the architecture of Figure 2-4. As noted above, not all function approximation strategies can be mapped onto this structure, and those which
do for univariate approximation, may fail to do so as the dimension increases. The challenge in the analysis which follows will be to identify approximation strategies which are sufficiently complex to be capable of reproducing large classes of functions, yet which still conform to the parallel processing model discussed above.

2.3 Constructive Network Approximations

A practical implementation of a network can employ only a finite number of components. Thus, regardless of the theoretical approximation capabilities of these devices, for most functions there will be points where the output of the “best” possible network approximation (measured with respect to an appropriate norm) deviates from the actual values of the function, i.e.

\[ f(x) - \sum_{i=1}^{N} c_i g(\varphi(x, \xi_i)) = d_f(x). \]

Indeed, unless \( f \) can be exactly expressed as a combination of the network elements, \( d_f(x) \) will generally be nonzero for almost every \( x \).

This fact has great significance in the design of learning strategies, for it implies that there is inherently some amount of error which no amount of tuning of the network parameters can alleviate. Unless knowledge of this limitation is built into the adaptation mechanism, the network parameters may keep wandering in a futile attempt to eliminate every error. As will be discussed in the next chapter, under the right conditions this wandering can in fact be pathological, causing the parameters to grow in an unbounded fashion. Controlling this behavior requires some means of characterizing the behavior of the error term \( d_f(x) \).

That is, having created an instance of a feedforward network by choosing specific input and activation functions, it is necessary to understand how well this structure can approximate different kinds of functions. For example, the network approximation theorems cited above state that, if \( \varphi \) is a biased dot product and \( g \) is a sigmoid, or
if \( g \circ \varphi \) is a radial Gaussian, then for any \( \epsilon_f > 0 \) and any compact subset, \( A \), of \( \mathbb{R}^n \), there exist constants \( N, c_k \), and \( \xi_k \) such that

\[
|f(x) - \sum_{k=1}^{N} c_k g(\varphi(x, \xi_k))| \leq \epsilon_f, \quad \forall x \in A
\]

for any continuous \( f \). Unfortunately, however, these theorems are not constructive; the relations among \( \epsilon_f \), \( A \) and the network parameters \( N, c_k \), and \( \xi_k \) are not known, only the theoretical existence of values which make the above inequality true.

To accurately predict the magnitude of \( d_f(x) \), and hence \( \epsilon_f \), a more constructive analysis method must be developed, one which is capable of quantifying the dependence of the network structure on the properties of the function to be learned. Thus, after discussing in Chapter 3 the potentially destabilizing effects of these disturbance terms when networks are used for identification and adaptive control purposes, Chapter 4 will develop explicit algorithms for choosing the parameters \( N, c_k \), and \( \xi_k \) to achieve a specific accuracy \( \epsilon_f \) for a given range of network inputs, \( A \). Later chapters will then develop algorithms which allow stable learning of these parameters.
Figure 2-3: Alternative representation of a dynamic network. The dynamic elements are modeled by placing a static, feedforward network in feedback with a linear dynamic system. In a recurrent network, for example, $W(z)$ would implement a one step delay in each channel, i.e. $W(z) = z^{-1}I$. 

$\hat{y}[t] = N(u[t], \hat{y}[t-1])$
Figure 2-4: Generalized representation of the static, feedforward networks used in this thesis.
### Table

<table>
<thead>
<tr>
<th>Model</th>
<th>( \varphi(x, \xi) )</th>
<th>( g(q) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>perceptron</td>
<td>( x^T \xi + \xi_0 )</td>
<td>([0, \infty])</td>
</tr>
<tr>
<td>sigmoid</td>
<td>( x^T \xi + \xi_0 )</td>
<td>((1 + e^{-q})^{-1})</td>
</tr>
<tr>
<td>general RBF</td>
<td>( |x - \xi|_2^2 )</td>
<td>( g \circ \varphi \in \mathcal{P}_0 )</td>
</tr>
<tr>
<td>Gaussian RBF</td>
<td>( |x - \xi|_2^2 )</td>
<td>( \exp(-\pi \sigma^2 q) )</td>
</tr>
<tr>
<td>radial wavelet</td>
<td>( \xi_0 |x - \xi|_2^2 )</td>
<td>( g(\sqrt{q}) \text{ admissible} )</td>
</tr>
<tr>
<td>Fourier series</td>
<td>( x^T \xi )</td>
<td>( \exp(2\pi j q) )</td>
</tr>
<tr>
<td>look up table</td>
<td>( |x - \xi|_{\infty} )</td>
<td>( \chi_k, \mathcal{K} \text{ compact.} )</td>
</tr>
</tbody>
</table>

**Figure 2-5:** Possible choices for input and activation functions for the networks pictured in Figure 2-4. Notes: *General*: \( \xi_0 \) is an arbitrary scalar which, without loss of generality, can be included in the parameterization of \( \varphi \). *RBFs*: \( \mathcal{P}_0 \) is the class of positive definite functions; positive definite functions of different orders are also often used, although strictly speaking the corresponding networks then require augmentation with polynomials in order to solve interpolation problems [126]; Micchelli [102] gives different conditions for \( g \) which ensure it can solve the interpolation problem; *Wavelets*: [42] gives the admissibility conditions for a mother wavelet, as well as the allowable scaling parameters and arrangement of the \( \xi \); *Look up tables*: these carve the input space into boxes; to build a mathematically precise representation two kinds of nodes are actually needed: one whose activation function is the characteristic function of an open interval, the other of a closed interval.
Chapter 3

Prior Designs and Instability Mechanisms

The recent resurgence of interest in neural network models has brought with it a plethora of new algorithms for using these devices to control complex, nonlinear dynamic systems. This chapter will briefly summarize some popular methods, then examine in greater detail the features and dangers of the most common methods for using networks in identification and control applications. This analysis will serve to guide the development of the algorithms explored in later chapters.

The new methods for using neural networks in control systems can be loosely divided into two categories. The first can be described as “trajectory following” designs, in which networks are trained so the resulting control law forces the process to closely track a specific, user-specified behavior. Figure 3-1 illustrates two possible configurations for such neural network controllers; in the first, the network attempts to force the process to exactly follow a reference signal, in the second, the network forces the plant to follow a model response to the reference signal. These techniques are very close to the designs studied by control systems theorists, and it is thus not surprising that much of the recent formal development in this area has been driven by veteran control researchers [29, 114, 115].

This first category of network control algorithms can be further subdivided into indirect and direct methods. In the indirect approach, an explicit model of the process
Figure 3-1: Block diagram representation of two possible trajectory following control systems which use neural networks. The boxes labeled "SR" are shift registers, which maintain a finite length, sliding window into the input-output data record for the process. Alternately, for continuous time systems, they represent measurements of the process state vector and control input.
is first determined, and this model then used to develop the necessary control signals, while in the direct approach, the controller is immediately trained using only the observed tracking errors. Section 3.1 and Section 3.2 respectively review how neural networks have been used in each of these approaches.

The second class of methods are more "goal-directed" in that precise specifications of the required process outputs are not assumed to be available, rather only a general "evaluation function" which provides some indication of how well the control decisions made have advanced the system toward the goal. In this case the goal may be as general as "balance the broom upright", "throw the ball through the hoop", or even "move to position x with a minimum amount of control effort".

There are a great number of techniques which fall into this second category, including reinforcement learning [15, 16, 167], Q-learning [17], Dyna [166], temporal difference (TD) learning [165], and heuristic dynamic programming (HDP) techniques [174, 175]. Until recently, these different techniques were developed and analyzed independently, lately however, it has been suggested that the underlying principle in each method consists of approximating the dynamic programming procedure for optimizing the evaluation functions used. They are thus each, in a sense, iterative algorithms for developing policies which solve problems of optimal nonlinear control [175, 177]. In these applications, networks are trained to represent either the optimal control policy, $u^*$, as a function of the observed process states, or to represent the optimal cost to go, $J^*$, and the control decisions at each step are then computed using the network representation of the optimal cost. From this perspective, although neural networks are nowhere mentioned in the research, several new AI techniques for developing nonlinear control strategies can be grouped into this class of learning models [24, 188].

This new unifying rigor has already resulted in some preliminary proofs of convergence for this second set of methods [17], and will undoubtedly permit even stronger results in the future. A problematic feature of this method of using networks for
control in this fashion, however, is that an extensive exploratory phase is required. If the goal is to develop an optimal policy for a process with well-understood dynamics, then this policy exploration can be performed inside a digital or analog computer using the simulated dynamics, and the resulting neural network representation used as a component of a real-time control system. Often, however it is assumed in these methods that the process is completely unknown; the information needed to determine the optimal cost or policy is then generated by injecting exploratory control signals into the actual process.

"Failure" of a given control strategy, for example a robotic arm smacking into a wall, a broom falling to the floor, or a submersible spinning wildly, is considered to be an integral component of this type of learning. Indeed, in some algorithms, a failure indication is the only type of feedback the evaluation signal provides [5, 16]. Clearly these failures pose no problems when the exploration is carried out using simulated dynamics. Further, to the extent that the actual process can tolerate this kind of abuse—for example, if a skilled human operator can be assumed present who will correct the failures and restart the learning—this kind of training can also be useful when the network experiments directly with the physical system.

Most commonly, however, one wants guarantees that learning can proceed unattended and will not cause this kind of pathological behavior; that is, one wants guarantees of the stability of the learning and resulting control. Note also that while developing optimal strategies off-line based upon a computer model of the process dynamics may avoid these kinds of physical instabilities while learning, there is still the very real possibility of instability when the resulting control law is brought on-line unless the actual process behaves precisely as the model predicts—a dangerous assumption to make in any situation.

Stability analyses of approximate dynamic programming methods are extremely difficult, and indeed may be fundamentally at odds with the exploratory nature of these algorithms. The past 25 years, however, have seen remarkable progress in the
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

analysis of the stability of adaptive trajectory following control algorithms, and the remainder of this thesis will attempt to extend these results to the new neural network control methods. Unfortunately, when formal stability analysis is turned to the popular trajectory following neural network control designs reviewed in Section 3.1, serious flaws are revealed. In both the training mechanism and the actual control laws potential instability mechanisms exist. The last section of this chapter discusses each of these mechanisms, and in particular shows how they are related to the approximation capabilities of the chosen neural network architecture. This lays the foundation both for the constructive network analysis undertaken in Chapter 4, as well as the development of new algorithms for neural identification and tracking control in Chapter 5 and Chapter 6 which are explicitly designed to avoid the potential instability mechanisms identified below.

3.1 Indirect Adaptive Control Algorithms

Looking at the control diagrams in Figure 3-1, one must decide how to adjust the parameters of the neural network issuing control signals to the process so as to reduce the observed tracking errors, e. Certainly, these adjustments will depend critically upon how the process reacts to changes in the control signal. However, by assumption the process dynamics are unknown, and thus the information required to make accurate changes to the parameters of the control network is not generally available.

This situation can be remedied somewhat if, before attempting to train the control network, a model of the process is constructed using a second neural network. Since the behavior of this new network will be known for any choice of its parameters, if it can be trained to accurately model the observed input-output behavior of the process, it can also be used to predict the changes in the control signal required to force the process to follow a particular trajectory. The structure of the identifier network can thus be used to train the control network.
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

This type of adaptive control is known as indirect, that is, it requires developing explicit, accurate models of the process dynamics during an initial training phase which uses either pre-collected input-output data or (assuming the process is stable) limited on-line, open loop interaction, in which the required input-output data is generated as the plant evolves in response to probing inputs. Once the quality of the model thus developed is assured, the control network is either trained using, or created directly from, the identification model. The trained control network is then finally brought on-line, and its outputs used to drive the actual process.

The general framework for this approach in neural network control applications can be traced to [134], and [78, 79]; specific mathematical description of the processes were introduced by [114, 115] enabling a comprehensive treatment for both nonlinear identification and control, while [31, 32, 33] have extensively explored similar ideas for nonlinear identification and adaptive noise equalization. This indirect adaptive control method has also been extensively explored for robotic manipulators, using different neural and related architectures, by [11, 12, 103, 104] among others. The sections below will review this popular technique in detail in preparation for the stability analysis to follow in Section 3.3.

3.1.1 Identification Methods and Neural Networks

The first step in an indirect approach to the control problem is thus to develop an accurate model of the process: either its forward dynamics, that is, the relation between the process outputs in response to its inputs, or its inverse dynamics, that is, the relation which describes the process inputs required to produce a particular set of outputs. Since techniques for accomplishing this with a great variety of processes are well known in the system identification literature [94], it is necessary to first understand the conditions under which a neural network would ever be needed to solve the problem.

Creating an identifier requires first introducing some prior knowledge about the
process so that an appropriate model structure can be selected. For example, in
discrete time a general process model would be

\begin{equation}
y[t] = f(y[t - 1], \ldots, y[t - N], u[t], \ldots, u[t - M + 1]).
\end{equation}

Most physical systems usually have at least a one step delay between the control and
the output; the instantaneous dependence on \( u[t] \) is included here so as to subsume
iterative supervised learning models into the same architecture. In continuous time
a similarly general description would be

\begin{equation}
\dot{z}(t) = f(z(t), u(t)), \quad y(t) = h(z(t), u(t)),
\end{equation}

where \( z(t) \in \mathbb{R}^n \). In both models \( y \) is the process output, and \( u \) is the process input,
both assumed scalar for convenience. For simplicity the nonlinear functions \( f \) and
\( h \) will be assumed to be at least continuously differentiable on \( \mathbb{R}^{N+M} \) or \( \mathbb{R}^{n+1} \) as
appropriate.

Several assumptions have already been made about the behavior of the process
at this level. First, in the discrete time case, it is assumed that upper bounds are
known on the extent of the dependence of the process output on its past outputs and
the past values of the control input; equivalently, in the continuous time description,
appropriate signals describing the state, \( z \), of the system are assumed known, and will
in the sequel be assumed available for measurement. Second, these models assume
that the process is autonomous, that is, there is no explicit dependence on time in
the equations of motion.

For convenience in discussing these models, define the signal vector \( x \) to be \( x[t] =
[y[t - 1], \ldots, y[t - N], u[t], \ldots, u[t - M + 1]] \) in discrete time models, and \( x^T(t) =
[z^T(t) \ u(t)] \) in continuous time. To simplify the ensuing discussion, and to mirror
the historical development, the discrete time model will be used to illustrate the
techniques described in this section; similar ideas can be applied to the continuous
model, which may appear more naturally in nonlinear physical systems, and which
will be analyzed in detail in later sections.
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

With these assumptions about the process behavior, two common structures can be utilized to model it. The first is the so-called parallel identification model [111],

$$\hat{y}[t] = \hat{f}(\hat{x}[t], \hat{c}[t - 1]),$$

where \(\hat{x}\) is defined similarly to \(x\) but using the past estimates \(\hat{y}[t - i]\) in place of \(y[t - i]\). The function \(\hat{f} : \mathbb{R}^{N+M} \times \mathbb{R}^p \rightarrow \mathbb{R}\) is parameterized by the adjustable values \(\hat{c} \in \mathbb{R}^p\), and in classical identification algorithms is chosen such that, for at least one choice of parameters, \(c \in \mathbb{R}^p\), \(\hat{f}(\hat{x}, \hat{c}) = f(\hat{x})\) for any \(\hat{x} \in \mathbb{R}^{N+M}\).

Notice that this parallel structure receives no direct information from the process it attempts to model, but rather feeds its own estimates of the process behavior back to itself to predict future behavior. This is in contrast to the so-called series-parallel model [111], which has the form

$$\hat{y}[t] = \hat{f}(x[t], \hat{c}[t - 1]),$$

the chief difference being that predictions of future process behavior are here made based on the observed time history of the process itself, not that of the model. Figure 3-2 illustrates these two different observer structures.

In both identifier models, different functional forms are possible for \(\hat{f}\) depending upon the amount of additional prior information available about the process. For example, if the process is known to be linear an appropriate structure is

$$\hat{f}(x[t], \hat{c}) = \sum_{k=1}^{N} \hat{c}_k y[t - k] + \sum_{k=0}^{M-1} \hat{c}_{k+N+1} u[t - k] = \sum_{k=1}^{p} \hat{c}_k x_k[t]$$

where here \(p = N + M\). Similarly, if the nonlinearity \(f\) is known to lie in the span of a finite set of basis functions \(Y_k\), one can choose

$$\hat{f}(x[t], \hat{c}) = \sum_{k=1}^{p} \hat{c}_i Y_k(x[t]).$$

It is now clear how neural networks may be used to extend this paradigm; as discussed in the previous chapter, networks provide efficient, parameterized expansions for entire classes of nonlinear functions. By using an appropriately designed
Figure 3-2: Two possible identification structures. Top: parallel model. Bottom: series-parallel model.
neural network, $\mathcal{N}(x,c)$, to implement the function $\hat{f}$, the resulting structure can theoretically recover any $f$ belonging to an appropriate function class (for example, any continuous function), instead of the quite restricted forms which can be recovered by the above models. The symbol $\mathcal{N}(x,c)$ here stands for a neural network structure with inputs $x$, parameterized by the weights in $c$, and its exact composition and configuration may depend upon the specific function class to which $f$ belongs. In the discussion which follows, any network with differentiable components can be assumed; later chapters of this thesis will focus on a specific class of network models.

A finite size neural network, however, can only approximate a given function class, and this only on closed bounded subsets of the input space. As a result, the precise equality between $\hat{f}(x,c)$ and $f(x)$ exploited in the classical algorithms no longer holds, but is replaced by the weaker condition that, given an $\epsilon_f > 0$, it is possible to choose a neural network structure such that, for some $c \in \mathbb{R}^p$

\begin{equation}
\hat{f}(x[t],c) = \mathcal{N}(x[t],c) = f(x[t]) + d_f(x[t])
\end{equation}

with $|d_f| \leq \epsilon_f, \forall x \in A$, where $A$ is a closed, bounded subset of $\mathbb{R}^{N+M}$. That is, in contrast to classical identification algorithms, which provide globally exact parameterizations of the process dynamics, neural networks provide parameterizations which are both local (i.e., accurate only in a finite region) and approximate.

Additional prior information about the process may of course be introduced into the structure of the identifier model; this additional information can often substantially facilitate the subsequent design of control laws. For example, if

$$y[t] = f(y[t-1],\ldots,y[t-N]) + \sum_{k=0}^{M-1} c_k u[t-k]$$

the nonlinear function $f$ can be identified by an appropriate neural network, and a more conventional linear identifier used to determine the influence of the control terms $u[t-k]$. 
3.1.2 Gradient Network Training

Training neural networks typically requires selection of an appropriate cost function to minimize. In supervised learning applications (many of which reduce to a special case of the above identification problem by assuming a memoryless process with \( N = 0 \) and \( M = 1 \)), the cost is the sum of the squares of the errors \( \mathcal{N}(\mathbf{x}, \hat{\mathbf{c}}) - f(\mathbf{x}) \) at a finite set of test points, \( \mathbf{x}_i \) [139]. The network weights are then iteratively adjusted to minimize this cost. For more general identification problems, one cannot assume the availability of a finite, prior set of input-output responses, but rather must be prepared for any possible input-output sequence which may be generated by the process. One generalization of the supervised learning cost function to this setting would be to compute the cost over a moving window into the evolving time series,

\[
J(\hat{\mathbf{c}}) = \sum_{t=1}^{T-1} e^2[t]
\]

where \( e[t] = \hat{y}[t] - y[t] \) is the prediction error at time \( t \), and hence implicitly depends upon the parameters \( \hat{\mathbf{c}} \) being used in the model.

Commonly (even in supervised learning applications), \( T \) is chosen to be 1, so that the cost to be minimized is simply the instantaneous squared error. In this case, the gradient descent law for parameter adjustment gives:

\[
\hat{c}[t] = \hat{c}[t-1] - k_a e[t] \frac{\partial \hat{y}[t]}{\partial \hat{c}}.
\]

For the series-parallel identifier model, this training law is exactly equivalent to back-propagation (omitting the common "momentum" term [139]); that is

\[
\hat{c}[t] = \hat{c}[t-1] - k_a e[t] \frac{\partial \mathcal{N}}{\partial \hat{c}}.
\]

where the network partial derivatives are evaluated at \( (\mathbf{x}[t], \hat{c}[t-1]) \), i.e. using the "old" weights and the node activations produced by the current network inputs.

For the parallel model, however, the feedback structure of the identifier requires use of the sensitivity function method [38], producing a more complicated mechanism
for computing the gradient of a squared prediction error metric. The sensitivity
method applied to neural networks uses the following system of equations to adjust
the weights [115]:
\[
\gamma_k[t] = (D_1\mathcal{N})[t]\gamma_k[t - 1] + \cdots + (D_N\mathcal{N})[t]\gamma_k[t - N] + (D_{N+M+k}\mathcal{N})[t]
\]
\[
\hat{c}_k[t] = \hat{c}_k[t - 1] - k_a e[t]\gamma_k[t],
\]
where the notation \((D_i\mathcal{N})[t]\) is used to indicate the partial derivative of the function
\(\mathcal{N}\), with respect to its \(i\)th argument, evaluated as above at \((x[t], \hat{c}[t - 1])\). Notice
that unlike the adjustment mechanism for the series-parallel model, which directly
uses the derivatives of the network output with respect to each weight, the sensitivity
method instead passes this signal through a time-varying filter. A filter of this form is
associated with each weight of the network, creating a substantial computational bur-
den. In principal, however, this filtering could be accomplished in hardware with the
same degree of parallelism which characterizes the forward operation of the network,
and thus is not necessarily an impediment to realization of this technique.

These sensitivity methods give only an approximation to gradient descent in the
chosen cost functional; the use of partial derivatives in this fashion implicitly assumes
that the adjustable parameters are constant, while in fact they are continually chang-
ing. To ensure that these methods provide a reasonable approximation to the gradient,
the network parameters should change only quite slowly, requiring very small adapta-
tion gains, or should change only after several time steps, instead of changing at each
instant. A complete discussion of the tradeoffs and limitations can be found in [115].
Choosing a large adaptation rate, so that the parameters change rapidly, may result
in very erratic learning, and even instability. In fact, it was precisely this difficulty
of determining the exact conditions under which the stability of sensitivity methods
could be assured that originally led control researchers to redesign adaptation mech-
anisms from first principles using stability methods [119]. Methods for accomplishing
a comparable “Lyapunov redesign” for identification and control algorithms which
use neural network models will be explored in Chapter 5 and Chapter 6.
3.1.3 Indirect Controller Structures and Training

The design of an effective control law for a dynamic system is often hampered by lack of complete knowledge about the structure of its equations of motion. The goal of the above identification algorithms has been to reconstruct the missing information by observing the input-output behavior of the process over a period of time. Assuming that a sufficiently slow adaptation rate has permitted the development of an identification model which adequately describes the observed behavior of the process, for example such that $|\hat{y}[t] - y[t]| \leq \epsilon$ for a sufficiently long period of time, two possibilities exist for utilizing the information contained in the models. If, given exact prior information about the process, a nonlinear control design methodology exists which will produce the desired behavior, the components of the identification model may be used in place of the actual plant information in the design. Often, however, the design of an effective control law based upon the identification model is more difficult; in this case, neural networks and gradient training techniques can again be helpful in developing a representation of the required control law.

To illustrate the first possibility, suppose that it is known that the true process is modeled by

$$y[t] = u[t - 1] - f(y[t - 1]).$$

Given $f$, the control law

$$u[t - 1] = f(y[t - 1]) + \alpha y[t - 1] + (1 - \alpha)r[t - 1],$$

where $r$ is the reference input the plant must follow, and $|\alpha| < 1$, will cause the process to asymptotically follow the model response

$$y_m[t] = \alpha y_m[t - 1] + (1 - \alpha)r[t - 1],$$

since the tracking error $e[t] = y[t] - y_m[t]$ satisfies $e[t] = \alpha e[t - 1]$ and hence decays asymptotically to zero. In the absence of prior knowledge about $f$, one could identify this function using a neural network as in the previous section, and instead use for
the control
\[ u[t - 1] = \mathcal{N}(y[t - 1], \hat{c}) + \alpha y[t - 1] + (1 - \alpha) r[t - 1], \]
where \( \hat{c} \) is the parameter choice learned by the network during the identification phase; note that this is not necessarily the same as the parameter choice \( c \) which minimizes the term \( d_f \) in (3.3). Even if the optimal parameters are learned, however, \( \mathcal{N} \) will not usually exactly cancel \( f \), and generally the ideal asymptotic decay of the error dynamics provided by the exact control law will be perturbed by the approximation error \( \mathcal{N}(x[t], \hat{c}) - f(x[t]) \). In this example the structure of the error dynamics is such that, provided the approximation error is small in magnitude, its effects on the tracking error will also be small. Hence, the general conditions under which \( |\mathcal{N} - f| \) can be guaranteed to be small is clearly an important factor in evaluating the quality of this design, and will be discussed at length in the later sections.

When the process assumes a more general form, it may be quite difficult to directly specify the form of the control signal required to asymptotically track the output of a particular model. Assuming, however, that a suitable feedback control law exists and can be implemented by an appropriately designed neural network, the network could be placed in series with the process, the feedback loop closed by using the process outputs as inputs to the network, and the network output used as the control input to the process. This is the configuration shown in Figure 3-1. The network parameters could then be adjusted so as to continually reduce the model tracking error. For example, if it is known \textit{a priori} that there exists a continuous control law of the form,

\[ u[t - 1] = \theta(y[t - 1], \ldots, y[t - N], r[t - 1], u[t - 2], \ldots, u[t - M + 1]) \]
such that \( y[t] \) asymptotically tracks \( y_m[t] \), a (recurrent) neural network with the same inputs could be employed to learn the continuous function \( \theta \).

\[
\hat{u}[t - 1] = \mathcal{N}_c(y[t - 1], \ldots, y[t - N], r[t - 1], \hat{u}[t - 2], \ldots, \hat{u}[t - M + 1], \hat{c}[t - 1])
\]
\[
= \mathcal{N}_c(x_c[t - 1], \hat{c}[t - 1])
\]
(the $c$ sub- and superscripts are used to distinguish the control network and its parameters from those of the network, $N_i$, and parameters $c^i$, used in the identification model). With this control strategy the closed-loop process output can be written as

$$y[t] = f(y[t - 1], \ldots, y[t - N], \hat{u}[t - 1], \ldots, \hat{u}[t - M + 1]),$$

but unfortunately, a gradient descent procedure to reduce the error $y[t] - y_m[t]$ cannot be implemented, since evaluation of the closed loop dependence of the process output on parameter variations in the control network, $\frac{\partial y[t]}{\partial c}$, would require exact knowledge of the process dynamics, $f$, which by assumption are unknown. This is a more formal statement of the problem observed in the introduction to this section.

To avoid this complication, the control network is instead placed in series with a trained parallel identification model of the process and the feedback loop closed as above. The parameters of the control network are then adjusted to generate the inputs required to force the identifier output to follow a model response, i.e. so that $|\hat{y}[t] - y_m[t]| \leq \epsilon$ for sufficiently large $t$. The known dynamics of the identifier can then be used to compute the required sensitivity functions needed to train the control network [115]. When acceptable tracking has been obtained by using the control network with the identification model, the control network can be brought on-line with the actual process as described above. Figure 3-3 illustrates the structure of this algorithm.

More precisely, using the general model above, the control network parameters would change according to the output of the following system of equations [116]:

$$\gamma_k[t] = (D_1N_i)[t] \gamma_k[t - 1] + \cdots + (D_NN_i)[t] \gamma_k[t - N] + (D_{N+1}N_i)[t] \eta_k[t - 1] + \cdots + (D_{N+M-1}N_i)[t] \eta_k[t - M + 1]$$

$$\eta_k[t - 1] = (D_1N_c)[t - 1] \gamma_k[t - 1] + \cdots + (D_NN_c)[t - 1] \gamma_k[t - N] + (D_{N+2}N_c)[t - 1] \eta_k[t - 2] + \cdots + (D_{N+M-1}N_c)[t - 1] \eta_k[t - M + 1] + (D_{N+M+k-1}N_c)[t - 1]$$

$$\hat{c}_k[t] = \hat{c}_k[t - 1] - k_\alpha \hat{e}_k[t] \gamma_k[t],$$
where \( \hat{e}[t] = \hat{y}[t] - y_m[t] \), and as before, the identifier network partial derivatives are evaluated at \((x_i[t], \hat{c}^i[t - 1])\), and here the controller network partial derivatives are evaluated at \((x_c[t - 1], \hat{c}^c[t - 1])\). Note that the parameters of the identifier model remain fixed while training the control network, and that, as above, the sensitivity of the control network output to changes in its \( k \)th parameter propagate through a high dimensional, time-varying filter before being used to update the corresponding weight.

Jordan has also proposed that, rather than attempt to drive the identifier model response to zero, the control network instead be allowed to interact directly with the process while it is trained, using the actual tracking error \( e[t] = y[t] - y_m[t] \) to adjust the network parameters. The identifier model in this case is then used by the algorithm only to generate the required gradient information [79]. This procedure has the advantage of continuing to update the control law as long as the actual plant fails to track the model, potentially avoiding inaccuracies in the control signal which might arise from the inevitable small discrepancies between the identifier model and the actual plant, but at the expense of using a gradient signal which only approximates the required information.

To speed the training in the above algorithm, Narendra [114, 115, 116] has proposed first training the identification network in series-parallel fashion, avoiding the need for the complex gradient filters in the identification process, then using the same network in a parallel fashion when training the control network. This approach also has its disadvantages, for, as Section 5.4.3 will show, the output of a trained series-parallel network may in some cases deviate rapidly from that of the true process when it is used in a parallel fashion.

Often in these applications, the model response is actually the identity mapping, \( y_m[t] = r[t] \). In this case the control network is being trained to precisely invert the identifier model (and hopefully therefore also the process itself) [71, 104, 105]. This inverse can also be learned in a more direct fashion, by using the identification phase
to instead develop a model of the inverse dynamics from the available input-output data. That is, instead of predicting the process outputs in response to its inputs, the network is used to make predictions of the control input to the process based upon observations of the current and past process outputs, and the past control inputs, i.e.

$$\hat{u}[t - 1] = \mathcal{N}_c(y[t], \ldots, y[t - N], u[t - 2], \ldots, u[t - M], \hat{e}_c[t - 1]).$$

In this case, the cost to be minimized during training would be the squared control prediction error, $\epsilon[t] = \hat{u}[t - 1] - u[t - 1]$ (or possibly a summation of these terms over a moving window) for the range of signals seen during the identification phase. Once a network has accurately identified the inverse dynamics, it can be used to control the process by replacing the input signal $y[t]$ with the desired process behavior $r[t]$, and using the network output as the control input to the process. Of course, identifying the inverse in this fashion assumes that a functional relation exists in the input-output data, that is, that the relation between $u[t - 1]$ and the signals input to the network is not one-to-many; if this is not the case, inverting an identified forward model may provide better results [78, 79].

In these latter methods of training and utilizing networks for control, it is very difficult to assess a priori what happens when the forward or inverse estimates are not precisely equal to the actual functions required to force the process to follow the model response. Unlike the simpler control models considered at the beginning of this section, the inevitable differences between the ideal control signal and that produced by a network trained in the fashion above may not produce small discrepancies in the process output compared to the model. That is, training a control network such that $|\hat{y}[t] - y_m[t]| \leq \epsilon$ for sufficiently large $t$ does not necessarily guarantee that, when the actual plant is controlled by the resulting network, $|y[t] - y_m[t]|$ will be comparably small. Similarly, learning the inverse dynamics so that $|\hat{u}[t - 1] - u[t - 1]|$ is as small as required, does not necessarily guarantee good tracking when the inverse model is used to control the process as indicated above.

Further, the required inverses may not even exist, and when they do exist may
not be representable by continuous functions. The arguments justifying the use of
differentiable network models are no longer applicable in this latter case, and more
complex network designs, employing discontinuous elements, are required in order to
ensure the required inverse can be learned to a specified tolerance [160]. Even assum-
ing the existence of a continuous inverse, the process may not be stably invertible,
that is, the control signals required to force the process to follow the given model may
grow without bound [114, 159]. The methods presented above may indeed learn to
implement such a control, but clearly this would not represent a desirable solution to
the tracking problem.

There are thus many questions regarding the efficacy of the control methods pre-
sented in this section which require answers. Before attempting to address these,
however, there are much more fundamental concerns which must be addressed re-
arding the stability and convergence properties of the gradient training process used
to generate the identification model. Following a brief digression in the following sec-
tion to discuss direct approaches to the control problems, these latter questions will
be examined in Section 3.3.1, before exploring the stability properties of the control
laws themselves in Section 3.3.3.

3.2 Direct Adaptive Control Methods

Despite the apparent difficulty of meaningfully adjusting the parameters of the con-
trol networks pictured in Figure 3-1 based only upon tracking error measurements,
there are situations in which such a direct approach to the adaptive control problem
can be accomplished. Successful application of this technique usually requires more
prior knowledge about the structure of the process dynamics and the form of the
required control law than do the indirect approaches examined above. As a result,
the discussion in this section will be more general than that given above, foregoing
detailed discussion of specific controller designs until the detailed analysis given in
Chapter 6.

While direct adaptive control techniques for certain types of nonlinear systems have been available for almost ten years, recently a number of unique methods have appeared in the literature. Neural networks are explicitly proposed in many of these new techniques, however there are a few which, while not ostensibly employing neural networks, use controller structures which are sufficiently close to the kinds of representations provided by networks that they should be considered in this category.

Arimoto's work [6, 7, 82], for example, provides methods for directly estimating the entire control function required to force a robot manipulator to carry out a particular task, free from any prior assumptions about the correct structure for this control signal. Moreover, his learning algorithm can be demonstrated to implement a contraction mapping in the function space in which the control function lives, thus proving convergence of the algorithm. Of course, one cannot realistically learn every point of a continuous function independently—there are uncountably many of them—and thus a practical implementation must at least discretize the control signal, learning instead constant values on a suitably fine partition. This effectively creates an adaptive lookup table which defines the required control function, and such a table can be easily mapped onto a neural network as discussed in the previous chapter. In fact, such an adaptive look-up table approach has also been suggested in the indirect robotic control algorithms proposed by Atkeson [11, 12].

In an attempt to extend Arimoto's work to more general classes of problems, Messner and Horowitz [101, 69, 70] have devised a method for adaptive robotic control which does not require iteration over a single motion as Arimoto's algorithm does. This method assumes certain continuity and differentiability properties of the required control signal to define an appropriate kernel function which spreads to adjacent points information about the required control signal learned along a specific trajectory through the state space. The continuity and differentiability properties assumed of the control signal are thus used to build up a general representation of the required
control strategy from limited specific experience.

The control law in this method is represented as an integral transform on \( \mathbb{R}^n \) of the form

\[
\hat{u}(x(t)) = \int_{\mathbb{R}^n} \hat{c}(t, \xi) K(x(t), \xi) d\xi
\]

and the weighting function \( \hat{c}(t, \xi) \) is continuously updated, instead of the control function itself. Once again, however, a practical implementation can hardly adapt this entire function, and thus the integral representation must be discretized, instead producing a representation of the form

\[
\hat{u}(x(t)) = \sum_k \hat{c}_k(t) K(x(t), \xi_k).
\]

The number and location of evaluation points, \( \xi_k \) required to form an accurate approximation to the integral strongly depends upon the structure of the kernel and the smoothness of the ideal weighting function \( c \). The relation to neural network methods is clear, however, by identifying this expansion with a one hidden layer network whose input weights are \( \xi_k \), whose output weights are \( \hat{c}_k \) and whose nodes each implement the nonlinear transform \( K(x, \xi_k) \) of the input vector \( x \). In this form, the algorithm is quite similar to the early potential function method of function approximation introduced by Aiserman et al. [1].

Kawato has also proposed direct adaptive neural network controllers for robotic manipulators [83]. His initial algorithms used one hidden layer networks whose inputs were the desired trajectories for the arm, and whose nodes implemented specific nonlinear transforms of the input signals, to issue control signals to the robot. The node functions were chosen to match known pieces of the robotic dynamic equations, so that Kawato's original algorithm is effectively a massively parallel implementation of the DC'AL adaptive feedforward robot control algorithm proposed by Sadegh and Horowitz [141, 142].

In his more recent designs, however, Kawato has made the inputs to the network the actual states of the robot, making the controller a true feedback mechanism, and
replaced the specific basic function nodes with more general neural network designs, which are assumed to be capable of *exactly* representing the required control function [84]. Similarly, in [28, 29] Chen and Khalil proposed adaptive feedback linearizing designs for a class of discrete time nonlinear systems in which the required control functions are assumed to be exactly represented by appropriately designed neural networks whose inputs are the measured states of the process.

In all of the algorithms described above there are two distinct features which distinguish them from the indirect methods of the previous section. First, the form of the required control signal is directly estimated, with no attempt to form an explicit model of the process to be controlled. Second, learning and control proceed *simultaneously* while still ensuring stable operation. Arimoto gives a contraction mapping argument to prove the stability and convergence of his algorithm, while Messner and Horowitz use Lyapunov theory to the same end. Kawato has recently published Lyapunov stability proofs for his robot control algorithm and neural network training mechanism [84], although these make some restrictive assumptions about the functions to be learned and their behavior while learning occurs. Similarly, Chen and Khalil provide at least local Lyapunov stability results for their algorithm.

Unfortunately, each of these proofs assume that the structure used to represent the control function is capable of exactly reproducing the signals required for perfect tracking. Regardless of how fine a partition is used in a look-up table, or how well the integral is discretized, or how many nodes are used in a neural network, the resulting structure will only be able to *approximately* represent the required function, although it may be possible to make the mismatch arbitrarily small over a particular range of inputs. Practical implementation of the methods described in this section thus also suffer from the drawbacks discussed above for the indirect methods, and it is again necessary to quantify the impact of this mismatch on the stability and convergence of the training and control.
3.3 Instability in Network Adaptive Controllers

Despite the possible problems, there have been many successful demonstrations of the above direct and indirect algorithms applied to practical problems in identification and control, although often heuristic tuning of certain of the training procedures and controller parameters is required to obtain satisfactory results. If, however, neural networks are to be truly useful in control applications there must be no guesswork governing their utilization; heuristics should be replaced by precise criteria delineating a successful design from a failure. The intention of this section is to show that it is really not so difficult to obtain poor results from gradient training procedures and network control algorithms, even under ideal conditions, and to identify the factors which influence the success of these methods.

3.3.1 Stability of Gradient Training

Consider, for example, a special case of the above discrete-time identification model where the process is described by the memoryless input-output relation \( y[t] = f(u[t]) \) and \( f(u) = \sin(2\pi u)/(\pi u) \). To identify the function \( f \) describing the process, a series-parallel identifier structure is employed, using a neural network whose input is \( u[t] \) and whose output is the prediction, \( \hat{y}[t] \) of the process value at time \( t \). In this application, the network is constructed using a single hidden layer of Gaussian radial basis function nodes [26, 126]. There are 17 nodes in the hidden layer of this network, and their centers lie on a regular mesh of size \( \Delta = 4/9 \), so that the network prediction at time step \( t \) of the required value \( f(u[t]) \) can be written as:

\[
\hat{y}[t] = \mathcal{N}(u[t], \hat{c}[t - 1]) = \sum_{k=-8}^{8} \hat{c}_k[t - 1] \exp(-\pi^2(u[t] - k\Delta)^2).
\]

(for convenience, the elements of \( \hat{c} \) have been labeled according to their position in the mesh). To reflect the total lack of prior information about the underlying process, the initial estimates of each required output weight is taken as zero.
To train this network, the 17 pairs of training data, \((\xi_i, f(\xi_i))\), shown in Figure 3-4 are used. Since there is only a finite data record, and since a very large amount of training time may be required before satisfactory results are obtained, the inputs \(u[t]\) in the identification model are chosen to cycle through the data in the training set; i.e. \(u[t] = \xi_{1+(t \mod 17)}\).

The error metric to be minimized is the instantaneous squared prediction error, and to give a gradient algorithm the best possible opportunity for success, the input weights of this network are held fixed and only the output weights are adjusted, so that the squared prediction error is quadratic in the parameter error. The gradient descent algorithm for each output weight is then

\[
\hat{c}_k[t] = \hat{c}_k[t - 1] - k_a \epsilon[t] \exp(-\pi^2(u[t] - k\Delta)^2)
\]

where \(\epsilon[t]\) is the prediction error, \(\epsilon[t] = \hat{y}(t) - y[t]\). In this simulation the adaptation gain is chosen as \(k_a = 0.2\).

Figure 3-5 shows the worst case and total square error over the training set as a function of the time index \(t\), while Figure 3-6 shows the worst case error over the compact set \(A = [-2.0, 2.0]\), during the first 5000 training steps. These plots demonstrate the classic overtraining phenomenon; as the network constructs a better fit to the available data, the error generalizing to points outside the training set worsens. The actual reason for this behavior, however, is cause for alarm: the steady worsening of the generalization error is caused by a monotonic growth in one of the network output weights, as shown in Figure 3-7. Increasing the number of training iterations produces minimal improvement in the training set error, but virtually linear growth in this weight and in the corresponding generalization error. Though it appears from the training errors in Figure 3-5 that the learning has "converged", in fact every additional training cycle serves to worsen the overall quality of the identification model, as Figure 3-8 shows.

Insight into this phenomenon can be obtained by recasting the identification problem posed to the network as an interpolation problem. Removed of its iterative
structure, the network is being asked to find a weight vector \( c \) such that

\[
G c = t
\]

where \( G_{ij} = \exp(-\pi^2(\xi_i - (j - 9)\Delta)^2) \), and \( t_i = f(\xi_i) \). In this case, there are exactly as many training examples as network nodes, but the square matrix \( G \) is very nearly singular (with a condition number exceeding \( 10^{30} \)) and part of the training data in \( t \) lies in its "effective" nullspace. Solving the interpolation problem exactly thus requires effectively unbounded weights, which the gradient algorithm merrily attempts to recover, oblivious to the catastrophic effect this has on the overall quality of the identification model.

If one were solving this interpolation problem directly, the magnitude of the weights used could be restricted by computing a pseudo-inverse of \( G \) which omits eigenvalues smaller than a particular threshold, thus making a tradeoff between the accuracy of the fit to the data and the size of the weights used. However, nothing in a pure gradient descent solution procedure is capable of performing this tradeoff; moreover, there is no prior method for determining when such techniques might be necessary, aside from monitoring in real time the eigenvalues of the evolving interpolation matrix (in fact this is a viable strategy, and is related to the idea of persistency monitoring [10] in adaptive systems theory, as the discussion in Section 5.5 will reveal).

There are a number of heuristics which can be used to address the problem observed above. For example, by adjusting the centers of the Gaussian network to correspond to the unique points in the process input, the interpolation matrix \( G \) is guaranteed to be nonsingular [102, 126]. Similarly, by introducing a "stopping criterion" to halt adaptation when the "true" minimum is reached, that is, when \(|c(t)| \) is as small as the best possible network approximation to \( f \) on \( A \) would allow, the network will not fruitlessly strain to eliminate the last bit of error from the training set, and unbounded growth in the parameters can again be avoided [122, 56]. But each of these heuristics merely replaces one problem with another. Adjusting the
network centers is only effective if there is a finite number of distinct points in the input-output stream, and if the network is of comparable size. This will rarely be the case in practical applications. On the other hand, deciding when to halt adaptation requires a method of a priori characterizing the representational capabilities of the chosen network design.

More quantitative insight into this problem can be gained by instead considering the dynamic aspects of the training. Performing a Taylor series expansion of the prediction error in terms of the network parameters, and recalling the limitations of network approximations discussed in Section 2.3,

\[
e[t] = \hat{y}[t] - y[t] = N(u[t], \hat{c}[t-1]) - N(u[t], c) + d_f(x[t])
\]

\[
(3.5) = \frac{\partial N}{\partial \hat{c}} \hat{c}[t-1] + d_g(u[t], \hat{c}[t-1]) + d_f(u[t]),
\]

where as usual, the indicated gradient is evaluated using the instantaneous values of the network inputs and parameter values. The term \(d_f\) represents the inevitable error incurred approximating a continuous function using a finite extent neural network and \(\hat{c}[t-1] = \hat{c}[t-1] - c\) is then the mistuning between the parameters being used at time \(t-1\) and the parameter values, \(c\), which provide the smallest possible \(\epsilon_f\) such that \(|d_f(x)| \leq \epsilon_f\) for all \(x \in A\). The term \(d_g\) contains the terms of the expansion higher than first order in the weight mistuning.

In the first two identifier/process models considered in Section 3.1.1, for which stability based analyses are well developed, the terms \(d_f\) and \(d_g\) vanish identically, leaving only the gradient term of the expansion. The vanishing of \(d_f\) indicates that the chosen basis is capable of a globally exact representation of the process dynamics for an appropriate choice of parameters, while the vanishing of \(d_g\) indicates that the error approximating \(f\) with \(\hat{f}\) can be linearly parameterized in terms of the mistuning of the parameter values. It is these two features which have been crucial in determining conditions which ensure the stability and convergence of identification algorithms employing these models [56, 111].
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

When a neural network is used to estimate the unknown function, however, these terms are generally nonzero. Yet, the training algorithms considered above account only for the gradient term in their attempts to minimize the squared error, neglecting completely the effects of $d_f$ and $d_g$. Analysis of the linearly parameterized models shows that when such perturbations to the error equation are ignored by the adaptation mechanism, it is no longer possible to assure stability and convergence [46, 137, 72]. The adjustable parameters may wander away from their optimal values in response to the perturbations, and even, for particular disturbances, diverge to infinity; this is known as parameter drift instability [72].

In the specific example considered above, keeping the input weights fixed renders $d_g$ identically zero but, as will be shown below, $d_f$ is nonnegligible for the chosen network architecture. The overtraining phenomenon witnessed, and the monotonic growth of one of the network weights, can thus be seen as a manifestation of the parameter drift problem. Fortunately, methods exist in the adaptive systems literature for modifying the gradient algorithm to guarantee its stability. One such technique, exactly equivalent to the “stopping criterion” heuristic, is to place an appropriately chosen deadzone into the error signal used for adaptation, so that no adaptation occurs when $e[t]$ is smaller than the worst expected magnitude of $d_f$. With this modification, here using a deadzone of 0.26 units, no parameter drift occurs and overtraining is avoided, as illustrated in the above figures and Figure 3-9. Chapter 5 and Chapter 6 will formalize this notion of using robust adaptation techniques to accommodate the use of inexact expansions of the functions to be learned.

3.3.2 Generalization, Excitation, and Network Design

Even assuming that the pathological behavior described above can be avoided, the success of the indirect control methods described in Section 3.1 depend critically upon the fidelity of the identification model. Unfortunately, as is well demonstrated by the previous example, convergence of the prediction error does not necessarily imply a high
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

fidelity identification model; that is, asymptotic satisfaction of the inequality $|\hat{y}[t] - y[t]| \leq \epsilon$ does not imply that the quantity $|\hat{f}(x, \hat{c}) - f(x)|$ is necessarily small for every choice of $x$. This is well known in the neural network literature as the generalization problem—convergence to minimal error on a training set does not guarantee good predictions of the required values for points outside the training set. Just how acute this fidelity problem can be is graphically illustrated by Figure 3-8, which shows the approximation implemented by the network for the previous example over the range of inputs $[-2, 2]$. Clearly this approximation deviates in places substantially from the worst case 0.2 units observed during training.

This phenomenon is well known in the recursive identification literature as well, which has actually developed criteria, known as the persistence of excitation (PE) conditions, which can guarantee when convergence of the prediction error to zero will ensure asymptotically perfect fidelity in the identification model, at least for the linearly parameterized identifier structures discussed in Section 3.1.1 [111, 149]. In the absence of persistency of excitation conditions for neural networks, researchers have again used heuristics to attempt to ensure high fidelity identification. A typical procedure is to train the identification model by injecting random inputs into the process, in the hope that the resulting training will be sufficiently exciting. Often, however, models are trained using physical data collected by passively observing a process over a period of time, and there may not be an easy way to obtain new data, or to directly inject stochastic perturbations so as to obtain more “interesting” input-output examples. Ideally, what is required is either a precise specification of the PE conditions for neural networks, permitting a designer to judge a priori the quality of a particular input-output record, or else the development of control algorithms which are not so critically dependent on a precise identification model. These options will be explored in Section 5.5 and Chapter 6 respectively.

These convergence questions are moot, however, if the networks used in the identification algorithm are incapable of providing an approximation of sufficient quality.
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

As discussed in the previous chapter, while the oft-cited network approximation theorems provide a justification for using these devices to learn the functions required for control applications, they provide only general existence statements, and do not specify the size, topology, or composition of a network required to achieve a given accuracy for a particular range of input signals. Once again, in the absence of a comprehensive theory, heuristics have arisen to fill the void: usually a particular network structure is assumed, for example, a one hidden-layer, sigmoidal network, and if, after a sufficiently long and (assumed) exciting training period, the results are unsatisfactory, the number of hidden layer nodes is increased, and the training is continued (or restarted). This procedure continues until the prediction errors are within the required tolerance.

In addition to the above generalization and learning stability issues, there is a more subtle problem underlying this strategy. A neural network with a finite number of components can provide an approximation with a particular accuracy only for a limited range of input signals; after the training phase this range is implicitly fixed by the size of the network and the weights it has learned. As noted at the end of Section 3.1.3, even the small errors incurred approximating the ideal control law over this range may cause large deviations between the true process output and the model response. If these deviations are such that the outputs of the process leave the set for which network accuracy can be assured, the difference between the network output and the required control will become quite large, potentially forcing the process outputs even further from the accuracy range, and instability may result. The following section examines in greater detail the possible instabilities which can occur when using networks to control dynamic systems.

3.3.3 Stability of Control Laws Using Neural Networks

Assuming a successful and persistently exciting identification phase, the resulting trained networks in the identifier, which now accurately model the dynamics of the
process, can either be used directly as the basis of nonlinear control laws, or used to train separate control networks as detailed in Section 3.1.3. There remain, however, still the questions raised at the end of that section about the ultimate effectiveness of these strategies. In particular, assuming no additional learning occurs when these networks are used to control the actual process, under what circumstances can guarantees of stability and convergence be given for the resulting closed loop behavior?

To approach this problem in its most general form, several intermediate questions must be answered on the basis of the available prior information. Given the process dynamics, does there exist a control law capable of achieving the desired objectives (i.e. tracking a model response) using bounded control inputs? Can a neural network be used to accurately approximate this required control law? What is the effect of the inevitable small perturbations between the network approximation and the ideal control law?

The simplest situation to analyze is when the structure of a control law is known which permits direct utilization of components of the identification model. For example, the continuous time system \( \dot{y}(t) = -f(y(t)) + u(t) \) can be forced to follow the output of a reference model, \( y_m(t) \), by exactly canceling the nonlinearity \( f \). Indeed, a linearizing control law of the form \( u(t) = f(y(t)) + \dot{y}_m(t) - k_D e(t) \) forces the tracking error \( e(t) = y(t) - y_m(t) \) to converge exponentially to zero from any initial condition, since with this control \( \dot{e}(t) = -k_D e(t) \). As discussed in Section 3.1.3, this ideal control law can instead be approximated by using a network which has accurately identified \( f \) over a range of states.

However, by using a network approximation in place of \( f \) in the control law, the convergence properties of the ideal control may not be globally preserved. As an example, consider a linear process with \( f(y) = cy \), and suppose an identification model of the process has been developed using the "network" approximation \( \hat{V}(y, \dot{c}) = \dot{c}y \), such that \( |\hat{V}(y, \dot{c}) - f(y)| \leq \epsilon_f \) everywhere on the set \( A = [-1, 1] \). If the output of the model response is identically zero, i.e. if it is desired to regulate the process about
the origin, the closed loop output of the process evolves according to the differential equation

\[ \dot{y}(t) = \mathcal{N}(y(t), \hat{c}) - cy(t) + \dot{y}_d(t) - k_D(y(t) - y_d(t)) \]

\[ = (\hat{c} - k_D)y(t), \]

where \( \hat{c} = \hat{c} - c \). Since \( \hat{c} \) can be as large as \( \epsilon_f \) and still provide the uniform bound given by the identification model, the feedback gain must at least satisfy \( k_D > \epsilon_f \) to ensure global stability of the closed loop system. The parameters of the controller required to ensure stable operation thus depend upon the approximation capability of the structure chosen to estimate the function \( f \).

The situation is more complex if \( f \) is nonlinear, for example \( f(y) = cy^2 \). Using a "network" in the regulation law of the form \( \mathcal{N}(y, \hat{c}) = \hat{c}y^2 \), trained to approximate \( f \) with the same range and uniform bounds as above, results in the closed loop differential equation \( \dot{y}(t) = \hat{c}y^2(t) - k_Dy(t) \), and it is easy to show that the origin is a stable equilibrium point of this differential equation, whose basin of attraction is \( (-\infty, k_D/\hat{c}) \) if \( \hat{c} > 0 \), and \( (k_D/\hat{c}, \infty) \) if \( \hat{c} < 0 \). The point \( y^* = k_D/\hat{c} \) is an unstable equilibrium, and for initial conditions \( y(0) > k_D/\hat{c} \) when \( \hat{c} > 0 \), or \( y(0) < k_D/\hat{c} \) when \( \hat{c} < 0 \), the process output will diverge to infinity. Thus, provided that \( k_D \) is large enough that the set \( A \) lies within the basin of attraction of the origin, and provided that the initial process condition is contained within this set, the regulated closed loop system will converge to the origin, and the process output will remain within the set \( A \) for all \( t \geq 0 \).

When following a time varying trajectory, however, it may not be possible to ensure that the state remains within \( A \), even if it initially begins inside this set. The tracking errors caused by the mismatch between \( c \) and \( \hat{c} \) in the control law can easily carry the process state outside \( A \), especially if the desired trajectory comes close to the boundary of this set. Suppose then that the set \( A = [-k_D/|\hat{c}|, k_D/|\hat{c}|] \), for example, and that at some time, \( t_0 \), the output \( y(t_0) \) lies a certain distance outside \( A \) as it attempts to follow a model trajectory which lies near its border. If the model
CHAPTER 3. PRIOR DESIGNS AND INSTABILITY MECHANISMS

trajectory suddenly becomes \( y_m(t) = \dot{y}_m(t) = 0 \) for \( t \geq t_0 \), from the above analysis the state may be "stranded" outside the basin of attraction of the origin, and, depending upon the sign of \( \dot{c} \) may diverge to infinity using the above control law. Even if continuity and differentiability constraints are placed on the model trajectory, it is still possible to devise a similar situation with a large enough mistuning \( \dot{c} \).

To prevent this situation from occurring, if \( |\dot{c}(0)| \) can be bounded, one could choose \( k_D \) to make \( A \) very large with respect to the desired trajectories commanded, so that the state has no chance to ever "escape" this set. Otherwise, additional modifications to the structure of the control law are needed, designed to provide additional robustness to the perturbations caused by the network mistuning. In this situation, for example, if the linearizing control law is augmented with the term \( -\bar{c} y^2(t) \sgn(y(t)) \), with \( \bar{c} > |\dot{c}| \), the regulation will again be successful, since the Lyapunov function \( V(t) = y^2(t) \), has time derivative,

\[
\dot{V}(t) \leq -k_D y^2(t) + y^2(t)|y(t)|||\dot{c}| - \bar{c}| \leq -k_D V(t),
\]

and \( y \) thus converges exponentially to the origin from any initial condition.

In the two examples above, the "networks" used did have the capacity to develop globally accurate models of the process dynamics, and would produce stable, convergent closed-loop dynamics if \( \dot{c} = 0 \), without further constraints on \( k_D \) or need for additional robustness measures. In actual network designs, however, even when the adjustable parameters assume their best possible values, there will still be discrepancies between the network output and the required linearizing control, and these discrepancies will become more pronounced outside any compact set \( A \), necessitating a more careful stability analysis of the type given above.

Even in these relatively simple uses of networks in tracking control laws, one again observes a delicate interplay between the approximation capabilities of the chosen network and the designs which ensure stability and convergence. In addition to the modifications required for the training mechanism, a control law which employs a trained network must also be robust to the perturbations introduced by use of neu-
ral network approximations to the functions which would allow theoretically perfect tracking. Similar closed loop stability analyses for more complex nonlinear processes depend crucially upon the availability of appropriate tools from nonlinear control theory (some of which will be introduced below), but will not change the demonstrated importance of the robustness in the control and adaptation mechanisms.

It is hence crucial to be able to quantify the ability of a particular network architecture to approximate the functions required for identification and control applications. This analysis is so basic to the success of these designs that it must form an integral part of any neural network control system design. Accordingly, the next chapter examines in detail network analysis and synthesis techniques before turning, in Chapter 5 and Chapter 6, to identification and control structures which explicitly incorporate the required robustness measures.
Figure 3-3: Technique for using a neural network identification model to train a neural network controller. The trained identifier provides a path through which to "back propagate" the sensitivity of the tracking error to changes in the control network parameters.

<table>
<thead>
<tr>
<th>Example</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ξ_i</td>
<td>-1.250</td>
<td>-1.083</td>
<td>-0.833</td>
<td>-0.750</td>
<td>-0.583</td>
<td>-0.333</td>
<td>-0.250</td>
<td>-0.083</td>
</tr>
<tr>
<td>f(ξ_i)</td>
<td>0.255</td>
<td>0.146</td>
<td>-0.331</td>
<td>-0.424</td>
<td>-0.272</td>
<td>0.829</td>
<td>1.273</td>
<td>1.911</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>ξ_i</td>
<td>0.00</td>
<td>0.167</td>
<td>0.250</td>
<td>0.417</td>
<td>0.500</td>
<td>0.667</td>
<td>0.917</td>
<td>1.167</td>
<td>1.500</td>
</tr>
<tr>
<td>f(ξ_i)</td>
<td>2.000</td>
<td>1.653</td>
<td>1.273</td>
<td>0.380</td>
<td>0.000</td>
<td>-0.414</td>
<td>-0.173</td>
<td>0.237</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 3-4: Data set used to train the network.
Figure 3-5: Total square error and worst case error on training set as a function of training iterations. The deadzone robustness modification results in slightly worse performance on the training set.
Figure 3-6: Worst case errors recorded on the compact set $A = [-2, 2]$ in the Gaussian network approximation to the function underlying the training data. Without robust adaptation this error begins to increase as training continues.
Figure 3-7: Evolution of network output weights. Without the robustness modification, the drift of weight 5 causes the worsening generalization errors seen in Figure 3-6.
Figure 3-8: The drift in the output weights causes the overall fit to the function underlying the data to become much worse as training increases, although behavior near the training points is almost constant. The different approximations were recorded after 400, 800, 1600, 3200, 6400, 12800, and 25600 training cycles. The dashed line is the actual function $f$ which generates the training set data, and the filled triangles are the points in the training set.
Figure 3-9: The modified learning mechanism prevents parameter drift and avoids overtraining, resulting in a higher quality identification model. The approximations were recorded at the same intervals as in Figure 3-8. As before, the dashed line is the actual function $f$ which generates the training set data, and the filled triangles are the points in the training set.
Chapter 4

Gaussian Network Analysis and Synthesis

Given the structure of the network representations considered in Chapter 2, one is motivated to study in detail the following problem: given the representation:

\[ \hat{f}(x) = \sum c_k g(\varphi(x, \xi_k)) \]

when does the table of numbers, \( \{\xi_k\}, \{c_k\} \), together with the functional form of \( g \) and \( \varphi \) serve to characterize a particular class of functions? That is, to what extent can more "interesting" functions be constructed out of the fundamental building blocks \( g \) and \( \varphi \)? As discussed above, for many of the choices \( g \) and \( \varphi \) commonly used in "neural" network applications, uniform approximation to continuous functions on compact subsets of \( \mathbb{R}^n \) is assured with a structure of this type. However, these theorems leave unanswered several important practical questions: for a specified uniform tolerance on a particular subset, exactly how many nodes are required, and how should the weights \( \xi_k \) and \( c_k \) be chosen, to ensure that this theoretical capability is achieved?

In order to evaluate the quality of a particular function approximation strategy, it is necessary to choose a metric which quantifies the severity of the approximation errors. For the applications to be considered in this thesis, uniform approximation, as opposed to mean square or other integral metrics, is the most useful measure of
the quality of an approximation, that is, knowledge of a bound $\epsilon_f > 0$ such that
\[ \sup_{x \in A} |\hat{f}(x) - f(x)| \leq \epsilon_f \]
for a chosen compact subset $A \subset \mathbb{R}^n$. Knowledge of such a uniform bound on the ability of a network to approximate certain classes of functions will allow precise characterization of the resulting asymptotic prediction and tracking errors which result when the network is used in identification and control applications; this will be extensively discussed in Chapters 5 and 6. The current chapter lays the foundation for this discussion by developing network analysis and synthesis techniques which provide constructive answers to the questions posed above. These techniques will be shown to be capable of precisely quantifying the dependence of the degree and extent of the uniform approximation capability on the size of the network and the specific values of its free parameters.

A convenient method for characterizing different function classes is to use the Fourier transform [25, 185], and Fourier analysis will be used extensively in the analysis below, both in its classical ($L^1$ and $L^2$) formulations, and in its distributional formulation [186, 77]. In particular, Fourier analysis immediately provides an answer to the representation problem posed above, and which is intimately familiar to most engineers. Given the Fourier representation, the question of when a discrete set of numbers completely characterizes the behavior of a function is elegantly answered by multidimensional sampling theory [121, 98]. Indeed, sampling theory has deep connections to many of the important techniques of approximation theory, and the excellent survey papers [76, 27, 62] show the remarkable breadth of this important theorem.

Recall that the sampling theorem states that a function whose Fourier transform vanishes outside of a closed and bounded subset of $\mathbb{R}^n$ can be exactly represented in terms of the values it assumes at a countable set of points, with use of an appropriately chosen interpolating function. Further, by spacing these points slightly closer together than ordinarily required ("oversampling"), it is possible to employ
a much larger class of interpolating functions which possess the same reconstruction properties as the canonical interpolator. As will be shown below, this extended class of interpolating functions includes several of the "neural" models currently in widespread use, suggesting that sampling theory can be used to predict and tailor the approximating ability of these networks.

Sampling theory produces an expansion which exactly fits into the above framework, but which requires an infinite number of terms. This is certainly unsatisfactory if the goal is to develop a representation which can be realized with a finite number of components. Further, the class of functions which are Fourier transformable and whose transforms have compact support is relatively small. However, the structure of the exact representation for bandlimited functions produced by the sampling theorem suggests methods by which these assumptions can be relaxed and the number of parameters reduced. In particular, by limiting the reconstruction to a compact subset of $\mathbb{R}^n$, one can explicitly construct a representation which uniformly approximates the original function on this set, and for which only a finite number of sampling points is required. By considered choice of the network parameters, moreover, the approximation error of the resulting expansion can be made vanishingly, and quantifiably, small on the chosen set.

In the sections which follow, these observations are made precise and the conclusions used to suggest designs for a class of analog networks. Section 4.2 reviews the canonical reconstruction of bandlimited multivariable functions, while Section 4.3 discusses how the sampling theorem can be mapped onto the network topology proposed in Chapter 2, and discusses the limitations of this structure for approximation purposes. In Section 4.4, it is shown that by sacrificing globally exact reconstruction in favor of uniform approximation on a chosen compact set, a much larger class of functions can be approximated, while using only a finite number of interpolating functions. Section 4.5 introduces a particular interpolating function, the Gaussian radial basis function which has found increasing use in network designs, and shows how the repre-
CHAPTER 4. GAUSSIAN NETWORK ANALYSIS AND SYNTHESIS

sentation derived from sampling theory also maps naturally onto networks which use this basis function for the hidden layer nodes. Section 4.6 then extends the classical sampling analysis using distribution theory to characterize Gaussian approximations to polynomial and trigonometric functions.

Section 4.7 outlines a procedure for choosing each component of these network architectures to reduce the uniform approximation errors below a prespecified threshold. Section 4.8 then presents a series of examples which demonstrate the effectiveness of the proposed network construction techniques. Finally, Section 4.9 concludes the chapter with some general observations about the analysis and proposed construction procedure.

4.1 The Smoothness Assumption

The central problem in a supervised learning application is to construct a "reasonable" approximation based upon only a limited amount of information about the process generating the observed data. Even if this data can be assumed to be generated by a continuous, but unknown, function, there are clearly still an infinite number of continuous functions which can pass through any given data set. The problem of reconstructing the underlying function on the basis of limited data is thus ill-posed, and additional prior information must be utilized to somehow select the "best" continuous function which fits the available information. Of course, at this point, it is still far from clear that the structure of such a representation will at all resemble the network structures considered in Chapter 2.

The regularization procedure formalizes the heuristic ideas of the above paragraph, and thus attempts to solve this class of learning problems from first principles, that is, without prior assumptions about an appropriate network architecture. This is accomplished by augmenting the usual training set error metric with a measure of the smoothness of a candidate function which would be used to fit the training data.
thus constraining the choice of continuous functions. This smoothness measure is called the stabilizer and is usually an $\mathcal{L}^2(\mathbb{R}^n)$ norm of a weighted combination of the derivatives of the candidate function. Performing the optimization of the augmented cost in the function space $\mathcal{L}^2(\mathbb{R}^n)$ then yields as the structural form of the regularized solution

$$\hat{f}(x) = \sum_{i=1}^{N} c_i g(x, \xi_i)$$

where $g$ is the Green's function associated with the stabilizer, and the $\xi_i$ are the domain points contained in the training data [126]. Choosing the stabilizer to be both rotationally and translationally invariant, the function $g$ is in fact a radial function, and this representation can be easily mapped onto the parallel, distributed processing model outlined in Chapter 2, by taking $\varphi$ as the Euclidean norm on $\mathbb{R}^n$ [26, 126].

An alternate characterization of the smoothness information encoded in the stabilizers used by the regularization method can be given in terms of the spatial Fourier transform:

$$F(\nu) = (\mathcal{F}f)(\nu) \triangleq \int_{\mathbb{R}^n} f(x) E_\nu(-x) \, dx,$$

where $E_\nu(x) = \exp(2\pi j \nu^T x)$ (this same function may also be denoted $E_x(\nu)$ depending upon the desired independent variables). Roughly speaking, large values of $|F|$ for high values of $\|\nu\|$ indicates that $f$ is much less smooth than a function whose transform nearly vanishes for large $\|\nu\|$. In terms of the regularization stabilizers, the more square integrable partial derivatives possessed by the function, the faster $F$ approaches 0 as a function of $\|\nu\|$ [185]: an ideally bandlimited function, for example one for which $F$ identically vanishes outside of some compact cube centered at the origin, can be considered as a limiting case of this asymptotic decay.

Unfortunately, in the applications which will be examined in the chapters to follow, a set of training data is not assumed to be available a priori; on the other hand, potentially new data about the function to be learned will be continually arriving during the learning process. Under these conditions it is difficult to employ the regularized constructions developed above, since the required centers of the ra-
dial functions cannot be specified beforehand, and certainly new centers cannot be allocated in real time to cover a potentially infinitely large data set. Moreover, the regularized solutions minimize an $L^2$ error metric, and it is not immediately obvious how this translates to the uniform approximation capability desired for estimation and control applications. The insight underlying the regularized solutions above can still be exploited, however, by first assuming that the function generating the observed data satisfies certain smoothness constraints, expressed in terms of the decay properties of its Fourier transform, and then investigating the structures required to represent this class of functions to a given uniform accuracy. It is this strategy which will drive the development of the architectures proposed in this chapter.

Throughout the development to follow, it is assumed that the function being approximated is continuous on the subset on which reconstruction is required. This condition is not sufficient, however, to ensure that the above Fourier transform is well defined. Even when it is meaningful, the “inversion” formula:

$$(\mathcal{F}^{-1}F)(\mathbf{x}) \triangleq \int_{\mathbb{R}^n} F(\mathbf{\nu}) E_x(\mathbf{\nu}) \, d\mathbf{\nu}$$

may not, in fact, be the same as $f(\mathbf{x})$ at every $\mathbf{x} \in \mathbb{R}^n$. If, however, both $f$ and $F$ are absolutely integrable, this inversion formula does in fact return the values of the original function at each $\mathbf{x}$ [185], that is $f = \mathcal{F}^{-1}F$. In general, assuming these conditions to ensure that the transform is well defined and that the inversion relation holds for the nonlinearity being approximated provides a convenient starting point; these conditions will then be weakened in the subsequent development. It is important to keep in mind in the Fourier analysis which follows that $f$ is being treated as a multidimensional function of $\mathbf{x} \in \mathbb{R}^n$ for the purpose of computing (or estimating) its spectral characteristics, and not as an implicit function of time, as is more usual in control and filtering applications.
4.2 Classical Sampling Theory

Suppose then that a continuous function, \( f \), is absolutely integrable and that its spatial Fourier transform has compact support. Under these conditions, the transform pair given above is valid [25, 185], and further, \( f \) admits an exact expansion in terms of its samples on an appropriately defined lattice in \( \mathbb{R}^n \) [121]. For simplicity in this exposition, the sampling is taken as uniform with respect to a square lattice, \( \Delta k, k \in \mathbb{Z}^n \), but this is by no means necessary: Section 4.9 will briefly discuss extensions to more general lattices.

The sampling operation can be viewed as a modulation of the hypersurface \( f \) with a field of Dirac distributions centered at the lattice points, i.e.

\[
(4.1) \quad f_s(x) = \psi(x)f(x)
\]

where

\[
\psi(x) = \sum \delta(x - \Delta k)
\]

(summations without explicit limits will be assumed to range over all of \( \mathbb{Z}^n \)). Fourier transforming (4.1), one obtains:

\[
F_s(\nu) = (F * \Psi)(\nu)
\]

where \(*\) indicates convolution and,

\[
\Psi(\nu) = \Delta^{-n}\sum \delta(\nu - \Delta^{-1}k).
\]

Hence

\[
(4.2) \quad F_s(\nu) = \Delta^{-n}\sum F(\nu - \Delta^{-1}k).
\]

implying that the spectrum of the sampled waveform consists of copies of the original spectrum centered at the frequency lattice points, \( \nu = \Delta^{-1}k \) [25, 121].

Let \( \mathcal{K}(\beta) \) be the smallest \( n \)-cube, centered at the origin, which completely encloses the support of \( F(\nu) \) (in general \( \mathcal{K}(\beta) = [-\beta, \beta]^n \) for any \( \beta > 0 \)). If \( \Delta \) is chosen to be
CHAPTER 4. GAUSSIAN NETWORK ANALYSIS AND SYNTHESIS

less than \(1/(2\Delta)\), the copies of \(F(\nu)\) in the sampled spectrum will be non-overlapping, and the canonical reconstruction function, whose spectrum is \(\Delta^n\) on \(\Lambda'\left(\frac{1}{2\Delta}\right)\) and zero elsewhere, can be used to assemble \(f(x)\) on the basis of its samples, since, under these conditions:

\[
F(\nu) = F_s(\nu) G_c(\nu)
\]

where \(G_c(\nu)\) is the spectrum of the canonical reconstructor (Figure 4-1 illustrates in 1-dimension). Inverse transforming this equation leads to the classical result:

\[
f(x) = \sum f(k\Delta) g_c(x - k\Delta)
\]

where \(g_c\) is the tensor product

\[
g_c(x - k\Delta) = \prod_{i=1}^{n} \frac{\sin(\Delta^{-1}\pi (x_i - k_i\Delta))}{\Delta^{-1}\pi (x_i - k_i\Delta)}.
\]

The impulse response of the ideal low-pass filter, \(\frac{\sin(\pi q)}{\pi q}\), is often referred to as the sinc function, and denoted sinc\((q)\).

4.3 Sampling Theory and Feedforward Networks

The elegance of this discovery, that there are classes of functions which can be exactly reconstructed using just the discrete translates of a single function, so moved the mathematician E.T. Whittaker that he proclaimed sinc to be "the function of royal blood in the family of entire functions" [176].

In the univariate case, this representation also has great significance in terms of the parallel, analog networks outlined in the previous chapter, for not only does it map precisely onto these structures, it does so in such a manner as to allow a precise interpretation of every component of the resulting network. That is, if the input function in Figure 2-4 is chosen to be the dilated translation, \(\varphi(x, \xi) = \Delta^{-1}(x - \xi)\), and the node activation function is taken as the sinc function: \(g(q) = \text{sinc}(q)\), then
Figure 4-1: Graphical illustration of the sampling theorem in the frequency domain. Here $f$ is a univariate function whose spectrum is the triangle pictured.
the network implements the expansion

\[ \mathcal{N}(x, \mathbf{e}) = \sum c_k \text{sinc}(\Delta^{-1}(x - \xi_k)). \]

By identifying the input weights with the sample points \( \xi_k = \Delta k \), and the output weights with the samples of the function being approximated, \( c_k = f(\Delta k) \), this structure exactly implements the univariate sampling theorem, and hence can exactly represent any function bandlimited to the interval \([\lambda, \rho] \), \( \beta < \frac{1}{2\Delta} \).

There are two drawbacks to using this insight as the basis for a constructive theory of networks. The first is that, as shown, the canonical reconstructing function in higher dimensions is the tensor product of the univariate sinc functions. This would require use of the more complex, multilayer tensor product net structure discussed in Chapter 2, since the node construction constraints assume the structure of an activation function to be independent of the number of signals input to the network. The second, more serious, problem is one well known to approximation theorists: while the cardinal series has an elegantly simple structure, it has very poor convergence properties. To quote I.M. Schoenberg, who attempted to use this expansion to approximate aerodynamic functions during WWII, "its excessively slow rate of damping makes the classic cardinal series inadequate for numerical purposes" [151].

That is, to be practical, and indeed implementable on a network, only a finite number of neighboring sample points should be required to compute an approximation to \( f(x) \) accurate to a chosen tolerance,

\[ |f(x) - \sum_{|x - \Delta k| \leq \rho} f(\Delta k)g(\varphi(x, \Delta k))| \leq \epsilon. \]

In the classical sampling theorem, however, very large values of \( \rho \), and hence a great number of terms in the series (or nodes in the network), are required to achieve reasonably small values of \( \epsilon \). The central problem lies with the decay properties of the reconstructing function \( g(q) = \text{sinc}(q) \); this function is not well localized, indeed it is not even absolutely integrable, and hence in most cases the series converges very slowly.
Schoenberg's proposed solution to this problem was to develop function approximation methods which, like the sampling theorem, employ translates of a single prototype function, but ones which exhibit far better spatial localization than sinc. The spline approximation methods which emerged from his analysis will be mentioned further below, for now his insight will be used to guide an exploration of methods which address both the convergence and the network equivalence problem in the sampling theorem. Perhaps it is possible to identify multidimensional low-pass filters which produce a simpler, one hidden layer network structure in higher dimensions, and which are also well localized in space so as to assure rapid convergence of the corresponding cardinal series. These new filters may introduce some error into the exact representation otherwise provided by sampling theory, which holds only for strict low-pass filters, but since approximation errors will occur in any case by truncating the series expansion, these new error sources can be tolerated so long as they are small compared to the reduction of truncation error afforded by the increased spatial localization.

4.4 Generalized Sampling Theory

4.4.1 Oversampling and General Low Pass Filters

If a bandlimited hypersurface, \( f \), is oversampled, i.e. if \( \Delta < 1 / (2\beta) \), then \( \mathcal{K}(\beta) \subset \mathcal{K}_\Delta \), where \( \mathcal{K}_\Delta = \mathcal{K}(\frac{1}{\Delta} - \beta) \), and the spectrum of the required interpolating function is actually arbitrary on \( \mathcal{K}_\Delta - \mathcal{K}(\beta) \). Oversampling thus gives the spectrum of the reconstructing function room to approach zero more smoothly than in the classical case, which in turn gives this function better spatial localization properties, since generally, the more absolutely integrable derivatives possessed by the spectrum of this function, the faster it will approach zero. More precisely, repeated application of the Riemann-Lebesgue lemma shows that, if all partial derivatives of a spectrum \( G \) of order up to \( m \) exist and are absolutely integrable, the interpolating function \( g \)
satisfies [132]
\[
\lim_{\|x\| \to \infty} \|x\|^m |g(x)| = 0.
\]

Furthermore, there is no reason to require the coefficients of the cardinal series to be exactly the samples of \( f \) itself. With this restriction removed, the structure of equality (4.3) suggests that a wide range of interpolating functions can also be used for the reconstruction. In fact, as long as its spectrum is bounded, real valued and strictly positive on \( \mathcal{K}(\beta) \), and vanishes outside of \( \mathcal{K}_\Delta \), any Fourier transformable function can be used in place of \( g_c \) in expansion (4.4), by changing the coefficients in this series to the samples of a different bandlimited function, related to \( f \) in a straightforward manner.

To see this, assume that \( g_1(x) \) is such a function, and using its spectrum, define a new function \( c(x) = (\mathcal{F}^{-1}C')(x) \), where:

\[
(4.5) \quad C'(\nu) = \Delta^n F(\nu) G_1^{-1}(\nu).
\]

Since, by assumption \( G_1(\nu) \) is strictly positive and real on \( \mathcal{K}(\beta) \), this new spectrum is well-defined; further, \( C'(\nu) \) is bounded since both \( F(\nu) \) and \( G_1^{-1}(\nu) \) are on \( \mathcal{K}(\beta) \). Hence \( C'(\nu) \) is absolutely integrable, and its inverse transform \( c(x) \) is a bounded, continuous function.

Consider applying the above sampling algorithm to \( c(x) \). It is straightforward to verify from the definitions of \( C'(\nu) \) and \( G_1(\nu) \) that:

\[
(4.6) \quad F(\nu) = C_s(\nu) G_1(\nu)
\]

where \( C_s(\nu) \) is the sampled spectrum of \( c(x) \), defined as in (4.2). Inverse transforming this equation reveals an exact expansion of \( f(x) \) in terms of the samples of the bounded function \( c(x) \):

\[
f(x) = \mathcal{F}^{-1}(C_s G_1)(x) = c_s(x) * g_1(x) = (c(x) * \nu(x)) * g_1(x)
\]

thus,

\[
(4.7) \quad f(x) = \sum c(k_\Delta)g_1(x - k_\Delta).
\]
Figure 4-1 illustrates these ideas for a one-dimensional function with a spectrum \( F(\nu) = 1 - |\nu| \) for \( |\nu| \leq 1 \) and \( F(\nu) = 0 \) otherwise. The sampling mesh in this figure is \( \Delta = 1/2 \) and the canonical reconstructing spectrum \( G_c(\nu) = 1/2 \) for \( |\nu| \leq 1 \), and \( G_c(\nu) = 0 \) otherwise is illustrated. Figure 4-2 illustrates reconstruction of the same function using instead an interpolating function whose spectrum is \( G_1(\nu) = \exp(-\nu^2) \).

The function \( C(\nu) \) is given in this example by

\[
C(\nu) = \frac{1}{5} \exp(\nu^2)(1 - |\nu|) \quad \text{for} \quad |\nu| < 1, \quad \text{and} \quad C(\nu) = 0 \quad \text{otherwise.}
\]

which is then oversampled with mesh \( \Delta = 1/5 \).

Notice that the spectrum \( G_1 \) does not vanish identically outside of \( \mathcal{K}_\Delta \) in this case, however at the beginning of the first copy in the repeating spectrum of \( c(x) \) one has \( G_1 = \exp(-16) \approx 10^{-7} \), so that the spurious high frequency components are almost completely negligible in the reconstruction. This suggests that it is possible to extend the class of interpolating functions still further, if vanishingly small approximation errors are allowed. The next section makes this observation precise, and identifies the exact source of the approximation errors introduced.

### 4.4.2 Approximation on Compact Sets

In this section it is shown that by sacrificing the globally exact representation afforded by expansion (4.7), it is possible to approximate a much larger class of functions than considered in the previous section, using a wider range of interpolating functions. The principal result demonstrated is that if the function can be smoothly truncated outside a compact set in such a way that the resulting spatial Fourier transform is absolutely integrable, then the function itself can be uniformly approximated on that set with a finite linear superposition of appropriately chosen interpolating functions.

As noted above, the actual nonlinearity \( f \) to be approximated may not be Fourier transformable, or if transformable, the inversion equation might not be well defined. However, if \( f \) is absolutely integrable, with an absolutely integrable spectrum, then
Figure 4-2: Illustration of Gaussian reconstruction in the frequency domain. By sampling a slightly different function, whose spectrum is at top left, then using a Gaussian low pass filter, the triangular spectrum of the original function $f$ is recovered. There are residual high frequency components since the Gaussian filter does not vanish identically for high frequencies, but these are invisible at the scale pictured.
the Fourier transform pair above is valid [185]. The absolute integrability of the spectrum reflects local properties of the function capturing its degree of smoothness; the absolute integrability of the function itself, however, is a global condition which is easily relaxed. Since approximation is required only on a chosen compact set, \( A \), if the original function \( f \) does not satisfy the integrability condition, consider instead an absolutely integrable function, \( f_F \) (read "\( f \) Fourier"), which is exactly equal to \( f \) on \( A \); i.e.,

\[
f_F(x) = m_1(x)f(x)
\]

where \( m_1(x) \) is an infinitely smooth function which is unity on \( A \) and outside this set decays to zero sufficiently rapidly that the product \( |m_1f| \) is integrable. For example, if, outside of \( A \), \( |f| \) has a rate of growth which can be polynomially bounded, a convenient form for \( m_1 \) is a Gaussian,

\[
m_1(x) = \exp\left(-\frac{d_x^2}{\sigma^2}\right).
\]

where \( d_x = \text{dist} (A, x) = \inf_{y \in A} ||x - y|| \). One thus has

\[
|f(x) - f_F(x)| = |1 - m_1(x)||f(x)| = \alpha_1(x)
\]

where \( \alpha_1(x) = 0 \) if \( x \in A \), and otherwise \( \alpha_1(x) \leq |f(x)| \).

While the transform of \( f_F \) exists, it will not in general be compactly supported, approaching zero only asymptotically. In order to extend the class of interpolating functions which can be used, allow the same to be true of the transforms of the functions \( g_1 \). Under these conditions, application of the sampling theorem will provide representations which are only approximately accurate. As is now demonstrated, however, if the rates of decrease of \( |F_F(\nu)| \) and \( |G_1(\nu)| \) are sufficiently fast as \( ||\nu|| \to \infty \), choice of a sufficiently fine sampling mesh on \( \mathbb{R}^n \) results in an expansion, \( f_A(x) \), in terms of the chosen interpolating functions \( g_1 \), which can uniformly approximate the actual function to a specified accuracy on a given compact set, i.e. \( |f_F(x) - f_A(x)| \leq \epsilon_f \) for all \( x \in A \).
To prove this assertion, define $f_{BL}(x)$ ("f bandlimited") to be the function which would obtain by truncating the spectrum of $f_F$ at a particular spectral radius, i.e.

$$f_{BL}(x) = F^{-1}(F_F \cap \beta)(x)$$

where $\chi_{\beta}(\nu)$ is the characteristic function of the hypercube $\mathcal{K}(\beta)$ in the frequency domain, that is, $\chi_{\beta}(\nu) = 1$ if $\nu \in \mathcal{K}(\beta)$, and $\chi_{\beta}(\nu) = 0$ otherwise. It is then easy to see that

$$|f_F(x) - f_{BL}(x)| \leq \int_{\mathcal{K}(\beta)^c} |F_F(\nu)|d\nu \leq \epsilon_c$$

where $\mathcal{K}(\beta)^c$ is the complement of the set $\mathcal{K}(\beta)$ in $\mathbb{R}^n$, i.e. $\mathcal{K}(\beta)^c = \mathbb{R}^n - \mathcal{K}(\beta)$. Hence, if the spectrum $F_F(\nu)$ is absolutely integrable on $\mathbb{R}^n$, it is possible to choose the spectral truncation radius $\beta$ so that the error $\epsilon_c$ is as small as desired, uniformly on $\mathbb{R}^n$.

Given this value of $\beta$, choose a $g_1(x)$ which satisfies the above requirements for reconstructing $f_{BL}(x)$, that is, $G_1(\nu)$ is bounded, real, and positive for $\nu \in \mathcal{K}(\beta)$, but perhaps approaches zero only asymptotically outside of $\mathcal{K}(\beta)$, and define $C(\nu)$ by applying (4.5) to $f_{BL}(x)$. Sampling the resulting function $c(x)$ with a mesh size $\Delta < 1/(2\beta)$, produces an expansion, $f_R(x)$ ("f reconstructed"), which can be expressed (using (4.7)) as:

$$f_R(x) = \sum c(\Delta k)g_1(x - \Delta k) = \int_{\mathbb{R}^n} G_1(\nu)C_1(\nu)E_x(\nu)d\nu$$

$$= \int_{\mathcal{K}(\beta)} G_1(\nu)G_1^{-1}(\nu)F_{BL}(\nu)E_x(\nu)d\nu$$

$$+ \int_{\mathcal{K}(\beta)} G_1(\nu) \left[ \Delta^{-n} \sum_{\mathbf{k} \neq 0} C(\nu - \Delta^{-1}\mathbf{k}) \right] E_x(\nu)d\nu$$

$$+ \int_{\mathcal{K}(\beta)} G_1(\nu) \left[ \Delta^{-n} \sum_{\mathbf{k} \neq 0} C(\nu - \Delta^{-1}\mathbf{k}) \right] E_x(\nu)d\nu$$

where the definition of $C(\nu)$ has been used.

Now, the first integral is exactly $f_{BL}(x)$, since $F_{BL}(\nu)$ vanishes outside of $\mathcal{K}(\beta)$ by construction, and the choice of the sampling rate ensures that the second integral...
is identically zero. Thus 

\[ f_R(x) - f_{BL}(x) = d_a(x) \]

where \( d_a(x) \) is an “aliasing” error incurred by approximating a bandlimited function \( f_{BL}(x) \) with a interpolating function which is not an ideal low-pass filter, and is given by the integral

\[ d_a(x) = \int_{\kappa*_{\Delta}} G_1(\nu) \left[ \Delta^{-n} \sum_{l \in \mathbb{Z}^n} C'(\nu - \Delta^{-1}l) \right] E_x(\nu) d\nu. \]

This use of the term aliasing is a bit nonstandard, since aliasing is usually a phenomenon associated with undersampling, occurring when the repeating spectra overlap. The terms contained in \( d_a \) represent in a sense a problem dual to aliasing, and [189] propose the new term impostor to describe these disturbances, but this does not appear to be in wide use. Thus the definition of the term “aliasing” will be broadened below to indicate any distortion to exact reconstruction caused by the repeating spectra.

Since \( G_1(\nu) \) is bounded away from zero on \( \kappa(\beta) \), and since, by the choice of mesh size, the repeating spectra in the summation are non-overlapping, one has conservatively

\[ |\Delta^{-n} \sum_{l \in \mathbb{Z}^n} C'(\nu - \Delta^{-1}l)| \leq \sup_{\nu \in \kappa(\beta)} |F_{BL}(\nu)G_1^{-1}(\nu)| \triangleq \kappa_C \]

for each frequency point \( \nu \). Using the fact that \( G_1 \) is always positive produces the conservative bound,

\[ |d_a(x)| \leq \kappa_C \int_{\kappa*_{\Delta}} G_1(\nu) d\nu \triangleq \epsilon_a \] (4.10)

so that if \( G_1(\nu) \) is also absolutely integrable it is possible to chose a \( \Delta \) (which defines the domain of integration \( \kappa*_{\Delta} \)) such that \( \epsilon_a \) is as small as desired. Hence, by appropriate choice of the mesh size, \( \epsilon_a \) can be made arbitrarily small, again uniformly on \( \mathbb{R}^n \).

Finally, truncation error of series (4.9) with absolute magnitude less than or equal to a prespecified tolerance, \( \epsilon_t \), can be achieved at each \( x \) by omitting the terms
corresponding to samples \( \Delta^{-1}k \) which lie outside an \( n \)-cube of radius \( \rho \) surrounding \( x \). To achieve this level of approximation everywhere on a particular compact set \( A \), take the union of all such \( n \)-cubes centered at each point \( x \in A \). This union defines a new region which encloses only a finite number of mesh points used in the reconstruction. Designating by \( f_A(x) \) the truncated series obtained by restricting the summation in the expansion of \( f_R(x) \) to these nodes, i.e.

\[
(4.11) \quad f_A(x) = \sum_{\text{dist}(A,k\Delta) \leq \rho} c(k\Delta)g_1(x - k\Delta),
\]

leads to the truncation error

\[
(4.12) \quad d_t(x) = \sum_{\text{dist}(A,k\Delta) > \rho} c(k\Delta)g_1(x - k\Delta),
\]

where the range of these summations is defined implicitly by the distance metric

\[
\text{dist}(A,k\Delta) \overset{\Delta}{=} \inf_{z \in A} \|z - k\Delta\|_\infty.
\]

For a given \( \epsilon_t \), of course, the \( \rho \) required to ensure that \( |d_t(x)| \leq \epsilon_t \) for all \( x \in A \) will be extremely sensitive to the decay properties of the specific interpolating function employed. As will be demonstrated below, one of the advantages of the radial Gaussian is the rapid convergence of the series expansion, permitting low values of \( \rho \) (relative to the mesh size) to be used in the truncation.

Using these definitions, the truncation error can also be written as

\[
(4.13) \quad |f_R(x) - f_A(x)| \leq \epsilon_t + \alpha_2(x),
\]

where \( \alpha_2(x) = 0 \) if \( x \in A \), and when \( x \in A^c \), \( \alpha_2(x) \) provides an upper bound on the approximation error resulting from the terms omitted in the series expansion. It follows that, using (4.4.2), (4.8), (4.10), and (4.13):

\[
|f(x) - f_A(x)| \leq |f(x) - f_F(x)| + |f_F(x) - f_{BL}(x)| + |f_{BL}(x) - f_R(x)| + |f_R(x) - f_1(x)| \leq \alpha_1(x) + \epsilon_c + \epsilon_a + \epsilon_t + \alpha_2(x) = \epsilon_f + \alpha(x)
\]
where \( \alpha = \alpha_1 + \alpha_2 \) is such that \( \alpha(x) = 0 \) if \( x \in A \).

The above analysis thus specifies a constructive methodology for uniformly approximating continuous functions observing fairly mild smoothness constraints. The approximations thereby developed provide a prespecified accuracy everywhere on a compact subset of \( \mathbb{R}^n \) using only a finite number of components. It should be noted that this construction can be applied even to what appear to be pathologically degenerate examples, for example

\[
f(x_1, x_2) = \frac{\sin(4\pi x_1)}{\pi x_1}
\]

since this expression can be seen as a product of a bandlimited function of \( x_1 \), and a function of \( x_2 \) which is constant (in this case unity), and can hence be approximated (albeit inefficiently) on a compact subset of \( \mathbb{R}^2 \) by the methods detailed above. Section 4.6 will extend this idea still further by deriving similar methods to directly approximate non-integrable functions, such as constants and polynomials, without explicitly computing the smooth space-frequency truncation required above.

### 4.5 Generalized Sampling and Gaussian Networks

To be an effective low-pass filter, and thus to be capable of well approximating the sampling theorem, requires that the spectrum of the interpolating function be well localized about the origin in frequency. On the other hand, rapid convergence of the corresponding cardinal series requires good localization about the origin in space. These observations can be quantified by defining the space and frequency localization measures of a reconstructing function \( g : \mathbb{R} \to \mathbb{R} \) (assuming, for clarity, that the function and its transform are even functions)

\[
w(g)^2 \triangleq \|g\|^{-2} \int_{\mathbb{R}} x^2 |g(x)|^2 dx
\]

\[
w(G)^2 \triangleq \|G\|^{-2} \int_{\mathbb{R}} \nu^2 |G(\nu)|^2 d\nu.
\]
whenever each of the above $L^2$ norms is well-defined. However, these two quantities are linked by the uncertainty relation [35]

$$w(g)w(G) \geq \frac{1}{4\pi}.$$  

It is thus impossible to independently reduce both quantities to arbitrarily small levels: good spatial concentration implies a slowly decaying spectrum, and conversely a concentrated spectrum implies a slowly decaying reconstruction function.

Among all possible functions for which these localization quantities are well defined, the lower bound in the uncertainty relation is attained if and only if $g(x) = e^{-\pi \sigma^2 x^2}$ [35, 52] (this representation is slightly non-standard in the placement of $\sigma^2$ for reasons to be explained below). That is, of all low-pass filters, a Gaussian achieves the best possible space-frequency localization tradeoff. Taking tensor products, similar arguments can be made in multiple dimensions [43], in which case $g$ is the Gaussian radial basis function. These are quite attractive results, especially given the widespread popularity of the radial Gaussian function in neural network models [26, 126].

In fact, the suitability of the radial Gaussians for functions of this class has also been demonstrated in the regularization setting. Functions possessing a Fourier transform with compact support are infinitely differentiable [8], and a regularized solution to the function approximation problem using a stabilizer which penalizes a convergent sum of derivatives of all orders yields a linear superposition of Gaussian radial basis functions [126]. Moreover, as discussed above, [54] have proven that any continuous function, not necessarily infinitely smooth, can be uniformly approximated on compact subsets of $\mathbb{R}^n$ by linear combinations of radial Gaussian functions.

Seen in the context of sampling theory, Gaussian radial basis functions are particularly attractive: they are bounded, strictly positive and absolutely integrable on $\mathbb{R}^n$, and further, they are their own Fourier transforms (modulo a scale factor). This function thus satisfies each of the criteria discussed in the previous section, and hence can be used as the reconstructing function in expansion (4.11). Furthermore, unlike other reconstructing functions which satisfy the constraints discussed in Section 4.4, for
example, tensor products of sinc functions, whose structure changes depending upon the dimension of the underlying space, the radial Gaussian filter and its spectrum have exactly the same functional representation regardless of dimension [25]:

\[
\exp\left(\frac{-\pi \nu^T \nu}{\sigma^2}\right) \leftrightarrow \sigma^n \exp\left(-\pi \sigma^2 x^T x\right).
\]

so that, in particular, the Gaussian low-pass filter response is

\[
G(\nu) = \exp\left(\frac{-\pi \nu^T \nu}{\sigma^2}\right).
\]

Hence, if the smooth truncation of \( f \) outside a compact set produces a function with a bounded, absolutely integrable spatial Fourier transform, \( f \) can be uniformly approximated on that set using a finite number of evaluations of the radial Gaussian:

\[
f_A(x) = \mathcal{V}(x, c) = \sum_{\text{dist}(A, k\Delta) \leq \rho} c_k \exp(-\pi \sigma^2 \|x - k\Delta\|^2)
\]

where \( c_k = \sigma^n c(\Delta k) \), and \( c(x) \) is defined by (4.5) using the Gaussian reconstructing filter in (4.15). Here \( \Delta k \) is the "center" of each radial Gaussian, and \( \sigma^2 \) is a measure of its essential "width". The only change which occurs in this structure as the dimension of \( x \) increases is in the computation of the inner product, resulting in an additional multiplication and addition in the local input function.

The radial Gaussian thus satisfies both the approximate low-pass filter constraints and the network construction constraints, and hence the Gaussian sampling expansion maps directly onto the network topology discussed in Chapter 2. There is one hidden layer node for every term in the truncated series, and thus the number of nodes in the network, \( S \), is equal to the number of elements in the index set \( \{k | \text{dist}(A, k\Delta) \leq \rho\} \). The current point, \( x \), at which the approximation \( f_A(x) \) is required, forms the input to the network, and the different input weights of each Gaussian node encode a subset of the uniform lattice \( k\Delta \). The local input function of each node computes a quadratic activation energy from the network inputs and its local weight set, given by

\[
r_k = \varphi(x, \Delta k) = \|x - k\Delta\|^2 = (x - k\Delta)^T(x - k\Delta).
\]
while the activation function computes the exponential of this scalar signal, \( g(r_k) = \exp(-\pi \sigma^2 r_k) \). The network output is formed by multiplying each nodal output by the corresponding \( c_k \), and summing together all the weighted outputs; the result represents the approximation \( f_A(x) \).

Note the placement of the parameter \( \sigma^2 \) in (4.15): as will be shown in Section 4.7, when designing a network it is convenient to use the canonical form for the Gaussian (the left hand side of (4.15) ) in the frequency domain representation. Although there is a discrepancy of \( \pi/2 \) between the formally defined variance of the frequency domain representation and the parameter \( \sigma^2 \), for convenience the word "variance" will refer to the parameter \( \sigma^2 \) as it appears in both the above frequency and spatial representations of the radial Gaussian. Note especially that each node has the same variance in these network designs.

### 4.6 Polynomial and Trigonometric Approximation

While one could use the above analysis to approximate constants and polynomials on compact subsets of \( \mathbb{R}^n \), it is possible to avoid the space/frequency truncation arguments which would otherwise be required and directly analyze the ability of the Gaussian to approximate such functions. As will be shown below, just as the use of the Gaussian on a regular grid closely approximates the sampling theorem for bandlimited functions, so too it closely approximates cardinal spline expansions, permitting the development of another synthesis tool for these networks.

It is a remarkable fact that networks composed of cardinal translates of certain radial basis functions can exactly reproduce multivariate polynomials, as demonstrated by Powell and his co-workers over the past several years [132]. Unfortunately, this analysis immediately dismisses the Gaussian as a practical alternative since it "is so impoverished that it cannot even provide an approximation which is a nonzero constant" [132]. While this observation is certainly true, the aliased "ripple" which
characterizes the departure of the Gaussian approximation from a constant can be made arbitrarily small by proper choice of mesh size and variance. Further, the constructions which allow exact polynomial reconstruction using other types of radial functions depend intimately upon the entire infinite grid to achieve their results, and it is not clear how well this analysis fares when the grid is truncated to a finite number of nodes.

This section thus analyzes the ability of Gaussian radial basis functions to approximate functions which are multivariate polynomials, trigonometric polynomials, or constants. To facilitate the discussion, the standard multi-index notation will be used; i.e. for \( m \in \mathbb{Z}_+^n \).

\[
x^m = \prod_{i=1}^{n} x_i^{m_i}, \quad |m| = m_1 + m_2 + \cdots + m_n
\]

\[
D^m f = \frac{\partial^{|m|} f}{\partial x_1^{m_1} \partial x_2^{m_2} \cdots \partial x_n^{m_n}} \quad f^{(m)} = D^m f
\]

\[
\binom{m}{\alpha} = \prod_{i=1}^{n} \binom{m_i}{\alpha_i}
\]

More general properties of Dirac distributions will be required below than those used in the preceding sections. To prepare for this extended analysis, recall again that the multidimensional Dirac distribution is defined by

\[
\int_{\mathbb{R}^n} \delta(x) f(x) dx = f(0)
\]

for all testing functions \( f \) belonging to the Schwartz space, \( \mathcal{S} \), of rapid decrease [186]. A periodic field of such distributions arranged on a regular grid with a mesh size of \( \Delta \) can be expanded in a Fourier series with coefficients

\[
a_k = \Delta^{-n} \int_{\Lambda(\frac{1}{\Delta})} \delta(x) e^{2\pi i x \cdot \frac{k}{\Delta}} dx
\]

\[
= \Delta^{-n}
\]

and hence one obtains the distributional equality [186]

\[
\sum \delta(x - k\Delta) = \Delta^{-n} \sum e^{2\pi i x \cdot \frac{k}{\Delta}}.
\]
CHAPTER 4. GAUSSIAN NETWORK ANALYSIS AND SYNTHESIS

Since the expressions for the truncation error of the Gaussian expansions to be examined below is identical to (4.12) above, only expressions for the aliasing error will be developed in this section.

4.6.1 Polynomial Approximation

Suppose that a monomial \( f(x) = x^m \) is sampled, and that this procedure is again modeled as a modulation by a field of Dirac distributions. Noting that for any \( \mathbf{k} \in \mathbb{Z}^n \) (abusing the \( \mathcal{F} \) operator notation slightly for clarity)

\[
\mathcal{F}(x^m e^{2\pi j \Delta^{-1} x^T \mathbf{k}})(\nu) = (-2\pi j)^{-|m|} \delta^{(m)}(\nu - \Delta^{-1} \mathbf{k}),
\]

where the distributional derivative \( \delta^{(m)} \) is such that

\[
\int_{\mathbb{R}^n} \delta^{(m)}(x) \psi(x) dx = (-1)^{|m|} (D^m \psi)(0),
\]

for any \( \psi \in \mathcal{S} \) [152], then since Fourier transformation is a continuous, linear operation over the space of tempered distributions [186], the generalized transform of the sampled function can be expressed as

\[
F_s(\nu) = (\mathcal{F} x^m \sum \delta(x - \Delta \mathbf{k})) \psi(\nu)
= \Delta^{-n} (\mathcal{F} \sum x^m e^{2\pi j x^T \mathbf{k}})(\nu)
= \Delta^{-n} \sum (\mathcal{F} x^m e^{2\pi j x^T \mathbf{k}})(\nu)
= \Delta^{-n} (-2\pi j)^{-|m|} \sum \delta^{(m)}(\nu - \Delta^{-1} \mathbf{k})
\]

If, as in the previous sections, the sampled spectrum is “filtered” using a low-pass filter \( G \in \mathcal{S} \), such as the Gaussian, application of the above formulae yields

\[
\int_{\mathbb{R}^n} G(\nu) F_s(\nu) E(x)(\nu) d\nu = \Delta^{-n} (-2\pi j)^{-|m|} \sum (D^m G E)(\Delta^{-1} \mathbf{k}).
\]

Since \( G \in \mathcal{S} \) and the magnitude of the indicated derivatives can be bounded by a polynomial times \( G \), the summation is well defined at each \( x \in \mathbb{R}^n \). On the other
hand, using the same technique as in Section 4.2
\[
\int_{\mathbb{R}^n} G(\nu) F_s(\nu) E_x(\nu) d\nu = (g \ast f_s)(x) = \sum f(\Delta k) g(x - k \Delta)
\]
provided the decay of \( g \) is fast enough to ensure absolute convergence of the series at every \( x \) \([152]\); note that if \( f \) is a polynomial and \( g \) is a Gaussian, this condition is certainly satisfied.

Together with the previous result, this implies that if \( f \) is an arbitrary polynomial
\[
f(x) = \sum_{0 \leq m \leq M_i} \kappa_m x^m
\]
by defining the operator
\[
p_f(D) = \sum_{0 \leq m \leq M_i} \kappa_m (2\pi j)^{-|m|} D^m
\]
one has
\[
\sum f(\Delta k) g(x - \Delta k) = \Delta^{-n} \sum (p_f(D) G E_x)(\Delta^{-1} k).
\]
Since each term of the summation can be expanded using the Leibniz formula
\[
(D^m G E_x)(\nu) = E_x(\nu) \sum_{0 \leq \alpha_i \leq m_i} \binom{m}{\alpha} (2\pi j)^{|\alpha|} (D^{m-\alpha} G)(\nu) x^\alpha,
\]
if \( G \) is such that \( G(0) = \Delta^n, G^{(\alpha)}(0) = 0 \) for all \( \alpha \neq 0, \alpha_i \leq M_i \), and \( G^{(\alpha)}(\Delta^{-1} k) = 0 \) for all \( 0 \leq \alpha_i \leq M_i \), then the terms \((p_f(D) G E_x)(\Delta^{-1} k)\) are identically zero, except at \( k = 0 \) where
\[
\Delta^{-n} (p_f(D) G E_x)(0) = \Delta^{-n} \sum_{0 \leq m \leq M_i} \kappa_m (2\pi j)^{-|m|} (D^m G E_x)(0) = \Delta^{-n} \sum_{0 \leq m \leq M_i} \kappa_m (2\pi j)^{-|m|} [G(0) x^m] = f(x).
\]
Thus, under these conditions, exact reconstruction of the polynomial \( f \) is possible based upon its samples, since
\[
f(x) = \sum f(\Delta k) g(x - \Delta k).
\]
CHAPTER 4. GAUSSIAN NETWORK ANALYSIS AND SYNTHESIS

The above constraints on the spectrum of the reconstructing function are often called the "Strang and Fix" [164] conditions for polynomial reproduction, although they were also known to Schoenberg for the one-dimensional case [151]. Indeed, a tensor product B-spline, \( s_m(x) = s_m(x_1) s_m(x_2) \cdots s_m(x_n) \), where

\[
\begin{align*}
    s_m(x) &= s_0(x) * s_m-1(x) \\
    s_0(x) &\triangleq \chi_{[-\Delta/2,\Delta/2]}(x)
\end{align*}
\]

has Fourier transform

\[
S_m(\nu) = \prod_{i=1}^{n} \left( \frac{\sin(\pi \Delta \nu)}{\pi \nu} \right)^{m+1},
\]

and thus clearly satisfies the required constraints. Hence the translates of \( s_m \) on a regular lattice can exactly reproduce any polynomial of degree up to \( m \) in each variable.

The inadequacy of the Gaussian for polynomial reproduction is, however, now apparent, since only its odd derivatives vanish at the origin, and none of its derivatives identically vanish away from the origin. However, if the goal is only polynomial approximation, not exact reproduction, the situation is not as grim. The effect of the non-zero derivatives of the Gaussian at the origin can be offset by sampling a slightly different polynomial of the same order, in exactly the same way that approximate reconstruction of bandlimited functions required sampling a distorted version of the original function. The non-zero derivatives away from the origin then introduce an approximation error equivalent to the aliasing error in the bandlimited case; since these derivatives decrease rapidly in magnitude with \( \| \nu \| \), proper choice of the mesh size can reduce them to a specified threshold on any compact set.

That is, by proper choice of polynomial \( c. f(x) = \Delta^{-n} \sigma^{-n} (p_c(D) G E_x)(0) \), where \( \sigma^{-n} G(\nu) \) is the Fourier transform of the radial Gaussian, so that

\[
\sum c(\Delta k) \exp(-\pi \sigma^2 \| x - k \Delta \|^2) = f(x) + d_n(x)
\]

and the second term is the aliasing error whose dependence upon the choice of mesh size and variance will be quantified below. To see this, assume \( f \) is a polynomial as
above, and let

\[(4.17) \quad c(x) = \sigma^n \Delta^n \sum_{0 \leq m_i \leq M_i} \kappa''_{m} x^m \]

(as before, the definition of the required weights \(c\) includes the constant \(\sigma^n\) arising in the inverse transform of the Gaussian low-pass filter). To reproduce \(f\) (modulo the aliasing error), the coefficients of \(c\) must be chosen to satisfy

\[
\sum_{0 \leq m_i \leq M_i} \kappa_m x^m = \sigma^n \Delta^n (p_c(D)G E_x)(0) = \sum_{0 \leq m_i \leq M_i} \kappa''_{m}(2\pi j)^{-|m|}(D^m G E_x)(0) = \sum_{0 \leq m_i \leq M_i} \kappa''_{m} \sum_{0 \leq \alpha_i \leq m_i} \binom{m}{\alpha}(2\pi j)^{\alpha-|m|}(D^{m-\alpha} G)(0)x^\alpha.
\]

Collecting the coefficients of like powers of \(x\), this equation can be satisfied by choosing \(\kappa''_{m}\) to simultaneously satisfy

\[(4.18) \quad \kappa_m = \sum_{m_i \leq \alpha_i \leq M_i} \kappa''_{m} \binom{\alpha}{m}(2\pi j)^{|m|-|\alpha|}(D^{\alpha-m} G)(0).\]

Working backwards from \(m_i = M_i\), these equations can be uniquely solved for the coefficients of the required polynomial. Notice that, since all odd partial derivatives of the radial Gaussian vanish at the origin, all coefficients of this polynomial will be real.

The aliasing error is then given explicitly by

\[(4.19) \quad d_a(x) = \sigma^n \Delta^n \sum_{k \neq 0} (p_c(D)G E_x)(\Delta^{-1} k).\]

Substituting the definition of \(p_c\) and expanding,

\[
d_a(x) = \sum_{k \neq 0} E_x(\Delta^{-1} k) \sum_{0 \leq m_i \leq M_i} \kappa''_{m}(2\pi j)^{-|m|} \sum_{0 \leq \alpha_i \leq m_i} \binom{m}{\alpha}(2\pi j)^{|\alpha|}(D^{m-\alpha} G)(\Delta^{-1} k)x^\alpha
\]

and hence,

\[(4.20) \quad d_a(x) = \sum_{0 \leq m_i \leq M_i} x^m \sum_{k \neq 0} E_x(\Delta^{-1} k)p_m(\Delta^{-1} k)G(\Delta^{-1} k),\]
where each \( p_m \) is the multivariate polynomial,

\[
p_m(\nu) = \sum_{m, M \leq M_0 \leq M} \frac{\alpha}{m!} (2\pi j)^{|m| - |\alpha|} H_{\alpha - m}(\nu)
\]

and the \( H_m \) are Hermite polynomials defined so that

\[
H_m(\nu) = G^{-1}(\nu)(D^m G)(\nu).
\]

Since the Gaussian spectrum, \( G(\nu) \), decays to zero faster than any polynomial in \( \nu \), the series defining \( d_a \) converges absolutely at each \( x \). Unlike the bandlimited analysis of the previous section, however, the presence of the polynomial terms in \( x \) in this expression means that the aliasing error is not uniformly bounded on \( \mathbb{R}^n \), but depends upon the magnitude of \( ||x|| \). However, on any compact subset of \( \mathbb{R}^n \), the magnitude of this aliasing can be uniformly bounded, and a variance and a mesh size can be chosen to reduce the aliasing to reduce (4.20) to a chosen tolerance.

### 4.6.2 Trigonometric Approximation

The generalized Fourier transform of a sine or cosine function is also composed of Dirac distributions:

\[
\int_{\mathbb{R}^n} \cos(2\pi \eta^T x) E_\nu(x) \, dx = \frac{1}{2} [\delta(\nu - \eta) + \delta(\nu + \eta)]
\]

\[
\int_{\mathbb{R}^n} \sin(2\pi \eta^T x) E_\nu(x) \, dx = \frac{1}{2j} [\delta(\nu - \eta) - \delta(\nu + \eta)]
\]

so that, following the development of the preceding section, and taking for example \( c(x) = \sigma^n \Delta^n \kappa \cos(2\pi j \eta^T x) \),

\[
\sum c(\Delta k) \exp(-\pi \sigma^2 ||x - k\Delta||^2) = \frac{1}{2} \kappa \sum \left[ G(\Delta^{-1} k + \eta) E_x(\Delta^{-1} k + \eta) + G(\Delta^{-1} k - \eta) E_x(\Delta^{-1} k - \eta) \right].
\]

Since the Gaussian is radially symmetric, \( G(\eta) = G(-\eta) \), choosing \( \kappa = G^{-1}(\eta) \) allows recovery of the unscaled cosine, plus the aliasing error. That is,

\[
\sum c(\Delta k) g(x - \Delta k) = \frac{1}{2} [E_x(\eta) + E_x(-\eta)] + d_a(x)
\]

\[
= \cos(2\pi j \eta^T x) + d_a(x)
\]
where here the aliasing error is given by

\[ d_a(x) = \frac{G^{-1}(\eta)}{2} \sum_{k \neq 0} \left[ G(\Delta^{-1}k + \eta)E_x(\Delta^{-1}k + \eta) + G(\Delta^{-1}k - \eta)E_x(\Delta^{-1}k - \eta) \right]. \]

An arbitrary trigonometric polynomial can thus be recovered in this fashion by sampling appropriately scaled versions of the constituent sinusoids.

The aliasing error here is easily seen to be uniformly bounded on \( \mathbb{R}^n \), since the above series which defines it can be bounded independently of \( x \). Note, however, that as in the bandlimited case, \( \Delta^{-1} \) should be chosen larger than twice the magnitude of the largest frequency in the trigonometric expansion of \( f \) lest the aliasing become non-negligible.

### 4.7 Choice of Gaussian Network Parameters

The Gaussian thus appears to sit on the boundary between the classical sampling theorem, with its requirements for strict low-pass filters, and multivariate spline and finite element polynomial reconstruction abilities, which require satisfaction of the corresponding Strang and Fix conditions. The Gaussian satisfies none of these requirements exactly, but in each case the effects of the discrepancy can be made arbitrarily small on compact subsets of \( \mathbb{R}^n \) by proper choice of the network design parameters. This section first summarizes the error analysis above, identifying each source of approximation error, then discusses how each can be independently reduced by the proper choice of network parameters to ensure a prespecified uniform approximation capability, \( \varepsilon_f \), everywhere on a given set \( A \).
4.7.1 Review of Error Sources

The above analysis reveals three classes of approximation errors introduced using the Gaussian; i.e. writing

\[ \mathcal{N}(x, c) - f(x) = d_f(x). \]

where \( \mathcal{N}(x, c) \) is an expansion in Gaussian radial basis functions the approximation error \( d_f \) can be decomposed as \( d_f = d_c + d_a + d_t \). The first of these, \( d_c \), is the class membership error; most functions encountered in practice will not exactly belong to one of the function classes examined above, and \( d_c \) quantifies the extent to which the actual function can be uniformly approximated on \( A \) by an element of one of the above function classes. The second error source, \( d_a \) is the aliasing error which arises from the fact that an infinite grid of radial Gaussians does not exactly satisfy the conditions required for exact reconstruction of either bandlimited or polynomial functions. Finally, the error \( d_t \) is the truncation error, which arises by truncating the infinite expansions considered above to a finite number of terms.

No choice of network design parameters can reduce the class membership error. The magnitude of this error amounts to a prior assumption which must be made when designing the network for a particular application, for example, the extent to which the required function can be uniformly approximated on the desired set by one of the function classes examined above. The error sources \( d_a \) and \( d_t \), however, can be bounded in terms of the Gaussian network design parameters and the assumed function class. The parameters of this class are also required prior assumptions and consist of: in the bandlimited case an upper bound on the bandwidth, \( \beta \), and \( F_{\text{max}} \) which provides an upper bound on the magnitude of \( F_F \) or, equivalently, on the integrated absolute value of \( f_F \). In the polynomial case, the required bounds are on the order of the polynomial to be approximated and on the maximum magnitudes of its coefficients, while in the trigonometric case, bounds on the magnitudes and frequencies of the sinusoids comprising the function are needed.

Given a target tolerance, \( \epsilon_f \), the bound on the class membership error is assumed
to be uniformly less than, e.g., $\epsilon_f / 3$. A mesh size $\Delta$ ($\leq 1/(2\beta)$) and a variance $\sigma^2$ can then be chosen so as to reduce the aliasing errors to this level. Finally, a truncation radius $\rho$, of the infinite Gaussian expansion, can be chosen to similarly reduce the truncation errors to this level. For convenience, the truncation radius will be expressed as a multiple of the mesh size, $\rho = l\Delta$.

The magnitude of each of the errors sources can, of course, be numerically evaluated directly from their definitions in terms of the assumed prior information for any given set of design parameters. Indeed this may be necessary for the more complicated sampling geometries suggested in Section 4.9. Some simplification of the inequalities, however, is possible in the case of a square sampling lattice, and the following sections explore this simplification to gain additional insight into the network design procedure.

### 4.7.2 Design for bandlimited functions

From the expression for the aliasing error (4.10) it is clear that the mesh size and variance are coupled, and it is useful to exploit the low-pass structure of the radial Gaussian to gain intuition into the selection of specific values for these parameters. The magnitude response of the filter, $G(\nu)$ is given by (4.15); written this way, $\sigma^2$ is proportional to the essential support of the filter in frequency, and to be an effective low-pass filter this support must contain $\mathcal{K}(\beta)$ and be contained in $\mathcal{K}_\Delta$. If $\beta$ is large, reflecting a function with very high assumed frequency content, $\sigma^2$ will have to be large as well, so that the profile of $G(\nu)$ is broad enough to cover the frequencies of interest. To minimize the aliasing errors, $\Delta$ must then be chosen so that $1/(2\Delta)$ is large also (refer again to Figure 4-2). On the other hand, if $\beta$ is small, the frequency range of interest can be covered using a small value of $\sigma^2$, allowing a smaller value for $1/(2\Delta)$.

This interpretation is intuitively pleasing. Noting the reciprocal role played by the parameter $\sigma^2$ in the right and left sides of (4.15), it follows that if $\beta$ is large, the
resulting network will consist of a densely packed array of Gaussians with very narrow profiles, which is a logical choice to approximate a function which can exhibit large localized variations. Conversely, if $\beta$ is small, the resulting network will consist of a loosely packed array of Gaussians with very broad profiles, which would be sufficient to capture a function which varies slowly.

If $\sigma^2$ is chosen to be small with respect to $\beta$, $G^{-1}$ in (4.5) will be very large near the boundary of the set $K(\beta)$, potentially creating a weighting function $c(x)$ with large magnitude and substantial high frequency behavior. To keep this weighting function in a sense close to $f$ itself (on $A$) requires that $G^{-1}$ be on the order of unity near the boundary of $K(\beta)$. Since the magnitude of $G^{-1}$ can be bounded on $K(\beta)$ by $\exp(n\pi\beta^2\sigma^2)$, this consideration suggests a choice of $\sigma^2 = n\pi\beta^2$. However, this dependence on the dimension $n$ assumes that the product $|F_F(\nu)G^{-1}(\nu)|$ can achieve its maximum arbitrarily near the corners of the $n$-cube $K(\beta)$. By choosing instead,

\begin{equation}
\sigma^2 = \pi \beta^2.
\end{equation}

and introducing the additional (rather mild) assumption that, with this choice of $\sigma^2$, the product $|F_F(\nu)G^{-1}(\nu)|$ achieves its maximum inside the ball of radius $\beta$ in $\mathbb{R}^n$, the desired constraints on the magnitude of the spectrum of $c(x)$ are assured.

The reciprocal relation between the variance and the mesh size then suggests that $\Delta$ be chosen such that

\begin{equation}
\Delta = \frac{1}{20\beta},
\end{equation}

where $\theta > 1$ is an "oversampling parameter", which controls the distance to the first copies of the repeating spectra.

Using the above low-pass filtering insights, the error bounds can be simplified considerably. In particular (see the Appendix for details), if $\Delta$ and $\sigma^2$ are chosen
according to (4.24) and (4.25) respectively, then

\begin{align*}
(4.26) \, |d_n(x)| &\leq \left[ e \, F_{\text{max}}(\sqrt{\pi} \beta)^n \right] \{1 - \text{erf}^n (2\theta - 1)\} \\
(4.27) \, |d_t(x)| &\leq \left[ e \, F_{\text{max}}(\sqrt{\pi} \beta)^n \right] \theta^{-n} \sum_{j \in \mathcal{J}} 2^{j+1} \binom{n}{j} \sum_{m=l+1}^{\infty} m^j \exp \left(-\left(\frac{\pi m}{2\theta}\right)^2\right)
\end{align*}

The index set \( \mathcal{J} \) is:

\[ \mathcal{J} = \{ 0 \leq j \leq n - 1 \mid n - j \text{ is odd} \} \]

and \( \text{erf}(\theta) \) is the error function (whose tabulated values can be found in many standard references). Since the Gaussians in (4.27) decay faster than any power of \( m \), the series converges, in fact, quite rapidly. Note that if the function being approximated has an integrable Fourier transform, a crude uniform bound for the class membership error is

\[ |d_c(x)| \leq \int_{\kappa(\beta)^c} |F_F(\nu)| d\nu \]

The above expressions for the aliasing and truncation error are quite convenient to compute, and show explicitly the rapid convergence of the expansions defining each error source; they are, however, quite conservative. At the expense of more complex computation, the bounds \( \epsilon_a \) and \( \epsilon_t \) can be sharpened to, respectively

\begin{align*}
|d_n(x)| &\leq F_{\text{max}} \sum_{k \neq 0} \int_{\kappa(\beta)+\Delta^{-1}k} G(\nu)G^{-1}(\nu - \Delta^{-1}k) d\nu \\
|d_t(x)| &\leq \left[ e \, F_{\text{max}}(\sqrt{\pi} \beta)^n \right] \theta^{-n} \sup_{x \in A_{\text{dist}(A,\Delta k)>l\Delta}} \sum \exp(-\pi \sigma^2 \|x - k\Delta\|^2),
\end{align*}

where \( \kappa(\beta)+\Delta^{-1}k \) is the cube of radius \( \beta \) centered at \( \Delta^{-1}k \). This aliasing bound is quite tight, equality being obtained at \( x = 0 \), for example, for a Gaussians approximation to the function whose spectrum is unity on \( \kappa(\beta) \).

### 4.7.3 Design for trigonometric polynomials

Suppose now that the function to be approximated can be represented on \( A \) with a uniform error no worse than \( \epsilon_c \) by an expansion of the form (using cosines without
loss of generality)

\[ f(x) = \sum_{i=1}^{m} \kappa_i \cos(2\pi \eta_i^T x). \]

The insights of the preceding section can also be used to select the parameters for a Gaussian approximation to this representation which reduce the aliasing and truncation errors to desired levels. In particular, by assuming the prior information \( F_{\text{max}} \geq \sum_i |\kappa_i|, \) and \( \beta \geq \max_i(\|\eta_i\|_\infty), \) the equations (4.24) and (4.25) can be used as above to select the Gaussian variance and the mesh size.

Using (4.23) and (4.12) conservative bounds for the aliasing and truncation error can then be written as

\[ |d_a(x)| \leq \frac{e^n F_{\text{max}}}{2} \sup_{\eta \in \mathcal{K}(\beta) \setminus \{0\}} \left[ G(\Delta^{-1} k - \eta) + G(\Delta^{-1} k + \eta) \right] \]

\[ |d_t(x)| \leq e^n F_{\text{max}} \left( \frac{\sqrt{\pi}}{2\theta} \right)^n \sup_{x \in \mathcal{A}_{\text{dist}(A, \Delta k) \setminus \Delta}} \sum \exp(-\pi \sigma^2 \|x - k\Delta\|^2), \]

where as above \( \theta \) is the oversampling ratio. For computational convenience, the Gaussian summation defining the truncation error may be replaced with the summation used in (4.27) above. Similarly, the exponent in the constant \( e^n \) assumes that one of the \( \eta_i \) can occur arbitrarily near the corners of \( \mathcal{K}(\beta) \); if instead these frequencies are known to be within the ball of radius \( \beta \) in frequency, the exponent \( n \) may be omitted.

### 4.7.4 Design for polynomials and constants

The situation is somewhat different with polynomials. Since their spectrum is supported only at the origin, the balance maintained above by choosing the variance to ensure the Gaussian remains larger than a certain magnitude at the edge of the frequency range of interest is no longer a factor. Thus, by varying \( \Delta \) and \( \sigma \) reciprocally, an infinite combination of variances and mesh sizes can yield exactly the same aliasing error.

The error expressions are much simpler for Gaussian approximation to constants, for example \( f(x) = \kappa \). Here, for any specific choice of mesh and variance, the errors
are uniformly bounded by

\begin{align}
(4.28) \quad |d_a(x)| & \leq |\kappa| \sum_{k \neq 0} G(\Delta^{-1} k) \\
(4.29) \quad |d_t(x)| & \leq |\kappa| (\sigma \Delta)^n \sup_{x \in A} \sum_{d \in A, d \Delta k > l \Delta} \exp(-\pi \sigma^2 \|x - k \Delta\|^2). 
\end{align}

As before, the multidimensional Gaussian summations can be reduced to a single summation, at the expense of some conservatism, using the counting argument in the Appendix.

For a general polynomial

\[ f(x) = \sum_{0 \leq m_i \leq M_i} \kappa_m x^m \]

expanding out the differentials in (4.20), the error bounds can be conservatively expressed as

\begin{align}
(4.30) |d_a(x)| & \leq \sup_{x \in A} \sum_{0 \leq m_i \leq M_i} |x^m| \sum_{k \neq 0} |p_m(\Delta^{-1} k)| G(\Delta^{-1} k) \\
(4.31) |d_t(x)| & \leq (\sigma \Delta)^n \sup_{x \in A} \sum_{0 \leq m_i \leq M_i} |\kappa'_m| \sum_{d \in A, d \Delta k > l \Delta} |(\Delta k)^m| \exp(-\pi \sigma^2 \|x - k \Delta\|^2). 
\end{align}

where the polynomials \( p_m \) are as defined in Section 4.6.1. Given bounds on the order of the polynomial and the magnitudes of its coefficients, bounds on the magnitude of the coefficients \( \kappa'_m \) can be computed from the matching conditions (4.18), and these in turn can be used to bound the above expressions for any choice of mesh, variance, and truncation. Recall that the Gaussians in both bounds decay faster than any polynomial in \( \nu \) or \( x \), and hence both series converge, in fact quite rapidly.

As mentioned at the end of Section 4.6.1, and unlike all the other cases considered above, it is clear from these expressions that for polynomial approximation the aliasing error is not uniform on \( \mathbb{R}^n \), but must be bounded as a function of \( \|x\| \). Note also that the coefficients in the Gaussian expansion are no longer uniformly bounded; hence, for a given \( l \), the magnitude of the coefficients in the closest omitted terms can grow rapidly as the mesh size increases. This places practical limits on the range of mesh sizes which can be used since, while the aliasing error can be kept to an arbitrary
level by appropriately selecting a variance, the truncation error for a fixed \( l \) will grow with the mesh size.

### 4.7.5 Choice of the output weights

When approximating bandlimited functions, the output-layer weights, \( c_k \), required to achieve the above uniform bounds could be found by explicitly evaluating the function

\[
(4.32) \quad c(\mathbf{x}) = \Delta^n\sigma^n\mathcal{F}^{-1}(F_{FG^{-1}}\chi_\beta)(\mathbf{x})
\]

at the lattice points \( \Delta k \). Similarly, for polynomial and trigonometric functions, the required output weights are the samples of the polynomials and sinusoids defined in Section 4.5 and Section 4.6 respectively. However, by assumption only the above function class information is assumed known about the function to be learned; no other prior information about its exact values or its spectrum are known. The actual weighting function \( c(\mathbf{x}) \) thus cannot be specified \emph{a priori}, and the goal of the learning algorithms examined in the chapters to follow will be to recursively adjust estimates, \( \hat{c}_k \), of the output weights to attempt to match their correct values \( c_k \).

### 4.8 Construction Examples

This section gives some examples of the construction procedures given in the preceding sections. These examples will also be used extensively in the identification and control applications which will be explored in later chapters.

#### 4.8.1 Bandlimited function: \( f_1(x) = \sin(2\pi x)/(\pi x) \)

This is the function used to generate the training data for the network considered at the end of Chapter 2, and has the Fourier transform \( F_1(\nu) = \chi_{[-1,1]}(\nu) \). To approximate functions bandlimited to \( \lambda'(1) \) in \( \mathbb{R} \), the analysis of Section 4.7.2 suggests
to $\theta = 1.25$, improves the uniform accuracy on $A$ to $|d_n(x)| \leq .12$, a factor of two improvement, without increasing the size of the network.

It is worth noting also at this point that there is a fundamental difference between the solution obtained by solving the interpolation problem at the lattice points, and that obtained using the Fourier analysis above. Both procedures give well defined solution for the output weights, since using Micchelli’s theorems [102] the interpolation matrix is nonsingular in this case. However, the interpolation method does not necessarily guarantee good uniform approximation, only that the network approximation will actually pass through the observed data points. To illustrate, Figure 4.8.1 shows the output weights obtained by solving the interpolation problem, i.e.

$$\mathcal{M}c_I = d$$

where $\mathcal{M}_{i,j} = \exp(-\pi \sigma^2 \Delta^2 (i - j)^2)$, and $d$ are the data points $d_i = f_i((i - 9)\Delta)$. Figure 4-6 compares the uniform approximation errors to $f_1$ using the output weight set $c_I$ instead of those used above. Here the worst case error is occurring at $x = \pm 0.66$, and is approximately 10% worse than that obtained using the weights computed by Fourier analysis.

Of course, the Fourier method uses more information to determine the output weights than the interpolation method. However, in the actual applications to follow, the output weights will be learned recursively based upon incoming observations. The only significant factor in this case is to obtain the tightest bound possible on the uniform approximation capability of the network.

But do these Gaussian networks really achieve better accuracy with fewer components than classical sampling theory would provide? To answer this question, Figure 4-7 shows the results of a 25 node approximation to the function $f_1(x - 0.33)$ using both sinc and Gaussian nodes a zero-centered grid of mesh $\Delta = 0.25$. The Gaussian approximation is approximately 60 times more accurate on the interval $[-2.2]$, although it degrades much more rapidly outside this set. Notice that the translation does not change the spectral support of the function, and was performed since oth-
the choices $\sigma^2 = \pi \beta^2 = \pi$ and $\Delta = (2\theta\beta)^{-1}$, so that the mesh used in the simulation of Section 3.3.1 corresponded to a rather small oversampling $\theta \approx 1.1$.

The magnitude of the expected aliasing error for a Gaussian network attempting to approximate this function can be predicted from (4.10) as

$$d_n(x) = \sum_{k \neq 0} \int_{\Delta^{-1}k - 1}^{\Delta^{-1}k + 1} G(\nu)G^{-1}(\nu - \Delta^{-1}k) d\nu$$

and with the chosen values for $\sigma^2$ and $\Delta$ this can be numerically evaluated to give the uniform bound $|d_n(x)| \leq 0.255$. Indeed, the values of the output weights which will achieve this approximation can be explicitly computed from (4.32) as

$$c(\Delta k) = \sigma \Delta \int_{-1}^{1} e^{\nu^2} E_{\Delta k}(\nu) d\nu$$

and the values corresponding to the nodes used in this network, $-8 \leq k \leq 8$, are listed in Figure 4.8.1.

If reconstruction is limited to the set $A = [-2, 2]$, then the network of Section 3.3.1 corresponds to a truncation parameter $l = 4$, and the truncation error,

$$d_l(x) = \sum_{|k| > 8} c(\Delta k)g(x - \Delta k)$$

is bounded on $A$ by a negligible fraction of the aliasing error. Figure 4-5 shows the actual approximation implemented by the network which confirms the above bound on the approximation error.

Notice that a small increase in the oversampling can bring a great improvement in the uniform accuracy. For example, reducing the mesh to $\Delta = 4/10$, corresponding
<table>
<thead>
<tr>
<th>$k$</th>
<th>$-8$</th>
<th>$-7$</th>
<th>$-6$</th>
<th>$-5$</th>
<th>$-4$</th>
<th>$-3$</th>
<th>$-2$</th>
<th>$-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(c_I)_k$</td>
<td>-0.0437</td>
<td>0.0925</td>
<td>-0.1443</td>
<td>0.1962</td>
<td>-0.2439</td>
<td>0.2799</td>
<td>-0.2705</td>
<td>-0.0014</td>
</tr>
<tr>
<td>$k$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$(c_I)_k$</td>
<td>2.0006</td>
<td>-0.0014</td>
<td>-0.2705</td>
<td>0.2799</td>
<td>-0.2439</td>
<td>0.1963</td>
<td>-0.1443</td>
<td>0.0925</td>
</tr>
</tbody>
</table>

Figure 4-4: Output weights determined by interpolation at the lattice points.

otherwise $f$ is exactly a multiple of one of the basis functions in a sampling theoretic expansion, which would result in identically zero approximation error on $\mathbb{R}^n$.

### 4.8.2 Polynomial function: $f_2(x) = ax(1-x)$

Using the polynomial matching condition (4.18), with $\kappa_2 = -a$, $\kappa_1 = a$, $\kappa_0 = 0$

\[
-a = \kappa'_2
\]

\[\implies \kappa'_2 = -a = a
\]

\[a = [\kappa'_1 + 2(2\pi j)^{-1}\kappa'_2 G''(0)]
\]

\[\implies \kappa'_1 = a
\]

\[0 = [\kappa'_0 + (2\pi j)^{-1}\kappa'_1 G''(0) + (2\pi j)^{-2}\kappa'_2 G'''(0)]
\]

\[\implies \kappa'_0 = a \left(\frac{1}{2\pi \sigma^2}\right).
\]

Thus the polynomial defining the correct values for the network output weights is

\[c(x) = a\Delta \sigma \left[ x - x^2 + \frac{1}{2\pi \sigma^2}\right].
\]

whose associated operator is

\[p_c(D) = \frac{a\Delta \sigma}{2\pi} \left(\frac{1}{2\pi}D^2 - jD + \frac{1}{\sigma^2}\right).
\]

Taking $a = 1$, and choosing the set $A = [0, 1]$, a mesh size of $\Delta = 1/4$, and a variance $\sigma^2 = 2\pi$ produces a bound for the aliasing error on $A$ of $|d_a(x)| \leq 0.0005$. Choosing
$l = 5$ again renders the truncation error negligible, and the resulting network consists of 15 nodes (the 5 nodes in $A$, plus 5 on either side of this set). Figure 4-8 shows the results, which again confirms the error bounds provided by the above analysis.

### 4.8.3 Polynomial term: $f_3(x) = f_2(f_2(x))$

Expanding out the iterated mapping

$$f_3(x) = -a^2 \left[ a x^4 - 2a x^3 + (1 + a) x^2 - x \right]$$

so that here $\kappa_0 = 0$, $\kappa_1 = a^2$, $\kappa_2 = -a^2(1 + a)$, $\kappa_3 = 2a^3$, and $\kappa_4 = -a^3$. Working backwards from $\kappa_4$,

$$-a^3 = \kappa_4 = \sum_{\alpha=4}^4 \kappa_\alpha' \binom{\alpha}{4} (2\pi j)^{\alpha-4} (D^{\alpha-4}G')(0)$$

so that $\kappa_4' = -a^3$. The constant $\kappa_3'$ can then be found from

$$\kappa_3 = \sum_{\alpha=3}^4 \kappa_\alpha' \binom{\alpha}{3} (2\pi j)^{3-\alpha} (D^{\alpha-3}G')(0) = \left[ \kappa_3' G'(0) + 4(2\pi j)^{-1} \kappa_4' G''(0) \right] \implies \kappa_3' = 2a^3;$$

similar computations yield

$$\kappa_2' = -a^2 \left[ 1 + a + \frac{3a}{2\pi^2} G'''(0) \right]$$

$$\kappa_1' = a^2 \left[ 1 + \frac{3a}{2\pi^2} G''(0) \right]$$

$$\kappa_0' = \frac{a^2}{4\pi^2} \left[ \frac{a}{4\pi^2} G^{(4)}(0) - \kappa_2' G''(0) \right].$$

Thus, using an array of Gaussians and the samples of the polynomial

$$c(x) = \Delta \sigma \sum_{r = 0}^4 \kappa_m' \kappa_m''$$

allows reconstruction of the polynomial $f_3(x)$ with an error bounded as a function of $x$ by

$$|d_a(x)| \leq \left| \sum_{m=0}^4 \sum_{k \neq 0} \kappa_m' (D^m G E_x)(\Delta^{-1}k) \right|,$$
which can be used with the formulae of the previous section to compute a conservative bound on the aliasing error.

In this example, reconstruction is limited to the set \( A = [0, 1] \), with \( a = 3.75 \). The Gaussian variance is set to \( \sigma^2 = 1.5\pi \) and the grid is taken to have mesh \( \Delta = 0.25 \). Expanding out the differentials defining the aliasing error, these choices result in the inequality \(|d_a(x)| \leq 0.0025\) on \( A \). Truncating the grid with \( l = 4 \), the truncation error is negligible in comparison to the aliasing, and Figure 4-9 shows the actual errors on \( A \) of the network approximation to \( f \).

4.8.4 Constant and Trigonometric function: \( f_4(x) = 2 + \cos(4\pi x) \)

Substituting directly into the equations with \( \eta = 2 \), by sampling the function

\[
c(x) = \Delta \sigma \left[ 2 + G^{-1}(2) \cos(4\pi x) \right]
\]

and using Gaussian interpolating functions with a variance \( \sigma^2 = 4\pi \) arranged on a sampling grid with a mesh size \( \Delta = 1/7 \), allows reconstruction of \( f \) with the aliasing error

\[
d_a(x) = 2 \sum_{k \neq 0} G(7k)E_x(7k) + \frac{G^{-1}(2)}{2} \sum_{k \neq 0} [G(7k + 2)E_x(7k + 2) + G(7k - 2)E_x(7k - 2)].
\]

which can be uniformly bounded by \(|d_a(x)| \leq 0.0055\). The set on which reconstruction is desired is \( A = [-.28, 1.28] \); taking again a truncation \( l = 5 \) renders the truncation error negligible, the network consists of 22 nodes. Figure 4-10 compares the network approximation with the actual function over the set \( A \).
CHAPTER 4. GAUSSIAN NETWORK ANALYSIS AND SYNTHESIS

4.8.5 Polynomial term in $\mathbb{R}^2$: $f_5(x_1, x_2) = ax_1 + bx_2 + c$

The polynomial which must be sampled to determine the correct output weights for the Gaussian expansion has the general form

$$c(x_1, x_2) = \sigma^2 \sum_{m_1=0}^{1} \sum_{m_2=0}^{1} \kappa'_{m_1 m_2} x_1^{m_1} x_2^{m_2}$$

where the polynomial coefficients must satisfy

$$\kappa_{m_1 m_2} = \sum_{m_1 \leq \alpha_1 \leq 3} \sum_{m_2 \leq \alpha_2 \leq 2} \kappa'_{m_1 m_2} (2\pi j)^{m_1 + m_2 - \alpha_1 - \alpha_2} \binom{\alpha_1}{m_1} \binom{\alpha_2}{m_2} G_1^{(\alpha_1 - m_1, \alpha_2 - m_2)}(0, 0).$$

For the chosen function $\kappa_{00} = c$, $\kappa_{01} = b$, $\kappa_{10} = a$, and $\kappa_{11} = 0$.

Solving these simultaneous equations, starting with $\kappa_{11}$ and working backwards, one finds the required values $\kappa_{11}' = 0$, $\kappa_{10}' = a$, $\kappa_{01}' = b$, and $\kappa_{00}' = c$. Note that in this case the polynomial $c$ is a scaled version of $f_5$.

Thus, using the samples of the polynomial

$$c(x_1, x_2) = \Delta^2 \sigma^2 (ax_1 + bx_2 + c)$$

as coefficients of a Gaussian expansion allows the desired recovery of $f_5$, modulo the small aliasing error. The operator $p_c(D)$ needed to evaluate the aliasing is then

$$p_c(D) = -\frac{\Delta^2 \sigma^2}{2\pi} [ajD^{1,0} + bjD^{0,1} - 2\pi c].$$

In this example, the set $A = [-3, 1.3] \times [-3, 3]$, and a mesh of $\Delta = 0.5$ is chosen for the centers. If, for example, $a = -3$, $b = 2$, and $c = 1$, then the choice of variance of $\sigma^2 = 0.4\pi$, with use of (4.20) produces a conservative bound for the aliasing error $|d_a(x_1, x_2)| \leq .002$ on $A$. Truncating the grid with $l = 4$, creating a network with 231 nodes, again renders the truncation error negligible, and Figure 4-11 shows a density plot giving a graphical picture of the error distribution.

4.9 Remarks

Using additional smoothness information: Any additional information known about the unknown function can be used in sharpening the above bounds or reducing the size
Figure 4-5: Error approximating $f_1$ on $A$ using the Gaussian network construction in Section 4.8.1. Note that, while the indicated accuracy is formally guaranteed only on $A = [-2, 2]$, the actual magnitude of $f_1$ beyond the edge of this set is on the order of the errors shown.
Figure 4-6: Error approximating $f_1$ on $A$ using the weights determined by solving the interpolation problem at the mesh points.
Figure 4-7: Comparison of a truncated sinc and a truncated Gaussian approximation to \( f_1(x - .33) \); the Gaussian expansion achieves much greater accuracy on the set \( \mathcal{A} \) using the same number of nodes.
Figure 4-8: Error approximating $f_2$ on $A$ using the Gaussian network construction in Section 4.3.2
Figure 4-9: Error approximating $f_3$ on $A$ using the Gaussian network construction in Section 4.8.3
Figure 4-10: Error approximating $f_4$ on $A$ using the Gaussian network construction in Section 4.8.4
Figure 4-11: Density plot of the error approximating $f_5$ on $A$ using the Gaussian network construction in Section 4.8.5. The darkest squares represent an error of approximately -.0013, while the lightest correspond to errors of approximately .0012. The shades in between correspond to errors which range between these two limiting values.
of the required representation. If information about the shape and orientation of the spectrum is known, the sampling lattice and mesh size can be adjusted accordingly. For example, if the spectrum is entirely contained in a ball in \( \mathbb{R}^n \), instead of a cube, the required sampling density (and hence the number of lattice points contained in any compact set) can be reduced by employing a hexagonal, as opposed to rectangular, sampling scheme [98, 121]. If \( f \) is known to have a transform whose support is much smaller in one direction in \( \mathbb{R}^n \), the variance in the filter response for that direction can be taken as proportionally smaller, and the mesh size for the corresponding direction in the sampling grid on \( \mathbb{R}^n \) can be taken proportionally larger. In this latter case, the Gaussian filter would be "stretched" to accommodate non-square supports of \( F_{BL} \) by scaling each frequency variable independently:

\[
G(\nu) = \exp \left[ -\pi \left( \frac{\nu_1^2}{\sigma_1^2} + \frac{\nu_2^2}{\sigma_2^2} + \ldots + \frac{\nu_n^2}{\sigma_n^2} \right) \right].
\]

Similarly, the filter response could be rotated by introducing scaled cross-couplings of the frequency variables, providing the more general representation,

\[
G(\nu) = \exp(-\pi \nu^T \Sigma \nu)
\]

for positive definite matrices \( \Sigma \). These more general quadratic arguments for the Gaussian have also been derived in the regularization analysis of [127], who reports promising experimental results using these additional degrees of freedom.

**Approximation breakdown outside of \( A \):** The approximation error outside of the set \( A \) represented by \( \alpha(\mathbf{x}) \) in (4.14) is of great importance. This error bound reflects the fact that the accuracy of the estimate implemented by the representations considered above may degrade drastically outside \( A \), especially using Gaussian interpolating functions which decay rapidly to zero as \( \mathbf{x} \) moves radially away from this set. Indeed, the examples in Section 4.8 graphically demonstrate this phenomenon. This possibly rapid degradation of the network approximation outside of the subset for which it was designed will significantly influence the adaptive algorithms which the following two chapters now examine in detail.
4.10 Appendix: Aliasing and truncation bounds

In this appendix are derived crude upper bounds for the components in the uniform approximation error to a bandlimited function $f$ on a compact set $A \subset \mathbb{R}^n$. Both the aliasing error $\epsilon_a$, arising from the departure of the radial Gaussian from the ideal low pass filter, and the truncation error $\epsilon_t$ are considered. The derivation assumes a uniform, square mesh, and that the set $A$ is square and aligned with the sampling lattice. These bounds are surely not the sharpest which can be computed, but, as conservative estimates, are sufficient for use in the robust adaptation mechanisms developed in Chapter 5 and Chapter 6.

Gaussian approximation to low-pass filter

Consider first the aliasing error. Starting from the definition (4.10), and using (4.25), a change of variables $\nu = \beta \eta$ produces:

$$\epsilon_a \leq \kappa_C \beta^n \int_{\mathbb{R}^n - A(2\theta - 1)} \exp(-||\eta||^2) \, d\eta$$

which can be written as:

$$\epsilon_a \leq \kappa_C \beta^n \left\{ \left( \int_{-\infty}^{\infty} \exp(-\zeta^2) \, d\zeta \right)^n - \left( \int_{-2\theta}^{2\theta - 1} \exp(-\zeta^2) \, d\zeta \right)^n \right\}$$

$$= \left[ \exp(1) F_{\max}(\sqrt{\pi} \beta)^n \right] \left( 1 - \text{erf}^n (2\theta - 1) \right),$$

where the last line uses (4.24) together with the assumption that the maximum of $F_F G^{-1}(\nu)$ occurs inside the ball of radius $\beta$ in frequency.

Truncation error

Consider a point $x \in A$ taken to coincide with one of the points in the sampling lattice, i.e. $x = j\Delta$ for some $j \in \mathbb{Z}^n$. Assume for simplicity that the truncation radius $\rho$ is taken as a multiple of the lattice mesh size, so that $\rho = l\Delta$ for some positive integer $l$. 
CHAPTER 4. GAUSSIAN NETWORK ANALYSIS AND SYNTHESIS

To begin the analysis, picture the sampling lattice as consisting of a series of nested hypercubes surrounding the point \( j \Delta \). Each point in the lattice thus lies on the boundary of exactly one of these nested hypercubes. Introducing the notation for \( m \geq 1 \)

\[
I_m = \{ k \in \mathbb{Z}^n \mid \|k \Delta - j \Delta\|_\infty = m \Delta \}
\]

to denote the indices of those nodes which lie on the boundary of the hypercube of radius \( m \Delta \) centered at \( j \Delta \), and let

\[
I_0 = \{ k \in \mathbb{Z}^n \mid \text{dist}(k \Delta, A) \leq l \Delta \}.
\]

These definitions allow the truncation error at \( x = j \Delta \) to be expressed as:

\[
|f_R(j \Delta) - f_A(j \Delta)| = \left| \sum_{k \in \mathbb{Z}^n} c_k \exp(-\pi \sigma^2 \|j \Delta - k \Delta\|^2) - \sum_{k \in I_0} c_k \exp(-\pi \sigma^2 \|j \Delta - k \Delta\|^2) \right|
\]

\[
= \left| \sum_{m=1}^{\infty} \sum_{k \in I_m} c_k \exp(-\pi \sigma^2 \|j \Delta - k \Delta\|^2) \right|
\]

where \( I'_m = I_m - I_0 \).

From its definition, \( C(\nu) \) is bounded with compact support, hence, using (4.32), the magnitude of each \( c_k \) is bounded by the constant:

\[
|c_k| \leq \sigma^n \ Vol(B) \sup_{\nu \in B} |C(\nu)| \leq \exp(1) F_{\max} \sigma^n \Delta^n (2\beta)^n \Delta \leq \kappa_T,
\]

where formula (4.24) has been used, together with the assumption that \( |F_F(\nu)G^{-1}(\nu)| \) achieves its maximum inside the ball of radius \( \beta \) in \( \mathbb{R}^n \).

For \( m \leq l \), since by definition of the truncation one has \( I'_m = \emptyset \), the inner summation vanishes identically. For \( m > l \), since the Gaussian is always positive, it is certainly true, although somewhat conservative, to write

\[
|f_R(j \Delta) - f_A(j \Delta)| \leq \kappa_T \sum_{m=l+1}^{\infty} \sum_{k \in I_m} \exp(-\pi \sigma^2 \|j \Delta - k \Delta\|^2).
\]

Part of the conservatism here ariees since, for \( l + 1 \leq m \leq m_0 \), \( I'_m \) is a strict subset of \( I_m \), where \( m_0 \) depends upon the diameter of \( A \) and the location within \( A \) of \( j \Delta \); hence
this inequality counts as omitted some of the nodes actually present in the network design.

Noting that each hypercube of radius \( m \Delta \) aligned with the sampling lattice includes \((2m + 1)^n\) lattice points, where \( n \) is the dimension \( x \), one obtains \( p(m) \triangleq \text{Card}(\mathcal{I}_m) = (2m + 1)^n - (2m - 1)^n \) is the number of terms in the inner summation for each \( m \). Since \( \min_{k \in \mathcal{I}_m} \| k \Delta - \hat{z} \Delta \| = m \Delta \), and the radial Gaussian is a monotone decreasing function of this norm, the error bound can be rewritten as

\[
|f_R(\hat{z} \Delta) - f_A(\hat{z} \Delta)| \leq \kappa_T \sum_{m = l+1}^{\infty} p(m) \exp(-\pi m^2 \sigma^2 \Delta^2).
\]

To evaluate \( p(m) \) explicitly, use the binomial theorem to compute

\[
p(m) = \sum_{j \in \mathcal{J}} 2^{j+1} \binom{n}{j} m^j,
\]

where the index set \( \mathcal{J} \) is

\[
\mathcal{J} = \{ 0 \leq j \leq n - 1 \mid n - j \text{ is odd} \}.
\]

Hence,

\[
|f_R(\hat{z} \Delta) - f_A(\hat{z} \Delta)| \leq \kappa_T \sum_{j \in \mathcal{J}} 2^{j+1} \binom{n}{j} \sum_{m = l+1}^{\infty} m^j \exp(-\pi m^2 \sigma^2 \Delta^2).
\]

and using the expressions (4.24), (4.25), produces

\[
|f_R(\hat{z} \Delta) - f_A(\hat{z} \Delta)| \leq \left[ e F_{\max} \left( \frac{\sqrt{\pi}}{2 \theta} \right) \right]^n (2\beta)^n \sum_{j \in \mathcal{J}} 2^{j+1} \binom{n}{j} \sum_{m = l+1}^{\infty} m^j \exp\left( -\left( \frac{\pi m}{2 \theta} \right)^2 \right)
\]

which gives (4.27) for any node point contained in the set \( A \). The symmetry of the construction and the monotonicity of the radial Gaussian ensures the total contribution of the omitted nodes becomes less as one moves away from nodes on the boundary toward the center of the set \( A \), thus the bound \( \epsilon_i \) can be expected to hold for all interior points of \( A \).
Chapter 5

Recursive Identification

One problem with the identification algorithm used in Section 3.3.1 is that the cost function the training attempts to minimize does not penalize the magnitude of the weights used. A gradient descent method is thus free to follow a potential minimizing solution along a direction in which the weights increase effectively without bound in order to try to squeeze the last bit of error out of the estimates to the values in a training set. A stability theoretic approach to the learning problem creates a positive definite cost function which contains all the time varying parameters of the identification or control problem, including the magnitudes of the adjustable network parameters. The adaptation mechanism is then designed analytically to ensure that this cost function is non-increasing at every instant in time.

In this chapter, the characterization of the approximating power of Gaussian networks given in the preceding chapter is developed into a complete specification of identification architectures exploiting these models, which are then proven stable and effective using Lyapunov stability theory. Following some initial observations reviewing classical recursive system identification techniques derived using stability methods, and the important unique aspects Gaussian network identifier structures introduce, Section 5.2 presents the design, and stability and convergence analysis for discrete time systems, followed by the continuous time analysis in Section 5.3. Section 5.4 then illustrates each of the algorithms with simulations.

As discussed in Section 3.3.2, however, even a stable and convergent identifier
CHAPTER 5. RECURSIVE IDENTIFICATION

will not necessarily produce high fidelity models of the plant dynamics unless the persistency of excitation conditions are satisfied. Section 5.5 formally defines these conditions and shows how the Gaussian network constructions of the previous section also permit precise characterization of the sequence of network inputs necessary to satisfy the persistency conditions. This analysis moreover demonstrates the inherent robustness which PE training lends to the learning process.

5.1 Stable Identification Methods

A stability analysis of a recursive identification structure begins by relating the adjustable parameters of the proposed identifier model to an error metric which describes the quality of the predictions made using the current parameter set. Before examining particular identifier structures in detail, the following section presents an overview of the kinds of relations which frequently arise in the analysis of identification methods. Section 5.1.2 then discusses the fact that, while appropriately designed neural networks are quite useful in extending these models, special precautions must be taken to retain the stability and convergence results of the classical models.

5.1.1 Analysis of Classical Identification Models

Denoting the error metric by $e(t)$ ($e[t]$ in discrete time), so that $e(t) = 0$ describes a perfect prediction of the process output at time $t$, one of the simplest possible relations between the adjustable outputs of the prediction mechanism and the actual process output is

$$e(t) = \hat{f}(t, \hat{c}(t)) - f(t) = \tilde{f}(t)$$

where $f(t)$ is an unknown function which governs the dynamic behavior of the process being observed, and $\hat{f}(t, \hat{c}(t))$ is the identifier's estimate of the correct value of this function at time $t$ using the parameters $\hat{c}(t)$. Assuming that $e(t)$ can be measured, the system identification problem can in this case be reduced to the problem of
recursively learning the function $f$. More often, however, the relation between the function estimation errors and the measured estimation error has a more complicated dynamic structure, such as

$$
\dot{\xi}(t) = K_D \xi(t) + b \dot{f}(t)
$$
$$
\epsilon(t) = \lambda^T \xi(t)
$$

where $K_D$ is a stable matrix and $b = [0 \ 0 \ \ldots \ 1]^T$. Here, direct measurements of the function estimation errors are not available, but must rather be inferred from the evolution of the observations $\epsilon(t)$.

In classical identification theory, it is assumed that $f$ lies in the linear span of a set of regressors, $Y_k(t)$, which are measured or computable from other measured signals, i.e.

$$
f(t) = \sum_{k=1}^{N} c_k Y_k(t).
$$

Provided that such a linear parameterization is known, the corresponding identifier employs the same structure, using estimates $\hat{c}_k(t)$ in place of the true parameter values $c_k$. When there are no dynamics between the function estimation errors and the measured error signal $\epsilon$, as above, the linear error equation is obtained

$$
\epsilon(t) = \sum_{k=1}^{N} \hat{c}_k(t) Y_k(t) - \sum_{k=1}^{N} c_k Y_k(t) = \hat{c}(t)^T Y(t),
$$

where $\hat{c}_k(t) = \hat{c}_k(t) - c_k$ is the mistuning between the estimate of the $k$th parameter at time $t$ and its true value $c_k$. This linear relation is extremely important in system identification theory, for assuming that the regressors are uniformly bounded, the recursive gradient descent algorithm

$$
\dot{\hat{c}}(t) = -k_a \epsilon(t) Y(t)
$$

for any $k_a > 0$ can then be shown to guarantee the convergence of the prediction errors to zero while maintaining the uniform boundedness of the estimates $\hat{c}(t)$ [111, 149]. Similar results hold in discrete time [56], as will be demonstrated below.
CHAPTER 5. RECURSIVE IDENTIFICATION

Even assuming a linear parameterization, however, the situation is much more complex when the measured errors have a dynamic relation to the function estimation errors. An important special case is when the error system satisfies the strictly positive real (SPR) condition, that is, when \( \lambda \) is such that the stable, rational transfer function \( \lambda^T (pI - K_D)^{-1} b \) is minimum phase (\( p \) here is used for the Laplace variable), has a frequency response whose phase shift at any frequency lies in the interval \([ -\pi, \pi ]\), and satisfies the asymptotic condition \( \lim_{\omega \to \infty} \omega^2 \text{Re}[\lambda^T (j\omega I - K_D)^{-1} b] > 0 \) [111, 149]. Intuitively, such a system has no worse than a first order lag between the input and the observed output, and is thus qualitatively similar to a first order differential equation. Indeed, it is often possible to choose the identifier structure such that

\[
\dot{e}(t) = -k_D e(t) + \dot{c}(t)^T Y(t),
\]

for some constant \( k_D > 0 \), which renders \( n - 1 \) of the error states unobservable but detectable, and which clearly satisfies the SPR conditions.

The importance of the SPR condition is that, in some sense, it "preserves" the information needed for gradient descent, so that use of the gradient adaptation law for the adjustable parameters also guarantees that \( e \) converges to zero, and that both the estimates \( \dot{c}(t) \) and all the states of the error system are uniformly bounded [111, 149].

5.1.2 Gaussian Networks for Identification

The above facts form the basis for most of the currently popular methods of system identification. The limiting assumption in these models is the knowledge of an appropriate set of regressors, \( Y \), which is usually highly problem specific. On the other hand, the results of the previous chapter have shown that a great variety of complicated functions can be reproduced using networks of very simple analog processing elements. That is, using the above Gaussian network designs, the outputs of the hidden layer nodes provide a set of regressors capable of closely approximating entire classes of functions. By using such networks to approximate the required functions,
one not only reduces the amount of prior information which must be encoded into the structure of the identifier, but also enjoys the advantages of the extremely efficient computational paradigm offered by these models.

As briefly discussed in the stability discussion of Chapter 3, the fact that a network can only approximate a particular function will produce slight changes to the above error models. Assuming that series-parallel identifier structures of Section 3.1.1 are used, so that the networks estimating of the required functions are non-recurrent, a linear error model would become

\[ \epsilon(t) = \mathcal{N}(x(t), \dot{c}(t)) - f(x(t)) \]
\[ = \frac{\partial \mathcal{N}}{\partial \dot{c}} \dot{c}(t) + d_f(x(t)) + d_g(x(t), \dot{c}(t)) \]

and similarly the above SPR model would be

\[ \dot{\epsilon}(t) = -k \epsilon(t) + \frac{\partial \mathcal{N}}{\partial \dot{c}} \dot{c}(t) + d_f(x(t)) + d_g(x(t), \dot{c}(t)). \]

where the partial derivatives of the network with respect to each parameter are evaluated using the current network inputs, \( x(t) \), and weights, \( \dot{c}(t) \).

By equating the partial network derivatives with the regressors \( Y \), these error equations are almost identical to the classical forms discussed above. The only difference is the presence of a perturbation \( d_f \), which describes the difference between the actual function \( f \) and the best uniform approximation to \( f \) on the set \( A \) possible given the chosen network architecture, and a perturbation \( d_g \) which contains the higher order terms in the weight mistuning for the Taylor expansion of the difference between the current network output and the best possible network approximation to \( f \) on \( A \). Despite the structural similarity, however, the presence of the perturbing terms fundamentally changes the stability properties of classical adaptive algorithms. Even if the magnitude of these terms can be uniformly bounded in time by a small constant, the classical algorithms can still under these conditions produce estimates \( \dot{c}_k(t) \) which grow without bound \([46, 137, 72]\). This is again the parameter drift phenomenon briefly encountered in Chapter 2.
To extend stability methods to identification and control algorithms which use neural network models, one possible approach is to try to develop stable adaptation algorithms which directly address the nonlinear effect of parameter mistuning represented by the term \(d_g\). While some preliminary work in this direction has begun [18], a complete solution which could be applied to the present application is still lacking. A second alternative is to treat the terms \(d_f\) and \(d_g\) in the expressions above as disturbances to existing linearly parameterized adaptation algorithms, whose stability properties have been exhaustively explored, and to call upon robust adaptation techniques to preserve stability in the face of these perturbations. This latter is the most promising approach at present, and will be extensively exploited in the techniques detailed below.

A number of robustness modifications have been developed for error equations perturbed in this manner [72, 111]. In order to implement them, however, and to be capable of predicting the ultimate capabilities of neural identification and control structures, it is necessary to quantify the magnitude of the disturbances in the error equations. Fortunately, the analysis and synthesis techniques for Gaussian networks detailed in the previous chapter provide just the tools required to evaluate these terms for a large class of functions.

The design procedure presented in the previous chapter exactly specifies the input weights, variances, and number of Gaussian nodes required to approximate with a chosen accuracy any function from a particular class, for example, any function banklimited to \([-\beta, \beta]^{n}\) whose transform is bounded in magnitude by \(F_{\text{max}}\). The assumed function class (measured by \(\beta\) and \(F_{\text{max}}\)) and the target accuracy (measured by \(\epsilon_f\) and \(A\)) thus determine a priori a correct set of input weights and variances; only the correct set of output weights must be learned to develop an accurate representation of a particular function in this class. Thus, by taking the network as

\[
N(x(t), \hat{c}(t)) = \sum_{\text{dist}(A, k\Delta) \leq \rho} \hat{c}_k(t) \exp(-\pi \sigma^2 \|x(t) - k\Delta\|^2),
\]

the prior analysis renders the network estimate linear in the adjustable output weights,
making the perturbing term $d_g$ identically zero. Provided the actual function underlying the observed data lies in the assumed function class, the magnitude of the term $d_f$ is then by design bounded by $\epsilon_f$ for every network input in the set $A$.

Compared with the network expansions discussed in Chapter 2, the input weights $kA$ are considered a fixed part of this architecture, and only the output weights are adjusted, so that $\hat{c} \in \mathbb{R}^p$ where here $p = \text{Card}(\{k \mid \text{dist}(A, kA) \leq \rho\})$. Chapter 7 will examine architectures in which the input weights also change with time.

With the magnitude of the disturbances in the resulting linearly parameterized expansion thus quantified, one of the several techniques of robust adaptation theory may be employed to modify the classical adaptation algorithm to prevent the kind of (possibly unbounded) parametric drift observed in Section 3.3.1. Not surprisingly, many of these modifications strongly resemble heuristics independently discovered by neural network researchers. For example, weight decay methods, which add a term to the adaptation law biasing the parameter estimates toward zero, thus preventing unbounded growth, are equivalent to the sigma modification technique in the robust adaptation literature [73]. Similarly, explicitly limiting parameter magnitudes, for example limiting $||\hat{c}|| \leq C_2$ or $|\hat{c}_k| \leq C_\infty$, is equivalent to a parameter projection robustness technique [46]; such limits are often implicitly imposed by hardware implementations of neural networks. Note that the Gaussian network construction given above also provides the necessary bounds on the magnitudes of the true parameters $||c||$ or $|c_k|$ needed to implement this latter modification.

A more common heuristic, demonstrated at the end of Section 3.3.1, involves stopping adaptation when the measured errors fall below a threshold determined by the bounds on the magnitude of the disturbance. Adaptation is thus stopped when the error signal is indistinguishable from the effects of the disturbance. This is equivalent to a deadzone robustness modification [122], and the remainder of this thesis will rely heavily on this technique, starting first with the stable identification algorithms developed in Section 5.1.
CHAPTER 5. RECURSIVE IDENTIFICATION

Each of these modifications will serve to halt the potentially unbounded growth of the adjustable parameters, but only the deadzone modification actually prevents parameter drift in response to the disturbance $d_f$; the other robustness techniques merely place (direct or indirect) bounds on how far the parameters can drift. Thus, adaptation algorithms which rely upon either parameter projection or sigma-modification to preserve stability can be subject to bursting behavior, that is, short periods of very bad predictions caused by the wandering of the parameters far from their true values [4]. For this reason, the analysis below will employ deadzone techniques where possible, although, assuming sufficient excitation, both parameter projection or switched sigma-modification techniques have the potential to perform better than deadzone techniques if the disturbance is much smaller than expected, or is absent altogether [111].

The design of a stable identification model using Gaussian networks thus requires a slightly different set of prior knowledge than classical designs. The functions driving the process dynamics should belong to one of the function classes considered in the previous chapter, and bounds must be known for the parameters of the function class to which the process dynamics are assumed to belong. Furthermore, prior bounds are required for the magnitude of control signals input to the process, and on the magnitude of the resulting process states and outputs. These bounds on the input-output behavior of the process define the boundaries of the set $A$ on which the network approximation will be required; together with the function class information, they completely specify the structure of a Gaussian network required in the identification model. Only the output weights must be recursively learned during the training procedure, and the following sections present the identifier structures and adaptation mechanisms which guarantee that this learning will be both stable and convergent.
CHAPTER 5. RECURSIVE IDENTIFICATION

5.2 Discrete Time Identification

This section focuses on the design of a recursive identification algorithm for single-input, single-output processes which can be modeled as

\[ y[t] = f(y[t - 1], \ldots, y[t - N], u[t], \ldots, u[t - M + 1]). \]

where \( f \) is a smooth, possibly nonlinear function. The outputs and control signals are assumed to be measured at each time step, and to be confined respectively to the sets \( A_y \) and \( A_u \) for all \( t \). The identification process begins at \( t = 0 \), without loss of generality, at which time a record of the last \( N \) outputs and \( M \) inputs is assumed to be available based upon past measurements of input-output activity.

Assuming that the function \( f \) belongs to one of the function classes examined in Chapter 4, based upon the assumed properties of the function class, treating each of the \( N + M \) arguments as independent variables, the variance, \( \sigma \), mesh size, \( \Delta \), and truncation radius, \( \rho \), of a network of Gaussian radial basis functions can be chosen so that, for the proper choice of output weights, \( c \), the network uniformly approximates \( f \) to a chosen degree of accuracy, \( \epsilon_f \), everywhere on the set

\[ A = \left( \prod_{i=1}^{N} A_y \right) \times \left( \prod_{i=1}^{M} A_u \right) \subset \mathbb{R}^{N+M}. \]

Defining as network inputs the collection of signals

\[ x[t] = [y[t - 1], \ldots, y[t - N], u[t], \ldots, u[t - M + 1]]^T; \]

a series-parallel identification structure

\[ \hat{y}[t] = \mathcal{N}(x[t], \hat{c}[t - 1]) \]

\[ = \sum_{\text{dist}(A_k, \Delta) \leq \rho} \hat{c}_k[t - 1] \exp(-\pi \sigma^2 \| x[t] - k \Delta \|^2). \]

can be used to predict outputs of the process for \( t \geq 0 \). The input weights of this network, which encode the sampling lattice, will remain fixed in this adaptation algorithm, while the output weights will be recursively adjusted to drive the prediction error, \( \epsilon[t] = \hat{y}[t] - y[t] \), toward zero.
5.2.1 Stability and Convergence Analysis

Expanding the prediction error in terms of the mistuning in the network output weights, \( \hat{c}_k[t - 1] = \hat{c}_k[t - 1] - c_k \), produces a linear error equation of the type discussed above

\[
\epsilon[t] = \sum_{\text{dist}(A, k\Delta) \leq \rho} \hat{c}_k[t - 1] \exp(-\pi \sigma^2 \norm{x[t] - k\Delta}^2) + d_f[t].
\]

Provided the function \( f \) in fact belongs to the assumed function class, the disturbance term \( d_f[t] = \mathcal{N}(x[t], c) - f(x[t]) \) satisfies \( |d_f[t]| \leq \epsilon_f \) as long as \( x[t] \in A \), and hence by assumption for all \( t \geq 0 \).

To learn the specific function from the observed input-output data, the output weights of this network should be adjusted according to the modified gradient adaptation law,

\[
(5.1) \quad \hat{c}_k[t] = \hat{c}_k[t - 1] - k_a \epsilon_{\Delta}[t] \exp(-\pi \sigma^2 \norm{x[t] - k\Delta}^2).
\]

where the adaptation signal \( \epsilon_{\Delta}[t] \) continuously incorporates a deadzone of size \( \Phi \) into the prediction error \( \epsilon[t] = \hat{y}[t] - y[t] \), i.e.

\[
\epsilon_{\Delta}[t] = \epsilon[t] - \Phi \text{sat}(\epsilon[t]/\Phi),
\]

and \( \text{sat} \) is the saturation function ( \( \text{sat}(y) = y \) if \( |y| < 1 \), and \( \text{sat}(y) = \text{sign}(y) \) otherwise). If the adaptation gain is chosen to satisfy

\[
(5.2) \quad 0 < \delta = 2 - k_a \sup_{x \in A} \sum_{\text{dist}(A, k\Delta) \leq \rho} \exp(-2\pi \sigma^2 \norm{x - k\Delta}^2),
\]

and the deadzone is chosen so that \( \Phi \geq \epsilon_f \), then each of the parameter estimates \( \hat{c}_k \) will remain bounded, and further, the prediction error will converge asymptotically to \( |\epsilon[t]| \leq \Phi \) as \( t \to \infty \).

As proof, consider the Lyapunov function candidate:

\[
V[t] = \sum_{\text{dist}(A, k\Delta) \leq \rho} (\hat{c}_k[t])^2
\]
so that, using the robust adaptation law (5.1), the difference $V[t] - V[t - 1]$ can be expanded as

$$
V[t] - V[t - 1] = k_a^2 \epsilon_\Delta[t]^2 \sum_{\text{dist}(A,k\Delta) \leq \rho} \exp(-\pi \sigma^2 \lVert x[t] - k\Delta \rVert^2)^2 \\
- 2k_a \epsilon_\Delta[t] \sum_{\text{dist}(A,k\Delta) \leq \rho} c_k[t - 1] \exp(-\pi \sigma^2 \lVert x[t] - k\Delta \rVert^2) .
$$

But since,

$$
\sum_{\text{dist}(A,k\Delta) \leq \rho} c_k[t - 1] \exp(-\pi \sigma^2 \lVert x[t] - k\Delta \rVert^2) = \epsilon_\Delta[t] + \Phi \text{ sat}(\frac{\epsilon[t]}{\Phi}) - d_f[t] ,
$$

substituting, re-arranging, and introducing obvious bounds reveals

$$
V[t] - V[t - 1] \leq -k_a \epsilon_\Delta[t]^2 (2 - k_a \epsilon_\Delta[t]) \sum_{\text{dist}(A,k\Delta) \leq \rho} \exp(-\pi \sigma^2 \lVert x[t] - k\Delta \rVert^2)^2 - 2k_a \epsilon_\Delta[t] (\Phi - \epsilon_f)
$$

(5.3) \leq -k_a \delta \epsilon_\Delta[t]^2 .

By the discrete time Lyapunov theorem [93], one can thus conclude that $V[t]$ and hence each of the $\hat{c}_k[t]$ are all uniformly bounded. Moreover, by summing both sides of this inequality

$$
\sum_{\tau=0}^{t} \epsilon_\Delta[\tau]^2 \leq \frac{1}{k_a \delta} [V[0] - V[t] ,
$$

and noting that the inequality (5.3) implies that $V[0] \geq V[t] \geq 0$ for any $t \geq 0$, one concludes that the signal $\epsilon_\Delta$ is in $l^2$, and thus tends to zero as $t \to \infty$. Finally, for any $\epsilon > 0$, $|\epsilon_\Delta[t]| < \epsilon$ implies that $|\epsilon| \leq \Phi + \epsilon$ and thus as $t \to \infty$ the inequality $|\epsilon[t]| \leq \Phi$ is obtained asymptotically.

### 5.2.2 Remarks and Comments

These results exactly quantify the relations between the approximation capabilities of the network and the adaptation parameters required for stable, convergent operation. The allowable adaptation gains are restricted by (5.2), while the required deadzone is determined by the bound $\epsilon_f$. 
CHAPTER 5. RECURSIVE IDENTIFICATION

The above stability proof shows that the parameter estimates remain bounded during adaptation, and that the prediction errors asymptotically converge to a neighborhood of zero whose size depends continuously on the uniform approximation capability of the network. Nothing can be concluded yet, however, about the convergence of the output weights to their correct values, nor about the rate at which the prediction error converges. Section 5.5 will discuss the conditions necessary to make these sharper conclusions.

Note that only an upper bound on the the number of inputs $N$ and $M$ required to characterize the process are necessary. Indeed, one can consider the functional dependence on any extraneous inputs to be an approximation to the constant unity on the set $A$. As long as the mesh chosen for the centers is sufficient to capture this function to the specified tolerance, the stability and asymptotic convergence properties will be exactly as detailed above.

One might be tempted to turn a series-parallel network trained to predict one step ahead into a parallel network in order to make multistep predictions, that is, feeding back the output of a trained predictor network to its inputs and iterating. In general, it is quite difficult to guarantee stability and convergence for this technique; the unavoidable, small one-step approximation errors can become magnified, resulting in significant divergence from the actual output of the process after only a few steps, as will be illustrated in Section 5.4.3. However, note that if, for example, the dynamics are autonomous

\[
y[t + 1] = f(f(y[t - 1], \ldots, y[t - N]), y[t - 1], \ldots, y[t - N + 1])
\]

\[
\triangleq f^{[1]}(y[t - 1], \ldots, y[t - N])
\]

\[
y[t + 2] = f(f(y[t - 1], \ldots, y[t - N]), f(y[t - 1], \ldots, y[t - N]), y[t - 1], \ldots, y[t - N + 2])
\]

\[
\triangleq f^{[2]}(y[t - 1], \ldots, y[t - N])
\]

and so on, so that it is possible to guarantee convergence when predicting an arbitrary (but fixed) number of steps into the future by using the same techniques described above directly on the iterated mapping $f^{[n]}$. Similar ideas apply for nonautonomous
CHAPTER 5. RECURSIVE IDENTIFICATION

processes, although the number of network inputs will have to increase to accommodate the larger number of forcing inputs which must be included in the iterated mapping. Note that \( f^{[n]} \) will usually be less smooth than the original mapping, and hence the corresponding network will generally require more nodes to maintain the same level of uniform approximation.

5.3 Continuous Time

This section designs recursive identifiers for continuous time systems which can be modeled by the nonlinear differential equations

\[
\dot{z}(t) = f(z(t), u(t)), \quad y(t) = h(z(t), u(t)),
\]

where \( u(t) \in \mathbb{R} \) is the control input. The state vector \( z(t) \in \mathbb{R}^n \), the process output \( y(t) \in \mathbb{R} \), and the input \( u(t) \) are assumed continually measured. Each component of the vector field \( f(z, u) \) is assumed to be sufficiently smooth that unique solutions to (5.4) exist, and further it is assumed that the magnitude of the state vector and the output are bounded in response to bounded inputs. It is also assumed that prior bounds are known for both the maximum magnitudes of the control input, so that \( u(t) \in A_u \forall t \), and the resulting maximum values of the state variables given the initial condition \( z(0) \), e.g. \( z(t) \in A_z, \forall t \geq 0 \). Finally, \( u(t) \) is assumed to be a differentiable function of time, with \( u(t) \) uniformly bounded, and the output mapping \( h \) is assumed to be at least continuously differentiable.

Again assuming the functions \( f_j \) and \( h \) belong to classes which can be approximated by Gaussian networks, and that information about the respective function classes is available to allow the proper choice of network construction parameters, a series-parallel identifier structure can be used

\[
\dot{\hat{z}}_j(t) = -k_D \dot{z}_j(t) + \sum_{\text{dist}(A,k\Delta_j) \leq \rho_f} \hat{c}_{j,k}^f(t) \exp(-\pi\sigma_f^2 \|x(t) - k\Delta_j\|^2)
\]

\[
\hat{y}(t) = \sum_{\text{dist}(A,k\Delta_h) \leq \rho_h} \hat{c}_{k}^h(t) \exp(-\pi\sigma_h^2 \|x(t) - k\Delta_h\|^2),
\]
where \( j = 1 \ldots n \), \( e_j(t) = \hat{z}_j - z_j(t) \), and recall that here \( x(t)^T = [z^T(t) \ u(t)] \in \mathbb{R}^{n+1} \).

Note that this identification structure effectively attempts to identify each of the nonlinear functions \( f_j \) and \( h \) simultaneously; \( n + 1 \) different networks can be used for this task, allowing different choices for the network design parameters, or, if \( f \) and \( h \) belong to similar function classes, the same network with \( n + 1 \) outputs can be used. The structure above uses a single network with \( n \) outputs to estimate each component of the dynamics \( f \), and a different network to estimate the output mapping \( h \).

For each network, the design parameters, \( \sigma, \Delta \), and \( \rho \) are fixed by the function classes to which each \( f_j \) and \( h \) are assumed to belong. They are chosen using the techniques of Chapter 4 so that, for proper sets of network output weights, \( c^f_j \) and \( c^h \), the worst case approximation errors to each component, \( \epsilon_f = \sup_j \epsilon_{f,j} \), and \( \epsilon_h \) are as small as necessary on the set \( A = A_z \times A_u \) to achieve a desired asymptotic estimation accuracy. The task now is to determine an adaptation algorithm for the output weights which drives the output prediction errors toward zero, while ensuring that the identifier states and network output weights remain bounded.

From (5.4) and (5.5) each component of the state estimation error, \( e_j(t) \), can be expanded in terms of the output weight mistuning, \( \hat{c}_{j,k}^f [t - 1] = \hat{c}_{j,k}^f [t - 1] - c_{j,k}^f \) to produce the perturbed SPR error equations

\[
(5.6) \quad \hat{e}_j(t) = -k_P e_j(t) + \sum_{\text{dist}(A,k\Delta_f) \leq \rho_f} \hat{c}_{j,k}^f(t) \exp(-\pi \sigma_f^2 \|x(t) - k\Delta_f\|^2) + d_{f,j}(t). \]

The disturbance terms \( d_{f,j}(t) \) are here given by

\[
d_{f,j}(t) = \sum_{\text{dist}(A,k\Delta_f) \leq \rho_f} c_{j,k}^f \exp(-\pi \sigma_f^2 \|x(t) - k\Delta_f\|^2) - f(x[t]).
\]

Provided the functions \( f_j \) in fact belong to the function classes assumed when designing the network, the prior analysis and above assumptions about the process input-output behavior then guarantees that the disturbances satisfy \( |d_{f,j}(t)| \leq \epsilon_{f,j} \) for all \( t \).

The output prediction error, \( \hat{y}(t) = \hat{y}(t) - y(t) \), can similarly be expanded to
produce the perturbed linear error equation

\[ \dot{y}(t) = \sum_{\text{dist}(A, k\Delta_h) \leq \rho_h} c_k^h(t) \exp(-\pi \sigma_h^2 \|x(t) - k\Delta_h\|^2) + d_h(t) \]

where here

\[ d_h(t) = \sum_{\text{dist}(A, k\Delta_h) \leq \rho_h} c_k^h \exp(-\pi \sigma_h^2 \|x(t) - k\Delta_h\|^2) - h(x[t]). \]

Once again, assuming that \( h \) actually belongs to the assumed function class, the inequality \(|d_h(t)| \leq \epsilon_h\) holds for all \( t \).

To incorporate the necessary robustness measures into the adaptation mechanism, a continuous deadzone signal

\[ e_\Delta(t) = e(t) - \Phi_f \sigma(e(t)). \]

can be constructed using the vector saturation function, \( \sigma(e) \),

\[ (\sigma(e))_i = \text{sat}(e_i/\Phi_f), \]

where the scalar, \( \Phi_f \), describes the size of the deadzone, to be selected below. A deadzone can similarly be incorporated into the output estimation error \( s(t) = \hat{y}(t) - y(t) \), resulting in a signal \( s_\Delta(t) \) defined exactly as the \( e_\Delta(t) \), using instead a deadzone of width \( \Phi_h \):

\[ s_\Delta(t) = s(t) - \Phi_h \text{sat}(s(t)/\Phi_h). \]

To stably train these networks, the modified gradient adaptation laws

\[
\begin{align*}
\dot{c}_{jk}^f(t) &= -k_1 e_\Delta_j(t) \exp(-\pi \sigma_f^2 \|x(t) - k\Delta_f\|^2) \\
(5.7) \quad \dot{c}_k^h(t) &= -k_2 s_\Delta(t) \exp(-\pi \sigma_h^2 \|x(t) - k\Delta_h\|^2),
\end{align*}
\]

are used. If the deadzone \( \Phi_f \) is then chosen such that \( k_D \Phi_f \geq \epsilon_f \), and if \( \Phi_h \) is chosen to satisfy \( \Phi_h \geq \epsilon_h \) then the states of the identifier and the output weights of the network remain bounded, and further the state and output estimation errors asymptotically converge to their respective deadzones; in particular \( \lim_{t \to \infty} |\hat{y}(t) - y(t)| \leq \Phi_h \).
CHAPTER 5. RECURSIVE IDENTIFICATION

To prove this, consider the Lyapunov function candidate

\begin{equation}
V = \frac{1}{2} e_\Delta^T e_\Delta + \frac{1}{2k_1} \sum_{j=1}^{n} \sum_{\text{dist}(A,k\Delta_f) \leq \rho_f} (\hat{c}_{j,k}^f)^2 + \frac{1}{2k_2} \sum_{\text{dist}(A,k\Delta_h) \leq \rho_h} (\hat{c}_{k}^h)^2,
\end{equation}

which has time derivatives along trajectories of the adaptive system given by (5.6) and (5.7):

\begin{equation}
\dot{V} = e_\Delta^T \left( I - \Phi \frac{\partial \sigma(e)}{\partial e} \right) \dot{e} + \frac{1}{k_1} \sum_{j=1}^{n} \sum_{k} \hat{c}_{j,k}^f \hat{c}_{j,k}^f + \frac{1}{k_2} \sum_{k} \hat{c}_{k}^h \hat{c}_{k}^h
\end{equation}

\begin{equation}
= -k_D e_\Delta^T e_\Delta - s_\Delta^2 + e_\Delta^T d_f + s_\Delta d_h - k_D \dot{\Phi}_f e_\Delta^T \sigma(e) - \dot{\Phi}_h s_\Delta \text{sat}(s/\Phi_h)
\end{equation}

\begin{equation}
\leq -k_D e_\Delta^T e_\Delta - s_\Delta^2 - \|e_\Delta\|_1(k_D \Phi - \|d_f\|_{\infty}) - |s_\Delta|(\Phi_h - |d_h|)
\end{equation}

\begin{equation}
\leq -k_D e_\Delta^T e_\Delta - s_\Delta^2
\end{equation}

whenever \( x \in A \).

Thus, if \( e_\Delta \) and all the output weight mistunings are bounded at time \( t = 0 \), they remain bounded for all \( t \geq 0 \). Since the tuned values of the output weights are finite, each output weight estimate also remains bounded. To complete the proof and establish asymptotic convergence of the estimation errors, it is necessary to show that \( s_\Delta \to 0 \) and \( e_\Delta \to 0 \) as \( t \to \infty \). This can be accomplished by applying Barbalat’s lemma [159] to the continuous, non-negative function,

\begin{equation}
V_1(t) = V(t) - \int_0^t (\dot{V}(\tau) + k_D e_\Delta^T(\tau)e_\Delta(\tau) + s_\Delta^2(\tau)) \, d\tau
\end{equation}

with \( \dot{V}_1(t) = -k_D e_\Delta^T(t)e_\Delta(t) - s_\Delta^2(t) \).

Note that use of (5.8) has already established that all signals in the system are bounded, the function \( V_1 \) is used only to conclude asymptotic convergence of the estimation errors. By Barbalat’s lemma, since \( V_1(t) \) is monotonically non-increasing and bounded below, if \( \dot{V}_1(t) \) is bounded, then \( \dot{V}_1(t) \to 0 \), and hence so also do \( e_{\Delta_1}(t) \) and \( s_\Delta(t) \).

To obtain the desired convergence result, it is thus necessary to demonstrate that \( e_\Delta^T(t)e_\Delta(t) \) and \( s_\Delta^2(t) \) have bounded rates of change. The definition of these terms
is such that \( \dot{s}_\Delta(t) \) is not well defined when \(|s(t)| = \Phi_h \), and similarly for \( \dot{e}_\Delta(t) \) when \(|e_{\Delta_j}(t)| = \Phi_f \). However, the quantities \((d/dt)s_{\Delta}^2(t)\) and \((d/dt)e_{\Delta}^T(t)e_{\Delta}(t)\) are both well defined and continuous everywhere and can be written as \((d/dt)s_{\Delta}^2(t) = 2s_{\Delta}(t)\dot{s}(t)\), and similarly for \((d/dt)e_{\Delta}^T(t)e_{\Delta}(t)\), so that the required rates of change vanish identically at those points where the derivatives of the deadzone signals themselves fail to exist.

For \( \dot{e}_j(t) \) to be bounded, each term on the right side of (5.6) must be bounded. By the assumed continuity of the functions being approximated, each \( f_j(x) \) is finite for finite \( x \). Since, by assumption \( x(t) \) remains in the compact set \( A \), each term \( f_j(x(t)) \) is thus bounded for all \( t \). Since the output weights remain bounded during operation, and since the output of each Gaussian node is bounded for any network input \( x \), the outputs of the network remain bounded. Thus, since the preceding Lyapunov proof has shown the identifier states remain bounded, each term on the right side of (5.6) is bounded, so that \( \dot{e}_j(t) \) is bounded for any \( t \).

Demonstrating that \( \dot{s}(t) \) is bounded is slightly more involved. Note that

\[
\dot{s}(t) = \sum_{\text{dist}(A,k\Delta_h) \leq \rho_h} \dot{c}_k^h(t) \exp(-\pi \sigma_h^2 \|x(t) - k\Delta_h\|^2)
- \dot{x}^T(t) \left( \frac{\partial}{\partial x} h(x(t)) - \sum_{\text{dist}(A,k\Delta_h) \leq \rho_h} \dot{c}_k^h(t) \frac{\partial}{\partial x} \exp(-\pi \sigma_h^2 \|x(t) - k\Delta_h\|^2) \right)
\]

where \( \dot{x}^T(t) = [f^T(x(t)), \dot{u}(t)] \). The above arguments have shown that each component of \( f \) is bounded, and by assumption \( \dot{u}(t) \) is bounded so that \( \dot{x} \) is bounded. From the Lyapunov proof \( s_\Delta(t) \) is bounded, and the Gaussian is certainly bounded, so \( \dot{c}_k^h(t) \) is bounded for any \( t \). Finally, the indicated partial derivatives of the Gaussian are uniformly bounded, and the continuous differentiability of \( h \) implies that its partial derivatives are bounded on any compact subset of \( \mathbb{R}^{n+1} \); so that the term in brackets is bounded for any \( t \). Together, these facts show that \( \dot{s}(t) \) is bounded for any \( t \).

The terms \((d/dt)e_{\Delta}^T(t)e_{\Delta}(t)\) and \((d/dt)s_{\Delta}^2(t)\) are thus bounded, and hence applying Barbalat's lemma to the function \( V_1(t) \) one concludes that \( s_\Delta(t) \) and each
\( e_{\Delta_j}(t) \) approach zero as \( t \to \infty \), and thus asymptotically \( \lim_{t \to \infty} |e_j(t)| \leq \Phi_f \) and \( \lim_{t \to \infty} |\sigma(t)| \leq \Phi_h \).

Once again, this analysis serves to quantify the network adaptation parameters required for stable, convergent operation. Unlike the discrete time case, there are no limits on the allowable adaptation gains (which can, in fact, be different for each output weight, as will be discussed in more detail in Chapter 7), while the deadzones required for the estimates of the functions \( f_j \) are actually smaller than those required in the previous section, since the filtering provided by the structure of the identifier helps to attenuate the effects of the network approximation errors. Finally, as before, the asymptotic accuracy of the identifier is limited only by the uniform accuracy of the network on the set \( A \).

### 5.4 Examples

This section examines some applications of the above ideas for tasks in time series prediction, supervised learning, and continuous time identification. In each case, the smoothness properties of the function to be learned and the range of inputs over which the network approximation will be required determine the network topology and design parameters, as discussed in Chapter 4, but the output weights of the networks used are initialized to zero, reflecting a complete lack of knowledge about the specific functions appearing. The stable adaptation algorithms derived above are then used to recursively adjust the output weights as discussed above so as to achieve the desired level of performance from the network.

#### 5.4.1 Supervised learning

The kind of supervised learning algorithm discussed in Section 3.3.1 is precisely an example of a problem in time series prediction, where the nonlinear process is given by \( y[t] = \frac{\sin(2\pi u[t])}{\pi u[t]} \) and \( u[t] \) cycles through the points indicated in Figure 3.3.1.
The set $A = [-2, 2]$ here, and from the construction example in Section 4.8.1 the network topology used in the example is capable, for the correct values of the 17 output weights, of approximating the function $f(u) = \sin(2\pi u)/(\pi u)$ with a uniform accuracy of at least .255 on $A$. Thus taking a deadzone of size $\Phi = 0.26$, as was done heuristically at the end of Section 3.3.1, the stability of the output weight adaptation algorithm and the convergence of the prediction error to the deadzone is guaranteed, as indeed Figure 3.5 confirms.

Note, however, from Figure 5-1 that, while the total squared error between the estimated output weights and those computed in Section 4.8.1 monotonically decreases, as predicted by the Lyapunov proof, it does not necessarily decrease to zero. This in particular means that, if learning is halted after convergence on the training set, the resulting network may not produce accurate predictions of the values of $f$ at points not in the training set: that is, the network may not generalize well. As Section 3.3.2 has noted, generalization, output weight convergence, and the concept of persistency of excitation in the adaptation mechanism are all related concepts, and Section 5.5 will explore these issues for the Gaussian identifiers proposed above.

This same figure illustrates once again the instability mechanism discussed in Section 3.3.1: without the deadzone modification the total squared error in the output weights initially decreases, then begins to climb rapidly.

### 5.4.2 Time series prediction

Consider the problem of recursively predicting the output of the chaotic time series [95],

$$y[t] = f(y[t - 1]) = 3.75y[t - 1](1 - y[t - 1]).$$

Choosing an identifier structure as a Gaussian network with the scalar input $x[t] = y[t - 1]$, the predicted process output at time $t$ can be written as

$$\hat{y}[t] = \mathcal{N}(y[t - 1], \hat{c}[t - 1]) = \sum_{d \in \langle A, k\Delta \rangle \leq \rho} \hat{c}_k[t - 1] \exp(-\pi \sigma^2(y[t - 1] - k\Delta)^2),$$
where the network design parameters must be chosen based upon prior assumptions about the process.

For this process, provided that \( y[0] \in [0,1] \), then also \( y[t] \in [0,1] \) for all \( t \) and hence, the set on which reconstruction of \( f \) is required is \( A = [0,1] \). From the construction example of Section 4.8.2, the choices \( \Delta = 1/4 \), \( \sigma^2 = 2\pi \), and \( \rho = 5\Delta \) will produce \( \sup_{x \in A} |f(x) - N(x,c)| \leq .002 \) for the choices of coefficients \( c_k \) computed for that example. Accordingly, the network is configured with this choice of parameters \((\sigma^2, \Delta, \rho)\), and the 15 required output weights are recursively estimated based upon observations of the process output.

To ensure stable recursive adaptation of the weights the deadzone size, \( \Phi \), is taken conservatively as .005. A quick calculation shows that, with the given mesh size, the adaptation gain \( k_a = 0.7 \) easily satisfies the above constraints on the adaptation gains. These two parameters are used in the output weight adaptation law (5.1).

Figure 5-2 shows a typical time series for this system, starting from an initial condition \( y[0] = 0.25 \). Using the adaptation laws of Section 5.2 together with the above identifier structure and adaptation parameters, the behavior of prediction error, \( e[t] \), using the adaptive Gaussian predictor is shown in Figure 5-3. The process and identifier dynamics were simulated in C on a NextStation. As expected, the prediction error eventually converges to the small deadzone. The monotonically decreasing mistuning in the output weights, predicted by the above Lyapunov proof, is demonstrated in Figure 5-4.

### 5.4.3 Multistep Prediction

Consider now the goal of predicting this same chaotic time series multiple time steps into the future. If the method in [114] were followed, after training the one-step, series-parallel predictor to the desired accuracy, it would be used to predict subsequent process outputs by operating in parallel mode. As the discussion in Section 5.2 indicated, however, this may not produce satisfactory operation since the small pre-
prediction errors observed in Figure 5-3 will become amplified in subsequent steps if they are fed back to the inputs of the Gaussian network. This is illustrated in Figure 5-5, which shows the degradation of the subsequent predictions obtained using a trained one-step Gaussian predictor in a parallel fashion.

On the other hand, stable convergent multistep prediction for a fixed number of steps into the future can be obtained by using a series-parallel identifier directly on the iterated mapping. To predict two steps into the future, for example, a Gaussian network predictor uses the single input, \( y[t - 1] \), and attempts to predict the output \( y[t + 1] \). Given the above assumptions about the process, \( y[t + 1] = f(f(y[t - 1])) \), and the right hand side of this equation is precisely the polynomial \( f_3 \) examined in the construction of Section 4.8.3 above. Thus, a Gaussian network predictor with exactly the same structure used in the previous section, but whose 15 nodes now have variance \( \sigma^2 = 1.5\pi \) and are centered on a regular grid of mesh \( \Delta = 0.25 \) over the set \([-1.25, 2.25]\), can approximate the iterated mapping with a worst case error of 0.0025 on \( A = [0, 1] \). Choosing again adaptation gains of \( k_a = 0.7 \) and a deadzone size \( \Phi = 0.005 \), Figure 5-6 shows the resulting error sequence, which again shows rapid convergence to the deadzone.

### 5.4.4 Continuous Time Identification

Consider now the problem of identifying the continuous time, linear process

\[
y'(t) = -3y(t) + 2u(t) + 1.
\]

Defining \( u'(t) = 1 + 2u(t) \), the structure of this differential equation implies that since

\[
y(t) = y(0)e^{-3t} + \int_0^t e^{-3(t-\tau)}u'(\tau)d\tau
\]

one can conclude that, if \( y(0) = 0 \) then \( \frac{u'_\infty}{3} \leq y(t) \leq \frac{u'_\infty}{3} \), where \( u'_\infty = \inf_{t \geq 0} u'(t) \) and similarly \( u'_\infty = \sup_{t \geq 0} u'(t) \). Thus, the assumptions that \( y(0) = 0 \) and that \( 0 \leq u'(t) \leq 3 \) ensure that \( y(t) \) remains in the set \( A_x = [0, 1] \) for all \( t \geq 0 \). From the definition of \( u'(t) \), this then requires \( u(t) \) to be contained in the set \( A_u = [-.5, 1] \).
CHAPTER 5. RECURSIVE IDENTIFICATION

To recursively identify this process, it is convenient to view instead the process dynamics as

\[ \dot{y} = f(y(t), u(t)) \]

where, clearly, \( f(x_1, x_2) = -3x_1 + 2x_2 + 1 \). This is precisely the function \( f_5 \) for which a Gaussian network construction was detailed in Section 4.8.5. Thus, the identification structure

\[ \dot{\hat{y}}(t) = -k_D e(t) + \sum_{k_1 = -4}^{6} \sum_{k_2 = -8}^{6} \hat{c}_{k_1, k_2}(t) \exp \left( -\pi \sigma^2 \left( (y(t) - k_1 \Delta)^2 + (u(t) - k_2 \Delta)^2 \right) \right) \]

where \( e(t) = \hat{y}(t) - y(t) \) is capable of learning the unknown function \( f \) with an accuracy of \( \epsilon_f = .002 \) everywhere on \( A = A_x \times A_u \), using the network construction parameters \( \sigma^2 = 0.4\pi \) and \( \Delta = 0.5 \). The resulting network contains 132 Gaussian nodes in the hidden layer, and thus there are 132 output weights which must be recursively learned by the adaptation law.

To ensure stable adaptation of the output weights, the adaptive algorithm (5.7) is used for the adjustable output weights with \( k_D = 5 \) and \( \Phi = 0.001 \), so that certainly the constraint \( k_D \Phi \geq \epsilon_f \) is satisfied. The adaptation gains are taken to be \( k_a = 200 \).

The continuous plant and identifier dynamics were simulated in C on a NeXTStation, using a fourth order, fixed stepsize Runge-Kutta algorithm with \( \Delta t = .0005 \) sec. Figure 5-7 shows the error in estimating the process output using this identifier structure and adaptation law when the input is taken as \( u(t) = 0.25 [1 + 3 \sin(3t) \cos(2t)] \). After a long settling time, the estimates are within the predicted bound \( |\hat{y}(t) - y(t)| \leq \Phi \). Once again, as Figure 5-7 shows, this does not imply convergence of all the output weights to their correct values, and in fact does not even imply that the total squared output weight mistuning is monotonically decreasing, as occurred in the discrete time examples considered above. Since the Lyapunov function here includes both the estimation errors \( \epsilon_{\Delta}(t) \) as well as the (weighted) parameter mistunings, the adaptive system can "trade off" estimation error for mistuning, so long as the sum squared total is nonincreasing.
Figure 5-1: Total square mistuning between the estimated output weights and those computed in the constructions of Section 4.8.1.
Figure 5-2: First 10000 outputs of the chaotic time series used to test the prediction algorithm.
Figure 5-3: The initial and asymptotic errors in the Gaussian network predictions of the output of the chaotic time series $y[t] = 3.75y[t-1](1 - y[t-1])$. 
Figure 5-4: Total squared mistuning between the estimated output weights and the output weights computed for $f_2$ in Section 4.8.2.
Figure 5-5: Prediction errors when using a trained one-step-ahead Gaussian predictor in parallel fashion for multistep prediction.
Figure 5-6: The initial and asymptotic errors in the Gaussian network predictions of the output of the chaotic time series two steps into the future.
Figure 5-7: Top: Error estimating the output of the continuous time, linear process when driven by the input signal specified in Section 5.4.4; Bottom: Total squared mistuning between the estimated output weights and those computed in Section 4.8.5.
CHAPTER 5. RECURSIVE IDENTIFICATION

5.5 Persistency of Excitation

With a perturbed linearly parameterized error model, both the linear and the SPR varieties, if the adaptive system is persistently excited, there may be no need for robust modification such as a deadzone in the face of uniformly bounded disturbances; the excitation is itself a form of robustness which prevents the parameters from drifting too far from their correct values [56, 111, 149]. If the parameterization is globally exact, with no disturbances present, persistently exciting signals in the adaptation mechanism also guarantee asymptotically perfect identification of the components of the process dynamics. That is, when the signals in the adaptation process are PE, not only does the error $e(t) \to 0$, but also the parameter mistuning $\dot{c}(t) \to 0$ as well [56, 111, 149].

While the above stability proofs have demonstrated the asymptotic convergence of the prediction errors to a vicinity of zero, they do not necessarily imply this stronger convergence result for the output weights of the networks used. A method is still needed to predict when the training of these neural networks will be persistently exciting. This section thus identifies particular sequences of network inputs which ensure satisfaction of the PE conditions in the above adaptive algorithms, and bounds the asymptotic fidelity of the identification models which result when these conditions are satisfied.

5.5.1 Persistency and Gaussian Networks

Given an adaptation mechanism of the form

$$\dot{c}[t] = \hat{c}[t-1] - k_a e[t]g(x[t])$$

the sequence of regressors, $g$, driving this equation are said to be persistently exciting [111, 149, 56] if for some positive, finite $T, \alpha_1, \alpha_2$ and all $t \geq 0$,

$$\alpha_1 I \leq \sum_{\tau=t}^{t+T-1} g(x[\tau])g(x[\tau])^T \leq \alpha_2 I.$$  \hfill (5.9)
If the regressor vector is uniformly bounded, satisfaction of the rightmost inequality is trivial (take, for example, \( \alpha_2 = T \sup_t \|g(x[t])\|^2 \)); it is ensuring a nonzero lower bound that presents a challenge.

Thus showing that the regressors are PE involves demonstrating that the inequality

\[
P(t) \triangleq \sum_{\tau=t}^{t+T-1} g(x[\tau]) g(x[\tau])^T \geq \alpha_1 I
\]

holds in discrete time, and similarly in continuous time

\[
P(t) \triangleq \int_t^{t+T} g(x(\tau)) g(x(\tau))^T d\tau \geq \alpha_1 I
\]

for some \( T > 0, \alpha > 0 \) and all \( t > t_0 \). To begin this analysis, let \( \mathcal{K} = \{ k \in \mathbb{Z}^n \mid \text{dist}(A, k\Delta) \leq \rho \} \), let \( S \) be the size of the network, so that \( S = \text{Card}(\mathcal{K}) \), and define \( \gamma : \{1, \ldots, S\} \to \mathcal{K} \) to be a one-to-one mapping of the integers from 1 to \( S \) to the indices contained in \( \mathcal{K} \). With this notation, the relevant signals for a persistency analysis of the preceding algorithms are the outputs of the Gaussian nodes, and hence

\[
g_i(x[t]) = \exp(-\pi \sigma^2 \|x[t] - \gamma(i)\Delta\|^2), \quad i = 1 \ldots S.
\]

Clearly \( T \geq S \); in fact, for the Gaussian radial basis function, as well as for networks constructed from many of the other radial functions in common use, if the sequence of network inputs corresponds to the centers of the hidden layer nodes, e.g. \( x[\tau] = \Delta k(\tau) \), where \( k(\tau) \) cycles through the elements of \( \mathcal{K} \) as \( \tau \) increases, then the hidden layer outputs are persistently exciting with \( T = S \). This is a consequence of results in the theory of positive definite functions, due to Schoenberg [151] and more recently extended by Micchelli [102]. These theorems establish sufficient conditions on functions \( h : [0, \infty) \to \mathbb{R} \) which ensure that the symmetric \( S \times S \) matrix whose elements are \( h(\|x_i - x_j\|), i, j = 1 \ldots S \), has a determinant bounded away from zero whenever the \( x_i \) are distinct points in \( \mathbb{R}^n \). These conditions are satisfied by many popular radial basis functions [126, 132, 102] and certainly by the radial Gaussian function [151].

To show the utility of this result for characterizing the PE conditions in the
CHAPTER 5. RECURSIVE IDENTIFICATION

Identification models considered above, denote by $\mathcal{M}$ the $S \times S$ matrix whose $(i, j)$th entry is $\exp(-\pi\sigma^2 \Delta^2 \|\gamma(i) - \gamma(j)\|^2)$; Schoenberg's and Micchelli's results thus show that this matrix has a determinant bounded away from zero. Choosing the input sequence to step through the centers of the nodes in the network, that is $x[t] = \gamma(1 + (t \mod S))\Delta$, for all $t \geq 0$ one obtains:

$$P_{i,j}[t] = \sum_{\tau=t}^{t+S-1} g(x[\tau])_i g(x[\tau])_j$$

$$= \sum_{\tau=0}^{S-1} g(x[\tau])_i g(x[\tau])_j$$

$$= \sum_{\tau=0}^{S-1} \exp(-\pi\sigma^2 \Delta^2 \|\gamma(\tau) - \gamma(i)\|^2) \exp(-\pi\sigma^2 \Delta^2 \|\gamma(\tau) - \gamma(j)\|^2)$$

$$= \sum_{\tau=0}^{S-1} \mathcal{M}_{r,i} \mathcal{M}_{r,j} = (\mathcal{M}^T \mathcal{M})_{i,j}$$

from which it follows that $P[t] > \alpha_1 \mathbf{1}$, $\alpha_1 > 0$, for any $t \geq 0$.

More generally, since any additional inputs to the network will only add additional semi-definite terms to the series, if a subset of the input sequence corresponds to the center of each Gaussian node at least once during every time interval of length $T \geq S$, the outputs of the hidden layer nodes in the network will also be persistently exciting.

The number $\alpha_1$ is determined by the smallest singular value of the matrix $\mathcal{M}$; as indicated below, this number controls both the rate of convergence of the errors, and the bounds on the amount of asymptotic mistuning in the network output weights; the larger this constant, the faster the convergence and the tighter the asymptotic parameter bounds [149]. Developing tight lower bounds for this number has been the subject of recent research in approximation theory, and the papers [109, 110] have estimates for most of the common radial basis functions. For the Gaussian networks considered here, the results of [109] show that, for any $x \in \mathbb{R}^n$, $x^T M x \geq \alpha_M \|x\|^2$,

where

$$\alpha_M = \kappa_2 (\sqrt{\pi} \sigma \Delta)^{-n} \exp \left( - \left( \frac{2\kappa_1}{\sqrt{\pi} \sigma \Delta} \right)^2 \right)$$
where
\[
\kappa_1 \triangleq 12 \left( \frac{\pi \Gamma^2 \left( \frac{n+2}{2} \right)}{9} \right)^{\frac{1}{n+1}}, \quad \kappa_2 \triangleq \frac{\kappa_1^2}{2\Gamma \left( \frac{n+2}{2} \right)},
\]
and \( \Gamma \) is Euler's gamma function.

The above results are easily exploited for the continuous time case. In this setting sufficient conditions for PE are that there exists a \( T > 0 \) such that, for every \( t > 0 \), there exist \( S \) constants \( \alpha_i \geq \delta > 0 \) and a collection of \( 2S \) points \( \{t_i^-, t_i^+\} \) in each interval \([t, t + T]\) with \( t_{i-1}^+ \leq t_i^- < t_i^+ \), so that the following equalities hold for each \( i = 1 \ldots S \)
\[
\int_{t_i^-}^{t_i^+} g(x(\tau))g(x(\tau))^T d\tau = \alpha_i g(\gamma(i))g(\gamma(i))^T.
\]
Then, using the above result
\[
P(t) \geq \sum_{i=1}^{S} \alpha_i g(\gamma(i))g(\gamma(i))^T \geq \delta \mathcal{M}^T \mathcal{M} \geq \alpha_1 I
\]
for some \( \alpha > 0 \) and all \( t \geq 0 \). The interpretation of these conditions in terms of the required network inputs is the same as in the discrete time case, but in a time averaged fashion. Certainly they will be satisfied if the network inputs dwell at each center for a nonzero length of time during every interval of length \( T \), but in practice this situation will rarely occur, and the above conditions give a more general formulation.

### 5.5.2 Output Weight Convergence and Generalization

When an unperturbed linearly parameterized model is globally exact, the unmodified gradient algorithm is known to create an adaptive system whose \((e, \dot{e})\) dynamics have the origin as a \textit{globally exponentially stable} equilibrium point, provided that the regressors \( g \) are sufficiently smooth and satisfy the persistency of excitation conditions [111, 149]. That is, not only the prediction errors, but also the mistuning in the identifier parameters converge exponentially to zero under these conditions, thus ensuring asymptotically perfect fidelity in the identifier model.

When the disturbance and deadzone are present, however, the situation is more complex. This section examines the asymptotic fidelity, and hence the generalization
capabilities, which can be expected from the Gaussian network identification algorithms examined above, under the assumption that the persistency conditions of the previous section are satisfied. In particular, it will be shown that, despite the fact that the Gaussian networks rarely provide an exact parameterization for the functions they attempt to estimate, certain exponential convergence properties for the output weights are still preserved. The analysis also makes clear the intrinsic robustness provided by the PE conditions.

Discrete Time

For a classical discrete-time recursive identification algorithm, in which the linear parameterization is globally exact allowing use of the unmodified gradient algorithm, the dynamics of the adaptive system can be expressed as

$$\dot{\mathbf{c}}[t] = \left( I - k_a \mathbf{g}(\mathbf{x}[t])\mathbf{g}^T(\mathbf{x}[t]) \right) \dot{\mathbf{c}}[t - 1].$$

Provided the regressors $\mathbf{g}$ are bounded and satisfy the PE conditions, and assuming that $k_a$ is chosen to satisfy (5.2), this system is globally exponentially stable to the origin [10, 56]. When deadzones and disturbances are present, as in the algorithms considered in Section 5.2, this classical equation becomes

$$\dot{\mathbf{c}}[t] = \dot{\mathbf{c}}[t - 1] - k_a e_\Delta[t] \mathbf{g}(\mathbf{x}[t]) \quad \text{for } t = 0, 1, \ldots, n - 1,$$

(5.10)

$$= \left( I - k_a \mathbf{g}(\mathbf{x}[t])\mathbf{g}^T(\mathbf{x}[t]) \right) \dot{\mathbf{c}}[t - 1] + \mathbf{d}[t].$$

The term $\dot{\mathbf{c}}$ here is a vector describing the mistuning in the collection of network output weights, ordered using the same mapping $\gamma$ which orders the hidden layer outputs contained in $\mathbf{g}$. The new forcing term is given by

$$\mathbf{d}[t] = k_a \mathbf{g}(\mathbf{x}[t]) \left[ \Phi \text{sat}(e[t]/\Phi) - d_f[t] \right],$$

and, using the assumptions in Section 5.2 on the behavior of the process and the approximation capabilities of the Gaussian network, can be uniformly bounded by

$$\sup_t \|\mathbf{d}[t]\|^2 \leq 4k_a^2 \Phi^2 \sup_{\mathbf{x} \in \Delta, \text{dist}(\mathbf{A}, \mathbf{K}) \leq \rho} \sum \exp(-2\pi \sigma^2 \|\mathbf{x} - k\mathbf{A}\|^2) \leq d_\infty^2.$$
The exponential stability of the unforced system implies the output of (5.10) will be bounded in response to the bounded input $\bar{d}$ [181]. Indeed, using the state transition matrix, $M(t, \tau)$, which here satisfies

$$M(t + 1, \tau) = (I - k_\alpha g(x[t])g^T(x[t]))M(t, \tau)$$

$$M(\tau, \tau) = I$$

the variation of constants formula is [181]

$$\dot{\hat{c}}[t] = M(t, 0)\hat{c}[0] + \sum_{r=0}^{t-1} M(t, \tau + 1)\bar{d}[\tau]$$

and by taking norms

$$\|\hat{c}[t]\| \leq \|M(t, 0)\|\|\hat{c}[0]\| + \bar{d}_\infty \sum_{r=0}^{t-1} \|M(t, \tau + 1)\|,$$

where the induced matrix norm is given by $\|M\|^2 = \lambda_{\text{max}}(M^TM)$ for any real matrix $M$. The exponential stability of the unforced equation implies the existence of a finite, positive constant $m$ and $0 \leq a < 1$ such that the induced norm of the state transition matrix satisfies the inequality [181]

$$\|M(t, \tau)\| \leq ma^{t-\tau}$$

so that

$$\frac{1}{m} \|\hat{c}[t]\| \leq a^t\|\hat{c}[0]\| + \bar{d}_\infty \sum_{r=0}^{t-1} a^r$$

and thus,

$$\lim_{t \to \infty} \|\hat{c}[t]\| \leq \frac{m\bar{d}_\infty}{1 - a}.$$

Thus, even with the disturbance and the deadzone, PE guarantees exponential decrease of the parameter mistuning to a neighborhood of the origin whose size depends continuously on the approximating capability of the network. Since the prediction errors in this identification model are a linear combination of the parameter mistuning and the network outputs, and since these latter are uniformly bounded, bounds on
the prediction errors will also decrease exponentially under these conditions. Indeed, asymptotically

\[
\lim_{t \to \infty} |N(x, \hat{c}[t]) - f(x)| \leq \lim_{t \to \infty} \left| \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}_k[t] \exp(-\pi \sigma^2 \|x - k\Delta\|^2) \right| + |d_f(x)|
\]

\[
\leq \epsilon_f + \kappa \frac{m d_{\infty}}{1 - a}
\]

for any input \( x \) in the set \( A \), where the constant \( \kappa \) is given by

\[
\kappa^2 = \sup_{x \in A} \sum_{\text{dist}(A,k\Delta) \leq \rho} \exp(-2\pi \sigma^2 \|x - k\Delta\|^2)
\]

and is easily evaluated from the network architecture. Noting that \( d_{\infty} \) approaches zero continuously as \( \epsilon_f \) does, if the training is persistently exciting the resulting identifier will develop a model of the process dynamics whose fidelity on \( A \) is asymptotically limited only by the approximation capabilities of the Gaussian network.

This argument is unchanged if the deadzone is omitted, that is by taking \( \Phi = 0 \) and removing the saturation term in the definition of \( d \). Making the appropriate change in the definition of \( d_{\infty} \), the above argument shows that the unmodified gradient algorithm will exhibit similar exponential convergence properties if the network inputs satisfy the conditions established in the previous section. Thus, persistency is a form of robustness, capable of preventing parameter drift without further modification of the adaptation mechanism. In practice, however, one may not have control over the evolution of the system being identified, and thus satisfaction of the PE conditions cannot be guaranteed. Thus, practical applications should not depend upon PE to provide the required robustness.

**Continuous Time**

The differential equations which govern the evolution of the output weight mistuning for the network estimate of the mapping \( h \) can be written as

\[
\ddot{c}^h(t) = -k_2 \left[ g(x(t))g^T(x(t)) \right] \dot{c}^h(t) + \bar{d}(t),
\]

(5.12)
where, as above, the forcing term is given by

$$
\overline{d}(t) = k_2 g(x(t)) [\Phi_h \text{sat}(e(t)/\Phi_h) - d_h(t)].
$$

Once again, using the assumptions in Section 5.3 on the behavior of the process and the approximation capabilities of the Gaussian network, this term is uniformly bounded in magnitude by $\overline{d}_\infty$ as above.

In the classical identification algorithms, with no deadzones and a globally exact linear parameterization of $h$, the term $\overline{d}$ is identically zero. If the regressors $g$ are sufficiently smooth and persistently exciting, this system is known to have $\dot{c} = 0$ as a globally exponentially stable equilibrium point [111, 149]. In the Gaussian network identification model, with the assumed constraints on the vector field $f$ and on the process input $u$, the regressors are bounded and continuous, hence satisfying the smoothness constraint [111, 149], and there remains only to characterize the impact of the disturbance term when the PE conditions are satisfied.

Since $x(t)$ is evolving independently of the parameter estimates, and is assumed to remain bounded, the differential equation (5.12) describes the response of a linear time-varying system to the bounded input $\overline{d}(t)$. Since the unforced system is exponentially stable, the system output in response to a bounded input is bounded [111]. Using the state transition matrix, $M(t, \tau)$, defined by

$$
\frac{d}{dt} M(t, \tau) = -k_2 g(x(t))g^T(x(t))M(t, \tau)
$$

$$
M(\tau, \tau) = I
$$

the value of the weight mistuning at any point in time can then be expressed as

$$
\dot{c}^h(t) = M(t, 0) \dot{c}^h(0) + \int_0^t M(t, \tau)\overline{d}(\tau)d\tau.
$$

Taking Euclidean norms on both sides of this equation produces the bound

$$
\|\dot{c}^h(t)\| \leq \|M(t, 0)\|\|\dot{c}^h(0)\| + \overline{d}_\infty \int_0^t \|M(t, \tau)\|d\tau.
$$
Since the unforced system is exponentially stable, this norm of the state transition matrix satisfies the inequality [173]
\[
\|M(t, \tau)\| \leq me^{-a(t-\tau)}
\]
for all \( t \geq \tau \), and some finite, positive constants \( m \) and \( a \). Substituting and integrating produces
\[
\lim_{t \to \infty} \|\tilde{z}^h(t)\| \leq \frac{m\tilde{d}_\infty}{a},
\]
and the bound is approached exponentially.

Thus, even in the presence of the disturbance and deadzone, if the network outputs are persistently exciting, the parameters exponentially converge to a neighborhood of the origin whose size depends continuously on the size of the deadzone, and hence on the quality of the approximation afforded by the Gaussian networks. Once again, even if no deadzone is used in the adaptation mechanism, provided the network outputs are PE, the above argument holds by simply deleting the saturation terms from the definition of \( \tilde{d} \). In particular, no instability will occur under these conditions.

Similar results can be obtained for the convergence of the network output weights which estimate the individual vector field components, \( f_j \), of the process. Combining the adaptation and error dynamics of these components one has,

\[
(5.13) \quad \dot{\xi}_j(t) = \Lambda(t)\xi(t) + \tilde{d}_j(t)
\]

where \( \xi_j^T = [e_j \ \dot{e}_j] \), \( \tilde{d}_j^T(t) = [d_{f_j,j}(t), \quad k_1 g^T(x(t))\Phi_j \text{ sat}(e_j/\Phi_j)] \), and
\[
\Lambda(t) = \begin{bmatrix}
-k_D & g^T(x(t)) \\
-k_1 g(x(t)) & 0
\end{bmatrix}.
\]
The term \( \dot{e}_j \) is here a vector describing the mistuning in the collection of weights associated with the \( j \)th network output.

Once again, provided the regressors are bounded with bounded derivatives and satisfy the PE conditions, the unforced dynamics have \( \xi_j = 0 \) as a globally exponentially stable equilibrium point [149]. Since the magnitude of each \( \tilde{d}_j(t) \) is easily seen
to be bounded in this case by

$$\sup_t \| \mathbf{d}_j(t) \|^2 \leq \Phi_j^2 \left( 1 + k_j^2 \sup_{x \in A_{\text{dist}(A, k\Delta_j) \leq \rho_j}} \exp(-2\pi\sigma_j^2 \| x - k\Delta_j \|^2) \right) \leq d_{\infty},$$

exactly the same argument used above produces

$$\lim_{t \to \infty} \| \mathbf{\xi}_j(t) \| \leq \frac{m'd_{\infty}}{a'},$$

for two different finite constants $m' > 0$ and $a' > 0$, showing that similar convergence properties hold for the output weights of the networks estimating the functions $f_j$.

As in the discrete case, the convergence of each of these sets of the output weights to a neighborhood of their tuned settings implies that the fidelity of the identification model is asymptotically bounded by,

$$\lim_{t \to \infty} |N_j(x, \hat{c}(t)) - f_j(x)| \leq \lim_{t \to \infty} \left| \sum_{\text{dist}(A, k\Delta_j) \leq \rho_j} \hat{c}_k(t) \exp(-\pi\sigma_j^2 \| x - k\Delta_j \|^2) \right| + |d_{f,j}(x)|$$

$$\leq \epsilon_f + \kappa \frac{m'd_{\infty}}{a'}$$

for any $x \in A$, where $\kappa$ is as above. Exactly the same inequality is obtained for the network approximation to $h$ with appropriate substitutions.

### 5.5.3 Examples

The fact that the process outputs appear in the network input vector makes it difficult to use the above analysis to specify a priori probing inputs $u$ which guarantee that the persistency conditions will be satisfied. Characterizing the process inputs which force the process outputs to pass through a given set of points would require both explicit knowledge of the process dynamics as well as methods for obtaining closed-form solutions to nonlinear differential and difference equations. The first of these is by assumption not available, and the second is available only for a limited class of dynamic systems, making it difficult to extend the persistency analysis much further than that given above.
CHAPTER 5. RECURSIVE IDENTIFICATION

In a supervised learning application, however, the only inputs to the network are points in the domain of the functions to be learned. Translating the persistency conditions into the language of learning theory, it is clear that by selecting the training examples to cycle through the lattice points, the resulting sequence of network inputs will be persistently exciting. Adaptation is thus robust to perturbations under these conditions, and no instability of the form observed at the end of Chapter 3 will occur in the training algorithm even if the deadzone is omitted.

This agrees nicely with the fact that, under these conditions, the corresponding interpolation problem can be solved exactly and the minimum training set error will in fact be zero [126, 102]. To illustrate, Figure 5-8 shows the results of using the unmodified gradient training algorithm for the learning problem considered in Section 3.3.1, but where the training data now correspond to the lattice points, $t_i = f_1((i - 9)\Delta)$. Cycling though this data, the training set errors exponentially approach zero, and moreover, the unbounded growth of the approximation in the vicinity of $x = -16/9$, which was the remarkable feature of the simulation discussed in Section 3.3.1, no longer occurs.

An examination of the output weights of the network after 60 training cycles (about 1000 training steps) reveals, as expected, that they are almost identical with the interpolation weights obtained by direct computation in the analysis of Section 4.8.1, indicating that this gradient learning procedure has recursively inverted the interpolation matrix. Recall, however, that this weight set is not the same as that obtained from the Fourier constructions developed in Section 4.8.1, confirming the above prediction that, with a nonzero disturbance present, the parameters will converge only to a neighborhood of values which actually minimize the uniform approximation error. Figure 5-9 illustrates the convergence of the weights for this situation, while Figure 5-10 shows the convergence of the successive network approximations to the function underlying the training data.

The above persistency argument also indicates that similar exponential conver-
gence will occur on any sequence which merely contains the lattice points, for example, a training sequence formed by injecting random examples into a sequence which cycles through the lattice points. Figure 5-11 shows the results obtained by interweaving examples drawn uniformly from \([-1,1]\) with the persistently exciting training sequence used above. Once again, the learning is stable, with no pathological weight drift. Exponential convergence of the output weights now occurs to within a much tighter ball of the Fourier weights computed in Section 4.8.1, and the worst case errors recorded on \(A\) exhibit similar exponential convergence to a neighborhood of the theoretical minimum of 0.26 as shown in Figure 5-12.
Figure 5-8: Convergence of the unmodified gradient training algorithm when the training set data corresponds to the centers used in the Gaussian network.
Figure 5-9: While the prediction errors on the training set converge, the output weights converge not to the Fourier weights computed in Section 4.8.1, which guarantee superior predictive ability across the entire set [-2,2], but rather to the interpolation weights computed in that section.
Figure 5-10: A portrait of the approximation abilities of the Gaussian network developed when the training data corresponds to the centers. The successive approximations in the bottom figure were recorded after 400, 800, 1600, 3200, 6400, 12800, and 25600 training cycles respectively. The dashed line is again the function underlying the training data. Compare with Figure 3-8 which also used the unmodified gradient algorithm, but with training data which was not persistently exciting.
Figure 5-11: Convergence of the unmodified gradient training algorithm with a more general PE training sequence which includes the Gaussian centers as a subset. Top: errors on training set. Bottom: convergence of the output weights to the Fourier weights computed in Section 4.8.1. Note that this convergence is to a much tighter neighborhood than that shown in Figure 5-9.
Figure 5-12: A portrait of the approximation abilities of the Gaussian network developed with the more general PE sequence. The successive approximations in the bottom figure were recorded after 400, 800, 1600, 3200, 6400, 12800, and 25600 training cycles respectively.
Chapter 6

Direct Adaptive Control

The uses of networks in the indirect control laws discussed in Section 3.1.3 assume that good identification models have been obtained. Despite the explicit characterization given above of the algorithms and conditions which ensure stable, high fidelity identification, it may not always be possible to maintain the prerequisite persistency conditions during the identification phase. Indirect control laws thus developed may hence be forced to employ approximations which are quite poor in certain parts of the state space, leading to the possibility of closed-loop instability as discussed in Section 3.3.3. While it may be possible to make the control law robust to the use of low quality approximations by appropriate modifications to its structure, as noted in Section 3.3.3 this may require the use of undesirably large control signals (e.g. large $k_D$ and $\bar{e}$ in the examples of that section), which may not be physically implementable by the available actuation.

Moreover, the separation of adaptive control algorithms into explicit identification and control stages seems artificial; ideally learning and control should occur simultaneously, allowing low quality approximations to be improved in real-time as needed. In contrast to the indirect algorithms examined in Section 3.1, one would thus prefer on-line algorithms of the type discussed in Section 3.2 which directly estimate the required control signal and are less sensitive to the persistency requirements; these are the subject of the current chapter.

Of course, it is possible to continue training the identifier and controller networks
in the indirect algorithms even after the networks have been brought on-line. Indeed, provided that an off-line training phase has produced network estimates sufficiently close to the actual functions required, and that the network parameters are changed sufficiently slowly, this strategy may result in a stable, closed loop adaptive system capable of providing quite satisfactory results [114]. Once again, the difficulty lies in quantifying exactly how accurate the models developed off-line must be, and just how slowly adaptation must proceed, in order to guarantee stability.

Carrying out learning and control simultaneously in this fashion introduces a level of complexity into the stability analysis which goes far beyond that used to design the identifier adaptation laws above. Whereas in an identification model the network merely passively observes the (assumed stable) process and adjusts its parameters accordingly, in an on-line, direct adaptive control setting the network interacts immediately with the process it observes, and must react quickly enough to stabilize even a potentially unstable process. The adaptation mechanism then couples the evolution of the time-varying parameters in the network to the dynamics of the system being controlled, producing a high-dimensional, nonlinear, closed-loop system, whose states are those of the original process together with the adjustable parameters of the network.

Generalities aside, the ability to carry the analysis further depends crucially upon the availability of suitable direct adaptive control structures for nonlinear systems. If no such designs are available given the prior assumption of the structure of the process dynamics, further advances in control theory are required before one can consider a stable neural network solution to the problem. The class of nonlinear systems for which stable techniques are known has grown substantially in the past few years [3, 75, 81, 88], but still represents only a limited subset of possible nonlinear process dynamics. Nonetheless, these special cases have wide applicability to a variety of practical applications [159].

To clearly illustrate the central ideas, this chapter starts by presenting direct
adaptive control structures for nonlinear systems in the canonical form \cite{[74, 159]}. Following an overview of adaptive feedback linearizing control strategies for such systems, Section 6.2.1 discusses the design specifics for the plant dynamics of (6.1) when the control gains \( b \) are known, and formally proves the stability of the design and the convergence of the tracking errors to a neighborhood of zero. Section 6.2.2 then extends the design and stability proof developed in Section 6.2.1 to include plants with unknown control gains. Section 6.3 next presents some general observations about the performance of the controller and the details of its implementation, based upon the stability proofs. Section 6.4 then presents several examples demonstrating the efficacy of the algorithms for the direct adaptive control of several different processes, both linear and nonlinear.

Using the insights developed from these algorithms, Section 6.5 extends the use of adaptive Gaussian networks to develop a direct adaptive control law for an important class of multivariable nonlinear systems, which include robot and spacecraft attitude dynamics as special cases. Section 6.6 then illustrates this design for a two degree of freedom robotic manipulator.

\section{6.1 Stable Adaptive Feedback Linearization}

\subsection{6.1.1 Adaptive Feedback Linearization}

A single-input, single-output nonlinear system is said to be in the canonical form \cite{[74, 159]} if the differential equation relating its observed output to the control input can be written as

\begin{equation}
 y^{(n)}(t) + f(x(t)) = b(x(t))u(t)
\end{equation}

where \( b \) is assumed globally bounded away from zero. The state of this plant, consisting of the output and \( n - 1 \) of its derivatives, are assumed to be available for measurement. The control objective is then to force the state vector, \( x^T = [y, \dot{y}, \ldots, y^{(n-1)}] \in \)
\( \mathbb{R}^n, t \) asymptotically follow a specified model trajectory, \( x_m^T = [y_m, y_m', \ldots, y_m^{(n-1)}] \).
(Note that this definition of \( x(t) \) omits the input \( u(t) \) compared with the definition used for the identification models). Defining a tracking error vector, \( \ddot{x}(t) = x(t) - x_m(t) \), the problem is thus to design a control law \( u(t) \) which ensures that \( \ddot{x}(t) \to 0 \) as \( t \to \infty \).

For nonlinear processes governed by these dynamics, the so-called feedback linearizing control law

\[
u(t) = b^{-1}(x(t))[y_m^{(n)}(t) + f(x(t)) - \alpha^T \ddot{x}(t)]\]

would accomplish the control objective if the functions \( f \) and \( b \) were known exactly [74, 159], since, with proper choice of the feedback gains, \( \alpha \), this control law causes the tracking error to evolve as the zero-input response of a stable, linear time-invariant system

\[
y^{(n)}(t) + \alpha_n \dot{y}^{(n-1)}(t) + \cdots + \alpha_1 \dot{y}(t) = 0.
\]

As in the identification case, a classical adaptive approach to this problem can be implemented if the functions \( f \) and \( b \) lie in the linear span of a set of prior, known basis functions so that

\[
f(x) = \sum_{k=1}^{N} c_k^f Y_k^f(x) \]

\[
b(x) = \sum_{k=1}^{N} c_k^b Y_k^b(x).
\]

By expressing estimates to these functions as

\[
f(x, \hat{c}^f(t)) = \sum_{k=1}^{N} \hat{c}_k^f(t) Y_k^f(x) \]

\[
b(x, \hat{c}^b(t)) = \sum_{k=1}^{N} \hat{c}_k^b(t) Y_k^b(x),
\]

a direct adaptive control solution to this tracking problem employs the certainty equivalence control law [159]

\[
u(t) = \frac{1}{b(x(t), \hat{c}^b(t))}[y_m^{(n)}(t) + \hat{f}(x(t), \hat{c}^f(t)) - \alpha^T \hat{x}(t)].
\]
Thus, at each instant in time, control signals are input to the process using a control law which replaces the actual function values in the feedback linearizing control law with the most recent estimates of the required values.

Stable adjustment of the parameters in this control law can be accomplished in real time if a measure of the tracking error can be found which satisfies two criteria. First, the error measure must contain sufficient information about the state of the plant so that, if the error can be proven uniformly bounded, the state of the plant is necessarily also uniformly bounded. Second, the error measure must be related to the parameter mistuning through one of the error models discussed in the previous chapter, that is, either the linear or the SPR model. Since \( y^{(n)}(t) \) is not generally considered available for measurement, a linear error equation will not usually be possible, since some dynamics must then exist between the measurable signals and the effects of the parameter mistuning. Fortunately, as will be shown, it is possible to construct an error signal from the available measurements which satisfies the SPR conditions.

By taking for these plants the error metric

\[
(6.2) \quad s(t) = \left( \frac{d}{dt} + \lambda \right)^{n-1} \dot{y}(t) = \lambda^T \dot{x}(t)
\]

for any \( \lambda > 0 \) so that \( \lambda_i = \left( \lambda^{n-1} \right) \lambda^{n-i} \), and choosing the feedback gains in the control law as \( \alpha_i = k_D \lambda_i + \lambda_{i-1} \) with \( \lambda_0 \triangleq 0 \) and \( k_D > 0 \), this measure evolves as

\[
\dot{s}(t) = -k_D s(t) + \sum_{k=1}^N \hat{c}_k^f(t) Y_k^f(x(t)) - \hat{u}(t) \sum_{k=1}^N \hat{c}_k^b(t) Y_k^b(x(t)),
\]

where, as usual, \( \hat{c}_k^f(t) = \hat{c}_k^f(t) - c_k^f \), and similarly for \( \hat{c}_k^b(t) \). This is precisely the kind of first order relation which typifies the SPR condition. Moreover, by considering the state realization of this error system,

\[
\begin{align*}
\dot{x}(t) &= \Lambda \dot{x}(t) + b \left( \sum_{k=1}^N \hat{c}_k^f(t) Y_k^f(x) - \hat{u}(t) \sum_{k=1}^N \hat{c}_k^b(t) Y_k^b(x) \right) \\
 s(t) &= \lambda^T \dot{x}(t)
\end{align*}
\]
where

\[
\Lambda = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_n \\
\end{bmatrix}
\]

and \(b = [0 \ 0 \ \cdots \ 1]^T\), it is easy to confirm that the pair \((\Lambda, \lambda)\) is detectable; in fact, using the above definitions of the \(\alpha_i\)

\[
\lambda^T(pI - \Lambda)^{-1}b = \frac{(p + \lambda)^{n-1}}{(p + k_D)(p + \lambda)^{n-1}}
\]

where again \(p\) is the Laplace variable, so that the \(n - 1\) unobservable modes correspond to stable poles at \(p = -\lambda\).

Hence uniform boundedness of \(s(t)\) implies uniform boundedness of each of the tracking error states, and moreover, if \(s(t)\) asymptotically approaches zero, so also do the error states [158]. Similarly, if the magnitude of \(s\) can be shown to be asymptotically bounded by a constant \(\Phi\), then the actual tracking errors can be shown [154] to be asymptotically bounded by:

\[
(6.3) \quad |\dot{x}^{(i)}(t)| \leq 2^i \lambda^{i-n+1} \Phi, \quad i = 0, \ldots, n - 1.
\]

Note that in the continuous time designs of Chapter 5 each of the \(e_j(t)\) evolved in a similar first order fashion. In the identification problems considered there, this was easy to arrange, since the "obvious" error measure, i.e. comparing each identifier state with the corresponding measured state, immediately produced a stable, first order differential equation relating the measured error to the function estimation error. In this direct adaptive control setting, however, the "obvious" metric with which to train the controller would be the discrepancy between the ideal linearizing control signal and that generated using the adaptive estimates; this is not a measurable quantity, since one cannot measure the true function values \(f(x(t))\) and \(b(x(t))\), and
hence more care is required in constructing a suitable error signal from the available measurements.

Most importantly, naively using the model tracking error \( e(t) = y(t) - y_m(t) \) as the controller training signal will generally not produce stable, closed-loop adaptive algorithms, since this signal will usually not satisfy the required first order relation with the estimation errors. The measure \( s(t) \) defined above, however, both satisfies the required relation, and provides a suitable indication of how well the controller is satisfying the tracking requirements: driving \( s(t) \) to zero asymptotically ensures that \( e(t) \) also approaches zero.

With this unperturbed SPR error system, the gradient adaptation mechanism

\[
\begin{align*}
\dot{\hat{c}}_k^f &= -k^f_a s(t) Y_k^f(x(t)) \\
\dot{\hat{c}}_k^b &= P_k \{ k_b^a \hat{u}(t) s(t) Y_k^b(x(t)) \}
\end{align*}
\]

then ensures that \( s(t) \) and all the parameter estimates are uniformly bounded, and further that \( s(t) \), and hence each of the tracking error states, asymptotically approach zero [150, 155]. The projection operators \( P_k \) here serve to ensure that the estimate \( \hat{b} \) remains globally invertible; this requires knowledge of a subset of the parameter space such that, if the estimates \( \hat{\theta}(t) \) are confined within it, the estimate \( \hat{b} \) will remain bounded away from 0. The simplest example is when the regressors \( Y_k^b \) are themselves strictly positive for all \( x \), and it is known that \( \hat{c}_k^b \geq C_k > 0 \). Under these conditions, and assuming that each weight estimate is initialized so that \( \hat{c}_k^b(0) > C_k \), the projection can be implemented as

\[
P_k(a) = \begin{cases} 
    a & \text{if } \hat{c}_k(t) > C_k \\
    a & \text{if } \hat{c}_k(t) = C_k \text{ and } a > 0 \\
    0 & \text{otherwise.}
\end{cases}
\]

In certain cases it is possible to modify the algorithm to avoid use of the projection operator; these will be examined in greater detail below.
These facts form the basis for many of the stable, direct adaptive tracking controller designs. The drawback is the relatively large amount of prior information which must be incorporated in the form of the exact structure of the basis functions, \( Y_k^f \) and \( Y_k^b \), required for the computation of \( \hat{u}(t) \). As in the identification models considered in Chapter 5, with much less prior knowledge these functions can be approximated to a chosen degree of accuracy using the output of the Gaussian networks considered in Chapter 4, producing an error equation whose mathematical structure strongly resembles that used above. The next section considers this possibility in more detail.

### 6.1.2 Gaussian Networks and Feedback Linearization

To begin combining the results of the previous section and of Chapter 4, notice that if the desired trajectories are contained in a compact subset, \( A_d \subseteq \mathbb{R}^n \), of the state space, in principle the tracking problem posed could be solved by an adaptive component of the control law capable of reconstructing the unknown functions \( f \) and \( b \) everywhere on \( A_d \). If the functions \( f \) and \( b \) are continuous, they can be uniformly approximated on compact subsets of the state space by neural networks whose inputs are the measured values of the states of the process. Using the certainty equivalence principle, estimates of these functions generated using a particular weight set can then immediately be used in place of the actual functions in the feedback linearizing control law, and an adaptation law devised to tune the weights toward the values which permit vanishingly small tracking errors.

Adopting this strategy, using networks \( \mathcal{N}_f \) and \( \mathcal{N}_b \) in place of the parameterizations \( \hat{f} \) and \( \hat{b} \) respectively, and choosing the same error metric as above, it is immediate that

\[
\dot{s}(t) = -k_D s(t) + [\hat{b}(x(t)) - \mathcal{N}_b(x(t), \hat{c}^b(t))] \hat{u}(t) + [\mathcal{N}_f(x(t), \hat{c}^f(t)) - f(x(t))].
\]

If, in addition, the functions to be approximated are known to belong to one of the
function classes discussed in Chapter 4, Gaussian network architectures can be chosen which permit linear expansions of the functions \( f \) and \( b \) in terms of the chosen Gaussian basis, and which can approximate these functions with uniform error bounded by \( \epsilon_f \) and \( \epsilon_b \) respectively on any compact set \( A \) containing \( A_d \). Thus, expressing

\[
\begin{align*}
\mathcal{N}_f(x(t), \hat{c}_f(t)) &= \sum_{\text{dist}(A, k\Delta_f) \leq \rho_f} \hat{c}_k^f(t) \exp(-\pi \sigma_f^2 \|x(t) - k\Delta_f\|^2) \\
\mathcal{N}_b(x(t), \hat{c}_b(t)) &= \sum_{\text{dist}(A, k\Delta_b) \leq \rho_b} \hat{c}_k^b(t) \exp(-\pi \sigma_b^2 \|x(t) - k\Delta_b\|^2)
\end{align*}
\]

the evolution of the error metric can be further expanded as

\[
\dot{s}(t) = -k_D s(t) + d_f(x(t)) - d_b(x(t)) \hat{u}(t) + \sum_{\text{dist}(A, k\Delta_f) \leq \rho_f} \hat{c}_k^f(t) \exp(-\pi \sigma_f^2 \|x(t) - k\Delta_f\|^2) - \hat{u}(t) \sum_{\text{dist}(A, k\Delta_b) \leq \rho_b} \hat{c}_k^b(t) \exp(-\pi \sigma_b^2 \|x(t) - k\Delta_b\|^2))
\]

where \( \hat{c}_k^f(t) = \hat{c}_k^f(t) - c_k^f \) and each \( c_k^f \) would be determined by the construction algorithms given in Chapter 4; the \( \hat{c}_k^b(t) \) are defined similarly. Alternately, if \( f \) and \( b \) belong to similar function classes, the estimates \( \hat{f} \) and \( \hat{b} \) could be generated by a single Gaussian network with two outputs.

The disturbances in this perturbed SPR error relation are given by \( d_f(x) = \mathcal{N}_f(x, c^f) - f(x) \) and similarly for \( d_b \) so that \( |d_f(x)| \leq \epsilon_f \) and \( |d_b(x)| \leq \epsilon_b \) for all \( x \in A \). The impact of these small terms on the closed-loop adaptive system could thus in principle be accommodated using the robust adaptation techniques discussed in the previous chapter. For example, by incorporating a deadzone of width \( \Phi \) into the tracking error metric,

\[
(6.4) \quad s_{\Delta}(t) = s(t) - \Phi \text{ sat}(s(t)/\Phi).
\]

and, as in the identification algorithms of the previous chapter, using this new signal in place of \( s(t) \) in the gradient adaptation algorithms, the tendency of the parameters to drift in response to the disturbance will be halted. The required size for \( \Phi \) will
be discussed in the specific designs which follow; for now, note that this signal $s_\Delta$ has several additional properties which will be exploited in the stability analysis conducted below: if $|s| < \Phi$ then $\dot{s}_\Delta = 0 = s_\Delta$, while if $|s| > \Phi$, then $\dot{s}_\Delta = \dot{s}$, and $|s_\Delta| = |s| - \Phi$.

The disturbances will remain bounded by $\epsilon_f$ and $\epsilon_b$, however, only if the plant state never leaves $A$. Unlike the identification case, where this situation was assumed in the problem statement, in an adaptive control setting, where the network estimation errors directly influence the evolution of the plant state vector, it is not generally possible to guarantee a priori that the plant will remain within any subset of its state space. Indeed, it is possible that, during the early stages of learning when the initial network approximations may be quite poor, the tracking errors would become so large that the plant state would leave any prespecified, compact $A$. Further, impulsive, unmodeled disturbances might also take the plant state outside this set.

As noted in the examples in Section 4.9, the disturbance terms may become quite large outside the set $A$, reflecting the rapid degradation of the ability of these networks to approximate functions outside of the set for which they were designed. Since the impact of these large disturbances on the tracking error will become quite significant if the state enters this region, evidently more is required to ensure closed-loop stability than a simple modification to the gradient adaptive algorithm; structural modifications to the control law of the kind discussed in Section 3.3.3 are required as well.

Having identified the problem, however, it can easily be overcome by including in the control law a component which takes over from the adaptive component as its approximation ability begins to degrade, and which forces the state back into $A$. Given the first order relation which exists between $s(t)$ and the uncertainty on $f$ and $b$, a nonlinear control methodology known as sliding control [159, 172] can be used to "overpower" large disturbances and drive $s(t)$ toward zero, given only crude bounds on the magnitudes of $d_f$ and $d_b$. The sliding control component usually employs the
signum function, as in the example in Section 3.3.3, however it is possible to smooth out the resulting discontinuity by instead using the control input

\[ u_{sl}(t) = - \text{sat}(s(t)/\Phi) \tilde{b}(\mathbf{x}(t))^{-1} \left[ \tilde{d}_b(\mathbf{x}(t))|\dot{\mathbf{x}}(t)| + \tilde{d}_f(\mathbf{x}(t)) \right] \]

where \( 0 < \tilde{b}(\mathbf{x}) \leq b(\mathbf{x}) \), \( \tilde{d}_f(\mathbf{x}) \geq |d_f(\mathbf{x})| + \eta \), and \( \tilde{d}_b(\mathbf{x}) \geq |d_b(\mathbf{x})| + \eta \), for some small constant \( \eta > 0 \). The constant \( \Phi \) describes the width of a a boundary layer, which can be taken to be the same as the deadzone width in the adaptation mechanism.

This sliding component can be added to the adaptive linearizing control law as needed to ensure closed-loop stability. However, the definitions of the required gains include the (inaccurate) outputs of the tuned neural networks when the state lies outside of \( A \); this means that the sliding component must actually fight against the output of the networks in these regions. To prevent this contention, it is useful to simply shut off the adaptive components as the state moves beyond the boundaries of \( A \), relying only upon the sliding controller in this region. Similarly, since the adaptation mechanism is robust to small disturbances and small tracking errors can be tolerated, the sliding component can be omitted from the control when the state is within \( A \).

The complete control law analyzed below thus has a dual character, acting as either a sliding or as an adaptive controller depending upon the instantaneous location of the plant state vector. To avoid discontinuously switching between the adaptive and sliding components, the set \( A \), on which the network approximation accuracy can be guaranteed, is chosen to be slightly larger than the "nominal operating range" for the process, \( A_d \). Pure sliding operation is then restricted to the exterior of \( A \), while pure adaptive operation is restricted to the interior of the set \( A_d \); in between, in the region \( A - A_d \), the two modes are effectively blended using a continuous modulation.
function, \( m(t) \), of the form

\[
\begin{align*}
m(t) &= 0 & \text{if } x(t) \in A_d \\
0 < m(t) < 1 & \text{ if } x(t) \in A - A_d \\
m(t) &= 1 & \text{ if } x(t) \in A^c.
\end{align*}
\]

A block diagram of this controller structure is shown in Figure 6-1 for reference, while Figure 6-2 shows the relations among the various subsets of the state space used to design the controller.

To aid in the construction of this modulation function, \( A_d \) can be chosen to correspond to a unit ball with respect to an appropriate weighted norm function. This certainly poses no problem since, as long as \( A_d \) contains all the trajectories the plant will be required to follow, and \( A \) is chosen to completely contain \( A_d \), the actual dimensions of these sets are arbitrary. Thus, \( A_d \) and \( A \) can be defined by

\[
(6.5) \quad A_d = \{ x \mid \| x - x_0 \|_{p,w} \leq 1 \} \quad \text{and} \quad A = \{ x \mid \| x - x_0 \|_{p,w} \leq 1 + \Psi \}.
\]

Here \( \Psi \) is a positive constant representing the width of the transition region, \( x_0 \) fixes the absolute location of the sets in the state space of the plant, and \( \| x \|_{p,w} \) is a weighted \( p \)-norm of the form

\[
\| x \|_{p,w} = \left\{ \sum_{i=1}^{n} \left( \frac{|x_i|}{w_i} \right)^p \right\}^{\frac{1}{p}},
\]

or, in the limiting case \( p = \infty \),

\[
(6.6) \quad \| x \|_{\infty,w} = \max \left( \frac{|x_1|}{w_1}, \frac{|x_2|}{w_2}, \ldots, \frac{|x_n|}{w_n} \right),
\]

for a set of strictly positive weights \( \{ w_i \}_{i=1}^{n} \). In \( \mathbb{R}^2 \), for example, with \( p = 2 \) the sets \( A_d \) and \( A \) are ellipses, and with \( p = \infty \) these sets are rectangles. With these definitions, one possible choice for a modulation function is

\[
(6.7) \quad m(t) = \max (0, \text{ sat} \left( \frac{r(t) - 1}{\Psi} \right)),
\]

where \( r(t) = \| x(t) - x_0 \|_{p,w} \). When \( r(t) \leq 1 \), meaning that \( x \in A_d \), the output of the saturation function is negative, hence the maximum which defines \( m(t) \) is zero,
as desired. When \( r(t) \geq 1 + \Psi \), corresponding to \( x \in A^c \), the saturation function is unity, hence \( m(t) = 1 \), again as desired. In between, for \( x \in A - A_d \), it is easy to check that \( 0 < m(x) < 1 \).

Thus, unlike the classical adaptive linearizing algorithms considered in the previous section, whose basis functions are sufficient to achieve a globally exactly match to the plant dynamics, the representation implemented by an adaptive Gaussian network is only approximately accurate on a subset of the entire state space. This introduces additional complexities in the design of a stable tracking control strategy whose solutions require a combination of techniques from both robust adaptive and robust nonlinear control theory. The resulting composite controller smoothly transitions between adaptive and nonadaptive operation and, as will be shown below, is capable of a globally stable solution to the tracking problem posed in this section.

### 6.2 Controller Designs and Stability Analysis

#### 6.2.1 Unity Control Gain

In this section, the above ideas are applied to design a control law for a system in the canonical form:

\[
y^{(n)}(t) + f(y(t), \dot{y}(t), \ldots, y^{(n-1)}(t)) = u(t),
\]

where it is assumed that a prior upper bound \( M_0(x) \) is known on the magnitude of \( f \) for points outside of the set \( A_d \), i.e.

\[
|f(x)| \leq M_0(x) \quad \text{when} \quad x \in A_d^c.
\]

Having chosen a set \( A \) containing \( A_d \) as outlined above, assume that the function class to which \( f \) belongs is known, and allows a choice of Gaussian network parameters \((\sigma, \Delta, \rho)\) such that the network can represent functions in this class to a uniform accuracy of \( \epsilon_f \) everywhere on \( A \). Denoting as usual this network expansion as \( \mathcal{N}(x, c) \),
the time derivative of the error metric (6.2) can then be written as

\[ \dot{s}(t) = a_r(t) - \mathcal{N}(x(t), c) + u(t) + d_f(t) \]  

where \( a_r(t) = \lambda_v^T \hat{x}(t) - y_m^{(n)}(t) \) with \( \lambda_v^T = [0, \lambda^{n-1}, (n-1)\lambda^{n-2}, \ldots, (n-1)\lambda] \), \( y_m^{(n)}(t) \) is the \( n \)th derivative of the desired trajectory, and the disturbance \( d_f(t) = \mathcal{N}(x(t), c) - f(x(t)) \) satisfies \(|d_f(t)| \leq \epsilon_f \) for all \( t \) such that \( x(t) \in A \).

This expression, and the considerations of the previous section, suggests use of the control law:

\[ u(t) = y_m^{(n)}(t) - \alpha^T \hat{x}(t) - m(t) M_0(x(t)) \text{ sat}(s(t)/\Phi) 
+ (1 - m(t)) \mathcal{N}(x(t), \hat{c}(t)), \]

where the gains \( \alpha = k_D \lambda + \lambda_v \) as defined in Section 6.1.1, and the modulation function which blends the two controller modes is specified by equation (6.7). Figure 6-3 illustrates the structure of the network which implements the adaptive component of this control law.

Using this control law, the time derivative of the error metric can be written as

\[ \dot{s}(t) = -k_D s(t) + (1 - m(t)) (\dot{f}_A(t) + d_f(t)) - m(t) (M_0(x(t)) \text{ sat}(s(t)/\Phi) + f(x(t))), \]

where for compactness

\[ \dot{f}_A(t) = \mathcal{N}(x(t), \hat{c}(t)) - \mathcal{N}(x(t), c) \]

\[ = \sum_{\text{dist}(A, k\Delta) \leq \rho} \hat{c}_k(t) \exp(-\pi \sigma^2 \|x(t) - k\Delta\|^2) \]

and, as usual the output weight mistuning is \( \dot{c}_k(t) = \hat{c}_k(t) - c_k \), where the tuned output weights, \( c_k \), are those which would be determined using the construction algorithms of Chapter 4.

If the network output weights are continuously updated as:

\[ \dot{c}_k(t) = -k_a (1 - m(t)) s_\Delta(t) \exp(-\pi \sigma^2 \|x(t) - k\Delta\|^2) \]
where \( k_a \) is a positive constant determining the adaptation rate, and if \( \Phi \) is chosen so that \( \Phi \geq \epsilon_f/k_D \) in both the sliding controller and the calculation (6.4) of \( s_\Delta \), then all states in the adaptive system will remain bounded, and moreover the tracking errors will asymptotically converge to a neighborhood of zero given by (6.3).

To prove this assertion, consider the Lyapunov function candidate:

\[
V(t) = \frac{1}{2}(s_\Delta^2(t)) + \frac{1}{k_a} \sum_{\text{dist}(A,k\Delta) \leq \rho} \dot{c}_k(t)^2.
\]

While \( \dot{s}_\Delta \) is not defined when \( |s| = \Phi \), \( (d/dt)s_\Delta^2 \) is well defined and continuous everywhere and can be written \( (d/dt)s_\Delta^2 = 2s_\Delta \dot{s}_\Delta \). Hence, using (6.12), \( \dot{V} = 0 \) when \( |s| \leq \Phi \). More generally, since \( s_\Delta \text{sat}(s/\Phi) = |s_\Delta| \), using (6.11) one has

\[
\dot{V} = -(k_D s_\Delta^2 + |s_\Delta| k_D \Phi) + s_\Delta (1-m)(\dot{f}_A + d_f) + k_a^{-1} \sum_{\text{dist}(A,k\Delta) \leq \rho} \dot{c}_k \dot{c}_k - m(s_\Delta f + M_0|s_\Delta|)
\]

so that, substituting (6.12) and using obvious bounds,

\[
\dot{V} \leq -k_D s_\Delta^2 + (1-m) |s_\Delta|(|d_f| - k_D \Phi) + m |s_\Delta|(|f| - k_D \Phi - M_0).
\]

By construction of the Gaussian network and modulation function, \( |d_f| \leq \epsilon_f \) whenever \( m < 1 \). Since \( k_D \Phi \geq \epsilon_f \), and \( 0 \leq m(t) \leq 1 \) for all \( t \geq 0 \), the second term on the right is non-positive. Similarly, the choice of sliding controller gains ensures that the third term is also non-positive.

Thus, one has \( \dot{V} \leq -k_D s_\Delta^2 \) for all \( t \geq 0 \), and hence if \( s_\Delta \) and all the \( \dot{c}_k \) are bounded at time \( t = 0 \), they remain bounded for all \( t \geq 0 \). Since \( s(t) \) is uniformly bounded, it is easily shown [158] that, if \( \dot{x}(0) \) is bounded, then \( \dot{x}(t) \) is also bounded for all \( t \), and thus, since \( x_m(t) \) is bounded by design, \( x(t) \) is as well. To complete the proof and establish asymptotic convergence of the tracking error, it is necessary to show that \( s_\Delta \to 0 \) as \( t \to \infty \). This can again be accomplished by applying Barbalat’s lemma [159] to the continuous, non-negative function,

\[
V_1(t) = V(t) - \int_0^t (\dot{V}(\tau) + k_D s_\Delta^2(\tau))d\tau \quad \text{with} \quad \dot{V}_1(t) = -k_D s_\Delta^2(t).
\]
As before, use of (6.13) has already established that all signals in the system are bounded, the function $V_1$ is used only to conclude asymptotic convergence of the tracking error, as is now demonstrated.

By definition, $\dot{s}_\Delta(t)$ is either 0 or $\dot{s}(t)$, where $\dot{s}(t)$ is given by (6.11). By assumption $f(x)$ is continuous, and hence since $x(t)$ is bounded $f(x(t))$ is bounded for all $t$; similarly the modulation $m(t)$ and sliding controller gains are both bounded. Since the parameter error is bounded, and the tuned parameter values $c_k$ are finite, the parameter estimates $\hat{c}_k(t)$ are bounded for all $t \geq 0$; this fact, and the fact that the Gaussian basis functions are bounded for all values of $x$ shows that the term $\dot{f}_A + df$ is bounded.

Every term on the right side of (6.11) is thus bounded, hence $\dot{s}$ is bounded, which proves $\dot{V}_1$ to be a uniformly continuous function of time. Since $V_1$ is bounded below by 0, and $\dot{V}_1 \leq 0$ for all $t$, use of Barbalat's lemma proves that $\dot{V}_1(t) \to 0$ and hence from (6.15) that $s_\Delta(t) \to 0$ as $t \to \infty$. This means that the inequality $|s(t)| \leq \Phi$ is obtained asymptotically, and the asymptotic tracking error bounds follow using (6.3).

### 6.2.2 Non-Unity Control Gains

The previous design procedure can be extended to plants with non-unity control gains, of the form:

\begin{equation}
(6.16) \quad y^{(n)}(t) + f(x(t)) = b(x_T(t)) u(t).
\end{equation}

where $x_T = [y, \dot{y}, \ldots, y^{(n-2)}]^T$, and it is assumed that the control gain $b(x_T(t))$ is finite, non-zero, and of known sign for all $x_T$; without loss of generality this sign can be taken as positive.

As in the previous section, it is assumed that the unknown nonlinear functions can be bounded outside the set $A_d$, so that in this region of the state space

$$b^{-1}(x_T) \leq M_1(x_T).$$
In light of the development below, it is most convenient to combine the assumed bound on $f$ with that on $b^{-1}$, to produce the corresponding bound for $x \in A_d^c$

$$|b^{-1}(x_T)f(x)| \leq M_0(x).$$

It is further assumed that a prior bound is known for the norm of the gradient of $b^{-1}(x_T)$, so that for any $x \in A_d^c$

$$|\dot{b}^{-1}(x_T)| = |\nabla b^{-1}(x) \dot{x}_T| \leq M_2(x)||x||. \quad (6.17)$$

These additional restrictions on $b$ will be used to develop an adaptive linearization algorithm which does not require the use of projections in the adaptation mechanism. To this end, define the nonlinear function $h = b^{-1}f$, and assume that it is known that both $h$ and $b^{-1}$ belong to one of the function classes examined in Chapter 4, allowing Gaussian network design parameters to be chosen so that the network expansions $\mathcal{N}_h(x, c^h)$ and $\mathcal{N}_b(x, c^b)$ approximate the functions $h$ and $b^{-1}$ respectively with the properties, valid on the chosen set $A$:

$$|\mathcal{N}_h(x, c^h) - h(x)| \leq \epsilon_h \quad \text{and} \quad |b^{-1}(x_T) - \mathcal{N}_b(x, c^b)| \leq \epsilon_b \quad (6.18)$$

where $\epsilon_h$ and $\epsilon_b$ are as small as desired. As mentioned at the end of Section 4.4, there is no conceptual problem approximating $b^{-1}(x_T)$ by a function of the entire state vector instead of just $x_T$; this fact will also allow a possible simplification in the design of the adaptive subsystem, as will be discussed below.

With these definitions, the time derivative of the error metric (6.2) satisfies

$$b^{-1}(x_T(t)) \dot{s}(t) = \mathcal{N}_b(x(t), c^b) a_r(t) - \mathcal{N}_h(x(t), c^h) + u(t) + d(t), \quad (6.19)$$

where the disturbance term $d$ is here given by

$$d(t) = \left[ b^{-1}(x_T(t)) - \mathcal{N}_b(x(t), c^b) \right] a_r(t) - \left[ h(x(t)) - \mathcal{N}_h(x(t), c^h) \right]$$

and thus satisfies the inequality $|d(t)| \leq \epsilon_h + \epsilon_b |a_r(t)|$ when $x(t) \in A$. 
CHAPTER 6. DIRECT ADAPTIVE CONTROL

Equation (6.19) and the considerations of Section 6.1.1 suggest a control law of the form:

\[
\begin{align*}
  u(t) &= -k_D \, s(t) - \frac{1}{2} M_2(x(t)) \|x(t)\| \, s_{\Delta}(t) + m(t) \, u_{sl}(t) \\
     &\quad - (1 - m(t)) \left[ a_r(t) \mathcal{N}_b(x(t), \hat{c}^b(t)) - \mathcal{N}_h(x(t), \hat{c}^h(t)) \right],
\end{align*}
\]

(6.20)

for some positive constant \(k_D\). The sliding term, \(u_{sl}(t)\), is again given by \(u_{sl}(t) = -k_{sl}(t) \, \text{sat}(s(t)/\Phi)\) with gain

\[
k_{sl}(t) = M_0(x(t)) + M_1(x(t)) |a_r(t)|.
\]

(6.21)

Note that the adaptive component of this control law estimates \(b^{-1}\) directly, as opposed to inverting an estimate of \(b\) as in the classical designs considered above, thus avoiding the need for projections in the adaptation mechanism.

The adaptive components of this control law could each be realized as the output of a network of radial Gaussian nodes similar to the one described for the unity gain system of Section 6.2.1. Each network would be designed considering the properties of the function classes to which \(h\) and \(b^{-1}\) belong, as well as the set of desired trajectories the system is required to track, and the approximation accuracy desired. If the function classes are sufficiently similar, this design can be simplified somewhat by employing a single network, with the \(n\) state variables as inputs, and the two adaptive approximations as the outputs. The number of Gaussian nodes and the corresponding input weights for this network are chosen to correspond to a sampling grid with the smaller of the two mesh sizes which would be used in the separate networks. This network then has two sets of output weights for each node, \(\hat{c}_h^k(t)\) and \(\hat{c}_b^k(t)\), creating two outputs which implement the approximations \(\mathcal{N}_h(x(t), \hat{c}^h(t))\) and \(\mathcal{N}_b(x(t), \hat{c}^b(t))\) respectively:

\[
\begin{align*}
  \mathcal{N}_h(x(t), \hat{c}^h(t)) &= \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}_h^k(t) \exp(-\pi \sigma^2 \|x(t) - k\Delta\|^2) \\
  \mathcal{N}_b(x(t), \hat{c}^b(t)) &= \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}_b^k(t) \exp(-\pi \sigma^2 \|x(t) - k\Delta\|^2).
\end{align*}
\]

(6.22)
As before, the input weights encoding the sampling mesh are fixed in this architecture, and the output weights \( \hat{c}^h_k(t) \) and \( \hat{c}^b_k(t) \) are adjusted to match their tuned values, \( c^h_k \) and \( c^b_k \), in the Gaussian expansions on \( A \) to \( h \) and \( b^{-1} \) respectively. Figure 6-4 illustrates the structure of the adaptive component of this control law.

Using control law (6.20) in (6.19) produces

\[
(6.23) \quad b^{-1} \dot{s} = -k_D s - \frac{1}{2} M_2 \| x \| s_\Delta + (1 - m) (\hat{h}_A - \overline{b}_{\Delta} a_r + d) + m (u_{sl} + b^{-1} a_r - h)
\]

where as above the output weight mistuning \( \hat{c}^h = \hat{c}^h_k(t) - c^h_k \) gives rise to the function estimation errors

\[
\hat{h}_A(t) = \mathcal{N}_h(x(t), \hat{c}^h(t)) - \mathcal{N}_h(x(t), c^h) = \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}^h_k(t) \exp(-\pi \sigma^2 \| x(t) - k\Delta \|^2)
\]

and similarly for \( \overline{b}_{\Delta}^{-1} \) and \( \hat{c}^b(t) \).

Choose the adaptation law for the output weights of the network to be:

\[
\hat{c}^h_k(t) = -k_a \left[ (1 - m(t)) s_\Delta(t) \exp(-\pi \sigma^2 \| x(t) - k\Delta \|^2) \right]
\]

\[
(6.24) \quad \hat{c}^b_k(t) = k_{a2} a_r(t) \left[ (1 - m(t)) s_\Delta(t) \exp(-\pi \sigma^2 \| x(t) - k\Delta \|^2) \right]
\]

where \( k_{a1} \) and \( k_{a2} \) are strictly positive constants determining the adaptation rate, and define

\[
\epsilon_r = \epsilon_h + \epsilon_b(2L \| r \|_{\max} + \| y^{(n)}(m) \|_{\max}),
\]

with \( r \) being the radius of the smallest \( n \)-ball completely containing the set \( A \), and \( \| y^{(n)}(m) \|_{\max} \) an upper bound on the magnitude of the \( n \)th derivative of the reference trajectory. If \( \Phi \) is then chosen such that \( \Phi \geq \epsilon_r/k_D \) in both the sliding controller and the calculation (6.4) of \( s_\Delta \), all states of the adaptive system will remain bounded, and moreover the tracking errors will asymptotically converge to a neighborhood of zero given by (6.3).
To prove stability and convergence, consider the non-negative function:

\[ V(t) = \frac{1}{2} \left[ b^{-1}(x_T(t)) s_\Delta(t)^2 + \frac{1}{k_{a1}} \sum_{\text{dist}(A,k) \leq \rho} \dot{c}_k(t)^2 + \frac{1}{k_{a2}} \sum_{\text{dist}(A,k) \leq \rho} \ddot{c}_k(t)^2 \right] . \]

As before, when \(|s| \leq \Phi\), \(\dot{V} = 0\) trivially; more generally, using (6.23)

\[
\dot{V} = \frac{1}{2} \left( \dot{b}^{-1} - M_z \|x\| - 2k_D s_\Delta^2 - |s_\Delta| k_D \Phi + m (s_\Delta (b^{-1}a_r - h) - |s_\Delta|k_{st}) \right. \\
+ (1 - m) s_\Delta (\dot{h}_A - b^{-1}a_r + d) + \left. k_{a1}^{-1} \sum_{\text{dist}(A,k) \leq \rho} \dot{c}_k \dot{c}_k + k_{a2}^{-1} \sum_{\text{dist}(A,k) \leq \rho} \ddot{c}_k \ddot{c}_k \right)
\]

where \(\dot{b}^{-1}\) is the time derivative of \(b^{-1}\). Using (6.17) and substituting (6.24) produces

\[
\dot{V} \leq -k_D s_\Delta^2 + (1 - m) |s_\Delta| (|d| - k_D \Phi) + m |s_\Delta| (|b^{-1}|a_r| + |h| - k_D \Phi - k_{st}).
\]

By construction, \(m < 1\) implies that \(x \in A\), hence \(|a_r| \leq \|x\| (2r_A) + \|y_m^{(n)}\|_{\text{max}}\), and the uniform bounds (6.18) hold in this region. The inequality \(|d| \leq \epsilon_r\) thus holds when \(m < 1\); since \(\Phi\) has been chosen such that \(\Phi \geq \epsilon_r/k_D\), and \(0 \leq m(t) \leq 1\) for all \(t \geq 0\), the second term on the right is less than or equal to zero. The choice of sliding controller gains ensures that the third term on the right is also non-positive.

Thus it has been established that the specified control and adaptation laws produce \(\dot{V}(t) \leq -k_D s_\Delta^2\) for all \(t \geq 0\). Hence, if the initial values of the states of the plant and of the parameter estimates are bounded, they remain bounded for all positive time. Finally, given the assumed constraints on \(b^{-1}\) and using (6.23), an argument exactly parallel to that of the unity gain case of Section 6.2.1 can be used to establish that \(s_\Delta(t) \rightarrow 0\) as \(t \rightarrow \infty\); use of (6.3) then provides the asymptotic bounds on the individual tracking error states. Note that, given the definitions of \(\Phi\) and \(\epsilon_r\), these bounds imply that a good approximation to \(b^{-1}\) is more important in minimizing asymptotic tracking errors than is the approximation to \(h\); this consideration should be reflected in the network design.
CHAPTER 6. DIRECT ADAPTIVE CONTROL

6.3 Remarks

*Asymptotic controller properties:* Since the desired trajectories are contained completely within $A_d$, and since the tracking errors converge to a neighborhood of zero, the above proofs also show that eventually all the plant trajectories will converge to a set which is either within a small neighborhood of the set $A_d$, or else is completely contained inside it. In particular, the sliding subsystem will asymptotically be used less and less (if, indeed, it is *ever* used) and, since the negative feedback terms decay to zero as the tracking error does, the control input will eventually be dominated by the feedforward term and the outputs of the Gaussian network, regardless of the initial conditions on the plant or network output weights. This is a particularly attractive result, since the sliding controller is present only as a backup system, meant to stabilize the system during the initial phases of learning.

*Relation to learning control:* The sliding controller and modulation can be thought of as a formalization of the heuristic procedure, often used in connectionist learning control applications (e.g. [5]), of monitoring and "restarting" the learning process when a prespecified "failure condition" is indicated. Here the failure condition is the plant state leaving the nominal operating range, $A_d$, and learning is gradually halted as the state moves outside this set. However, since the plant is assumed to be in continual operation, it cannot be simply "shut down" and "restarted". Instead, the combined action of the modulation and the sliding controller smoothly force the plant state back into the nominal range, whereupon learning is resumed.

*Other network models:* From the standpoint of the controller design or stability proof there is nothing special about the role of the Gaussian. Its use in this paper stems from the attractive space-frequency localization properties which make possible the analysis and synthesis procedure discussed in Chapter 4. However, provided the uniform approximation capability of the chosen network model can be characterized, enabling the proper choice of deadzone size, the Gaussian can be replaced by any of
the other popular functions in the neural network and radial basis function literature, without changing the stability and convergence properties of the adaptive system.

Parameter convergence: Once again, it must be emphasized that the fact that the tracking errors converge to a neighborhood of zero does not, in general, imply that the output weights converge to a neighborhood of their correct values. This latter situation will not obtain unless the signals which appear in the output weight adaptation laws are persistently exciting, as discussed in Section 3.3.2. In the unity gain case, these conditions will be satisfied if the network inputs satisfy the criteria discussed in Section 5.5. The regressors which must satisfy the persistency condition for the estimates \( \hat{c}^b \) in the no-unity gain case, however, include the network outputs multiplied by the signal \( a_r \), and the conditions on the network inputs discussed in the previous chapter will not, in general, ensure persistency.

Adaptation gains: An adaptive controller can compensate for dynamic effects attributable only to the states it "sees", i.e. those which are measured and which form the inputs to the Gaussian networks. There is no way for the adaptive architecture described to compensate for effects due to (temporal) high frequency modes inevitably neglected in assuming a finite dimensional dynamic model for the process. It is thus necessary in practice to limit the magnitude of the adaptation gains used to ensure that the control signal issued by the network is free from any temporal high frequency components resulting from rapid adaptation which might excite high frequency modes of the process. Inspection of the adaptation laws suggests that the magnitude of the adaptation signals will be much larger for the \( \hat{c}^k \) than for the \( \hat{c}^h \). To prevent excessively rapid adaptation from adding high frequency components to the control signal, the adaptation gain \( k_{a2} \) should be chosen much smaller than \( k_{a1} \).

Deadzone size: Using the same parameter to control both the adaptation deadzone and the sliding controller boundary layer allows one to guarantee convergence of the tracking errors to the same neighborhood of zero regardless of the location of the
state vector. However, if extreme tracking accuracy is required, and the network is designed to produce the required reconstruction accuracy in the nominal operating range of the process, the resulting choice of \( \Phi \) might produce a sliding controller with a boundary layer so thin it risks exciting high frequency dynamics when this component is active [154]. This suggests that the value of \( \Phi \) used, and hence the resulting tracking accuracy, must arise as a tradeoff between the frequency range of the unmodeled dynamics and the trajectory following requirements. One possible way to implement such a tradeoff is to actually let \( \Phi \) vary depending on the state location, and careful tuning of \( \Phi \) in this spirit is extensively studied in [154, 155].

*Transition region:* The above control and adaptation laws are unique not only for the fact that the adaptive component is implemented by a parallel network of extremely simple basis functions, but also for the manner in which a seamless transition between adaptive and robust operation has been effected. It is important to note that this transition is *necessary* given the limitations on the approximating abilities of the adaptive controllers examined in this thesis; such a scheme may also prove useful in conventional adaptive applications as well. The width, \( \Psi \), of this transition in a sense also represents the width of a boundary layer, serving to smooth the transition between the two controller modes. Since, with a zero width transition region, there is a possibility the controller might chatter along the boundary of the set \( A_d \), the same comments which apply to \( \Phi \) above also apply to \( \Psi \): the transition region should be "thick" enough that there is no possibility that the transition between adaptive and sliding operation might be so sudden as to excite unmodeled dynamics.

*Biological Plausibility:* Although the intention of this chapter has been to derive stable adaptive controllers for nonlinear dynamic systems, intuitively the composite structure of the above control law is compatible with the observed movements of biological organisms. Large magnitude, gross control movements are employed when outside of accustomed regions of precision operation, for example when one momentarily loses
balance or swings an arm rapidly toward a pad of paper prior to writing a letter, while the finer, tuned movements occur in a much smaller subset of the degrees of freedom which describe the motion. It is thus possible that actual biological motion control structures also utilize a kind of blended control law.

6.4 Examples

In this section, the preceding theoretical development is applied to three example systems. Section 6.4.1 examines the performance of the proposed adaptive controller for an open-loop unstable, linear process model, while in Section 6.4.2, a controller is designed for the unity gain plant described by (6.8) where the unknown nonlinear function has a strictly bandlimited spectrum. In Section 6.4.3, a more realistic example is considered using a non-unity gain nonlinear process with more general forms of nonlinearities.

To gauge the effectiveness of the adaptive component of the control law, it is useful to compare the tracking performance of the closed loop system both with and without the output of the adaptive Gaussian network. Since use of the sliding controller is sufficient to keep the tracking errors bounded, even without the adaptive contribution, these simulations can be conducted without fear of instability.

In the simulations which follow, a two dimensional plant is required to track a set of desired trajectories which are specified as the output of a third order, critically damped system with a bandwidth of 10 rad/sec driven by a unity amplitude, 0.5 mean, 0.4 Hz. square wave. Figure 6-5 shows a phase space portrait of the desired position and velocity. Based upon this portrait, the set \( A_d \) was chosen to be the unit ball with respect to the weighted infinity norm

\[
\|x\|_{\infty,w} = \max\left(\frac{|x|}{0.75}, \frac{|\dot{x}|}{3.0}\right),
\]

centered at \( x_0 = [0.5, 0]^T \). This corresponds to the rectangular subset of the state space, \( A_d = [-0.25, 1.25] \times [-3.0, 3.0] \). A relatively thin transition region between
adaptive and sliding operation was chosen for these simulations, corresponding to a value of \( \Psi = 0.05 \) in (6.7), so that

\[
A = \{ x \mid \| x - x_0 \|_{\infty, u} \leq 1.05 \},
\]

so that \( A \approx [-0.3, 1.3] \times [3.2, 3.2] \).

As in the previous chapter, all simulations were performed using C running on a 25 MHz NeXTStation, and employed a fourth order, fixed stepsize Runge-Kutta algorithm with a time step of \( 10^{-4} \) sec.

### 6.4.1 Unity Control Gain: Linear Plant

In this first example, the process is the linear, time invariant plant given by

\[
\ddot{y}(t) - 2\dot{y}(t) + 3y(t) = u(t) + 1;
\]

note in particular that the natural dynamics of this process are unstable. Assuming that the output \( y(t) \) and the derivative of the output \( \dot{y}(t) \) can be measured, this process can be recast into the canonical form by taking \( x_1(t) = y(t) \), \( x_2(t) = \dot{y}(t) \), and \( f(x) = -3x_1 + 2x_2 + 1 \).

Since the function \( f \) is exactly the same as the function \( f_5 \) studied in the Gaussian network construction of Section 4.8.5, the network structure

\[
\mathcal{N}(x(t), \dot{c}(t)) = \sum_{k_1=-4}^{6} \sum_{k_2=-10}^{10} \hat{c}_{k_1,k_2}(t) \exp \left( -\pi \sigma^2 \left( (y(t) - k_1 \Delta)^2 + (u(t) - k_2 \Delta)^2 \right) \right)
\]

with \( \sigma^2 = 0.4\pi \) and \( \Delta = 0.5 \) satisfies the inequality \( |\mathcal{N}(x, c) - f(x)| \leq .002 \) for every \( x \) in the set \( A \) when the output weights \( c_\mathbf{k} \) are chosen as indicated in Section 4.8.5. By assumption, however, the process is unknown, and only the approximation abilities of the network relative to the function are assumed. The goal of the adaptation mechanism is hence to learn the required network output weights sufficiently well that the resulting control input forces the process to follow the model response.

Accordingly, the above network structure is used in the control law (6.11), together with the feedback gains \( k_D = \lambda = 10 \). A deadzone size \( \Phi = 0.001 \) is then clearly
sufficient to ensure \( k_D \Phi \geq \epsilon_f \), and is used in the adaptation law (6.12) together with an adaptation gain \( k_a = 200 \). Two simulations were conducted, one in which the complete control law (6.11), including the adaptive Gaussian network, was employed, and one in which only the PD and sliding components were used. Figure 6-6 compares the output tracking error, \( \hat{y}(t) \), and the total control signal issued to the plant during the first 5 seconds for both of these simulations. Use of the adaptive controller clearly results in superior tracking performance; worst case tracking error is reduced significantly compared with that obtained using PD and sliding components alone during this time period, and this ratio continually improves as the adaptive controller learns more about the plant dynamics. Figure 6-10 shows the tracking error using the adaptive controller during a twenty-five second period which does indeed demonstrate the predicted asymptotic convergence toward the deadzone.

To illustrate again the point about persistency of excitation, Figure 6-8 demonstrates that, while the tracking error is tending toward the deadzone, there is still a significant amount of mistuning in the network output weights. On the other hand, the errors in approximating the function \( f \) along the actual trajectory followed by the closed-loop system, is asymptotically becoming quite small, as Figure 6-8 shows.

### 6.4.2 Unity Control Gain: Nonlinear Plant

In this section the ideas of Section 6.1.2 are illustrated on a two dimensional plant with unity control gain; i.e. (6.8) with \( n = 2 \). The goal of the controller in this example is to obtain asymptotic position tracking accuracy to within \( 5 \times 10^{-4} \) units (any tighter bounds might be limited by, e.g., the resolution of the sensors in a physical application).

Taking \( k_D = \lambda = 10 \) and using (6.3), the desired asymptotic tracking accuracy requires the uniform approximation bound \( \epsilon_f \leq 0.05 \). The function \( F_{BL} \) is here assumed to be supported on a cube with \( \beta = 2 \), and to have a maximum value of \( F_{\max} = 10 \) which may occur arbitrarily near the corners of \( \mathcal{K}(\beta) \). With this value of
\( \beta \), the class membership error is assumed to be uniformly bounded on \( A \) by 0.005; use of (4.26) and (4.27) (with \( e \) replaced by \( e^2 \)), suggests that values of \( \theta = 2 \) and \( l = 4 \) will achieve the required bounds for the aliasing and truncation errors. From (4.24) and (4.25) these parameters indicate that a network of radial Gaussian nodes with variance \( \sigma^2 = 4\pi \), whose input weights encode a sampling grid with mesh size \( \Delta = 0.125 \), will be sufficient to achieve the control objectives, provided \( f \) actually belongs to the assumed function class.

Based upon definitions of the sets \( A_d \) and \( A \), together with the truncation radius \( \rho = 5\Delta \), the adaptive control network consists of the 1403 Gaussian nodes centered at the mesh points contained in the set \([-7\Delta, 15\Delta] \times [-30\Delta, 30\Delta]\). The complete control law is given by equations (6.11), together with the definitions of \( s(t) \), \( s_\Delta(t) \), and \( m(t) \). Assuming that the nonlinearity \( f \) is uniformly upper bounded on \( \mathcal{A}_d^\circ \) by 20, the gain \( k_d = 20 \) is used in the sliding controller. The network output weights are adjusted according to (6.12) with \( k_a = 750 \) and a deadzone size of \( \Phi = 0.005 \).

The true plant is modeled by (6.8) with \( n = 2 \) and:

\[
f(y, \dot{y}) = -4 \left( \frac{\sin(4\pi y)}{\pi y} \right) \left( \frac{\sin(\pi \dot{y})}{\pi \dot{y}} \right)^2
\]

which is indeed ideally bandlimited, its spectrum resembling a "roof" of length 4 and width 2 centered at the origin. All initial conditions were taken as 0, reflecting the fact that the only prior knowledge assumed about the plant are the number of states, and upper bounds on the absolute magnitude of the function \( f \), and on magnitude and support of its spatial Fourier transform.

Once again, two simulations were conducted, one in which the complete control law included the adaptive Gaussian network, and one in which only the PD and sliding components were used. Figure 6-9 compares the position tracking error, \( \dot{y}(t) \), and the total control signal issued to the plant during the first 5 seconds for both of these simulations. As before, the worst case tracking error is reduced significantly when using the adaptive components compared with that obtained using PD and sliding components alone during this time period. Figure 6-10 shows the tracking error using
the adaptive controller during a twenty-five second period which again demonstrates the predicted asymptotic convergence to the deadzone.

6.4.3 Non-Unity Control Gains

Again, a two dimensional plant is used, but now it is assumed that the control gains, \( b \), may be a function of the process output \( y \). The maximum desired acceleration is \( |\ddot{y}_m|_{max} \leq 25 \) for the desired trajectories pictured in Figure 6-5; together with the definition of the set \( A \) this implies that \( |a_r| \leq 100 \) for all \( x \in A \). Thus, the error bound \( \epsilon_r \) can be taken as \( \epsilon_r = \epsilon_h + 100 \epsilon_b \).

Separate networks are used to implement each of the required control components. The network approximating \( h \) is exactly the same as that used to approximate \( f \) in the previous example, while the network approximating \( b^{-1} \) has only the plant output \( y(t) \) as its input. The Gaussians in this second network are chosen to have a variance \( \sigma_b^2 = 0.4\pi \) and are centered on a regular grid \( \Delta_b = 0.5 \) which covers the set \([-6\Delta, 8\Delta] \).

To achieve asymptotic tracking accuracy of 0.003 units, using here the values \( k_D = 20 \), \( \lambda = 10 \) requires that \( \epsilon_r \leq 0.6 \). From the above expression for this quantity, this requires that the above networks be capable of approximating \( h \) and \( b^{-1} \) on \( A \) with a uniform error no worse than, for example, 0.05 and 0.0055 units respectively. Assuming this to be the case, the complete control law is given by equations (6.20), (6.21), and (6.23), using a value of \( \Phi = 0.03 \) as the adaptation deadzone and sliding controller boundary layer, together with adaptation gains \( k_{a1} = 300 \) and \( k_{a2} = 100 \). Assuming uniform upper bounds of \( M_0(x) = 20 \) and \( M_1(x) = 3 \), the sliding controller gains are taken as \( k_{sl} = 20 + 3|a_r| \); finally, the assumed uniform bound \( M_2(x) = 15 \) is used in (6.20).

The actual plant is given by (6.16) with

\[
 f(x, t) = -4 \left( 2 + \cos(4\pi x) \right)^{-1} \left( \frac{\sin(4\pi x)}{\pi x} \right) \left( \frac{\sin(\pi \dot{x})}{\pi \dot{x}} \right)^2
\]
CHAPTER 6. DIRECT ADAPTIVE CONTROL

and

\[ b(x) = (2 + \cos(4\pi x))^{-1} \]

so that \( h = b^{-1}f \) is precisely the process nonlinearity examined in the previous section, hence indeed \( \epsilon_h \leq 0.05 \) units for the chosen network architecture. Similarly, \( b^{-1} \) is precisely the function \( f_4 \) from the construction examples of Section 4.8.4, and as the network architecture used here is the same used in that construction, the bound \( \epsilon_b \leq 0.0055 \) as desired. The validity of the assumed bounds \( M_i(x) \) are also clear for these functions.

The same simulation parameters and initial conditions as in the previous section are chosen. As before, two simulations were conducted, one with and the other without the contribution of the adaptive network. Figure 6-11 shows both the position tracking error and control signals input to the plant during the first 5 seconds for both these simulations. Once again, it is evident that use of the adaptive controller provides superior performance; initial tracking errors are about two times better than those of the PD and sliding controller alone, with this ratio improving dramatically after the first few seconds. Figure 6-12 shows the tracking error for the adaptive controller over the first forty seconds of operation; again, it is evident that the errors have converged into the deadzone.

Figure 6-13 shows the evolution of the function estimation errors along the trajectory followed by the closed-loop system. Note the slower convergence of these estimates; the adaptive system is effectively using errors in one estimate to offset errors in the other so as to still meet the tracking specification. Unless the adaptive system is persistently excited, for example by commanding a sufficiently "interesting" set of trajectories, the controller need not "untangle" the different function estimation errors from the available error measurements in order to meet the tracking requirements.
6.5 Robots and Spacecraft

In this section, the above direct adaptive control designs are extended to a special class of multiple input and output nonlinear processes. As in the non-unity gain designs of Section 6.2.2, the special structure possessed by these dynamics will permit development of an algorithm which does not require explicit inversion of an estimate of the control gain, and hence avoids projection mechanisms in the adaptation algorithm. There is a price to be paid for this, however; the same simplification which permits the design to avoid projections and inversions makes it difficult to employ the deadzone robustness method used above, and hence to obtain precise asymptotic bounds on the magnitude of each tracking error state. Nonetheless, by instead using the sigma-modification robustness technique [73], and settling for asymptotic bounds on the total energy contained in the tracking errors, a very satisfactory direct adaptive control method can be developed.

An important class of nonlinear dynamic equations, which includes models of \( n \) joint robotic manipulators and rigid body rotations of spacecraft as special cases [159], can be written as

\[
H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = \tau
\]  

(6.27)

where, for example, \( q \in \mathbb{R}^n \) are the robotic joint angles or the spacecraft rotation parameters, \( H \in \mathbb{R}^{n \times n} \) is a symmetric, positive definite inertia matrix, \( C \in \mathbb{R}^{n \times n} \) is such that the vector \( C(q, \dot{q}) \dot{q} \) contains the Coriolis and centripetal torques, and \( G \in \mathbb{R}^n \) contains, for the robotic manipulator dynamics, any gravitational torques. The control input \( \tau \in \mathbb{R}^n \) represents the torques applied at each robotic joint, or a transformed version of the torques applied about the spacecraft body axes. In both of these physical systems, these dynamics have an important passivity property, which ensures that, by proper choice of \( C \) in the above parameterization,

\[
z^T(\dot{H} - 2C)z = 0
\]
for any \( z \in \mathbb{R}^n [156, 159, 158] \).

In order to develop a direct adaptive control law for these systems, similar to the classical algorithm developed in [158], but assuming here that the component functions of \( H, C \) and \( G \) are unknown, first generalize the error metric used in the previous sections to \( n \) dimensions, by defining

\[
(6.28) \quad s(t) = \dot{q}(t) + \Lambda \ddot{q}(t)
\]

where \( \Lambda = \Lambda^T > 0 \), \( \dot{q}(t) = q(t) - q^m(t) \), and \( q^m(t) \) is the trajectory the coordinates \( q \) are required to follow, assumed to be bounded, and at least twice continuously differentiable, with bounded first and second derivatives. It is also convenient to rewrite this expression as \( s(t) = \dot{q}(t) - \dot{q}^r(t) \) where

\[
(6.29) \quad \dot{q}^r(t) = \dot{q}^m(t) - \Lambda \ddot{q}(t).
\]

The complete state vector for the process is specified in terms of the coordinates \( q \) and their derivatives so that \( x^T = [q^T, \dot{q}^T] \in \mathbb{R}^{2n} \).

While the nonlinear functions which form the components of \( H, C \) and \( G \) are assumed unknown, for robots and spacecraft these functions are typically either polynomials, trigonometric terms, or combinations of such functions. Thus, the design procedures of Chapter 4 can be used to choose network construction parameters so that the components of these matrices can be approximated by radial Gaussian networks with a chosen accuracy on any compact subset of \( \mathbb{R}^n \) or \( \mathbb{R}^{2n} \). To understand how these network representations might be effectively used in a tracking control structure for systems whose dynamics can be written as \( (6.27) \), first define a set \( A_d \subset \mathbb{R}^{2n} \) containing the trajectories the system must follow, a set \( A \supset A_d \), and a modulation function \( m(t) \) as outlined above. The proposed adaptive control law can then be written as

\[
(6.30) \quad \tau(t) = -k_D s(t) + (1 - m(t)) \left[ \dot{\hat{r}}^H(t) + \dot{\hat{r}}^C(t) + \dot{\hat{r}}^G(t) \right] + m(t) \tau^{sl}(t)
\]
where the sliding component is \( \tau_i^s(t) = -k_i(x, t) \text{sgn}(s_i(t)) \), whose gains are chosen so that [159]

\[
k_i(x, t) \geq \|[H(q)\dot{q}(t) + C(x)\dot{q}(t) + G(q)]\|_i.
\]

The remaining torques in the control law are generated using the output of radial Gaussian networks, \( \mathcal{N}^H, \mathcal{N}^C, \) and \( \mathcal{N}^G \), designed to approximate the elements of \( H, C, \) and \( G \) respectively; thus

\[
\hat{\tau}_i^H(t) = \sum_{j=1}^{n} \mathcal{N}_{i,j}^H(q(t), \dot{c}^H(t)) \dot{q}_j^H(t)
\]

(6.31)

\[
\hat{\tau}_i^C(t) = \sum_{j=1}^{n} \mathcal{N}_{i,j}^C(x(t), \dot{c}^C(t)) \dot{q}_j^C(t)
\]

\[
\hat{\tau}_i^G(t) = \mathcal{N}_{i}^G(q(t), \dot{c}^G(t)).
\]

The Gaussian network \( \mathcal{N}^H \) used in this control law has \( n \) inputs, \( q_i \), and \( n^2 \) outputs, \( \mathcal{N}_{i,j}^H \), so that

(6.32)

\[
\mathcal{N}_{i,j}^H(q, \dot{c}^H) = \sum_{\text{dist}(A_{q}, k\Delta Hi) \leq \rho_H} \dot{c}_{i,j,k}^H \exp(-\pi \sigma_H^2 q - k\Delta H) ^2),
\]

where the set \( A \) has been decomposed as \( A = A_{q} \times A_{q} \), and for convenience the vector \( \dot{c}^H \in \mathbb{R}^{PH} \) contains each of the \( p_{H} = n^2 \mathcal{S}_H \) output weights, \( \dot{c}_{i,j,k}^H \). The network design parameters, \( \sigma_H^2 \), \( \Delta_H \), and \( \rho_H \) are chosen, based upon the function class to which the elements of \( H \) are assumed to belong, so that for a prespecified \( c_{i,j}^H \), there is a \( c^H \in \mathbb{R}^{PH} \) such that

\[
|\mathcal{N}_{i,j}^H(q, c^H) - H(q)_{i,j}| \leq c_{i,j}^H
\]

for any \( q \in A_{q} \), and any function \( H_{i,j} \) in the assumed class. As discussed at the end of Section 4.7, the same bounds which guide the selection of the design parameters also give bounds on the magnitudes of the required output weights, i.e. \( \|c^H\| \leq c_{max}^H \). Unlike the algorithms considered in the previous sections, the adaptation laws below will make explicit use of these bounds on the required output weights, instead of the bounds \( c_{i,j}^H \) on the network's uniform approximation capability.
The signals $N^C_{i,j}$ are the $n^2$ outputs of the Gaussian network $N^C$ which has the $2n$ inputs, $x_i$, so that
\begin{align}
N^C_{i,j}(x, c^C) &= \sum_{\text{dist}(A, k\Delta C) \leq \rho C} \hat{c}^C_{i,j,k} \exp(-\pi \sigma_C^2 \|x - k\Delta C\|^2).
\end{align}

and the network design parameters $\sigma_C^2$, $\Delta C$, and $\rho C$ are chosen based upon the assumed function classes of the elements of $C$ such that, given a tolerance $\epsilon_{i,j}^C$, there is a $c^C \in \mathbb{R}^{pc}$ with
\begin{align}
\left|N^C_{i,j}(x, c^C) - C(x)_{i,j}\right| &\leq \epsilon_{i,j}^C
\end{align}
for any $x \in A$. As above, the assumed prior information also provides an upper bound $c^C_{max}$ such that $\|c^C\| \leq c^C_{max}$.

Similarly, the terms $N^G_{i,j}$ represent the $n$ outputs of the Gaussian network $N^G$ which has $n$ inputs, $q_i$, so that
\begin{align}
N^G_{i,j}(q, c^G) &= \sum_{\text{dist}(A, k\Delta G) \leq \rho G} \hat{c}^G_{i,k} \exp(-\pi \sigma_G^2 \|q - k\Delta G\|^2).
\end{align}

Its design parameters are chosen based upon the assumed information about the elements of $G$ such that, given an $\epsilon_{i,j}^G$, there is a $c^G \in \mathbb{R}^{pg}$,
\begin{align}
\left|N^G_{i,j}(q, c^G) - G(q)_{i,j}\right| &\leq \epsilon_{i,j}^G
\end{align}
for any $q \in A$, with $\|c^G\| \leq c^G_{max}$; note that here the number of output weights is only $p_G = nS_G$.

Inspired by the robustness analysis in [135] and the designs of Section 6.2, chose as the output weight adaptation laws
\begin{align}
\hat{c}^H_{i,j,k}(t) &= -k^H_a (1 - m(t)) \left( \omega^H(t) \hat{c}^H_{i,j,k}(t) + s_i(t) \hat{q}^H_j(t) \exp(-\pi \sigma_H^2 \|q(t) - k\Delta H\|^2) \right),
\hat{c}^C_{i,j,k}(t) &= -k^C_a (1 - m(t)) \left( \omega^C(t) \hat{c}^C_{i,j,k}(t) + s_i(t) \hat{q}^C_j(t) \exp(-\pi \sigma_C^2 \|x(t) - k\Delta C\|^2) \right),
\hat{c}^G_{i,k}(t) &= -k^G_a (1 - m(t)) \left( \omega^G(t) \hat{c}^G_{i,k}(t) + s_i(t) \exp(-\pi \sigma_G^2 \|x(t) - k\Delta G\|^2) \right).
\end{align}
where

\[
\omega^H(t) = \begin{cases} 
0 & \text{if } \|\hat{c}^H(t)\| < c_0^H \\
\omega_0 \left( \frac{\|\hat{c}^H(t)\|}{c_0^H} - 1 \right) & \text{if } c_0^H \leq \|\hat{c}^H(t)\| \leq 2c_0^H \\
\omega_0 & \text{if } \|\hat{c}^H(t)\| > 2c_0^H 
\end{cases}
\]

(6.36)

for any \( c_0^H \geq c_{\max}^H \) and any positive constant \( \omega_0 \). Exactly similar expressions are used to compute \( \omega^C(t) \) and \( \omega^G(t) \), replacing superscripts as appropriate. Notice that the robustness in this adaptation law comes from introducing a decay mechanism into the weights whenever they begin to exceed the worst case bounds on their magnitudes.

To prove the stability of this simultaneous training and control strategy, consider the Lyapunov function candidate

\[
V = \frac{1}{2} s^T H s + \frac{1}{2k_a^H} \sum_{i,j=1}^{n} \sum_{\text{dist}(A_q, k\Delta H) \leq \rho_H} (\hat{c}_{i,j,k}^H)^2 \\
+ \frac{1}{2k_g^C} \sum_{i,j=1}^{n} \sum_{\text{dist}(A_q, k\Delta C) \leq \rho_C} (\hat{c}_{i,j,k}^C)^2 + \frac{1}{2k_g^G} \sum_{i=1}^{n} \sum_{\text{dist}(A_q, k\Delta G) \leq \rho_G} (\hat{c}_{i,k}^G)^2
\]

where, as usual, each \( \hat{c} \) is the mistuning between the network output weights, and the corresponding tuned weight, \( c \), which provides the uniform approximation on \( A \) given above. This function has a time derivative along the closed loop trajectory of the adaptive system (6.27), (6.30), (6.36), and (6.36) which can be written as

\[
\dot{V} = -k_D s^T s + \frac{1}{2} s^T (H - 2C) s + m s^T \left( \tau^l - H \ddot{q} - C \dot{q} - G \right) \\
+ (1 - m) \left( \sum_{i,j=1}^{n} s_i (N_{i,j}^H(q, \hat{c}^H) - H_{i,j}) \ddot{q}_j^c + \sum_{i,j=1}^{n} s_i (N_{i,j}^C(x, \hat{c}^C) - C_{i,j}) \dot{q}_j^c \\
+ \sum_{i=1}^{n} s_i (N_i^G(q, \hat{c}^G) - G_i) - \sum_{i,j=1}^{n} s_i N_{i,j}^H \ddot{q}_j^c - \sum_{i,j=1}^{n} s_i N_{i,j}^C \dot{q}_j^c - \sum_{i=1}^{n} s_i N_i^G \right) \\
- (1 - m) \left( \omega^H(\hat{c}^H)^T \dot{c}^H + \omega^C(\hat{c}^C)^T \dot{c}^C + \omega^G(\hat{c}^G)^T \dot{c}^G \right)
\]

where

\[
N_{i,j}^H = N_{i,j}^H(q, \hat{c}^H) - N_{i,j}^H(q, c^H) \\
= \sum_{\text{dist}(A_q, k\Delta H) \leq \rho_H} \hat{c}_{i,j,k}^H \exp(-\pi \sigma^2 \|x - k\Delta H\|^2)
\]
and similarly for $\mathcal{N}^C_i$, and $\mathcal{N}^G_i$.

Define as usual a vector, $d$, measuring the discrepancy between the best possible network approximations and the actual functions driving the system dynamics, i.e.,

$$d_i(x, t) = \sum_{j=1}^n \left( \mathcal{N}^H_{i,j}(q, c^H) - H_{i,j}(q) \right) \dot{q}_j^r(t) + \sum_{j=1}^n \left( \mathcal{N}^C_{i,j}(x, c^C) - C_{i,j}(x) \right) \dot{q}_j^r(t) + \left( \mathcal{N}^G_i(q, c^G) - G_i(q) \right),$$

so that

$$|d_i(x, t)| \leq \sum_{j=1}^n |\varepsilon_{i,j}^H| |\dot{q}_j^r(t)| + \sum_{j=1}^n |\varepsilon_{i,j}^C| |\dot{q}_j^s(t)| + \varepsilon_i^G$$

for any $x \in A$. Using the definitions of the sliding controller gains, and this definition of the disturbance, the above expression for $\dot{V}$ can be simplified as

$$\dot{V} \leq -\|s\|(k_D\|s\| - (1 - m)\|d\|)$$

$$\quad - (1 - m) \left( \omega^H(\dot{c}^H)^T \dot{c}^H + \omega^C(\dot{c}^C)^T \dot{c}^C + \omega^G(\dot{c}^G)^T \dot{c}^G \right).$$

Now, since $(1 - m)$ is 0 whenever $x \notin A$, the effects of the disturbance can be bounded as $(1 - m)\|d\| \leq d_\infty$, where

$$d_\infty^2 \triangleq \sup_{t} \sup_{x \in A} \sum_{i=1}^n |d_i(x, t)|^2.$$ 

Using the above bounds on each $d_i$, the definition (6.29) of $\dot{q}^r$, and recalling that the first and second derivatives of the model trajectories are bounded by assumption, it is clear that $d_\infty^2$ is finite, and can be made arbitrarily small by proper choice of network design parameters.

To address the effect of the last terms in (6.37), note that

$$-\omega^H(\dot{c}^H)^T \dot{c}^H \leq -\omega^H(\|\dot{c}^H\|^2 - (c^H)^T \dot{c}^H) \leq -\omega^H\|\dot{c}^H\|(\|\dot{c}^H\| - \|c^H\|)$$

since $\omega^H \geq 0$ by design. But $\omega^H > 0$ only if $\|\dot{c}^H\| > \|c^H\|$, and thus

$$-\omega^H(\dot{c}^H)^T \dot{c}^H \leq 0$$

for all $\dot{c}^H$
for all $t \geq 0$, with strict inequality whenever $\|\hat{c}^H(t)\| > c_0^H$. A similar argument holds for the terms of (6.37) multiplied by $\omega^C$ and $\omega^G$. As a result, one can now conclude that $\hat{V}$ is nonpositive on the set $\Omega^c$, where $\Omega$ is the bounded neighborhood of the origin defined by $\|s\| \leq d_\infty/k_D$, $\|\hat{c}^H\| \leq 2c_0^H$, $\|\hat{c}^C\| \leq 2c_0^C$, and $\|\hat{c}^G\| \leq 2c_0^G$. This shows $V(t)$, and hence $s(t)$ and each output weight mistuning, to be uniformly bounded [111]. Moreover, since by definition (6.28) of the error metric $s$, the tracking errors $\hat{q}$ can be seen as the output of a stable filter driven by $s$, provided that the initial conditions $\hat{q}(0)$ and $\dot{\hat{q}}(0)$ are bounded, $\hat{q}(t)$ and $\dot{\hat{q}}(t)$ remain bounded for all $t \geq 0$ [158]. Since by assumption the model trajectories and their derivatives are bounded, the actual process states are then also uniformly bounded.

To determine the convergence properties of this algorithm, rewrite the expression for $\hat{V}$ as

$$
\hat{V} \leq -k_D\|s\|^2 + (1 - m)\|s\|\|d\|
\leq (0.5 - k_D)\|s\|^2 + \frac{1 - m}{2}\|d\|^2,
$$

Integrating both sides, recalling that $V(t)$ is bounded, and assuming $k_D > 0.5$, reveals for any $T \geq 0$

$$
\int_0^T \|s(t)\|^2 dt \leq \kappa_1 + \frac{T\|d\|^2}{2k_D - 1}
$$

for some finite constant, $\kappa_1$. The convergence of $s$ can thus be specified as

$$
\lim_{T \to \infty} \frac{1}{T} \int_0^T \|s(t)\|^2 dt \leq \frac{\|d\|^2}{2k_D - 1}.
$$

Thus, unlike the precise asymptotic bounds on the magnitudes of each of the individual tracking errors obtained for the algorithms of previous sections, this analysis gives instead a bound on the asymptotic energy contained in the error metric $s$.

Note that the weight decay mechanism used in the above robust adaptation law could also be designed to trigger separately for each individual weight, instead of using a single trigger $\omega$ for all of the output weights of an entire network, by using assumed upper bounds on the individual magnitudes. The above stability and convergence
arguments would be left unchanged in this case, except for the definition of the set Ω which would then be defined using |\check{c}_{i,j,k}^H| \leq 2c_{0,i,j,k}^H, and similarly for the other network output weights.

6.6 Robotic Example

Before applying the tracking control algorithm developed in the previous section, note that the following important simplification can be made for robotic manipulators [37] by writing

\[ C(x)\dot{q}^r = C'(q)[\dot{q} \dot{q}^r] \]

where \( C' \in \mathbb{R}^{n \times n^2} \), and \([\dot{q} \dot{q}^r] \in \mathbb{R}^{n^2}\) is a vector containing all possible combinations \( \dot{q}_i \dot{q}^r_j \). A network, \( N^{C'} \), can now be used to approximate the components of the matrix \( C' \), replacing the network, \( N^C \) which approximates the components of the matrix \( C \).

This strategy requires a slight modification to the corresponding component of the control law, so that now

\[ \dot{\tau}_i^C(t) = \sum_{j=1}^{n^2} N_{i,j}^{C'}(q(t), \dot{q}^r(t))[\dot{q}(t) \dot{q}^r(t)]_j \]

where

\[ N_{i,j}^{C'}(q, \dot{q}^r) = \sum_{\text{dist}(A_{q,k\Delta C'}) \leq R_C} \check{c}_{i,j,k}^{C'} \exp(-\pi \sigma_{C'}^2 \|q - k\Delta C'\|^2), \]

and the adaptation mechanism for the kth output weight becomes

\[ \dot{c}_{i,j,k}(t) = -k_{a}(1 - m(t)) \left( \omega^{C'}(t) \check{c}_{i,j,k}^{C'}(t) + s_i(t) [\dot{q}(t) \dot{q}^r(t)]_j \exp(-\pi \sigma_{C'}^2 \|q(t) - k\Delta C'\|^2) \right). \]

The new network \( N^{C'} \) thus has the \( n \) inputs, \( q_i \), and \( n^3 \) outputs.

This modification allows a great reduction in the number of nodes needed in the network component of the control law, since the sampling mesh which defines the number and location of the centers of the Gaussian nodes needs only cover a subset of \( \mathbb{R}^n \), instead of \( \mathbb{R}^{2n} \) as in the previous design. For example, assuming that
\[ A = A_q \times A_q = [-\kappa, \kappa]^{2n}, \] 
and that \( \Delta_C = \Delta_{C'} < \kappa, \) then the number of nodes in \( \mathcal{N}^C \) is \( S_C = \kappa_1^{2n} \), for some \( \kappa_1 > 3 \), and the number of output weights in this network is then \( p_C = n^2 \kappa_1^{2n} \). On the other hand \( S_{C'} = \kappa_1^n \), and hence \( p_{C'} = n^3 \kappa_1^n \), so that the difference \( p_C - p_{C'} = n^2 \kappa_1^n (\kappa_1^n - n) > n^2 3^n (3^n - n) \) grows rapidly with dimension. Note that, since all networks now have only the joint angles, \( q_i \), as inputs, the notation \( A_q \) is redundant, and both \( A \) and \( A_d \) will be understood in what follows to be subsets of \( \mathbb{R}^n \).

To illustrate the algorithm, the process is chosen as a planar, two joint robotic manipulator, whose dynamics can be written in the form (6.27) with

\[
\begin{align*}
H_{1,1} &= a_1 + 2a_3 \cos(q_2) + 2a_4 \sin(q_2) \\
H_{1,2} &= H_{2,1} = a_2 + a_3 \cos(q_2) + a_4 \sin(q_2) \\
H_{2,2} &= a_2 \\
C_{1,1} &= -h \dot{q}_2 \\
C_{1,2} &= -h (\dot{q}_1 + \dot{q}_2) \\
C_{2,1} &= h \dot{q}_1 \\
C_{2,2} &= 0 \\
G_1 &= G_2 = 0
\end{align*}
\]

where \( h = a_3 \sin(q_2) - a_4 \cos(q_2) \) [159]. For this simulation the parameters \( a_1 = 3.3, \ a_2 = 0.97, \ a_3 = 1.04, \) and \( a_4 = 0.6 \) are used. Note that, given this structure, the matrix \( C' \in \mathbb{R}^{2 \times 4} \) can be written as

\[
C' = h \begin{bmatrix}
0 & -1 & -1 & -1 \\
1 & 0 & 0 & 0
\end{bmatrix}.
\]

The desired trajectories are chosen to be

\[
\begin{align*}
q_1^n(t) &= 1.33(1 - \cos(0.75\pi t)) \\
q_2^n(t) &= 0.75 \cos(2\pi t)
\end{align*}
\]
so that the set $A_d \subset \mathbb{R}^2$ can be taken as $A_d = [0, 2.66] \times [-.75, .75]$, or, for convenience, as the unit ball centered at $x_0^T = [1.5, 0]$ with respect to the weighted infinity norm using weights $w_1 = 1.5$, $w_2 = 1$. The modulation function is then computed as in the simulations of Section 6.4, using a transition region width of $\Psi = 0.1$.

For simplicity, the networks $\mathcal{N}^H$ and $\mathcal{N}^{C'}$ are merged to produce a single network with the two inputs $q_1$ and $q_2$ and the $2^2 + 2^3 = 12$ outputs needed to implement the approximations to the functions in $H(q_1, q_2)$ and $C'(q_1, q_2)$ (actually, note here that the true matrices $H$ and $C'$ are functions of $q_2$ only). The network is designed using the construction parameters $\Delta = 0.25$, $\sigma^2 = 2\pi$ and $\rho = 5\Delta = 1.25$ so that, given the above definition of the set $A$, this network has a total of 437 nodes and 5244 output weights which must be learned.

Since each entry of $H$ and $C'$ is either a sine, cosine, constant, or a sum of these, assuming that each component function has the general form $\kappa_1 + \kappa_2 \cos(2\pi n_1^T q) + \kappa_3 \sin(2\pi n_2^T q)$ with $\|n_i\| \leq 1$ and $|\kappa_i| \leq 5$, the choice of construction parameters and the analysis of Chapter 4 suggests that each of the required output weights is conservatively bounded in magnitude by $|c_{i,j,k}| \leq 15\sigma\Delta\kappa_G$ where

$$\kappa_G = \exp\left(\frac{\pi}{\sigma^2}\right) = \exp(0.5)$$

so that $|c_{i,j,k}| \leq 16$. This bound is used with the weight decay adaptation laws (6.36), (6.36), and (6.39), using a different $\omega$ for each output weight, as discussed at the end of Section 6.5. The weight decay parameter, $\omega_0$, is taken as $\omega_0 = 5$ and the adaptation gains are $k_a = 2$. As usual, the initial condition on each output weight is taken to be zero.

The gains of the sliding controller are taken so that

$$k_1^s(q_1, q_2, t) = k_2^s(q_1, q_2, t) = 25\|\dot{q}(t)\| + 30\|\ddot{q}(t)\|$$

using worst case bounds on the magnitude of the elements of $H$ and $C'$, and the error metric $s$ is computed using (6.28) with $\Lambda = 20I$. Finally, the gains $k_D = 100$ are used for the linear feedback components of the control law (6.30).
CHAPTER 6. DIRECT ADAPTIVE CONTROL

Figure 6-14 and Figure 6-15 compares the results of attempting to force the robot to follow the model trajectory, first without use of the adaptive Gaussian networks, i.e. using just the linear feedback and (if required) sliding components of the control law (6.30), then including the adaptive network contributions. Use of the networks improves the initial worst case tracking errors by a factor of three over the PD controller, and this ratio rapidly improves as the network learns more about the dynamics of the arm. Figure 6-16 plots the average energy of each component of the error metric $s$ during the simulation. As predicted above, these quantities are asymptotically converging to small values.
Figure 6-1: Illustration of the structure of the proposed adaptive controller.
Figure 6-2: Illustration of the different subsets of the plant state space involved in the design of the proposed adaptive controller.
Figure 6-3: Illustration of the structure of the adaptive component of the control law for a unity gain plant.
Figure 6-4: Illustration of the structure of the adaptive component of the control law for a non-unity gain plant.
Figure 6-5: Desired trajectories vs. time for simulations; note that $x_2 = \frac{dx}{dt}x_1$. Top: desired trajectories vs. time. Bottom: phase space portrait of desired trajectories.
Figure 6-6: PD and adaptive controller performance during the first 5.0 seconds, linear process example. Top: tracking errors. Bottom: control signals.
Figure 6-7: Adaptive controller tracking errors during the first 25 seconds of operation, linear process example.
Figure 6-8: Evolution of the approximation to the required linearizing control signal implemented by the Gaussian network. Top: total square mistuning between the output weight estimates and the values computed in Section 4.8.5. Bottom: Gaussian network estimate of the function $f(x) = -3x_1 + 2x_2 + 1$ evaluated along the closed-loop trajectory.
Figure 6-9: PD and adaptive controller performance during the first 5.0 seconds, unity gain example. Top: tracking errors. Bottom: control signals.
Figure 6-10: Adaptive controller tracking errors during the first 25 seconds of operation, unity gain example. The dashed lines indicate the predicted asymptotic bounds.
Figure 6-11: PD and adaptive controller performance during the first 5.0 seconds, non-unity gain example. Top: tracking errors. Bottom: control signals.
Figure 6-12: Adaptive controller tracking errors during the first 40 seconds of operation, non-unity gain example. The dashed lines indicate the predicted asymptotic bounds.
Figure 6-13: Evolution of the estimates implemented by the Gaussian networks to the functions $h$ and $1/b$ required for the linearizing control signal.
Figure 6-14: Comparison of robot tracking performance under PD and adaptive control laws.
Figure 6-15: Comparison of control signals used in the PD and adaptive tracking simulations for the two degree of freedom robot.
Figure 6-16: Convergence of the average energy in each component of the error metric $s$. 
Chapter 7

Enhanced Adaptive Techniques

7.1 Laterally inhibitive adaptation laws

The learning algorithms that have been developed in the previous chapters have been almost purely "Hebbian" in that the adjustment of the kth output weight at time \( t \) depends only upon the instantaneous product of the error measure and the output of the node centered at \( k\Delta \). However, the structure of the above Lyapunov proofs indicates immediately how lateral inhibition and excitation can be stably introduced to the adaptation mechanism, allowing the activity of other nodes in the network to influence the learning at node \( k \).

To see this it is necessary to use again the ordering imposed on the hidden layer outputs in Section 5.5 by a one-to-one mapping \( \gamma : \{1, \ldots, S\} \rightarrow \mathcal{K} \), where \( S \) is the total number of nodes in the network, and \( \mathcal{K} \) is the set of indices which define the centers of the hidden layer nodes. Again ordering the vector of hidden layer outputs, \( g(x) \) so that \( g_i(x) = \exp(-\pi \sigma^2 \|x - \gamma(i)\Delta\|^2) \), \( i = 1 \ldots S \), and imposing the same ordering on \( \dot{c} \), i.e. \( \dot{c}_i = \dot{c}\gamma(i) \), the adaptation laws for the output weights of the continuous time identification algorithms developed in Chapter 5 can be compactly written as, for example

\[
\dot{c}(t) = -k_a e_\Delta(t)I g(x(t)),
\]

where \( I \) is the \( S \times S \) identity matrix. A similar simplification holds for the tracking
control networks developed in Chapter 6, replacing $-\varepsilon(t)$ with $-(1 - m(t))s_\Delta(t)$ or $(1 - m(t))s_\Delta(t)\sigma_r(t)$ as appropriate.

Stable cross-coupling of the learning in the output weights can be introduced now by replacing the adaptive gain matrix $k_a I$ with any symmetric positive definite matrix $K_a$. That such a modification preserves the stability of the above continuous-time algorithms is easily seen by noting that, with this modification

$$\frac{1}{2} \frac{d}{dt} \dot{c}(t)^T K_a^{-1} \dot{c}(t) = -\varepsilon(t) \dot{c}(t)^T g(x(t))$$

$$= -\varepsilon(t) \sum_{\text{dist}(\Delta, k\Delta) \leq \rho} \dot{c}_{k}(t) \exp(-\pi \sigma^2 \|x - k\Delta\|^2)$$

so that by changing the sum squared mistuning terms in the Lyapunov functions used in previous chapters to be the more general quadratic form $\frac{1}{2} \dot{c}^T K_a^{-1} \dot{c}$, and using the new adaptation law, the resulting time derivatives are identical to those obtained previously and thus the above proofs are left unchanged.

In the discrete-time algorithms of Section 5.2, by replacing the sum-squared mistuning in the Lyapunov function with the more general form $\dot{c}[t]^T K_a^{-1} \dot{c}[t]$, where again $K_a = K_a^T > 0$, using the new adaptation law

$$(7.2) \quad \dot{c}[t] = \dot{c}[t - 1] - \varepsilon(t) K_a g(x[t])$$

and setting $\Delta V[t] = V[t] - V[t - 1]$, one obtains

$$\Delta V[t] = \dot{c}[t]^T K_a^{-1} \dot{c}[t] - \dot{c}[t - 1]^T K_a^{-1} \dot{c}[t - 1]$$

$$= 2\dot{c}[t]^T K_a^{-1} (\dot{c}[t] - \dot{c}[t - 1]) + \varepsilon(t)^2 g(x[t])^T K_a g(x[t]) - 2\varepsilon(t) \dot{c}[t]^T g(x[t])$$

$$= -2\varepsilon(t) \dot{c}[t] - 1^T g(x[t]) + \varepsilon(t)^2 g(x[t])^T K_a g(x[t])$$

$$\leq -\varepsilon(t)^2 (2 - g(x[t])^T K_a g(x[t])).$$

assuming as above the deadzone is chosen so that $\Phi \geq \varepsilon_f$. Thus, if the adaptation gain matrix is chosen to satisfy

$$(7.3) \quad 0 < \delta = 2 \sup_{x \in A} g(x)^T K_a g(x)$$
then $V[t] - V[t-1]$ is non-positive, and the stability and convergence properties of the prediction algorithm are unchanged using the new adaptation law.

Notice that there is quite a bit of freedom in determining these new adaptive laws. The mapping $\gamma$ can be chosen to impose any desired ordering on the hidden layer nodes, and the matrix $K_a$ allows specification of the extent to which neighboring nodes (measured with respect to the $\gamma$ ordering) are inhibitive or excitative.

To illustrate, Figure 7-1 shows the results of using the new adaptation law (7.2) to predict the chaotic time series examined in Section 5.4.2. The prediction network is the same as in the earlier simulation, and the mapping $\gamma: \{1, \ldots, 17\} \rightarrow \{-8, \ldots, 8\}$ is chosen to impose the “natural” ordering on the output weights, i.e. $\gamma(i) = i - 9$. The gain matrix $K_a$ is taken as band diagonal, so that $(K_a)_{i,j} = (-1)^{|i-j|}2^{-|i-j|}$ for $|i - j| \leq 2$, and $(K_a)_{i,j} = 0$ otherwise. This is a symmetric, positive definite matrix, and satisfies the constraint (7.3) with the construction parameters $\sigma^2 = 2\pi$ and $\Delta = 1/4$ used in the network design of Section 5.4.2. The adaptive law for each output weight can thus be written explicitly as

$$\hat{c}_i[t] = \hat{c}_i[t-1] - \epsilon_{\Delta}[t] \sum_{\substack{1 \leq j \leq 17 \\left|\frac{i-j}{2}\right| \leq 2}} (-1)^{|i-j|}2^{-|i-j|} \exp(-\pi \sigma^2(y[t] - \gamma(j)\Delta)^2).$$

Adjacent nodes in this adaptation law are thus inhibitive, while nodes one step removed are excitative. As seen from the figure, the convergence properties of the new algorithm are almost identical with that used in Section 5.4.2.

## 7.2 Dynamic Network Structure

Intuitively, those Gaussian nodes in the network which are centered sufficiently far (relative to their variance) from the current input vector will contribute minimally to the approximation $\mathcal{N}(x)$. In fact, the truncation error bounds given in Section 4.7, hold at any point $x \in \mathbb{R}^n$, while the network itself is constructed by ensuring that these bounds are satisfied for all points in the compact set $A$. Thus, for any particular
Figure 7-1: Adaptive Gaussian network predictions of the output of the chaotic time series of Section 5.4.2, using an adaptation law with lateral inhibition and excitation. Top: bandwidth estimates and prediction errors. Bottom: close up view of the prediction errors in the last 10000 time steps of the simulation.
input, one can completely neglect the contribution of nodes in the network centered further than $\rho_1 = (l + 1) \Delta$ from this point while still maintaining the uniform bound $\epsilon_f$. Hence, at any instant only a small subset of the hidden layer outputs need to be computed, and only output weights associated with these "active" nodes need to be adjusted. Distance, here and for the rest of the discussion in this section, is measured using the infinity norm on $\mathbb{R}^n$, in accordance with the truncation arguments in Section 4.7.

Further, if the evolution of the plant state vector is such that, up to a given time $T$, it has not passed closer than $\rho_1$ to the center of a particular node then, adopting the above truncation strategy, this node did not need to "exist" on the time interval $[0, T]$. Similarly, nodes whose output weights were zero up to time $T$ also did not contribute to the control law and thus did not need to exist. Since the output weights will change only when $s_\Delta$ or $\epsilon_\Delta$ is nonzero then, by considering all output weights to be initialized to zero, each individual node in the network can be brought into existence only if the plant state passes within $\rho_1$ of its center during times when $s_\Delta$ or $\epsilon_\Delta$ is nonzero. The sampling mesh thus defines a lattice of latent node positions, and nodes can be created only as they are required. The analysis of the previous chapters hence naturally specifies a mechanism for stable, on-line construction of the network in response to a particular sequence of inputs.

This strategy offers great advantages in a recursive identification application, as it allows the structure of the network to be driven by the actual behavior the process exhibits instead of worst case bounds. In an adaptive control application, this online construction can also be pursued as long as the plant state remains within the set $A$ as designed in Chapter 6. If the state should ever leave this set, construction and adaptation should be stopped as described in the previous chapter, and the sliding controller should again be used to return the state to $A$, whereupon network construction and output weight adaptation can recommence.

In fact, each of the results shown in Chapter 5 and Chapter 6 used the above trun-
CHAPTER 7. ENHANCED ADAPTIVE TECHNIQUES

cation ideas to speed execution of the simulations. The chaotic prediction example of Section 5.4.2 and the nonunity gain tracking control simulation of Section 6.4.2 were also conducted using the output of every node in fixed-size networks to check the accuracy of the truncation argument: no measurable difference was detected between these larger simulations and those reported in the previous chapters.

7.3 Moving Centers and Changing Variances

The construction algorithm developed in Chapter 4 requires a prior estimate of smoothness, measured in terms of the effective spatial bandwidth of the function to be approximated. This estimate is then used to define a fixed mesh on which the nodes are centered, and a fixed variance for each node. Conservative estimates of the required bandwidth can therefore lead to designs with a possibly unnecessarily large number of nodes, a problem which worsens as the dimension, n, increases.

To attempt to alleviate this problem, note that it is certainly possible, for a particular bandlimited function, that the required approximation accuracy could be obtained by using only a few, sparsely distributed nodes. In this sense the constructive algorithms developed in Chapter 4 are “worst case” designs: they guarantee the required accuracy for any member of the chosen function class. An algorithm capable of also adjusting the centers and variances might thus be capable (using truncation arguments similar to the above) of finding a more parsimonious network representation of the actual function underlying the observed data. Ideally, the stable, on-line algorithms discussed above for learning the output weights should be extended to the other parameters in the Gaussian expansion.

Developing a constructive theory of such scattered-center RBF networks, to even ascertain the existence of such representations, is a formidable task, although the analysis in Chapter 8 represents a promising first step. More importantly, however, as discussed in Chapter 5, stable on-line adjustment of network parameters other
Figure 7-2: Illustration of the on-line construction algorithm: a node is brought into existence only if the input passes within the truncation distance of the corresponding lattice point, which are represented by circles in the figure. In this example, the truncation distance is 1.25 times the lattice mesh size, and the arrowed line represents the evolution in time of the two inputs to the network. The dark circles correspond to nodes which are currently active to compute the approximation at the current input (represented by the head of the arrow), while the lighter grey circles correspond to those which have come into existence as a result of the past inputs. The unfilled circles represent “latent” lattice points: no node centered at these points currently exists in the network, but will come into existence if the future evolution of the input passes within 1.25 units.
than the output weights is extremely difficult. There is not yet a fully developed theory for how to accommodate the nonlinear impact changes in these parameters have on the overall approximation. On the other hand, using the gradient descent algorithms developed above as a first approximation to these nonlinearities results in the potentially large disturbing term $d_g$ which must then be accommodated in the robust adaptation mechanism, possibly resulting in large steady-state tracking or prediction errors.

A satisfactory mechanism for stable, independent adjustment of the centers and variances thus awaits further developments in the field. However, as will be shown below, by coupling the adjustments together it is possible to make the approximation error $\mathcal{N} - f$ instead depend linearly on a measure of the parameter mistuning, allowing direct application of the gradient algorithms used above. In fact, the analysis in Section 4.7 immediately suggests a natural method for coupling the centers and variances together: if the centers are constrained to lie on a mesh of size $\Delta$, and all nodes are constrained to have the same variance, then the equations (4.24) and (4.25) couple the choices of centers and variances together through the assumed bandwidth, $\beta$, of the function being approximated.

Thus, assume that at a given time only an estimate, $\hat{\beta}(t)$, of the required spatial bandwidth is available, and that the variance and mesh size of the network are selected using (4.24) and (4.25), with the estimated value $\hat{\beta}(t)$. Intuitively, this corresponds to continually tuning the input weights and variances of the network to ensure that it has the capacity to reproduce functions bandlimited to $\hat{\beta}(t)$ with a uniform accuracy of $\epsilon_f$ on $A$. More precisely, it is a certainty equivalence network construction using the estimated parameter $\hat{\beta}(t)$ in the design procedure detailed in Section 4.7. As the estimate $\hat{\beta}(t)$ increases, the mesh contracts, the centers move closer together, and variances increase; similarly as $\hat{\beta}(t)$ decreases, the centers move further apart and the variances decrease. With an eye to Figure 4-2 and Figure 7-3, this corresponds to varying the essential width of the Gaussian low-pass filter, and adjusting the sample
CHAPTER 7. ENHANCED ADAPTIVE TECHNIQUES

points accordingly. The following sections explore this idea rigorously, first explicitly expanding the approximation error in terms of both the bandwidth and the output weight mistuning, then using this expansion to develop stable laws for adjusting these parameters.

To indicate the added dependence of the structure of the Gaussian network on the estimated bandwidth, the network output will now be written as \( N(x, \hat{c}, \hat{\beta}) \), so that

\[
N(x, \hat{c}, \hat{\beta}) = \sum_{\text{dist}(k\hat{\Delta}, A) \leq \hat{\Delta}} \hat{c}_k \exp\left(-\pi \hat{\sigma}^2 \|x - k\hat{\Delta}\|^2\right)
\]

where

(7.4)

\[
\hat{\sigma}^2 = \pi \hat{\beta}^2 \quad \text{and} \quad \hat{\Delta} = \frac{1}{2\hat{\beta}^2},
\]

for some fixed oversampling ratio \( \theta \) and truncation parameter \( l \). The analysis below also assumes that the set \( A \) is a rectangle, containing the origin and aligned with the coordinate axes of \( \mathbb{R}^n \). This is to ensure that as the mesh size monotonically decreases, lattice points remain within \( A \) once they have entered this set.

7.3.1 Effects of bandwidth mistuning

With the certainty equivalence design strategy, the approximation error can be expanded in terms of the mistuning in the single bandwidth parameter, \( \hat{\beta}(t) \), instead of the mistuning in the individual centers and variances. To begin this analysis, define a frequency truncation operator, \( T \), such that, for an integrable function, \( f \),

\[
T(f, \hat{\beta})(x) = \mathcal{F}^{-1}(F_{\lambda \hat{\beta}})(x),
\]

and hence \( T(f, \hat{\beta}) \) is the function obtained by passing \( f \) through an ideal spatial low-pass filter with a cut-off of \( \hat{\beta} \). If \( f \) is bandlimited, and the support of \( F \) is contained in \( \mathcal{K}(\hat{\beta}) \), then

\[
|f(x) - T(f, \hat{\beta})(x)| = |T(f, \beta)(x) - T(f, \hat{\beta})(x)|
\]

\[
\leq \int_{\mathcal{K}(\beta) - \mathcal{K}(\hat{\beta})} |F(\nu)|d\nu \leq F_{\max}(\beta - \hat{\beta})p(\beta, \hat{\beta}),
\]
uniformly on $\mathbb{R}^n$, where
\[ p(\beta, \hat{\beta}) = \begin{cases} 
2^n \sum_{i=0}^{n-1} \beta^i \hat{\beta}^{n-i-1} & \text{if } \beta > \hat{\beta} \\
0 & \text{otherwise,}
\end{cases} \]
and $F_{max}$ is an upper bound on the magnitude of the spectrum of $f$. Given a function, $f$, a set, $A$, and a tolerance $\epsilon_c$, define
\[ \beta^* = \inf_{\beta \geq 0} \{ \beta \mid |f(x) - T(f, \beta)(x)| \leq \epsilon_c \ \forall x \in A \}, \]
that is, $T(f, \beta^*)$ is the bandlimited approximation to $f$ with smallest spectral support capable of ensuring the required class membership accuracy everywhere on $A$. Usually, $\beta^*$ is unknown, and only a crude upper bound, $\beta \geq \beta^*$, is known which makes this inequality true. This bound would ordinarily be used in the conservative designs of Chapter 4; in this section it will be used only as an upper limit for estimates $\hat{\beta}(t)$, and the algorithms developed below will attempt to find a bandwidth significantly smaller that $\beta$ which still reproduce the function to the required accuracy.

As in previous chapters, to develop a stable adaptation mechanism for $\hat{\beta}(t)$ it is necessary to show that
\[ d(x) = N(x, \hat{c}, \hat{\beta}) - f(x) \]
can be expanded linearly in $\hat{\beta} - \beta^*$, without destroying the linear parameterization in the $\hat{c}_k$. Equivalently, this expansion can be in terms of $\hat{\beta} - \beta$, since the analysis of Chapter 4 shows that oversampling will only increase the accuracy of the Gaussian approximation to bandlimited functions.

Expanding $d$ in terms of the bandwidth parameter yields
\[ d = (T(f, \beta) - f) + (T(f, \hat{\beta}) - T(f, \beta)) + N(\cdot, \hat{c}, \hat{\beta}) - T(f, \hat{\beta}). \]
Since $T(f, \hat{\beta})$ is bandlimited, the analysis in Chapter 4 gives
\[ T(f, \hat{\beta}) = N(x, c^{\hat{\beta}}, \hat{\beta}) - d_a(\hat{\beta}) - d_t(x, \hat{\beta}) \]
where $d_a(\hat{\beta})$ is the aliasing error incurred approximating a function bandlimited to $\hat{\beta}$ with the specified mesh and variance using a Gaussian approximation to the low pass
filter, and similarly \( d_t(x, \hat{\beta}) \) is the truncation error at point \( x \in \mathbb{R}^n \) incurred truncating this Gaussian expansion using only the nodes in the set \( \{ k \mid \| x - k\hat{\Delta} \|_{\infty} \leq \hat{\Delta} \} \). The required output weights in this expansion are given by \( c_k^{\beta} = c^{\beta}(k\hat{\Delta}) \) where

\[
c^{\beta}(x) = \left( \frac{\sqrt{\pi}}{2\theta} \right)^n \mathcal{F}^{-1}(FG^{-1})(x).
\]

Thus combining the above results

\[
d = (T(f, \beta) - f) + d_a(\hat{\beta}) + d_t(\cdot, \hat{\beta}) + (T(f, \hat{\beta}) - T(f, \beta)) + \mathcal{N}(\cdot, \hat{\beta}, \hat{\beta}) - \mathcal{N}(\cdot, c^{\beta}, \hat{\beta})
\]

\[
= (T(f, \beta) - f) + d_a(\hat{\beta}) + d_t(\cdot, \hat{\beta}) + (T(f, \hat{\beta}) - T(f, \beta))
\]

\[
+ (\mathcal{N}(\cdot, \hat{\beta}, \hat{\beta}) - \mathcal{N}(\cdot, c^{\beta}, \hat{\beta})) + (\mathcal{N}(\cdot, c^{\beta}, \hat{\beta}) - \mathcal{N}(\cdot, c^{\beta}, \hat{\beta}))
\]

where now \( c_k^{\beta} = c^{\beta}(k\Delta) \), with

\[
c_k^{\beta}(x) = \left( \frac{\sqrt{\pi}}{2\theta} \right)^n \mathcal{F}^{-1}(FG^{-1})(x),
\]

and

\[
\sigma^2 = \pi\beta^2, \quad \Delta = \frac{1}{2\theta\beta}.
\]

By defining the output weight mistuning as \( \hat{c}_k = \hat{c}_k - c_k^{\beta} \) the expression for \( d \) becomes

\[
d = (T(f, \beta) - f) + d_a(\hat{\beta}) + d_t(\cdot, \hat{\beta}) + \sum_{\text{dist}(A,k\Delta)\leq \hat{\Delta}} \hat{c}_k \exp(-\pi\sigma^2\|x - k\hat{\Delta}\|^2)
\]

\[
+ (T(f, \hat{\beta}) - T(f, \beta)) + (\mathcal{N}(\cdot, c^{\beta}, \hat{\beta}) - \mathcal{N}(\cdot, c^{\beta}, \hat{\beta})).
\]

The first term in this expansion is simply the class membership error \( d_c \), and thus the first four terms resemble the expansion of \( \mathcal{N} - f \) used in the previous chapters; the second to last term has a dependence on \( \beta - \hat{\beta} \) which is quantified by (7.5), what remains is to similarly quantify the bandwidth dependence of the last term in parentheses. Noticing that \( c_k^{\beta} - c_k^{\Delta} = c(\Delta k) - c^{\beta}(k\Delta) \), that \( c^{\beta} = T(c^{\beta}, \beta) \), and that

\[
T(c^{\beta}, \beta)(\Delta k) = T(c^{\beta}, \hat{\beta})(\hat{\Delta} k) + (\Delta - \hat{\Delta}) \frac{\partial T(c^{\beta}, \beta)}{\partial x}(\zeta_k) k
\]
for some \( \zeta_k \in \{ k \Delta + (1 - \alpha)\Delta, \ 0 \leq \alpha \leq 1 \} \), the last term in the expansion of \( d \) can be written as

\[
(\text{7.6}) \quad \mathcal{N}(\cdot, c^\beta, \hat{\beta}) - \mathcal{N}(\cdot, c^{\hat{\beta}}, \hat{\beta}) = \sum_{\text{dist}(A,k\Delta) \leq l\Delta} (T(c^\beta, \hat{\beta})(k\Delta) - T(c^{\hat{\beta}}, \hat{\beta})(k\Delta)) \exp(-\pi\hat{\sigma}^2\|x - k\hat{\Delta}\|^2)
\]

\[
+ (\Delta - \hat{\Delta}) \sum_{\text{dist}(A,k\Delta) \leq l\Delta} \frac{\partial T(c^{\hat{\beta}}, \hat{\beta})}{\partial x}(\zeta_k) \ k \exp(-\pi\hat{\sigma}^2\|x - k\hat{\Delta}\|^2).
\]

For clarity, express now

\[
d(x) = \sum_{\text{dist}(A,k\Delta) \leq l\Delta} c_k \exp(-\pi\hat{\sigma}^2\|x - k\hat{\Delta}\|^2) + d_f(x, \hat{\beta}) + d_\beta(x, \hat{\beta}),
\]

where

\[
d_f(x, \hat{\beta}) = T(f, \hat{\beta})(x) - f(x) + d_a(\hat{\beta}) + d_t(x, \hat{\beta})
\]

and \( d_\beta(\hat{\beta}) \) is the sum of (7.6) and \((T(f, \hat{\beta}) - T(f, \beta))\). Assuming the parameters \( \theta \) and \( l \) are chosen so that, for any \( \hat{\beta} \in [\kappa_\beta, \beta] \), for some small \( \kappa_\beta > 0 \), the aliasing and truncation errors of a Gaussian network approximation to a function bandlimited to \( \mathcal{K}(\hat{\beta}) \) are bounded on \( A \) by \( \epsilon_a \) and \( \epsilon_t \) respectively, then \( |d_f| \) is bounded by \( \epsilon_f = \epsilon_c + \epsilon_a + \epsilon_t \) for all \( x \in A \) and all \( \hat{\beta} \in [\kappa_\beta, \beta] \). Similarly, from the above expansions, for any bandwidth in this range, and any \( x \in A \), \( |d_\beta| \leq (\beta - \hat{\beta})\psi(\hat{\beta}) \) where

\[
\psi(\hat{\beta}) = F_{\max} p(\beta^*, \hat{\beta}) + \sup_{x \in A} \sum_{\text{dist}(A,k\Delta) \leq l\Delta} \left[ C_{\max} p(\beta^*, \hat{\beta}) + \frac{\kappa_B(\hat{\beta})}{\beta}\|k\hat{\Delta}\| \right] \exp(-\pi\hat{\sigma}^2\|x - k\hat{\Delta}\|^2),
\]

and \( C_{\max} = e^n F_{\max} \Delta^n \). The positive constants in this expression, \( \kappa_B(\hat{\beta}) \), which uniformly bound the total magnitude of the vector of partial derivatives appearing in (7.6), by Bernstein's theorem [8] are bounded for any bounded \( \hat{\beta} \), and the dependence of \( \hat{\sigma}^2 \) and \( \hat{\Delta} \) on the estimated bandwidth is through (7.4) above.
7.3.2 Stable bandwidth adaptation

From the analysis of the above section, for any given choice of bandwidth at time $t$, $\hat{\beta}(t)$, the function approximation error at any point $x \in \mathbb{R}^n$ may be expanded as

\[
\mathcal{N}(x, \hat{c}(t), \hat{\beta}(t)) - f(x) = d_f(x, \hat{\beta}(t)) + d_\beta(x, \hat{\beta}(t)) + \sum_{\text{dist}(A, k\hat{\Delta}(t)) \leq l\hat{\Delta}(t)} \hat{c}_k(t) \exp(-\pi \hat{\sigma}^2(t))\|x - k\hat{\Delta}(t)\|^2.
\]

(7.7)

This section will use this characterization of the error to develop an adaptation mechanism for $\hat{\beta}(t)$, and to modify slightly the adaptation mechanisms for the output weights, so as to ensure stability and convergence of identification and control algorithms which employ these networks.

Consider first a continuous time identification algorithm. For both the linearly parameterized and “canonical” SPR error equations, i.e. $\varepsilon(t) = \hat{c}(t)^T g(x(t))$ and $\dot{\varepsilon}(t) = -k_D \varepsilon(t) + \hat{c}(t)^T g(x(t))$ respectively, the only additional term to accommodate in the analysis is the bandwidth mistuning, $d_\beta$, which then adds the term $\varepsilon_\Delta d_\beta$ to the time derivative of the quadratic Lyapunov functions used in the previous chapters.

This inspires the choice of a new Lyapunov function

\[
V(t) = \frac{1}{2} \varepsilon_\Delta(t)^2 + \frac{1}{2k_a} \sum_{\text{dist}(A, k\hat{\Delta}) \leq l\hat{\Delta}} \hat{c}_k(t)^2 + \frac{K_\beta}{2k_b} (\hat{\beta}(t) - \beta)^2,
\]

for the SPR case, or simply

\[
V(t) = \frac{1}{2k_a} \sum_{\text{dist}(A, k\hat{\Delta}) \leq l\hat{\Delta}} \hat{c}_k(t)^2 + \frac{K_\beta}{2k_b} (\hat{\beta}(t) - \beta)^2,
\]

for the linear error equation. Here, $K_\beta$ is a finite, positive constant which will be determined below.

In each case, if the output weights for the network are adjusted using

(7.8)

\[
\dot{\hat{c}}_k(t) = \begin{cases} 
- k_a \varepsilon_\Delta(t) \exp(-\pi \hat{\sigma}^2(t))\|x(t) - k\hat{\Delta}(t)\|^2 & \text{if dist}(A, k\hat{\Delta}(t)) \leq l\hat{\Delta}(t) \\
0 & \text{otherwise},
\end{cases}
\]
and the bandwidth estimate is updated as

\[
\hat{\beta}(t) = \begin{cases} 
k_b |e_\Delta(t)| & \text{if } \kappa_\beta \leq \hat{\beta}(t) < \beta \\
0 & \text{otherwise,}
\end{cases}
\]

for any \(k_a, k_b > 0\), with \(\hat{\beta}(0) \in [\kappa_\beta, \beta]\), for some small \(\kappa_\beta > 0\), and if the deadzone is chosen as \(\Phi \geq \epsilon_f/k_D\) for the SPR case, or \(\Phi \geq \epsilon_f\) in the linear case, the error measure \(e\) will asymptotically converge to the deadzone.

To prove this, first note that, since \(A\) is assumed to be a rectangle as specified above, and since the bandwidth adaptation mechanism ensures that \(\Delta \leq \hat{\Delta}(t)\) for all \(t \geq 0\), the inclusion relation

\[\{k \mid \text{dist}(A, k\hat{\Delta}) \leq l\hat{\Delta}\} \subset \{k \mid \text{dist}(A, k\Delta) \leq l\Delta\}\]

holds, so that

\[\frac{1}{k_a} \sum_{\text{dist}(A, k\Delta) \leq l\Delta} \hat{c}_k(t) \hat{c}_k = -e_\Delta(t) \sum_{\text{dist}(A, k\Delta) \leq l\Delta} \hat{c}_k(t) \exp(-\pi \hat{\sigma}^2(t) \|x(t) - k\hat{\Delta}(t)\|^2)\]

and thus using (7.7), (7.8), (7.9), and the two error models above, the time derivative of both these \(\dot{V}(t)\) can be written as

\[\dot{V}(t) \leq -\delta e_\Delta(t)^2 + |e_\Delta(t)| (\beta - \hat{\beta}(t)) (\psi(\hat{\beta}(t)) - K_\beta)\]

where \(\delta = k_D\) in the SPR case, and \(\delta = 1\) in the linear case. By choosing the arbitrary constant \(K_\beta\) such that

\[K_\beta = \sup_{\kappa_\beta \leq \alpha \leq \beta} \psi(\alpha)\]

\(\dot{V}(t)\) is clearly nonpositive for all \(t \geq 0\), thus ensuring that the estimation error and parameter estimates are uniformly bounded. An argument based on Barbalat's lemma, entirely similar to that given in Section 5.3, then establishes the convergence of the error to the deadzone.

Of course, this same argument can also be applied to each of the individual \(\epsilon_j\) used in Section 5.3. To extend this algorithm to the control algorithms developed in
Chapter 6, notice that the error measure, \( s(t) \), used in that chapter obeys exactly the SPR equation used above, and only the modulation needs to be incorporated into the above adaptation laws. Thus, for example, changing \( \epsilon_{\Delta}(t) \) to \( (1 - m(t))s_{\Delta}(t) \) in (7.8) and (7.9) one obtains the new adaptation mechanisms for the algorithm of Section 6.2.1.

For the discrete time identification algorithm of Section 5.2, the prediction error can be expanded as

\[
\epsilon[t] = \mathcal{N}(x[t], \hat{c}[t - 1], \hat{\beta}[t - 1]) - f(x[t]) \\
= d_f(x[t], \hat{\beta}[t - 1]) + d_\beta(x[t], \hat{\beta}[t - 1]) \\
+ \sum_{\text{dist}(A, k\hat{\Delta}[t-1]) \leq \hat{\Delta}[t-1]} \hat{c}_k[t - 1] \exp(-\pi \sigma^2[t - 1] \|x[t] - k\hat{\Delta}[t - 1]\|^2).
\]

which, together with the above bound on \( d_\beta \) and the analysis of Section 5.2, suggests a new Lyapunov function of the form

\[
V[t] = \sum_{\text{dist}(A, k\hat{\Delta}) \leq \hat{\Delta}} \hat{c}_k[t]^2 + K_1 (\hat{\beta}[t] - \beta)^2,
\]

where \( K_1 \) is a finite, positive constant to be determined below. Taking as the adaptive laws

\[
(7.10) \quad \hat{c}_k[t] = \begin{cases} 
\hat{c}_k[t - 1] - k_a e_{\Delta}[t] \exp(-\pi \sigma^2[t - 1] \|x[t] - k\hat{\Delta}[t - 1]\|^2) & \text{if dist}(A, k\hat{\Delta}[t - 1]) \leq \hat{\Delta}[t - 1] \\
0 & \text{otherwise},
\end{cases}
\]

and

\[
(7.11) \quad \hat{\beta}[t] = \begin{cases} 
\hat{\beta}[t - 1] + k_b |e_{\Delta}[t]| & \text{if } \hat{\beta}[t - 1] + k_b |e_{\Delta}[t]| < \beta \\
\beta & \text{otherwise},
\end{cases}
\]
with again $\beta[0] \in [\kappa_\beta, \beta)$, the difference $V[t] - V[t-1]$ can be expanded as

$$V[t] - V[t-1] \leq -k_a \epsilon^2 \left( 2 - k_a \sum_{\text{dist}(A_k \Delta[t-1]) \leq t \Delta[t-1]} \exp(-2\pi \delta^2 \|x[t] - k \Delta[t-1]\|^2) \right)$$

$$+ 2k_a |\epsilon \Delta[t]| (|d_f(x[t], \hat{\beta}[t-1])| - \Phi) + 2k_a \epsilon \Delta[t] d_\beta(\hat{\beta}[t-1])$$

$$+ K_1 \left[ (\hat{\beta}[t] - \beta)^2 - (\hat{\beta}[t-1] - \beta)^2 \right]$$

$$\leq -k_a \delta \epsilon \Delta[t]^2 + 2k_a |\epsilon \Delta[t]| \psi(\hat{\beta}[t-1]) (\beta - \hat{\beta}[t-1])$$

$$+ 2K_1 \beta(\hat{\beta}[t-1] - \beta[t]) + 2K_1 k_b \hat{\beta}[t-1] |\epsilon \Delta[t]| + K_1 \epsilon^2 \Delta[t]^2$$

$$\leq (K_1 k_b^2 - k_a \delta) \epsilon^2 \Delta + 2|\epsilon \Delta[t]| (\beta - \hat{\beta}[t-1]) (k_a K_\beta - K_1 k_b)$$

provided the deadzone is at least $\Phi \geq \epsilon_f$ and $k_a$ is chosen so that

$$0 < \delta = 2 - k_a \sup_{\kappa_\beta \leq \beta \leq \beta} \sup_{x \in A} \sum_{\text{dist}(A_k \Delta) \leq t \Delta} \exp(-2\pi \delta^2 \|x - k \Delta\|^2).$$

$K_\beta$ uniformly bounds $\psi(\hat{\beta}[t-1])$ as above, so that, if the arbitrary constant $K_1$ and the bandwidth adaptation gain $k_b$ are chosen to satisfy

$$K_1 k_b^2 < k_a \delta \quad \text{and} \quad K_1 k_b > k_a K_\beta$$

$V[t] - V[t-1]$ is nonpositive, and hence the parameter estimates are uniformly bounded. Since the outputs of the Gaussian network are also uniformly bounded, and the process inputs and outputs are bounded by assumption, the prediction errors are also uniformly bounded, and the same argument given at the end of Section 5.2.1 suffices to show convergence of $\epsilon[t]$ to the deadzone. Note that these conditions require $k_b$ to be quite small, on the order of $\delta / K_\beta$.

### 7.3.3 Comments

The above Lyapunov-like arguments exploit the fact that an upper bound, $\beta$, is known which makes the approximation errors uniformly less than $\epsilon_f$ on the set $A$. Not surprisingly, this results in an adaptive strategy which monotonically contracts the
mesh toward this upper bound in response to non-zero tracking or prediction errors. However, it is not immediately obvious how to develop a mechanism for expanding the mesh when \( \hat{\beta}(t) \) exceeds \( \beta^* \). Indeed, the sampling theory interpretation shows that any bandwidth greater than \( \beta^* \) is sufficient to reproduce \( f \) to the desired accuracy, hence the error signal contains no direct information about when the bandwidth estimate is too large.

Since the bandwidth estimate will continue to increase as long as \( s_\Delta(t) \) or \( e_\Delta(t) \) is non-zero, its adaptation gain, \( k_b \), should be significantly lower than the gains used for the output weights. Adapting the output weights faster than the bandwidth estimates in this fashion reflects the intuitive idea that the network should attempt the "best fit" with the current mesh size before contracting it. Since the regressors in the adaptation laws for the output weights in a Gaussian network are uniformly bounded by unity, the ratio \( k_a/k_b \) is proportional to the relative rates of adaptation.

The bandwidth assumption can thus be relaxed, but not completely avoided, since an ultimate upper bound on the essential frequency content is still required. However, if the network is initialized with a very small initial estimate \( \hat{\beta}(0) \), and a very conservative upper bound \( \beta \) is used (reflecting, perhaps, the maximum number of nodes which can be allocated by the physical implementation of the network), this new adaptation mechanism can potentially find an "effective" bandwidth significantly lower than \( \beta \) which still provides the required tracking or prediction accuracy. This is especially true if the observed data lie in regions within \( A \) in which the underlying function exhibits less high frequency variation than might be expected by a global analysis over the entire set.

### 7.3.4 Examples

To illustrate the bandwidth adaptation idea, Figure 7-4 shows the result of using the above algorithms to predict the output of the chaotic time series examined in Section 5.4.2. Notice that, although the function underlying the observed data from this
process is not bandlimited in the classical sense discussed above, by smoothly truncating $f_1$ as discussed in Section 4.4 it can be uniformly approximated on the interval $[0,1]$ by a bandlimited function, and it is this function the bandwidth adaptation mechanism attempts to recover.

For this simulation, the initial conditions are $\hat{c}_k[0] = 0$, $\hat{\beta}[0] = 0.01$, and $\lambda[0] = 0.25$. The adaptation gains in (7.10) and (7.11) are $k_a = 0.7$ and $k_b = 0.005$, while the network construction parameters in (7.4) are $\theta = 1.5$ and $l = 5$. The bound $\beta = 5$ was used for the projection in the bandwidth adaptation law, and the deadzone size was $\Phi = 0.005$.

As seen in the figures, the eventual steady state reached by the network uses approximately $\sigma^2 = 4\pi$ and $\Delta = 1/6$ resulting in a total of 17 Gaussian nodes with nonzero output weights. Using the dynamic network algorithm specified in Section 7.2, these are thus the only nodes which exist in the network representation of the function generating the observed time series. This should be compared with the values obtained by the direct distributional construction in Section 4.8.2 in which $\sigma^2 = 2\pi$ and $\Delta = 1/4$, for a total of 15 nodes in that network. Thus, while the additional adaptation in this example has not reduced the size of the network representation, it has found centers and variances very close to those obtained by direct computation, with significantly less prior information.

Similarly, Figure 7-5 shows the result of using the above bandwidth adaptation ideas for the second order, unity control gain plant examined in Section 6.4.2. The desired trajectory is again that used for the simulations in Section 6.4, while the initial conditions of the output weights and bandwidth estimate are $\hat{c}_k(0) = 0$, $\hat{\beta}(0) = 0.05$, and $\lambda(0) = \hat{\lambda}(0) = 0$. The dynamic network construction procedure of Section 7.2 is also employed, so that the controller creates nodes only as they are needed, determined by the instantaneous mesh configuration. Also as discussed in that section, the controller considers as “active”, for the purposes of computing the output and for updating the output weights, only those nodes centered at lattice points within the
instantaneous truncation distance, $l\Delta(t)$, of the current network input.

The output weight adaptation law (7.8), and the bandwidth adaptation law (7.9) are used with the control law (6.11) and the values $k_d = \lambda = 10.0$, $k_a = 300.0$, $\Phi = 0.005$, $k_b = 0.75$. The value $\beta = 2.5$ is used as the upper bound in the bandwidth projection, and the network construction parameters are $\theta = 2$, $l = 5$. As can be seen from the figures, after 25 seconds tracking within the tolerance predicted by (6.3) is obtained using an estimated bandwidth about 4 times less than the true bandwidth of the nonlinearity $f$; compare also with Figure 6-10 which uses a network designed using the fixed bandwidth estimate $\beta = 2$. At $t = 25$ seconds, the adaptive component of this controller consists of only 148 nodes, which is a substantial reduction over the conservative, fixed bandwidth network used in Section 6.4.2 which required 1403 nodes.

### 7.4 Incorporating Prior Plant Knowledge

The developments of the preceding chapters by no means suggest that more conventional adaptive architectures should be discarded. Every piece of information available to the designer should be employed in constructing a control system, and in most cases, good estimates of basis functions driving the process dynamics are available. A Gaussian network controller or identifier can then be employed in conjunction with these to account for uncertainties which cannot be so easily parameterized over a particular operating range. Even in the absence of knowledge of a particular set of basis functions, rather than have the network attempt to synthesize linear components of the nonlinearities directly from Gaussians, the terms $Y_k(x) = x_k$, and a bias term $Y_0(x) = 1$, can be directly added to the adaptive component of the control law in practical applications.

Adding linear terms to the representation can be effected by introducing direct connections in the original network between the input layer and the output node,
Figure 7-3: Illustration of the strategy for coupled adjustment of the centers and variances. By coupling adjustment of these parameters through the bandwidth, using the construction procedure of Section 4.7, the network approximates the sampling theorem using low-pass filters of different spatial bandwidths. The adaptation mechanism then needs only to find the correct bandwidth to accurately represent the function.
Figure 7-4: Adaptive Gaussian network predictions of the output of the chaotic time series of Section 5.4.2 using the bandwidth adaptation algorithm of Section 7.3. Top: bandwidth estimates and prediction errors. Bottom: close up view of the prediction errors in the last 10000 time steps of the simulation.
Figure 7-5: Tracking errors using a Gaussian network with bandwidth adaptation as the adaptive element of the unity gain control law developed in Section 6.2.1. Top: bandwidth estimates vs. time. The dashed line represents the actual spatial bandwidth of the nonlinearity being estimated in the control law. Bottom: tracking error versus time. The dashed line is the predicted asymptotic bound.
plus a constant bias term on the output, producing a "fully connected" feedforward architecture, as shown in Figure 7-6. Incorporating the known basis functions can also be accomplished using an analog network design, which operates in parallel with the original Gaussian network. Each hidden layer node of this new network is connected to each input with unity weight, but unlike the Gaussian network which uses nodes with identical input-output behavior, the kth hidden node of the new network computes the nonlinear transform \( Y_k(x) \) of the incoming signals. As discussed in Section 3.2, mapping specific basis functions onto a parallel network in this fashion has also been explored with some success in the early "neural" robotic control literature [83], although without the formal stability and convergence analysis. The general representation of this new adaptive structure is then

\[
\mathcal{N}(x, \hat{c}) = \hat{c}_0 + \sum_{i=1}^{S} \hat{c}_\gamma(i) \exp(-\pi \sigma^2 \|x - \gamma(i) \Delta\|^2) + \sum_{i=1}^{n} \hat{c}_{S+i} x_i + \sum_{i=1}^{N_{bf}} \hat{c}_{S+n+i} Y_i(x)
\]

where, as above, \( S \) is the total number of nodes in the network, \( N_{bf} \) is the number of known basis functions, and \( \gamma \) orders the indices of the lattice points which define the centers of the Gaussian nodes.

The relative weights of the known basis functions are recursively adjusted using adaptive laws exactly like (6.12) or (5.7), replacing the Gaussian with the corresponding \( Y_i \). However, since these additional basis functions are generally assumed to be *globally* exact, the modulation is omitted in computing their contribution to the control law and when adjusting their relative weights. Further, outside the set \( A_d \), the sliding component of the controller needs only offset the incremental difference between the tuned output of the known basis functions and the effects of the unknown dynamics which the network attempts to eliminate when the state is within \( A_d \).

The advantages of incorporating prior known basis functions are clear: the residual components of the "true" nonlinear functions influencing the dynamics are likely to be either very small in magnitude, or very restricted in their domain of influence. In either case, the Gaussian network required to approximate these residual components should be very much smaller than the network required if no prior information were
available. Gaussian network approximation, is thus seen as a method for augmenting existing adaptive control schemes. The networks can be used to characterize elements of the plant dynamics for which explicit models do not exist, while the known dynamic structure can be directly exploited.

A good example of this idea can be found in robotic applications, for which the basis functions governing the motion of the joints as a function of the applied torques are known with great accuracy, and for which stable adaptive schemes exploiting this knowledge are well known [158, 159]. What is much more difficult to accurately characterize in these applications is the impact of Coulomb friction forces at very low velocities, and the effects of torque ripple in the joint actuators. A network of the type proposed could be designed to estimate the friction forces and ripple effects and operate alongside a conventional adaptive controller (which also includes, perhaps, explicit friction and ripple models). An investigation of this idea is currently underway on the Whole Arm Manipulator (WAM) in the MIT AI lab [118].

Similarly, these adaptive Gaussian networks would also be useful in aeronautical or ocean engineering by approximating state dependent elements of the physical dynamics for which prior models are extremely difficult to develop accurately especially over wide ranges of the state variables, such as aerodynamic or hydrodynamic lift and drag coefficients. These approximations could then be used in conjunction with the known equations of motion, substantially enhancing performance.

As an illustration of these ideas, consider a simplified model of the single degree of freedom rotation of an underwater vehicle with an attitude dependent drag characteristic, for example

\begin{equation}
\ddot{\theta}(t) = -(2 + \cos 4\pi\theta(t)) \dot{\theta}(t)|\dot{\theta}(t)| + u(t)
\end{equation}

where \(\theta(t)\) is the rotation angle, and \(u(t)\) is the torque input. The system is required to track the outputs of the model trajectories used for the simulations in Section 4.8.

This dynamic equation is clearly in the canonical form discussed in Chapter 6, allowing use of an adaptive feedback linearization scheme, provided measurements of
the rotation angle and angular velocity are available. For the purposes of controller
design, the dynamics can be considered to be in one of two possible forms, depending
upon the amount of prior information available to the designer. In Model 1, nothing
is assumed known about the dynamics, so that

$$\ddot{\theta}(t) = f_{m1}(\theta(t), \dot{\theta}(t)) + u(t),$$

while in Model 2, the rigid body dynamics of the vehicle are assumed known, and
only the hydrodynamic drag coefficient is assumed unknown:

$$\ddot{\theta}(t) = f_{m2}(\theta(t))\dot{\theta}(t)|\dot{\theta}(t)| + u(t).$$

For Model 1, the function $f_{m1}$ is estimated using a Gaussian network whose inputs
are the rotation angle and the angular velocity. Rather than attempt to develop
a tight constructive bound for the approximation capability of a particular fixed
network architecture, the bandwidth adaptation method and on-line construction
methods discussed in the previous sections are used to learn the mesh size, variance,
and distribution of centers required to solve the tracking problem to the specified
tolerance. The upper bound $\beta = 5$ is assumed to provide an accuracy of at least
$|d_f(x)| \leq 0.05$ on the set $A$.

For Model 2, $f_{m2}$ is estimated by using a Gaussian network with only the rota-
tion angle as input, then multiplying the network output by the the known dynamic
component $\dot{\theta}(t)|\dot{\theta}(t)|$ to form the linearizing component of the control signal. This
multiplicative use of the Gaussian network estimate is yet another technique for utilizing
the flexibility of these structures. Since the function $f_{m2}$ is precisely the function
$f_4$ examined in Section 4.8.4, the same network used in that section can be used to
implement the adaptive approximation for Model 2. However, as the network output
is multiplied before it is used in the control law, this must be taken into account
when bounding the magnitude of $d_f$ in order to select an appropriate deadzone. Noting
that $\dot{\theta}|\dot{\theta}|$ is conservatively bounded in magnitude by 10 on the set $A$, using the
bounds computed in Section 4.8.4 the disturbance term satisfies $|d_{f_{m2}}(x)| \leq 0.055$ for any $x \in A$.

Each of these adaptive estimates are then used in the control law (6.11) to determine the required control input. The output weights of the network used in Model 1 are updated using (6.12) and the centers and variances are adjusted using (7.9) and (7.4), while the multiplicative use of the network in Model 2 requires the modified adaptation mechanism:

$$\dot{c}_k(t) = -k_a(1 - m(t))s_\Delta(t)\dot{\theta}(t)|\dot{\theta}(t)|\exp(-\pi \sigma^2(\theta(t) - k_\Delta)^2).$$

For both simulations, $k_D = \lambda = 10$, $k_a = 100$, and $\Phi = 0.0075$; note that the bound $k_D \Phi \geq \epsilon_f$ is satisfied. For the Model 1 simulation the bandwidth adaptation rate is $k_b = 0.25$ with $\hat{\beta}(0) = 0.75$, and the construction parameters $\theta = 2.25$, and $l = 5$ are used in (7.4). As seen in Figure 7-7, both models produce the asymptotic tracking accuracy predicted by (6.3), however the Model 2 network uses only 22 nodes while the Model 1 network is using 520 nodes, corresponding to a bandwidth estimate of $\hat{\beta} = 1.67$, to achieve the desired performance.
Figure 7-6: Adding linear terms to the estimate can be effected by introducing direct, weighted connections from the network inputs to its output. The unit labeled "1" stands for a node with a constant unity output, allowing for an adjustable bias on the network output.
Figure 7-7: The advantages of incorporating as much prior information as possible into the controller design. Top: Tracking errors obtained using an adaptive bandwidth network and Model 1; at 50 sec. the controller is using 520 nodes. Bottom: Tracking errors obtained using a 22 node network and Model 2. Both give the same asymptotic performance.
Chapter 8

Directions for Future Research

This chapter is devoted to a brief discussion of how the ideas in this thesis might be profitably extended in the future, as well as to a tentative exploration of how the constructions and algorithms developed above relate to, and may be enhanced by, new work being done in the fields of approximation theory and adaptive nonlinear control.

Since this thesis has described how to use neural networks for identification and control applications, the recommendations for future research divide naturally into two categories: those which extend the control algorithms of Chapter 6 to more general classes of nonlinear systems, and those which extend the network construction methods of Chapter 4. Section 8.1 takes up the first of these, discussing methods by which Gaussian networks may be used in two of the newer strategies for adaptively controlling nonlinear systems, specifically the adaptive input-output linearization and the adaptive integrator backstepping algorithms. While these newer designs may allow an extension of the results in Chapter 6 to much larger classes of nonlinear processes, there are several theoretical details which must be addressed before they can be used with assurance, and Section 8.1 attempts to point out several of the more obvious difficulties in extending the existing stability and convergence results.

Following this, Section 8.2 examines how the network designs of Chapter 4 might be productively extended. In particular, some method of addressing the conservatism inherent in the fixed grid used by these designs is sought, first by exploring techniques
of *irregular* sampling theory, then by turning to newer ideas in windowed Fourier analysis and wavelet theory. These techniques are found to hold great promise for constructive designs of networks which conform to the specifications proposed in Chapter 2, and the Gaussian again plays a central role in each model.

### 8.1 Control Extensions

The most important research direction from the control standpoint is to extend the above results to more general classes of nonlinear systems. A number of new techniques have been developed in the past few years, which include the canonical form design of Section 6.1.1 as a special case, and this section briefly sketches how Gaussian networks could be used in each of these algorithms.

#### 8.1.1 Adaptive input-output linearization

This section considers extending the adaptive control designs in Chapter 6 to continuous-time nonlinear processes whose dynamics have the more general form

\[
\dot{x}(t) = f(x(t)) + b(x(t))u(t)
\]

\[
y = h(x(t))
\]

The functions \( f: \mathbb{R}^n \to \mathbb{R}^n \), \( b: \mathbb{R}^n \to \mathbb{R}^n \), and \( h: \mathbb{R}^n \to \mathbb{R} \) are assumed to be smooth, and the state, \( x \in \mathbb{R}^n \), and output, \( y \), are assumed to be measured. Given this information, it is desired to design a control law such that the output asymptotically follows a model trajectory \( y_m(t) \).

To this end, an explicit input-output dynamic model relating the control to the output is needed. Differentiating \( y \) yields \( \dot{y} = L_f h + L_b hu \), where

\[
L_f h = \frac{\partial h}{\partial x} f,
\]

and more generally \( L_f^i h = L_f (L_f^{i-1} h) \) with \( L_f^0 h \triangleq h \), and similarly for \( L_b h \). If \( L_b h \) is bounded away from zero on \( \mathbb{R}^n \), the system has *strong relative degree one* [150].
and there are internal dynamics of order $n - 1$ which are, in some sense, hidden from the input-output representation. If $L_b h$ is identically zero on $\mathbb{R}^n$, then differentiation again yields $\ddot{y} = L_f^2 h + (L_b L_f h)u$, and if $L_b L_f h$ is bounded away from zero on $\mathbb{R}^n$, the system has strong relative degree two, with internal dynamics of order $n - 2$. If not, this process of differentiating the output continues until the function multiplying the control signal is not identically zero, and the number of differentiations required defines the relative degree of the system. The more general case, in which $L_b L_f^i h$ is zero only on a subset of $\mathbb{R}^n$, is much more complex and will not be considered here.

For simplicity in this exposition, it is assumed that the system has strong relative degree $n$, i.e. there are no internal dynamics in the input-output representation, and the control gain function is globally bounded away from zero. In this case, the dynamics of the original system are completely captured in the dynamics of the input-output representation, and can be expressed as $y^{(n)} = L_f^i h + (L_b L_f^{n-1} h)u$, which is exactly the nonlinear canonical form with respect to the new coordinates $y^{(i)} = L_f^{(i)} h$.

Since the input-output dynamics are in the canonical form, if the smooth functions $L_f^i h(x)$ and $L_b L_f^{n-1} h(x)$ were known, asymptotically convergent tracking would be produced by the linearizing control law

$$u = \frac{1}{L_b L_f^{n-1} h} [y_m^{(n)} - L_f^i h - \alpha^T \dot{y}]$$

where $\dot{y}_i = y^{(i-1)} - y_m^{(i-1)}$ and the $\alpha_i$ correspond to the coefficients of a Hurwitz polynomial. However, by assumption the functions $f$, $b$, and $h$ are unknown, and hence $L_f^i h(x)$ and $L_b L_f^{n-1} h(x)$ cannot be used in the control law. Worse, however, is the fact that the tracking error vector, $\dot{y}$, is itself also unknown, since the higher derivatives of the output are not measured and hence cannot be used in the control law.

In the adaptive designs of Chapter 6, the output derivatives were assumed to be measured, and only the nonlinear functions $L_f^i h(x)$ and $L_b L_f^{n-1} h(x)$ needed to be learned by Gaussian networks. In the current, more general setting, note that while the output derivatives cannot be measured, by definition they are related to the
CHAPTER 8. DIRECTIONS FOR FUTURE RESEARCH

state variables through the smooth nonlinear functions $L_{ij}h$, and could hence also be learned by Gaussian networks.

Recall that stable training of the network output weights in such a control law required construction of a signal which was either a direct measure of the (linearly parameterized) function estimation errors, or which was related to these estimation errors by a strictly positive real transfer function. It was noted in Section 6.1.1 that generally the direct tracking error measure $e(t) = \dot{y}(t)$ will not satisfy the SPR conditions, but that, when the dynamics are in canonical form, appropriately chosen linear combinations of the tracking error and its derivatives will satisfy these conditions. Unfortunately, in this case these derivatives are unavailable, so this method cannot be used.

A different technique is available, however, known as error augmentation [150], which adds to the measured error an additional signal, $\eta(t)$, such that $e_1(t) = e(t) + \eta(t)$ can be linearly parameterized by the mistuning of the network output weights, allowing use of robust gradient identification techniques. It should be possible to incorporate into the structure of the Gaussian network the filters required to construct the auxiliary signal and the regressors in the parameterization of $e_1$, creating a new kind of dynamic network architecture capable of solving a much larger class of control problems. Moreover, new robust control designs [157] have been developed for systems in this form, providing the required globally stabilizing component.

A crucial assumption in the convergence proof of the original algorithm [150] is that the regressors used in the adaptation mechanism have derivatives with respect to both the states and the parameter estimates which are globally bounded in these variables, and that the coordinate change is globally Lipschitz. These restrictive assumptions have spurred research into different methods of adaptive nonlinear control, such as those discussed below. Since the Gaussian regressors certainly satisfy the first of these constraints, their use in this algorithm may enable a relaxation of this restriction, however it is unknown what effect the local nature of the parameterizations
provided by these networks will have on the convergence analysis.

8.1.2 Backstepping designs

The canonical form process examined in Section 6.1 can be seen as a subset of the more general classes of dynamic systems described by the so-called parameter-strict-feedback form:

\[
\begin{align*}
\dot{x}_i &= x_{i+1} + f_i(x_1, \ldots, x_i) \\
\dot{x}_n &= f_n(x) + b(x)u, \\
y &= x_1.
\end{align*}
\]

where, as usual, the states \(x_i\) are all assumed to be measured, and \(b(x)\) is assumed globally bounded away from zero. Clearly this reduces to the canonical form (6.1) if \(f_i = 0, i = 1 \ldots n - 1\). The special structure of these dynamics allows use of a new adaptive control design procedure known as integrator backstepping [3, 81, 88, 133, 75, 153] to achieve globally stable adaptive tracking control, assuming only that the nonlinear functions \(f_i\) are smooth and can be linearly parameterized in terms of known regressors. In particular, there are no growth constraints imposed on the nonlinearities in this design, as is required to establish the asymptotic convergence of the design in [150]. Moreover, exactly the same insight which has driven the development of these new adaptive control laws can be used to derive new robust nonlinear control laws, resulting in a new class of sliding controller capable of combating uncertainty on the \(f_i\) to enforce a desired tracking accuracy for these systems [157].

In principle then, since the Gaussian networks used in this thesis provide a locally approximate linear parameterization of functions \(f_i\) belonging to a wide variety of function classes, they can be used to implement the adaptive component of a backstepping design whenever the state is within a compact set \(A \supset A_d\). Outside this set, the new sliding control laws can be used to reduce the tracking error and force
the state back into the set on which the network has good approximating capability. What is not yet clear in this approach is what the effects of the small network approximation errors on $A$ will be, and to what extent they can be accommodated by robust adaptive techniques.

There is, however, a deeper difficulty using neural networks in these designs. To more clearly demonstrate, consider the "benchmark" problem which drove much of the research into these systems [86]:

$$
\dot{x}_1 = x_2 + f_1(x_1)
$$

$$
\dot{x}_2 = x_3
$$

$$
\dot{x}_3 = u.
$$

Assuming for simplicity that $f_1$ can be exactly represented by a Gaussian network $\mathcal{N}(x_1, bve) = c^Tg(x_1)$, the adaptive backstepping design in [81, 88], would suggest the control law

$$
u = \ddot{y}_d - z_3 - z_2 + \frac{\partial \alpha_2}{\partial x_1}(x_2 + \mathcal{N}(x_1, \hat{c})) + \frac{\partial \alpha_2}{\partial x_2}x_3 + \frac{\partial \alpha_2}{\partial \hat{c}}\hat{c} - z_2\frac{\partial \alpha_2}{\partial x_1}\frac{\partial \alpha_1}{\partial \hat{c}}g(x_1)
$$

and the adaptation law

$$
\dot{\hat{c}} = (z_1 - z_2\frac{\partial \alpha_1}{\partial x_1} - z_3\frac{\partial \alpha_2}{\partial x_1})\exp(-\pi\sigma^2(x_1 - k\Delta)^2).
$$

to track the output $y = x_1$. The auxiliary signals in these equations are defined as $z_1 = y - y_d$, $z_2 = x_2 - \dot{y}_d - \alpha_1$, $z_3 = x_3 - \ddot{y}_d - \alpha_2$, where

$$
\alpha_1 = -z_1 - \mathcal{N}(x_1, \hat{c})
$$

$$
\alpha_2 = -z_2 - z_1 + \frac{\partial \alpha_1}{\partial x_1}(x_2 + \mathcal{N}(x_1, \hat{c})) + \frac{\partial \alpha_2}{\partial \hat{c}}\tau_2
$$

$$
\tau_1 = z_1g(x_1)
$$

$$
\tau_2 = \tau_1 - z_2\frac{\partial \alpha_1}{\partial x_1}g(x_1).
$$

Unfortunately, the partial derivatives required in the control and adaptation laws generate a large number of additional signals from the original Gaussian network, for
example
\[
\frac{\partial \alpha_1}{\partial x_1} = -1 + 2\pi \sigma^2 \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}_k (x_1 - k\Delta) \exp(-\pi \sigma^2 (x_1 - k\Delta)^2)
\]
and hence \( \alpha_2 \) contains the term
\[
2\pi \sigma^2 \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}_k (x_1 - k\Delta) \exp(-\pi \sigma^2 (x_1 - k\Delta)^2)
\cdot \sum_{\text{dist}(A,k\Delta) \leq \rho} \hat{c}_k \exp(-\pi \sigma^2 (x_1 - k\Delta)^2),
\]
which resembles the output of a "Hermite" network, whose basis functions are \( xg(x) \) translated on a regular grid, multiplying the output of the original network. The situation becomes even more complex for the derivatives of \( \alpha_2 \) required in the control law, generating higher order "Hermite" networks and products of their outputs.

While these signals are not difficult to construct, the design rapidly loses the elegant simplicity of the network models used in this thesis, and indeed ceases resembling at all the parallel processing model discussed in Chapter 2. It may, however, be possible to re-parameterize the uncertainty in the control law at each stage of the design so as to regain simpler network representations of the required control inputs, and this would certainly be a productive topic for exploration.

### 8.2 Network Extensions

The prime goal for future developments in the neural network models used in this thesis should be to achieve the smallest possible network representation of the functions required by the application. In a sense, this task is similar to image compression; one wants the highest fidelity representation of a complicated function using the smallest possible number of components.

Unfortunately, the designs given by sampling theory are somewhat conservative, and the use of a regular mesh for the location of the centers means that there is the possibility of an exponential explosion in the number of nodes required to approximate
a function, unless the functions become much smoother as the dimension increases. While this smoothness/dimension tradeoff is inherent in approximation theory [128], it is possible that more flexible representations, or different design methodologies, might permit more efficient (i.e. smaller) networks which achieve the same accuracy for a given function class. This section thus presents some preliminary ideas aimed at reducing the conservative, fixed grid used in classical sampling theory, and discusses some new network architectures which are suggested by this analysis.

8.2.1 Alleviating the "curse of dimensionality"

It is clear that if the assumed bandwidth is held constant as the number of independent variables in the function increases, the number of nodes required by the designs of Chapter 4 will grow exponentially with the dimension. Perhaps, however, a more sophisticated design strategy for selecting the network configuration would produce a more compact representation. In fact, there have recently appeared several major new theorems [13,14,53] regarding the theoretical approximation capabilities of feedforward networks, indicating that proper choices of the connection weights can result in perhaps surprisingly rapid convergence, for certain function classes, of the worst case approximation error as a function of network size, $S$.

Starting from a lemma due to Jones [77], Barron [13,14] was able to prove the following: suppose $f$ can be expressed on a compact subset, $A$, of $\mathbb{R}^n$ by the inverse Fourier transform $f = F^{-1}F$, where $\|\nu\|F(\nu)$ is integrable. Then there exist constants $c_k, \xi_k, \theta_k$, and a constant $\kappa_1(f,n,A)$, which depends upon the function, the dimension, and the set $A$, such that

$$\text{ess sup}_{x \in A} |f(x) - \sum_{k=1}^{S} c_k g_s(\xi_k^T x - \theta_k)|^2 \leq \frac{\kappa_1^2(f,n,A)}{S}$$

where $g_s(r)$ is a "sigmoid" function; that is $g_s$ is bounded with a limit of 0 as $r \to -\infty$ and a limit of 1 as $r \to \infty$. Thus, on any compact subset of $\mathbb{R}^n$, a sigmoid type network can uniformly approximate any continuous function satisfying the above frequency
domain constraints with an accuracy whose square improves \textit{linearly} with the size of
the network.

Recently, Girosi and Anzellotti [53], starting from the same technical lemma, have
been able to establish similar rates of convergence for radial basis function networks.
Of particular interest, in view of the results on approximation of bandlimited functions
by Gaussian networks considered above, is the following consequence of these newer
theorems: Suppose \( \mathcal{F}f \) has compact support and is in addition \( m > n/2 \) times
continuously differentiable. Then there exist constants \( c_k, \xi_k \), and a constant \( \kappa_2(f, n) \)
which depends upon the function and the dimension, such that
\[
\text{ess sup}_x |f(x) - \sum_{k=1}^{S} c_k g(x - \xi_k)|^2 \leq \frac{\kappa_2(f, n)}{S},
\]
where \( g \) is the radial Gaussian. That is, it is possible to choose the centers of a
Gaussian RBF expansion in such a way that bandlimited functions with sufficiently
differentiable spectra can also be uniformly approximated with a degree of accuracy
whose square improves \textit{linearly} with the size of the network.

These linear rates of convergence have attracted a great deal of attention lately.
It is important to note, however, that the constants \( \kappa_1 \) and \( \kappa_2 \) in the above formulae
\textit{do} depend upon the dimension. They can be expressed respectively, as
\[
\kappa_1(f, n, A) = \gamma_1(n) \int_{\mathbb{R}^n} \|\nu\|_A |F(\nu)|^2 d\nu
\]
\[
\kappa_2(f, n) = \gamma_2(n) (\|\lambda\|_1^2 - \|f\|_{H^{s,2}}^2)
\]
where \( \gamma_1 \) and \( \gamma_2 \) are constants depending only on the dimension, \( \|\nu\|_A = \sup_{x \in A} |\nu^T x| \).
\( \lambda = \sigma^n \mathcal{F}^{-1}(FG^{-1}) \), and \( \|f\|_{H^{s,2}} \) is a Sobolev norm
\[
\|f\|_{H^{s,2}}^2 = \int_{\mathbb{R}^n} (1 + \|\nu\|^2)^s F^2(\nu) d\nu.
\]
with \( s > n/2 \).

In the case of functions which are bandlimited to a ball of radius \( \beta > 1 \), for exam-
ple, it is easy to see that the constant \( \kappa_1 \) can explode exponentially with dimension.
CHAPTER 8. DIRECTIONS FOR FUTURE RESEARCH

This has an important practical implication: to maintain a specified level of accuracy with a constant number of nodes as the input dimension increases, it would still be necessary to exponentially contract the class of functions approximated (as measured by the bandwidth parameter $\beta$). It is not so obvious, however, how rapidly $\kappa_2$ grows with dimension for bandlimited functions.

On the other hand, for a particular function class and dimension, this $1/S$ rate of convergence is not generally achievable using classical linear subspace methods [13, 53], so there appears to be a benefit to exploiting these properties of feedforward networks, and attempting to identify constructive techniques which may realize this theoretical potential. The following two sections thus examine several techniques in approximation theory which offer additional insight into methods for selecting the network parameters, and how they relate to properties of the function being approximated.

8.2.2 Scattered-center networks and irregular sampling

Use of classical sampling theory in the constructions of Chapter 4 led to the natural identification of the centers in a RBF expansion with a lattice of uniformly spaced sample points; this interpretation allowed each component of the network to be given a precise interpretation, and its impact on the overall approximation capability to be easily quantified. A natural extension of these ideas, which retains this intuitive interpretation of each of the network components, is to use irregular sampling theory to analyze the approximation capability of arbitrary distributions of centers.

Girosi and Anzellotti's results can be interpreted as stating the existence of a particular set of (possibly irregular) sample points which, when used as the centers of a radial Gaussian expansion, allows the indicated rate of uniform convergence for certain bandlimited functions. Rather than attempt to explicitly identify which points in the domain are required to achieve the theoretical convergence rate (since they may in fact not be unique), it is useful to instead explore a related question:
when is a given finite collection of centers sufficient to recover a bandlimited function to a prespecified accuracy on a given subset of \( \mathbb{R}^n \)\

Irregular sampling expansions of bandlimited functions have been known since at least the mid 1960's, starting with the pioneering work of Beutler [22, 23] and Yao and Thomas [183], the latter being the first to realize the distinction between the theoretical existence of an irregular expansion, and the practical stability of the reconstruction procedure with respect to, e.g. round-off errors, and sample noise. As demonstrated in the recent work by Benedetto and Heller [20, 21], however, the development of a general theory of irregular sampling is made much more elegant and comprehensive using the formalism of frame theory. While frame theory was originally developed in the 1950s [45], it has only recently become of central importance thanks to its role in the development of wavelet analysis techniques [40, 42].

Why are frames so important? The expansion of an element of certain function classes by means of orthonormal bases is a well known technique in engineering and mathematics, and its utility has motivated quite a large amount of research to the development of such bases. Often however, especially in considering the approximation capabilities of neural networks, the functional form of the “basis” elements has been prescribed, by e.g., ease of implementation, or observed neurobiology. The resulting collection is then almost surely not an orthonormal basis, and the question naturally arises as to how to meaningfully evaluate its representational properties.

Certainly one important requirement is that the given collection be capable of uniquely characterizing each element of the chosen function class, in some appropriate sense. In practice, however, one demands more than this; one requires assurances that the algorithm which reconstructs the function using these basis elements be stable, in the sense that “small” errors made in computing the coefficients of the expansion result in “small” errors in the reconstructed function, where “small” is measured with respect to an appropriate metric. If the functions to be approximated live in a Hilbert space, the theory of frames both establishes criteria which ensure the
desired completeness and stability properties, and provides explicit formulae for the reconstruction procedure [35, 42, 60].

Frame analysis and the equivalence, under Fourier transformation, of translates of a prototype function with modulation by complex exponentials, show [20, 21, 61] that the question of when a bandlimited function can be reconstructed using irregular translates of a prototype function hinges on the frame properties of collections of complex exponentials. That is, if the set of irregular sample points corresponds to a set of frequencies for which the resulting complex exponentials form a frame for $\mathcal{L}^2(\Lambda_\Delta)$, where $\Lambda_\Delta$ is a compact subset of $\mathbb{R}$, then any function bandlimited to a strict subset of $\Lambda_\Delta$ can be stably reconstructed using the corresponding translates of an appropriate low-pass filter. This analysis can be extended to more general low-pass filters such as the Gaussian, and moreover provides formulae defining the aliasing error and the required output weights, although these are much more complex than those devised in Chapter 4 [61, 148].

The conditions under which a set of complex exponentials forms a frame for compact subsets of $\mathbb{R}$, have been extensively explored [20, 184] and thus the tools are available to judge the “admissibility” of a candidate collection of irregular sample points. Moreover, tensor products of exponential frames are again frames, so that the admissibility criterion can be applied to each component of a set of sample points in $\mathbb{R}^n$ to adjudge its suitability [61]. What is unknown at this time is how tightly the approximation errors can be bounded compared to the regularly sampled case, and whether these irregularly sampled representations will actually result in smaller networks to achieve the same accuracy.

8.2.3 Local bandwidth: Gabor models and wavelets

This thesis has made fundamental use of the Whittaker/Shannon model of function reconstruction, exploiting the low-pass filtering properties and rapid spatial decay of the radial Gaussian. There are, however, other, potentially more powerful techniques
CHAPTER 8. DIRECTIONS FOR FUTURE RESEARCH

which make use of the simultaneous space-frequency localization properties of this function. These techniques may be viewed as logical extensions of the methods detailed above, and can be analyzed in a similar framework, suggesting constructive designs for new classes of neural networks which use Gaussian nodes. As in the previous section, the analysis will start in one dimension and then consider multidimensional extensions.

Gabor Models

The sampling theorem says that if a set of basis functions is formed by taking a single prototype function, corresponding to the impulse response of an ideal low-pass filter with cut-off frequency $(2\Delta_x)^{-1}$, and translating it to points in a regular mesh, $k\Delta_x$, the resulting collection can represent any square-integrable function bandlimited to $\beta < (2\Delta_x)^{-1}$. If instead the function has significant frequency content for $|\nu| > (2\Delta_x)^{-1}$, the sample points and prototype function can be adjusted to open the filter bandwidth as required. However, if the additional frequency content is itself very concentrated, for example in the range $[\nu^-, \nu^+]$, where $\nu^- > 0$ is much greater than $(2\Delta_x)^{-1}$, it would be quite inefficient to extend the filter bandwidth to accommodate these new frequencies. Such a strategy would then require an extremely small $\Delta_x$, resulting in a very large number of basis functions needed to achieve a certain accuracy on a compact set.

Instead, if the range $\nu^+ - \nu^- < \Delta^{-1}$, the original low-pass filter can instead be translated in frequency to cover this new frequency range. Since translation in frequency becomes modulation by a complex exponential in the spatial domain, the sampling theoretic argument can now be applied to show that the components of $f$ which lie in the range $[\nu^-, \nu^+]$ can be exactly reconstructed using the translates of the new basis function

$$g'_c(x) = e^{2\pi j \bar{\nu} x} g_c(x)$$

on a regular grid with mesh size $\Delta_x$, where $\bar{\nu} = (\nu^- + \nu^+)/2$, and $g_c$ is the impulse
CHAPTER 8. DIRECTIONS FOR FUTURE RESEARCH

response of the original low-pass filter.

This suggests that, rather than try to choose a particular prototype basis function based upon the worst case frequency content of \( f \), instead consider a family of basis functions corresponding to the impulse responses of identical filters centered at different points in frequency, i.e.

\[
f(x) = \sum_{k \in \mathbb{Z}} \sum_{m \in \mathbb{Z}} c_{km} e^{2\pi j \Delta \nu m} g_c(x - k \Delta_x).
\]

Of course, if one hopes to recover any Fourier transformable function by this method, the sequence \( \Delta \nu m \) should be chosen such that the entire frequency axis is covered, and so that there are no "holes" in the coverage of the array of low-pass filters. Making the logical choice \( \Delta \nu m = m \Delta \nu \), this then requires that \( \Delta_x \Delta \nu \leq 1 \). In fact, the choice \( \Delta_x = \Delta \nu = 1 \) produces an orthonormal basis for \( \mathcal{L}^2(\mathbb{R}) \) [42].

If this argument is now applied to the Gaussian sampling models developed in Chapter 4, the corresponding expansion is

\[
f(x) = \sum_{k \in \mathbb{Z}} \sum_{m \in \mathbb{Z}} c_{km} \exp(2\pi jm \Delta \nu x) \exp(-\pi \sigma^2 (x - k \Delta_x)^2)
\]

which is precisely the representation developed by D. Gabor [52] in his 1946 paper on communication theory. This expansion exploits the simultaneous localization properties of the Gaussian in both space and frequency to produce a local Fourier decomposition of the function: each coefficient \( c_{km} \) represents the extent to which frequencies in a neighborhood of \( m \Delta \nu \) contribute to reconstruction of \( f \) in a neighborhood of \( x = k \Delta_x \). As with the array of ideal low-pass filters considered above, if the mesh sizes are chosen appropriately this collection in fact forms a frame, permitting numerically stable reconstruction of \( f \) from the sequence \( \{c_{km}\} \) [42].

One of the chief advantages of such a representation is that it permits the idea of local bandwidth to be introduced, by truncating the inner summation at different values depending upon \( x \), for example:

\[
\hat{f}(x) = \sum_{\text{dist}(k \Delta_x, A) \leq \rho_x} \sum_{m \in \mathcal{M}(k)} c_{km} \exp(2\pi jm \Delta \nu x) \exp(-\pi \sigma^2 (x - k \Delta_x)^2)
\]
where $\mathcal{M}(k)$ consists of those points $m$ such that, in a neighborhood of $x = k\Delta x$, $f$ has significant frequency content near $\nu = m\Delta \nu$. An especially convenient expression for each $\mathcal{M}(k)$ could take the form of a local bandlimit, $\beta_k$, so that each $\mathcal{M}(k)$ would consist of all frequency lattice points, $m\Delta \nu$ contained in $\Lambda(\beta_k)$.

Provided that the collection $\{\mathcal{M}(k)\}_{k \in \mathbb{Z}}$ is chosen to accurately reflect the frequency content of $f$, the approximating capability of this expansion can be quantified, especially given the analytic tools developed in [42]. In fact, the constructions of Section 4.5 could instead be viewed as truncated Gabor representations, where the assumed global bandlimit on the function being approximated means that $\Delta x$ can be chosen such that minimal truncation error is incurred by choosing $\mathcal{M}(k) = 0$ for every $k$. The analysis of the aliasing error in Section 4.7 is then precisely equivalent to this frequency truncation error of the Gabor expansion.

Expansion (8.1) can also be mapped onto a parallel network, as already noted by some vision researchers [43], and is in fact a straightforward extension of the designs proposed above. Each Gaussian node in this new network is centered at a spatial lattice point $k\Delta x$ as before, but is now modulated by a weighted sum of complex exponentials, or equivalently sine and cosine functions, corresponding to the frequency lattice points contained in the set $\mathcal{M}(k)$. This modulation can be implemented using a second set of "Fourier" nodes in the network, which compute the corresponding exponential or sine and cosine; the weighted outputs of these new nodes are then summed together to form the individual Gaussian modulations.

The approximation implemented by this new network representation is evidently linear in the output weights, and hence estimates of the required weights can be recursively adjusted to produce stable identification and control algorithms, exactly as detailed in Chapter 5 and Chapter 6. Yet, in this form the Gabor model does not offer much more flexibility than the design considered in Chapter 4, since accurate characterization of $f$ requires prior knowledge about the location of the regions of local high frequency behavior. However, if the bandwidth adaptation and latent
node ideas of the previous chapter are applied to local bandwidth estimates \( \hat{\beta}_k(t) \), the resulting network may have the desirable capability of determining by itself the regions of high frequency content. Instead of globally contracting the mesh, as done in Section 7.3, each local estimate would \( \hat{\beta}_k(t) \) change only when the state is within the truncation distance, \( \rho_x \), of the node centered at \( k\Delta \), and as this estimate grows progressively more Fourier nodes are created and added to the modulation of the Gaussian centered at this point. The node density may thus become high only in those regions of the state space where the function is rapidly varying, and a given degree of uniform approximation accuracy can be ensured with potentially far fewer nodes. This idea merits further exploration.

The extension to multiple dimensions is straightforward by taking tensor products of one dimensional expansions, producing an expansion in radial Gaussians modulated by multivariate complex exponentials. Irregularly sampled versions of this representation are also possible, and can be analyzed using the frame formalism mentioned above. Both of these are considered in the abstract analysis of [47, 48].

Unfortunately, while these representations would potentially expand the classes of functions for which constructive network designs are available, and potentially produce smaller representations than the designs of Chapter 4, they do not conform well to the one hidden layer network designs proposed in Chapter 2, resembling rather the tensor product of a Fourier series network with a radial Gaussian network. It is thus desirable to explore representations which do not require modulations by a complex exponential in order to achieve these enhanced functional representation abilities.

**Wavelet Models**

Suppose that, unlike the Gaussian or ideal low-pass filter, the frequency response of the prototype function is zero at \( \nu = 0 \) and otherwise has localized "lobes" for \( |\nu| > 0 \). A classic example is the "Mexican hat" function \( h(x) = (1 - x^2) \exp(-\pi x^2) \), whose
frequency response is plotted in Figure 8-1. The points around which these lobes are localized in frequency, and the width of the lobes themselves, can be adjusted now by a simple rescaling of the dependent variable \( x \), as shown in the figure. Rescaling the argument of \( h \) thus translates the effective centers of the approximate low-pass filter implemented by the lobes of its frequency response. Combining this with the sampling theoretic viewpoint of the previous section suggests an exploration of representations of the form

\[
f(x) = \sum_{k \in \mathbb{Z}} \sum_{m \in \mathbb{Z}} c_{km} h(\Delta_{\nu}^m x - k \Delta_x)
\]

where now \( \Delta_{\nu} \) is a dilation parameter such that \( \Delta_{\nu} > 1 \).

This is in fact the class of representations examined by the rapidly developing branch of approximation theory known as wavelet analysis, in which a function is reconstructed from translations and dilations of a single template function called the "mother wavelet" [42]. The template must satisfy certain admissibility conditions, which serve to ensure that \( (\mathcal{F}h)(0) = 0 \), that \( \mathcal{F}h \) has the desired localization in frequency, and that \( h \) has good spatial localization properties. Provided these conditions are satisfied, it is possible to choose the dilation and translation parameters such that the resulting collection of basis functions forms a frame, and numerically stable reconstruction of \( f \) from the coefficients \( c_{km} \) is then guaranteed for any \( f \in L^2(\mathbb{R}) \) [42].

Most importantly, however, this representation maps directly onto the class of single layer networks considered in Chapter 2, by taking the activation function to be the mother wavelet, \( h \), and choosing as the input function the dilated translation \( \varphi(x, m \Delta_{\nu}, k \Delta_x) = \Delta_{\nu}^m x - k \Delta_x \). The input weights of this network thus parameterize the different dilations and translations of the network input signal, and the required output weights could be explicitly computed by determining the functions dual to each of the dilated and translated wavelets.

To develop a finite sized network expansion, the space-frequency localization properties of the mother wavelet can be exploited to select, based upon the size of the set
CHAPTER 8. DIRECTIONS FOR FUTURE RESEARCH

upon which reconstruction is desired and assumed upper bounds on the information about $f$ which can be extracted at the $m$th scale, a subset of the above infinite expansion which provides a prespecified approximation capability. The analysis in [42] provides many of the tools necessary to carry out this program rigorously. In fact, constructive analyses of such wavelet networks have already begun to appear in the neural network literature, both for sigmoidal networks in one dimension [120], and for tensor product “Gaussian derivative” networks in two dimensions [187].

As in the designs considered above, for the purposes of adaptive control or recursive identification it is not necessary to have explicit formulae for the $c_{km}$; it is sufficient to know that such a linearly parameterized expansion of the required function exists. Given bounds on the approximation capability of the chosen finite expansion, estimates $\hat{c}_{km}(t)$ of the required output weights can then be recursively adjusted by the methods developed above while retaining the stability and convergence properties of the adaptive algorithms.

The extension to multiple dimensions of these techniques is usually effected by taking tensor products of one dimensional wavelet bases, which could be implemented using the more complex tensor product networks discussed in Chapter 2. Fortunately, recent work also shows “radial wavelet” expansions to be possible, using a mother wavelet $h(r)$, with $r = \|x\|$, provided that the one dimensional admissibility conditions are satisfied with respect to the variable $r$ [42]. These expansions would map directly onto the simpler radial network representations used in this thesis, using scaled and translated Euclidean norms for the node input functions.

There are many potential benefits to using, for example, radial “Mexican hat” wavelets instead of the radial Gaussian networks considered above. First, the wavelet expansions extend the class of functions for which constructive network designs would be available. Second, and perhaps more important, a wavelet expansion in a sense offers a direct method of addressing the scattered-center problem and potentially developing more compact representations of the function. The above representa-
tion directly introduces the freedom to use high node densities in regions where the function has a substantial amount of high frequency behavior, and low node density elsewhere. While the centers of the wavelet responses are constrained to lie at the points \( k \Delta^m \Delta_x \), by using different ranges of \( k \) at different scale levels \( m \), there is still great flexibility in the locations for the centers, although they are not truly "scattered".

Moreover, because the spacing of the centers of such an expansion is coupled with the scaling of the response of the basis function, it also offers a formal framework in which to evaluate the "HyperBF" representations suggested in [125]. The HyperBF technique suggests approximating functions using radial basis functions operating at different scales, for example using Gaussians with different variances. This is very similar to what is being implemented in a radial wavelet expansion, and hence the analytic tools provided by wavelet theory may be useful to suggest optimal HyperBF representations.

It should be noted, however, that the bounds provided by existing wavelet theory may not always be very tight, and that closed form representations for the optimal coefficients \( c_{km} \) are not always available. Nor is there any reason to expect a wavelet expansion to be more parsimonious when approximating the same class of functions considered by sampling theory, and it would be quite useful to explore the circumstances under which wavelet methods should be preferred to, e.g., the generalized sampling theoretic methods examined in Chapter 4. Recent theoretical work into the optimality of wavelet expansions may offer additional insight into this topic [44].
Figure 8-1: Fourier transform of scaled "Mexican hat" functions, $h(ax)$ where $h(x) = (1 - x^2) \exp(-x^2/2)$. Rescaling widens the side lobes of the response and moves their effective centers further out the frequency axis.
Chapter 9

Summary and Conclusions

The most general themes running through this thesis may be summarized as follows:

- In most cases, neural networks can only *approximately* represent the functions which permit perfect identification or control of a nonlinear dynamic system.

- This fact requires use of *robust* design techniques for control laws and recursive adaptation mechanisms in order to ensure stability and convergence.

- Robust designs require a quantification of the approximating capability of the chosen network architecture, and hence demand a *constructive* theory of neural networks.

Drawing from many different disciplines, this thesis has attempted to synthesize a unified framework capable of addressing these three fundamental observations. In particular, specific designs have been presented for both a constructive theory of a class of neural networks, as well as for robust identification and direct adaptive control structures which use these constructions, and which have been formally proven to be both stable and convergent.

Sampling theory has been extensively used to provide an analysis and synthesis tool for neural networks. By restricting the centers of a *Gaussian radial basis function* network to lie on a regular grid covering the set on which the function must be well approximated, and restricting each hidden layer node to have the same variance, a Gaussian network can be viewed as an approximation to the multidimensional
CHAPTER 9. SUMMARY AND CONCLUSIONS

tsampling theorem. This allows a precise interpretation of each of the network components:

- The centers represent sample points;
- The variance controls the effective bandwidth of the Gaussian low-pass filter;
- The output weights are the samples of a continuous function, related to the function being approximated by a simple spatial convolution.

This construction also allows a precise diagnosis and quantification of the sources of approximation error:

- Class membership error: since few functions are ideally bandlimited, but may be well approximated by bandlimited functions on compact sets.
- “Aliasing” error: since the Gaussian deviates from an ideal low-pass filter
- Truncation error: since the sampling theoretic expansion is infinite, and a practical network can contain only a finite number of nodes.

By interpreting polynomials, sines, and cosines as bandlimited distributions, exactly the same construction and interpretation can be given for the components of a Gaussian network approximation to these functions, and the resulting approximation errors similarly quantified.

This constructive theory of networks provides conservative estimates of the worst case uniform errors, on a prespecified compact subset of $\mathbb{R}^n$, that a correctly designed Gaussian network will commit when approximating any function belonging to one of the above function classes. Conversely, given the assumed function class and the set on which accurate approximation is required, the design exactly specifies the required variance and the number and distribution of centers needed to approximate any member of this class to a specified accuracy. Only the output weights of the network then need to be learned in order to represent a specific member of the function
class, and hence the network estimate of the function is linear in the adjustable output weights.

Assuming that the functions underlying the observed input-output behavior of a nonlinear process belong to classes which can be well approximated by these Gaussian network designs, stable, convergent identification structures which employ these networks have been specified. The linear parameterization provided by the sampling theoretic constructions, together with the bound they provide on the degree and extent of the network's approximation capability, can be used with recursive gradient adaptation methods made robust using deadzone modifications, to produce stable recursive identifiers whose asymptotic prediction capabilities are limited only by the approximating capability of the network. Moreover, use of radial basis function networks for these applications allows an exact specification of the conditions which must be satisfied by the signals input to the network to ensure that the adaptive mechanism is persistently excited.

In identification applications, the process state is assumed to remain within a bounded subset of $\mathbb{R}^n$, allowing networks to be designed with a guaranteed approximating capability for any behavior the process may exhibit. In a direct adaptive control application, however, since the output of the adaptive network directly influences the evolution of the process state, it is not possible to specify a priori a bounded set to which the state is confined; depending upon the (assumed unknown) amount of initial mistuning in the adjustable parameters of the network, the process state may be forced out of any prespecified bounded set by the erroneous outputs of the network.

Robust adaptive techniques are thus insufficient in a control application to accommodate the inexact parameterization provided by a network; robust elements must be incorporated into the control law as well. By combining sliding control with an adaptive feedback linearization control strategy, in which Gaussian networks adaptively cancel the natural nonlinear dynamics of the plant, and using again robust
adaptation mechanisms, a class of stable, convergent direct adaptive controllers has been developed. The asymptotic tracking accuracy of these controllers is limited only by the ability of the adaptive networks to approximate the nonlinear functions driving the plant dynamics. A unique feature of the adaptive control laws proposed is their ability to smoothly transition from adaptive to robust modes of operation, becoming a robust nonlinear controller in the regions of the state space where the network has poor approximating capability, a purely adaptive controller where the network approximating power is good, or a stabilizing blend of the two modes in an intermediate transition region.

This blended control strategy has also been combined with a robust version of the classical adaptive robot control algorithms to create stable adaptive control laws for a class of multivariable nonlinear systems with a passive input-output structure. Gaussian networks are used to estimate the different nonlinear functions required in this control law, and sigma-modification robustness techniques are used in the adaptation mechanism for the output weights of these networks. While pointwise convergence results are not available for the tracking errors in this algorithm, asymptotically the average "energy" of the tracking errors is shown to be proportional to a bound determined by the uniform approximation capabilities of the networks used in the control law.

The unification of approximation and stability theory in these designs has many beneficial side effects. For example, by noting that the accuracy of a Gaussian network approximation at a given point in \( \mathbb{R}^n \) depends only upon nodes in the sampling mesh within the truncation distance of this point, new nodes of the network can be brought into existence only as needed, i.e., only when the inputs to the network pass within the truncation distance of the corresponding center. The sampling mesh thus defines a lattice of latent node positions, and the approximation theoretic analysis hence naturally specifies a mechanism for minimally expanding the network in response to a particular sequence of inputs, without sacrificing any of the stability or convergence
properties of the above algorithms. Similarly, by replacing the sum-squared output weight mistuning terms in the Lyapunov functions used in the stability proofs with more general positive definite forms, lateral inhibition and excitation can be stably added to the adaptive algorithm, permitting the learning of the output weight for a particular node to be influenced by the outputs of neighboring hidden layer nodes.

The unified viewpoint also offers insight into mechanisms for stable, on-line adjustment of the centers and variances in a Gaussian network. By appropriately coupling changes to these parameters via the assumed bandwidth of the function being approximated, then making the bandwidth another estimated parameter, the mistuning in the bandwidth estimate appears linearly when the resulting function estimation errors are bounded. Robust recursive gradient methods can thus be used to simultaneously adjust both the output weights and the bandwidth estimate. While this results in a strategy of monotonically contracting the mesh size, it can potentially find much coarser mesh sizes, and hence much smaller networks, which still provide the required accuracy.

This unified framework is expected to continue to yield valuable insights as more complex network architectures and adaptation mechanisms are pursued, and as the control algorithms developed herein are extended to more complicated dynamical system structures, possibly along the lines suggested in Chapter 8. Of course, in so doing one must be careful not to overlook existing, non-neural solutions for the problems under consideration, which may indeed more effectively exploit available prior information to achieve the required performance. A neural network should be viewed not as a panacea, but rather as yet another utility in an engineer's toolbox, whose use in any particular application should be rigorously justified. This thesis has attempted to develop the tools to enable such judgments, and to identify specific instances in which these devices can enhance existing techniques of identification and control.
Bibliography


BIBLIOGRAPHY


