Learning Augmented Recursive Estimation for Uncertain Nonlinear Dynamic Systems

Stark Christiaan Draper

B.A., History (1994)
Leland Stanford Junior University

B.S., Electrical Engineering (1994)
Leland Stanford Junior University

submitted to the department of
electrical engineering and computer science
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1996

Author ....
Department of Electrical Engineering and Computer Science
May, 1996

Certified by ....
Peter Dayan, Ph.D.
Assistant Professor of Computational Neuroscience

Certified by ....
Rami Mangoubi, Ph.D.
Thesis Supervisor, Charles Stark Draper Laboratory

Accepted by ....
F. R. Morgenthaler, Ph.D.
Professor of Electrical Engineering
Chair, Department Committee on Graduate Students

Learning Augmented Recursive Estimation for Uncertain Nonlinear Dynamic Systems

Stark Christiaan Draper

submitted to the department of
electrical engineering and computer science
in partial fulfillment of the requirements for the degree of
master of science at the Massachusetts Institute of Technology

May, 1996

Abstract

This thesis addresses the joint problems of state estimation and system identification for partially unknown dynamic systems. We focus on the case where the unknown dynamics are nonlinear time-invariant function of the state. We use a spatially localized learning system to construct a state-dependent mapping of the unknown nonlinearities. This map then serves as a source of evolving long-term memory for use in tasks of estimation and identification. Standard filtering algorithms (EKFs) that exploit the knowledge stored in the learning system are used to estimate both the system state and the correction terms for the learned mapping. These state/correction term pairings are then used to refine the learned mapping of the unknown dynamics. This approach contrasts with many existing identification techniques that treat the unknown nonlinearities as time-varying quantities. By not storing any information about previous estimates of the unknown dynamics, these techniques fail to improve their performance as they explore the state space. Numerical results for a number of dynamic systems are included that demonstrate the improved performance of this approach and provide insight into the strengths and weaknesses of the algorithm.

Academic Supervisor:	Peter Dayan, Ph.D. Assistant Professor of Computational Neuroscience
Thesis Supervisor:	Rami Mangoubi, Ph.D. Senior Member of the Technical Staff, Draper Laboratory

Acknowledgment

I must say that when I was first offered support at Draper Laboratory to fund my graduate studies at MIT, I was a bit hesitant. I was not quite sure what it would be like to walk into a building with the prominent portion of my name on the door every day, or to hear stories of lore about my grandfather. People had commonly enough made the connection at Stanford, and that was a good couple of thousand miles from the Infinite Corridor. Being a graduate student in course VI, ten courses away from Aero/Astro, might be close enough. But on further, and deeper, reflection I came to realize that it might be a great opportunity, not only to study at MIT, but to find out what went on in that black-glassed building, to work with the people who had sent Apollo to the moon, and indeed to hear those stories of my grandfather. I have not been disappointed. While being a Draper Fellow might isolate one a little from graduate life at MIT, the staff at Draper makes a concerted effort to interact with the Draper Fellows and to make up for that isolation, to both parties' benefit I think. Indeed, though I really had nothing to do with it, the whole mission of the Lab to support graduate education is really quite notable nowadays, so... sure, you guys can keep my name on the door.

I would especially like to thank Rami Mangoubi who, although our first discussions were about the Palestinian-Israeli conflict, eventually took over supervision of my project and really spent much of his busy schedual helping me work on it. Peter Dayan was my academic supervisor and although he entered into the project after the problem and approach had already been firmed up, he took an active interest and made some significant contributions in how I went about experimenting with the architecture, and in clarifying the connection between learning and estimation. I would also like to thank Walter Baker who suggested the problem I ended up working on in the first place, and though he's currently doing his doctorate at Harvard, took a continuing interest in this project's progress (I won't tell Roger). Dean Cerrato was my first real (third actual — but that's another story) supervisor at Draper, and he continued to leave his door (and bookcase) open even after the focus of my project moved away from learning systems and towards estimation and identification.

I would also like to thank the rest of the staff (and eventually students) who participated in learning group meetings and made suggestions on my, and other students', projects — Milt Adams, Neil Adams, Jim Negro, Keith Rogers, Matt Welborn. There were other students and staff, both at Draper and MIT, who helped me through various stages of the project: Elaine Chew, Larry McGovern, Homer Pien, Afshan Hamid, Bill Hall, Rich Barron, Sridevi Sarma, Austin Frakt, Cathy (the cute) O'neil. Of course, the Draper folks who like to head over to the Cambridge Brewing Company on those snowy Friday afternoons should be included in the last category as well. My parents even contributed a little, on both the technical and dining-out sides. I think in closing it is appropriate to mention my academic advisor, Professor Alan Oppenheim, who first thought to forward my MIT application to Draper Labs, and without whose acquaintance I'd still be in sunny California... hmmmm...

This thesis was prepared at The Charles Stark Draper Laboratory, Inc. Funding was provided by a Draper Laboratory Independent Research and Development (IR&D) project, number 96-0-718.

[illegible]

Stark C. Draper

Contents

List of Figures

Chapter 1

Introduction

Many physical phenomena, natural and man-made, are well modeled by dynamic systems. The state of the dynamic system encapsulates all information needed to describe the system at the current time, and the system dynamics mathematically describe how the state evolves with time. Given noisy observations of the state, or a function thereof, a central question in system theory is how best to estimate the state of the system. Under certain conditions this problem has an explicit solution, but in general it is quite difficult.

A further difficulty arises when one's description of the system dynamics is incomplete. We concentrate on this case where the unknown dynamics are time-invariant and nonlinear. The problem at hand is thus one of joint state estimation and system identification. We will investigate the application of learning ideas, taken from the wealth of research on learning systems that has emerged in the past decade, to this joint problem of estimation and identification. We will show the potential benefits of drawing on both fields as compared to some standard nonlearning techniques.

Often problems that require both state estimation and system identification are approached via state augmentation and nonlinear filtering methods [8, 13]. In this approach, unknown parameters are treated as additional states and introduced into the original state vector. Conventional filtering methods are then used to estimate this augmented state vector.[1] Although the unknown parameters may actually be nonlinear functions of the original state, they appear in the augmented filtering framework as time-varying quantities that are estimated in a recursive fashion.

A deficiency in this approach arises if the system switches repeatedly among different operating points. This type of estimation and identification algorithm will be forced to identify the corresponding unknown dynamics from scratch each time — it will have to readjust fully every time the system revisits an operating point it has been to in the past. Effectively then, the influence of previous estimates is age-weighted. Old estimates of the nonlinearities matter little after sufficient time has elapsed. If some form of memory could be incorporated into the system, then the estimator could partially rely on "remembered" information from previous visits to this area of the state space, when attempting to carry out its identification task. It is in the identification and storage of such information that ideas from the learning literature will come to the fore.

The primary objective of this thesis is to formulate a learning augmented estimation methodology for partially unknown dynamic systems. As well as incorporating ideas of memory, we must also determine how to combine this learned information with new information collected as the system operates.

Over the past decade a number of researchers have applied learning to problems of estimation, identification, and control of dynamic systems (e.g., see [6, 10, 11, 14, 16, 17, 18, 23]). In general these approaches fall into two categories. The first category addresses the estimation problem. These ap-

[1]An illustrative example is described in Section 9.4 of [8], where the problem is to design a filter for a missile autopilot, given only a partial model of the dynamics and noisy measurements. We will develop this example in Section 2.4.

proaches use the universal function approximation capability of many learning systems to attempt to obtain unbiased, minimum-variance filters for stochastic nonlinear systems. One recent approach is discussed in [14] where a training set (incorporating the actual system states) is used to build a mapping between measurements and estimates of the system state. In many situations, however, such a training set it not available, nor is a complete model of the state dynamics from which to generate such a training set. The second category addresses the problems of identification and control for a deterministic but unknown nonlinear dynamic system. Again exploiting the universal approximator capability of many learning systems, these approaches try to minimize the output error by adjusting the parameters of their function approximations. Our problem is a bit different from these in that we want to do both estimation and identification for a nonlinear dynamic system that is stochastic and unknown.

1.1 Problem formulation

Consider the discrete-time nonlinear system given by

$$\begin{aligned} x_{k+1} &= f(x_k, \theta(x_k)) + b(x_k, \theta(x_k))u_k + Gw_k && (1.1) \\ z_k &= h(x_k) + v_k && (1.2) \end{aligned}$$

where $x_k \in \Re^n$ is the state of the system, $u_k \in \Re^m$ is a known driving signal, $z_k \in \Re^p$ is the measurement, $\theta(x_k) \in \Re^r$ is the unknown state-dependent parameter vector, and w_k and v_k are process and measurement noises, respectively (assumed to be zero mean white Gaussian sequences). In addition, f and b are possibly nonlinear time-invariant functions describing the plant model. In our problem, these functions are only partially known, with the unknown portion characterized by $\theta(x)$. Finally, h is a known function describing the sensor model. The reason why this function must be known will become evident in

Chapter 4 when we discuss blending.

As is discussed in Chapter 2, in the absence of model uncertainty (i.e., when the mapping $x \mapsto \theta(x)$ is completely known), a problem of great interest is the design of filters that give estimates of the state $\hat{x}_k$ at each time k, based on the observations z_ℓ up to time k, $\ell = 0, \ldots, k$.

In the presence of model uncertainty, however, an additional goal, and a necessary one for achieving the original state estimation objective, is the identification of the unknown mapping $\theta(x)$. The case of a linear mapping with unknown constant parameters (i.e., $f(x_k, \theta(x_k)) = \theta\, x_k$) has been widely treated in the literature [12, 13, 22]. In this thesis we focus on systems whose model uncertainties are nonlinear functions of the state.

Figure 1.1 displays a schematic of the algorithm designed to address the problem of state estimation and system identification for a partially unknown nonlinear time-invariant system. The *learning system* encapsulates "remembered" state-dependent knowledge of the unknown portion of the dynamics $\theta(x)$. The learned mapping $\hat{\theta}_{LS}$ is then fed to two filters to help in estimation: first, to an *augmented extended Kalman filter* (AEKF) that provides refined knowledge of the unknown dynamics by estimating a correction term $\delta\theta$ as well as the state; and second, to an *extended Kalman filter* (EKF) that provides state estimates if the AEKF is not working well. The two filters also take the known driving signal u and the sensor measurements z produced by the *plant* as inputs. Finally, the estimates of each filter are fed to the *blending* block that combines the two sets of estimates to produce final state and parameter estimates, $\hat{x}_B$ and $\hat{\theta}_B$. These estimates are then used in the next step of the state estimation problem and to further refine the learning system mapping. Because we are using Kalman filters as our recursive estimators, we refer to the overall estima-

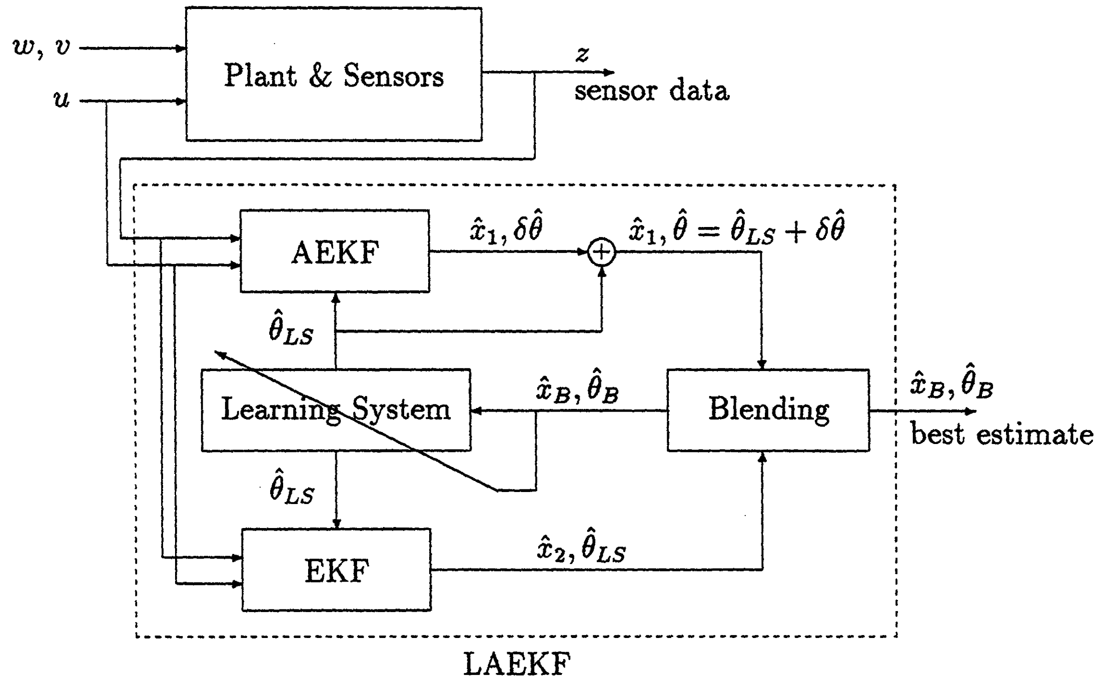
w, v
u
Plant & Sensors
z
sensor data
AEKF
$\hat{x}_1, \delta\hat{\theta}$
$\hat{x}_1, \hat{\theta} = \hat{\theta}_{LS} + \delta\hat{\theta}$
$\hat{\theta}_{LS}$
Learning System
$\hat{x}_B, \hat{\theta}_B$
Blending
$\hat{x}_B, \hat{\theta}_B$
best estimate
$\hat{\theta}_{LS}$
EKF
$\hat{x}_2, \hat{\theta}_{LS}$
LAEKF

Figure 1.1: Block diagram of learning augmented estimator

tion system as a *learning augmented extended Kalman filter*, or LAEKF. This architecture is discussed more fully later in the thesis, as is the design of each of the subsystems.

1.2 Organization of the thesis

In Chapter 2 a brief review is given of estimation theory. We first discuss the general problem of estimation of probabilistic quantities. We then address the case of state estimation for linear systems and discuss the Kalman filter algorithm. From there we move into the more general, and less tractable, nonlinear estimation framework. We conclude this section with a discussion of the assumptions that lie behind the design of the suboptimal, but implementable, extended Kalman filter (EKF). In both the linear and nonlinear discussions we assumed full knowledge of the system dynamics. In the final section of Chapter 2 we give a description of how the EKF can be used to address the joint problems of state estimation and system identification — a development resulting in the so called augmented extended Kalman filter (AEKF).

Chapter 3 gives some background on learning systems. Issues that face the designer of such systems are discussed. Following this general discussion the particular learning system used in the thesis is described, as are some learning issues that are raised by our specific application.

Chapter 4 formulates anew the problem of the thesis. Based on the material developed in Chapters 2 and 3, a full description of the learning augmented estimation architecture can finally be given. In addition, the question of combining the information contained in the learning system's mapping, and the new information entering the filters via observations, is addressed and resolved in the development of the blending system.

In Chapter 5 several applications of the learning augmented architecture are presented. Performance of the LAEKF is compared with that of nonlearning approaches to the estimation and identification problem, as well as with the state estimation performance of nonlinear filters that are designed with full knowledge of the nonlinear dynamics. It is this latter performance that the learning augmented system should approach as it identifies the unknown portion of the model with increasing accuracy.

Finally, Chapter 6 presents conclusions emanating from this research and ideas on future directions that might be pursued.

Chapter 2

State Estimation for Dynamic Systems

Nature provides abundant examples of situations where one wishes to infer information about a certain object or process from observation of a distinct, but related, object or process. An example might be a mountaineer's prediction of future weather based upon his or her observations of local conditions (clouds, winds, barometric pressure, temperature) throughout the day, or a bat's hunting method where the bat determines the distance and direction to its prey by chirping and listening to the echoes. While the mountaineer faces the problem of attempting to infer global information from sparse observations, the bat faces the problem of translating its echoed information into distances. In both cases the relationship of the received information to the desired knowledge may be unclear and the received information may be corrupted (by local fluctuations in the mountaineer's case, by extraneous echoes and other noises in the bat's). It is often useful to cast the desired knowledge, the observed data, and the relationship between, into a probabilistic framework. Then reasonable statements can be made about the desired knowledge based upon the observations.

In Section 2.1 general approaches to probabilistic estimation will be presented. Section 2.2 specializes these approaches to linear dynamic systems and

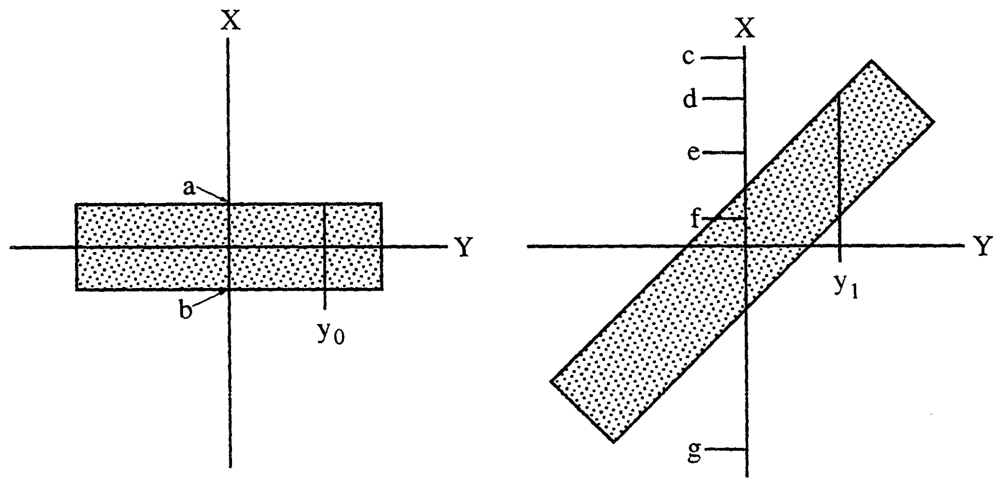

Figure 2.1: Two examples of joint densities, $p_{X,Y}(x, y)$

discusses the Kalman filtering algorithm. Section 2.3 generalizes this discussion to nonlinear systems, providing insight into both the linear and nonlinear cases. Finally, Section 2.4 provides a brief introduction to the joint problems of state estimation and system identification.

2.1 Bayesian estimation

Suppose that the unknown quantity in which we are interested, X, is well described as a probabilistic quantity. In that case all a priori information about X is contained within its probability density function, $p_X(x)$. Now say that some other data related to the unknown quantity is gathered, and call this data Y. We would then like to refine our information about X based on the new data. The quantity in which we are now interested is the conditional density, $p_{X|Y}(x|y)$.

To help understand why $p_{X|Y}(x|y)$ is the right thing to look at, we present two examples of joint densities $p_{X,Y}(x, y)$ in figure 2.1. We are interested in the distribution of X conditioned on measurements of Y. In the figure both densities are two-dimensional and uniform (i.e. $p_{X,Y}(x,y) = c$ in the shaded region and zero elsewhere, where $c = 1/(\text{area of shaded region})$). On the left we see that the marginal density, $p_X(x)$ is the same as $p_{X|Y}(x|y_0)$ $\forall\, y_0$ within the shaded region. Thus, given any possible observed values (i.e., within the shaded region) of Y, we gain no knowledge about X. This means that observations of Y are useless as they are not related to X and we should observe something else. In the plot on the right a different situation prevails. Given no information about Y, the random variable X is constrained to lie between g and c, i.e. $g < X < c$. However, given $Y = y_1$, we see that the conditional probability $p_{X|Y}(x|y_1) \neq p_X(x)$, rather it is uniformly distributed between f and d. Thus, the conditional probability gives us refined, seemingly sensible, information about X.

Now, say that instead of looking at densities, we want to derive a single number, a "best" guess of the quantity X based on the observations y, i.e. $\hat{X}(y)$. To determine whether a guess is "good," we must first have some measure of quality. Such a measure is a cost function that penalizes the difference between the true value of x and the estimate. Given a cost function, we can then determine our estimator by finding the function that minimizes the expected value of the cost over the conditional distribution of X given Y. That is

$$\begin{aligned} \hat{X}(y) &= argmin_a \mathsf{E}[C(x,a)] && (2.1) \\ &= argmin_a \int_{-\infty}^{\infty} C(x,a) p_{X|Y}(x|y) dx. && (2.2) \end{aligned}$$

There are many possible choices of cost function; all of which generally result in different estimators. One choice is to penalize the absolute error $C(x,a) = |x - a|$. This choice yields the *minimum absolute error* (MAE) estimator. Another is to seek the maximum of the conditional density, i.e., $C(x,a) = \delta(x - a)$, where $\delta(\cdot)$ is a dirac delta function. This cost function yields the *maximum a posteriori* (MAP) estimator.

One of the most common cost function is the quadratic, i.e. $C(x,a) = (x-a)^2$. This is the cost criterion we will focus on below.[1] Given the quadratic cost criterion, we want to solve the following problem

$$\hat{X}(y) = argmin_a \int_{-\infty}^{\infty} (x-a)^2 p_{X|Y}(x|y)dx. \tag{2.3}$$

Taking the derivative of the expected cost with respect to a — which we can easily do since the cost criterion is a smooth polynomial — and setting the result equal to zero, we get

$$\begin{aligned} -2a\int_{\infty}^{\infty}(x-a)p_{X|Y}(x|y)dX &= 0 \\ a\int_{-\infty}^{\infty}p_{X|Y}(x|y)dx &= \int_{-\infty}^{\infty}xp_{X|Y}(x|y)dx \\ a &= \mathsf{E}[X|y]. \end{aligned} \tag{2.4}$$

Since the function to be minimized is convex in a, any local extremum is global, and we can take another derivative to show that this extremum is a minimum. Thus, we have derived the estimator that globally minimizes the quadratic cost criterion,

$$\hat{X}(y) = \mathsf{E}[X|y]. \tag{2.5}$$

Returning to our example of figure 2.1, we see that on the left $\mathsf{E}[X] = 0$, and $\mathsf{E}[X|y_0] = \mathsf{E}[X] = 0 \ \forall\, y_0$ in the shaded region. So, just as we saw in the discussions of conditional densities, observation of Y gives us no additional information about X. Again in the plot on the right a different situation prevails. Given no information about Y, $\mathsf{E}[X] = 0$. However, given $Y = y_1$ we see that $\mathsf{E}[X|y] = e = \frac{d+f}{2}$. And again, as in the discussion of joint densities, observation of Y gives us refined knowledge of X.

This is the *Bayes Least-Squares* estimator, an estimator that has many nice properties. It is unbiased, $\mathsf{E}[X - \hat{X}(y)] = \mathsf{E}[X] - \mathsf{E}[\mathsf{E}[X|y]] = \mathsf{E}[X] - \mathsf{E}[X] = 0$.

[1]The squared form of the last equation reveals that in this discussion we will be dealing with scalar quantities, though the ideas extend to the multi-dimensional case.

In the general multi-dimensional case it is uncorrelated with any vector-valued function of the data, i.e. $\mathsf{E}[(\hat{X}(y) - X)g^T(y)] = 0$. A corollary of this last property is that the Bayes estimator is the estimator that yields the minimum mean-squared error. All these are desirable properties of estimators. The difficulty is that equation (2.5) is often a highly nonlinear function of the data that is not easily computable. A case where equation (2.5) is easily computable is the case where X and Y are jointly Gaussian. In that case the bayesian estimator is a linear function of the observations,

$$\hat{X}(y) = \mathsf{E}[X] + \Lambda_{XY}\Lambda_{YY}^{-1}(y - \mathsf{E}[Y]) \tag{2.6}$$

where $\Lambda_{XY} = \mathsf{E}[(X - \mathsf{E}[X])(Y - \mathsf{E}[Y])]$, $\Lambda_{YY} = \mathsf{E}[(Y - \mathsf{E}[Y])(Y - \mathsf{E}[Y])]$ and the estimate generated is the *linear least squares estimate* (LLSE). This estimate can also be calculated for non-Gaussian distributions, but in general it is then not equal to the Bayesian estimate.

2.2 State estimation for linear systems

In the last section the central idea was that the conditional probability density $P_{X|Y}(x|y)$ contained the information in which we were most interested. In this section, the unknown X evolves in time as a Markov process, and we want to make sequential estimates. Given knowledge of how X evolves in time, an analogous idea to the last section is to determine the conditional density of the unknown at the next time step based on all the observations up to the previous time step. We then to condition this density on the next observation, before finally determining the estimate based on this updated conditional density.

The unknown we wish to estimate is the state of a dynamic system where the evolution of the state with time is described by the following relations,

$$x_{k+1} = A_k x_k + G_k w_k \tag{2.7}$$

$$z_k = C_k x_k + v_k \tag{2.8}$$

where $x \in \Re^n$ is the state of the system; $z \in \Re^m$ are the observations; $A \in \Re^{n \times n}$ is the state-transition matrix, describing the evolution of the state with time; $G \in \Re^{n \times n}$ is the process-noise intensity matrix, describing how randomness enters into the state evolution; $C \in \Re^{m \times n}$ is the observation matrix, describing the linear combination of states measured by the sensors; and w_k and v_k are zero-mean, uncorrelated, white random processes with covariances Q_k and R_k respectively. In addition, they are uncorrelated with the initial state estimate $\hat{x}_0$ that has covariance P_0^+.

If we again choose the quadratic cost criterion we can find the *linear least squares estimator* (LLSE) for X based on the observations z. This is the estimator that minimizes (2.3) subject to the additional constraint that the estimate $\hat{X}(z)$ be linear in z. The form of the LLSE was given by equation (2.6). It turns out that given the relations (2.7) and (2.8) the calculation of the linear least squares estimate can be implemented in a recursive manner. The algorithm that does this is the Kalman filter.

The form of the Kalman filtering algorithm is as follows where $\hat{x}_k^-$ is the "predicted" state estimate of x_k based on the observations $z_0, \ldots z_{k-1}$, and $\hat{x}_k^+$ is the "updated" state estimate of x_k based on the observations $z_0, \ldots z_k$. The same notation holds for the error covariance, P.

Prediction:

$$\hat{x}_{k+1}^- = A_k \hat{x}_k^+ \tag{2.9}$$

$$P_{k+1}^- = A_k P_k^+ A_k^T + G_k Q_k G_k^T \tag{2.10}$$

Gain:

$$K_{k+1} = P_{k+1}^- C_{k+1}^T (C_{k+1} P_{k+1}^- C_{k+1}^T + R_{k+1})^{-1} \tag{2.11}$$

Update:

$$\hat{x}_{k+1}^{+} = \hat{x}_{k+1}^{-} + K_{k+1}(z_{k+1} - C_{k+1}\hat{x}_{k+1}^{-}) \tag{2.12}$$

$$(P_{k+1}^{+})^{-1} = (P_{k+1}^{-})^{-1} + C_{k+1}R_{k+1}^{-1}C_{k+1}^{T} \tag{2.13}$$

The Kalman filtering algorithm is a recursive algorithm divided into two parts. First, a *predicted* estimate of the state at time $k+1$, or $\hat{x}_{k+1}^{-}$, is calculated using knowledge of system dynamics, A_k, and the estimate at time k, $\hat{x}_k^{+}$. Likewise, the error covariance of the predicted estimate is a function of the plant dynamics, the error in the last estimate, P_k^{+}, and the process noise covariance, Q_k. Second, based on these extrapolations, the observation function, C_{k+1}, and the sensor noise covariance, R_{k+1}, a *gain* term is computed. Finally, using both the extrapolated quantities and the gain, the estimate $\hat{x}_{k+1}^{-}$ is *updated* to $\hat{x}_{k+1}^{+}$ based on the new observation z_{k+1}. The error covariance P_{k+1}^{-} is also updated at this point, but its update is not a function of the new observation.

Insight into the algorithm is gained by examining the form of each of the equations (2.9-2.13). The state prediction equation (2.9) reveals that extrapolation of the state estimate is based solely on system dynamics. This makes some intuitive sense as between measurements we receive no additional information nor can we predict what the process noise will be because it's white. Additionally, we know that the speed of the system dynamics is be controlled by the A matrix; the same thing is happening in (2.9) — the difference between $\hat{x}_k^{+}$ and $\hat{x}_{k+1}^{-}$ is determined by the A matrix.

If we now examine equation (2.10), we will gain an understanding of how randomness enters and propagates in the system. We will turn to the propagation of uncertainty first. The first term in (2.10) is a propagation of the error covariance at the last step and is a function of A. Depending on system stability, $A_kP_k^{+}A_k^{T}$ can be larger or smaller than P_k^{+}. The second term in (2.10) is always non-negative. Because of this non-negativity, the process noise can be viewed as a constant source of uncertainty in the system.

The expression for the gain (2.11) contains two terms of interest, P_{k+1}^- and R_{k+1}. The first is the covariance of the predicted estimate and is a measure of the quality of information contained within the prediction. The second term is the covariance of the observation noise, and is a measure of the quality of new information coming into the system by way of the sensors. The Kalman filter blends these two sources of information in a statistically optimal way. Note that if R is very small, i.e. if the sensor measurements are practically free of noise, and if C is invertible, the gain term would look something like C^{-1}, simply inverting the observation matrix. We will get back to this when we discuss the state update equation (2.12). Alternately, if $P_{k-1}^+ = 0$, and there is no process noise, $Q_{k-1} = 0$, then the state prediction is perfect and the gain would go to zero — i.e. no information from the sensors would be needed.

Examining the state update equations (2.12), we see that the new estimate is derived from the best guess of the current state based upon all previous measurements, $\hat{x}_{k+1}^-$, and some function of the new information contained in the current observation. This new information — the difference between the predicted and the actual measurement — is called the residue, $(z_{k+1} - C\hat{x}_{k+1}^-)$. Now, if we again consider the situation where the sensor noise is very small and C is invertible, the state update equation would reduce to,

$$\hat{x}_{k+1}^+ = C_{k+1}^{-1} z_{k+1}$$

that is, a simple inversion of the sensor model and a cancellation of the predicted estimate. So although $\hat{x}_{k+1}^+$ is conditioned on $z_0, \ldots z_{k+1}$, in this case only z_{k+1} is used.

Finally, when considering equation (2.13) one should realize that the inverse of a covariance matrix is a measure of information. In this way we can see that each new measurement adds some amount of information to the knowledge one has of the state. While the process noise pumps uncertainty into the system, the measurements removes it. If we now consider equations (2.10) and (2.13), we can see that the error covariance matrix evolves in time independently of the state trajectory. That is, P_k^- and P_k^+ are not functions of z_k and so can be calculated off-line, thus significantly lowering the on-line computational burden of the Kalman filter.

For purposes of comparison with results in the next section we present the continuous-time Kalman filtering equations below,

$$\begin{aligned} \dot{\hat{x}}_t &= A_t\hat{x}_t + K_t[z_t - C_t\hat{x}_t] & (2.14)\\ \dot{P}_t &= A_tP_t + P_tA_t^T + G_tQ_tG_t^T - K_tR_tK_t^T & (2.15)\\ K_t &= P_tC_t^TR_t^{-1}. & (2.16)\end{aligned}$$

Note that because sensor data is now continuous in time, the prediction and update steps collapse into one set of equations. Although the form of the algorithm changes, the underlying intuition remains the same. The state estimate derivative is of the exact form of the discrete-time state estimate update where the state estimate is corrected by a linear function of the residue. As in discrete-time, the error covariance increases proportionally to the strength of the process noise and decreased proportionally to the information content of the observations (each gain factor K_t contains a R^{-1}). Finally, the general form of the gain as a ratio of error covariance to measurement covariance remains the same.

2.3 State estimation for nonlinear systems

While the problem of state estimation for linear dynamic systems with white noise processes is tractable, for more general dynamic systems it is quite difficult.

In this thesis we are concerned with nonlinear dynamic systems and the estimation techniques applied to such systems are necessarily approximate. One may begin from the linear context as developed above, and try to extend the ideas to nonlinear problems through linearization and other techniques. We prefer to take the opposite tack and instead outline the theory from a general nonlinear framework, thereby more clearly seeing what specializations and assumptions are needed to arrive at usable, though suboptimal, filtering algorithms. This section will present a general overview of this complex theory, often presenting deep results without proof, but with the intention of providing some insight and intuitive feeling for the subject.[2]

In section 2.3.1 we will discuss the mathematics behind the evolution of continuous-time stochastic processes without observations, while in 2.3.2 we will discuss how to condition these processes on observations. Finally, in 2.3.3 we will discuss implementable filtering algorithms based on expressions for the evolution of the conditional probabilistic density of the state in time, where the conditioning is based on the observations received.

2.3.1 State evolution without observations

As was discussed in the introduction, in many cases the temporal evolution of natural phenomena or man-made devices can be well modeled by dynamic systems. As opposed to Section 2.2, in more general cases these dynamics consist of nonlinear stochastic differential equations

$$dX_t = m(X_t, t)dt + \sigma(X_t, t)dW_t \tag{2.17}$$

[2]A more complete exposition can be found in [21], upon which much of this development is based. In addition, while in the last section the discussion was carried out in discrete-time, in this section it will be easier to consider the continuous-time scalar case.

where if X_0 is the initial condition, m and σ are real-valued function on $\Re \times (0, \infty)$ with $\sigma(x,t) > 0$, and $(W_t; t \geq 0)$ is a Wiener process independent of X_0. The reason for expressing this equation in this form, rather than differential form is that Wiener processes are continuous, hence dW_t and dX_t exist, but not differentiable, so $\frac{dW_t}{dt}$ does not. Equation (2.17) can alternately be expressed in integral form as

$$X_t = X_0 + \int_0^t m(X_s, s)ds + \int_0^t \sigma(X_s, s)dW_s. \tag{2.18}$$

Subject to certain regularity conditions, (2.17) and (2.18) give rise to well defined processes, see [21]. We will assume such conditions hold and simply note that $m(X_t, t)dt$ and $\sigma(X_t, t)dW_t$ are often referred to as the drift and variance terms.

Conditioned on the state at a certain time, X_τ, any future state X_t can be generated without knowledge of past data $(X_t; t < \tau)$ since $(W_t - W_\tau; t > \tau)$ is independent of the past. Thus,

$$X_t = X_\tau + \int_\tau^t m(X_s, s)ds + \int_\tau^t \sigma(X_s, s)dW_s. \tag{2.19}$$

Random processes that have such a property are referred to as *Markovian*, and the property leads to a very nice statistical description. Given any finite collection of time instants, $0 \leq t_1 < t_2 < \ldots < t_n < \infty$, the joint density of the corresponding time samples, $x_{t_1}, x_{t_2} \ldots x_{t_n}$, is given by,

$$p_{X_{t_1},\ldots X_{t_n}}(x_1, \ldots x_n) = p_{X_{t_1}}(x_1) \prod_{k=2}^{n} p_{X_{t_k}|X_{t_{k-1}}}(x_k | x_{k-1}). \tag{2.20}$$

So, given expressions for the density of the initial condition, and for the transition densities $p_{X_t|X_s}(x_t|x_s)$, $t \geq s > 0$, any finite dimensional distribution of sample points can be obtained.

It happens that the transition densities satisfy the partial differential equation known as the *Fokker-Planck Equation* or *Kolmogorov's forward equation* *of diffusion*, as follows, where the $*$ represents the complex conjugate and we present results only for the scalar case.

$$\frac{\partial}{\partial t} p_{x_t|x_\tau}(X_t|X_\tau) = A_t^* p_{x_t|x_\tau}(X_t|X_\tau) \tag{2.21}$$

where for any function $f(x)$, A is the Fokker-Planck operator given by

$$A_t^* f(x) \equiv \frac{1}{2}\frac{\partial^2}{\partial x^2}[\sigma^2(x,t)f(x)] - \frac{\partial}{\partial x}[m(x,t)f(x)]. \tag{2.22}$$

In the end our objective is to derive information about the distribution of the random process X_t at any instance. Since knowledge of all density moments is equivalent to knowledge of the probabilistic density, it may be more productive to try to find expressions for the evolution of the moments rather than a single expression for the entire density. We will pursue this idea hereafter.

Starting from either the Fokker-Planck equations or (2.18), one can derive the following equation for the time derivative of the expectation of any function of the state, $f(x)$,

$$\frac{d}{dt}\mathsf{E}[f(X_t)] \;=\; \mathsf{E}[m(X_t,t)\left\{\frac{d}{dX_t}f(X_t)\right\} + \tfrac{1}{2}\sigma^2(X_t,t)\left\{\frac{d^2}{dX_t^2}f(X_t)\right\}] \tag{2.23}$$

However, since $f(X_t)$ can be any function of X_t, we can choose $f(X_t) = X_t$ and evaluate (2.23) to derive the time derivative of the mean (μ) of the density,

$$\dot{\mu}_t = \frac{d}{dt}\mathsf{E}[X_t] = \mathsf{E}[m(X_t,t)] \tag{2.24}$$

To derive a similar expression for the covariance (ν) of the density, we proceed by exploiting the result in (2.23) with $f(X_t) = X_t^2$ as follows where $\nu_t = \mathsf{E}[X_t^2] - \mu_t^2$,

$$\begin{aligned}\dot{\nu}_t &= \frac{d}{dt}\mathsf{E}[X_t^2] - 2\mu_t\dot{\mu}_t \\ &= \mathsf{E}[2X_t m(X_t,t) + \sigma^2(X_t,t)] - 2\mu_t\mathsf{E}[m(X_t,t)] \\ &= 2\mathsf{E}[(X_t - \mu_t)(m(X_t,t) - \mathsf{E}[m(X_t,t)])] + \mathsf{E}[\sigma^2(X_t,t)] \\ &= 2Cov(X_t, m(X_t,t)) + \mathsf{E}[\sigma^2(X_t,t)] \end{aligned} \tag{2.25}$$

As seen in (2.21), it is possible to derive the general equations needed to propagate the probability density of the state. However, (2.24) and (2.25) demonstrate that even the propagations of only the first two central moments are not easily solved as the expectations depend upon the two nonlinear functions $m(X_t, t)$ and $\sigma(X_t, t)$, whose expectations in turn rely on further nonlinear functions, etc.

For linear systems, however, the result is quite tractable. In this case, $m(X_t, t) = A_t X_t$ and $\sigma(X_t, t) = G_t$. Substituting back into (2.24) and (2.25) one gets

$$\dot{\mu}_t = A_t \mu_t \tag{2.26}$$

$$\dot{\nu}_t = A_t \nu_t + \nu_t A_t + G_t^2. \tag{2.27}$$

If we remember that there are no observations, then these equations are obtainable from the linear continuous-time Kalman filter algorithm, (2.14) and (2.15), specialized to the scalar case with $K_t = 0$.

2.3.2 Observations and nonlinear filtering

Thus far we have only discussed the propagation of probability densities for dynamic systems without observations. Knowledge how the density evolves allows us to estimate the state at any time. However, if we have some other information about the state, sensor data for instance, we should condition our density on this new information. If we were to then look for the mean of this conditioned density, we would arrive at the Bayesian estimate. In our case the additional information is from integrated observations that are of the form

$$dZ_t = h(X_t, t)dt + R_t^{1/2} dV_t \tag{2.28}$$

where $(V_t, t > 0)$ is a standard Wiener process independent of X_0 and $(W_t, t > 0)$.

Now, as discussed above, we would like to be able to estimate the conditional expectation of any function of the process $f(X_t)$ based on all the observations up to time t. Since we are seeking the Bayesian minimum variance unbiased (MVU) estimate, we want to calculate the conditional expectation of X_t, where the conditioning is represented by an overbar,

$$\begin{aligned} \overline{f(X_t)} &= \mathsf{E}[f(X_t)|Z_0^t] \\ &= \int_{-\infty}^{\infty} f(x) q_{X_t}(x_t) dx \end{aligned} \tag{2.29}$$

where $q_{X_t}(x_t)$ is the conditional density of X_t given Z_0^t, the observations from time 0 to time t.

Inspection of (2.29) reveals that if one had an expression for the evolution of the conditional density in time, one could compute a simple integral to get the MVU estimate at any time. This expression would be a function both of the state diffusion (2.17) and the observations (2.28). Indeed, such a description of the evolution of the conditional density has been derived and is termed the *Kushner Equation.* Within regularity — again refer to [21] — $q_{X_t}(x_t)$ satisfies the following stochastic partial differential equation

$$dq_{X_t}(x_t) = A_t^* q_{X_t}(x_t)dt + q_{X_t}(x_t)[h(x_t, t) - \overline{h(X_t, t)}]R_t^{-1} dI_t \tag{2.30}$$

where again the overbar denotes the mean of the function conditioned on Z_0^t. The process $(I_t, t > 0)$ is the nonlinear innovations process as follows

$$I_t = Z_t - \int_0^t \overline{h(X_s, s)} ds \tag{2.31}$$

$$dI_t = dZ_t - \overline{h(X_t, T)} dt \tag{2.32}$$

The equations (2.29) and (2.30) can be combined to obtain a stochastic differential equation for the evolution of $\overline{f(X_t)}$,

$$d\overline{f(X_t)} = \overline{A_t f(X_t)}dt + \{\overline{f(X_t)h(X_t,t)} - \overline{f(X_t)}\,\overline{h(X_t,t)}\}R_t^{-1}dI_t \qquad (2.33)$$

It should be noted that if $h \equiv 0$ (i.e. back to the case of no observations), then the Kushner equation (2.30) reduces to the Fokker-Planck equation (2.21). However, these relations are fundamentally different in that the Kushner equation is stochastic while Fokker-Planck is deterministic.

For completeness we include a relationship that is analogous to (2.30), but is simpler and lends itself more easily to analytical work. This is the unnormalized conditional density given by the Zakai equation,

$$d\rho_{X_t}(x) = A_t^* \rho_{X_t}(x)dt + h(x,t)\rho_{X_t}(x)dZ_t \qquad (2.34)$$

where

$$q_{X_t}(x) = \frac{\rho_{X_t}(x)}{\int_{-\infty}^{\infty} \rho_{X_t}(x)dx}$$

The problem facing any real-world implementation of these equations is that, in general, they are of infinite dimensionality. The expression for the evolution of the conditional expectation of $f(X_t)$, (2.33), depends on knowledge of other quantities that themselves rely on nonlinear recursions. Unless at some finite point the equations close, any filter based on these equations will be of infinite order and therefore not implementable.

To better see under what conditions the equations do form a closed set, we examine in greater depth the two functions of the process in which we are most interested. First is the conditional mean of the state and second is the conditional mean squared error $P_t = \overline{X_t^2} - \overline{X}_t^2$. Substituting $f(x) = x$ into equation (2.33) we get

$$d\overline{X}_t = d\mathsf{E}[X_t|Z_0^t] = \overline{m(X_t,t)}dt + \{\overline{X_t h(X_t,t)} - \overline{X}_t\overline{h(X_t,t)}\}R_t^{-1}dI_t. \qquad (2.35)$$

To gain insight into this equation we will make a number of successive simplifications. For linear observations, $h(X_t, t) = C_t X_t$, and remembering that we are only dealing with the scalar case, we have

$$d\overline{X_t} = \overline{m(X_t, t)}dt + C_t P_t R_t^{-1} dI_t$$

where $P_t \equiv \overline{X_t^2} - \overline{X}_t^2$, the conditional variance of X_t given Z_0^t. The innovations term $C_t P_t R_t^{-1} dI_t$ is identical to that for the Kalman filter (2.14). Further simplification to a linear drift term $(m(X_t, t) = A_t X_t)$ leads to

$$dX_t = A_t \overline{X}_t dt + P_t C_t R_t^{-1} dI_t$$

which is identical in form to the state estimation equation of the continuous-time Kalman filter (2.14). There may yet be a difference between the filters, however, in that it is not yet clear whether P_t evolves independently of Z_0^t, as is the case in the Kalman filter formulation.

Because of the stochastic nature of the quantities with which we are dealing, we must turn to stochastic calculus when integrating the time derivatives of these processes. Because of this, we will simply state the time derivative of the error variance as

$$\begin{aligned} dP_t &= \{2\overline{(X_t - \overline{X}_t)} + \overline{\sigma^2(X_t, t)} - [\overline{X_t h(X_t, t)} - \overline{X}_t \overline{h(X_t, t)}]^2 R_t^{-1}\}dt \\ &+ \{\overline{(X_t - \overline{X}_t)^2 h(X_t, t)} - P_t \overline{h(X_t, t)}\} R_t^{-1} dI_t. \end{aligned} \tag{2.36}$$

Evidently this expression is quite complicated. We simplify to linear observations and drift term, and a process noise function independent of state $(\sigma(X_t, t) = G_t)$ to get

$$dP_t = \{2A_t P_t + G_t^2 + C_t^2 P_t^2 R_t^{-1}\}dt + \overline{(X_t - \overline{X}_t)^3} R_t^{-1} dI_t. \tag{2.37}$$

The first three terms are familiar from the continuous-time Kalman variance equation (2.15). But even with the assumption of a completely linear model, we see that the variance P_t is still dependent on the observations through the last cubic term. This is distinctly different from the Kalman filtering case. If, however, the assumption is made that the initial condition is jointly gaussian with the noise, then X_t is conditionally gaussian given Z_0^t. Because of the symmetry of the Gaussian distribution all its odd central moments are zeros, and thus $\overline{(X_t - \overline{X}_t)^3} = 0$. The variance now reduces to its familiar form (2.15),

$$dP_t = \{A_t P_t + P_t A_t + G_t G_t + C_t P_t R_t^{-1} P_t C_t\} dt. \tag{2.38}$$

Note again that while (2.37) is stochastic, (2.38) is deterministic, and so can be calculated off-line as mentioned in Section 2.2. This then is a fundamental difference between the linear and nonlinear cases. When considering the first two conditional central moments, in the linear case only the conditional mean (the estimate) is dependent on the observations, while in the nonlinear case both the mean and the error covariance are dependent on the observations. (For the linear case refer to equations (2.9)- (2.13) for discrete-time, and equations (2.14)-(2.16) for continuous-time).

2.3.3 Implementable nonlinear filters and approximations

As we have seen in the preceding discussion, it is only under very stringent conditions that the optimal filter equations are finite dimensional and therefore implementable. It is thus of great interest to see what approximations can be made to allow implementation more generally. Expanding the innovations term in equation (2.35) we have

$$\begin{aligned} d\overline{X}_t &= \overline{m(X_t,t)}dt + \{\overline{X_t h(X_t,t)} - \overline{X}_t \overline{h(X_t,t)}\} R_t^{-1} dZ_t \\ &+ \{\overline{X_t h(X_t,t)} - \overline{X}_t \overline{h(X_t,t)}\} R_t^{-1} \overline{h(X_t,t)} dt \end{aligned}$$

where again the overbar indicates the conditional mean, conditioned on Z_0^t.

To address the difficulties posed by the conditional expectations, $\overline{m}$, $\overline{Xh}$ and $\overline{h}$, we first suppose the estimation error $(\overline{X}_t - X_t)$ to be generally small and assume $m(x,t)$ and $h(x,t)$ are smooth functions of the state, x. In this case we can approximate the state transition and observation functions by their first order Taylor expansions

$$\begin{aligned} m(X_t,t) &\cong m(\overline{X}_t,t) + (X_t - \overline{X}_t)m'(\overline{X}_t,t) \\ h(X_t,t) &\cong h(\overline{X}_t,t) + (X_t - \overline{X}_t)h'(\overline{X}_t,t) \end{aligned}$$

where $m'(x,t) = \partial m(x,t)/\partial x$ and $h'(x,t) = \partial h(x,t)/\partial x$. If we take the conditional expectation of each expansion, noting that $E[X_t - \overline{X}_t | Z_0^t] = 0$, we have

$$\begin{aligned} \overline{m(X_t,t)} &\cong m(\overline{X}_t,t) \\ \overline{h(X_t,t)} &\cong h(\overline{X}_t,t) \end{aligned}$$

After substitution and simplification, we have the relation

$$d\overline{X}_t \cong m(\overline{X}_t,t)dt + P_t h'(\overline{X}_t,t)R_t^{-1}[dZ_t - h(\overline{X}_t,t)dt] \tag{2.39}$$

Since the system model described by m, h, and R_t and the observations are known, the only unknown in the expression is P_t; were that quantity calculable, then the equations could be implemented.

Applying the same approximations to equation (2.36) as well as expanding $\sigma^2(x,t)$ in its first order Taylor approximation, we get

$$\begin{aligned} dP_t &\cong \{2P_t m'(\overline{X}_t,t) + \sigma^2(\overline{X}_t,t) - P_t^2[h'(\overline{X}_t,t)]^2 R_t^{-1}\}dt \\ &+ \overline{(X_t - \overline{X}_t)^3} h'(\overline{X}_t,t)R_t^{-1}[dZ_t - h(\overline{X}_t,t)dt]. \end{aligned}$$

We see that we again run into the troubling third moment term. If, as before, we assume the errors are symmetrically distributed, this term drops out and we are left with

$$\frac{d}{dt}P_t = m'(\overline{X}_t,t)P_t + P_t m'(\overline{X}_t,t) + \sigma^2(\overline{X}_t,t) - [h'(\overline{X}_t,t)]P_t R_t^{-1} P_t [h'(\overline{X}_t,t)]. \tag{2.40}$$

These equations, (2.39) and (2.40), are the equations upon which the *extended Kalman filter* (EKF) design is based. Continual linearization must be performed about the state estimate to implement this filter. Thus, as well as being subject to the assumptions needed to justify the use of a Taylor expansion, the EKF is much more computationally intensive than the linear Kalman filter as the evolution of the error variance is dependent on the state trajectory of the system.

In summation then, we have seen that the extended Kalman filtering algorithm can be derived directly from the optimal nonlinear filtering equations based upon three key assumptions. The first is that the system functions are smooth so that a Taylor series can be used to approximate the functions. The second is that the estimation error is generally small so that only a first order Taylor series is needed. The third assumption is that the errors are symmetrically distributed so that the third order term, and all coupling to higher order statistics, drop out of the equations. Since the EKF is such a widely used tool, much work has been dedicated to analyzing it properties; for further reading see [1, 8, 12, 13].

As much of our work will be on systems that have continuous-time plants, but only discrete-time sensors, we will now list the extended Kalman filtering equations for such a system where measurements are collected once every T seconds.

System dynamics:

$$\dot{x}_t = f(x_t, t) + G_t w_t \tag{2.41}$$

$$z_{nT} = h(x_{nT}) + v_{nT} \tag{2.42}$$

Propagation:

$$\dot{\bar{x}}_t = f(\bar{x}_t, t) \tag{2.43}$$

$$\dot{P}_t = F(\bar{x}_t, t)P_t + P_t F^T(\bar{x}_t, t) + G_t Q_t G_t^T \tag{2.44}$$

Gain:

$$K_n = P_{nT}^- H^T(\bar{x}_{nT}^-)(H(\bar{x}_{nT}^-)P_{nT}^- H^T(\bar{x}_{nT}^-) + R_n)^{-1} \tag{2.45}$$

Update:

$$\bar{x}_{nT}^+ = \bar{x}_{nT}^- + K_n(z_{nT} - h(\bar{x}_{nT}^-)) \tag{2.46}$$

$$P_{nT}^+ = (I - K_n H(\bar{x}_{nT}^-))P_{nT}^- \tag{2.47}$$

where

$$F(\bar{x}_t, t) = \frac{\partial f(x_t, t)}{\partial x_t}\Big|_{x_t = \bar{x}_t} \tag{2.48}$$

$$H(\bar{x}_{nT}^-) = \frac{\partial h(x_{nT})}{\partial x_{nT}}\Big|_{x_{nT} = \bar{x}_{nT}^-} \tag{2.49}$$

where $w_t \sim (0, Q_t)$, $v_{nT} \sim (0, R_n)$, $x_0 \sim (\bar{x}_0, P_0)$ and $E[w_t v_{nT}^T] = 0 \; \forall \; n, t$. In addition, the notation $\hat{X}_{nT}^-$ refers to the state estimate just prior to the n^{th} measurement, and $\hat{X}_{nT}^+$ to the estimate just after the measurement. The notation is analogous for the error covariances, P_{nT}^- and P_{nT}^+.

2.4 Parameter estimation using an AEKF

As we have seen in the last two sections, methods of state estimation assume complete knowledge of the system dynamics. Such knowledge is not always available, however, and therefore some amount of system identification must be done in conjunction with state estimation. For this discussion we return to the case where the dynamics are linear,

$$\begin{aligned}\dot{x}_t &= A_t x_t + G_t w_t \\ z_{nT} &= C_{nT} x_{nT} + v_{nT}.\end{aligned}$$

If we knew A_t, G_t and C_{nT}, we could implement slightly modified versions (to account for the continuous-time plant) of equations (2.9-2.13) to get estimates of the state. However if, for example, we did not know the parameters of the A matrix, we would need an alternate approach. One such approach is to include the parameter in the original state vector as additional states and then do estimation on the augmented state. Then, if we had two original states, x_1 and x_2, our augmented state vector would be

$$x = [x_1\ x_2\ a_{11}\ a_{12}\ a_{21}\ a_{22}]^T$$

where the a_{ij}'s are the entries of the A matrix. If we wish to estimate this augmented state we need to use a nonlinear estimation technique as the state derivatives are no longer linear in the states, e.g. $(\dot{x}_{2,t} = a_{21,t}\, x_{1,t} + a_{22,t}\, x_{2,t})$. One method is to use an EKF as as described in (2.41-2.49). Because the nonlinear filter used is an EKF, and the state has been augmented with the unknown parameters, this approach is referred to as *augmented extended Kalman filtering* (AEKF). Before we can implement the nonlinear Kalman filtering algorithm, however, we must decide how to model the time derivatives of the unknowns $a \equiv [a_{11}\ \ a_{12}\ \ a_{21}\ \ a_{22}]^T$.

If the dynamics we are trying to identify are constant, we could say

$$\dot{a}_t = 0. \qquad (2.50)$$

If, on the other hand, the parameters are time-varying, i.e. $\dot{a}_t \neq 0$, we need to model this variation. One method is to model the parameters as polynomial functions of time and then estimate the time-invariant coefficients p_i, i.e.

$$a_t = p_0 + p_1 t + p_2 t^2 + \dots + p_m t^m \tag{2.51}$$

A problem with this method is that the a_{ij}'s may vary in a very non-polynomial way, resulting in a huge number of coefficients p_i that need to be identified. A way to avoid this dimensionality issue is to model the time variation of the parameters as a random walk. In this case (2.50) becomes

$$\dot{a}_t = w_{a,t} \tag{2.52}$$

where $w_a \sim N(0, Q_a)$. The strength of the noise Q_a corresponds to the speed of parameter variation in time. That is, if $Q_a = 0$ we reduce to the case of equation (2.50) where the parameters are constant. As Q_a increases, the bandwidth of the time variations that the parameters can track also increases. Evidently this noise modeling is a crucial user-determined aspect of the approach that can greatly influence the success of the identification technique. We will return to this issue when we discuss our approach to the joint tasks of state estimation and system identification.

Our dynamic system model can now be recast as follows,

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} A_t x_t \\ 0 \end{bmatrix} + \begin{bmatrix} G w_t \\ w_{a,t} \end{bmatrix} \tag{2.53}$$

$$z_{nT} = C_{nT} x_{nT} + v_{nT}. \tag{2.54}$$

And we see that we now have a nonlinear continuous-time plant with discrete measurements, and so we can use the EKF algorithm described in equations (2.41-2.49) to address this problem.

2.4.1 An application

Following the theoretical exposition given in the last section, we would like to demonstrate some of these ideas on a real system. We choose a three-state, tail-surface-controlled airframe.[3] We desire the airframe's lateral acceleration to follow a deterministic square-wave u_t of amplitude 10 ft/sec^2. The states are pitch rate q_t, lateral acceleration l_t, and tail deflection δ_t, which is limited to $\pm 30.5°$. The sensors yield full state measurements, but only periodically. The dynamics are as follows, where in this particular example process noise has been excluded

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} q_t \\ l_t \\ \delta_t \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & -100 \end{bmatrix}\begin{bmatrix} q_t \\ l_t \\ \delta_t \end{bmatrix} + \begin{bmatrix} 0 \\ -a_{23} \\ 100 \end{bmatrix} u_t \qquad (2.55) \\ z_{nT} &= x_{nT} + v_{nT} \qquad (2.56) \end{aligned}$$

To use the AKEF, the state is augmented with the unknowns, $a = [a_{11}\, a_{12} a_{13} a_{21}\, a_{22}\, a_{23}]^T$, resulting in the same dynamics as in (2.53) if $x = [q\, l\, \delta]^T$, i.e

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} q_t \\ l_t \\ \delta_t \\ a_t \end{bmatrix} &= \begin{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & -100 \end{bmatrix}\begin{bmatrix} q_t \\ l_t \\ \delta_t \end{bmatrix} + \begin{bmatrix} 0 \\ -a_{23} \\ 100 \end{bmatrix} u_t \\ 0 \end{bmatrix} \\ &\quad + \begin{bmatrix} 0 \\ 0 \\ 0 \\ Q_a \end{bmatrix} \qquad (2.57) \\ z_{nT} &= x_{nT} + v_{nT} \qquad (2.58) \end{aligned}$$

Figure 2.2 is a schematic of the overall estimation/identification/control system. The sensor noise is small, so the challenge is in the identification of the entries of the A matrix, not in the estimation of the state. The AEKF provides state and parameter estimates, both of which are used in controller design. The "first-order actuator" box models the elevator dynamics as a simple pole at $s = -100$. Control is implemented by a full state feedback (using the estimated state) pole-placement (using the estimated parameters) algorithm,

[3]This example is taken from Section 9.4 of [8]

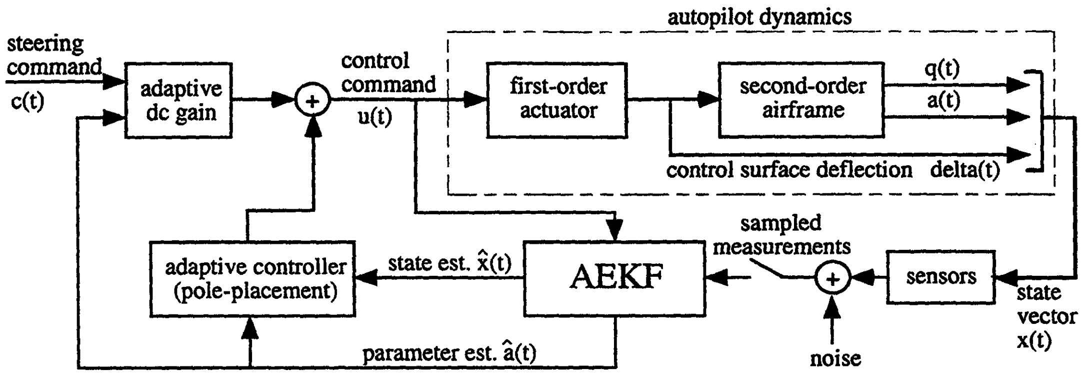

Figure 2.2: Adaptive controller, plant, and AEKF

where the poles are nominally placed at $s = -110$, and $-10 \pm j5$. In addition, the estimated parameters are used to determine a gain scaling at the input to achieve zero steady state error to a step input.

A typical simulation is shown in figures 2.3 and 2.4. Figure 2.3 shows the parameter identification performance of the time-varying parameters. Figure 2.4 shows the state estimation performance of the system, and the ability of the control system to track the desired trajectory. The a_{11} and a_{22} estimates track especially poorly. This is explained by the dynamics of the airframe as those parameters are relatively unobservable. Because of this decoupling, however, these parameters have little impact on the system response, and so the inability to identify them is of diminished importance. The reason for the large transient over the first second of the a_{12} estimate is unclear as it was a feature that did not always occur and seemed to be relatively dependent upon initial conditions. All the parameter estimates can be seen to have a vaguely step-like character. This is a function of the control input. Generally it was when a sharp change in the control input occurred that dynamics were excited and the measurements contained useful information about the dynamics of the airframe. Between such shocks the parameter estimates could not track the changing parameters even

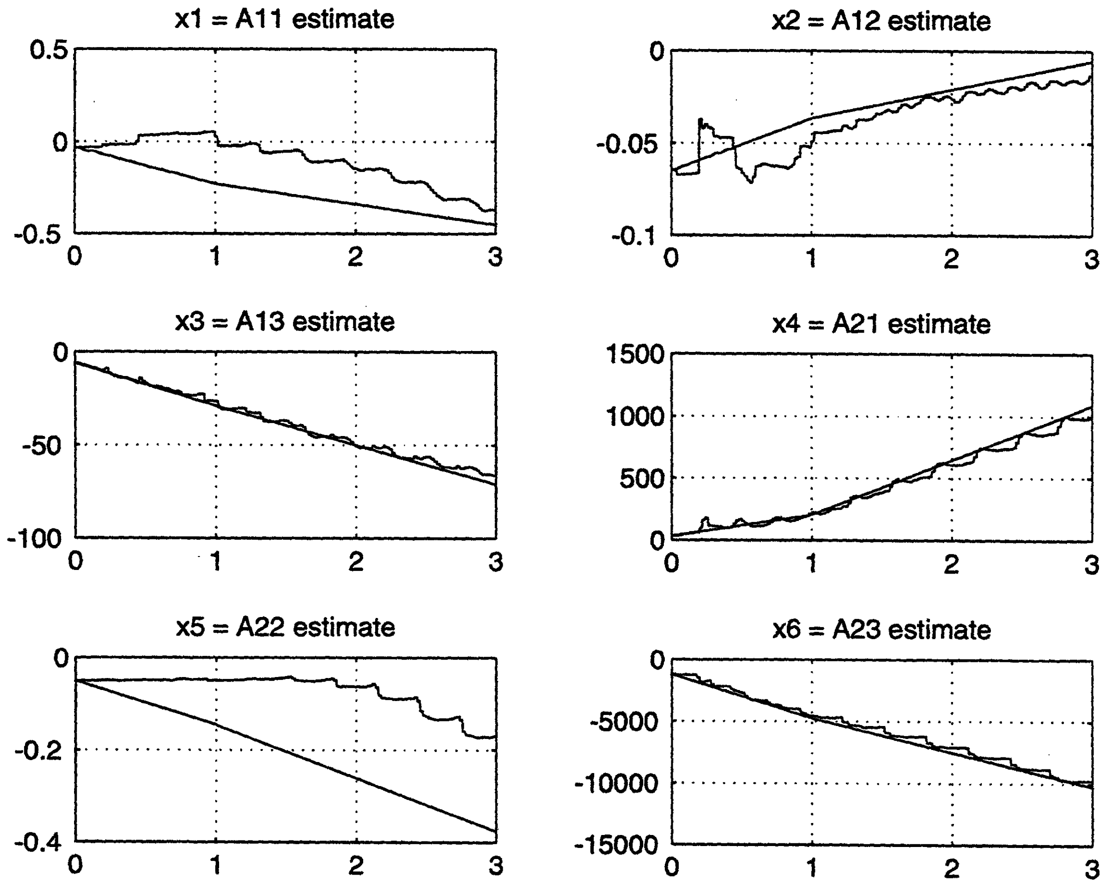

Figure 2.3: Parameter identification performance

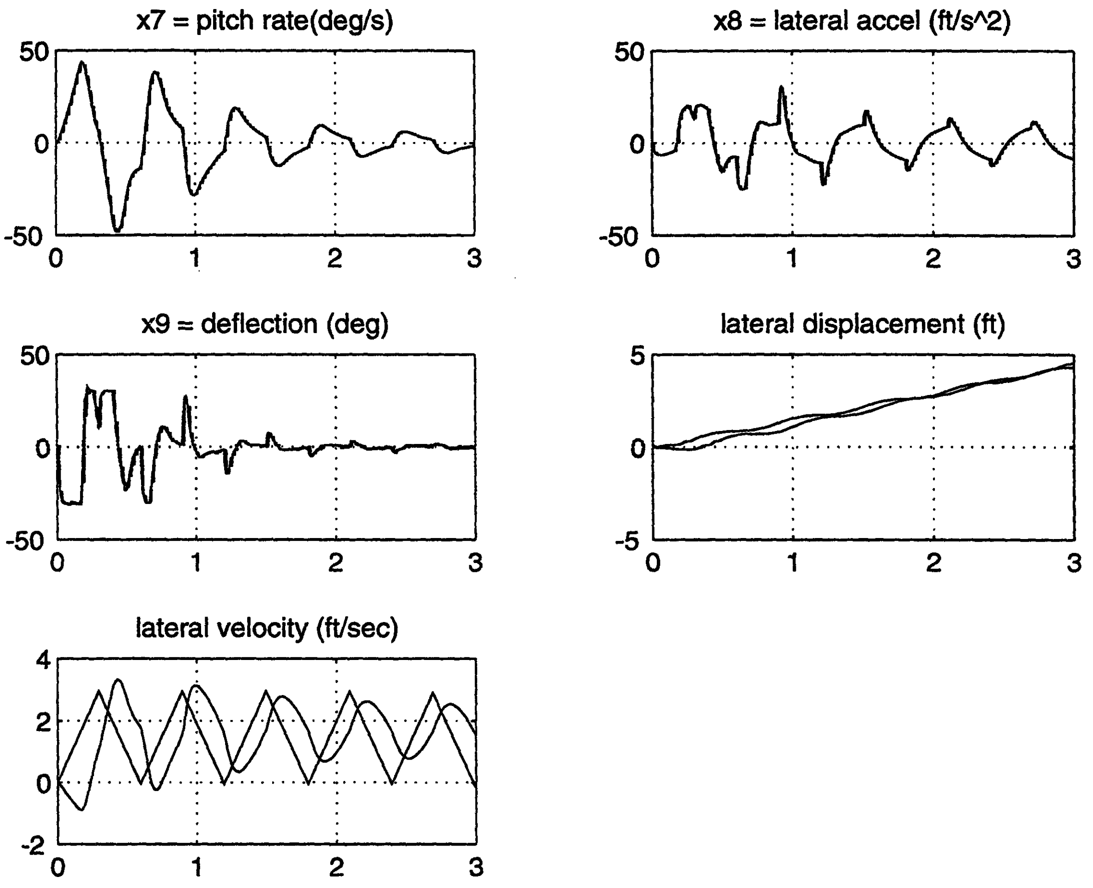

Figure 2.4: State estimation performance

though the state estimates tracked well. This is an especially important effect when considering system identification schemes based on on-line observations. If the system dynamics are not continually excited by the driving signal then it will not be possible to continually identify the system (the requirement of continual excitation is often referred to as *persistent excitation*). A trivial example of a non-persistently exciting signal is a zero input. Note that in figure 2.4 the actual velocity is the curve that lags the triangle wave. This lag is a function of what we are tracking — lateral acceleration. Some error integrates up in the initial transient until the controller gets going, and since we only care about error in the lateral acceleration, the error in the velocity never gets corrected.

In the example we have presented there was very little sensor and no real process noise. To make the relationship between noise levels and identification performance more clear we return to the linear Kalman filtering equations (2.9-2.13). The time variation of the unknown dynamics is modeled as a disturbance (through the Q_a matrix, see equation (2.52)). By determining the transfer function from disturbances to estimation error we can develop a feeling for what kind of disturbances the system filters out and what kind can be passed through. This will give us an idea of the frequencies that can be tracked by this system identification scheme. Under linear time-invariant assumptions, and after a little fiddling, we can come up with,

$$\begin{aligned} \tilde{x}_k^+ &= x_k - \hat{x}_k^+ & (2.59) \\ &= x_k - \hat{x}_k^- - K_k(Y_k - C\hat{x}_k^-) & (2.60) \\ &= (A - K_kCA)\tilde{x}_{k-1}^+ + (I - K_kC)w_{k-1} - K_kv_k. & (2.61) \end{aligned}$$

If we assume we have reached steady state so that $K_k = K$ and if, for the moment, we view w_k and v_k as deterministic quantities so that we can take their Z-transform to determine the transfer function, we get

$$(I - (A - KCA)z^{-1})\tilde{X}^+(z) = (I - KC)z^{-1}W(z) - KV(z) \qquad (2.62)$$

which results in two transfer function,

$$\frac{\tilde{X}^+(z)}{W(z)} = \frac{(I - KC)z^{-1}}{I - (I - KC)Az^{-1}} \qquad (2.63)$$

$$\frac{\tilde{X}^+(z)}{V(z)} = \frac{-K}{I - (I - KC)Az^{-1}} \qquad (2.64)$$

The major quantity of interest within these transfer functions is the location of the poles as the poles determine the pass-band of the filter. Assuming that C is invertible, we get

$$0 = I - (I - KC)Az^{-1}$$

$$
\begin{aligned}
z &= (I-KC)A \\
&= (I-P^-C^T(CP^-C^T+R)^{-1}C)A \\
&= (I-[(AP^+A^T+GQG^T)C^T(C(AP^+A^T+GQG^T)C^T+R)^{-1}C])A \\
&= C^{-1}RC^{T}(AP^+A^T+GQG^T+C^{-1}RC^{-T})^{-1}A \\
&= \left[\frac{C^{-2}R}{A^2P^+ + G^2Q + C^{-2}R}\right] A \qquad (2.65)
\end{aligned}
$$

where the last equation holds only for scalar systems.

By examining this equation we will gain an understanding of what kind of frequency selective filtering the Kalman filter performs in its attempts to filter out disturbances. If, for example, we discover that under some conditions the Kalman filter acts as a low-pass filter, we wouldn't expect the system to be able to track high-bandwidth parameter variations since they are also modeled as disturbances. If we now look at the scalar case, equation (2.65), we see that as R increases, the quality of the sensors lessens, and $z \rightarrow A$. This means that the transfer function from disturbances to error gains an increasing low-pass characteristic. In such cases only low-frequency information from the sensors is allowed through and, analogously, only low-frequency parameter variations will be tracked. This makes sense intuitively as if we do not have a system model to start with, and if in addition our sensors degrade beyond worth (as $R \rightarrow \infty$), we have nothing. Therefore, the overall system should have little success in either of its tasks of estimation or identification. Now, if we examine the Q and P^+ terms, both of which are in the denominator, we see that as either of those two quantities increases, $z \rightarrow 0$. This means that the filter attains an increasingly all-pass character and higher bandwidth variations can be tracked. This makes some sense since as P^+ increases we should trust the model less and thus allow further refinements. In addition we modeled the unknowns as disturbances, a high Q_a indicating high-frequency parameter variations. Thus as Q increases in equation (2.65) the bandwidth of the Kalman filter grows and we would expect that higher frequency variations could be tracked. However, the sensor noise is still the dominated quantity, for if R gets really large the transfer function will have to have a low-pass character no matter what the Q or P^+, and the system won't track the unknowns well.

This last observation brings us back to the idea of incorporating memory into a system identification system. If we take advantage of periods of good data, then we will be able to exploit this stored knowledge in times of poor data. That is exactly what the LAEKF architecture is formulated to do.

In Section 1.1 where we gave an overview of the LAEKF we mentioned that an AEKF was used to do system identification for the nonlinear system. In this section we have described how the AEKF can do system identification for a time-varying linear system. We have also seen that the EKF performs successive linearizations of the system dynamics in order to calculate its estimate. Thus, one can alternately think of the EKF as a linear Kalman filter doing state estimation on a time-varying, but linear, system where the time variation of the system is determined on-line. Because the difference between linear time-varying and nonlinear systems is opaque to an EKF, this approach can also be used on nonlinear systems. However, since the "time" variations are now a function of state trajectory, rather than time, the question of how to model the time variations of the unknown parameters becomes even more important.

Chapter 3

Learning Systems

In life one is continually faced with new situations, but these situation are often not very different from ones faced in the past. It would be silly to ignore all one's accumulated experience and adapt to every situation from scratch. Rather, one wants to remember those experiences and make use of them in the new situations faced every day. This is the basic idea of learning from experience, to use the past as a vast reservoir of accumulated knowledge for future generalization. An example might again be the mountaineer of the last chapter trying to come to a conclusion about a potentially dangerous weather situation. The mountaineer relies on all his or her accumulated knowledge when deciding whether to try to gain the summit or not. In this chapter we will discuss how the idea of learning, based on the repetition of examples paradigm, can be implemented in a mathematical manner on a training set of numerical data.

This chapter opens with a general overview of concerns in learning system design in Section 3.1, while Section 3.2 discusses in more detail the particular learning system used in this thesis. Finally, Section 3.3 discusses additional issues and ideas concerning learning that arose in the course of this research, but were not investigated.

3.1 General concerns in learning system design

We will focus on the idea of learning as one of function approximation. Viewed in this light, learning systems are closely related to estimation systems as addressed in the last chapter. We are again given a batch of data somehow related to the quantities we wish to determine. Before the data was a time sequence probabilistically related to another, desired, time sequence (observation and state sequences, respectively). Now the batch of data is a set of input-output pairings that we want to use to determine the parameters of an approximate mapping that could have generated the data set.

The data set used for training can be of greatly varying origin. It can be specified by an outside user; generated on-line by the system itself; or, as in our case, can be a result of a combination of both where an operator might specify the deterministic driving signal, but the estimator generates its own training data during system operation. A question of great relevance is how to weight old training data as compared to data collected more recently. Ideally we would have some knowledge about how data relevance decays with time. If we return to the probabilistic context of the last chapter, this is exactly what the Kalman filter does. The Kalman filter's propagated covariance matrix is a measure of the information contained in the old measurements about the current situation, while the covariance of the sensor noise is a measure of information in the new data collected. Based on these two measures of quality the Kalman filter decides how best to update its estimates. However, as we are operating in a world where we don't have such measures of information, we instead use the "viscous" training methodology of gradient descent on a cost function to update our learned mapping. This methodology puts much store in the old information, only updating its mapping slightly based on each new training example.

Having decided that we will use gradient-based algorithms to do our training, let us delve deeper into the probabilistic context of Chapter 2 to gain insight into what these means for our learning algorithm. We recall first that in the discussion of Bayesian estimation (Section 2.1) the quantity of real interest was the conditional density, i.e. $p_{X|Y}(x|y)$. In our current context we will also be concerned with conditional quantities, $p_{D|H_i}(d|h)$ and $p_{H_i|D}(h|d)$, where D is the data set received and H_i is the hypothesis that the data set was generated by the i^{th} approximation architecture. In this discussion we will only have one architecture available, H_1. The problem then reduces to one of how to optimize the internal structure of that architecture to approximate the unknown function. Following in the same vein as before, we represent the internal structure as a vector of "weights," w, that parameterize the mapping. Now, the quantity in which we are most interest would be the conditional density of the weights given the training data and the system structure, i.e.

$$p_{W|D,H_1}(w|d,h) = \frac{p_{D|W,H_1}(d|w,h)p_{W|H_1}(w|h)}{p_{D|H_1}(d|h)}. \tag{3.1}$$

From the development of Section 2.1 we know that to minimize the expected value of the squared error in weight values we would use the conditional mean of the weights, $\mathsf{E}[w|D,H_1]$, as our mapping weights. However, as discussed above, we have instead decided to use gradient methods. Using gradient methods to find the maximum over the conditional density of equation (3.1) yields the "most likely" weights, w_{ML}. The is the same approach as in Section 2.1 where we derived the *maximum a posteriori* estimator by finding the maximum over the probability density. Thus by choosing to use gradient techniques we have fundamentally changed our approach from a Bayesian one to a maximum likelihood one. Note that in this problem the denominator of (3.1) is simply a normalizing factor independent of W and will not enter into the calculation for w_{ML}.

The final characteristic of learning systems that we want to discuss is that of "spatial localization." The idea of spatially localized networks is most easily understood when contrasted with "global" networks. A global architecture is one where an adjustment of a single parameter of the function approximation has a significant effect across the whole mapping. A spatially localized architecture is one where the adjustment of a single parameter generally has a nonzero effect only in a certain, bounded, region. Thus while a spatially localized system distinguishes between training examples based upon the location of each in the input space, a global architecture does not.

Let us define our functional approximation as $f(x; w)$. In this notation x is the point in the input space where we want an approximate value of the unknown function, and w is the vector of real numbers, or weights, that parameterizes the approximation. Then say we change only a single one of those weights, w_j, to $w_j + \Delta w_j$, and let $\Delta w = [0\,0\,0 \ldots \Delta w_j \ldots 0]^T$. If

$$f(x; w + \Delta w) - f(x; w) >> 0 \quad \forall x$$

then $f(x; w)$ has a global architecture.

On the other hand, say we had an approximation $f(x; x^c, w)$ where now each weight is associated with a point in the input space, x^c_j. Given the same change in weights, Δw, if

$$f(x; x^c, w + \Delta w) - f(x; x^c, w) \sim 0 \quad \forall x \;\; s.t. \; |x - x^c_j| >> 0$$

then $f(x; x^c, w)$ has a spatially localized architecture. Each weight is associated with a point in the input space, x^c_j. At some distance from this point, the effect of changing this weight on the mapping is negligible. See [2, 4, 5, 7, 20] for additional discussions on this topic.

The central strength of spatially localized architectures is that as well as not effecting the approximation in distant areas of the space, if we train weights in one region, we need not modify the weights associated with distant centers. Thus, if the system spends a lot of time in one area of the space, and then transitions into a different area, the relevant weights for that distant area will be left unmodified until the system returns. Of course, this idea is based on the assumption that the functional value in one area of the space has little relevance for the value in other areas. If one had little a priori information, this would be a safe assumption. (Alternately one could incorporate such information as is available into the Bayesian approach for model comparison that will be outlined at the end of the chapter.) However if, for example, one knew that the unknown function was in fact quadratic, then it would not make sense to use a localized system because data from one area of the space has a known relationship to the approximation across the whole space.

3.2 Basis/influence function networks

The particular learning system we use in this work is a simple type of *basis/influence* function network [2, 5]. Many other types of schemes might have been applied, but we are motivated to use this architecture because of its spatially localized character. In the basis-influence scheme, there are a number of (spatially) local models F_j, $j = 1, \ldots, N$, centered about various points in the input space. Each local model is intended to provide an accurate representation of the global input-output map whenever the input signal falls within a compact subset of the input space about the center of the local model. The scope of each local model is determined by an associated influence function d_j, $j = 1, \ldots, N$. Based on the influence functions, the outputs of the local models are combined to yield the global output. Because of the general character of local models knitted together to yield a global approximation, this network results in the cases, the function approximation is no longer linear in its adjustable parameters and true nonlinear optimization methods such as gradient descent must be used.

The weight update rule is derived from the following squared output error

$$J(x) = \tfrac{1}{2}e(x)^T e(x) \tag{3.5}$$

where the output error e is given by

$$e(x) = \theta_{targ}(x) - \hat{\theta}_{LS}(x) \tag{3.6}$$

and $\theta_{targ}(x)$ is the target data while x is the input data. The update rules are obtained by calculating the gradient of J with respect to the adjustable weights W_j and w_j^0 as follows,

$$\begin{aligned}
\frac{dJ(x)}{dW_k} &= \frac{d}{dW_k}\left\{\frac{1}{2}(\theta_{targ}(x) - \hat{\theta}_{LS}(x))^T(\theta_{targ}(x) - \hat{\theta}_{LS}(x))\right\} \\
&= (\theta_{targ}(x) - \hat{\theta}_{LS}(x))\frac{d}{dW_k}\left\{\hat{\theta}_{LS}(x)\right\} \\
&= (\theta_{targ}(x) - \hat{\theta}_{LS}(x))\frac{d}{dW_k}\left\{\sum_{j=1}^{N}(W_j(x - x_j^c) + w_j^0)D_j(x)\right\} \\
&= (\theta_{targ}(x) - \hat{\theta}_{LS}(x))(x - x_k^c)^T D_k(x)
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
\frac{dJ(x)}{dw_k^0} &= \frac{d}{dw_k^0}\left\{\frac{1}{2}(\theta_{targ}(x) - \hat{\theta}_{LS}(x))^T(\theta_{targ}(x) - \hat{\theta}_{LS}(x))\right\} \\
&= (\theta_{targ}(x) - \hat{\theta}_{LS}(x))\frac{d}{dw_k^0}\left\{\sum_{j=1}^{N}(W_j(x - x_j^c) + w_j^0)D_j(x)\right\} \\
&= (\theta_{targ}(x) - \hat{\theta}_{LS}(x))D_k(x)
\end{aligned} \tag{3.8}$$

If we now remember that $e(x) = (\theta_{targ}(x) - \hat{\theta}_{LS}(x))$ and that $D_k(x)$ is a scalar quantity, we get the following simple form for the weight update rules

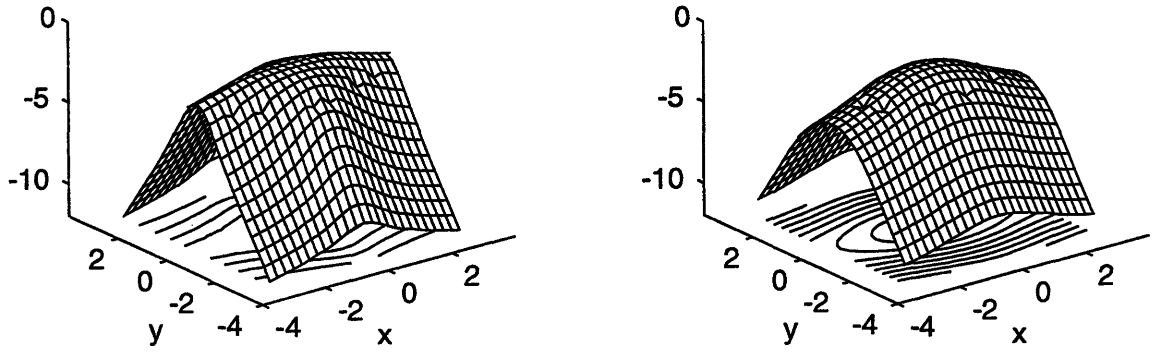

Figure 3.1: Learned mappings of $f(x,y) = -(1+x^2) - (1+\frac{8}{27}y^2)$

where $\lambda_{LR} > 0$ is the learning rate,

$$\Delta W_k = \lambda_{LR} D_k(x) e(x)(x - x_k^c)^T \tag{3.9}$$

$$\Delta w_k^0 = \lambda_{LR} D_k(x) e(x). \tag{3.10}$$

In figure 3.1 we see the results of using this architecture to learn a multi-dimensional inverse parabola with two different affine/gaussian, basis/influence, function networks. In the left hand plot there are five centers at $x^c = \{(-2,-2),(-2,2),(0,0),(2,-2),(2,2)\}$, each associated with a very narrow influence function, $V = 0.01 * I$, where I is the identity matrix. Because of these choices the affine character of the basis functions becomes very apparent. In the plot on the right, however, there are nine centers at $x^c = \{(-2,-2),(-2,0),(-2,2),(0,-2),(0,0),(0,2),(2,-2),(2,0),(2,2)\}$, and $V = I$. Because of the greater density of basis functions, the more gradual knitting together provided by the wider influences, and the lack of noise on the training set, the learned mapping is a much better approximation of the true parabolic function. Evidently the choice of basis/influence function locations, and influence widths, is an important factor in the use of this sort of learning system. This choice can be made adaptive, however, as is done in [5].

spatially localized character desired.

In our application "linear" (affine) maps are used as the local models F_j while Gaussian functions are used as the associated influence functions d_j. Assuming our network has N local models, the overall output $\hat{\theta}_{LS}(x)$ corresponding to the input x is given by

$$\hat{\theta}_{LS}(x) = \sum_{j=1}^{N} F_j(x) \frac{d_j(x)}{\sum_{k=1}^{N} d_k(x)} = \sum_{j=1}^{N} F_j(x) D_j(x). \tag{3.2}$$

where D_j are the *normalized* influence functions. Since F_j is affine, it can be written as

$$F_j(x) = W_j(x - x_j^c) + w_j^0 \tag{3.3}$$

where W_j is the "slope" matrix, w_j^0 is the "bias," and x_j^c is the "center" (local origin) about which the approximation is intended to hold. The influence function d_j is simply an unnormalized Gaussian function

$$d_j(x) = \exp[-\tfrac{1}{2}(x - x_j^c)^T V_j (x - x_j^c)] \tag{3.4}$$

where V_j is a positive definite matrix.

A gradient learning algorithm is used to update the adjustable weights of the learning system. For the results presented in this thesis, the adjustable weights are limited to W_j and w_j^0, associated with the local affine approximations. Because the weights enter linearly into the approximation scheme, one could solve for the minimum error mapping directly using a recursive least squares type algorithm. As discussed above, we choose not to do this because one would need knowledge about how data relevance decays with time. In the absence of such knowledge gradient descent algorithms are much more robust to such modeling uncertainties. More elaborate basis/influence learning systems are possible where the parameters associated with the influence functions (e.g., the matrices V_j and the center positions x_j^c) are also be updated [5]. In such

Alternately, different learning system architectures (basis/influence centers and widths in this case) can be compared according to their Bayesian "evidence," an approach that will be discussed in the next section.

3.3 Discussion

If we return to Section 1.1 we now better understand the motivations behind our choice of learning system architecture for the LAEKF. The central characteristic that attracted us to basis/influence function approximation networks was their spatially localized character. Our system will be collecting training information only along state trajectories. If there is some area of the state space that it has not visited, we want to leave the mapping in that area alone until the system gets there. Similarly, once the system leaves a certain area of the state space, we want the mapping to remain unmodified in that region until the system returns. Spatially localized networks exactly provide such a property.

An issue that arises in our design is that the training set that the learning system uses is made up of estimates coming from Kalman filters. Because of this there is error both in the input values (the state estimates) and in the output targets (the estimates of the unknown portion of the dynamics). The uncertainty in both halves of the training set means that the learning problem is analogous to a total least squares problem, as opposed to a least squares problem where error is confined to the outputs. We have not taken any steps to address this issue, but future work could be along such lines.

Another situation alluded to, but not addressed, above was that of having multiple learning architectures to choose from. When there are multiple model architectures available, one is not only faced with the problem of picking the best weights to match the learned mappings to the data, but the question of which model to use, or how far to trust each. Note that the the quality of each learned mapping is not necessarily proportional to how well each architecture fits the data points. To see this we turn to the familiar case of fitting a polynomial to some noisy measurements of an underlying function in a least squares manner. If the order of the polynomial is the same as the number of data points, then those points can be fitted exactly, but there will be wild variations between them. This is the problem of overfitting data. We want a method for minimizing the error between the underlying function and approximation, not just the error in the training set. One method often used is to train networks on some percentage of the available training set, and then to do cross-validation with the un-trained-upon examples to see how the error increases at points between those trained upon. However, if the amount of data available for training is limited, one would prefer to train and test on the same batch of data. Bayesian analysis focusing on conditional densities lets us do this in a consistent manner.

Again the quantities of interest are the conditional densities, $p_{D|H_i}(d|h)$ and $p_{H_i|D}(h|d)$ where now $i = 1, \ldots N$. To decide the amount to which each mapping architecture can be trusted, we turn to the posteriori probability, or the "evidence," of each hypothesis, i.e.,

$$p_{H_i|D}(h|d) \propto p_{D|H_i}(d|h) p_{H_i}(h). \tag{3.11}$$

If we set all the prior probabilities $p_{H_i}(h)$ equal we are saying that we have no a priori reason to favor one architecture over another. In this case the evidence used to compare models comes directly from the data, $p_{D|H_i}(d|h)$. This is the *predicted* probability that the hypothesized model could generate the received data set. To help see why this approach works to avoid the overfitting problem, we turn to figure 3.2. In this figure the horizontal axis represents the possible range of data sets D used for training. If the approximation architecture is sim-

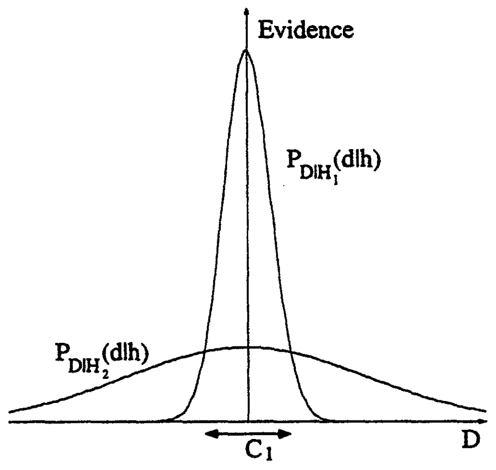

Figure 3.2: A Bayesian decision between architectures

ple, it can only approximate a limited number of functions well, functions that would only be able to generate a limited range of data sets. The first hypothesis has such a limited range. If the data set used for training lies in that limited set, it is quite likely that the simple architecture is a good one to use for the approximation. A more complex model can approximate a much wider set of functions which, if true, could generate much more varied data sets, and thus its conditional density, $p_{D|H_2}(d|h)$ is much wider (though it still integrates to one). So, if our priori probabilities are equal ($p_{H_1}(h) = p_{H_2}(h)$), and if the data fell in the region C_1 we would trust the first model more, and if the data fell outside of this region we would turn increasingly to the more complex architecture. An example of two architectures with differing complexities would be two back-propagation networks, one with ten and the other with one hundred hidden nodes. The more complex network can approximate a much wider range of functions (including all the ones that the simple network can — just zero out the contributions from ninety of the nodes). However, given a data set that could well be generated by the simpler architecture, there is much less evidence that it was generated by the complex one. To determine the overall approximation then, the Bayesian approach averages over all possible architecture and all possible weights of each architecture, to the degree that it trusts each architecture and each set of weights. For a more complete discussion of these ideas, and especially how to extend these ideas to back-propagation networks, see [15].

Finally, although the dynamics we are trying to learn are in fact deterministic, it would be useful to learn an output distribution, rather than an output value. Such a distribution could represent the quality of the mapping in that area. Having knowledge of the quality of the mapping embedded into the learning system would be very useful, especially for blending. A second motivation for learning an output distribution, rather than a deterministic output point, would be if the output is truly stochastic. While in our application this is not the case, it would be if we were trying to learn the mapping from state and driving signal to next state. That next state value would be uncertain because of the stochastic nature of the system. Thus learning an output distribution would be the right thing to do.

Chapter 4

Learning Identification for Estimation

In this chapter the problem of this thesis is set forth anew and the design of the learning augmented extended Kalman filter (LAEKF) is fully discussed. Section 4.1 casts the problem into a mathematical formulation. Section 4.2 provides a brief outline of the complete architecture so that in the subsequent discussions the eventual objective is not lost sight of. Section 4.3 presents a basic approach to the solution of the estimation/identification problem, and the deficiencies of this approach that motivated our work are also described. Section 4.4 presents a refinement of the approach of Section 4.3 that incorporates the idea of "memory" for the first time. Significant caveats remain, however, and the ways these issues are addressed leads to the final architecture as described fully in Section 4.5. Finally, in Section 4.6 we present a summary of the LAEKF algorithm and give a brief discussion of how we would expect the algorithm to behave.

4.1 Problem statement

Consider anew the discrete-time nonlinear system described in Chapter 1 given by

$$x_{k+1} = f(x_k, \theta(x_k)) + b(x_k, \theta(x_k))u_k + Gw_k \tag{4.1}$$

$$z_k = h(x_k) + v_k \tag{4.2}$$

where $x_k \in \Re^n$ is the state of the system, $u_k \in \Re^m$ is a known driving signal, $z_k \in \Re^p$ is the measurement, $\theta(x_k) \in \Re^r$ is the unknown state-dependent parameter vector, and w_k and v_k are process and measurement noises, respectively (assumed zero mean and white). In addition, f and b are possibly nonlinear time-invariant functions describing the plant model. In our problem, these functions are only partially known, with the unknown portion characterized by $\theta(x)$. Finally, h is a known function describing the sensor model. It should be noted that the system represented by (4.1) and (4.2) is assumed to have a sensor suite such that, given an appropriate input signal, all observability and identifiability conditions are satisfied.

As we have seen in Chapter 2, if the dynamics are fully known, a problem of great interest is the design of filters that give estimates of the state at each time $\hat{x}_k$, based on the observations z_ℓ up to time k, $\ell = 0, \ldots, k$. In the linear case the Kalman filter algorithm (equations 2.9-2.13) yields such estimates, and in the nonlinear case the extended Kalman filter can be used instead (equations 2.41-2.49).

In the presence of model uncertainty, however, an additional goal, and a necessary one for achieving the original state estimation objective, is to identify the unknown mapping $\theta(x)$. The case of a linear mapping with unknown constant parameters (i.e. $f(x_k, \theta(x_k)) \equiv \theta\, x_k$) has been widely investigated, and

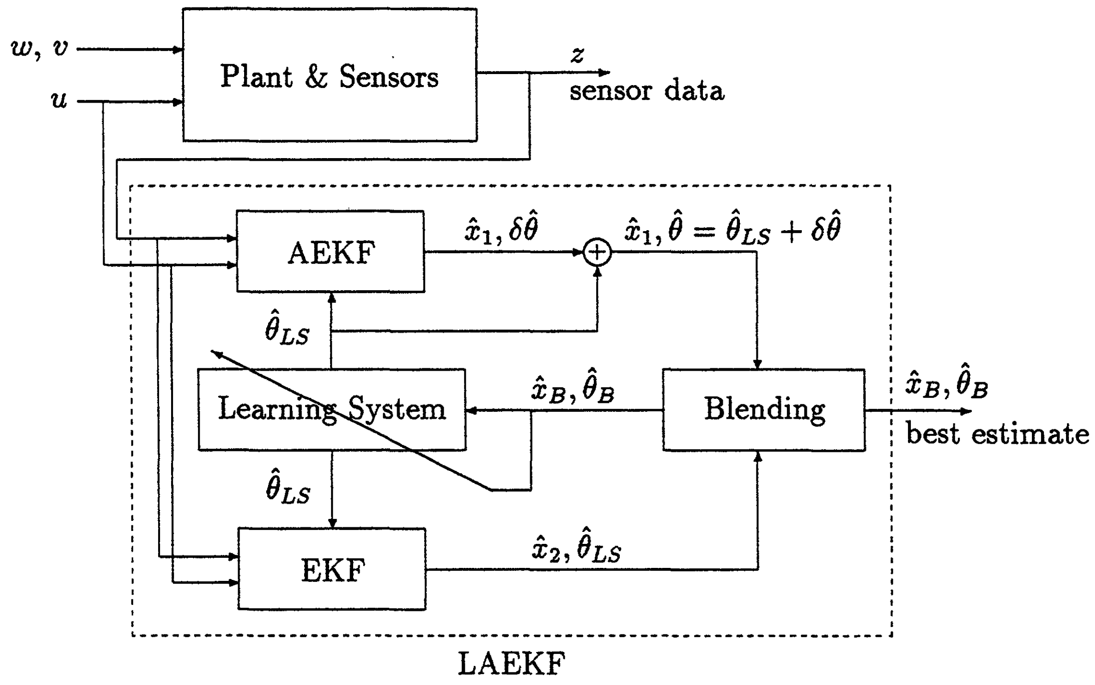
w, v
u
Plant & Sensors
z
sensor data
AEKF
$\hat{x}_1, \delta\hat{\theta}$
$\hat{x}_1, \hat{\theta} = \hat{\theta}_{LS} + \delta\hat{\theta}$
$\hat{\theta}_{LS}$
Learning System
$\hat{x}_B, \hat{\theta}_B$
Blending
$\hat{x}_B, \hat{\theta}_B$
best estimate
$\hat{\theta}_{LS}$
EKF
$\hat{x}_2, \hat{\theta}_{LS}$
LAEKF

Figure 4.1: Block diagram of learning augmented estimator

one approach based on augmented extended Kalman filtering was presented in Section 2.4. In this thesis, however, we are focusing on systems whose model uncertainties are nonlinear, but time-invariant, functions of the state. Our objective is to formulate a methodology to address this problem by drawing upon ideas from the learning literature, as discussed in Chapter 3.

4.2 The learning augmented architecture described

Figure 4.1 displays a schematic of the algorithm devised to address this problem. The *learning system* is the heart of this architecture. It encapsulates a priori state-dependent knowledge of the unknown portion of the dynamics $\theta(x)$ based on all state estimates up to the current time. The particular learning algorithm used has been fully described in Chapter 3. The mapping $\hat{\theta}_{LS}(x)$ is fed to two filters to help in estimation. The mapping is fed first to an *augmented extended*

Kalman filter (AEKF) that provides refined knowledge of the unknown dynamics by estimating the error in the learned mapping $\delta\theta$ as well as the state; and second, the mapping is fed to an *extended Kalman filter* (EKF) that provides state estimates if the AEKF is not working well. The two filters also take the known driving signal u and the sensor measurements produced by the *plant* z as inputs. Finally, the estimates of each filter are sent to the *blending* block that, based on the quality of the estimates coming from the AEKF and the EKF, produces final state and parameter estimates, $\hat{x}_B$ and $\hat{\theta}_B$. These estimates are then used in the next step of the state estimation problem, to further refine the learning system mapping, and can be used for control or other applications. For simplicity of discussion, in the rest of the paper we assume that the uncertainty resides only in the plant dynamics, $f(x_k, \theta(x_k))$.

In the following sections we examine this architecture block by block, first describing the reasons for each subsystem, and then the mathematical implementation of each. We start with the AEKF in Section 4.3. That discussion motivates the inclusion of a learning system as described in Section 4.4. At the conclusion of this section we will understand the reasons for including a second EKF and a blending system, as is discussed in Section 4.5.

4.3 Naive approach

A simple way to address the joint problems of estimation and identification is to implement a nonlinear filter on a state vector augmented by the unknowns, exactly as in Section 2.4. See figure 4.2 for the schematic of such an approach. Here the plant takes a driving signal and process and sensor noises, u, w and v, respectively, as inputs; and yields sensor data z as an output. The AEKF takes the driving signal and sensor data as inputs and produces estimates of state and

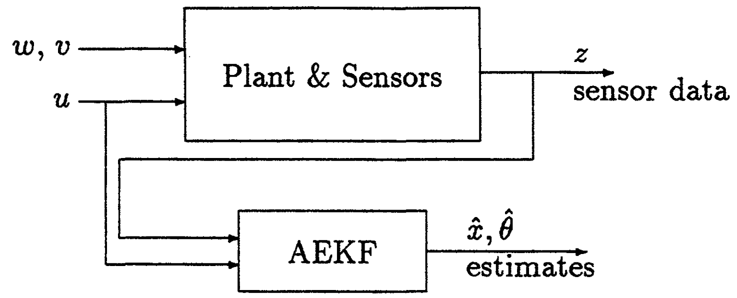

Figure 4.2: AEKF addressing joint tasks of estimation and identification

unknown dynamics, $\hat{x}$ and $\hat{\theta}$.

The augmented state equations upon which the AEKF design is based are

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ \theta_{k+1} \end{bmatrix} &= \begin{bmatrix} f(x_k, \theta_k) \\ A_\theta \theta_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} G & 0 \\ 0 & B_\theta \end{bmatrix} \begin{bmatrix} w_k \\ \tilde{w}_k \end{bmatrix} && (4.3) \\ z_k &= Hx_k + v_k. && (4.4) \end{aligned}$$

An important point to these equations is the way in which the time variation of θ is modeled. Here we have modeled it as a Gauss-Markov process,

$$\theta_{k+1} = A_\theta \theta_k + B_\theta \tilde{w}_k \qquad (4.5)$$

where the matrices A_θ and B_θ are diagonal and selected so as to give good tracking quality. In the linearized context this means that $||\mathrm{T}_{\hat{\theta}\theta}|| \simeq 1$ where $\mathrm{T}_{\hat{\theta}\theta}$ is the mapping $\theta \mapsto \hat{\theta}$. If one wishes to simplify the problem further, A_θ can be chosen equal to zero. In this case the modeling of the time variation of θ reduces to a random walk. This simplification was made in Section 2.4 and will be made again for the results presented in Chapter 5.

The deficiencies in this approach are two-fold. First, as discussed before, the unknowns are modeled as time-varying, so if the state trajectory takes the system out of a certain area of the state space, and subsequently returns, old estimates of the unknown dynamics will have little influence on later ones. Some sort of "memory" is needed so that when the system revisits an area of the state space, it does not have to reidentify the dynamics from scratch.

A second weakness is that the performance of the AEKF is very dependent on the way the time variation of the unknowns is modeled. If nothing is known beforehand about this variation, great mistakes can be made. Therefore, an architecture that minimizes the dependency of the overall performance on this user-determined modeling is desired.

4.4 Incorporating memory

In order to address the deficiencies described in the last section, we make two changes in the architecture. We first add a spatially localized learning system that uses the state/parameter estimates as training data and, over time, constructs a model of the unknown dynamics. By making this change we have incorporated memory into the identification architecture. The second change is to use the nonlinear filter to identify the error in the learning mapping, $\delta\theta$, rather than the full unknown dynamics, θ. A schematic of this refined system is given in figure 4.3. Note that in the figure the diagonal arrow through the learning system indicates that the estimates being fed to the learning system are used to modify the learned mapping. However, while training data accumulates at the rate of the sensors, it is stored in a buffer and the learning system mapping is updated only periodically.

The function approximation form of the learning system is as follows,

$$\hat{\theta}_{LS}(x) = \sum_{j=1}^{N} F_j(x)\frac{d_j(x)}{\sum_{k=1}^{N} d_k(x)} = \sum_{j=1}^{N} F_j(x)D_j(x). \tag{4.6}$$

where the local basis function are affine and the influences Gaussian, as in Chapter 3. The complete algorithm for training the learning system mapping is

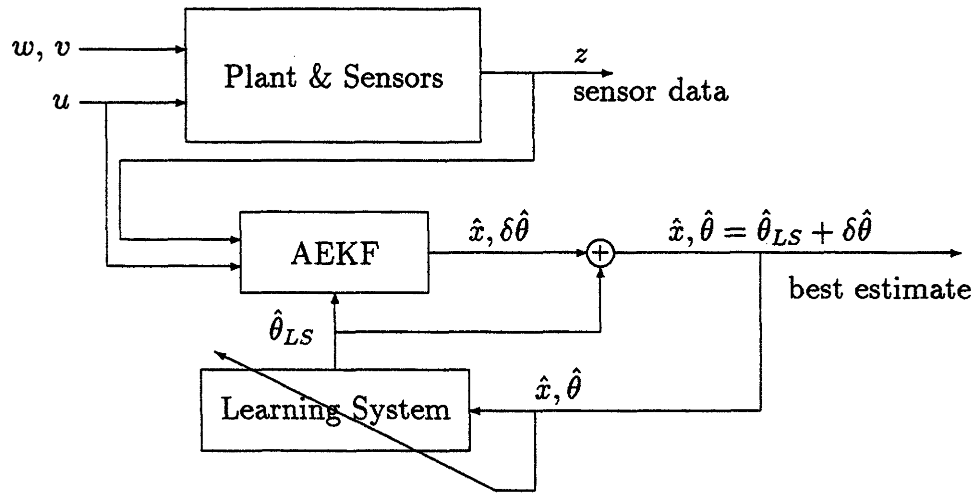

Figure 4.3: AEKF merged with a learning system

given by equations (3.2-3.10). In this application the training data is provided by the state and parameter estimates, $\{\hat{x}_k, \hat{\theta}_{LS} + \delta\hat{\theta}_k\}$, with the state estimates serving as inputs, and the corrected parameter estimates as targets.

The augmented state equations upon which the AEKF design is based are

$$\begin{bmatrix} x_{k+1} \\ \delta\theta_{k+1} \end{bmatrix} = \begin{bmatrix} f(x_k, \hat{\theta}_{LS} + \delta\theta_k) \\ A_\theta \delta\theta_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} G & 0 \\ 0 & B_\theta \end{bmatrix} \begin{bmatrix} w_k \\ \tilde{w}_k \end{bmatrix} \quad (4.7)$$

$$z_k = Hx_k + v_k. \quad (4.8)$$

The reason for estimating a correction term, $\delta\theta$, rather than the whole value of the unknown function at each time step, is that by doing so we explicitly make use of the information stored in the learned mapping in our current estimation and identification steps. In this way we incorporate a priori knowledge gained from previous visits to this area of the state space into our current tasks of estimation and identification. As we will see, making this change yields system performance that is much more invariant to user choice of A_θ and B_θ.

The problem with this design is that if the nonlinear filter (the AEKF) starts to produce poor state/parameter estimates, the learning system will blithely train on this bad data and will learn the wrong function. If this continues, the system performance can degrade past that of the memoryless AEKF as described in Section 4.3. Examples of situations where this might happen are if the quality of the sensors degrades, or if the driving signal's bandwidth is above that which the filter can track.

Two remedies can help to avoid this pitfall. First is a way to compute state estimates based solely on the old information stored in the learning system whenever the nonlinear filter (then AEKF) is not working properly. Second is a way to judge the quality of the information already stored in the learning system versus the new information being collected by the AEKF. The first requires the use of an additional filter, the EKF block in figure 4.1, and the second a blending mechanism also seen in the same figure. The next section will discuss these additions.

4.5 The learning augmented architecture

The architecture of the last section suffers from the lack of a check on the learning system. To address this issue we first add a second estimator to our design, an extended Kalman filter. The dynamic model upon which the EKF is based is,

$$\begin{aligned} x_{k+1} &= f(x_k, \hat{\theta}_{LS}) + Bu_k + Gw_k && (4.9) \\ z_k &= Hx_k + v_k. && (4.10) \end{aligned}$$

We see then that this filter does not attempt to do any system identification, rather it completely trusts the learned model as the correct model of the unknown portion of the dynamics. As we will see, this filter will come to the fore when the AEKF is not producing good estimates.

Now that there are two filters, the AEKF and the EKF, there are also two sets of estimates, $\hat{x}_1$, $\hat{\theta}_{LS} + \delta\hat{\theta}$, and $\hat{x}_2$, $\hat{\theta}_{LS}$, respectively, as diagrammed in figure 4.1. (Note that because the EKF performs no system identification, its estimate of the unknown dynamics is the same as the learning system's.) A second additional subsystem is needed to combine these two sets of estimate into the final *blended* estimates, $\hat{x}_B$ and $\hat{\theta}_B$. The following blending law is used where the derivation will be given in subsequent subsections,

$$\begin{aligned} \hat{x}_{B,k} &= I_{2,k}(I_{1,k} + I_{2,k})^{-1}\hat{x}_{1,k} + I_{1,k}(I_{1,k} + I_{2,k})^{-1}\hat{x}_{2,k} && (4.11) \\ \hat{\theta}_{B,k} &= I_{2,k}(I_{1,k} + I_{2,k})^{-1}(\theta_{LS,k} + \delta\theta_k) + I_{1,k}(I_{1,k} + I_{2,k})^{-1}\theta_{LS,k} && (4.12) \\ & && (4.13) \end{aligned}$$

where

$$\begin{aligned} I_{1,k} &= \hat{P}_{1,k} - \tfrac{1}{2}(\hat{Q}_{12,k} + \hat{Q}_{21,k}) && (4.14) \\ I_{2,k} &= \hat{P}_{2,k} - \tfrac{1}{2}(\hat{Q}_{12,k} + \hat{Q}_{21,k}) && (4.15) \end{aligned}$$

and making the following definition of error

$$e_{i,k} = (z_k - H\hat{x}_{i,k}) \qquad i = \{1,2\} \qquad (4.16)$$

we can compute the sample statistics

$$\begin{aligned} \hat{P}_{i,k} &= \lambda e_{i,k}\, e_{i,k}^T + (1-\lambda)\hat{P}_{i,k-1} \qquad i = \{1,2\} && (4.17) \\ \hat{Q}_{12,k} &= \lambda e_{1,k}\, e_{2,k}^T + (1-\lambda)\hat{Q}_{12,k-1} && (4.18) \end{aligned}$$

It is because of the nonlinear and unknown character of this system, and because we need a measure of the quality of the learned mapping, that we cannot calculate such blending statistics analytically. We have thus turned to the output squared error of the AEKF as a measure of the quality of the corrected mapping, $\hat{\theta}_{LS} + \delta\hat{\theta}$, and to the output squared error of the EKF as a measure of the quality of the quality of the learning system mapping, $hat\theta_{LS}$. These measures are then used in the blending block to yield $\hat{x}_B$ and $\hat{\theta}_B$, the outputs of the learning augmented estimation system, as well as the data upon which the learning system is trained. This is the point in the architecture where knowledge of the observation function H is crucial. Without such knowledge it would be impossible to compute such sample statistics.

4.5.1 A motivating example for blending

The blending block is introduced into the architecture to provide a check on the learning system. This check is needed so that an unstable feedback loop is not set up where the learning system learns the wrong function, thereby causing the quality of the estimates to degrade, so that the learning system is fed even worse training data, and so forth. A method is needed to emphasize the correction term $\theta_{LS} + \delta\theta$ if the learning mapping is incorrect (as at network initialization and visits to unexplored areas of the state space), and to emphasize the learned mapping θ_{LS} if the AEKF is performing poorly. To motivate our approach, we consider the problem of having information from two separate, but correlated, sensors of a random vector y,

$$y_1 = y + w_1 \tag{4.19}$$

$$y_2 = y + w_2 \tag{4.20}$$

where $y_1, y_2, y, w_1, w_2 \in \Re^p$. Assume that w_1 and w_2 are zero-mean random vectors such that $\mathsf{E}[w_1 w_1^T] = P_1$, $\mathsf{E}[w_2 w_2^T] = P_2$, and $\mathsf{E}[w_1 w_2^T] = Q_{12} = Q_{21}^T$.

We want to combine these two measurements linearly so as to achieve an unbiased estimate, and to minimize the trace of the resulting covariance matrix. We start as follows, remembering that the measurements are assumed to be unbiased,

$$\begin{aligned}
\mathsf{E}[y_B - y] &= \mathsf{E}[Ay_1 + By_2 - y] \\
&= A\,\mathsf{E}[y_1] + B\,\mathsf{E}[y_2] - \mathsf{E}[y] \\
&= \{A + B - I\}\mathsf{E}[y] = 0. \qquad (4.21)
\end{aligned}$$

To get an unbiased estimate, our matrices must satisfy the relationship $A+B = I$, where I is the identity matrix. Now, to minimize the covariance of the blended estimate, y_B, our optimization problem is

$$\begin{aligned}
\min_{A,B}\ \text{trace}\,\mathsf{E}[ee^T] &= \min_{A,B}\ \text{trace}\,P_e \qquad (4.22) \\
\text{s.t. } A + B &= I
\end{aligned}$$

where e is the error, and the error covariance matrix P_e can be expanded as follows,

$$\begin{aligned}
\mathsf{E}[ee^T] &= \mathsf{E}[(y_B - y)(y_B - y)^T] \\
&= \mathsf{E}[(Ay_1 + By_2 - (A+B)y)(Ay_1 + By_2 - (A+B)y)^T] \\
&= \mathsf{E}[(A(y_1 - y) + B(y_2 - y))(A(y_1 - y) + B(y_2 - y))^T] \\
&= AP_1A^T + AQ_{12}B^T + BQ_{21}A^T + BP_2B^T \\
&= AP_1A^T + AQ_{12}(I - A)^T + (I - A)Q_{21}A^T \\
&+ (I - A)P_2(I - A)^T. \qquad (4.23)
\end{aligned}$$

Taking the derivative of the trace of P_e and setting it equal to zero results in

$$\begin{aligned}
\frac{\partial}{\partial A}(\text{trace}P_e) &= A(P_1 + P_1^T) + Q_{12}^T - A(Q_{12} + Q_{12}^T) + Q_{21}^T(\frac{\partial}{\partial A}A^T) \\
&- A(Q_{21} + Q_{21}^T) + (I - A)(P_2 + P_2^T)(-I) \\
&= 2AP_1 - 2A(Q_{12} + Q_{21}) + 2AP_2 - 2P_2 + (Q_{12} + Q_{21}) \\
&= 0.
\end{aligned}$$

Solving for A gives us

$$A = (P_2 - \tfrac{1}{2}(Q_{12} + Q_{21}))(P_1 + P_2 - (Q_{12} + Q_{21}))^{-1} \tag{4.24}$$

Now, if we make the identifications,

$$I_1 = P_1 - \tfrac{1}{2}(Q_{12} + Q_{21}) \tag{4.25}$$

$$I_2 = P_2 - \tfrac{1}{2}(Q_{12} + Q_{21}) \tag{4.26}$$

The following linear blending law yields the minimum covariance estimate,

$$y_B = I_2(I_1 + I_2)^{-1}y_1 + I_1(I_1 + I_2)^{-1}y_2 \tag{4.27}$$

where we have assumed that $(I_1 + I_2)$ is invertible.

4.5.2 Blending with sample statistics

The approach described in the last Subsection (4.5.1) would be a good one if we had second order statistics. However, because of the plant's nonlinear and unknown character we don't have these statistics. Still, we will model our approach on the example given above. In our case the two sources of information are the adaptive filter and the learning system. Although Ljung has shown that while being used as a system identification tool the AEKF can become biased, and need not converge even in the case of identifying constant parameters for a linear time-invariant system [12], we assume that the state dynamics are such that the filter remains unbiased and non-divergent. Under these assumption we wish to apply (4.27) to our system where

$$y_1 = \begin{bmatrix} \hat{x}_1 \\ \hat{\theta}_{LS} + \delta\hat{\theta} \end{bmatrix} \tag{4.28}$$

$$y_2 = \begin{bmatrix} \hat{x}_2 \\ \hat{\theta}_{LS} \end{bmatrix}. \tag{4.29}$$

To implement equation (4.27) we need a measure of the quality of these estimates. The measure we use is derived from the output squared error given by

$$e_{1,k}e_{1,k}^T = (z_k - H\hat{x}_{1,k})(z_k - H\hat{x}_{1,k})^T. \tag{4.30}$$

and

$$e_{2,k}e_{2,k}^T = (z_k - H\hat{x}_{2,k})(z_k - H\hat{x}_{2,k})^T. \tag{4.31}$$

The difference between the two error calculations is that in (4.30) the AEKF relies on the corrected mapping, $\hat{\theta}_{LS}+\delta\hat{\theta}$, while in (4.31) the EKF relies solely on $\hat{\theta}_{LS}$. This exclusive reliance on the learned map means that the EKF's output squared error is a relative measure of learning system quality. Finally, smoothed versions of the output squared errors are calculated,

$$\hat{P}_{i,k} = \lambda e_{i,k}e_{i,k}^T + (1-\lambda)\hat{P}_{i,k-1} \qquad i = \{1,2\}. \tag{4.32}$$

Equation (4.32) is the expression for a low pass filter where the filter bandwidth is controlled by λ. These quantities are then used in blending in the same way that P_1 and P_2 were in (4.25) and (4.26). An analogous measure is calculated for the pseudo cross-covariance,

$$\hat{Q}_{12,k} = \lambda e_{1,k}e_{2,k}^T + (1-\lambda)\hat{Q}_{12,k-1} \tag{4.33}$$

Since the output error is a function of the current learning mapping, if the mapping is incorrect, this will be reflected in a large output error for the EKF. If the mapping is correct, but the adaptive $\delta\theta$ term is not, this will be reflected in a large output error for the AEKF.

This approach is limited by the relationship between sensor data and unknown dynamics. If the sensor data is strongly affected by the unknown dynamics then the approach can be usefully applied. Of course, if the sensor data is not closely related to the unknown dynamics, it may not be possible to do system identification in the first place. A second limitation to the approach is in the case of high sensor noise. Under high sensor noise conditions the blending ratio will approach $1:1$ because the noise will dominate the residue term $(z - H\hat{x} = H\left(x - \hat{x}\right) + v)$ from which the sample statistics are calculated. In such cases it would be very useful to have a learning system that kept track of its own error statistics, rather than relying on the output error for such information. Finally, dimensionality issues arise when considering this blending approach. Even if multiple sensors are available, they do not necessarily yield covariance-matrix-like information without further knowledge. Thus, a priori information on the relation of sensor data to the domain and range of the unknown functions can greatly help this identification scheme. In the examples we present in Chapter 5, the observation functions are linear, scalar, and known.

This method seems to work quite well even though the measures of quality are calculated from sample statistics and are of the wrong dimensionality. The reason is that all that is really needed is a *relative* measure of quality to judge between the two sources of information. This measure need not be a true measure of information, as is the inverse of a covariance matrix, it must simply measure the quality of one information source with respect to the other. It is such a relative quality of measure that the output error covariance yields.

4.6 Summary of the LAEKF algorithm

The LAEKF algorithm can now be summarized. If we refer back to figure 4.1, we can walk through the implementation of each block. The system dynamics are given by equations (4.1) and (4.2). It is the θ in these equations that we wish to identify, and a priori information about the value of θ is given by the learning system whose mapping is of the form of equation (4.6).

The learning mapping is fed to two filters, an AEKF and an EKF, to help in estimation and identification. The AEKF design is based on the dynamic models of equations (4.7) and (4.8), while the EKF design is based on slightly different dynamics, equations (4.9) and (4.10). Each of these filters generates a set of estimates. These two sets of estimates must be combined to yield the final estimates of states and unknown dynamics. This operation is performed by the blending block.

The blending block is founded on the idea of generating second-order sample statistics. The blending law seeks to determine a minimum variance unbiased combination of the two estimates. Its design is based upon the calculation of sample statistics as given by equations (4.11-4.18).

Finally, the output of the blending block provides not only the final estimates of states and unknown dynamics, but also training data for the next step in the refinement of the learning system mapping. The learning system is of a spatially localized type and the algorithm for updating its mapping was described fully in Chapter 3 and is given by equations (3.2-3.10).

Chapter 5

Applications to Physical Systems

In the last chapter we discussed the motivations behind, and the design of, the LAEKF. There were three main design points. The first was to incorporate ideas of memory through inclusion of a learning system in the estimation architecture. The second was to use an EKF coupled with a blending system to provide a check on the influence of the learning system mapping. The final idea harkens back to Chapter 3 and was to make the learning system spatially localized in order to accommodate the way that training data is generated, i.e. along state trajectories.

In this chapter we present three systems that illustrate each of these design points. The example of Section 5.1 is a spring-mass-damper system with an unknown spring stiffness function. This is a very familiar dynamic system and so the performance of the LAEK will be easy to understand; it is also a system that well demonstrates the advantages in constructing a mapping of the unknown dynamics. In Section 5.2 we present a second simulation of the spring-mass-damper system. However, this time the conditions under which the system operates change in time. These changes to some degree illustrate the strength of using a spatially localized approximation, but more significantly demonstrate the importance of using a blending system in the learning augmented architecture. Finally, in Section 5.3 we implement the architecture on a more complex physical system, the aeroelastic oscillator. The response of this system is characterized by two limit cycles. These means that training data is collected along very constrained state trajectories, thereby fully illustrating the importance of using a spatially localized learning system.

5.1 Spring-mass-damper

The first system on which we will implement our LAEKF is a second-order mass-spring-damper system with an unknown nonlinear spring stiffness. The system dynamics are described by the scalar equations

$$\ddot{x} = -\dot{x} + \theta(x)x + u \tag{5.1}$$

where x is position, u is the known input signal, and the unknown nonlinear spring stiffness is a function of position

$$\theta(x) = -(1 + \tfrac{8}{27}x^2). \tag{5.2}$$

The sensor data is scalar, observing only position, with measurements received at 10 Hz and a noise covariance matrix of $R^{1/2} = 0.2$. The system is driven by the superposition of two sine waves of the same amplitude, but one of period $T = 30$ seconds, and the other of period $T = 15$ seconds, i.e. $u(t) = 15\sin(0.2t) + 15\sin(0.4t)$. As we will see, this input provides sufficient excitation of the dynamics so as to allow system identification. The time variation of the unknown spring stiffness function is modeled as a random walk with $A_\theta = 0$, and $\mathsf{E}[B_\theta \tilde{w}\tilde{w}^T B_\theta^T] = 2$ (see equation (4.7)). The learning system is initialized at a constant zero with basis-influence functions placed at $x^c = \{-4, -2, 0, 2, 4\}$ with $V_j = I$. The decay parameter for the sample statistics, see equation (4.32), is set to $\lambda = 0.2$. The simulation runs for 300 seconds

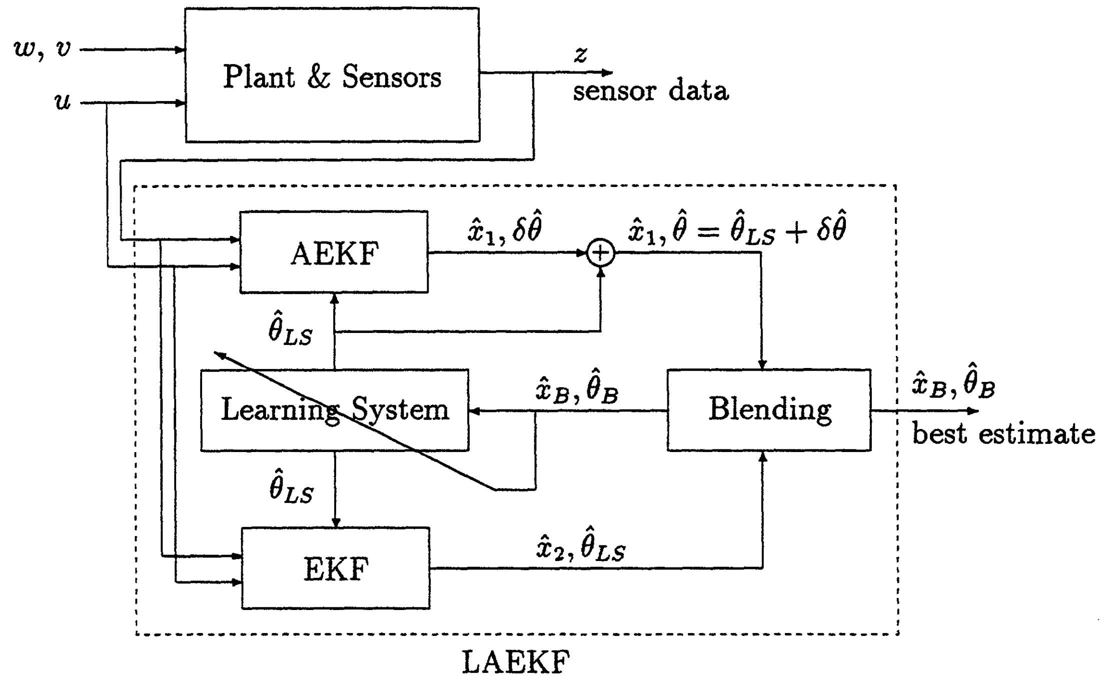
w, v
u
Plant & Sensors
z
sensor data
AEKF
$\hat{x}_1, \delta\hat{\theta}$
$\hat{x}_1, \hat{\theta} = \hat{\theta}_{LS} + \delta\hat{\theta}$
$\hat{\theta}_{LS}$
Learning System
$\hat{x}_B, \hat{\theta}_B$
Blending
$\hat{x}_B, \hat{\theta}_B$
best estimate
$\hat{\theta}_{LS}$
EKF
$\hat{x}_2, \hat{\theta}_{LS}$
LAEKF

Figure 5.1: Block diagram of learning augmented estimator

and learning is done off line on six data batches, each consisting of 50 seconds of estimates. That is, the learning system first collects 50 seconds worth of data and then trains; then that new mapping is used to help in the next 50 seconds of estimation and so on. Figure 5.1 presents anew the schematic of the LAEKF from Chapter 4 for reference in the following discussions.

To see how learning affects estimation, figure 5.2 plots the estimation performance of the filters in terms of root-mean-squared (RMS) error. The error is calculated over the 30 seconds of estimates prior to each plotted point (i.e. a causal error calculation); this is why the error plots are at zero until time $t = 30$. So, if there are N samples in the past 30 seconds, then the RMS error for a single scalar state estimate at time k is calculated as follows

$$\mathrm{RMS}_k = \frac{1}{N} \sum_{i=0}^{N-1} (\hat{x}_{k-i} - x_{k-i})^2. \tag{5.3}$$

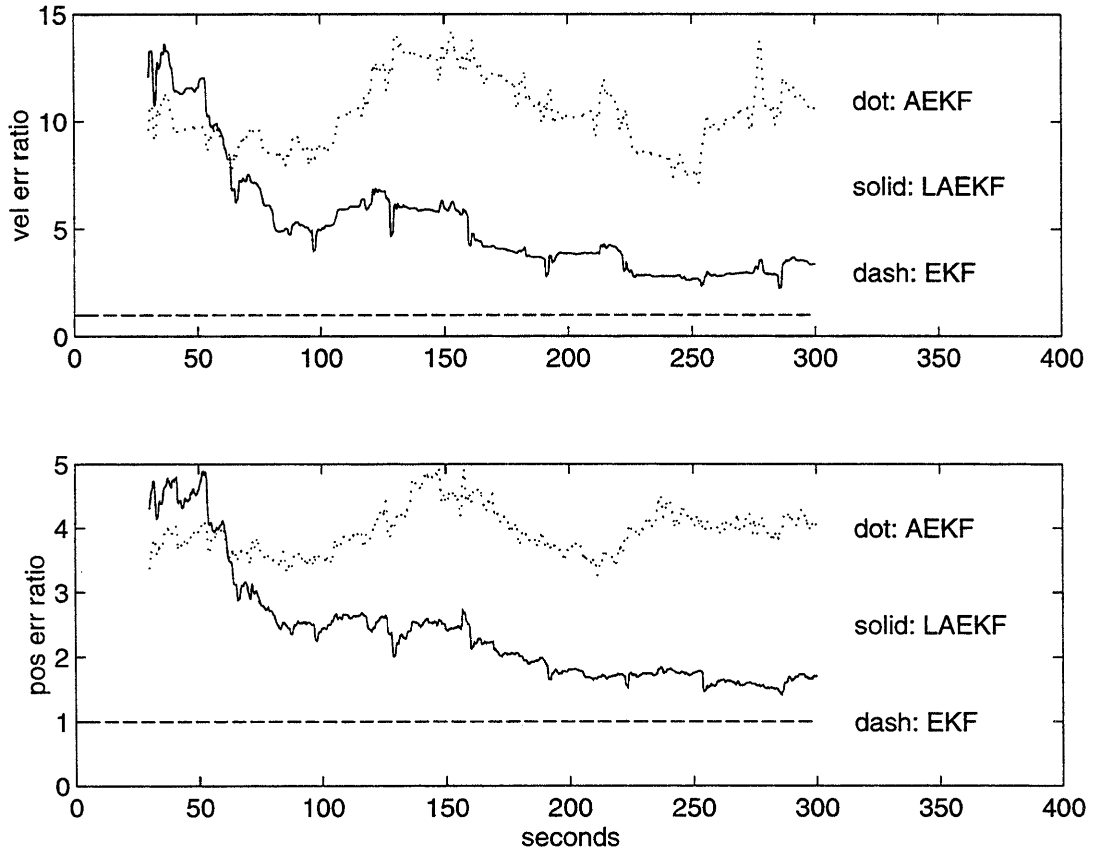

Figure 5.2: RMS state estimation errors normalized by performance of standard EKF (designed with exact nonlinear model).

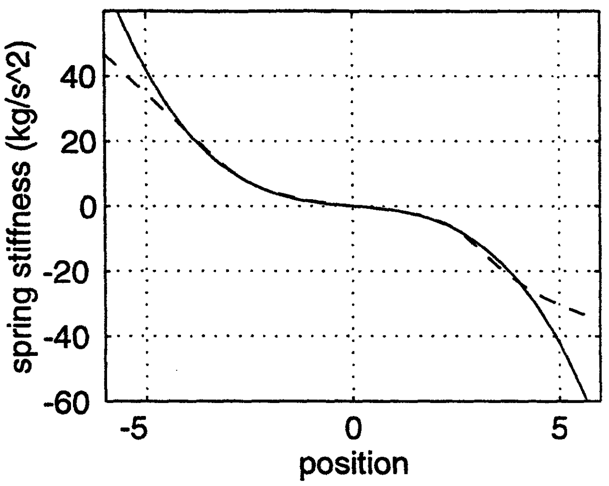

Figure 5.3: Learned mapping of spring stiffness. Dashed line corresponds to mapping at $t = 300$; solid line is truth.

Each subplot of Figure 5.2 contains three curves. First is the performance of an EKF *designed with full knowledge of the nonlinear dynamics.* This filter's accuracy gives us a base-line from which to evaluate the performance of the other estimation methods. If the error of either the AEKF or LAEKF approaches this level of accuracy, then we would say it is approaching the best possible performance. We plot the ratio of the RMS error of each estimator to the RMS error of the EKF. Thus, the ratio of EKF error to EKF error is plotted as a reference dashed line at 1.0. The error of the AEKF normalized by the EKF error is plotted as a dotted curve, and the normalized error of the LAEKF is the solid curve. The top graph in Figure 5.2 shows the velocity error ratios, and the bottom one shows the position error ratios.

The second quantity of interest is the quality of the system identification done by the LAEKF. First, in figure 5.3, we plot the learned system function (dashed line) versus the true unknown system function (solid line). This figure shows the true mapping versus the learned mapping after the six training batches of data have been used, i.e. at $t = 300$.

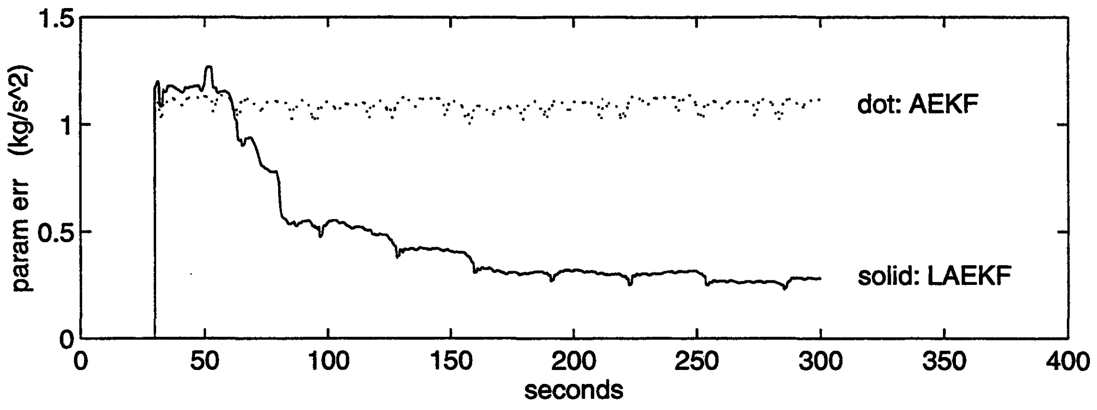

Figure 5.4: RMS errors in spring stiffness identification.

Figure 5.4 uses the same error measure as in figure 5.2, and compares the performance of the AEKF and LAEKF as system identification approaches under the same operating conditions. Since an EKF designed with full knowledge of the nonlinear dynamics need not do any system identification, in this figure the EKF performance is not plotted for comparison.

As can be seen in both figures 5.2 and 5.4, the errors in the learning augmented system are substantially smaller than those for the system without learning. It should also be noted that all the information from which the learning map is constructed comes from a single driving signal. With more variation in the driving signal — a richer input — the system identification could be done to an even greater degree of accuracy. On the other hand, it is also possible to feed the system unfriendly signals, such as very high frequency signals above the bandwidth of the filter, that can lower the performance gains of this architecture. So, as with other identifications schemes, the performance of the system remains, in part, dependent on the characteristics of the driving signals.

It should also be noted that in this application the blending did not come into play. This is because the system was driven by the same double sine wave

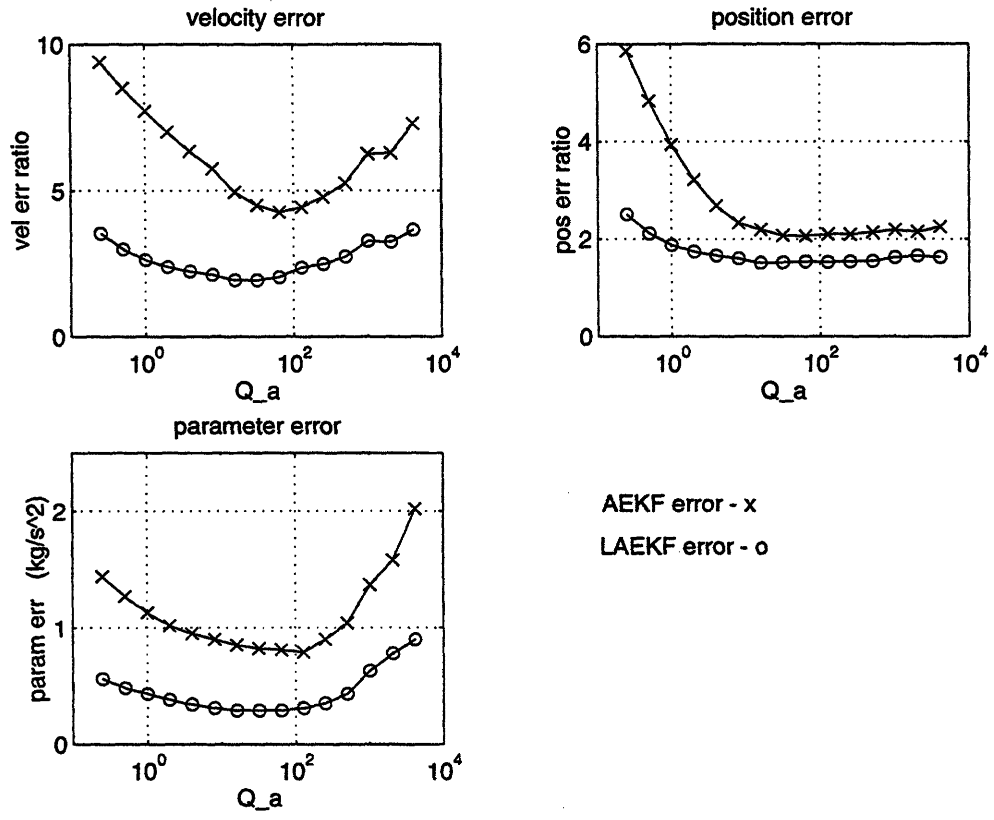

Figure 5.5: Identification performance versus noise modeling level.

signal under the same noise conditions for the entire run. Since there was no change in conditions there was no reason why the AEKF incorporated into the LAEKF architecture should diverge, and thus no reason why the system should rely exclusively on the old learned mapping and the EKF instead of continuing to correct the mapping. In the next section we will see another application to a spring-mass-damper system where the blending will play a very central role in preserving system performance after a change in operating conditions.

The last aspect of LAEKF performance to investigate goes back to our discussion of how to model the time variation of the unknowns. As discussed in Section 2.4.1, this user-determined quantity can have a great effect on the success of the identification scheme. In figure 5.5 we plot the system identification performance versus noise modeling level. That is, if we model the parameter variations as

$$\dot{a}_t = w_{a,t}$$

where $w_a \sim N(0, Q_a)$, then the independent axes in figure 5.5 plots the intensity of the Q_a matrix. As can be seen, across the noise modeling spectrum, the error of the LAEKF is significantly below that of the AEKF. In addition, the performance of the LAEKF is a less strong function of the noise level — i.e. the plot of LAEKF performance is flatter, and therefore more invariant to the user's choice of Q_a. These are both objectives in the design of this architecture, and are a function of choosing to estimate a correction term $\delta\theta$ rather than the whole unknown θ. This choice yields improved performance because as the unknown dynamics are learned the quantity that remains to be estimated decreases and thus eases the estimation problem. In this way learning helps in the identification as well as the estimation problems.

If we think for a moment about figure 5.5, we realize that the characteristic "U" shape of the plots in figure 5.5 shows that there is an optimal choice of Q_a. That is, there exists a choice of Q_a that results in the best LAEKF performance. If Q_a is chosen too small, parameter variations cannot be tracked, and both identification and estimation suffer. If, on the other hand, Q_a is chosen too large, both tasks suffer again from designs optimized for excessive levels of process noise.[1] Ideally then, after each training batch, and for each region of the state space, a new Q_a would be chosen that would best model the speed of parameter variation in that region of the space. In the results presented here, this is not done, rather for each 300 second experiment Q_a is set equal to a constant. Each experiment corresponds to a different value of Q_a and to a

[1]The reason that the error in the position estimate levels out with increasing Q_a is that there is a position sensor that upper bounds the error in that estimate.

different data point in figure 5.5 (the x's and the o's). The idea of treating Q_a as an adaptive quantity could work well with the spatially localized character of the learning system. With each local approximation some measure of approximation quality could be associated that could then be used in either tasks of blending or noise modeling. As uncertainty in the mapping increases, the noise level could be increased, and vice-versa. Although not implemented, this would be an interesting path for future investigation.

5.2 Spring-mass-damper II

In this section we also present results for the system used in 5.1, but this time the conditions are less friendly for system identification. This example will help explain why the spatially localized character of the learning system is important, and why the use of a blending block is critical. Initially the system is driven by a biased signal; this situation reveals the advantage of using a spatially localized function approximation system. Then, after a time, the sensor the system relies on degrades significantly. This situation is disastrous for an identification approach that treats the unknown as time-varying as the algorithm must continue to rely heavily on the degraded sensor. However, we will see that the LAEKF is not so adversely effected by this sensor degradation as it *remembers* the system dynamics identified earlier. While this example may seem a bit constructed, it is analogous to real time operations where the quality of the received data is changing in time as a function of location in state space.

Again, the same second-order spring-mass-damper system is used with the same unknown and learning system architecture. The system is again driven by the superposition of two sine waves, one of period $T = 15$ seconds, the other of period $T = 30$ seconds. However, this time the simulation is run for 500 seconds and the driving signal is initially biased positive, and subsequently negative, for the first and second 150 seconds, before the bias is dropped at 300 seconds. That is

$$\begin{aligned} u(t) &= 15\sin(0.2t) + 15\sin(0.4t) + 10 & 0 \le t < 150 \\ u(t) &= 15\sin(0.2t) + 15\sin(0.4t) - 10 & 150 \le t < 300 \\ u(t) &= 15\sin(0.2t) + 15\sin(0.4t) & 300 \le t \le 500 \end{aligned}$$

In addition, the sensor quality degrades at $t = 300$, so

$$\begin{aligned} \mathsf{E}[vv^T]^{1/2} &= [0.2] & 0 \le t < 300 \\ \mathsf{E}[vv^T]^{1/2} &= [5.0] & 300 \le t \le 500 \end{aligned}$$

and the filters are designed with knowledge of this degradation. That is, it is assumed that this failure was detected.

As before, we look to state estimation performance as a measure of quality. In figure 5.6 we see that the error for the AEKF (dotted line) stays relatively constant for the first 300 seconds regardless of which way the driving signal is biased. For the LAEKF (solid line), on the other hand, there is a large jump in error at 150 seconds when the bias switches from +10 to −10 and the system transitions into an unexplored region of the state space. This is caused by the incorrect (unlearned) mapping that the system partially relies on (and also indicated that the blending can be improved somewhat). However, after an additional training cycle (at 200 seconds), the error again drops below that of the AEKF. At 300 seconds, the time of sensor degradation, both estimates worsen in quality. However, the LAEKF error stays significantly below that of the AKEKF as it has already learned the unknown dynamics and can now rely on its learned mapping to help in estimation. The AEKF, on the other hand, must continue to identify the unknown dynamics based on its observations and

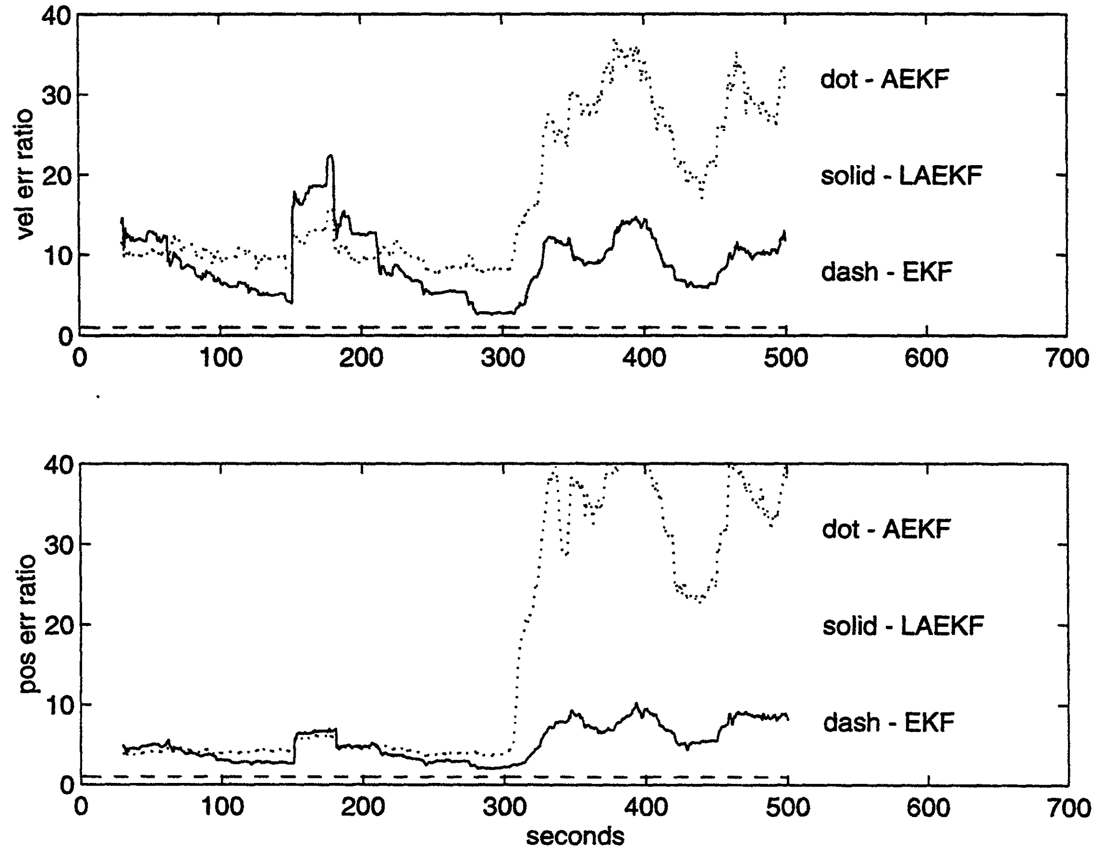

Figure 5.6: RMS state estimation errors normalized by performance of standard EKF (designed with exact nonlinear model).

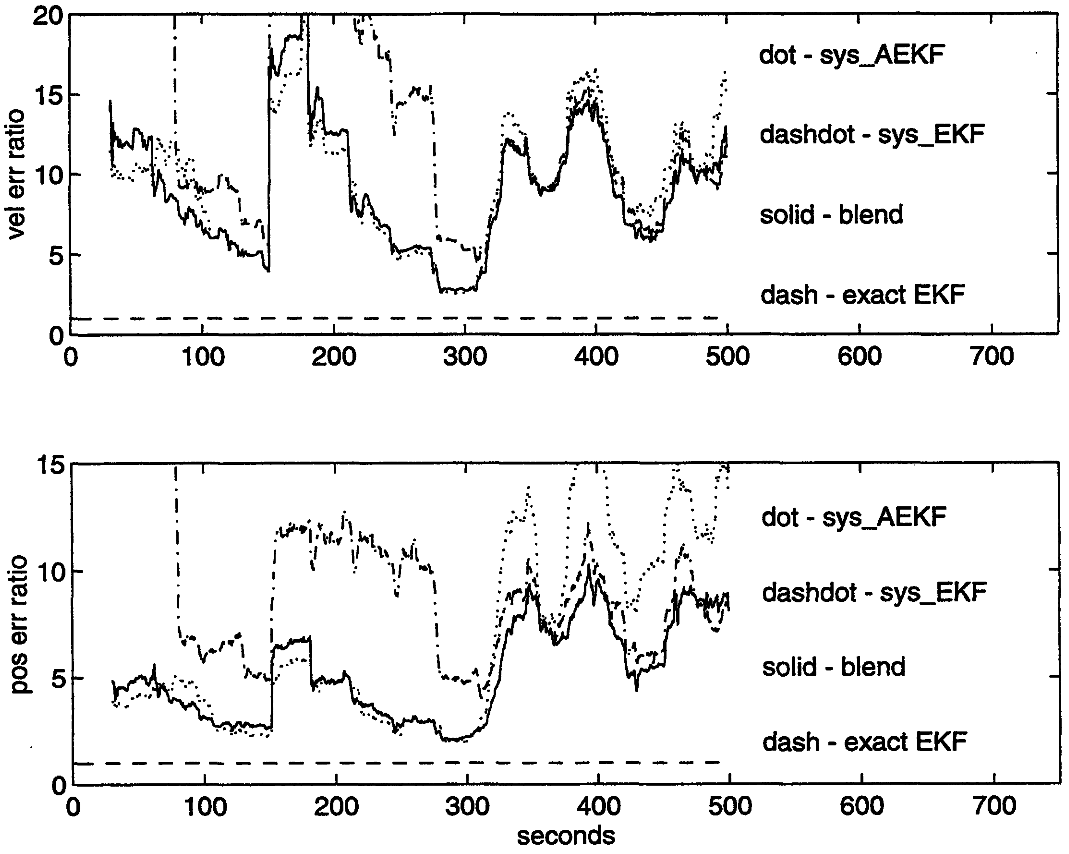

Figure 5.7: Normalized RMS state estimation errors of the AEKF and EKF systems incorporated into the LAEKF, of the final blended estimates, and of an EKF designed with full knowledge of the dynamics.

thus its performance degrades in step with the sensor. Finally, note again that the dashed line is the performance of an EKF designed with full knowledge of the state dynamics, a measure of performance by which the other plots have been normalized.

To help understand how blending comes to the rescue after sensor degradation, in figure 5.7 we plot the contributions of the two filters subsystems of the LAEKF (the AEKF and EKF) to the final, blended estimate. It helps at this point to refer back to figure 5.1 to aid in the following discussion. In figure 5.7 the error of the AEKF subsystem ($\hat{x}_1$ in figure 5.1) is plotted as a dotted line; the error of the EKF subsystem designed based solely on the learned mapping ($\hat{x}_2$) is the dash-dot line; and the blended estimate is the solid curve. For the first 300 seconds, before the sensor degrades, the AEKF subsystem provides very good estimates, and the LAEKF relied on them. After the sensor degrades, on the other hand, it is the EKF subsystem's state estimates that the LAEKF relies on. Indeed, this is exactly the purpose of blending, to direct the system to rely more heavily on the AEKF subsystem and its correction term when it was producing the better data, and more heavily on the learned mapping and EKF subsystem when the AEKF is not working well. Because in this simulation the conditions for attempting system identification degrade significantly, the AEKF that is still trying to accomplish this task suffers greatly in both its estimation and identification tasks. The EKF, which does not attempt to do any system identification, yields much better state estimates under such conditions, and the learned mapping is left unchanged.

To see what is happening to the approximation over the course of the experiment, we plot the learned mappings for this system. Figure 5.8 shows the learned mapping at two distinct times as compared with the true unknown function (solid curve). The mapping is plotted at $t = 150$ (dotted) and $t = 300$ (dashed). These plots make it clear that spatial localization has really come into play. Until $t = 150$, the driving signal is biased positive, and thus the negative region of the state space is not explored, and a correct mapping could not be learned there. Between $t = 150$ and $t = 300$, however, the system transitions into the negative state-space, explores, and learns the dynamics there. The dashed plot, corresponding to $t = 300$, has been corrected for negative position, but for positive positions the mapping has been left alone (the dotted and dashed lines are superimposed). This exploration is also made clear in figure 5.9 where the system identification RMS error has been plotted versus time. As in the

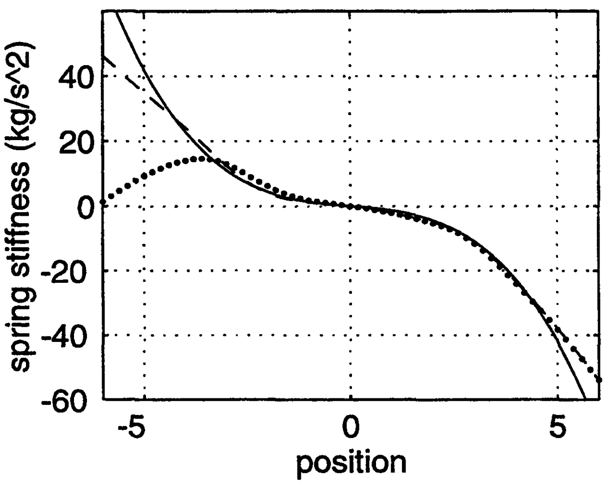

Figure 5.8: Learned mapping of spring stiffness. Dotted line corresponds to mapping at $t = 150$; dashed line corresponds to mapping at $t = 300$; solid line is truth.

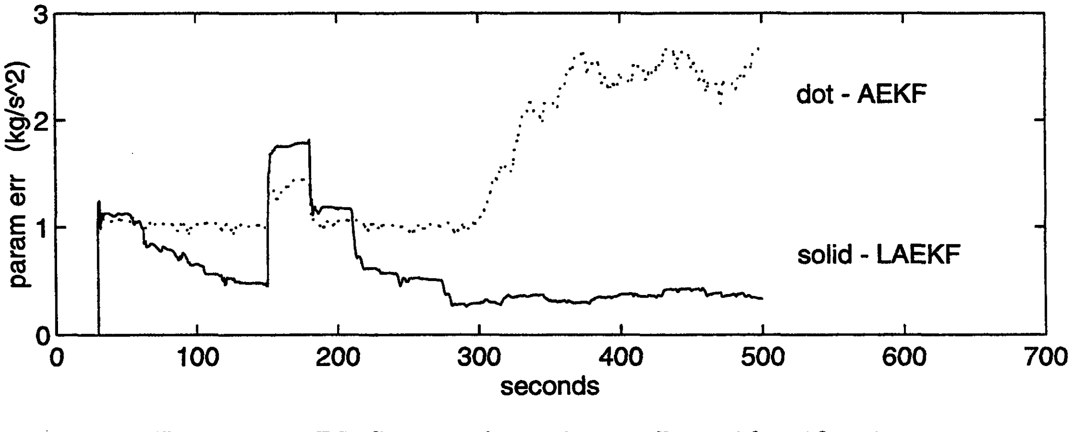

Figure 5.9: RMS errors in spring stiffness identification.

state error plots, figure 5.6, there is a error hump around $t = 150$ when the system transitions into an unexplored area of the state space. After $t = 300$, however, the identification performance of the memoryless AEKF (dotted line) worsens drastically because of the loss of the quality sensor. On the other hand, the error in the estimates of the unknown dynamics made by the LAEKF doesn't increase at all. This is because the blending has figured out the AEKF subsystem is producing poor correction terms and so turns to the EKF subsystem for state estimation. But this EKF does not attempt to do any system identification, and so the learned mapping is left alone and will stay unchanged until conditions for system identification improve.

5.3 Aeroelastic oscillator

In this third example we turn to a different system to demonstrate the strengths of this approach when it comes to accumulating training data along state trajectories. The system we turn to is a type of aeroelastic oscillator. These oscillators are systems of great practical interest with perhaps the most famous example being the Tacoma Narrows Bridge in Washington state. Our oscillator is a second-order system with unknown dynamics residing in the damping term as follows,

$$\begin{aligned} \ddot{x} &= \theta(\dot{x})\,\dot{x} - x + u + Gw \\ z &= [0 \quad 1]\begin{bmatrix} \dot{x} \\ x \end{bmatrix} + v \end{aligned}$$

where again the measurements are received periodically at 10 Hz. In the equations of motion, the unknown function is actually

$$\theta(\dot{x}) = 1.61 - 105\,\dot{x}^2 + 1531\,\dot{x}^4 - 5712\,\dot{x}^6. \tag{5.4}$$

These equations correspond to a physical system consisting of a mass with a rectangular cross section attached to a surface by a spring and damper. In

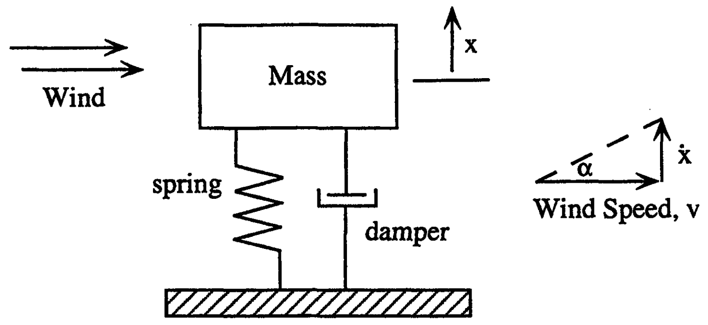

Figure 5.10: Structural diagram of aeroelastic oscillator.

addition, wind is blowing over the block and, depending on the wind velocity and angle of attack α, the oscillatory character of the mass changes; see figure 5.10. While at low angles of attack and wind speeds the mass settles out to zero if displaced, as the wind increases the mass can settle into either of two limit cycles, depending on initial conditions. This is the region in which our system operates, resulting in the functional form for $\theta(\dot{x})$ as given in equation (5.4). For a complete discussion of this system see [19].

In addition to the equation of motion discussed above, the system is driven by a small signal, $u(t) = 0.5\sin(0.2t)$, that provides some extra exploration as the system settles into its limit cycle. The learning system is initialized with centers at $\{-0.45, -0.3, -0.15, 0, 0.15, 0.3, 0.45\}$, and with narrow influences, $V = 0.01 * I$. Having influence functions this narrow means that the input space is basically divided up into a number of affine approximations. In addition, the noise statistics are as follows,

$$\mathsf{E}[vv^T]^{1/2} = [0.02] \tag{5.5}$$

$$\mathsf{E}[Gww^TG^T]^{1/2} = \text{diag}[\tfrac{\sqrt{2}}{10}\ 0.1\ 6]. \tag{5.6}$$

In figure 5.11 we present a set of plots illustrating the performance of the LAEKF for this system. We do not present state estimation performance, but

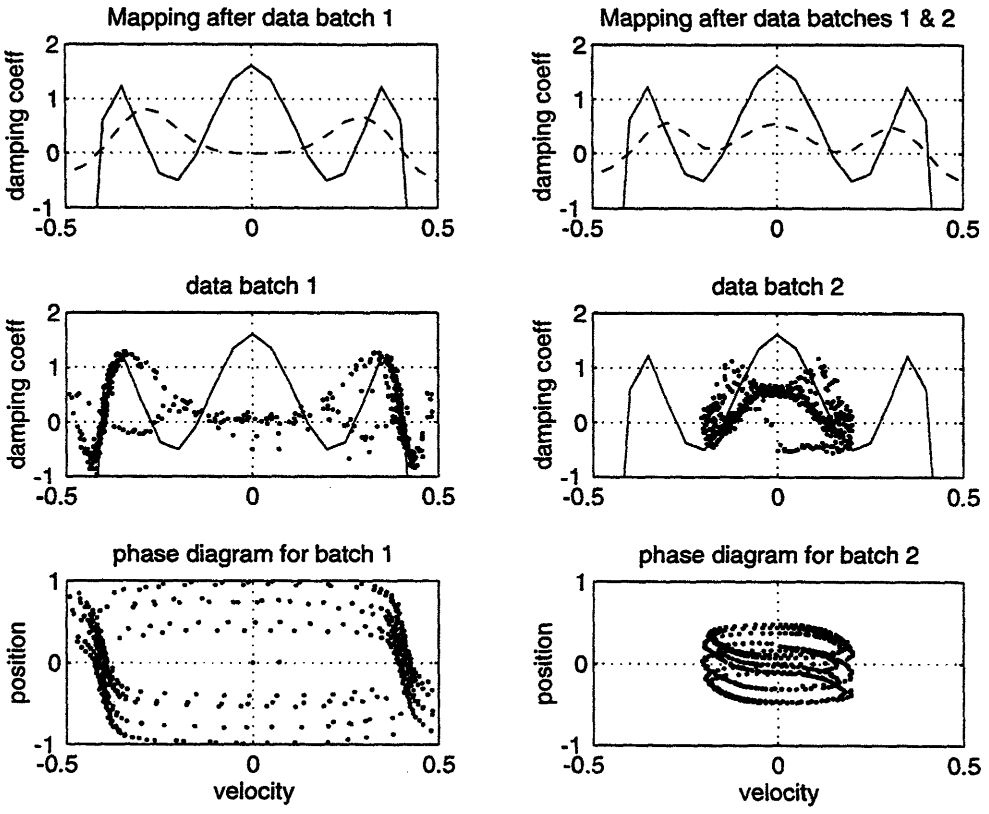

Figure 5.11: Learned mappings; data batches; and phase diagrams of two training sets for the aeroelastic oscillator.

rather graph the data and the learned functions to better comprehend how the spatially localized character of the learning system assists in these problems. It is important to understand the way this figure is set up. The figure is divided into two columns where the left hand column presents the results after the first set of training data has been received, and where the right hand column presents results after the first two training sets have been received. The top row of plots show the respective learned curves (dashed) versus the true mapping (solid). The second row of plots show the training data graphed versus the true function. The bottom row of plots are the phase diagrams of the training sets, showing the state-space trajectory of the system. The reason why these trajectories are so different is because of the limit-cycle nature of the dynamics.

The first and second training runs were started with different initial conditions and therefore settled into different limit-cycles. All plots have velocity — the independent variable in the mapping $\theta(\dot{x})$ — plotted along the horizontal axis.

This simulation demonstrates why the spatially localized character of the learning system is so important when it comes to systems that have limit cycles or only sequentially explore regions of the input space. In the first training batch (first column) no real data was collected about the center hump of the unknown dynamics because the system transitioned through that part of the state space so quickly. It was not until the system fell into the inner limit-cycle that information about the central hump could be collected. The learning system's spatially localized nature allows it to concentrate only on those areas where it had data. Thus the outer humps were trained in the first batch, and the central hump in the second.

A further benefit of this experiment was that we discovered that the LAEKF could only construct an accurate mapping for low sensor noises. This observation motivated further investigation and led to the analysis of Chapter 2, equations (2.59-2.65). Basically what was found was that as the sensor noise increases, the bandwidth of the Kalman filter drops, and the AEKF begins to treat fast spatial variations in the dynamics increasingly as noise. Since they are classified as such, the AEKF does not even attempt to track system function variations in those high bandwidths. This makes some intuitive sense since a priori we do not have a complete description of the dynamics and if, in addition, we only have very poor sensor measurements, then we really do not have much information with which to approach the tasks of estimation and identification.

Chapter 6

Conclusion

This thesis presents a learning augmented filtering algorithm for dynamic systems having unknown nonlinearities. To demonstrate the advantages of incorporating a learning system into a state estimation architecture, three different nonlinear filters are compared: (1) a standard *extended Kalman filter* (EKF) derived from an exact system model; (2) an *augmented extended Kalman filter* (AEKF) designed by augmenting an EKF derived from an incomplete description of system dynamics with additional states that model the unknown dynamics in a time-dependent manner; and (3) a *learning augmented* AEKF (LAEKF) formed by combining an AEKF with a learning system that models unknown nonlinear effects in a state-dependent manner. As the first filter is designed with full knowledge of the system dynamics (i.e., all nonlinearities), it is included only for purposes of state estimation performance comparison.

The results presented here confirm an intuitive expectation that the standard EKF, based on an exact system model, outperforms the AEKF, based on an approximate model, despite the adaptive capability of the latter. The difference in state estimation performance between these two filters is likely to be greater in situations where the unknown dynamics cannot simply be considered as a small set of constant parameters to be identified on-line. For example, the spring-mass-damper system considered in this thesis involves a spring stiffness parameter that is really an unknown nonlinear function of position. It is in such situations that learning may be advantageously applied, as demonstrated by the experiments in Chapter 5.

The underlying idea is that a complete system model can be built up over time, given estimates of the states and unknown state-dependent mappings, so that in the long-run the overall estimation performance can be improved. We have provided some background theory in Chapters 2 and 3, theory that we draw upon to design and explain the learning augmented estimator as presented in Chapter 4. Finally, in Chapter 5 we implement this design on a number of practical systems, both to see how it works in practice and to gain deeper insight into its strengths and weaknesses.

In Chapter 5 we first demonstrate the approach on a spring-mass-damper system with an unknown spring stiffness. We learn that the LAEKF improves state and parameter estimation performance dramatically, as compared to a memoryless AEKF attempting the same tasks. We also learn that if we model the time variation of the unknown spring stiffness as a random walk, the LAEKF outperforms the AEKF regardless of how we choose to pick the parameters of the model, i.e. the noise driving term. Our second application is again to a spring-mass-damper system, but this time the operating conditions are less friendly for system identification. There are biases in the driving signal, and part-way through the experiment the quality of the sensor degrades greatly. Because of the biases for the first time we see the spatially localized character of the learning system coming into play, and because of the degradation of the sensor quality we really see the need for the blending system. Our last application of the LAEKF is to an aeroelastic oscillator. This system has dynamics characterized by two limit-cycles. We run the system twice, with two different sets of initial conditions that cause the system to settle out in the different limit-cycles. Because of these distinct state trajectories, and because the LAEKF only collects training data along state trajectories, this experiment clearly shows the importance of using a spatially localized learning system. The results presented were designed to demonstrate the need for each subsystem of the LAEKF architecture. The results also show the merits of learning augmentation for a class of nonlinear estimation and identification problems and provide motivation for further work in this area.

6.1 Future directions

The results presented in this thesis suggest both immediate extensions and future areas of research.

1. *Smoothing:* An immediate direction in which this work could be taken is to implement an extended Kalman smoother to calculate the estimates used in training. Because these estimates are stored up over time in a buffer, and only periodically used to update the learned mapping, they can be smoothed to increase their accuracy.
2. *Blending and whiteness:* The blending scheme might well benefit from casting it in the light of an optimization problem. In such an optimization problem the objective would be to whiten the blended residue sequence. That is, given $\hat{x}_{1,k}$ and $\hat{x}_{2,k}$ $\forall\, k$, our objective would be to choose α_k and β_k at each time step to whiten the residue $(z_k - \alpha_k \hat{x}_{1,k} - \beta_k \hat{x}_{2,k})$, under the constraint that $\alpha_k + \beta_k = I$ where I is the identity matrix.
3. *Adaptive modeling of parameter variations:* A third extension of the current framework is to make the time variation modeling of the unknown dynamics adaptive. The motivation for such a change originated with

the discussion of figure 5.5 in Chapter 5. In that discussion it became clear that the optimal noise modeling for parameter variation is not a constant across all time and areas of the state space. Rather, the noise modeling should be made a function of the quality of the learned map. Where the learned map is poor, the noise level should be increased to allow identification; where it is refined, the noise level should be decreased so as to not interfere with state estimation.

4. *Learning approximation quality:* A final direction in which the research can be taken is to develop learning systems that keep track of their own approximation quality. This idea was discussed briefly at the close of Chapter 4. There the motivation was to help in blending. Now we see a second benefit would be that the noise modeling of parameter variations could be made a dependent function of such knowledge. Indeed, a third benefit of having such a measure of learning system error also exists. Knowledge of such error can be used to determine what areas of the state space need further exploration. Then, if we have some control over the driving signal, the system can actively direct itself into such areas in order to extend and refine its mapping. Some progress in addressing this question of learning system quality has been made from the direction of the Bayesian approach to learning as discussed briefly in Chapter 3.

Bibliography

[1] B. Anderson and J. Moore (1979). *Optimal Filtering*, Prentice Hall.

[2] W. Baker and J. Farrell (1992). An introduction to connectionist learning control systems, in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. White and D. Sofge (eds.), Van Nostrand Reinhold.

[3] W. Baker and P. Millington (1993). Design and evaluation of a learning augmented longitudinal flight control system, *32nd IEEE Conference on Decision and Control.*

[4] D. Broomhead and D. Lowe (1988). Multivariable functional interpolations and adaptive networks, *Complex Systems*, vol. 2, pp. 321–355.

[5] D. Cerrato (1993). *Modeling Unknown Dynamical Systems Using Adaptive Structure Networks*, M.S. Thesis, Department of Electrical Engineering and Computer Science, M.I.T., and Draper Laboratory Report T-1194.

[6] J. Farmer and J. Sidorowich (1987). Predicting chaotic time series, *Physical Review Letters*, vol. 59, pp. 845–848.

[7] J. Farrell (1996). Motivations for local approximations in passive learning control, to appear in *Journal of Intelligent Systems and Control.*

[8] A. Gelb (ed.) (1974). *Applied Optimal Estimation*, M.I.T. Press.

[9] A. Jazwinski (1970). *Stochastic Processes and Filtering Theory*, Academic Press.

[10] A. Lapedes and R. Farber (1987). *Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling*, Los Alamos Report LA-UR 87-2662.

[11] M. Livstone, J. Farrell, and W. Baker (1992). A computationally efficient algorithm for training recurrent connectionist networks," *1992 American Control Conference.*

[12] L. Ljung (1979). Asymptotic Behavior of the Extended Kalman Filter as a Parameter Estimator for Linear Systems, *IEEE Transactions on Automatic Control*, vol. AC-24, no. 1, pp. 36–50.

[13] L. Ljung and T. Söderström (1983). *Theory and Practice of Recursive Estimation*, M.I.T. Press.

[14] J. Lo (1994). Synthetic approach to optimal filtering, *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 803–811.

[15] D. Mackay (1992). *Bayesian Methods for Adaptive Models*, PhD thesis, California Institute of Technology.

[16] M. Matthews (1990). Neural network nonlinear adaptive filtering using the extended Kalman filter algorithm, *1990 International Neural Network Conference.*

[17] M. Matthews (1990). A state-space approach to adaptive nonlinear filtering using recurrent neural networks, *1990 IASTED Symposium on Artificial Intelligence Applications and Neural Networks.*

[18] K. Narendra and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27.

[19] G. Parkinson and J. Smith (1964). The Square Prism as an Aeroelastic Non-linear Oscillator, *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 17, pt. 2.

[20] T. Poggio and F. Girosi (1989). *A theory of networks for approximation and learning*, Artificial Intelligence Laboratory Memo No. 1140, M.I.T.

[21] H. Poor (1988). *An Introduction to Signal Detection and Estimation*, Springer-Verlag.

[22] D. Wiberg and D. Dewolf (1988). The Wiberg estimator: continuous-time case, *27th IEEE Conference on Decision and Control*, pp. 845–850.

[23] R. Williams (1990). Adaptive state representation and estimation using recurrent connectionist networks, in *Neural Networks for Control*, W. Miller, R. Sutton, and P. Werbos (eds.), M.I.T. Press.